(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 September 2014 (18.09.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/145933 A2

(51) International Patent Classification:
F26B 3/20 (2006.01)

(21) International Application Number:
PCT/US2014/030789

(22) International Filing Date:
17 March 2014 (17.03.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/799,427 15 March 2013 (15.03.2013) US

(71) Applicant: **BLINKX, INC.** [US/US]; One Market Plaza, Spear Tower, Suite 1810, San Francisco, CA 94105 (US).

(72) Inventors: **STEEN, Henrik, Ambjoern**; 13803 70th Avenue Ne, Kirkland, WA 98034 (US). **WEINBERGER, Keith**; 2707 Mount St. Helens Place S, Seattle, WA 98144 (US).

(74) Agent: **SCHWAAB, Andrew, B.**; Dla Piper LLP, 2000 University Avenue, East Palo Alto, CA 94303 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) Title: SYSTEMS AND METHODS OF PROCESSING DATA INVOLVING PRESENTATION OF INFORMATION ON ANDROID DEVICES

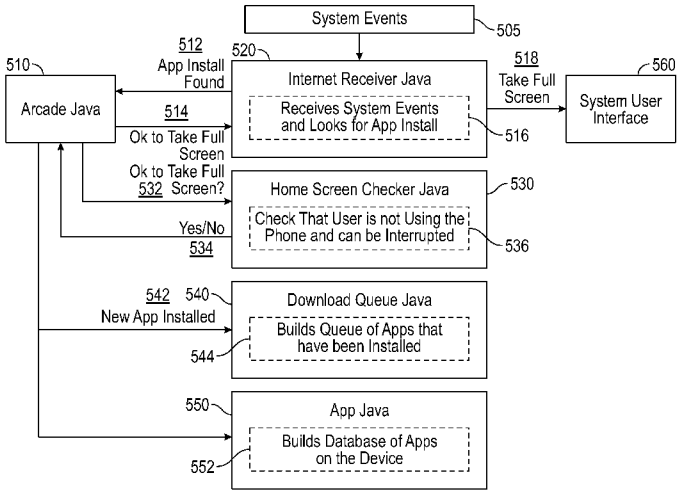


FIG. 5

(57) Abstract: Systems and methods may display an advertisement on a mobile computing device comprising a processor and a memory. An application module coupled to the processor may be constructed and arranged to identify at least one application of the mobile computing device and store a database including at least one listing of the at least one application in the memory. A queue module coupled to the processor may be constructed and arranged to build a queue of the at least one application from the at least one listing in the database. A listener module coupled to the processor may be constructed and arranged to identify at least one broadcast event generated by the processor and associated with the at least one installed application and display at least one advertisement in response to the at least one broadcast event.

WO 2014/145933 A2

Systems and Methods of Processing Data Involving Presentation of Information on Android Devices

Cross-Reference to Related Application Information

This application claims benefit/priority to U.S. provisional patent application No. 61/799,427, filed March 15, 2013, which is incorporated herein by reference in entirety.

Appendix

This application incorporates the attached Appendices A and B of computer code and representative commands, in connection with aspects of the innovations herein may be implemented and utilized.

Background

Conventional advertising methods on mobile devices such as those utilizing the Android operating system rely upon application developers to call for an advertisement at specific points within the user mechanics of an application. For instance, a game application may call for an ad in between levels of a game. Another instance is for an ad to be called once the user launches or exits an application.

Overview

The inventions herein relate to the presentation of an advertising offer immediately after a user installs an application on a mobile computing device. In some implementations, a method of displaying an advertisement on a mobile computing device is provided and includes determining reception of an operating system broadcast event output by the device indicating application installation completion, performing processing to determine an advertisement based on the received operating system broadcast event, and displaying the advertisement on the device immediately upon reception of the application installation completion associated with the operating system broadcast event.

Furthermore, the advertisement may be contextually relevant to an application associated with the application installation completion. The contextually relevant advertisement may promote a product that shares at least one characteristic with the application associated with the application installation completion. The product may be a second application that shares a same type as the application associated with the application installation completion.

In some implementations, the method includes executing the determining step, the performing step, and the displaying step as a function of a third party application to monetize the third party application. The operating system broadcast event may be output by the device to applications of the device. The determined advertisement may be requested and received from a server. The method may further include receiving user interaction with a user interface of the displayed advertisement on the device, transmitting the user interaction to a server, and storing the user interaction on the server. The broadcast event may be a package install or package added action.

In another implementation, a method of displaying an advertisement on a mobile computing device, includes determining reception of an operating system broadcast event output by the device indicating application installation completion, performing processing to determine an advertisement based on the received operating system broadcast event after application installation, and displaying the advertisement on the device prior to execution of the application associated with the operating system broadcast event signal.

Furthermore, the broadcast event may be an operating system broadcast intent. The method may include determining whether the advertisement may be displayed by interrupting control of the device. The method may include generating a queue of installed applications including an application corresponding to the application installation completion. The method may include generating a database of applications installed on the device based on the received operating system broadcast event. The advertisement may take full display screen control of the device.

The advertisement may allow user input to operate the installed application with one tap. The advertisement allows user input to download another application with one tap. The advertisement may allow user input to display a website with one tap. The

advertisement may provide reminder functionality to execute the application. The reception determination step may include listening for an operating system broadcast action. The determining reception step may include listening for a broadcast intent signal. The broadcast event may be a package installed signal. The broadcast event may be an install referrer output from an application download site. The advertisement may offer an application to download from an external server. The advertisement may be any one of an offer to download another application, a banner advertisement, and a video advertisement. The displayed advertisement may be provided in a graphical user interface. The device may be an Android device.

In another implementation, a method of displaying an advertisement on a device includes determining reception of a broadcast event, determining an advertisement based on the determined broadcast event, and displaying the advertisement on the device.

In another implementation, a method of displaying an advertisement on a mobile computing device includes determining reception of an operating system broadcast event output by the device indicating application installation completion, generating a queue of installed applications including an application corresponding to the application installation completion, generating a database of applications installed on the device based on the received operating system broadcast event, and determining whether control of the device may be interrupted by full screen control of a display of the device by another application.

In another implementation, a method of displaying an advertisement on a mobile computing device includes browsing applications for download from an application server, downloading an application from the application server, broadcasting an Android operating system broadcast action indicating application installation. The method further includes executing in a third party application, listening for the Android operating system broadcast action indicating application installation, determining an advertisement that is contextually relevant to the application based on receiving the Android operating system broadcast action, retrieving the determined advertisement from an external server, and displaying the advertisement on the device immediately upon reception of the application installation associated with the operating system

broadcast action. The advertisement may be displayed on a full screen of the device and include an offer to download another application contextually relevant to the downloaded application. The method may further include receiving user interaction with a user interface of the displayed advertisement on the device, transmitting the user interaction to a server, and storing the user interaction on the server, wherein the user interaction downloads the contextually relevant another application.

It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the invention, as described. Further features and/or variations may be provided in addition to those set forth herein. For example, the present invention may be directed to various combinations and subcombinations of the disclosed features and/or combinations and subcombinations of several further features disclosed below in the detailed description.

Description of the Drawings

The accompanying drawings, which constitute a part of this specification, illustrate various implementations and aspects of the innovations herein and, together with the description, help illustrate the principles of the present inventions. In the drawings:

FIG. 1 is a system block diagram illustrating operation of a device consistent with certain aspects related to the innovations herein.

FIG. 2 is a sequence diagram illustrating operation of a system consistent with certain aspects related to the innovations herein.

FIG. 3 is another sequence diagram illustrating operation of a system consistent with certain aspects related to the innovations herein.

FIG. 4 is a flowchart illustrating operation of a device consistent with certain aspects related to the innovations herein.

FIG. 5 is another flowchart illustrating operation of a device consistent with certain aspects related to the innovations herein.

Detailed Description of Illustrative Implementations

Reference will now be made in detail to the inventions herein, examples of which are illustrated in the accompanying drawings. The implementations set forth in the following description do not represent all implementations consistent with the claimed inventions. Instead, they are merely some examples consistent with certain aspects related to the systems and methods described herein. Wherever possible, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

In connection with development functions of the Android developer tools, the systems and methods described herein listen for an Android system broadcast action that indicates an application has been installed on the device. In one embodiment the system listens for the broadcast intent named ACTION_PACKAGE_ADDED to detect when an application is installed on the device. Upon detection of such action, the system takes full screen control of the device and presents the user with advertising. The types of advertising possible within this space include offers to download related apps, banner ads, video ads and others. In another embodiment, the full screen may also display utility functions such as a one tap method to run the installed app, a method to set a reminder to run the installed app, and other functions useful to the user. This system may be developed as a standalone application or embedded in another application as a module. Some embodiments may use the specific system event associated with the installation of an app to take full screen control of the device and display an ad. .

Systems and methods described herein may comprise one or more computers, which may also be referred to as processors. A computer may be any programmable machine or machines capable of performing arithmetic and/or logical operations. In some embodiments, computers may comprise processors, memories, data storage devices, and/or other commonly known or novel components. These components may be connected physically or through network or wireless links. Computers may also comprise software which may direct the operations of the aforementioned components. Computers may be referred to with terms that are commonly used by those of ordinary skill in the relevant arts, such as servers, PCs, mobile devices, routers, switches, data centers, distributed computers, and other terms. Computers may facilitate

communications between users and/or other computers, may provide databases, may perform analysis and/or transformation of data, and/or perform other functions. It will be understood by those of ordinary skill that those terms used herein are interchangeable, and any computer capable of performing the described functions may be used. For example, though the term "server" may appear in the following specification, the disclosed embodiments are not limited to a server.

