



ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #
18/478,360

RECEIPT DATE / TIME
09/29/2023 04:06:06 PM ET

ATTORNEY DOCKET #
10875-10150 US

Title of Invention

SYSTEM AND METHOD FOR DETECTING A CHANGE IN CONTEXT OF AN APPLICATION USING SUBSECTIONS

Application Information

APPLICATION TYPE	Utility - Nonprovisional Application under 35 USC 111(a)	PATENT #	-
CONFIRMATION #	7247	FILED BY	Edward Van Gieson
PATENT CENTER #	62900386	FILING DATE	-
CUSTOMER #	93219	FIRST NAMED INVENTOR	Sneha Saxena
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	Edward Van Gieson

Documents

TOTAL DOCUMENTS: 6

DOCUMENT	PAGES	DESCRIPTION	SIZE (KB)
10150 US - Application Data Sheet.pdf	8	Application Data Sheet	2174 KB
10150 US - Declaration.pdf	3	Oath or Declaration filed	242 KB
10150 US - Power of Attorney.pdf	3	Power of Attorney	317 KB
10150 US - Specification-APP.TEXT.docx	40	Application body structured text document	57 KB

Warning: Text decorations have been removed. Bookmarks were found and have been removed.

10150 US - Drawings.pdf	17	Drawings-only black and white line drawings	2500 KB
10150 US - Specification.pdf	40	Auxiliary PDF of Application	190 KB

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
10150 US - Application Data Sheet.pdf	47BB162DB7A87A0294EBF4125060024232A8886D76BCE8B0E4E66E006FB5076777412ADB24B1B49CE78E61D6ADEE4A6192370C93DA7ADAB1A9B5F148CCB38575
10150 US - Declaration.pdf	379269A8D39E3BECF0447DBDB728D717D43CCD90A0A75CCDE7DF8DE5ACA3A8865B005DA1E362A3469008E957AFD2B64A22CB97BEC5A0FFC0ADA0958D655994D2
10150 US - Power of Attorney.pdf	190F0A6784467FB1839EA9B2B615ECB7CF52FE5D129BFDB86A4C8DD5ABC716D95C590D2B9101713B600B70625A2F09C0935FACF09D0D4374E3519D87AE42B1D6
10150 US - Specification-APP.TEXT.docx	F883D63B5780D634DE056C020F375E63961DF2FC04682918C8D4C1257DBF184A6D346FC1538E2D733CEA91573A3443F2F6F77DC2B770DE089B3F034FBF7BC7BE
10150 US - Drawings.pdf	4192875D03FD98F5924A89055B3052EEAED30EF63D665A731456E7795977CA45E30A6926D824A2385F51F59E03C574D94292A7BCDFE5BBE767EF099790B22CCA
10150 US - Specification.pdf	21C7F920146E7729E9C1BB8BFA635C6695BA86BBFC7D4C2D8F8949A6458FCA83ECF0426FFA4CDF4B619FFBAC27F35DEFDC2FD2505008D36B29E4E9AEA5EA2E6F

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d))

and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.



ELECTRONIC PAYMENT RECEIPT

APPLICATION #
18/478,360

RECEIPT DATE / TIME
09/29/2023 04:06:06 PM ET

ATTORNEY DOCKET #
10875-10150 US

Title of Invention

SYSTEM AND METHOD FOR DETECTING A CHANGE IN CONTEXT OF AN APPLICATION USING SUBSECTIONS

Application Information

APPLICATION TYPE	Utility - Nonprovisional Application under 35 USC 111(a)	PATENT #	-
CONFIRMATION #	7247	FILED BY	Edward Van Gieson
PATENT CENTER #	62900386	AUTHORIZED BY	Edward Van Gieson
CUSTOMER #	93219	FILING DATE	-
CORRESPONDENCE ADDRESS	-	FIRST NAMED INVENTOR	Sneha Saxena

Payment Information

PAYMENT METHOD CARD / 1002	PAYMENT TRANSACTION ID E20239SG08076200	PAYMENT AUTHORIZED BY Edward Van Gieson
PRE-AUTHORIZED ACCOUNT 603148	PRE-AUTHORIZED CATEGORY 37 CFR 1.16 (National application filing, search, and examination fees)	

FEE CODE	DESCRIPTION	ITEM PRICE(\$)	QUANTITY	ITEM TOTAL(\$)
1111	UTILITY PATENT APPL. SEARCH FEE	700.00	1	700.00
1011	BASIC FILING FEE - UTILITY (PAPER FILING ALSO REQUIRES NON-ELECTRONIC FILING FEE UNDER 1.16(T))	320.00	1	320.00
1311	PATENT APPL. EXAMINATION FEE	800.00	1	800.00
1202	EACH CLAIM IN EXCESS OF 20	100.00	1	100.00

TOTAL	\$1,920.00
AMOUNT:	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

SYSTEM AND METHOD FOR DETECTING A CHANGE IN CONTEXT OF AN APPLICATION USING SUBSECTIONS

TECHNICAL FIELD

[0001] The present disclosure is related to determining a change in a page or context of an application for use in a digital adoption platform.

BACKGROUND

[0002] A digital adoption platform (DAP) is a type of software that is layered on top of another software, app, or website to help facilitate end user proficiency by helping to guide users through key tasks and provide contextual information as users navigate the user interface of the product. Users are provided with information to help familiarize them and become more proficient. This helps to drive adoption.

[0003] For example, a DAP may generate a help tip. Background information on an example DAP implementation is found in various sources, including U.S. Pat. No. 11,372,661 and U.S. Pat. No. 11,461,090, assigned to Whatfix Private Limited, the contents of each of which is hereby incorporated by reference. A DAP supports content authoring modules and content playback modules to generate, for example, smart tips as a user navigates elements of a user interface of an underlying software application.

[0004] A DAP supports content creators creating new flows or other guided features to enable higher adoption of client applications. Content Creators of the product can create content, record a flow, and the content is played back the same flow as and when required when end-user clients navigate the client application. This requires the DAP to find the visual UI elements on the application the user is looking at (e.g., finding where the user's cursor is located on a graphical user interface).

[0005] An application may have different pages with different screen element formats. For example, an application may generate graphical user interfaces with pages for different purposes, such as a calendar page, a teleconference page, etc. Also, even for a visual UI for a particular purpose (e.g., document management), there may be a variety of different pages for different features that have variations in the arrangement or display of user interface elements. That is, as a user navigates an application, there may be major changes in the visual UI as well as minor changes in the visual UI. For either case, the DAP needs to know what page the user is looking at to provide the correct smart tip.

[0006] Consequently, a DAP needs to know what page the user is looking at to aid in understanding visual UI elements on the application that the user is looking at. Conventionally, identifying pages on a desktop application relies on Automation API Interfaces for different technologies like SAP-GUI, UIAutomation, Java Access Bridge etc. However, this is a slow process because uniquely identifying the pages is a memory and CPU intensive task. This makes the process unusable on minimum configuration machines. Some of the problems with these approaches include that each technology like WinForm, SAPGui, Java Swings app needs a different library and algorithm to identify the elements. Additionally, automation APIs are slow and are CPU intensive so identifying context takes time in every technology.

[0007] The current state of the art for detecting a change in a page or a context of an application is based on primarily three approaches:

[0008] 1) Window foreground change,

[0009] 2) Window title change, and

[0010] 3) Unique user interface element on the page.

[0011] These three approaches are achieved through System level and Object level events generated by the operating system. System-level events describe situations affecting all applications in the system. Object-level events pertain to situations specific to objects (UI elements) within one application.

[0012] The events used in the current state of the art to identify a change in context of the active client application are:

[0013] A. EVENT_SYSTEM_FOREGROUND: In this approach, the event when raised by the operating system depicts a change in the foreground. The system sends this event even if the foreground window has changed to another window in the same thread;

[0014] B. EVENT_OBJECT_NAMECHANGE: In this approach, the system sends this event for the following user interface elements: check box, cursor, list-view control, push button, radio button, status bar control, tree view control, and window object. This can be used to determine if there is any change in the title of the current window. This is specifically useful when there is no change in the foreground but the page within the current window is changed; and

[0015] C. UIAutomation Element Visibility: This approach looks for an element on the page using properties which denote a single element on a page. For example, consider looking for an element with object level properties like Name=Chat and AutomationID=Chat0021 at regular intervals. When an element with the given properties is found, an assumption can be made that the current page is a Chat window. Elements that are unique to a particular context of application are chosen further to identify change of context. Properties such as name, automation id, control type, etc., of these unique elements are then classified to identify context that is

required for segmentation, where segmentation is a technique to customize DAP experiences based on individual or group preferences or needs.

[0016] The above system and object events are usually sufficient when building applications which are not context aware. Digital adoption platforms like Whatfix DAP rely heavily on the context of the application to provide relevant and most accurate help content on the page. Smart Tips, Beacons, Guided Walkthroughs are all built around the context of the application, so a highly accurate algorithm is an integral part of a DAP solution.

[0017] The above events suffer from the drawback that it's impractical to achieve a high accuracy in terms of identifying the context with close to 100% accuracy. There are four basic challenges.

[0018] First, one obstacle is that most applications do not have reliable titles sufficient to identify if a context change has happened. Some legacy applications built using WinForms have a single title throughout all pages. This makes title-based identification of context useless.

[0019] Second, in some customer applications, the applications do not raise any title change event even when there is a visible change in the title. This usually happens when the system cannot detect a title change due to the way the application is built by the developers.

[0020] Third, foreground change events are usually the most accurate when it comes to context change identification when the entire window is changed. But on some customer applications a foreground change event is triggered when the user hovers over certain UI elements on a page. Certain UI elements creating a system help text tooltip which is also considered a new window by the operating system. This triggers an unwanted foreground change event causing the DAP application to re-check the current page context. This can be partially

mitigated by various means, including checking the size of the window which raised the foreground change event.

[0021] Fourth, the UI Automation based approach for looking at certain elements on all pages at regular intervals has two main issues. One is that it is a CPU intensive operation, so it can be used judiciously and only when customers have better than average hardware. Second is that this requires a lot of manual effort by a technical person to identify unique elements on all required pages. This work cannot be done by the content author as this is a highly technical work.

[0022] Due to the above four challenges, page context change detection remains a difficult task with low accuracy.

SUMMARY

[0023] A digital adoption platform includes a page change identification technique. The page change identification technique is platform and application agnostic. Changes in screenshots of a UI of an application are compared. In one implementation, differences between screenshots are compared using a cross-correlation technique. In one implementation, trigger conditions are defined to reduce the computational resources required.

[0024] An example method of detecting page changes of a graphical user interface (GUI) application screen includes determining if a trigger condition is satisfied for a page change, the trigger condition being based on a combination of events associated with a likely page change of an application. Subsequent to satisfying the trigger condition, a sequence of at least two screenshots of the application is captured taken at different times, dividing each screenshot into at least two subsections, and comparing analogous subsections of the at least two screenshots using a cross correlation function. The method includes identifying a page change in response to

the cross-correlation function, for at least one subsection, having a value below a threshold value indicative of a page change.

[0025] In one implementation, the at least two subsections consist of quadrants.

[0026] In one implementation, the method further includes performing a pre-processing operation to identify blank portions of a screenshot and in response apply at least one rule for dividing screenshots into at least two subsections.

[0027] In one implementation, the pre-processing operation identifies an order for processing subsections based at least in part on the blank portions.

[0028] In one implementation, the pre-processing operation applies at least one rule to improve an accuracy of cross-correlation computations.

[0029] In one implementation, dividing each screenshot into at least two subsections is selected to improve an accuracy of cross-correlation computations.

[0030] In one implementation, the cross-correlation function is evaluated based on a reference screenshot captured in response to the trigger condition and a subsequent screenshot.

[0031] In one implementation, the trigger condition comprises operating system events.

[0032] In one implementation, the trigger condition comprises a combination of reorder events, focus events, mouse events, and keyboard events.

[0033] In one implementation, the trigger condition comprises a first trigger condition comprising detecting mouse events and keyboard events followed by a second trigger condition comprising detecting reorder events and focus events.

[0034] In one implementation, the threshold value is in a range between 0.5 to 1.0.

[0035] In one implementation, the method includes performing a stability test, based on screenshots of the application, to determine if the user interface is stable.

[0036] In one implementation, the stability test is based on performing a cross-correlation check for at least two screenshots taken after detecting a change in page content.

[0037] In one implementation, the method includes performing at least one retry if the cross-correlation function does not have a value below a threshold value indicative of a page change.

[0038] In one implementation, the sequence of at least two consecutive screenshots is converted into grayscale images to reduce the number of dimensions for analysis.

[0039] In one implementation, the method includes caching grayscale versions of at least a first screenshot in the sequence of at least two consecutive screenshots.

[0040] In one implementation, the sequence of at least two consecutive screenshots comprises screenshots of at least two different color images, with the cross-correlation function being performed on color images.

[0041] In one implementation, the method includes utilizing the page change identification information for a digital adoption platform to recognize page changes for which guidance is provided by the digital adoption platform.

[0042] In one implementation, a system for identifying page changes of a graphical user interface, includes a page change identification engine configured to determining if a trigger condition is satisfied for a page change, the trigger condition being based on a combination of events associated with a likely page change of an application. Subsequent to satisfying the trigger condition, the method includes capturing a sequence of at least two screenshots of the application taken at different times, dividing each screenshot into at least two sub-sections, and comparing analogous subsections of at least two screenshots using a cross correlation function.

The method includes identifying a page change in response to the cross-correlation function having a value below a threshold value indicative of a page change.

[0043] In one implementation, the at least two sub-sections consist of quadrants.

BRIEF DESCRIPTION OF THE DRAWINGS

[0044] Fig. 1 is a high-level drawing of a DAP system illustrating components to identify page changes of an application in accordance with an implementation.

[0045] Fig. 2 is a high-level flowchart of a method to identify page changes of an application in accordance with an implementation.

[0046] Fig. 3A is high-level flow chart illustrating an example of operating system events, keyboard events, and mouse events for triggering a page identification process in accordance with an implementation.

[0047] Fig. 3B illustrates an example of a cross-correlation function in accordance with an implementation.

[0048] Fig. 4 is an interaction diagram illustrating a process of page identification in accordance with an implementation.

[0049] Fig. 5A is a flowchart of a method of performing page identification in accordance with an implementation.

[0050] Fig. 5B is a continuation of the flowchart of Fig. 5A, from line A-A, in accordance with an implementation.

[0051] Fig. 6 illustrates a first example of an application context, showing an example user interface in accordance with an implementation.

[0052] Fig. 7 illustrates a second example of an application context, showing an example user interface in accordance with an implementation.

[0053] Fig. 8 illustrates a third example of an application context, showing an example user interface in accordance with an implementation.

[0054] Fig. 9 illustrates a visualization of the difference between UI pages in the first context of Fig. 6 and the second context of Fig. 7.

[0055] Fig. 10 illustrates a visualization of the difference between UI pages in the first context of Fig. 6 and the third context of Fig. 8.

[0056] Fig. 11 illustrates an example of a UI page in which the blank area is a majority.

[0057] Fig. 12A illustrates an example of segmentation of the UI page of Fig. 11 into quadrants to aid in comparing images in accordance with an implementation.

[0058] Fig. 12B illustrates an example of segmentation of the UI page of Fig. 11 into three segments to aid in comparing images in accordance with an implementation.

[0059] Fig. 12C illustrates an example of segmentation of the UI page of Fig. 11 into six segments to aid in comparing images in accordance with an implementation.

[0060] Fig. 12D illustrates an example of segmentation of the UI page of Fig. 11 into two segments to aid in comparing images in accordance with an implementation.

