

EXHIBIT M3

This claim chart is meant to be illustrative for purposes of meeting Plaintiff's pleading obligations and should not be construed as binding or limiting.

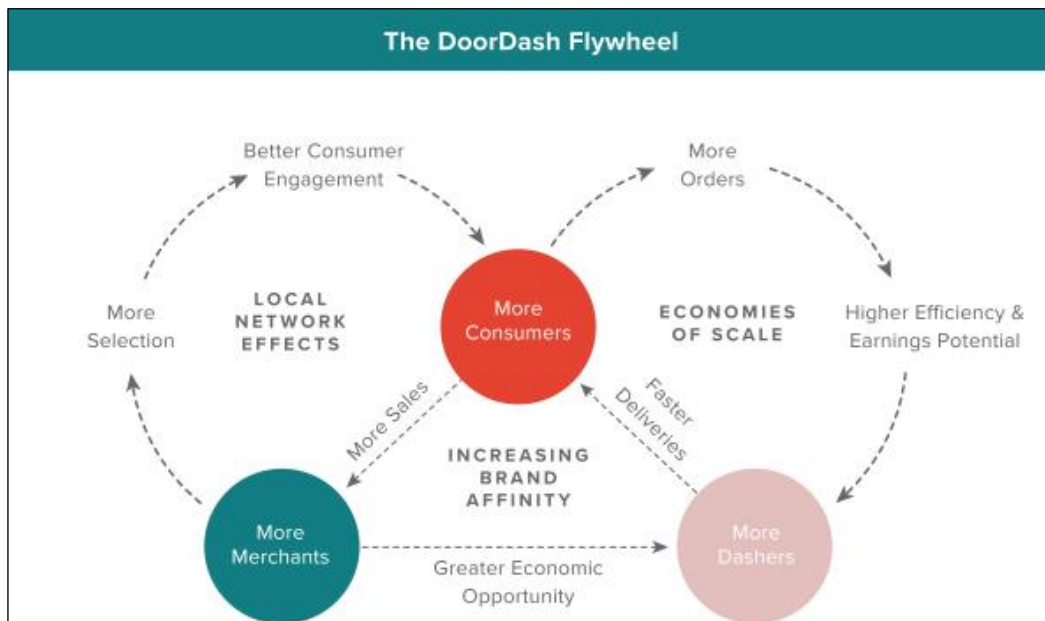
Ameranth

U.S. Patent No. 11,847,587

Claim 1 Chart

1. An intelligent back office and handheld/mobile distributed computing network with varying, multi-modes of contact, and parallel operational capabilities for use in completing remotely initiated hospitality tasks in the hospitality market comprising:

DoorDash develops, tests, integrates, uses, operates, manages, and maintains a ‘parallel operational’ based cluster set of interconnected and improved “Intelligent Web Servers with multi modes of contact”, utilizing Machine Learning (ML) with parallel operational capabilities for multi-users, via multi-communications protocols while integrating with its “central data platform” (i.e. “master database”). This system meets all the limitations of the ‘587 claims, as is detailed below, to complete remotely initiated hospitality food and drink delivery and pick-up tasks triggered by and from consumers, produced by merchants (restaurants, etc.), and delivered by Dashers. The DoorDash Flywheel diagram, shown below, makes evident that the DoorDash operates in accordance with its overall series of linked services and via its “360 degree picture” central (“Flywheel”) and technology platform framework:



This was confirmed on August 17, 2022, by DoorDash’s Vice President of Analytics and Data Science Jessica Lachs in an interview on “Leveraging Data to Delight Customers Despite a Challenging Supply Chain” (See Exh. #98) in which she states:

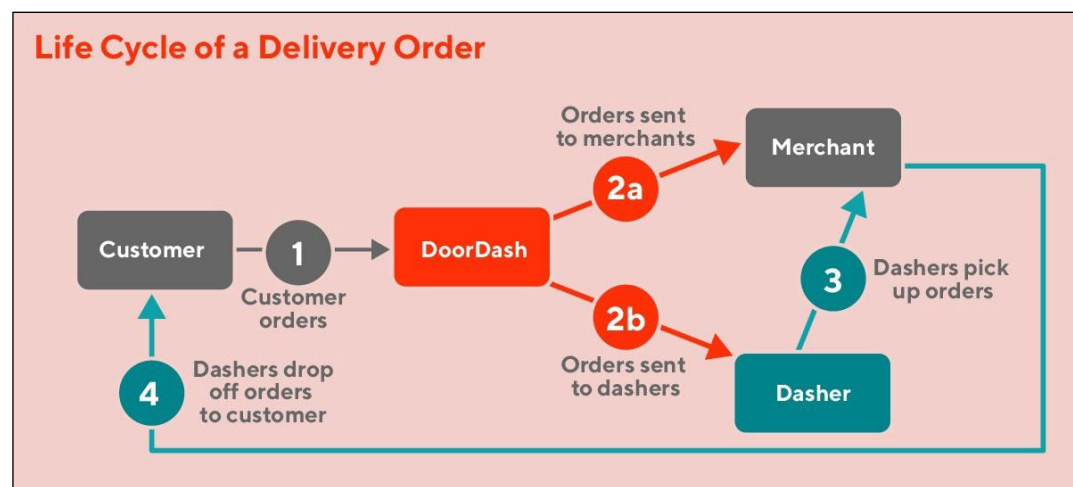
“And so for us, it’s really about collecting as much information as we can about all sides of the marketplace, bringing all of that data together into a central data platform, where all of that data is accessible no matter the source. Whether it is coming from our production systems, transactional data, whether it is event data in our apps, whether that’s the consumer app, the dasher app, the merchant app... whether it is coming from our CRM systems. All of that data needs to come in to one central place so that we can tie it together and use the insights together to create a 360 degree picture of what’s happening on our platform and off our platform so that we can use that information not just to provide accurate menus and inventory

for consumers but also so we can send the right email communications to consumers, to dashers, so that we really have a full picture of what's happening and can use that for personalization and to help all three sides of our marketplace really optimize that they are at their peak efficiency."

"So, for us, we want data to be easily accessible to all the different teams that need access to it. Analytics, being one of the largest customers of data at DoorDash, of course, but the way we think about our data models is really about increasing accessibility and consistency to that data. So, having all of our data in one central place and making sure that it is high in performance and so like query speeds are fast and that data models are thoughtful, so that it makes it a lot easier for data scientists, analysts, operators, product managers to be able to query the data that is needed and use the data in our production, in our production systems as well. So, we try to be thoughtful about how we structure our data models and how we ensure that all of the different production systems tie together into that central, as you mentioned, that central data lake."

The DoorDash Life Cycle of a Delivery Order

In the blog article from the DoorDash Engineering team titled "Next-Generation Optimization for Dasher Dispatch at DoorDash" (see Exh. 84) they state that the DoorDash platform "...powers an on-demand marketplace involving real-time order demand and dasher [(drivers)] supply. Consumers ask for goods to be delivered from a merchant to their location. To fulfill this demand, [DoorDash] present dashers with delivery routes, where they move between picking up orders at merchants and delivering them to consumers", followed by the diagram below:



Moreover, they assert the intelligence of their platform "Our dispatch system seeks high dasher efficiency and fulfillment quality by considering driving distance, time waiting for an order to be ready, delivery times as seen by the consumer, and more. Given incomplete information about the world, the system generates many predictions, such as when we expect an order to be ready for pickup, to model the real-world scenarios. With this data, the dispatch system generates future states for every possible matching and decides the best action to take, given our objectives of efficiency and quality." See also Note 1 and Note 2 below.

NOTE 1: To the extent, if any that DoorDash contends that it does not infringe, one or more of the '587 claim elements, they infringe under the Doctrine of Equivalents (DOE).

NOTE 2: One or more of the 149 exhibits shown below, are incorporated into the support for their infringement of each of the elements the '587 claim elements.

How DoorDash Work for Restaurants

In the DoorDash Blog article "How Does DoorDash Work for Restaurants?" (See Exh. 116), DoorDash states:

"Restaurants can grow online with DoorDash by driving more sales on the app and through their own website, using Delivery, Pickup, and DashPass to unlock the active (and hungry) customer base."

Regarding how customers place DoorDash Orders, the article states:

"Restaurants can grow online with DoorDash by driving more sales on the app and through their own website, using Delivery, Pickup, and DashPass to unlock the active (and hungry) customer base."

When customers are ready to place an order on Marketplace, they browse restaurants on the DoorDash app or website. Customers can search for their favorite establishments, or filter restaurants by cuisine, location, promotions and more. They enjoy the ease and convenience of browsing thousands of restaurants in a single app, as well as the ability to track orders and get restaurant-quality food anytime, anywhere.

Customers can also place orders through Storefront, an online ordering system that gives restaurants direct, commission-free orders through their website, social media, and Google Search & Maps.

Once the customer chooses a restaurant, they place their order, pay for it on the app or online, and receive an estimated delivery time. That's when restaurants receive the order via their chosen order protocol.

Regarding how restaurants receive DoorDash Orders, the article states:

"The order protocol is the way that restaurants receive orders from DoorDash. Restaurants can choose to receive orders on the device of their choice to maximize their off-premise efficiency. Storefront orders are processed in the same way as Marketplace orders. Restaurants have two options for their DoorDash order protocol.

Option 1: Use the POS integration

Maximize efficiency by integrating DoorDash with your point of sale (POS) system or aggregator. DoorDash integrates with leading POS and technology systems, including Square, Toast, Deliverect, ItsaCheckmate, Redcat and many more.

This allows you to receive DoorDash orders directly to your POS system and then straight to your kitchen, which reduces the risk of human error. DoorDash also has an open API which enables partners to build integrations to manage their menu, store, and order data.

Option 2: Use a tablet

You can download the DoorDash Order Manager app, our all-in-one tool for receiving, organizing, and tracking pickup and delivery orders. DoorDash can also provide a tablet for you for a small weekly fee. Just order it by contacting support or choosing the option when signing up for DoorDash.

With the Order Manager app, restaurant operators can adjust prices and menu items in real time, including deactivating (and reactivating) specific items that are out of stock. The Order Manager app also helps restaurant managers coordinate with Dashers, prepare for large scheduled orders, pause orders when it gets too busy, and contact both the customer and DoorDash support.

When a customer places a DoorDash order, you will receive a notification on the Order Manager app allowing you to quickly and efficiently process orders. Just be sure your tablet volume is on loud and the device is placed somewhere that is easily accessible by staff.”

The article also discusses using the Business Manager App for iPhone or Android:

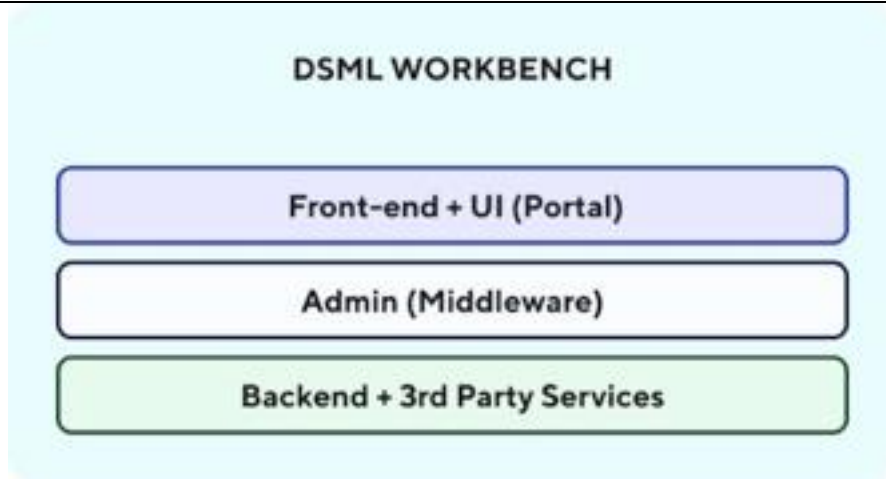
Bonus: Business Manager App

“Download the Business Manager App on the App Store or Google Play Store to manage your restaurant on the go. The Business Manager app makes it easy to track orders and review performance, all from your phone. Monitor a Dasher's location, access 24/7 support, mark items out of stock, update store hours, get analytics, and more from the app.”

In the blog article by the DoorDash Engineering team titled “Transforming MLOps at DoorDash with Machine Learning Workbench”, states how DoorDash has developed an internal Machine Learning (ML) Workbench to enhance data operations and support their data scientists, analysts, and AI/ML engineers. The importance of ML at DoorDash is emphasized, given its applications across the platform's ecosystem involving customers, Dashers, and merchants. The ML Workbench serves as a centralized hub for tasks in the machine learning lifecycle, including building, training, tuning, and deploying machine learning models. (See Exh. 121)

DoorDash's ML Workbench streamlines the machine learning process by providing a space for data collection, organization, and use in ML algorithms. It was designed with user-centered principles, aiming to create a best-in-class internal tool that is functional, usable, aesthetically pleasing, and integrates well with DoorDash's internal tools ecosystem. The development strategy was iterative, focusing on understanding user pain points, designing solutions, running user tests, and optimizing for better velocity and productivity.

Key components of the ML Workbench include front-end UI (Portal), Admin (Middleware), and Backend + 3rd Party Services. Through user research and interviews, DoorDash categorized users into admins, end users, and operators, tailoring the Workbench's features to their needs. Features such as model viewing, testing predictions, and model performance monitoring were added to streamline daily workflows and accelerate model development velocity.



The implementation of the ML Workbench has led to significant improvements in the efficiency and user experience of DoorDash's engineering and data science teams. The tool facilitated better observability of features and models, and the future vision includes diversifying adoption, improving feature and model observability, and continuing a user-centric development approach.

This ML Workbench and its capabilities align well with Claim 1 of the patent, as it is an embodiment of an interconnected, intelligent web server network with multi-modes of contact, multi-communications protocols, and multi-user and parallel operational capabilities. The master database's role within this framework is also showcased, where it integrates with the network of web servers and is accessible via a database API, which intelligently learns, updates, and stores data. The Workbench's integration with various ML lifecycle stages and its emphasis on efficiency and reliability reflect the innovations described in Claim 1, particularly in the context of improving network efficiency and reliability.

The accused instrumentality meets this limitation under the doctrine of equivalents because the accused instrumentality performs substantially the same function, in substantially the same way to achieve substantially the same result. For example, DoorDash operates a network of interconnected "Intelligent Web Servers with multi modes of contact" that utilize Machine Learning (ML) for parallel operational capabilities, serving multiple users through various communication protocols while integrating with a central data platform, also known as the "master database." This system effectively meets the core elements of the claim preamble, enabling DoorDash to remotely initiate and complete hospitality food and drink delivery and pick-up tasks, all triggered by consumers, facilitated by merchants (such as restaurants), and executed by Dashers. The interconnectedness of these web servers, as depicted in the DoorDash Flywheel diagram, mirrors the multi-user and parallel operational capabilities described in the claim preamble.

See Exh. 1 - 149

a network of distributed and linked back-office servers that are

As stated above, as to the 'at least one said web server', and confirmed below, DoorDash develops, tests, integrates, uses, operates, manages, and maintains a "distributed computing" platform of clustered set of interconnected and vastly improved "Intelligent

| | |
|--|---|
| <p>continuously synchronized in real time and which are enabled to be remotely accessed and managed by a system administrator via a web based interface;</p> | <p>Web Servers with multi modes of contact”. DoorDash uses a combination of web server hardware, software, and networking services to create a very powerful and effective online web-based hospitality system, as confirmed below.</p> <p>The DoorDash web server system includes a website, mobile website, APIs, mobile apps (iOS and Android), and backend-for-frontend (BFF) architecture and functionality. Their interconnected web servers, as is shown herein, is a DoorDash engineered custom developed set of interconnected web servers while including several well-known and widely used web server-based technologies.</p> <p>A webserver serves content (images, text, menu, video, audio, video stream, etc.) over the internet (or network) to a calling device (client) that runs on computer hardware. There can be one or more calling clients at any given moment. The computer hardware that the webserver runs on can be any device with a network connection (laptop, server computer, mobile device, etc.).</p> <p>To serve content, a webserver listens on a network port (virtual point where network connections start and end) for a request from a client (browser, mobile app, webapp, desktop app, etc.) sent via a networking communication protocol (i.e., HTTP or HTTPS). The webserver interprets and processes that request and returns a response to the client with the resources requested (text, images, menus, code, etc.). Depending on the scale of the webserver running web application(s), for example the DoorDash system, more powerful linked computer servers or sets/clusters of servers will be used to run a bigger system.</p> <p>Each web server has an IP address associated with it that allows it to communicate with other computers over the internet. When a user types in a name of a website in the browser, such as DoorDash.com, the Domain Name System (DNS) finds the correct IP Address of that domain name and directs the user accordingly (see Exh. 83). Using the “ping” command from any Windows or MacOS PC on DoorDash.com returns the IP address of 104.18.29.209, as screen in this screenshot:</p> |
|--|---|

```

-Pro ~ [11:25:28]
> $ ping doordash.com
PING doordash.com (104.18.29.209): 56 data bytes
64 bytes from 104.18.29.209: icmp_seq=0 ttl=58 time=9.813 ms
64 bytes from 104.18.29.209: icmp_seq=1 ttl=57 time=11.659 ms
64 bytes from 104.18.29.209: icmp_seq=2 ttl=58 time=14.275 ms
64 bytes from 104.18.29.209: icmp_seq=3 ttl=57 time=14.175 ms
64 bytes from 104.18.29.209: icmp_seq=4 ttl=57 time=9.486 ms
64 bytes from 104.18.29.209: icmp_seq=5 ttl=58 time=13.449 ms
64 bytes from 104.18.29.209: icmp_seq=6 ttl=57 time=11.120 ms
64 bytes from 104.18.29.209: icmp_seq=7 ttl=57 time=14.248 ms
64 bytes from 104.18.29.209: icmp_seq=8 ttl=58 time=10.253 ms
64 bytes from 104.18.29.209: icmp_seq=9 ttl=57 time=10.014 ms
64 bytes from 104.18.29.209: icmp_seq=10 ttl=57 time=10.555 ms
64 bytes from 104.18.29.209: icmp_seq=11 ttl=57 time=13.055 ms
64 bytes from 104.18.29.209: icmp_seq=12 ttl=57 time=9.903 ms
64 bytes from 104.18.29.209: icmp_seq=13 ttl=58 time=10.249 ms
64 bytes from 104.18.29.209: icmp_seq=14 ttl=57 time=9.591 ms
64 bytes from 104.18.29.209: icmp_seq=15 ttl=57 time=12.271 ms
64 bytes from 104.18.29.209: icmp_seq=16 ttl=57 time=10.375 ms
64 bytes from 104.18.29.209: icmp_seq=17 ttl=57 time=9.300 ms
64 bytes from 104.18.29.209: icmp_seq=18 ttl=57 time=11.719 ms
64 bytes from 104.18.29.209: icmp_seq=19 ttl=57 time=10.877 ms
64 bytes from 104.18.29.209: icmp_seq=20 ttl=57 time=10.174 ms
64 bytes from 104.18.29.209: icmp_seq=21 ttl=57 time=14.072 ms
64 bytes from 104.18.29.209: icmp_seq=22 ttl=57 time=9.514 ms
AC
--- doordash.com ping statistics ---
23 packets transmitted, 23 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 9.300/11.311/14.275/1.703 ms
-Pro ~ [11:26:12]
> $ █

```

In the blog article “Improving Web Page Performance at DoorDash Through Server-Side Rendering with Next.JS” (see Exh. 1) by the DoorDash Engineering Team, they affirm multiple facts regarding the DoorDash system including the use of a webserver and how it works. Specifically, they state that that they needed to upgrade the user experience for desktop and mobile and employ Google’s web metrics as a benchmark for creating a faster web service. Furthermore, they state that they use Next.js, Express.js, Node.js as part of their web server system, technologies which are very popular today with developers and are in fact web server (backend) technologies. Lastly, and more importantly, they state that their system needed more “**flexibility**” from the web server hosting and therefore they decided to build and implement their own custom web server using next.js and express.js. The article states the following:

“The DoorDash app was running on a client-side system prone to loading issues, poor SEO, and other issues. By moving to server-side rendering, we hoped that we could upgrade a number of key elements, including:

Enhancing the user experience: We wanted to improve the user experience by shortening page-load times. This aligns with the recent introduction of Google’s web metrics that favor fast, lightweight pages on modest mobile devices. These metrics have significant influence on the page rank assigned by Google.

Enabling Bundle Size Optimization: Our existing client-side rendered single-page app (CSR, SPA) was becoming difficult to optimize because the size of the JavaScript and other resource bundles had become bloated.

Improving SEO: We set out to deliver optimal SEO metadata using server-side rendered content. Whenever possible, it is better to deliver fully formed web content to search engines rather than waiting for client-side JavaScript to render the content. One approach: Move API calls from the client browser (north-south) to the server-side (east-west), where performance typically is better than on a user's device."

*"Many engineers at DoorDash are huge fans of the Next.js team and Vercel. Vercel's infrastructure was built for Next.js, providing both an amazing developer experience and a **hosting infrastructure** that make working with Next.js easy and maximally optimized."*

*"At DoorDash, however, we needed a little more flexibility and customization than Vercel could offer when it comes to how we deploy, build, and host our apps. We opted instead for the **custom-server approach to serving pages via Next.js** because it provided us more flexibility in how we hosted our app within our existing Kubernetes infrastructure."*

*"**Our custom server is built** with Express.js and leverages our in-house JavaScript server toolkit, which provides out-of-the-box functionality like logging and metrics collection."*

"Before rolling out our new service to production, we needed to know how much traffic it could support and what resources it required... After an initial audit we saw that not all cores were being utilized to spread the processing load. As a result we used Node.js's cluster API to make use of all the pod's cores, which quadrupled the pod's request capacity".

Node.js "...is an open-source, cross-platform runtime environment that allows developers to create all kinds of server-side tools and applications in JavaScript. The runtime is intended for use outside of a browser context i.e., running directly on a computer or server OS (see Ex. 2). (For a simple example of how to create a server to start listening to HTTP requests see Ex. 11).

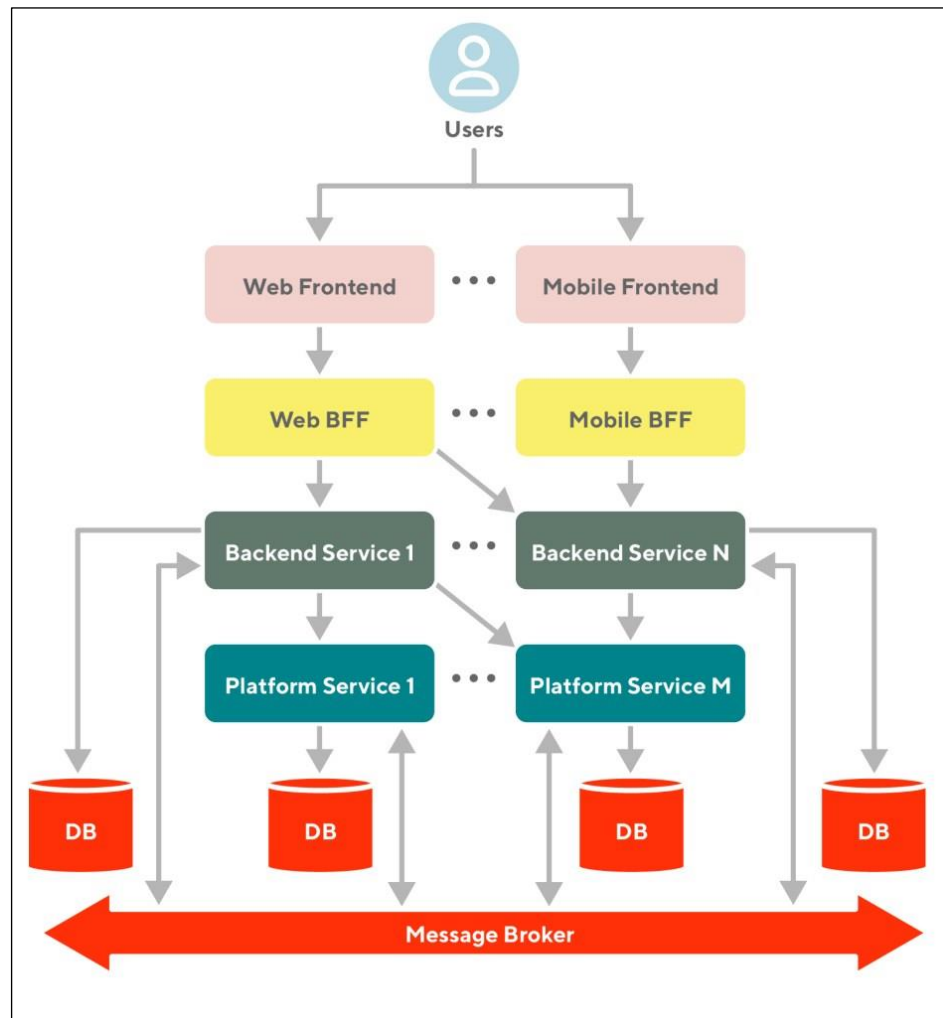
Express.js "... is a back-end web application framework for Node.js. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js" (see Ex. 3).

Next.js is a server rendered React (JavaScript) framework used to create high performing web applications and has become very popular in the tech world. *"Next.js handles the tooling and configuration needed for React, and provides additional structure, features, and optimizations for your application. Next.js is a JavaScript framework that enables you to build superfast and extremely user-friendly static websites, as well as web applications using React. This feature allows Next.js to build hybrid applications that contain both server-side rendered and statically generated pages"* (see Ex. 4.).

Furthermore, in 2019, DoorDash rebuilt their web UI using the React and Next.js frameworks. At that time, they also implemented server-side rendering (a method of rendering web applications on the server and serving a static HTML page loaded with data specific to the requested page to the user). This helped to improve the performance of the overall web application by shortening page-load times, improve bundle size optimizations, and improving SEO (see Exh. 27).

The original DoorDash platform was originally a monolithic application written in Python using the Django web framework with a PostgreSQL database. As the platform grew, they started to have problems with reliability and scaling. Around 2018 they instituted a code freeze and began migrating to microservices. At this time, they also migrated to the Kotlin language, and their services now run on the java virtual machine (JVM) (See Exh. 20, 21, 22, 40).

DoorDash was able to improve the efficiency and reliability of their platform using a microservices architecture. By breaking up their application into domain-specific parts they were able to reduce errors and latency. They state in the article “Future-proofing: How DoorDash Transitioned from a Code Monolith to a Microservice Architecture” (see Exh. 20) that “... *final design for our new microservice architecture consisted of five different layers, ranging from the user experience to the core infrastructure. Each layer provides functionality to the upper layer and leverages the functionality exposed by the lower layer [as shown below]*”:



These layers include (See Exh. 20):

“Frontend layer: Provides frontend systems (like the DoorDash mobile app, Dasher web app, etc.) for the interaction with consumers, merchants, and Dashers that are built on top of different frontend platforms.

BFF layer: The frontend layer is decoupled from the backend layer via BFFs. The BFF layer provides functionality to the frontend by orchestrating the interaction with multiple backend services while hiding the underlying backend architecture.

Backend Layer: Provides the core functionality that powers the business logic (order cart service, feed service, delivery service, etc.).

Platform layer: Provides common functionality that is leveraged by other backend services (identity service, communication service, etc.).

Infrastructure layer: Provides the infrastructural components that are required to build the site (databases, message brokers, etc.) and lays the foundation to abstract the system from the underlying environment (cloud service provider).”

In this, DoorDash’s new architecture they introduced the backend-for-frontend (BFF) “...an application connecting the consumer-facing client and the services providing general purpose APIs. Client requests go to the BFF, which then orchestrates the aggregation of information needed by the client”. The BFF is a software architecture pattern (See Exh. 40) used by microservices which “...shifted from thick-client applications to **interfaces delivered via the web, a trend that has also enabled the growth of SAAS-based solutions in general**”. As such BFF can be considered as “...the user-facing application as being two components - a client-side application living outside your perimeter, and a **server-side component (the BFF)** inside your perimeter”. The perimeter of the BFF is the webserver.

In the blog article by the DoorDash engineering team Platform Optimization Through Better API Design (see Exh. 22) they state in the introduction:

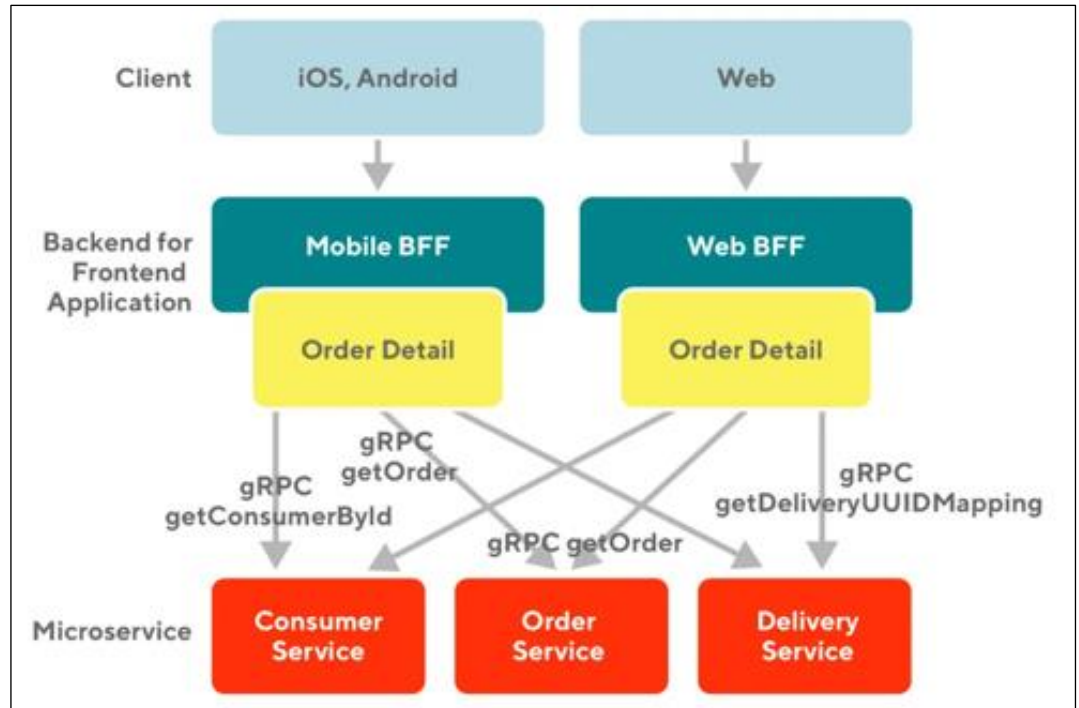
“As DoorDash migrated from a monolithic codebase to a microservices architecture, we found an opportunity to refine our API design. Beyond simple functionality, we determined best practices in making APIs that help our applications load quickly, use minimal storage, and, most importantly, avoid failures.”

“APIs, the connective tissue of a software platform, can offer performance improvements when properly designed. At DoorDash, the APIs relay frontend client requests to backend services and provide the information that users see in our apps, such as the estimated time when a food order will be delivered. If the APIs perform at maximum efficiency, client requests and responses all process more quickly” (See Exh. 20, 21, 40).

In their new architecture they introduced the backend-for-frontend (BFF) “... an application connecting the consumer-facing client and the services providing general purpose APIs. Client requests go to the BFF, which then orchestrates the aggregation of information needed by the client”. The BFF is a software architecture pattern (See Exh. 40) used by microservices which “...shifted from thick-client applications to interfaces delivered via the web, a trend that has also enabled the growth of SAAS-based solutions in general”. As such BFF can be considered as “...the user-facing application as being two components – a client-side application living outside your

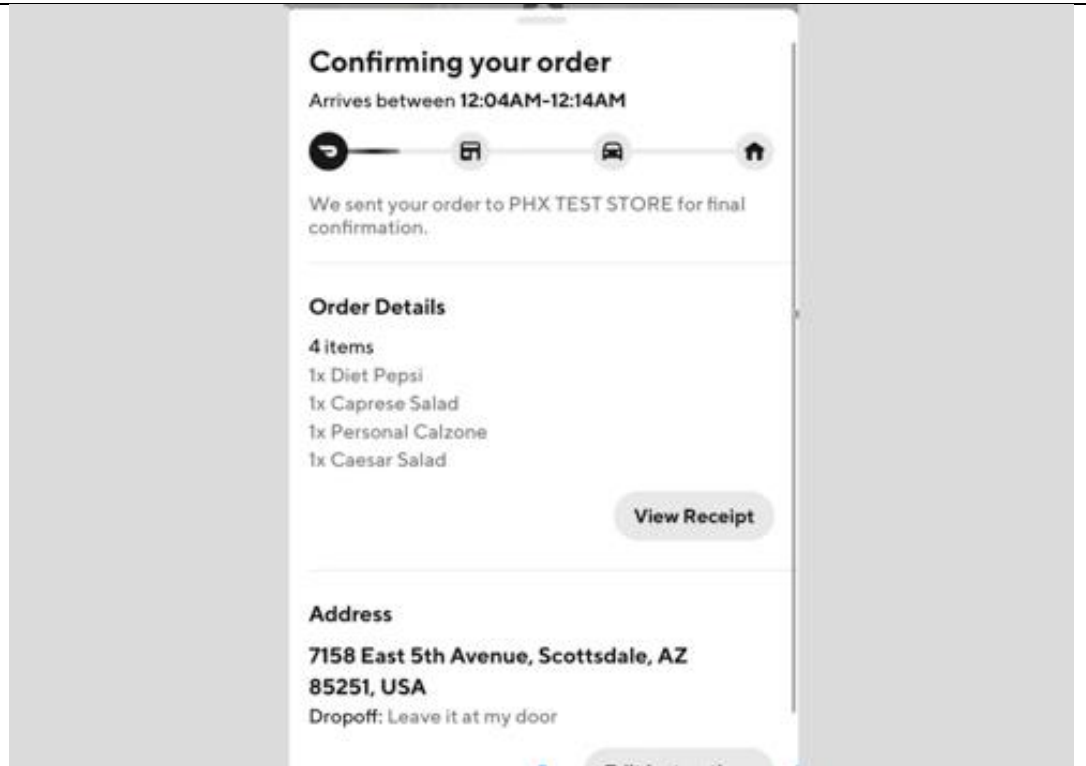
perimeter, and **a server-side component (the BFF)** inside your perimeter”. The perimeter of the BFF is the webserver.

As shown “...in the diagram below ...when the BFF receives the order details request, it orchestrates the calls to the consumer service, order service, and delivery service, **ultimately assembling the response details into a consolidated order detail response**. Building APIs using domain-specific services, orchestrated by the BFF, makes it easier to understand which RPCs are called and how they are executed” (see Exh. 41).

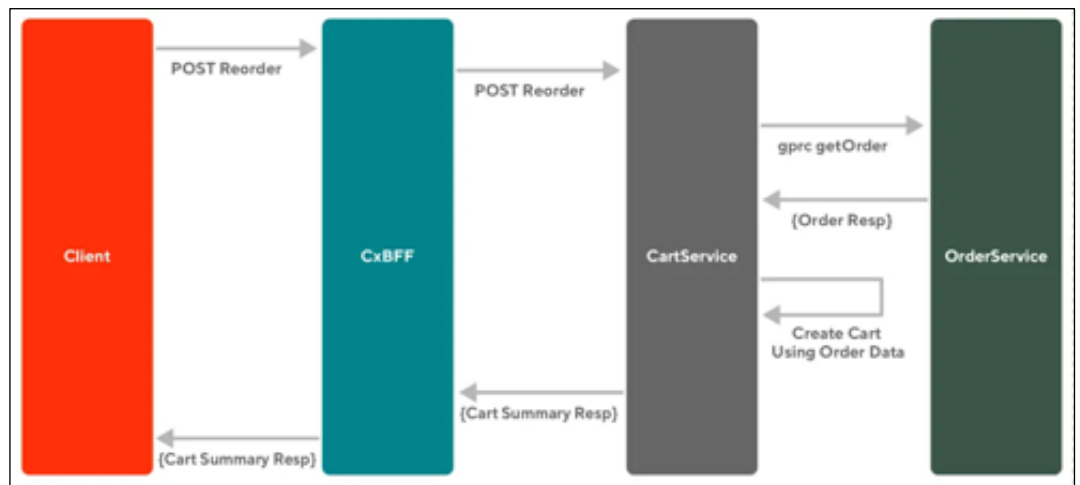


BFF is the core service that permits their large system to be pieced into various backend sub-systems (microservices) for performance improvements and easier maintainability. They go on to state that:

“At DoorDash, our APIs primarily support information on food orders, whether that’s an order placed by a consumer and sent to a restaurant, a delivery request sent to a Dasher, or a Dasher’s progress in bringing a food order to a consumer. For example, two main endpoints, order detail and order delivery status, populate our order tracking page, [as] shown in [the figure below]”



“When we redesigned the Reorder endpoint, we wrote the Reorder API so it would read the order information it needs to create the new cart instead of passing the large order data over the network to create the new cart, as shown [below] ... [here] in our redesigned order flow, we reduce network traffic by writing the Reorder API so that it reads the order detail and creates the new cart, simplifying its role.... The client only needs metadata for the new cart from the Reorder endpoint, so we only return a cart summary response to the client instead of the whole cart. Compared to the previous flow, shown in Figure 4, we not only eliminate the request to call GET order cart from the client, but also make both the request and response very light.”



Fault Tolerance with RPC Fallbacks in DoorDash’s Microservices

In the blog article “Improving Fault Tolerance with RPC Fallbacks in DoorDash’s Microservices” by the DoorDash Engineering team (See Exh. 119), they state that:

“...the industry today is replacing legacy monolithic architectures with microservices to capitalize on the promise of better scalability and developer productivity. As with anything in distributed systems, this new architecture comes with tradeoffs — most notably complexity. Interservice communication is done through remote procedure calls, or RPCs, which add a whole new set of issues to consider. But designing for fault tolerance from the start can ensure the system continues to operate properly in the event of failures.”

“DoorDash uses a microservices architecture that employs gRPC to communicate between services. Product microservices return content to clients in the user’s preferred language. The translated content is fetched from the platform’s translation microservice. A translation client simplifies the interaction with the translation systems, exposing a simplified interface that different teams can use to fetch translated content.”

“With the myriad strategies described here to mitigate failures, it is possible to keep systems operational despite them. The fallbacks we have deployed to improve our system’s fault tolerance is just one of the many initiatives DoorDash undertakes to provide our customers the best service possible.”

Node.js SDK

Recently, DoorDash further customized its webservers (again which runs on node.js) by creating a new API called DoorDash Node.js SDK which *“...care of the boilerplate work of using our API, like setting up the authentication token and making HTTP requests. SDKs generally make it faster to use an API because they take care of this boilerplate work so you can get right to the task you’re trying to achieve—say, creating a delivery and requesting a Dasher. In the screenshot below, the code on the left creates a delivery using the Create Delivery API directly; the code on the right, using the DoorDash Node.js SDK.”* See Exh. 41.

```

1 import axios, { AxiosResponse } from "axios";
2 import { v4 as uuidv4 } from "uuid";
3 import { iDropOff, iPickup } from "../models";
4 import jwt from "jsonwebtoken";
5
6 export async function createDelivery(): Promise<AxiosResponse<string>> {
7   const body = JSON.stringify({
8     external_delivery_uuid: uuidv4(),
9     pickup_address: "1000 4th Ave, Seattle, WA, 98104",
10    pickup_phone_number: "+1(650)5555555",
11    dropoff_address: "1201 3rd Ave, Seattle, WA, 98101",
12    dropoff_phone_number: "+1(650)5555555",
13  });
14
15  try {
16    const response = await axios.post(
17      "https://openapi.doorcrawl.com/drive/v2/deliveries",
18      body,
19      {
20        headers: {
21          Authorization: "Bearer " + getToken(),
22          "Content-Type": "application/json",
23        },
24      }
25    );
26
27    return response;
28  } catch (error) {
29    console.error(error);
30  }
31 }
32
33 export function getToken() {
34   return jwt.sign(
35     // Payload
36     {
37       aud: "doordash",
38       iss: process.env.DEVELOPER_ID,
39       kid: process.env.DEVELOPER_KEY_ID,
40     },
41     // secretOrPrivateKey
42     Buffer.from(process.env.BASE_64_SIGNING_SECRET, "base64"),
43     // Options
44     {
45       algorithm: "HS256",
46       expiresIn: "1800s",
47     }
48   );
49 }

```

```

1 import { DeliveryResponse,
2   DoorDashAuthorizationError,
3   DoorDashClient,
4   DoorDashResponse } from "@doordash/sdk";
5
6 import { v4 as uuidv4 } from "uuid";
7
8 const client = new DoorDashClient({
9   developer_id: "{your developer_id}",
10  key_id: "{your key_id}",
11  signing_secret: "{your signing_secret}"
12 });
13
14 client.createDelivery({
15   external_delivery_id: uuidv4(),
16   pickup_address: "1000 4th Ave, Seattle, WA, 98104",
17   pickup_phone_number: "+1(650)5555555",
18   dropoff_address: "1201 3rd Ave, Seattle, WA, 98101",
19   dropoff_phone_number: "+1(650)5555555",
20 }).then((response: DoorDashResponse<DeliveryResponse>) => {
21   // do something
22 }).catch((err: any) => {
23   // handle error
24 });

```

AWS Web Server Hosting and Location

DoorDash migrated their hosting from Heroku to Amazon Web Services (AWS) sometime in 2015. They built a Docker image of their app to make the transition easier. At that time DoorDash adopted “Amazon RDS Postgres” as primary data store and then in 2018 migrated to “Amazon Aurora Postgres.”

Since 2019, they have been using the following AWS services: Amazon ElastiCache (in-memory caching service), Amazon Relational Database Service (aka RDS is a collection of database services), Amazon Aurora (scalable MySQL and PostgreSQL database service), Amazon CloudWatch (monitoring and observability service), Amazon DynamoDB (NoSQL database service), Amazon Kinesis (real time video process and analyzing service), and Amazon Redshift (data warehouse service). DoorDash also uses Amazon EC2 compute resources. This is likely how they host their Kotlin/Java application (see Exh. 23, 24, 25, 26).

Web Technology Surveys

W3Techs, World Wide Web Technology Surveys (See Ex. 8) identifies DoorDash as using multiple server, client, and programming technologies to operate their system (see Ex 13). These include but not limited to the following:

Content Management System: *Salesforce Customer 360 (used on a subdomain)*

Server-side Programming Language: *ASP.NET Microsoft’s Active Server Pages technology on the .NET framework.*

Client-side Programming Language: *JavaScript, Bootstrap is an HTML, CSS.*

Web Servers: *Cloudflare Server and Envoy. Cloudflare Server is a web server developed by Cloudflare. Envoy Proxy is a proxy server designed for large service-oriented architectures.*

Reverse Proxy Service: *Cloudflare provides a content delivery network.*

DNS Server Provider through Cloudflare. *DNS allows websites to found by name such as DoorDash.com on webservers.*

Home Technologies Reports Sites Quality Users Blog Forum FAQ Search

Learn to build a website, step-by-step - WebsiteSetup | WordPress Tutorials & Themes - Themeisle | Free WordPress SEO plugin - SEOPress | Web Design Tips, Tutorials, and Guides - DesignBombs

lookup another site: Site Info
Get our [site info tools](#).

Site Info - Doordash.com

Overview of web technologies used by Doordash.com.

Website Background

| Description on Homepage | DoorDash Food Delivery & Takeout - From Restaurants Near You Delivery & takeout from the best local restaurants. Breakfast, lunch, dinner and more, delivered safely to your door. Now offering pickup & no-contact delivery. | | | | | | | | | |
|-------------------------|--|--------------|----------|--------------|---------------|-------|-----|--------|------|-------|
| Popularity rank | Number 643 of all websites according to Alexa | | | | | | | | | |
| Main visitors locations | <table border="1"> <tr> <th>Country</th> <th>Visitors</th> <th>Country Rank</th> </tr> <tr> <td>United States</td> <td>94.5%</td> <td>521</td> </tr> <tr> <td>Canada</td> <td>4.3%</td> <td>1,237</td> </tr> </table> | Country | Visitors | Country Rank | United States | 94.5% | 521 | Canada | 4.3% | 1,237 |
| Country | Visitors | Country Rank | | | | | | | | |
| United States | 94.5% | 521 | | | | | | | | |
| Canada | 4.3% | 1,237 | | | | | | | | |

Content Management System

| | |
|---|---|
| Salesforce Customer 360 used on a subdomain | Salesforce Customer 360 provides platforms that enables companies to connect with customers and partners. |
|---|---|

Server-side Programming Language

| | |
|---------|---|
| ASP.NET | Microsoft's Active Server Pages technology on the .NET framework. |
|---------|---|

Client-side Programming Language

| | |
|------------|---|
| JavaScript | JavaScript is a lightweight, object-oriented, cross-platform scripting language, often used within web pages. |
|------------|---|

JavaScript Library

| | |
|-------------------------------|--|
| Bootstrap used on a subdomain | Bootstrap is an open source HTML, CSS, and JavaScript framework. |
|-------------------------------|--|

Web Servers

| | |
|-------------------|--|
| Cloudflare Server | Cloudflare Server is a web server developed by Cloudflare. |
| Envoy | Envoy Proxy is a proxy server designed for large service-oriented architectures. |

Web Hosting Provider

| | |
|--------------------------------|--|
| Salesforce used on a subdomain | Salesforce is a US-based provider of cloud-based software. This includes the former Demandware brand. <small>hosting info partly based on data from ipinfo.io</small> |
|--------------------------------|--|

Data Center Provider

| | |
|--------------------------------|---|
| Salesforce used on a subdomain | Salesforce is a US-based provider of cloud-based software. This includes the former Demandware brand. |
|--------------------------------|---|

Reverse Proxy Service

| | |
|------------|---|
| Cloudflare | Cloudflare provides a content delivery network. |
|------------|---|

DNS Server Provider

| | |
|------------|---|
| Cloudflare | Cloudflare provides DNS servers and other web services. |
|------------|---|

StatsCrop


StatsCrop.com is a free online website analyzer. Using this service, the following website information for DoorDash.com is generated including the location of their servers, name server info, DNS information and whois profile (see Ex. 10):

Server Location

Site Server

Server Location

Where are Doordash.com website's servers located? The servers are being hosted in United States.



| # | IP Address | Country / Region | City |
|---|--------------------------|------------------|------|
| 1 | 104.18.41.135 | United States | - |
| 2 | 172.64.146.121 | United States | - |
| 3 | 2606:4700:4400:6812:2987 | United States | - |
| 4 | 2606:4700:4400:ac40:9279 | United States | - |

Name Servers

Site Domain

Domain Profile
Domain Whois
DNS Record
Name Server

Name Server

The table below shows the Name Server for the domain name Doordash.com.

| # | Name Server | IP Address |
|---|------------------|---------------|
| 1 | ns1.doordash.com | 162.159.6.188 |
| 2 | ns2.doordash.com | 162.159.7.107 |

Domain Whois

Domain Whois

Domain Whois is a query and response protocol that is widely used for querying databases that store the registered users or assignees of a domain name. The following information is the Whois of the domain Doordash.com. [Whois Lookup](#)

```

Domain Name: doordash.com
Registry Domain ID: 1689302483_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.registrar.amazon.com
Registrar URL: https://registrar.amazon.com
Updated Date: 2023-09-28T04:05:34Z
Creation Date: 2013-06-18T19:11:18Z
Registrar Registration Expiration Date: 2029-06-18T19:11:18Z
Registrar: Amazon Registrar, Inc.
Registrar Abuse Contact Email: abuse@amazonaws.com
Registrar Abuse Contact Phone: +1.206.760.2482
Reseller:
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Registry Registrant ID:
Registrant Name: On behalf of doordash.com user
Registrant Organization: Whois Privacy Service
Registrant Street: P.O. Box 81226
Registrant City: Seattle
Registrant State/Province: WA
Registrant Postal Code: 98188-1226
Registrant Country: US
Registrant Phone: +1.2065771368
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: emer-7142752@doordash.com.whoisprivacyservice.org
Registry Admin ID:
Admin Name: On behalf of doordash.com administrative contact
Admin Organization: Whois Privacy Service
Admin Street: P.O. Box 81226
Admin City: Seattle
Admin State/Province: WA
Admin Postal Code: 98188-1226
Admin Country: US
Admin Phone: +1.2065771368
Admin Phone Ext:
Admin Fax:
Admin Fax Ext:
Admin Email: admin-7142752@doordash.com.whoisprivacyservice.org
Registry Tech ID:
Tech Name: On behalf of doordash.com technical contact
Tech Organization: Whois Privacy Service
Tech Street: P.O. Box 81226
Tech City: Seattle
Tech State/Province: WA
Tech Postal Code: 98188-1226
Tech Country: US
Tech Phone: +1.2065771368
Tech Phone Ext:
Tech Fax:
Tech Fax Ext:
Tech Email: tech-7142752@doordash.com.whoisprivacyservice.org
Name Servers: ns1.doordash.com
DNSSEC: signedDelegation
URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/
-- Last update of WHOIS database: 2023-07-28T04:05:24.00Z --
For more information on Whois status codes, please visit https://www.icann.org/resources/pages/status-codes
By submitting a query to the Amazon Registrar, Inc. WHOIS database, you agree to abide by the following terms. The data in Amazon Registrar, Inc.'s WHOIS database is provided by Amazon Registrar, Inc. for the sole purpose of assisting you in your research of domain name registrants. For more information, please visit https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/domain-contact-support.html
Contact information available here: https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/domain-contact-support.html
© 2022, Amazon.com, Inc., or its affiliates
    
```

More importantly, StatsCrap.com shows the following webserver DNS records for DoorDash.com (see Exh. 10).

Note: DNS is “...a hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. The table below shows the DNS record for the domain name DoorDash.com” (see Exh. 83).

DNS Record

The Domain Name System (DNS) is a hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. The table below shows the DNS record for the domain name Doordash.com.

| Host | Type | Content |
|------------------|------|--|
| doordash.com | A | 104.18.41.135 |
| doordash.com | A | 172.64.146.121 |
| doordash.com | NS | ns1.doordash.com |
| doordash.com | NS | ns2.doordash.com |
| doordash.com | MX | aspmx3.googlemail.com |
| doordash.com | MX | aspmx2.googlemail.com |
| doordash.com | MX | aspmx1.google.com |
| doordash.com | MX | alt1.aspmx1.google.com |
| doordash.com | MX | alt2.aspmx1.google.com |
| doordash.com | TXT | google-site-verification=2rcRtS1Udxw6DyypdVQDk9KxUhcNvMlp74YRRsA |
| doordash.com | TXT | google-site-verification=oinRvdbIMV4Ncm6r26vumPZ8wTqV43myYzProPB1M |
| doordash.com | TXT | google-site-verification=pESHdBYNwhG8AJm--kYJD-HSjW7Y90Gaxgslqds |
| doordash.com | TXT | ALIASES for doordash-prod-lb-1965037934.us-west-2.elb.amazonaws.com |
| doordash.com | TXT | google-site-verification=rdKNCs8nOYSMdrp6GqZcA882ULU3FWNHqMzHTaHUK |
| doordash.com | TXT | google-site-verification=GWpsrxGOKS4wrPZn6vo6CpWIS4UuA32W0qMrOT1esM |
| doordash.com | TXT | google-site-verification=_nhAl3u-62bMXOhq8KHCRaIYgz_iggCvNuE9l_3nVdW |
| doordash.com | TXT | google-site-verification=UcYKd6s6iOERFexTTouDRNxxKR8HfH_DxMgDQyFMWg |
| doordash.com | TXT | stripe-verification=972a79921adf9a3d42e966695a8aba1a145b5eb4d7c2373d05ba7a1bbd535fb4 |
| doordash.com | TXT | cloudbees-domain-verification=485857a9eee140720ace1308ce419d23b24d6062 |
| doordash.com | TXT | docuSign=b7b01054-c73b-4575-9b14-e509789cf05 |
| doordash.com | TXT | atlassian-domain-verification=WynfBPjKThUeFbQhY1YvhBd+YgTvHjYK7+H7ZbQjdyS1eNehVzWxkFA5KU6RB |
| doordash.com | TXT | dropbox-domain-verification=zqgh8mdbs1rt |
| doordash.com | TXT | MS=ms82589317 |
| doordash.com | TXT | google-site-verification=ixlt07NjESGULS-Txeg7Qu2T6qhzv6zHEgJ8l3R10 |
| doordash.com | TXT | v=spf1 include:doordash.com._nsfp.vali.email include:%(j)_lp.%(h)_ehlo.%(d)_spf.vali.email -all |
| doordash.com | TXT | atlassian-domain-verification=Nhx5JpLlPH2KCXSAAyEmsQTJ0dBCbNf5J00Q95K09LhSO2euZJ0t9HF0dHQQJ |
| doordash.com | TXT | apple-domain-verification=gLrxbW5elM6T0bPk |
| doordash.com | TXT | facebook-domain-verification=ke0mqxx0oqg5awcoellkz177oxn2u |
| doordash.com | TXT | adobe-idp-site-verification=4c435e2e07ad31e38e2b8eef37dd3422b6eaf6dec8746a240258d8a2f96e9 |
| doordash.com | TXT | nintex.60aff12781259a0069a20271 |
| doordash.com | TXT | docuSign=5b94c41e-39e0-479d-b682-d928d5cd0d54 |
| doordash.com | TXT | stripe-verification=90559cf6e421e37c8322149535e5ee37d6683aac1844c3a9d3c41d9783e1ca63 |
| doordash.com | TXT | stripe-verification=fc0a51eed89484d07c775f7cc99a8267651b02c0e4c7b23bc0940e6f04be1 |
| doordash.com | AAAA | 2606:4700:4400::6812:2987 |
| doordash.com | AAAA | 2606:4700:4400::ac40:9279 |
| www.doordash.com | A | 172.64.155.210 |
| www.doordash.com | A | 104.18.32.46 |
| www.doordash.com | AAAA | 2606:4700:4400::6812:202e |
| www.doordash.com | AAAA | 2606:4700:4400::ac40:9bd2 |
| doordash.com | SOA | class: IN ttl: 300 mname: ns1.doordash.com rname: dns.cloudflare.com serial: 2285555019 refresh: 10000 retry: 2400 expire: 604800 minimum-ttl: 300 |

Who.is also reports similar information (See Exh. 27). In the trace root, we an IP of 216.182.229.160. A search in who.is for that IP reveals an amazon AWS server:

who.is Search for domains or IP addresses... Premium Domains Transfer Features Login Sign Up

Interested in domain names? [Click here](#) to stay up to date with domain name news and promotions at Name.com

216.182.229

DNS Information

Whois **DNS Records** Diagnostics

DNS Records for 216.182.229 cache expires in 4 minutes and 23 seconds

| Hostname | Type | TTL | Priority | Content |
|-------------|------|------|----------|---|
| 216.182.229 | SOA | 900 | | dns-external-master.amazon.com root@amazon.com 2006082303 3600 900 604800 900 |
| 216.182.229 | NS | 3600 | | x1.amazonaws.com |
| 216.182.229 | NS | 3600 | | x2.amazonaws.com |
| 216.182.229 | NS | 3600 | | x3.amazonaws.org |
| 216.182.229 | NS | 3600 | | x4.amazonaws.org |
| 216.182.229 | NS | 3600 | | pdns1.ultradns.net |

[Transfers](#) [Premium Domains](#) [Web Hosting](#) [Website Builder](#) [Contact Us](#) [FAQs](#) [Terms of Service](#)

WebTechSurvey

This site lists the technologies used by a website (called technology stack). In the case of doordash.com, webtechsurvey.com provides the following high-level summary report (See Exh. 44):

Web Tech Survey Home Web Technologies HTTP Headers Sign In [Free Sign Up](#)

www.doordash.com

website information and technology stack

- Web technologies detected on this domain: **29** technologies
- Web technologies detected across all doordash.com subdomains: **93** technologies
- Website rank: **3,105**
- Website Location: **US**
- Language: **English**
- Last technology update: **Jul 1, 2022**
- Added technologies: **1**
- Total doordash.com subdomain count: **27** subdomains
- Top ranking subdomain: **www.doordash.com** (Rank: 3,105)
- IP Address: **172.64.155.210**
- Reverse IP Lookup: **13** websites share **172.64.155.210** IP address.
- Autonomous System Number (ASN): **AS13335 (CLOUDFLARENET)**
- Total linking domains: **12,640** domains
- Total linking IP: **5,380** IPs
- Certificate Issuer: **Cloudflare, Inc.**
- Certificate Expiration Date: **Oct 2, 2022** (Valid)

Available Reports

- Technology Stack
- Technology History
- Subdomains
- Backlinks
- Domain Forwarding
- Reverse IP

Security Profile

- Live site
- Secure connection (HTTPS)
- Certificate valid
- No outdated technologies detected

Social Media

- Facebook: <https://facebook.com/doordash>
- Twitter: <https://twitter.com/doordash>
- Instagram: <https://instagram.com/doordash>










Subdomains

- www.doordash.com
- cdn.doordash.com
- get.doordash.com
- help.doordash.com
- blog.doordash.com
- work.doordash.com
- identity.doordash.com
- typography.doordash.com
- support.doordash.com
- stories.doordash.com

For the technology stack implemented by DoorDash.com for their webserver, webtechsurvey.com reports the following (See Exh. 44):

www.doordash.com





















Website technology stack

| TECHNOLOGY | TECHNOLOGY TYPE | VERSION | FIRST DETECTED |
|---|-----------------------|---------|----------------|
|  Node.js | Programming Languages | | Jan 1, 2022 |
|  Polyfill | JavaScript Libraries | | Jan 1, 2022 |
|  Polyfill IO | JavaScript Libraries | | Jan 1, 2022 |
|  React | JavaScript Frameworks | | Jan 1, 2022 |
|  srcset | Document Standards | | Jan 1, 2022 |
|  styled-components | JavaScript Frameworks | | Jan 1, 2022 |
|  UTF-8 | Document Encoding | | Jan 1, 2022 |
|  Viewport Meta | Document Standards | | Jan 1, 2022 |
|  webpack | Miscellaneous | | Jan 1, 2022 |

First Previous 1 **2** Next Last

www.doordash.com 

Website technology stack

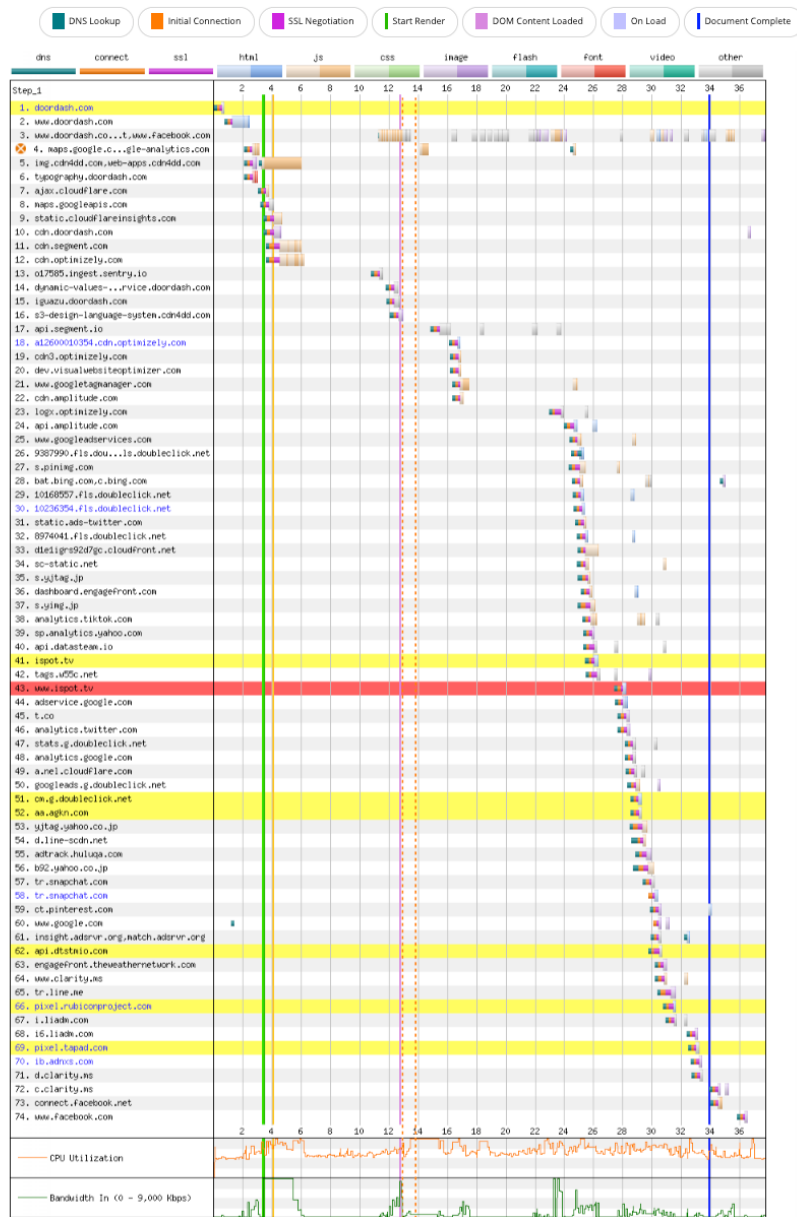
| TECHNOLOGY | TECHNOLOGY TYPE | VERSION | FIRST DETECTED |
|--|--------------------------------|---------|----------------|
|  SurveyMonkey | Surveys | | Jul 1, 2022 |
|  Adjust | Fraud Detection and Prevention | | May 1, 2022 |
|  AWS | Platform as a Service | | May 1, 2022 |
|  Bing Domain Verification | Domain Verification | | May 1, 2022 |
|  Cloudflare Rocket Loader | Widgets | | May 1, 2022 |
|  Envoy | Reverse Proxy | | May 1, 2022 |
|  Google Site Verification | Domain Verification | | May 1, 2022 |
|  Google Workspace | Web Mail | | May 1, 2022 |
|  X-UA-Compatible | Document Standards | | May 1, 2022 |
|  Cart Functionality | Ecommerce | | Feb 1, 2022 |
|  CloudFlare | Content Delivery Networks | | Jan 1, 2022 |
|  Cloudflare Certificate | Certificate Authority | | Jan 1, 2022 |
|  Cloudflare DNS | Name Server | | Jan 1, 2022 |
|  Cloudflare Hosting | Web Hosting | | Jan 1, 2022 |
|  Cloudflare Insights | Real user monitoring | | Jan 1, 2022 |
|  Cloudflare Server | Web Servers | | Jan 1, 2022 |
|  Description Meta | Document Standards | | Jan 1, 2022 |
|  Google Maps API | Maps | | Jan 1, 2022 |
|  JavaScript | Document Standards | | Jan 1, 2022 |
|  Next.js | Web Application Frameworks | | Jan 1, 2022 |

WebPageTest.org

WebpageTest.org is a free internet service to test the performance of the websites, web server and web services. It's used primarily to identify bottlenecks in a website and web server performance so developers can optimize accordingly. Similar services include GTMetrix and Lighthouse by Google.

Running a test of <https://DoorDash.com/> on WebpageTest returned detailed tests results of the network connections, servers, and resources that DoorDash.com encompasses further evidencing DoorDash's use of a webserver (see Ex 16). The test results show that door dash implements a content delivery network (CDN) for resource caching and website performance. They also show DoorDash's integration with Google Maps Facebook, CloudFlare, Bing, Pinterest, Amazon Web Services (S3), SnapChat, and connections to and from APIs as seen in this screen shot.

Connection View



For a full detail of the test results in JSON format see Exh. 16.

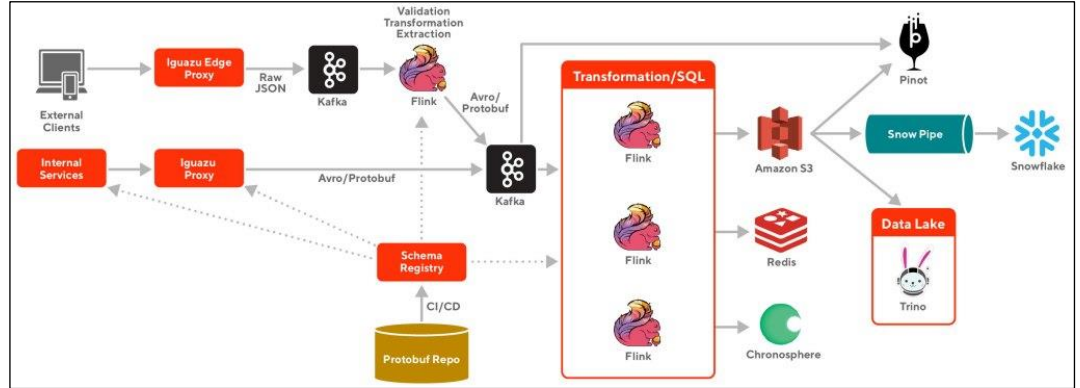
DoorDash Website

A Web Debugging Proxy tool captures request/response, messaging data and information about the connection to the server between a calling device and the DoorDash System.

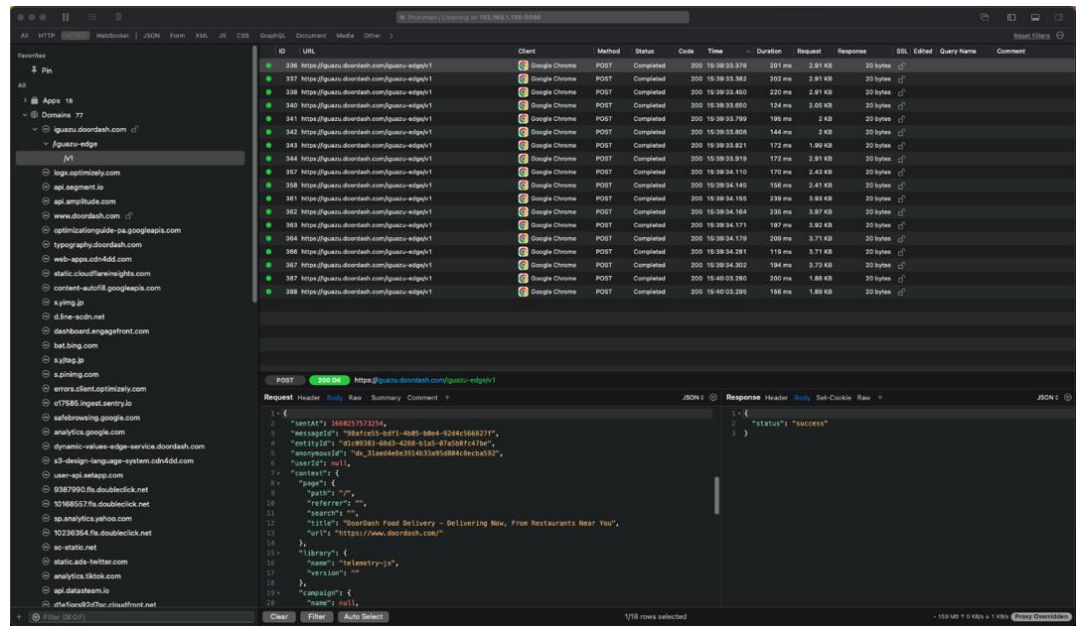
Displayed are a series of “messages” or request/response being communicated between server and client from the address Iguazu.DoorDash.com which is the address to their custom built DoorDash Event engine (called Iguazu). The diagram below depicts the inner functioning of Iguazu taking from the DoorDash technical video presentation “Scaling our Data Platform” (See Exh. 47) and the DoorDash engineering blog article

“Building Scalable Real Time Event Processing with Kafka and Flink” (See Exh. 53). In this blog article they state the following:

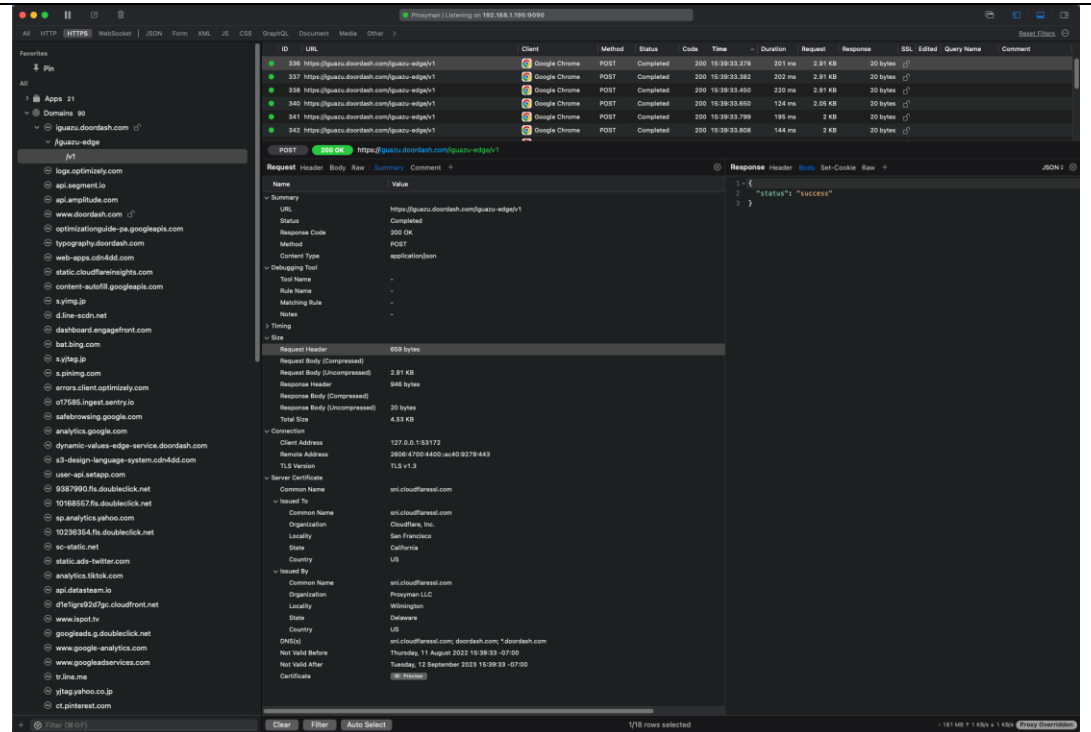
“...[in] the past two years, we built the real time events processing system and scaled it to process hundreds of billions of events per day with a 99.99% delivery rate. The overall architecture of the system is depicted in [the figure below]” (see Exh. 53):



Each message is a dataset containing various parameters used by the DoorDash system (client, server, etc.). Each message contains header information that details facts about the connection as should be below:




The summary tab provides the web server certificate authenticating the HTTPS connection, in this case as sni.cloudflaressl.com. This indicates that DoorDash has integrated CloudFlare’s SNI (Server Name Indication) into its custom webserver. According to CloudFlare “...SNI is an extension for the TLS protocol (formerly known as the SSL protocol), which is used in HTTPS. It’s included in the TLS/SSL handshake process in order to ensure that client devices are able to see the correct SSL certificate for the website they are trying to reach” (see Ex 7). We also see that the DNS lists CloudFlare and DoorDash.com.



Further, one can view the certificate authorizing the DoorDash web server as valid as seen below:

View Certificate

Proxyman CA (10 Aug 2022,
sni.cloudflaressl.com

 **sni.cloudflaressl.com**
 Issued by: Proxyman CA (10 Aug 2022,)
 Expires: Tuesday, September 12, 2023 at 3:39:33 PM Pacific Daylight Time
 This certificate is valid

Trust
 When using this certificate: Use System Defaults ?
 X.509 Basic Policy no value specified

Details

Subject Name
 Country or Region US
 Locality San Francisco
 Organization Cloudflare, Inc.
 Common Name sni.cloudflaressl.com
 State/Province California

Issuer Name
 Country or Region US
 Locality Wilmington
 Organization Proxyman LLC
 Common Name Proxyman CA (10 Aug 2022,
 Organizational Unit <https://proxyman.io>
 State/Province Delaware

Serial Number 2520296785
Version 3
Signature Algorithm SHA-256 with RSA Encryption (1.2.840.113549.1.1.1)
Parameters None

Not Valid Before Thursday, August 11, 2022 at 3:39:33 PM Pacific Daylight Time
Not Valid After Tuesday, September 12, 2023 at 3:39:33 PM Pacific Daylight Time

Public Key Info
Algorithm RSA Encryption (1.2.840.113549.1.1.1)
Parameters None
Public Key 256 bytes : F3 75 D9 72 0D 11 D4 57 ...
Exponent 65537
Key Size 2,048 bits
Key Usage Encrypt, Verify, Wrap, Derive
Signature 256 bytes : 7B 35 27 A5 F4 0E 0C EA ...

Extension Key Usage (2.5.29.15)
Critical NO
Usage Digital Signature, Non-Repudiation, Key Encipherment, Data Encipherment

Extension Basic Constraints (2.5.29.19)
Critical YES
Certificate Authority NO

Extension Extended Key Usage (2.5.29.37)
Critical NO
Purpose #1 Server Authentication (1.3.6.1.5.5.7.3.1)
Purpose #2 Client Authentication (1.3.6.1.5.5.7.3.2)

Extension Subject Key Identifier (2.5.29.14)
Critical NO
Key ID A4 03 8F DA 26 CE E8 D0 F5 EE 24 6B D4 4B 0F C4 51 8D CF BA

Extension Subject Alternative Name (2.5.29.17)
Critical NO
DNS Name sni.cloudflaressl.com
DNS Name doordash.com
DNS Name *.doordash.com

Extension Authority Key Identifier (2.5.29.35)
Critical NO
Data 30 00

Extension Netscape Certificate Comment (2.16.840.1.113730.1.13)
Critical NO
Data This Root certificate was locally generated by Proxyman for SSL Proxying. If you are browsing a website through Proxyman with SSL Proxying enabled, you would see this certificate as a part of a certificate chain. Please see <https://proxyman.io> for more in...

Fingerprints
SHA-256 02 4F 22 0A AF CB 39 71 0E B1 24 9A 96 A2 44 FA D0 76 5E 94 49 C9 D7 58 5E 13 64 F3 23 0A 78 69
SHA-1 7B B0 9B 12 96 03 3B D6 01 B7 B3 91 1C 6A BB 75 1A B0 CE 68

? OK

BuiltWith

BuiltWith is a tool that provides insights into the technologies used on websites. It can be especially helpful for understanding the tech stack of a particular company by analyzing the various web technologies and tools they utilize for their online presence.

For DoorDash, analytics and tracking technologies are a crucial part of their operation. According to the data from BuiltWith, some of the top analytics technologies used in the top 1 million sites include Google Analytics, Global Site Tag, Facebook Pixel, Facebook Signal, and Google Conversion Linker, among others. These tools are essential for administrators and executives as they provide real-time data and metrics which are vital for decision-making. The ability to track user behavior, conversion rates, and other key performance indicators (KPIs) helps in making informed strategic decisions, optimizing user experience, and improving service delivery. (See Exh. 122)

Developers integrate analytics tools with a website and its server by inserting specific code snippets provided by the analytics service into the site's pages. These code snippets, often referred to as tracking codes or tags, are designed to capture and send data to the analytics platform each time a user interacts with the website. This can include page views, clicks, form submissions, and more. The data collected is then processed and presented in the analytics dashboard. For more sophisticated tracking, developers may use tag management systems to deploy and manage multiple analytics and marketing tags to streamline integration and data collection processes.


Built With Reports ▾ Tools ▾ Plans Customers Account ▾ Help ▾


Home / doordash.com Technology Profile


DOORDASH.COM


Technology Profile Detailed Technology Profile Meta Profile Performance Profile Relationship


Analytics and Tracking View Global Trends


 **Optimizely**
[Optimizely Usage Statistics](#) · [Download List of All Websites using Optimizely](#)
Optimizely empowers companies to deliver more relevant and effective digital experiences on websites and mobile through A/B testing and personalization.
A/B Testing · Conversion Optimization · Personalization · Site Optimization


 **Adjust**
[Adjust Usage Statistics](#) · [Download List of All Websites using Adjust](#)
Mobile app tracking system.
Mobile

 **Recruitics**
[Recruitics Usage Statistics](#) · [Download List of All Websites using Recruitics](#)
Recruitment marketing and analytics.
Marketing Automation

 **Hubspot**
[Hubspot Usage Statistics](#) · [Download List of All Websites using Hubspot](#)
Hubspot provides marketing information and leads via inbound marketing software.
CRM · Marketing Automation

 **Spendgo**
[Spendgo Usage Statistics](#) · [Download List of All Websites using Spendgo](#)
Digital marketing solution for eCommerce.
Marketing Automation

 **Segment**
[Segment Usage Statistics](#) · [Download List of All Websites using Segment](#)
Segment gives you the ability to instrument your web app for analytics once, and then send your data to any number of analytics services. Previously known as Segment.io
Customer Data Platform

 **Amplitude**
[Amplitude Usage Statistics](#) · [Download List of All Websites using Amplitude](#)
Mobile analytics platform.
Mobile

Cloudflare Insights

[Cloudflare Insights Usage Statistics](#) · [Download List of All Websites using Cloudflare Insights](#)

Visitor analytics and threat monitoring.
Application Performance · Audience Measurement

Sift Science

[Sift Science Usage Statistics](#) · [Download List of All Websites using Sift Science](#)

Sift Science monitors a site's traffic in real time and alerts you instantly to fraudulent activity.
Fraud Prevention

New Relic

[New Relic Usage Statistics](#) · [Download List of All Websites using New Relic](#)

New Relic is a dashboard used to keep an eye on application health and availability while monitoring real user experience.
Application Performance

Google Analytics

[Google Analytics Usage Statistics](#) · [Download List of All Websites using Google Analytics](#)

Google Analytics offers a host of compelling features and benefits for everyone from senior executives and advertising and marketing professionals to site owners and content developers.
Application Performance · Audience Measurement · Visitor Count Tracking

Google Universal Analytics

[Google Universal Analytics Usage Statistics](#) · [Download List of All Websites using Google Universal Analytics](#)

The analytics.js JavaScript snippet is a new way to measure how users interact with your website. It is similar to the previous Google tracking code, ga.js, but offers more flexibility for developers to customize their implementations.

Google Analytics 4

[Google Analytics 4 Usage Statistics](#) · [Download List of All Websites using Google Analytics 4](#)

Google Analytics 4 formerly known as App + Web is a new version of Google Analytics that was released in October 2020.

Twitter Analytics

[Twitter Analytics Usage Statistics](#) · [Download List of All Websites using Twitter Analytics](#)

A tool that helps website owners understand how much traffic they receive from Twitter and the effectiveness of Twitter integrations on their sites. Includes Twitter Conversion Tracking.
Conversion Optimization

Bing Universal Event Tracking

[Bing Universal Event Tracking Usage Statistics](#) · [Download List of All Websites using Bing Universal Event Tracking](#)

Universal Event Tracking (UET) is a simple and powerful campaign measurement solution that allows you to track key conversion goals important to your business.
Conversion Optimization · Retargeting / Remarketing

Facebook Signal


[Facebook Signal Usage Statistics](#) · [Download List of All Websites using Facebook Signal](#)

Journalists use Signal to surface relevant trends, photos, videos and posts from Facebook and Instagram for use in their storytelling and reporting.

Facebook Pixel

[Facebook Pixel Usage Statistics](#) · [Download List of All Websites using Facebook Pixel](#)

Facebook Pixel is Facebook's conversion tracking system for ads on Facebook to websites.

 **Facebook Conversion Tracking**


[Facebook Conversion Tracking Usage Statistics](#) · [Download List of All Websites using Facebook Conversion Tracking](#)

Conversion tracking functionality from Facebook, allows a user to track advertisement clicks.
Conversion Optimization

 **DoubleClick Floodlight**

[DoubleClick Floodlight Usage Statistics](#) · [Download List of All Websites using DoubleClick Floodlight](#)

Floodlight is feature of DoubleClick ads that allows advertisers to capture and report on the actions of users who visit their website after viewing or clicking on one of the advertiser's ads.

 **Global Site Tag**

[Global Site Tag Usage Statistics](#) · [Download List of All Websites using Global Site Tag](#)

Google's primary tag for Google Measurement/Conversion Tracking, Adwords and DoubleClick.

 **Twitter Website Universal Tag**

[Twitter Website Universal Tag Usage Statistics](#) · [Download List of All Websites using Twitter Website Universal Tag](#)

A tool from Twitter that makes it possible for advertisers to track website conversions and manage tailored audience campaigns.

 **Twitter Conversion Tracking**

[Twitter Conversion Tracking Usage Statistics](#) · [Download List of All Websites using Twitter Conversion Tracking](#)

Twitter ads conversion tracking code.
Conversion Optimization · Conversion Tracking

 **Google Conversion Linker**

[Google Conversion Linker Usage Statistics](#) · [Download List of All Websites using Google Conversion Linker](#)

Detects the ad click information in your conversion page URLs and stores this information to associate an ad click with a conversion.

 **LinkedIn Insights**

[LinkedIn Insights Usage Statistics](#) · [Download List of All Websites using LinkedIn Insights](#)

The LinkedIn Insight Tag is a piece of lightweight JavaScript code that you can add to your website to enable in-depth campaign reporting and unlock valuable insights about your website visitors and for conversion optimization of ads.
Conversion Optimization

 **Cloudflare Web Analytics**

[Cloudflare Web Analytics Usage Statistics](#) · [Download List of All Websites using Cloudflare Web Analytics](#)

Privacy-first web analytics from Cloudflare.
Audience Measurement

 **TikTok Conversion Tracking Pixel**

[TikTok Conversion Tracking Pixel Usage Statistics](#) · [Download List of All Websites using TikTok Conversion Tracking Pixel](#)

TikTok advertising conversion tracking pixel.
Conversion Optimization · Conversion Tracking

| | |
|--|--|
| | <div data-bbox="462 119 1469 745" style="border: 1px solid #ccc; padding: 10px;"> <p> Reddit Conversion Tracking Reddit Conversion Tracking Usage Statistics · Download List of All Websites using Reddit Conversion Tracking Conversion tracking system from Reddit. Conversion Optimization</p> <p> Claritas Claritas Usage Statistics · Download List of All Websites using Claritas Custom audience segments & consumer insights for over 120 million households</p> <p> Riskified Riskified Usage Statistics · Download List of All Websites using Riskified eCommerce fraud and charge back solutions. Fraud Prevention</p> <p> Crimson Hexagon Crimson Hexagon Usage Statistics · Download List of All Websites using Crimson Hexagon AI-Powered Consumer Insights tracking platform. Audience Measurement</p> </div> <p>The accused instrumentality meets this limitation under the doctrine of equivalents because the accused instrumentality performs substantially the same function, in substantially the same way to achieve substantially the same result. For example, DoorDash's distributed computing platform, powered by web servers, mobile apps, and backend architecture, effectively serves as the backbone of their online hospitality service, akin to a web-based communication network for users and restaurants. Their use of web server technologies like Next.js, Express.js, and Node.js is equivalent to backend systems that enable rapid web page rendering and optimization, while the transition to a microservices architecture reflects a shift towards a more flexible and reliable structure. The implementation of a backend-for-frontend (BFF) architecture ensures seamless interaction, and the DoorDash Node.js SDK streamlines API integration. Additionally, their hosting on Amazon Web Services (AWS) signifies a reliable and scalable infrastructure for web-based services, altogether representing an equivalent comprehensive online platform for facilitating user-restaurant interactions.</p> |
| <p>one or more hospitality software applications linked with the back-office servers and with handheld/mobile compatible versions available to be remotely accessed and used by handheld/mobile equipped users and including two or more different handheld/mobile computers with their respective and different mobile operating systems;</p> | <p>DoorDash's architecture exemplifies this claim element with its suite of hospitality software applications seamlessly integrated with back-office servers. These applications are designed for compatibility across a diverse range of handheld devices and mobile operating systems, ensuring remote access and usability for users on the go. This cross-platform functionality facilitates uninterrupted and efficient service delivery, allowing DoorDash to maintain robust connectivity with customers and partners, regardless of the device or operating system in use.</p> <p>Redact POS</p> <p>DoorDash integrates with the Redcat point-of-sale (POS) "...[seamlessly] integrate orders from DoorDash, Caviar, and Storefront" (See Exh. 95). In this DoorDash Merchant Support document concerning Redcat integration, it states the following overview of this integration:</p> <p style="text-align: center;"><i>"Integrating DoorDash with your POS through Redcat simplifies order management and operations with:</i></p> |

- *Streamlined orders - No need to manually enter orders from a tablet, email, or phone to your POS. Orders will go right to the kitchen, minimizing room for error and freeing up your staff to focus on your in-person guests.*
- *Automatic menu updates - Any changes you make to your POS menu will be automatically reflected on your DoorDash menu.*
- *Increased order accuracy - Because menu changes are synced through Redcat, customers are less likely to order an out-of-stock item.*
- *Consolidated accounting & reporting on your POS - DoorDash orders are tagged to a unique payment type and dining option to let you view your business's performance on your POS."*

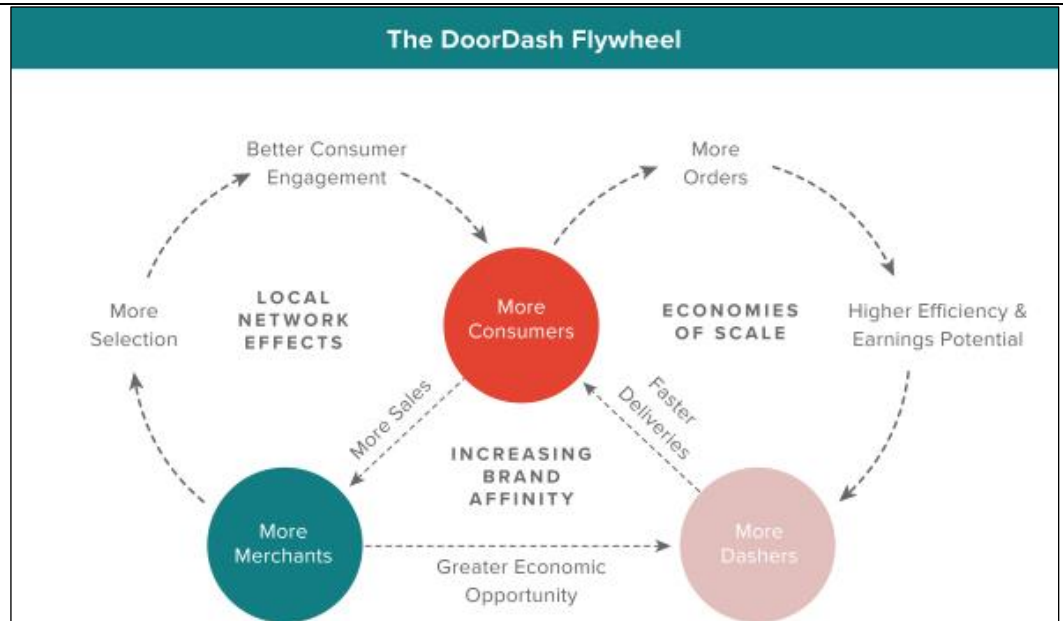
ItsaCheckmate POS

DoorDash integrates with the ItsaCheckmate point-of-sale (POS) "...[seamlessly] integrate orders from DoorDash, Caviar, and Storefront" (See Exh. 96). In this DoorDash Merchant Support document concerning Redcat integration, it states the following overview of this integration:

"Integrating DoorDash with your POS through ItsaCheckmate simplifies order management and operations with:

- *Streamlined orders - No need to manually enter orders from a tablet, email, or phone to your POS. Orders will go right to the kitchen, minimizing room for error and freeing up your staff to focus on your in-person guests.*
- *Automatic menu updates - Any changes you make to your POS menu will be automatically reflected on your DoorDash menu.*
- *Fewer order cancellations - Menu changes, including 86'd items, are synced as needed through ItsaCheckmate which saves customers from ordering an out-of-stock item.*
- *Consolidate accounting & reporting on your POS - DoorDash orders are tagged to a unique payment type and dining option to let you view your business's performance on your POS."*

As confirmed by DoorDash Vice President Jessica Lachs, (above/below) as part of its '360 degree picture' and framework approach, DoorDash integrates its intelligent web servers, with its MFCCS, and hospitality food/drink ordering application, (while including programming to intelligently choose and apply multiple modes of communications and/or different protocols if applicable) to enable the completion of the user requested hospitality food/drink, ordering tasks for delivery or pick up and as part of its Flywheel marketplace. The DoorDash Flywheel diagram, shown below, makes evident that the DoorDash operates in accordance with its overall series of linked services and via its "360-degree picture" central ("Flywheel") and technology platform framework:



DoorDash's Vice President of Analytics and Data Science Jessica Lachs in an August 17, 2022, interview on "Leveraging Data to Delight Customers Despite a Challenging Supply Chain" in which she states (See Exh. #98):

"And so for us, it's really about collecting as much information as we can about all sides of the marketplace, bringing all of that data together into a central data platform, where all of that data is accessible no matter the source. Whether it is coming from our production systems, transactional data, whether it is event data in our apps, whether that's the consumer app, the dasher app, the merchant app... whether it is coming from our CRM systems. All of that data needs to come in to one central place so that we can tie it together and use the insights together to create a 360 degree picture of what's happening on our platform and off our platform so that we can use that information not just to provide accurate menus and inventory for consumers but also so we can send the right email communications to consumers, to dashers, so that we really have a full picture of what's happening and can use that for personalization and to help all three sides of our marketplace really optimize that they are at their peak efficiency."

"So, for us, we want data to be easily accessible to all the different teams that need access to it. Analytics, being one of the largest customers of data at DoorDash, of course, but the way we think about our data models is really about increasing accessibility and consistency to that data. So, having all of our data in one central place and making sure that it is high in performance and so like query speeds are fast and that data models are thoughtful, so that it makes it a lot easier for data scientists, analysts, operators, product managers to be able to query the data that is needed and use the data in our production, in our production systems as well. So, we try to be thoughtful about how we structure our data models and how we ensure that all of the different production systems tie together into that central, as you mentioned, that central data lake."

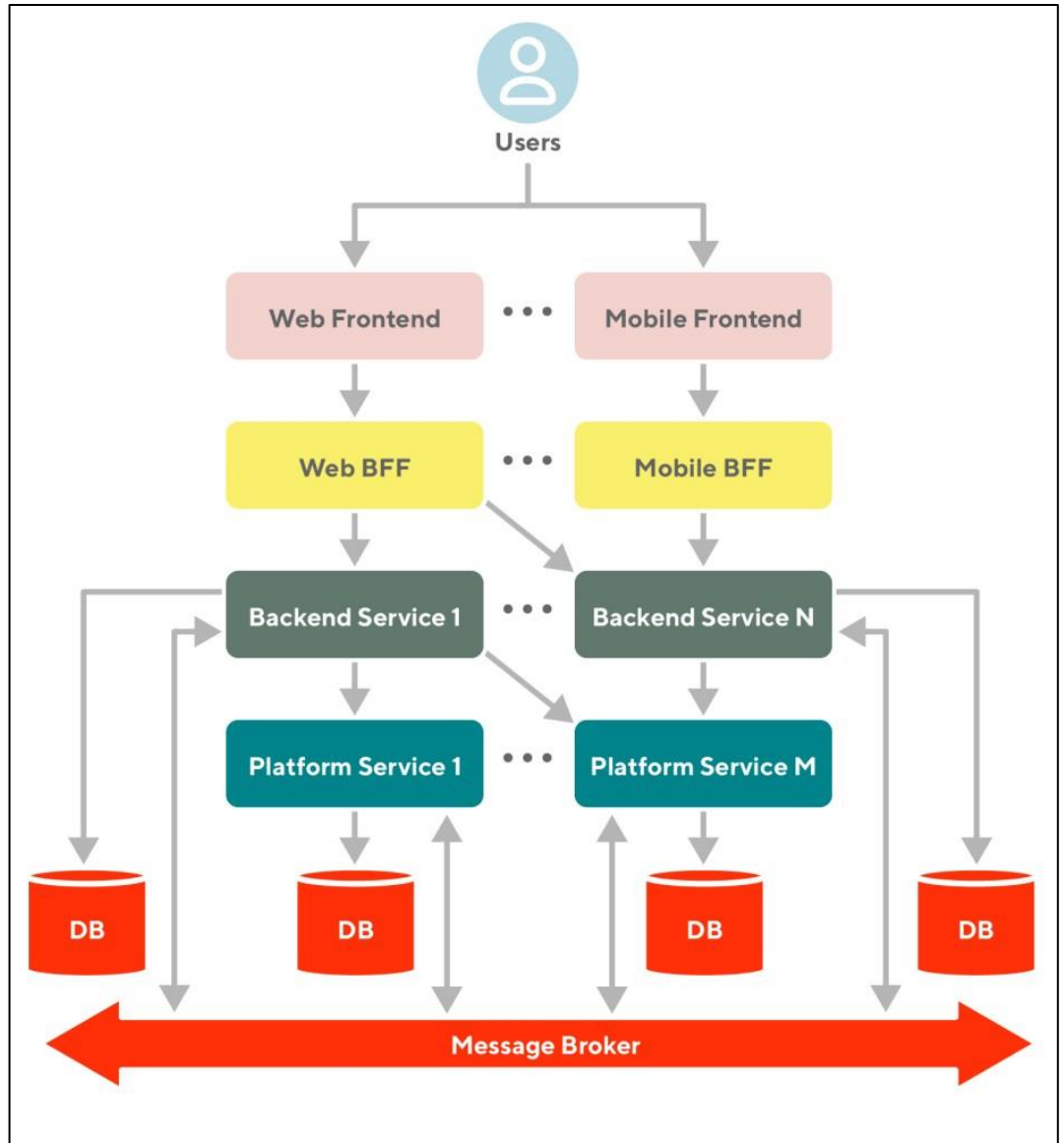
Microservices Architecture

The original DoorDash platform was originally a monolithic application written in Python using the Django web framework with a PostgreSQL database. As the platform

grew, they started to have problems with reliability and scaling. Around 2018 they institute a code freeze and began migrating to microservices. At this time, they also migrated to the Kotlin language, and their services now run on the JVM (See Exh. 19, 20, 21, 22).

DoorDash was able to improve the efficiency and reliability of their platform using a microservices architecture. By breaking up their application into domain-specific parts they were able to reduce errors and latency. They state in the article “Future-proofing: How DoorDash Transitioned from a Code Monolith to a Microservice Architecture” (see Exh. 20) that

“... final design for our new microservice architecture consisted of five different layers, ranging from the user experience to the core infrastructure. Each layer provides functionality to the upper layer and leverages the functionality exposed by the lower layer [as shown below]”:



These layers include (see in Exh 20):

“Frontend layer: Provides frontend systems (like the DoorDash mobile app,

Dasher web app, etc.) for the interaction with consumers, merchants, and Dashers that are built on top of different frontend platforms.

BFF layer: The frontend layer is decoupled from the backend layer via BFFs. The BFF layer provides functionality to the frontend by orchestrating the interaction with multiple backend services while hiding the underlying backend architecture.

Backend Layer: Provides the core functionality that powers the business logic (order cart service, feed service, delivery service, etc.).

Platform layer: Provides common functionality that is leveraged by other backend services (identity service, communication service, etc.).

Infrastructure layer: Provides the infrastructural components that are required to build the site (databases, message brokers, etc.) and lays the foundation to abstract the system from the underlying environment (cloud service provider)."

In their new architecture they introduced the backend-for-frontend (BFF) "... an application connecting the consumer-facing client and the services providing general purpose APIs. Client requests go to the BFF, which then orchestrates the aggregation of information needed by the client.". The BFF is a software architecture pattern (See Exh. 40) used by microservices which "...shifted from thick-client applications to **interfaces delivered via the web, a trend that has also enabled the growth of SAAS-based solutions in general**". As such BFF can be considered as "...the user-facing application as being two components - a client-side application living outside your perimeter, and **a server-side component (the BFF)** inside your perimeter". The perimeter of the BFF is the webserver.

Integrated In-App Texting

DoorDash integrates texting between customers and Dashers into its iOS and Android app. In the blog article by the DoorDash engineering team "Building Chat Into the DoorDash App to Improve Deliveries" they state

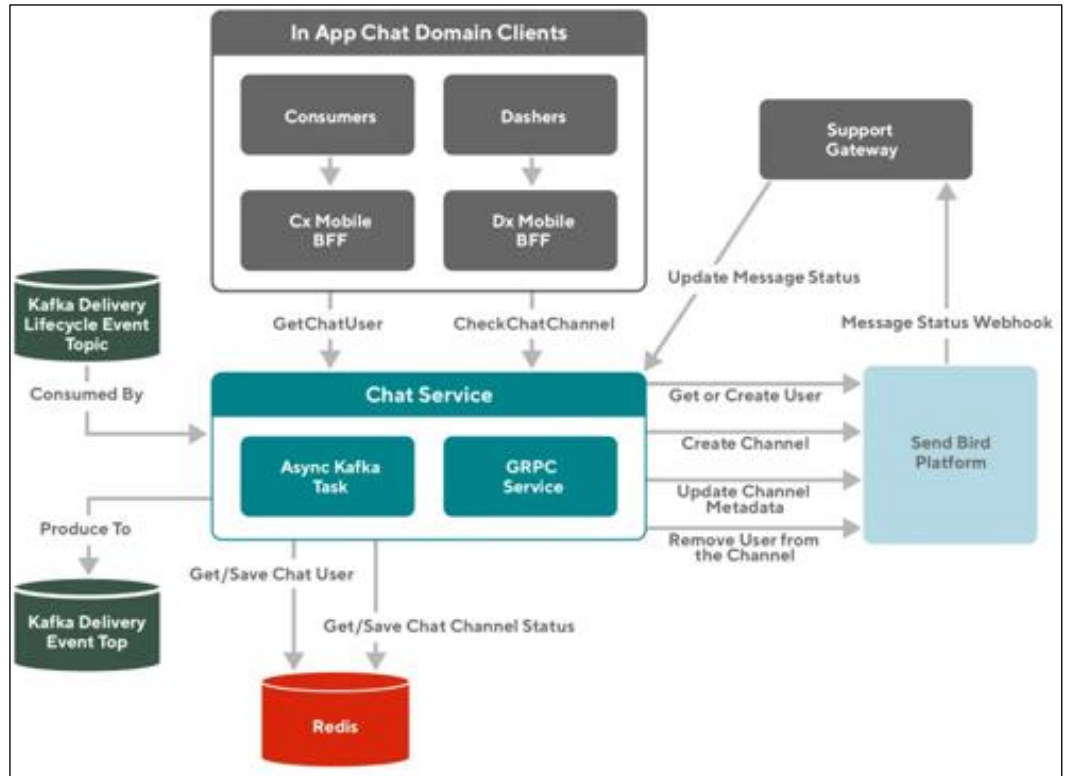
"... [b]uilding chat into our apps may seem redundant, as smartphones already support calling and texting. However, building our own chat client has several advantages over these external channels. Our chat client creates a buffer between Dashers and customers, ensuring privacy for both parties, and lets us better resolve customer support issues."

The problem with using external communication channels to connect Delivery Drivers with customers is that "...a Dasher attempting a delivery might try to call or text the customer if finding the location proved difficult or other issues arose. Although this method may be effective, it requires the Dasher to switch away from the app, which shows the delivery address and other important details" (See Exh. 71, 72).