Computers may be linked to one another via a network or networks. A network may be any plurality of completely or partially interconnected computers wherein some or all of the computers are able to communicate with one another. It will be understood by those of ordinary skill that connections between computers may be wired in some cases (e.g., via Ethernet, coaxial, optical, or other wired connection) or may be wireless (e.g., via Wi-Fi, WiMax, or other wireless connection). Connections between computers may use any protocols, including connection oriented protocols such as TCP or connectionless protocols such as UDP. Any connection through which at least two computers may exchange data can be the basis of a network.

FIG. 1 is a system block diagram illustrating operation of a device consistent with certain aspects related to the innovations herein. In FIG. 1, a mobile device 110, such as an Android device, is shown broadcasting a system event, and the device may listen to the broadcast events. There may be many standard and/or custom broadcast events utilized by Android devices, as those of ordinary skill in the art will appreciate. For example, the device may listen for certain specific events, and if one is detected, the device may call out for an advertisement to display. The advertisement may be combined into a UI that may be displayed to the user. An external ad server may be called and may deliver ads in response. An external database may record user interaction with the advertisement.

FIG. 1 depicts a method of displaying an advertisement on a mobile computing device and includes determining reception of an operating system broadcast event output by the device indicating application installation completion, performing processing to determine an advertisement based on the received operating system broadcast event, and displaying the advertisement on the device immediately upon

reception of the application installation completion associated with the operating system broadcast event.

The mobile device 110 may output an operating system event at step 120, for example an Android operating system intent broadcast message 130. The mobile device 110 may include a processor. A resident application 140 may be executed on the processor to perform the steps 142-156. In some embodiments, the resident application may be embedded within a third party application that performs other functions, such as a game or a utility application. At step 142, the intent listener listens to the intent broadcast 130 and determines at step 144 whether a desired intent broadcast is received. If so, the process proceeds to step 146 where a type of advertisement to display is determined. For example, the advertisement may be contextually relevant to an application associated with the application installation completion. The contextually relevant advertisement may promote a product that shares at least one characteristic with the application associated with the application installation completion. The product may be a second application that shares a same type as the application associated with the application installation completion, for example. The mobile device 110 then may retrieve an advertisement 148 from an external server 160, such as an advertisement server 162. Once an advertisement is retrieved, externally or from storage within the device 110, the advertisement may be displayed 150 on a display of the device 110. The resident application 140 may also record the user interaction 152 with the advertisement and transmit the user action 154. The user action may be recorded 164 on the external server 160.

In some embodiments, the determining step, the performing step, and the displaying step may be executed as a function of a third party application to monetize the third party application. The operating system broadcast event may be output by the device to applications of the device. The determined advertisement may be requested and received from a server. The broadcast event may be a package install or package added action.

FIG. 2 is a sequence diagram illustrating operation of a system consistent with certain aspects related to the innovations herein. As shown in FIG. 2, displaying an advertisement on a mobile computing device may include determining reception of an

operating system broadcast event output by the device indicating application installation completion, performing processing to determine an advertisement based on the received operating system broadcast event after application installation, and displaying the advertisement on the device prior to execution of the application associated with the operating system broadcast event signal.

Furthermore, the broadcast event may be an operating system broadcast intent. The method may include determining whether the advertisement may be displayed by interrupting control of the device. The method may include generating a queue of installed applications including an application corresponding to the application installation completion. The method may include generating a database of applications installed on the device based on the received operating system broadcast event. The advertisement may take full display screen control of the device.

FIG. 2 shows a sequence of events that may occur in a system including a user 210, internet browser 220 on the device, the Google Play Store 230 , the Android system 240, the system/methods described herein 250, advertisement server 260, and an advertisement network 270. The sequence starts with the user 210 interacting 214 with the Android system of the mobile device 110. The interaction may cause a broadcast event 218 to be sent by the mobile device 110. The broadcast event may be a package installed signal or an install referrer output from an application download site, for example. In response, a broadcast intent 222 may be transmitted, and the system/methods described herein may listen to the broadcast intent and call to an advertisement server 260 for an advertisement 224. The advertisement server 260 may call for an advertisement from the advertisement network 270. The advertisement network 270 may return an advertisement 228 to the advertisement server 260, and the advertisement server 260 may send the advertisement to the mobile device at step 232. A graphical user interface (GUI) may be presented to the user at step 234. The GUI may take control of the display screen space, display the advertisement, and provide any associated functionality with the advertisement. The advertisement may be displayed to the user at step 236, and when the user interacts 238 with the advertisement, the user may initiate functionality to visit a website 242, to download a different application, and/or other functionality.

In some embodiments, the advertisement may allow user input to operate the installed application with one tap. The advertisement may allow user input to download another application with one tap. The advertisement may allow user input to display a website with one tap. The advertisement may provide reminder functionality to execute the application. The advertisement may offer an application to download from an external server. The advertisement may be any one of an offer to download another application, a banner advertisement, and a video advertisement. The displayed advertisement may be provided in a graphical user interface.

FIG. 3 is a sequence diagram illustrating operation of a system consistent with certain aspects related to the innovations herein. This diagram shows one embodiment in which the specific broadcast event named "PACKAGE_INSTALL" is detected and operated upon. PACKAGE_INSTALL may be broadcast by the Android operating system when a new application completes installation on the device.

FIG. 3 illustrates the sequence of events occurring in a system including a user 210, internet browser 220 on the device 110, the Google Play Store 230 , the Android system 240, the system/methods described herein 250, advertisement server 260, and an advertisement network 270. The sequence starts with the user 210 interacting 302 with the Android system of the mobile device to transmit 302 and receive 306 data from an application store. The user may select to view information on applications 308 and may receive corresponding data 310 from the application store. The user may select to start installation of an application 312, and the user may receive an indication 314 of application selection. The application store then may communicate with the operating system 240 to start the installation process 316. The user may perform other actions at step 330 while waiting for installation of the selected application to complete. The operating system 240 may transmit a broadcast intent signal such as PACKAGE_INSTALL at step 318. The system/methods described herein may listen to the broadcast intent and call to an advertisement server 260 for an advertisement 320. The advertisement server 260 may call for an advertisement 322 from the advertisement network 270. The advertisement network 270 may return an advertisement 324 to the advertisement server 260, and the advertisement server 260 may send the advertisement to the mobile device at step 326. A GUI may be presented

to the user at step 234. The GUI may take control of the display screen space, display the advertisement, and provide any associated functionality with the advertisement. The advertisement may be displayed to the user at step 236, and when the user interacts 238 with the advertisement, the user may initiate functionality to visit a website 242, to download a different application, and/or other functionality.

FIG. 4 is a flowchart illustrating operation 400 of a device consistent with certain aspects related to the innovations herein. This example operation may be performed with an Android device. The system may listen for broadcast events sent by the Android operating system and determine if an event should be acted upon. If so, an advertisement to show to the user may be produced by calling for an ad from an external server and determining the best way to show the advertisement. The advertisement may be shown to the end user, and the end user's interaction with the advertisement may be recorded.

At step 402, the device 110 may for the Android system to broadcast a standard action, detect the type of action, and determine if the action is one that should be acted upon. If the action should be acted upon, in step 404 the device 110 may determine what kind of advertisements should be shown, given the action in question. The device 110 may also determine the way to display advertisements. In step 406, webservers may be called for advertisements, and the advertisements may be built for display. At step 408, the advertisements may be shown on the device. In step 410, user interaction with the advertisements may be allowed. The user interaction with advertisements may be recorded, and user interaction data may be sent to external recording servers.

FIG. 5 is a flowchart illustrating operation of a device consistent with certain aspects related to the innovations herein. In particular, FIG. 5 describes a processing by code module. A main code module may be provided. For example, in FIG. 5, Arcade.java 510 is the main code module. Arcade.java 510 may manage the application's set up and loading of resources and other code modules for proper operation on the mobile device 110. Once loaded and running on the device, the other code modules may function as follows.

Module App.java 550 may build and maintain a database of apps 552 that reside on the device. The database may be referenced by the DownloadQueue.java 540

module to determine if an application should be displayed in the device's interface. The determination may be based on whether the application has already been run by the user or not, for example. When a new application is installed, Arcade.java 510 may transmit 542 to DownloadQueue.java 540 and App.Java 550 that a new application has been installed. DownloadQueue.Java 540 may build a queue of applications that have been installed 544.

A listener module, e.g., IntentReceiver.java 520, may listen for, receive, and processes system events 505 that are broadcast by the Android operating system. The broadcast event may be an operating system broadcast intent, for example. IntentReceiver.java 520 may look for system events 516 that can be used to trigger the call and display for an advertisement. In some implementations, IntentReceiver 520 may look for a system event 516 that means a new application has been installed on the device, for example. IntentReceiver.java 520 may communicate information about a system event event 512 to Arcade.java 510.

When Arcade.java 510 receives information about a new application install event, it may cause 532 HomeScreenChecker.java 530 module to run a set of rules to determine if it is permissible 536 to display information full screen on the device and/or whether the advertisement may be displayed by interrupting control of the device. Various rules may be established to control aspects of advertisement display. For example, rules may prevent the display from interrupting a user if they are using the device as a phone, and/or rules may ensure the screen is not in screen-lock mode before showing an advertisement. Other rules useful to the end user experience may also be established.