[0061] Figs. 13A and 13B illustrate a general computer environment for deploying the page identification technique in accordance with an implementation.

DETAILED DESCRIPTION

[0062] Fig. 1 is a block diagram of a high-level implementation of an example of system for context/page detection for a digital adoption platform 102. The digital adoption platform 102 provides guidance for a target client application 101, which may, for example, be displayed on a display screen of a desktop computer 100. A target client application 101 running on the desktop of a client computer 100 generates a graphical user interface (GUI) 102 that may be displayed on

a computer display screen of the client computer 100. The target client application may, for example, generate different application display screens (e.g., a calendar screen, a teleconference screen, etc.). Each of the application screens corresponds to a page of the application. Accurately identifying a page change/context change of the application that is being displayed on the user's computer display screen is important to provide the correct DAP guidance.

[0063] In an exemplary implementation, a digital adoption platform (DAP) 102 requires page change information to generate smart tips from content providers. The DAP is an additional software layer to provide help tips for the target client application.

[0064] In one implementation, the DAP 102 includes a DAP player process engine 104 to implement a player process; a Web process manager 106 to implement a Web process; a native system process engine 108 to implement a native process; and a guided assistance process engine 110 to implement a guided assistance process.

[0065] In one implementation, a page/context change detection engine 120 is provided. In one implementation, it includes a screenshot capture unit 122 to capture screenshots of screen images (or subsections thereof). In one implementation, a screenshot of an image may be resized and converted into grayscale for further analysis regarding changes to a displayed page. A trigger condition may be used to determine when an initial screenshot is captured. A configurable time may be selected for each consecutive subsequent screen capture, with 300 milliseconds being an exemplary time interval between consecutive screenshots.

[0066] In one implementation, a screen stability unit 124 determines screen stability by using a stability score based on comparing a sequence of screenshots. For example, if a screen UI page is still loading, it's not yet stable. For example, if it takes about 2 seconds for a screen UI page to load, and if the interval between screenshots is 300 milliseconds, then a number of

screenshots will be captured before the UI page has finished loading. In one implementation, a cross-correlation technique is used to evaluate screen stability.

[0067] In one implementation, a screen similarity engine 126 analyzes differences between screenshots, or in some implementations, subsections thereof. In one implementation, a cross-correlation technique is used and cross-correlation thresholds are used to identify a page/context change. In one implementation, the difference between images is measured using a cross-correlation technique that has a score between -1 and 1, where -1 denotes a complete change and 1 denotes no change at all.

[0068] For example, a cross correlation technique close to 1, such as a cross correlation reference value of 0.95, may be used to identify when images are stable. In particular, when a page has finished loading (or is nearly finished loading), then two consecutive screen images taken when the page loading is nearly complete will have almost no change, and hence the cross-correlation value will be close to 1.

[0069] However, a lower cross-correlation value between two images, is useful to identify when there is a page change where one of the images is a reference screenshot image taken prior to the page change (e.g., a screenshot taken when a trigger condition is satisfied). For example, a cross-correlation threshold value of 0.80 to 0.90 in comparison to an initial reference image may be used to detect a page change.

[0070] In one implementation, a threshold value of the cross-correlation computation of around 0.8-0.95 is used to define a context change for an application. Cross-correlation scores below this threshold indicate that there is a context change compared to the previous state of an application prior to a trigger condition being satisfied. This is an empirically derived range of threshold values, which may be adjusted for a particular application. This is because for some

applications, the application's UI and colors are mostly similar for each page (or for many pages) but may differ slightly in data format or presentation. For a particular application, the cross-correlation coefficient threshold for each page/context change may be logged under the file and could be analyzed through it. In one implementation, a cross correlation threshold value is an incremental change (a delta change value) relative to an average value.

[0071] Performing cross-correlation calculations on grayscale screenshots is computationally less intensive than for colored images, which is a consideration given that some end users may have low end computer devices. It should be noted that better correlation score accuracy can be achieved on color images. It will be understood that the process may be configurable or otherwise be adapted to the capabilities of a user's client computer. For higher end computing devices, the cross-correlation scores may be calculated from color images. However, the process may calculate cross-correlation scores for grayscale images for low end client devices. In some implementations, calculating cross-correlation scores on grayscale images may be the default setting.

[0072] In some instances, a significant portion of a screen may be either blank or otherwise have very little content that reflects a difference in a screen in a cross-correlation computation. For some cases, a significant percentage of the UI screen may be sparse or blank in terms of changeable content. In some implementations, dividing the screen image into subsections may be used to reduce computational effort and improve the accuracy of cross-correlation computations. For example, a screen image may be divided into two or more subsections, with a division of a screen image into four quarters (quadrants) being one example of this. Rules may be defined for a particular application to identify conditions for dividing the screen image into subsections is performed. As an illustrative example, consider sectioning a

screenshot into quarters (i.e., four quadrants of equal area). However, the technique may be generalized to include dividing a screen image into two or more subsections. A screen subsection engine 128 is included in one implementation to divide a screen into smaller sub-sections (e.g., by rectangular quarter sections as one example).

[0073] In one implementation, a subsection pre-processing unit may identify empty (or comparatively empty) portions of a UI screen. This may, in some implementations, be based on image pre-processing to identify what portions of a displayed page are blank or that are otherwise irrelevant for analyzing differences in images.

[0074] This section pre-processing can be implemented in an application agnostic manner. However, if additional application information is available, this could also be considered. For example, if additional information is available on how a particular application lays out content that might also be considered. For example, some applications are more likely to have blank sections in particular sections of a page (e.g., top vs. bottom of page; left vs. right section of page).

[0075] In one implementation, a subsection rules module 132 is provided to implement rules to determine when to divide a screen into smaller subsections, and rules to process subsections. For example, a UI screen may be sectioned into two or more subsections (e.g., two, three, four, six, eight, etc.). The subsections may also be oriented/arranged differently (e.g., vertical strip segments vs. horizontal strip segments as one possibility; or a grid array such as a 2x2 into quarter sections or a 3 x3 array into 9 sections as another example). While the subsections may be of equal size, more generally they may be of different size. For example, some applications may have pages with an uneven distribution of content in the sense that the blank sections are not symmetrically distributed. For example, some pages may have more of the

content near the top or the bottom of the page; or more of the content located on the right or the left side of the page.

[0076] An order for processing subsections and performing cross-correlation computations on analogous subsections may also be defined. For example, if a screen image is divided into four quadrants Q1, Q2, Q3, and Q4, the cross-correlation computations may be performed on in analogous quadrants. For example, if all subsections but one is blank, then only the one non-blank subsection needs to be used in an image comparison. For example, if the division is into quadrants, and only one quadrant had content, then the cross-correlation may be limited to that one quarter of the image.

[0077] In one implementation, an event trigger engine 134 identifies conditions for triggering computationally expensive image process calculations for identifying page changes. The trigger conditions may include, for example, keyboard events, mouse events and operating system events that are indicative of a likely upcoming page change. For a particular operating system, a table may be defined of a combination of OS events, keyboard events, and mouse events. An optimal sequence of events may also be considered. For example, in one implementation, Keyboard and Mouse events are first detected before moving on to listening for focus and UI reorder events and then deciding to start an image difference algorithm check.

[0078] In one implementation, a screen pixel caching module 136 includes a cache (or access to a cache) and cache rules to cache screen pixels. For example, after a trigger condition is satisfied, an initial screenshot image (or a grayscale version thereof) may be cached and used during a page identification process that may include, for example, a series of comparisons with a series of consecutive screenshot images. Performing screen pixel caching in a page change

identification process provides a variety of performance benefits. In one implementation, grayscale pixels are cached according to cache rules optimized for page change detection.

[0079] Retry logic 138 supports performing retries, such as during a time period when the screen UI is not stable. For example, a maximum number of retries (e.g., 3, 4, 5, or 6) may be supported. Alternatively, there may be a maximum time interval during which retries are permitted. In some implementations, the maximum number of retries is configurable.

[0080] An event listening module 140 may be provided to support the page/context change detection engine 120. For example, in one implementation, Operating System event listeners may listen for a Focus change, Keyboard event, Mouse Activities and System UI elements reorder events.

[0081] In one implementation, a configuration engine 142 is provided for a user or an administrator to adjust/select configurable parameters. This may include, by way of example, the number and arrangement of subsections and arrangements of subsection 144, one or more other subsection rules 146, retry parameters 148, cross-correlation thresholds 150, and may also include other parameters, such as a timeout value.

[0082] One aspect of the technique for identifying page/context change detection is that because it relies on analyzing a difference in images it is platform and application technology independent for desktop applications. Desktop applications are typically built using technologies like .Net Framework, SAP GUI, JAVA etc. Using an image difference-based algorithm is platform and application technology independent because detecting differences between images is not tied to title change event, a foreground change event, or a specific element on the page. This solution is independent of the application technology on which it is built in and can work on a wide variety of different desktop applications be it Automation, SAP GUI, JAVA, Citrix or

Remote desktop compliant because they use system events and because capturing screen shots and performing image processing of captured screenshot images is technology agnostic.

[0083] In one implementation, when a trigger condition is satisfied, an initial screen shot image is captured, which is followed by one or more additional screen shot images taken slightly later in time, where, for example, the time interval between consecutive screenshots may be selected to be any reasonable number, such as 300 milliseconds, as an illustrative but non-limiting example. The time interval between consecutive screen shots may, for example, be based on an empirical understanding of possible minimal time periods after the trigger condition is satisfied for a page change to occur. There is thus an initial screen shot that is captured along with at least one subsequent screenshot image that is captured. If a page change occurs, there will be a difference in screenshot images that can be determined based on whether the cross-correlation value is less than a first threshold value.

[0084] In one implementation, a stability analysis is performed based on analyzing the cross-correlation value of later screenshots, relative to a second threshold value. The stability analysis may be used to determine that screen image is stable (e.g., the page is loaded or nearly loaded). This provides a number of benefits, including supporting providing DAP guidance when a page is stably loaded.

[0085] As previously discussed, in one implementation, differences between successive images are calculated using a cross correlation coefficient. A cross correlation threshold value is used to decide if there is a change in the page. As some examples, a threshold value of 0.8 to 0.85 may be used, although more generally a wider range of 0.7 to 0.9 might be possible in some implementations. In one implementation, the threshold value is configurable in the range of 0.5 to 1.0 to provide flexibility in configuring the page change detection process. For example, if

there is a default range of threshold values in the default range 0.7 to 0.9, the threshold value may be configurable to improve the accuracy of deciding if there is a change in a page. A default threshold value of 0.8 to 0/85 is arrived at through multiple experiments. These cross-correlation threshold values are in an approach in which the cross-correlation values are in the range between -1.0 (negative 1.0) to 1.0 (Positive 1).

[0086] The cross-correlation threshold value can be adjusted, if necessary, based on empirical results for particular types of applications, which may vary to the degree different pages change in content. The trigger events may be selected to include operating system events that are found in common operating systems and can, if necessary, be adjusted to account for minor differences in operating system events between different operating systems.

[0087] In one implementation, the trigger for starting the process of a page change identification algorithm utilizes a combination of OS events Focus change, UI reorder events, keyboard events and mouse click events.

[0088] In one implementation, a listener is used for a set of OS events such as focus change and UI reorder events. Reorder or Focus events come up whenever there is a change in UI. It was observed that this event is received even if the mouse is moved on the screen of the target application (for e.g., SAP), because it usually displays certain text elements on the screen at that point. Ordinarily, to switch between pages, mouse clicks or keyboard keys (Enter, Tab, Esc) are pressed by a user. Therefore, in one implementation, the trigger conditions include mouse events (e.g., mouse up) and a few keyboard events combined with reorder and focus events. In one implementation, there is a sequence in which a reorder/focus event is only considered if a mouse/keyboard event was received earlier. Thus, the trigger condition(s) identify when a page change is likely to occur. Once the combination of trigger events is received, image

processing starts, A difference in images is used to identify a page change. The identified page change, in turn, is eventually used to segment digital guidance content. That is, the page change is used to aid in determining the DAP guidance provided to an end user (e.g., providing DAP guidance based on their individual preferences or needs).

[0089] Fig. 2 is a high-level flowchart of a method of page/context change detection in accordance with an implementation. In block 202, a determination is made if a trigger condition is satisfied based on a combination of events. In block 204, an initial comparison of screenshot images may be performed in decision block 204 to determine if an image difference exists. If there is no difference, a page has not changed (block 206). If there is a difference, a determination is made in decision block 208 if a change to the UI is in process. If not, then the process moves to decision block 212. If yes, the process moves to block 210 to wait for the UI to become stable (e.g., satisfying a stability condition threshold). In block 212 a decision is made whether the change in the screenshot image is significant, which may be based on a cross-correlation similarity threshold condition. If not, then the process moves to block 206. If yes, in block 214 the process identifies that that page/context has changed.

[0090] Fig. 3A illustrates an example of a table for determining a combination of events that triggers the computationally expensive image processing aspects of the page/context change detection process. In some implementations, for the Windows operating system, this includes a combination of focus events, UI reorder events, keyboard events, and mouse events. However, more generally, other combinations of events could in theory be utilized that are indicative of a likely page change. To the extent that there are minor differences in the events (or event names) in different operating systems, the trigger conditions could be adapted for different operating systems. Thus, it will be understood that the basic technique may be applied to different

operating systems, including Windows, MAC, Linux operating systems, etc. It will also be understood that the trigger conditions can be varied from those illustrated.

[0091] In one implementation, the event listener listens for different OS events such as UI focus change and UI reorder events and when these events occur at a certain predefined combination as described in the table of Fig. 3A, the page change identification process is initiated for checking the difference between consecutive screenshots of the application to identify a context/page change.

[0092] Reorder or Focus events come up whenever there is a change in UI. It was observed this event is received, even if a user moves a mouse on the screen of the target application (for e.g., SAP), because it usually displays certain text elements on the screen at that point. However, greater reliability, in terms of a trigger condition, is achieved by considering a combination of focus or reorder events and mouse events and keyboard events. This is because ordinarily to switch between pages, mouse clicks or keyboard keys (e.g., Enter, Tab, Esc) are pressed by a user. Therefore, better trigger results are achieved by combining mouse events (e.g., mouse up) and a few keyboard events with reorder and focus events. In one implementation, reorder/focus events are listened to only if a mouse/keyboard event was received earlier.

[0093] Once a combination of trigger events is received, image processing is initiated to determine if there is a difference in images. Fig. 3B illustrates an example of a cross-correlation formula. However, more generally the cross-correlation performed between screenshot images (or sections thereof) other well-known cross-correlation equations and optimizations thereof known in the art.

[0094] Fig. 4 illustrates a process for illustrating the interaction of four systems corresponding to a Player Process 404, Web Process Manager 406, Native System Process 408,

and Guided Assistance Process 410 in accordance with an implementation. Each system has its own set of core responsibilities. The page/context change detection process may be coordinated across the four systems to identify a page/context change.

[0095] In one implementation, the Player Process 404 is started by the end user. This is the entry point to start the DAP platform on the user's machine (e.g., on a desktop computer operating a target client application for which DAP assistance is to be provided).

[0096] In one implementation, a Web Process Manager 406 holds multiple responsibilities. In one implementation, it retrieves configuration details for an enterprise. In one implementation, it starts the Native System process 408. In one implementation, it forms the layer to overlay the DAP product's widgets on a target client application. In one implementation, it helps in communicating between the Native System process 408 and the Guided Assistance process 410.