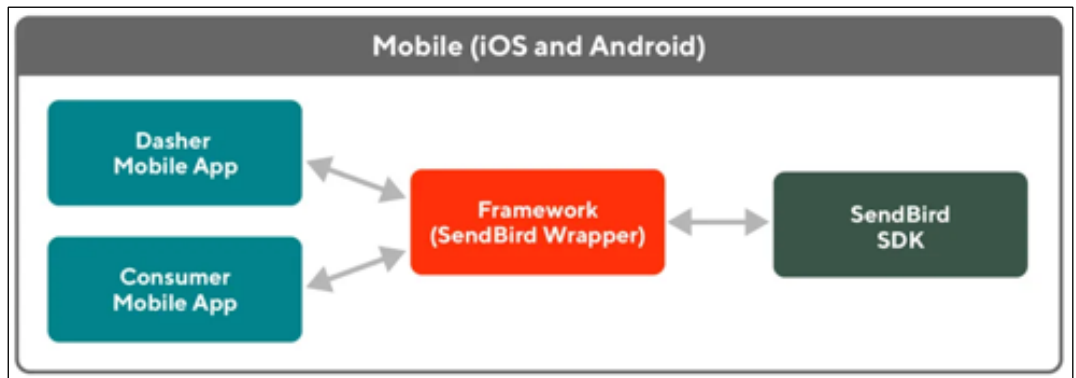
DoorDash integrates SendBird's chatting technology into its platform by implementing their Chat API and native Chat SDK to solve these problems. They state that the system consists of

"...four main components: mobile clients, chat service, data store, and SendBird SDKs. The in-app chat entry points have been added to all of DoorDash's mobile clients. Mobile clients connect to the chat service to get the chat user and chat channels. The chat service handles communications between the data store and

SendBird platform APIs. In the data store we maintain users and channels that get created by SendBird. The chat service uses SendBird to create users, chat channels, and channel metadata, and then stores that data”, as illustrated by the diagram below:



The DoorDash Mobile Apps, iOS and Android are integrated with SendBird’s SDK as depicted in this diagram:



In the blog article “Building Chat Into the DoorDash App to Improve Deliveries” by the DoorDash Engineering team (See Exh. 71) they state that DoorDash integrated chat into their apps as opposed to relying in the built-in chat/texting of the smartphones because “...creates a buffer between Dashers and customers, ensuring privacy for both parties, and lets us better resolve customer support issues.” The article states:

Every delivery enabled by the DoorDash platform is different. Dashers (our term for delivery drivers) meet customers in a wide range of contexts, from apartment and office building lobbies to suburban homes. This variety of circumstances and the timely nature

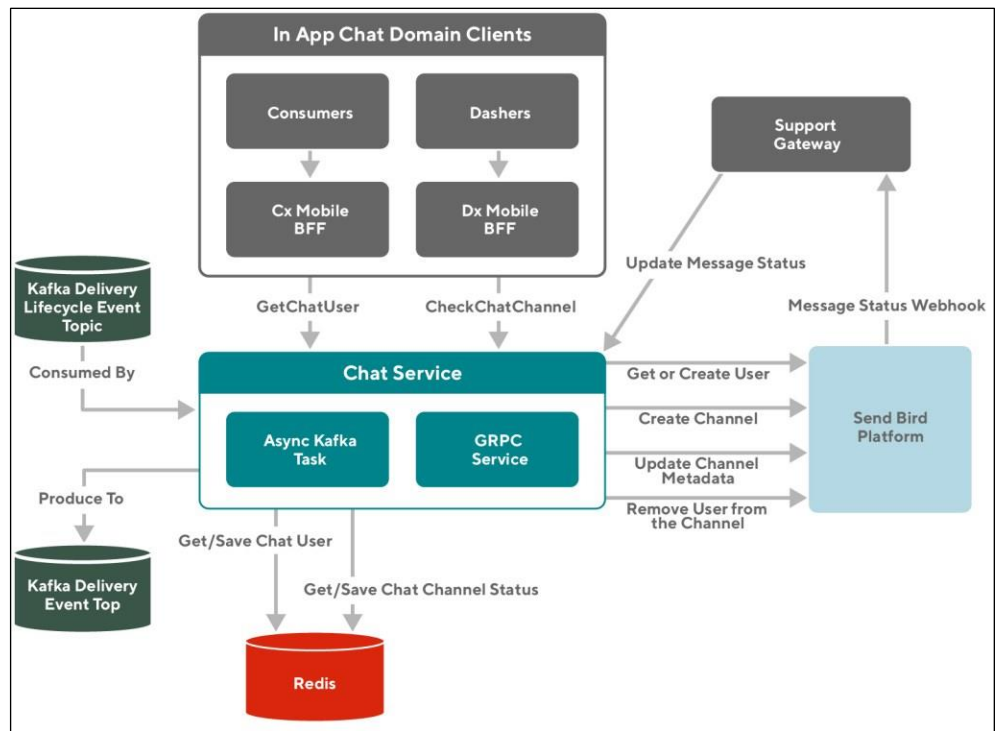
of contact makes communication essential, which is why we built chat into the DoorDash apps.

“Building chat into our apps may seem redundant, as smartphones already support calling and texting. However, building our own chat client has several advantages over these external channels. Our chat client creates a buffer between Dashers and customers, ensuring privacy for both parties, and lets us better resolve customer support issues.

Creating a chat client requires quite a bit of engineering and, given our past success integrating existing solutions, we began this project by evaluating third-party software. After settling on a chat technology, we integrated it into our platform and apps.”

Regarding the integration of SendBird (See Exh. 72) into the DoorDash iOS and Android apps, the article specifies the main aspects of the technical system design, how they integrated the SendBird UI into their custom apps, and the final results, stating specifically the following:

“Our system design consists of four main components: mobile clients, chat service, data store, and SendBird SDKs. The in-app chat entry points have been added to all of DoorDash’s mobile clients. Mobile clients connect to the chat service to get the chat user and chat channels. The chat service handles communications between the data store and SendBird platform APIs. In the data store we maintain users and channels that get created by SendBird. The chat service uses SendBird to create users, chat channels, and channel metadata, and then stores that data.”



“[In the above figure, the DoorDash] ...in-app chat system design consists of four main components: mobile clients, chat service, data store, and SendBird SDKs. The chat service receives chat user and channel requests from the mobile

clients, sending them to the SendBird platform. SendBird creates channels and users and updates the message status. The chat service also stores the chats in a Redis data store.”

Note: Redis is an “...open source, in-memory data store used by millions of developers as a database, cache, streaming engine, and message broker” (See Exh. 90).

For the DoorDash mobile apps (iOS and Android), the article explains how they built a “...a framework that serves as a wrapper around the SendBird UI SDK” and states the following:

“On the mobile side we built a framework that serves as a wrapper around the SendBird UI SDK. The framework adds UI styling and configuration on top of the chat SDK UI. The framework is integrated in both consumer and Dasher mobile apps. Building the framework helped us to isolate the chat-related logic and maintain it in one place.

Integrating chat into our mobile apps required work on four separate codebases. We have the consumer and Dasher apps, with Android and iOS versions of each.

Because chat sessions are temporary events, we chose Redis as our data store, using it as a cache layer for the user info and chat channel status. Redis was also an easy choice, as it is already available in our cloud infrastructure.

SendBird stores the actual chat text on its platform, which is SOC 2 certified and GDPR compliant, ensuring security and privacy. A pipeline between SendBird and our customer service system lets authorized users view chat histories to help resolve support issues.”

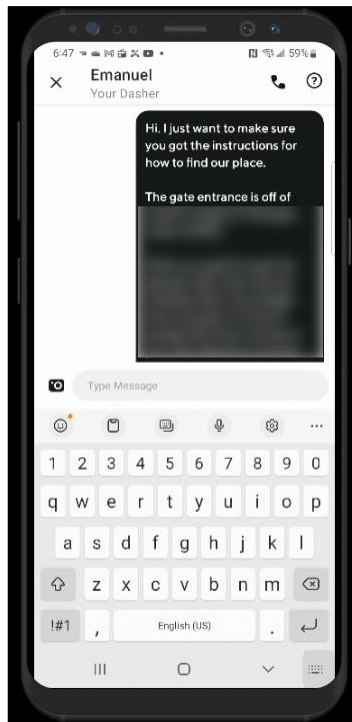
The article includes with very interesting results integrating a custom chat technology into their mobile apps:

“Adding in-app chat to our consumer and Dasher apps decreased the number of orders showing as Never Delivered, a metric we use to measure how many deliveries were missed. One of the most common causes of Never Delivered is poor communication between a consumer and Dasher.”

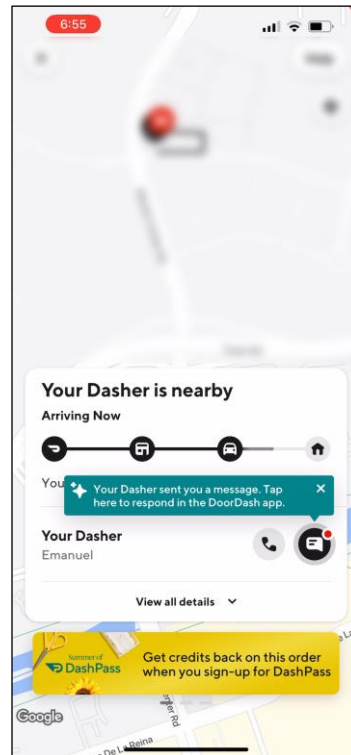
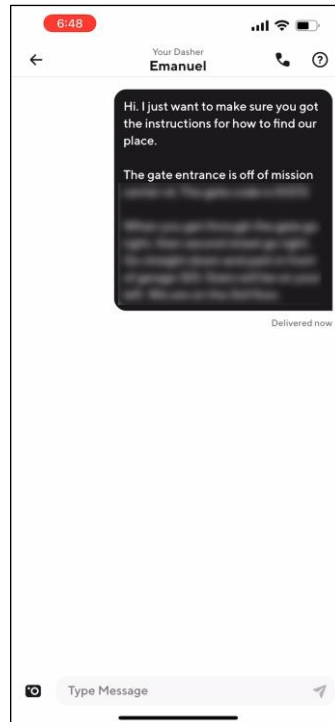
Order Tracking and Notification

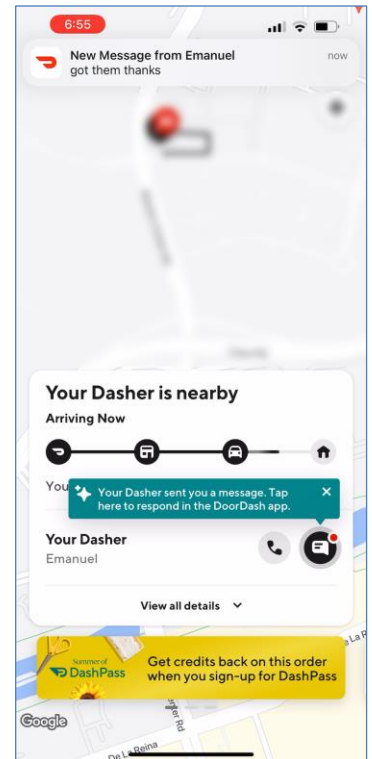
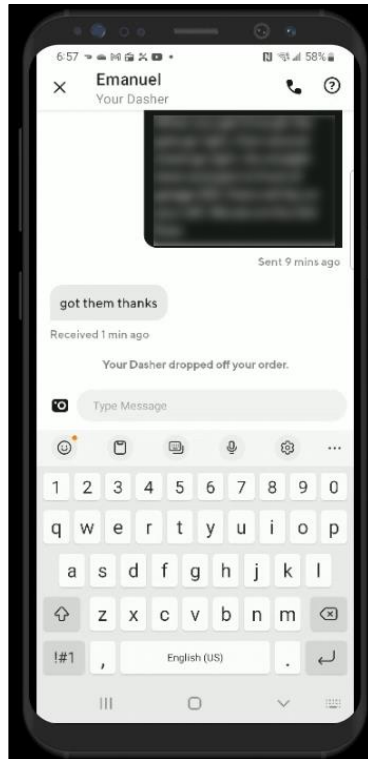
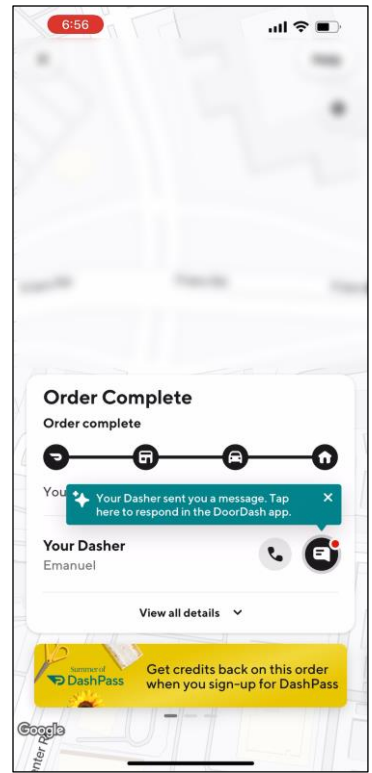
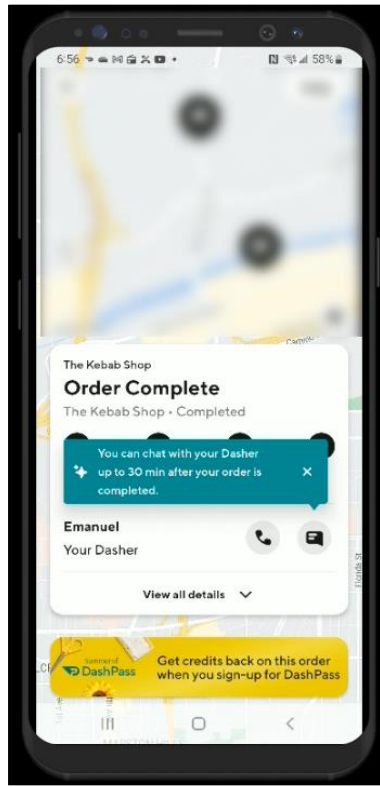
The same holds true for order notifications, order tracking and texting/chatting between the dasher and user. Below are a series of screens taken from one iPhone and one Android each tracking the same order placed by a user in the Android App. They depict the notifications and updates in the app for the user to be able to track their order. In this case, we show see text messages placed and sent by the user to the Dasher on the android app, being reflected the iOS app on an iPhone that is tracking the order. When the Dasher responds to the chat from the user, the user is notified on the android phone and on the iOS phone.

Android DoorDash App



iOS DoorDash App





Business Manager App

DoorDash offers the Business Manager app for iOS and Android. It lets restaurants “...track orders in progress, resolve issues, access support, monitor business performance, and get real-time notifications — all on your phone.” Selective

screenshots from the DoorDash Business Manager App Guide describes the app, and it's features as below (See Exh. 59, 62, 65). With the Business Manager App, managers can in real-time:

- “track sales, orders, and trends, for one or multiple stores
- change store status
- update menu hours
- view active orders
- call the customer or Dasher
- contact DoorDash Support
- mark items out of stock
- cancel an order”

INTRODUCTION

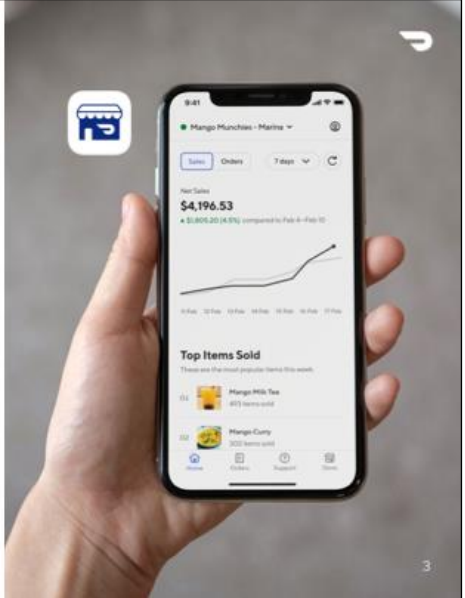
DoorDash - Business Manager App

Manage your DoorDash business right on your phone

Running a business isn't easy, but it could definitely be a little easier.

The new DoorDash - Business Manager app lets you track orders in progress, resolve issues, access support, monitor business performance, and get real-time notifications — all on your phone.

This guide will walk you through downloading, installing, and using the new app.



INTRODUCTION

A note on Point-of-Sale (POS) Integrations



What's a POS integration?

A POS integration is a streamlined connection between your point-of-sale (POS) software and DoorDash's platform. This connection lets you receive DoorDash orders directly on your POS, so everything is in one place and there's no need to manually enter orders from one system into another.

Why is the user flow different depending on whether or not I have a POS?

When businesses use a POS system, some of their store information is stored directly on that system instead of on DoorDash. This means that the user experience for stores using POS system differs for certain actions and updates.

| Use your POS system to: | Use the DoorDash Business Manager app to: | Call DoorDash Support to: |
|---|--|--|
| <ul style="list-style-type: none"> Update your menu (Items, modifiers, and photos) Change menu and store hours Mark items out of stock | <ul style="list-style-type: none"> Receive push notifications Track business health View active orders Call customer, dasher, or support | <ul style="list-style-type: none"> Cancel live orders |

Throughout this guide, sections with separate POS instructions will be marked using this tag:

[POS Integrations](#)

Integration features vary by provider. Reach out to your integration provider or refer to these [help articles](#) for more information.

4

HOW TO

Install the app - iOS

- 1 Download the app with [this link](#) or QR Code below.
- 2 Click on **Install** and let the app download.
- 3 Once installed, tap **Open** to launch the app for the first time.



Apple App Store



HOW TO

Install the app - Android

- 1 Download the app with [this link](#) or QR Code below.
- 2 Click on **Install** and let the app download.
- 3 Once installed, tap **Open** to launch the app for the first time.



Google Play Store

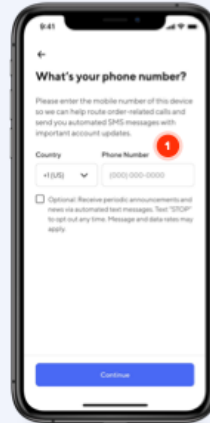


5

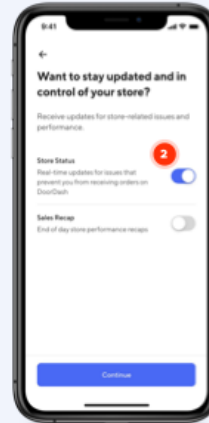
HOW TO

Enable push notifications

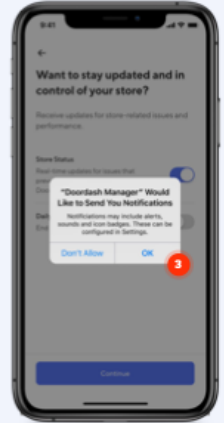
- 1 When you first log in, you'll be asked to provide your **phone number for authentication**. Once you've done that, tap Continue.
- 2 **Enable push notifications** in the toggle bar to receive real-time updates about store issues and end-of-day performance recaps. Tap Continue.
- 3 You'll see a pop-up asking you to allow **DoorDash Manager to send you notifications**. Be sure to select **OK** for full functionality.
- 4 If you select Don't Allow, you'll need to **re-enable notifications** via your device's Settings menu.



1st time authentication



Notifications selector



Accept device notifications

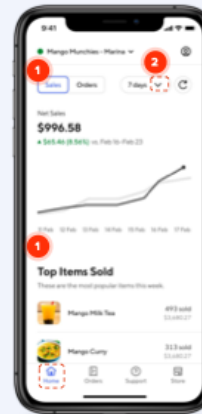
6

HOW TO

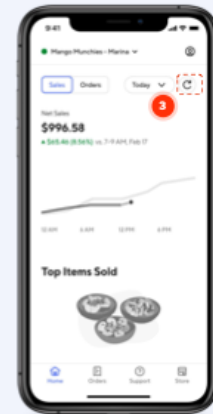
Track business health (single store)

- 1 The Home tab of your Dashboard allows you to see **sales, orders, trends and top items sold** in a specific time range.
- 2 You'll have the option to choose from today, yesterday, 7 days, 30 days. Just tap the dropdown arrow.
- 3 If you select **today**, you'll be able to:
 - Monitor sales and orders in real time by tapping the refresh arrow
 - Analyze partial-day trends

Menu item performance is calculated at 7 days and 30 days, so to view that data just change the time range.



View top-line metrics and trends

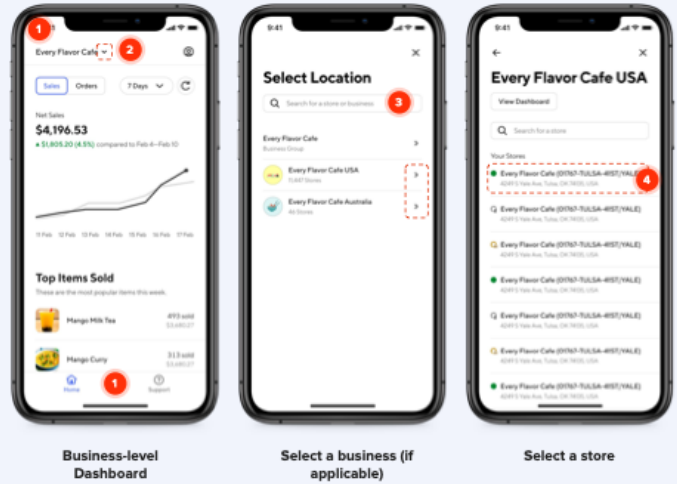


Monitor day-of performance

8

HOW TO
Track business health (multiple stores)

- 1 If you own **multiple locations**, the home screen will display metrics across all stores by default. You can confirm this by noting:
 - The business (or group) name at the top left
 - Your bottom navigation bar will only show 2 tabs (Home and Support)
- 2 To view **store-level metrics and features**, tap the arrow in the top navigation bar.
- 3 Depending on how your business was set up on DoorDash, it may take 1-2 taps to get to an individual store (tap the **arrow to navigate**). You can also locate the store in the **search bar**.
- 4 **Tap the store** you'd like to view and both the app dashboard and metrics will automatically refresh.



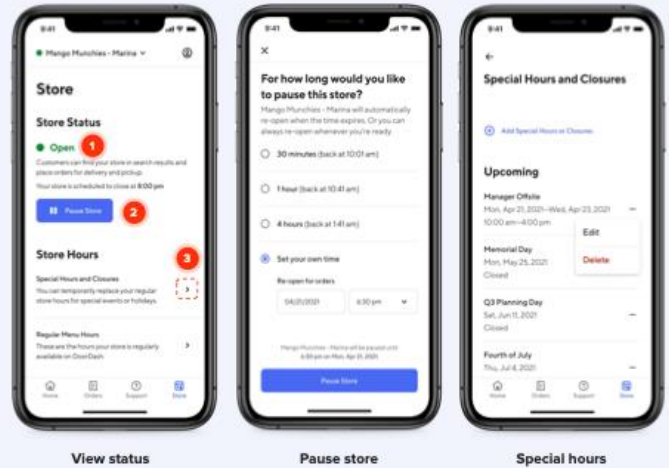
HOW TO
Change store status

POS Integrations

- 1 You can view and change your store status (**open, paused, closed**) in the Store tab. If you have access to multiple stores, select the specific store from the dropdown menu.
- 2 Tap **Pause Store** to pause an open store and choose when it reopens (in an hour, the next day, etc).
- 3 View or add special hours or closures by tapping the right-hand arrow next to **Special Hours and Closures**.*

When you change your store status in the app, it will be reflected on the Tablet and Portal (and vice versa).

*If you use a POS protocol, we recommend making special hours updates and closures using your POS. When you make these changes via the app, they may be overwritten during a POS sync.



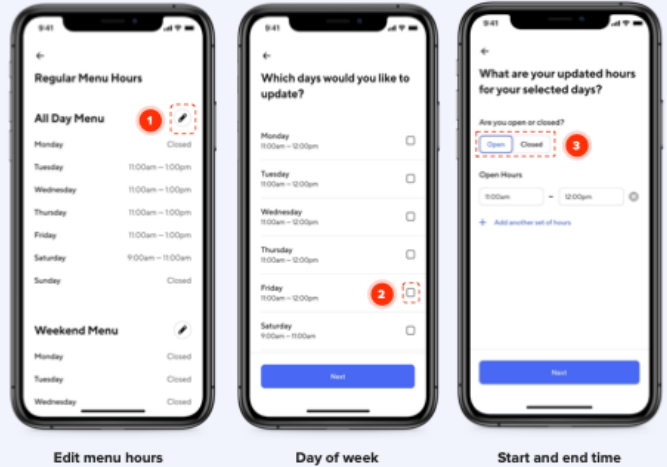
HOW TO
Update menu hours*

POS Integrations

- 1 From the Store tab (refer to previous page for a screenshot) you can update your menu hours by tapping the right-hand arrow next to **Regular Menu Hours**.
- 2 If you have multiple menus for your store, scroll down to find the specific menu. Then tap the **pencil to edit**.
- 3 **Check the box** next to the day(s) of the week you'd like to update the menu hours for and tap **Next**.

Indicate whether the new hours reflect when you're open or closed by clicking on the top toggle. Then **tap on the start or end time** and select the desired time from the dropdown. Tap **Next** and then **Confirm**.

*If you use a POS protocol, we recommend updating menu hours using your POS. When you make these changes via the app, they may be overwritten during a POS sync.



Edit menu hours

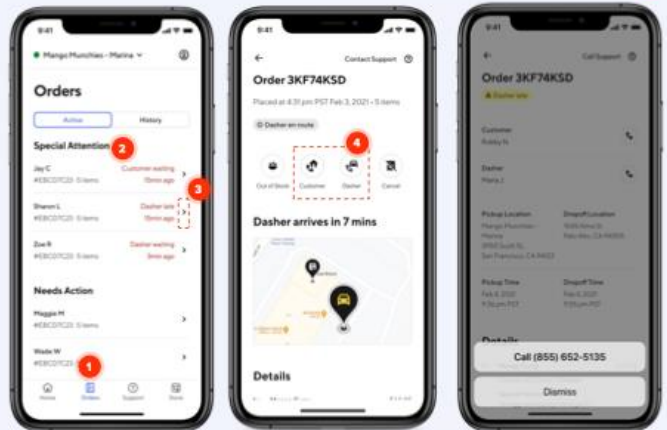
Day of week

Start and end time

11

HOW TO
View active orders

- 1 **Tap on the Orders button** in the bottom navigation to view active orders. For already completed orders, you can also toggle to History.
- 2 **Note the order status** on the right-side to see if an order needs attention.
 - ▲ Dasher late
 - 🕒 Dasher en route
 - 👤 Dasher arriving
 - 👤 Customer pickup
 - ▲ Dasher waiting
 - 👤 Customer arriving
 - 📦 Delivered
 - 👤 Customer picked up
- 3 **See order details** by tapping on the right-hand arrow by the individual order.
- 4 **Call a customer or Dasher directly** by tapping the phone icon.



Orders

Order details

Call customer or Dasher

12

HOW TO

POS Integrations

Call the customer or Dasher*

If you use a Tablet protocol, you can continue to call customers or Dashers for in-progress orders by tapping the phone icon or Issue with Order.

Now you can also call from the app by **clicking on the phone icon by the customer's or Dasher's name**.

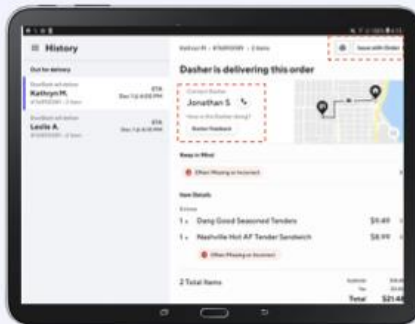
Common reasons to call a customer or Dasher:

- Out-of-stock item
- Clarification on special instructions
- Order running late
- Dasher arrives early
- Dasher waiting but not coming inside

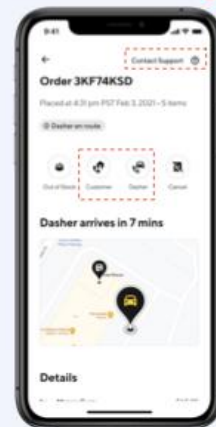
Afterwards, if you need to make any requested **order changes**, you can do so by chatting with our Support team — just tap **Contact Support** at the top right.

Your Tablet is still the best way to adjust prep time or mark an order ready for pickup.*

*If you use a POS integration, there's no need to mark orders ready for pickup; they will be confirmed and assigned a Dasher automatically.



Using Tablet to call customer or Dasher



Call customer or Dasher

13

HOW TO

Contact Support

1 The **Support icon** is always accessible on the bottom navigation.

2 If you have access to multiple stores, be sure to select the right store from the dropdown.

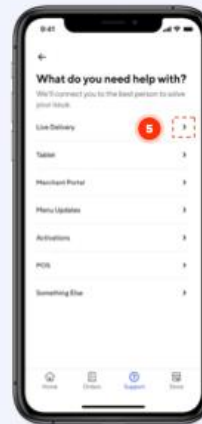
3 To start a new request, tap **General Support** and then select the nature of your question in the next screen.

4 If you'd like to reopen a previous conversion, you can do so by tapping one of the Recent Messages.

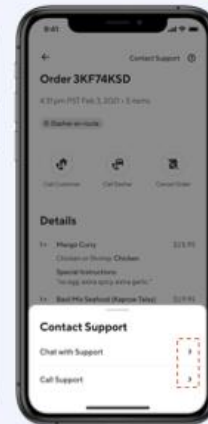
5 For any issues with In-progress orders, tap **Live Delivery**. You can then choose if you'd like to **call Support** or **chat with Support**.



Support



Live delivery



Chat or call

14

HOW TO POS Integrations

Mark out of stock

- 1 Click on an active order to note when an item is **out of stock**.
- 2 Based on the order details, you'll select which **item or modifier** is out of stock.
- 3 Click on Next to **choose how long it will be out of stock** – hours, until the end of the day, or indefinitely. The item will automatically be made available after the selected time. You are also able to mark the item back in stock manually on your Tablet or POS.
- 4 Handle the **active order** in question by tapping Next to **chat with Support**. They'll get in touch with the customer to confirm if they want to replace the out-of-stock item or remove the item.

*If you use a POS protocol, we recommend marking items out of stock using your POS. When you make these changes via the app, they may be overwritten during a POS sync.

Active order

Select item or modifier

Chat for live order help

15

HOW TO POS Integrations

Cancel a live order*

- 1 If you're not able to fill an active order, you can choose to **cancel** it from the order details.
- 2 **Provide more detail** on why you can't fill the order so we can help you with next steps.
- 3 If you choose to cancel the order, we'll **confirm** this with you and let the customer know. As a reminder, cancellations that you initiate will **not be paid** for, even if you already prepared the food.

*This feature is not available on POS order protocol. If you're using a POS system and [need to cancel a live order](#), contact DoorDash Support.

Cancel a live order

Tell us what's going on

Confirmation

16

Checkout Process Architecture

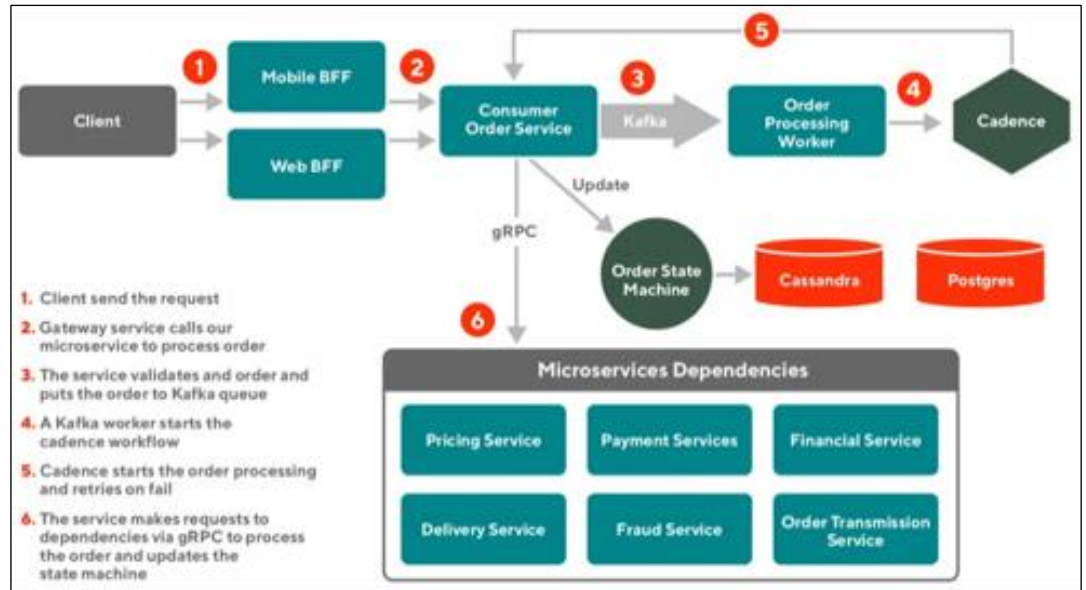
In the engineering blog article “Building a More Reliable Checkout Service at Scale with Kotlin” (see Exh. 78) the DoorDash development team explains how they reimplemented the customer checkout process into a new Kotlin microservices architecture to increase the DoorDash platforms performance, reliability, and scalability. As they state:

“...[t]he consumer checkout flow is one of the most critical flows on the DoorDash food ordering platform. The flow is responsible for submitting consumers’ order carts, processing payments, and connecting the order to our logistics system.”.

They describe this new architecture with this detailed diagram stating that:

“...[our] re-engineered checkout flow takes advantage of our new microservice architecture’s components. Cassandra, a distributed database, has replaced Postgres, and become our primary data storage. Kafka is used to decouple the

complex workflow. Cadence helps us build fault-oblivious business logic” (see Exh. 78).



This diagram depicts the master database through the Order State Machine which gets its data from the Casandra database. The order lifecycle flow depicted in the above diagram is as follows according to the article:

1. *“The clients (web/mobile) send requests to our gateway microservice to checkout an order.*
2. *The gateway microservice calls the newly built order service.*
3. *The order service fetches data associated with the order and validates the item, merchant, and delivery availability. Then it materializes the order into a Cassandra database and drops off a Kafka message in the queue, indicating the order has been submitted.*
4. *The Kafka consumers poll order submission events and create a Cadence workflow to initialize the order processing.*
5. *Cadence calls an order service endpoint to start the order processing.*
6. *The consumer order service starts its state machine to process the order. In each state machine step, such as performing a payment or creating a delivery within our logistics systems, the consumer order service interacts with a related microservice using gRPC and updates the order we stored in the Cassandra database.*
7. *The client checks the order processing state using a pull model. The client polls an endpoint on the order service, which looks up the order state in the Cassandra table and returns a success if the order processing has finished all of the state machine’s steps. If the order processing fails, Cadence retries it.”*

Dasher Dispatch Engine with Intelligent Machine Learning Technology

Pipelines for order completion logic is confirmed and admitted in these screenshots take

from the video presentation “DoorDash Technical Showcase Event: Logistics Team” (See Exh. #49) and as stated in this video as below.

Some of our Pipelines

Dasher wait time

Kitchen capacity

Other metrics we track

Influx of orders

Low Ideal High

Kitchen capacity varies over time

- Order cancellations
- POS outages
- Store closures
- Merchant tablet activity

55

Details on One of our Pipelines: Dasher Wait Time

Store Geofence

Dasher At Store Food prep time Dasher Leaving Store

Kafka Flink Pipeline

Use ml models and heuristics to decide if store can handle more orders

Calculate dasher wait time and aggregate median

We use median in place of average to discard 1 bad delivery

55

Multi-Modes of Communication

DoorDash affirms in the merchant support document “Which order protocol works best for your business” (See Exh. 112) that DoorDash merchants/partners can receive orders through multiple modes of communication including:

- “Your existing point-of-sale (POS) system via an integration
- A DoorDash tablet or your own Android tablet using the Order Manager app
- Email”

This document presents a table that specifies which mobile, tablet or web app is best to use for a given features. Below are screenshots of this specification (See Exh. 112):

| | Direct POS Integration (POS protocol) | Middleware Integration (POS protocol) | Tablet | Email |
|-----------------------------|---|---|---|--|
| Summary | <p>Popular with high-medium volume stores (~300+ orders per month) Great operational efficiency Most preferred</p> | | <p>Popular with medium-low volume stores (60-300 orders per month) Good operational efficiency</p> | <p>Suited for low volume stores (less than 60 orders per month) Lowest operational efficiency Least preferred</p> |
| Best For | <p>Businesses of all sizes, from quick-service to full-service, with an existing POS system.</p> | <p>Businesses of all sizes, from quick-service to full-service, with an existing POS system that does not support integration. Also for businesses using multiple third-party delivery providers.</p> | <p>Businesses of all sizes without a POS system or where POS integration is not available.</p> | <p>Small businesses that do not have a POS system and/or efficient internet connectivity.</p> |
| It Provides | <p>An easy, seamless way to manage delivery and pickup orders using your POS.</p> | <p>An easy, seamless way to manage delivery and pickup orders using your POS.</p> | <p>Quick order management, access to Dasher status monitoring, item availability updates, and real-time prep time adjustment.</p> | <p>Manual work to receive, confirm, and update orders.</p> |
| Country Availability | <p>Varies by provider</p> | <p>Varies by provider</p> | <p>All markets</p> | <p>All markets</p> |

| Functionalities | | | | |
|--|---|---|---|---|
| Receive Delivery and Pickup Orders | ✓ | ✓ | ✓ | ✓ |
| Receive Scheduled Orders | ✓ | ✓ | ✓ | ✓ |
| Managed Menu Changes | ✓ | ✓ | Merchant Portal | Merchant Portal |
| Menu Item modifiers | Varies by provider | Varies by provider | ✓ | Merchant Portal |
| Manage Menu Photos | Varies by provider | Varies by provider | ✓ | Merchant Portal |
| Manage Prep Times | Varies by provider | Varies by provider | ✓ | × |
| Track DoorDash Business Health | ✓ | Varies by provider; available on Business Manager app or Merchant Portal | Business Manager app or Merchant Portal | Business Manager app or Merchant Portal |
| Pricing Merchant charges/fees paid to DoorDash | FREE Some providers charge a fee for 3rd party integrations. Reach out to your provider for details | FREE Some providers charge a fee for 3rd party integrations. Reach out to your provider for details | \$6/ WEEK | FREE |
| Extra Hardware | None | None | DoorDash or Personal Android Tablet | Computer or tablet |

| Menu Management Features | | | | |
|--|--------------------|--------------------|---|---|
| Item 86'ing (Item OOS in real-time) | Varies by provider | Varies by provider | ✓ | Business Manager app or Merchant Portal |
| Manage Store Hours | ✓ | ✓ | ✓ | Business Manager app or Merchant Portal |
| Manage Store Holiday/Special Hours | ✓ | ✓ | ✓ | Business Manager app or Merchant Portal |
| Update Menu Hours | Varies by provider | Varies by provider | ✓ | Business Manager app or Merchant Portal |
| Separate and Day-part Menus | ✓ | ✓ | ✓ | Merchant Portal |
| Special Instructions* | ✓ | ✓ | ✓ | Merchant Portal |
| Menu Item Sort Order | ✓ | ✓ | ✓ | Merchant Portal |
| Menu Item Level Hours** | Varies by provider | Varies by provider | ✓ | Merchant Portal |
| Price Override/Inflation | Varies by provider | Varies by provider | ✓ | Merchant Portal |

| Live Operations Features | | | | |
|--|--|--|---|--|
| Busy Kitchen | Contact DoorDash Support | Contact DoorDash Support | ✓ | Contact DoorDash Support |
| Temporarily Pause Store | Business Manager app or Merchant Portal | Business Manager app or Merchant Portal | ✓ | Business Manager app or Merchant Portal |
| Contact Customer | Business Manager app or contact DoorDash Support | Business Manager app or Contact DoorDash Support | ✓ | Business Manager app or contact DoorDash Support |
| Cancel/Refund Customer | Business Manager app or contact DoorDash Support | Business Manager app or Contact DoorDash Support | ✓ | Business Manager app or contact DoorDash Support |
| Dasher Updates | Business Manager app or contact DoorDash Support | Business Manager app or Contact DoorDash Support | ✓ | Business Manager app or contact DoorDash Support |
| Contact DoorDash Merchant Support | Business Manager app, phone, or email | Business Manager app, Merchant Portal, phone, or email | ✓ | Business Manager app or contact DoorDash Support |

Furthermore, in the merchant support document “How do I receive orders with DoorDash?” DoorDash affirms (See Exh. 113):

“If you would prefer orders to come through a tablet, all orders will come through DoorDash's Order Manager App on Android tablets.

- *With the tablet, you can see:*
- *Sections for each order stage lifecycle to easily identify where an order is*
- *Scheduled order timing when it appears in-app is changing (used to show up/print immediately)*
- *Cancellations won't just disappear anymore*
- *Tags for scheduled orders, large orders, canceled orders, modified orders,*

customer pickup

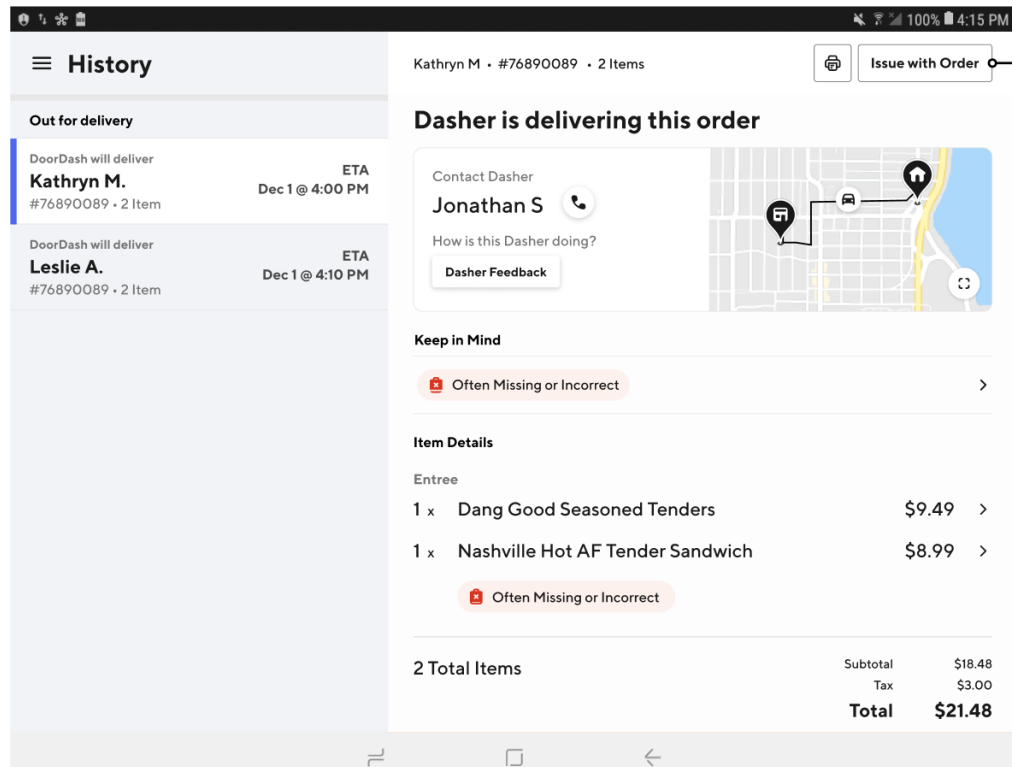
- *Item count per order*
- *Large/clearer map and Dasher location information*

You can receive orders by POS, email, or email/fax. However, you will not be able to receive orders from phone+tablet or email+fax.

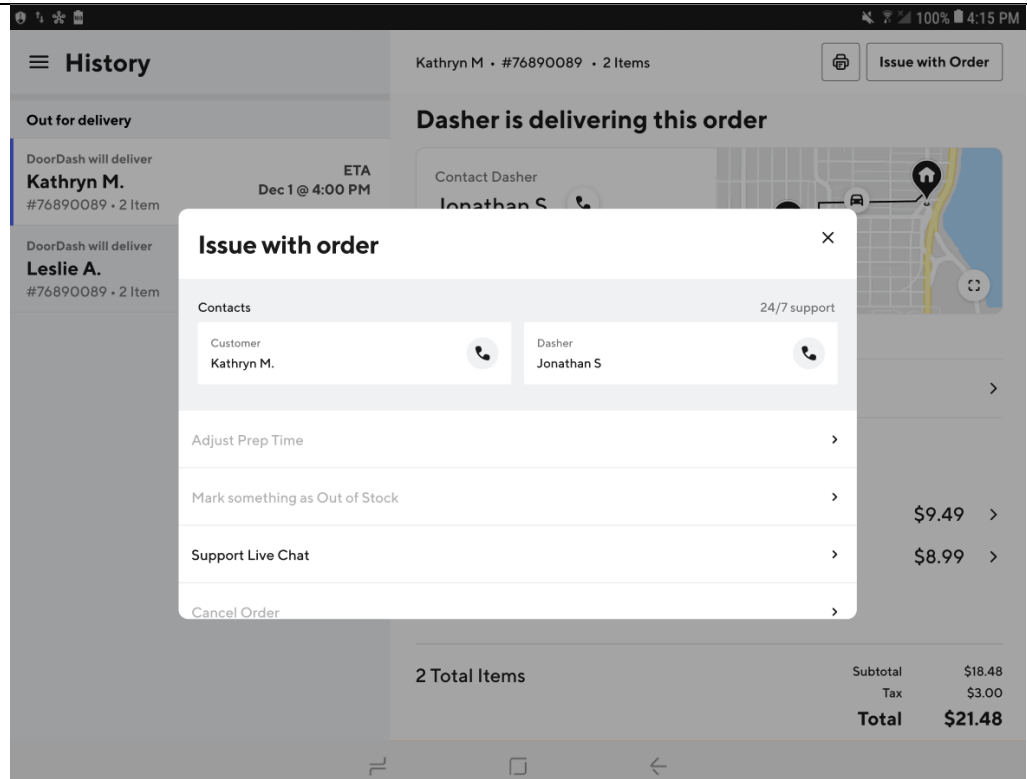
DoorDash can also place a follow-up automated call to ensure that all orders were received.”

The DoorDash Support article “How can I contact the Customer or Dasher through the tablet” (See Exh. 114) instructs the DoorDash merchant how to communicate with a Dasher (delivery driver) or Customer for an order that’s in process or has been picked up, should any questions arise. According to the article, the merchant uses the tablet and:

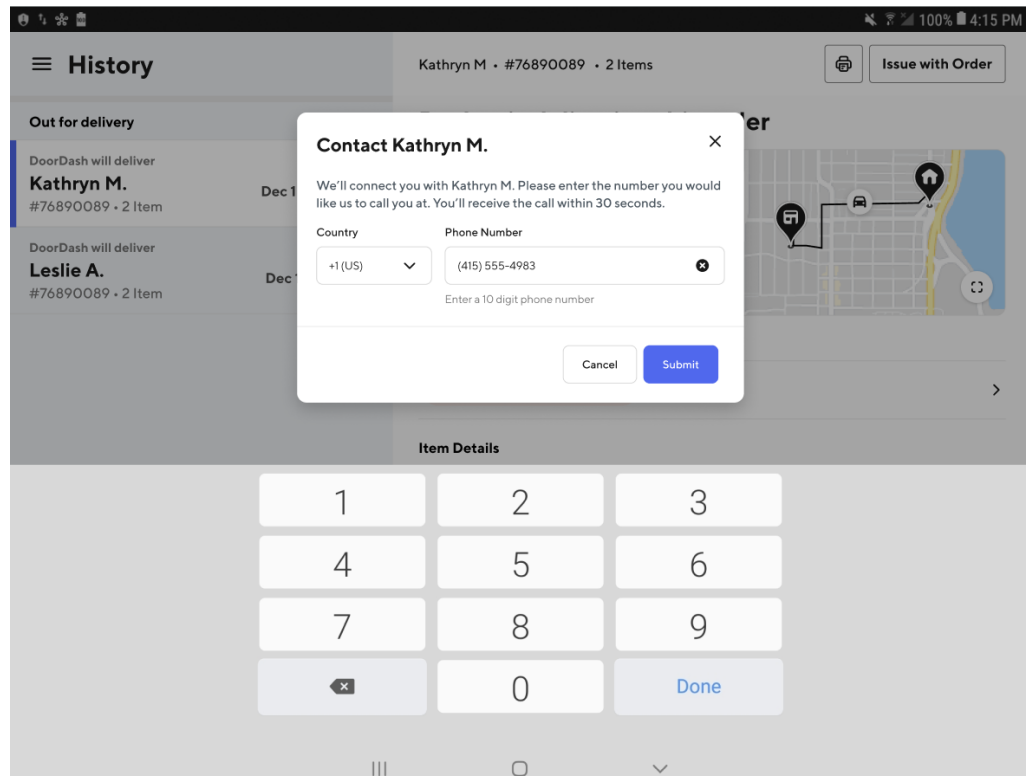
“...On the Order History page, tap on the Issue with Order button in the top right corner.”



“Next, tap on either the Customer or Dasher that you would like to contact. “



“Next, enter the phone number you would like to use to be connected to either the customer or Dasher. Tap Submit and we will call you on the number you enter within 30 seconds to connect your call!”



The DoorDash Merchant Support article *“How can I tell if my Dasher is arriving or*

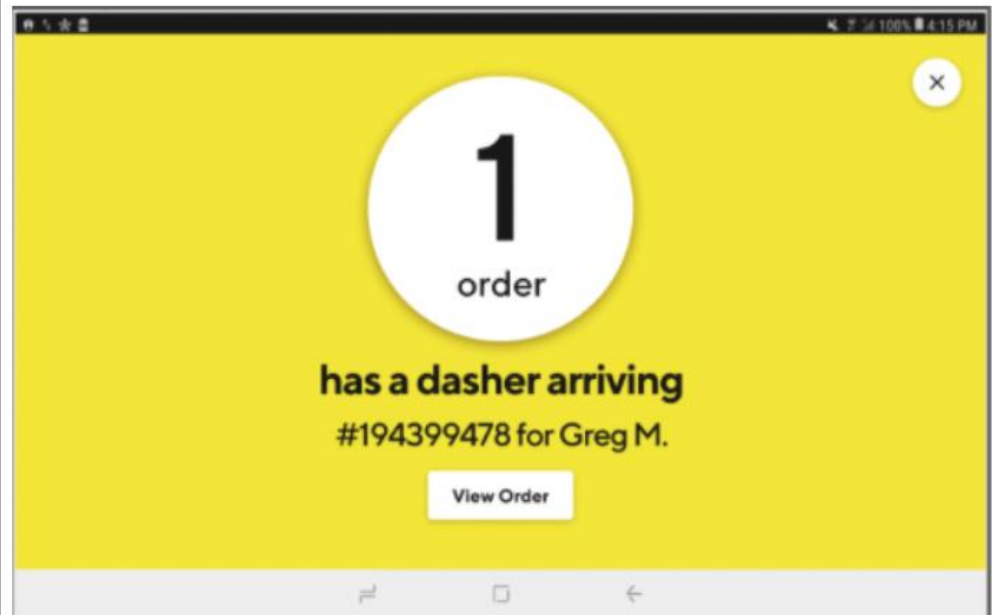
waiting” (See Exh. 115), discusses how merchants receive real-time notifications on their DoorDash Tablet:

“...when a Dasher is nearby to pickup an order! The new Dasher Arriving feature will send you a secondary chime and visual cue that will alert when a Dasher is 5 minutes away. With this feature, Merchants will now know exactly when to start preparing and bagging those final items (cold drinks, ice cream, fries, etc) right before a Dasher's arrival, preserving food quality and also reducing overall wait times!”

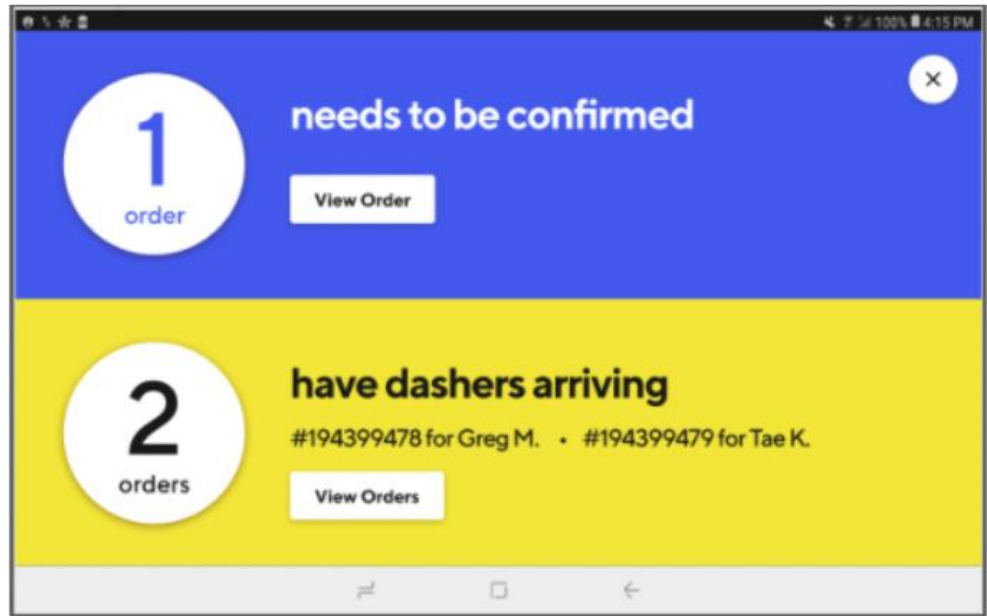
Notification

Once a Dasher is 5 minutes away, a full screen take-over will appear in yellow notifying staff on an approaching Dasher

Note: A unique chime will sound, which will help staff differentiate between a normal incoming order versus an incoming Dasher.

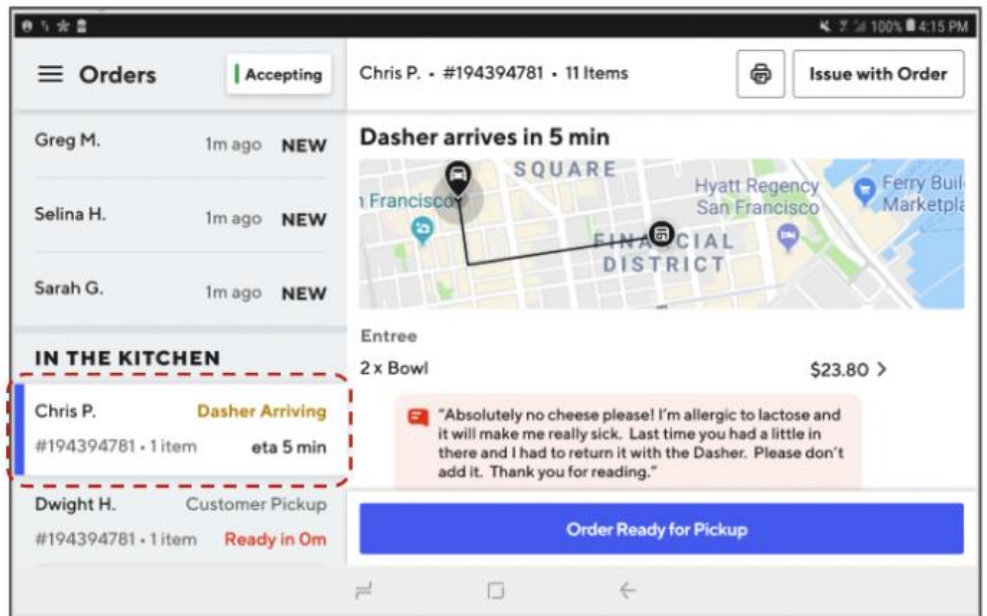


If there are new orders and Dashers arriving at the same time, the screen will appear as below, and the chime for "new order" will alarm. You will also be able to see which orders have Dashers coming for them (Customer name(s) shown).



Dasher Arriving Label

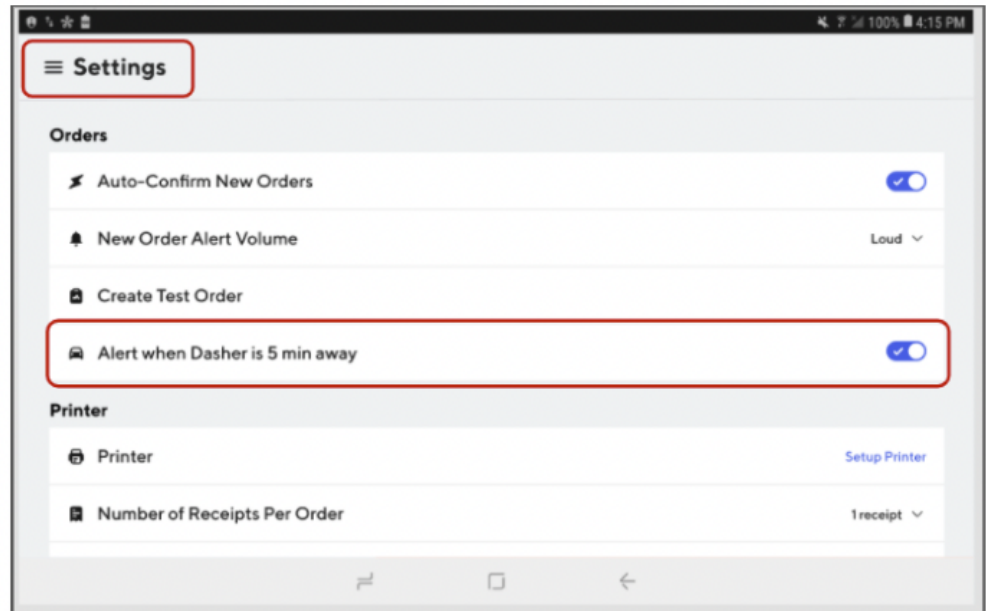
Within each order, there will be a Dasher Arriving label with eta x min shown, counting down until the Dasher arrives.



Feature Settings

Enter the flow directly in the tablet by selecting Settings from the main navigation page

Settings → Alert when Dasher is 5 min away → Toggle On / Off



In the DoorDash merchant support article “Can I have a tablet and also still get orders through email/fax/phone calls” (See Exh. 118), DoorDash states the following:

“[The Merchant] ...can receive DoorDash orders on a tablet and get orders through email or fax at the same time. As long as your tablet is logged in, you will continue to receive orders.”

DoorDash’s Unified Chat Experience

The DoorDash engineering team stated in an engineering blog post, titled **Building a Unified Chat Experience at DoorDash**, how they successfully integrated chat into their overall system, addressing the challenges and strategies employed to create a unified chat experience. Their implementation aligns with MFCCS as it provides a centralized system layer architecture enhancing communication between the web server and multiple wireless handheld devices, each operating on distinct mobile systems thus allowing customers to communicate with customer support and dashers. (See Exh. 102)

DoorDash recognized the crucial need to address customer issues on a large scale, leading them to acknowledge the essential nature of a unified chat feature. Their initial challenge was a fragmented chat system, which was scattered across various user groups and platforms, causing duplicated efforts and central adoption of best practices to be challenging.

By leveraging the architecture MFCCS, DoorDash could centralize its chat functionalities, allowing for seamless interactions between consumers, Dashers, and merchants. The MFCCS would facilitate the web server's ability to communicate with diverse handheld devices, presenting mobile-compatible versions of the hospitality

application. This integration is crucial for enabling user-initiated actions and subsequent selection of choices directly from the touchscreens of various devices, fostering a multi-modal, multi-protocol communication environment.

According to the engineering article, a significant objective for DoorDash was to streamline their support processes through automation, allowing customers to retrieve information through the app itself, thus minimizing the need for human support interaction. They aimed to develop a single chat support platform that would ensure consistent user experiences across different platforms and facilitate the reuse of components.

In building this unified platform, DoorDash created a single chat implementation that functioned for both customer support and communication between consumers and Dashers. This system, powered by Sendbird, was directly integrated into the consumer and Dasher apps, providing a uniform experience across all DoorDash chats and allowing for shared enhancements across different chat flows.

The backend system was strategically designed with multiple layers, distinguishing between internal services and third-party functionalities such as chat natural language processing. This separation expedited development by eliminating the need to create distinct, non-essential features.

Integration with their Decision Engine platform was another crucial step, which enabled operations to be automated, such as order updates, issuing credits, and personalized messaging. This significantly decreased the dependency on manual operations. DoorDash also carefully monitored several key performance indicators, including feedback from agents and customers, error rates, customer satisfaction, average handle time, first contact resolution, and manual tasks per order.

The outcomes of this updated chat support system, according to the article, were noteworthy, with reduced escalation rates and improved customer satisfaction. The automation led to a decrease in manual interactions per delivery, and customers were able to solve their problems more quickly, sometimes without the need for manual assistance. The time needed to implement additional automation was also reduced, owing to the common platform processing. Moreover, the common UI layers ensured that the chat feature was seamlessly integrated into the applications (iOS and Android), providing a consistent brand experience across all platforms.

The advancements DoorDash has made in their chat system encapsulate the essence of the Middleware/Framework Communications Control Software (MFCCS) described in the patent claim. By creating a unified chat platform that seamlessly integrates with multiple user interfaces and operating systems, DoorDash has effectively embodied the centralized system layer architecture that MFCCS envisions. Their backend, which differentiates between core and third-party functionalities, mirrors the multi-layered approach of MFCCS, allowing for efficient, scalable communication across various devices. Moreover, the phased rollout and meticulous performance monitoring align with the adaptive, intelligent operational capabilities that MFCCS aims to provide. This demonstrates a real-world application of the patent's claim, showcasing a system that not only supports multi-mode communication but also enhances user experience and operational efficiency in a dynamic hospitality market.

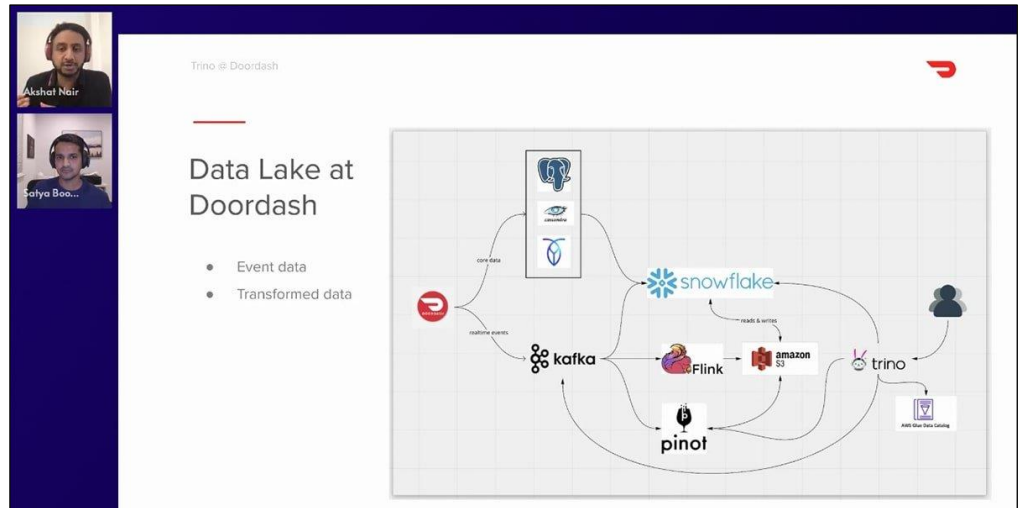
The accused instrumentality meets this limitation under the doctrine of equivalents because the accused instrumentality performs substantially the same function, in

substantially the same way to achieve substantially the same result. For example, the Middleware/Framework Communications Control Software (MFCCS) integration into the DoorDash System empowers the network to process hospitality application task requests from wireless handheld users, utilizing a spectrum of communication modes. Notably, the DoorDash system seamlessly transitions to alternative hospitality entities upon detecting inventory shortages, demonstrating operational efficiency guided by intelligent rules that prevent redundant attempts with unresponsive entities. This approach optimizes resource allocation and minimizes processing time by circumventing fruitless communication endeavors with hospitality entities known to be unresponsive, effectively embodying the core elements of the claim in the DoorDash system.

a master database comprising multiple linked and continuously synchronized in realtime databases throughout the network and with data and parameters of the one or more hospitality software applications integrated with the said network and with predefined formats, the master database comprising a usable file structure dictated prior to execution, thus improving efficiency and reliability, wherein the one or more hospitality software applications learns, updates, stores and intelligently applies varying modes of contact with the handheld/mobile equipped hospitality users and in accordance with their preferences, if any;

DoorDash designs, tests, deploys, integrates and maintains the claimed ‘master database’ (it’s ‘central data platform’ and/or its ‘central data lake’ and/or its ‘core data platform’) as is detailed below, including a recently confirmed addition on August 17, 2022 by the Jessica Lachs, DoorDash Vice President of Analytics and Data Science, and in a recent job posting for a Director of DoorDash’s Core Data Platform.

DoorDash’s “master database” has its own API and operates “intelligently” via “learning” (DoorDash’s platform features extensive “machine learning” technology), while integrating with the claimed “web servers” and storing and utilizing “data and parameters” of the said hospitality applications with a usable menu structure (and with imported merchant data via the external API as shown further below) and using at least one predetermined format that is dictated prior to task execution and which enables the intelligent application of multiple modes of contact and related operational parameters for hospitality entities and remote hospitality users (consumers/dashers).



DoorDash’s Vice President of Analytics & Data Science, Jessica Lachs, in a August 17, 2022 video interview titled “Leveraging Data to Delight Customers Despite a Challenging Supply Chain”, stated the following (See Exh. 98):

“And so for us, it’s really about collecting as much information as we can about all sides of the marketplace, bringing all of that data together into a central data platform, where all of that data is accessible no matter the source. Whether it is coming from our production systems, transactional data, whether it is event data in our apps, whether that’s the consumer app, the dasher app, the merchant app... whether it is coming from our CRM systems. All of that data needs to come in to one central place so that we can tie it together and use the insights together to create a 360 degree picture of what’s happening on our platform and off our platform so that we can use that information not just to provide accurate menus and inventory for consumers but also so we can send the right email communications to consumers, to dashers, so that we really have a full picture of what’s happening and can use that for personalization and to help all three sides of our marketplace really optimize that they are at their peak efficiency.”

“So, for us, we want data to be easily accessible to all the different teams that need access to it. Analytics, being one of the largest customers of data at DoorDash, of course, but the way we think about our data models is really about increasing accessibility and consistency to that data. So, having all of our data in one central place and making sure that it is high in performance and so like query speeds are fast and that data models are thoughtful, so that it makes it a lot easier for data scientists, analysts, operators, product managers to be able to query the data that is needed and use the data in our production, in our production systems as well. So, we try to be thoughtful about how we structure our data models and how we ensure that all of the different production systems tie together into that central, as you mentioned, that central data lake.”

Director of Engineering, Core Data Platform

In a recent job posting for a Director of Engineering for DoorDash’s Core Data Platform in San Francisco, CA (See Exh. 43), it describes the purpose of the Core Data Platform as follows:

“...DoorDash is a data driven organization and relies on timely, accurate and reliable data to drive many business and product decisions. The Core Data Platform owns all the infrastructure necessary to run an operationally efficient analytical data stack. This will include data ingestion (batch and real time), data compute & transformation, data storage (warehouse, data lake, OLAP etc.), querying infrastructure as well as data compliance, quality and governance.”

Further See Exhibit #45:

In the customer case study published by Snowflake (See Exh. 45) they state that DoorDash serves “...more than 4,000 cities” and “...ingests and analyzes large amounts of operational data and data from its web and mobile app.” They go on to state that “...DoorDash’s legacy data architecture could not keep pace with its data-driven culture. Consumer, merchant, and Dasher data were spread across data silos, creating challenges for operational reporting and business decision-making. Resource contention caused by 70,000 dashboard queries and 3,000 ad hoc queries per day led to missed SLAs and stale data.” Therefore, Snowflake, provided the ideal solution to DoorDash that being “...[a] platform for scalable delivery insights”. This solution according to Snowflake (See Exh. 45):

“Realizing the need for a modern data infrastructure, DoorDash turned to Snowflake’s platform. Snowflake’s multi-cluster shared data architecture scaled to handle all of DoorDash’s data, users, and workloads with speeds twice as fast as before. Snowflake’s network of connectors, drivers, and programming languages accelerated the migration of 600 ETL jobs.

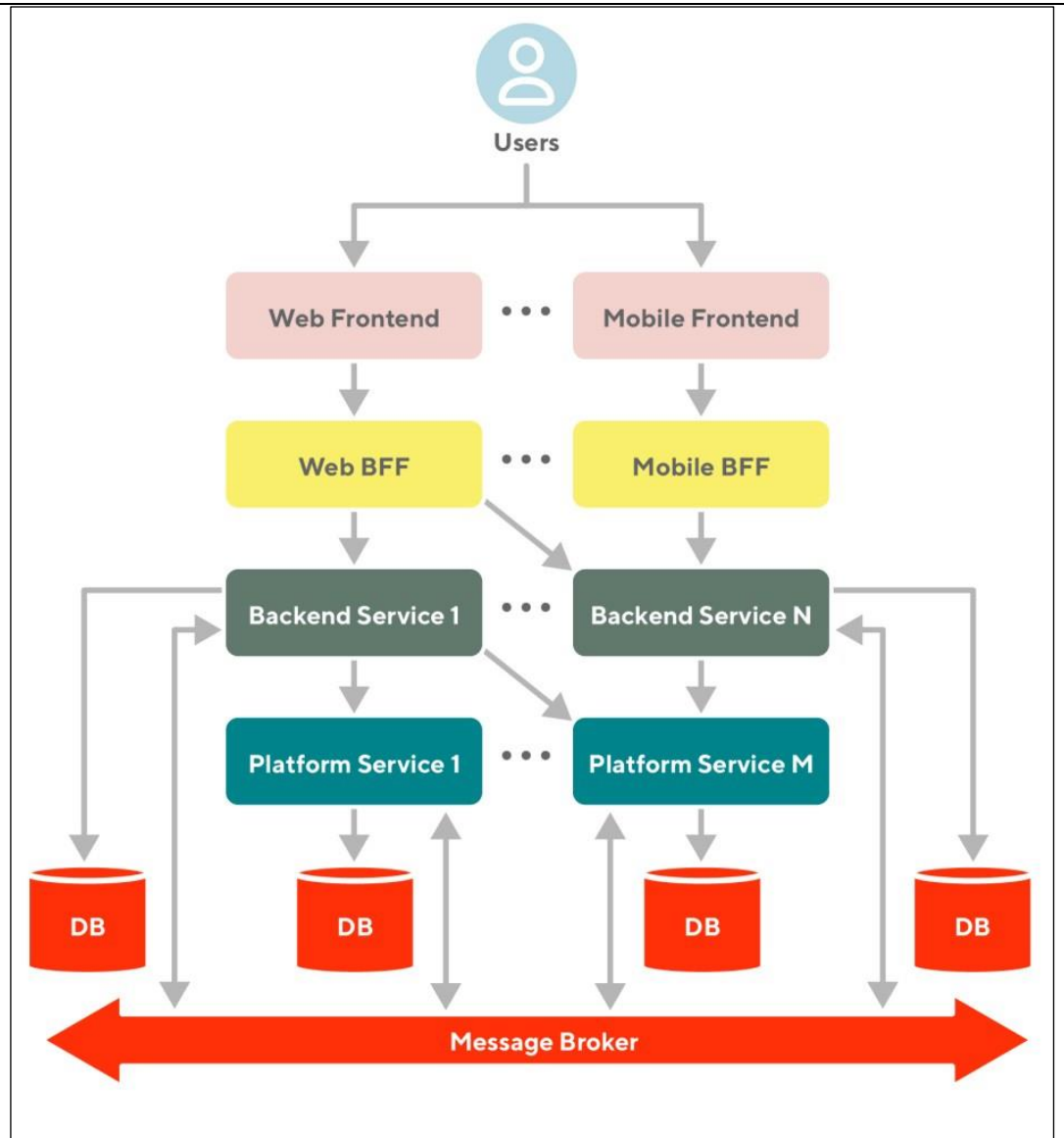
ETL jobs finished 23% faster, enabling the BI team to meet its reporting SLAs 99.7% of the time. Snowflake’s fully managed infrastructure with near-infinite scalability kept the team focused on data analytics and modeling. “Snowflake’s elasticity keeps us on track no matter what life brings,” DoorDash’s Director of Business Intelligence Marta Vovchenko said.

Ingesting all of DoorDash’s consumer, merchant, and Dasher data into Snowflake provides market managers across the globe with the latest supply and demand insights by 7 a.m. daily.

Architecting DoorDash’s merchant portal on Snowflake provides merchants with data-driven reports for managing orders, inventory, and staffing. “Offering our data as a product for end users is a big differentiator,” Vovchenko said. Snowflake’s instant scalability enables the BI team to quickly spin up separate data warehouses and develop data models that measure new business-line performance. DoorDash’s marketing attribution model powered by Snowflake analyzes advertising data from Facebook, Google, and other platforms to optimize campaigns and budgets. Machine learning algorithms use data stored in Snowflake to make personalized product recommendations to customers at scale.”

DoorDash’s intricate architecture includes the ‘master database’. DoorDash was able to improve the efficiency and reliability of their data platform using a microservices architecture. By breaking up their application into domain-specific parts they were able to reduce errors and latency. They state in the article “Future-proofing: How DoorDash Transitioned from a Code Monolith to a Microservice Architecture” (see Exh. 20) that

“... [the] final design for our new microservice architecture consisted of five different layers, ranging from the user experience to the core infrastructure. Each layer provides functionality to the upper layer and leverages the functionality exposed by the lower layer [as shown below]”:



As the above diagram depicts, the four DB (or database) icons represent linked database all managed through the Message Broker, which is their database API. Also in this diagram, the Backend microservice and Platform microservice connect to and interact with linked database. As describe in “How DoorDash transitioned from a code monolith to microservices” (See Exh. 20) by the DoorDash engineering team, their microservices architecture depicted in this diagram provided the following:

“Frontend layer: Provides frontend systems (like the DoorDash mobile app, Dasher web app, etc.) for the interaction with consumers, merchants, and Dashers that are built on top of different frontend platforms.

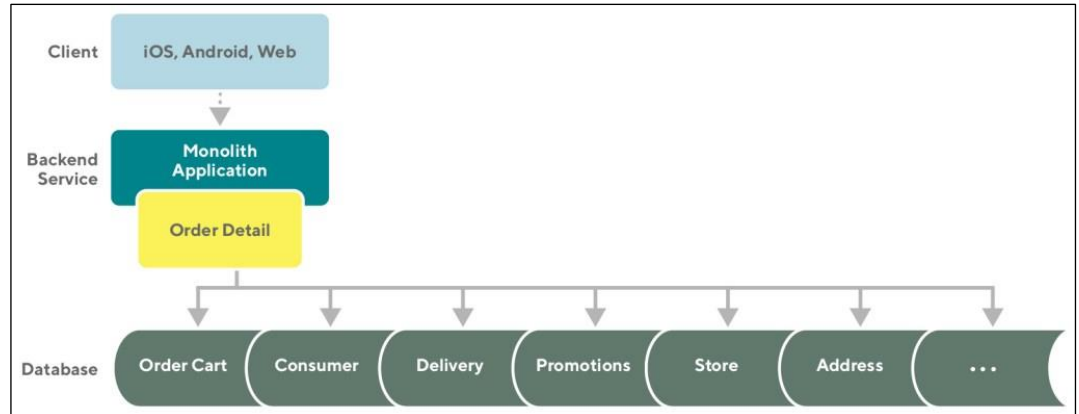
BFF layer: The frontend layer is decoupled from the backend layer via BFFs. The BFF layer provides functionality to the frontend by orchestrating the interaction with multiple backend services while hiding the underlying backend architecture.

Backend Layer: Provides the core functionality that powers the business logic (order cart service, feed service, delivery service, etc.).

Platform layer: Provides common functionality that is leveraged by other backend services (identity service, communication service, etc.).

Infrastructure layer: Provides the infrastructural components that are required to build the site (databases, message brokers, etc.) and lays the foundation to abstract the system from the underlying environment (cloud service provider).”

As depicted by the diagram below, DoorDash states “... our microservices architecture, APIs orchestrated by a BFF are targeted towards specific services, making it easier to design precise calls and appropriate execution.”



“From database infrastructure to app programming, there are many ways technology companies can improve the customer experience. Good, targeted API design might be one that falls under the radar, yet can deliver significant improvements. In our case, we were given the opportunity to redesign our APIs during a migration from a monolithic codebase to a microservices architecture. However, any technology company, especially those with a great amount of legacy code in their platform, might find it useful to assess their APIs. There might be opportunities to reposition the APIs, removing overly large requests and reducing load on networks, while making clients run more quickly and becoming less error-prone, ultimately delivering a better customer experience.” See Exhibit 41.

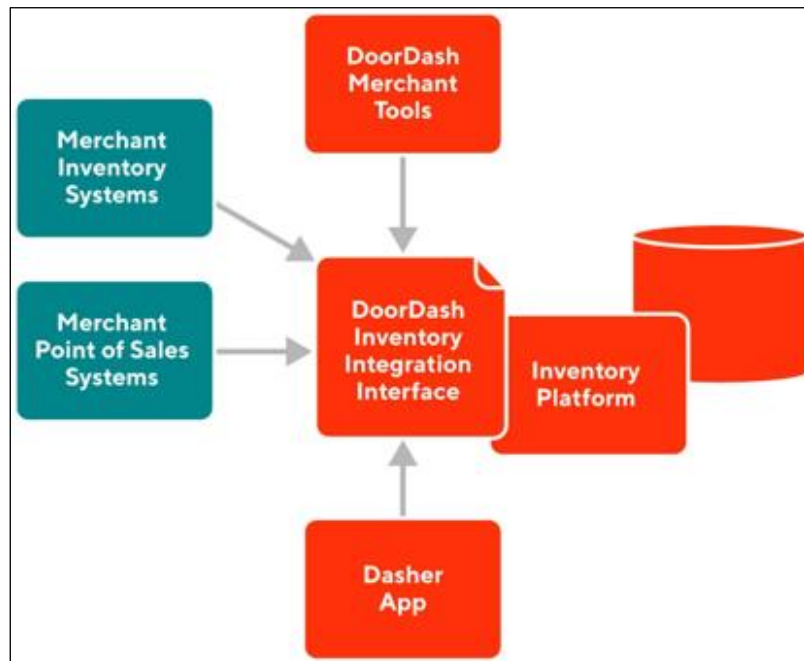
Data Warehousing

In a customer case study published by Starburst, which promotes itself as the fastest query engine for data warehouse, analytics, etc., it states that “DoorDash connects customers and businesses in more than 4,000 cities. Its network of merchants and local drivers, known as “Dashers,” enables on-demand delivery of takeout food, groceries, and household essentials. To provide insights to internal stakeholders and merchants, DoorDash ingests and analyzes large amounts of operational data and data from its web and mobile apps” (see Exh. 45). Also, the Director of Business Intelligence at DoorDash goes on to state “[m]igrating to Snowflake has been fundamental to the growth of BI and the expansion of data’s impact at DoorDash”. Lastly, in a case study published by Starburst (See Exh. 54), an engineering manager for DoorDash is quoted as saying “With Starburst, the future of data analytics looks bright. We hope to

achieve our vision of a unified query engine and a secure, single point of access to all of our data with Starburst Enterprise”.

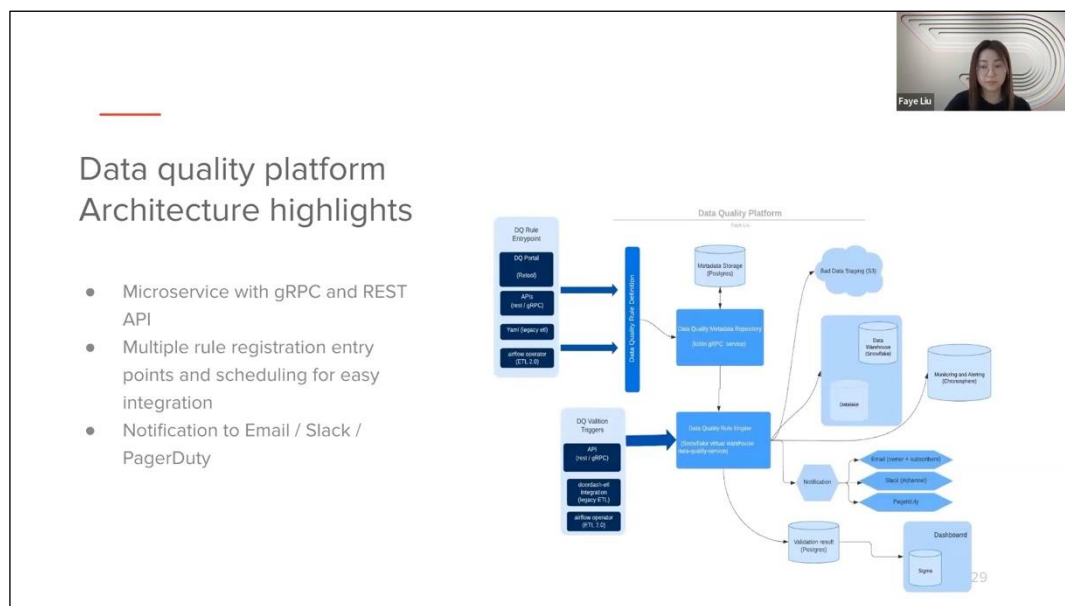
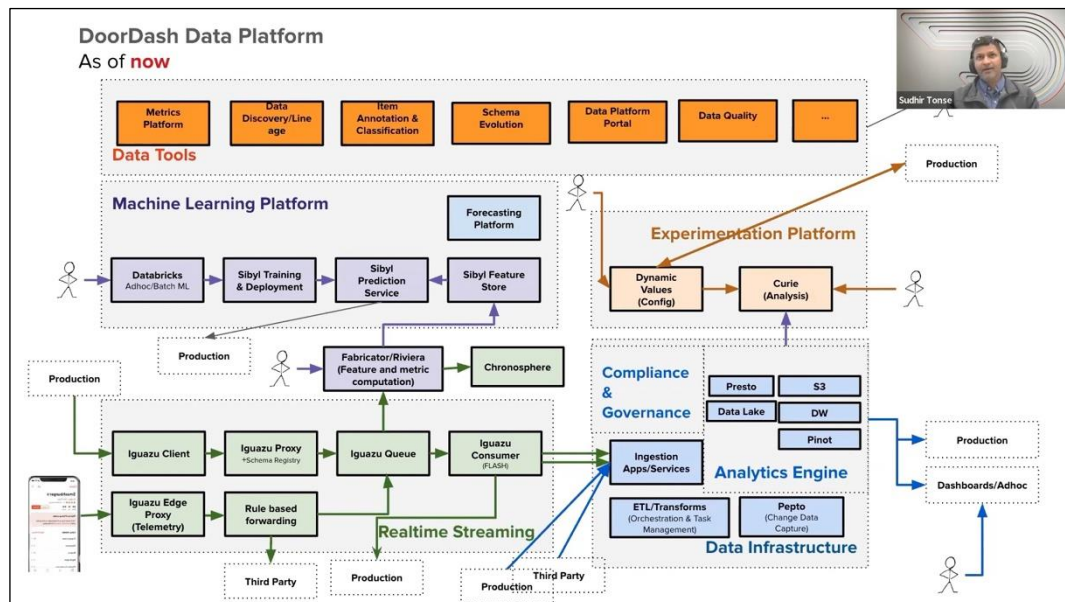
In the DoorDash engineering article “Building a Source of Truth for an Inventory with Disparate Data Sources” (See Exh. 55), they state that they have faced serious challenges managing data inventory and admit to the use of a master database. As such they state:

“...[they] must ensure that all merchant processes are recorded electronically, including receiving inventory, putting inventory on shelves, and removing items from shelves. But this is not feasible in light of the large operational and capital investments that would be required. Instead, we must optimize use of the data that we can collect and extrapolate inventory states from that... [as shown in figure below] ...[by] collecting partial or incomplete data from across multiple sources and then reconciling that input within a comprehensive inventory dataset, we unlock the ability to crowdsource inventory data for a physical store. Already, we are collecting hundreds of millions of inventory data points from these disparate sources.”



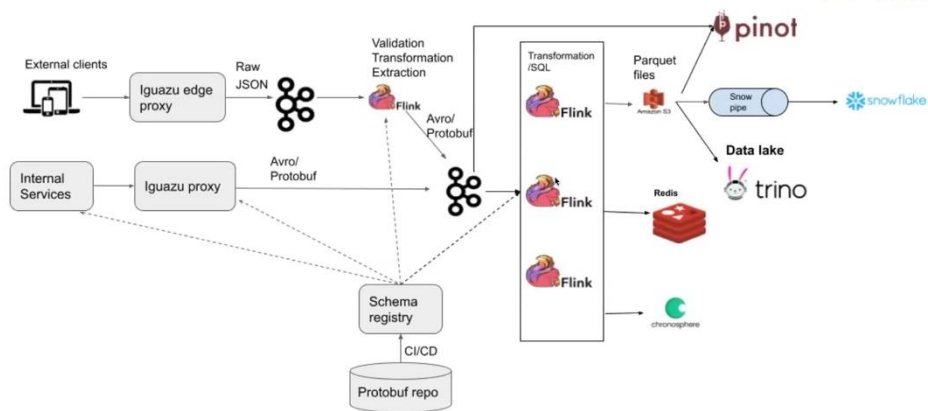
In the video DoorDash Technical meetup event: **Scaling our Data Platform** by the DoorDash engineering team (See Exh. 47), they clearly demonstrate how they use a of collecting master database in the DoorDash Data Platform. Below are several screenshots from this video highlighting the integration of a master database, sectioned into several microservices. For example, the two charts below, taken from this video (timestamp 00:06:43 and 00:21:42) illustrates the development that DoorDash has undertaken “over the last three years” on their Data Platform gathering, analyzing, retrieving, updating and storing massive amounts of data. They added services like real-time streaming and machine learning (i.e., artificial intelligence system processes), all integrated with the data warehousing master database located on Amazon’s S3 cloud servers.

The second chart “Data quality platform Architecture highlights” (See Exh. 56) illustrates DoorDash’s microservice for data quality using a RESTful API (note: RESTful API is an interface that two computer systems use to exchange information securely over the internet) shows the integration of several data stores for data warehousing, dashboard, monitoring and alerts, all of which culminate into a database operating as one master database.



The video goes on to describe how Iguaza works (timestamp 00:29:42) and is integrated into the data platform, DoorDash’s ideal workflow that includes a “single entry point” (timestamp 00:39:40) in which data is being “served” into their ‘master database’, and their development of Fabricator for a unified execution environment with an “online storage”.

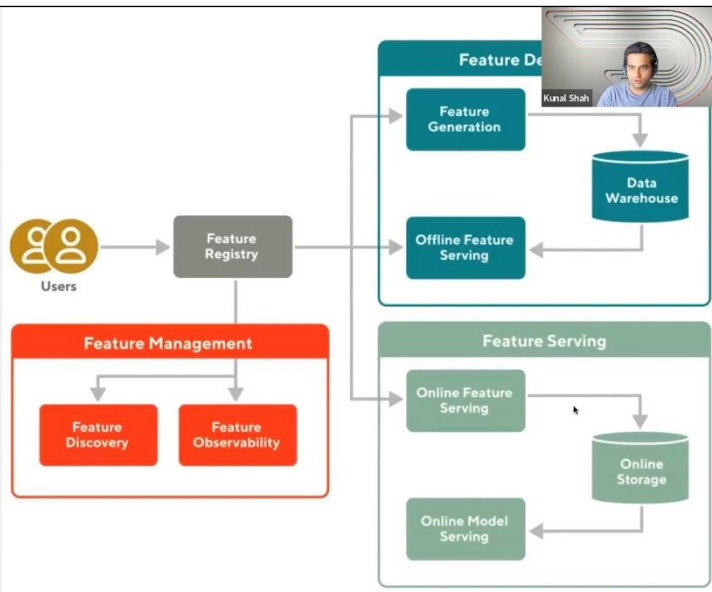
The Big Picture of Iguazu



40

What's a better ideal?

- Single entry point
- Automated and standard downstream interactions
- Seamless sharing with code and feature reuse



Fabricator

Fabricator delivers this premise using three core ideas :

- Central feature registry
- Unified execution environment
- Automated infra integrations

The diagram illustrates the Fabricator architecture. On the left, 'Users' (represented by two person icons) have two paths: 'Develop' leading to 'Feature Generation' and 'Productionize' leading to 'Feature Registry'. 'Feature Generation' is part of the 'Unified Execution Environment' (a blue box) which also includes 'Data Warehouse' and 'Online Feature Serving'. 'Feature Registry' is a central component that feeds into 'Feature Generation' and also connects to the 'Infrastructure Automation' (a red box). The 'Infrastructure Automation' box contains 'Online Feature Serving', 'Feature Observability', and 'Feature Discovery', all of which are connected to 'Online Storage' (a red cylinder icon). A small video inset of Kunal Shah is in the top right corner.

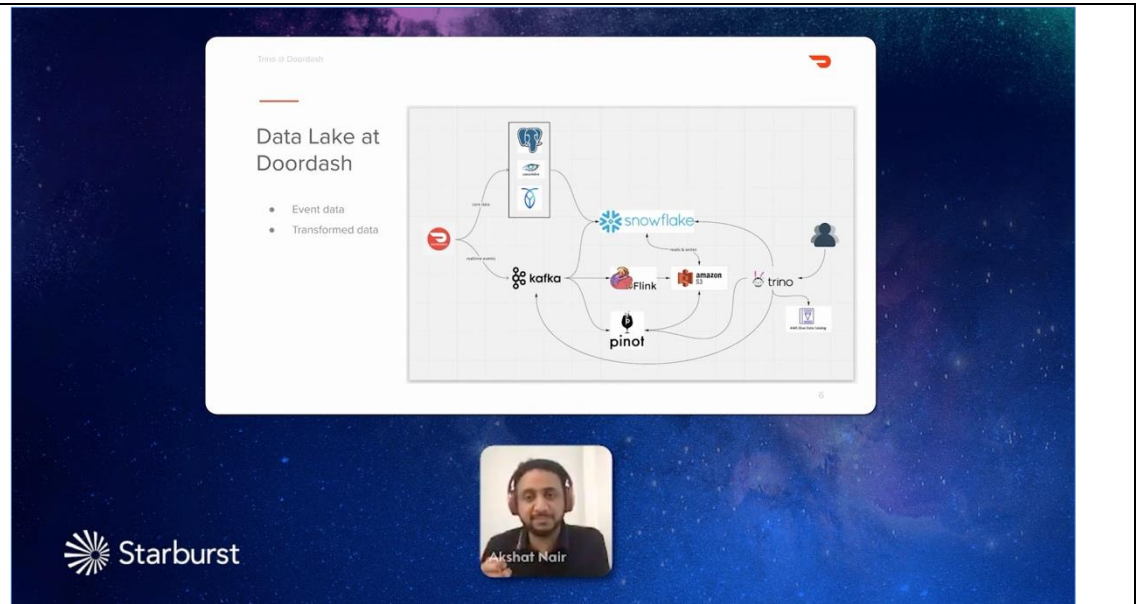
Working at DoorDash’s Scale

- **Analytics:** Data Warehouse active users ~2500 with XX PB+ data powering 5M+ queries/day
- **Realtime Events:** Iguazu handles around 100s of billions messages per day
- **Experimentation Platform:** ~250 experiments analyzed L28 via our Experimentation Analysis Platform Curie
- **Machine Learning Platform:** Sibyl Prediction Service - ~3 Million Predictions per sec and ~200+ Models

10

Trino Query Engine Integration

In the video “How DoorDash processes petabytes of data by utilizing Trino” the DoorDash engineering team (see Exh. 48) presents how data flows and queried in their current setup. They state that users can query data from “...snowflake, and data from S3 and data from Pinot using Trino”. Trino is described as a system that “...is a distributed SQL query engine designed to query large data sets distributed over one or more heterogeneous data sources” (See Exh. 57-58), in other words, gather data from multiple internal sources through one interface, a master database.



DoorDash's Intelligent Machine Learning

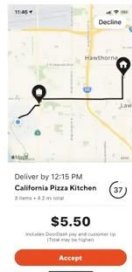
The DoorDash system intelligently learns, updates, and stores multiple communication modes of contact and related operational parameters for hospitality entities and for remote hospitality users along with their prior attributes or preferences, if any and then intelligently applies them, as the systems maintains an account for each user this account is accessible and updateable through the webapp, the iOS app and the Android App Any change to the account made by the user through one app, is automatically reflected in the other apps. In other words, a change the user makes through the iOS app of their account, will be reflected when they login texting/chatting between the dasher and user, and to their account through the web browser.

In the video “DoorDash Technical Showcase Event- Logistics team” (See Exh. 49), the engineering logistics team uses Machine Learning (known as ML) to power their Dispatch engine so that Dashers and orders are matched in real time, and that the deliveries are completed as “effective as possible” for the merchant, dashers, and customers. Screenshots from this video is as follows:



What We Do

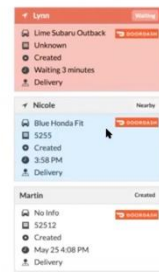
Fulfillment of Deliveries



Build a fantastic product for Dashers



Make optimal decisions so every delivery goes smoothly



Streamline merchant operations

The lifecycle of an order starts at the merchant (i.e., restaurant), to the dasher (delivery driver), and onto the customer.



Delivery lifecycle



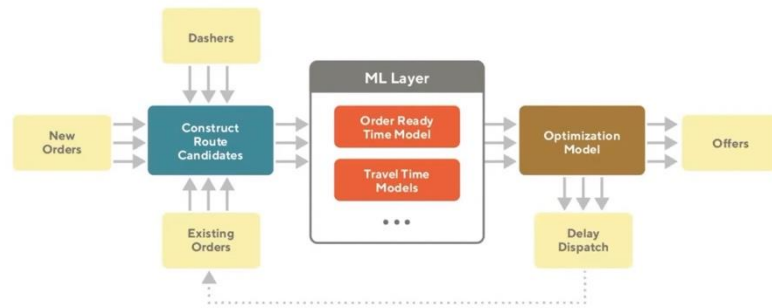
The dispatch problem: Decide which offer to send to which Dasher in order to:

- Maximize Dasher efficiency and earnings opportunities
- Deliver orders fast & on time for best possible consumer and merchant experience

An order goes through an extensive journey. As an order flows through the Dispatch, it goes through several stages making intelligent decisions based on the data available.



An Order's Journey Through Dispatch



11

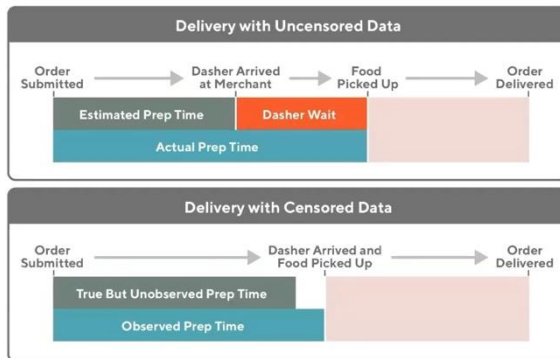
The Dispatcher system figures out which dasher to send the order to, how long the merchant will take to prepare the order, when to send the dasher to pick up the order, how long the travel time will be, and when the customer can anticipate the delivery time. The Dispatcher system makes these intelligent decisions in real time for each order.