If Arcade.java 510 receives confirmation 534 from HomeScreenChecker.java 530 that it is OK to display information for the user full screen, Arcade.java 510 may communicate a display command 514 to IntentReceiver.java 520. IntentReceiver.java 520 may take control 518 of the system user interface device screen 560 and display the application interface for the user. The application interface may provide the end user with the tools and controls to run the application that was just installed, set a reminder, and/or run other applications that were installed before but not yet run, for example. The application interface may also display advertising along with these tools and controls.

In some implementations, the advertisements are contained within the application interface using a webview and a call to a URL to retrieve a webpage containing an advertisement. In some other implementations, the advertisements may be called directly by the application and added to the application interface or displayed on the mobile device in a way congruent with the functioning of the application.

As disclosed herein, features consistent with the present inventions may be utilized via and/or involved with various circuits/circuitry, hardware, software and/or firmware. For example, the systems and methods disclosed herein may be embodied in or used in connection with various forms including, for example, memory, data processors, such as in computing devices that also includes memory, a database, digital electronic circuitry, firmware, software, or in combinations of them. Further, while some of the disclosed implementations describe specific hardware components, systems and methods consistent with the innovations herein may be implemented in the context of any combination of hardware, software and/or firmware. Moreover, the above-noted features and other aspects and principles of the innovations herein may be implemented in various memory environments. Such environments and related applications may be specially constructed for performing the various routines, processes and/or operations according to the invention or they may include a general-purpose computer or computing platform selectively activated or reconfigured by code to provide the necessary functionality. The processes disclosed herein are not inherently related to any particular computer, network, architecture, environment, or other apparatus, and may be implemented by a suitable combination of hardware, software, and/or firmware. For example, various general-purpose machines may be used with programs written in accordance with teachings of the invention, or it may be more convenient to construct a specialized apparatus or system to perform the required methods and techniques.

Aspects of the method and system described herein, such as the logic, may be implemented as functionality programmed into any of a variety of circuitry, including programmable logic devices ("PLDs"), such as field programmable gate arrays ("FPGAs"), programmable array logic ("PAL") devices, electrically programmable logic and memory devices and standard cell-based devices, as well as application specific

integrated circuits. Some other possibilities for implementing aspects include: memory devices, microcontrollers with memory (such as EEPROM), embedded microprocessors, firmware, software, etc. Furthermore, aspects may be embodied in microprocessors having software-based circuit emulation, discrete logic (sequential and combinatorial), custom devices, fuzzy (neural) logic, quantum devices, and hybrids of any of the above device types. The underlying device technologies may be provided in a variety of component types, e.g., metal-oxide semiconductor field-effect transistor ("MOSFET") technologies like complementary metal-oxide semiconductor ("CMOS"), bipolar technologies like emitter-coupled logic ("ECL"), polymer technologies (e.g., silicon-conjugated polymer and metal-conjugated polymer-metal structures), mixed analog and digital, and so on.

It should also be noted that the various logic and/or functions disclosed herein may be enabled using any number of combinations of hardware, firmware, and/or as data/instructions embodied in various machine-readable or computer-readable media, in terms of their behavioral, register transfer, logic component, and/or other characteristics. Computer-readable media in which such formatted data and/or instructions may be embodied include, but are not limited to, non-volatile storage media in various forms (e.g., optical, magnetic or semiconductor storage media), though does not include transitory media such as carrier waves.

Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense as opposed to an exclusive or exhaustive sense; that is to say, in a sense of "including, but not limited to." Words using the singular or plural number also include the plural or singular number respectively. Additionally, the words "herein," "hereunder," "above," "below," and words of similar import refer to this application as a whole and not to any particular portions of this application. When the word "or" is used in reference to a list of two or more items, that word covers all of the following interpretations of the word: any of the items in the list, all of the items in the list and any combination of the items in the list.

Although certain presently preferred implementations of the invention have been specifically described herein, it will be apparent to those skilled in the art to which the

inventions pertain that variations and modifications of the various implementations shown and described herein may be made without departing from the spirit and scope of the innovations herein. Accordingly, it is intended that the inventions be limited only to the extent required by the applicable rules of law.