[0097] In one implementation, a Native System Process 408 is responsible for interacting with target applications and listening to OS events triggered by the application. In one implementation, it identifies if the application's context has changed or not, deciding further whether to trigger segmentation or not.

[0098] In one implementation, the Guided Assistance Process 410 receives the message to trigger segmentation and performs the refresh.

[0099] In one implementation, as illustrated by arrow 421, the Player Process 404 loads any stored local information and starts the Web process Manager 406.

[0100] In one implementation, as illustrated by in arrow 422, the Web Process Manager 406 retrieves the Enterprise configuration details for the end user's account.

[0101] In one implementation, as illustrated by arrow 423, the Web Process manager 406 starts the Native System Process 408.

[0102] In one implementation, once the Native System Process 408 is started in arrow 423, it registers in arrow 424-1, the hooks for global events, which includes Foreground, Keyboard and Mouse events.

[0103] In one implementation, as illustrated by arrow 424-2, as soon as a target application comes into focus, the Native System Process 408 sends a message to the Web Process Manager 406 to alert it that the target app is in focus.

[0104] The Web process manager 406 listens to this message and creates an IFrame over the target application to overlay widgets as indicated by arrows 425-1 and 425-2, which results in sending the target application's related configurations to the Native System Process 408.

[0105] In one implementation, the Native System Process 408 receives the target application's related configurations, as illustrated by arrow 426 it registers hooks for Reorder and Focus events that are application specific.

[0106] In one implementation, as illustrated by As illustrated by arrow 427, the Native System Process 408 loads the image configurations. The received configurations also hold the values that are required to perform Image processing (other than other configuration details), which are loaded on the Native System process 408. Some examples of parameters that are associated with Image processing include the following:

[0107] 1) Coefficient Threshold : Helps to set an upper threshold for a cross-correlation threshold value to identify context change. An example of a default value is 0.80, although other values that are higher or lower could be selected, such as 0.50, 0.60, 0.70, 0.90, or 1.0.

[0108] 2) Image Evaluation Retries : Identifies the number retries to be performed over an application for identification of a context change. This may vary depending on the type of application and depending on application's performance. An example for default retries is 6, although other number of retries could be selected such as 3, 4, or 5.

[0109] 3) Enabled Segmentation Debug Mode: This is useful for engineers in debugging purposes. It allows saving the images on the machine, to identify and debug the process. In one implementation, by default, it will be disabled.

[0110] In some implementations, users have the flexibility to configure the values of the coefficient thresholds for identifying context change the number of retries, etc.

[0111] Once the associated image configuration is loaded, the process builds the initial cache image, which generates an initial image to compare with. This is necessary for the very first comparison to take place. In one implementation, as soon as the target application is in focus, a first screenshot is taken and it becomes a cache image.

[0112] In one implementation, since the events are already hooked, listeners are used to identify when to process segmentation triggers 428-1. As illustrated in arrow 428-2, steps to initiate image process to identify if a page is changed are initiated. If the correlation coefficient is less than the threshold value, the context is known to be changed. If not, the process performs retries, until they are exhausted, assuming the page is taking time to load. If all the retries are exhausted, in one implementation the process assumes that page has not changed. Once a process receives a page that has changed, further consecutive screenshots may be compared to identify if the page is stable. In one implementation, there is also a timeout value (e.g., 5 minutes but more generally a configurable time period such as 1 minute, 2 minutes etc.). If the timeout value is exceeded, a determination is made that the application is in a hung state, and evaluation is

stopped (to prevent and not go into infinite computations). In one implementation if the cross-correlation coefficient value is greater than a threshold value, such as 0.95, the process considers a page to be stable. However, more generally other threshold cross-correlation coefficient values could be selected that would be consistent with the application being stable enough that there is little or no additional loading of content on the page.

[0113] Once the page is stable, this aids in confirming that the context has changed. If a page change is identified, then as illustrated by arrow 429, the Native System Process 408 sends a message to the Web Process Manager 406 to trigger segmentation (e.g., providing guidance to an end-user, may include providing guidance based on user preferences or user needs). In one implementation, as illustrated by arrow 430, the Guided Assistance Process 410 will perform a refresh to perform segmentation (e.g., provide DAP guidance to a user based on any applicable user preferences).

[0114] Figs. 5A and 5B illustrate another example of a method in accordance with an implementation. Referring to Fig. 5A, the process in block 502 listens for Mouse/Keyboard Events. In block 504, the process checks for mouse/keyboard events. If no Mouse/Keyboard events are detected, the process moves to block 544 and does nothing. However, if a Mouse/Keyboard event is received, the process in block 406 keeps track of Reorder/Focus events.

[0115] In one implementation, if a Mouse/Keyboard event is detected but no reorder/focus events are further detected, the process moves to block 544 and does nothing. However, if a Mouse/Keyboard event is detected and a reorder/focus event is detected, the trigger condition for a combination of events is satisfied, and the process then moves on to perform image processing steps that are more computationally expensive.

[0116] In block 510, a screenshot is captured of the foreground window. In block 512, resizing of the image is performed. There is a possibility that the previous and current screenshot differ in size. So, to make sure the process does not process the wrong set of data, the current image is resized, if necessary, based on the previous image.

[0117] In block 514, the image is converted to grayscale, which reduces the number of variables to check in making an image comparison, which can also be described as reducing the number of dimensions for analysis. Reducing the number of variables improves speed, which aids in avoiding the problem of stale data while forming a comparison between images. Therefore, in one implementation an image is converted into a grayscale image and then further processing is performed on it.

[0118] In block 518, images are compared to identify differences between images. For the very first comparison, a screenshot is taken of the screen that appears first for the foreground window. Therefore, now the process has the data to compare. Both the images are in the form of grayscale, and a comparison is performed for the data for each pixel to get a cross-correlation coefficient.

[0119] If the cross-correlational coefficient is not less than a defined threshold, the process in block 522 determines if there are retries left to perform the same steps again. For example, a pre-selected maximum number of retries may be configured.

[0120] If all the retries are exhausted, the process will end and do nothing, moving on to block 544.

[0121] In block 520, if the cross-correlational coefficient value is less than a defined upper threshold, the process proceeds further to block 550.

[0122] The process continues in Fig. 5B at line A-A.

[0123] In one implementation, if the difference between two images is found, the process waits for the page to become stable (e.g., no longer loading elements). In one implementation, to figure out stability, in block 550, the process keeps on taking screenshots and compares them with the consecutive ones, to get a high correlation coefficient (e.g., greater than 0.95).

[0124] In decision block 552, if the cross-correlation coefficient is greater than 0.95, the page is considered to be stable. In block 554, the cache image and grayscale matrix are updated.

[0125] If not, the process performs the same steps until the process receives a timeout in decision block 558.

[0126] If the process reaches a timeout limit, this means that the application is hung and something is wrong. Therefore, the process ends in block 560.

[0127] After a page is stable, a determination can be made that the context has changed. The previous image, which is cache image, is updated with the current one. This will aid in subsequent comparisons to identify context change. Since the page context change is identified, segmentation is triggered in block 556, where the segmentation is part of the process of providing DAP guidance to a user.

Use of Subsections

[0128] As previously discussed, in one implementation the screenshot is divided into subsections, such as quadrants. Cross-correlation comparisons may be performed on specific subsections of the screenshot. This implementation is particularly beneficial for the case that a large portion of a screenshot is blank. In performing an image difference algorithm, the screenshot of the application is taken and compared with the cached image which was taken at the trigger point of the algorithm as described in the previous section. The image comparison happens in the following ways:

[0129] 1) Full image comparison in which the process compares the full screenshot of the application with the previous and the next screenshots.

[0130] 2) Sectional image comparison in which the image is divided into subsections and then each subsection is compared with the exact same subsection in the next image. For example, if the screenshot is divided into four quadrants Q1, Q2, Q3, and Q4, then quadrant Q1 in a first image is compared with quadrant Q1 in a second image, and so on. Dividing a screenshot into subsections may be necessary to achieve a high degree of accuracy in the page change identification when the application is maximized. This is because maximizing an application increases the likelihood a page has less content to show and that most of the area is blank. In theory each and every subsection could be evaluated. However, in some implementations, the sectional image comparison includes performing a pre-processing operation to identify regions of the full image that are blank or otherwise irrelevant. Alternatively, a pre-processing operation could identify potential opportunities to improve the accuracy of cross-correlation computations as a pre-condition to dividing an image into sectioning. In any case, once a single subsection is demonstrated by cross-correlation computation to indicate a page change, the other subsections do not need to be evaluated.

[0131] 3) Additionally, another possible option would be to perform a full image comparison and then do a sectional image comparison for greater accuracy. For example, if the full image comparison generated a cross-correlation value that was marginal, in regard to having a cross correlation value indicative of a context change, an additional sectional image comparison could be done. That is, a sectional image comparison could be performed when one or more factors suggests a risk of a false negative result of the full image cross-correlation computation.

[0132] When a page has a large percentage of blank regions this increases the likelihood of false negatives when a cross-correlation function is computing. Dividing the image into subsections improves accuracy for the cross-correlation computations, particularly under some scenarios, such as when the application is maximized or if the application itself includes pages that are largely blank. This way, the process can accurately identify changes even if there are too many white spaced regions in the page as a whole.

[0133] Note that a page change has to be only detected in one subsection (e.g., one quarter).

[0134] Some aspects of cross-correlation and subsections will now be illustrated with some example UI pages. Fig. 6 illustrates a first example of an application context, in which a screenshot 610 of a UI page for Bluetooth & devices.

[0135] Fig. 7 illustrates a second example of an application context, in which a screenshot 710 is of a UI page.

[0136] Fig. 8 illustrates a third example of an application context, in which a screenshot 810 is of a UI page.

[0137] Fig. 9 is a visualization of differences 905 between the pages in the first context of Fig. 6 and the second context of Fig. 7 to indicate visually aspects of performing a cross-correlation comparison. As the left-side of each page is nearly identical, there is almost no difference in that section of the screenshot. However, there is still substantial change between Fig. 6 and Fig. 7.

[0138] Fig. 10 is a visualization of differences 1002 between the pages in the first context of Fig. 6 and the third context of Fig. 8. Fig. 6 and Fig. 8 are nearly identical. As such, only a small portion of the screen differs. One aspect illustrated by Fig. 10 is that the cross-correlation

threshold may be selected to identify differences in pages that approximate the smallest difference that the typical human user would be aware of. As seen in the figures, the blank area is in the majority. While comparing images, the cross-correlation output can give a false negative when there is a large fraction of blank area in an image. So, dividing the image into multiple quadrants isolates the difference computation. Once a page change has been identified in one subsection (e.g., one quarter), there is no need to perform a comparison analysis on the remaining subsections.

[0139] Fig. 11 illustrates an example of screenshot of a UI page in which the blank area is in the majority. This has the downside that the cross-correlation calculation performed on the entire page is less reliable, potentially giving a false negative. There are various ways that it could be determined that much of the page is blank. For example, this could be detected by performing a preliminary image analysis to identify blank sections of a page. Alternatively, in some cases, other information may be available about a page to indicate that it is a type of page in which much of the page is blank.

[0140] Fig. 12 illustrates a screenshot of the page of Fig. 11 but divided into smaller subsections. In this example, the screen image is divided into quarter sections (Q1, Q2, Q3, and Q4). This permits the difference computation to be isolated to the subsection that is not blank (Q1) resulting in a more accurate difference computation. As previously discussed, while dividing into quadrants is one option, more generally the number and arrangement of subsections could include at least two subsections, as the objective of segmentation is to identify page differences with greater accuracy.

[0141] **Memoization**

[0142] The capturing of images of UI pages is a resource heavy process. It was observed in one test by the inventors that consumption of CPU utilization increases during the evaluation process up to 50%. To reduce CPU usage a memoization process was developed to cache the previous evaluated pixels in the memory.

[0143] In one implementation, gray-scale pixels at full and quadrant level are cached, as they are the basic origin for evaluation purposes. (Note that if other sub-sectioning schemes besides quadrants are used, pixels may be cached at the level of the sub-sectioning (e.g., one half of the pixels of a page for a page divided into two subsections)).

[0144] In one implementation, as soon as the target active window appears, the first screenshot is taken. Then the process of building a cache begins. Pixels are converted into grayscale for full screen images and for quadrants and they are cached. The screen's width and height may also be cached as a variable. When the next screenshot is taken, the process does not have to calculate the previous image's pixels, because the cache can be used.

[0145] When a page/context change is identified as having occurred, the cache is updated with the then pixel values. This means that for the the next comparison the process does not have to re-evaluate the pixel values.

[0146] Note that if the active window has been resized, the previously built cache won't be the relevant candidate to be taken up for further computations. To verify if the window has been resized, the new active window's width and height are compared with that of the cached one. For the case that the active window was resized, the process updates the cache with resized window image's pixels.

[0147] Empirically this resulted in a significant decrease in the number of computations done, as well as supporting the offloading resources.

[0148] As previously discussed, the page/context change detection engine 120 includes a screen pixel caching unit 120. The screen pixel caching unit 120 may be designed to build an initial cache for processing the computations. It may also be designed to identify if the cache needs to be re-built, and whether the cache needs to be updated. It may be designed to get pixels in grayscale and cache them. It may also be designed to identify that a window has been resized. In one implementation, the screen pixel caching unit is designed to cache both full and subsection-level pixels (e.g., quadrant level pixels)

[0149] The page change identification engine 120 may be implemented as software instructions. It may, for example, be implemented to operate with other software on a user's desktop. Referring to Fig. 13A, the software instructions may be stored on a user's computer, which may in turn have conventional hardware components such as a memory 1310, data store 1320, output device 1314 (e.g., a display screen), communication bus 1302, an input device 1312, a processor 1308, and a communication unit 1304 to communicate with a computer network, such as a LAN, WAN, the internet, etc.

[0150] In some instances, various implementations may be presented herein in terms of algorithms and symbolic representations of operations on data bits within a computer memory. An algorithm is here, and generally, conceived to be a self-consistent set of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like New, Upload, Sync, and Open in One Drive.

[0151] To ease description, some elements of the system and/or the methods are referred to using the labels first, second, third, etc. These labels are intended to help to distinguish the elements but do not necessarily imply any particular order or ranking unless indicated otherwise.

[0152] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout this disclosure, discussions utilizing terms including "processing," "computing," "calculating," "determining," "displaying," or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0153] Various implementations described herein may relate to an apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, including, but is not limited to, any type of disk including floppy disks, optical disks, CD ROMs, and magnetic disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, flash memories including USB keys with non-volatile memory or any type of media suitable for storing electronic instructions, each coupled to a computer system bus.

[0154] The technology described herein can take the form of an entirely hardware implementation, an entirely software implementation, or implementations containing both hardware and software elements. For instance, the technology may be implemented in software, which includes, but is not limited to, firmware, resident software, microcode, etc. Furthermore, the technology can take the form of a computer program object accessible from a computer-usable or computer-readable medium providing program code for use by or in connection with a computer or any instruction execution system. For the purposes of this description, a computer-usable or computer readable medium can be any non-transitory storage apparatus that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device.

[0155] A data processing system suitable for storing and/or executing program code may include at least one processor coupled directly or indirectly to memory elements through a system bus. The memory elements can include local memory employed during actual execution of the program code, bulk storage, and cache memories that provide temporary storage of at least some program code in order to reduce the number of times code must be retrieved from bulk storage during execution. Input or I/O devices (including, but not limited to, keyboards, displays, pointing devices, etc.) can be coupled to the system either directly or through intervening I/O controllers.