ML layer: Predicting How an Order Will Play Out

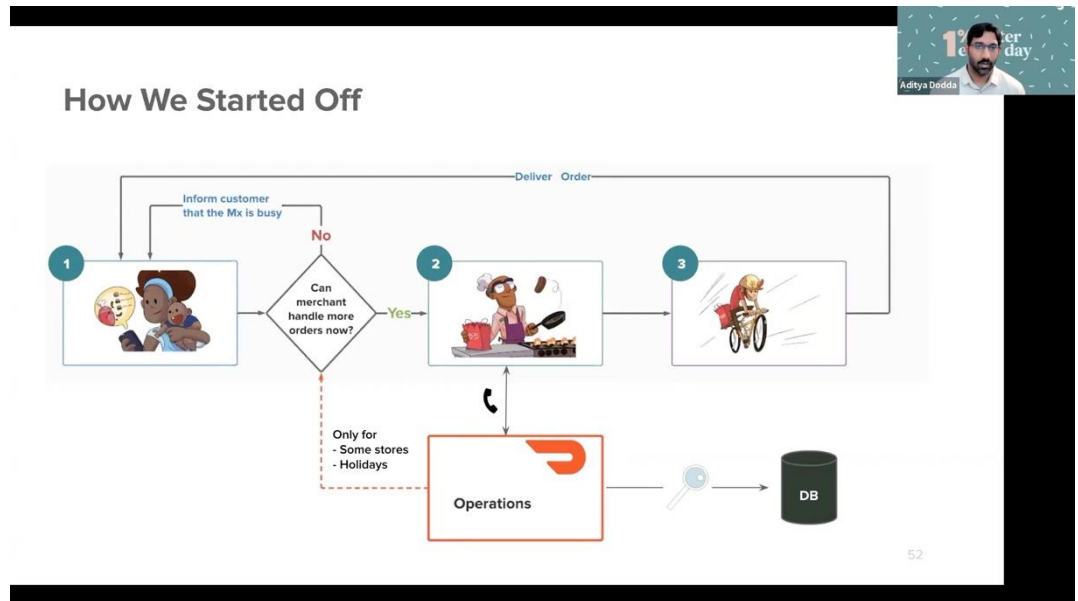
1. Order ready time
2. Travel times
3. Likelihood Dasher will accept the order

Example: How we deal with left-censored data in estimating prep times

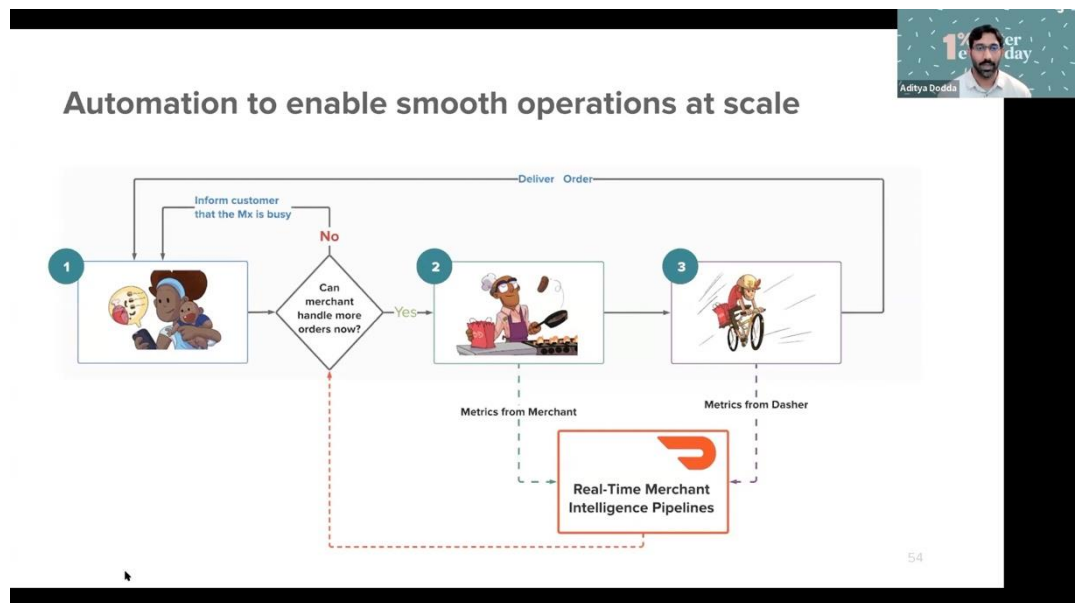


14

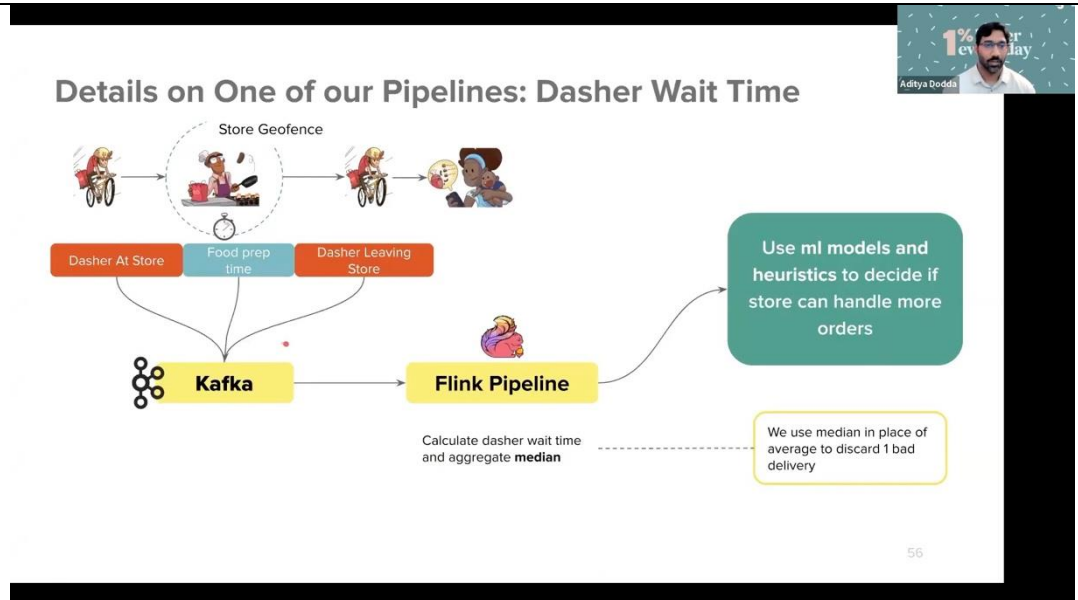
Here is an illustration of an orders flow:



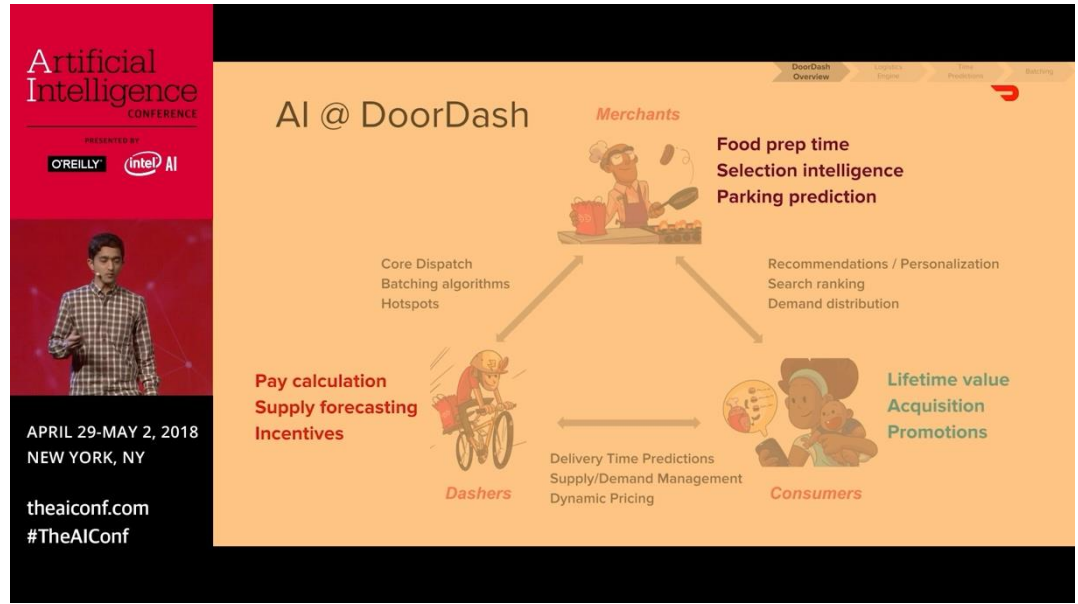
DoorDash implemented the real-time merchant intelligent pipelines to enable the smooth automation of the ordering and delivery process.



Dasher Wait Times in a machine learning model, depicted below, to intelligently calculate dasher wait times, and other factors.



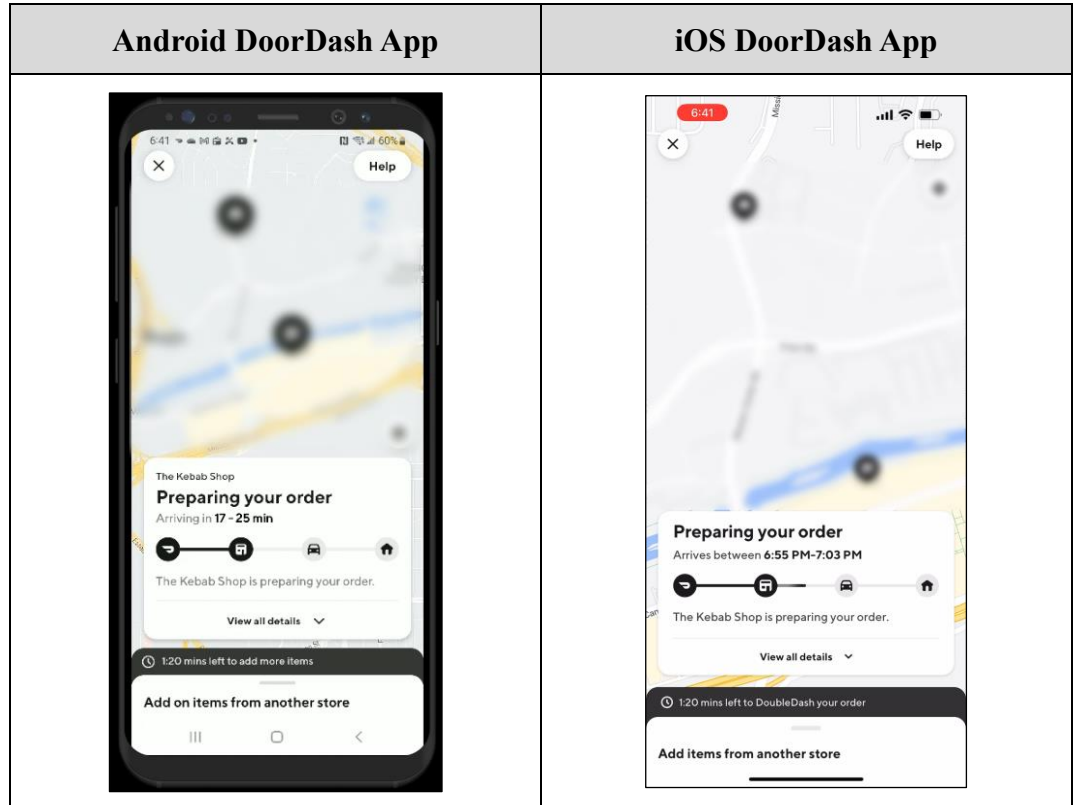
The technical video “How DoorDash Leverages AI In Its Logistics Engine” presented by the DoorDash engineering team describes how DoorDash is using Artificial Intelligence (AI) to create automated intelligent decisions regarding dasher wait time, delivery time, prep time, dynamic pricing, merchant availability, etc. (See Exh. 50).

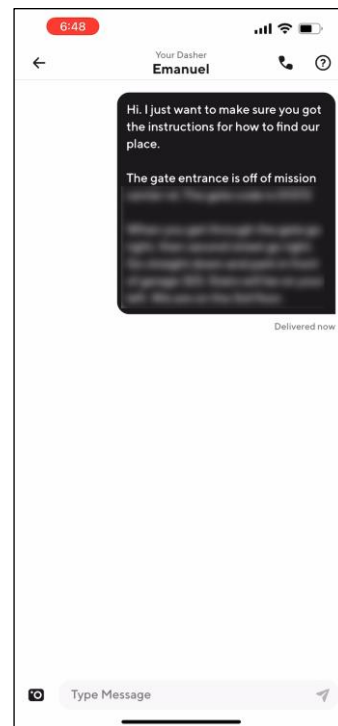
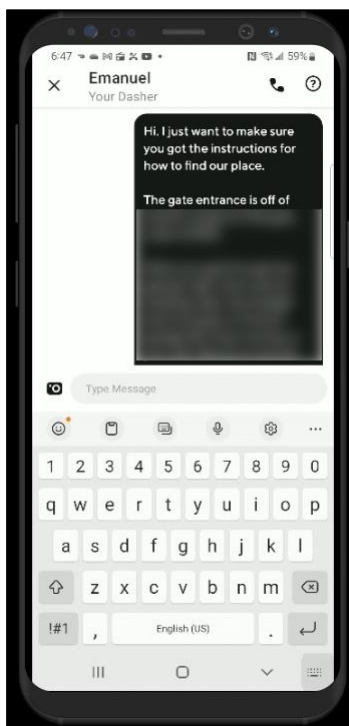
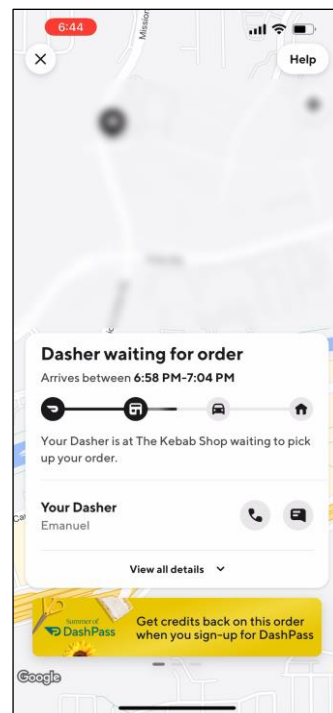


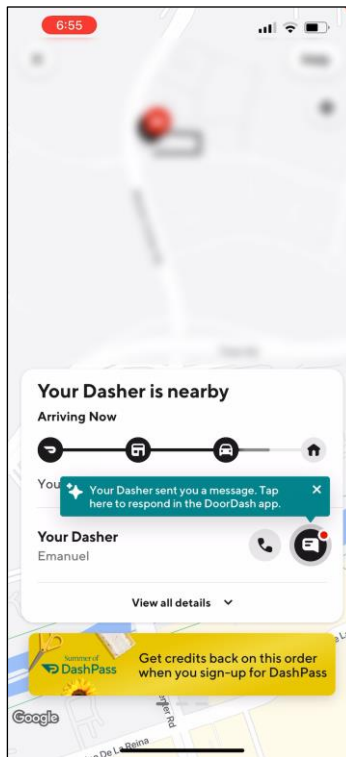
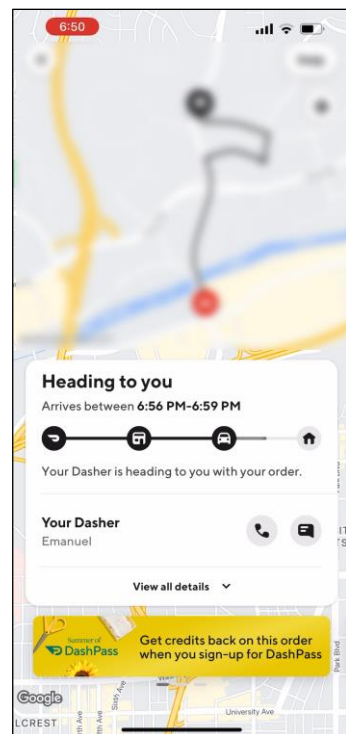
Order Tracking and Notification

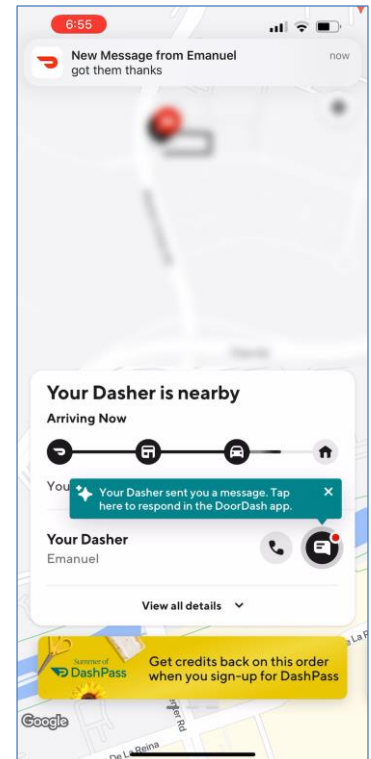
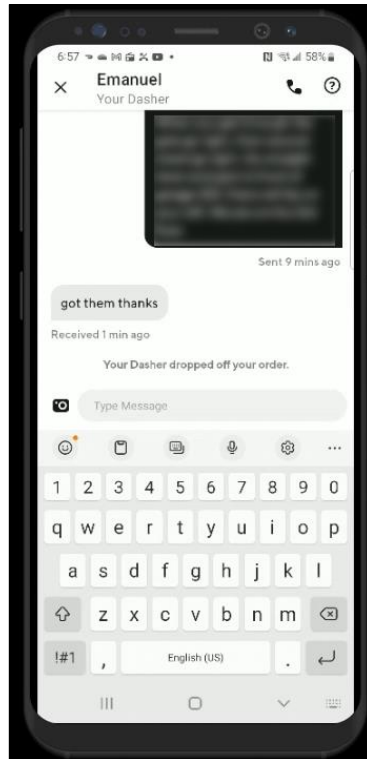
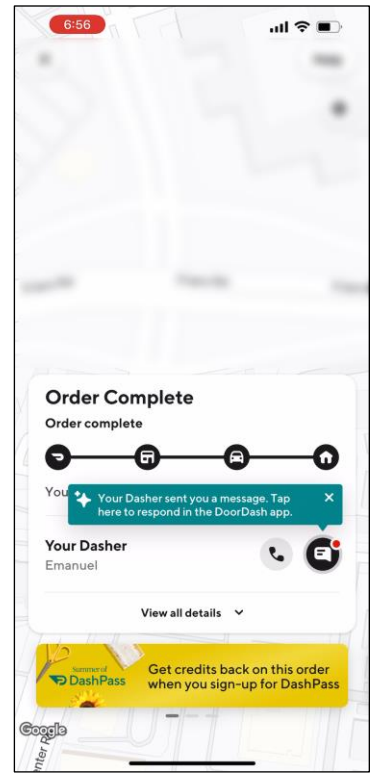
The same holds true for order notifications, order tracking and texting/chatting between the dasher and user. Below are a series of screenshots taken from one iPhone and one Android smartphone each tracking the same DoorDash order placed by a user in the DoorDash Android app (running on the android phone). They depict the order notifications and order updates in the app for the user to be able to track their order and delivery.

In this case, we have one iPhone and one Android device tracking the same order, we can see the intelligent decisions being made by the Dispatcher regarding estimated time of delivery, and the dasher time and distance from fulfilling the order. We also see text messages placed and sent by the user to the Dasher on the android app, being reflected in the iOS app on the iPhone that is tracking the order. When the Dasher responds to the chat from the user, the user is notified on the android phone and iOS phone.









Furthermore, in the engineering article “Building a More Reliable Checkout Service at Scale with Kotlin” (See Exh. 78) they state that integration of another large enterprise database called Casandra into their overall system. Thus, their databases and data

warehousing are integrated into their master database managed by the various microservices as stated below:

“The technologies we used to ensure reliability. We want to avoid losing orders or leaving them in a limbo state during the checkout process and we consider this as the most important reliability feature, we need to support in the consumer checkout flow. This essentially requires us to be able to:

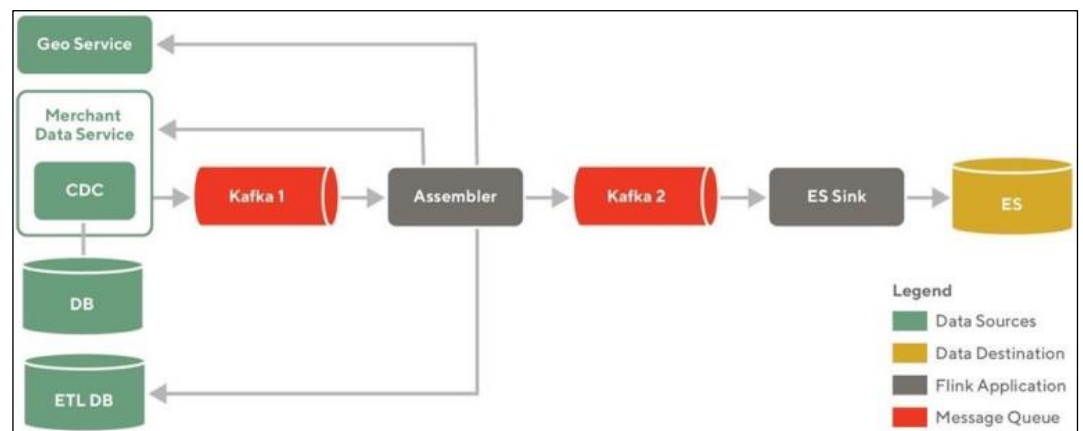
Cassandra: We use Cassandra as our primary data storage to persist order-related data. To maintain backward compatibility, data is also written back to the Postgres database. Cassandra's high availability, scalability, and multiple-AZ replication empower us to scale horizontally. The support of the KV model allows persisting order data in a more efficient and flexible way.

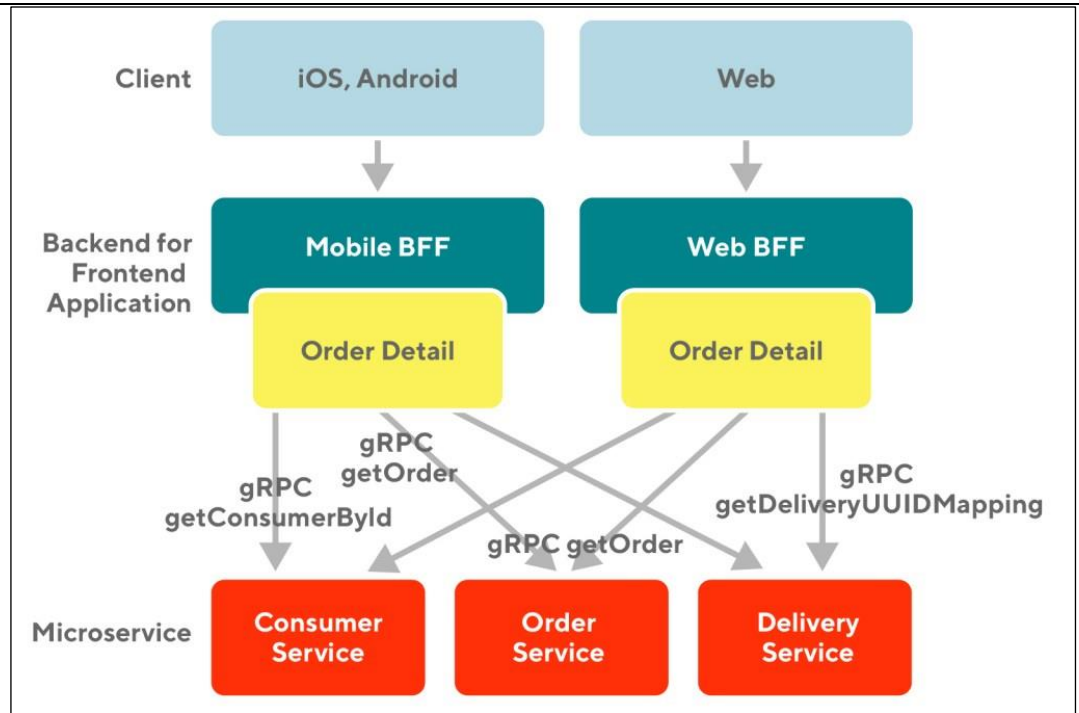
Conclusion: To summarize, a Kotlin and gRPC microservice architecture with a tech **stack** consisting of Cassandra, Kafka, and Cadence can help improve the reliability, performance, and scalability of the checkout flow.

Any company growing out of their monolith and facing similar problems with checkout or similar flows should consider the results shown by our migration.”

In the article “How does DoorDash build faster indexes using Apache Kafka and elasticsearch?” that DoorDash’s old database indexing system was neither reliable nor scalable. In fact, it was very slow. To fix this problem, DoorDash developers built an event driven architecture for fast indexing. As they state (see Exh. 80):

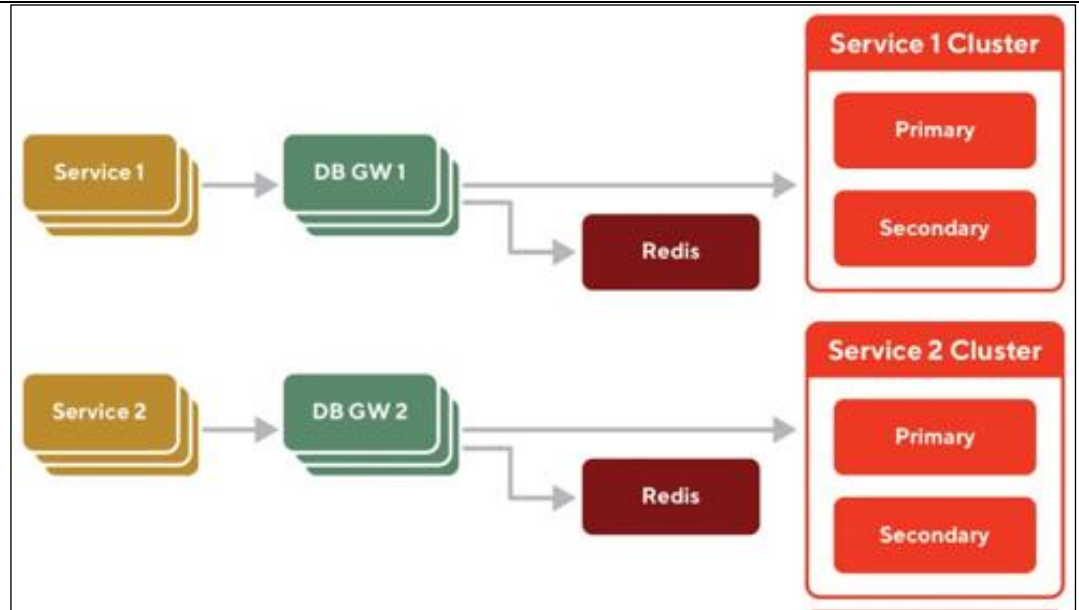
“...we solve[d] these problems by building a new search index platform. The platform provides fast and reliable indexes to support different vertical industries. It also improves search performance and the productivity of search teams. It uses Kafka as a message queue and data store ...Flink Perform data conversion and send data to Elasticsearch” as depicted by the diagram below .”





In the article “From Monolith to Microservices: Reducing the Migration’s Pain Points” from the DoorDash Engineering team (See Exh. 79), they further affirm that

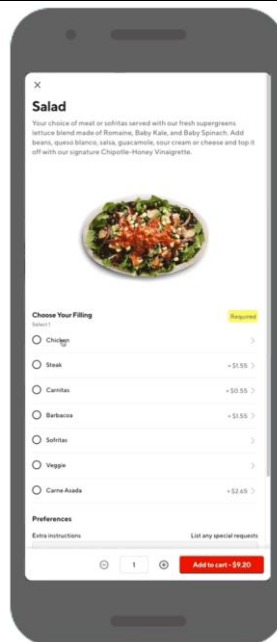
“...one of the advantages of moving to a microservice architecture is the ability to experiment with new database technologies that might fit a specific use case better than others. But, at the end of the day, there’s a chance that most services in an engineering organization are using homogeneous DB types”. The article continues with “...DoorDash’s Core Platform and Storage teams have recently invested in a centralized data access layer in the form of a DB gateway, which is deployed in isolation for each DB cluster and replaces the SQL interface for microservices with an abstract API served by a gRPC gateway. Such a gateway needs many precautions, such as isolated deployments, versioning, and configuration, to make sure it doesn’t become a single point of failure. [The figure] ... below, shows at a high level what this data gateway looks like.” It shows that “...[t]he data access layer becomes the point-of-contact between services and their storage, hiding complexity such as caching or routing [within the DoorDash microservices architecture”.



Ameranth has tested the DoorDash system by placing actual food/drink delivery orders through completion (as is shown below). We confirm that the information, data and operational parameters Ameranth entered into the DoorDash hospitality mobile applications (both Android and IOS based), mimicking typical consumer users, is updated and stored in DoorDash’s “master database”. This includes “multiple modes of communications” such as “adding the text messaging communication mode option” was intelligently applied and used. Moreover, we confirm this feature to be part of DoorDash’s merchant mobile application, and that it enables the optional ‘text messaging communication mode’, as is shown and confirmed further below.

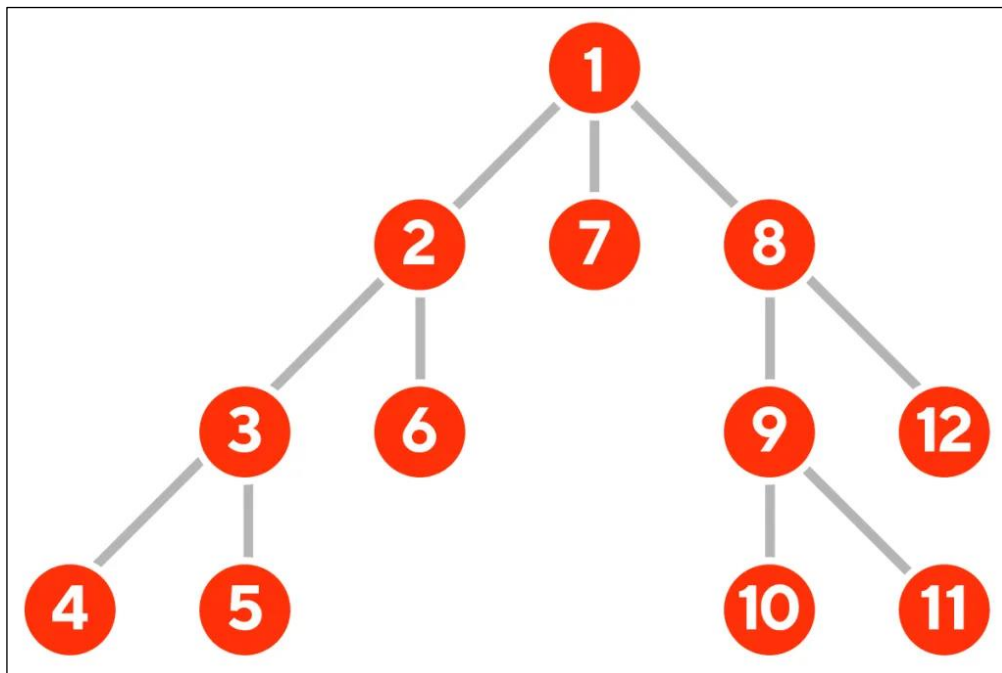
In the blog article “Managing React State on DoorDash’s Item Modal Using the Class Pattern” by the DoorDash Engineering team, further confirms the use of “menu tree structures” and “predetermined formats” (See Exh. 27). The article states that:

“[The figure below shows the] Item Modal is a dynamic form in which we display item data to our users, take and validate user inputs based on boundary rules set by the item data, dynamically calculate item prices based on these user inputs, and submit valid user-modified items to a persistent data store.”



It further describes the TreeState as:

“The most important characteristic of the Item Modal rebuild is that it is a TreeState, which is represented as an N-ary tree, as shown in [the figure below]:”

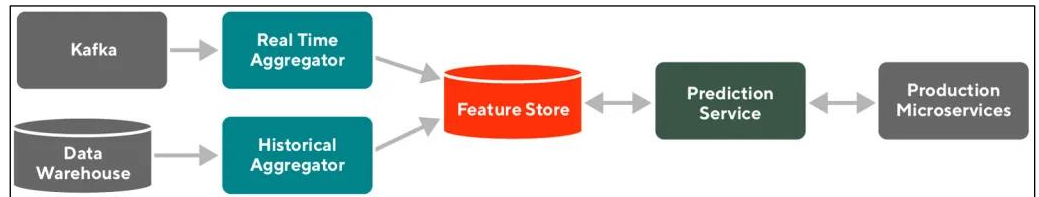


“[This figure shows that the] ... TreeState used in our Item Modal is represented as an N-ary tree. In this usage, every item, has an ItemNode, and this ItemNode can have any number of OptionListNode. Each OptionListNode can have any number of OptionNodes, and these OptionNodes can have any number of OptionListNodes, and so on.”

In the blog article “Building Riviera: A Declarative Real-Time Feature Engineering Framework” by the DoorDash Engineering Team (See Exh. 105) they further assert their database APIs and layered architectural approaches. Furthermore, they state:

“Leveraging the Apache Flink stream processing platform, we built an internal framework, which we call Riviera, that allows users to declaratively specify their feature transformation from source(s) to features stores through a simple configuration.

Within DoorDash’s ML Platform, we have worked on establishing an effective online prediction ecosystem. Figure 1, below, gives a high-level overview of our ML Infrastructure in production. We serve traffic on a large number of ML Models, including ensemble models, through our SibylPrediction Service. Because the foremost requirement of our prediction service is to provide a high degree of reliability and low latency (<100 ms), we built an efficient feature store to serve aggregated features. We use Redis to power our gigascale feature store to provide high throughput and availability for our features.”



“[The above figure shows] ...our ML Platform architecture, we serve ML models through a prediction service which relies on a Feature Store to provide aggregate features in production.”

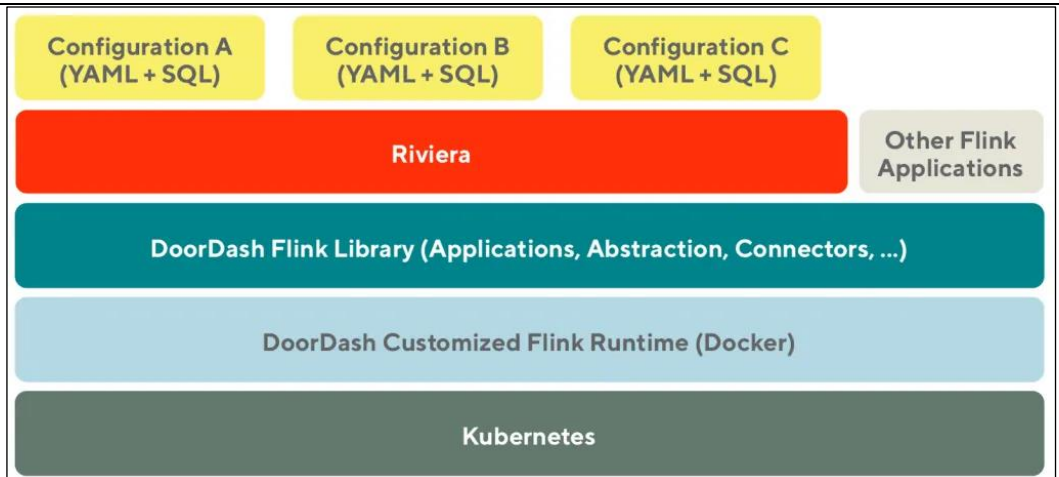
The article then provides an overview of the Flink-as-a-service platform:

“To help build sophisticated stream processing applications like Riviera, it is critical to have a high-quality and high-leverage platform to increase developer velocity. We created such a platform at DoorDash to achieve the following goals:

- Streamline the development and deployment process
- Abstract away the complexities of the infrastructure so that the application’s users can focus on implementing their business logic
- Provide reusable building blocks for applications to leverage

The following diagram shows the building blocks of our Flink-as-a-service platform together with applications, including Riviera, on top of it.”

“...Flink-as-a-service provides multiple levels of abstractions to make application development easier.”



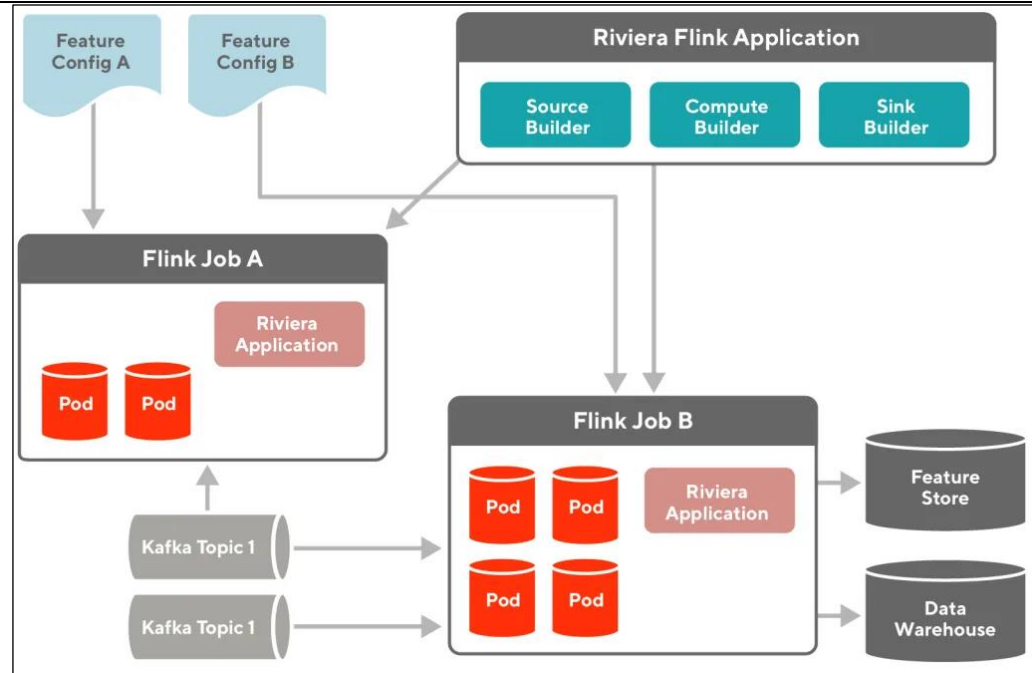
“Most of DoorDash’s infrastructure is built on top of Kubernetes. In order to adopt Flink internally, we created a base Flink runtime docker image from the open-source version. The docker image contains entry point scripts and customized Flink configurations (flink-conf.yaml) that integrate with DoorDash’s core infrastructure, providing integrations for metric reporting and logging.”

Creating a generic Flink application in Riviera

“Building on issues with Flink that needed to be addressed and the existing state of our infrastructure, we designed Riviera as an application to generate, deploy, and manage Flink jobs for feature generation from lean YAML configurations.

The core design principle for Riviera was to construct a generified Flink application JAR which could be instantiated with different configurations for each feature engineering use case. These JARs would be hosted asst and alone Flink jobs on our Kubernetes clusters, which would be wired to all our Kafka topics, feature store clusters, and data warehouses. Figure 3 captures the high-level architecture of Riviera.”

"[The figure below shows] ...A Riviera Flink application constructs sources, transformation operator graphs and sinks in Flink from their YAML configurations and then runs them on the Flink-as-a-service platform.”



“The launch of Riviera enabled feature development to become more self-serve and has improved iteration life cycles from a few weeks to a few hours. The plug-and-play architecture for the DSL also allows adapting to new sources and sinks within a few days.

The integration with the Flink-as-a-service platform has enabled us to automate our infrastructure by standardizing observability, optimization, and cost management behind the Flink applications as well, allowing us to bring up a large number of jobs in isolation with ease.

The library utilities we built around Flink’s API and state management have reduced codebase size by over 70%.”

In the video interview “SFBigAnalytics_20220920: DoorDash: Building Scalable Real Time Event Processing with Kafka and Flink” by Allen Wang “[the] lead engineer at DoorDash working on real-time data infrastructure” (See Exh. 120), he admits the following:

“...as mentioned in the beginning one important objective for Iguazu is to create a platform for easy data processing Apache flink layered API architecture fits perfectly with this objective.”

“...to support SQL based applications we created a framework called Vdara where all where all the necessary processing logic and wiring are captured in the yaml file which is simple enough for everyone to read or Auto the yaml file captures a lot of a high level abstractions for example connecting to Kafka sources and different things and the processing logic is expressed as a SQL statement and currently Riviera's primary use cases are real-time feature engineering and real-time monitoring of mobile application metrics.”

“...so the same person ozone as when you say framework should be created how do you design a framework please elaborate it's like a pretty general question um

yeah I think um as I mentioned before um you're keeping your customers in mind uh creating different kind of abstractions make sure that um that whenever you see opportunities uh try to um put all the common functionalities in in your platform and so that you can reduce the you know the necessary workload from on other applications uh and really just to make sure that um that as a as a platform you will capture uh all the all the necessary interactions with the infrastructure and then that the application uh to worry you know only about its things logic but not you know at the infrastructure level.”

See Exh. 120.

In a video by DoorDash's engineering team title “Databases at DoorDash | How DoorDash manages 1.9 PB of data & 1.2M QPS on CockroachDB”, DoorDash showcases their utilization of Cockroach DB as the “master database” for managing massive amounts of data, demonstrating their implementation of a master database. The video begins with Mike Czabator, an engineer with both DoorDash and Cockroach Labs experience, providing insights into DoorDash's database practices. DoorDash's core infrastructure team is responsible for a range of databases, including CockroachDB, showcasing a dedicated approach to database management. (See Exh. 122 – 123)

DoorDash's deployment strategy spans a single AWS region, supported by multiple Availability Zones (AZs), highlighting their scalability. The substantial growth metrics, including a 55% node increase and nearly doubling data size to two petabytes (PBs), highlight the extensive scope of their data management. Furthermore, DoorDash promotes a self-service model, allowing users to create applications and users for their clusters. They emphasize automation and feedback-driven improvement, enhancing the efficiency and reliability of their database operations. DoorDash's commitment to performance optimization and continuous enhancement aligns with the concept of a master database, reinforcing their capability in handling vast amounts of data.

In another video by the DoorDash Engineering team titled “DoorDash’s Journey from Aurora Postgres to CockroachDB”, they shared insights into their data management practices, particularly their utilization of CockroachDB for handling extensive data volumes with the “master database”. DoorDash's primary challenge was managing high traffic loads, which often led to database crashes during peak usage. To address this, the team recognized the need for a database solution that could provide horizontal scalability through multiple writers. (See Exh. 124 - 125)

During the COVID-19 lockdown, DoorDash faced a pivotal moment when their main database cluster experienced a peak query per second (QPS) load of 1.6 million, resulting in system downtime. To tackle this issue, they opted to migrate to CockroachDB, a distributed database system known for its scalability and fault tolerance. Their migration strategy involved the extraction of 54 tables to seven new database clusters, significantly reducing the QPS load on the main database cluster. To facilitate this migration, DoorDash developed a versatile data migration tool. This tool was designed to minimize manual intervention, support schema transformations, and streamline data synchronization. Key features of the tool included the ability to customize runtime behavior, accommodate schema changes, and perform data chunking for parallel processing. DoorDash also leveraged foreign data wrappers,

intelligent traffic routing, and data chunking to optimize the migration process.

In the Jul 28, 2022, technical video presentation “DoorDash Technical meetup event: Scaling our Data Platform”, the admit ‘predefined formats’ at 33:45 to 34:44 and at 39:45 (See Exh. 47), as shown by the following screenshots:

The screenshot shows a presentation slide with the title "Event format and schema enforcement". In the top right corner, there is a small video feed of Allen Wang. The slide contains the following text:

- Unified event format with envelope and schematized payload
- Supporting both Avro and Protobuf schema
- Build-time schema registration and update with schema registry

Below the text is a diagram showing the workflow: A "Protobuf repo" (represented by a blue cylinder) has an arrow labeled "CI/CD (Update)" pointing to a "Schema registry" (pink box). From the "Schema registry", an arrow labeled "Read-only" points to "Internal Services" (pink box). The slide number "44" is in the bottom right corner.

The screenshot shows a presentation slide with the title "What's a better ideal?". In the top right corner, there is a small video feed of Kunal Shah. The slide contains the following text:

- Single entry point
- Automated and standard downstream interactions
- Seamless sharing with code and feature reuse

Below the text is a complex diagram illustrating a feature management and serving architecture. It includes "Users" (represented by two people icons) interacting with a "Feature Registry". The "Feature Registry" is connected to "Feature Management" (a red box containing "Feature Discovery" and "Feature Observability"). The "Feature Registry" also feeds into "Feature Generation" and "Offline Feature Serving". "Feature Generation" is connected to a "Data Warehouse". "Offline Feature Serving" is also connected to the "Data Warehouse". Below this, there is a "Feature Serving" section (green box) containing "Online Feature Serving", "Online Storage", and "Online Model Serving". Arrows indicate the flow of data and features between these components.

In the video interview “SFBigAnalytics_20220920: DoorDash: Building Scalable Real Time Event Processing with Kafka and Flink” by Allen Wang “[the] lead engineer at Doordash working on real-time data infrastructure” (See Exh. 120), he admits the following:

“...two years ago, we started to rethink this approach and decide to build a new system to replace those Legacy pipelines and address the future event processing needs they'll be anticipated so first uh you should support heterogeneous data

sources including microservices and mobile or web applications and be able to deliver the events to different destinations. Secondly it should be easily accessible for data consumers to tap into streams of data and build their own processing logic to improve data quality we want to have end-to-end schema reinforcement and schema evolution”.

“...however without a unified event format it's still difficult for producers and consumers to understand each other so in the next section we will discuss the event format and schemas which serve as a protocol between producers and consumers so from the very beginning we Define a unified format for events produced and processed in Iguazu the unified event format greatly reduced the barrier in consuming events and reduce the frequency between the event producers and consumers so instead we concluded that it will be ideal for uh to register and update this schema at build time to reduce the update API call volume to schema registry and to catch incompatible schema changes early in the development cycle.”

“...well one challenge we faced is how we can essentially automate the schema update at build time and the solution we created is to leverage the fact that all of our approach above schemas are managed in a single Repository and integrate the schema registry update as part of the CI CD process when the protobot definition is updated.”

“...as I mentioned in the beginning of the talk data warehouse integration is one of the key goals of Iguazu snowflake is our main data warehouse solution we expect events to be delivered to snowflake with a strong consistency and low latency the data warehouse integration in Brazil is implemented as a two-step process secondly I'm picking the right framework and creating the right building blocks is crucial to ensure success researching and leveraging The Sweet Spot of those Frameworks dramatically reduces the time needed to develop and operate these large-scale event.”

(See Exh. 120.)

The accused instrumentality meets this limitation under the doctrine of equivalents because the accused instrumentality performs substantially the same function, in substantially the same way to achieve substantially the same result. For example, DoorDash's 'master database,' operating as a 'central data platform' or 'central data lake,' embodies the core essence of the 587 patent's master database claim element. It serves as a structured repository that enables efficient data management, integrates machine learning to intelligently handle communication modes and operational parameters, and is accessible through an API for various users. The database's design for learning, updating, and storing information aligns closely with the patent claim's emphasis on a master database that not only stores but also intelligently processes data to enhance efficiency and reliability, particularly within the realm of hospitality software applications. DoorDash's implementation of modern data warehousing techniques and microservices architecture further mirrors the 587 patent's vision for an adaptable and efficient data infrastructure. The integration with analytics and machine learning tools for real-time communication modes and operational adjustments demonstrates functional equivalency, as endorsed by the Doctrine of Equivalents.”

Middleware/Framework Communications Control Software (MFCCS) which enables via its centralized system layer architecture the parallel operations capable network to communicate with said two or more different wireless handheld computers, each with different mobile operating systems and with mobile compatible versions of the said hospitality application accessible from the back office servers;

DoorDash's implementation of Middleware/Framework Communications Control Software (MFCCS) is evident in their centralized system layer architecture, which facilitates communication between web server networks and various wireless handheld devices. This system, as described by Jessica Lachs, encompasses DoorDash's event processing system named Iguazu, developed with Apache Flink and Kafka, underscoring the platform's scalability and real-time processing capabilities.

Central System Layer Architecture

The central system layered architecture of Iguazu demonstrates DoorDash's commitment to a design that enables seamless interaction with multiple mobile operating systems and GUI screens. This allows for the initiation of user actions and subsequent selection of choices directly from handheld devices, aligning with the claim's requirement for MFCCS.

Furthermore, DoorDash's transition from a monolithic to a microservices architecture has bolstered their platform's efficiency and reliability, according to the DoorDash engineering team. Breaking up their application into domain-specific services reduced errors and latency, with the microservices structure providing separate layers for front-end systems, backend-for-frontend (BFF) services, backend logic, platform services, and foundational infrastructure.

The DoorDash engineering blog and various technical presentations provide insights into their advanced master database integration and the hospitality applications, confirming the existence of a multi-layered communication framework. This framework supports a variety of tasks for consumers, Dashers, and merchants, proving the platform's ability to handle multiple communication modes and protocols.

In practice, the MFCCS is manifested through the order tracking, notification systems, and in-app chat functionalities of DoorDash's consumer and Dasher applications across iOS and Android devices. This allows for real-time decision-making by the Dispatcher and seamless interaction between users and Dashers, showcasing the intelligent use of multiple modes of communication.

DoorDash's engineering articles and support documents further demonstrate the company's use of a unified chat experience, multi-modal communication, and a comprehensive approach to menu management through their Merchant Portal. This includes the ability for merchants to integrate with various POS systems and the provision of a Business Manager app, which allows for real-time operational management.

The integration of open-source frameworks, the application of machine learning for real-time event processing, and the orchestration of complex data flows through a centralized API are all indicative of a sophisticated MFCCS as described in the patent claim. DoorDash's system exemplifies a robust, intelligent, and adaptive communication control software that is integral to the modern hospitality market.

DoorDash Iguazu Event Processing System

In the technical presentation video "Building Scalable Real Time Event Processing with Kafka and Flink" by the DoorDash Engineering team (see Exh 91) they present that

"...[t]wo years ago, DoorDash started the journey of creating a real time event processing system to replace the legacy data pipelines and address our event

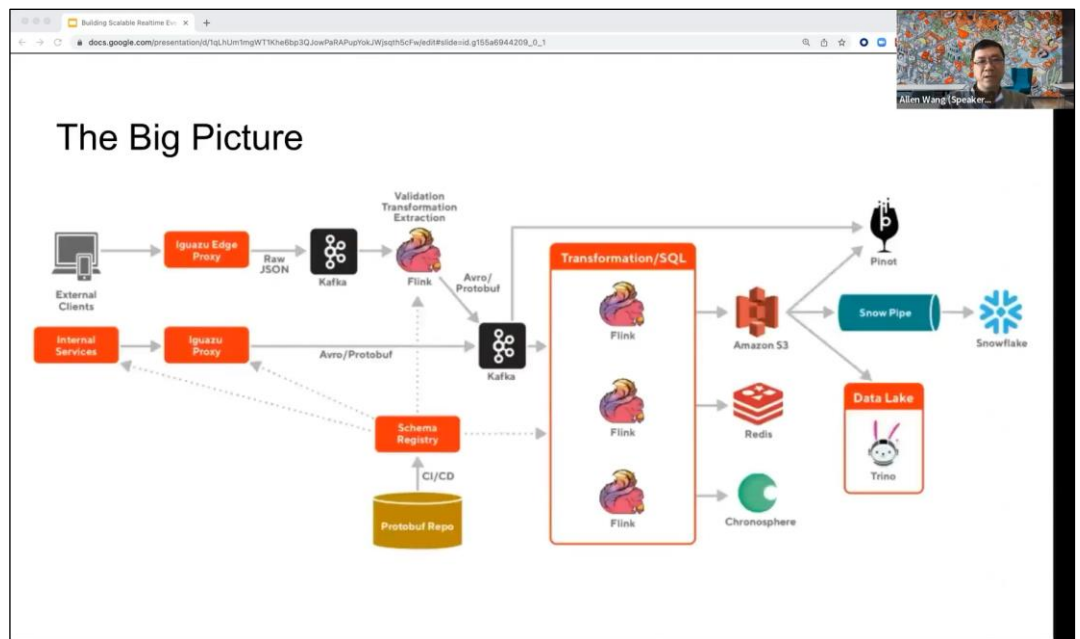
processing needs to scale our business. We created a scalable system that could handle heterogeneous data sources and destinations, was easily accessible with different levels of abstractions and had end to end schema enforcement. We were able to accomplish this by shifting our strategy from heavily relying on AWS and third-party data services to leveraging open source frameworks that can be customized and better integrated with our infrastructure.

In this session, I will share what we learned and how we put together this system with Apache Flink, Kafka as well as Kubernetes.”

They discuss the Iguazu real time event processing system they development from scratch and it’s architecture.”

Introducing Iguazu

- Iguazu - building real time event processing from scratch
 - Emphasis on integration of open source frameworks with DoorDash’s infrastructure
- Current scale and SLO
 - **Hundreds of billions** events processed per day
 - Hundreds of events and pipelines
 - **99.99%** delivery rate
 - Sub second latency for streaming consumers, and minutes latency to data warehouse



The diagram above depicts the layered system architecture of Iguazu, DoorDash’s event processing system. It depicts the inner functioning of Iguazu taking from the DoorDash

engineering blog article “Building Scalable Real Time Event Processing with Kafka and Flink” (see Exh 53) and the DoorDash technical video “Scaling our Data Platform” (see Exh 47).

In the DoorDash engineering blog article (see Exh 53) “Building Scalable Real Time Event Processing with Kafka and Flink”, they state that:

“...[t]wo years ago, we started the journey of creating a real time event processing system named Iguazu to replace the legacy data pipelines and address the following event processing needs we anticipated as the data volume grows with the business:

- ***Heterogeneous data sources and destinations:*** *Data ingest from a variety of data sources including the legacy monolithic web application, microservices and mobile/web devices, and delivery to different destinations including third-party data services. Reliable and low latency data ingest into the data warehouse is a high priority.*
- ***Easily accessible:*** *A platform that makes it easy for different teams and services to tap into the streams of the data and build their own data processing logic.*
- ***End-to-end schema enforcement and schema evolution:*** *Schema only improves data quality, but also facilitates easy integration with data warehouses and SQL processing.*
- ***Scalable, fault-tolerant, and easy to operate for a small team:*** *We want to build a system that can easily scale to the business need with minimal operational overhead.”*

Kafka is an open-source platform from Apache, excels in stream processing and data management. It adeptly handles the reception, storage, organization, and distribution of voluminous data streams to a diverse array of end-users and applications. However, the influx of substantial data payloads—spanning hundreds to thousands of messages—into Kafka's servers can precipitate challenges such as data overloading and duplication. These issues, in turn, can result in the data within Kafka servers becoming disorganized and obscured, complicating effective data management and utilization.

Apache Kafka's architecture includes a key structural element known as Topics, which serve as the primary unit for organizing events or messages. Essentially, Kafka Topics function as virtual groups or logs, systematically storing messages and events in a sequential manner. This organization facilitates the seamless transmission of data between Kafka servers. Each topic acts as a dynamic repository, where messages sent by producers are chronologically appended, forming an evolving log file.

In practice, producers inject messages into these topics, adding to the tail end of the log, while consumers extract messages from specified topics, ensuring a streamlined flow of data. This methodology enables logical segregation of messages and events, akin to how distinct tables in a database hold varied types of data. In the Kafka ecosystem, the creation of multiple topics is permitted, catering to diverse use cases. Crucially, each topic must bear a unique, identifiable name to maintain clear distinction among various Kafka brokers within a Kafka cluster, thereby ensuring efficient data management and retrieval.

The article from DoorDash Engineering Blog titled "API-First Approach to Kafka Topic

Creation" discusses how DoorDash's Engineering teams have improved their Kafka Topic creation process. They replaced a Terraform/Atlantis-based approach with an in-house API Infrastructure Service, which has led to a 95% reduction in real-time pipeline onboarding time and has saved numerous developer hours. (See Exh. 126)

DoorDash's Real-Time Streaming Platform (RTSP) team, part of the Data Platform organization, manages over 2500 Kafka Topics. Kafka functions as the publication-subscription layer of the Iguazu pipeline and processes around six billion messages per day.

The article explains the challenges with the legacy architecture, where provisioning Kafka Topics was slow due to on-call engineer approval and prone to failures, increasing the on-call load. The RTSP team worked with storage and cloud teams to automate Kafka resource creation, which allowed for a more streamlined process and reduced manual intervention.

DoorDash has introduced super-user accounts to resolve merge conflicts that occurred in ACL files for Iguazu users, which has significantly sped up applications by Atlantis. The new architecture integrates with the Storage Self-Serve Platform within Infra Service, which provides an API to perform CRUD operations on infrastructure components. This new approach allows for provisioning approximately 100 new topics every week without manual intervention and significantly faster onboarding times for customers.

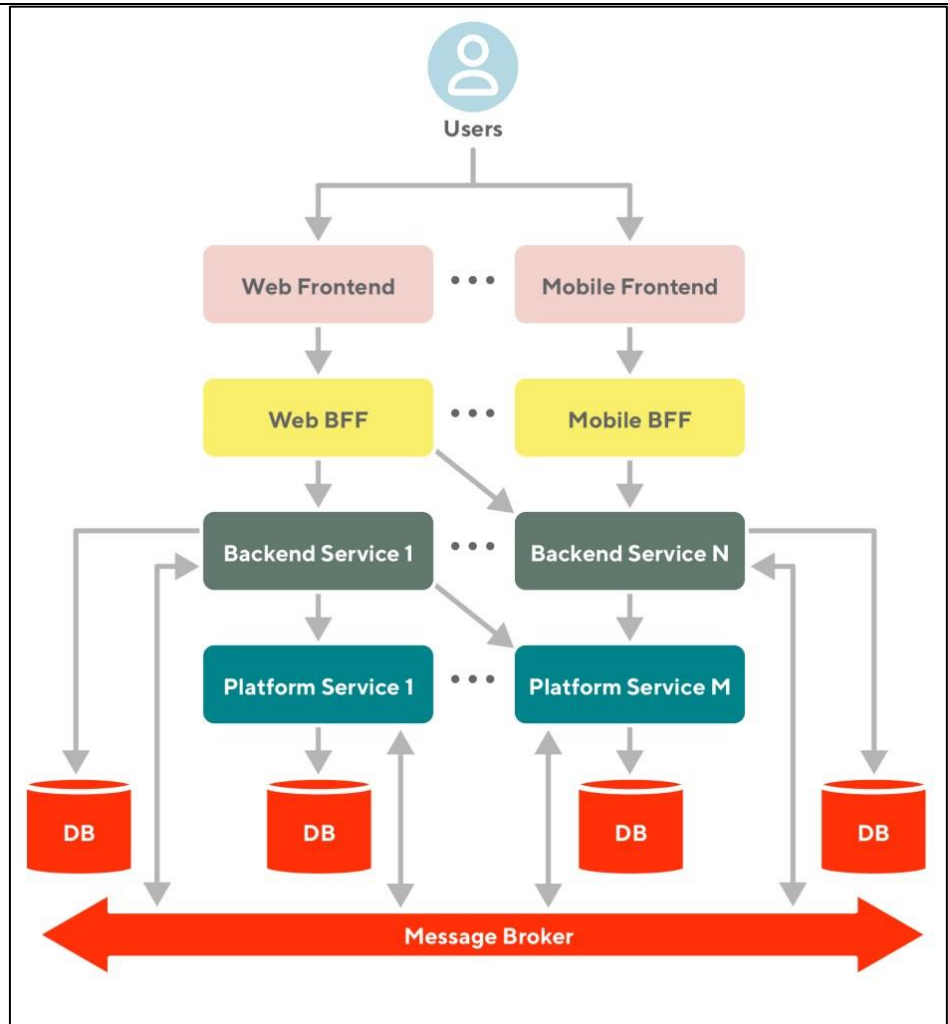
The article concludes by outlining the future direction of Kafka Automation and Storage Self-Serve, aiming to improve guardrails and customer experience. Acknowledgments are given to the various teams and engineers who contributed to this engineering win. This article demonstrates DoorDash's commitment to improving their operational efficiency and infrastructure management, ensuring faster onboarding times, and reducing the need for manual interventions, which contributes to a better overall experience for their customers and partners.

Microservices Layered Architecture

The original DoorDash platform was originally a monolithic application written in Python using the Django web framework with a PostgreSQL database. As the platform grew, they started to have problems with reliability and scaling. Around 2018 they institute a code freeze and began migrating to microservices. At this time, they also migrated to the Kotlin language, and their services now run on the java virtual machine (JVM) (see Exh 20, 21, 22).

DoorDash was able to improve the efficiency and reliability of their platform using a microservices architecture. By breaking up their application into domain-specific parts they were able to reduce errors and latency. They state in the article "Future-proofing: How DoorDash Transitioned from a Code Monolith to a Microservice Architecture" (see Exh. 20) that:

"... final design for our new microservice architecture consisted of five different layers, ranging from the user experience to the core infrastructure. Each layer provides functionality to the upper layer and leverages the functionality exposed by the lower layer [as shown below]":



These layers include: (see Exh 20):

“Frontend layer: Provides frontend systems (like the DoorDash mobile app, Dasher web app, etc.) for the interaction with consumers, merchants, and Dashers that are built on top of different frontend platforms.

BFF layer: The frontend layer is decoupled from the backend layer via BFFs. The BFF layer provides functionality to the frontend by orchestrating the interaction with multiple backend services while hiding the underlying backend architecture.

Backend Layer: Provides the core functionality that powers the business logic (order cart service, feed service, delivery service, etc.).

Platform layer: Provides common functionality that is leveraged by other backend services (identity service, communication service, etc.).

Infrastructure layer: Provides the infrastructural components that are required to build the site (databases, message brokers, etc.) and lays the foundation to abstract the system from the underlying environment (cloud service provider).”

In their new architecture they introduced the backend-for-frontend (BFF) “... an application connecting the consumer-facing client and the services providing general

purpose APIs. Client requests go to the BFF, which then orchestrates the aggregation of information needed by the client.”. The BFF is a software architecture pattern (see Exh 40) used by microservices which “...shifted from thick-client applications to **interfaces delivered via the web, a trend that has also enabled the growth of SAAS-based solutions in general**”. As such BFF can be considered as “...the user-facing application as being two components - a client-side application living outside your perimeter, and **a server-side component (the BFF) inside your perimeter**”. The perimeter of the BFF is the webserver.

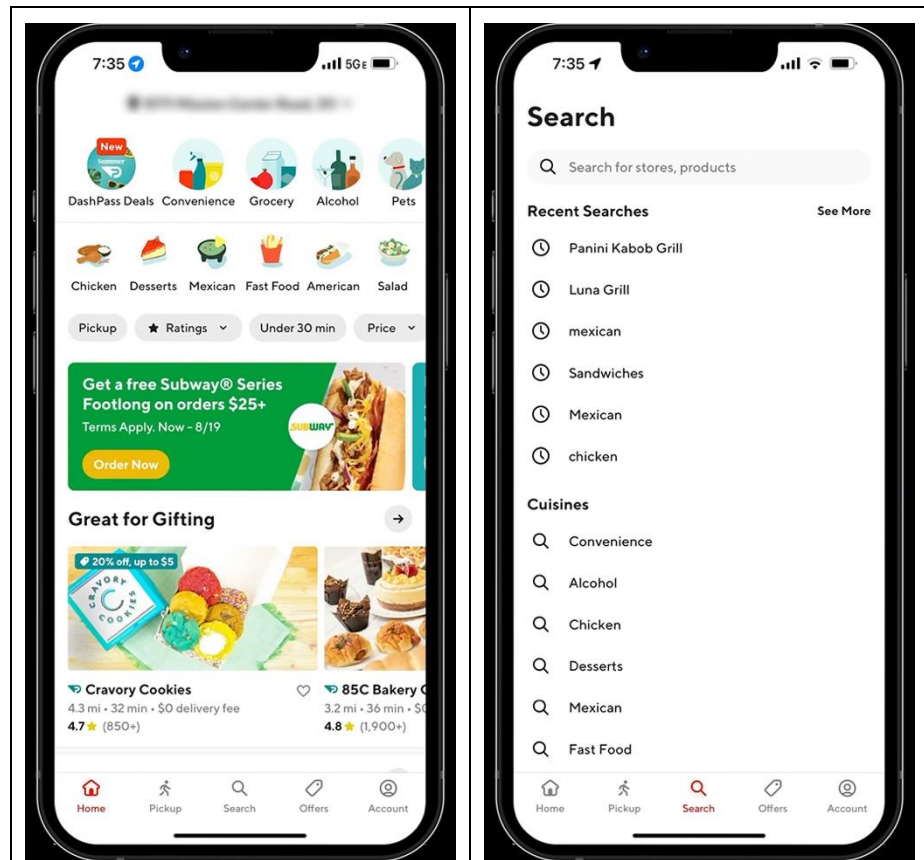
In the blog article “Building a Unified Chat Experience at DoorDash” by the Engineering Team they further admit to their back-end framework and its layered approach as follows (see Exh 102):

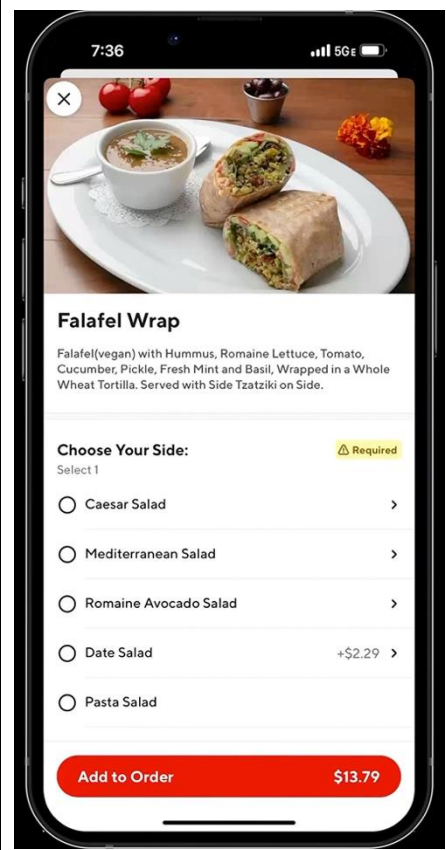
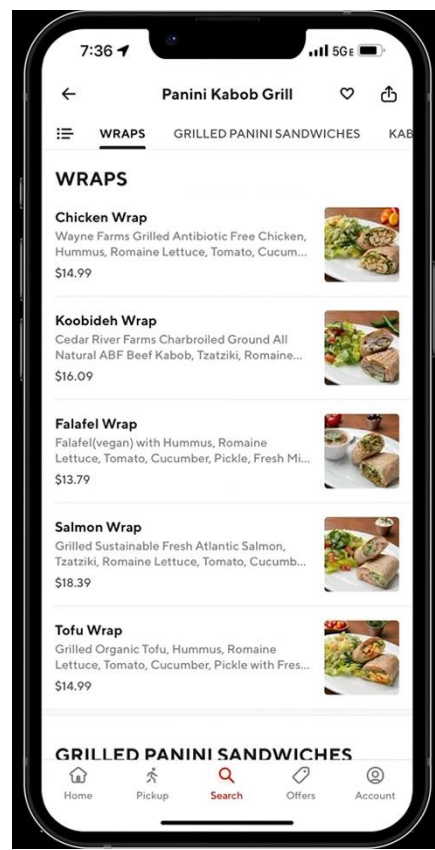
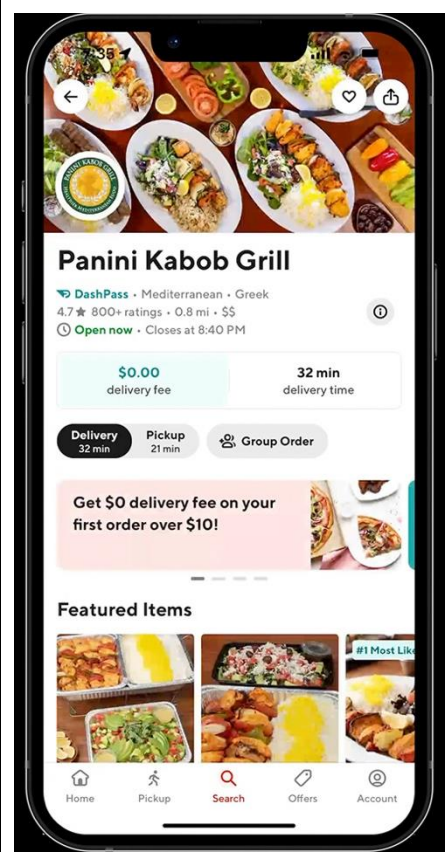
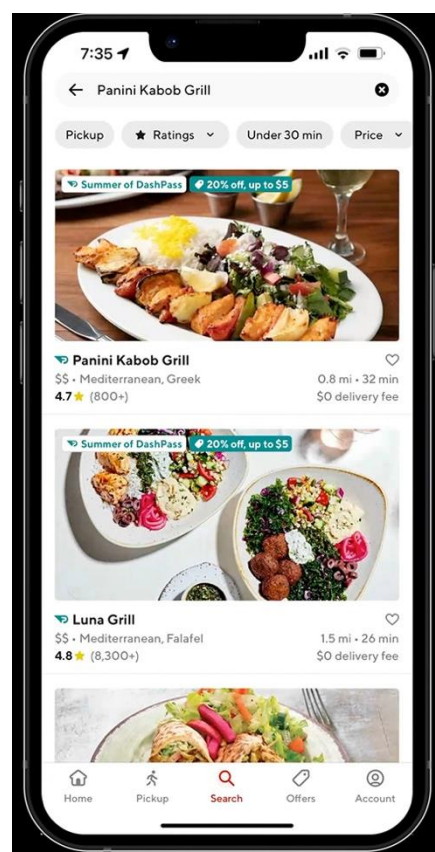
“Building an extensible backend system”

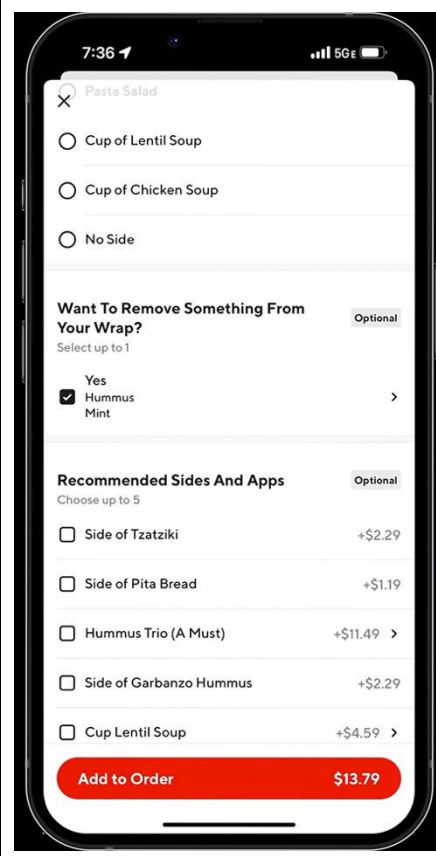
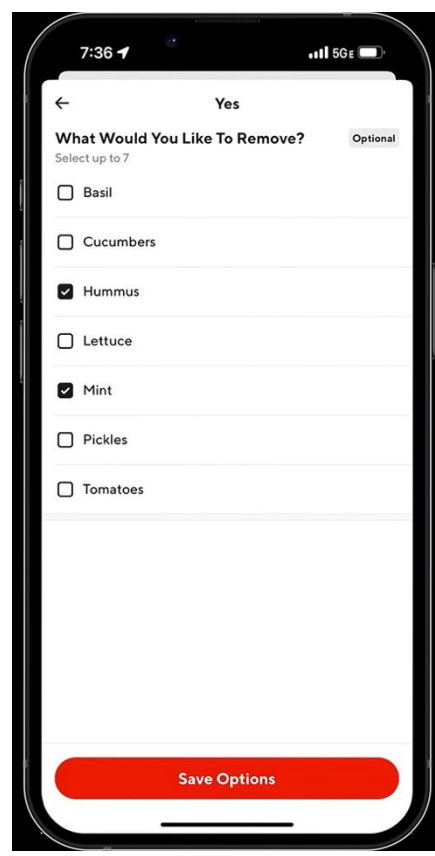
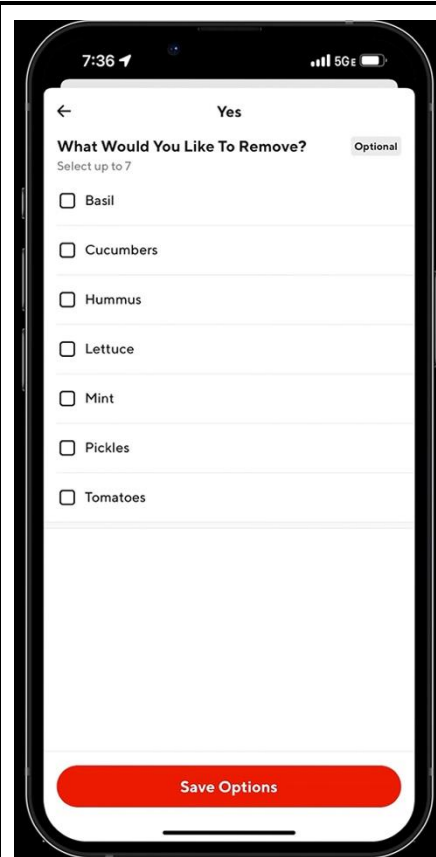
“The backend system was built with multiple layers, allowing us to split responsibilities between internal services and third parties. Utilizing third parties for functionality such as chat, natural language processing (NLP), and agent ticket management allowed us to move quicker without having to build functionality that does not differentiate ourselves directly”

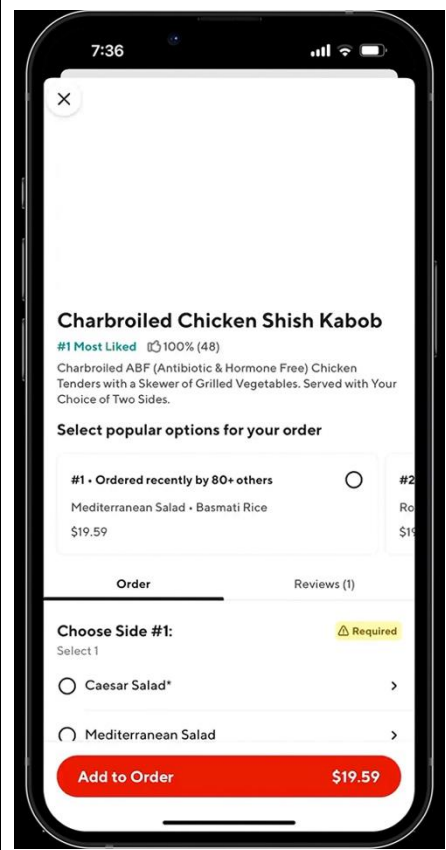
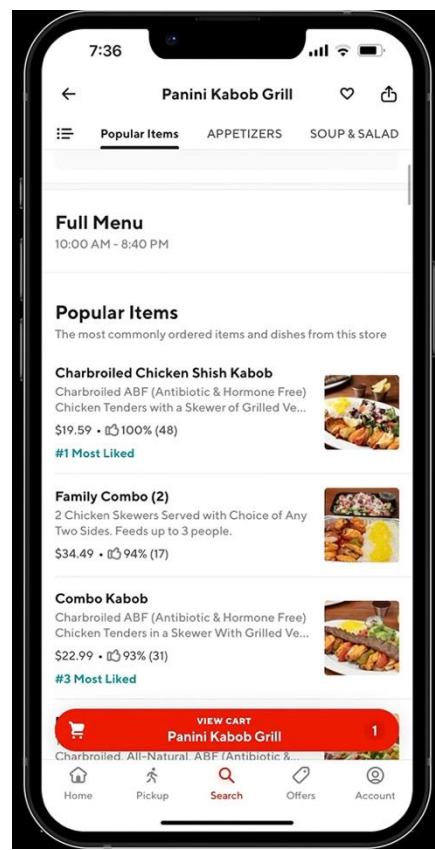
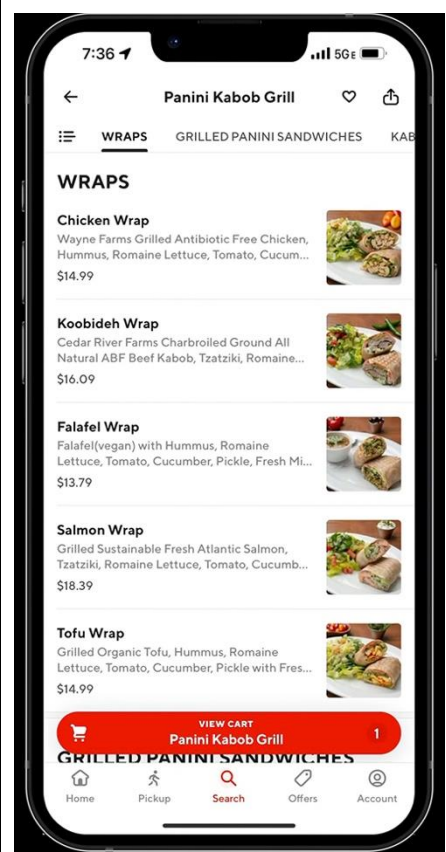
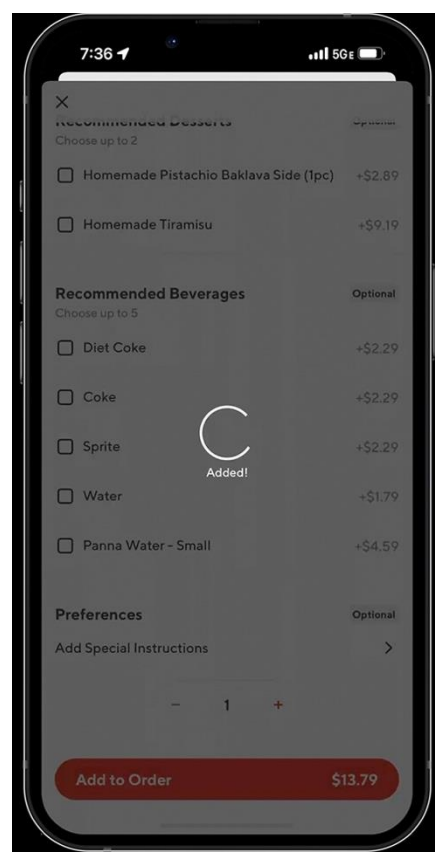
DoorDash Consumer App – iOS

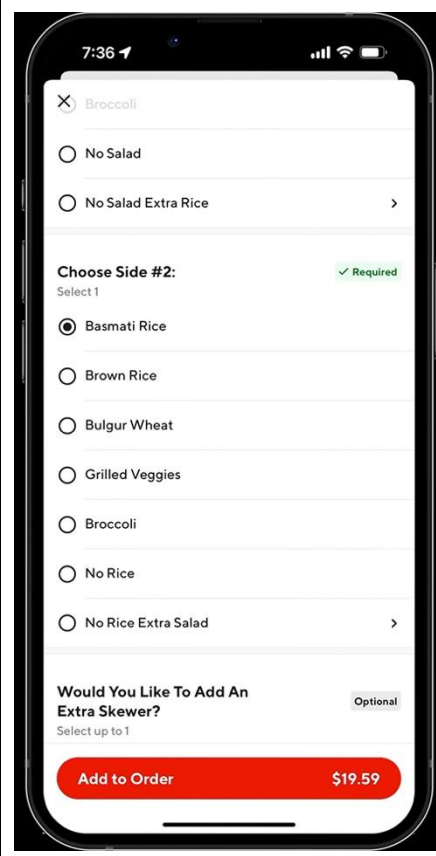
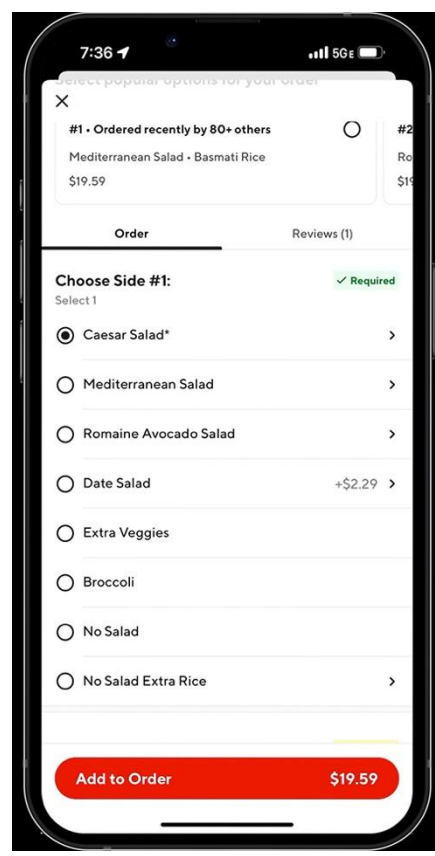
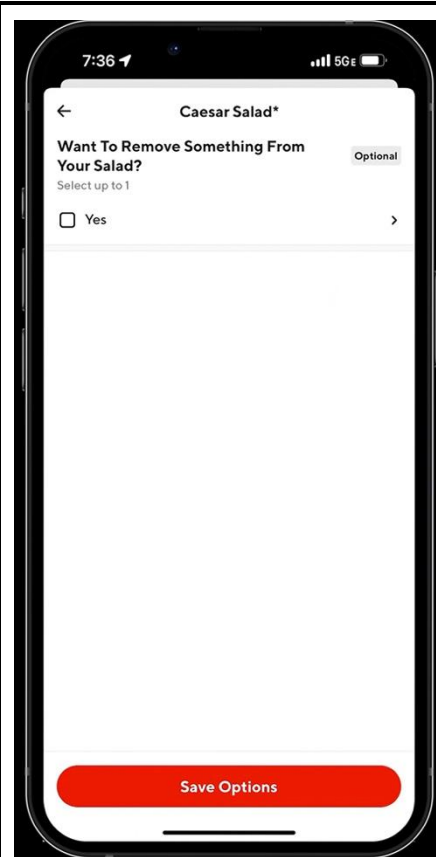
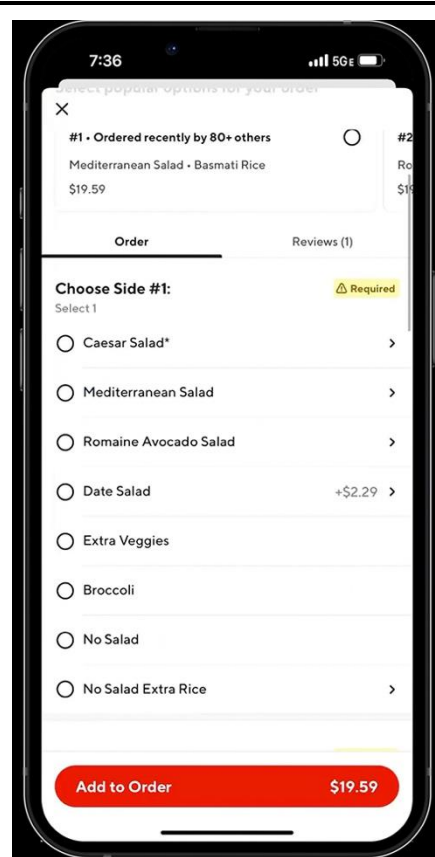
Below are screen shots of the ordering process on an iOS device. This process demonstrates the user launching the DoorDash app.

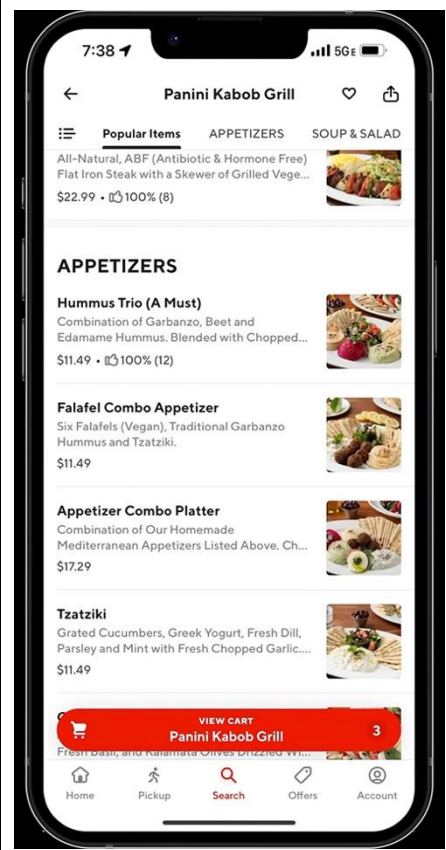
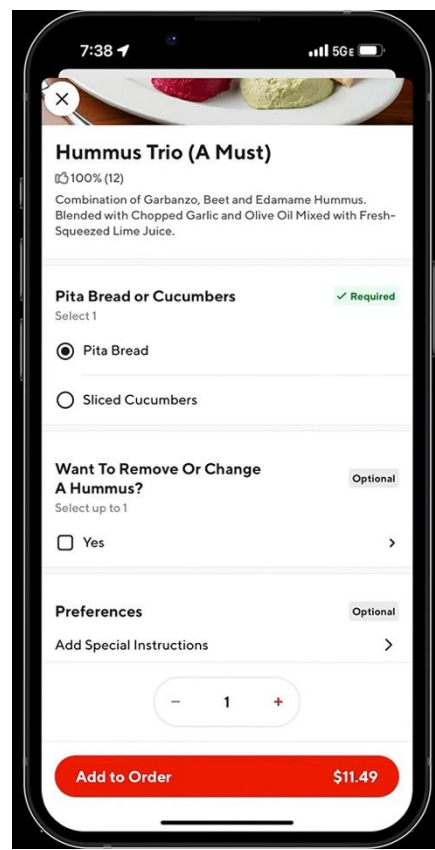
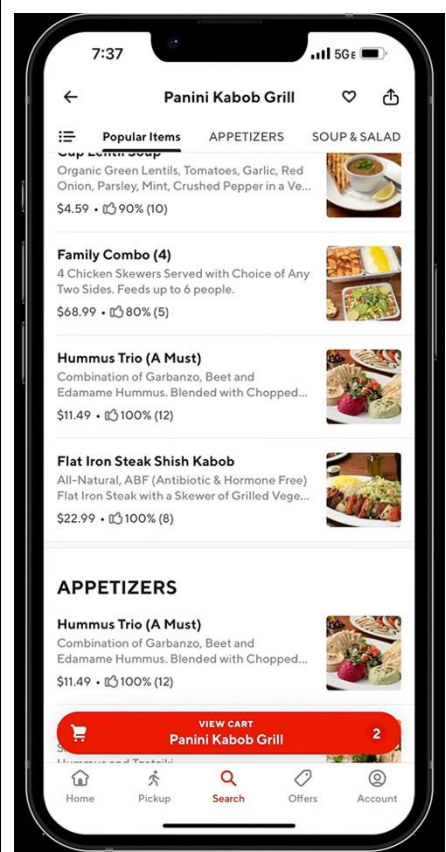
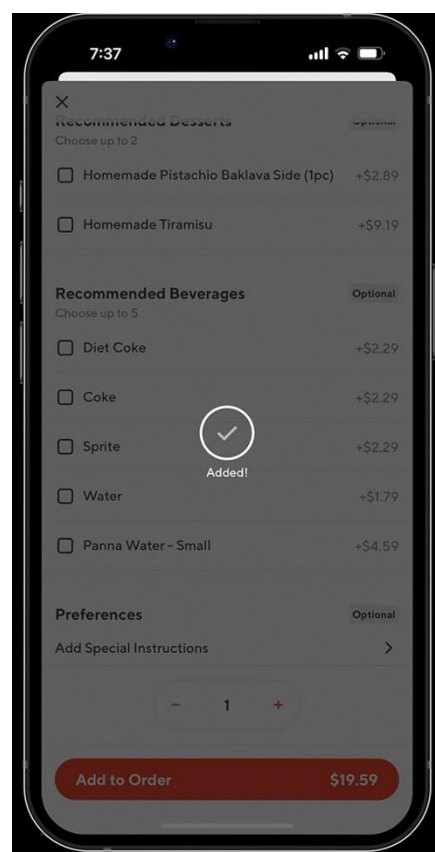


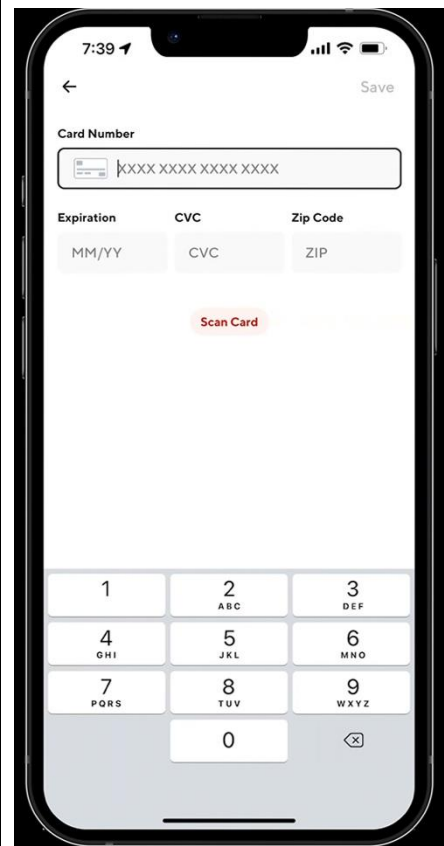
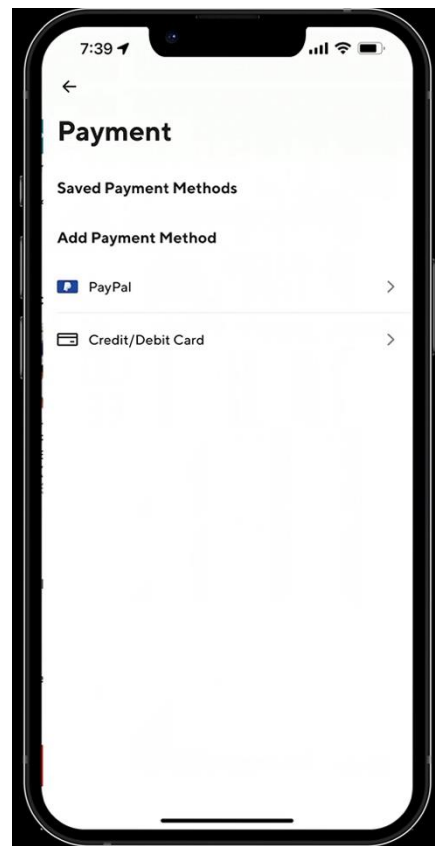
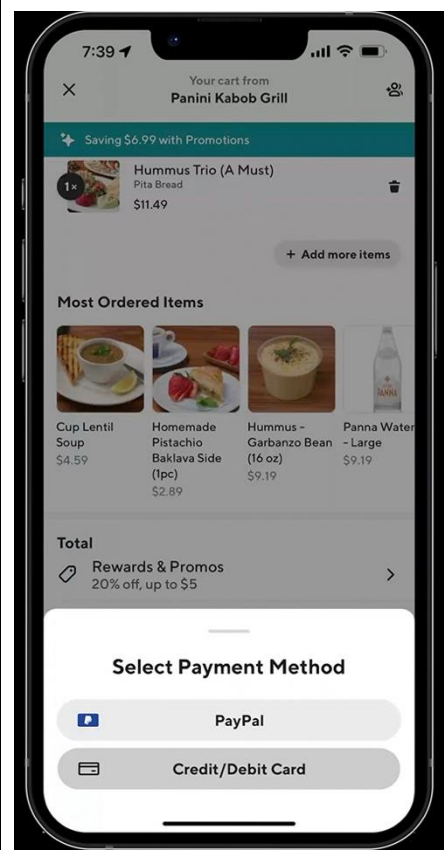
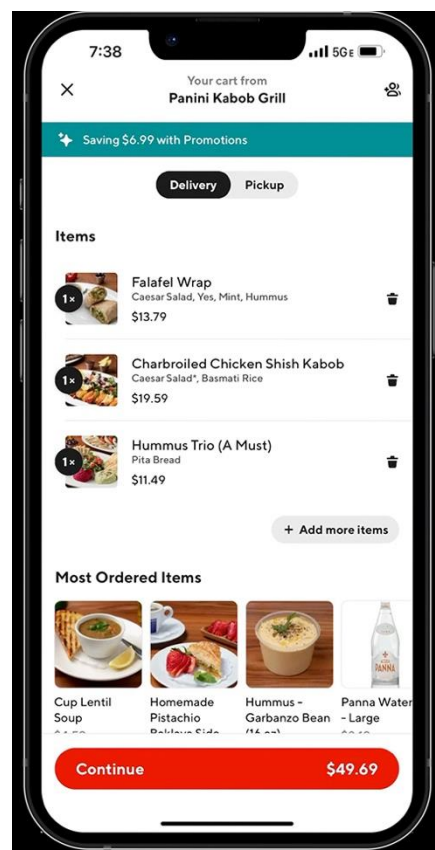


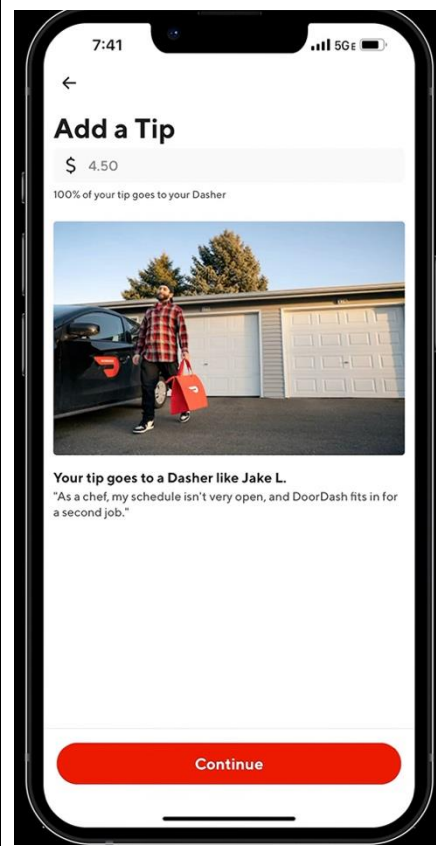
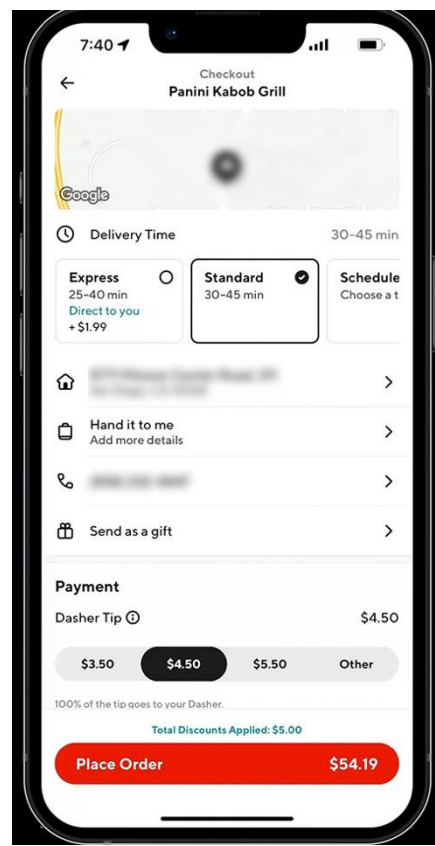
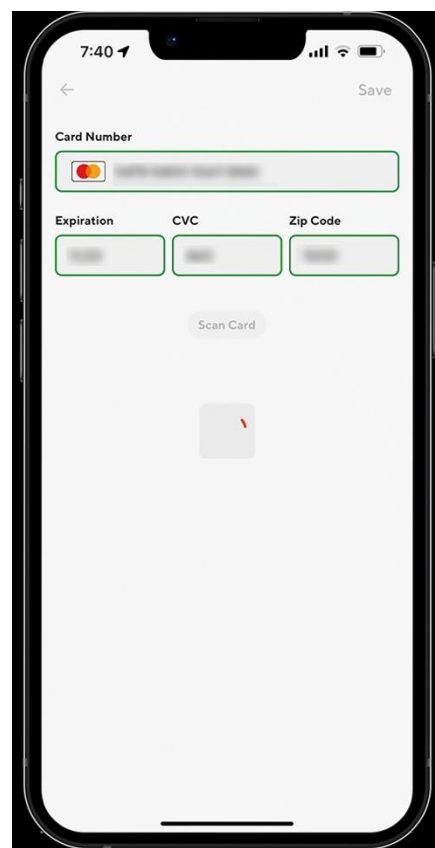


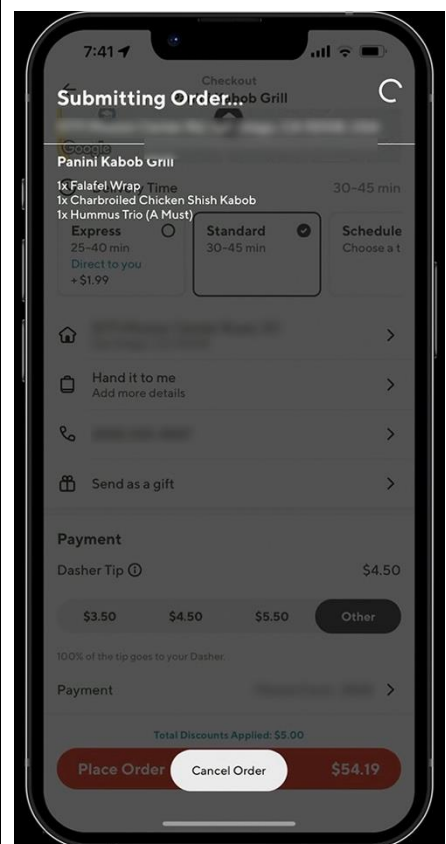
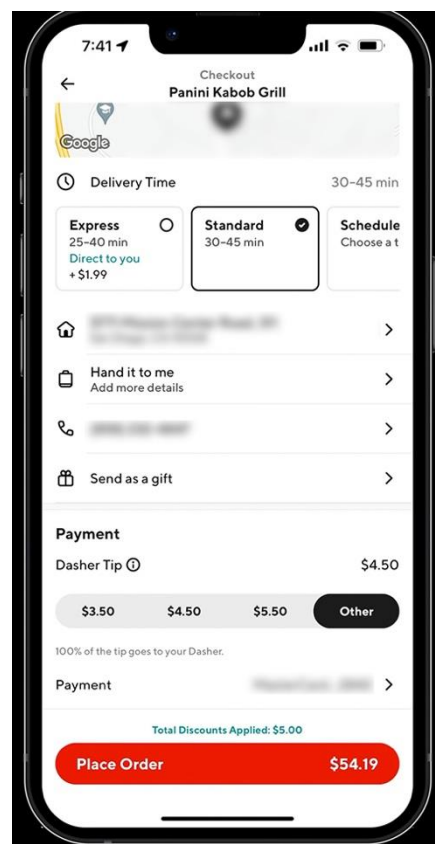


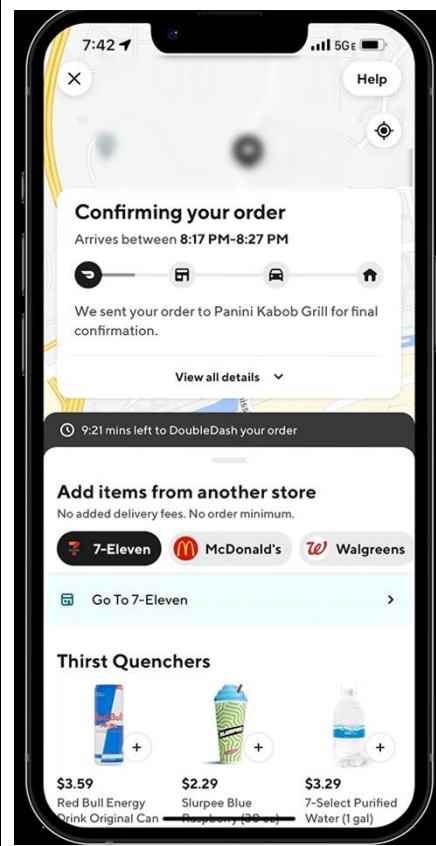
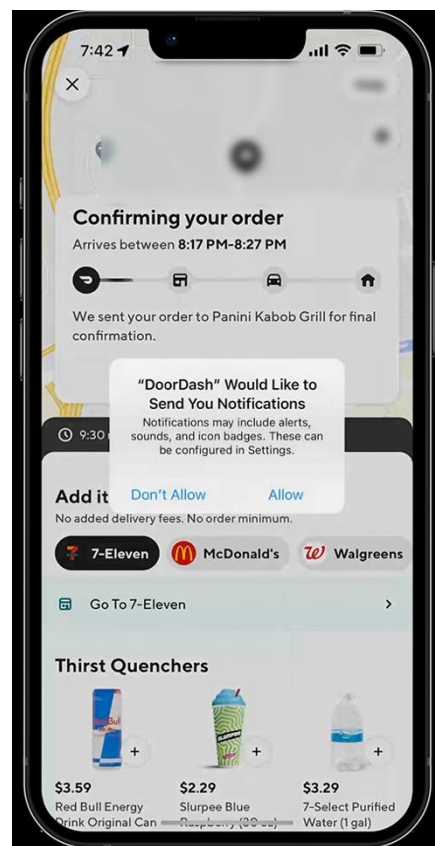
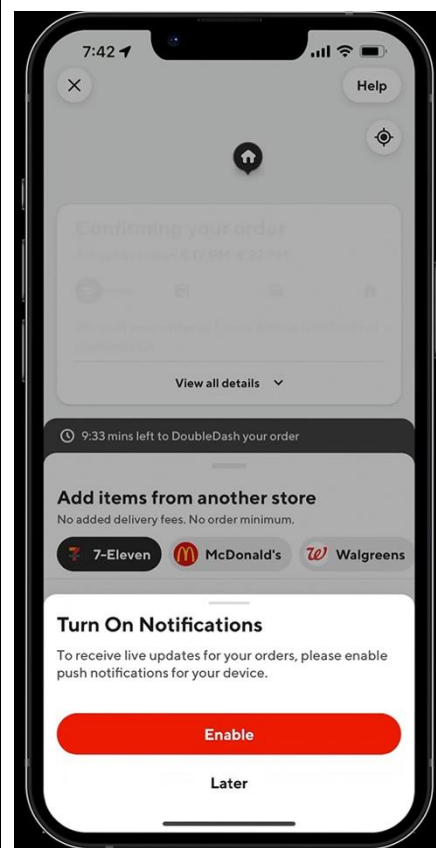
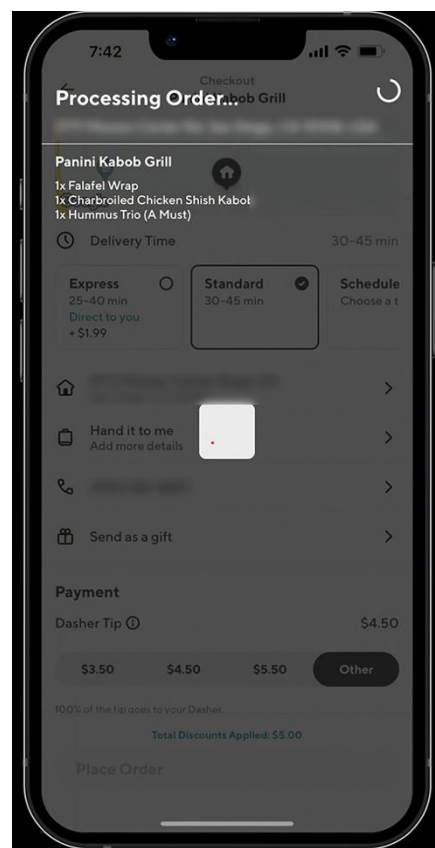