Appendix A – Representative Broadcast Actions Detected/Processed

Android Broadcast Actions - from Android SDK Version 17

android.app.action.ACTION_PASSWORD_CHANGED

android.app.action.ACTION_PASSWORD_EXPIRING

android.app.action.ACTION_PASSWORD_FAILED

android.app.action.ACTION_PASSWORD_SUCCEEDED

android.app.action.DEVICE_ADMIN_DISABLED

android.app.action.DEVICE_ADMIN_DISABLE_REQUESTED

android.app.action.DEVICE_ADMIN_ENABLED

android.bluetooth.a2dp.profile.action.CONNECTION_STATE_CHANGED

android.bluetooth.a2dp.profile.action.PLAYING_STATE_CHANGED

android.bluetooth.adapter.action.CONNECTION_STATE_CHANGED

android.bluetooth.adapter.action.DISCOVERY_FINISHED

android.bluetooth.adapter.action.DISCOVERY_STARTED

android.bluetooth.adapter.action.LOCAL_NAME_CHANGED

android.bluetooth.adapter.action.SCAN_MODE_CHANGED

android.bluetooth.adapter.action.STATE_CHANGED

android.bluetooth.device.action.ACL_CONNECTED

android.bluetooth.device.action.ACL_DISCONNECTED

android.bluetooth.device.action.ACL_DISCONNECT_REQUESTED

android.bluetooth.device.action.BOND_STATE_CHANGED

android.bluetooth.device.action.CLASS_CHANGED

android.bluetooth.device.action.FOUND

android.bluetooth.device.action.NAME_CHANGED

android.bluetooth.device.action.UUID

android.bluetooth.devicepicker.action.DEVICE_SELECTED

Appendix A – Representative Broadcast Actions Detected/Processed

android.bluetooth.devicepicker.action.LAUNCH

android.bluetooth.headset.action.VENDOR_SPECIFIC_HEADSET_EVENT

android.bluetooth.headset.profile.action.AUDIO_STATE_CHANGED

android.bluetooth.headset.profile.action.CONNECTION_STATE_CHANGED

android.bluetooth.input.profile.action.CONNECTION_STATE_CHANGED

android.bluetooth.pan.profile.action.CONNECTION_STATE_CHANGED

android.hardware.action.NEW_PICTURE

android.hardware.action.NEW_VIDEO

android.hardware.input.action.QUERY_KEYBOARD_LAYOUTS

android.intent.action.ACTION_POWER_CONNECTED

android.intent.action.ACTION_POWER_DISCONNECTED

android.intent.action.ACTION_SHUTDOWN

android.intent.action.AIRPLANE_MODE

android.intent.action.BATTERY_CHANGED

android.intent.action.BATTERY_LOW

android.intent.action.BATTERY_OKAY

android.intent.action.BOOT_COMPLETED

android.intent.action.CAMERA_BUTTON

android.intent.action.CONFIGURATION_CHANGED

android.intent.action.DATE_CHANGED

android.intent.action.DEVICE_STORAGE_LOW

android.intent.action.DEVICE_STORAGE_OK

android.intent.action.DOCK_EVENT

android.intent.action.DREAMING_STARTED

android.intent.action.DREAMING_STOPPED

Appendix A – Representative Broadcast Actions Detected/Processed

android.intent.action.EXTERNAL_APPLICATIONS_AVAILABLE

android.intent.action.EXTERNAL_APPLICATIONS_UNAVAILABLE

android.intent.action.FETCH_VOICEMAIL

android.intent.action.GTALK_CONNECTED

android.intent.action.GTALK_DISCONNECTED

android.intent.action.HEADSET_PLUG

android.intent.action.INPUT_METHOD_CHANGED

android.intent.action.LOCALE_CHANGED

android.intent.action.MANAGE_PACKAGE_STORAGE

android.intent.action.MEDIA_BAD_REMOVAL

android.intent.action.MEDIA_BUTTON

android.intent.action.MEDIA_CHECKING

android.intent.action.MEDIA_EJECT

android.intent.action.MEDIA_MOUNTED

android.intent.action.MEDIA_NOFS

android.intent.action.MEDIA_REMOVED

android.intent.action.MEDIA_SCANNER_FINISHED

android.intent.action.MEDIA_SCANNER_SCAN_FILE

android.intent.action.MEDIA_SCANNER_STARTED

android.intent.action.MEDIA_SHARED

android.intent.action.MEDIA_UNMOUNTABLE

android.intent.action.MEDIA_UNMOUNTED

android.intent.action.MY_PACKAGE_REPLACED

android.intent.action.NEW_OUTGOING_CALL

android.intent.action.NEW_VOICEMAIL

Appendix A – Representative Broadcast Actions Detected/Processed

android.intent.action.PACKAGE_ADDED

android.intent.action.PACKAGE_CHANGED

android.intent.action.PACKAGE_DATA_CLEARED

android.intent.action.PACKAGE_FIRST_LAUNCH

android.intent.action.PACKAGE_FULLY_REMOVED

android.intent.action.PACKAGE_INSTALL

android.intent.action.PACKAGE_NEEDS_VERIFICATION

android.intent.action.PACKAGE_REMOVED

android.intent.action.PACKAGE_REPLACED

android.intent.action.PACKAGE_RESTARTED

android.intent.action.PACKAGE_VERIFIED

android.intent.action.PHONE_STATE

android.intent.action.PROVIDER_CHANGED

android.intent.action.PROXY_CHANGE

android.intent.action.REBOOT

android.intent.action.SCREEN_OFF

android.intent.action.SCREEN_ON

android.intent.action.TIMEZONE_CHANGED

android.intent.action.TIME_SET

android.intent.action.TIME_TICK

android.intent.action.UID_REMOVED

android.intent.action.USER_PRESENT

android.intent.action.WALLPAPER_CHANGED

android.media.ACTION_SCO_AUDIO_STATE_UPDATED

android.media.AUDIO_BECOMING_NOISY

Appendix A – Representative Broadcast Actions Detected/Processed

android.media.RINGER_MODE_CHANGED

android.media.SCO_AUDIO_STATE_CHANGED

android.media.VIBRATE_SETTING_CHANGED

android.media.action.CLOSE_AUDIO_EFFECT_CONTROL_SESSION

android.media.action.OPEN_AUDIO_EFFECT_CONTROL_SESSION

android.net.conn.BACKGROUND_DATA_SETTING_CHANGED

android.net.nsd.STATE_CHANGED

android.net.wifi.NETWORK_IDS_CHANGED

android.net.wifi.RSSI_CHANGED

android.net.wifi.SCAN_RESULTS

android.net.wifi.STATE_CHANGE

android.net.wifi.WIFI_STATE_CHANGED

android.net.wifi.p2p.CONNECTION_STATE_CHANGE

android.net.wifi.p2p.DISCOVERY_STATE_CHANGE

android.net.wifi.p2p.PEERS_CHANGED

android.net.wifi.p2p.STATE_CHANGED

android.net.wifi.p2p.THIS_DEVICE_CHANGED

android.net.wifi.supplicant.CONNECTION_CHANGE

android.net.wifi.supplicant.STATE_CHANGE

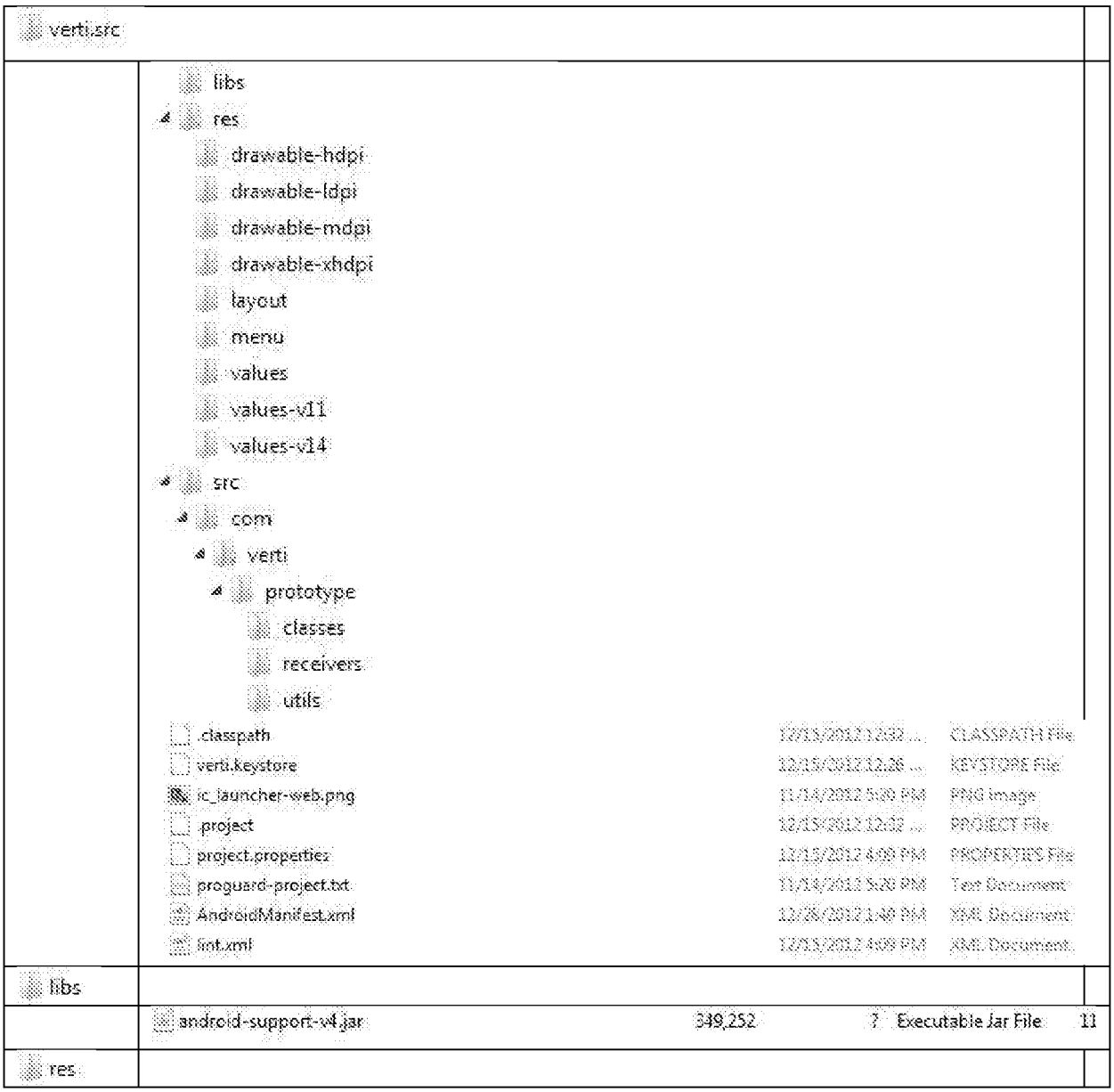
android.speech.tts.TTS_QUEUE_PROCESSING_COMPLETED

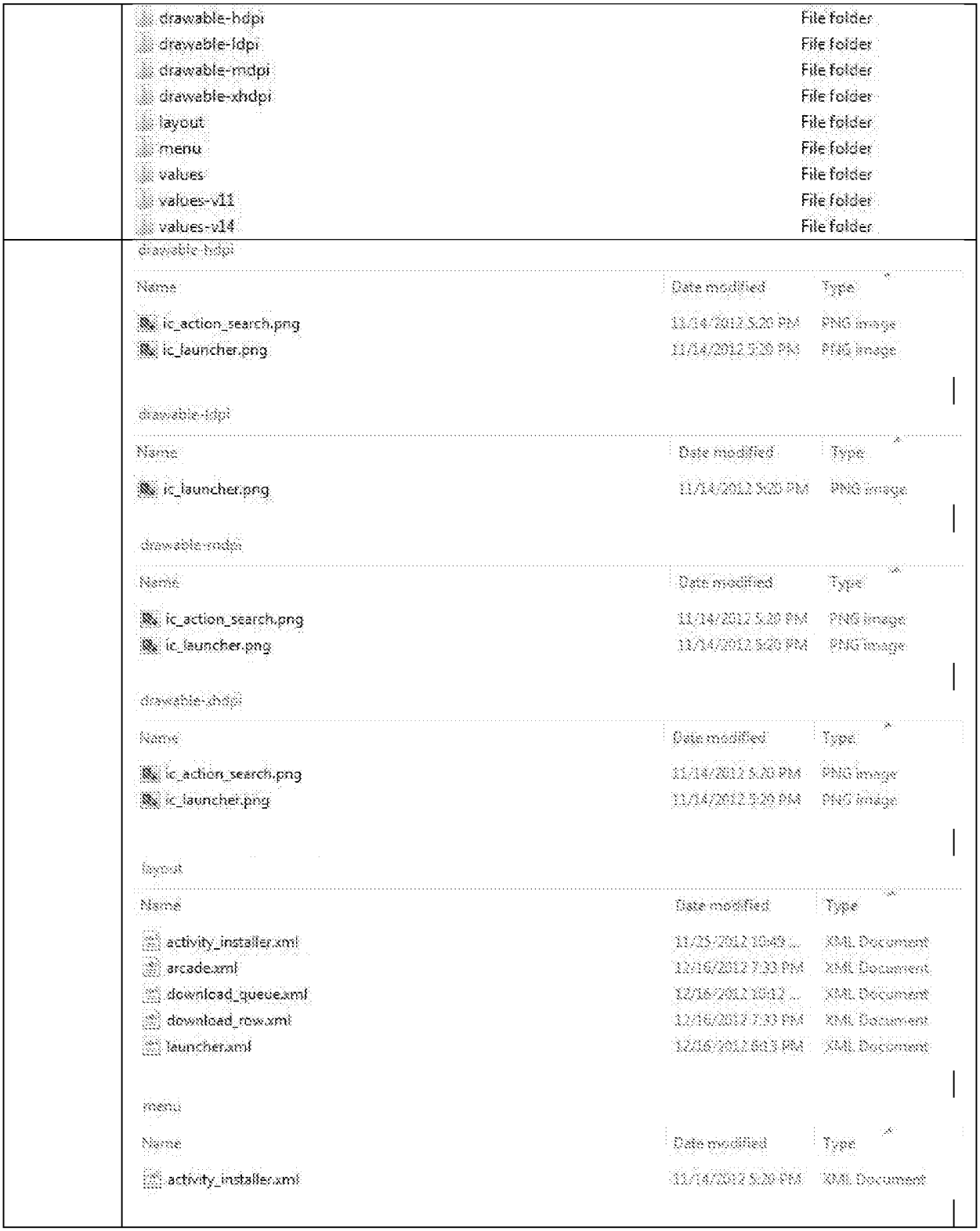
android.speech.tts.engine.TTS_DATA_INSTALLED