[0156] Network adapters may also be coupled to the system to enable the data processing system to become coupled to other data processing systems, storage devices, remote printers, etc., through intervening private and/or public networks. Wireless (e.g., Wi-Fi™) transceivers, Ethernet adapters, and Modems, are just a few examples of network adapters. The private and public networks may have any number of configurations and/or topologies. Data may be

transmitted between these devices via the networks using a variety of different communication protocols including, for example, various Internet layer, transport layer, or application layer protocols. For example, data may be transmitted via the networks using transmission control protocol / Internet protocol (TCP/IP), user datagram protocol (UDP), transmission control protocol (TCP), hypertext transfer protocol (HTTP), secure hypertext transfer protocol (HTTPS), dynamic adaptive streaming over HTTP (DASH), real-time streaming protocol (RTSP), real-time transport protocol (RTP) and the real-time transport control protocol (RTCP), voice over Internet protocol (VOIP), file transfer protocol (FTP), WebSocket (WS), wireless access protocol (WAP), various messaging protocols (SMS, MMS, XMS, IMAP, SMTP, POP, WebDAV, etc.), or other known protocols.

[0157] Finally, the structure, algorithms, and/or interfaces presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method blocks. The required structure for a variety of these systems will appear from the description above. In addition, the specification is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the specification as described herein.

[0158] The foregoing description has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the specification to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. As will be understood by those familiar with the art, the specification may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. Likewise, the

particular naming and division of the modules, routines, features, attributes, methodologies, and other aspects are not mandatory or significant, and the mechanisms that implement the specification or its features may have different names, divisions and/or formats.

[0159] Furthermore, the modules, routines, features, attributes, methodologies, and other aspects of the disclosure can be implemented as software, hardware, firmware, or any combination of the foregoing. Also, wherever a component, an example of which is a module, of the specification is implemented as software, the component can be implemented as a standalone program, as part of a larger program, as a plurality of separate programs, as a statically or dynamically linked library, as a kernel loadable module, as a device driver, and/or in every and any other way known now or in the future. Additionally, the disclosure is in no way limited to implementation in any specific programming language, or for any specific operating system or environment.

WHAT IS CLAIMED IS:

1. A computer-implemented method of detecting page changes of a graphical user interface (GUI) application screen, comprising:

determining if a trigger condition is satisfied for a page change, the trigger condition being based on a combination of events associated with a likely page change of an application;

subsequent to satisfying the trigger condition, capturing a sequence of at least two screenshots of the application taken at different times, dividing each screenshot into at least two subsections, and comparing analogous subsections of the at least two screenshots using a cross correlation function; and

identifying a page change in response to the cross-correlation function, for at least one subsection, having a value below a threshold value indicative of a page change.

2. The computer-implemented method of claim 1, where the at least two subsections consist of quadrants.
3. The computer-implemented method of claim 1, further comprising performing a pre-processing operation to identify blank portions of a screenshot and in response apply at least one rule for dividing screenshots into at least two subsections.
4. The computer-implemented method of claim 3, wherein the pre-processing operation identifies an order for processing subsections based at least in part on the blank portions.

5. The computer-implemented method of claim 3, wherein the pre-processing operation applies at least one rule to improve an accuracy of cross-correlation computations.
6. The computer-implementation method of claim 1, wherein dividing each screenshot into at least two subsections is selected to improve an accuracy of cross-correlation computations.
7. The computer-implemented method of claim 1, wherein the cross-correlation function is evaluated based on a reference screenshot captured in response to the trigger condition and a subsequent screenshot.
8. The computer-implemented method of claim 1, wherein the trigger condition comprises operating system events.
9. The computer-implemented method of claim 1, wherein the trigger condition comprises a combination of reorder events, focus events, mouse events, and keyboard events.
10. The computer-implemented method of claim 1, wherein the trigger condition comprises a first trigger condition comprising detecting mouse events and keyboard events followed by a second trigger condition comprising detecting reorder events and focus events.
11. The computer-implemented method of claim 1, wherein the threshold value is in a range between 0.5 to 1.0.
12. The computer-implemented method of claim 1, further comprising performing a stability test, based on screenshots of the application, to determine if the user interface is stable.

13. The computer-implemented method of claim 12, wherein the stability test is based on performing a cross-correlation check for at least two screenshots taken after detecting a change in page content.
14. The computer-implemented method of claim 1, further comprising performing at least one retry if the cross-correlation function does not have a value below a threshold value indicative of a page change.
15. The computer-implemented method of claim 1, wherein the sequence of at least two consecutive screenshots is converted into grayscale images to reduce the number of dimensions for analysis.
16. The computer-implemented method of claim 15, further comprising caching grayscale versions of at least a first screenshot in the sequence of at least two consecutive screenshots.
17. The computer-implemented method of claim 1, wherein the sequence of at least two consecutive screenshots comprises screenshots of at least two different color images, with the cross-correlation function being performed on color images.
18. The computer-implemented method of claim 1, further comprising utilizing the page change identification information for a digital adoption platform to recognize page changes for which guidance is provided by the digital adoption platform.
19. A system for identifying page changes of a graphical user interface, comprising:

a page change identification engine configured to determining if a trigger condition is satisfied for a page change, the trigger condition being based on a combination of events associated with a likely page change of an application;

subsequent to satisfying the trigger condition, capturing a sequence of at least two screenshots of the application taken at different times, dividing each screenshot into at least two sub-sections, and comparing analogous subsections of at least two screenshots using a cross correlation function; and

identifying a page change in response to the cross-correlation function having a value below a threshold value indicative of a page change.

20. The system of claim 19, wherein the at least two sub-sections consist of quadrants.

21. A computer-implemented method of detecting page changes of a graphical user interface (GUI) application screen, comprising:

determining if a trigger condition is satisfied for a page change, the trigger condition being based on a combination of events associated with a likely page change of an application;

subsequent to satisfying the trigger condition, capturing a sequence of at least two screenshots of the application taken at different times, dividing each screenshot into quadrants, and comparing analogous quadrants of the at least two screenshots using a cross correlation function;

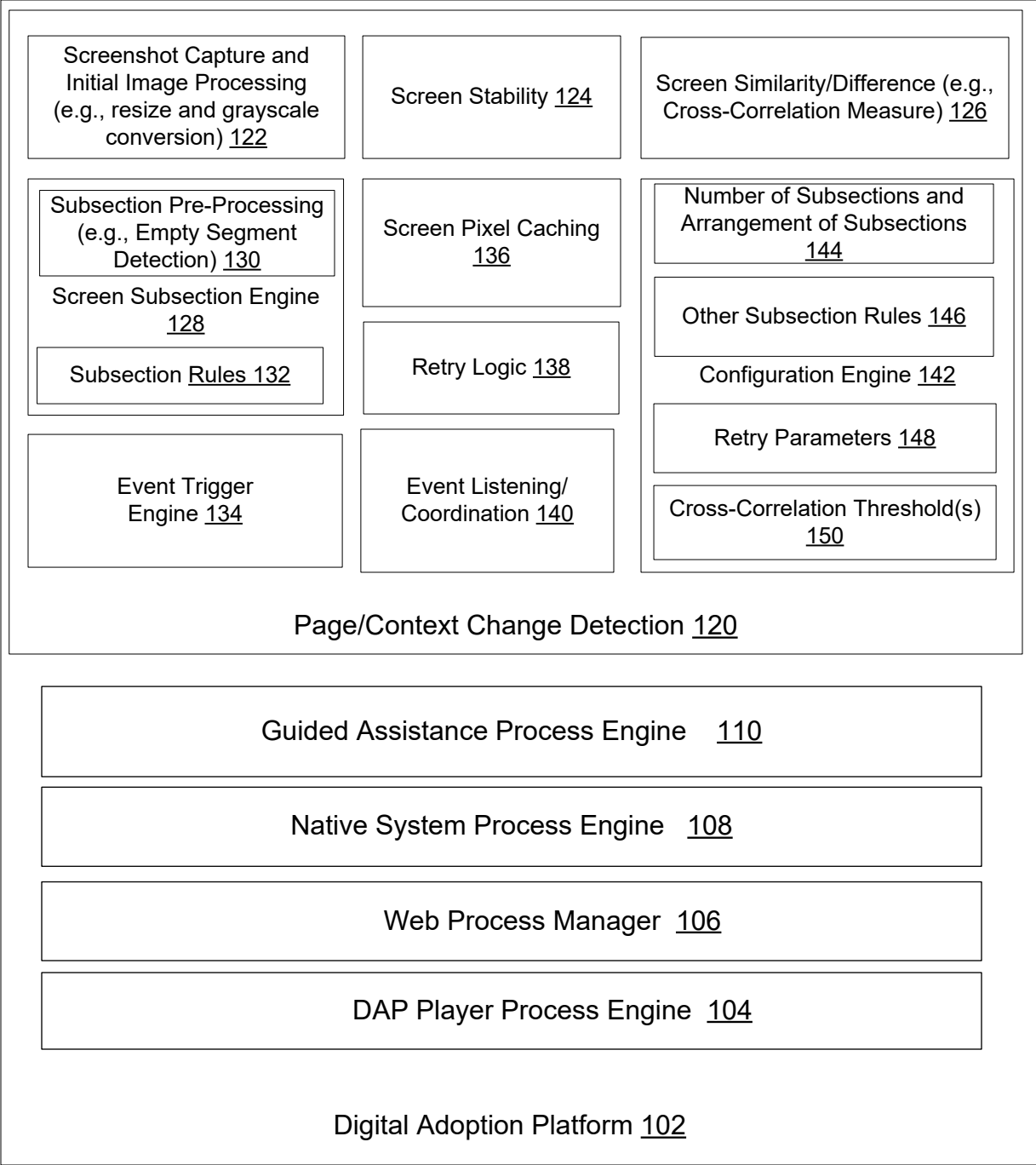
identifying a page change in response to the cross-correlation function having a value below a first threshold value indicative of a page change;

evaluating page stability using a stability score; and

in response to identifying a page change and that the page is stable, identifying a page change for a digital adoption platform.

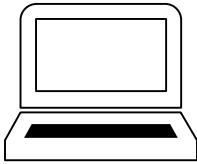
ABSTRACT

A digital adoption platform includes a page change identification technique. The page change identification technique is platform and application agnostic. Changes in screenshots of a UI of an application are compared. Differences between screenshots are compared using a cross-correlation technique. Trigger conditions are defined to reduce the computational resources required. Additional optimizations to reduce processing resources and optimize accuracy may also be included.



Target Client Application 101

Application GUI 102



100

Fig. 1

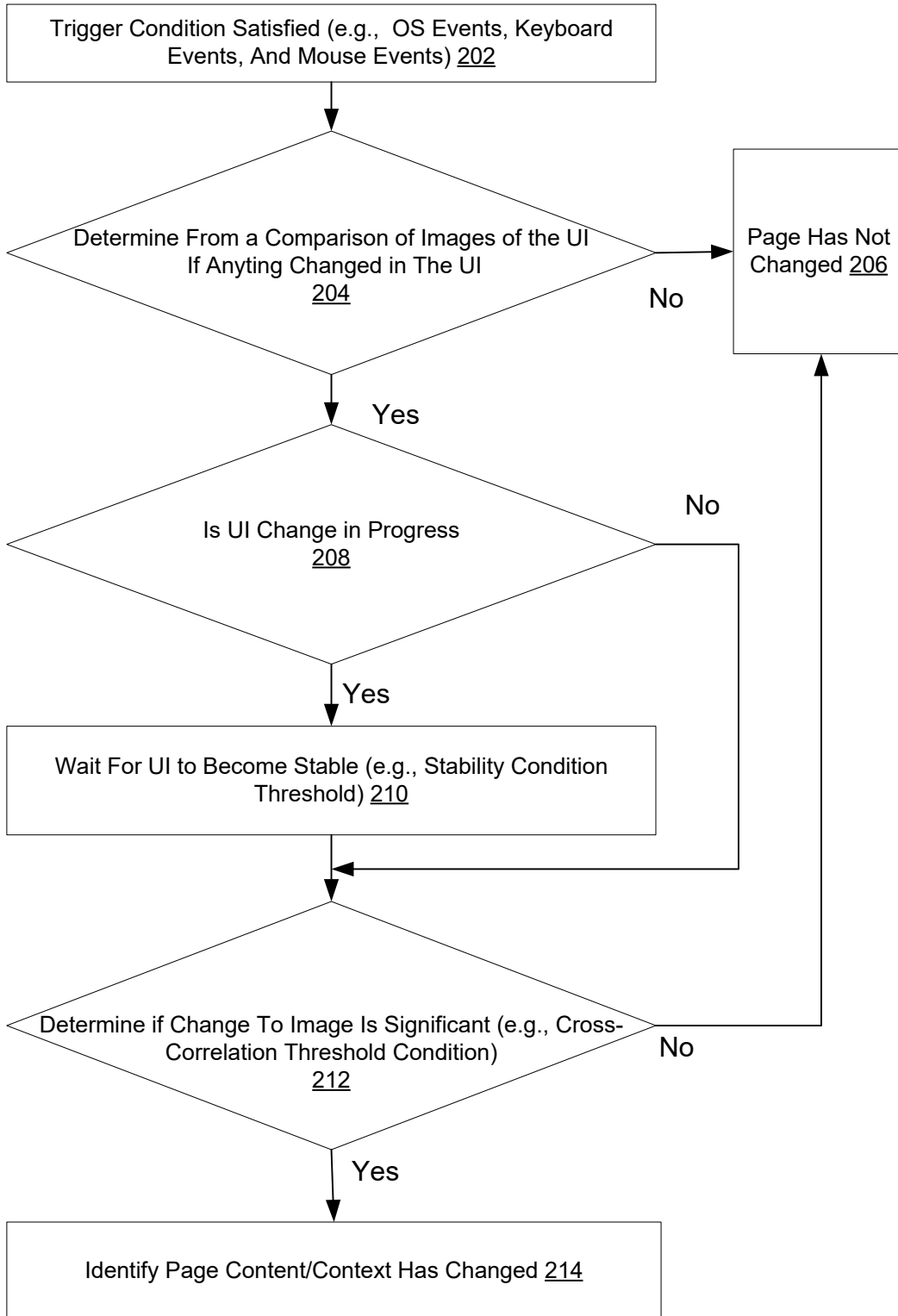


Fig. 2

Focus Event	UI Reorder Event	Keyboard Event (Tab/Esc/Enter)	Mouse Event (Mouse Up)	Process?
Yes	No	Yes	No	Yes
Yes	No	No	Yes	Yes
Yes	No	Yes	Yes	Yes
Yes	No	No	No	No
No	Yes	Yes	No	Yes
No	Yes	No	Yes	Yes
No	Yes	Yes	Yes	Yes
No	Yes	No	No	No