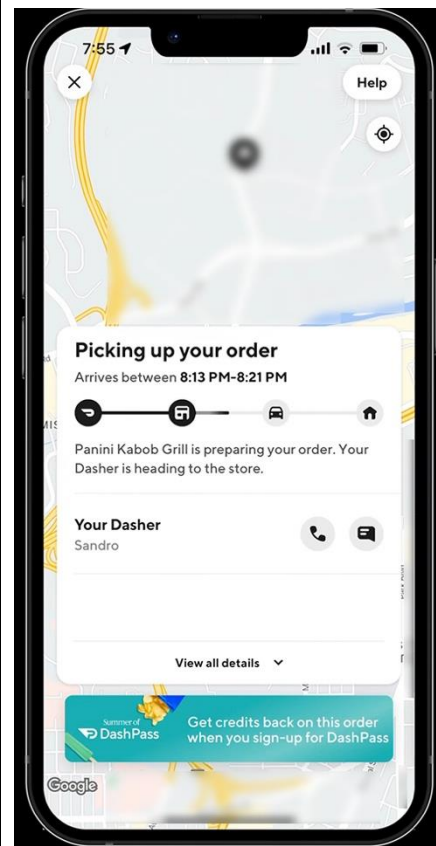
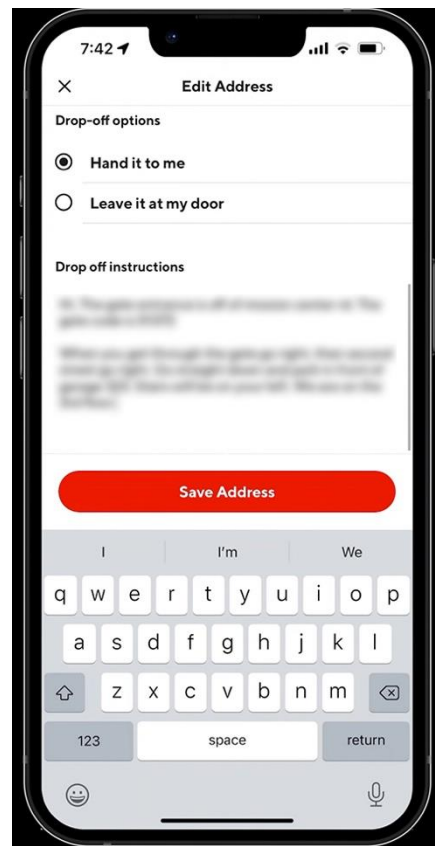
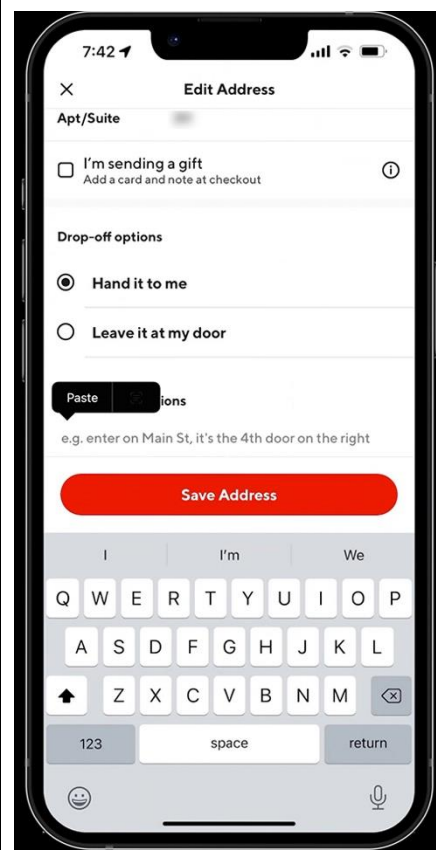
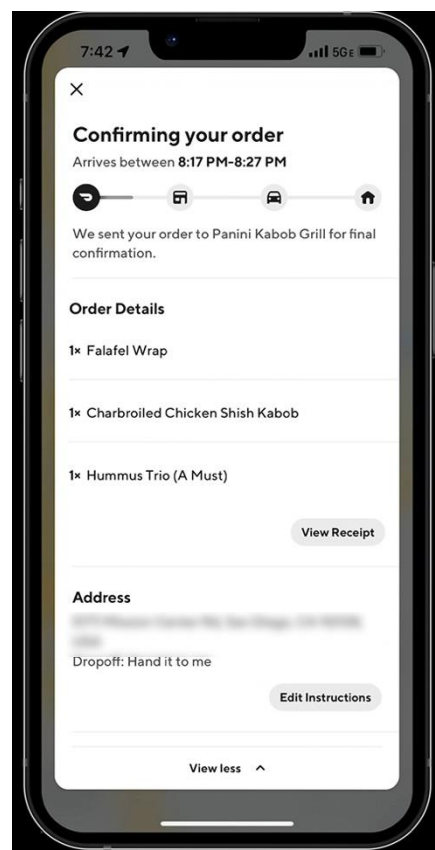


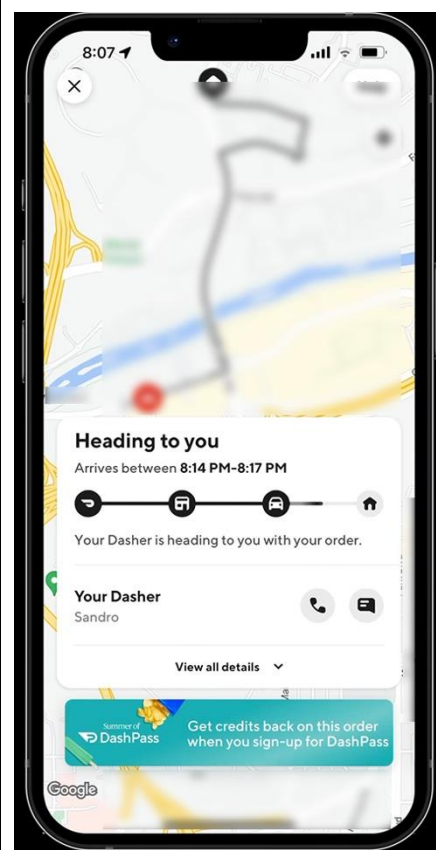
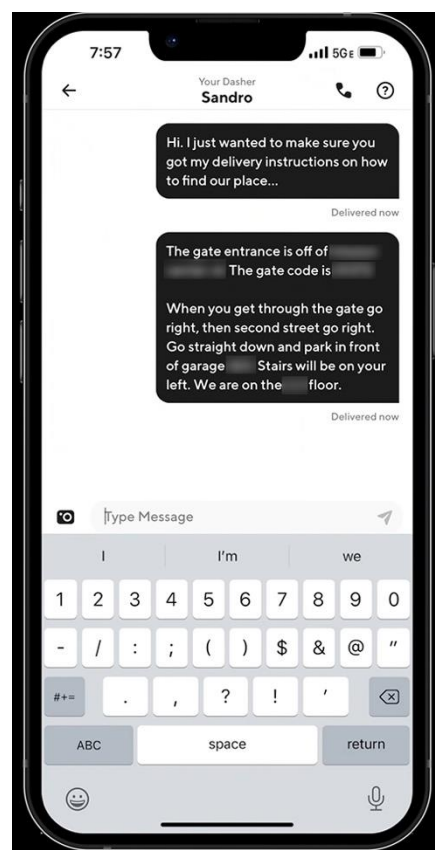
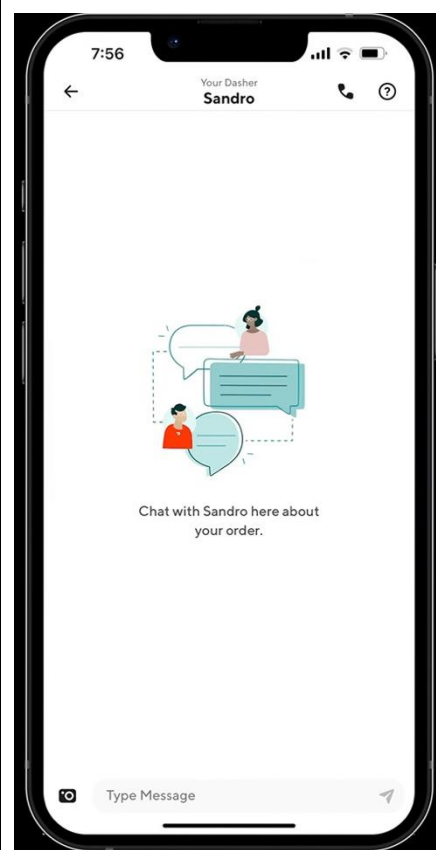
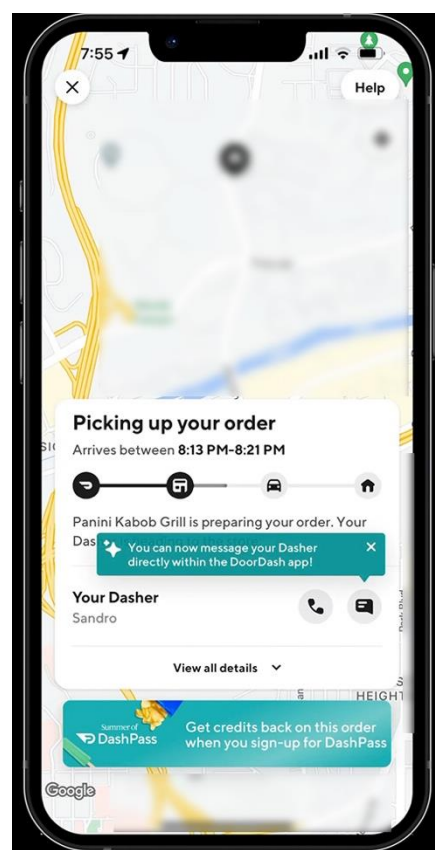


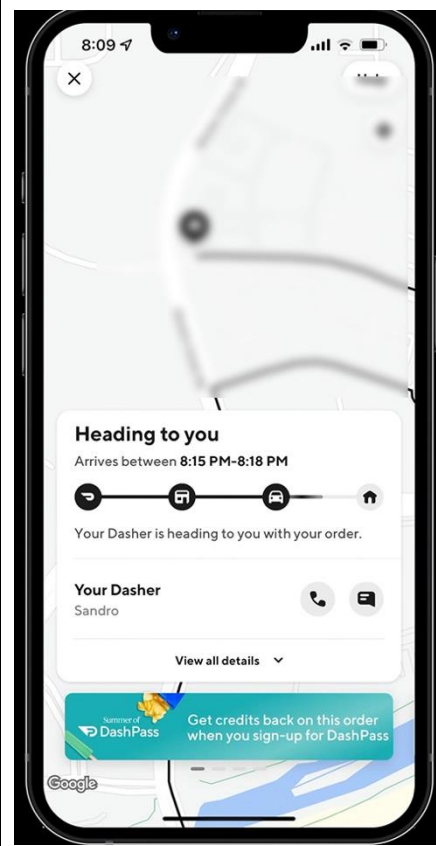
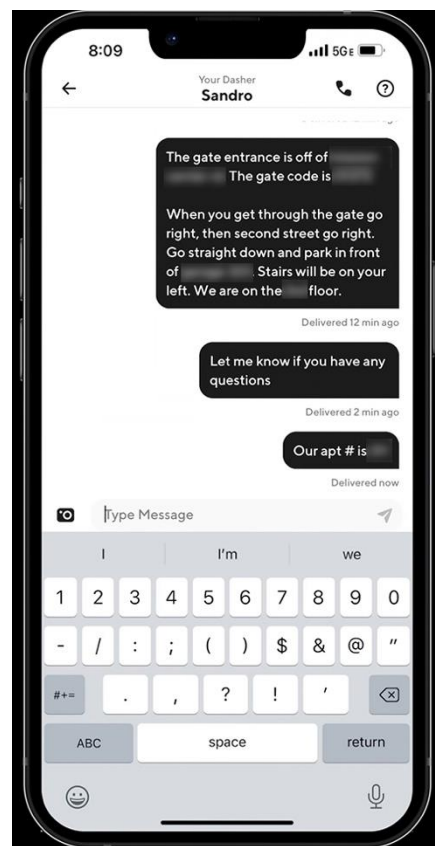
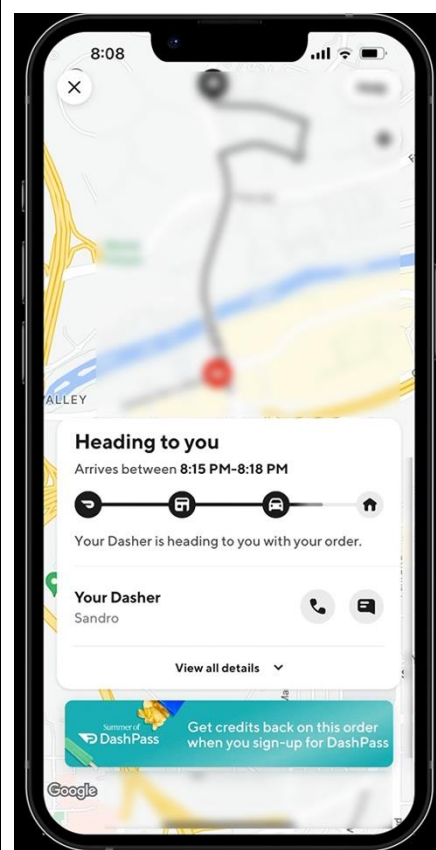
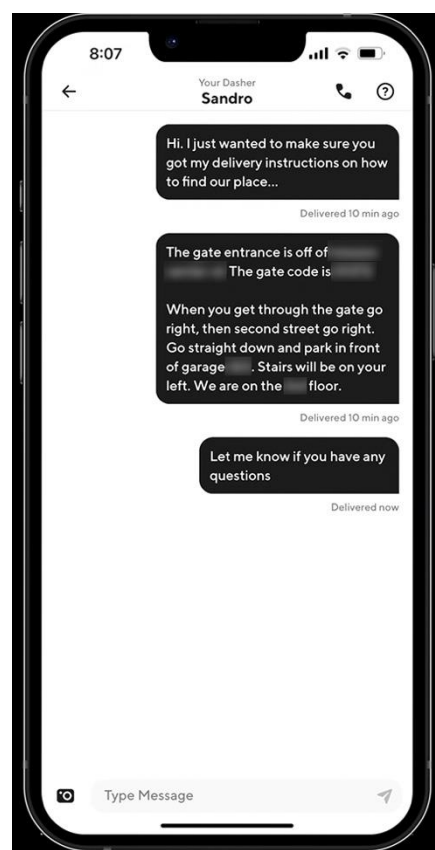


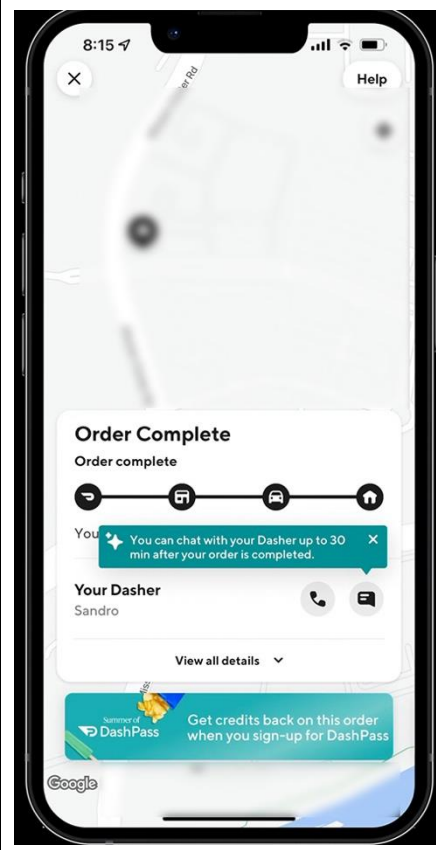
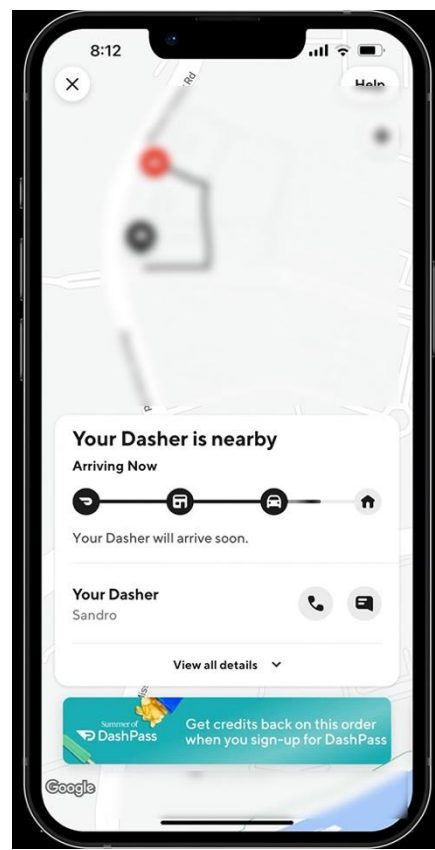
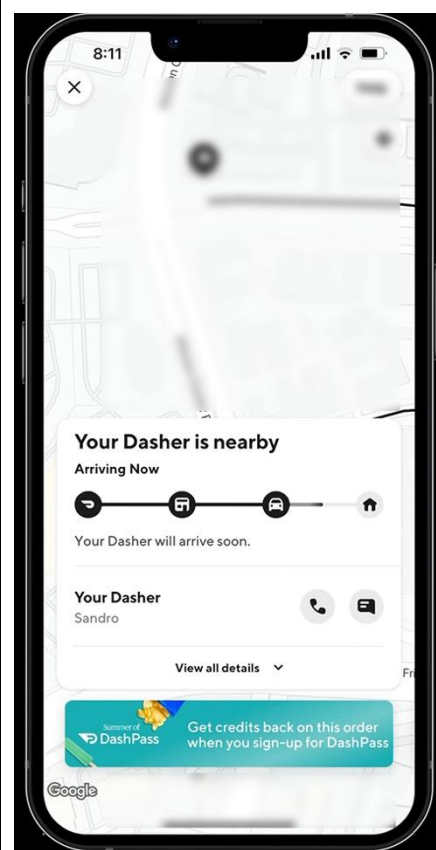






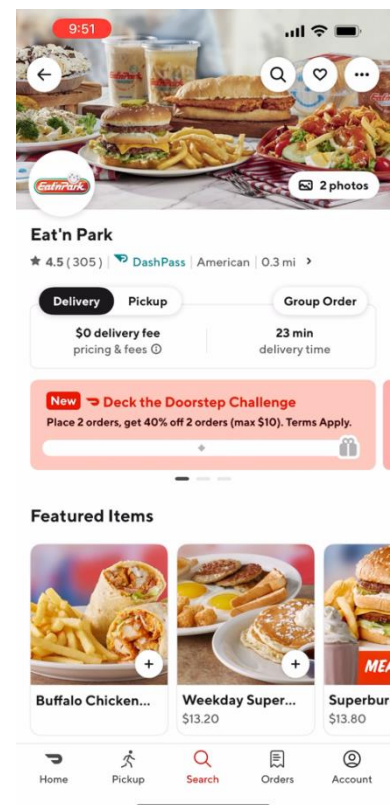
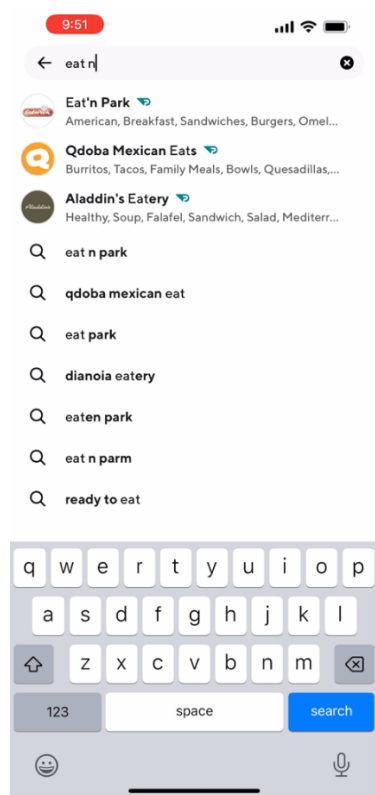
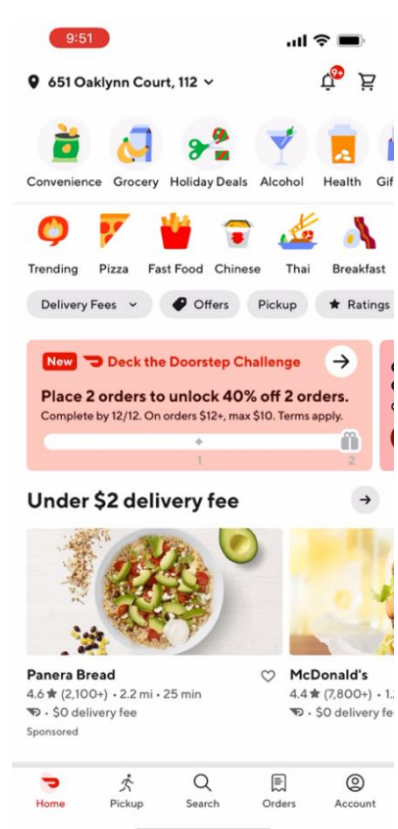


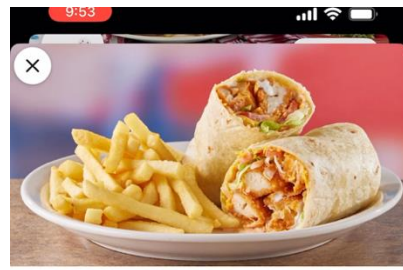




Below are screenshots of an DoorDash order for a customer located in Pittsburgh PA,

ordering from Eat 'N Park on the iOS app.





Buffalo Chicken Wrap

350 cal

Wrap-tastic! We load a flour tortilla with Buffalo chicken, cheddar cheese, lettuce, tomato, and your choice of homemade Ranch or bleu cheese dressing.

Select popular options for your order

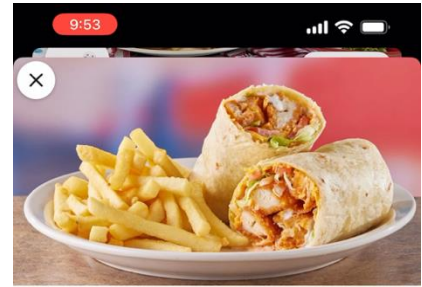
#1 • Ordered recently by 20+ others
 1 Side • Crispy Chicken Tenders • Ranch • French Fries
 \$13.80

Select Your Buffalo Chicken Wrap

Required • Select at least 1

2 Sides
 350 cal +\$16.30 >

Make 1 required selection \$0.00



Buffalo Chicken Wrap

350 cal

Wrap-tastic! We load a flour tortilla with Buffalo chicken, cheddar cheese, lettuce, tomato, and your choice of homemade Ranch or bleu cheese dressing.

Select popular options for your order

#1 • Ordered recently by 20+ others
 1 Side • Crispy Chicken Tenders • Ranch • French Fries
 \$13.80

Select Your Buffalo Chicken Wrap

Required • Select at least 1

2 Sides
 350 cal +\$16.30 >

Make 1 required selection \$0.00



1 Side

Select Your Protein:

Required • Select 1

Crispy Chicken Tenders
 470 cal

Grilled Chicken Breast
 180 cal

Select Your Dressing For Your Wrap:

Required • Select 1

Ranch
 200 cal

Bleu Cheese
 300 cal

No Dressing

Choose Your Side:

Required • Select 1

Fried Cheese Sticks with Marinara
 295 cal +\$1.40

Fried Cheese Sticks with Ranch
 545 cal +\$1.40

Make 3 required selections



1 Side

Select Your Protein:

Required • Select 1

Crispy Chicken Tenders
 470 cal

Grilled Chicken Breast
 180 cal

Select Your Dressing For Your Wrap:

Required • Select 1

Ranch
 200 cal

Bleu Cheese
 300 cal

No Dressing

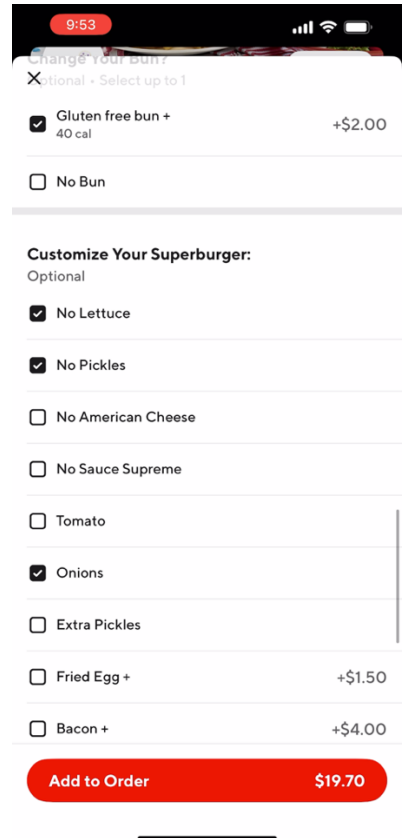
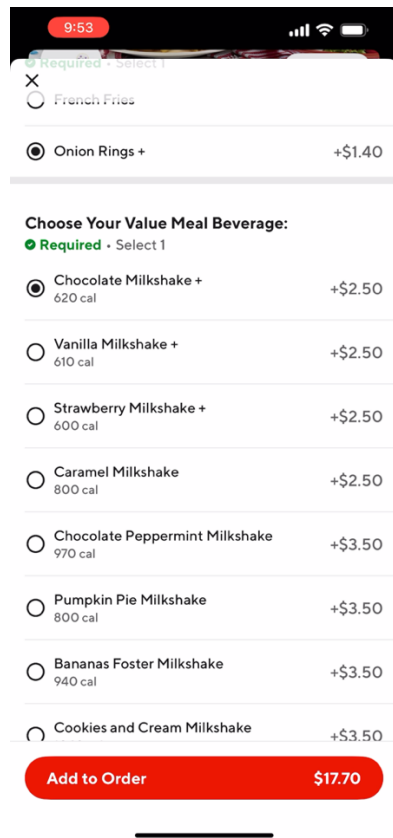
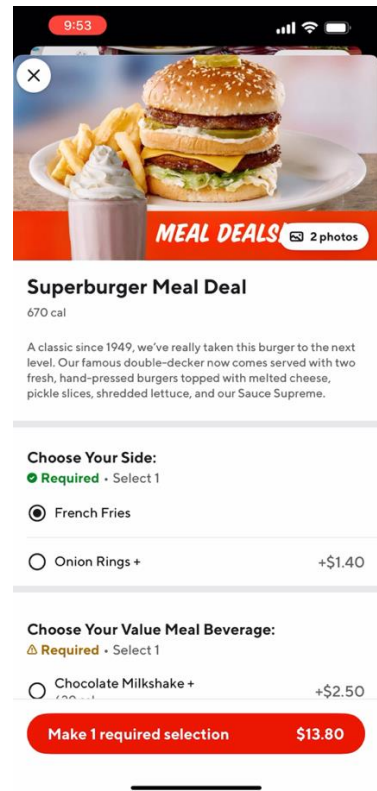
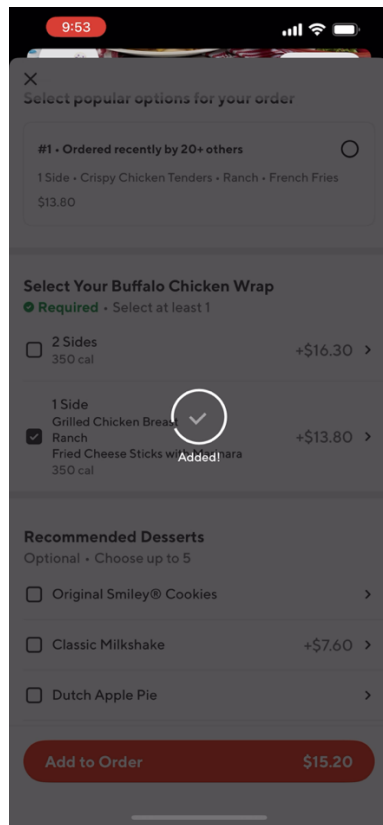
Choose Your Side:

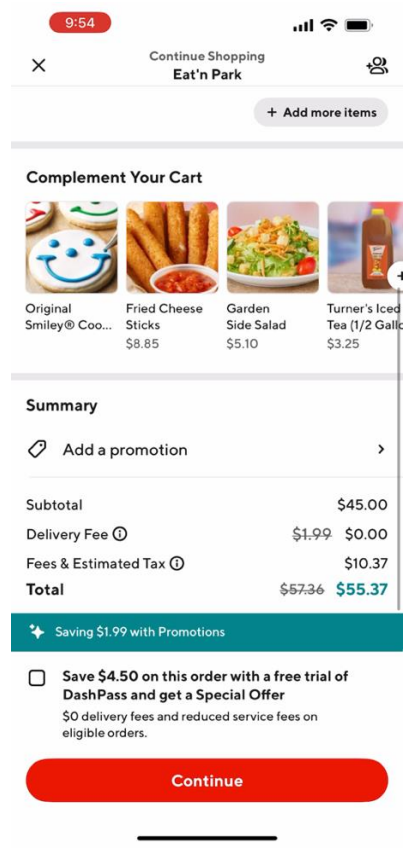
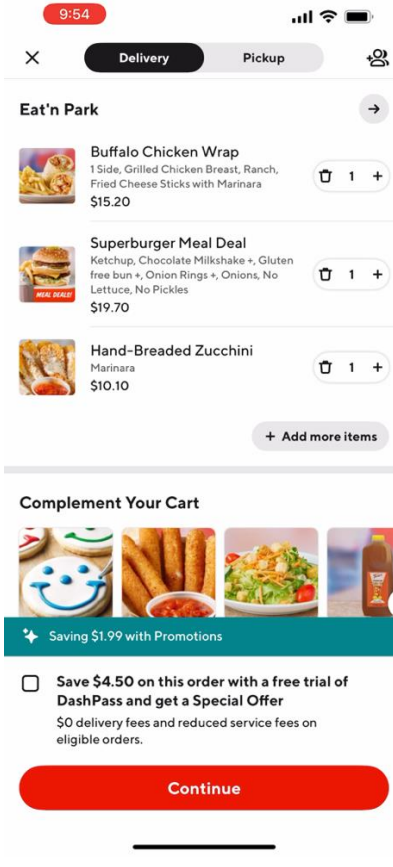
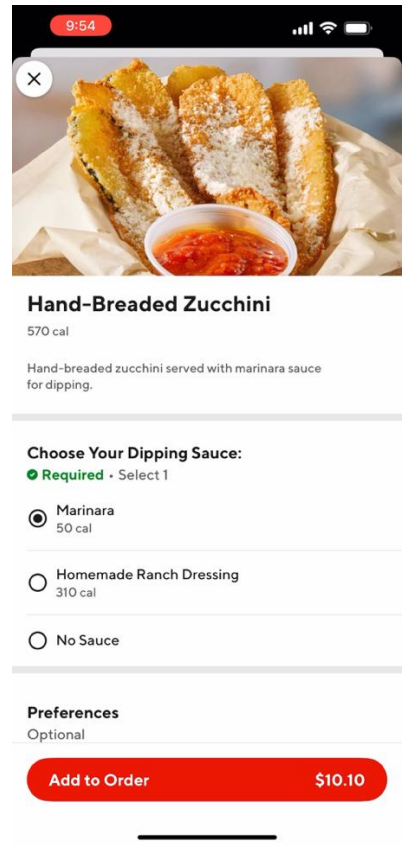
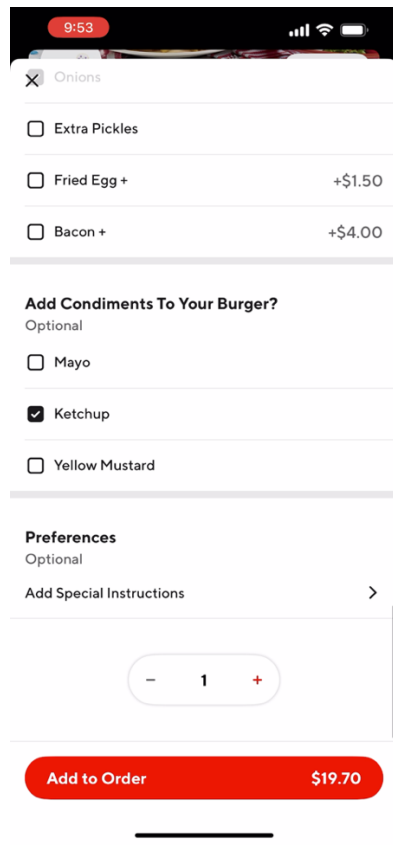
Required • Select 1

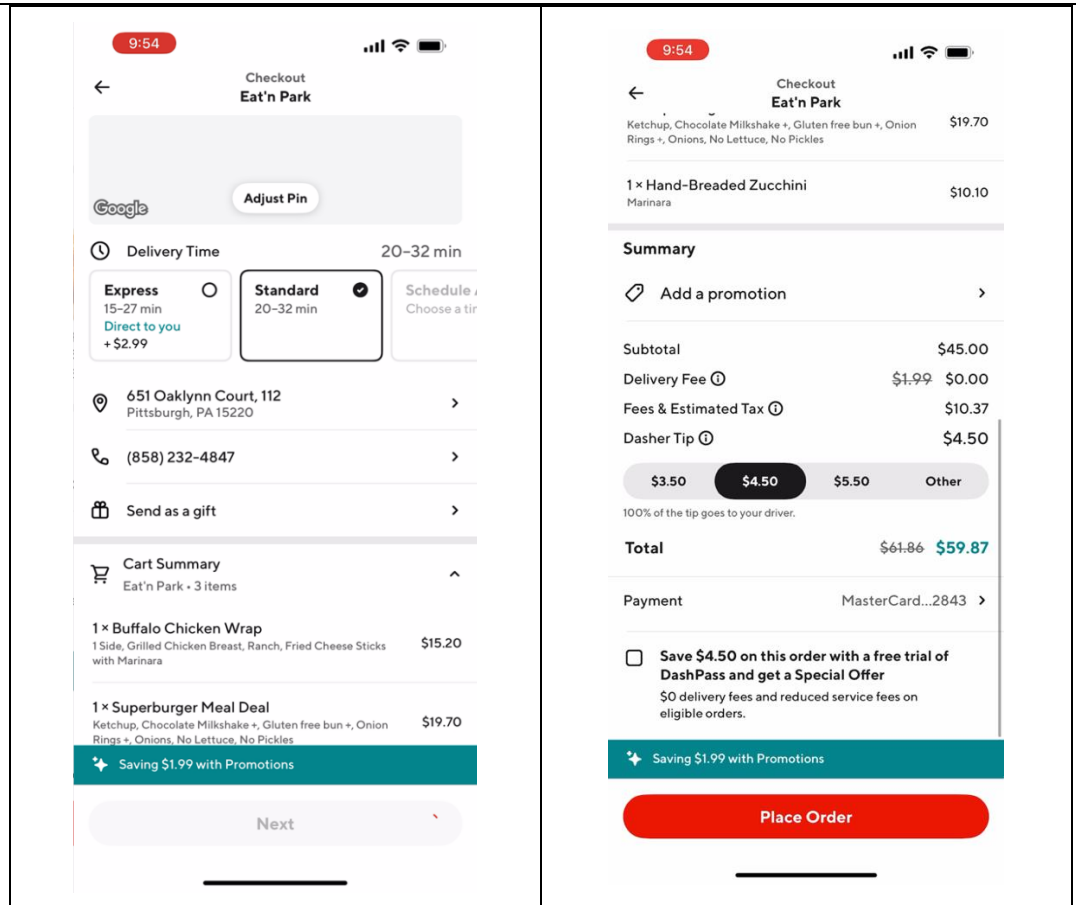
Fried Cheese Sticks with Marinara
 295 cal +\$1.40

Fried Cheese Sticks with Ranch
 545 cal +\$1.40

Make 1 required selection

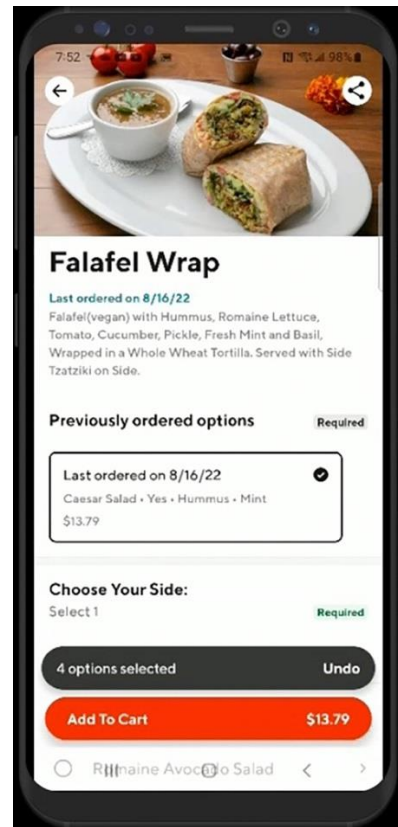
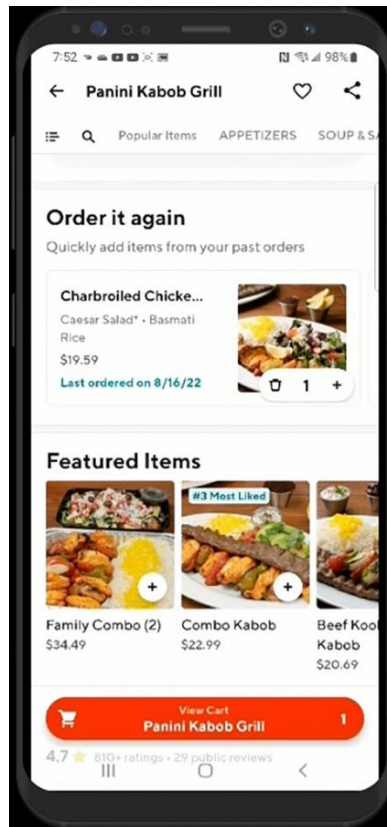
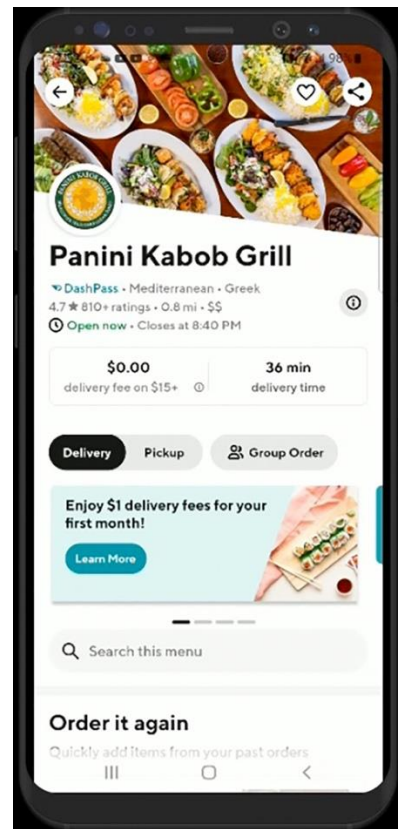


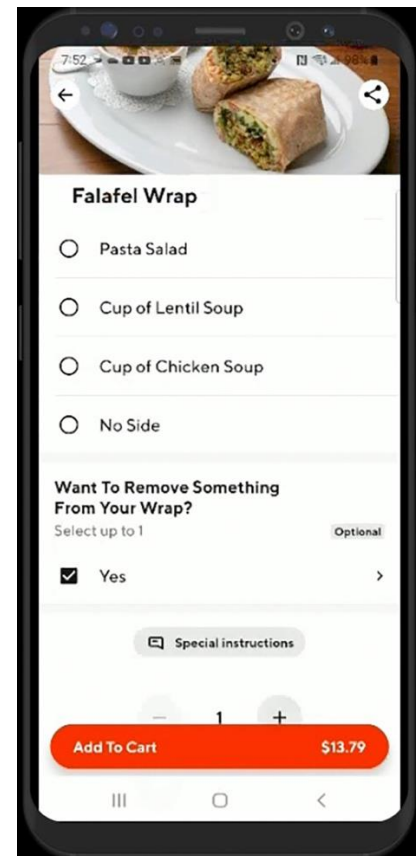
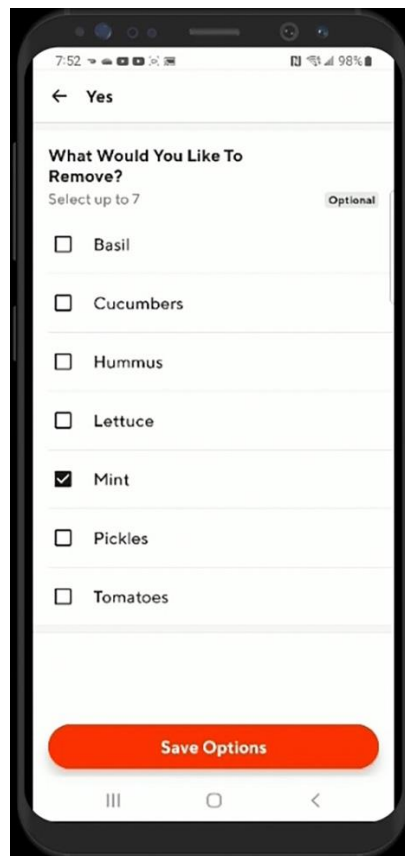
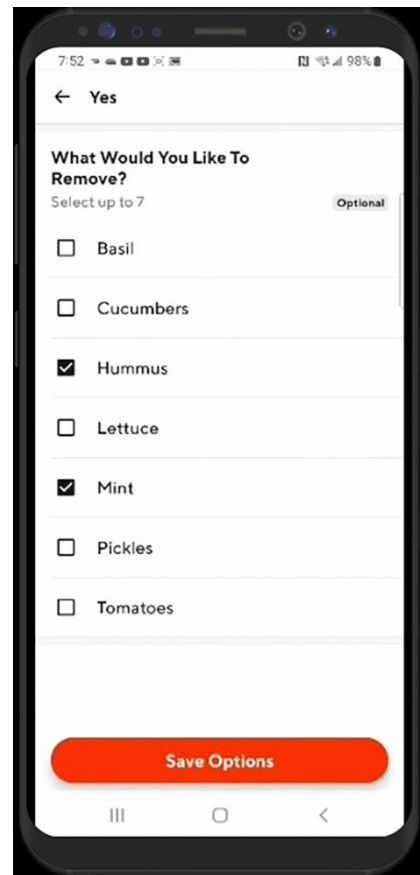
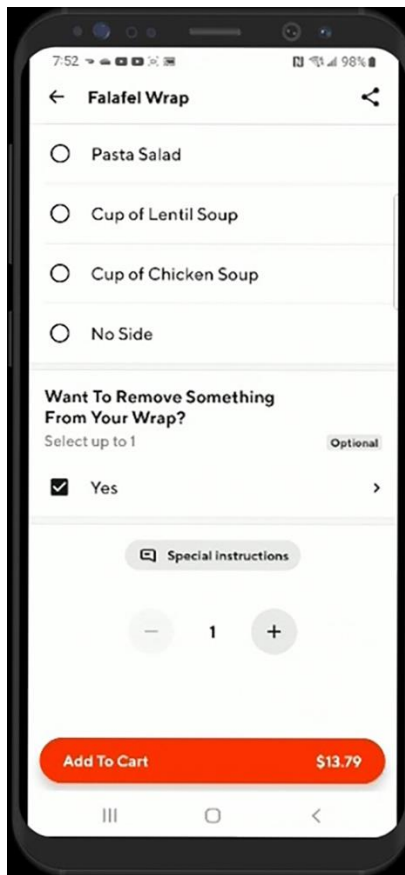


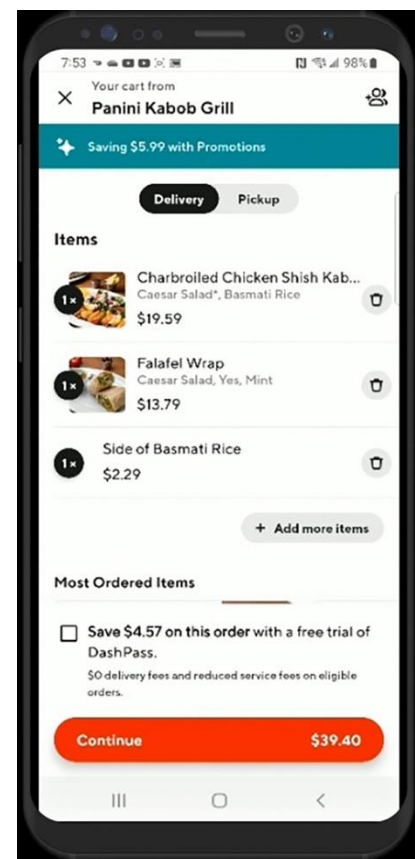
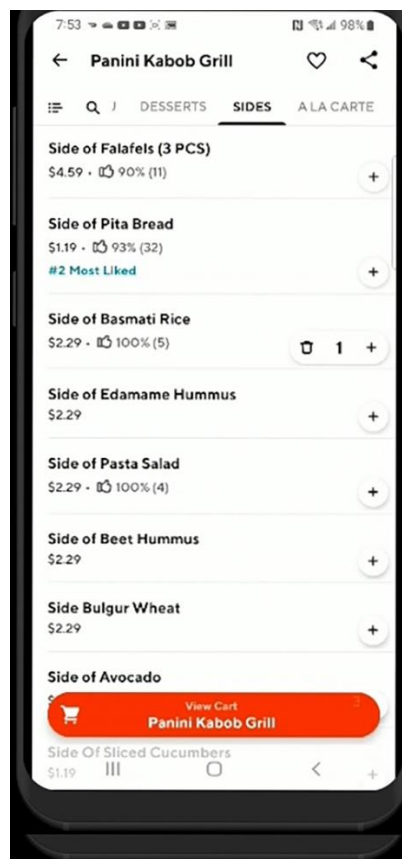
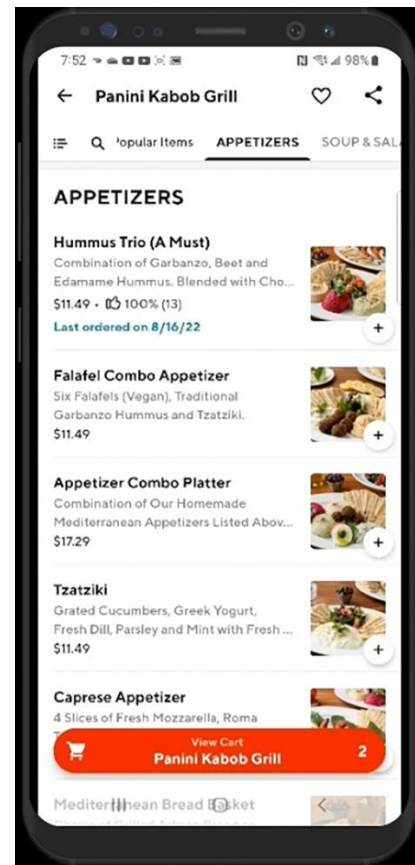
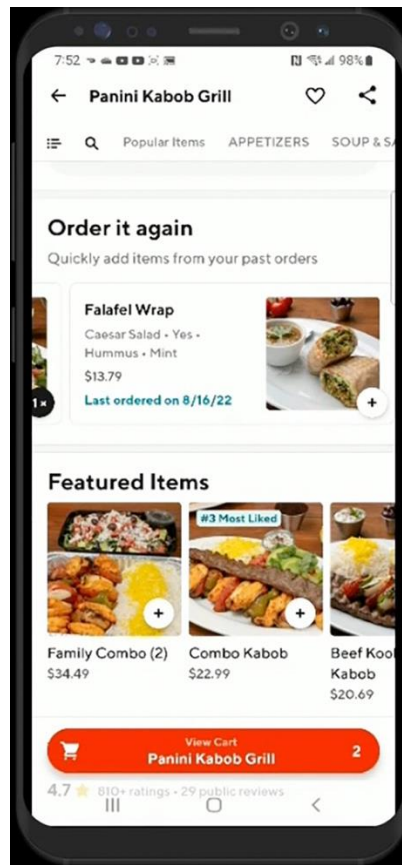


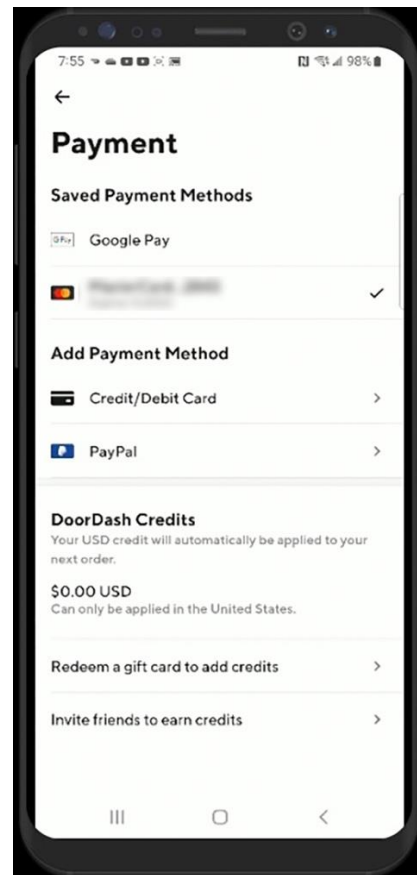
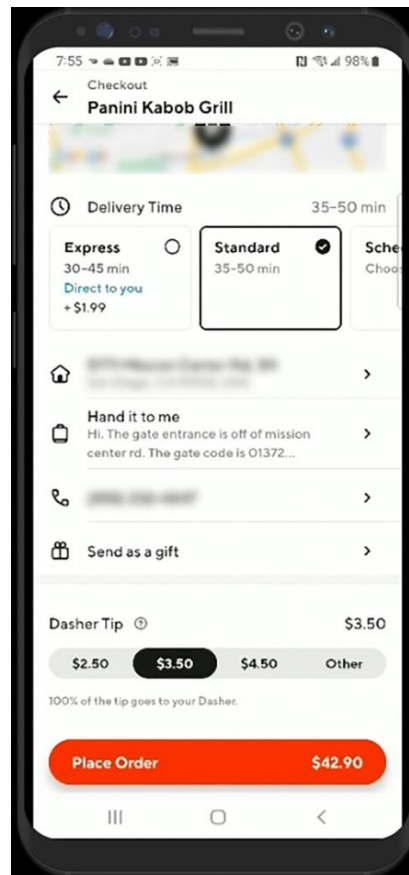
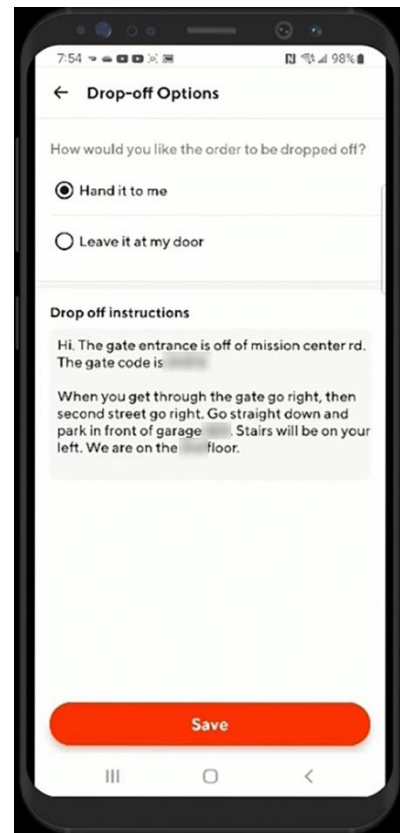
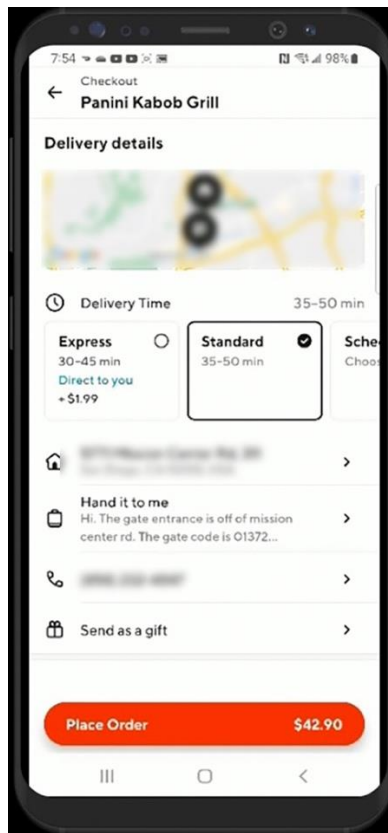
DoorDash Consumer App - Android

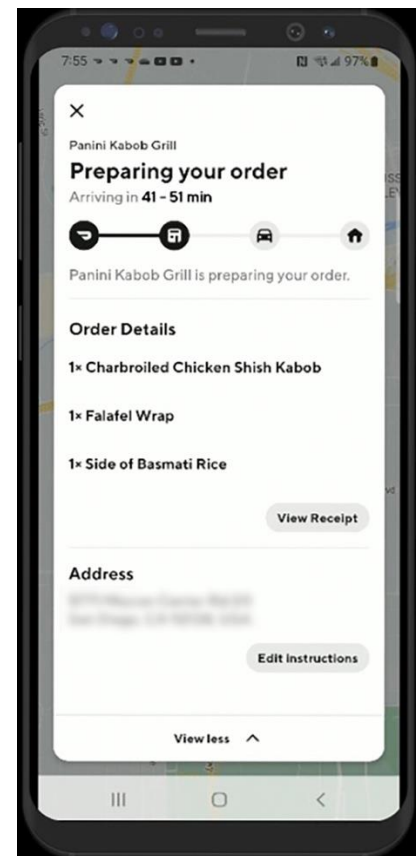
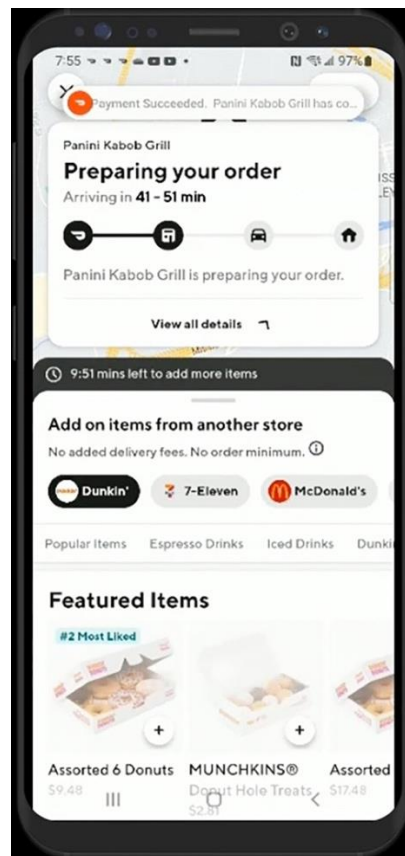
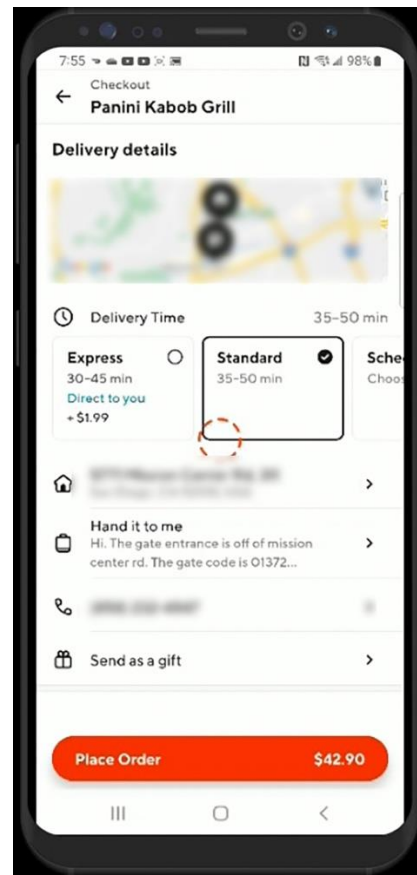
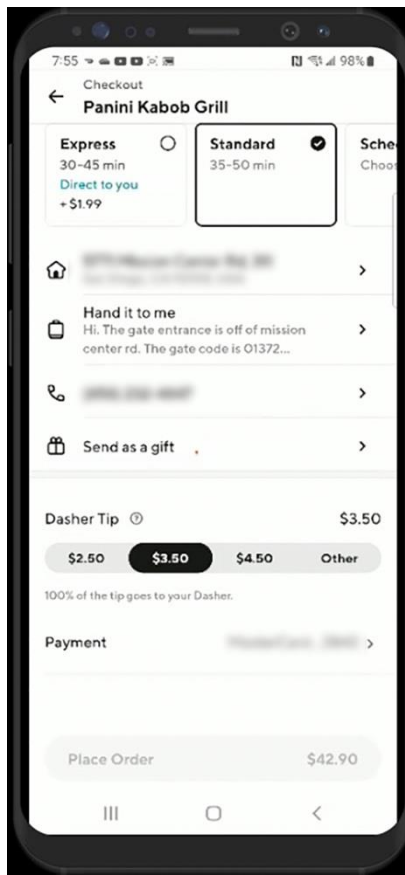
Below are screen shots of the ordering process on an Android device. This process demonstrates the user launching the DoorDash app.

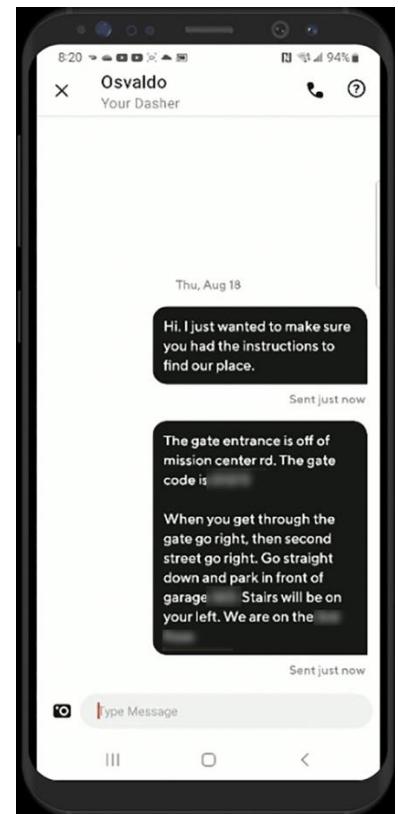
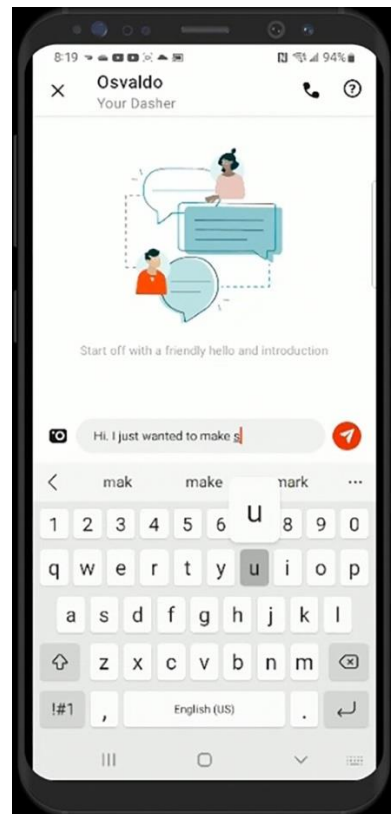
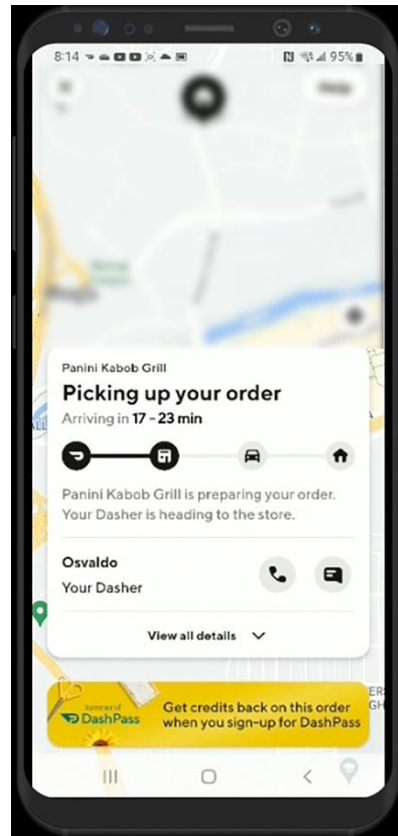
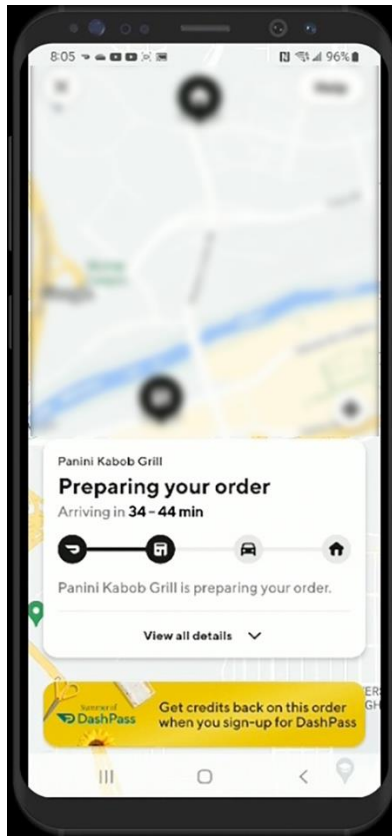


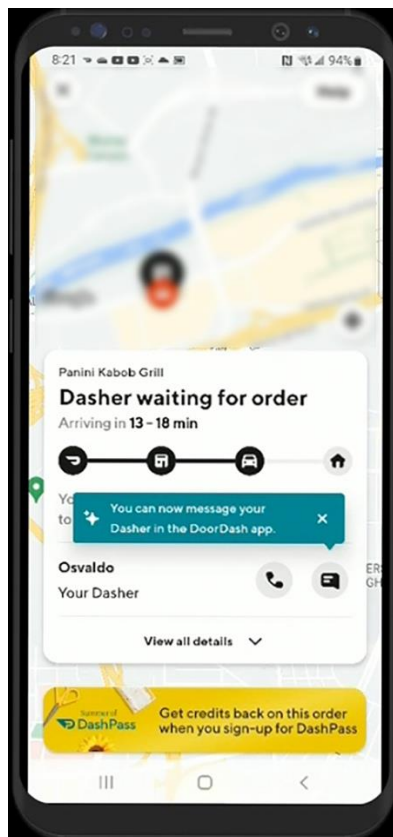
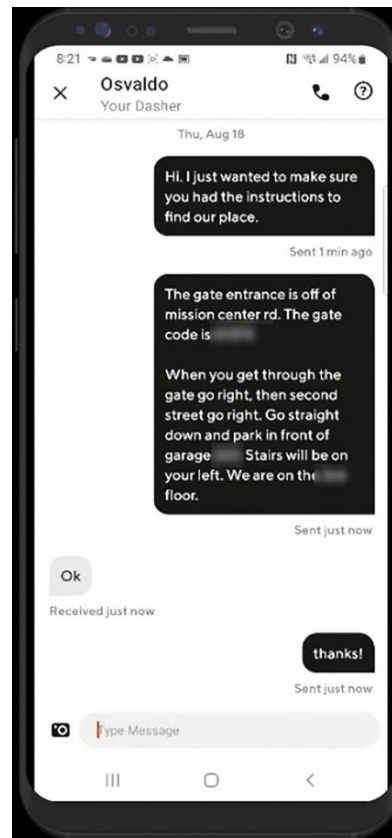
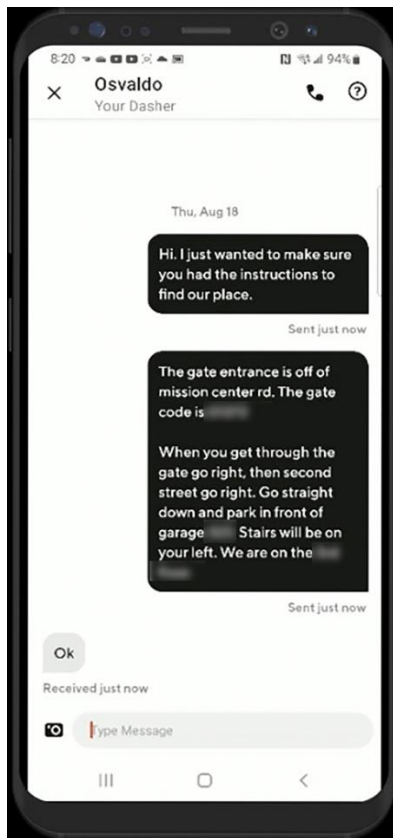


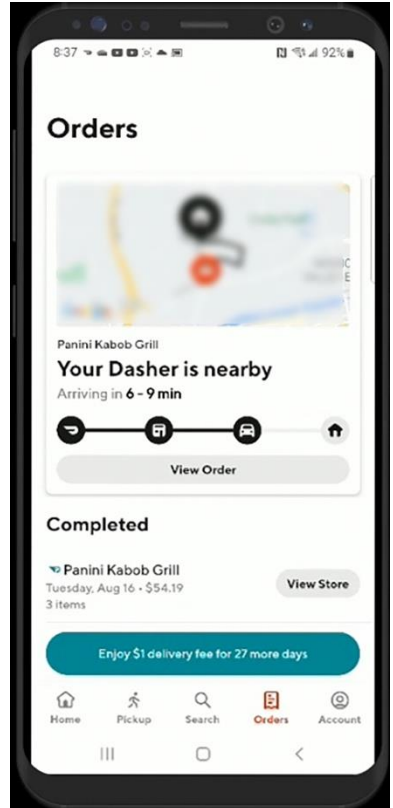
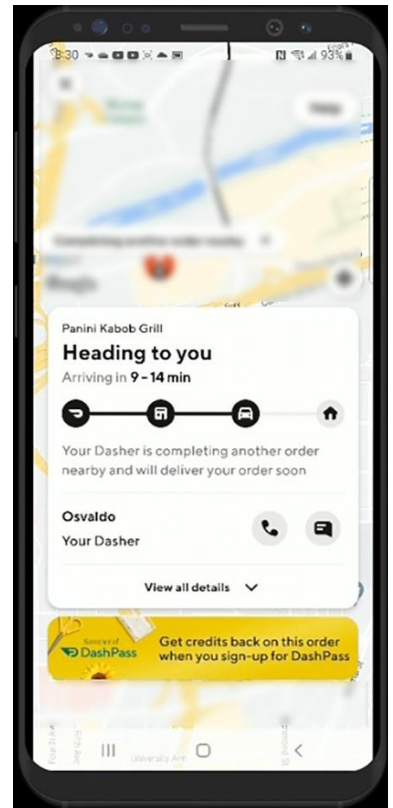


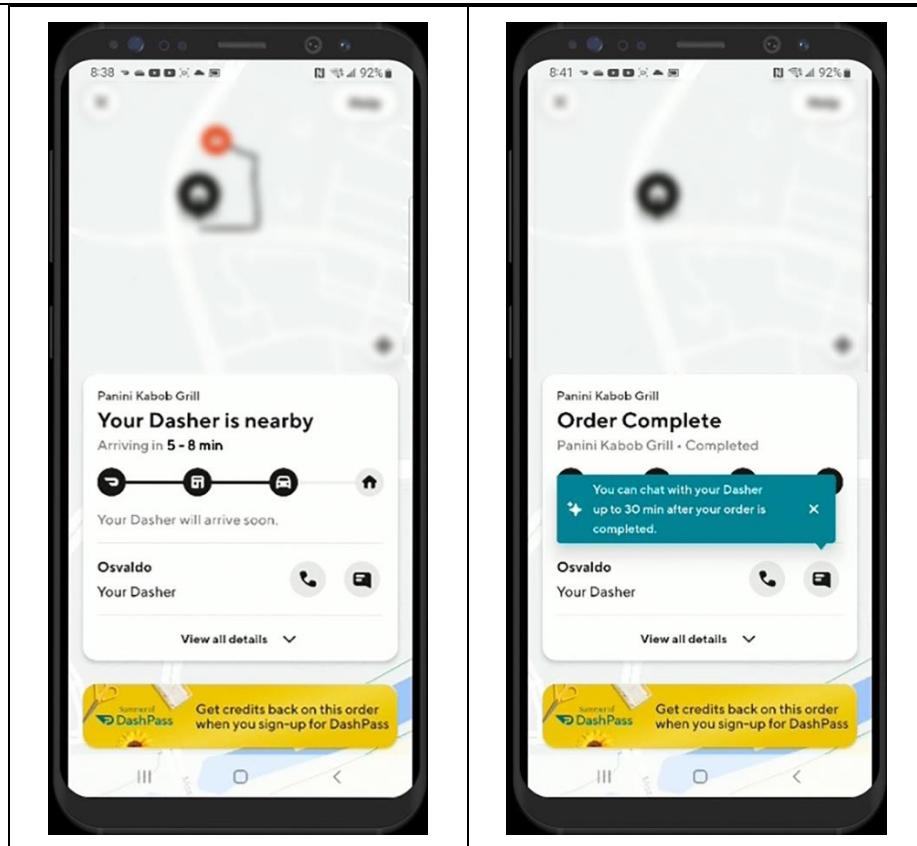






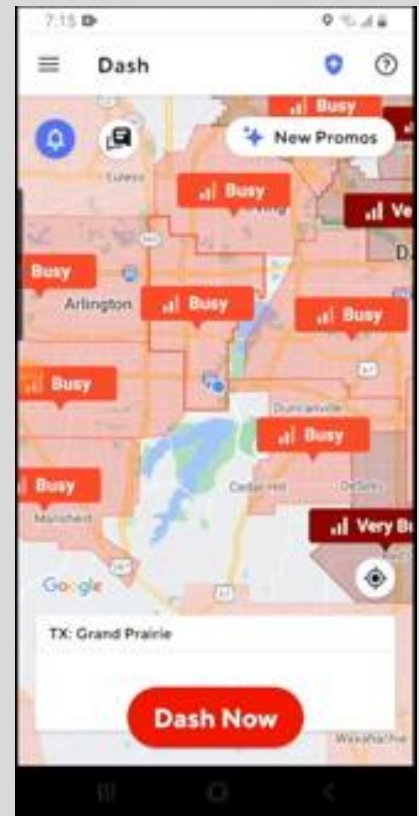
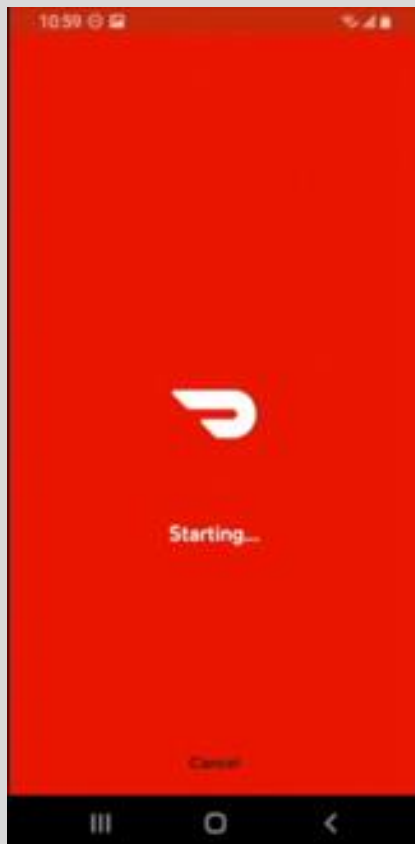


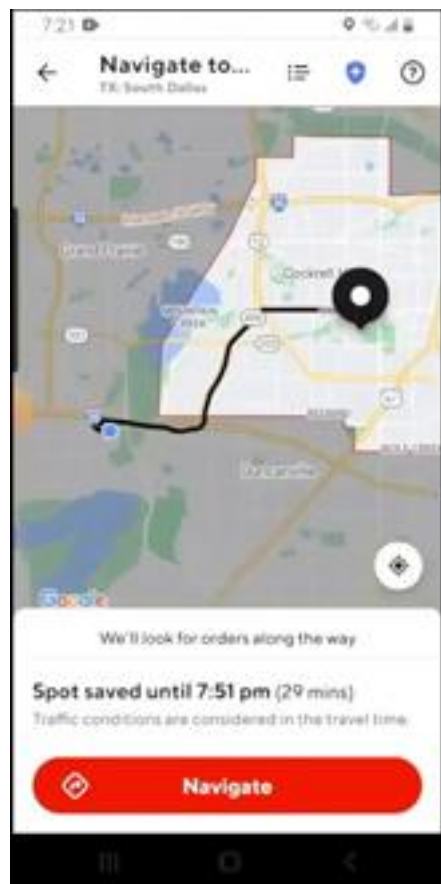
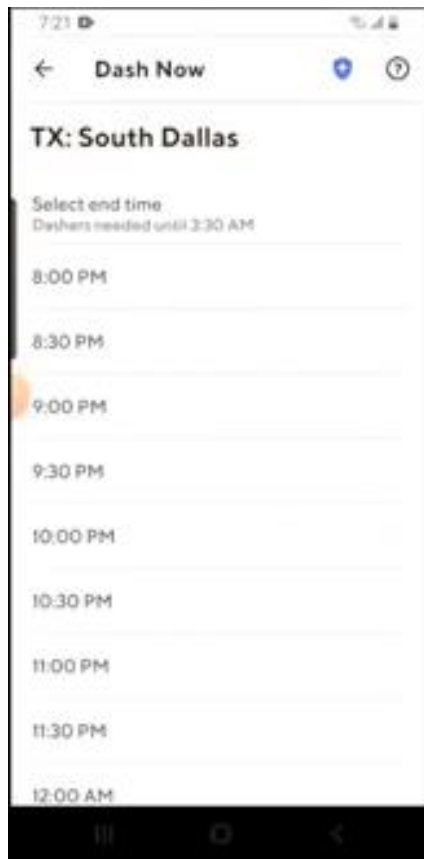


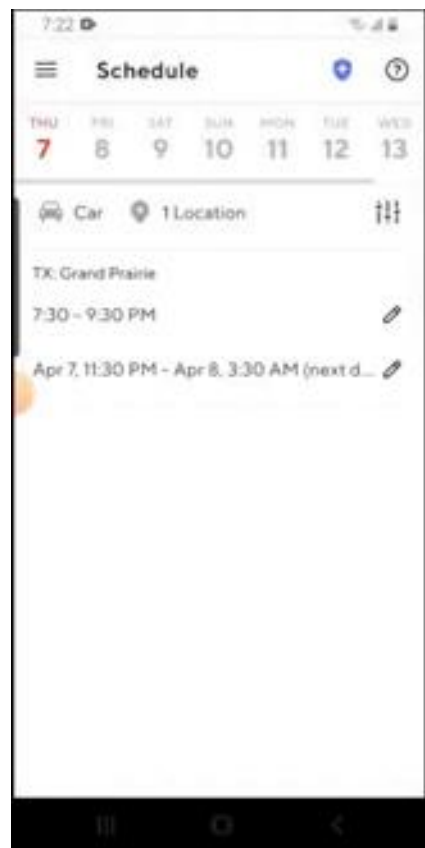
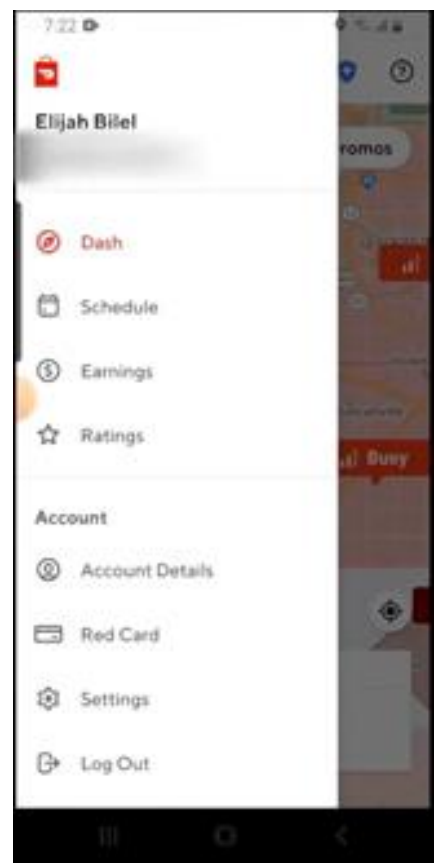


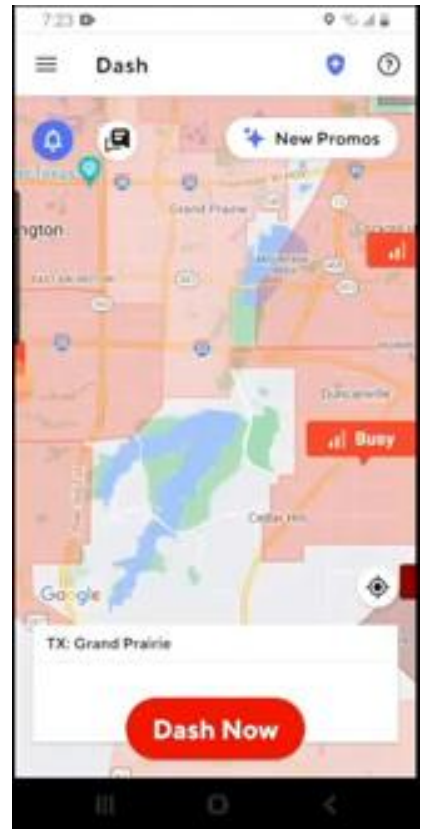
DoorDash Dasher App – Android

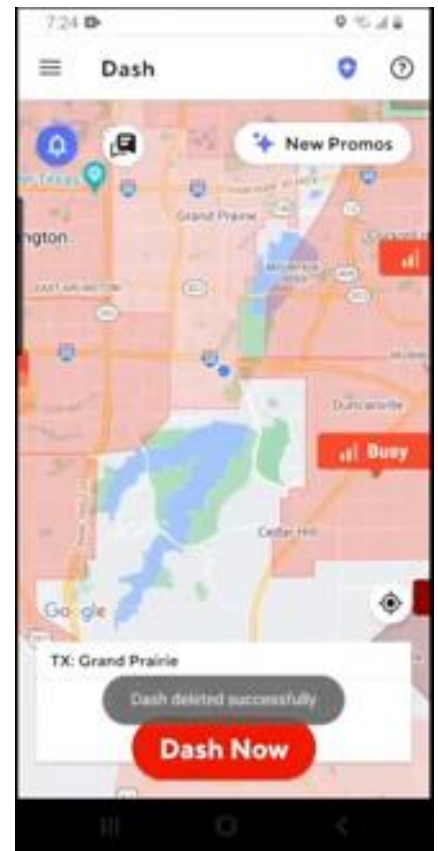
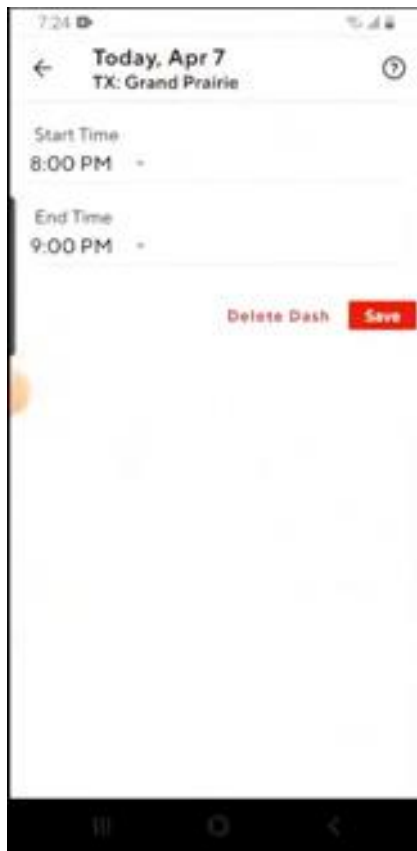
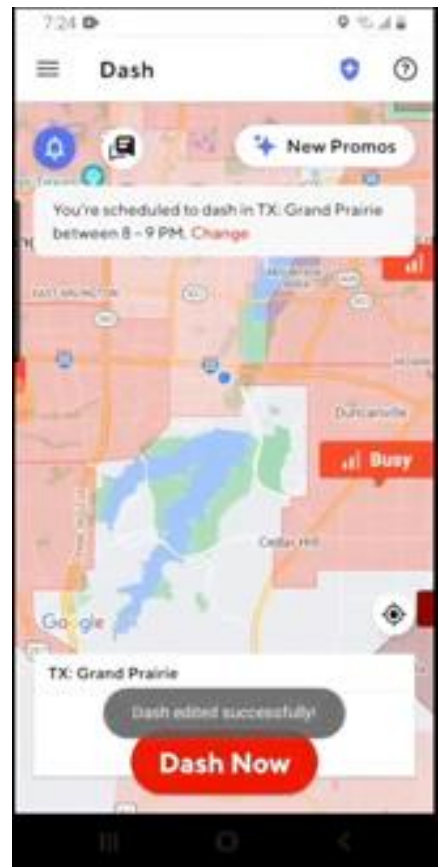
Below are screen shots of the Dasher App for Android. The Dasher app is the app that DoorDash drivers use to pick-up orders at restaurants and deliver orders to consumers (see Exh 86):

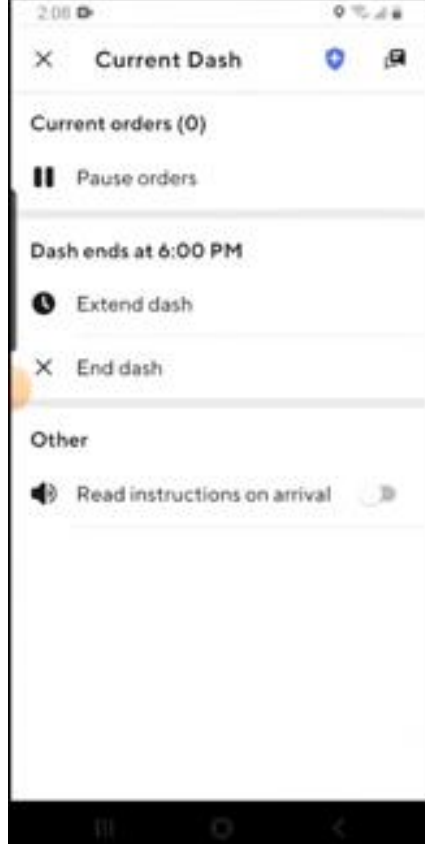
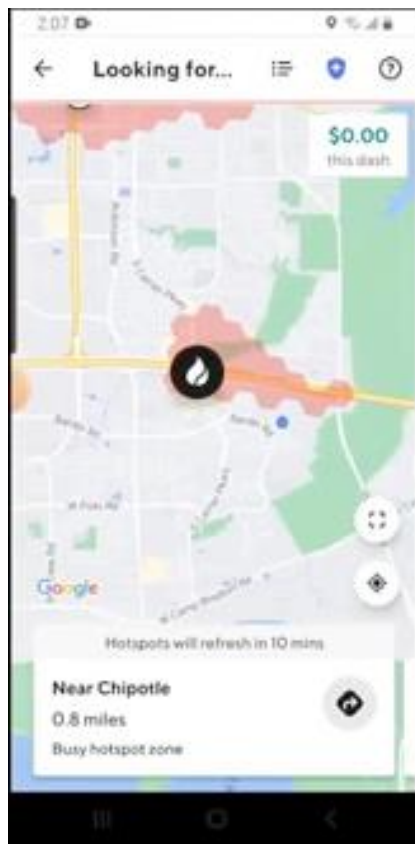
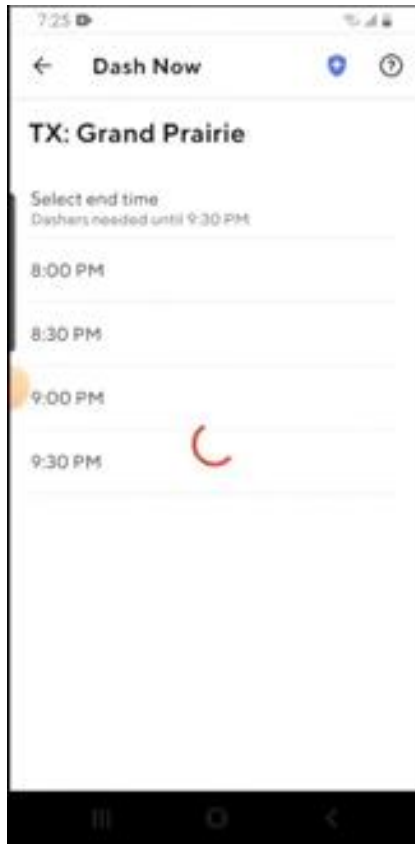


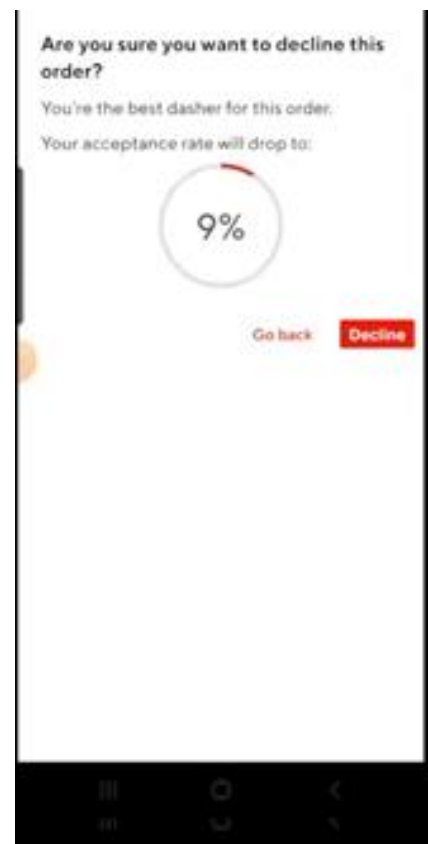
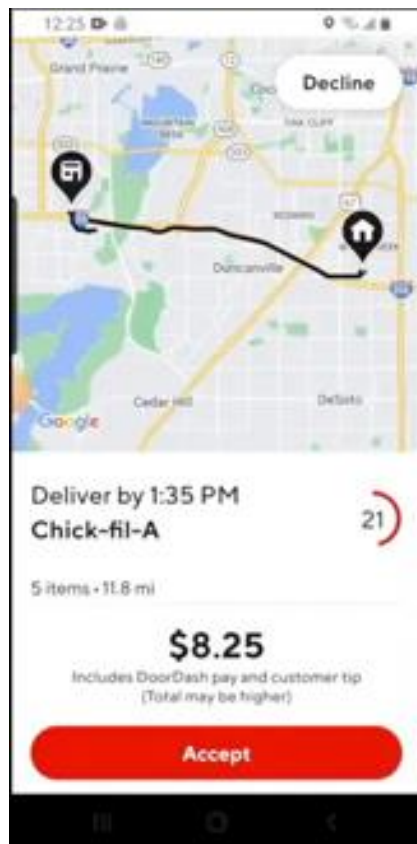
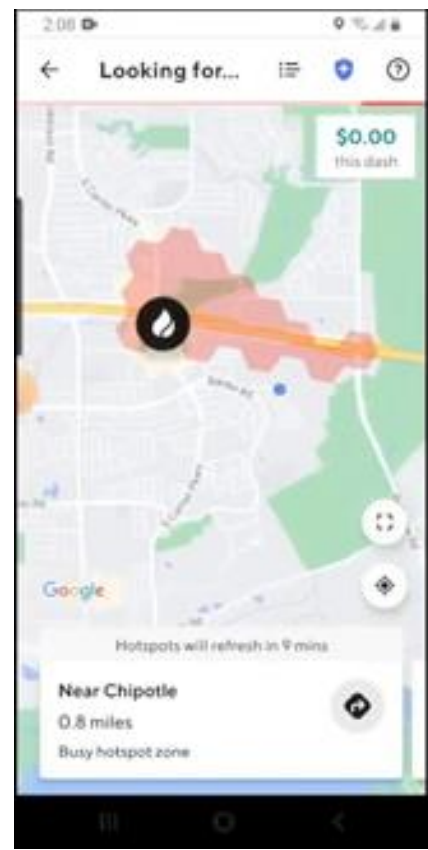
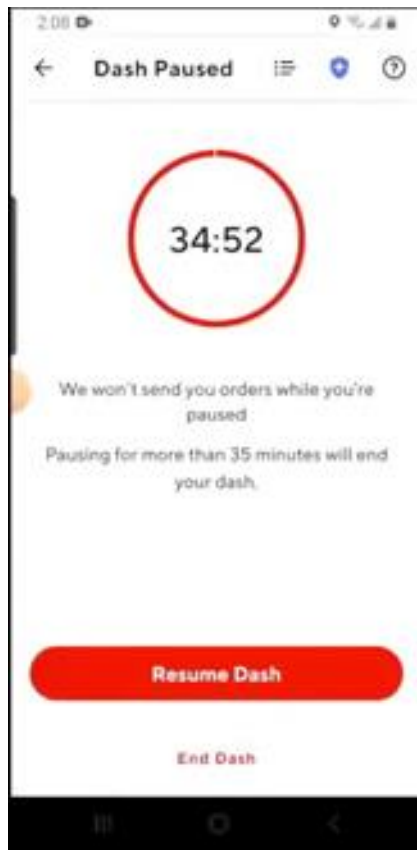


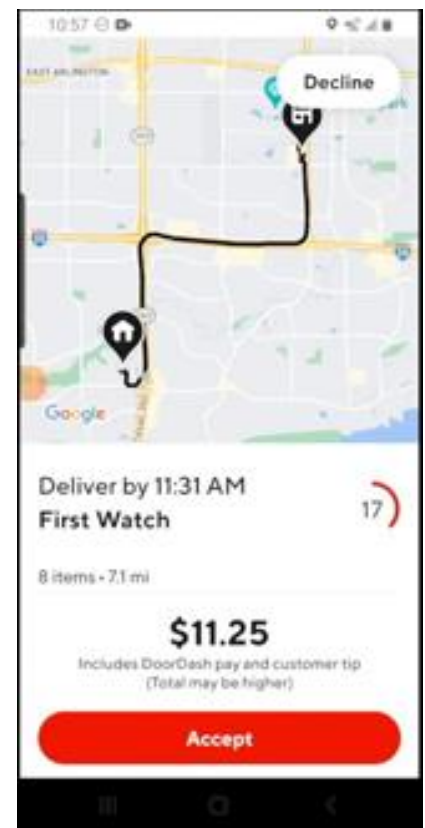
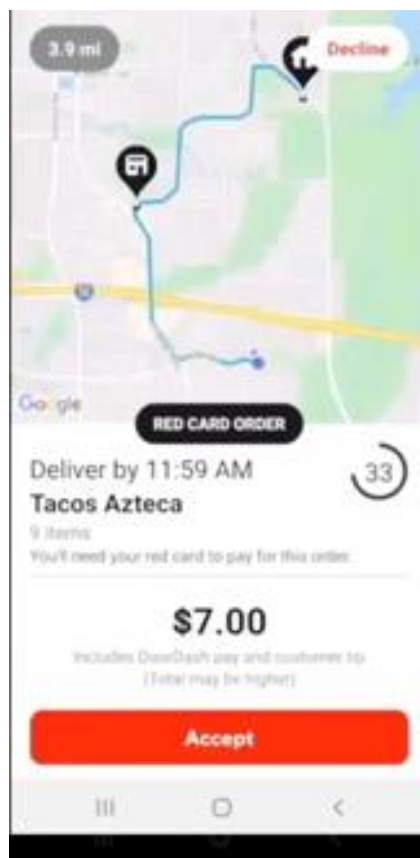
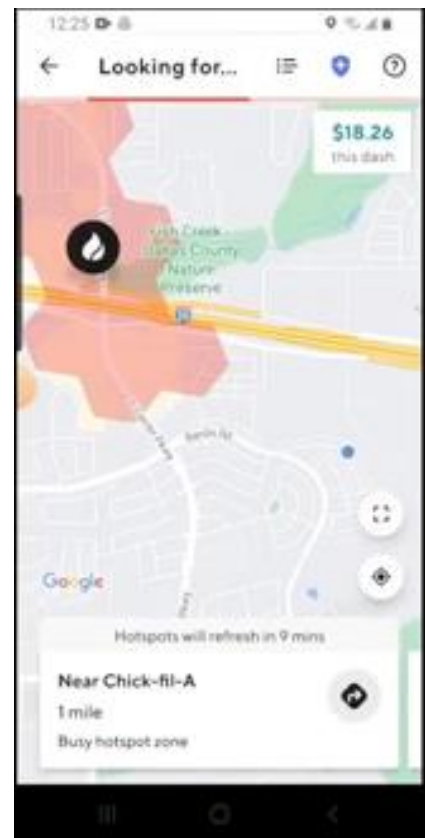
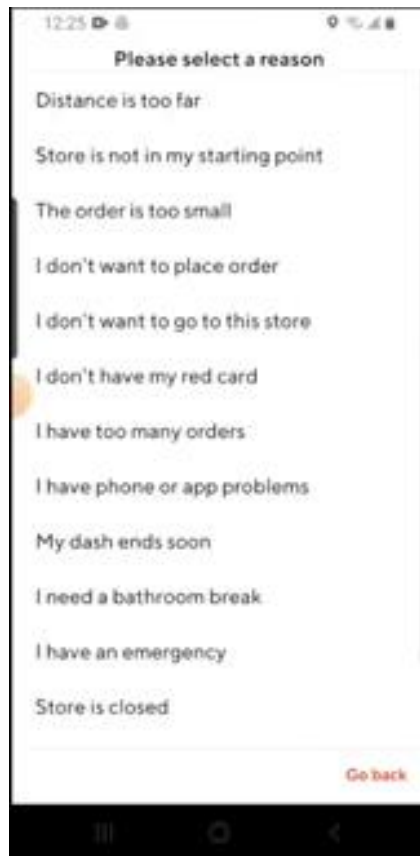


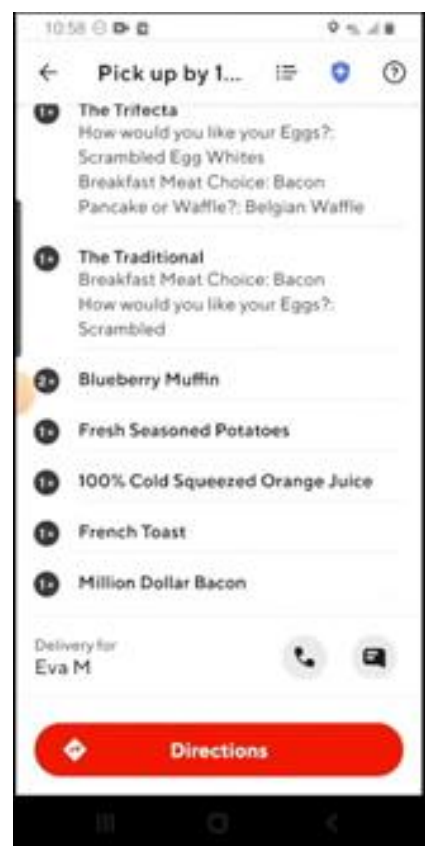
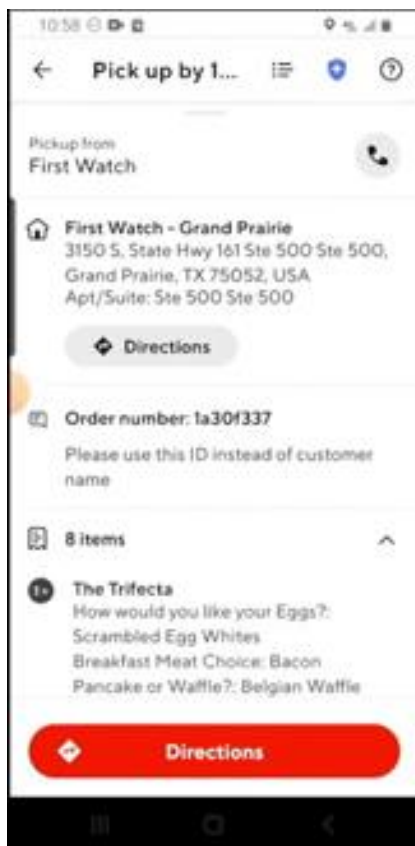
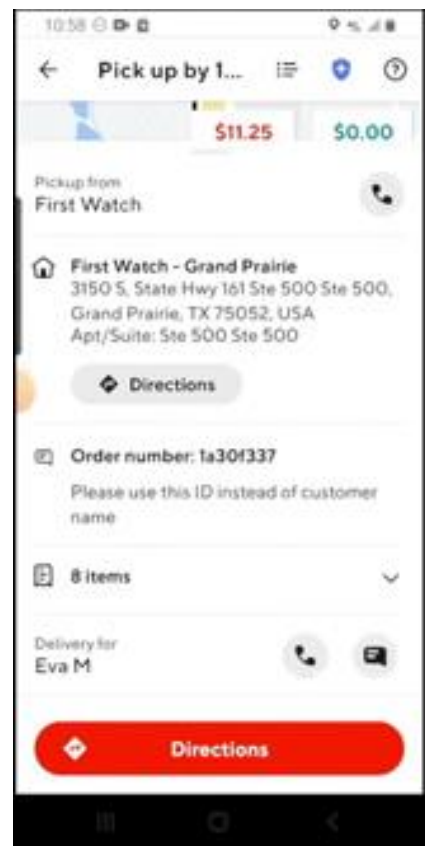
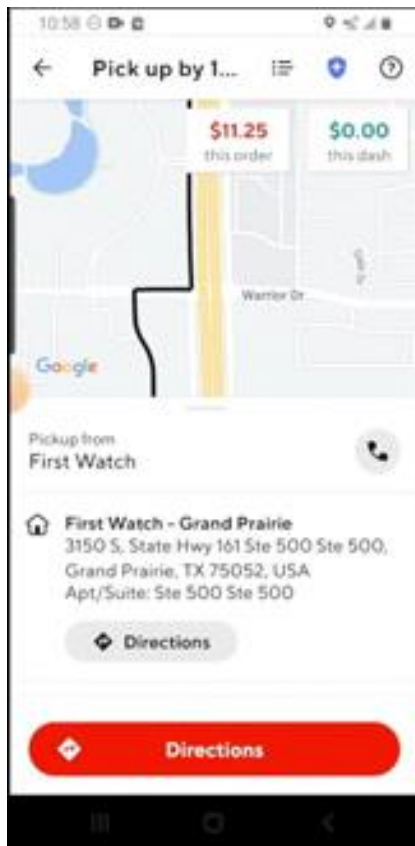