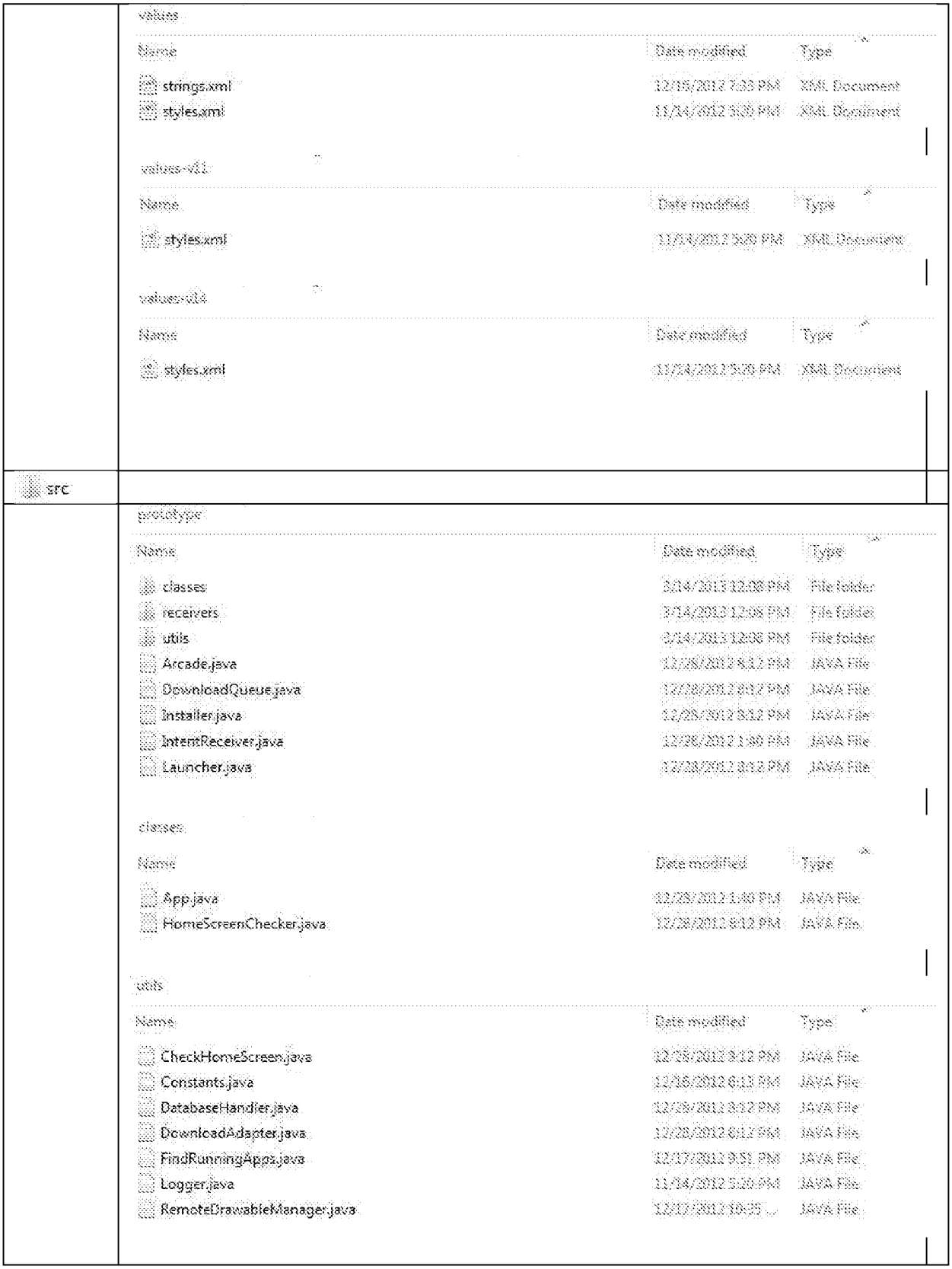
Appendix B

Source Code

Section 1: File Structure




drawable-hdpi File folder
drawable-ldpi File folder
drawable-mdpi File folder
drawable-xhdpi File folder
layout File folder
menu File folder
values File folder
values-v11 File folder
values-v14 File folder
drawable-hdpi
Name Date modified Type
ic_action_search.png 11/14/2012 5:20 PM PNG image
ic_launcher.png 11/14/2012 5:20 PM PNG image
drawable-ldpi
Name Date modified Type
ic_launcher.png 11/14/2012 5:20 PM PNG image
drawable-mdpi
Name Date modified Type
ic_action_search.png 11/14/2012 5:20 PM PNG image
ic_launcher.png 11/14/2012 5:20 PM PNG image
drawable-xhdpi
Name Date modified Type
ic_action_search.png 11/14/2012 5:20 PM PNG image
ic_launcher.png 11/14/2012 5:20 PM PNG image
layout
Name Date modified Type
activity_installer.xml 11/25/2012 10:49 ... XML Document
arcade.xml 12/16/2012 7:33 PM XML Document
download_queue.xml 12/16/2012 10:12 ... XML Document
download_row.xml 12/16/2012 7:33 PM XML Document
launcher.xml 12/16/2012 8:13 PM XML Document
menu
Name Date modified Type
activity_installer.xml 11/14/2012 5:20 PM XML Document


values
Name
Date modified
Type
strings.xml
12/16/2012 7:33 PM
XML Document
styles.xml
11/14/2012 5:20 PM
XML Document
values-v11
Name
Date modified
Type
styles.xml
11/14/2012 5:20 PM
XML Document
values-v14
Name
Date modified
Type
styles.xml
11/14/2012 5:20 PM
XML Document
src
prototype
Name
Date modified
Type
classes
3/14/2013 12:08 PM
File folder
receivers
3/14/2013 12:08 PM
File folder
utils
3/14/2013 12:08 PM
File folder
Arcade.java
12/28/2012 8:12 PM
JAVA File
DownloadQueue.java
12/28/2012 8:12 PM
JAVA File
Installer.java
12/28/2012 8:12 PM
JAVA File
IntentReceiver.java
12/28/2012 1:30 PM
JAVA File
Launcher.java
12/28/2012 8:12 PM
JAVA File
classes
Name
Date modified
Type
App.java
12/28/2012 1:40 PM
JAVA File
HomeScreenChecker.java
12/28/2012 8:12 PM
JAVA File
utils
Name
Date modified
Type
CheckHomeScreen.java
12/28/2012 8:12 PM
JAVA File
Constants.java
12/16/2012 6:13 PM
JAVA File
DatabaseHandler.java
12/28/2012 8:12 PM
JAVA File
DownloadAdapter.java
12/28/2012 8:12 PM
JAVA File
FindRunningApps.java
12/17/2012 9:51 PM
JAVA File
Logger.java
11/14/2012 5:20 PM
JAVA File
RemoteDrawableManager.java
12/17/2012 10:05 ...
JAVA File

not furnished upon filing

Section 2: Source Code by File

Filename: .classpath

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
	<classpathentry kind="src" path="src"/>
	<classpathentry kind="src" path="gen"/>
	<classpathentry kind="con" path="com.android.ide.eclipse.adt.ANDROID_FRAMEWORK"/>
	<classpathentry kind="con" path="com.android.ide.eclipse.adt.LIBRARIES"/>
	<classpathentry kind="output" path="bin/classes"/>
</classpath>
```

Filename: verti.keystore

(encrypted file)

Filename: ic_launcher-web.png

Filename: .project

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
	<name>verti</name>
	<comment></comment>
	<projects>
	</projects>
	<buildSpec>
		<buildCommand>
			<name>com.android.ide.eclipse.adt.ResourceManagerBuilder</name>
			<arguments>
			</arguments>
		</buildCommand>
		<buildCommand>
			<name>com.android.ide.eclipse.adt.PreCompilerBuilder</name>
			<arguments>
			</arguments>
```

```
            </buildCommand>
            <buildCommand>
                <name>org.eclipse.jdt.core.javabuilder</name>
                <arguments>
                </arguments>
            </buildCommand>
            <buildCommand>
                <name>com.android.ide.eclipse.adt.ApkBuilder</name>
                <arguments>
                </arguments>
            </buildCommand>
        </buildSpec>
        <natures>
            <nature>com.android.ide.eclipse.adt.AndroidNature</nature>
            <nature>org.eclipse.jdt.core.javanature</nature>
        </natures>
</projectDescription>
```

Filename: project.properties

```
# This file is automatically generated by Android Tools.
# Do not modify this file -- YOUR CHANGES WILL BE ERASED!
#
# This file must be checked in Version Control Systems.
#
# To customize properties used by the Ant build system edit
# "ant.properties", and override values to adapt the script to your
# project structure.
#
# To enable ProGuard to shrink and obfuscate your code, uncomment this (available properties: sdk.dir,
user.home):
#proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt

# Project target.
target=android-15
```

Filename: proguard-project.txt

```
# To enable ProGuard in your project, edit project.properties
# to define the proguard.config property as described in that file.
```

```
#
# Add project specific ProGuard rules here.
# By default, the flags in this file are appended to flags specified
# in ${sdk.dir}/tools/proguard/proguard-android.txt
# You can edit the include path and order by changing the ProGuard
# include property in project.properties.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# Add any project specific keep options here:

# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}
```

Filename: AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.verti.prototype"
   android:versionCode="1"
   android:versionName="1.0"
    >

   <uses-sdk
      android:minSdkVersion="9"
      android:targetSdkVersion="15" />

   <uses-permission android:name="android.permission.GET_TASKS"></uses-permission>
   <uses-permission android:name="android.permission.INTERNET" />