Fig. 3A

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient

x_i = values of the x-variable in a sample

\bar{x} = mean of the values of the x-variable

y_i = values of the y-variable in a sample

\bar{y} = mean of the values of the y-variable

Fig. 3B

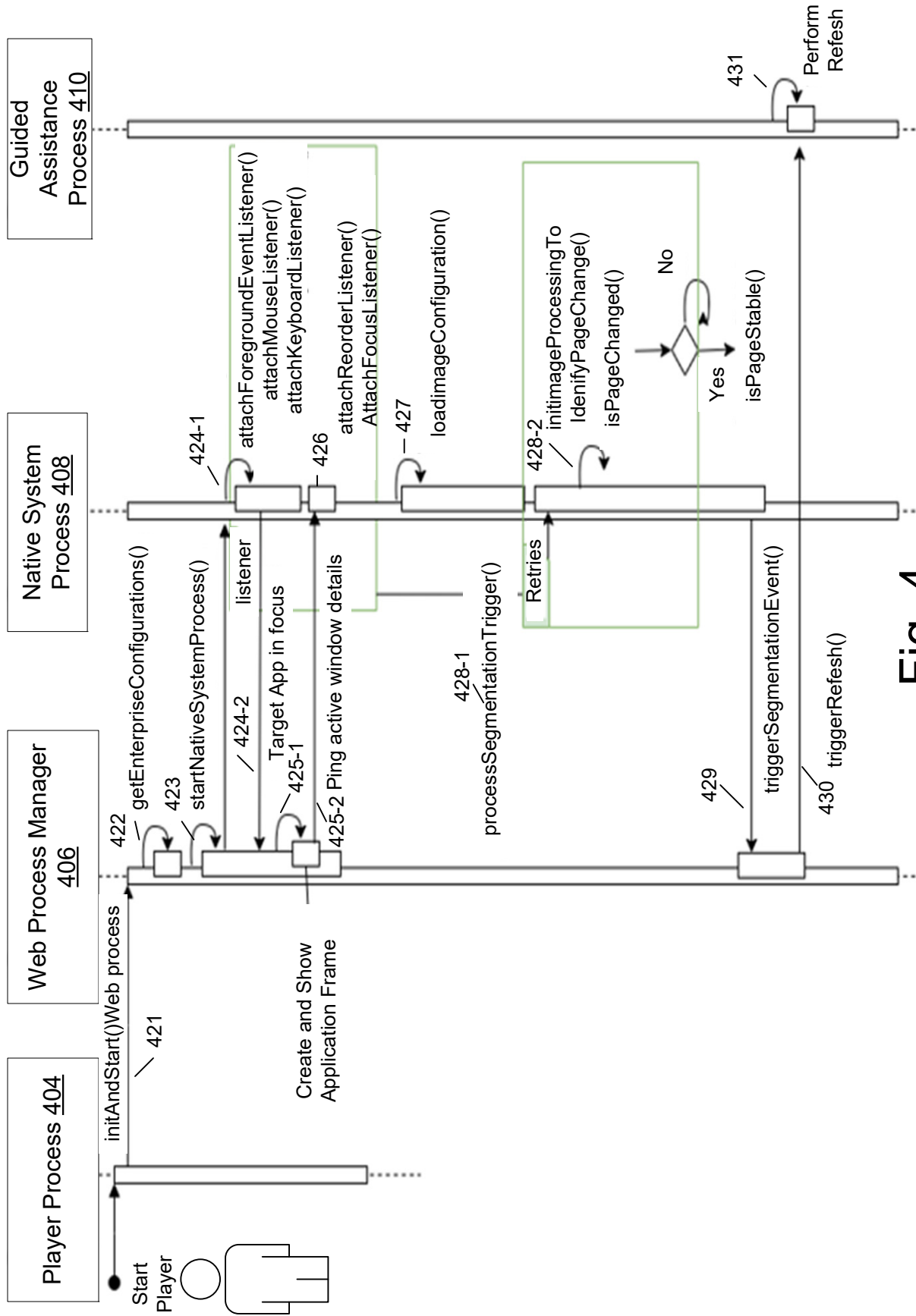


Fig. 4

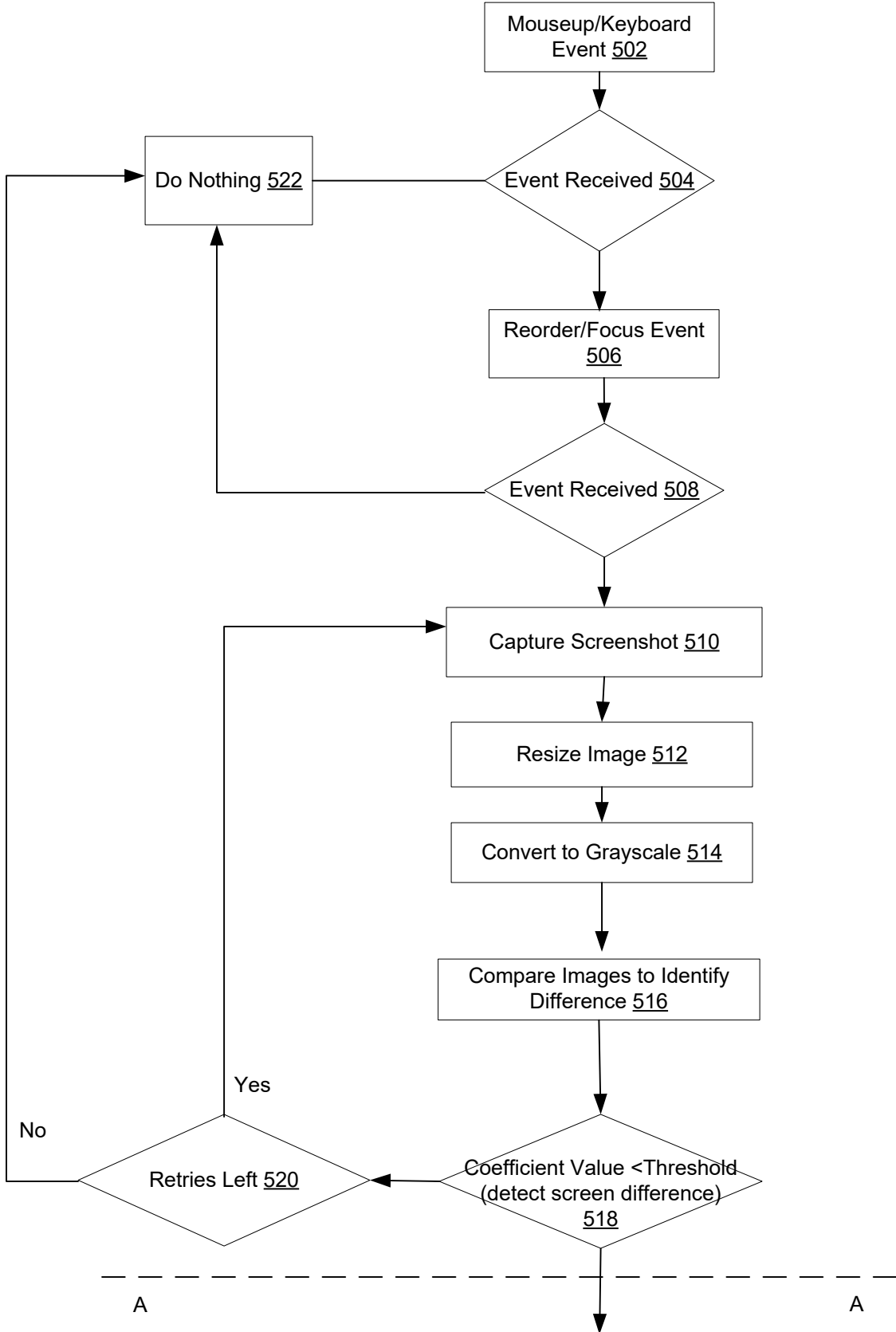


Fig. 5A

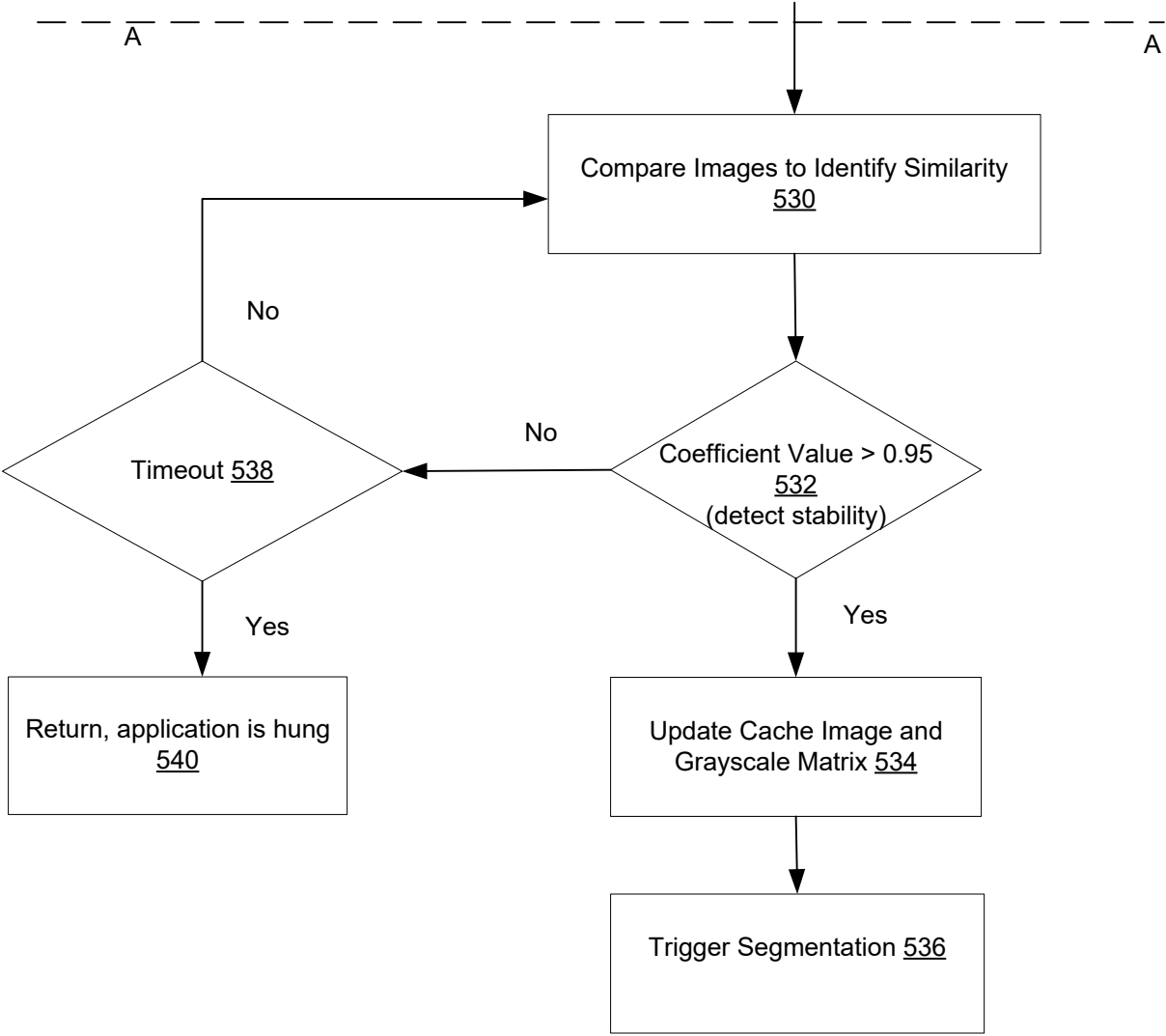
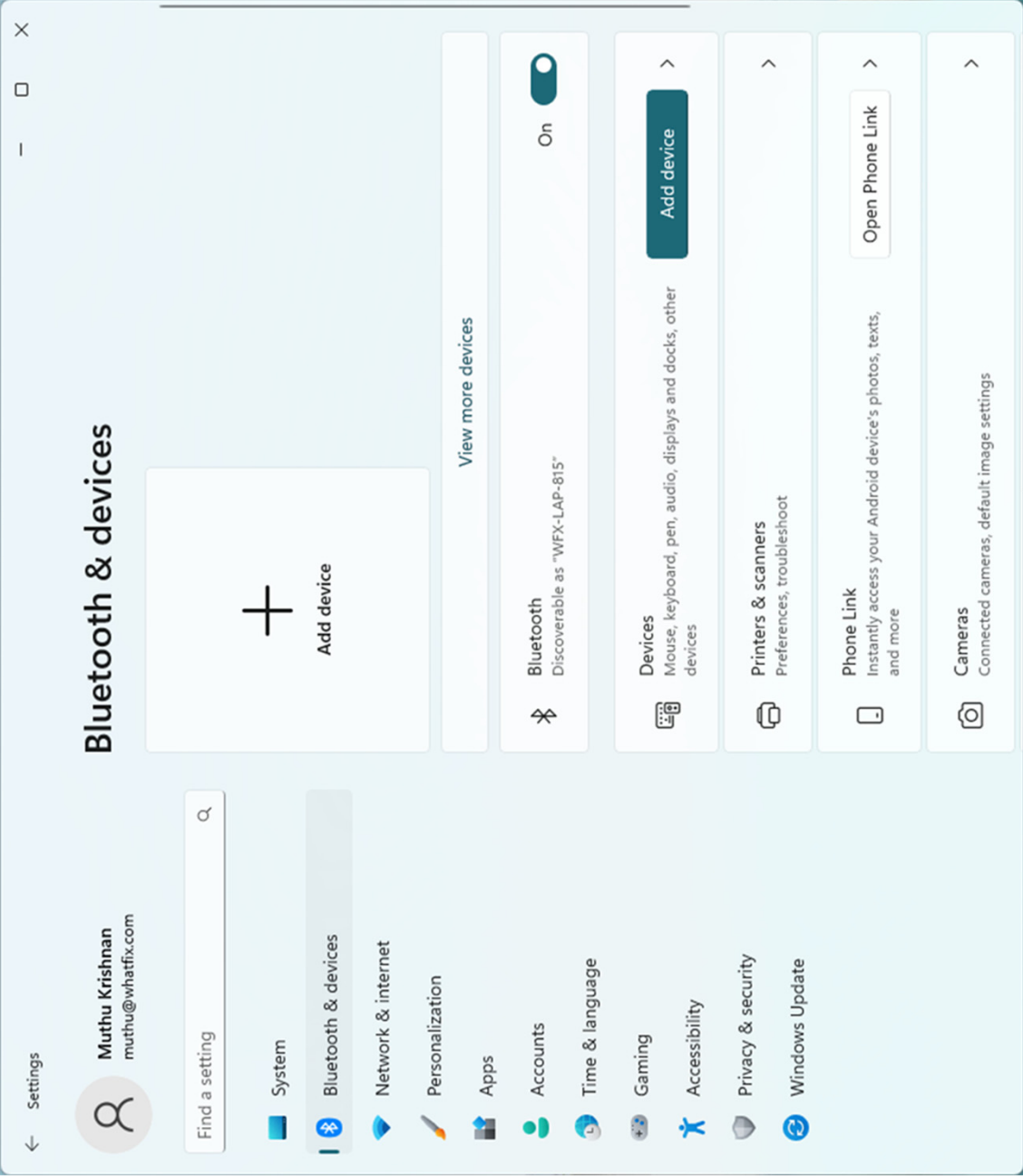
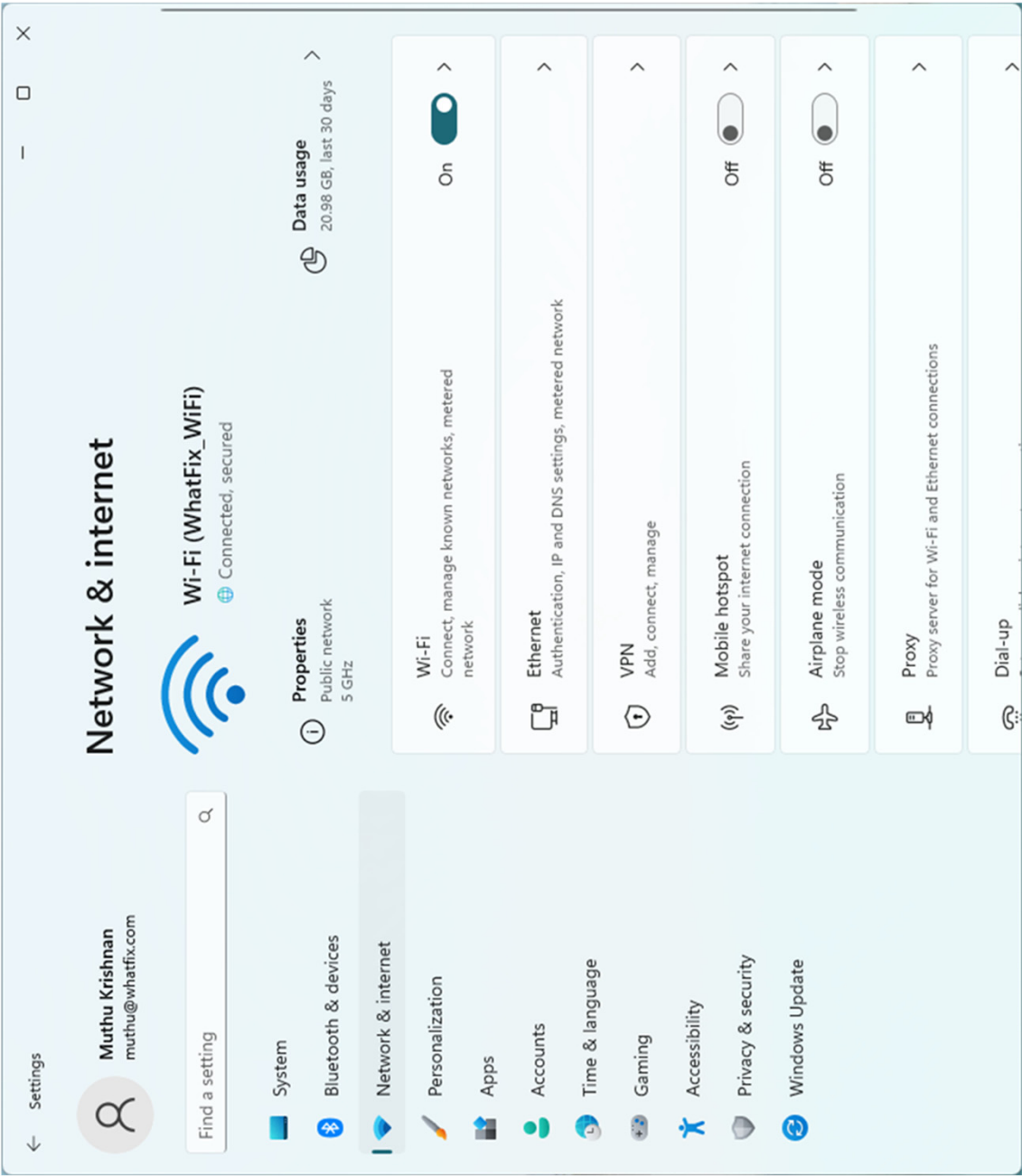


