AppLens and LaunchTile: Two Designs for One-Handed Thumb Use on Small Devices

Amy K. Karlson, Benjamin B. Bederson
Human-Computer Interaction Lab
Computer Science Department
Univ. of Maryland, College Park, MD, 20742
{akk, bederson}@cs.umd.edu

John SanGiovanni
Microsoft Research
One Microsoft Way, Redmond, WA 98052
johnsang@microsoft.com

ABSTRACT
We present two interfaces to support one-handed thumb use for PDAs and cell phones. Both use Scalable User Interface (ScUI) techniques to support multiple devices with different resolutions and aspect ratios. The designs use variations of zooming interface techniques to provide multiple views of application data: AppLens uses tabular fisheye to access nine applications, while LaunchTile uses pure zoom to access thirty-six applications. We introduce two sets of thumb gestures, each representing different philosophies for one-handed interaction. We conducted two studies to evaluate our designs. In the first study, we explored whether users could learn and execute the AppLens gesture set with minimal training. Participants performed more accurately and efficiently using gestures for directional navigation than using gestures for object interaction. In the second study, we gathered user reactions to each interface, as well as comparative preferences. With minimal exposure to each design, most users favored AppLens's tabular fisheye interface.

Author Keywords
One-handed, mobile devices, gestures, notification, Piccolo, thumb navigation, Zoomable User Interfaces (ZUIs).

ACM Classification Keywords
H.5.2. User Interfaces: Input Devices and Strategies, Interaction Styles, Screen Design; D.2.2 User Interfaces; I.3.6 Interaction Techniques

INTRODUCTION
The current generation of mobile computing hardware features a variety of devices for interaction with the system software. One such class of devices, typically referred to as "smartphones", features a numeric keypad or miniature thumb keyboard for input together with a screen for display output. These devices allow for single-handed interaction, which provides users with the ability to place calls and access information when one hand is otherwise occupied. However, because smartphones lack a touch-sensitive display, user interaction is constrained to a discrete set of keys, and thus the options for interaction design are limited to keypad-mapped functions and directional navigation. Another design approach, typically classified as a "Personal Digital Assistant" (PDA) features a touch-sensitive display surface designed primarily to be used with an included stylus. This design offers greater software design flexibility, but many of the small targets designed for a stylus are too small for fingertip actuation, making one-handed use difficult or impossible.

Our goal is to create a new single-handed interaction system for both smartphone and PDA devices. In this work, we have focused on the problem navigating device applications, and have adopted two design strategies. The first leverages Scalable User Interface (ScUI) techniques [2,3] to allow the system to adapt to different screen resolutions as well as support both portrait and landscape device rotation. This architecture allows developers to create a single application that can target a wide variety of screen resolutions, and provides users with a consistent interface and interaction model across devices. We accomplish this using the University of Maryland's Piccolo.NET development toolkit for zoomable and scalable user interfaces [5,18].

Our second design strategy provides access to rich notification information from multiple applications. Most current PDA interfaces are designed for focused interaction with a single task or application, with limited consideration or display real estate allocated for notifications (e.g., email, appointment reminders) or monitoring of ambient information streams (e.g., stocks, sport scores). In our proposed designs, each application has a dynamic launch tile in the place of a static launch icon. This feature offers high-value at-a-glace information for several applications at once, as well as on-demand application launch when users desire more detailed information.

In the work presented here, however, we limit our discussion of scalability and notification in favor of emphasizing the design features relevant to one-handed interaction. We proceed by describing two designs: AppLens (characterized by zoom+fisheye) and LaunchTile (characterized by zoom+pan). The two approaches employ variations of zooming interface techniques [1] to overview several notification tiles, each roughly the size of a postage stamp. AppLens uses a tabular fisheye approach to provide integrated access to and notification for nine applications. LaunchTile uses pure zooming within a landscape of thirty-six applications to accomplish the same goals.

Fisheye and pure zoomable techniques both offer promise, but there are no clear guidelines as to when each approach should be used. Our approach in this work, therefore, has been to design and build the best interfaces we could with roughly the same functionality, and then compare and contrast the results. In this way, we hope to understand which design direction makes the most sense for this domain and to learn something about the trade-offs between the two approaches.