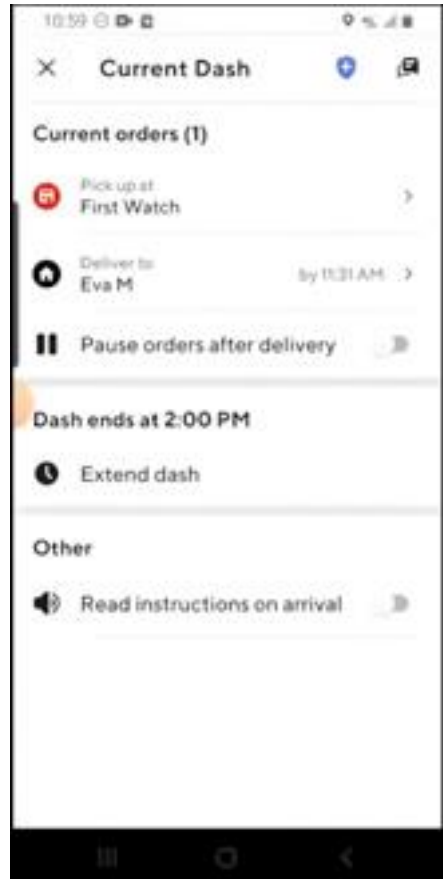
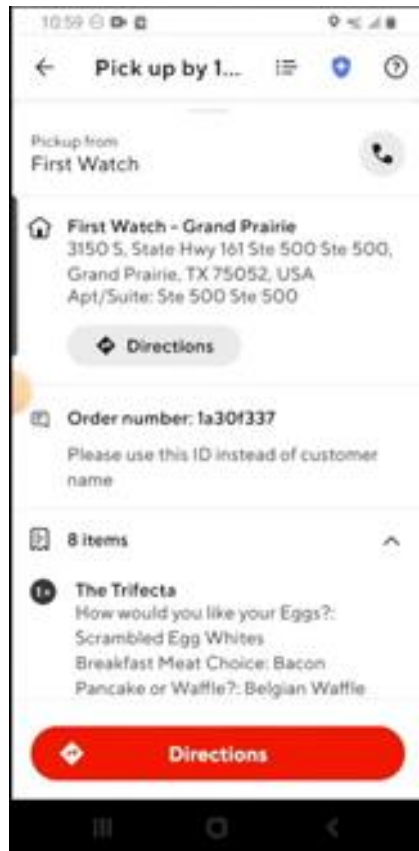
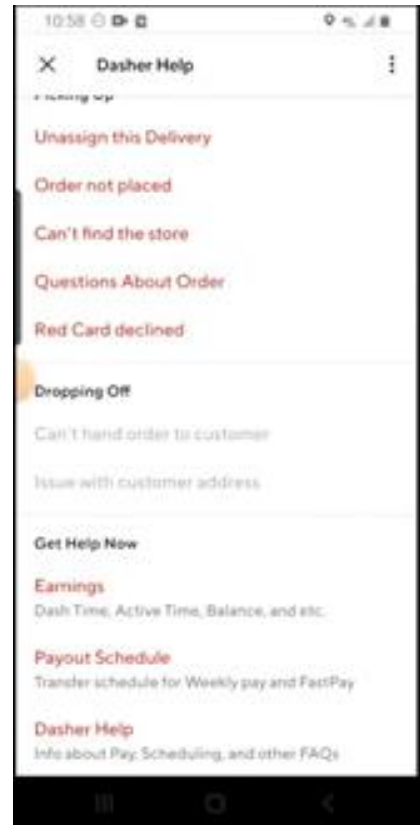
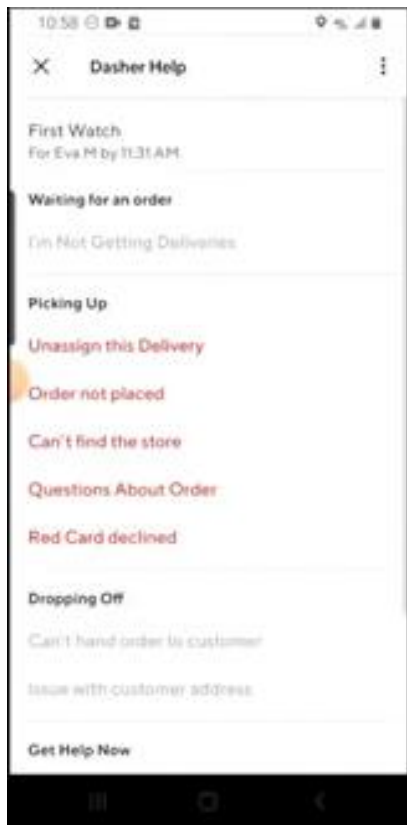


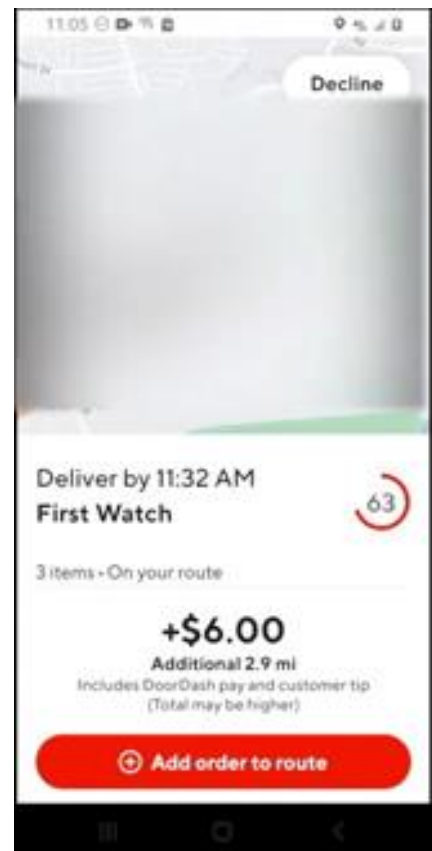
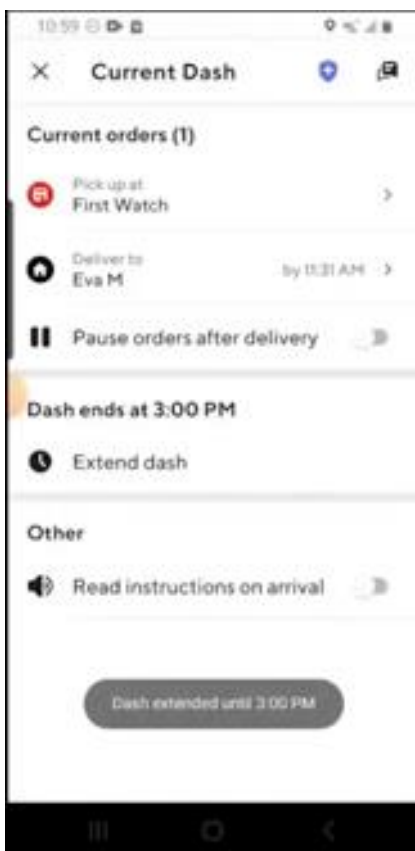
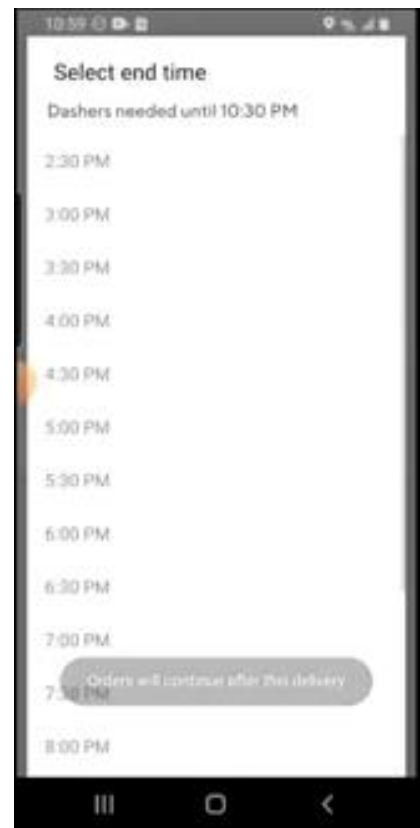
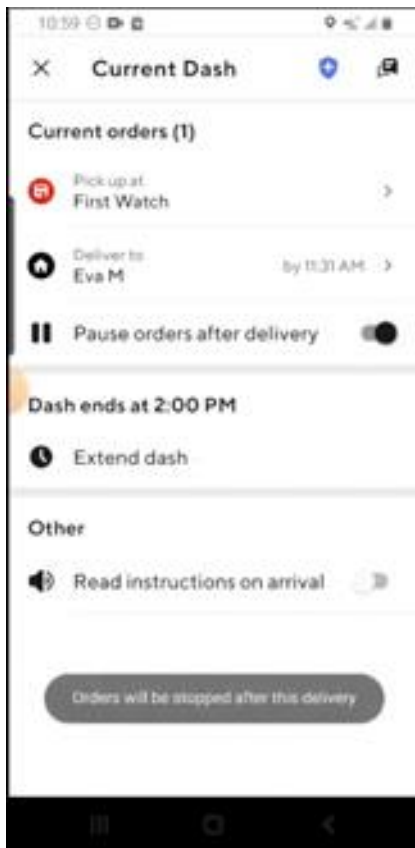


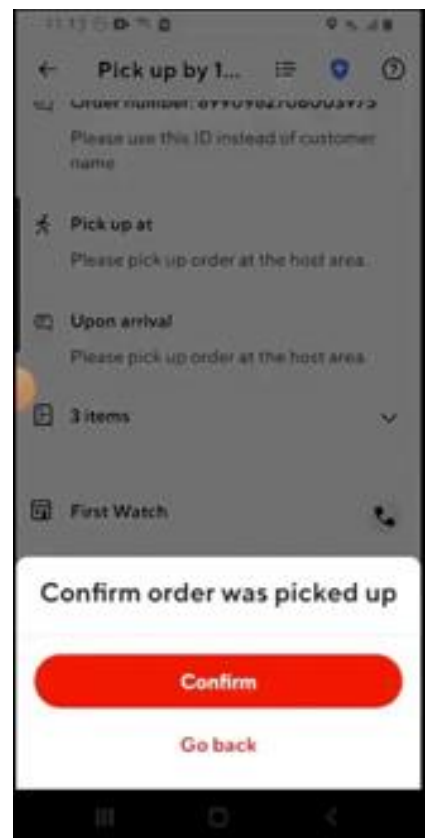
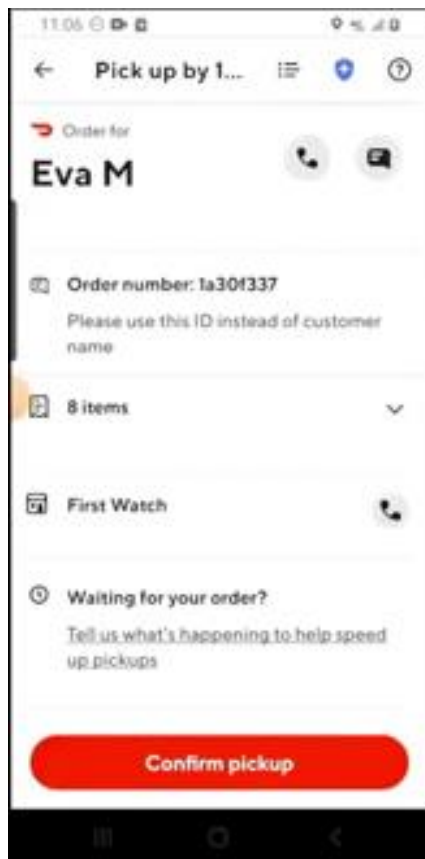
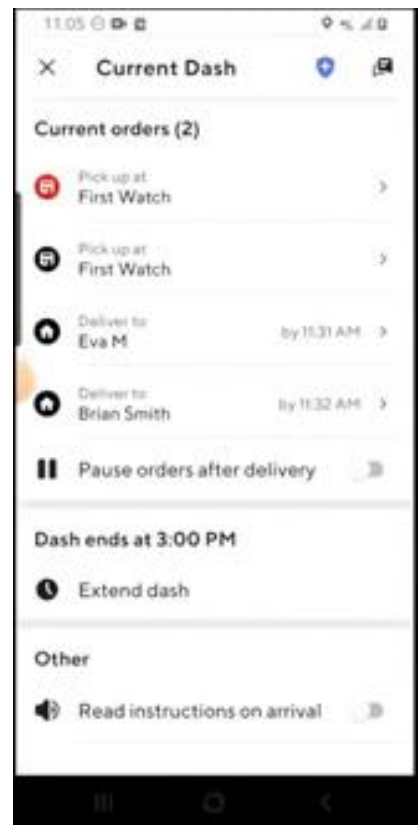
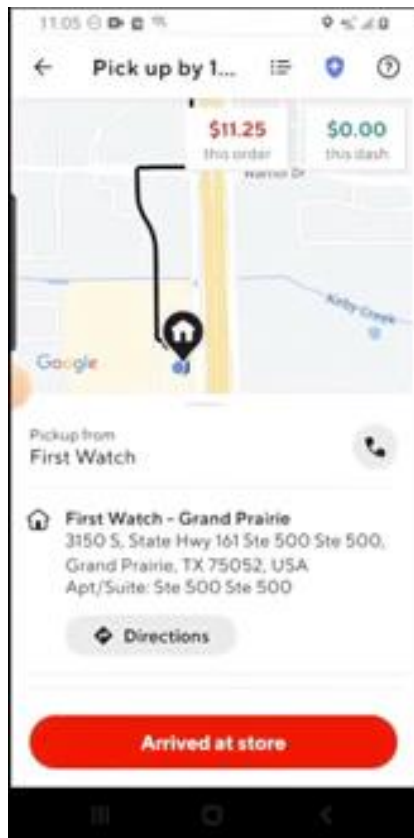


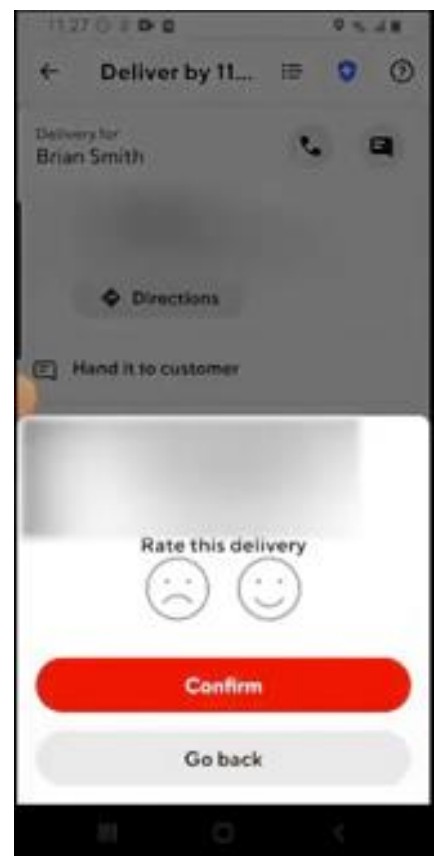
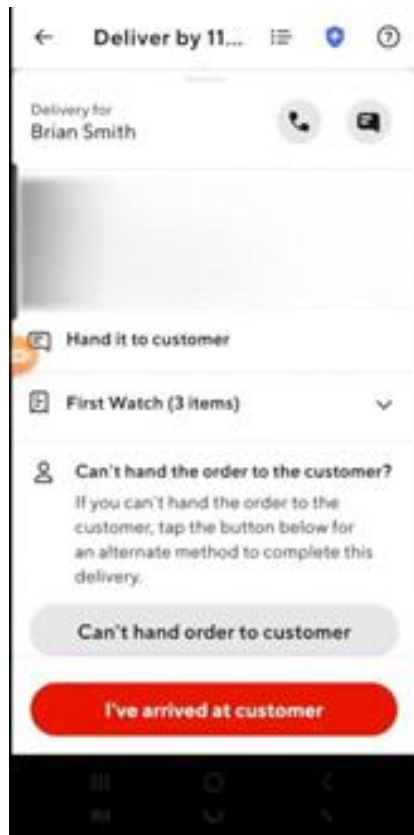


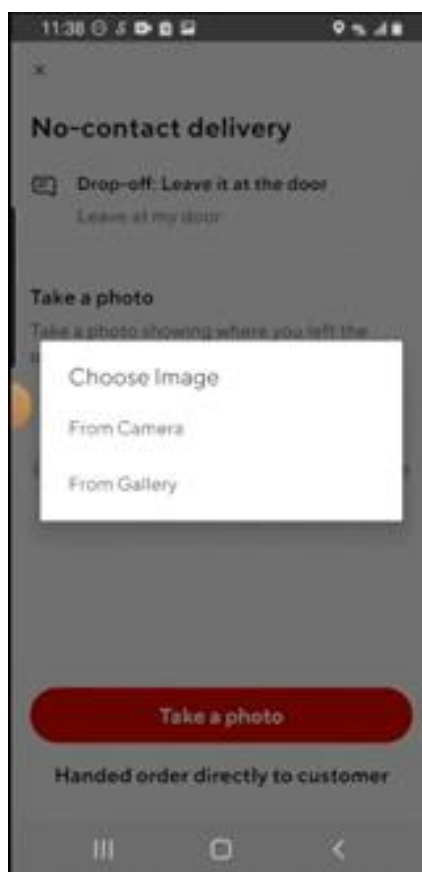
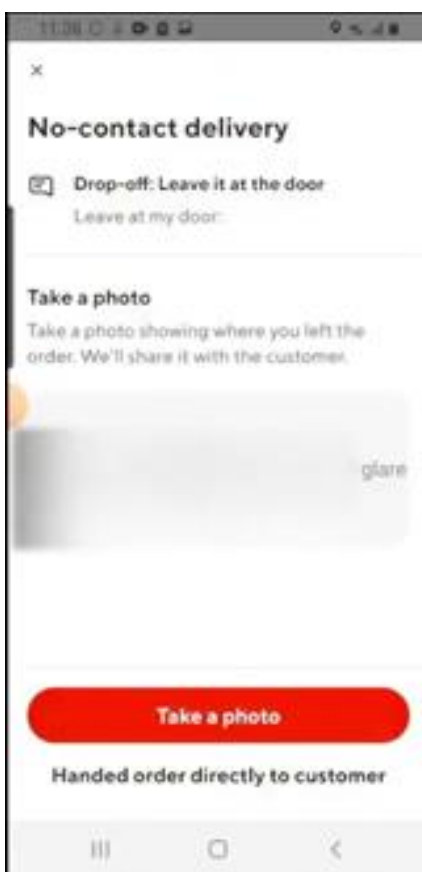
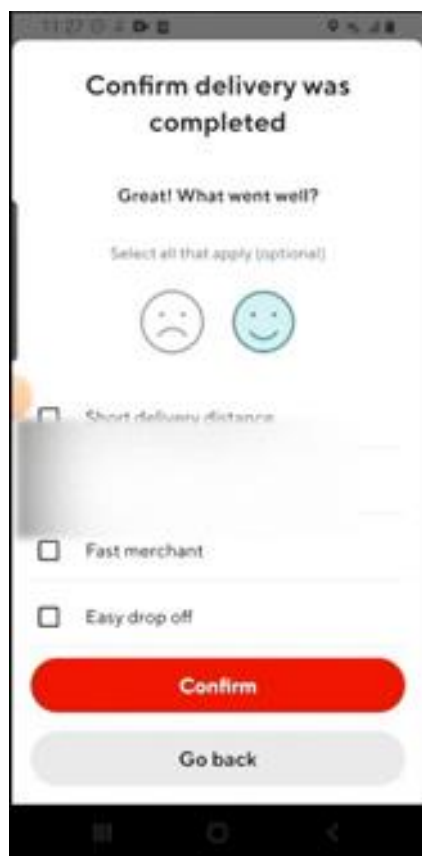


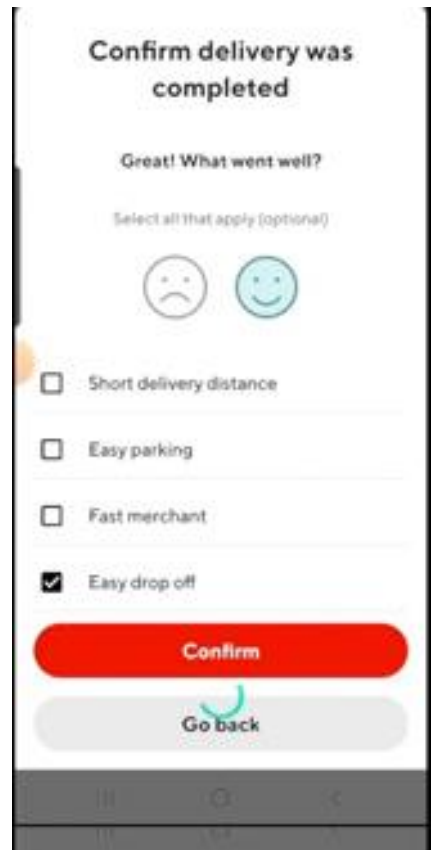
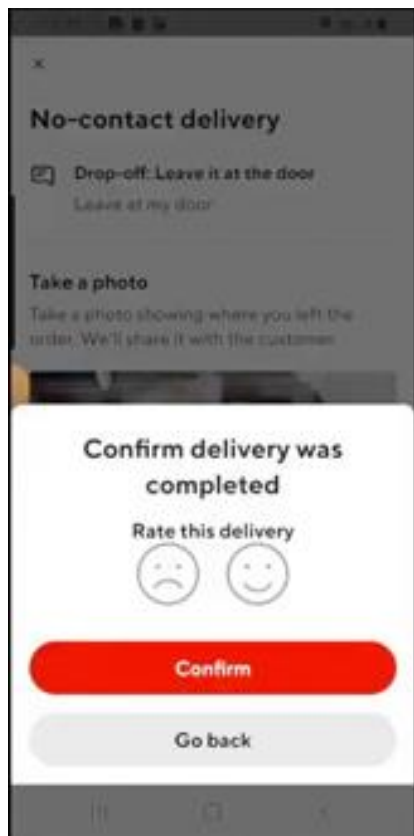
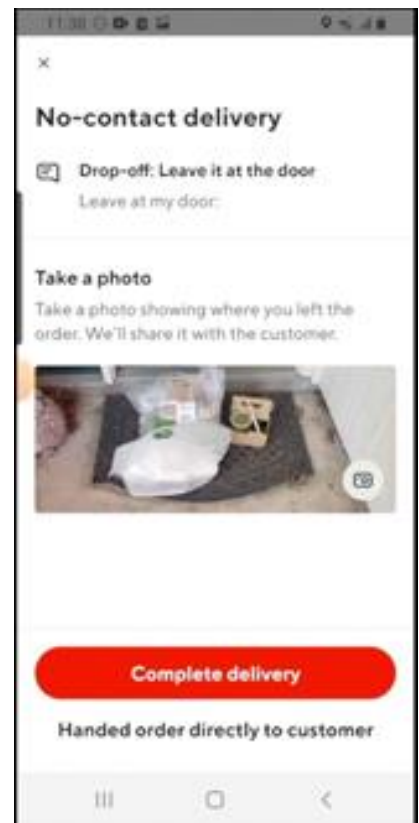


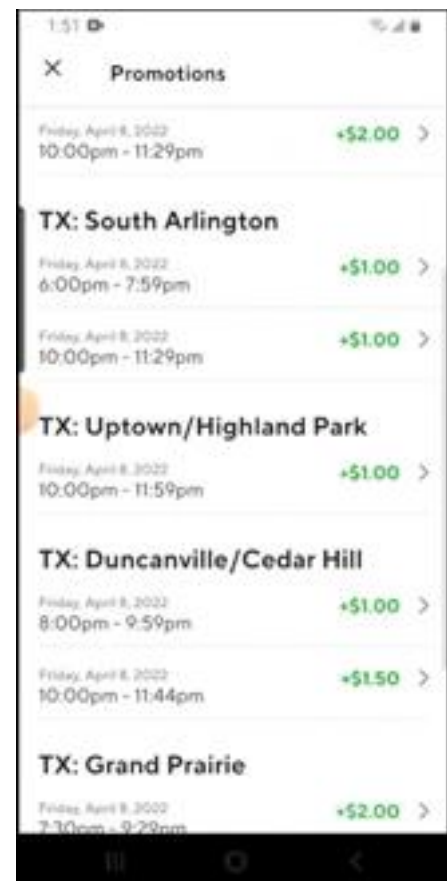
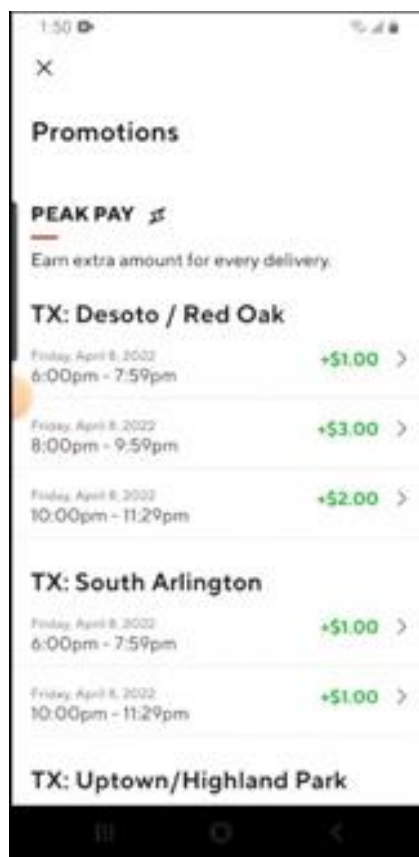
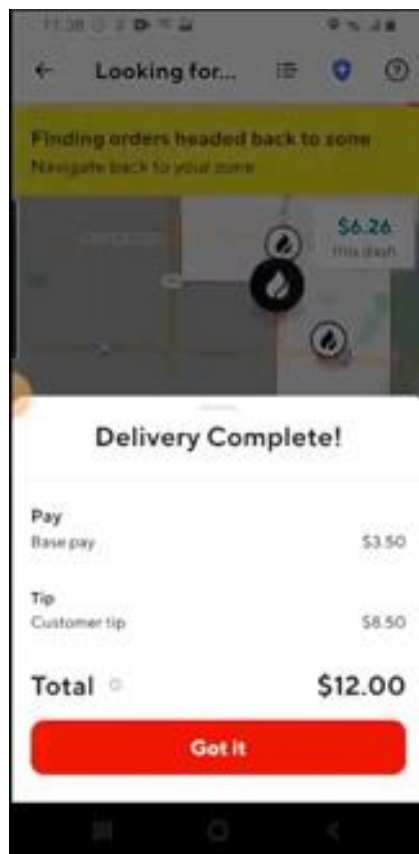


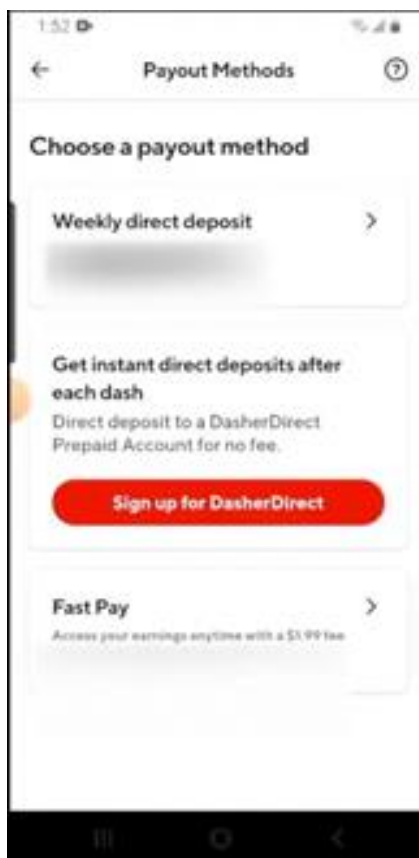
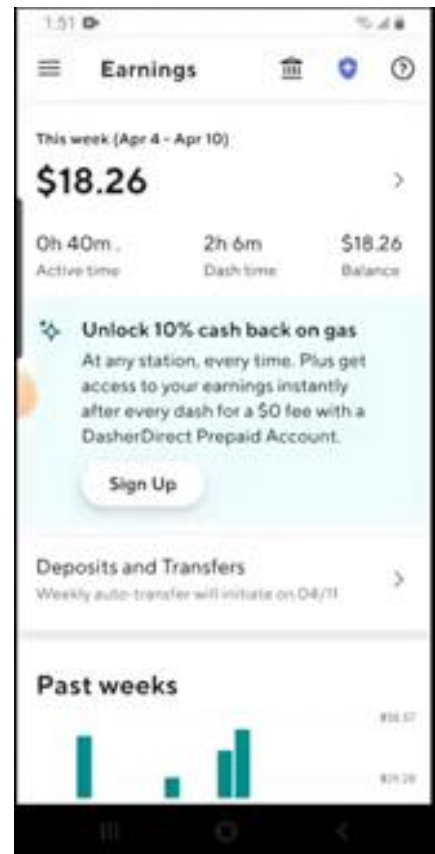
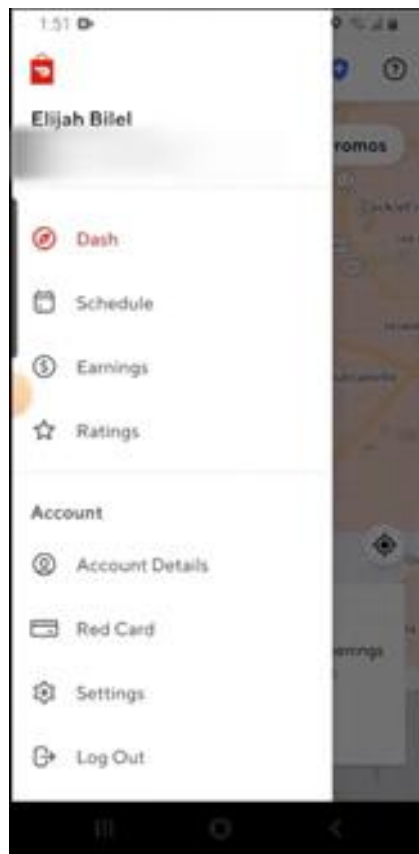


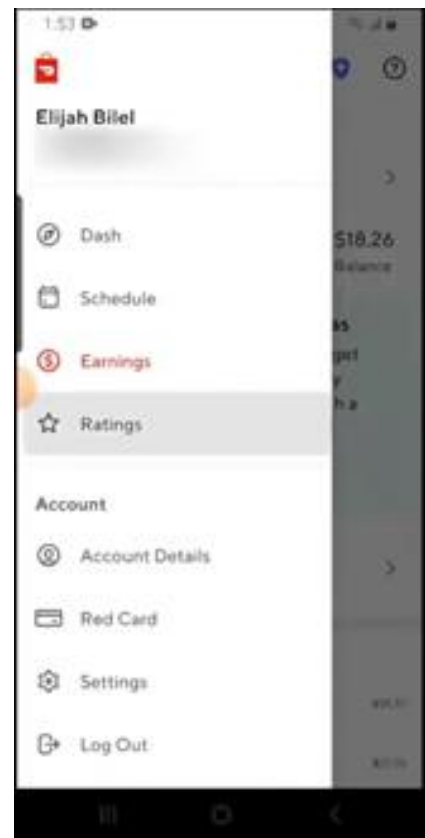
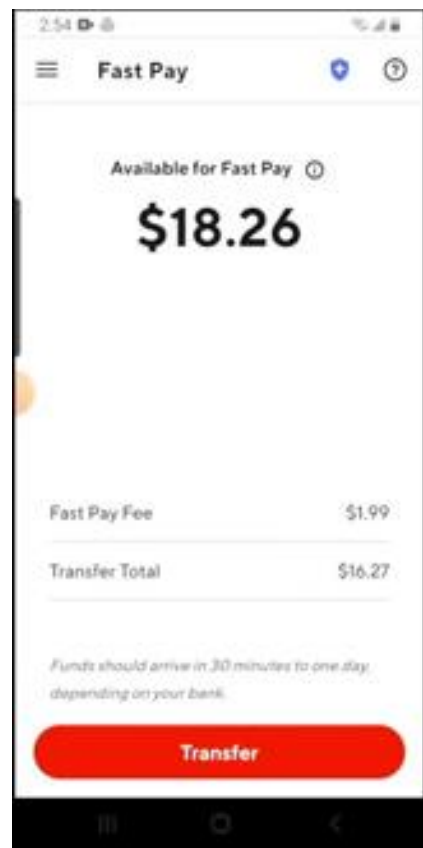
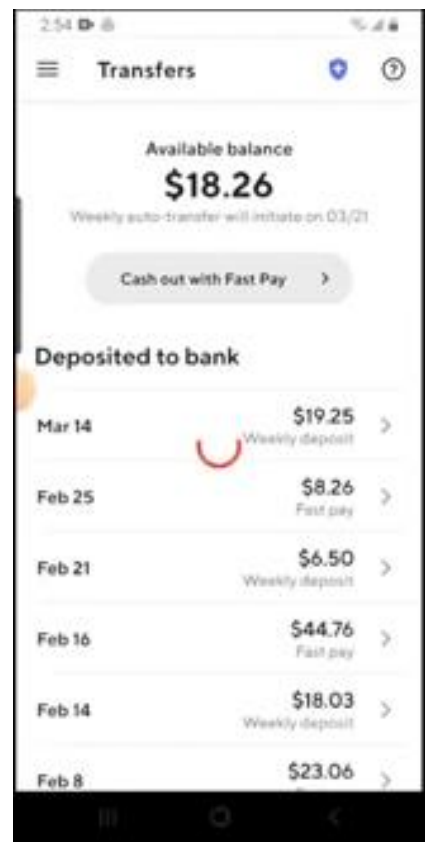
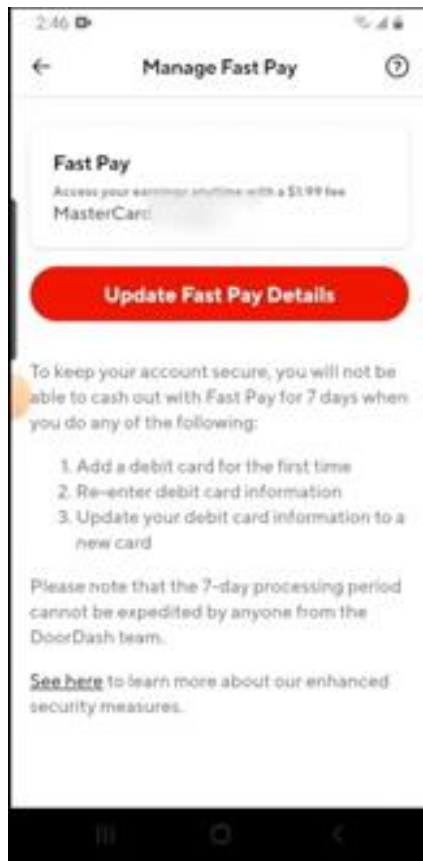


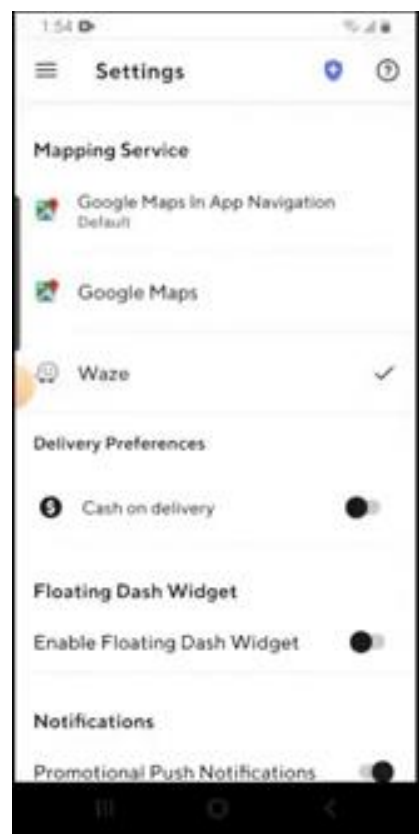






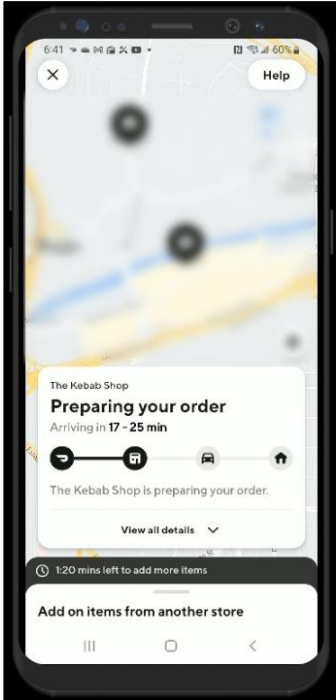
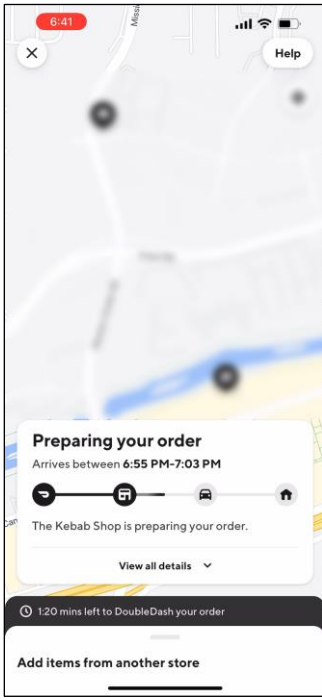


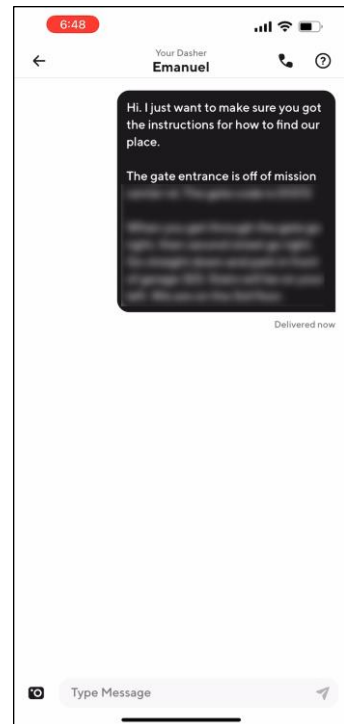
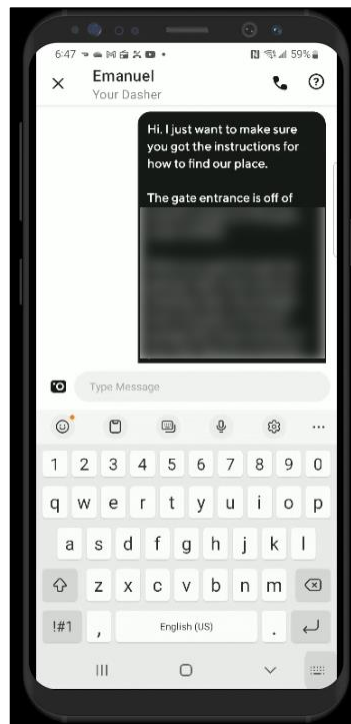
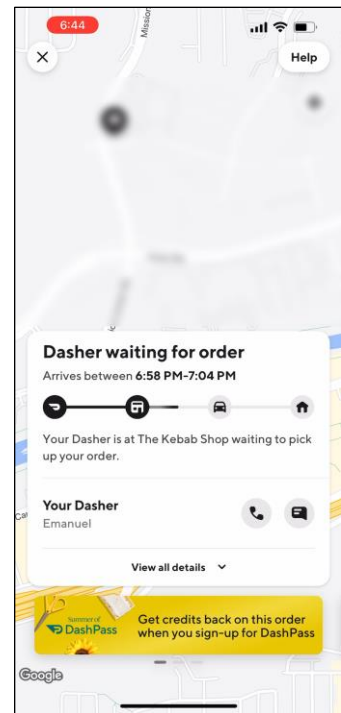


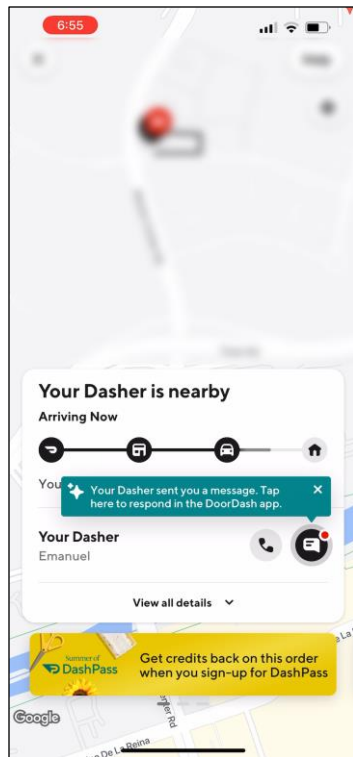
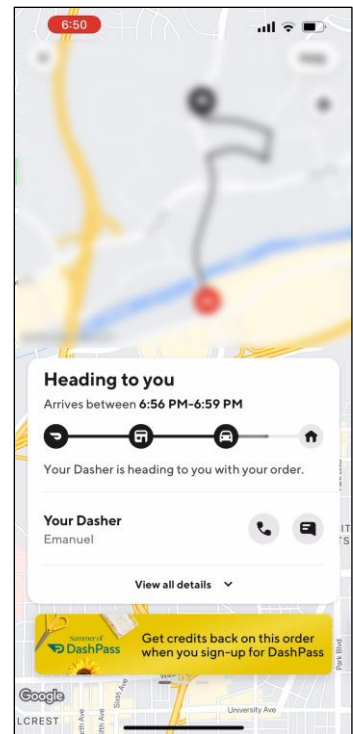


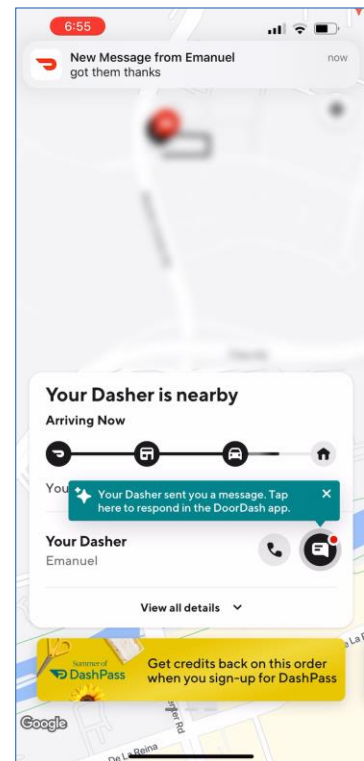
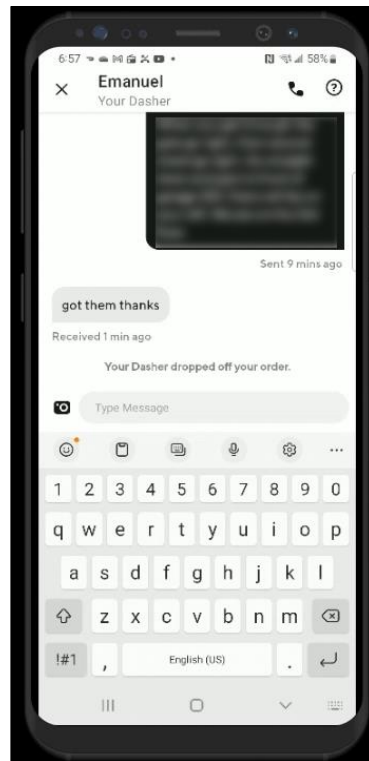
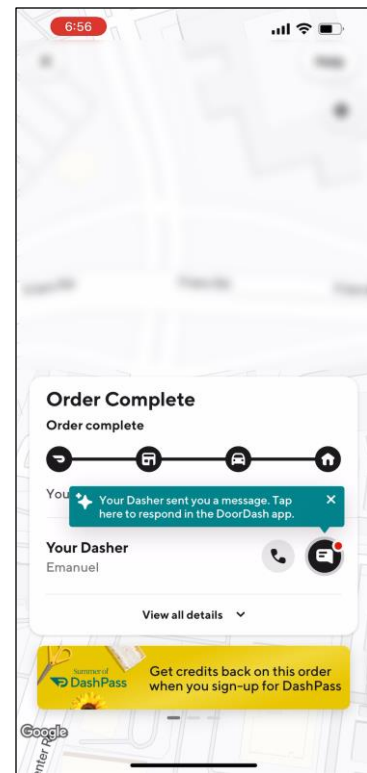
Mobile App Order Tracking, Notification, and in-app Chat

Below are a series of screen shots taken from one iPhone and one Android each tracking the same order placed by a user in the DoorDash Android App. They depict the order notifications and order updates in the app for the user to be able to track their order and delivery. In this case, we have one iPhone and one Android device tracking the same order, we can see the intelligent decisions being made by the Dispatcher regarding estimated time of delivery, and dasher time and distance from fulfilling the order. We also see text messages placed and sent by the user to the Dasher from the DoorDash android app, also viewable and accessible in the DoorDash iOS app on an iPhone that is tracking the order. When the Dasher responds to the chat from the user, the user is notified on the android phone and on the iOS phone. This functionality confirms multiple modes of contact are used by the two or more wireless handheld computers in support of the remote initiated hospitality tasks.

| Android DoorDash App | iOS DoorDash App |
|--|---|
|  <p>The screenshot shows the Android DoorDash app interface. At the top, the status bar displays the time 6:41 and 60% battery. The app header includes a 'Help' button. The main content area shows 'The Kebab Shop' with the status 'Preparing your order' and 'Arriving in 17 - 25 min'. Below this is a progress bar with icons for the shop, a driver, and a home icon. A message states 'The Kebab Shop is preparing your order.' and there is a 'View all details' link. At the bottom, a dark bar indicates '120 mins left to add more items' and a button to 'Add on items from another store'.</p> |  <p>The screenshot shows the iOS DoorDash app interface. The status bar at the top displays the time 6:41. The app header includes a 'Help' button. The main content area shows 'The Kebab Shop' with the status 'Preparing your order' and 'Arrives between 6:55 PM-7:03 PM'. Below this is a progress bar with icons for the shop, a driver, and a home icon. A message states 'The Kebab Shop is preparing your order.' and there is a 'View all details' link. At the bottom, a dark bar indicates '120 mins left to DoubleDash your order' and a button to 'Add items from another store'.</p> |







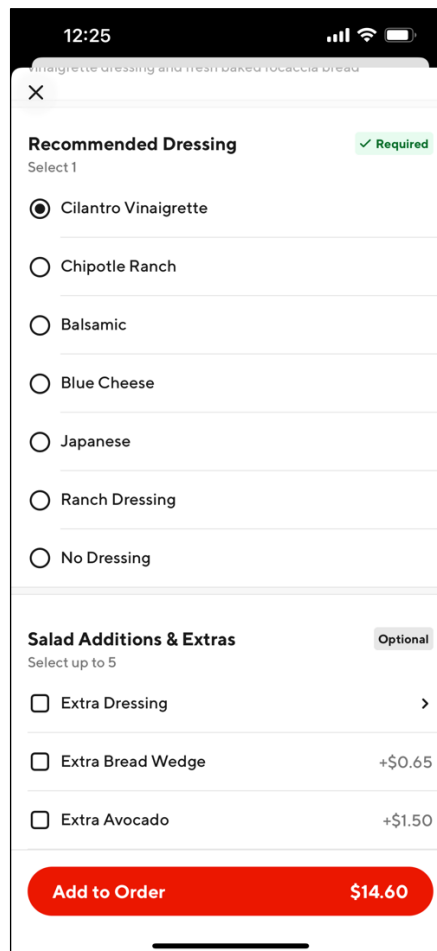
In the DoorDash Engineering blog article “Managing React State on DoorDash’s Item Modal Using the Class Pattern” (see Exh. 27) they state that DoorDash’s created an Item Modal to shows users what menu items (and their sub-modifiers) they can order from.

The Item Modal is a dynamic form in which they display menu item data (menus, modifiers, and sub-modifiers) to the user to;

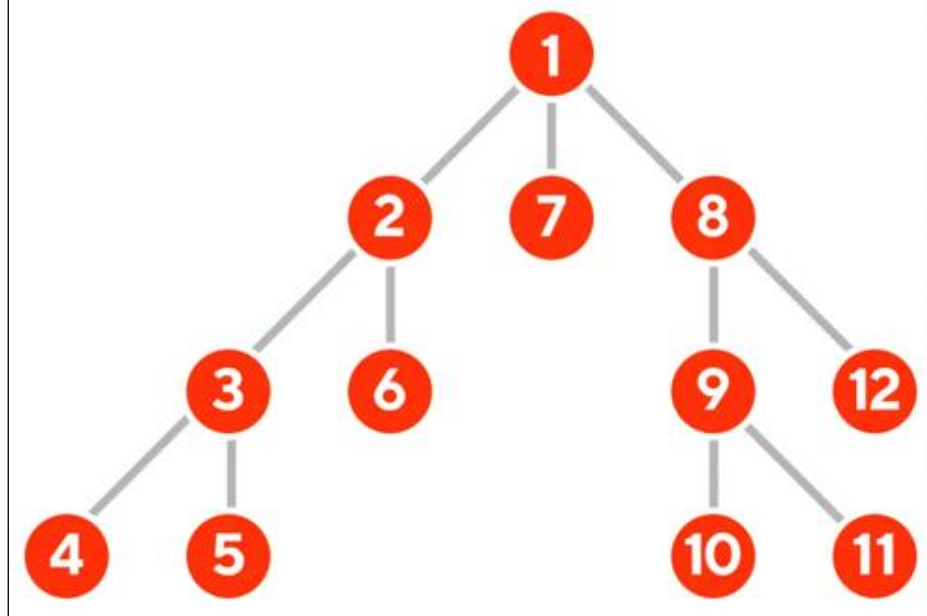
“...take and validate user inputs based on boundary rules set by the item data, dynamically calculate item prices based on these user inputs, and submit valid user-modified items to a persistent data store. and validate their inputs based on rules set by the item data, dynamically item prices” (see Exh 75). This Item Modal they claim is:

...one of the most complex components of our app and web frontends”. When DoorDash moved to a microservices architecture, it gave them the ability to “...rethink how we manage the Item Modal on our React-based web frontend.”

Here is a screenshot example:

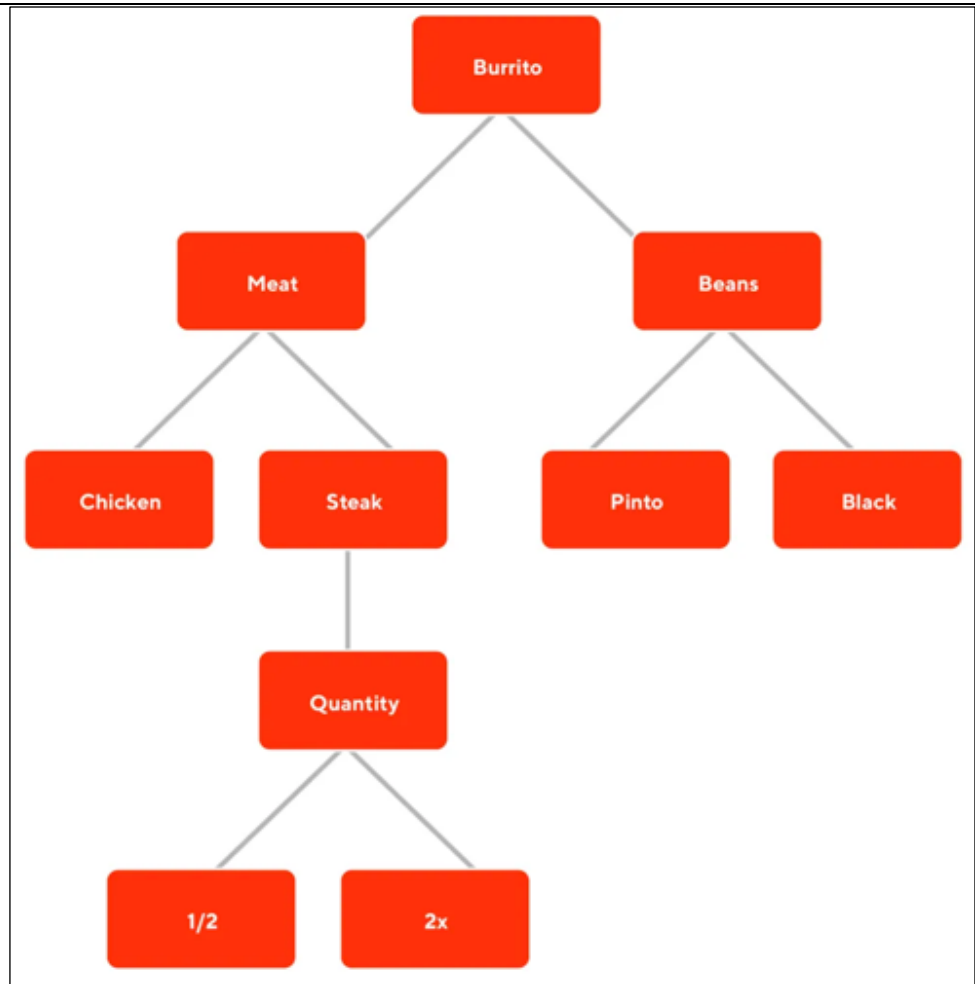


They describe the most important characteristic of the Item Modal rebuild was “...a TreeState, which is represented as an N-ary tree...” as shown below. In this design “...every item, has an ItemNode, and this ItemNode can have any number of OptionListNode. Each OptionListNode can have any number of OptionNodes, and these OptionNodes can have any number of OptionListNodes and so on.”

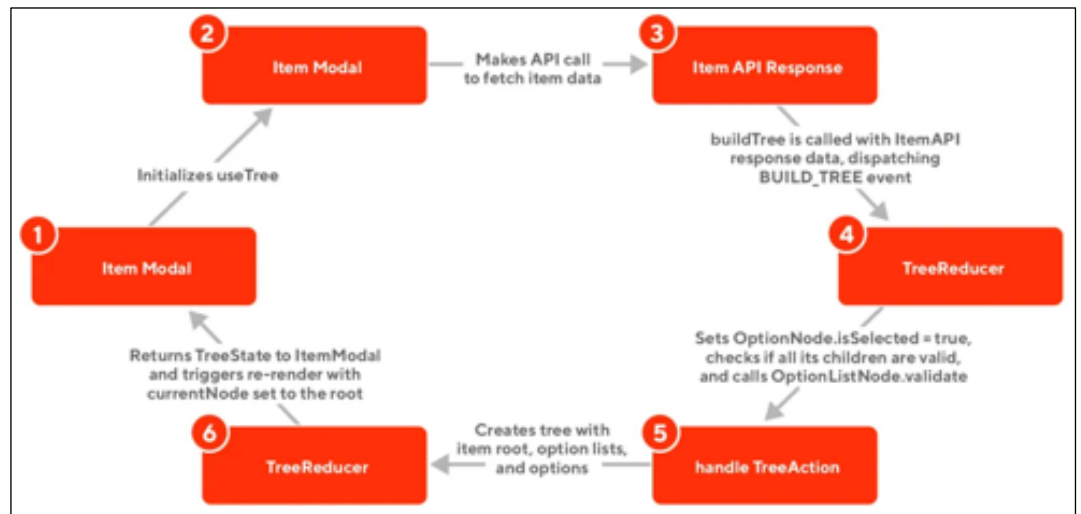


To better understand the model, they use the example of a user ordering a burrito and

“...in which a user can choose from two meats, chicken or steak, and two beans, pinto or black. We can take it further by allowing the user to select the quantity of meat via a nested option.” The figure below shows that “[o]rdering a burrito provides a variety of options, as displayed in the N-ary tree above, making it a helpful way to visualize an otherwise complex item.”



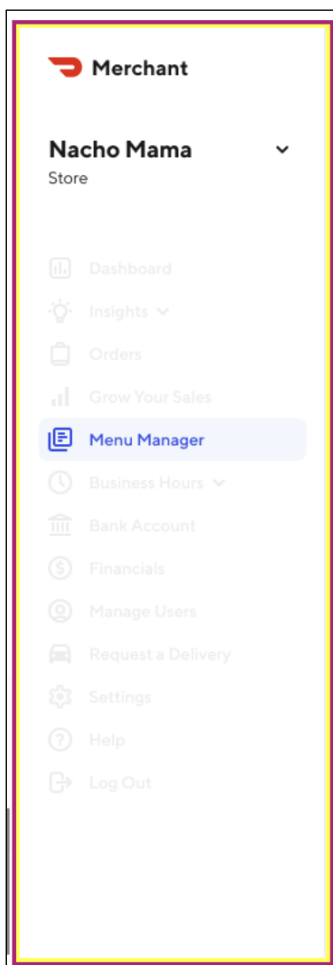
To render the Item Modal for a particular merchant, they must build the initial tree state with menu item data (including modifiers, and sub-modifiers) acquired via an API call to the web server (BFF) and validate accordingly. This requires a substantial workflow as exhibited by the chart below (see Exh. 75):

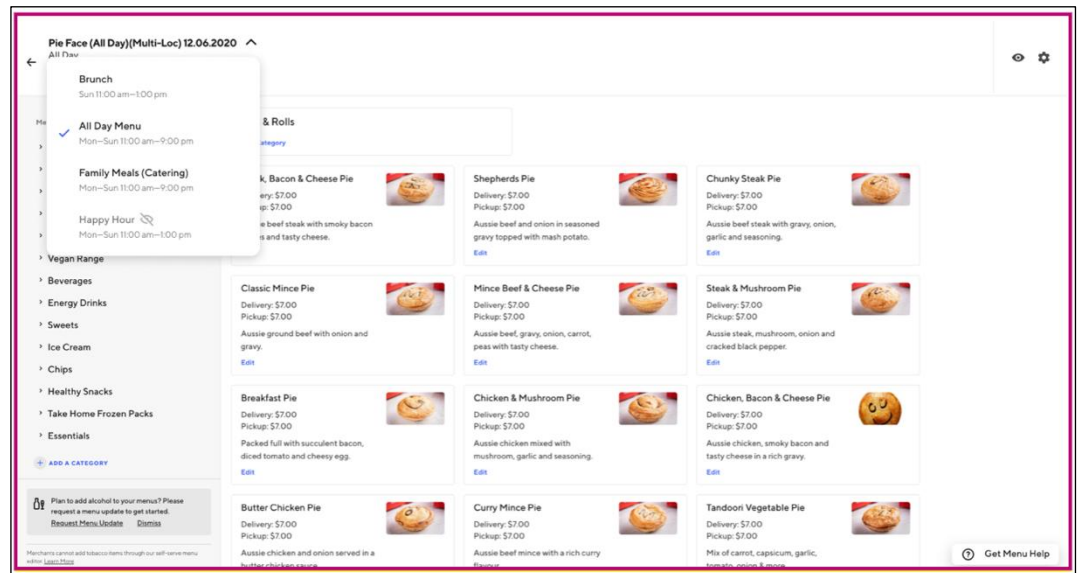
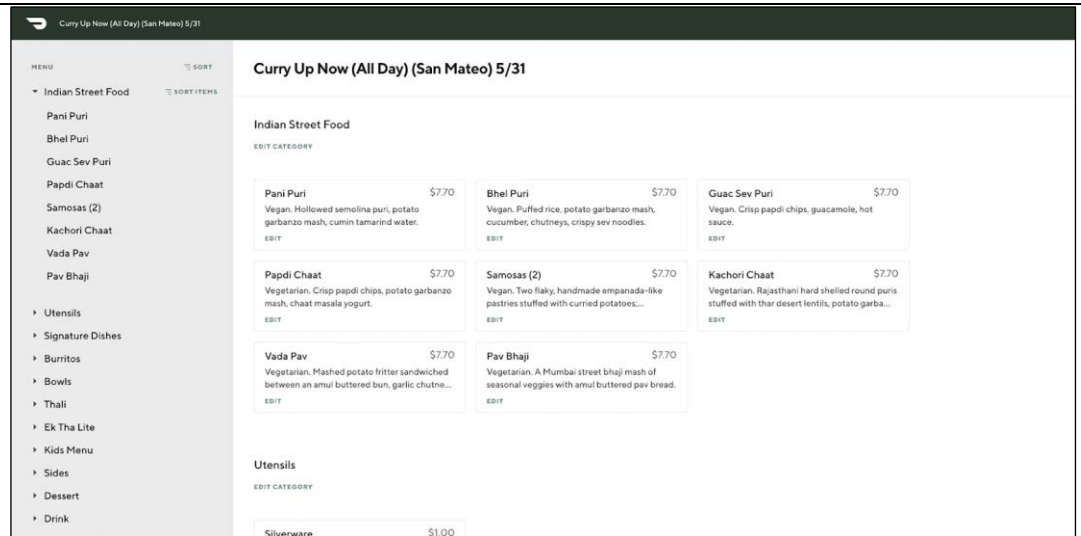


Menu Management in DoorDash Business Manager

DoorDash provides merchants two methods for managing their menus on the DoorDash platform which are categorized into “POS Integrated Users” and “Non-POS Integrated users.”

A “menu manager” on their merchant portal (website) allows the merchants to build and manage how their menu appears on the platform. The UI provides options to build menu items, categories, modifiers, add photos, and specify the sort order. Their Menu Editor Guide details how you can create menu items, categories, modifiers, etc. (see Exh 28, 29, 97). Example of this are screenshots from the Menu Editor tool below:





According to the DoorDash documentation by using the merchant portal, restaurants can: (See Exh. 29, 30)

“Add new items, modifiers, and options to your menu

Edit categories, items, modifiers, and options (setting prices, descriptions and names for each)

Sort categories, items, and modifiers

Temporarily deactivate items

Remove items from your menu permanently

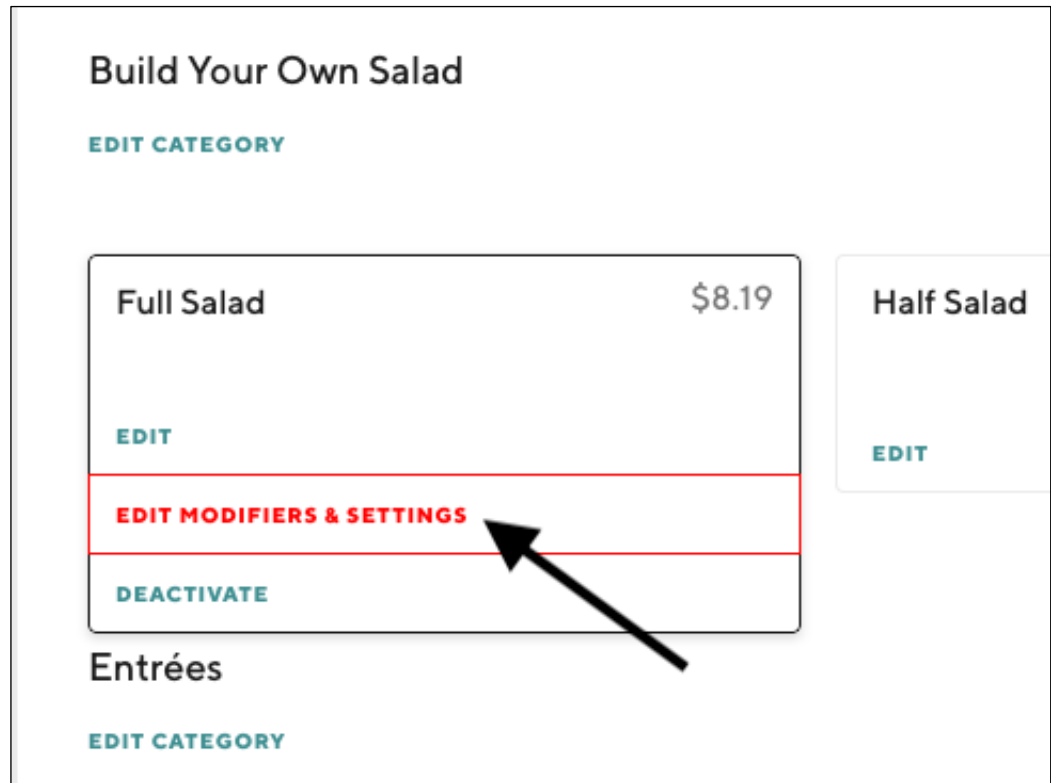
Update modifier settings”

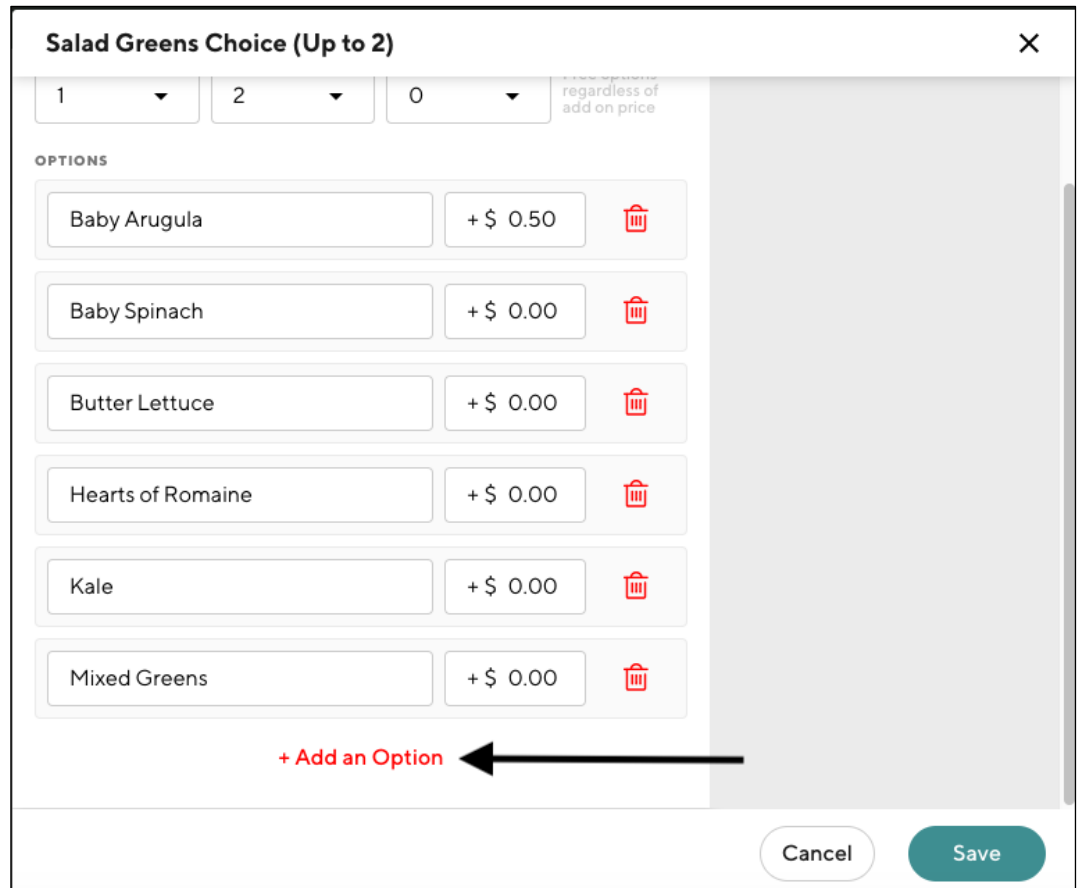
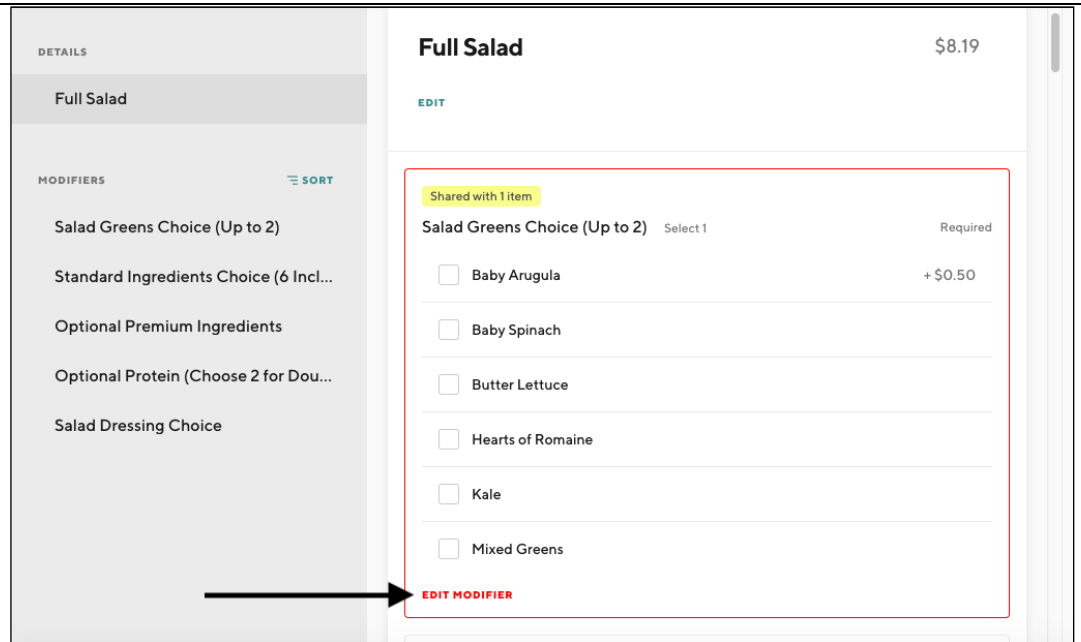
Menu: How to Add a New Option to a Modifier

In the user support article, “Adding a new option to a modifier on your menu” (see Exh. 73) they state “...[a] Modifier is a part that makes up an Item on your menu. For example, Greens are a modifier of a salad entree. Options within a Modifier are Kale,

Romaine, Mixed, etc.” The article shows the user how to add a new option to their restaurant’s menu which includes the following steps and screenshots:

1. Click Edit Modifiers & Settings on the item with the modifier that you would like to edit
2. On the next screen, click Edit Modifier at the bottom of the modifier
3. Scroll to the bottom and click Add an Option
4. Type in the new option and price and click Save”



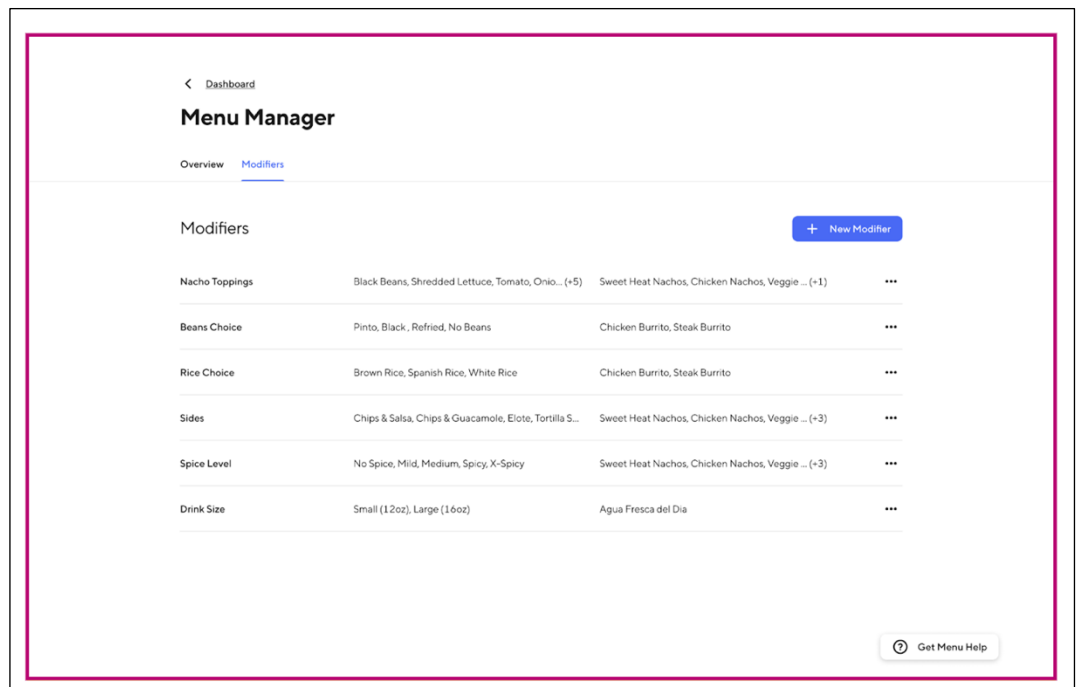
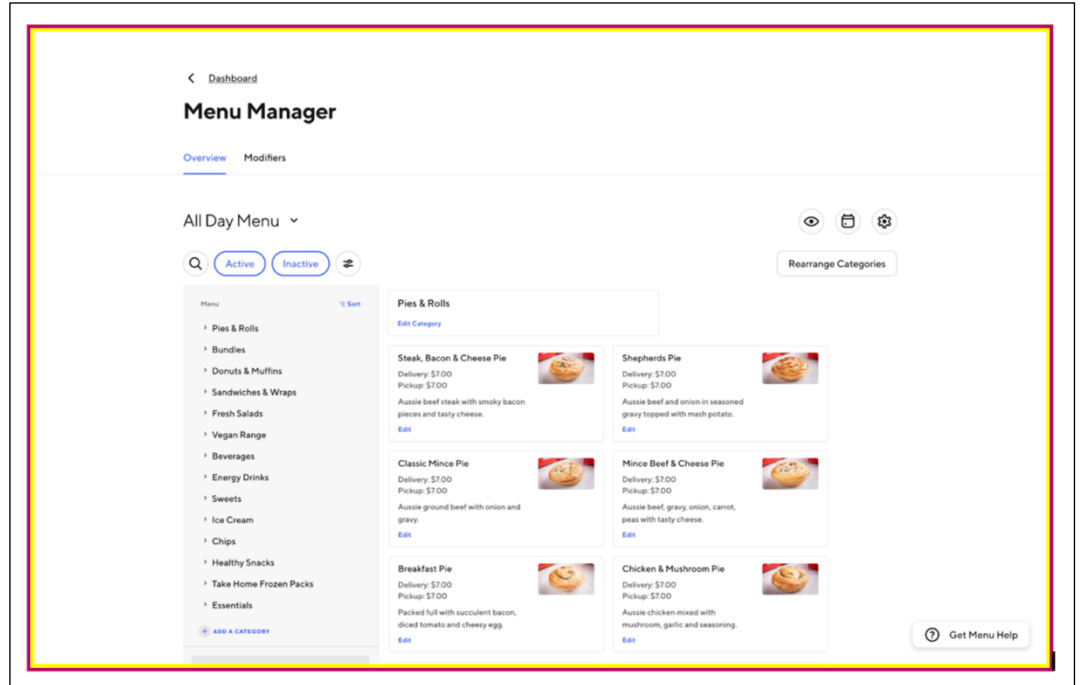


To edit the modifier settings for a single menu, according to the article, a user follows the following steps (as shown in the below screenshots):

“

1. *Click Modifiers at the top*

2. On the Modifiers page, click on the three dots to the right of the modifier that you would like to edit
3. In the pop-up that appears, click Edit Details
4. In the right panel, in "Add an option", enter new option name and price
5. Press enter and click Save Changes





Menu: How to Edit Modifier Settings on your Menu

In the user support article, “Editing Modifier Settings on your Menu” (see Exh. 74) they state, “Changes to modifier settings can be made in the Menu Editor in your Merchant Portal by following [these] steps”. The article proceeds to train the user on how to edit modifier settings by following these steps and screenshots:

“Select at least: The minimum selections a customer must make on the modifier. Set this to 0 if the modifier is optional.

Select at most: The maximum selections a customer can make on the modifier. This should be set to at least 1 and can not be less than the number in Select at least.

Free options: The number of free options available to the customer. The option is free regardless of add on price. For example, if Free options is set to 1 and each option costs \$1.00, the customer will not be charged \$1.00 for the first option they select, but they

will be charged for the second option they select.”

Build Your Own Salad

EDIT CATEGORY

| | |
|---|----------------------|
| Full Salad \$8.19 | Half Salad |
| EDIT | EDIT |
| EDIT MODIFIERS & SETTINGS | |
| DEACTIVATE | |

Entrées

EDIT CATEGORY

Note: A black arrow points to the 'EDIT MODIFIERS & SETTINGS' button.

Full Salad \$8.19

[EDIT](#)

Shared with 1 item

Salad Greens Choice (Up to 2) Required

| | |
|--|---------|
| <input type="checkbox"/> Baby Arugula | +\$0.50 |
| <input type="checkbox"/> Baby Spinach | |
| <input type="checkbox"/> Butter Lettuce | |
| <input type="checkbox"/> Hearts of Romaine | |
| <input type="checkbox"/> Kale | |
| <input type="checkbox"/> Mixed Greens | |

[EDIT MODIFIER](#)

Note: A black arrow points to the 'EDIT MODIFIER' button.

Salad Greens Choice (Up to 2)

NAME
Salad Greens Choice (Up to 2)

CUSTOMERS CAN
Select at least: 1 | Select at most: 2 | Free options: 0

OPTIONS

| | | |
|--------------|----------|--|
| Baby Arugula | +\$ 0.50 | |
| Baby Spinach | +\$ 0.00 | |

ITEMS USING THIS MODIFIER (1)
Half Salad

[Delete Modifier](#)

Menu Manager

Dashboard > Menu Manager

Overview | Modifiers

All Day Menu

Active | Inactive

Rearrange Categories

- Pies & Rolls
- Bundles
- Donuts & Muffins
- Sandwiches & Wraps
- Fresh Salads
- Vegan Range
- Beverages
- Energy Drinks
- Sweets
- Ice Cream
- Chips
- Healthy Snacks
- Take Home Frozen Packs
- Essentials

| Item Name | Delivery | Pickup | Description | Price |
|---------------------------|----------|--------|--|-------|
| Steak, Bacon & Cheese Pie | \$7.00 | \$7.00 | Aussie beef steak with smoky bacon pieces and tasty cheese. | |
| Shepherds Pie | \$7.00 | \$7.00 | Aussie beef and onion in seasoned gravy topped with mash potato. | |
| Classic Mince Pie | \$7.00 | \$7.00 | Aussie ground beef with onion and gravy. | |
| Mince Beef & Cheese Pie | \$7.00 | \$7.00 | Aussie beef, gravy, onion, carrot, peas with tasty cheese. | |
| Breakfast Pie | \$7.00 | \$7.00 | Packed full with succulent bacon, diced tomato and cheesy egg. | |
| Chicken & Mushroom Pie | \$7.00 | \$7.00 | Aussie chicken mixed with mushroom, garlic and seasoning. | |

[Get Menu Help](#)

< Dashboard

Menu Manager


Overview Modifiers


Modifiers + New Modifier

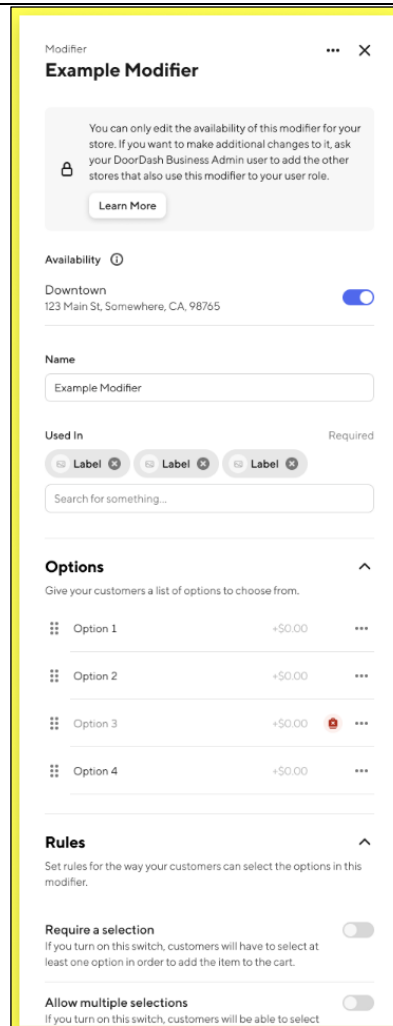
| | | | |
|----------------|--|--|-----|
| Nacho Toppings | Black Beans, Shredded Lettuce, Tomato, Onio... (+5) | Sweet Heat Nachos, Chicken Nachos, Veggie ... (-1) | ... |
| Beans Choice | Pinto, Black, Refried, No Beans | Chicken Burrito, Steak Burrito | ... |
| Rice Choice | Brown Rice, Spanish Rice, White Rice | Chicken Burrito, Steak Burrito | ... |
| Sides | Chips & Salsa, Chips & Guacamole, Elote, Tortilla S... | Sweet Heat Nachos, Chicken Nachos, Veggie ... (-3) | ... |
| Spice Level | No Spice, Mild, Medium, Spicy, X-Spicy | Sweet Heat Nachos, Chicken Nachos, Veggie ... (-3) | ... |
| Drink Size | Small (12oz), Large (16oz) | Agua Fresca del Dia | ... |

Get Menu Help

...

 Edit Details

 Delete



Here is another example. The slides below were extracted from a DoorDash merchant training video in which they show the user how to add a menu item using the Menu Editor in the Merchant Portal (see Exh 37):

Sprout Cafe 1/12

MENU SORT

- Small Bites
- Signature Soups
- Entrées
- Build Your Own Salad
- Signature Salads
- Grilled Cheese Sandwiches
- Cool Sandwiches
- Warm Sandwiches
- Snacks
- Drinks

Sprout Cafe 1/12

Small Bites

EDIT CATEGORY

| | |
|--|--|
| Summer Rolls \$5.90 With mixed green lettuce, somen noodles, carrots, bean sprouts, caramelized shallots, a... EDIT | Guacamole \$8.03 With sour cream and homemade chips. EDIT |
| Fried Calamari \$9.10 With chili lime aioli. EDIT | Garlic Sweet Potato Fries \$7.49 With chipotle ketchup and chili lime aioli. EDIT |
| Pork Potstickers \$8.03 6 pieces. With chili soy ginger sauce. EDIT | Bulgogi Lettuce Cups \$6.96 2 pieces. With rib eye steak, Gochujang sauce, crushed peanuts, carrots, and cilantro. EDIT |
| Shoestring Fries \$5.50 Chipotle ketchup, chili lime aioli EDIT | Sesame Tofu \$6.42 Crispy tofu in a tangy sesame sauce. EDIT |

Click + ADD AN ITEM

Sprout Cafe 1/12

MENU SORT

- Small Bites
- Signature Soups
- Entrées
- Build Your Own Salad
- Signature Salads
- Grilled Cheese Sandwiches
- Cool Sandwiches
- Warm Sandwiches
- Snacks
- Drinks

Sprout Cafe 1/12

Small Bites

EDIT CATEGORY

| | |
|--|--|
| Summer Rolls \$5.90 With mixed green lettuce, somen noodles, carrots, bean sprouts, caramelized shallots, a... EDIT | Guacamole \$8.03 With sour cream and homemade chips. EDIT |
| Fried Calamari \$9.10 With chili lime aioli. EDIT | Garlic Sweet Potato Fries \$7.49 With chipotle ketchup and chili lime aioli. EDIT |
| Pork Potstickers \$8.03 6 pieces. With chili soy ginger sauce. EDIT | Bulgogi Lettuce Cups \$6.96 2 pieces. With rib eye steak, Gochujang sauce, crushed peanuts, carrots, and cilantro. EDIT |
| Shoestring Fries \$5.50 Chipotle ketchup, chili lime aioli EDIT | Sesame Tofu \$6.42 Crispy tofu in a tangy sesame sauce. EDIT |

Click

Unnamed item \$ 0.00

Item description

+ ADD A MODIFIER & DETAILS

Cancel Save

+ ADD AN ITEM

Sprout Cafe 1/12

MENU SORT

- Small Bites
- Signature Soups
- Entrées
- Build Your Own Salad
- Signature Salads
- Grilled Cheese Sandwiches
- Cool Sandwiches
- Warm Sandwiches
- Snacks
- Drinks

Sprout Cafe 1/12

Small Bites

EDIT CATEGORY

| | |
|--|--|
| Summer Rolls \$5.90 With mixed green lettuce, somen noodles, carrots, bean sprouts, caramelized shallots, a... EDIT | Guacamole \$8.03 With sour cream and homemade chips. EDIT |
| Fried Calamari \$9.10 With chili lime aioli. EDIT | Garlic Sweet Potato Fries \$7.49 With chipotle ketchup and chili lime aioli. EDIT |
| Pork Potstickers \$8.03 6 pieces. With chili soy ginger sauce. EDIT | Bulgogi Lettuce Cups \$6.96 2 pieces. With rib eye steak, Gochujang sauce, crushed peanuts, carrots, and cilantro. EDIT |
| Shoestring Fries \$5.50 Chipotle ketchup, chili lime aioli EDIT | Sesame Tofu \$6.42 Crispy tofu in a tangy sesame sauce. EDIT |

Type

Type here \$ 0.00

Item description

+ ADD A MODIFIER & DETAILS

Cancel Save

+ ADD AN ITEM

S Sprout Cafe 1/12

MENU SORT

- ▶ Small Bites
- ▶ Signature Soups
- ▶ Entrées
- ▶ Build Your Own Salad
- ▶ Signature Salads
- ▶ Grilled Cheese Sandwiches
- ▶ Cool Sandwiches
- ▶ Warm Sandwiches
- ▶ Snacks
- ▶ Drinks

Sprout Cafe 1/12

Small Bites

EDIT CATEGORY

| | |
|--|--|
| <p>Summer Rolls \$5.90</p> <p style="font-size: 0.8em;">With mixed green lettuce, somen noodles, carrots, bean sprouts, caramelized shallots, a...</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Guacamole \$8.03</p> <p style="font-size: 0.8em;">With sour cream and homemade chips.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Fried Calamari \$9.10</p> <p style="font-size: 0.8em;">With chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Garlic Sweet Potato Fries \$7.49</p> <p style="font-size: 0.8em;">With chipotle ketchup and chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Pork Potstickers \$8.03</p> <p style="font-size: 0.8em;">6 pieces. With chili soy ginger sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Bulgogi Lettuce Cups \$6.96</p> <p style="font-size: 0.8em;">2 pieces. With rib eye steak, Gochujang sauce, crushed peanuts, carrots, and cilantro.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Shoestring Fries \$5.50</p> <p style="font-size: 0.8em;">Chipotle ketchup, chili lime aioli</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Sesame Tofu \$6.42</p> <p style="font-size: 0.8em;">Crispy tofu in a tangy sesame sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |

Hummus Click \$ 0.00

+ ADD A MODIFIER & DETAILS

Cancel
Save

ADD AN ITEM

S Sprout Cafe 1/12

MENU SORT

- ▶ Small Bites
- ▶ Signature Soups
- ▶ Entrées
- ▶ Build Your Own Salad
- ▶ Signature Salads
- ▶ Grilled Cheese Sandwiches
- ▶ Cool Sandwiches
- ▶ Warm Sandwiches
- ▶ Snacks
- ▶ Drinks

Sprout Cafe 1/12

Small Bites

EDIT CATEGORY

| | |
|--|--|
| <p>Summer Rolls \$5.90</p> <p style="font-size: 0.8em;">With mixed green lettuce, somen noodles, carrots, bean sprouts, caramelized shallots, a...</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Guacamole \$8.03</p> <p style="font-size: 0.8em;">With sour cream and homemade chips.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Fried Calamari \$9.10</p> <p style="font-size: 0.8em;">With chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Garlic Sweet Potato Fries \$7.49</p> <p style="font-size: 0.8em;">With chipotle ketchup and chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Pork Potstickers \$8.03</p> <p style="font-size: 0.8em;">6 pieces. With chili soy ginger sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Bulgogi Lettuce Cups \$6.96</p> <p style="font-size: 0.8em;">2 pieces. With rib eye steak, Gochujang sauce, crushed peanuts, carrots, and cilantro.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Shoestring Fries \$5.50</p> <p style="font-size: 0.8em;">Chipotle ketchup, chili lime aioli</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Sesame Tofu \$6.42</p> <p style="font-size: 0.8em;">Crispy tofu in a tangy sesame sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |

Hummus Type \$ 0.00

+ ADD A MODIFIER & DETAILS

Cancel
Save

ADD AN ITEM

S Sprout Cafe 1/12

MENU SORT

- ▶ Small Bites
- ▶ Signature Soups
- ▶ Entrées
- ▶ Build Your Own Salad
- ▶ Signature Salads
- ▶ Grilled Cheese Sandwiches
- ▶ Cool Sandwiches
- ▶ Warm Sandwiches
- ▶ Snacks
- ▶ Drinks

Sprout Cafe 1/12

Small Bites

EDIT CATEGORY

| | |
|--|--|
| <p>Summer Rolls \$5.90</p> <p style="font-size: 0.8em;">With mixed green lettuce, somen noodles, carrots, bean sprouts, caramelized shallots, a...</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Guacamole \$8.03</p> <p style="font-size: 0.8em;">With sour cream and homemade chips.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Fried Calamari \$9.10</p> <p style="font-size: 0.8em;">With chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Garlic Sweet Potato Fries \$7.49</p> <p style="font-size: 0.8em;">With chipotle ketchup and chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Pork Potstickers \$8.03</p> <p style="font-size: 0.8em;">6 pieces. With chili soy ginger sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Bulgogi Lettuce Cups \$6.96</p> <p style="font-size: 0.8em;">2 pieces. With rib eye steak, Gochujang sauce, crushed peanuts, carrots, and cilantro.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Shoestring Fries \$5.50</p> <p style="font-size: 0.8em;">Chipotle ketchup, chili lime aioli</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Sesame Tofu \$6.42</p> <p style="font-size: 0.8em;">Crispy tofu in a tangy sesame sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |

Hummus Click \$ 0.00

House made hummus served with warm pita

+ ADD A MODIFIER & DETAILS

Cancel
Save

ADD AN ITEM

➔ Sprout Cafe 1/12

MENU SORT

- ▶ Small Bites
- ▶ Signature Soups
- ▶ Entrées
- ▶ Build Your Own Salad
- ▶ Signature Salads
- ▶ Grilled Cheese Sandwiches
- ▶ Cool Sandwiches
- ▶ Warm Sandwiches
- ▶ Snacks
- ▶ Drinks

Sprout Cafe 1/12

Small Bites

EDIT CATEGORY

| | |
|--|--|
| <p>Summer Rolls \$5.90</p> <p style="font-size: 0.8em;">With mixed green lettuce, somen noodles, carrots, bean sprouts, caramelized shallots, a...</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Guacamole \$8.03</p> <p style="font-size: 0.8em;">With sour cream and homemade chips.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Fried Calamari \$9.10</p> <p style="font-size: 0.8em;">With chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Garlic Sweet Potato Fries \$7.49</p> <p style="font-size: 0.8em;">With chipotle ketchup and chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Pork Potstickers \$8.03</p> <p style="font-size: 0.8em;">6 pieces. With chili soy ginger sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Bulgogi Lettuce Cups \$6.96</p> <p style="font-size: 0.8em;">2 pieces. With rib eye steak, Gochujang sauce, crushed peanuts, carrots, and cilantro.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Shoestring Fries \$5.50</p> <p style="font-size: 0.8em;">Chipotle ketchup, chili lime aioli</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Sesame Tofu \$6.42</p> <p style="font-size: 0.8em;">Crispy tofu in a tangy sesame sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |

Hummus
\$ T

House made hummus served with warm pita

+ ADD A MODIFIER & DETAILS

Cancel
Save

ADD AN ITEM

➔ Sprout Cafe 1/12

MENU SORT

- ▶ Small Bites
- ▶ Signature Soups
- ▶ Entrées
- ▶ Build Your Own Salad
- ▶ Signature Salads
- ▶ Grilled Cheese Sandwiches
- ▶ Cool Sandwiches
- ▶ Warm Sandwiches
- ▶ Snacks
- ▶ Drinks

Sprout Cafe 1/12

Small Bites

EDIT CATEGORY

| | |
|--|--|
| <p>Summer Rolls \$5.90</p> <p style="font-size: 0.8em;">With mixed green lettuce, somen noodles, carrots, bean sprouts, caramelized shallots, a...</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Guacamole \$8.03</p> <p style="font-size: 0.8em;">With sour cream and homemade chips.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Fried Calamari \$9.10</p> <p style="font-size: 0.8em;">With chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Garlic Sweet Potato Fries \$7.49</p> <p style="font-size: 0.8em;">With chipotle ketchup and chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Pork Potstickers \$8.03</p> <p style="font-size: 0.8em;">6 pieces. With chili soy ginger sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Bulgogi Lettuce Cups \$6.96</p> <p style="font-size: 0.8em;">2 pieces. With rib eye steak, Gochujang sauce, crushed peanuts, carrots, and cilantro.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Shoestring Fries \$5.50</p> <p style="font-size: 0.8em;">Chipotle ketchup, chili lime aioli</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Sesame Tofu \$6.42</p> <p style="font-size: 0.8em;">Crispy tofu in a tangy sesame sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |

Hummus
\$ 7.00

House made hummus served with warm pita

+ ADD A MODIFIER & DETAILS

Cancel
Save

ADD AN ITEM

➔ Sprout Cafe 1/12

MENU SORT

- ▶ Small Bites
- ▶ Signature Soups
- ▶ Entrées
- ▶ Build Your Own Salad
- ▶ Signature Salads
- ▶ Grilled Cheese Sandwiches
- ▶ Cool Sandwiches
- ▶ Warm Sandwiches
- ▶ Snacks
- ▶ Drinks

Sprout Cafe 1/12

Small Bites

EDIT CATEGORY

| | |
|--|--|
| <p>Summer Rolls \$5.90</p> <p style="font-size: 0.8em;">With mixed green lettuce, somen noodles, carrots, bean sprouts, caramelized shallots, a...</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Guacamole \$8.03</p> <p style="font-size: 0.8em;">With sour cream and homemade chips.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Fried Calamari \$9.10</p> <p style="font-size: 0.8em;">With chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Garlic Sweet Potato Fries \$7.49</p> <p style="font-size: 0.8em;">With chipotle ketchup and chili lime aioli.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Pork Potstickers \$8.03</p> <p style="font-size: 0.8em;">6 pieces. With chili soy ginger sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Bulgogi Lettuce Cups \$6.96</p> <p style="font-size: 0.8em;">2 pieces. With rib eye steak, Gochujang sauce, crushed peanuts, carrots, and cilantro.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |
| <p>Shoestring Fries \$5.50</p> <p style="font-size: 0.8em;">Chipotle ketchup, chili lime aioli</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> | <p>Sesame Tofu \$6.42</p> <p style="font-size: 0.8em;">Crispy tofu in a tangy sesame sauce.</p> <p style="font-size: 0.7em; text-align: right;">EDIT</p> |

Hummus
\$ 7.00

House made hummus served with warm pita

+ ADD A MODIFIER & DETAILS

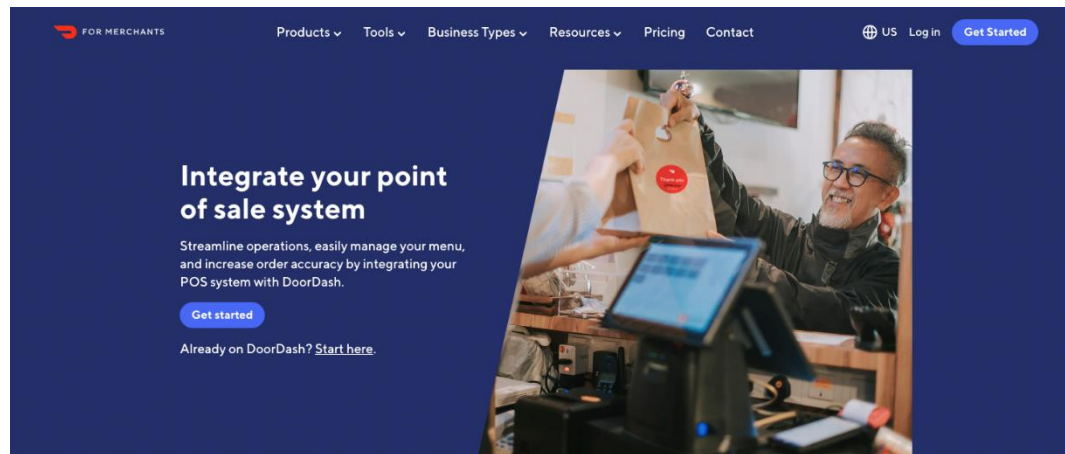
Cancel
Save

ADD AN ITEM

Integration Partners

DoorDash has “partners” that offer direct integration with the DoorDash platform. These third parties provide a means in their respective UIs to connect to the merchant’s DoorDash account.

There appear to be around 57 existing partners that represent different POS systems (Aloha, Square, Toast) and aggregators (All Day Kitchens, Chowly, MealCo). Within the category of POS integration there are two main features that a merchant can take advantage of: menu management and order fulfillment. Below are screenshots from the DoorDash Merchant portal listing the various DoorDash integration partners (see Exh 36, 37).