Fig. 5B



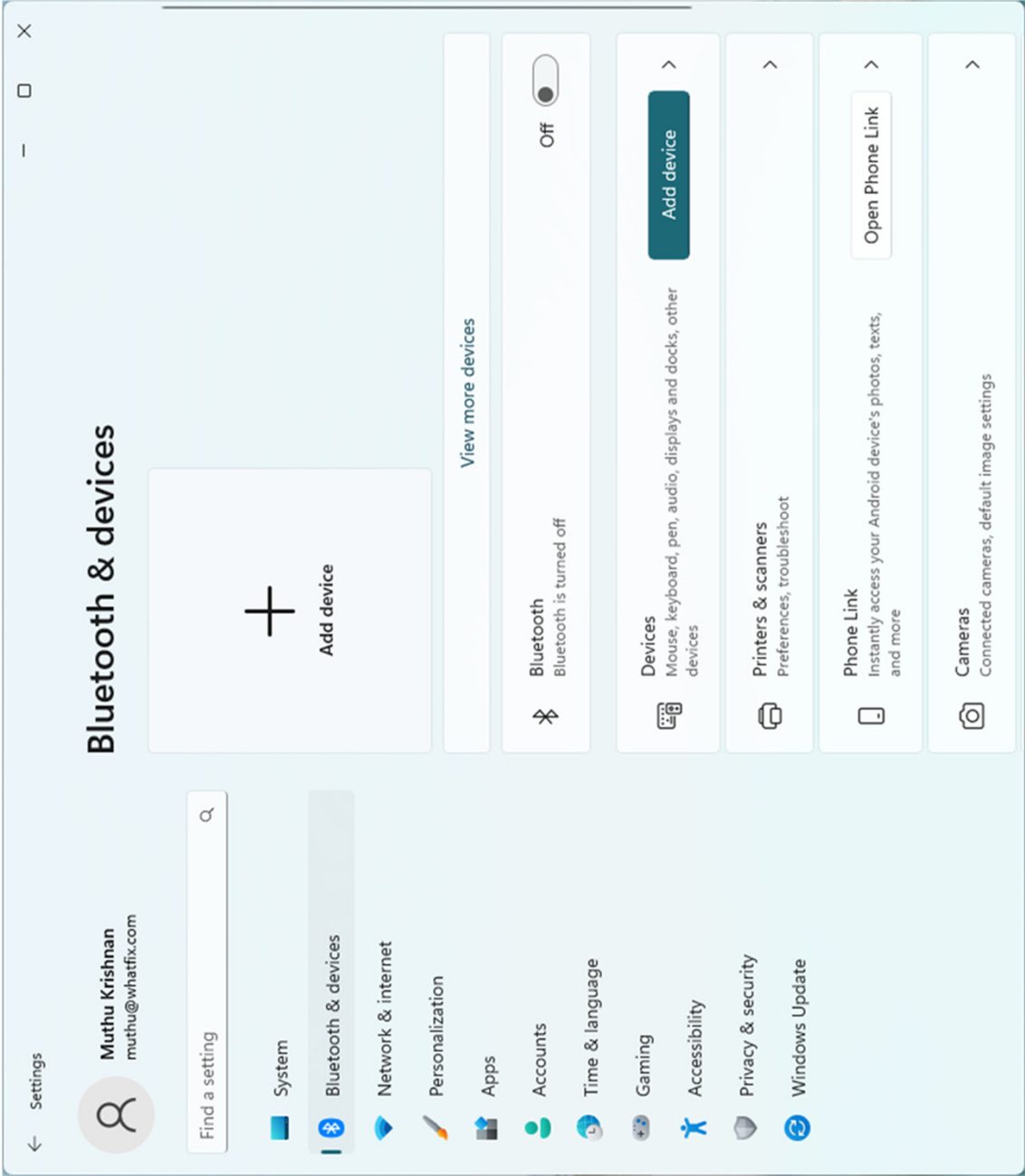
610

Fig. 6



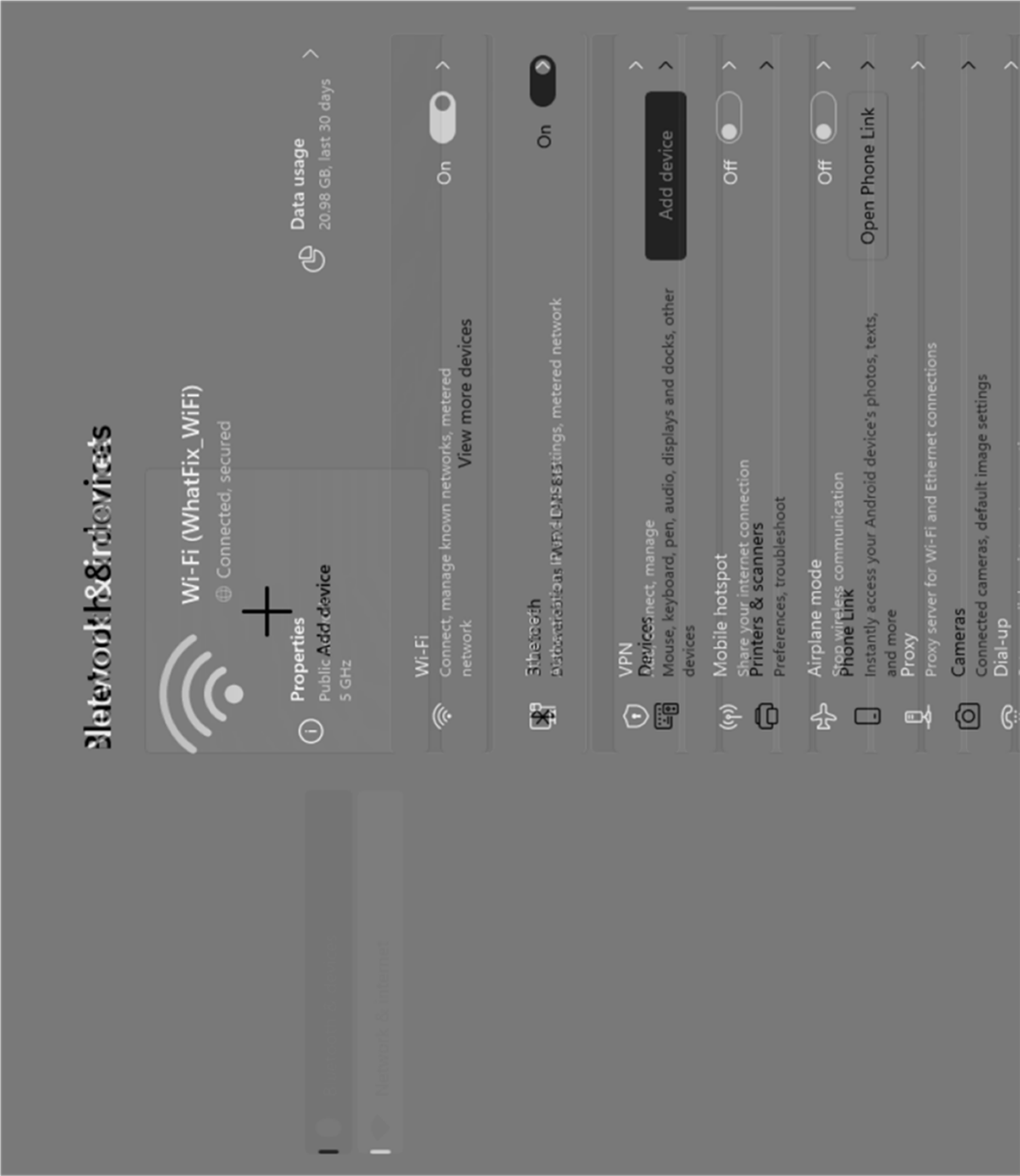
710

Fig. 7



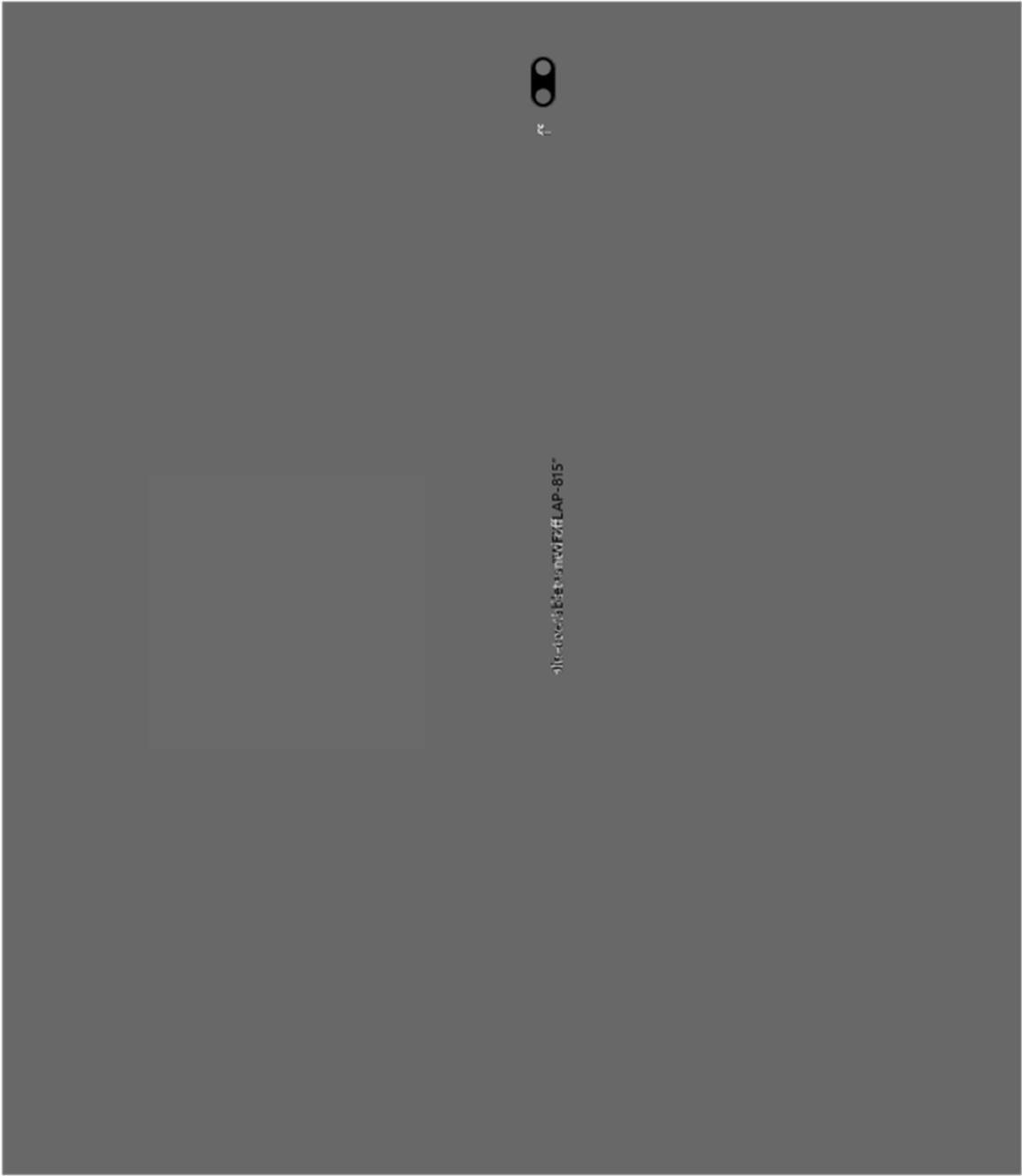
810

Fig. 8



902

Fig. 9



1002

Fig. 10

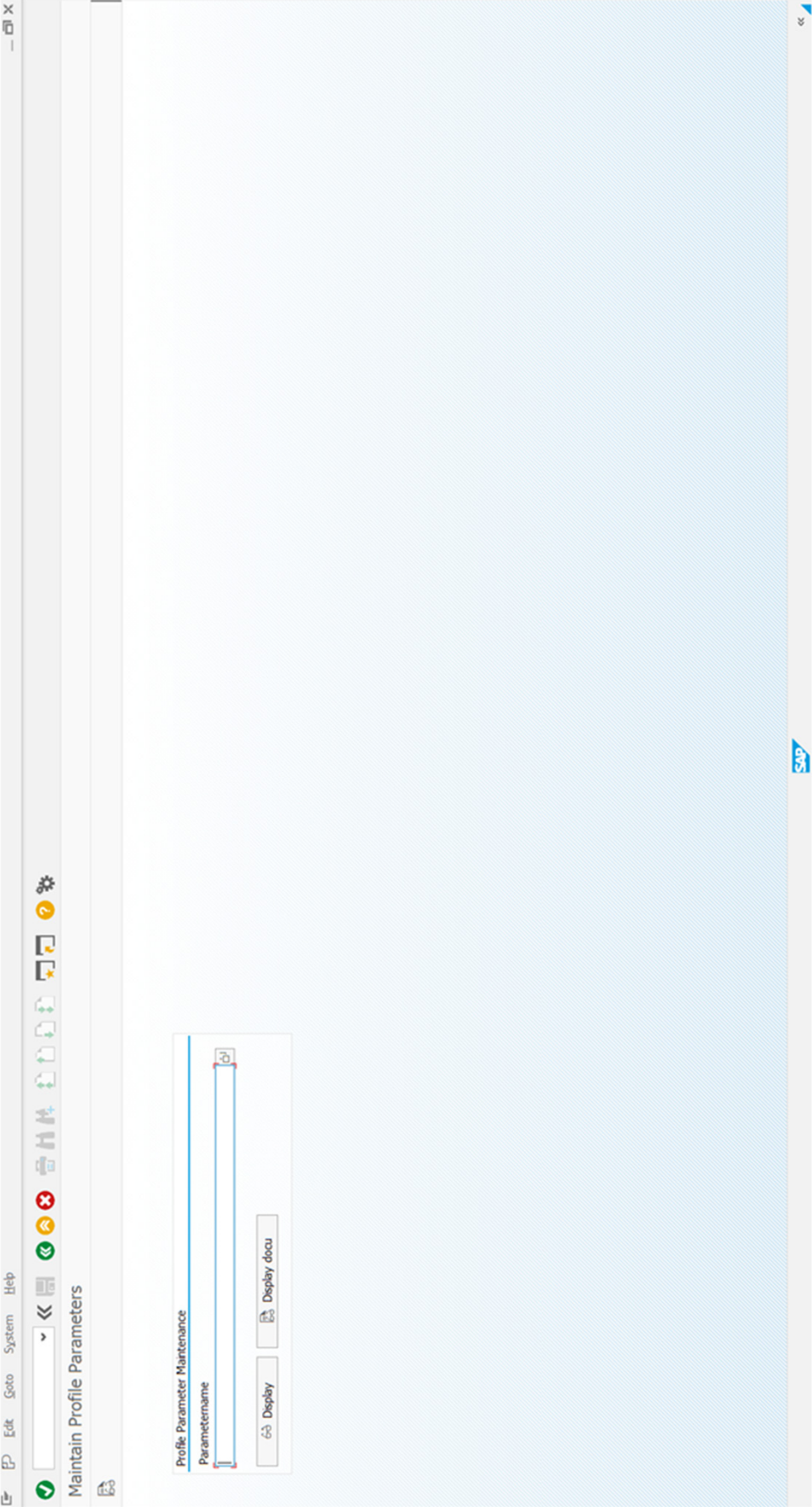


Fig. 11

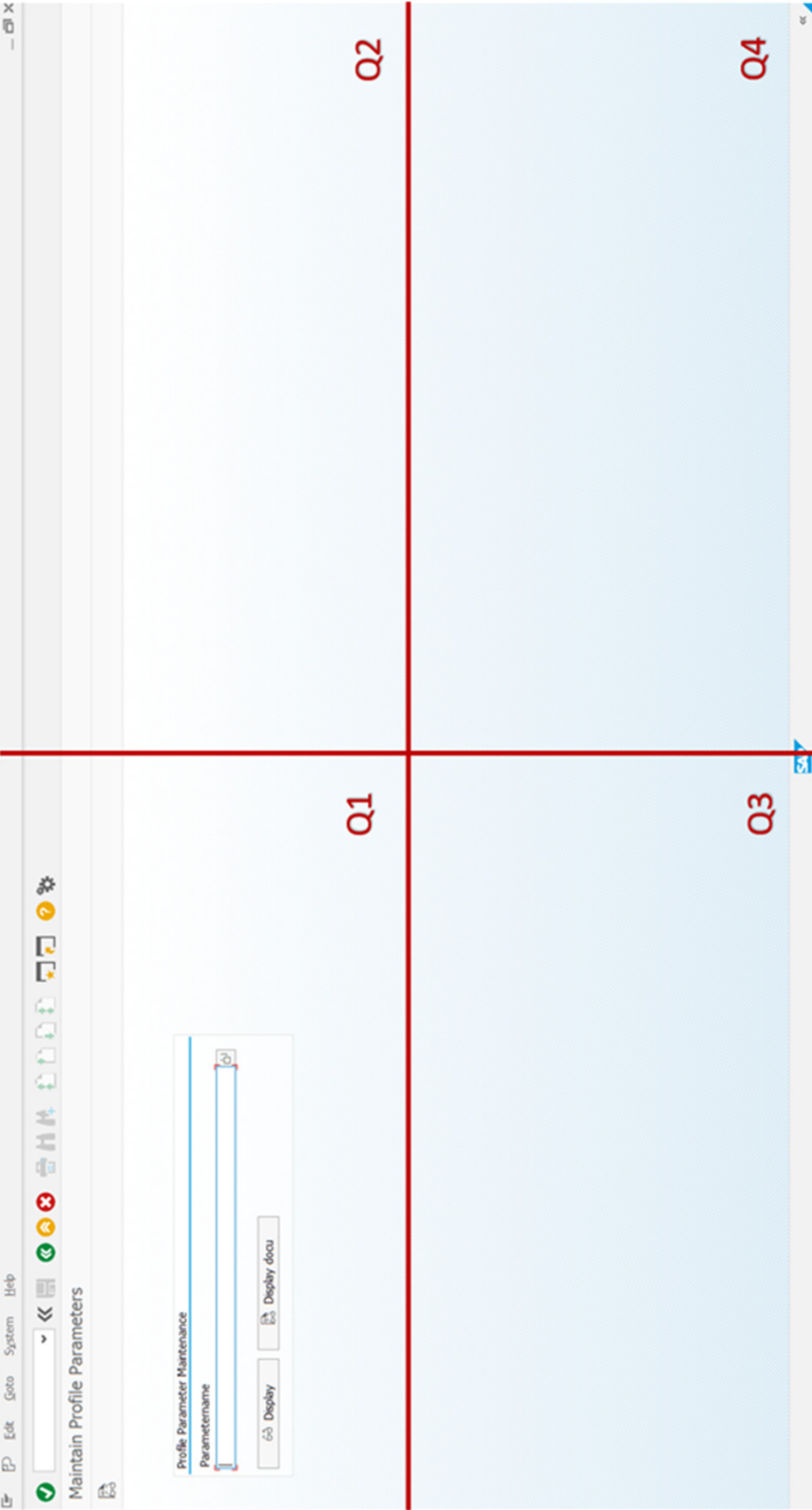


Fig. 12A

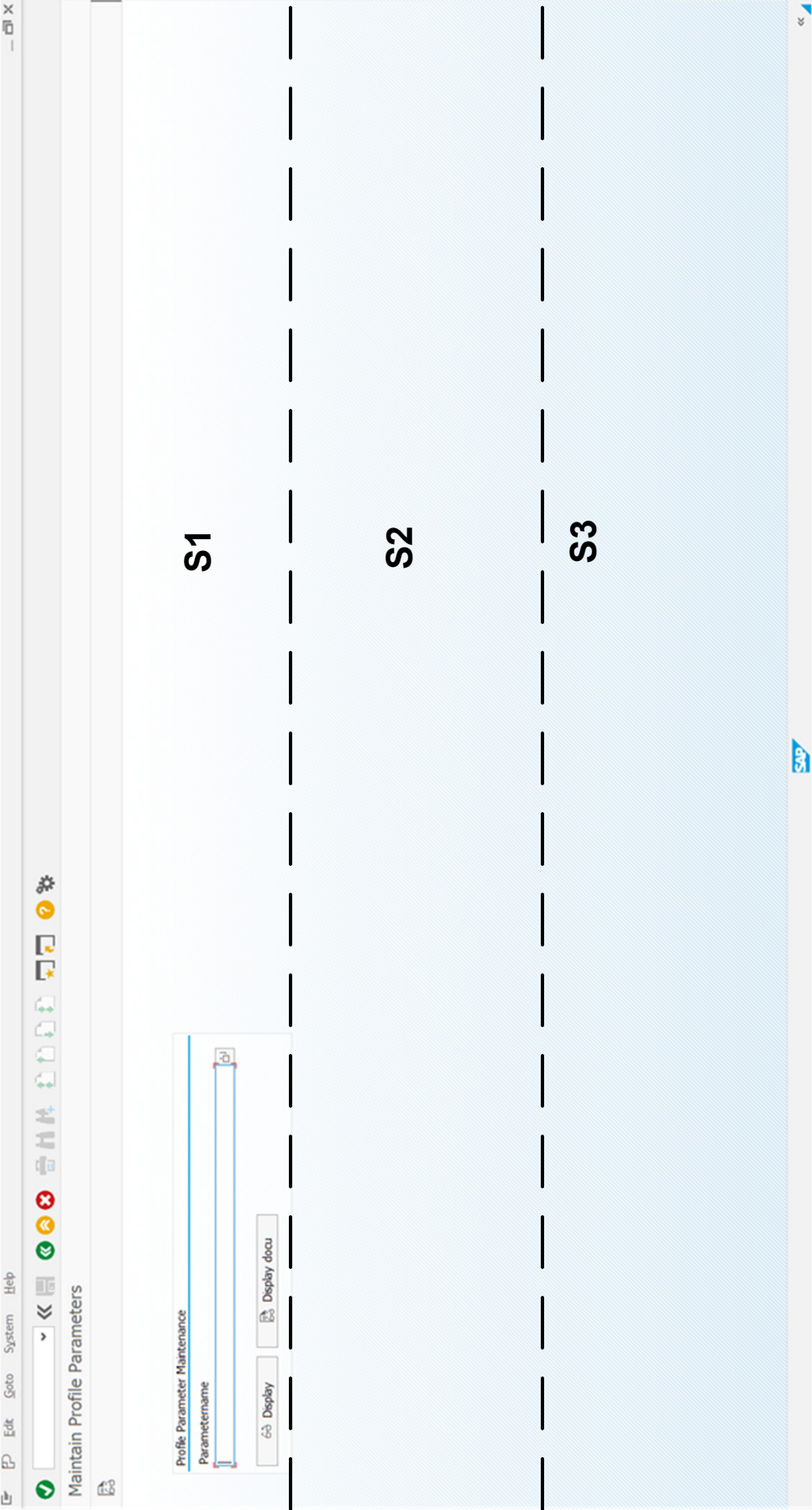


Fig. 12B

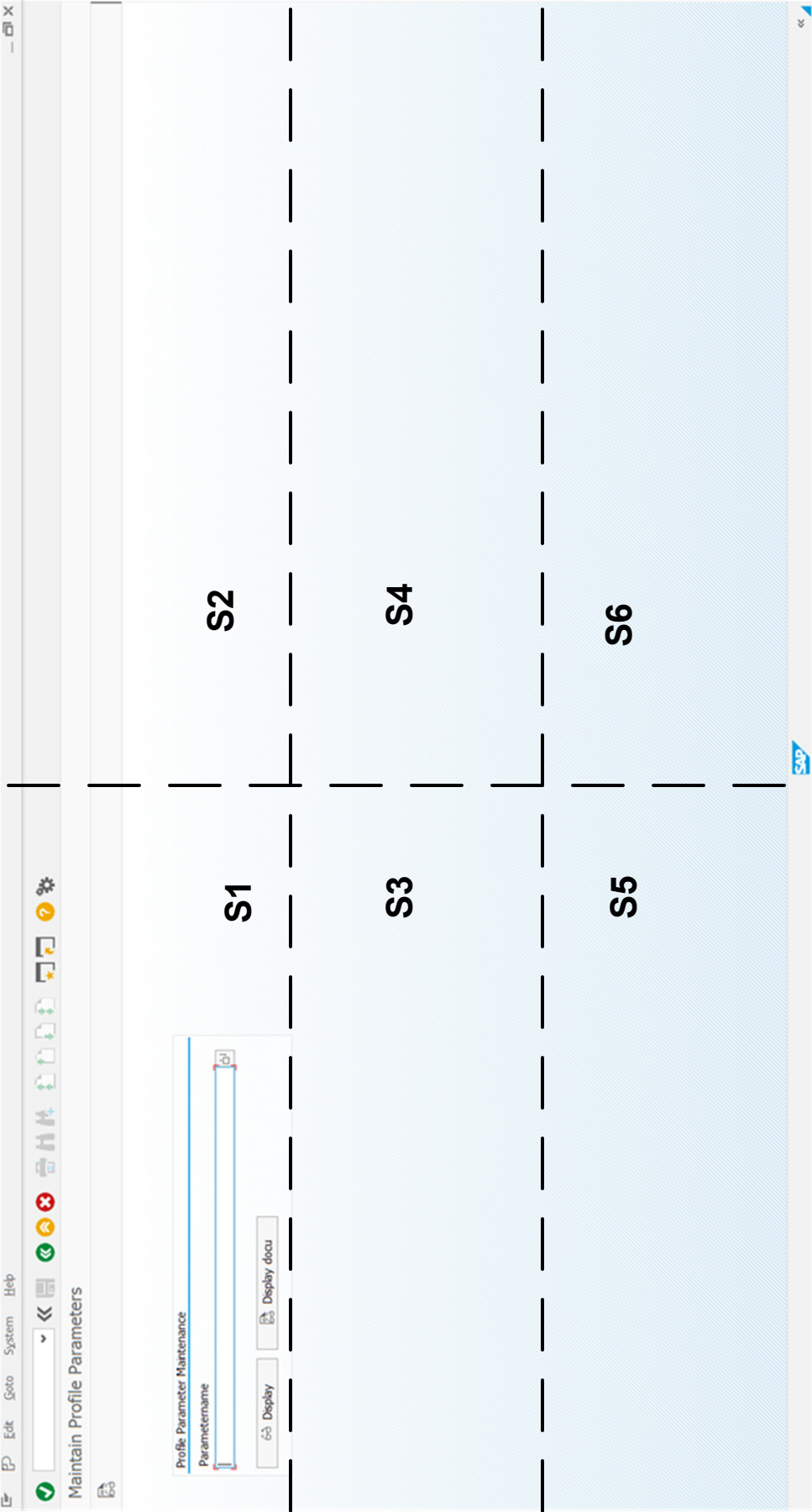


Fig. 12C

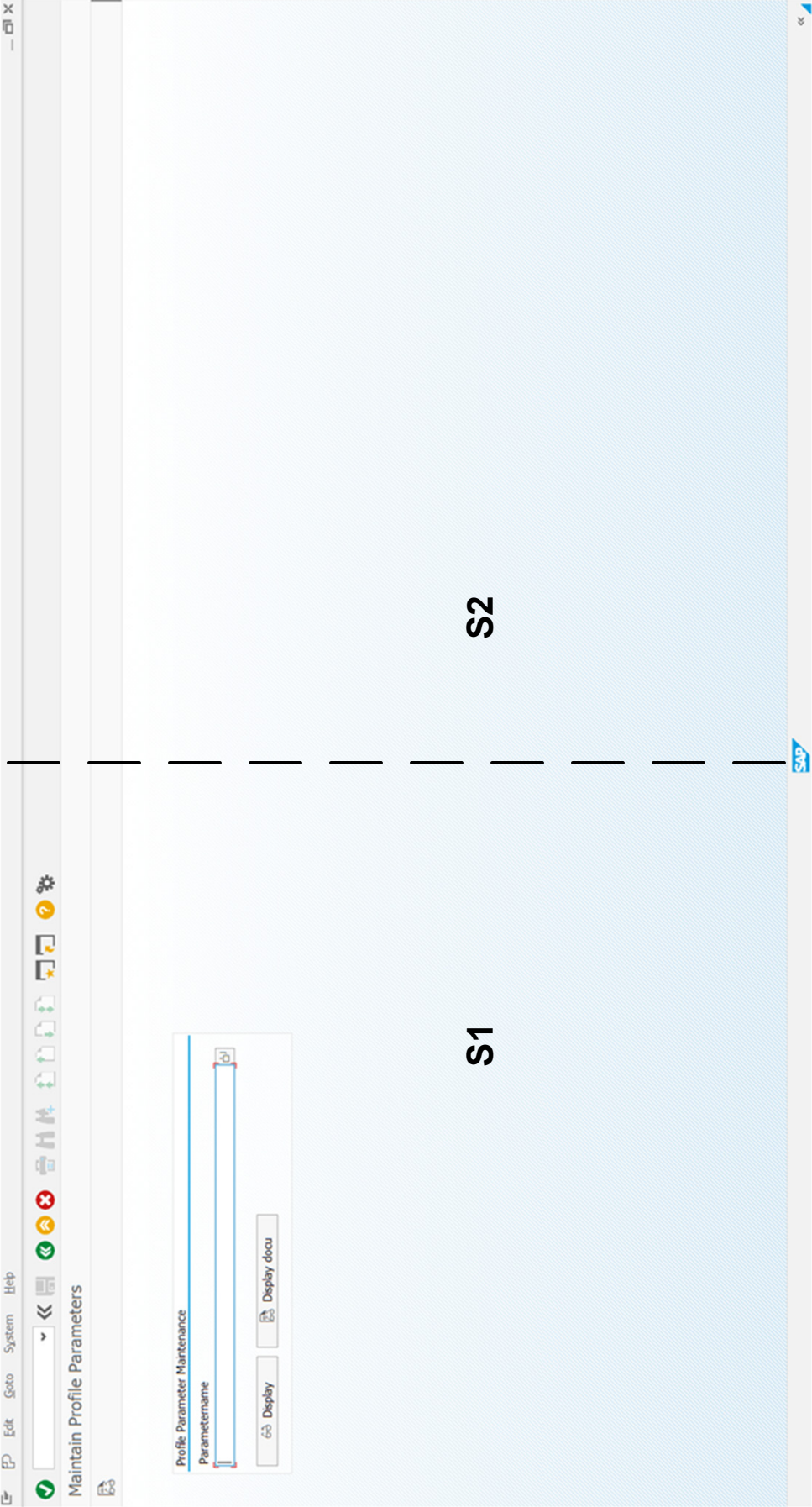


Fig. 12D

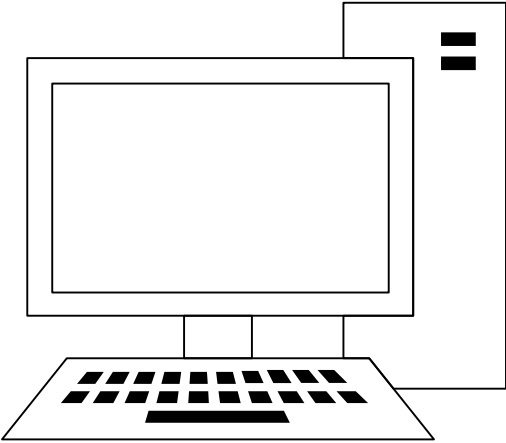


Figure 13A

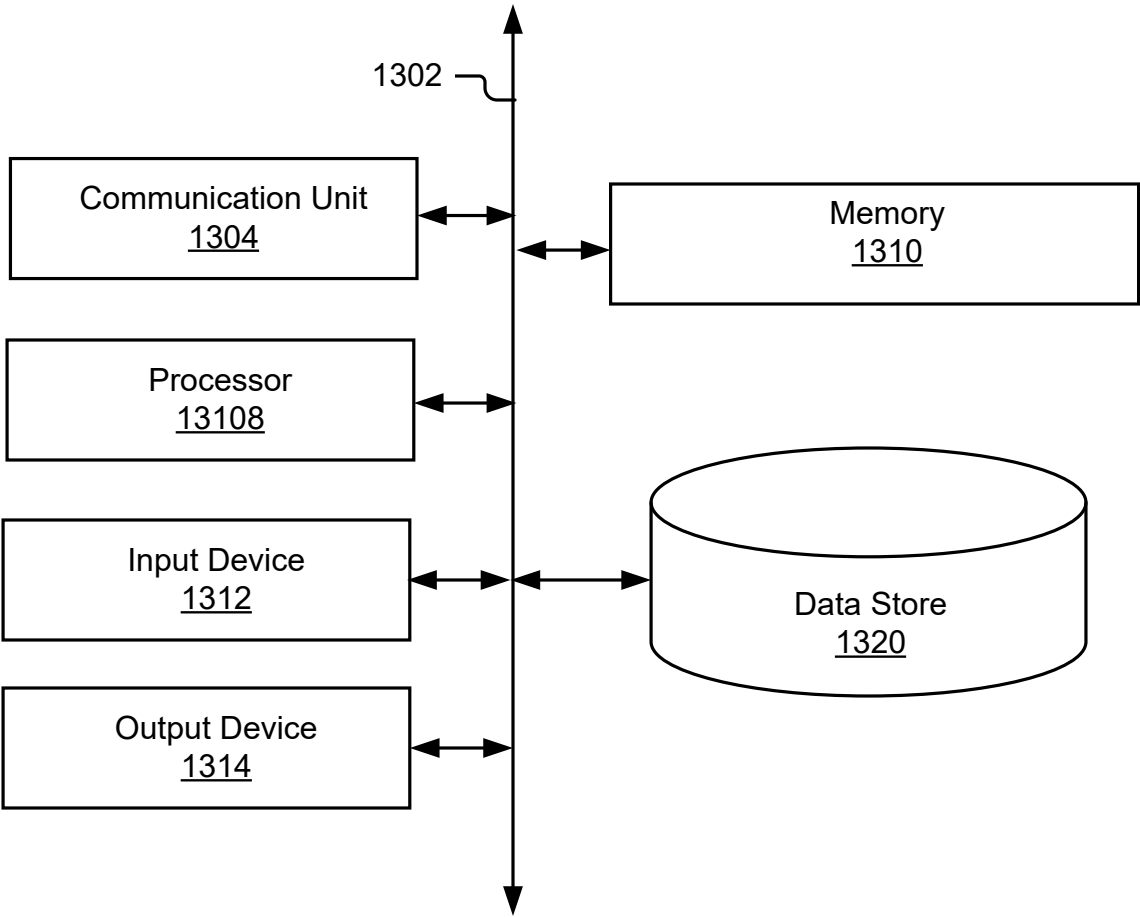


Figure 13B

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

DECLARATION (37 CFR 1.63) FOR UTILITY OR DESIGN APPLICATION USING AN APPLICATION DATA SHEET (37 CFR 1.76)

Title of Invention	SYSTEM AND METHOD FOR DETECTING A CHANGE IN CONTEXT OF AN APPLICATION USING SUBSECTIONS
---------------------------	--

As the below named inventor, I hereby declare that:

This declaration is directed to: The attached application, or
 United States application or PCT international application number _____
filed on _____.

The above-identified application was made or authorized to be made by me.


I believe that I am the original inventor or an original joint inventor of a claimed invention in the application.

I hereby acknowledge that any willful false statement made in this declaration is punishable under 18 U.S.C. 1001 by fine or imprisonment of not more than five (5) years, or both.

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

LEGAL NAME OF INVENTOR

Inventor: Sneha Saxena Date (Optional) : 09/29/2023Signature:  /
EF3902C5D7FD49A...

Note: An application data sheet (PTO/SB/14 or equivalent), including naming the entire inventive entity, must accompany this form or must have been previously filed. Use an additional PTO/AIA/01 form for each additional inventor.

A Federal agency may not conduct or sponsor, and a person is not required to respond to, nor shall a person be subject to a penalty for failure to comply with an information collection subject to the requirements of the Paperwork Reduction Act of 1995, unless the information collection has a currently valid OMB Control Number. The OMB Control Number for this information collection is 0651-0032. Public burden for this form is estimated to average 1 minute per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the information collection. Send comments regarding this burden estimate or any other aspect of this information collection, including suggestions for reducing this burden to the Chief Administrative Officer, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450 or email InformationCollection@uspto.gov. **DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS.** If filing this completed form by mail, send to: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

DECLARATION (37 CFR 1.63) FOR UTILITY OR DESIGN APPLICATION USING AN APPLICATION DATA SHEET (37 CFR 1.76)

Title of Invention	SYSTEM AND METHOD FOR DETECTING A CHANGE IN CONTEXT OF AN APPLICATION USING SUBSECTIONS
---------------------------	--

As the below named inventor, I hereby declare that:

This declaration is directed to: The attached application, or
 United States application or PCT international application number _____
 filed on _____.

The above-identified application was made or authorized to be made by me.


I believe that I am the original inventor or an original joint inventor of a claimed invention in the application.

I hereby acknowledge that any willful false statement made in this declaration is punishable under 18 U.S.C. 1001 by fine or imprisonment of not more than five (5) years, or both.

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

LEGAL NAME OF INVENTOR

Inventor: Muthukrishnan Thukkaram Date (Optional) : 09/29/2023Signature: 
DocuSigned by: Muthukrishnan Thukkaram
81FCE0D724E6437...