   <application
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name=".Installer"
```

```
        android:label="@string/title_activity_installer" >
    </activity>

    <activity
        android:name=".Arcade"
        android:label="@string/title_activity_arcade" >
    </activity>

    <activity
        android:name=".Launcher"
        android:theme="@android:style/Theme.NoTitleBar">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <activity
        android:name=".DownloadQueue"
        android:label="@string/title_activity_arcade" >
    </activity>

    <receiver android:name=".IntentReceiver" android:exported="true">
                <intent-filter>
                        <action android:name="com.android.vending.INSTALL_REFERRER" />
                        <action android:name="android.intent.action.PACKAGE_ADDED" />
                        <data android:scheme="package" />
                </intent-filter>

                </receiver>
  </application>

</manifest>
```

Filename: lint.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<lint>
</lint>
```

Filename: android-support-v4.jar
(compiled file)

Filename: ic_action_search.png
(blank image)

Filename: ic_launcher.png

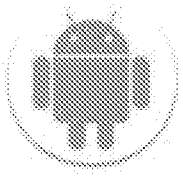
Filename: ic_launcher.png

Filename: ic_action_search.png
(blank image)

Filename: ic_launcher.png

Filename: ic_action_search.png
(blank image)

Filename: ic_launcher.png



Filename: activity_installer.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >

  <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="@string/installer_status"
    tools:context=".Installer" />

  <EditText
    android:id="@+id/url"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/textView1"
    android:layout_marginTop="66dp"
    android:ems="10"
    android:hint="@string/enter_url" />

  <Button
    android:id="@+id/submit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/url"
    android:layout_alignParentRight="true"
    android:text="@string/submit" />
```

```
</RelativeLayout>
```

Filename: arcade.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >

  <TextView
    android:id="@+id/app_ready_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="14dp"
    android:layout_marginTop="10dp"
              android:drawablePadding="5dp"
              android:drawableLeft="@drawable/ic_launcher"
              android:gravity="center|center"
    android:text="@string/default_app_text"
    android:textAppearance="?android:attr/textAppearanceMedium" />

  <Button
    android:id="@+id/run"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="5dp"
    android:layout_marginTop="10dp"
    android:layout_toRightOf="@+id/app_ready_text"
    android:text="@string/run" />

  <Button
    android:id="@+id/remind_me"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/app_ready_text"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="32dp"
    android:layout_marginBottom="10dp"
    android:text="@string/remind_me_later" />
```

```
<RelativeLayout
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:layout_below="@+id/remind_me"
   android:layout_above="@+id/back"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="10dp"
   android:background="#cccccc"
   >


   <WebView
      android:id="@+id/reco_webview"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true" />

</RelativeLayout>

<Button
      android:id="@+id/back"
      style="?android:attr/buttonStyleSmall"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"
      android:layout_centerHorizontal="true"
      android:text="@string/back" />

</RelativeLayout>
```

Filename: download_queue.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent" >

   <ListView
      android:id="@android:id/list"
      android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:layout_above="@+id/install_button"
    android:layout_alignParentTop="true"
    />

        <TextView android:id="@android:id/empty"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/nothing_to_download"/>

        <Button
          android:id="@+id/install_button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:paddingRight="10dp"
          android:layout_alignParentBottom="true"
          android:layout_alignParentRight="true"
          android:text="@string/install" />

</RelativeLayout>
```

Filename: download_row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >

        <TextView android:id="@+id/download_app_name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="14dp"
    android:layout_marginTop="10dp"
                android:layout_alignParentLeft="true"
                android:drawablePadding="5dp"
                android:drawableLeft="@drawable/ic_launcher"
                android:gravity="center|center"
    android:text="@string/app_name"
    android:textAppearance="?android:attr/textAppearanceMedium"
    />
```

```
        <CheckBox
          android:id="@+id/download_this"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_alignBottom="@+id/download_app_name"
          android:layout_alignParentRight="true"
          android:layout_marginRight="30dp" />

</RelativeLayout>
```

Filename: launcher.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >

  <TextView
    android:id="@+id/launcher_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
              android:drawablePadding="15dp"
              android:drawableLeft="@drawable/ic_launcher"
              android:gravity="center|center"
    android:text="@string/launcher_text"
    android:textAppearance="?android:attr/textAppearanceLarge" />
</RelativeLayout>
```

Filename: activity_installer.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/menu_settings"
    android:title="@string/menu_settings"
    android:orderInCategory="100"
    android:showAsAction="never" />
</menu>
```

Filename: strings.xml

```
<resources>

   <string name="app_name">Verti Arcade</string>
   <string name="installer_status">Installer Status</string>
   <string name="menu_settings">Settings</string>
   <string name="title_activity_installer">Verti Prototype</string>
   <string name="title_activity_arcade">Verti Arcade</string>
   <string name="submit">Submit</string>
   <string name="enter_url">Enter URL</string>
   <string name="remind_me_later">Reminde Me Later</string>
   <string name="default_app_text">App is ready to </string>
   <string name="run">Run</string>
   <string name="back">Back</string>
   <string name="launcher_text">Verti Arcade</string>
   <string name="nothing_to_download">No Apps in the Download Queue</string>
   <string name="install">Install</string>

</resources>
```

Filename: styles.xml

```
<resources>

   <style name="AppTheme" parent="android:Theme.Light" />

</resources>
```

Filename: styles.xml

```
<resources>

   <style name="AppTheme" parent="android:Theme.Holo.Light" />

</resources>
```

Filename: styles.xml

```
<resources>
```

```
    <style name="AppTheme" parent="android:Theme.Holo.Light.DarkActionBar" />

</resources>
```

Filename: Arcade.java

```
package com.verti.prototype;

import java.util.List;
import java.util.Timer;
import java.util.TimerTask;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.ActivityManager;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager.NameNotFoundException;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.verti.prototype.classes.App;
import com.verti.prototype.utils.Constants;
import com.verti.prototype.utils.DatabaseHandler;
import com.verti.prototype.utils.Logger;

@SuppressLint("SetJavaScriptEnabled")
public class Arcade extends Activity {
        private final static String TAG="Arcade";
        private DatabaseHandler dbh;
        private Button mRunButton;
        private Button mRemindMeButton;
```

```
    private TextView mAppLaunchText;
    private Button mBackButton;
    private WebView mWebView;
    private App mApp;
    private Timer timer;


@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.arcade);
  dbh = DatabaseHandler.getInstance(this);
  mApp = dbh.getApp(getIntent().getStringExtra(DatabaseHandler.APP_URI));
  mApp.printMe();
  mAppLaunchText = (TextView) findViewById(R.id.app_ready_text);
  mRunButton = (Button) findViewById(R.id.run);
  mRemindMeButton = (Button) findViewById(R.id.remind_me);
  mBackButton = (Button) findViewById(R.id.back);
  mRunButton.setOnClickListener(listener);
  mRemindMeButton.setOnClickListener(listener);
  mBackButton.setOnClickListener(listener);

  mWebView = (WebView) findViewById(R.id.reco_webview);
  mWebView.getSettings().setJavaScriptEnabled(true);
  mWebView.addJavascriptInterface(new VertiJsBridge(this), "vertiArcade");
  //mWebView.loadUrl("http://brookemaury.org/rec.html"); /* testing */
  mWebView.loadUrl("http://www.sunpeaksfun.com/rec.html");
  try {
                    //mPackageInfo = getPackageManager().getPackageInfo(mApp.getUri(),
PackageManager.GET_META_DATA);
                    Drawable img = getPackageManager().getApplicationIcon(mApp.getUri());
          mAppLaunchText.setCompoundDrawablesWithIntrinsicBounds( img, null, null, null );
          mAppLaunchText.setText(mApp.getName() + " is ready to ");
            } catch (NameNotFoundException e) {
                    Toast.makeText(this, "App was found!", Toast.LENGTH_SHORT).show();
            }
}

public void onStart() {
      super.onStart();
}
```

```
  private OnClickListener listener = new OnClickListener() {
                @Override
                public void onClick(View view) {
                        switch (view.getId()) {
                        case R.id.run:
                                mApp.setInstalled(true);
                                dbh.setInstalled(mApp);
                                Intent runApp = new
Intent(Arcade.this.getPackageManager().getLaunchIntentForPackage(mApp.getUri()));
                                startActivity(runApp);
                                finish();
                                break;
                        case R.id.remind_me:
                                relaunchTimer();
                                Toast.makeText(Arcade.this, "We'll remind you again in " +
Constants.REMINDER_INTERVAL_MILLIS/1000 + " seconds.", Toast.LENGTH_LONG).show();
                                finish();
                                break;
                        case R.id.back:
                                if (timer != null) timer.cancel();
                                finish();
                                break;
                        }
                }
  };

  /**
   * might need some work.
   */

  private void relaunchTimer() {
                timer = new Timer();
                TimerTask task = new TimerTask() {
              public void run() {
                ActivityManager am = (ActivityManager)
Arcade.this.getSystemService(Context.ACTIVITY_SERVICE);
                // get the info from the currently running task
                List<ActivityManager.RunningTaskInfo> taskInfo = am.getRunningTasks(1);
                ComponentName componentInfo = taskInfo.get(0).topActivity;
                if
(!componentInfo.getPackageName().equals(Arcade.this.getApplicationInfo().packageName)) {
                        Intent relaunchApp = new Intent(Arcade.this, Arcade.class);
```

```
                        relaunchApp.putExtra(DatabaseHandler.APP_URI, mApp.getUri());
                        relaunchApp.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_REORDER_TO_FRONT | Intent.FLAG_ACTIVITY_NEW_TASK);
                        Arcade.this.startActivity(relaunchApp);
                    }
                }
            };

            //timer.schedule(task, Constants.REMINDER_INTERVAL_MILLIS,
Constants.REMINDER_INTERVAL_MILLIS);
            //run only once now.
            timer.schedule(task, Constants.REMINDER_INTERVAL_MILLIS);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_installer, menu);
        return true;
    }

    final class VertiJsBridge {
        Context mContext;
        VertiJsBridge(Context context) {
            mContext = context;
        }
        public void addToQueue(String appName, String appPackage, String appIconURL) {
            Logger.d(TAG, "addToQueue: appName: " + appName + " appPackage: " + appPackage +
" iconString: " + appIconURL);
            App appToAdd = new App();
            appToAdd.setName(appName);
            appToAdd.setUri(appPackage);
            appToAdd.setIconPath(appIconURL);
            //appToAdd.setInstalled(false);
            appToAdd.rememberMe(mContext);
        }
        public void removeFromQueue(String appPackage) {
            Logger.d(TAG, "removeFromQueue: appPackage: " + appPackage);
            App appToForget = new App(appPackage);
            appToForget.forgetMe(mContext);
        }

        public void sendList(String[] apps) {
```

```
                for (final String appUri : apps) {
                        if (dbh.getApp(appUri).isInstalled()) {
                                mWebView.post(new Runnable() {
                                        public void run() {
                                        mWebView.loadUrl("javascript:hideApp('" + appUri + "')");
                                        }
                                });
                        }
                }
        }
  }
}
```