For device interaction when using a touch-sensitive screen, both designs utilize a gestural system for navigation within the application's zoomspace. While our designs do not directly address one-handed text entry, they are compatible with a variety of existing single-handed text input techniques, including single- and multi-tap alphanumeric keypad input, as well as miniature thumb keyboards and unistroke input systems executed with a thumb (e.g., Graffiti [6], Quikwriting [17]).

RELATED WORK

Gestures have proven a popular interaction alternative when hardware alone fails to effectively support user tasks, typical of many nontraditional devices, from mobile computers to wall-sized displays [9]. Gestures can be very efficient, combining both command and operand in a single motion, and are space-conserving, reducing the need for software buttons and menus. However, the invisible nature of gestures can make them hard to remember, and recognition errors can negatively impact user satisfaction [15]. Recent research efforts pairing gestures with PDA-sized devices have emphasized gestures based on changes in device position or orientation [10,21,28]. However, our work more closely resembles the onscreen gestures that have played a prominent role in stylus-based mobile computing (e.g., Power Browser [7]).

Our thumb-as-stylus designs support usage scenarios in which only one hand is available for device operation, such as talking on the phone or carrying a briefcase. Although existing stylus-based gesture systems do not preclude the use of the thumb, we are not aware of any systems that have been specifically designed for the limited precision and extent of the thumb. The EdgeWrite [32] gesture-based stylus text entry system is particularly suited to mobile usage scenarios due to its use of physical barriers. EdgeWrite adaptation to one-handed mobile computing is compelling, but would require expanding the dedicated input area to accommodate thumb-resolution gestures.

One handed device interaction has typically focused on text entry techniques, but beyond a numeric keypad, most one-handed text entry systems require specialized hardware, such as accelerometers [25,30] or customized keyboards [16]. We instead address the more general task of system navigation and interaction, and restrict our designs to existing hardware.

Commercial products have also emerged with related design goals. The Jackito PDA [14] supports thumb-only interaction, but presumes two handed use and is not gesture oriented. Lastly, Apple's iPod is an elegant one-handed interaction solution for audio play and storage [13], but is currently not designed for the type of generalized application interaction we propose.

THE ZOOM+FISHEYE APPROACH: APPLENS

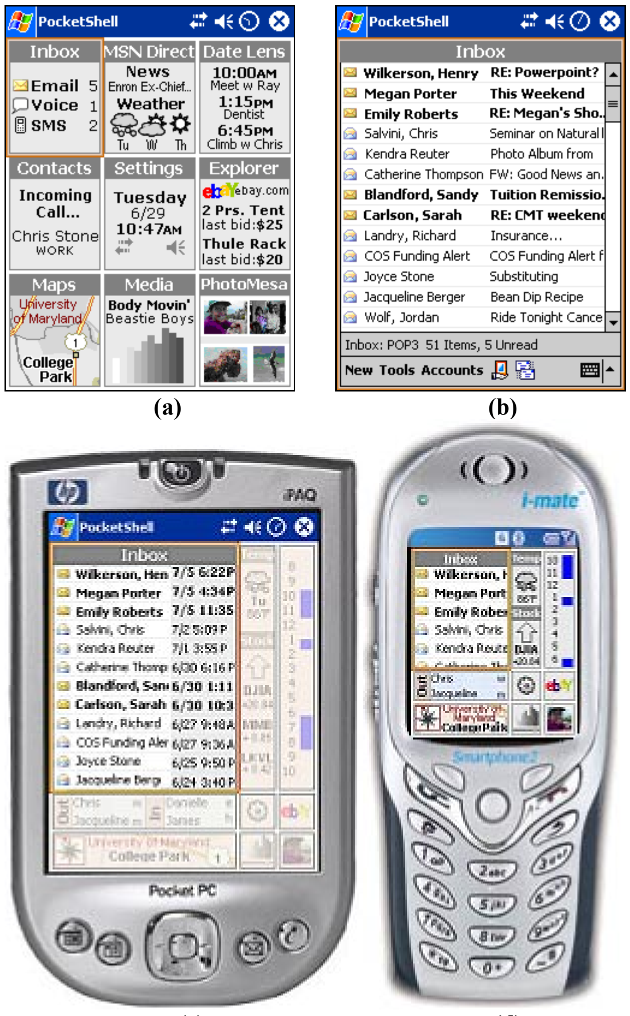
AppLens provides one-handed access to nine applications, and is strongly motivated by DateLens, a tabular fisheye calendar [3]. We refer to AppLens as a "shell" application for its role in organizing and managing access to other applications.