Note: An application data sheet (PTO/SB/14 or equivalent), including naming the entire inventive entity, must accompany this form or must have been previously filed. Use an additional PTO/AIA/01 form for each additional inventor.

A Federal agency may not conduct or sponsor, and a person is not required to respond to, nor shall a person be subject to a penalty for failure to comply with an information collection subject to the requirements of the Paperwork Reduction Act of 1995, unless the information collection has a currently valid OMB Control Number. The OMB Control Number for this information collection is 0651-0032. Public burden for this form is estimated to average 1 minute per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the information collection. Send comments regarding this burden estimate or any other aspect of this information collection, including suggestions for reducing this burden to the Chief Administrative Officer, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450 or email InformationCollection@uspto.gov. **DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS.** If filing this completed form by mail, send to: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. The United States Patent and Trademark Office (USPTO) collects the information in this record under authority of 35 U.S.C. 2. The USPTO's system of records is used to manage all applicant and owner information including name, citizenship, residence, post office address, and other information with respect to inventors and their legal representatives pertaining to the applicant's/owner's activities in connection with the invention for which a patent is sought or has been granted. The applicable Privacy Act System of Records Notice for the information collected in this form is COMMERCE/PAT-TM-7 Patent Application Files, available in the Federal Register at 78 FR 19243 (March 29, 2013). <https://www.govinfo.gov/content/pkg/FR-2013-03-29/pdf/2013-07341.pdf>

Routine uses of the information in this record may include disclosure to: 1) law enforcement, in the event that the system of records indicates a violation or potential violation of law; 2) a Federal, state, local, or international agency, in response to its request; 3) a contractor of the USPTO having need for the information in order to perform a contract; 4) the Department of Justice for determination of whether the Freedom of Information Act (FOIA) requires disclosure of the record; 5) a Member of Congress submitting a request involving an individual to whom the record pertains, when the individual has requested the Member's assistance with respect to the subject matter of the record; 6) a court, magistrate, or administrative tribunal, in the course of presenting evidence, including disclosures to opposing counsel in the course of settlement negotiations; 7) the Administrator, General Services Administration (GSA), or their designee, during an inspection of records conducted by GSA under authority of 44 U.S.C. 2904 and 2906, in accordance with the GSA regulations and any other relevant (i.e., GSA or Commerce) directive, where such disclosure shall not be used to make determinations about individuals; 8) another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)); 9) the Office of Personnel Management (OPM) for personnel research purposes; and 9) the Office of Management and Budget (OMB) for legislative coordination and clearance.

If you do not furnish the information requested on this form, the USPTO may not be able to process and/or examine your submission, which may result in termination of proceedings, abandonment of the application, and/or expiration of the patent.

Additional Uses

Additional USPTO uses of the information in this record may include disclosure to: 1) the International Bureau of the World Intellectual Property Organization, if the record is related to an international application filed under the Patent Cooperation Treaty; 2) the public i) after publication of the application pursuant to 35 U.S.C. 122(b), ii) after issuance of a patent pursuant to 35 U.S.C. 151, iii) if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections, or an issued patent, or iv) without publication of the application or patent under the specific circumstances provided for by 37 CFR 1.14(a)(1)(v)-(vii); and/or 3) the National Archives and Records Administration, for inspection of records.