Filename: DownloadQueue.java

```
package com.verti.prototype;

import java.util.ArrayList;

import android.app.ListActivity;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

import com.verti.prototype.utils.Constants;
import com.verti.prototype.utils.DatabaseHandler;
import com.verti.prototype.utils.DownloadAdapter;

public class DownloadQueue extends ListActivity {
        private DatabaseHandler dbh;
        private Button mButton;
        private ArrayList<String> appsToInstall = new ArrayList<String>();

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.download_queue);
```

```
    mButton = (Button) findViewById(R.id.install_button);
    mButton.setOnClickListener(listener);
    dbh = DatabaseHandler.getInstance(this);
    if (dbh.getAppsForDownload().size() == 0) finish();
    else getApps();
  }


  public void onStart() {
        super.onStart();
        checkAndInstall();
  }

  private OnClickListener listener = new OnClickListener() {
                @Override
                public void onClick(View arg0) {
                        checkAndInstall();
                }
  };

  @SuppressWarnings("deprecation")
        private void getApps () {
        Cursor cursor = dbh.getAppsForDownloadInCursor();
        startManagingCursor(cursor);
        String[] from = new String[] { DatabaseHandler.APP_URI };
        int[] to = new int[] { R.id.download_app_name };
        DownloadAdapter apps =
                        new DownloadAdapter(this, R.layout.download_row, cursor, from, to);
        setListAdapter(apps);
  }

  private void checkAndInstall() {
        appsToInstall = dbh.getDownloadQueue();
        if (appsToInstall != null && appsToInstall.size() > 0) {
                String getThisUri = appsToInstall.get(0);
                Intent getApp = null;
                        getApp = new Intent(Intent.ACTION_VIEW,Uri.parse(Constants.MARKET_URL +
getThisUri));
                        getApp.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                        startActivity(getApp); //TODO: for result?
        }
  }
```

```
    public void onPause() {
        super.onPause();

    }

}
```

Filename: Installer.java

```
package com.verti.prototype;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.List;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.verti.prototype.classes.App;
import com.verti.prototype.utils.FindRunningApps;
import com.verti.prototype.utils.Logger;

public class Installer extends Activity {
        private final static String TAG="Installer";
        private EditText mEnterUrl;
```

```
        private Button mSubmitButton;
        private FindRunningApps mFindApps;
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_installer);
          new CollectAppInfo().execute(this);
      SharedPreferences prefs = this.getSharedPreferences(
              "com.verti.prototype", Context.MODE_PRIVATE);

      if (prefs.getBoolean("first", true)) {
          //dbh.createTestData();
      }
      Editor ed = prefs.edit();
      ed.putBoolean("first", false);
      ed.commit();
      mEnterUrl = (EditText) findViewById(R.id.url);
      mSubmitButton = (Button) findViewById(R.id.submit);
      mSubmitButton.setOnClickListener(submitListener);
    }

    public void onStart() {
          super.onStart();

    }

    private OnClickListener submitListener = new OnClickListener() {
          String urlString;
                @Override
                public void onClick(View view) {
                      try {
                            urlString = (mEnterUrl.getText() != null) ?
mEnterUrl.getText().toString().toLowerCase() : "";
                            if (!urlString.startsWith("http://") && !urlString.startsWith("https://"))
urlString = "http://" + urlString;
                            @SuppressWarnings("unused")
                            URL url = new URL (urlString);
                            Intent launchUrl = new Intent(Intent.ACTION_VIEW,
Uri.parse(urlString));
                            startActivity(launchUrl);
                      } catch (MalformedURLException e) {
```

```
                                Toast.makeText(Installer.this, urlString + " is not a valid URL",
Toast.LENGTH_LONG).show();
                        }
                }


  };

  /*
  private void checkAndInstall() {
        dbh = DatabaseHandler.getInstance(this);
        List<App> apps = dbh.getAppsForDownload();
        if (apps.size() == 0) return; //nothing to do.
        App app = apps.get(0);
        if (app != null) {
                Intent getApp = null;
                getApp = new Intent(Intent.ACTION_VIEW,Uri.parse(Constants.MARKET_URL +
app.getUri()));
                startActivity(getApp); //TODO: for result?
        }
  }
  */

  private class CollectAppInfo extends AsyncTask<Context, Integer, Integer> {
        List<ApplicationInfo> packageAppsList;
        protected Integer doInBackground(Context ...contexts) {
        final Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
        mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
        final PackageManager pm = contexts[0].getPackageManager();
        packageAppsList  = pm.getInstalledApplications(PackageManager.GET_META_DATA);
        App app;
        for (ApplicationInfo appInfo : packageAppsList) {
                try {
                                app = new App(pm.getPackageInfo(appInfo.packageName,
PackageManager.GET_META_DATA), contexts[0]);
                        app.rememberMe(contexts[0]);
                        //app.printMe();
                        } catch (NameNotFoundException e) {
                                //this isn't an app i guess
                        }
        }
```

```
            return packageAppsList.size();
        }

        protected void onProgressUpdate(Integer... progress) {

        }

        protected void onPostExecute(Integer result) {
            mFindApps = new FindRunningApps(Installer.this);
            mFindApps.start();
            Logger.d(TAG, "App Info DB updated with " + result + " apps");
             Toast.makeText(Installer.this, result + " apps installed.", Toast.LENGTH_LONG).show();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_installer, menu);
        return true;
    }
}
```

Filename: IntentReceiver.java

```
package com.verti.prototype;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;

import com.verti.prototype.classes.App;
import com.verti.prototype.classes.HomeScreenChecker;
import com.verti.prototype.utils.DatabaseHandler;
import com.verti.prototype.utils.Logger;

public class IntentReceiver extends BroadcastReceiver {
        private final static String TAG = "IntentReceiver";
        private final static String REFERRER = "referrer";
        private DatabaseHandler dbh;
```

```
private final static String INSTALL_REFERRER="com.android.vending.INSTALL_REFERRER";
private String referrer;

@Override
public void onReceive(Context context, Intent intent) {
        Logger.d(TAG, "incoming: " + intent.getAction());
        dbh = DatabaseHandler.getInstance(context);
        if (INSTALL_REFERRER.equals(intent.getAction())) {
                referrer = intent.getStringExtra(REFERRER);
                //Logger.d(TAG, referrer);
                //TODO: place operations in a service

                String[] apps = referrer.split("&");
                for (String app : apps) {
                        String[] appUris = app.split("=");
                        dbh.addApp(new App(appUris[1]));
                }
                dbh.printApps();
        }

        /* Rules for what to display when a new app is displayed:
         * 1. Only if the criteria for "isHomeScreenOnTop" are satisfied will we show anything
         * 2. If there are more apps to install, show the download queue
         * 3. If not, show the "Arcade" view.
         */

        else if (Intent.ACTION_PACKAGE_ADDED.equals(intent.getAction())) {
                String appUri = intent.getData().getSchemeSpecificPart();

                        App app;
                        try {
                                app = new
App(context.getPackageManager().getPackageInfo(appUri, PackageManager.GET_META_DATA),
context);
                        } catch (NameNotFoundException e) {
                                app = new App(appUri);
                        }
                app.setInstalled(true);
                dbh.removeFromDownloadQueue(app.getUri()); /* removes from the download
queue */
                dbh.addApp(app);
                Logger.d(TAG, "app: " + appUri + " installed: " + app.isInstalled());
```

```
                        if (HomeScreenChecker.isHomeScreenOnTop(context)) {
                                if (dbh.moreAppsToInstall()) {
                                        Intent bringToFront = new Intent(context,
DownloadQueue.class);
                                        bringToFront.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_REORDER_TO_FRONT | Intent.FLAG_ACTIVITY_NEW_TASK);
                                        context.startActivity(bringToFront);
                                }
                                else {
                                        Intent bringToFront = new Intent(context, Arcade.class);
                                        bringToFront.putExtra(DatabaseHandler.APP_URI, appUri);
                                        bringToFront.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_REORDER_TO_FRONT | Intent.FLAG_ACTIVITY_NEW_TASK);
                                        context.startActivity(bringToFront);
                                }
                        }
                }
                else if (Intent.ACTION_PACKAGE_REMOVED.equals(intent.getAction())) {
                String appUri = intent.getData().getSchemeSpecificPart();

                        App app;
                        try {
                                app = new App(context.getPackageManager().getPackageInfo(appUri,
PackageManager.GET_META_DATA), context);
                        } catch (NameNotFoundException e) {
                                app = new App(appUri);
                        }
                app.setInstalled(true);
                dbh.removeFromDownloadQueue(app.getUri()); /* removes from the download queue
*/
                dbh.deleteApp(app);
                Logger.d(TAG, "app: " + appUri + " installed: " + app.isInstalled());
        }
        }
}
```