Generalized Data Access Using Tabular Fisheyes

Spence and Apperley [27] introduced the "bifocal display" as one of the first examples of fisheye distortion applied to computer interfaces. Furnas extended the bifocal display to include cognitive perceptual strategies and introduced a set of analytical functions to automatically compute generalized fisheye effects [8]. Since then, fisheye distortion has been applied with mixed success across a variety of domains, including graphs [24], networks [26], spreadsheets [20], and documents [12,23].

Bederson et al. [3] drew upon that work in developing DateLens, a space-conserving calendar for PDAs, which was shown to perform favorably for long-term scheduling and planning tasks when compared to a traditional calendar implementation. One of the strengths of DateLens was the pairing of distortion and scalability, which allowed the details of a single day to be viewed in the context of up to several months of appointment data. Also important was the design of effective representations for the variety of cell sizes and aspect ratios that resulted from tabular distortion. One drawback of DateLens, however, was that it required two-handed stylus interaction to actuate the small interface widgets. Our design extends the principles of DateLens to include one-handed thumb access and generalizes the approach for use across a variety of domains.

We developed a scalable framework that provides a grid, tabular layout, and default views for cell contents at a variety of sizes and aspect ratios. We also developed a general API to make it simple for applications to be built within this framework; developers need only to replace a small number of cell views with representations that are meaningful within the target domain.



Wilkerson, Henry	RE: Powerpoint?
Megan Porter	This Weekend
Emily Roberts	RE: Megan's Sho...
Salvini, Chris	Seminar on Natural I
Kendra Reuter	Photo Album from
Catherine Thompson	FW: Good News an...
Blandford, Sandy	Tuition Remissio...
Carlson, Sarah	RE: CMT weekend
Landry, Richard	Insurance...
COS Funding Alert	COS Funding Alert f
Joyce Stone	Substituting
Jacqueline Berger	Bean Dip Recipe
Wolf, Jordan	Ride Tonight Cance

Inbox: POP3 51 Items, 5 Unread

New Tools Accounts

Figure 1. AppLens Zoom Levels: (a) Notification, (b) Full, (c, d) Context.

AppLens Zoom Levels

The AppLens shell (Figure 1) has been implemented within our generalized tabular fisheye framework, using a 3x3 grid, and assigning one of nine applications to each cell. The support for tabular layout includes representations at 3 zoom levels: *Notification*, *Context* and *Full*.

Notification zoom distributes the available screen real estate equally among the 9 application tiles (Figure 1a). One tile (shown centered) remains reserved for settings, which can be used to configure the selection of applications which occupy the other 8 notification tiles. Generally, tiles at *Notification* size display high level static and/or dynamic application-specific notification information.

Context zoom (Figure 1c) allocates roughly half the available display area to a single *focus* tile, compressing the remaining tiles according to a tabular fisheye distortion technique [3,20]. A tile at *Context* size typically appears much like a fully functional application, but selectively displays features to accommodate display constraints, and is not interactive. Tiles on the periphery of a *Context* zoom, called *peripheral* tiles, may be rendered quite differently depending on their position relative to the focus tile, which dictates whether the peripheral tile is a square, a wide-flat rectangle, or a narrow-tall rectangle. To reduce visual overload, peripheral tiles are displayed at 40% transparency. The contents of distorted peripheral tiles are not themselves distorted, but rather change representation to provide the most meaning in the space available.

The third and final *Full* zoom level expands a tile to a fully interactive application that occupies 100% of the display (Figure 1b).