TRANSMITTAL FOR POWER OF ATTORNEY TO ONE OR MORE REGISTERED PRACTITIONERS

NOTE: This form is to be submitted with the Power of Attorney by Applicant form (PTO/AIA/82B) to identify the application to which the Power of Attorney is directed, in accordance with 37 CFR 1.5, unless the application number and filing date are identified in the Power of Attorney by Applicant form. If neither form PTO/AIA/82A nor form PTO/AIA82B identifies the application to which the Power of Attorney is directed, the Power of Attorney will not be recognized in the application.

Application Number	
Filing Date	
First Named Inventor	Sneha Saxena
Title	SYSTEM AND METHOD FOR DETECTING A CHANGE IN CONTEXT OF AN APPLICATION USING SUBSECTIONS
Art Unit	
Examiner Name	
Attorney Docket Number	10875-10150 US

SIGNATURE of Applicant or Patent Practitioner

Signature	/Edward Van Gieson/	Date (Optional)	
Name	Edward Van Gieson	Registration Number	44,386
Title (if Applicant is a juristic entity)	Attorney of Record		
Applicant Name (if Applicant is a juristic entity)	Whatfix Private Limited		

NOTE: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4(d) for signature requirements and certifications. If more than one applicant, use multiple forms.

*Total of _____ forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.32, and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

POWER OF ATTORNEY BY APPLICANT

I hereby revoke all previous powers of attorney given in the application identified in either the attached transmittal letter or the boxes below.

Application Number	Filing Date

(Note: The boxes above may be left blank if information is provided on form PTO/AIA/82A.)

I hereby appoint the Patent Practitioner(s) associated with the following Customer Number as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above:

93219

OR

I hereby appoint Practitioner(s) named in the attached list (form PTO/AIA/82C) as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the patent application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above. (Note: Complete form PTO/AIA/82C.)

Please recognize or change the correspondence address for the application identified in the attached transmittal letter or the boxes above to:

The address associated with the above-mentioned Customer Number

OR

The address associated with Customer Number:

OR

Firm or Individual Name

Address

City

State

Zip

Country

Telephone

Email

I am the Applicant (if the Applicant is a juristic entity, list the Applicant name in the box):

WHATFIX PRIVATE LIMITED

- Inventor or Joint Inventor (title not required below)
- Legal Representative of a Deceased or Legally Incapacitated Inventor (title not required below)
- Assignee or Person to Whom the Inventor is Under an Obligation to Assign (provide signer's title if applicant is a juristic entity)
- Person Who Otherwise Shows Sufficient Proprietary Interest (e.g., a petition under 37 CFR 1.46(b)(2) was granted in the application or is concurrently being filed with this document) (provide signer's title if applicant is a juristic entity)

SIGNATURE of Applicant for Patent

The undersigned (whose title is specified below) is authorized to act on behalf of the applicant (e.g., where the applicant is a juristic entity).

Signature  / Date (Optional) 11/29/2022

Name Khadim Hussain Ismail Batti

Title CEO

NOTE: Signature - This form must be signed by the applicant in accordance with 37 CFR 1.33. See 37 CFR 1.4 for signature requirements and certifications. If more than one applicant, use multiple forms.

Total of forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.32, and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.