DoorDash Integration Partners

Search for an Integration

All Categories All Countries

| | | |
|--|--|-----------------------------|
| Chowly Aggregator / Middleware | Clover POS System | Crisp POS System |
| Cuboh Aggregator / Middleware | Deliverect Aggregator / Middleware | Doshii POS System |

< 1 2 3 4 5 ... 9 >

DoorDash Integration Partners

Search for an Integration

All Categories All Countries

| | | |
|--|--|-----------------------------|
| Chowly Aggregator / Middleware | Clover POS System | Crisp POS System |
| Cuboh Aggregator / Middleware | Deliverect Aggregator / Middleware | Doshii POS System |

< 1 2 3 4 5 ... 9 >

DoorDash Integration Partners

Search for an Integration

All Categories All Countries

| | | |
|----------------------------------|------------------------------------|-------------------------------------|
| Focus POS POS System | FuturePOS POS System | GiveX POS System |
| HarborTouch POS System | Heartland POS POS System | HungerRush POS POS System |

< 1 2 3 4 5 ... 9 >

DoorDash Integration Partners

Search for an Integration

All Categories All Countries

| | | |
|---|---|-------------------------------------|
| ItsaCheckmate Aggregator / Middleware | KBOX Global Aggregator / Middleware | Local Kitchens POS System |
| MealCo Aggregator / Middleware | Mentum Aggregator / Middleware | Microworks POS System |

< 1 ... 3 4 5 ... 9 >

DoorDash Integration Partners

Search for an Integration

All Categories All Countries

| | | |
|--|---|--|
| Mobi2go Aggregator / Middleware | MYR POS Aggregator / Middleware | Nextbite/Ordermark Aggregator / Middleware |
| Novadine Aggregator / Middleware | Olo Aggregator / Middleware | Omnivore Aggregator / Middleware |

< 1 ... 4 5 6 ... 9 >

DoorDash Integration Partners

Search for an Integration

All Categories All Countries

| | | |
|---|---|------------------------------------|
| Ordermark Aggregator / Middleware | offer Otter Aggregator / Middleware | Paytouch POS System |
| Paytronix Aggregator / Middleware | POSitouch POS System | Precision POS POS System |

< 1 ... 5 6 7 ... 9 >

DoorDash Integration Partners

Q Search for an Integration

All Categories

All Countries

| | | |
|--|--|---|
| QuBeyond POS System | Recipe Aggregator / Middleware | Redcat Aggregator / Middleware |
| REEF Aggregator / Middleware | Restaurant Manager POS System | RRT (Restaurant Revolution...) POS System |

< 1 ... 5 6 7 8 9 >

DoorDash Integration Partners

Q Search for an Integration

All Categories

All Countries

| | | |
|--|---------------------------------------|---|
| Square POS System | Squirrel Systems POS System | Tacit Aggregator / Middleware |
| Tillster Aggregator / Middleware | Toast POS System | Tyro Aggregator / Middleware |

< 1 ... 5 6 7 8 9 >

DoorDash Integration Partners

Q Search for an Integration

All Categories

All Countries

| | | |
|--|---|-----------------------------|
| UEAT Aggregator / Middleware | VisualTouch POS Aggregator / Middleware | Xenial POS System |
| Xoikos POS System | YourFare Aggregator / Middleware | |

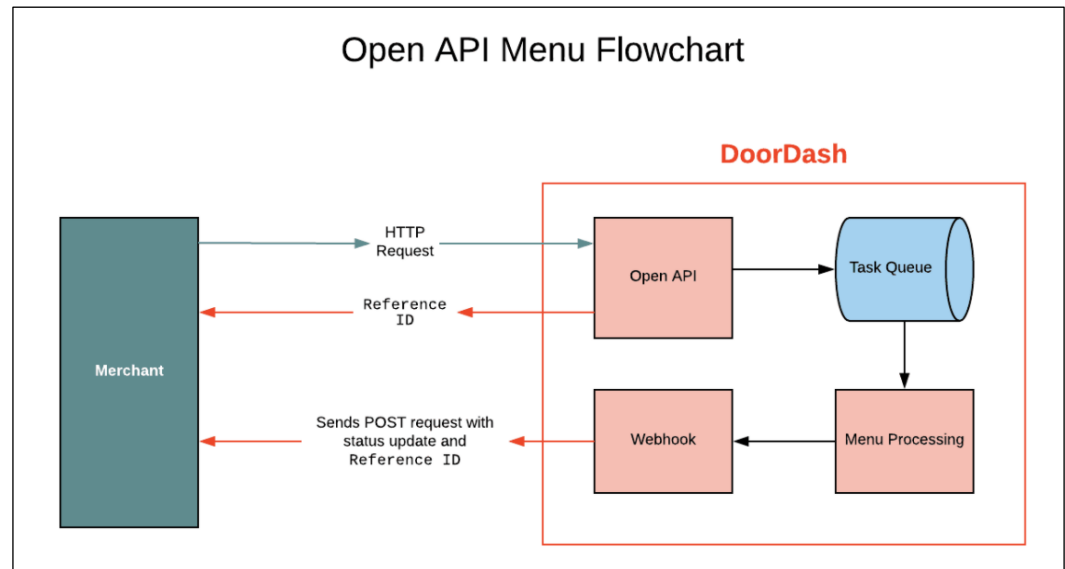
< 1 ... 5 6 7 8 9 >

Menu Management

Menu management automatically synchronizes the merchant’s menu with the DoorDash platform. This is a one-way synchronization from the POS system to DoorDash. Not all features of DoorDash’s menu manager are supported for all merchants. In some cases, the menu items will sync over but the sort order will not. The merchant is still able to utilize the menu manager to adjust imported items. Order fulfillment can happen through one of four different order protocols: direct POS integration, middleware integration, tablet (DoorDash Order Manager app), or email (see Exh. 31, 32)

Updating menus on the DoorDash platform involves both Push (partner to DoorDash) and Pull (DoorDash to partner). When a partner is first added to the DoorDash platform, a request is issued to the partner’s server to retrieve a merchant’s menu. The partner’s server must respond with the full menu in its body. Whenever there are changes to a

merchant's menu, the partner can then issue a request to DoorDash's server, pushing the new menu. In both push and pull scenarios, once the menu has been processed by the DoorDash platform, a request is issued to the partner server indicating the success or fail (see Exh. 38).



POS Integration

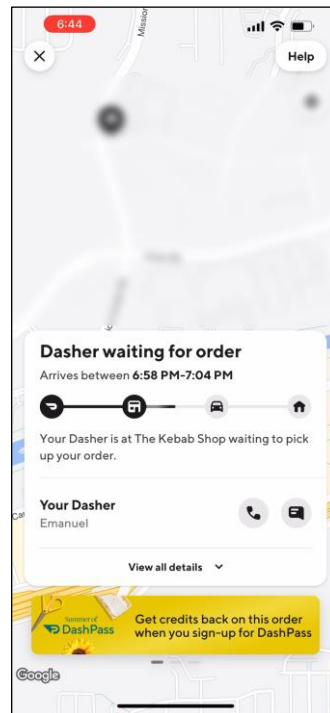
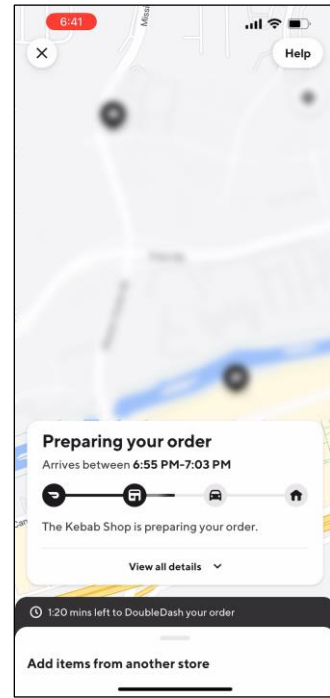
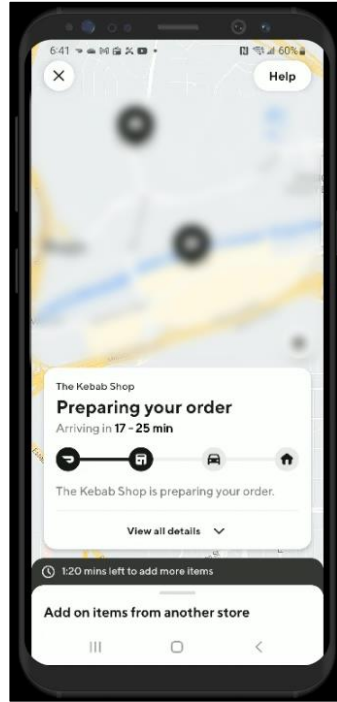
Two popular Point of Sale systems (OS) are Square and Clover. DoorDash provides the means for merchants to integrate with these POSs to fulfill DoorDash orders on said POS. Through this integration, merchants can manage their store menus (published on the DoorDash system) which included adding, editing, deleting, and updating menu item, menu photos, prices and price adjustment, special instructions, etc. Furthermore, through the integration merchants can directly interact with their POS to manage all aspects of DoorDash orders including fulfillment, refunds, cancellations, prep times, schedules, tips, etc. (See Exh 81, 82)

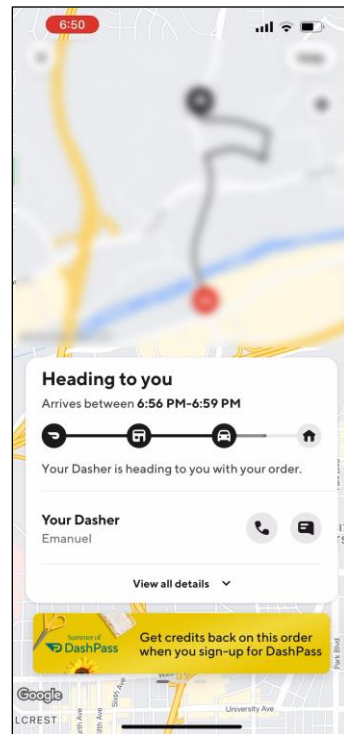
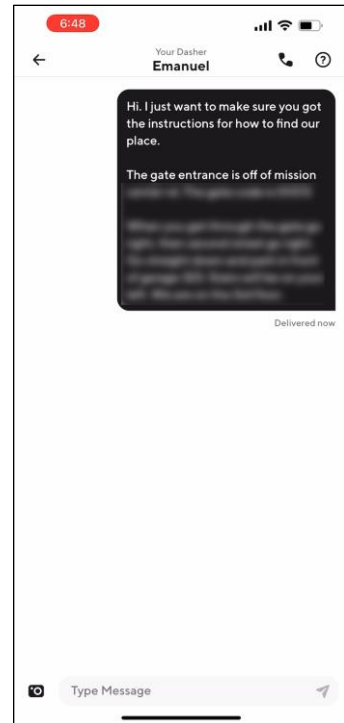
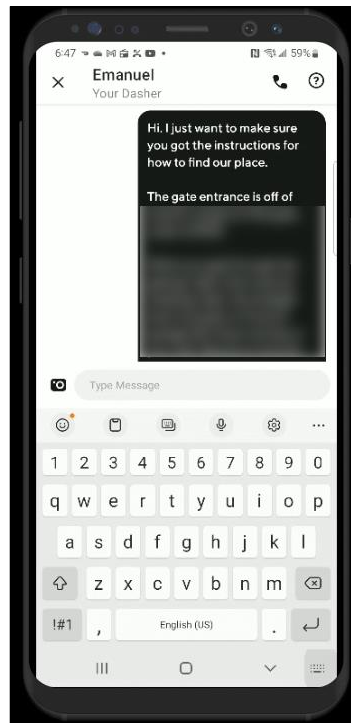
Order Tracking and Notification

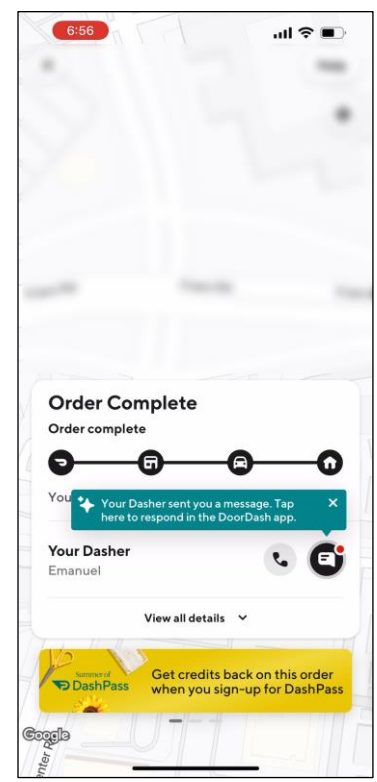
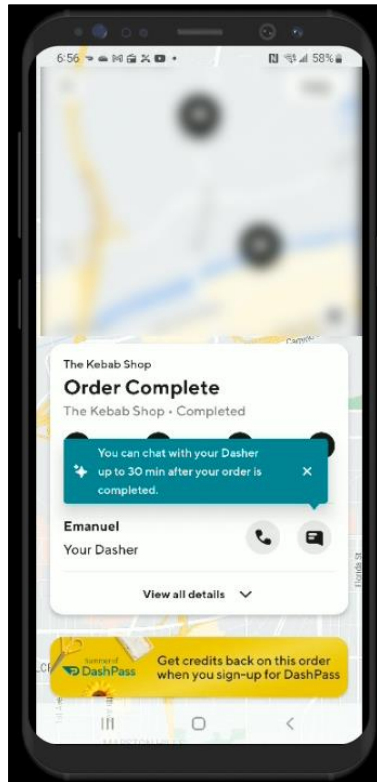
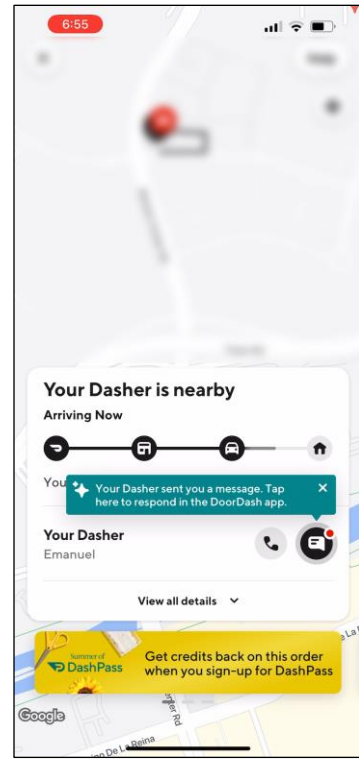
The same holds true for order notifications, order tracking and texting/chatting between the dasher and user. Below are a series of screens take from one iPhone and one Android each tracking the same order placed by a user in the Android App. They depict the notifications and updates in the app for the user to be able to track their order. In this case, we have one iPhone and one Android device tracking the same order, we can see the intelligent decisions being made by the Dispatcher regarding estimated time of delivery and dasher time and distance from fulfilling the order. We also see text messages placed and sent by the user to the Dasher on the android app, being reflected in the iOS app on an iPhone that is tracking the order. When the Dasher responds to the chat from the user, the user is notified on the android phone and on the iOS phone.

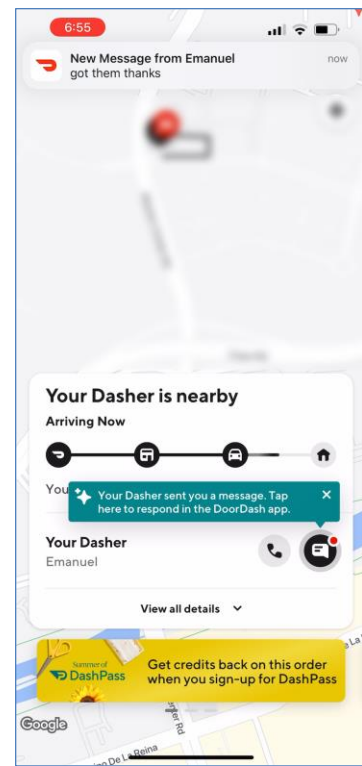
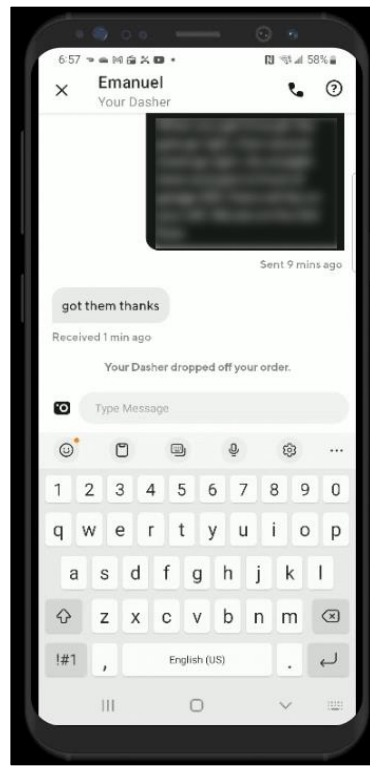
Android DoorDash App

iOS DoorDash App









DoorDash Tablet (wireless handheld computer)

Restaurants use the DoorDash’s custom app called **DoorDash Tablet** to receive and manage customer orders, manage their menus, manage hours of operation and kitchen status, and chat with DoorDash support. The following screenshots are taken from the DoorDash document “Your DoorDash Tablet A Step-by-Step Guide.pdf” (see Exh 88) and depict a step by step process as to how restaurants/merchants/entities use the tablet app:



SET UP YOUR TABLET

Turn on your tablet

Turn your tablet on by finding and pressing the power button.

- 1 A screen will appear with a DoorDash icon and a WiFi icon. Tap the **DoorDash** icon.
- 2 On the next screen, tap the **Let's Get Started** button to begin.

DD_258818

Welcome to DoorDash

We're so happy to have you onboard. Your store is almost ready to go live.

Let's Get Started

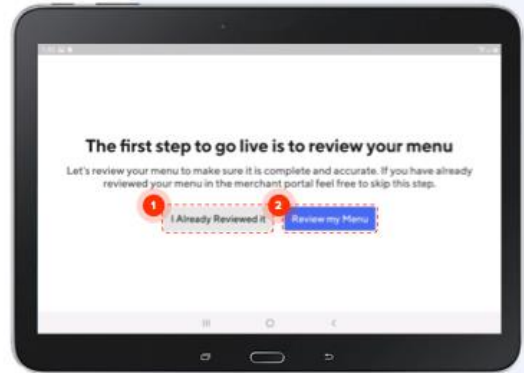
DoorDash Android App Order

SET UP YOUR TABLET

Review your menu

Next you'll be asked to review your menu, which reflects the menu located in the Merchant Portal.

- 1 If you've already reviewed your menu on the portal, tap **I Already Reviewed It**.
- 2 If you haven't reviewed it yet or don't know what that means, tap **Review My Menu**.



7

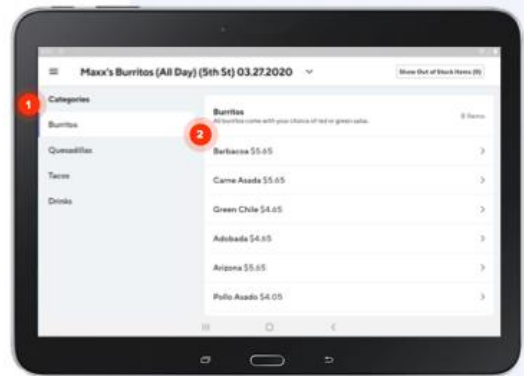
DoorDash Android App Order

SET UP YOUR TABLET

Review your menu

Your menu categories will be listed on the left-hand side of the screen.

- 1 Tap a category to view all the items in that category at once.
- 2 Tap an individual item to see its details.



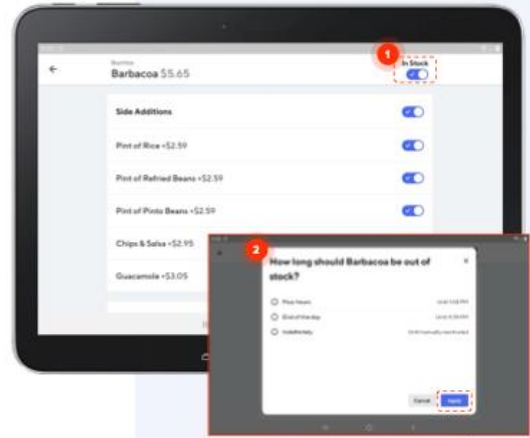
8

SET UP YOUR TABLET

Set item availability

Tapping into an individual item will display the options associated with that item.

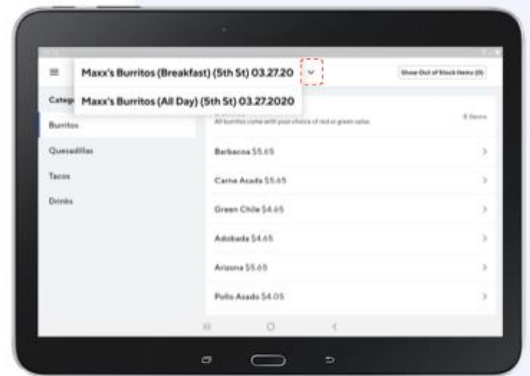
- 1 If any of these optional items are out of stock, switch the **In Stock** toggle next to the item.
- 2 Select the period of time you want an item to be listed as out of stock, then tap **Apply**. Once your store goes live, you'll be able to edit these settings at any time using your tablet.



SET UP YOUR TABLET

Manage multiple menus

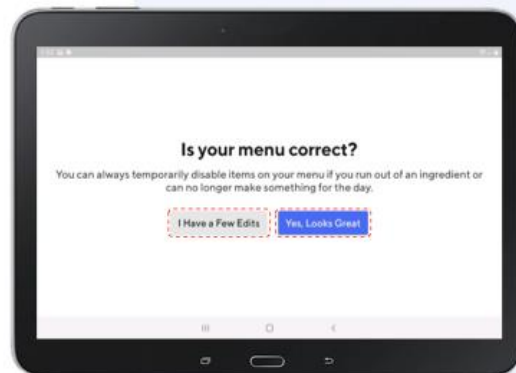
If you have different menus for different times of day or days of the week, you can access them using the **dropdown icon** next to your menu name.



SET UP YOUR TABLET

Approve your menu

Once you've reviewed your menu, tap **I Have a Few Edits** if you still want to make edits. If you have no further changes, tap **Yes, Looks Great**.



11

SET UP YOUR TABLET

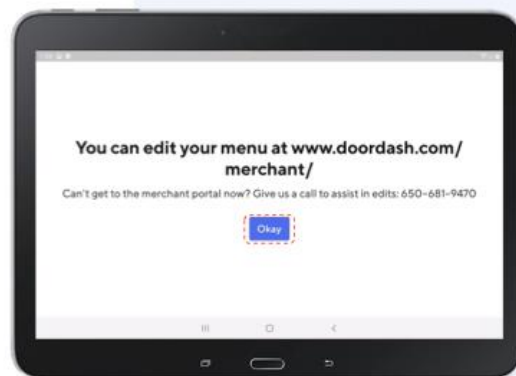
Edit your menu

Need to make additional edits to your menu like descriptions, options, and prices? The fastest and easiest way is by using the Menu Editor tool in the [Merchant Portal](#).

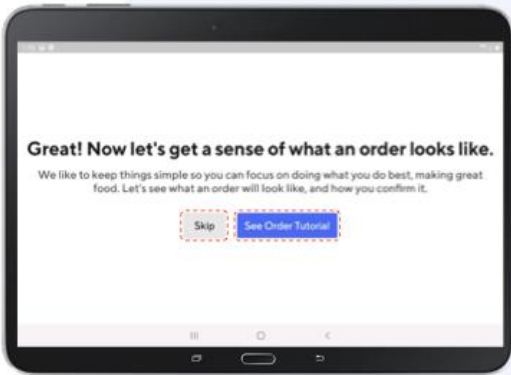
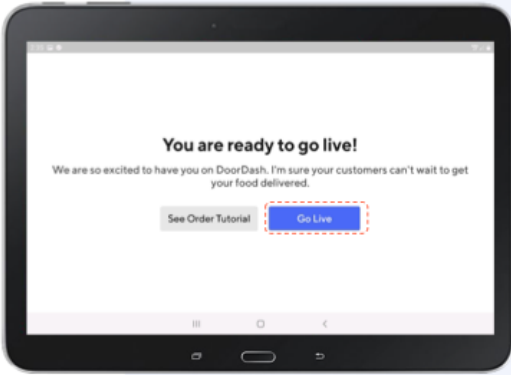
Review the Merchant Portal guide for step-by-step instructions on how to update your menu.

If you're stuck, you can call Support and talk to one of our agents.

Tap **Okay** to move on to the next step.



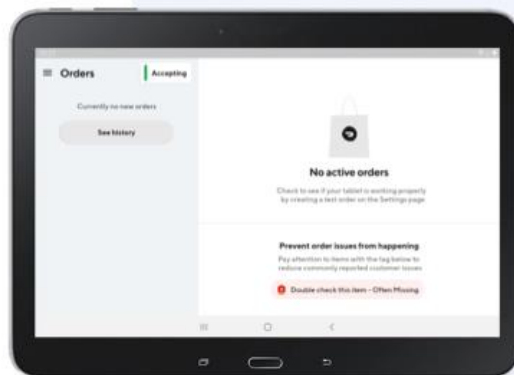
12

| | |
|--|--|
| | <p>SET UP YOUR TABLET</p> <h2>Take a quick order tutorial</h2> <p>If you'd like to see an overview on how orders work on your tablet, tap See Order Tutorial. If not, tap Skip to continue the setup.</p>  <p>Great! Now let's get a sense of what an order looks like.</p> <p>We like to keep things simple so you can focus on doing what you do best, making great food. Let's see what an order will look like, and how you confirm it.</p> <p>Skip See Order Tutorial</p> <p>13</p> |
| | <p>SET UP YOUR TABLET</p> <h2>Go live</h2> <p>It's time to go live! Tap Go Live to get your store live on the DoorDash platform. If you go live during your store open hours, customers will be able to find and order from you right away.</p>  <p>You are ready to go live!</p> <p>We are so excited to have you on DoorDash. I'm sure your customers can't wait to get your food delivered.</p> <p>See Order Tutorial Go Live</p> <p>14</p> |

SET UP YOUR TABLET

You're online

Once you see this screen, you're live. Congratulations! Next up: receiving and accepting new orders.



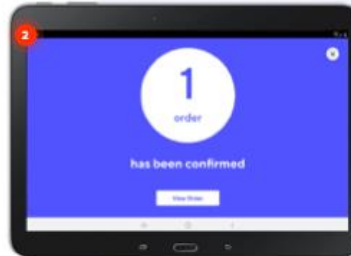
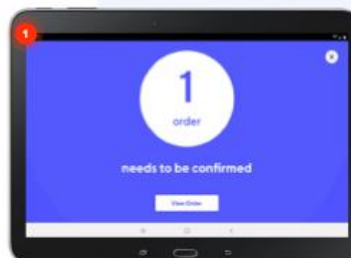
15

RECEIVE AN ORDER

Review & confirm orders

When a customer places an order, you have the option to either:

- 1 **Manually confirm each order:** this allows you to review and confirm the prep time for each order that comes in.
- 2 **Automatically confirm orders as they in:** a Dasher will be assigned right away to come pick up your order. This is a good option for restaurants that have fast prep times.



17

RECEIVE AN ORDER

Manually confirming orders

- 1 Tap **Confirm with # min Prep Time** if the default time frame is accurate.
- 2 Or Tap **Manual Prep Time** if you need want to select less or more time.
- 3 Tap **Confirm Order**.

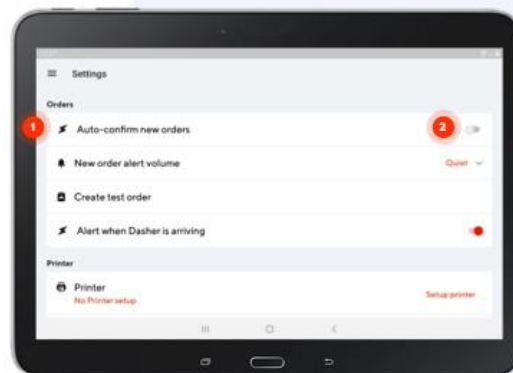
Note: If you've run out of an item, you'll need to report an issue with the order.



RECEIVE AN ORDER

Auto-confirm new orders

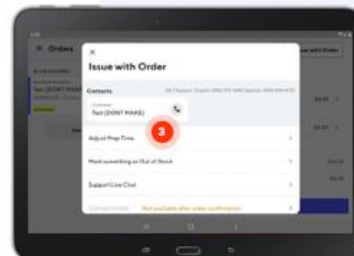
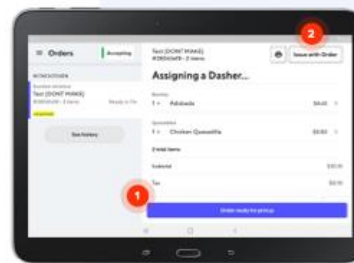
- 1 When **Auto-confirm new orders** is on, new customer orders will automatically be confirmed and a Dasher will be assigned immediately.
- 2 Use the toggle to switch this function on or off.



RECEIVE AN ORDER

Automatically confirm orders

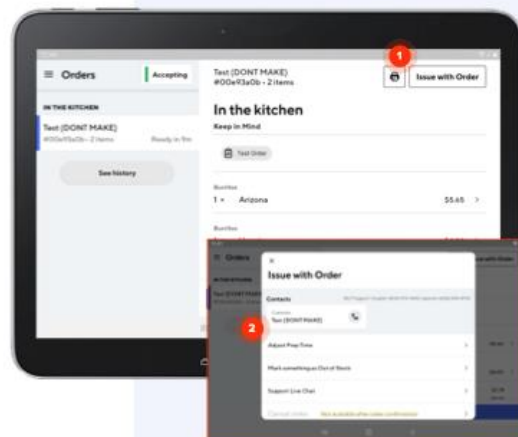
- 1 Tap **Order ready for pickup** once you are ready for Dasher pickup.
- 2 If you need to adjust your prep time, tap **Issue with Order**.
- 3 Tap **Adjust Prep Time** to select your desired order ready time



ADJUST PICKUP TIME

Add more time

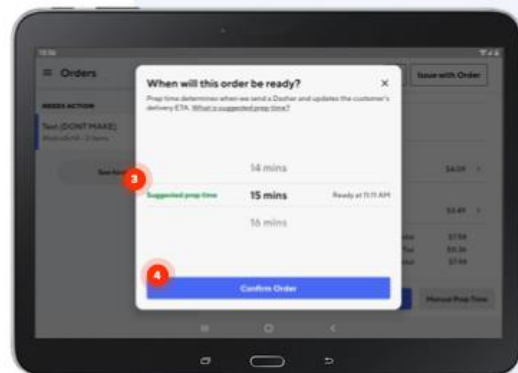
- 1 If an order will take longer than you originally estimated, you can move the pickup time back by tapping **Issue with Order**.
- 2 Select **Adjust Prep Time** in the pop-up menu.



ADJUST PICKUP TIME

Add more time, continued.

- 3 From here, you can choose when the order will be ready for Dasher or customer pickup.
- 4 Tap **Confirm** to update the pickup time

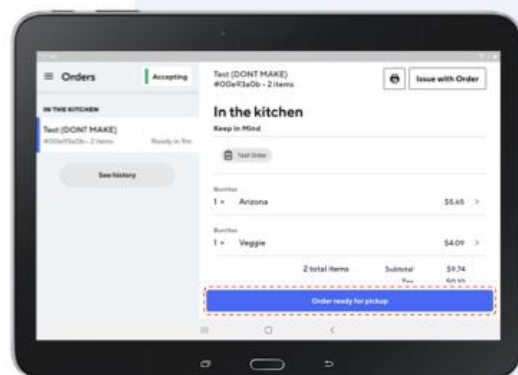


CONFIDENTIAL 25

ORDER READY

Order ready for pickup

If an order is ready before the estimated time you originally selected, simply tap **Order ready for pickup** and we will notify the Dasher to head to your restaurant as soon as possible.



CONFIDENTIAL 26

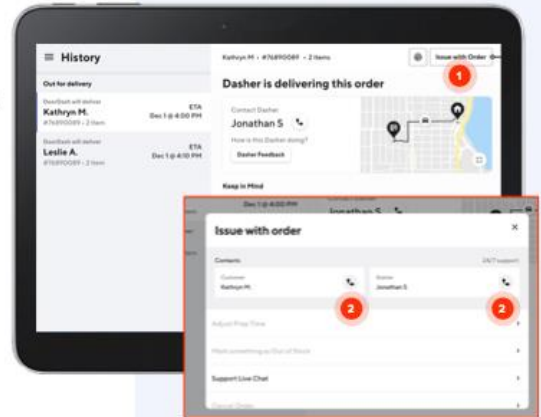
Contacting the Customer or Dasher on Tablet

Contact Customer or Dasher on Live Orders

We understand that at times, questions arise and you may need to contact the customer or Dasher during an active order or after the order has been picked up.

To do so:

- 1 Navigate to **Active Orders** tab or **Order history** tab and tap on **Issue with Order** at the top right.
- 2 Choose either **Customer** or **Dasher** to contact on this screen by **tapping the phone icon**.



CONFIDENTIAL 28

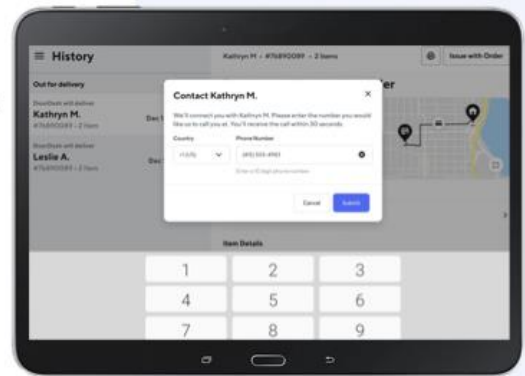
Contacting the Customer or Dasher on Tablet

Contact Customer or Dasher on Live Orders

After you hit the phone icon, you will be prompted to **enter your phone number**. This can be any phone you wish to use to speak with the Customer or Dasher.

After entering your phone number, you will shortly receive a phone call from DoorDash connecting you to the Customer or Dasher.

Note: Your phone number will be masked, the Customer or Dasher will not be able to view your number or call you back.



CONFIDENTIAL 29

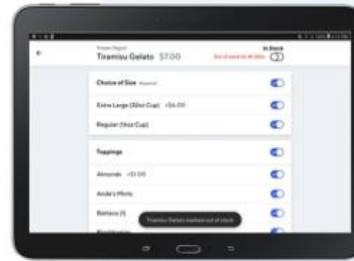
RESOLVE AN OUT OF STOCK ITEM

Resolve an out-of-stock item

When you run out of an item, there are two ways to resolve:

If you proactively notice the item is out of stock before an order is placed, you'll want to update your menu ASAP. Review the following slides, "Mark Items Out-of-Stock".

If you notice the item is out of stock as part of an existing order, you'll need to follow the order issue resolution process. Review the following slides, "Resolving Orders with an Out-of-Stock Item".



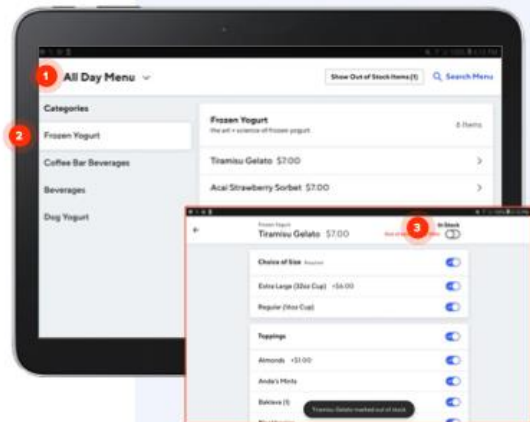
CONFIDENTIAL 31

OUT OF STOCK ITEMS

Mark items out-of-stock

When you notice an item is out of stock, you can remove it from your menu in the **Manage Menu** section of the tablet.

- 1 Click on the **menu** you would like to edit.
- 2 Select the **category** that contains the item(s) you would like to edit.
- 3 Tap the **item** and select the toggle to mark it out of stock. You can select the entire item or just specific modifiers



CONFIDENTIAL 33

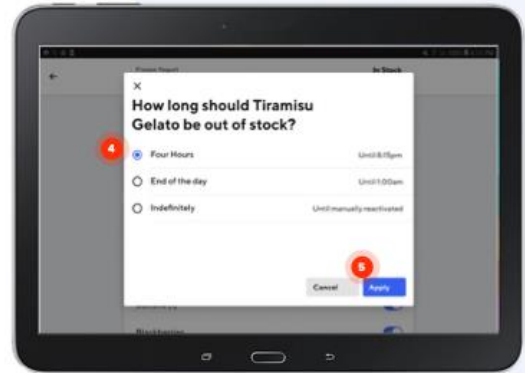
OUT OF STOCK ITEMS

Mark items out-of-stock

Once you update the item to be out of stock, you'll be prompted to specify for how long.

- 4 Select one of the time duration options
- 5 Tap **Apply**

In the example shown, the entire item "Tiramisu Gelato" and related modifiers are marked out of stock for 4 hours.



CONFIDENTIAL 34

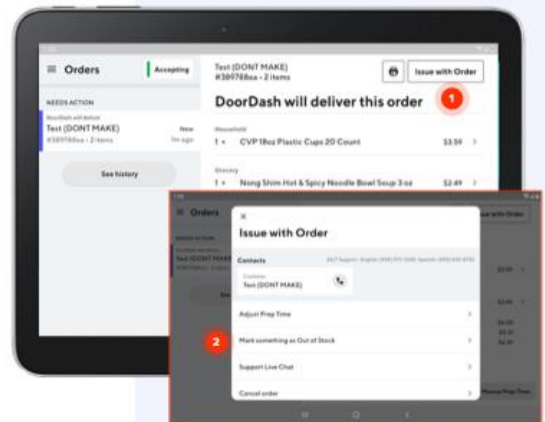
OUT OF STOCK ITEMS

Marking items out-of-stock

If your Customer ordered an item that is out-of-stock, resolve by:

- 1 Tapping **Issue with Order**
- 2 Tapping **Mark something as Out of Stock**

Before checkout, Customers specify how they want out of stock items resolved. You will be prompted to resolve based on the Customer's choice: *Merchant Recommendation*, *Refund*, *Contact Customer*, or *Cancel the Order*.



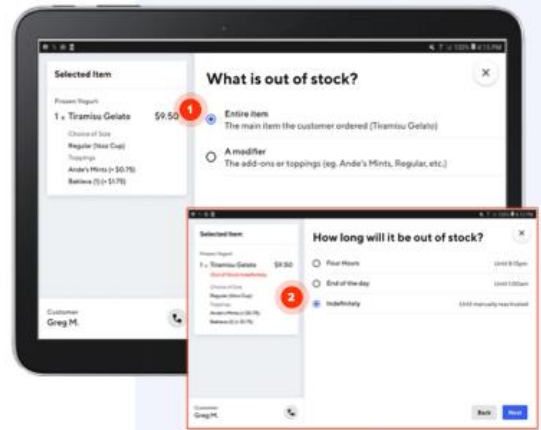
CONFIDENTIAL 36

OUT OF STOCK ITEMS

Merchant Recommendation

If the Customer selected **Merchant Recommendation**, they are leaving the substitution up to your discretion.

- 1 Once you have selected the item with an order issue, you can choose either the **entire item** or **just specific modifiers**.
- 2 You will then be prompted to select how long to deactivate the item or modifier.



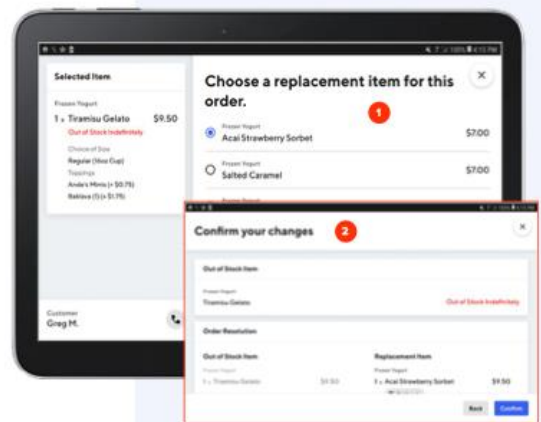
CONFIDENTIAL 37

OUT OF STOCK ITEMS

Merchant Recommendation

Next, you'll be asked to choose a replacement item for the order.

- 1 Select and customize a replacement item for your Customer.
- 2 Confirm your changes.



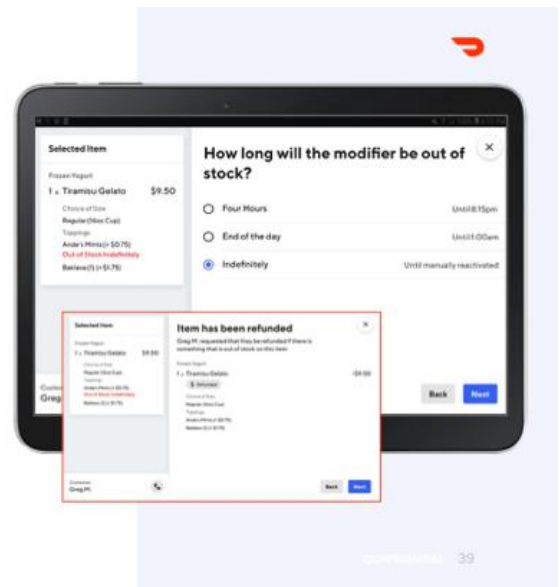
CONFIDENTIAL 38

OUT OF STOCK ITEMS

Refund

If the Customer selected **Refund**, they do not want a replacement and would like to be refunded for the item.

Once you mark the item as out of stock, we will automatically refund your Customer for the related item.



OUT OF STOCK ITEMS

Contact Customer

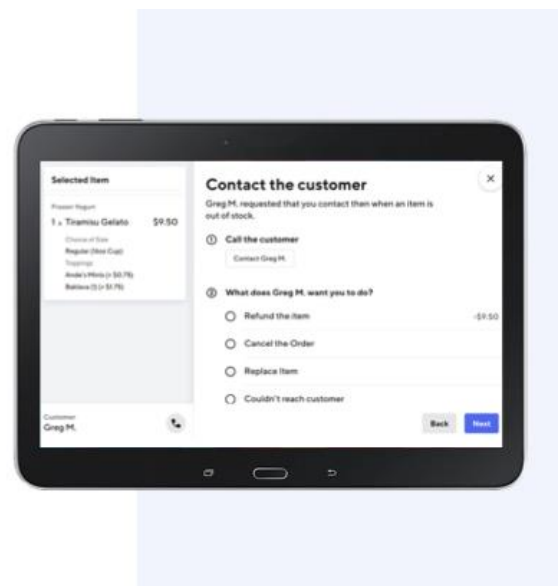
If the Customer selected **Contact Customer**, they want you to reach out to them to decide what to do about the out of stock item.

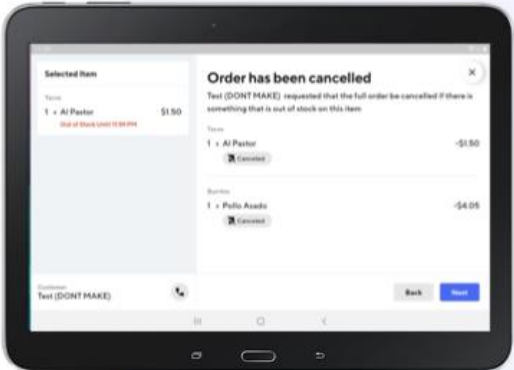
Call them by tapping the 'Contact Name' button in the active orders tab. Then ask how they would like to proceed. They can choose to:

- Receive a refund for the individual item;
- Cancel and refund the order;
- Or replace the item.

Choose the customer's preferred option and tap **Next**.

If you are unable to reach the customer, select **Couldn't reach customer** and then tap **Next**.



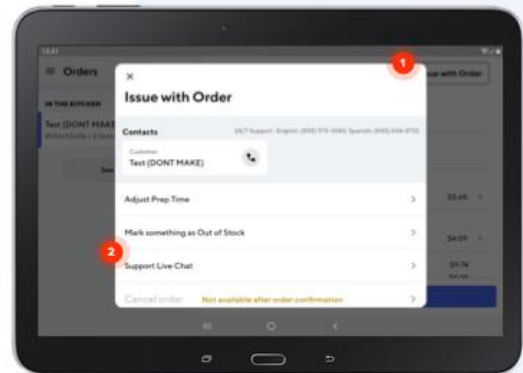
| | |
|--|---|
| | <p>ITEM OUT OF STOCK</p> <h2>Cancel and refund the order</h2> <p>If the Customer selected Cancel and refund the order, they no longer want their order and would prefer a refund.</p> <p>The entire order will be cancelled and the customer to be refunded after you have marked it out of stock for a specific duration.</p>  <p>CONFIDENTIAL 41</p> |
| | <h2>Live Chat with Support</h2> <p>CONFIDENTIAL 42</p> |

LIVE CHAT WITH SUPPORT

Contact support

If you need assistance beyond contacting a customer, adjusting pickup time or marking an item out of stock, you can reach out to the Support team.

- 1 Tap on **Issue with Order** at the top right.
- 2 Tap **Support Live Chat** to open a chat window with our Support team. This feature is accessible for all orders that are currently in progress or have already been picked up.

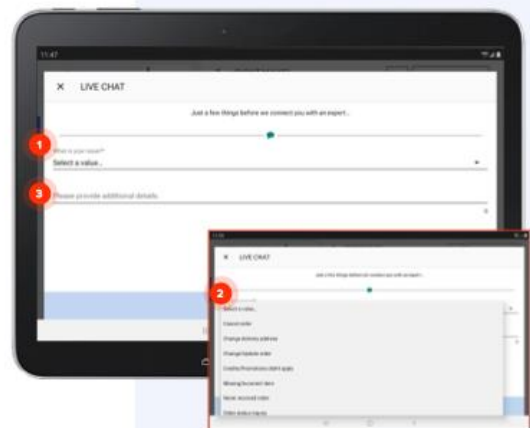


CONFIDENTIAL 43

LIVE CHAT WITH SUPPORT

Contact support, continued.

- 1 Tap **Select a value** to reveal a drop-down menu.
- 2 Select the issue you're experiencing.
- 3 Provide additional details so our Support team can resolve quickly.
- 4 Tap **Chat with an Agent** to start the chat.



CONFIDENTIAL 44

Dasher Arriving

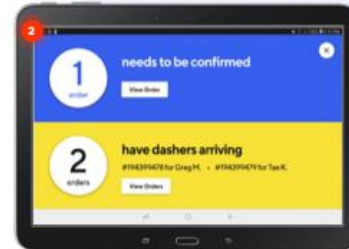
45

DASHER ARRIVING

Dasher is arriving alert

The Dasher Arriving feature will send you a secondary chime and visual cue that will alert you when a **Dasher is 5 minutes away**.

- 1 You will also be able to see which orders have Dashers coming for them (Customer name(s) shown).
- 2 If there are new orders and Dashers arriving at the same time, the screen will show both and the chime for "new order" will alarm.

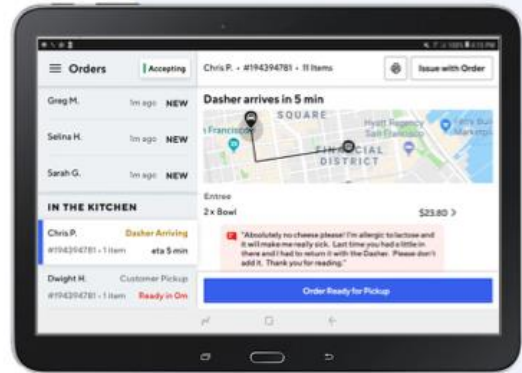


46

DASHER ARRIVING

Dasher is arriving label

Within each order, there will be a Dasher Arriving label with **eta x min** shown, counting down the time until the Dasher arrives.



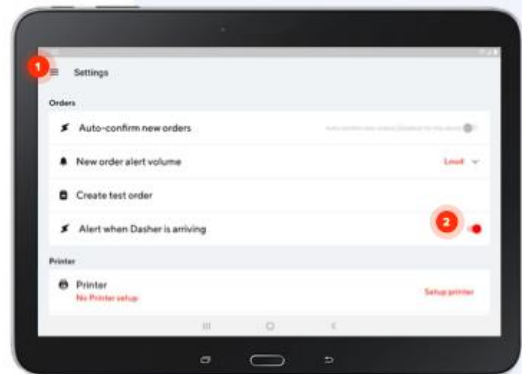
CONFIDENTIAL 47

DASHER ARRIVING

Dasher notifications

You can update your tablet settings to turn Dasher-related notifications on or off.

- 1 Tap on the hamburger icon to open up the sidebar menu and select **Settings**.
- 2 Toggle the **Alert when Dasher is arriving** button on or off based on your preference.



CONFIDENTIAL 48

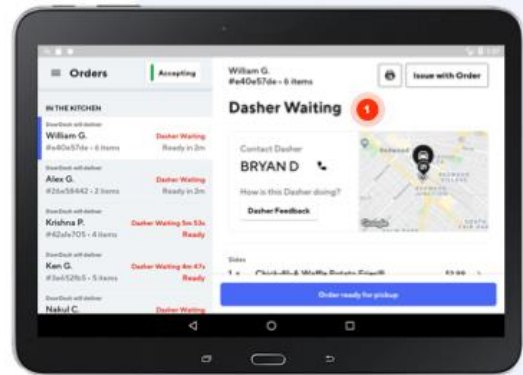
DASHER WAITING

Dasher is waiting status

The Dasher Waiting feature will let you know how long Dashers have been waiting at your store to pick up an order.

- 1 **Dasher Early Arrival:**
We will show a Dasher is waiting as soon as they arrive (Dasher Waiting), but we will wait to start the timer only after your confirmed prep time has lapsed

Note: If the Dasher is early, your order ready time will show in grey (e.g. Ready in 2m)

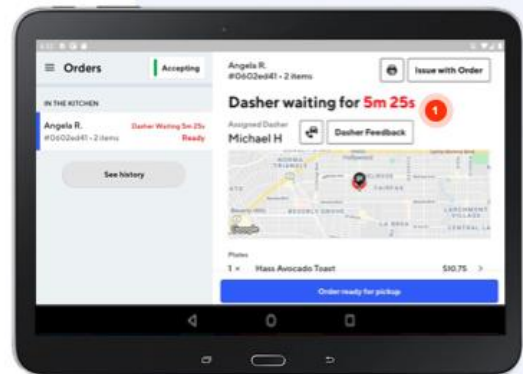


CONFIDENTIAL 50

DASHER WAITING

Dasher is waiting timer

- 1 **Merchant Running Late:**
In this example, the merchant-confirmed prep time has already lapsed (**Ready**); late orders will show **Dasher waiting for Xm XXs** (Dasher wait will align with your Avoidable wait time)



CONFIDENTIAL 51

Change an existing order

In this section, we'll cover how to:

- [Update the status of your kitchen](#)
- [Adjust store hours](#)

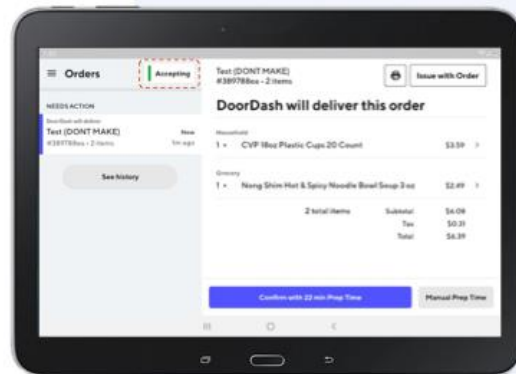
CONFIDENTIAL 53

MANAGE YOUR STORE

Update kitchen status

If you are experiencing a rush in the kitchen, you can update your kitchen status and choose to either slow down the volume of orders you receive or temporarily pause receiving new orders.

To change your kitchen status, tap on the **status button** on your order manager.



CONFIDENTIAL 55

MANAGE YOUR STORE

Update kitchen status

The default setting for your kitchen status is **Normal**.

When customers are on DoorDash, the estimated delivery time they see for your restaurant is partly calculated based on the amount of time it normally takes for your kitchen to prepare an order.



CONFIDENTIAL 56

MANAGE YOUR STORE

Update kitchen status to busy

Setting your kitchen status to **Busy** adjusts the estimated delivery time so you can set accurate expectations with customers before they order.

To set your status as busy, tap **Busy** and select how much extra prep time you need to prepare an order. Then tap **Update Status**.

Tip: If your Kitchen Status is set to Busy, your suggested Prep Time will include the additional prep time you have selected.



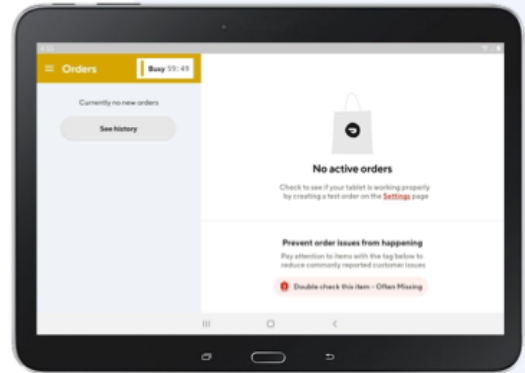
CONFIDENTIAL 57

MANAGE YOUR STORE

Update kitchen status to busy, continued

Your kitchen status will remain as **Busy** for one hour and then automatically shift back to **Normal**.

You can manually change your status to **Normal** any time within that hour by tapping on the **status button**.



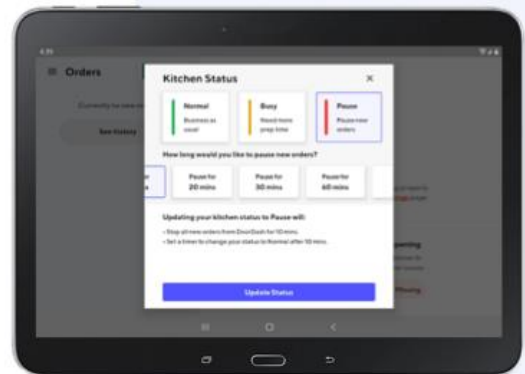
CONFIDENTIAL 58

MANAGE YOUR STORE

Pause new orders

If your kitchen becomes so busy that you are unable to accept more DoorDash orders, you can pause new orders. This means customers will temporarily lose the ability to place new orders, but existing orders will be unaffected.

The maximum amount of time your store can be paused via your tablet is 60 minutes. If you need to pause your store for a longer period, you can do so in the [Merchant Portal](#) or contact Support.

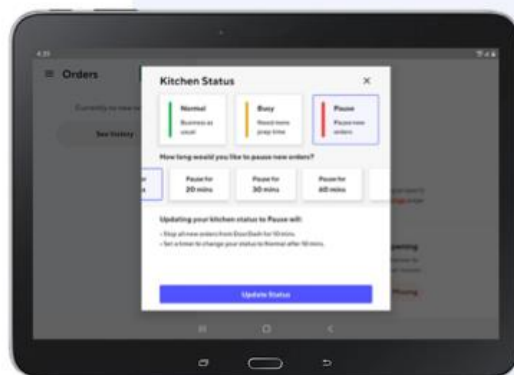


CONFIDENTIAL 59

MANAGE YOUR STORE

Pause new orders

To temporarily stop new orders, tap **Pause**, select the duration of time you want to pause new orders, and tap **Update Status**.

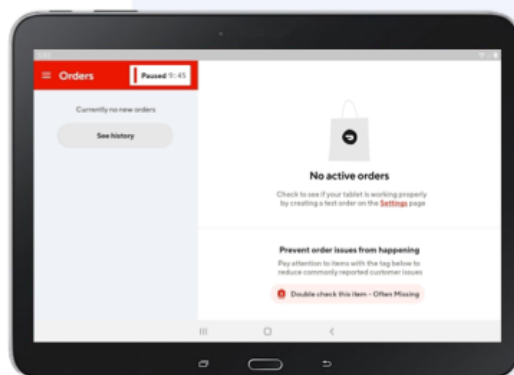


CONFIDENTIAL 60

MANAGE YOUR STORE

Pause new orders

Your store will remain **Paused** for the duration of time you selected and then automatically shift back to **Normal**. You can manually update your status to **Normal** at any time by tapping on the status button.

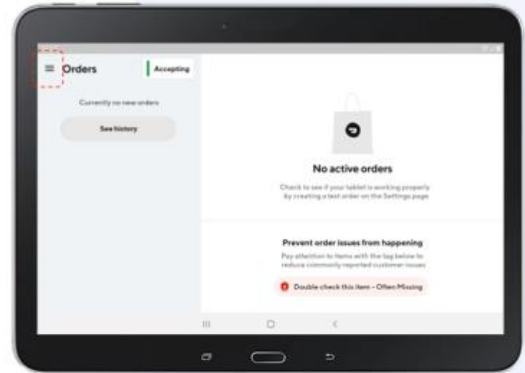


CONFIDENTIAL 61

MANAGE YOUR STORE

Adjust store hours

View your store hours by tapping on the **hamburger icon** to open up the sidebar menu.

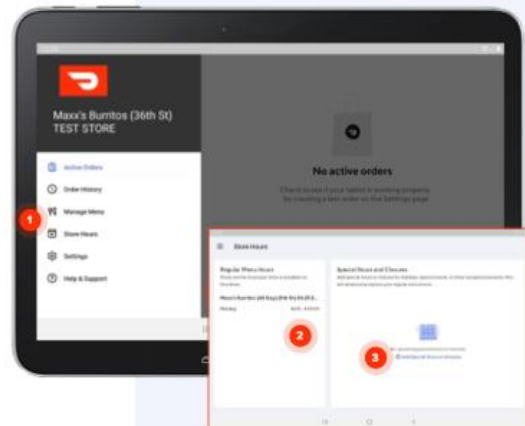


CONFIDENTIAL 63

MANAGE YOUR STORE

Adjust store hours, cont.

- 1 Tap **Store Hours** in the sidebar menu
- 2 Your regular store hours will be listed in the left hand column. If you need to change your store hours *permanently*, visit the [Merchant Portal](#) and follow the instructions in the how to guide.
- 3 If you need to adjust store hours or add a closure for a specific date or dates, these can be done on your tablet. Tap the **Add Special Hours or Closures** link to continue.

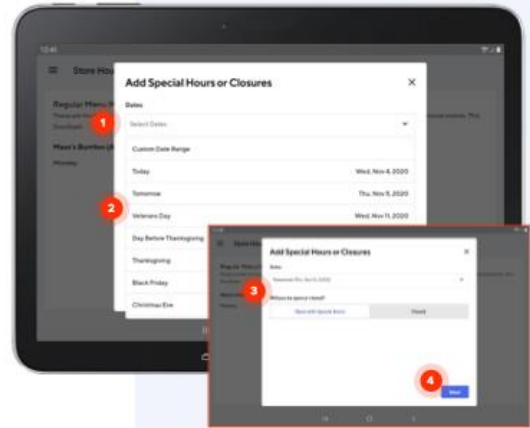


CONFIDENTIAL 64

MANAGE YOUR STORE

Adjust store hours, cont.

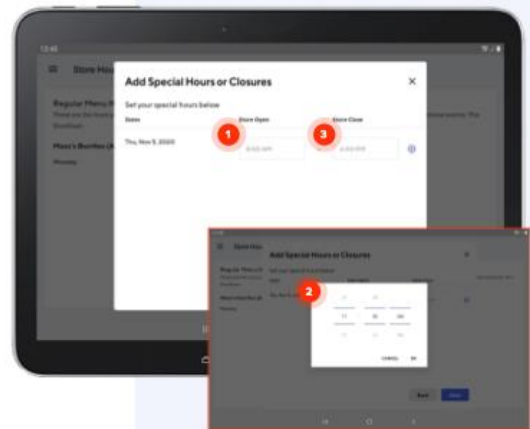
- 1 Tap **Select Dates** in the Dates section
- 2 Choose one of the options in the dropdown menu and tap **Next**
- 3 Choose whether your store will be **Open with Special Hours** or **Closed** for the date selected
- 4 Tap **Next** to continue



MANAGE YOUR STORE

Adjust store hours, cont.

- 1 Tap the box under **Store Open**
- 2 Use the pop-up to select the special opening time. Tap **OK**
- 3 Repeat steps 1 & 2 for **Store Close**
- 4 Tap **Next** to continue

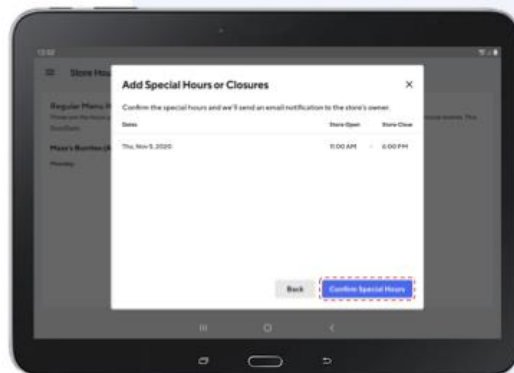


MANAGE YOUR STORE

Adjust store hours, cont.

Check the date and the open and close times to confirm they are correct. Tap **Confirm Special Hours** to finalize.

Your updates will also be reflected in the Merchant Portal under your **Business Hours → Special Closures/Hours** section.

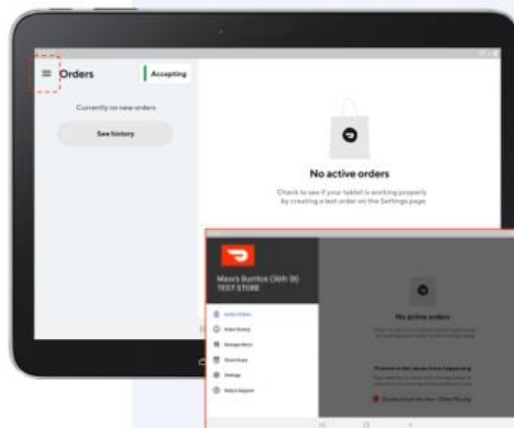


CONFIDENTIAL 67

SIDEBAR MENU

Access your sidebar menu

Access your sidebar menu by tapping on the **hamburger icon**.

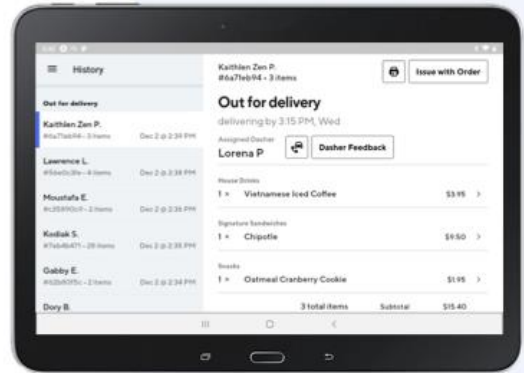


CONFIDENTIAL 70

SIDEBAR MENU

Order history

When you tap **History**, the your most recent orders from the last 7 days (up to 100 orders total) will be displayed. Orders more than 7 days old will not appear here and can only be accessed via the Merchant Portal.

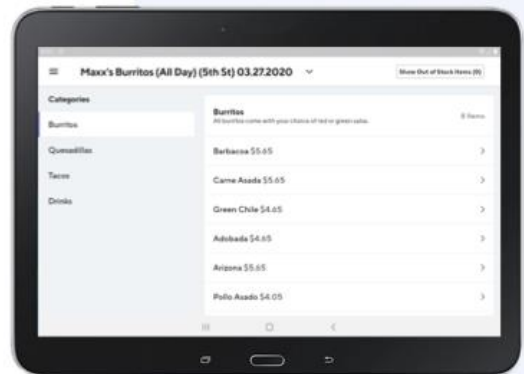


CONFIDENTIAL 72

SIDEBAR MENU

Manage your menu

Refer to the [Review Your Menu](#) section to learn how to review your menu and mark items out of stock.



CONFIDENTIAL 74

SIDEBAR MENU

Special store hours

Refer to the [Adjust Store Hours](#) section to learn how to add special holiday hours and closures. To permanently change your store hours, visit the [Merchant Portal](#).



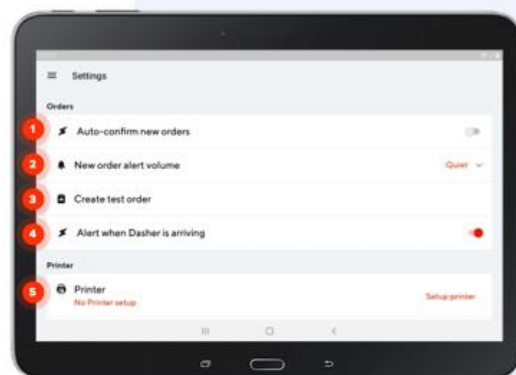
CONFIDENTIAL 76

SIDEBAR MENU

Settings

In the Settings section, you can manage multiple settings, including:

- 1 Turning the automatic confirm of new orders on or off
- 2 Changing the volume of new order alerts
- 3 Creating a test order
- 4 Turning the Dasher arrival alert on or off
- 5 Setting up a printer

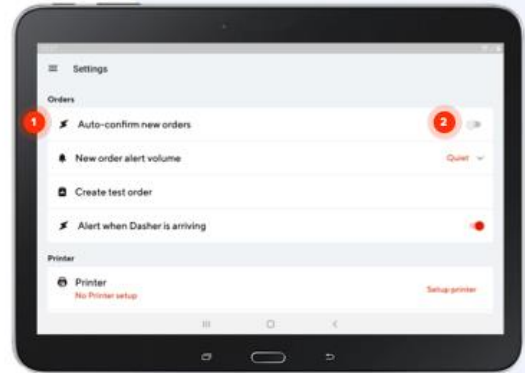


CONFIDENTIAL 78

SIDEBAR MENU

Auto-confirm new orders

- 1 When **Auto-confirm new orders** is on, new customer orders will automatically be confirmed.
- 2 Use the toggle to switch this function on or off.

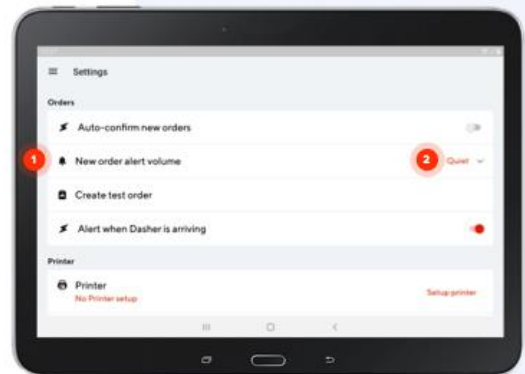


CONFIDENTIAL 79

SIDEBAR MENU

New order alert volume

- 1 When **new order alert volume** is on, your tablet will "ding" every time a new order is received.
- 2 Use the dropdown menu to choose whether the alert is loud or quiet.

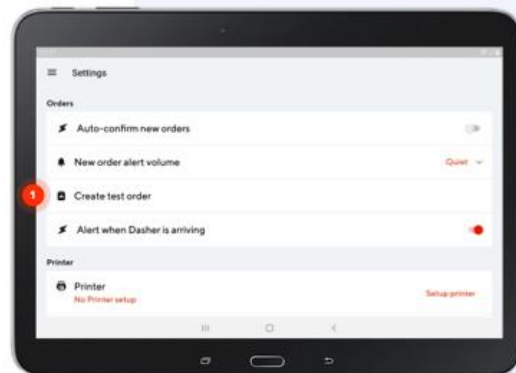


CONFIDENTIAL 80

SIDEBAR MENU

Create a test order

- 1 Creating a test order is a great way to familiarize yourself with order flow. Tap **Create test order** to initiate the test order process.

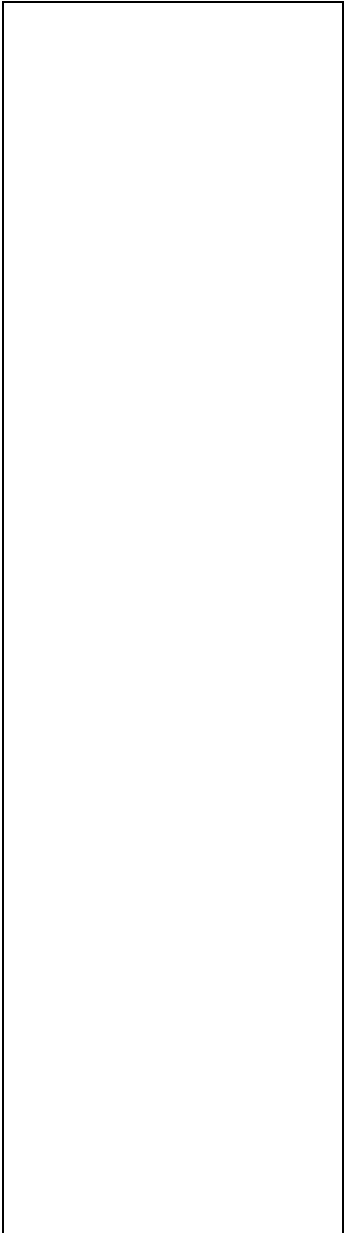


SIDEBAR MENU

Alert when Dasher is arriving

- 1 When the **Alert when Dasher is arriving** is turned on, you will be notified (with a full-screen notification and a unique sound) when a Dasher is arriving to pick up your customer's order
- 2 Use the toggle to switch this function on or off



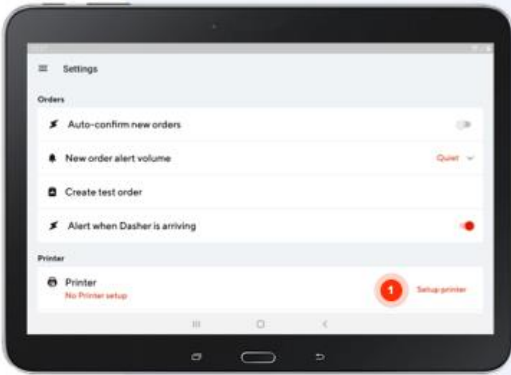


SIDEBAR MENU

Printer setup

If you selected a printer during your sign-up process, you need to pair it with your tablet.

- 1 Ensure your printer is turned on and tap **Setup printer** to begin.

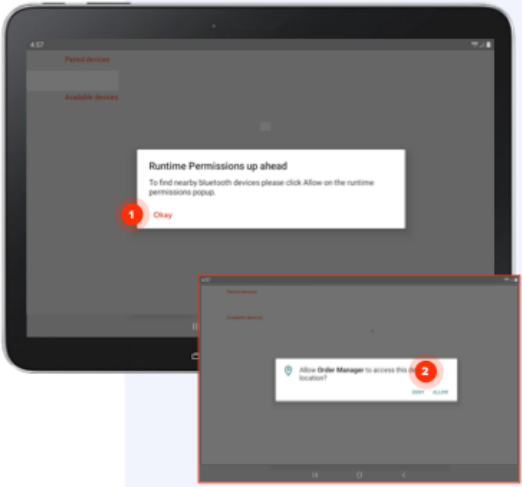


CONFIDENTIAL 84

SIDEBAR MENU

Printer setup

- 1 Tap **Okay** to allow your tablet to find your printer via bluetooth
- 2 Tap **Allow** to continue pairing process

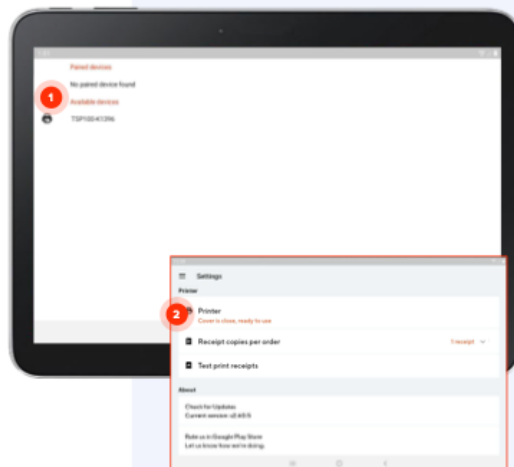


CONFIDENTIAL 85

SIDEBAR MENU

Printer setup

- 1 Under **Available devices**, tap on the printer name to pair it with your tablet
- 2 When your printer and tablet are successfully paired, you will see a **Cover is close, ready to use** message under Printer in Settings.



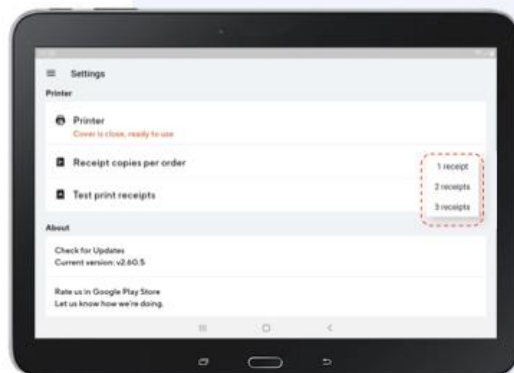
CONFIDENTIAL 86

SIDEBAR MENU

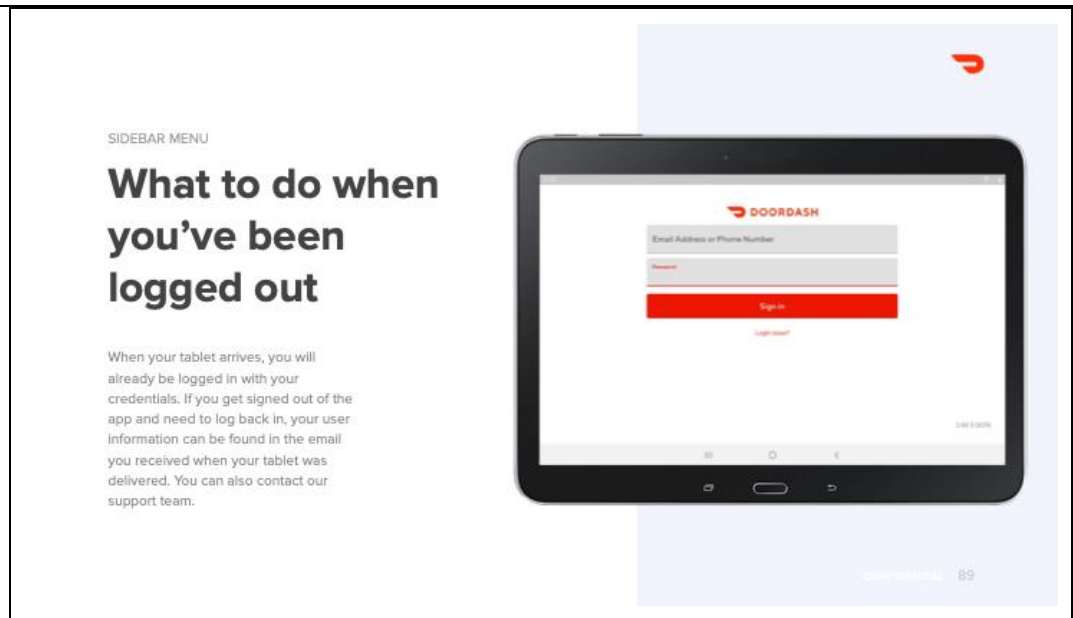
Printer setup

Your printer will now print a receipt of all new orders that you receive.

The default quantity is 1. To have multiple copies printed, tap on the **dropdown** to the right of Receipt copies per order and select how many copies you want printed (up to 3 per order).

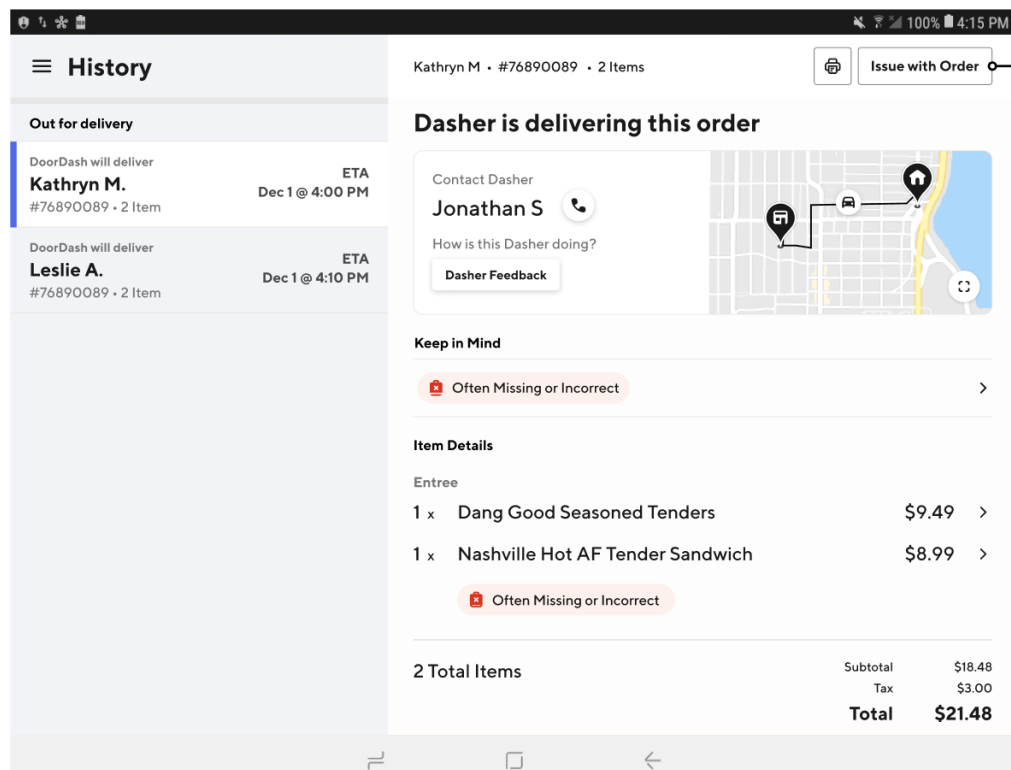


CONFIDENTIAL 87

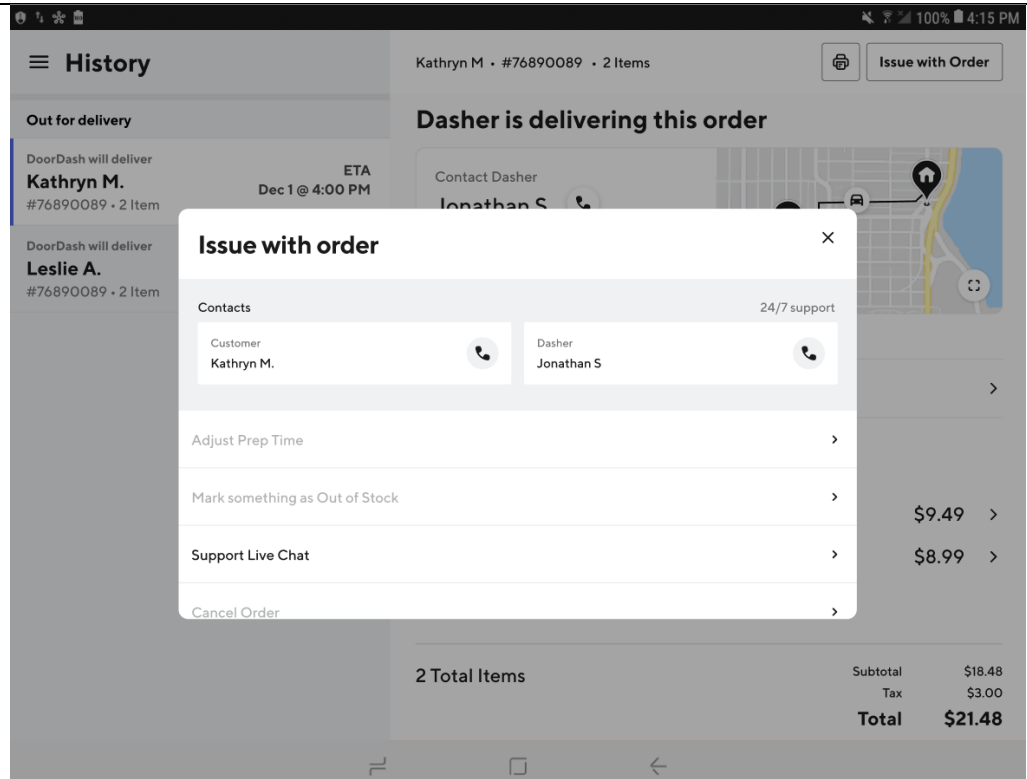


The DoorDash Support article “How can I contact the Customer or Dasher through the tablet” (see Exh 114) instructs the DoorDash merchant how to communicate with a Dasher (delivery driver) or Customer for an order that’s in process or has been picked up, should any questions arise. According to the article, the merchant uses the tablet and:

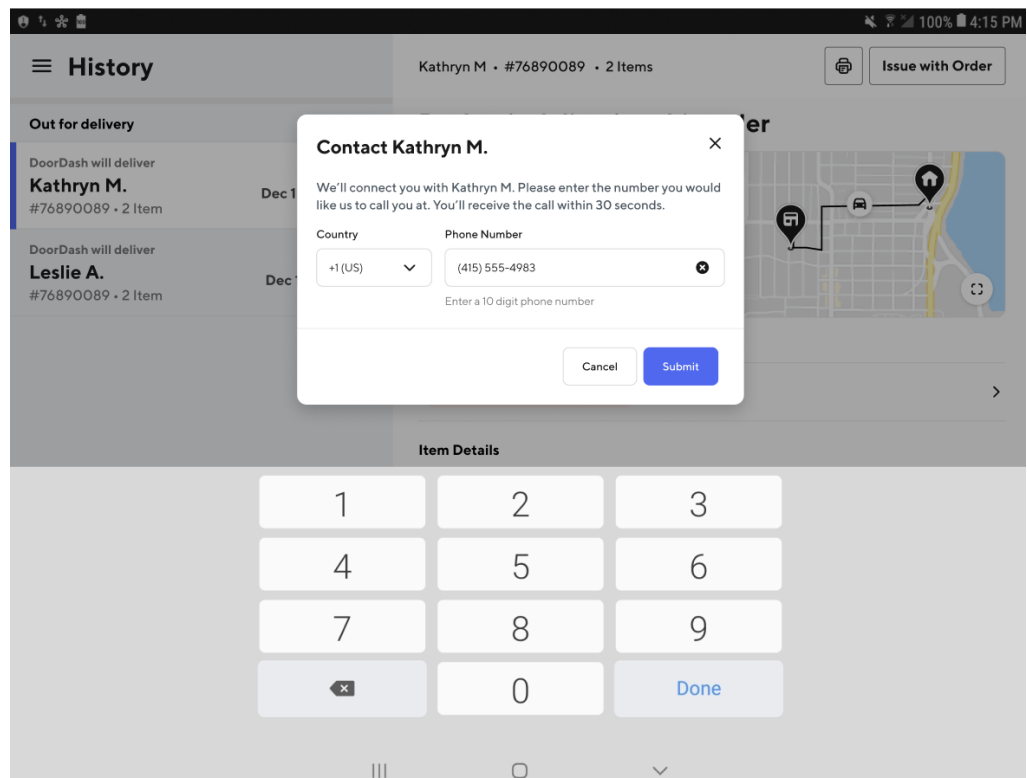
“...On the Order History page, tap on the Issue with Order button in the top right corner.”



“Next, tap on either the Customer or Dasher that you would like to contact.”



“Next, enter the phone number you would like to use to be connected to either the customer or Dasher. Tap Submit and we will call you on the number you enter within 30 seconds to connect your call!”



The DoorDash Merchant Support article *“How can I tell if my Dasher is arriving or waiting”* (see Exh 115), discusses how merchants receive real-time notifications on

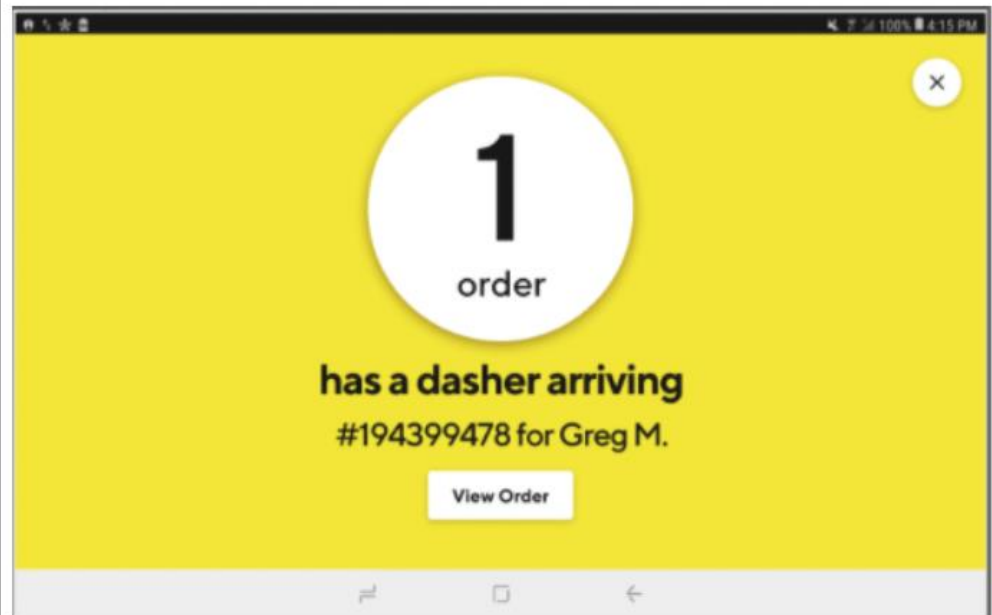
their DoorDash Tablet:

“...when a Dasher is nearby to pickup an order! The new Dasher Arriving feature will send you a secondary chime and visual cue that will alert when a Dasher is 5 minutes away. With this feature, Merchants will now know exactly when to start preparing and bagging those final items (cold drinks, ice cream, fries, etc) right before a Dasher's arrival, preserving food quality and also reducing overall wait times!”

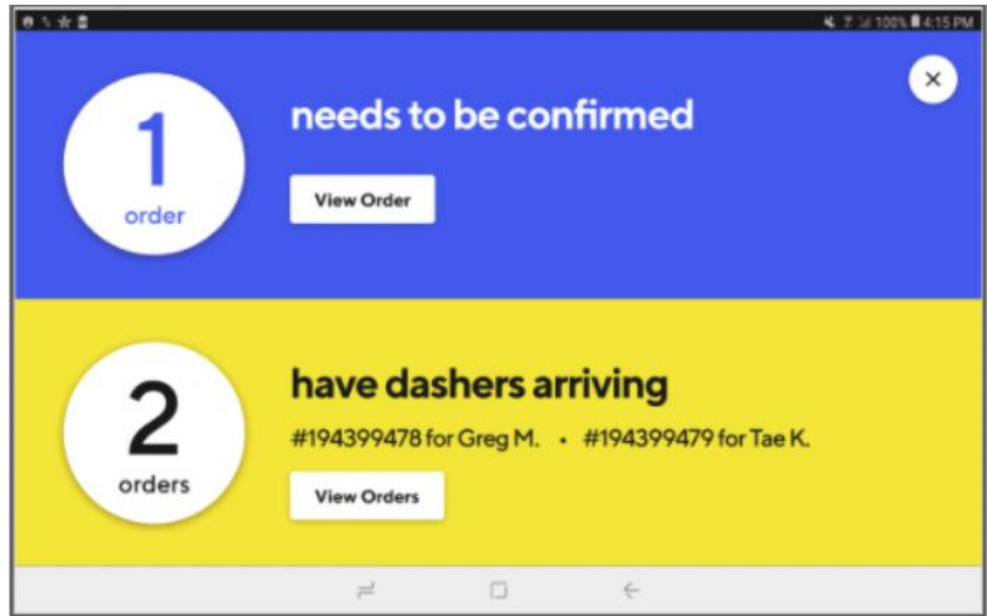
Notification

Once a Dasher is 5 minutes away, a full screen take-over will appear in yellow notifying staff on an approaching Dasher

Note: A unique chime will sound, which will help staff differentiate between a normal incoming order versus an incoming Dasher.

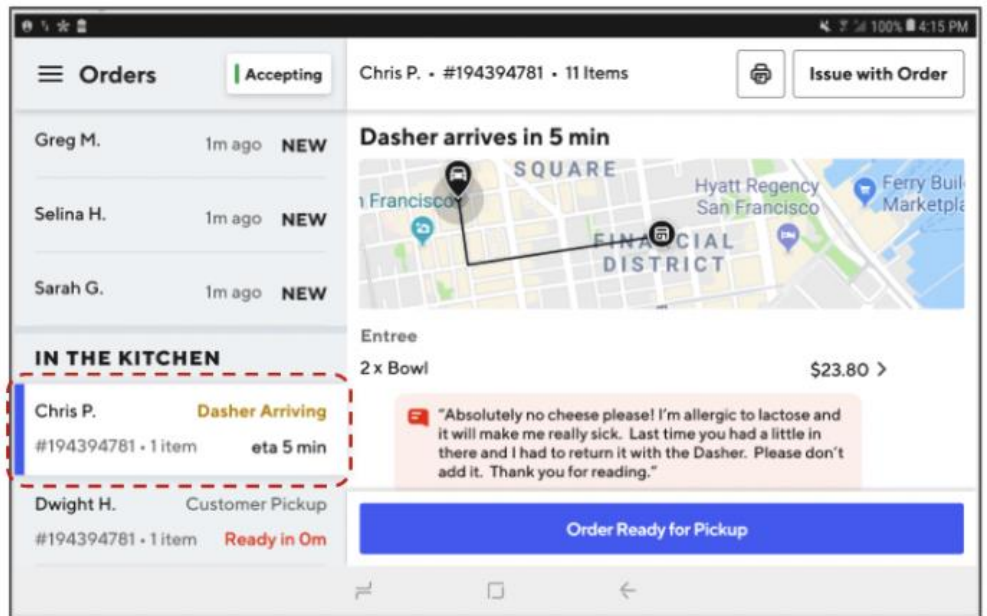


If there are new orders and Dashers arriving at the same time, the screen will appear as below, and the chime for "new order" will alarm. You will also be able to see which orders have Dashers coming for them (Customer name(s) shown).



Dasher Arriving Label

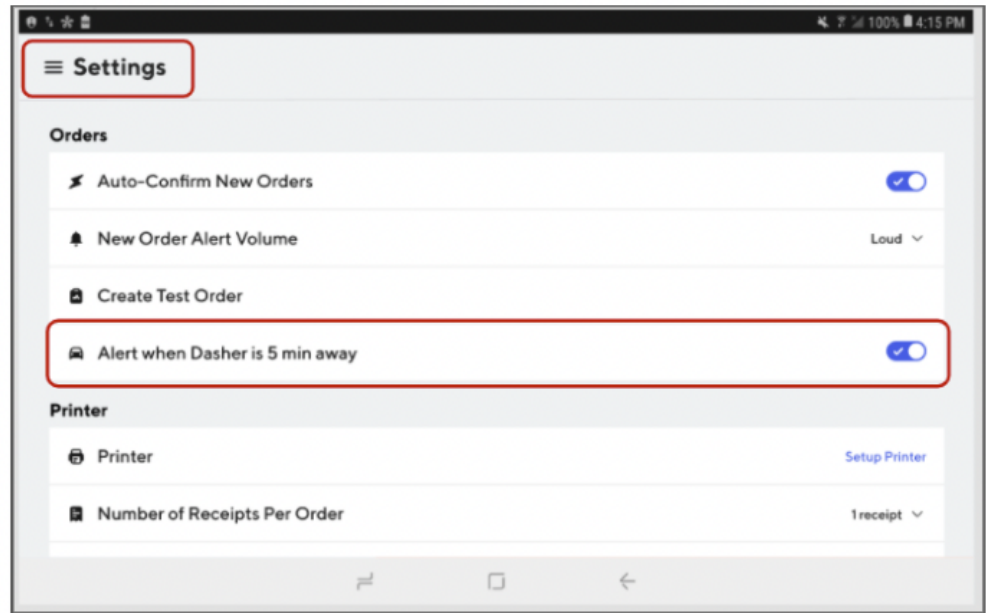
Within each order, there will be a Dasher Arriving label with eta x min shown, counting down until the Dasher arrives.



Feature Settings

Enter the flow directly in the tablet by selecting Settings from the main navigation page

Settings → Alert when Dasher is 5 min away → Toggle On / Off



DoorDash Business Manager Mobile Application

DoorDash offers the Business Manager app for iOS and Android and are found on the Apple App and Google Play Stores.

Google Play Games Apps Movies & TV Books Kids

Introducing the DoorDash Business Manager

The tools you need to manage your business with DoorDash, right on your phone.

DoorDash

3.5 ★ 67 reviews 10K+ Downloads Everyone 0

Install Add to wishlist

You don't have any devices You can share this with your family. [Learn more about Family Library](#)

Trailer

Developer contact

More by DoorDash

- DoorDash - Food Delivery DoorDash 4.6 ★
- DoorDash - Dasher DoorDash 2.7 ★
- DoorDash Order Manager DoorDash 2.5 ★

About this app

Track orders in progress, resolve issues, access Support, keep track of your business performance, and get real-time notifications on

App Store Preview

DoorDash - Business Manager ⁴⁺

Manage your DoorDash store
DoorDash, Inc.

★ ★ ★ ★ 1.0 • 15 Ratings

Free

iPhone Screenshots

Manage and grow your business on the go

Track orders and handle issues in real time

Easily manage your store, adjust hours, and more

Access DoorDash Support anytime, instantly

Track orders in progress, resolve issues, access Support, keep track of your business performance, and get real-time notifications on DoorDash.

TRACK LIVE ORDERS
Manage in-progress orders and see your Dasher's status, location, and arrival time. Mark an item [out](#) [more](#)

The DoorDash Manger lets restaurants “...*track orders in progress, resolve issues, access support, monitor business performance, and get real-time notifications — all on your phone.*” Selective screenshots from the DoorDash Business Manager App Guide

describes the app, and it’s features as below (See Exh 59, 62, 65). With the Business Manager App, managers can in real-time:

- track sales, orders, and trends, for one or multiple stores
- change store status
- update menu hours
- view active orders
- call the customer or Dasher
- contact DoorDash Support
- mark items out of stock
- cancel an order”

INTRODUCTION

DoorDash - Business Manager App

Manage your DoorDash business right on your phone

Running a business isn't easy, but it could definitely be a little easier.

The new DoorDash - Business Manager app lets you track orders in progress, resolve issues, access support, monitor business performance, and get real-time notifications — all on your phone.

This guide will walk you through downloading, installing, and using the new app.

INTRODUCTION

A note on Point-of-Sale (POS) Integrations

What's a POS integration?

A POS integration is a streamlined connection between your point-of-sale (POS) software and DoorDash's platform. This connection lets you receive DoorDash orders directly on your POS, so everything is in one place and there's no need to manually enter orders from one system into another.

Why is the user flow different depending on whether or not I have a POS?

When businesses use a POS system, some of their store information is stored directly on that system instead of on DoorDash. This means that the user experience for stores using POS system differs for certain actions and updates.

| Use your POS system to: | Use the DoorDash Business Manager app to: | Call DoorDash Support to: |
|---|--|--|
| <ul style="list-style-type: none"> Update your menu (Items, modifiers, and photos) Change menu and store hours Mark items out of stock | <ul style="list-style-type: none"> Receive push notifications Track business health View active orders Call customer, dasher, or support | <ul style="list-style-type: none"> Cancel live orders |

Throughout this guide, sections with separate POS instructions will be marked using this tag: POS Integrations

Integration features vary by provider. Reach out to your integration provider or refer to these [help articles](#) for more information.

HOW TO
Install the app - iOS

- 1 Download the app with [this link](#) or QR Code below.
- 2 Click on **Install** and let the app download.
- 3 Once installed, tap **Open** to launch the app for the first time.



Apple App Store



HOW TO
Install the app - Android

- 1 Download the app with [this link](#) or QR Code below.
- 2 Click on **Install** and let the app download.
- 3 Once installed, tap **Open** to launch the app for the first time.

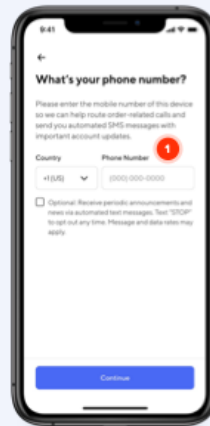


Google Play Store

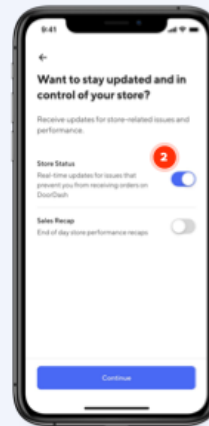


HOW TO
Enable push notifications

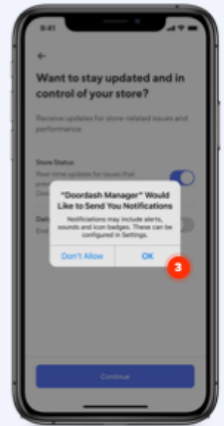
- 1 When you first log in, you'll be asked to provide your **phone number for authentication**. Once you've done that, tap Continue.
- 2 **Enable push notifications** in the toggle bar to receive real-time updates about store issues and end-of-day performance recaps. Tap Continue.
- 3 You'll see a pop-up asking you to allow **DoorDash Manager to send you notifications**. Be sure to select **OK** for full functionality.
- 4 If you select Don't Allow, you'll need to **re-enable notifications** via your device's Settings menu.



1st time authentication



Notifications selector



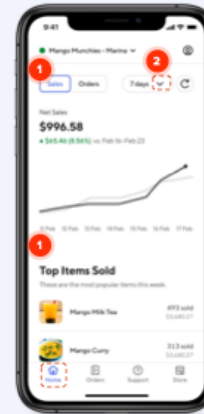
Accept device notifications

HOW TO

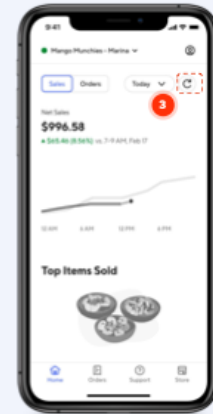
Track business health (single store)

- 1 The Home tab of your Dashboard allows you to see **sales, orders, trends and top items sold** in a specific time range.
- 2 You'll have the option to choose from today, yesterday, 7 days, 30 days. Just tap the dropdown arrow.
- 3 If you select **today**, you'll be able to:
 - Monitor sales and orders in real time by tapping the refresh arrow
 - Analyze partial-day trends

Menu item performance is calculated at 7 days and 30 days, so to view that data just change the time range.



View top-line metrics and trends



Monitor day-of performance

8

HOW TO

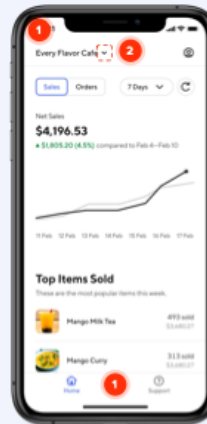
Track business health (multiple stores)

- 1 If you own **multiple locations**, the home screen will display metrics across all stores by default. You can confirm this by noting:
 - The business (or group) name at the top left
 - Your bottom navigation bar will only show 2 tabs (Home and Support)

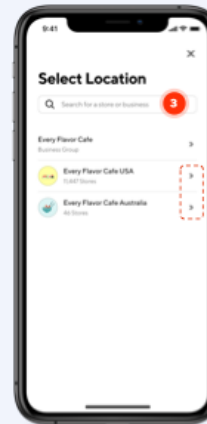
To view **store-level metrics and features**, tap the arrow in the top navigation bar.

- 2
- 3 Depending on how your business was set up on DoorDash, it may take 1-2 taps to get to an individual store (tap the **arrow to navigate**). You can also locate the store in the **search bar**.

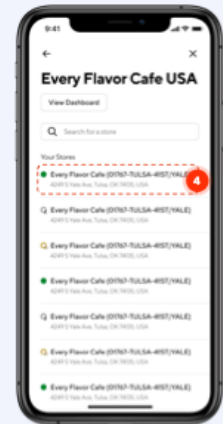
- 4 **Tap the store** you'd like to view and both the app dashboard and metrics will automatically refresh.



Business-level Dashboard



Select a business (if applicable)



Select a store

9

HOW TO

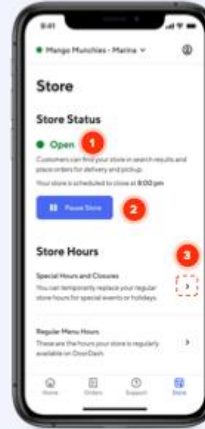
POS Integrations

Change store status

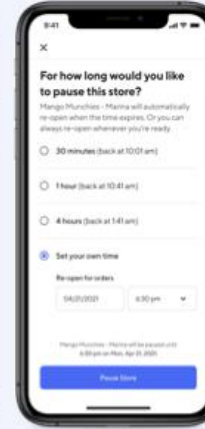
- 1 You can view and change your store status (**open, paused, closed**) in the Store tab. If you have access to multiple stores, select the specific store from the dropdown menu.
- 2 Tap **Pause Store** to pause an open store and choose when it reopens (in an hour, the next day, etc).
- 3 View or add special hours or closures by tapping the right-hand arrow next to **Special Hours and Closures.***

When you change your store status in the app, it will be reflected on the Tablet and Portal (and vice versa).

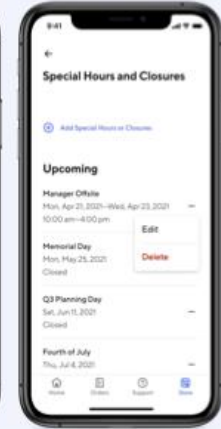
*If you use a POS protocol, we recommend making special hours updates and closures using your POS. When you make these changes via the app, they may be overwritten during a POS sync.