Gesture-Based Cursor Navigation

Existing application designs for PDAs are often inappropriate for one-handed use due to their reliance on screen regions that typically cannot be reached while maintaining control of the device (e.g., accessing the Start menu in the upper left-hand corner of a display while holding the device in the right hand), and the use of standard widgets that are too small for reliable thumb use (e.g., radio buttons, checkboxes, and on-screen keyboards). In support of traditional designs, AppLens uses an object cursor to identify the on-screen object that is the current interaction target. The cursor is depicted as a dynamically-sized rectangular orange border that users move from one on-screen object to another via command gestures. Cursors are not new to PDA interface design: the WebThumb [31] web browser includes a similar notion of cursor, but which is controlled via directional hardware, and others [29] have explored device tilting to manipulate PDA cursors.

Neither the cursor nor gestures interfere with the most common stylus interactions of tap and tap+hold. Although gestures do overlap stylus drag commands, dragging is rarely used in handheld applications and could be distinguished from gestures by explicitly setting a gesture input mode.

We established a core set of commands that would allow users to navigate applications using only the input cursor. Our command language supports directional navigation (UP, DOWN, LEFT, RIGHT) as well as two widget interaction commands: one equivalent to a stylus tap (ACTIVATE), and the other which negates widget activation (CANCEL), equivalent to tapping the stylus outside the target widget. We also include the convenience commands FORWARD and BACKWARD, equivalent to TAB and SHIFT-TAB on Windows PCs.

Command Gestures

Our use of gestures is motivated by Hirotaka's observation that the button positions on cell phones require interaction using a low, unstable grip [11]. PDA joysticks face a similar drawback in that they are typically located along the lower perimeter of the device. AppLens avoids this problem since its gestures can be issued anywhere on the screen. Each AppLens gesture is uniquely defined by a slope and direction, or vector, which allows gestures to be robust and highly personalizable; users can issue gestures of any length (beyond an activation threshold of 20 pixels) anywhere on the touch-sensitive surface. This flexibility lets users interact with our designs using the grasp that provides maximum stability in a mobile scenario.

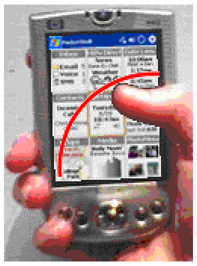


Figure 2. Screen area accessible with one hand.

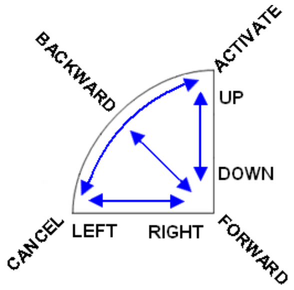


Figure 3. The gesture set.

We based the gesture set on the limited motion range of thumbs (Figure 2), with the goal of creating a gesture language that could be learned with minimal training. After informally experimenting with a variety of gestures, we developed a simple set of gestures with the aim of maximizing memorability and robustness of execution (Figure 3). We assigned the directional commands UP, DOWN, LEFT and RIGHT to spatial gestures that map directly to their function. We assigned ACTIVATE and CANCEL to the two gestures defined by pivoting the thumb from bottom to top and top to bottom respectively. We made these assignments both to reinforce their opposing relationship as well as for ergonomic ease in issuing common commands. Finally, we assigned the upper-left to lower-right diagonal to FORWARD due to its relative forward nature, and by similar reasoning, the reverse gesture to BACKWARD.

The eight gesture commands can also be activated with a numeric keypad by mapping each to the key corresponding to the gesture endpoint: 1-BACKWARD, 3-ACTIVATE, 7-CANCEL, and 9-FORWARD. Since smartphones have a joystick that can be used for directional navigation, the keypad-to-command mapping is not necessary, but can be assigned as follows: 2-UP, 4-LEFT, 6-RIGHT and 8-DOWN.

Using Command Gestures within AppLens

Users navigate between AppLens zoom levels using ACTIVATE and CANCEL gestures. As a rule, the ACTIVATE gesture behaves as a stylus tap on the current cursor target, thus its effects are target-specific. Since application tiles are non-interactive at *Notification* and *Context* zoom levels, the ACTIVATE gesture simply zooms in, animating the layout first from *Notification* to *Context* zoom, and then to *Full* zoom. Once at *Full* zoom, the input cursor transitions to the objects *within* the application, at which point the command gestures affect the current target widget. The CANCEL command negates the effects of the ACTIVATE command. At *Full* zoom, the effects of the CANCEL command depend on the location of the cursor and the state of its target. The CANCEL command will first deactivate an active target. If the current target is not in an active state, CANCEL will cause the application tile to animate from *Full* zoom to *Context* zoom, and if issued again, to *Notification* zoom.