View status



Pause store



Special hours

10

HOW TO

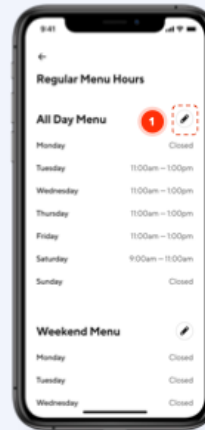
POS Integrations

Update menu hours*

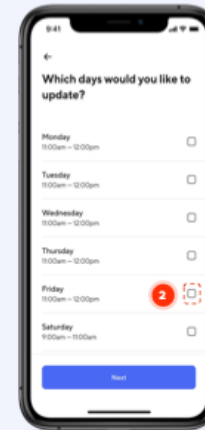
- 1 From the Store tab (refer to previous page for a screenshot) you can update your menu hours by tapping the right-hand arrow next to **Regular Menu Hours**.
- 2 If you have multiple menus for your store, scroll down to find the specific menu. Then tap the **pencil to edit**.
- 3 **Check the box** next to the day(s) of the week you'd like to update the menu hours for and tap **Next**.

Indicate whether the new hours reflect when you're open or closed by clicking on the top toggle. Then **tap on the start or end time** and select the desired time from the dropdown. Tap **Next** and then **Confirm**.

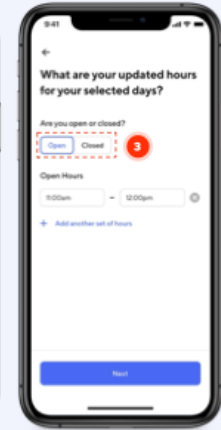
*If you use a POS protocol, we recommend updating menu hours using your POS. When you make these changes via the app, they may be overwritten during a POS sync.



Edit menu hours



Day of week



Start and end time

11

HOW TO
View active orders

- 1 Tap on the **Orders** button in the bottom navigation to view active orders. For already completed orders, you can also toggle to History.
- 2 Note the **order status** on the right-side to see if an order needs attention.
 - ▲ Dasher late
 - Dasher en route
 - Dasher arriving
 - Customer pickup
 - ▲ Dasher waiting
 - Customer arriving
 - Delivered
 - Customer picked up
- 3 See **order details** by tapping on the right-hand arrow by the individual order.
- 4 Call a customer or Dasher directly by tapping the phone icon.



Orders



Order details



Call customer or Dasher

12

HOW TO POS Integrations
Call the customer or Dasher*

If you use a Tablet protocol, you can continue to call customers or Dashers for in-progress orders by tapping the phone icon or Issue with Order.

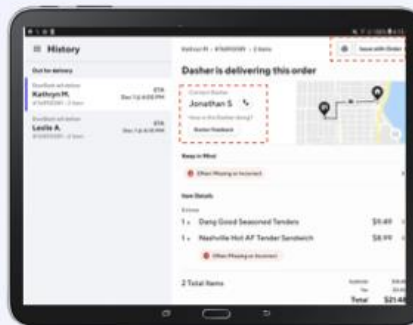
Now you can also call from the app by **clicking on the phone icon by the customer's or Dasher's name**.

- Common reasons to call a customer or Dasher:
- Out-of-stock item
 - Clarification on special instructions
 - Order running late
 - Dasher arrives early
 - Dasher waiting but not coming inside

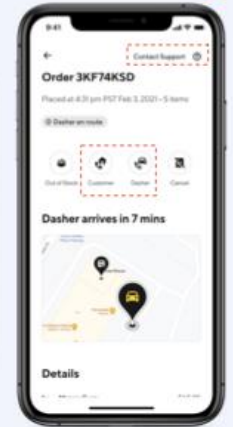
Afterwards, if you need to make any requested **order changes**, you can do so by chatting with our Support team — just tap **Contact Support** at the top right.

Your Tablet is still the best way to adjust prep time or mark an order ready for pickup.*

*If you use a POS Integration, there's no need to mark orders ready for pickup; they will be confirmed and assigned a Dasher automatically.



Using Tablet to call customer or Dasher

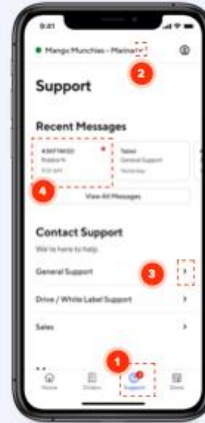


Call customer or Dasher

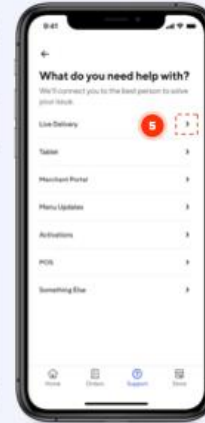
13

HOW TO Contact Support

- 1 The **Support icon** is always accessible on the bottom navigation.
- 2 If you have access to multiple stores, be sure to select the right store from the dropdown.
- 3 To start a new request, tap **General Support** and then select the nature of your question in the next screen.
- 4 If you'd like to reopen a previous conversion, you can do so by tapping one of the Recent Messages.
- 5 For any issues with in-progress orders, tap **Live Delivery**. You can then choose if you'd like to **call Support** or **chat with Support**.



Support



Live delivery



Chat or call

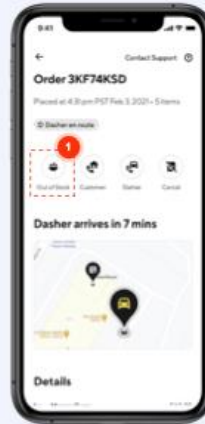
14

HOW TO Mark out of stock

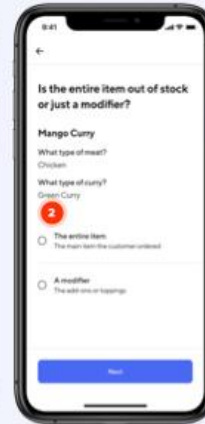
POS Integrations

- 1 Click on an active order to note when an item is **out of stock**.
- 2 Based on the order details, you'll select which **item or modifier** is out of stock.
- 3 Click on Next to **choose how long it will be out of stock** – hours, until the end of the day, or indefinitely. The item will automatically be made available after the selected time. You are also able to mark the item back in stock manually on your Tablet or POS.
- 4 Handle the **active order** in question by tapping Next to **chat with Support**. They'll get in touch with the customer to confirm if they want to replace the out-of-stock item or remove the item.

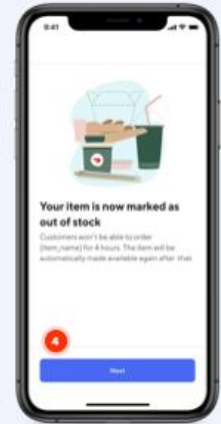
*If you use a POS protocol, we recommend marking items out of stock using your POS. When you make these changes via the app, they may be overwritten during a POS sync.



Active order



Select item or modifier



Chat for live order help

15

HOW TO POS Integrations

Cancel a live order*

- 1 If you're not able to fill an active order, you can choose to **cancel** it from the order details.
- 2 **Provide more detail** on why you can't fill the order so we can help you with next steps.
- 3 If you choose to cancel the order, we'll **confirm** this with you and let the customer know. As a reminder, cancellations that you initiate will **not be paid** for, even if you already prepared the food.

*This feature is not available on POS order protocol. If you're using a POS system and [need to cancel a live order](#), contact DoorDash Support.

Cancel a live order

Tell us what's going on

Confirmation

16

Multi-Modes of Communication

DoorDash affirms in the merchant support document “Which order protocol works best for your business” (see Exh 112) that DoorDash merchants/partners can receive orders through multiple modes of communication including:

- “Your existing point-of-sale (POS) system via an integration
- A DoorDash tablet or your own Android tablet using the Order Manager app
- Email”

This document presents a table that specifies which mobile, tablet or web app is best to use for a given features. Below are screenshots of this specification (see Exh 112):

210

| | Direct POS Integration (POS protocol) | Middleware Integration (POS protocol) | Tablet | Email |
|-----------------------------|---|---|---|--|
| Summary | <p>Popular with high-medium volume stores (~300+ orders per month) Great operational efficiency Most preferred</p> | | <p>Popular with medium-low volume stores (60-300 orders per month) Good operational efficiency</p> | <p>Suited for low volume stores (less than 60 orders per month) Lowest operational efficiency Least preferred</p> |
| Best For | <p>Businesses of all sizes, from quick-service to full-service, with an existing POS system.</p> | <p>Businesses of all sizes, from quick-service to full-service, with an existing POS system that does not support integration. Also for businesses using multiple third-party delivery providers.</p> | <p>Businesses of all sizes without a POS system or where POS integration is not available.</p> | <p>Small businesses that do not have a POS system and/or efficient internet connectivity.</p> |
| It Provides | <p>An easy, seamless way to manage delivery and pickup orders using your POS.</p> | <p>An easy, seamless way to manage delivery and pickup orders using your POS.</p> | <p>Quick order management, access to Dasher status monitoring, item availability updates, and real-time prep time adjustment.</p> | <p>Manual work to receive, confirm, and update orders.</p> |
| Country Availability | <p>Varies by provider</p> | <p>Varies by provider</p> | <p>All markets</p> | <p>All markets</p> |

| Functionalities | | | | |
|--|---|---|---|---|
| Receive Delivery and Pickup Orders | ✓ | ✓ | ✓ | ✓ |
| Receive Scheduled Orders | ✓ | ✓ | ✓ | ✓ |
| Managed Menu Changes | ✓ | ✓ | Merchant Portal | Merchant Portal |
| Menu Item modifiers | Varies by provider | Varies by provider | ✓ | Merchant Portal |
| Manage Menu Photos | Varies by provider | Varies by provider | ✓ | Merchant Portal |
| Manage Prep Times | Varies by provider | Varies by provider | ✓ | × |
| Track DoorDash Business Health | ✓ | Varies by provider; available on Business Manager app or Merchant Portal | Business Manager app or Merchant Portal | Business Manager app or Merchant Portal |
| Pricing Merchant charges/fees paid to DoorDash | FREE Some providers charge a fee for 3rd party integrations. Reach out to your provider for details | FREE Some providers charge a fee for 3rd party integrations. Reach out to your provider for details | \$6/ WEEK | FREE |
| Extra Hardware | None | None | DoorDash or Personal Android Tablet | Computer or tablet |

| Menu Management Features | | | | |
|--|--------------------|--------------------|---|---|
| Item 86'ing (Item OOS in real-time) | Varies by provider | Varies by provider | ✓ | Business Manager app or Merchant Portal |
| Manage Store Hours | ✓ | ✓ | ✓ | Business Manager app or Merchant Portal |
| Manage Store Holiday/Special Hours | ✓ | ✓ | ✓ | Business Manager app or Merchant Portal |
| Update Menu Hours | Varies by provider | Varies by provider | ✓ | Business Manager app or Merchant Portal |
| Separate and Day-part Menus | ✓ | ✓ | ✓ | Merchant Portal |
| Special Instructions* | ✓ | ✓ | ✓ | Merchant Portal |
| Menu Item Sort Order | ✓ | ✓ | ✓ | Merchant Portal |
| Menu Item Level Hours** | Varies by provider | Varies by provider | ✓ | Merchant Portal |
| Price Override/Inflation | Varies by provider | Varies by provider | ✓ | Merchant Portal |

| Live Operations Features | | | | |
|--|--|--|---|--|
| Busy Kitchen | Contact DoorDash Support | Contact DoorDash Support | ✓ | Contact DoorDash Support |
| Temporarily Pause Store | Business Manager app or Merchant Portal | Business Manager app or Merchant Portal | ✓ | Business Manager app or Merchant Portal |
| Contact Customer | Business Manager app or contact DoorDash Support | Business Manager app or Contact DoorDash Support | ✓ | Business Manager app or contact DoorDash Support |
| Cancel/Refund Customer | Business Manager app or contact DoorDash Support | Business Manager app or Contact DoorDash Support | ✓ | Business Manager app or contact DoorDash Support |
| Dasher Updates | Business Manager app or contact DoorDash Support | Business Manager app or Contact DoorDash Support | ✓ | Business Manager app or contact DoorDash Support |
| Contact DoorDash Merchant Support | Business Manager app, phone, or email | Business Manager app, Merchant Portal, phone, or email | ✓ | Business Manager app or contact DoorDash Support |

Furthermore, in the merchant support document “How do I receive orders with DoorDash?” DoorDash affirms (see Exh 113):

“If you would prefer orders to come through a tablet, all orders will come through DoorDash's Order Manager App on Android tablets.

- *With the tablet, you can see:*
- *Sections for each order stage lifecycle to easily identify where an order is*
- *Scheduled order timing when it appears in-app is changing (used to show up/print immediately)*
- *Cancellations won't just disappear anymore*
- *Tags for scheduled orders, large orders, canceled orders, modified orders,*

customer pickup

- *Item count per order*
- *Large/clearer map and Dasher location information*

You can receive orders by POS, email, or email/fax. However, you will not be able to receive orders from phone+tablet or email+fax.

DoorDash can also place a follow-up automated call to ensure that all orders were received.”

The DoorDash Merchant support article “How to switch to POS order protocol” states that (see Exh 117):

“As a DoorDash partner, you can choose to receive DoorDash orders through:

- Your existing point-of-sale (POS) system via an integration (see the complete list of integration partners here)
- A DoorDash tablet or your Android tablet using the Order Manager app
- Email

DoorDash’s Unified Chat Experience

The DoorDash engineering team stated in an engineering blog post, titled **Building a Unified Chat Experience at DoorDash**, how they successfully integrated chat into their overall system, addressing the challenges and strategies employed to create a unified chat experience. Their implementation aligns with MFCCS as it provides a centralized system layer architecture enhancing communication between the web server and multiple wireless handheld devices, each operating on distinct mobile systems thus allowing customers to communicate with customer support and dashers. (See Exh. 102)

DoorDash recognized the crucial need to address customer issues on a large scale, leading them to acknowledge the essential nature of a unified chat feature. Their initial challenge was a fragmented chat system, which was scattered across various user groups and platforms, causing duplicated efforts and central adoption of best practices to be challenging.

By leveraging the architecture MFCCS, DoorDash could centralize its chat functionalities, allowing for seamless interactions between consumers, Dashers, and merchants. The MFCCS would facilitate the web server's ability to communicate with diverse handheld devices, presenting mobile-compatible versions of the hospitality application. This integration is crucial for enabling user-initiated actions and subsequent selection of choices directly from the touchscreens of various devices, fostering a multi-modal, multi-protocol communication environment.

According to the engineering article, a significant objective for DoorDash was to streamline their support processes through automation, allowing customers to retrieve information through the app itself, thus minimizing the need for human support interaction. They aimed to develop a single chat support platform that would ensure consistent user experiences across different platforms and facilitate the reuse of components.

In building this unified platform, DoorDash created a single chat implementation that functioned for both customer support and communication between consumers and

Dashers. This system, powered by Sendbird, was directly integrated into the consumer and Dasher apps, providing a uniform experience across all DoorDash chats and allowing for shared enhancements across different chat flows.

The backend system was strategically designed with multiple layers, distinguishing between internal services and third-party functionalities such as chat natural language processing. This separation expedited development by eliminating the need to create distinct, non-essential features.

Integration with their Decision Engine platform was another crucial step, which enabled operations to be automated, such as order updates, issuing credits, and personalized messaging. This significantly decreased the dependency on manual operations. DoorDash also carefully monitored several key performance indicators, including feedback from agents and customers, error rates, customer satisfaction, average handle time, first contact resolution, and manual tasks per order.

The outcomes of this updated chat support system, according to the article, were noteworthy, with reduced escalation rates and improved customer satisfaction. The automation led to a decrease in manual interactions per delivery, and customers were able to solve their problems more quickly, sometimes without the need for manual assistance. The time needed to implement additional automation was also reduced, owing to the common platform processing. Moreover, the common UI layers ensured that the chat feature was seamlessly integrated into the applications (iOS and Android), providing a consistent brand experience across all platforms.

The advancements DoorDash has made in their chat system encapsulate the essence of the Middleware/Framework Communications Control Software (MFCCS) described in the patent claim. By creating a unified chat platform that seamlessly integrates with multiple user interfaces and operating systems, DoorDash has effectively embodied the centralized system layer architecture that MFCCS envisions. Their backend, which differentiates between core and third-party functionalities, mirrors the multi-layered approach of MFCCS, allowing for efficient, scalable communication across various devices. Moreover, the phased rollout and meticulous performance monitoring align with the adaptive, intelligent operational capabilities that MFCCS aims to provide. This demonstrates a real-world application of the patent's claim, showcasing a system that not only supports multi-mode communication but also enhances user experience and operational efficiency in a dynamic hospitality market.

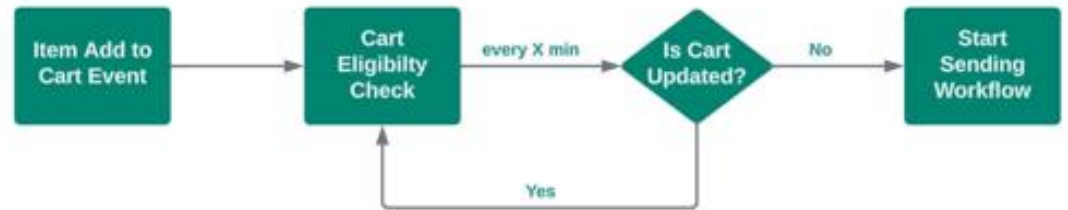
Notifications

The MFCCS is evident through DoorDash's strategic use of real-time notifications to engage consumers and enhance user experience as detailed by their engineering team in the blog article titled "**Leveraging Flink to Detect User Sessions and Engage DoorDash Consumers with Real-Time Notifications**". By harnessing the capabilities of Apache Flink, DoorDash has been able to detect user sessions and discern the critical moments to send push notifications, directly aligning with the objectives of the MFCCS to facilitate intelligent, timely communication between the server and the user's handheld devices. (See Exh. 127)

This approach to session detection and notification dispatch represents the essence of MFCCS. The sessions on DoorDash's platform are managed in a way that reflects the centralized control of MFCCS, grouping user activities into sessions and determining inactivity, thus optimizing the timing for sending out notifications. This reduces

unnecessary communication attempts, which is a core feature of the MFCCS, aiming to enhance the efficiency of the server's operations.

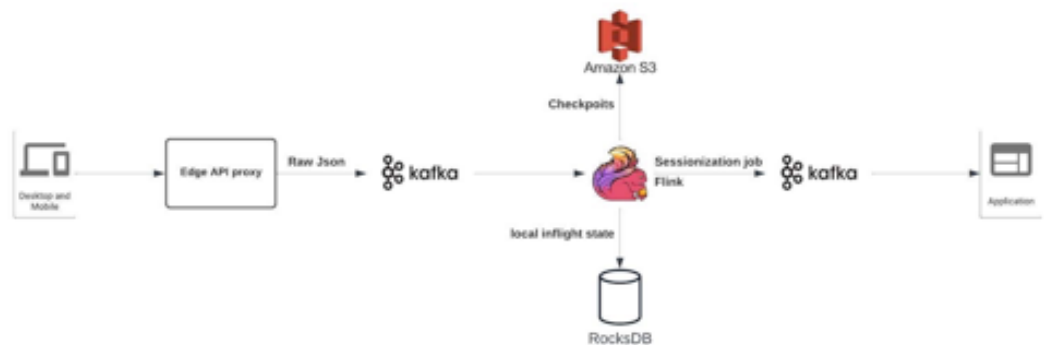
The previous notification system at DoorDash relied on a front-end triggered approach that lacked the precision of the current model, often resulting in untimely and less effective user notifications which did not reflect actual user engagement or session activity accurately, as shown below:



DoorDash's new backend-focused strategy (see diagram below) for triggering notifications ensures that the system accurately mirrors actual user activity, an essential aspect of the MFCCS's design to ensure relevance and personalization of user interactions. The sessionization platform developed by DoorDash utilizes in-memory streaming for processing events, further embodying the MFCCS principle of leveraging real-time data for improved service delivery.



The infrastructure supporting this system (see diagram below) mirrors the robustness required by MFCCS, capable of handling large-state computing and managing user events in a stateful manner until the appropriate action is determined. This is in line with MFCCS's aim to manage multiple communication modes and protocols, ensuring that the server selects the most efficient and effective form of user engagement.



The tangible results of integrating such an intelligent communication system is that DoorDash's cart abandonment notifications have not only increased order volume and revenue but also the speed of notification delivery by six-fold, leading to higher user engagement rates. This success is a testament to the MFCCS's capability to provide an adaptable.

| | |
|--|---|
| | <p>Robocalls for Store Status</p> <p>In the DoorDash customer support article “Using Robocalls to Verify Store Hours” DoorDash's states that it uses automated calls (or robocalls) to its customers locations to ensure the maintain efficient and reliable service of operations. As detailed in the support document, DoorDash implements robocalls to verify store hours and open status, especially during times of uncertainty such as holidays, severe weather, or unforeseen events. This automated system embodies the proactive and responsive communication strategy central to MFCCS. (See Exh. 128)</p> <p>The document outlines how robocalls are placed from the DoorDash backend to merchants, requiring a simple keypad response to ascertain the store's status, thereby minimizing avoidable cancellations, and ensuring a seamless customer experience. In cases of non-response, a temporary deactivation is initiated, emphasizing the MFCCS's focus on intelligent and adaptable operational processes. Merchants can reactivate their service through DoorDash's Merchant Portal or by contacting support, showcasing a system designed for rapid recovery and continuity, which is intrinsic to the robust nature of MFCCS.</p> <p>This proactive approach to merchant engagement via robocalls, as practiced by DoorDash, ensures that the service ecosystem remains agile and customer-focused, mirroring the MFCCS's goal of utilizing technology to enhance operational efficiency and user satisfaction.</p> <p>The accused instrumentality meets this limitation under the doctrine of equivalents because the accused instrumentality performs substantially the same function, in substantially the same way to achieve substantially the same result. For example, DoorDash's system mirrors the functionalities described in the patent claim for Middleware/Framework Communications Control Software (MFCCS). Their architecture facilitates seamless interactions across different mobile operating systems and devices, meeting the claim's requirements for centralized communication. This system supports varied user interfaces and adapts to real-time data processing needs, which exemplifies the adaptable and efficient communication control integral to the claim under the principle of equivalence.</p> |
| <p>at least one external software application program interface (API), which enables the full integration of the one or more hospitality software applications via the MFCCS and its layer architecture with one or more non-hospitality applications;</p> | <p>DoorDash has strategically implemented at least one external API showcasing their advanced master database and their ability to synchronize and integrate their food and drink ordering software application with non-hospitality applications like Google Maps and texting.</p> <p>In their enhancement from a monolithic codebase to a microservices architecture, DoorDash has focused on refining API designs for better platform functionality. They emphasize the importance of APIs in providing information to apps and facilitating quick, storage-efficient, and failure-resistant operations. This commitment to API optimization has led to a significant reduction in latency and errors, improving both backend services and user experience, according to the DoorDash engineering team.</p> <p>The API for menu management is another aspect of DoorDash's commitment to integration and efficiency. It allows for synchronization from the POS system to DoorDash, with the capability for merchants to adjust imported items using DoorDash's menu manager. The process of updating menus involves both pushing from the partner</p> |

to DoorDash and pulling from DoorDash to the partner, ensuring that menus are current and accurate.

DoorDash's Documentation highlights the selective access to Marketplace APIs, emphasizing the importance of managing menu, store, and order data. This approach aligns with their goal to offer a controlled and efficient API system that supports the seamless operation of their marketplace.

Moreover, DoorDash's integration with Google Maps via its API for real-time navigation in both iOS and Android apps showcases their ability to leverage external non-hospitality applications to enhance their service. This integration provides Dashers with precise delivery locations and navigation data, crucial for the delivery journey.

The DoorDash engineering team's blog post about building a unified chat experience illustrates their successful integration of a chat system that aligns with the MFCCS, providing a centralized layer that enhances communication across different devices and platforms. This system, powered by Sendbird and integrated with the consumer and Dasher apps, ensures consistent user experiences and leverages automation to reduce the need for human support interaction.

DoorDash's advancements in integrating a chat system embody the centralized system layer architecture that MFCCS envisions. Their multi-layered backend, differentiated between core functionalities and third-party services, allows for efficient, scalable communication across various devices, demonstrating the practical application of the patent's claim in enhancing user experience and operational efficiency in the hospitality market.

In the blog article "Platform Optimization Through Better API Design" by the Engineering Team (see Exh 22) they admit and confirm the DoorDash platforms framework and API optimizations. Specifically, the article states:

"As DoorDash migrated from a monolithic codebase to a microservices architecture, we found an opportunity to refine our API design. Beyond simple functionality, we determined best practices in making APIs that help our applications load quickly, use minimal storage, and, most importantly, avoid failures.

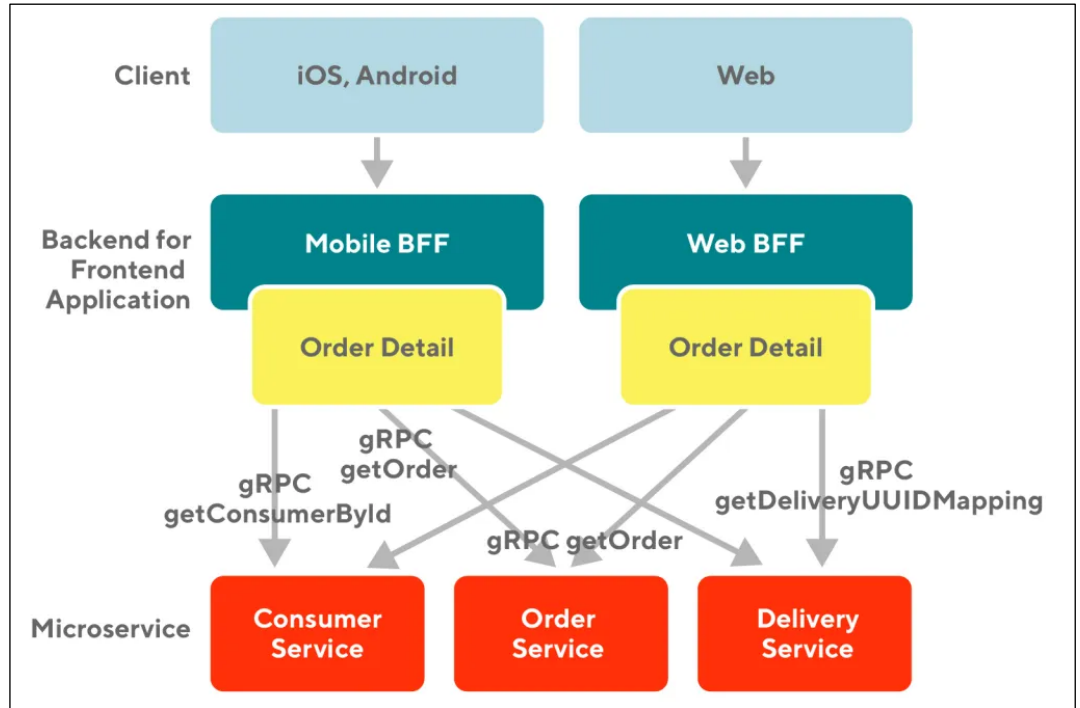
APIs, the connective tissue of a software platform, can offer performance improvements when properly designed. At DoorDash, the APIs relay front end client requests to backend services and provide the information that users see in our apps, such as the estimated time when a food order will be delivered. If the APIs perform at maximum efficiency, client requests and responses all process more quickly

The migration to a microservice architecture gave DoorDash the opportunity to revisit our API design, specifically determining how to optimize the information flows. We came up with best practices around targeted endpoint requests, resulting in significantly reduced latency and errors.

As shown in Figure 2, below, when the BFF receives the order details request, it orchestrates the calls to the consumer service, order service, and delivery service, ultimately assembling the response details into a consolidated order detail response. Building APIs using domain-specific services, orchestrated by the BFF, makes it easier to understand which RPCs are called and how they are

executed.”

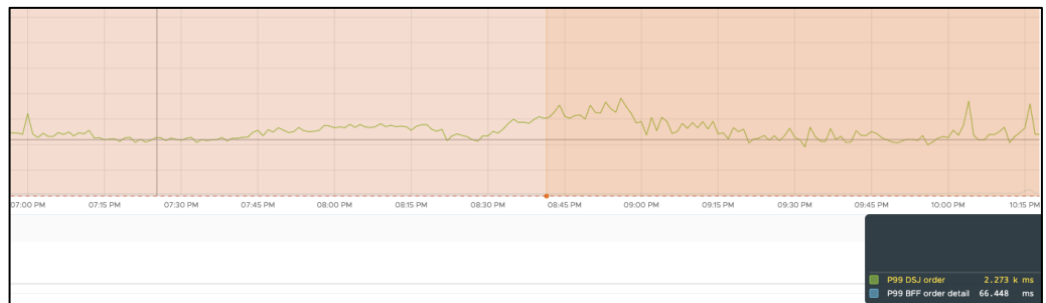
“[The figure below shows] ...our microservices architecture, APIs orchestrated by a BFF are targeted towards specific services, making it easier to design precise calls and appropriate execution.”



“At DoorDash, our APIs primarily support information on food orders, whether that’s an order placed by a consumer and sent to a restaurant, a delivery request sent to a Dasher, or a Dasher’s progress in bringing a food order to a consumer.

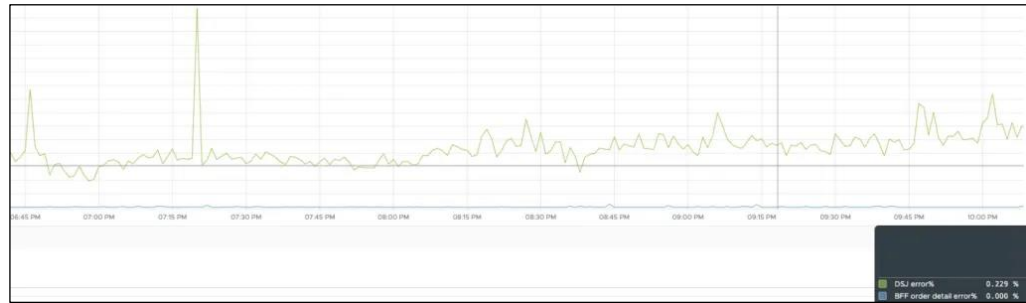
Redesigning our APIs to focus on their specific roles yielded significantly reduced latency and error rates. For the use cases cited above, both the order detail endpoint and order history endpoint process a high volume of traffic. Our API improvements show that:

- For the order detail endpoint, we reduced endpoint latency by over 90%, a 30 times improvement. For P99, as shown in Figure 6, the latency has been reduced from greater than 2,000ms to less than 100ms. In terms of error rate, shown in Figure 7, we achieved a 60 times improvement, going from 0.3% on average to 0.005% (actually, the new endpoint often shows 0% error).”



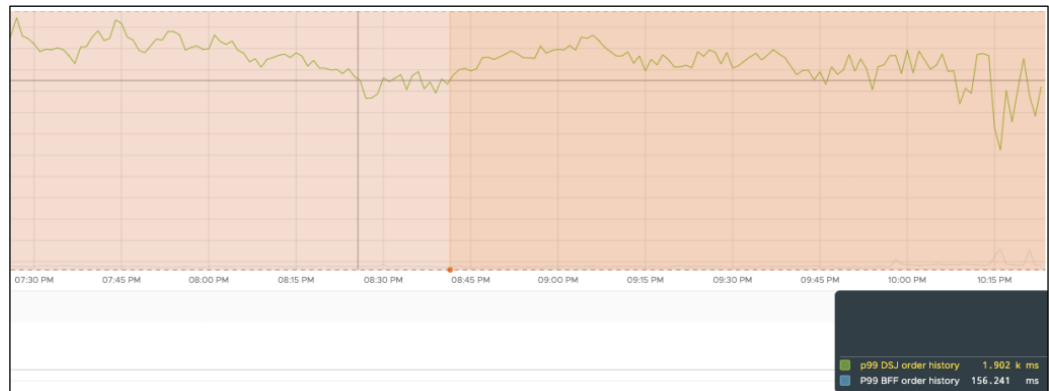
“[The figure above shows] ... redesigning our order detail API to better focus

on its role led to a 30 times reduction in latency.”

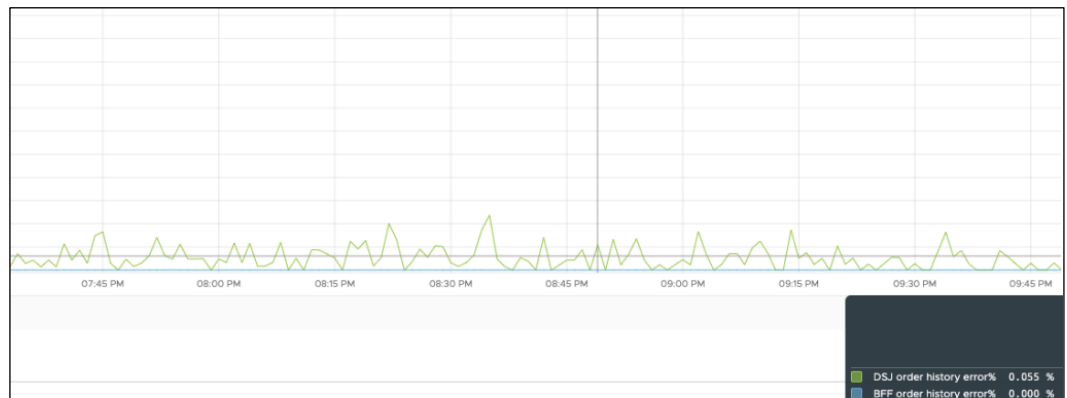


“[The figure above shows the] ...order detail error rate went down significantly with the redesigned API, and it frequently runs flawlessly.”

- *“For the order history endpoint, we achieved a 10 times reduction in endpoint latency, as shown in Figure 8, and an error rate for the new API of 0% for almost all the time, as shown in Figure 9.”*



“[The figure above shows] ... [t]he order history endpoint shows a 10 times reduction in latency with our new design.”



“[The figure above shows metrics] ...for the redesigned order history API rarely show any errors.

“These API improvements led to a perceivable improvement on the client side. When we place a test order with a couple of items the response size of the old API was 72kb (71.4kb exactly), while it is 1kb (1.1kb exactly) using the new API, as shown [these figures below]:”

| | | |
|------------------|-------|---|
| Request Headers | 14 | > |
| Response Headers | 17 | > |
| Response Body | 72 KB | > |

| | | |
|------------------|------|---|
| Request Headers | 14 | > |
| Response Headers | 15 | > |
| Response Body | 1 KB | > |

In a video by DoorDash's engineering team title “Databases at DoorDash | How DoorDash manages 1.9 PB of data & 1.2M QPS on CockroachDB”, DoorDash showcases their utilization of Cockroach DB as the “master database” for managing massive amounts of data, demonstrating their implementation of a master database. The video begins with Mike Czabator, an engineer with both DoorDash and Cockroach Labs experience, providing insights into DoorDash's database practices. DoorDash's core infrastructure team is responsible for a range of databases, including CockroachDB, showcasing a dedicated approach to database management. (See Exh. 122 -123)

DoorDash's deployment strategy spans a single AWS region, supported by multiple Availability Zones (AZs), highlighting their scalability. The substantial growth metrics, including a 55% node increase and nearly doubling data size to two petabytes (PBs), highlight the extensive scope of their data management. Furthermore, DoorDash promotes a self-service model, allowing users to create applications and users for their clusters. They emphasize automation and feedback-driven improvement, enhancing the efficiency and reliability of their database operations. DoorDash's commitment to performance optimization and continuous enhancement aligns with the concept of a master database, reinforcing their capability in handling vast amounts of data.

In another video by the DoorDash Engineering team titled “DoorDash’s Journey from Aurora Postgres to CockroachDB”, they shared insights into their data management practices, particularly their utilization of CockroachDB for handling extensive data volumes with the “master database”. DoorDash's primary challenge was managing high traffic loads, which often led to database crashes during peak usage. To address this, the team recognized the need for a database solution that could provide horizontal scalability through multiple writers. (See Exh 124 -125)

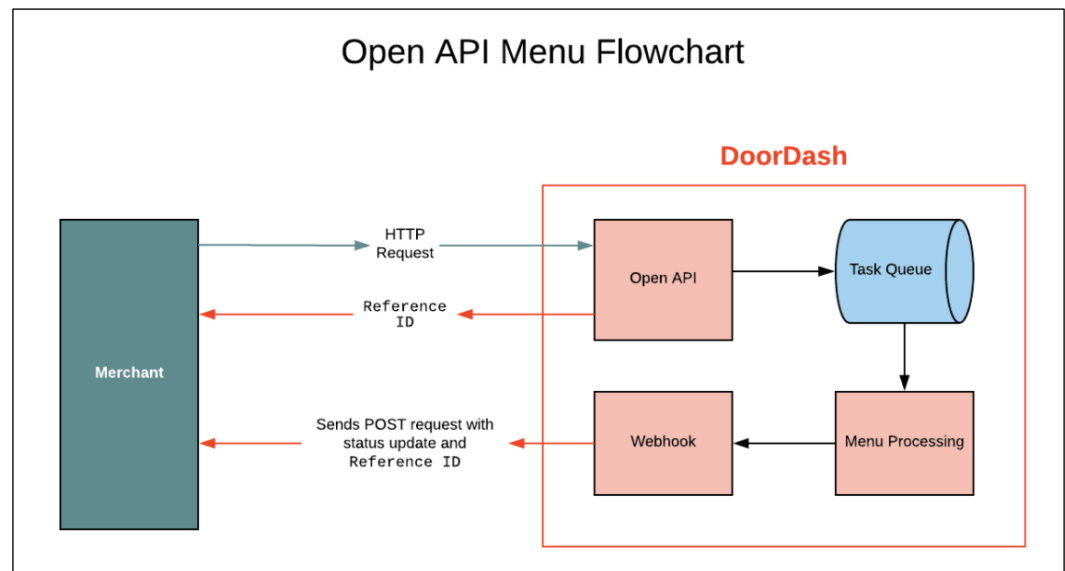
During the COVID-19 lockdown, DoorDash faced a pivotal moment when their main database cluster experienced a peak query per second (QPS) load of 1.6 million, resulting in system downtime. To tackle this issue, they opted to migrate to CockroachDB, a distributed database system known for its scalability and fault tolerance. Their migration strategy involved the extraction of 54 tables to seven new

database clusters, significantly reducing the QPS load on the main database cluster. To facilitate this migration, DoorDash developed a versatile data migration tool. This tool was designed to minimize manual intervention, support schema transformations, and streamline data synchronization. Key features of the tool included the ability to customize runtime behavior, accommodate schema changes, and perform data chunking for parallel processing. DoorDash also leveraged foreign data wrappers, intelligent traffic routing, and data chunking to optimize the migration process.

Menu Management/Open API

The Open API of Menu management automatically synchronizes the merchant's menu with the DoorDash platform. This is a one-way synchronization from the POS system to DoorDash. Not all features of DoorDash's menu manager are supported for all merchants. In some cases, the menu items will sync over but the sort order will not. The merchant is still able to utilize the menu manager to adjust imported items. Order fulfillment can happen through one of four different order protocols: direct POS integration, middleware integration, tablet (DoorDash Order Manager app), or email (see Exh. 31, 32).

Updating menus on the DoorDash platform involves both Push (partner to DoorDash) and Pull (DoorDash to partner). When a partner is first added to the DoorDash platform, a request is issued to the partner's server to retrieve a merchant's menu. The partner's server must respond with the full menu in its body. Whenever there are changes to a merchant's menu, the partner can then issue a request to DoorDash's server, pushing the new menu. In both push and pull scenarios, once the menu has been processed by the DoorDash platform, a request is issued to the partner server indicating the success or fail (see Exh. 38).



In the DoorDash Developer Documentation site, they state the following on the About DoorDash Marketplace page (see Exh 106):

“MARKETPLACE APIS ARE LIMITED ACCESS, The 2022 Marketplace API integration pipeline is closed. New integration requests will not be accepted. Please record interest for 2023 builds here, and we'll reach out when capacity opens.

The DoorDash Marketplace API enables partners to manage their menu, store, and order data. You can use the Menu API to create and update the menus you wish to display to consumers on DoorDash. With the Order Webhook, you can receive live order data from DoorDash directly to your system. And through our Store webhook, you're able to control store level data such as store availability and item availability.

DoorDash Merchant API is an asynchronous API organized around REST. JSON is returned by all API responses, and we use conventional HTTP response codes to indicate the success or failure of a request. All requests must be made using HTTPS or they will fail.”

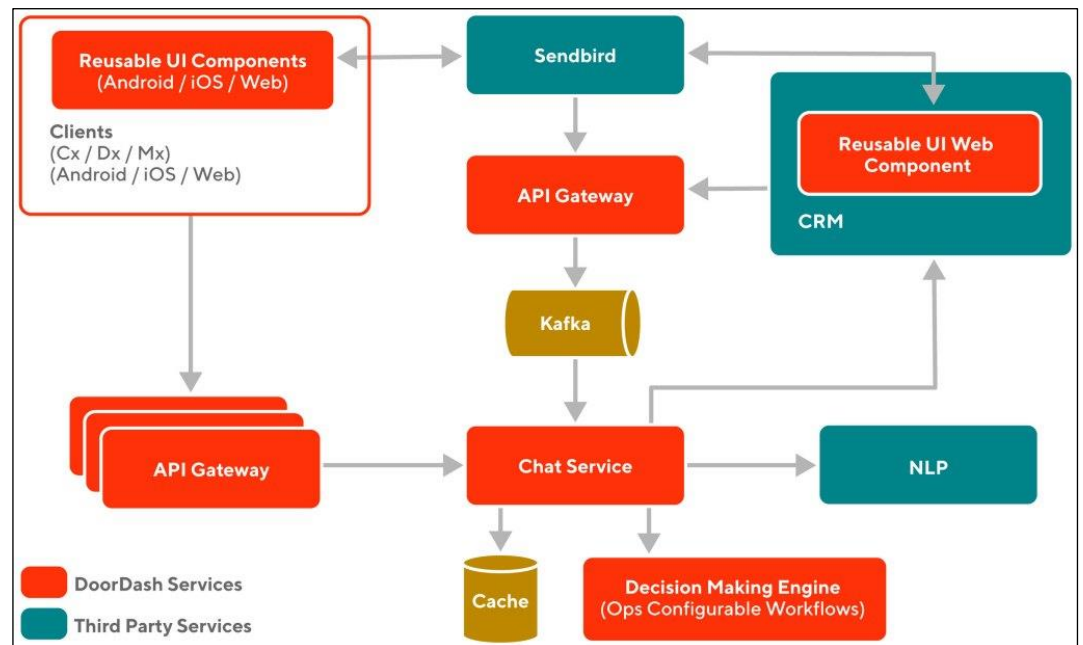
In the blog article “Building a Unified Chat Experience at DoorDash” by the Engineering Team, they provide further admissions of DoorDash’s layered architecture approach and integration with non-hospitality (third party) applications (see Exh 102):

“Building an extensible backend system

The backend system was built with multiple layers, allowing us to split responsibilities between internal services and third parties. Utilizing third parties for functionality such as chat, natural language processing (NLP), and agent ticket management allowed us to move quicker without having to build functionality that does not differentiate ourselves directly.

To allow the platform to handle processing at common layers, we created anew API gateway for all the different callbacks from third parties. Since the processing does not need to be synchronous, we are able to push the messages to a messaging queue for processing to ensure we process the messages reliably. The downstream processing is then agnostic of the source and data-driven based upon the different events.”

The figure below shows: “... [data] flow through centralized chat service, integrating with 3rd parties”


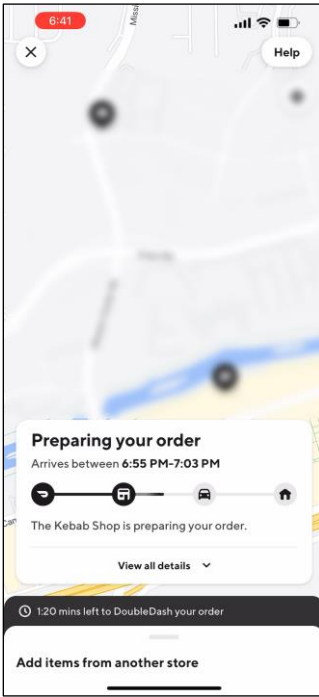


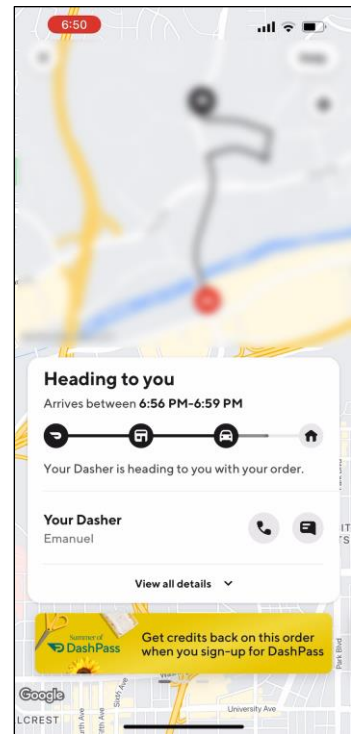
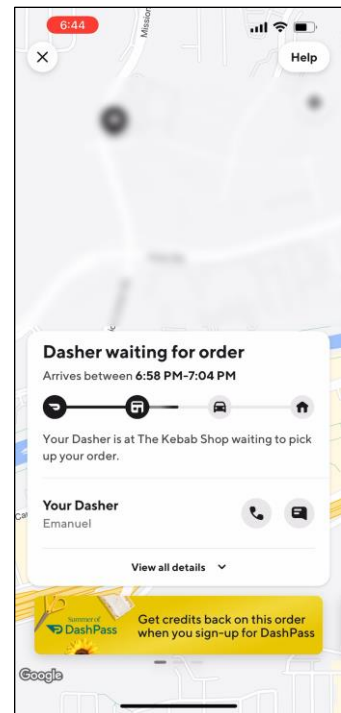
DoorDash also integrates via its API’s its website, webapp, DoorDash iOS app, DoorDash Android app, Dasher iOS App and Dasher Android app with Google Maps

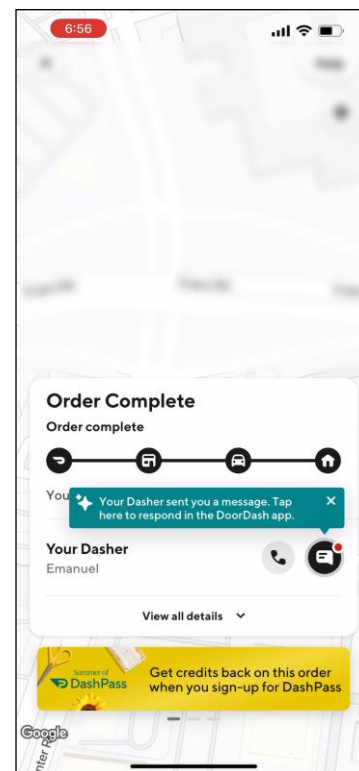
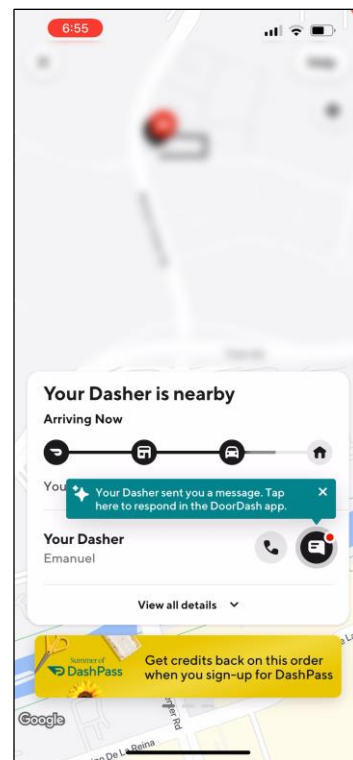
through the Google Maps API.

Google Maps and DoorDash Mobile Apps

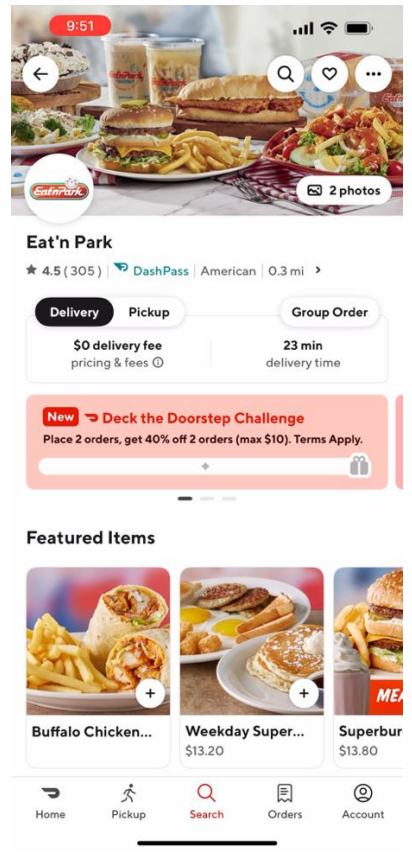
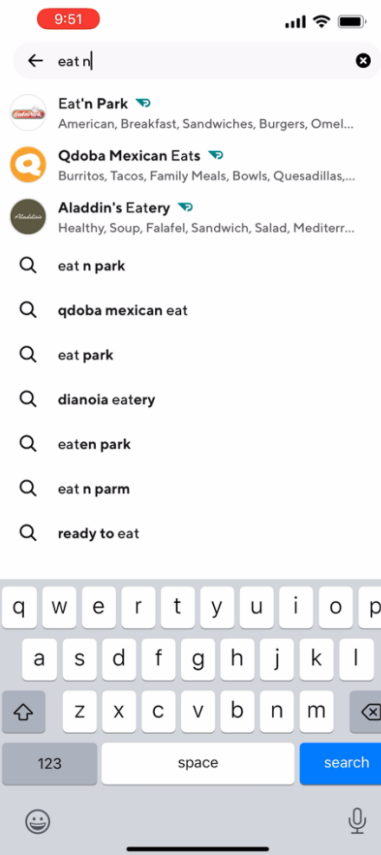
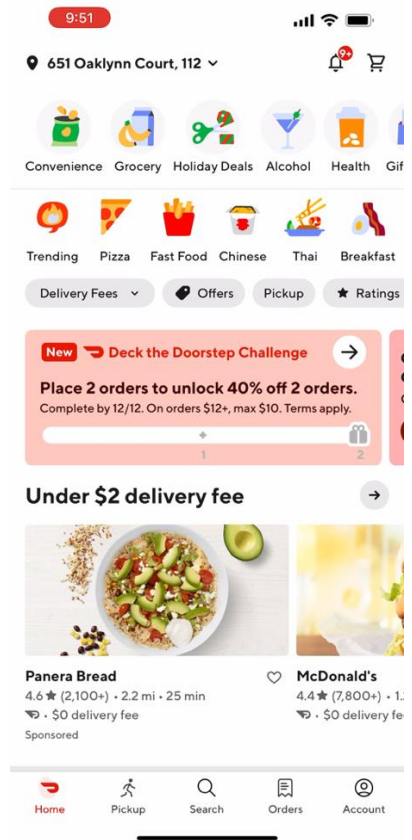
In both the iOS and Android DoorDash apps we see real time delivery information displayed in an embedded Google Map after the user has placed an order with a DoorDash partnered restaurant, and the order has been sent out for delivery. As the dasher gets closer to their destination, the DoorDash Android and iOS apps update the embedded Google Maps (through the Google Maps API) in real time from data from the DoorDash platform, through communication with the webserver and DoorDash Platform.

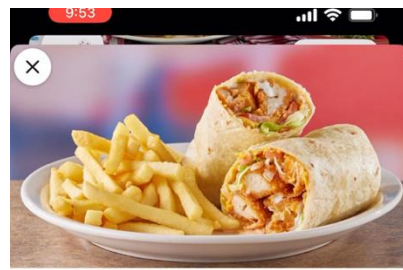
| Android DoorDash App | iOS DoorDash App |
|--|--|
|  |  |





Below are screenshots of an DoorDash order for a customer located in Pittsburgh PA, ordering from Eat 'N Park on the iOS app.





Buffalo Chicken Wrap

350 cal

Wrap-tastic! We load a flour tortilla with Buffalo chicken, cheddar cheese, lettuce, tomato, and your choice of homemade Ranch or bleu cheese dressing.

Select popular options for your order

#1 • Ordered recently by 20+ others

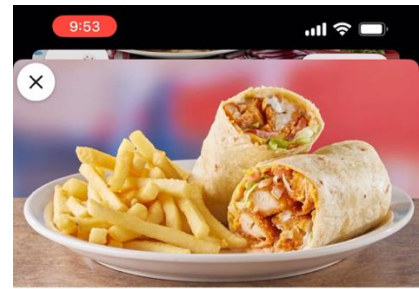
1 Side • Crispy Chicken Tenders • Ranch • French Fries
\$13.80

Select Your Buffalo Chicken Wrap

Required • Select at least 1

2 Sides
350 cal +\$16.30 >

Make 1 required selection \$0.00



Buffalo Chicken Wrap

350 cal

Wrap-tastic! We load a flour tortilla with Buffalo chicken, cheddar cheese, lettuce, tomato, and your choice of homemade Ranch or bleu cheese dressing.

Select popular options for your order

#1 • Ordered recently by 20+ others

1 Side • Crispy Chicken Tenders • Ranch • French Fries
\$13.80

Select Your Buffalo Chicken Wrap

Required • Select at least 1

2 Sides
350 cal +\$16.30 >

Make 1 required selection \$0.00



1 Side

Select Your Protein:

Required • Select 1

Crispy Chicken Tenders
470 cal

Grilled Chicken Breast
180 cal

Select Your Dressing For Your Wrap:

Required • Select 1

Ranch
200 cal

Bleu Cheese
300 cal

No Dressing

Choose Your Side:

Required • Select 1

Fried Cheese Sticks with Marinara +\$1.40
295 cal

Fried Cheese Sticks with Ranch +\$1.40
545 cal

Make 3 required selections



1 Side

Select Your Protein:

Required • Select 1

Crispy Chicken Tenders
470 cal

Grilled Chicken Breast
180 cal

Select Your Dressing For Your Wrap:

Required • Select 1

Ranch
200 cal

Bleu Cheese
300 cal

No Dressing

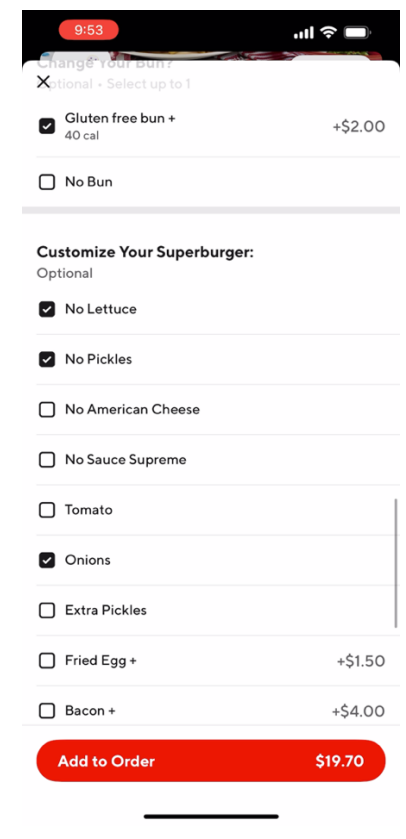
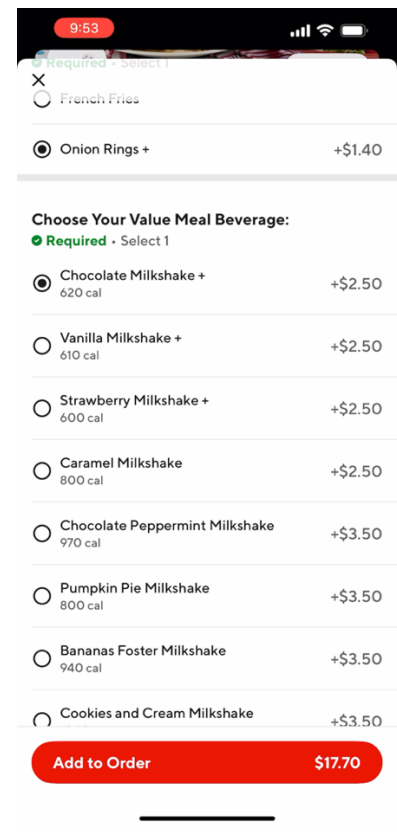
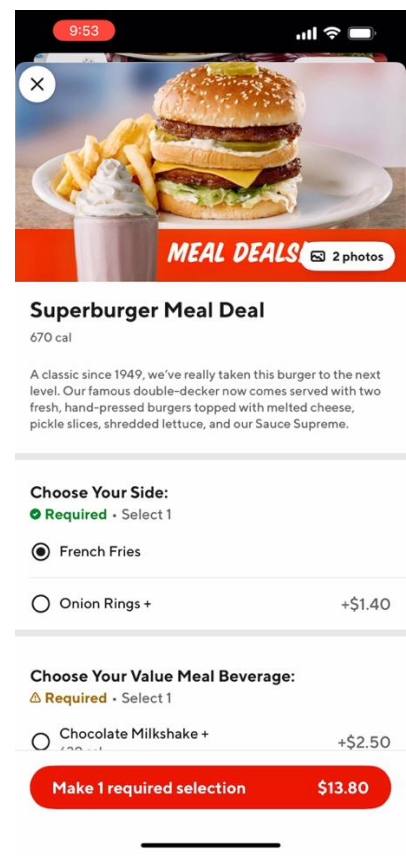
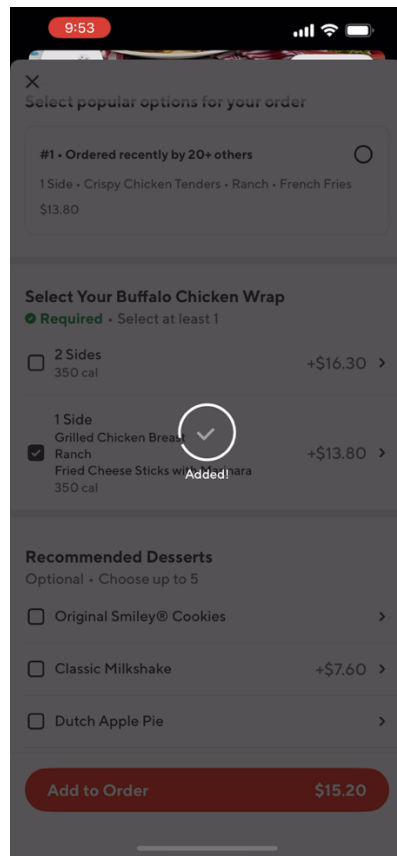
Choose Your Side:

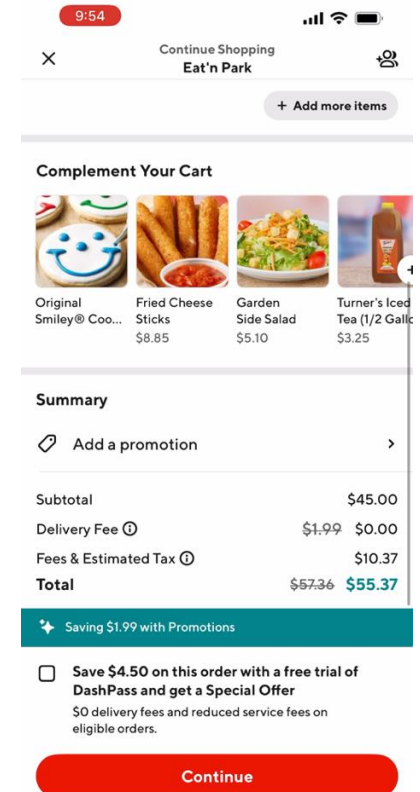
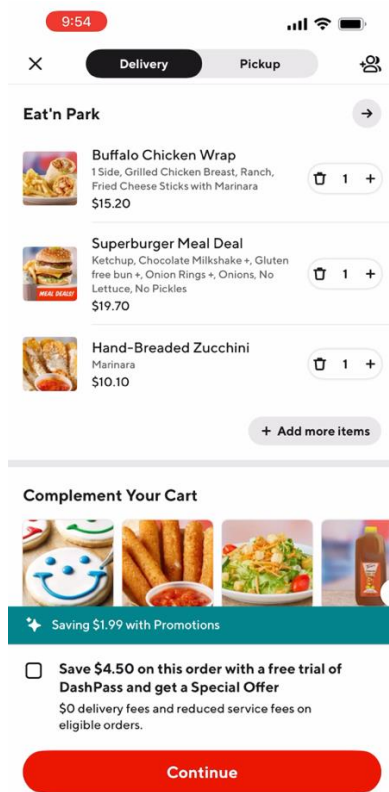
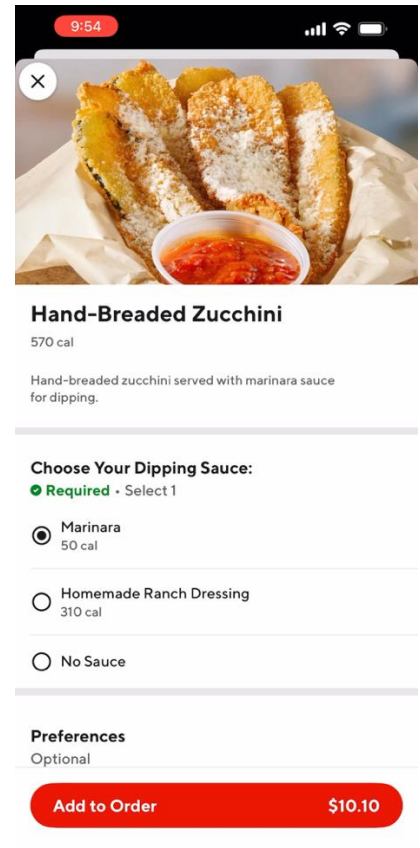
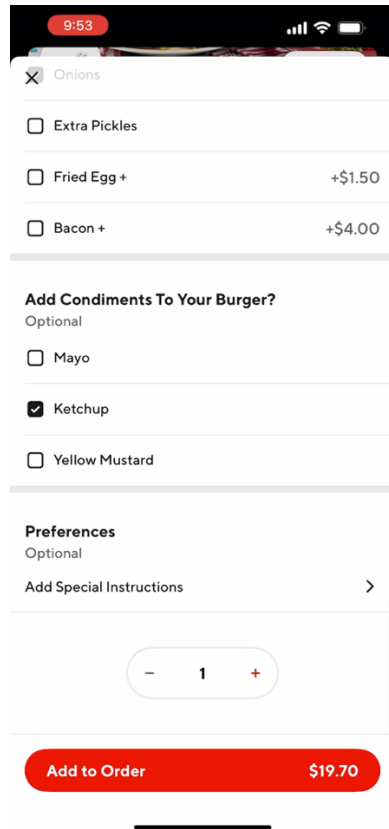
Required • Select 1

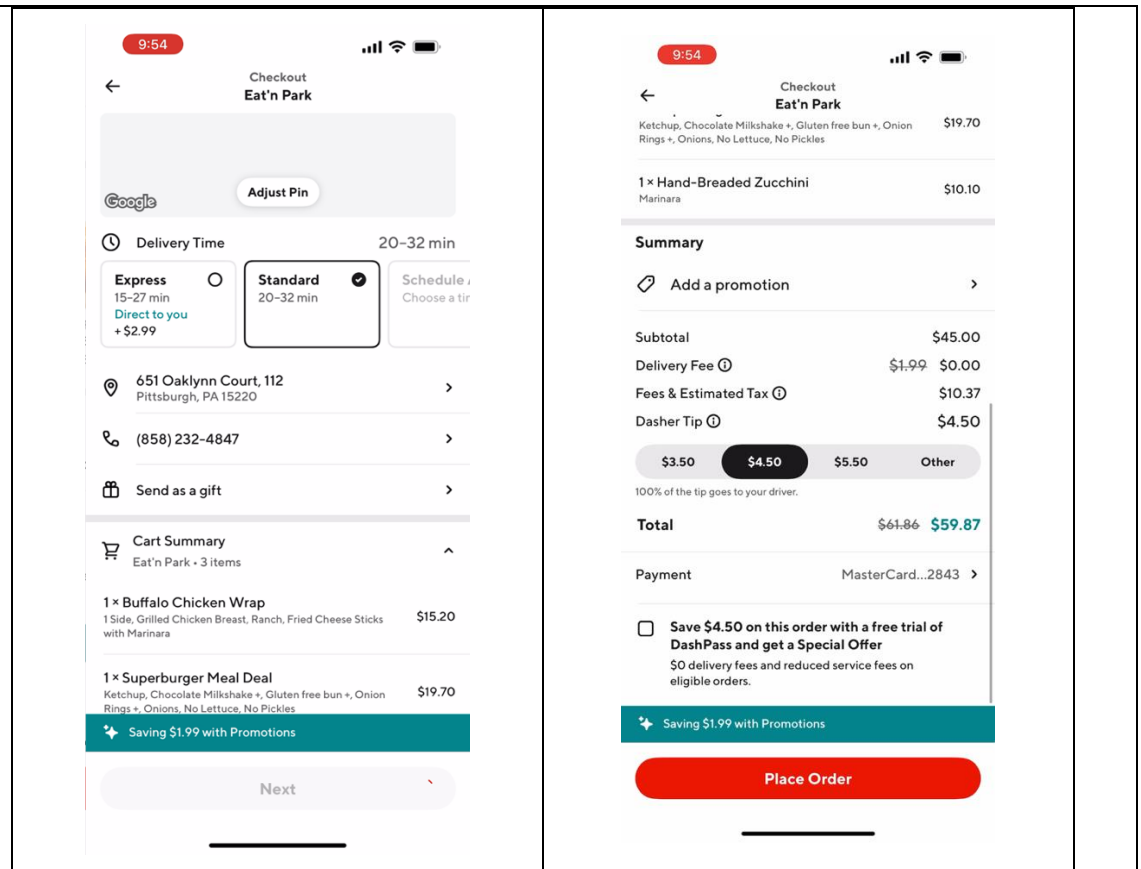
Fried Cheese Sticks with Marinara +\$1.40
295 cal

Fried Cheese Sticks with Ranch +\$1.40
545 cal

Make 1 required selection



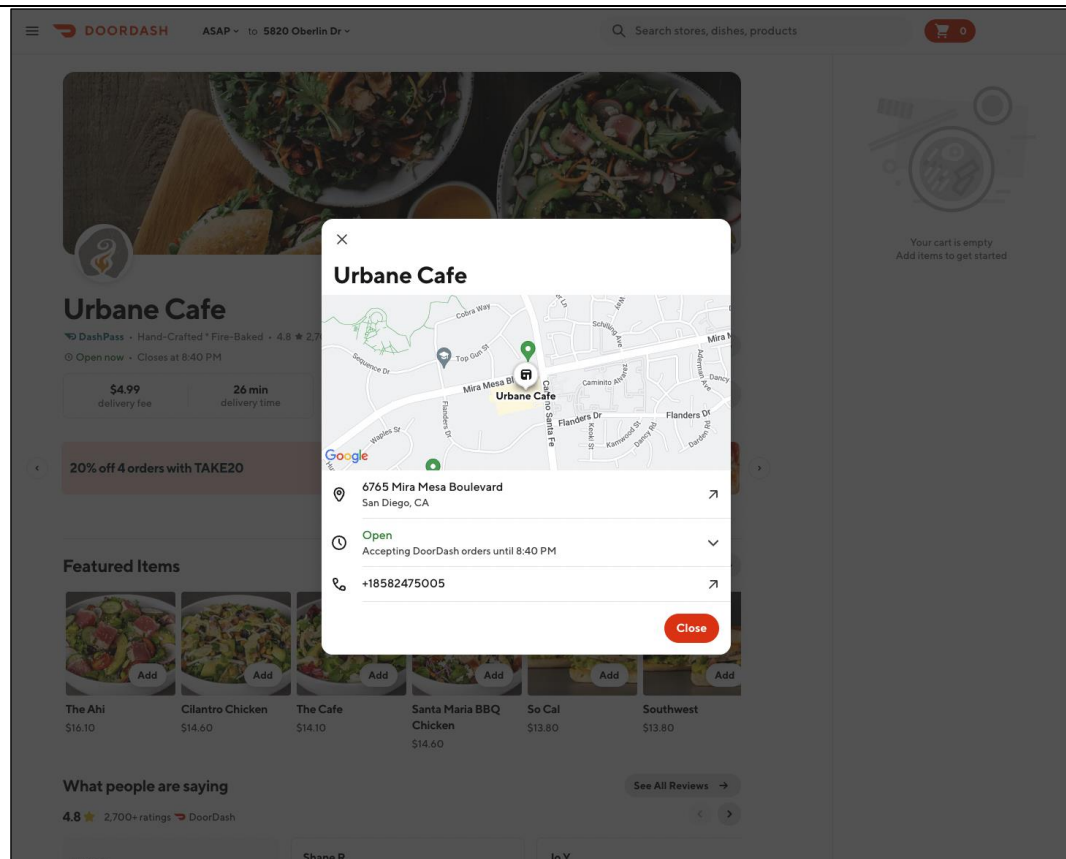




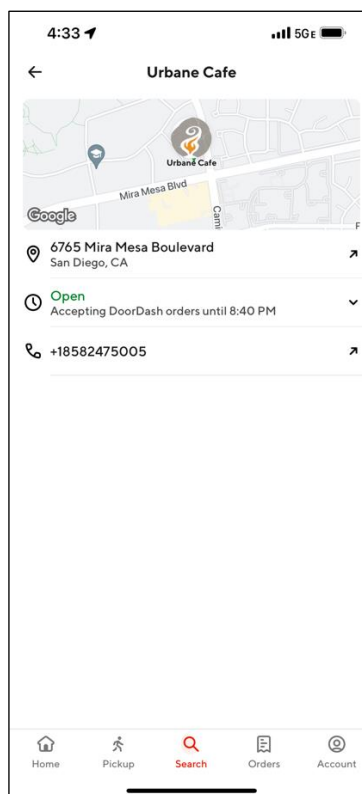
The Dasher App (iOS and android) integrate with Google Maps, through the Google Maps API (see Exh. 66, 67, 68, 69) to provide in app real time navigation to delivery drivers (Dashers). As stated in the google documentation:

“...with the Maps SDK for iOS, you can add maps based on Google maps data to your application. The SDK automatically handles access to the Google Maps servers, map display, and response to user gestures such as clicks and drags. You can also add markers, polylines, ground overlays and info windows to your map. These objects provide additional information for map locations and allow user interaction with the map.”

On the website the user can see where a particular restaurant from which they might order is located from the restaurants page on DoorDash.com. After sign-in, selecting the delivery address, and clicking on a restaurant from which to possibly order, takes the user to that restaurants page. There, the user clicks the more info icon, and the restaurant info popup appears displaying the exact location on a google map (that's integrated into the DoorDash webapp) like so:



The same holds true for the iOS and Android app. In the iOS app, once the user has navigated to the restaurant page, clicking the more info icon yields the following popup.



Google Maps API is part of the technology stack integrated with the DoorDash platform as indicated by the analysis at webtechsurvey.com (see Exh. 42)

The screenshot shows the WebTechSurvey website interface. The navigation bar includes 'Home', 'Web Technologies', 'HTTP Headers', 'Sign In', and 'Free Sign Up'. A search bar is present with the text 'Lookup website or web technology' and a 'SEARCH' button. The main content area displays a list of technologies used by DoorDash, categorized into sections:

- HubSpot**
 - get1.doordash.com, Used since Apr 2, 2022
 - partnerships.doordash.com, Used since Apr 5, 2022
- Maps**
 - Google Maps API
 - partnerships.doordash.com, Used since Apr 5, 2022
 - www.doordash.com, Used since Apr 1, 2022
- Marketing Automation**
 - Q4inc
 - ir.doordash.com, Used since Apr 8, 2022
- Marketing Analytics**
 - HubSpot Analytics
 - partnerships.doordash.com, Used since Apr 5, 2022
- Miscellaneous**
 - Amazon S3
 - cdn.doordash.com, Used since Apr 6, 2022
 - common.doordash.com, Used since Apr 9, 2022
 - driver.doordash.com, Used since Apr 6, 2022
 - support.doordash.com, Used since Apr 7, 2022
 - typography.doordash.com, Used since Apr 8, 2022
 - www.merchandising.doordash.com, Used since May 13, 2022

In the blog article by the DoorDash Engineering team titled “Augmenting Google Maps to Power Local Commerce Delivery” (See Exh 69) dated July 6, 2022, they describe how the DoorDash system is integrated with the Google Maps API. The premise of the article is that they had to augment:

“...Google Maps [integration] by capturing correct addresses and micro-routing Dashers with precise delivery locations and contextual navigation data during every step in the delivery journey.”

In the video interview “SFBigAnalytics_20220920: DoorDash: Building Scalable Real Time Event Processing with Kafka and Flink” by Allen Wang “[the] lead engineer at DoorDash working on real-time data infrastructure” (see Exh 120), he admits the following:

“...and therefore we fine-tuned our topic and the proxy producers configuration to achieve high efficiency and throughput on the topic level we use replication factor of two with one minimum in sync replica and compare with the typical configuration of three replicas and this saved this disk space and reduces the broker's CPU utilization on replication while still providing adequate data redundancy.”

DoorDash's Unified Chat Experience

The DoorDash engineering team stated in an engineering blog post, titled **Building a Unified Chat Experience at DoorDash**, how they successfully integrated chat into their overall system, addressing the challenges and strategies employed to create a unified chat experience. Their implementation aligns with MFCCS as it provides a centralized system layer architecture enhancing communication between the web server and multiple wireless handheld devices, each operating on distinct mobile systems thus allowing customers to communicate with customer support and dashers. (See Exh. 102)

DoorDash recognized the crucial need to address customer issues on a large scale, leading them to acknowledge the essential nature of a unified chat feature. Their initial challenge was a fragmented chat system, which was scattered across various user groups and platforms, causing duplicated efforts and central adoption of best practices to be challenging.

By leveraging the architecture MFCCS, DoorDash could centralize its chat functionalities, allowing for seamless interactions between consumers, Dashers, and merchants. The MFCCS would facilitate the web server's ability to communicate with diverse handheld devices, presenting mobile-compatible versions of the hospitality application. This integration is crucial for enabling user-initiated actions and subsequent selection of choices directly from the touchscreens of various devices, fostering a multi-modal, multi-protocol communication environment.

According to the engineering article, a significant objective for DoorDash was to streamline their support processes through automation, allowing customers to retrieve information through the app itself, thus minimizing the need for human support interaction. They aimed to develop a single chat support platform that would ensure consistent user experiences across different platforms and facilitate the reuse of components.

In building this unified platform, DoorDash created a single chat implementation that functioned for both customer support and communication between consumers and Dashers. This system, powered by Sendbird, was directly integrated into the consumer and Dasher apps, providing a uniform experience across all DoorDash chats and allowing for shared enhancements across different chat flows.

The backend system was strategically designed with multiple layers, distinguishing between internal services and third-party functionalities such as chat natural language processing. This separation expedited development by eliminating the need to create distinct, non-essential features.

Integration with their Decision Engine platform was another crucial step, which enabled operations to be automated, such as order updates, issuing credits, and personalized messaging. This significantly decreased the dependency on manual operations. DoorDash also carefully monitored several key performance indicators, including feedback from agents and customers, error rates, customer satisfaction, average handle time, first contact resolution, and manual tasks per order.

The outcomes of this updated chat support system, according to the article, were noteworthy, with reduced escalation rates and improved customer satisfaction. The automation led to a decrease in manual interactions per delivery, and customers were able to solve their problems more quickly, sometimes without the need for manual

| | |
|---|--|
| | <p>assistance. The time needed to implement additional automation was also reduced, owing to the common platform processing. Moreover, the common UI layers ensured that the chat feature was seamlessly integrated into the applications (iOS and Android), providing a consistent brand experience across all platforms.</p> <p>The advancements DoorDash has made in their chat system encapsulate the essence of the Middleware/Framework Communications Control Software (MFCCS) described in the patent claim. By creating a unified chat platform that seamlessly integrates with multiple user interfaces and operating systems, DoorDash has effectively embodied the centralized system layer architecture that MFCCS envisions. Their backend, which differentiates between core and third-party functionalities, mirrors the multi-layered approach of MFCCS, allowing for efficient, scalable communication across various devices. Moreover, the phased rollout and meticulous performance monitoring align with the adaptive, intelligent operational capabilities that MFCCS aims to provide. This demonstrates a real-world application of the patent's claim, showcasing a system that not only supports multi-mode communication but also enhances user experience and operational efficiency in a dynamic hospitality market.</p> <p>The accused instrumentality meets this limitation under the doctrine of equivalents because the accused instrumentality performs substantially the same function, in substantially the same way to achieve substantially the same result. For example, the doctrine of equivalence, in the context of the provided information regarding DoorDash's integration of external software APIs and optimization of their platform, refers to the principle that even if specific elements or technologies used may vary from those mentioned in a patent claim, the overall function, purpose, and outcome achieved by DoorDash remain substantially equivalent to what is described in the claim. In other words, DoorDash has implemented technologies and strategies that may not be identical to those explicitly outlined in the claim but still achieve the same essential objectives. This doctrine acknowledges that variations in implementation are acceptable as long as they maintain the core principles and goals described in the patent claim. DoorDash's integration of external APIs, optimization of API designs, and creation of a centralized chat system all align with this doctrine by achieving substantially equivalent results to the claim elements related to integration and efficiency in their platform.</p> |
| <p>additionally, the network is further enabled to automatically contact one or more other entities when the remotely initiated hospitality task cannot be completed with a first entity and to intelligently continue this, until the task is completed;</p> | <p>DoorDash's system architecture presents a compelling example of an intelligent network as described in Claim 1 of the patent, designed to foster seamless communication between its technological components and management staff. The company's use of Apache Flink enables real-time detection of user sessions and the strategic delivery of notifications, reflecting the intelligent and responsive nature of the network. This capability ensures that alerts are not only timely but also relevant, enhancing the decision-making processes and operational efficiency of management staff.</p> <p>The architecture's backend-focused approach to notifications ensures that communication is reflective of actual user engagement. This sophisticated handling of session data reduces unnecessary communications and targets the precise moments for interaction, a core feature of the intelligent network described in the patent. Such precision in communication exemplifies the patent's emphasis on the efficiency of server operations and the importance of real-time data processing.</p> |

DoorDash's implementation of Machine Learning further underscores the network's intelligent design. The system learns from vast datasets, optimizing the matching of orders with delivery personnel and predicting preparation and delivery times. This continuous learning process, integral to the network's intelligence, facilitates swift adjustments to operational parameters, ensuring a high level of service and responsiveness to changing conditions in the delivery environment.

The transition to a microservices architecture has enabled DoorDash to enhance the reliability and scalability of its platform. By compartmentalizing functions into domain-specific services, DoorDash has reduced errors and latency, a testament to the network's robust design. This architectural shift not only supports the scalability required for a growing platform but also embodies the resilience and efficiency envisioned by the patent.

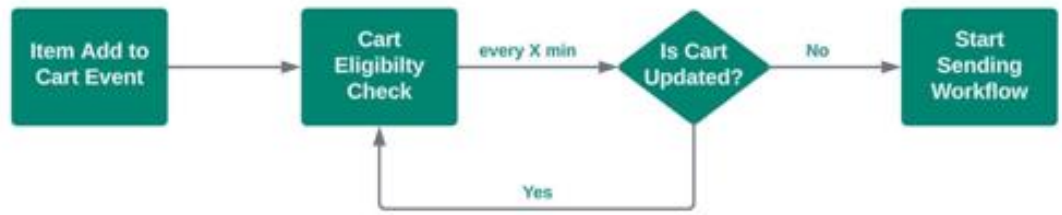
The central data platform and ML Workbench are the linchpins of DoorDash's intelligent network. They act as the operational core, where data is not just stored but intelligently processed to inform decisions across the platform. This centralized data handling and the application of Machine Learning algorithms for predictive analytics are in direct alignment with the intelligent network capabilities set forth in the patent. DoorDash's system architecture thus presents a clear real-world embodiment of the patent's visionary communication network, driving efficiency and intelligence in the domain of hospitality services.

Notifications

The MFCCS is evident through DoorDash's strategic use of real-time notifications to engage consumers and enhance user experience as detailed by their engineering team in the blog article titled "**Leveraging Flink to Detect User Sessions and Engage DoorDash Consumers with Real-Time Notifications**". By harnessing the capabilities of Apache Flink, DoorDash has been able to detect user sessions and discern the critical moments to send push notifications, directly aligning with the objectives of the MFCCS to facilitate intelligent, timely communication between the server and the user's handheld devices. (See Exh. 127)

This approach to session detection and notification dispatch represents the essence of MFCCS. The sessions on DoorDash's platform are managed in a way that reflects the centralized control of MFCCS, grouping user activities into sessions and determining inactivity, thus optimizing the timing for sending out notifications. This reduces unnecessary communication attempts, which is a core feature of the MFCCS, aiming to enhance the efficiency of the server's operations.

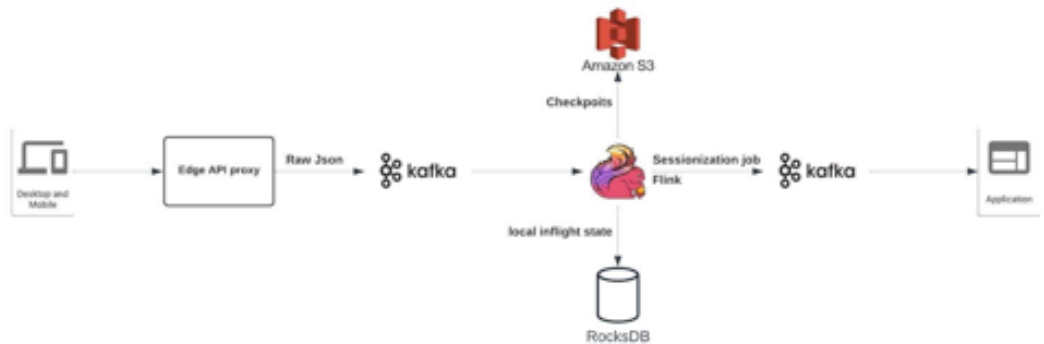
The previous notification system at DoorDash relied on a front-end triggered approach that lacked the precision of the current model, often resulting in untimely and less effective user notifications which did not reflect actual user engagement or session activity accurately, as shown below:



DoorDash's new backend-focused strategy (see diagram below) for triggering notifications ensures that the system accurately mirrors actual user activity, an essential aspect of the MFCCS's design to ensure relevance and personalization of user interactions. The sessionization platform developed by DoorDash utilizes in-memory streaming for processing events, further embodying the MFCCS principle of leveraging real-time data for improved service delivery.



The infrastructure supporting this system (see diagram below) mirrors the robustness required by MFCCS, capable of handling large-state computing and managing user events in a stateful manner until the appropriate action is determined. This is in line with MFCCS's aim to manage multiple communication modes and protocols, ensuring that the server selects the most efficient and effective form of user engagement.



The tangible results of integrating such an intelligent communication system is that DoorDash's cart abandonment notifications have not only increased order volume and revenue but also the speed of notification delivery by six-fold, leading to higher user engagement rates. This success is a testament to the MFCCS's capability to provide an adaptable.

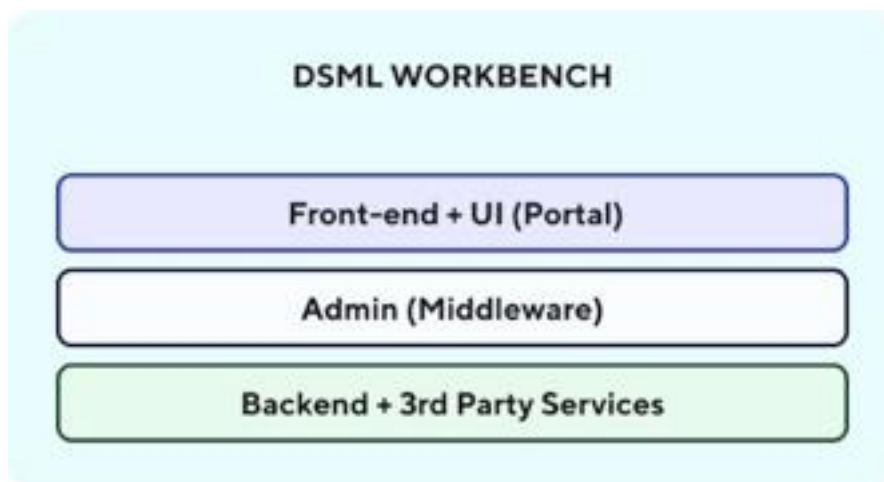
Machine Language

In the blog article by the DoorDash Engineering team title “Transforming MLOps at DoorDash with Machine Learning Workbench”, states how DoorDash has developed an internal Machine Learning (ML) Workbench to enhance data operations and support their data scientists, analysts, and AI/ML engineers. The importance of ML at DoorDash is emphasized, given its applications across the platform's ecosystem involving customers, Dashers, and merchants. The ML Workbench serves as a centralized hub for

tasks in the machine learning lifecycle, including building, training, tuning, and deploying machine learning models. (See Exh. 121)

DoorDash's ML Workbench streamlines the machine learning process by providing a space for data collection, organization, and use in ML algorithms. It was designed with user-centered principles, aiming to create a best-in-class internal tool that is functional, usable, aesthetically pleasing, and integrates well with DoorDash's internal tools ecosystem. The development strategy was iterative, focusing on understanding user pain points, designing solutions, running user tests, and optimizing for better velocity and productivity.

Key components of the ML Workbench include front-end UI (Portal), Admin (Middleware), and Backend + 3rd Party Services. Through user research and interviews, DoorDash categorized users into admins, end users, and operators, tailoring the Workbench's features to their needs. Features such as model viewing, testing predictions, and model performance monitoring were added to streamline daily workflows and accelerate model development velocity.



The implementation of the ML Workbench has led to significant improvements in the efficiency and user experience of DoorDash's engineering and data science teams. The tool facilitated better observability of features and models, and the future vision includes diversifying adoption, improving feature and model observability, and continuing a user-centric development approach.

This ML Workbench and its capabilities align well with Claim 1 of the patent, as it is an embodiment of an interconnected, intelligent web server network with multi-modes of contact, multi-communications protocols, and multi-user and parallel operational capabilities. The master database's role within this framework is also showcased, where it integrates with the network of web servers and is accessible via a database API, which intelligently learns, updates, and stores data. The Workbench's integration with various ML lifecycle stages and its emphasis on efficiency and reliability reflect the innovations described in Claim 1, particularly in the context of improving network efficiency and reliability.

DoorDash Iguazu Event Processing System

In the technical presentation video "Building Scalable Real Time Event Processing with Kafka and Flink" by the DoorDash Engineering team (see Exh 91) they present that

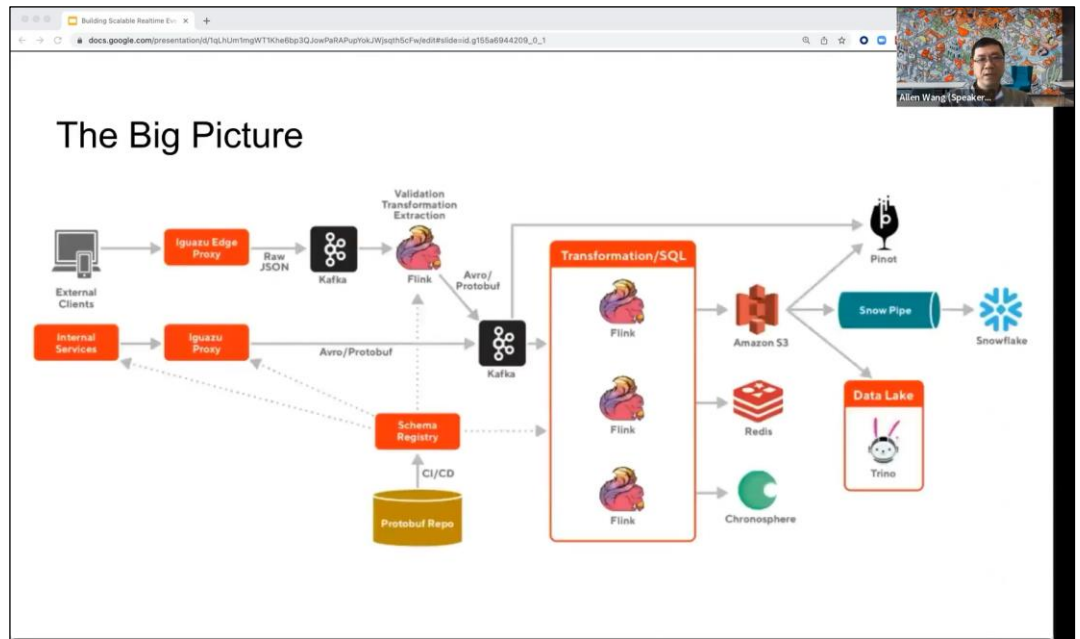
“...[t]wo years ago, DoorDash started the journey of creating a real time event processing system to replace the legacy data pipelines and address our event processing needs to scale our business. We created a scalable system that could handle heterogeneous data sources and destinations, was easily accessible with different levels of abstractions and had end to end schema enforcement. We were able to accomplish this by shifting our strategy from heavily relying on AWS and third-party data services to leveraging open source frameworks that can be customized and better integrated with our infrastructure.

In this session, I will share what we learned and how we put together this system with Apache Flink, Kafka as well as Kubernetes.”

They discuss the Iguazu real time event processing system they development from scratch and it’s architecture.”

The screenshot shows a presentation slide titled "Introducing Iguazu". The slide content is as follows:

- Iguazu - building real time event processing from scratch
 - Emphasis on integration of open source frameworks with DoorDash's infrastructure
- Current scale and SLO
 - **Hundreds of billions** events processed per day
 - Hundreds of events and pipelines
 - **99.99%** delivery rate
 - Sub second latency for streaming consumers, and minutes latency to data warehouse



The diagram above depicts the layered system architecture of Iguazu, DoorDash’s event processing system. It depicts the inner functioning of Iguazu taking from the DoorDash engineering blog article “Building Scalable Real Time Event Processing with Kafka and Flink” (see Exh 53) and the DoorDash technical video “Scaling our Data Platform” (see Exh 47).

In the DoorDash engineering blog article (see Exh 53) “Building Scalable Real Time Event Processing with Kafka and Flink”, they state that:

“...[t]wo years ago, we started the journey of creating a real time event processing system named Iguazu to replace the legacy data pipelines and address the following event processing needs we anticipated as the data volume grows with the business:

- ***Heterogeneous data sources and destinations:*** *Data ingest from a variety of data sources including the legacy monolithic web application, microservices and mobile/web devices, and delivery to different destinations including third-party data services. Reliable and low latency data ingest into the data warehouse is a high priority.*
- ***Easily accessible:*** *A platform that makes it easy for different teams and services to tap into the streams of the data and build their own data processing logic.*
- ***End-to-end schema enforcement and schema evolution:*** *Schema only improves data quality, but also facilitates easy integration with data warehouses and SQL processing.*
- ***Scalable, fault-tolerant, and easy to operate for a small team:*** *We want to build a system that can easily scale to the business need with minimal operational overhead.”*

Kafka is an open-source platform from Apache, excels in stream processing and data management. It adeptly handles the reception, storage, organization, and distribution of voluminous data streams to a diverse array of end-users and applications. However, the influx of substantial data payloads—spanning hundreds to thousands of messages—into Kafka's servers can precipitate challenges such as data overloading and duplication. These issues, in turn, can result in the data within Kafka servers becoming disorganized and obscured, complicating effective data management and utilization.

Apache Kafka's architecture includes a key structural element known as Topics, which serve as the primary unit for organizing events or messages. Essentially, Kafka Topics function as virtual groups or logs, systematically storing messages and events in a sequential manner. This organization facilitates the seamless transmission of data between Kafka servers. Each topic acts as a dynamic repository, where messages sent by producers are chronologically appended, forming an evolving log file.

In practice, producers inject messages into these topics, adding to the tail end of the log, while consumers extract messages from specified topics, ensuring a streamlined flow of data. This methodology enables logical segregation of messages and events, akin to how distinct tables in a database hold varied types of data. In the Kafka ecosystem, the creation of multiple topics is permitted, catering to diverse use cases. Crucially, each topic must bear a unique, identifiable name to maintain clear distinction among various

Kafka brokers within a Kafka cluster, thereby ensuring efficient data management and retrieval.

The article from DoorDash Engineering Blog titled "API-First Approach to Kafka Topic Creation" discusses how DoorDash's Engineering teams have improved their Kafka Topic creation process. They replaced a Terraform/Atlantis-based approach with an in-house API Infrastructure Service, which has led to a 95% reduction in real-time pipeline onboarding time and has saved numerous developer hours. (See Exh. 126)

DoorDash's Real-Time Streaming Platform (RTSP) team, part of the Data Platform organization, manages over 2500 Kafka Topics. Kafka functions as the publication-subscription layer of the Iguazu pipeline and processes around six billion messages per day.

The article explains the challenges with the legacy architecture, where provisioning Kafka Topics was slow due to on-call engineer approval and prone to failures, increasing the on-call load. The RTSP team worked with storage and cloud teams to automate Kafka resource creation, which allowed for a more streamlined process and reduced manual intervention.

DoorDash has introduced super-user accounts to resolve merge conflicts that occurred in ACL files for Iguazu users, which has significantly sped up applications by Atlantis. The new architecture integrates with the Storage Self-Serve Platform within Infra Service, which provides an API to perform CRUD operations on infrastructure components. This new approach allows for provisioning approximately 100 new topics every week without manual intervention and significantly faster onboarding times for customers.

The article concludes by outlining the future direction of Kafka Automation and Storage Self-Serve, aiming to improve guardrails and customer experience. Acknowledgments are given to the various teams and engineers who contributed to this engineering win. This article demonstrates DoorDash's commitment to improving their operational efficiency and infrastructure management, ensuring faster onboarding times, and reducing the need for manual interventions, which contributes to a better overall experience for their customers and partners.

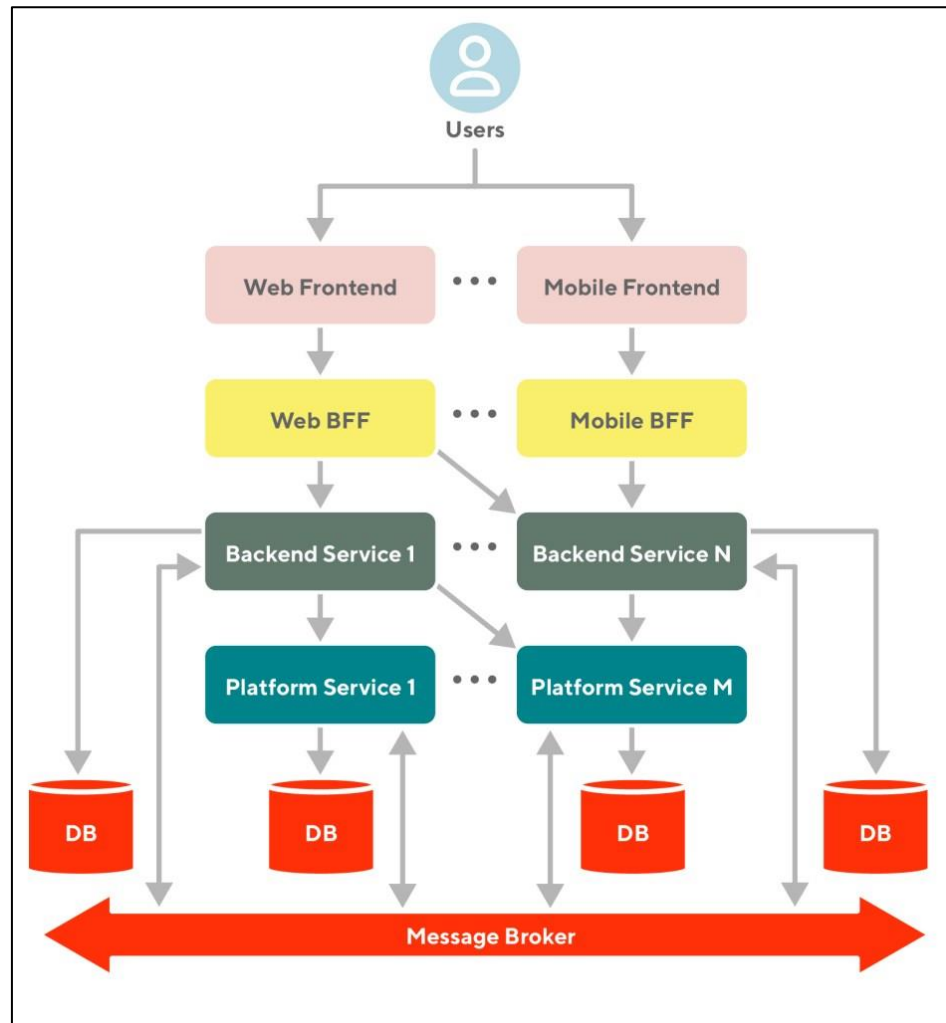
Microservices Layered Architecture

The original DoorDash platform was originally a monolithic application written in Python using the Django web framework with a PostgreSQL database. As the platform grew, they started to have problems with reliability and scaling. Around 2018 they institute a code freeze and began migrating to microservices. At this time, they also migrated to the Kotlin language, and their services now run on the java virtual machine (JVM) (see Exh 20, 21, 22).

DoorDash was able to improve the efficiency and reliability of their platform using a microservices architecture. By breaking up their application into domain-specific parts they were able to reduce errors and latency. They state in the article "Future-proofing: How DoorDash Transitioned from a Code Monolith to a Microservice Architecture" (see Exh. 20) that:

"... final design for our new microservice architecture consisted of five different layers, ranging from the user experience to the core infrastructure. Each layer

provides functionality to the upper layer and leverages the functionality exposed by the lower layer [as shown below]”:



These layers include: (see Exh 20):

“Frontend layer: Provides frontend systems (like the DoorDash mobile app, Dasher web app, etc.) for the interaction with consumers, merchants, and Dashers that are built on top of different frontend platforms.

BFF layer: The frontend layer is decoupled from the backend layer via BFFs. The BFF layer provides functionality to the frontend by orchestrating the interaction with multiple backend services while hiding the underlying backend architecture.

Backend Layer: Provides the core functionality that powers the business logic (order cart service, feed service, delivery service, etc.).