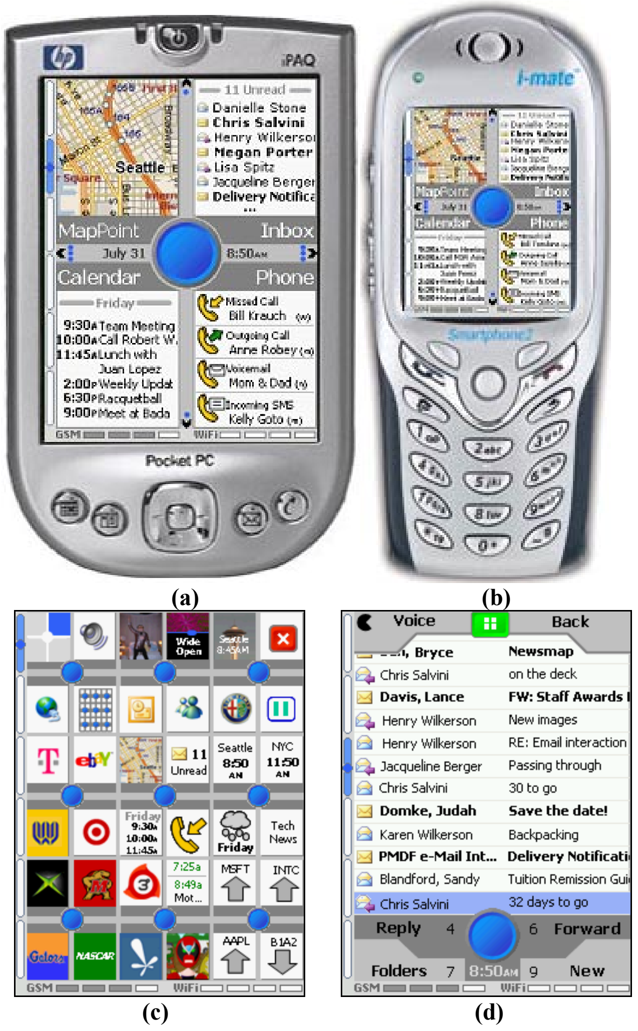


Figure 4. Three LaunchTile zoom levels: (a, b) Zone, (c) World, (d) Application

THE ZOOM+PAN APPROACH: LAUNCHTILE

Our second design, LaunchTile proposes another way to interact with a grid of notification tiles. The primary shell of the LaunchTile design is an interactive zoomspace consisting of 36 application tiles, divided into 9 zones of 4 tiles each (Figure 4c). The 36 tile configuration is an exploration of the maximum number of applications that can reasonably fit within the display space. Since the design is fundamentally scalable, however, it can display any number of tiles *up to* 36, and fewer may even be preferable. As a central design element, LaunchTile uses a large blue onscreen button, called Blue (Figure 4a), to unify the shell and applications with a consistent visual metaphor. Blue provides a consistent point of reference for zooming and panning navigation, with onscreen tiles, menus, and functions maintaining a consistent relative position to Blue. The LaunchTile zoomspace consists of 3 zoom levels: World (36 tiles, Figure 4c), Zone (4 tiles, Figures 4a and b) and Application (1 tile, Figure 4d).

Zone View

The Zone view of LaunchTile divides the screen area into 4 equally-sized application tiles (Figure 4a). To view other tiles, the user pans the zoomspace to neighboring 4-tile clusters, called *zones*. The zones are arranged as a 3x3 grid, each associated with a numerical designator from 1 to 9 as on a conventional telephone keypad. Zone 5 is the center zone, which defines the Home screen, and shows the 4 highest priority notification tiles, as configured by the user.

Panning Techniques

To support various input hardware and styles of interaction, there are several ways to pan the zoomspace within Zone view. If the device has a multidirectional control joystick, the user can push it in the direction of the targeted zone. From the Home screen, the 16 tiles in zones 2, 4, 6, and 8 are only a single tap away. As many directional pads do not support diagonal action, the 16 additional tiles in the corner zones 1, 3, 7, and 9 are two taps away. Alternatively, if the device has a touch-sensitive display, the user can use his or her thumb directly to drag the zoomspace. Dragging is performed "on rails", permitting the user to drag vertically and horizontally, but not diagonally. Although the zoomspace moves with the thumb during dragging, Blue remains centered and stationary. Because only one instance of Blue exists within Zone view, each zone is depicted with an empty center hub during dragging. Upon thumb release, the zoomspace animates to align Blue with the closest zone's empty hub. The visual and automated guidance ensures the user is never caught between zones.

Within each 4-tile zone, indicator widgets communicate the user's relative location within the zoomspace. The indicator has two components. First, directional arrows show where other zones are. If users only see indicators pointing up and right, they know they are in zone 7. Small blue dots featured with the arrows represent the location of all zones not currently in view. The blue dots could also be used to indicate an alert or status change in a neighboring zone, though this feature was not implemented in our prototype. The final way to pan the zoomspace is to tap the directional indicator itself. An oversized hit target ensures that the user can easily hit the indicator without using a stylus.

Zooming Out to the World View

From any one of the nine 4-tile Zone view zones, the user may tap Blue (or press the 5 key) to zoom out to view the entire 36-tile zoomspace (Figure 4c). Since all 36 tiles are visible at once, this view reduces each tile to a small icon. From this World view, the user can easily see the absolute location of each tile, as well as monitor all applications at once. In the World view, the display is divided into a grid of 9 hit targets, each mapping to a 4-tile zone. Single-tapping a zone animates to Zone view, displaying the zone's 4 notification tiles.

Zooming In to an Application

The user taps any of the 4 notification tiles within Zone view to launch the corresponding application. An animated zoom draws the zoomspace toward the user until the target application fills the entire display (Figure 4d). If the device has only a numeric keypad (no touchscreen), the user presses the numeric key in the corner that corresponds to the zone. Pressing 1 launches the upper left tile, 3 launches the upper right, 7 launches the lower left, and 9 launches the lower right. This technique provides quick, single-tap access to each visible tile, and was inspired by ZoneZoom of Robbins et al. [22].

As the system zooms, Blue stays in view and retains its function as a central point of reference (Figure 4d). Application menu commands are represented as on-screen buttons clustered around Blue, now positioned at the bottom of the display. Each menu button displays a numeric label, so that mobile phone users may activate each menu by pressing the corresponding number on the keypad. A visual indicator to the left of the zoomspace animates during interaction to reflect the user's current absolute zoom level within the LaunchTile environment.

Zoom Control

Pressing Blue typically moves the user deeper into the object hierarchy, while a dedicated Back button moves the user up the hierarchy. In the Zone view however, Blue toggles between Zone view (4 tiles) and World view (36 tiles). Once an application is launched, three dedicated software buttons along the top edge of the screen support inter- and intra-application navigation (Figure 4d). A green Home button immediately zooms the view out to the Home screen. There is also a Back button on the upper right edge of the screen, and another global command key. The placeholder for this function in our prototype is an icon for voice command and control. On a non-touchscreen device, Back and Home commands are executed with dedicated physical buttons, such as those provided on a smartphone.

Application-Level Interaction

Although the original focus of our designs was on the application "shell", we extended the LaunchTile interaction philosophy to the application level, where we sought to make interaction consistent with navigation among the application tiles. Several gestures have been designed to specifically support one-handed navigation and item selection within applications. Previously, others have demonstrated that for direct-manipulation interfaces, a grounded tap-drag-select-release technique is more accurate than a tap-to-select [19]. We therefore made all LaunchTile tap-to-select targets large enough for thumb activation. In cases when limited display real estate necessitates smaller targets, the central Blue widget serves as a moveable tool glass which can be positioned over the target object (e.g., email header, message text). The large thumb-friendly drag target is offset below the selection area to prevent the user's thumb from occluding the target. Alternatively, the user can drag the application

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.