Platform layer: Provides common functionality that is leveraged by other backend services (identity service, communication service, etc.).

| | |
|--|--|
| | <p><i>Infrastructure layer: Provides the infrastructural components that are required to build the site (databases, message brokers, etc.) and lays the foundation to abstract the system from the underlying environment (cloud service provider)."</i></p> <p>In their new architecture they introduced the backend-for-frontend (BFF) "... an application connecting the consumer-facing client and the services providing general purpose APIs. Client requests go to the BFF, which then orchestrates the aggregation of information needed by the client.". The BFF is a software architecture pattern (see Exh 40) used by microservices which "...shifted from thick-client applications to interfaces delivered via the web, a trend that has also enabled the growth of SAAS-based solutions in general". As such BFF can be considered as "...the user-facing application as being two components - a client-side application living outside your perimeter, and a server-side component (the BFF) inside your perimeter". The perimeter of the BFF is the webserver.</p> <p>The accused instrumentality meets this limitation under the doctrine of equivalents because the accused instrumentality performs substantially the same function, in substantially the same way to achieve substantially the same result. For example, DoorDash's network utilizes a sophisticated algorithmic framework that mirrors the intelligent operational capabilities of the patented system. It dynamically adjusts to real-time data for delivery logistics, employs predictive analytics for operational efficiency, and incorporates a centralized data management system for streamlined communication, much like the patent's description of an intelligent network. This implementation achieves the same overarching goal of efficient service delivery, adhering to the principles of the doctrine of equivalents.</p> |
| <p>additionally the network is further enabled to intelligently apply time constraints and/or hold times, associated with the hospitality task, when applicable;</p> | <p>DoorDash's system architecture presents a compelling example of an intelligent network as described in Claim 1 of the patent, designed to foster seamless communication between its technological components and management staff. The company's use of Apache Flink enables real-time detection of user sessions and the strategic delivery of notifications, reflecting the intelligent and responsive nature of the network. This capability ensures that alerts are not only timely but also relevant, enhancing the decision-making processes and operational efficiency of management staff.</p> <p>The architecture's backend-focused approach to notifications ensures that communication is reflective of actual user engagement. This sophisticated handling of session data reduces unnecessary communications and targets the precise moments for interaction, a core feature of the intelligent network described in the patent. Such precision in communication exemplifies the patent's emphasis on the efficiency of server operations and the importance of real-time data processing.</p> <p>DoorDash's implementation of Machine Learning further underscores the network's intelligent design. The system learns from vast datasets, optimizing the matching of orders with delivery personnel and predicting preparation and delivery times. This continuous learning process, integral to the network's intelligence, facilitates swift adjustments to operational parameters, ensuring a high level of service and responsiveness to changing conditions in the delivery environment.</p> <p>The transition to a microservices architecture has enabled DoorDash to enhance the</p> |

reliability and scalability of its platform. By compartmentalizing functions into domain-specific services, DoorDash has reduced errors and latency, a testament to the network's robust design. This architectural shift not only supports the scalability required for a growing platform but also embodies the resilience and efficiency envisioned by the patent.

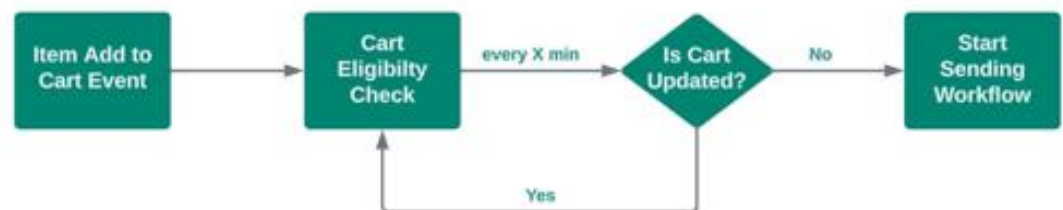
The central data platform and ML Workbench are the linchpins of DoorDash's intelligent network. They act as the operational core, where data is not just stored but intelligently processed to inform decisions across the platform. This centralized data handling and the application of Machine Learning algorithms for predictive analytics are in direct alignment with the intelligent network capabilities set forth in the patent. DoorDash's system architecture thus presents a clear real-world embodiment of the patent's visionary communication network, driving efficiency and intelligence in the domain of hospitality services.

Notifications

The MFCCS is evident through DoorDash's strategic use of real-time notifications to engage consumers and enhance user experience as detailed by their engineering team in the blog article titled "**Leveraging Flink to Detect User Sessions and Engage DoorDash Consumers with Real-Time Notifications**". By harnessing the capabilities of Apache Flink, DoorDash has been able to detect user sessions and discern the critical moments to send push notifications, directly aligning with the objectives of the MFCCS to facilitate intelligent, timely communication between the server and the user's handheld devices. (See Exh. 127)

This approach to session detection and notification dispatch represents the essence of MFCCS. The sessions on DoorDash's platform are managed in a way that reflects the centralized control of MFCCS, grouping user activities into sessions and determining inactivity, thus optimizing the timing for sending out notifications. This reduces unnecessary communication attempts, which is a core feature of the MFCCS, aiming to enhance the efficiency of the server's operations.

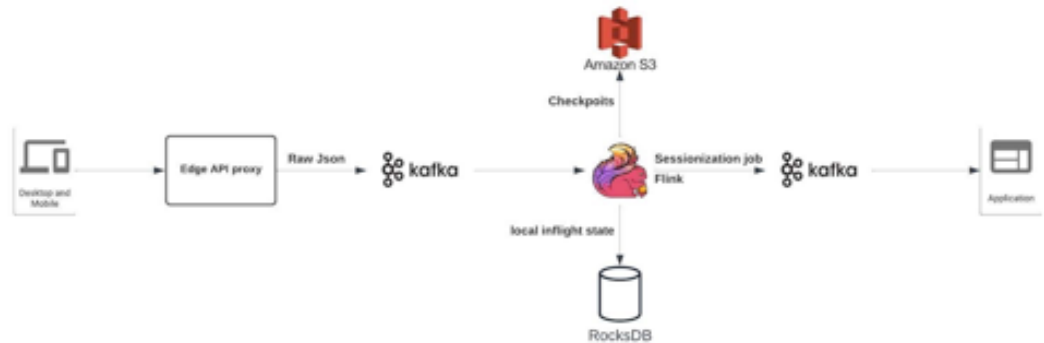
The previous notification system at DoorDash relied on a front-end triggered approach that lacked the precision of the current model, often resulting in untimely and less effective user notifications which did not reflect actual user engagement or session activity accurately, as shown below:



DoorDash's new backend-focused strategy (see diagram below) for triggering notifications ensures that the system accurately mirrors actual user activity, an essential aspect of the MFCCS's design to ensure relevance and personalization of user interactions. The sessionization platform developed by DoorDash utilizes in-memory streaming for processing events, further embodying the MFCCS principle of leveraging real-time data for improved service delivery.



The infrastructure supporting this system (see diagram below) mirrors the robustness required by MFCCS, capable of handling large-state computing and managing user events in a stateful manner until the appropriate action is determined. This is in line with MFCCS's aim to manage multiple communication modes and protocols, ensuring that the server selects the most efficient and effective form of user engagement.



The tangible results of integrating such an intelligent communication system is that DoorDash's cart abandonment notifications have not only increased order volume and revenue but also the speed of notification delivery by six-fold, leading to higher user engagement rates. This success is a testament to the MFCCS's capability to provide an adaptable.

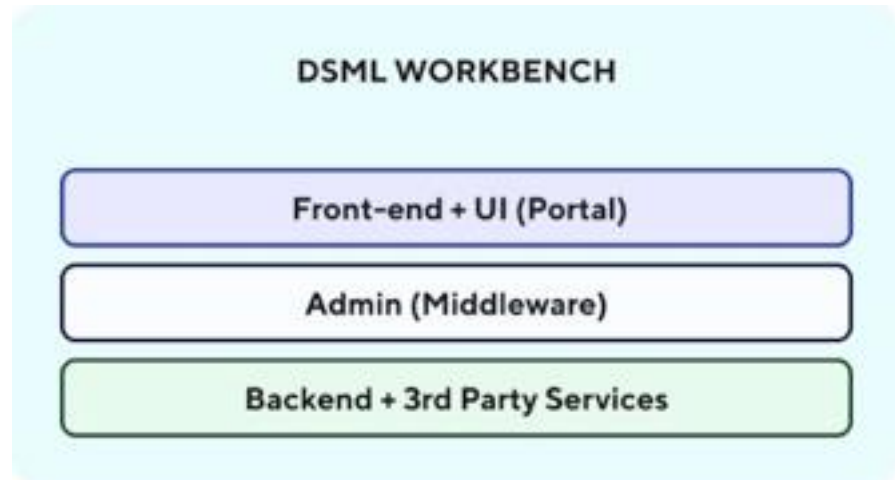
Machine Language

In the blog article by the DoorDash Engineering team titled “Transforming MLOps at DoorDash with Machine Learning Workbench”, states how DoorDash has developed an internal Machine Learning (ML) Workbench to enhance data operations and support their data scientists, analysts, and AI/ML engineers. The importance of ML at DoorDash is emphasized, given its applications across the platform's ecosystem involving customers, Dashers, and merchants. The ML Workbench serves as a centralized hub for tasks in the machine learning lifecycle, including building, training, tuning, and deploying machine learning models. (See Exh. 121)

DoorDash's ML Workbench streamlines the machine learning process by providing a space for data collection, organization, and use in ML algorithms. It was designed with user-centered principles, aiming to create a best-in-class internal tool that is functional, usable, aesthetically pleasing, and integrates well with DoorDash's internal tools ecosystem. The development strategy was iterative, focusing on understanding user pain points, designing solutions, running user tests, and optimizing for better velocity and productivity.

Key components of the ML Workbench include front-end UI (Portal), Admin (Middleware), and Backend + 3rd Party Services. Through user research and interviews, DoorDash categorized users into admins, end users, and operators, tailoring the Workbench's features to their needs. Features such as model viewing, testing

predictions, and model performance monitoring were added to streamline daily workflows and accelerate model development velocity.



The implementation of the ML Workbench has led to significant improvements in the efficiency and user experience of DoorDash's engineering and data science teams. The tool facilitated better observability of features and models, and the future vision includes diversifying adoption, improving feature and model observability, and continuing a user-centric development approach.

This ML Workbench and its capabilities align well with Claim 1 of the patent, as it is an embodiment of an interconnected, intelligent web server network with multi-modes of contact, multi-communications protocols, and multi-user and parallel operational capabilities. The master database's role within this framework is also showcased, where it integrates with the network of web servers and is accessible via a database API, which intelligently learns, updates, and stores data. The Workbench's integration with various ML lifecycle stages and its emphasis on efficiency and reliability reflect the innovations described in Claim 1, particularly in the context of improving network efficiency and reliability.

DoorDash Iguazu Event Processing System

In the technical presentation video “Building Scalable Real Time Event Processing with Kafka and Flink” by the DoorDash Engineering team (see Exh 91) they present that

“...[t]wo years ago, DoorDash started the journey of creating a real time event processing system to replace the legacy data pipelines and address our event processing needs to scale our business. We created a scalable system that could handle heterogeneous data sources and destinations, was easily accessible with different levels of abstractions and had end to end schema enforcement. We were able to accomplish this by shifting our strategy from heavily relying on AWS and third-party data services to leveraging open source frameworks that can be customized and better integrated with our infrastructure.

In this session, I will share what we learned and how we put together this system with Apache Flink, Kafka as well as Kubernetes.”

They discuss the Iguazu real time event processing system they development from scratch and it's architecture.”

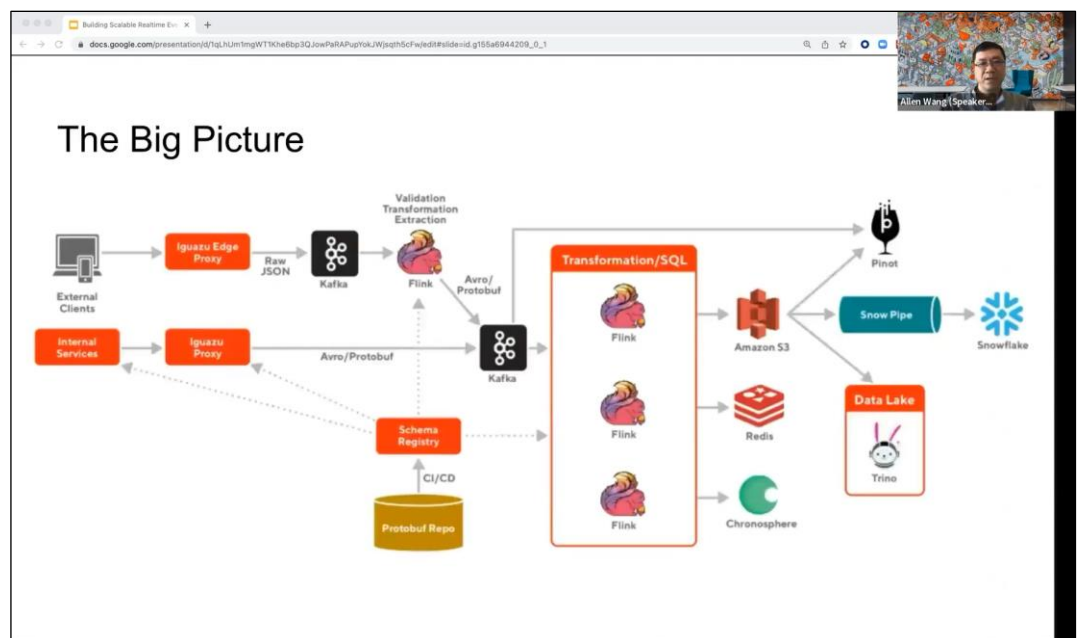
Building Scalable Realtime E...

docs.google.com/presentation/d/1qLhUm1mgWTk0he8p3QJowPaR4PuyYokJWjgqH5CFweldt#slide=id.g155a6944209_0_1

Allen Wang (Speaker...

Introducing Iguazu

- Iguazu - building real time event processing from scratch
 - Emphasis on integration of open source frameworks with DoorDash's infrastructure
- Current scale and SLO
 - **Hundreds of billions** events processed per day
 - Hundreds of events and pipelines
 - **99.99%** delivery rate
 - Sub second latency for streaming consumers, and minutes latency to data warehouse



The diagram above depicts the layered system architecture of Iguazu, DoorDash's event processing system. It depicts the inner functioning of Iguazu taking from the DoorDash engineering blog article "Building Scalable Real Time Event Processing with Kafka and Flink" (see Exh 53) and the DoorDash technical video "Scaling our Data Platform" (see Exh 47).

In the DoorDash engineering blog article (see Exh 53) "Building Scalable Real Time Event Processing with Kafka and Flink", they state that:

"...[t]wo years ago, we started the journey of creating a real time event processing system named Iguazu to replace the legacy data pipelines and address the following event processing needs we anticipated as the data volume grows with the business:

- **Heterogeneous data sources and destinations:** Data ingest from a variety of

data sources including the legacy monolithic web application, microservices and mobile/web devices, and delivery to different destinations including third-party data services. Reliable and low latency data ingest into the data warehouse is a high priority.

- ***Easily accessible:*** *A platform that makes it easy for different teams and services to tap into the streams of the data and build their own data processing logic.*
- ***End-to-end schema enforcement and schema evolution:*** *Schema only improves data quality, but also facilitates easy integration with data warehouses and SQL processing.*
- ***Scalable, fault-tolerant, and easy to operate for a small team:*** *We want to build a system that can easily scale to the business need with minimal operational overhead.”*

Kafka is an open-source platform from Apache, excels in stream processing and data management. It adeptly handles the reception, storage, organization, and distribution of voluminous data streams to a diverse array of end-users and applications. However, the influx of substantial data payloads—spanning hundreds to thousands of messages—into Kafka's servers can precipitate challenges such as data overloading and duplication. These issues, in turn, can result in the data within Kafka servers becoming disorganized and obscured, complicating effective data management and utilization.

Apache Kafka's architecture includes a key structural element known as Topics, which serve as the primary unit for organizing events or messages. Essentially, Kafka Topics function as virtual groups or logs, systematically storing messages and events in a sequential manner. This organization facilitates the seamless transmission of data between Kafka servers. Each topic acts as a dynamic repository, where messages sent by producers are chronologically appended, forming an evolving log file.

In practice, producers inject messages into these topics, adding to the tail end of the log, while consumers extract messages from specified topics, ensuring a streamlined flow of data. This methodology enables logical segregation of messages and events, akin to how distinct tables in a database hold varied types of data. In the Kafka ecosystem, the creation of multiple topics is permitted, catering to diverse use cases. Crucially, each topic must bear a unique, identifiable name to maintain clear distinction among various Kafka brokers within a Kafka cluster, thereby ensuring efficient data management and retrieval.

The article from DoorDash Engineering Blog titled "API-First Approach to Kafka Topic Creation" discusses how DoorDash's Engineering teams have improved their Kafka Topic creation process. They replaced a Terraform/Atlantis-based approach with an in-house API Infrastructure Service, which has led to a 95% reduction in real-time pipeline onboarding time and has saved numerous developer hours. (See Exh. 126)

DoorDash's Real-Time Streaming Platform (RTSP) team, part of the Data Platform organization, manages over 2500 Kafka Topics. Kafka functions as the publication-subscription layer of the Iguazu pipeline and processes around six billion messages per day.

The article explains the challenges with the legacy architecture, where provisioning Kafka Topics was slow due to on-call engineer approval and prone to failures,

increasing the on-call load. The RTSP team worked with storage and cloud teams to automate Kafka resource creation, which allowed for a more streamlined process and reduced manual intervention.

DoorDash has introduced super-user accounts to resolve merge conflicts that occurred in ACL files for Iguazu users, which has significantly sped up applications by Atlantis. The new architecture integrates with the Storage Self-Serve Platform within Infra Service, which provides an API to perform CRUD operations on infrastructure components. This new approach allows for provisioning approximately 100 new topics every week without manual intervention and significantly faster onboarding times for customers.

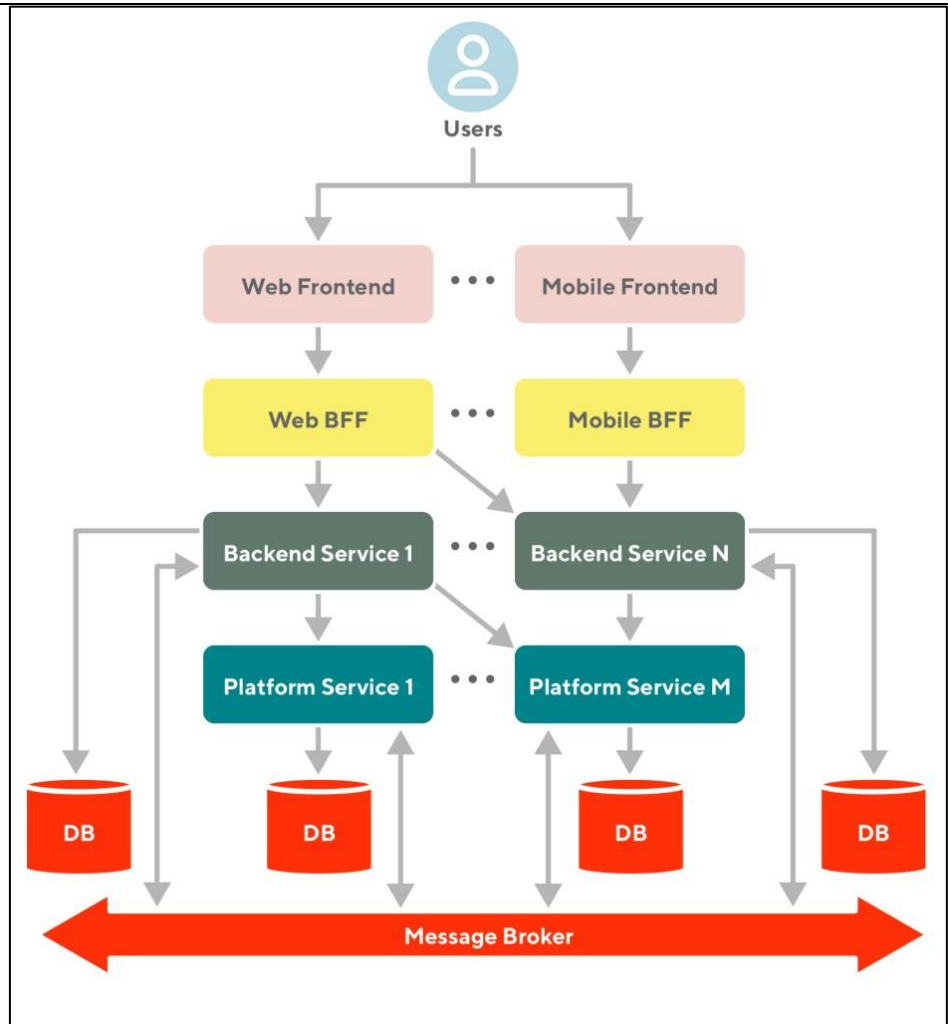
The article concludes by outlining the future direction of Kafka Automation and Storage Self-Serve, aiming to improve guardrails and customer experience. Acknowledgments are given to the various teams and engineers who contributed to this engineering win. This article demonstrates DoorDash's commitment to improving their operational efficiency and infrastructure management, ensuring faster onboarding times, and reducing the need for manual interventions, which contributes to a better overall experience for their customers and partners.

Microservices Layered Architecture

The original DoorDash platform was originally a monolithic application written in Python using the Django web framework with a PostgreSQL database. As the platform grew, they started to have problems with reliability and scaling. Around 2018 they institute a code freeze and began migrating to microservices. At this time, they also migrated to the Kotlin language, and their services now run on the java virtual machine (JVM) (see Exh 20, 21, 22).

DoorDash was able to improve the efficiency and reliability of their platform using a microservices architecture. By breaking up their application into domain-specific parts they were able to reduce errors and latency. They state in the article “Future-proofing: How DoorDash Transitioned from a Code Monolith to a Microservice Architecture” (see Exh. 20) that:

“... final design for our new microservice architecture consisted of five different layers, ranging from the user experience to the core infrastructure. Each layer provides functionality to the upper layer and leverages the functionality exposed by the lower layer [as shown below]”:



These layers include: (see Exh 20):

“Frontend layer: Provides frontend systems (like the DoorDash mobile app, Dasher web app, etc.) for the interaction with consumers, merchants, and Dashers that are built on top of different frontend platforms.

BFF layer: The frontend layer is decoupled from the backend layer via BFFs. The BFF layer provides functionality to the frontend by orchestrating the interaction with multiple backend services while hiding the underlying backend architecture.

Backend Layer: Provides the core functionality that powers the business logic (order cart service, feed service, delivery service, etc.).

Platform layer: Provides common functionality that is leveraged by other backend services (identity service, communication service, etc.).

Infrastructure layer: Provides the infrastructural components that are required to build the site (databases, message brokers, etc.) and lays the foundation to abstract the system from the underlying environment (cloud service provider).”

In their new architecture they introduced the backend-for-frontend (BFF) “... an application connecting the consumer-facing client and the services providing general

| | |
|--|---|
| | <p>purpose APIs. Client requests go to the BFF, which then orchestrates the aggregation of information needed by the client.”. The BFF is a software architecture pattern (see Exh 40) used by microservices which “...shifted from thick-client applications to interfaces delivered via the web, a trend that has also enabled the growth of SAAS-based solutions in general”. As such BFF can be considered as “...the user-facing application as being two components - a client-side application living outside your perimeter, and a server-side component (the BFF) inside your perimeter”. The perimeter of the BFF is the webserver.</p> <p>The accused instrumentality meets this limitation under the doctrine of equivalents because the accused instrumentality performs substantially the same function, in substantially the same way to achieve substantially the same result. For example, DoorDash's network utilizes a sophisticated algorithmic framework that mirrors the intelligent operational capabilities of the patented system. It dynamically adjusts to real-time data for delivery logistics, employs predictive analytics for operational efficiency, and incorporates a centralized data management system for streamlined communication, much like the patent's description of an intelligent network. This implementation achieves the same overarching goal of efficient service delivery, adhering to the principles of the doctrine of equivalents.</p> |
| <p>wherein the parallel operations capable network is integrated with the MFCCS and the backend servers are programmed with instructions executable to choose and apply varying modes of contact during the same remotely initiated hospitality task, for and with the handheld/mobile customers and/or handheld/mobile equipped entity staff, to intelligently execute and support completion of the hospitality application task requests.</p> | <p>This claim element focuses on the importance of a web server's ability to efficiently manage communications within the hospitality industry. This system is particularly relevant to DoorDash, which requires a robust and flexible communication channel to operate its systems and apps smoothly.</p> <p>The claim element describes a server that intelligently selects the best way to communicate with users and can switch methods if the primary one fails. This capability ensures that DoorDash can maintain consistent contact with both customers and restaurants, even when technical issues arise ensuring seamless operations through smart communication management.</p> <p>Redact POS</p> <p>DoorDash integrates with the Redcat point-of-sale (POS) “...[seamlessly] integrate orders from DoorDash, Caviar, and Storefront” (See Exh. 95). In this DoorDash Merchant Support document concerning Redcat integration, it states the following overview of this integration:</p> <p><i>“Integrating DoorDash with your POS through Redcat simplifies order management and operations with:</i></p> <ul style="list-style-type: none"> • <i>Streamlined orders - No need to manually enter orders from a tablet, email, or phone to your POS. Orders will go right to the kitchen, minimizing room for error and freeing up your staff to focus on your in-person guests.</i> • <i>Automatic menu updates - Any changes you make to your POS menu will be automatically reflected on your DoorDash menu.</i> • <i>Increased order accuracy - Because menu changes are synced through Redcat, customers are less likely to order an out-of-stock item.</i> |

- *Consolidated accounting & reporting on your POS - DoorDash orders are tagged to a unique payment type and dining option to let you view your business's performance on your POS."*

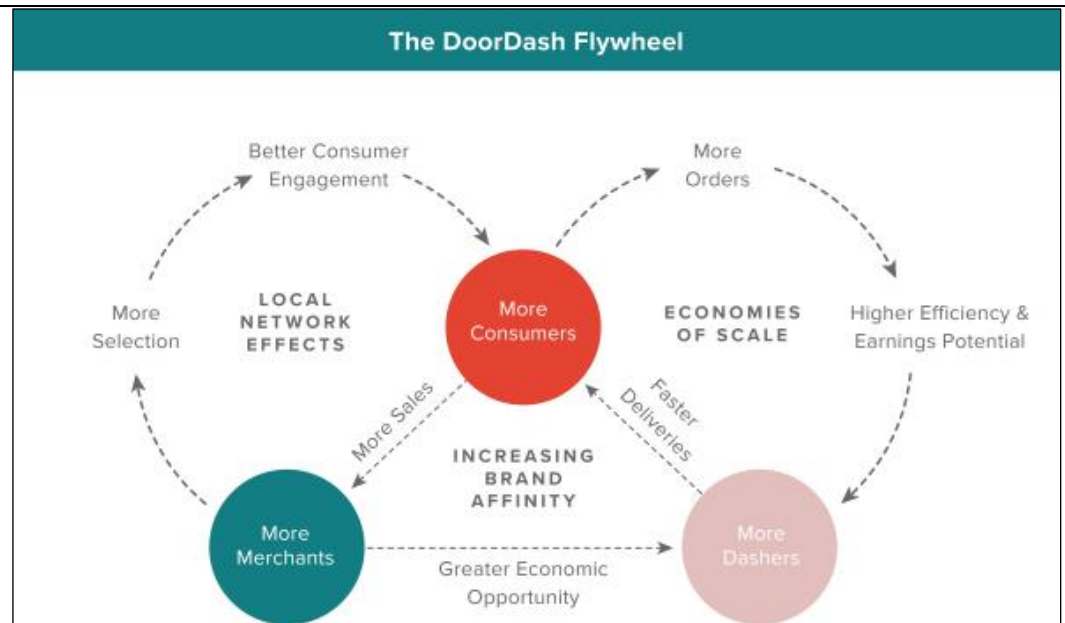
ItsaCheckmate POS

DoorDash integrates with the ItsaCheckmate point-of-sale (POS) "...[seamlessly] integrate orders from DoorDash, Caviar, and Storefront" (See Exh. 96). In this DoorDash Merchant Support document concerning Redcat integration, it states the following overview of this integration:

"Integrating DoorDash with your POS through ItsaCheckmate simplifies order management and operations with:

- *Streamlined orders - No need to manually enter orders from a tablet, email, or phone to your POS. Orders will go right to the kitchen, minimizing room for error and freeing up your staff to focus on your in-person guests.*
- *Automatic menu updates - Any changes you make to your POS menu will be automatically reflected on your DoorDash menu.*
- *Fewer order cancellations - Menu changes, including 86'd items, are synced as needed through ItsaCheckmate which saves customers from ordering an out-of-stock item.*
- *Consolidate accounting & reporting on your POS - DoorDash orders are tagged to a unique payment type and dining option to let you view your business's performance on your POS."*

As confirmed by DoorDash Vice President Jessica Lachs, (above/below) as part of its '360 degree picture' and framework approach, DoorDash integrates its intelligent web servers, with its MFCCS, and hospitality food/drink ordering application, (while including programming to intelligently choose and apply multiple modes of communications and/or different protocols if applicable) to enable the completion of the user requested hospitality food/drink, ordering tasks for delivery or pick up and as part of its Flywheel marketplace. The DoorDash Flywheel diagram, shown below, makes evident that the DoorDash operates in accordance with its overall series of linked services and via its "360-degree picture" central ("Flywheel") and technology platform framework:



DoorDash’s Vice President of Analytics and Data Science Jessica Lachs in an August 17, 2022, interview on “Leveraging Data to Delight Customers Despite a Challenging Supply Chain” in which she states (See Exh. #98):

“And so for us, it’s really about collecting as much information as we can about all sides of the marketplace, bringing all of that data together into a central data platform, where all of that data is accessible no matter the source. Whether it is coming from our production systems, transactional data, whether it is event data in our apps, whether that’s the consumer app, the dasher app, the merchant app... whether it is coming from our CRM systems. All of that data needs to come in to one central place so that we can tie it together and use the insights together to create a 360 degree picture of what’s happening on our platform and off our platform so that we can use that information not just to provide accurate menus and inventory for consumers but also so we can send the right email communications to consumers, to dashers, so that we really have a full picture of what’s happening and can use that for personalization and to help all three sides of our marketplace really optimize that they are at their peak efficiency.”

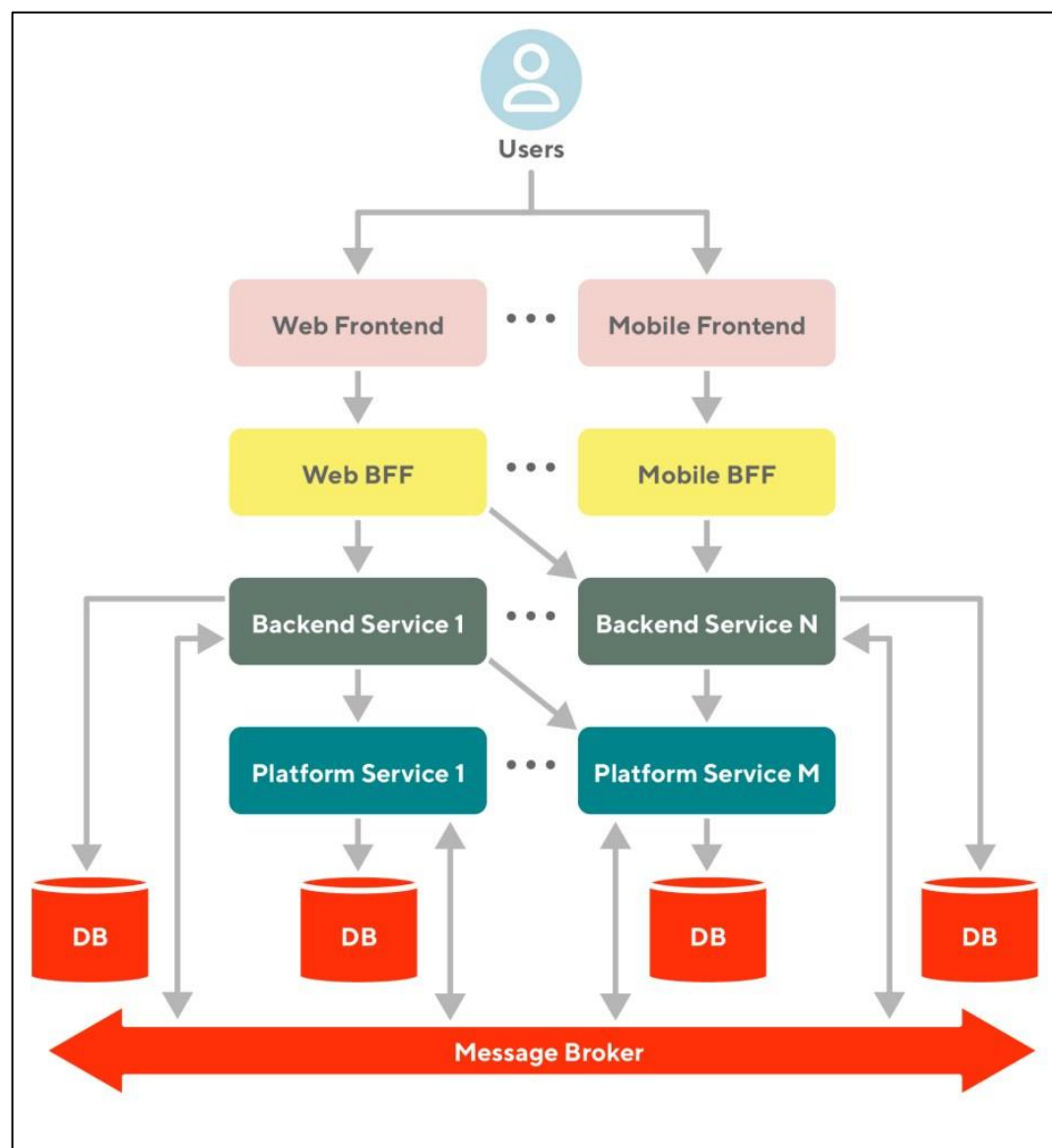
“So, for us, we want data to be easily accessible to all the different teams that need access to it. Analytics, being one of the largest customers of data at DoorDash, of course, but the way we think about our data models is really about increasing accessibility and consistency to that data. So, having all of our data in one central place and making sure that it is high in performance and so like query speeds are fast and that data models are thoughtful, so that it makes it a lot easier for data scientists, analysts, operators, product managers to be able to query the data that is needed and use the data in our production, in our production systems as well. So, we try to be thoughtful about how we structure our data models and how we ensure that all of the different production systems tie together into that central, as you mentioned, that central data lake.”

Microservices Architecture

The original DoorDash platform was originally a monolithic application written in Python using the Django web framework with a PostgreSQL database. As the platform grew, they started to have problems with reliability and scaling. Around 2018 they institute a code freeze and began migrating to microservices. At this time, they also migrated to the Kotlin language, and their services now run on the JVM (See Exh. 19, 20, 21, 22).

DoorDash was able to improve the efficiency and reliability of their platform using a microservices architecture. By breaking up their application into domain-specific parts they were able to reduce errors and latency. They state in the article “Future-proofing: How DoorDash Transitioned from a Code Monolith to a Microservice Architecture” (see Exh. 20) that

“... final design for our new microservice architecture consisted of five different layers, ranging from the user experience to the core infrastructure. Each layer provides functionality to the upper layer and leverages the functionality exposed by the lower layer [as shown below]”:



These layers include (see in Exh 20):

“Frontend layer: Provides frontend systems (like the DoorDash mobile app, Dasher web app, etc.) for the interaction with consumers, merchants, and Dashers that are built on top of different frontend platforms.

BFF layer: The frontend layer is decoupled from the backend layer via BFFs. The BFF layer provides functionality to the frontend by orchestrating the interaction with multiple backend services while hiding the underlying backend architecture.

Backend Layer: Provides the core functionality that powers the business logic (order cart service, feed service, delivery service, etc.).

Platform layer: Provides common functionality that is leveraged by other backend services (identity service, communication service, etc.).

Infrastructure layer: Provides the infrastructural components that are required to build the site (databases, message brokers, etc.) and lays the foundation to abstract the system from the underlying environment (cloud service provider).”

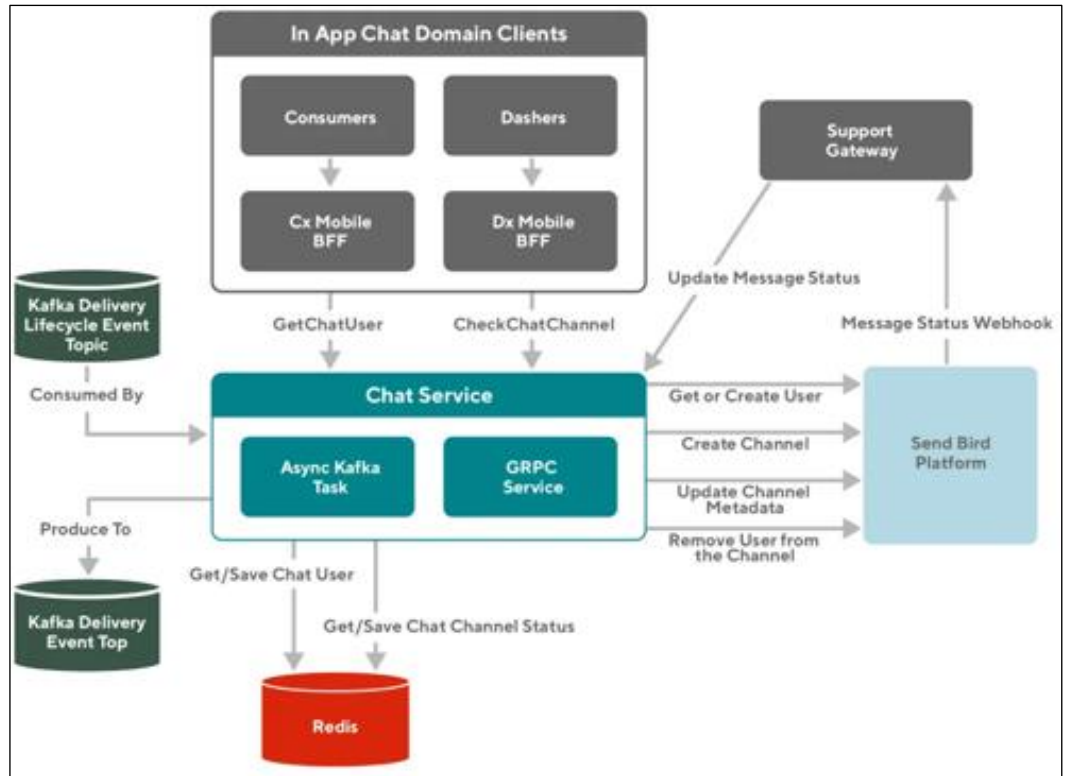
In their new architecture they introduced the backend-for-frontend (BFF) “... an application connecting the consumer-facing client and the services providing general purpose APIs. Client requests go to the BFF, which then orchestrates the aggregation of information needed by the client.”. The BFF is a software architecture pattern (See Exh. 40) used by microservices which “...shifted from thick-client applications to **interfaces delivered via the web, a trend that has also enabled the growth of SAAS-based solutions in general**”. As such BFF can be considered as “...the user-facing application as being two components - a client-side application living outside your perimeter, and **a server-side component (the BFF)** inside your perimeter”. The perimeter of the BFF is the webserver.

Integrated In-App Texting

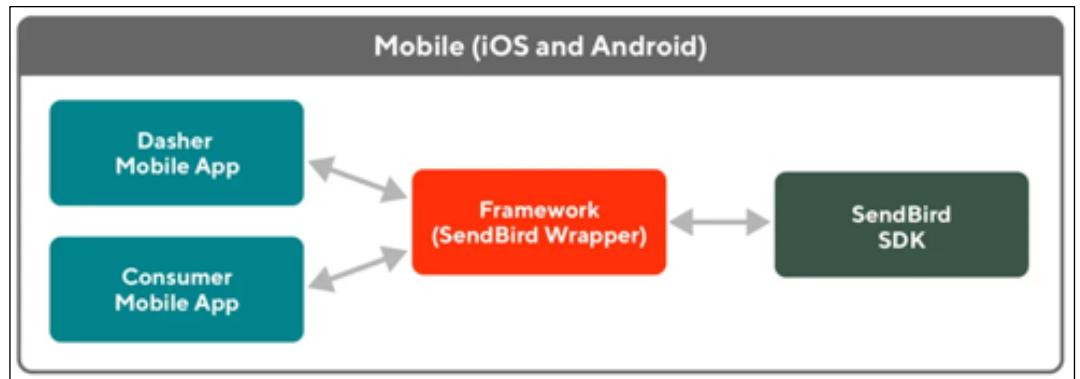
DoorDash integrates texting between customers and Dashers into its iOS and Android app. In the blog article by the DoorDash engineering team “Building Chat Into the DoorDash App to Improve Deliveries” they state “... [b]uilding chat into our apps may seem redundant, as smartphones already support calling and texting. However, building our own chat client has several advantages over these external channels. Our chat client creates a buffer between Dashers and customers, ensuring privacy for both parties, and lets us better resolve customer support issues.” The problem with using external communication channels to connect Delivery Drivers with customers is that “...a Dasher attempting a delivery might try to call or text the customer if finding the location proved difficult or other issues arose. Although this method may be effective, it requires the Dasher to switch away from the app, which shows the delivery address and other important details” (See Exh. 71, 72).

DoorDash integrates SendBird’s chatting technology into its platform by implementing their Chat API and native Chat SDK to solve these problems. They state that the system consists of “...four main components: mobile clients, chat service, data store, and SendBird SDKs. The in-app chat entry points have been added to all of DoorDash’s mobile clients. Mobile clients connect to the chat service to get the chat user and chat channels. The chat service handles communications between the data store and SendBird platform APIs. In the data store we maintain users and channels that get

created by SendBird. The chat service uses SendBird to create users, chat channels, and channel metadata, and then stores that data”, as illustrated by the diagram below:



The DoorDash Mobile Apps, iOS and Android are integrated with SendBird’s SDK as depicted in this diagram:



In the blog article “Building Chat Into the DoorDash App to Improve Deliveries” by the DoorDash Engineering team (See Exh. 71) they state that DoorDash integrated chat into their apps as opposed to relying in the built-in chat/texting of the smartphones because “...creates a buffer between Dashers and customers, ensuring privacy for both parties, and lets us better resolve customer support issues.” The article states:

Every delivery enabled by the DoorDash platform is different. Dashers (our term for delivery drivers) meet customers in a wide range of contexts, from apartment and office building lobbies to suburban homes. This variety of circumstances and the timely nature

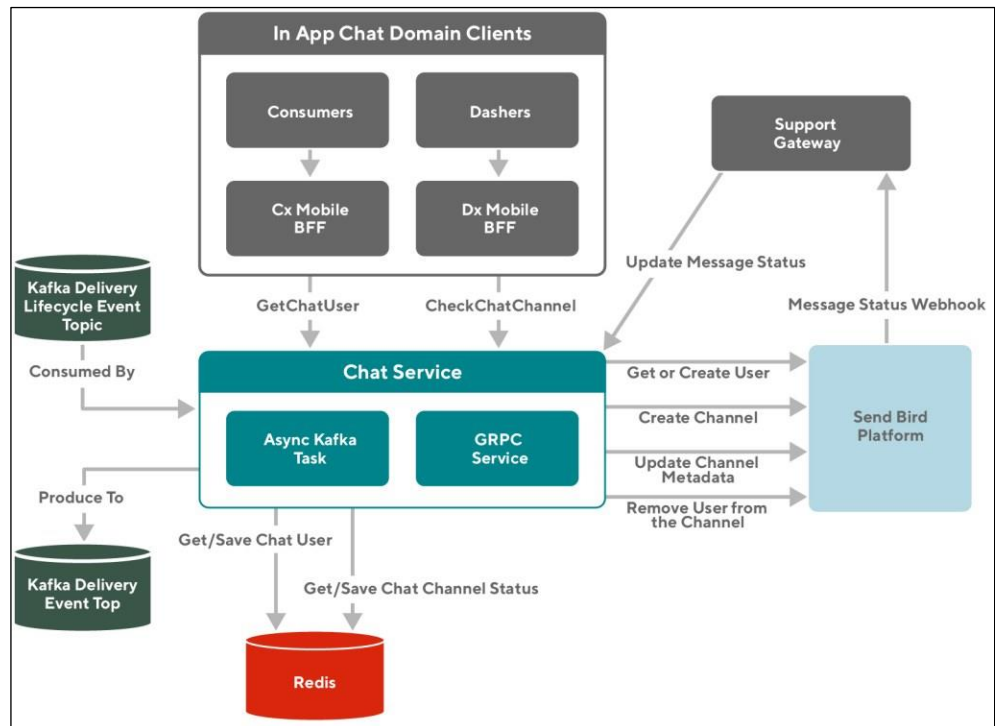
of contact makes communication essential, which is why we built chat into the DoorDash apps.

“Building chat into our apps may seem redundant, as smartphones already support calling and texting. However, building our own chat client has several advantages over these external channels. Our chat client creates a buffer between Dashers and customers, ensuring privacy for both parties, and lets us better resolve customer support issues.

Creating a chat client requires quite a bit of engineering and, given our past success integrating existing solutions, we began this project by evaluating third-party software. After settling on a chat technology, we integrated it into our platform and apps.”

Regarding the integration of SendBird (See Exh. 72) into the DoorDash iOS and Android apps, the article specifies the main aspects of the technical system design, how they integrated the SendBird UI into their custom apps, and the final results, stating specifically the following:

“Our system design consists of four main components: mobile clients, chat service, data store, and SendBird SDKs. The in-app chat entry points have been added to all of DoorDash’s mobile clients. Mobile clients connect to the chat service to get the chat user and chat channels. The chat service handles communications between the data store and SendBird platform APIs. In the data store we maintain users and channels that get created by SendBird. The chat service uses SendBird to create users, chat channels, and channel metadata, and then stores that data.”



“[In the above figure, the DoorDash] ...in-app chat system design consists of four main components: mobile clients, chat service, data store, and SendBird SDKs. The chat service receives chat user and channel requests from the mobile clients, sending them to the SendBird platform. SendBird creates channels and users and

updates the message status. The chat service also stores the chats in a Redis data store.”

Note: Redis is an “...open source, in-memory data store used by millions of developers as a database, cache, streaming engine, and message broker” (See Exh. 90).

For the DoorDash mobile apps (iOS and Android), the article explains how they built a “...a framework that serves as a wrapper around the SendBird UI SDK” and states the following:

“On the mobile side we built a framework that serves as a wrapper around the SendBird UI SDK. The framework adds UI styling and configuration on top of the chat SDK UI. The framework is integrated in both consumer and Dasher mobile apps. Building the framework helped us to isolate the chat-related logic and maintain it in one place.

Integrating chat into our mobile apps required work on four separate codebases. We have the consumer and Dasher apps, with Android and iOS versions of each.

Because chat sessions are temporary events, we chose Redis as our data store, using it as a cache layer for the user info and chat channel status. Redis was also an easy choice, as it is already available in our cloud infrastructure.

SendBird stores the actual chat text on its platform, which is SOC 2 certified and GDPR compliant, ensuring security and privacy. A pipeline between SendBird and our customer service system lets authorized users view chat histories to help resolve support issues.”

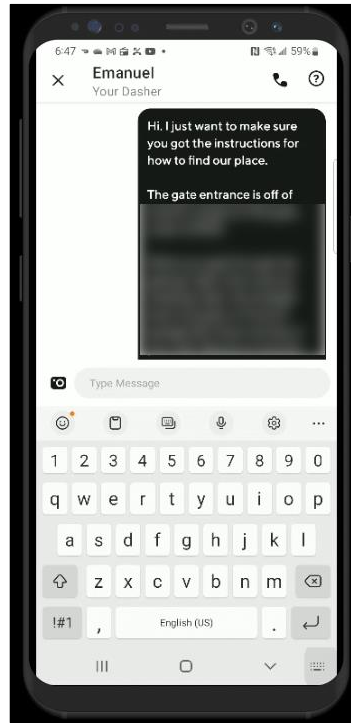
The article includes with very interesting results integrating a custom chat technology into their mobile apps:

“Adding in-app chat to our consumer and Dasher apps decreased the number of orders showing as Never Delivered, a metric we use to measure how many deliveries were missed. One of the most common causes of Never Delivered is poor communication between a consumer and Dasher.”

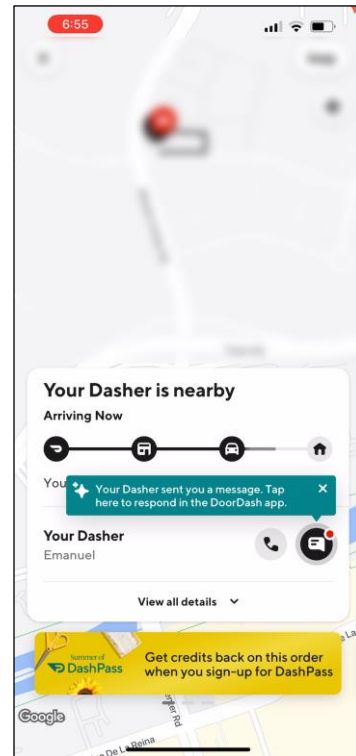
Order Tracking and Notification

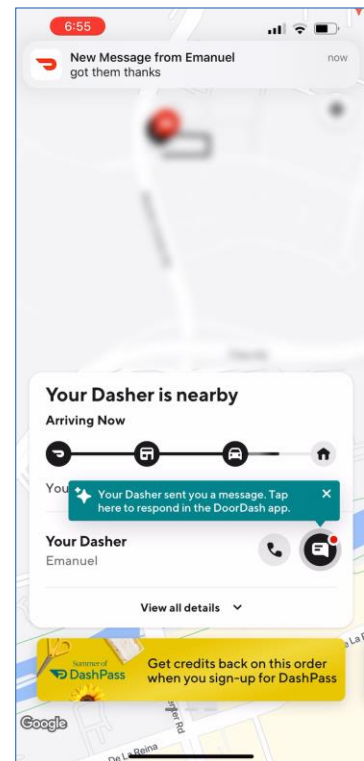
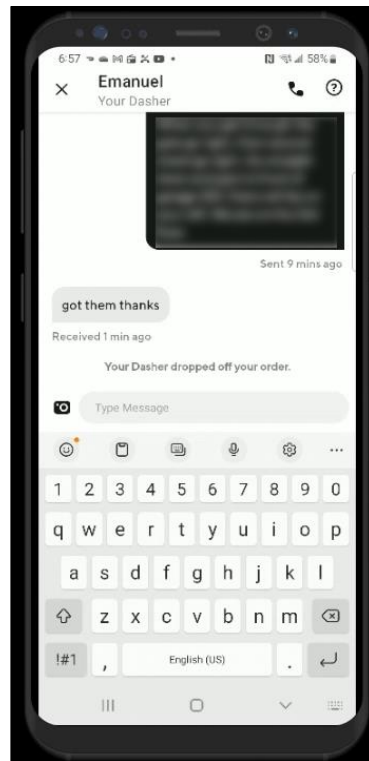
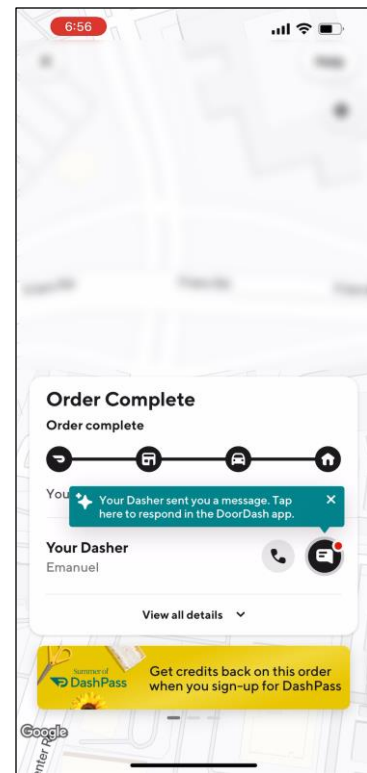
The same holds true for order notifications, order tracking and texting/chatting between the dasher and user. Below are a series of screens taken from one iPhone and one Android each tracking the same order placed by a user in the Android App. They depict the notifications and updates in the app for the user to be able to track their order. In this case, we show see text messages placed and sent by the user to the Dasher on the android app, being reflected the iOS app on an iPhone that is tracking the order. When the Dasher responds to the chat from the user, the user is notified on the android phone and on the iOS phone.

Android DoorDash App



iOS DoorDash App





Business Manager App

DoorDash offers the Business Manager app for iOS and Android. It lets restaurants “...track orders in progress, resolve issues, access support, monitor business performance, and get real-time notifications — all on your phone.” Selective

screenshots from the DoorDash Business Manager App Guide describes the app, and it's features as below (See Exh. 59, 62, 65). With the Business Manager App, managers can in real-time:

- “track sales, orders, and trends, for one or multiple stores
- change store status
- update menu hours
- view active orders
- call the customer or Dasher
- contact DoorDash Support
- mark items out of stock
- cancel an order”

INTRODUCTION

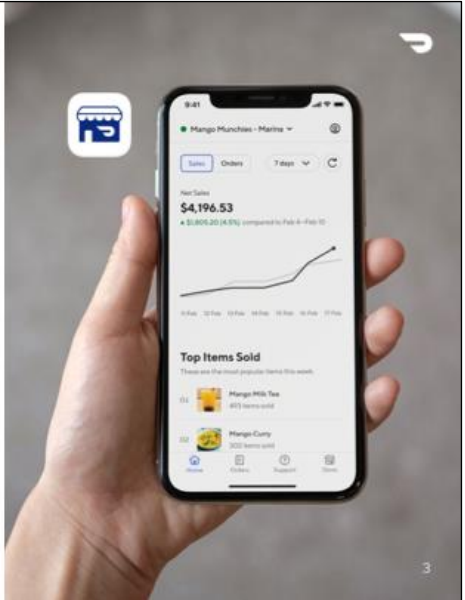
DoorDash - Business Manager App

Manage your DoorDash business right on your phone

Running a business isn't easy, but it could definitely be a little easier.

The new DoorDash - Business Manager app lets you track orders in progress, resolve issues, access support, monitor business performance, and get real-time notifications — all on your phone.

This guide will walk you through downloading, installing, and using the new app.



INTRODUCTION

A note on Point-of-Sale (POS) Integrations



What's a POS integration?

A POS integration is a streamlined connection between your point-of-sale (POS) software and DoorDash's platform. This connection lets you receive DoorDash orders directly on your POS, so everything is in one place and there's no need to manually enter orders from one system into another.

Why is the user flow different depending on whether or not I have a POS?

When businesses use a POS system, some of their store information is stored directly on that system instead of on DoorDash. This means that the user experience for stores using POS system differs for certain actions and updates.

| Use your POS system to: | Use the DoorDash Business Manager app to: | Call DoorDash Support to: |
|---|--|--|
| <ul style="list-style-type: none"> Update your menu (Items, modifiers, and photos) Change menu and store hours Mark items out of stock | <ul style="list-style-type: none"> Receive push notifications Track business health View active orders Call customer, dasher, or support | <ul style="list-style-type: none"> Cancel live orders |

Throughout this guide, sections with separate POS instructions will be marked using this tag:

[POS Integrations](#)

Integration features vary by provider. Reach out to your integration provider or refer to these [help articles](#) for more information.

4

HOW TO

Install the app - iOS

- 1 Download the app with [this link](#) or QR Code below.
- 2 Click on **Install** and let the app download.
- 3 Once installed, tap **Open** to launch the app for the first time.



Apple App Store



HOW TO

Install the app - Android

- 1 Download the app with [this link](#) or QR Code below.
- 2 Click on **Install** and let the app download.
- 3 Once installed, tap **Open** to launch the app for the first time.



Google Play Store

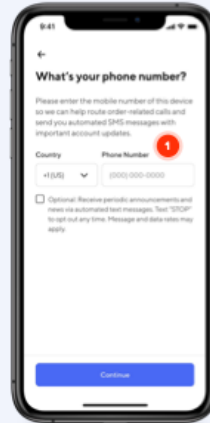


5

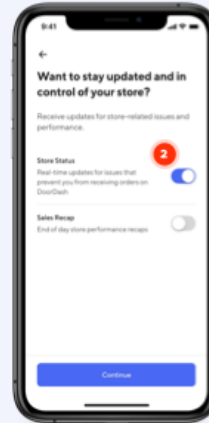
HOW TO

Enable push notifications

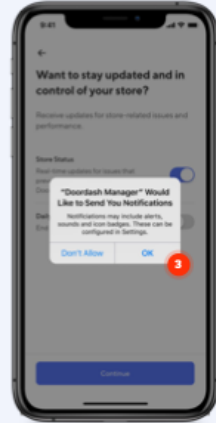
- 1 When you first log in, you'll be asked to provide your **phone number for authentication**. Once you've done that, tap Continue.
- 2 **Enable push notifications** in the toggle bar to receive real-time updates about store issues and end-of-day performance recaps. Tap Continue.
- 3 You'll see a pop-up asking you to allow **DoorDash Manager to send you notifications**. Be sure to select **OK** for full functionality.
- 4 If you select Don't Allow, you'll need to **re-enable notifications** via your device's Settings menu.



1st time authentication



Notifications selector



Accept device notifications

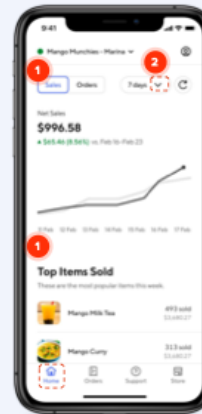
6

HOW TO

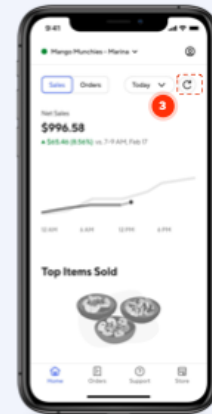
Track business health (single store)

- 1 The Home tab of your Dashboard allows you to see **sales, orders, trends and top items sold** in a specific time range.
- 2 You'll have the option to choose from today, yesterday, 7 days, 30 days. Just tap the dropdown arrow.
- 3 If you select **today**, you'll be able to:
 - Monitor sales and orders in real time by tapping the refresh arrow
 - Analyze partial-day trends

Menu item performance is calculated at 7 days and 30 days, so to view that data just change the time range.



View top-line metrics and trends

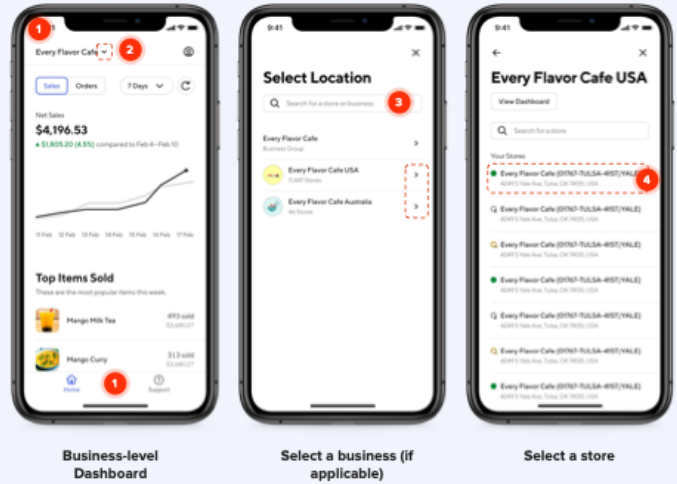


Monitor day-of performance

8

HOW TO Track business health (multiple stores)

- 1 If you own **multiple locations**, the home screen will display metrics across all stores by default. You can confirm this by noting:
 - The business (or group) name at the top left
 - Your bottom navigation bar will only show 2 tabs (Home and Support)
- 2 To view **store-level metrics and features**, tap the arrow in the top navigation bar.
- 3 Depending on how your business was set up on DoorDash, it may take 1-2 taps to get to an individual store (tap the **arrow to navigate**). You can also locate the store in the **search bar**.
- 4 **Tap the store** you'd like to view and both the app dashboard and metrics will automatically refresh.



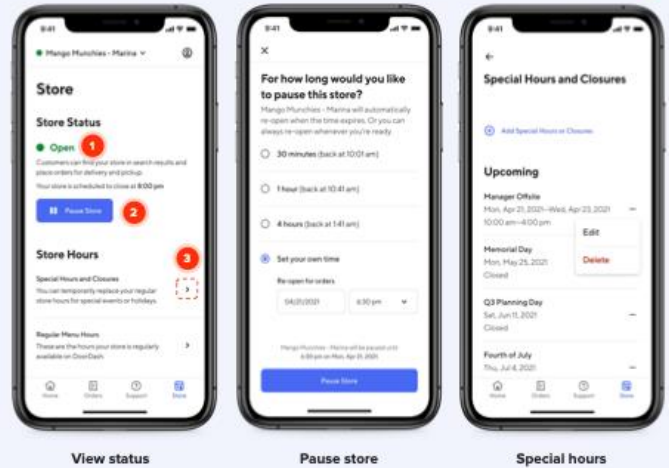
HOW TO Change store status

POS Integrations

- 1 You can view and change your store status (**open, paused, closed**) in the Store tab. If you have access to multiple stores, select the specific store from the dropdown menu.
- 2 Tap **Pause Store** to pause an open store and choose when it reopens (in an hour, the next day, etc).
- 3 View or add special hours or closures by tapping the right-hand arrow next to **Special Hours and Closures**.

When you change your store status in the app, it will be reflected on the Tablet and Portal (and vice versa).

*If you use a POS protocol, we recommend making special hours updates and closures using your POS. When you make these changes via the app, they may be overwritten during a POS sync.



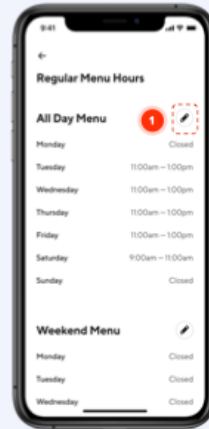
HOW TO
Update menu hours*

POS Integrations

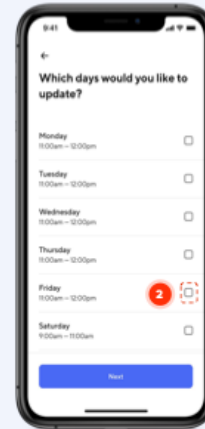
- 1 From the Store tab (refer to previous page for a screenshot) you can update your menu hours by tapping the right-hand arrow next to **Regular Menu Hours**.
- 2 If you have multiple menus for your store, scroll down to find the specific menu. Then tap the **pencil to edit**.
- 3 **Check the box** next to the day(s) of the week you'd like to update the menu hours for and tap **Next**.

Indicate whether the new hours reflect when you're open or closed by clicking on the top toggle. Then **tap on the start or end time** and select the desired time from the dropdown. Tap **Next** and then **Confirm**.

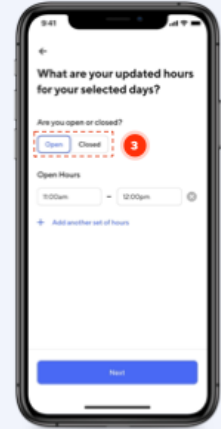
*If you use a POS protocol, we recommend updating menu hours using your POS. When you make these changes via the app, they may be overwritten during a POS sync.



Edit menu hours



Day of week



Start and end time

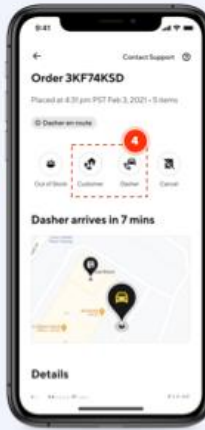
11

HOW TO
View active orders

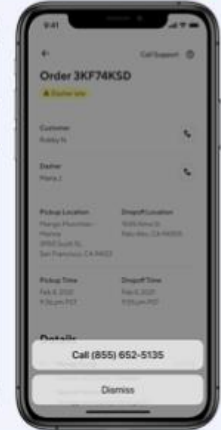
- 1 **Tap on the Orders button** in the bottom navigation to view active orders. For already completed orders, you can also toggle to History.
- 2 **Note the order status** on the right-side to see if an order needs attention.
 - ▲ Dasher late
 - Dasher en route
 - Dasher arriving
 - ▲ Dasher waiting
 - Delivered
 - Customer pickup
 - Customer arriving
 - Customer picked up
- 3 **See order details** by tapping on the right-hand arrow by the individual order.
- 4 **Call a customer or Dasher directly** by tapping the phone icon.



Orders



Order details



Call customer or Dasher

12

HOW TO

POS Integrations

Call the customer or Dasher*

If you use a Tablet protocol, you can continue to call customers or Dashers for in-progress orders by tapping the phone icon or Issue with Order.

Now you can also call from the app by **clicking on the phone icon by the customer's or Dasher's name**.

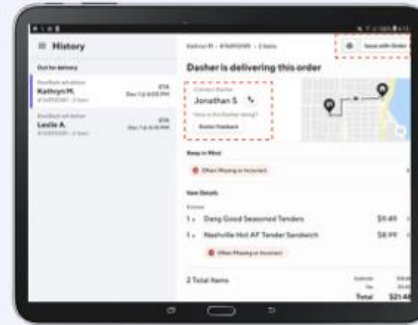
Common reasons to call a customer or Dasher:

- Out-of-stock item
- Clarification on special instructions
- Order running late
- Dasher arrives early
- Dasher waiting but not coming inside

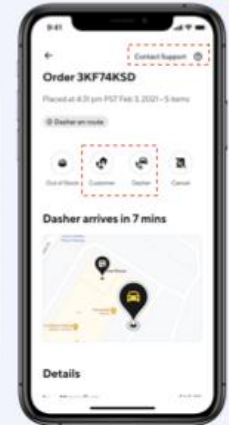
Afterwards, if you need to make any requested **order changes**, you can do so by chatting with our Support team — just tap **Contact Support** at the top right.

Your Tablet is still the best way to adjust prep time or mark an order ready for pickup.*

*If you use a POS integration, there's no need to mark orders ready for pickup; they will be confirmed and assigned a Dasher automatically.



Using Tablet to call customer or Dasher



Call customer or Dasher

13

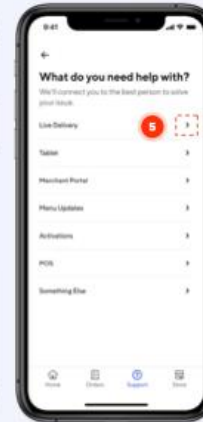
HOW TO

Contact Support

- 1 The **Support icon** is always accessible on the bottom navigation.
- 2 If you have access to multiple stores, be sure to select the right store from the dropdown.
- 3 To start a new request, tap **General Support** and then select the nature of your question in the next screen.
- 4 If you'd like to reopen a previous conversion, you can do so by tapping one of the Recent Messages.
- 5 For any issues with In-progress orders, tap **Live Delivery**. You can then choose if you'd like to **call Support** or **chat with Support**.



Support



Live delivery



Chat or call

14

HOW TO POS Integrations

Mark out of stock

- 1 Click on an active order to note when an item is **out of stock**.
- 2 Based on the order details, you'll select which **item or modifier** is out of stock.
- 3 Click on Next to **choose how long it will be out of stock** – hours, until the end of the day, or indefinitely. The item will automatically be made available after the selected time. You are also able to mark the item back in stock manually on your Tablet or POS.
- 4 Handle the **active order** in question by tapping Next to **chat with Support**. They'll get in touch with the customer to confirm if they want to replace the out-of-stock item or remove the item.

*If you use a POS protocol, we recommend marking items out of stock using your POS. When you make these changes via the app, they may be overwritten during a POS sync.

Active order **Select item or modifier** **Chat for live order help**

15

HOW TO POS Integrations

Cancel a live order*

- 1 If you're not able to fill an active order, you can choose to **cancel** it from the order details.
- 2 **Provide more detail** on why you can't fill the order so we can help you with next steps.
- 3 If you choose to cancel the order, we'll **confirm** this with you and let the customer know. As a reminder, cancellations that you initiate will **not be paid** for, even if you already prepared the food.

*This feature is not available on POS order protocol. If you're using a POS system and [need to cancel a live order](#), contact DoorDash Support.

Cancel a live order **Tell us what's going on** **Confirmation**

16

Checkout Process Architecture

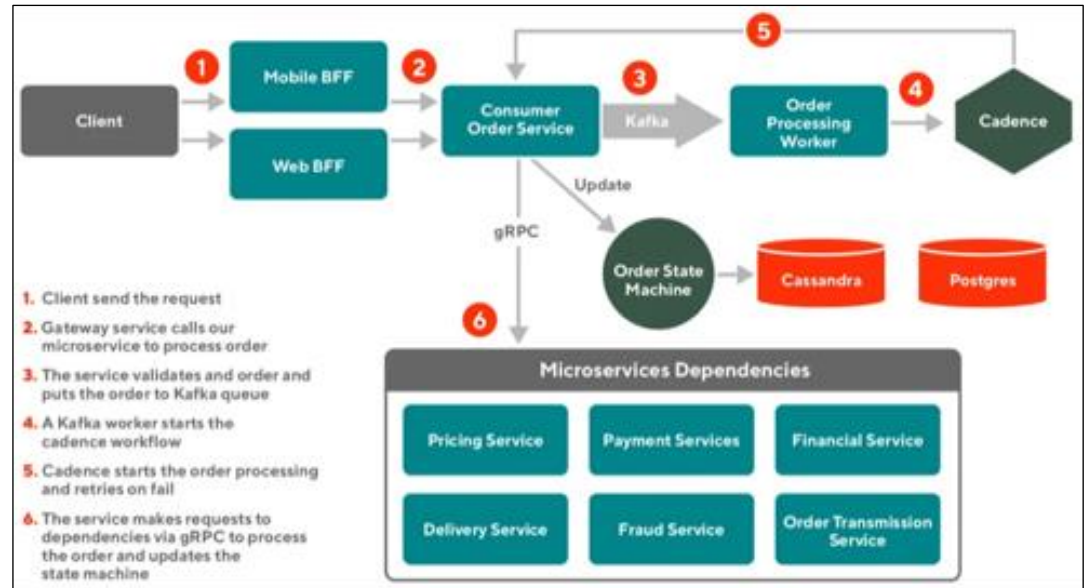
In the engineering blog article “Building a More Reliable Checkout Service at Scale with Kotlin” (see Exh. 78) the DoorDash development team explains how they reimplemented the customer checkout process into a new Kotlin microservices architecture to increase the DoorDash platforms performance, reliability, and scalability. As they state:

“...[t]he consumer checkout flow is one of the most critical flows on the DoorDash food ordering platform. The flow is responsible for submitting consumers’ order carts, processing payments, and connecting the order to our logistics system.”

They describe this new architecture with this detailed diagram stating that:

“...[our] re-engineered checkout flow takes advantage of our new microservice architecture’s components. Cassandra, a distributed database, has replaced Postgres, and become our primary data storage. Kafka is used to decouple the

complex workflow. Cadence helps us build fault-oblivious business logic” (see Exh. 78).



This diagram depicts the master database through the Order State Machine which gets its data from the Casandra database. The order lifecycle flow depicted in the above diagram is as follows according to the article:

“

1. The clients (web/mobile) send requests to our gateway microservice to checkout an order.
2. The gateway microservice calls the newly built order service.
3. The order service fetches data associated with the order and validates the item, merchant, and delivery availability. Then it materializes the order into a Cassandra database and drops off a Kafka message in the queue, indicating the order has been submitted.
4. The Kafka consumers poll order submission events and create a Cadence workflow to initialize the order processing.
5. Cadence calls an order service endpoint to start the order processing.
6. The consumer order service starts its state machine to process the order. In each state machine step, such as performing a payment or creating a delivery within our logistics systems, the consumer order service interacts with a related microservice using gRPC and updates the order we stored in the Cassandra database.
7. The client checks the order processing state using a pull model. The client polls an endpoint on the order service, which looks up the order state in the Cassandra table and returns a success if the order processing has finished all of the state machine's steps. If the order processing fails, Cadence retries it.”

Dasher Dispatch Engine with Intelligent Machine Learning Technology

Pipelines for order completion logic is confirmed and admitted in these screenshots take from the video presentation “DoorDash Technical Showcase Event: Logistics Team” (See Exh. #49) and as stated in this video as below.

Some of our Pipelines

- Dasher wait time**: Illustration of a Dasher on a bicycle and a kitchen worker.
- Kitchen capacity**: A bar chart showing 'Influx of orders' with levels 'Low', 'Ideal', and 'High'. Below it, it says 'Kitchen capacity varies over time' with an illustration of a kitchen worker.
- Other metrics we track**:
 - Order cancellations
 - POS outages
 - Store closures
 - Merchant tablet activity

55

Details on One of our Pipelines: Dasher Wait Time

Store Geofence

Dasher At Store → Food prep time → Dasher Leaving Store

Use ml models and heuristics to decide if store can handle more orders

Kafka → Flink Pipeline

Calculate dasher wait time and aggregate median

We use median in place of average to discard 1 bad delivery

56

Multi-Modes of Communication

DoorDash affirms in the merchant support document “Which order protocol works best for your business” (See Exh. 112) that DoorDash merchants/partners can receive orders through multiple modes of communication including:

- “Your existing point-of-sale (POS) system via an integration
- A DoorDash tablet or your own Android tablet using the Order Manager app

- *Email*

This document presents a table that specifies which mobile, tablet or web app is best to use for a given features. Below are screenshots of this specification (See Exh. 112):

| | Direct POS Integration (POS protocol) | Middleware Integration (POS protocol) | Tablet | Email |
|-----------------------------|---|--|--|--|
| Summary | <p>Popular with high-medium volume stores (~300+ orders per month) Great operational efficiency Most preferred</p> | | <p>Popular with medium-low volume stores (60-300 orders per month) Good operational efficiency</p> | <p>Suited for low volume stores (less than 60 orders per month) Lowest operational efficiency Least preferred</p> |
| Best For | Businesses of all sizes, from quick-service to full-service, with an existing POS system. | Businesses of all sizes, from quick-service to full-service, with an existing POS system that does not support integration. Also for businesses using multiple third-party delivery providers. | Businesses of all sizes without a POS system or where POS integration is not available. | Small businesses that do not have a POS system and/or efficient internet connectivity. |
| It Provides | An easy, seamless way to manage delivery and pickup orders using your POS. | An easy, seamless way to manage delivery and pickup orders using your POS. | Quick order management, access to Dasher status monitoring, item availability updates, and real-time prep time adjustment. | Manual work to receive, confirm, and update orders. |
| Country Availability | Varies by provider | Varies by provider | All markets | All markets |

| Functionalities | | | | |
|--|---|---|---|---|
| Receive Delivery and Pickup Orders | ✓ | ✓ | ✓ | ✓ |
| Receive Scheduled Orders | ✓ | ✓ | ✓ | ✓ |
| Managed Menu Changes | ✓ | ✓ | Merchant Portal | Merchant Portal |
| Menu Item modifiers | Varies by provider | Varies by provider | ✓ | Merchant Portal |
| Manage Menu Photos | Varies by provider | Varies by provider | ✓ | Merchant Portal |
| Manage Prep Times | Varies by provider | Varies by provider | ✓ | ✗ |
| Track DoorDash Business Health | ✓ | Varies by provider; available on Business Manager app or Merchant Portal | Business Manager app or Merchant Portal | Business Manager app or Merchant Portal |
| Pricing Merchant charges/fees paid to DoorDash | FREE Some providers charge a fee for 3rd party integrations. Reach out to your provider for details | FREE Some providers charge a fee for 3rd party integrations. Reach out to your provider for details | \$6/ WEEK | FREE |
| Extra Hardware | None | None | DoorDash or Personal Android Tablet | Computer or tablet |

| Menu Management Features | | | | |
|--|--------------------|--------------------|---|---|
| Item 86'ing (Item OOS in real-time) | Varies by provider | Varies by provider | ✓ | Business Manager app or Merchant Portal |
| Manage Store Hours | ✓ | ✓ | ✓ | Business Manager app or Merchant Portal |
| Manage Store Holiday/Special Hours | ✓ | ✓ | ✓ | Business Manager app or Merchant Portal |
| Update Menu Hours | Varies by provider | Varies by provider | ✓ | Business Manager app or Merchant Portal |
| Separate and Day-part Menus | ✓ | ✓ | ✓ | Merchant Portal |
| Special Instructions* | ✓ | ✓ | ✓ | Merchant Portal |
| Menu Item Sort Order | ✓ | ✓ | ✓ | Merchant Portal |
| Menu Item Level Hours** | Varies by provider | Varies by provider | ✓ | Merchant Portal |
| Price Override/Inflation | Varies by provider | Varies by provider | ✓ | Merchant Portal |

| <i>Live Operations Features</i> | | | | |
|--|--|--|---|--|
| Busy Kitchen | Contact DoorDash Support | Contact DoorDash Support | ✓ | Contact DoorDash Support |
| Temporarily Pause Store | Business Manager app or Merchant Portal | Business Manager app or Merchant Portal | ✓ | Business Manager app or Merchant Portal |
| Contact Customer | Business Manager app or contact DoorDash Support | Business Manager app or Contact DoorDash Support | ✓ | Business Manager app or contact DoorDash Support |
| Cancel/Refund Customer | Business Manager app or contact DoorDash Support | Business Manager app or Contact DoorDash Support | ✓ | Business Manager app or contact DoorDash Support |
| Dasher Updates | Business Manager app or contact DoorDash Support | Business Manager app or Contact DoorDash Support | ✓ | Business Manager app or contact DoorDash Support |
| Contact DoorDash Merchant Support | Business Manager app, phone, or email | Business Manager app, Merchant Portal, phone, or email | ✓ | Business Manager app or contact DoorDash Support |

Furthermore, in the merchant support document “How do I receive orders with DoorDash?” DoorDash affirms (See Exh. 113):

“If you would prefer orders to come through a tablet, all orders will come through DoorDash's Order Manager App on Android tablets.

- *With the tablet, you can see:*
- *Sections for each order stage lifecycle to easily identify where an order is*
- *Scheduled order timing when it appears in-app is changing (used to show up/print immediately)*
- *Cancellations won't just disappear anymore*

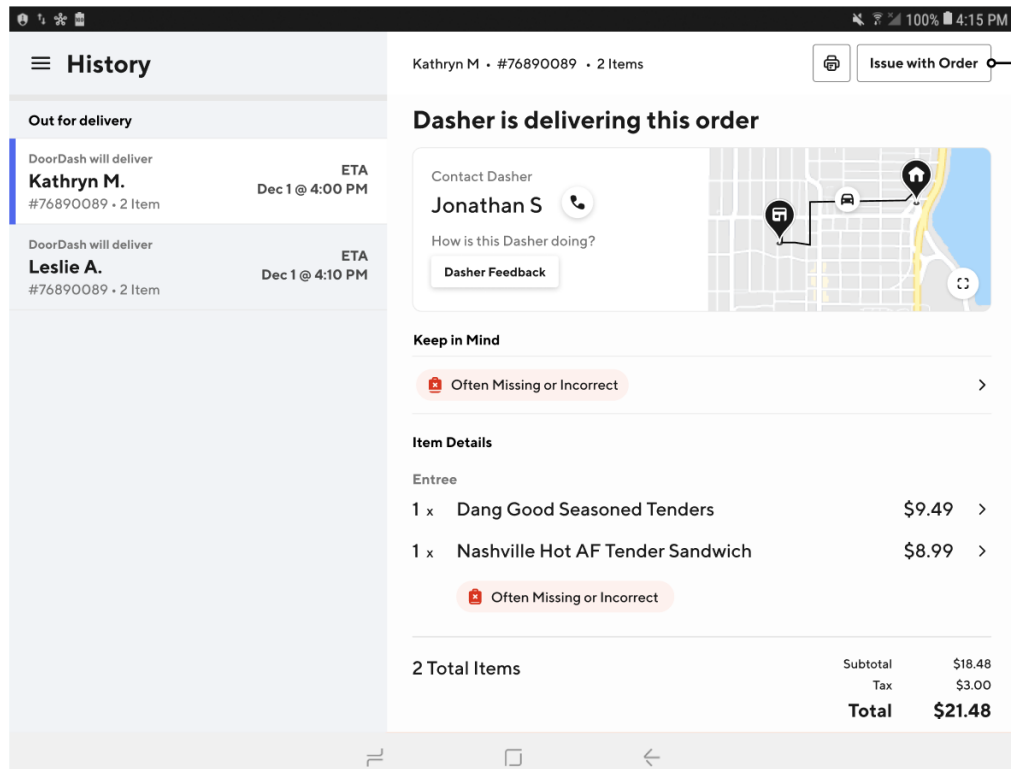
- *Tags for scheduled orders, large orders, canceled orders, modified orders, customer pickup*
- *Item count per order*
- *Large/clearer map and Dasher location information*

You can receive orders by POS, email, or email/fax. However, you will not be able to receive orders from phone+tablet or email+fax.

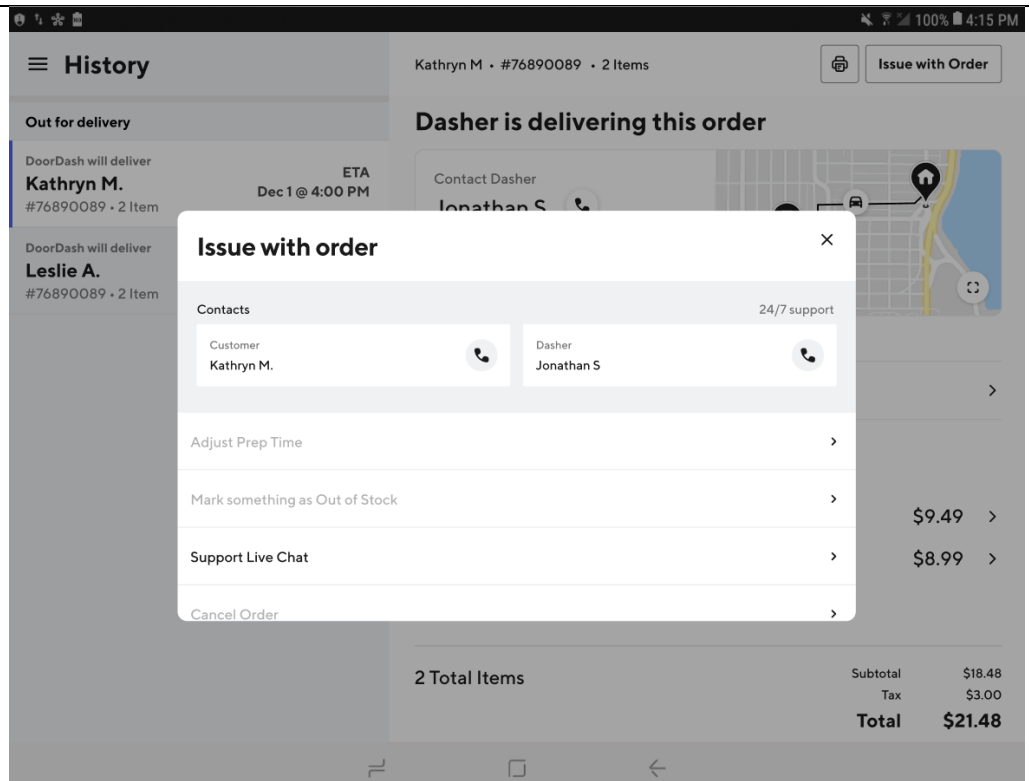
DoorDash can also place a follow-up automated call to ensure that all orders were received.”

The DoorDash Support article “How can I contact the Customer or Dasher through the tablet” (See Exh. 114) instructs the DoorDash merchant how to communicate with a Dasher (delivery driver) or Customer for an order that’s in process or has been picked up, should any questions arise. According to the article, the merchant uses the tablet and:

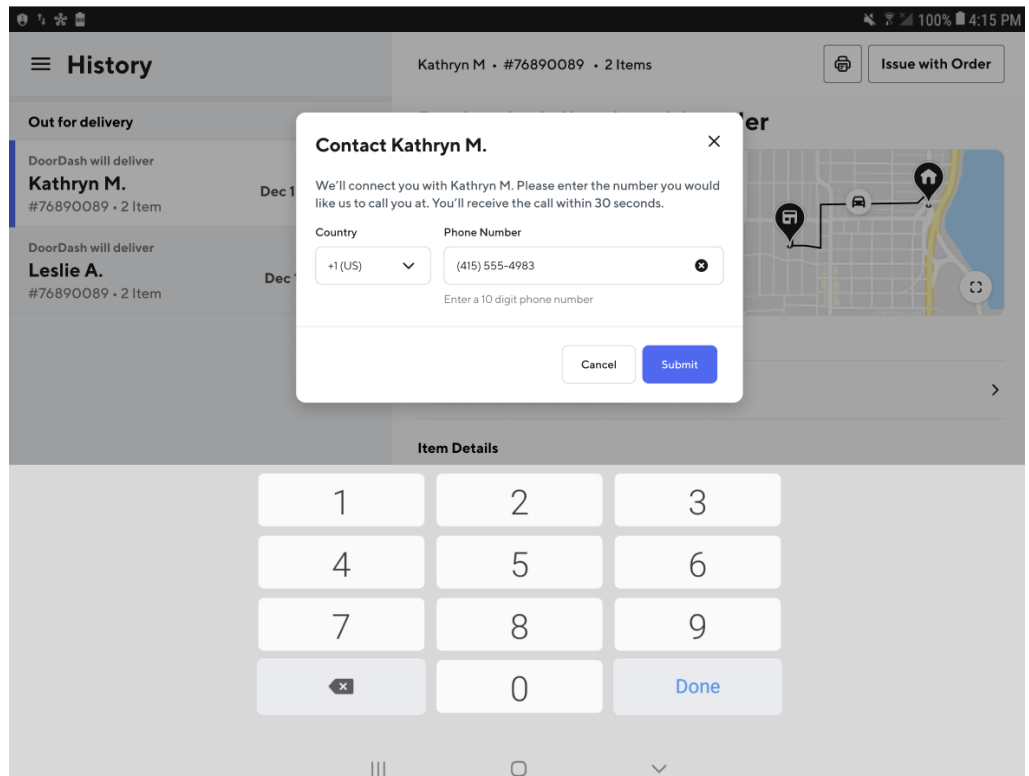
“...On the Order History page, tap on the Issue with Order button in the top right corner.”



“Next, tap on either the Customer or Dasher that you would like to contact. “



“Next, enter the phone number you would like to use to be connected to either the customer or Dasher. Tap Submit and we will call you on the number you enter within 30 seconds to connect your call!”



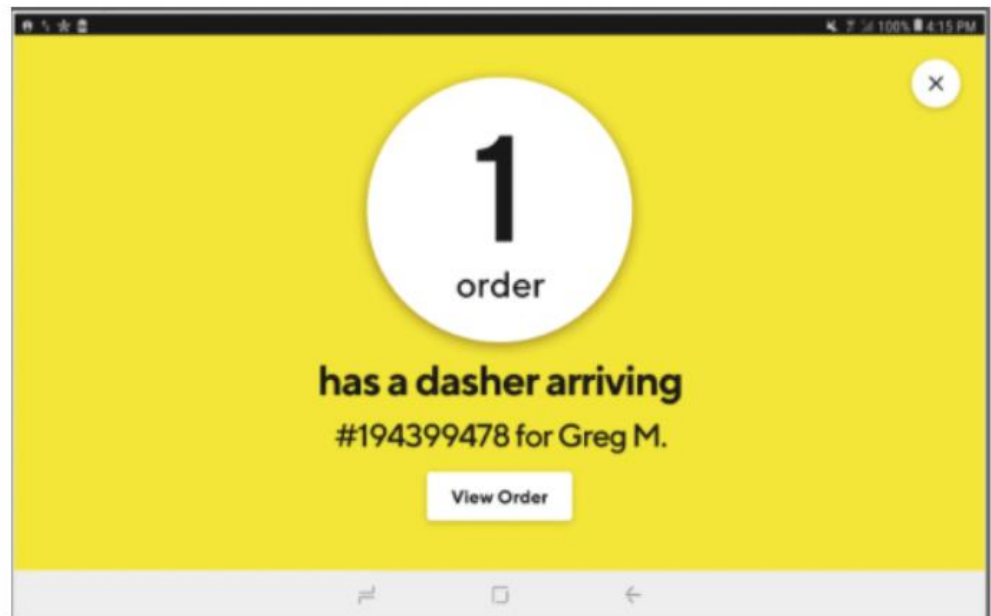
The DoorDash Merchant Support article *“How can I tell if my Dasher is arriving or waiting”* (See Exh. 115), discusses how merchants receive real-time notifications on their DoorDash Tablet:

“...when a Dasher is nearby to pickup an order! The new Dasher Arriving feature will send you a secondary chime and visual cue that will alert when a Dasher is 5 minutes away. With this feature, Merchants will now know exactly when to start preparing and bagging those final items (cold drinks, ice cream, fries, etc) right before a Dasher's arrival, preserving food quality and also reducing overall wait times!”

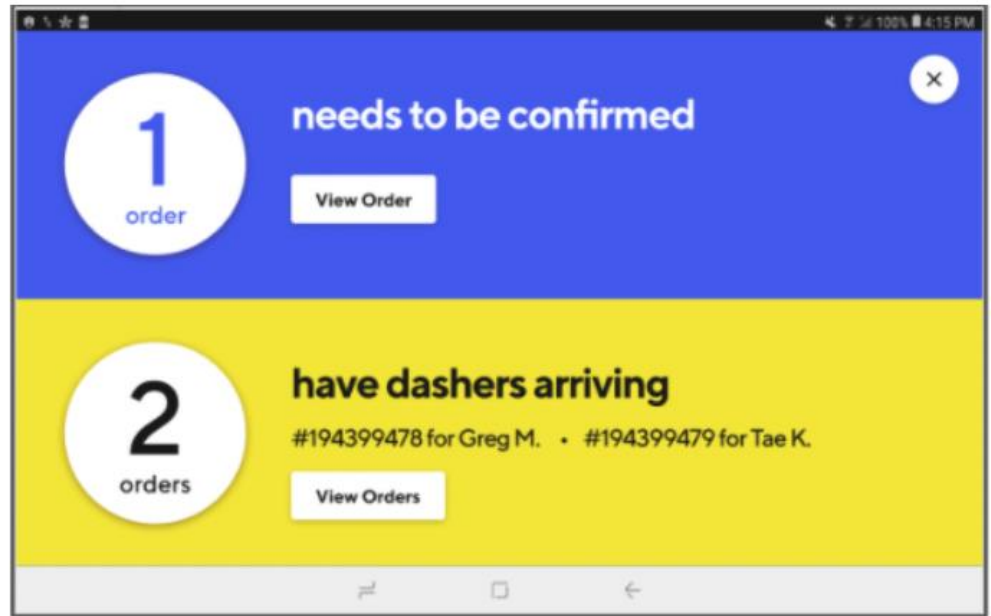
Notification

Once a Dasher is 5 minutes away, a full screen take-over will appear in yellow notifying staff on an approaching Dasher

Note: A unique chime will sound, which will help staff differentiate between a normal incoming order versus an incoming Dasher.

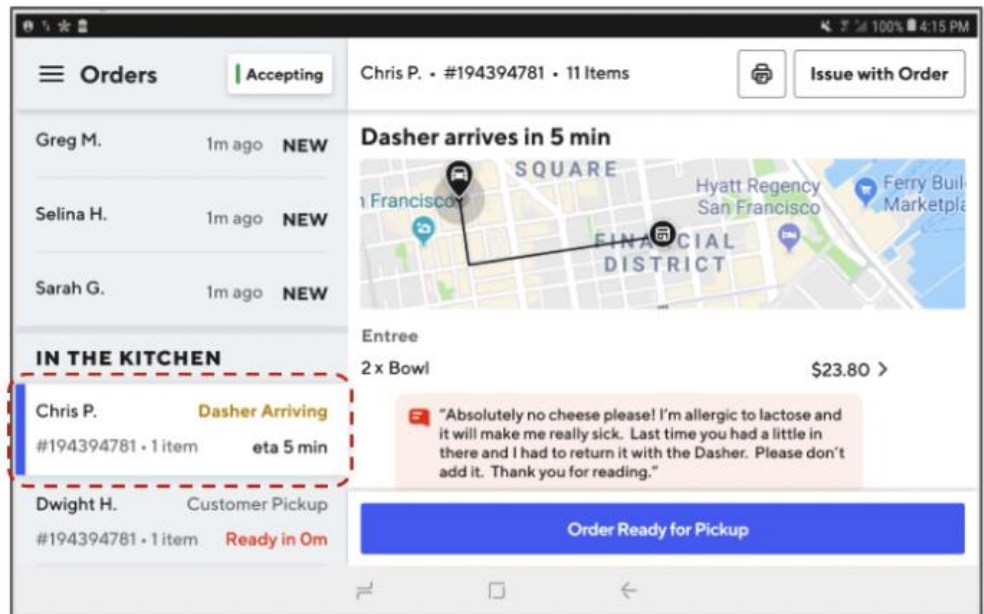


If there are new orders and Dashers arriving at the same time, the screen will appear as below, and the chime for "new order" will alarm. You will also be able to see which orders have Dashers coming for them (Customer name(s) shown).



Dasher Arriving Label

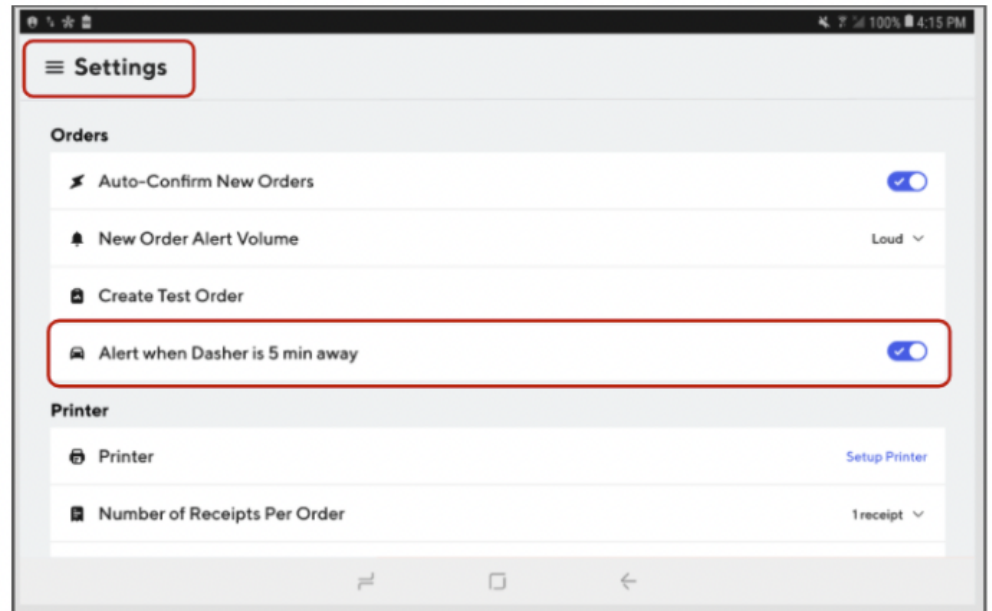
Within each order, there will be a Dasher Arriving label with eta x min shown, counting down until the Dasher arrives.



Feature Settings

Enter the flow directly in the tablet by selecting Settings from the main navigation page

Settings → Alert when Dasher is 5 min away → Toggle On / Off



In the DoorDash merchant support article “Can I have a tablet and also still get orders through email/fax/phone calls” (See Exh. 118), DoorDash states the following:

“[The Merchant] ...can receive DoorDash orders on a tablet and get orders through email or fax at the same time. As long as your tablet is logged in, you will continue to receive orders.”

DoorDash’s Unified Chat Experience

The DoorDash engineering team stated in an engineering blog post, titled **Building a Unified Chat Experience at DoorDash**, how they successfully integrated chat into their overall system, addressing the challenges and strategies employed to create a unified chat experience. Their implementation aligns with MFCCS as it provides a centralized system layer architecture enhancing communication between the web server and multiple wireless handheld devices, each operating on distinct mobile systems thus allowing customers to communicate with customer support and dashers. (See Exh. 102)

DoorDash recognized the crucial need to address customer issues on a large scale, leading them to acknowledge the essential nature of a unified chat feature. Their initial challenge was a fragmented chat system, which was scattered across various user groups and platforms, causing duplicated efforts and central adoption of best practices to be challenging.

By leveraging the architecture MFCCS, DoorDash could centralize its chat functionalities, allowing for seamless interactions between consumers, Dashers, and merchants. The MFCCS would facilitate the web server's ability to communicate with

diverse handheld devices, presenting mobile-compatible versions of the hospitality application. This integration is crucial for enabling user-initiated actions and subsequent selection of choices directly from the touchscreens of various devices, fostering a multi-modal, multi-protocol communication environment.

According to the engineering article, a significant objective for DoorDash was to streamline their support processes through automation, allowing customers to retrieve information through the app itself, thus minimizing the need for human support interaction. They aimed to develop a single chat support platform that would ensure consistent user experiences across different platforms and facilitate the reuse of components.

In building this unified platform, DoorDash created a single chat implementation that functioned for both customer support and communication between consumers and Dashers. This system, powered by Sendbird, was directly integrated into the consumer and Dasher apps, providing a uniform experience across all DoorDash chats and allowing for shared enhancements across different chat flows.

The backend system was strategically designed with multiple layers, distinguishing between internal services and third-party functionalities such as chat natural language processing. This separation expedited development by eliminating the need to create distinct, non-essential features.

Integration with their Decision Engine platform was another crucial step, which enabled operations to be automated, such as order updates, issuing credits, and personalized messaging. This significantly decreased the dependency on manual operations. DoorDash also carefully monitored several key performance indicators, including feedback from agents and customers, error rates, customer satisfaction, average handle time, first contact resolution, and manual tasks per order.

The outcomes of this updated chat support system, according to the article, were noteworthy, with reduced escalation rates and improved customer satisfaction. The automation led to a decrease in manual interactions per delivery, and customers were able to solve their problems more quickly, sometimes without the need for manual assistance. The time needed to implement additional automation was also reduced, owing to the common platform processing. Moreover, the common UI layers ensured that the chat feature was seamlessly integrated into the applications (iOS and Android), providing a consistent brand experience across all platforms.

The advancements DoorDash has made in their chat system encapsulate the essence of the Middleware/Framework Communications Control Software (MFCCS) described in the patent claim. By creating a unified chat platform that seamlessly integrates with multiple user interfaces and operating systems, DoorDash has effectively embodied the centralized system layer architecture that MFCCS envisions. Their backend, which differentiates between core and third-party functionalities, mirrors the multi-layered approach of MFCCS, allowing for efficient, scalable communication across various devices. Moreover, the phased rollout and meticulous performance monitoring align with the adaptive, intelligent operational capabilities that MFCCS aims to provide. This demonstrates a real-world application of the patent's claim, showcasing a system that not only supports multi-mode communication but also enhances user experience and operational efficiency in a dynamic hospitality market.

The accused instrumentality meets this limitation under the doctrine of equivalents

| | |
|--|---|
| | <p>because the accused instrumentality performs substantially the same function, in substantially the same way to achieve substantially the same result. For example, the Middleware/Framework Communications Control Software (MFCCS) integration into the DoorDash System empowers the network to process hospitality application task requests from wireless handheld users, utilizing a spectrum of communication modes. Notably, the DoorDash system seamlessly transitions to alternative hospitality entities upon detecting inventory shortages, demonstrating operational efficiency guided by intelligent rules that prevent redundant attempts with unresponsive entities. This approach optimizes resource allocation and minimizes processing time by circumventing fruitless communication endeavors with hospitality entities known to be unresponsive, effectively embodying the core elements of the claim in the DoorDash system.</p> |
|--|---|