Filename: Launcher.java

```
package com.verti.prototype;

import java.util.List;
```

```
import java.util.Timer;
import java.util.TimerTask;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Menu;
import android.widget.Toast;

import com.verti.prototype.classes.App;
import com.verti.prototype.utils.Constants;
import com.verti.prototype.utils.FindRunningApps;
import com.verti.prototype.utils.Logger;

public class Launcher extends Activity {
        private final static String TAG="Launcher";
        private FindRunningApps mFindApps;
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.launcher);
        new CollectAppInfo().execute(this);

   }

   public void onStart() {
        super.onStart();
                Timer timer = new Timer();
                TimerTask task = new TimerTask() {
            public void run() {
                Intent launchQueue = new Intent(Launcher.this, DownloadQueue.class);
//              Intent launchQueue = new Intent(Launcher.this, Arcade.class);
//              launchQueue.putExtra(DatabaseHandler.APP_URI, "com.twitter.android");
                startActivity(launchQueue);
                finish();
            }
          };
```

```
            timer.schedule(task, Constants.SHOW_SPLASH_MILLIS);
    }

    private class CollectAppInfo extends AsyncTask<Context, Integer, Integer> {
            List<ApplicationInfo> packageAppsList;
            protected Integer doInBackground(Context ...contexts) {
            final Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
            mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
            final PackageManager pm = contexts[0].getPackageManager();
            packageAppsList  = pm.getInstalledApplications(PackageManager.GET_META_DATA);
            App app;
            for (ApplicationInfo appInfo : packageAppsList) {
                    try {
                                    app = new App(pm.getPackageInfo(appInfo.packageName,
PackageManager.GET_META_DATA), contexts[0]);
                            app.rememberMe(contexts[0]);
                            } catch (NameNotFoundException e) {
                                    //this isn't an app i guess
                            }
            }

            return packageAppsList.size();
        }

        protected void onProgressUpdate(Integer... progress) {

        }

        protected void onPostExecute(Integer result) {
          mFindApps = new FindRunningApps(Launcher.this);
          mFindApps.start();
          Logger.d(TAG, "App Info DB updated with " + result + " apps");
           Toast.makeText(Launcher.this, result + " apps installed.", Toast.LENGTH_LONG).show();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.activity_installer, menu);
      return true;
    }
}
```

Filename: App.java

```
package com.verti.prototype.classes;

import android.content.Context;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;

import com.verti.prototype.utils.DatabaseHandler;
import com.verti.prototype.utils.Logger;

/**
 * Holds data about the apps to be downloaded
 * @author bmaury
 *
 */

public class App {
	private final static String TAG="App";
	private String uri;
	private String name;
	private boolean installed;
	private boolean systemApp;
	private String iconPath;
	private long installTime;
	private boolean hasRun;

	public App() {}

	public App(String uri, String name, boolean installed) {
		this.uri = uri;
		this.name = name;
		this.installed = installed;
	}
	public App(String uri) {
		this.uri = uri;
		this.name = "";
		this.installed = false;
	}
```

```
	public App(PackageInfo packageInfo, Context context) {
		PackageManager pm = context.getPackageManager();
		this.uri = packageInfo.applicationInfo.packageName;
		this.name = packageInfo.applicationInfo.loadLabel(pm).toString();
		this.installed = true;
		this.systemApp = ((packageInfo.applicationInfo.flags & ApplicationInfo.FLAG_SYSTEM)
!= 0) ? true : false;
	}

	/*
	public App(String uri, String appName, String imageUrl) {
		this.uri = uri;
		this.name = appName;
		this.iconPath = imageUrl;
	}
	*/

	public String getUri() {
		return uri;
	}
	public void setUri(String uri) {
		this.uri = uri;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public boolean isInstalled() {
		return installed;
	}
	public void setInstalled(boolean installed) {
		this.installed = installed;
	}

	public boolean isSystemApp() {
		return systemApp;
	}

	public void setSystemApp(boolean systemApp) {
```

```
        this.systemApp = systemApp;
}

public String getIconPath() {
        return iconPath;
}

public void setIconPath(String iconPath) {
        this.iconPath = iconPath;
}

/**
 * stores this apps metadata in the database
 * @param context
 */

public void rememberMe(Context context) {
        DatabaseHandler dbh =  DatabaseHandler.getInstance(context);
        dbh.addApp(this);
}

public void forgetMe(Context context) {
        DatabaseHandler dbh =  DatabaseHandler.getInstance(context);
        dbh.deleteApp(this);
}

public void printMe() {
        Logger.d(TAG, "uri: " + this.getUri()
                                + " name: " + this.getName()
                                +  " system: " + this.isSystemApp()
                                + " icon path: " + this.getIconPath()
                                + " installed: " + this.isInstalled()
                                );
}

public long getInstallTime() {
        return installTime;
}

public void setInstallTime(long installTime) {
        this.installTime = installTime;
}
```

```
	public boolean hasRun() {
		return hasRun;
	}

	public void setHasRun(boolean hasRun) {
		this.hasRun = hasRun;
	}
}
```

Filename: HomeScreenChecker.java

```
package com.verti.prototype.classes;

import android.content.Context;

public class HomeScreenChecker {

	//private final static String[] homeScreens = new String[] { "com.android.launcher",
"com.motorola.blur.home"};
	public static boolean isHomeScreenOnTop(Context context) {
	return true;
		/*
		ActivityManager am = (ActivityManager)
context.getSystemService(Context.ACTIVITY_SERVICE);
	// get the info from the currently running task
	List<ActivityManager.RunningTaskInfo> taskInfo = am.getRunningTasks(1);
	ComponentName componentInfo = taskInfo.get(0).topActivity;
	for (String homeScreen : homeScreens) {
		if (homeScreen.equals(componentInfo.getPackageName())) {
			return true;
		}
	}
	return false;
		*/
	}


}
```

Filename: CheckHomeScreen.java

```
package com.verti.prototype.utils;

import java.util.Timer;
import java.util.TimerTask;

import android.content.Context;
import android.content.Intent;

import com.verti.prototype.classes.HomeScreenChecker;

public class CheckHomeScreen extends Thread {
  private final static String TAG = "CheckHomeScreen";
        private Context mContext;

  public CheckHomeScreen(Context context) {
        this.mContext=context;
        this.setName(TAG);
  }

        @Override
        public void run() {
                Timer timer = new Timer();
                TimerTask task = new TimerTask() {
            public void run() {
                if (HomeScreenChecker.isHomeScreenOnTop(mContext)) {
                        //Intent startVerti = new
Intent(mContext.getPackageManager().getLaunchIntentForPackage(mContext.getApplicationInfo().pack
ageName));
                        Intent startVerti = new
Intent(mContext.getPackageManager().getLaunchIntentForPackage(mContext.getApplicationInfo().pack
ageName));
                        startVerti.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                        mContext.startActivity(startVerti);
                }
              }
            };
            timer.schedule(task, 100, 15000); //15 seconds
        }
}
```

Filename: Constants.java

```
package com.verti.prototype.utils;

public class Constants {
        public final static String MARKET_URL="http://market.android.com/details?id=";
        public final static int REMINDER_INTERVAL_MILLIS=15000;
        public final static int SHOW_SPLASH_MILLIS=3000;
}
```

Filename: DatabaseHandler.java

```
package com.verti.prototype.utils;

import java.util.ArrayList;
import java.util.List;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

import com.verti.prototype.classes.App;


public class DatabaseHandler extends SQLiteOpenHelper {
        private final static String DATABASE_NAME="verti_apps";
        private final static String APP_INFO_TABLE="application_info";
        public final static String APP_URI="app_uri"; /* used in other classes for querying DB */
        public final static String APP_NAME="app_name";
        private final static String APP_INSTALLED="installed";
        private final static String IS_SYSTEM_APP="system_app";
        private final static String APP_ICON_PATH="app_icon_path";
        private final static String APP_INSTALL_TIME="install_time";
        private final static String APP_HAS_RUN="has_run";
        private final static int DATABASE_VERSION = 1;
        private final static String TAG="DatabaseHandler";
```

```
    private final static String DOWNLOAD_QUEUE_TABLE ="download_queue";

    private static DatabaseHandler instance = null;

     public static DatabaseHandler getInstance(Context context) {
        if(instance == null) {
          instance = new DatabaseHandler(context);
        }
        return instance;
     }

    protected DatabaseHandler(Context context) {
         super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

     public DatabaseHandler(Context context, String name, CursorFactory factory,
                    int version) {
            super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
            String CREATE_TRACKER_TABLE="CREATE TABLE " + APP_INFO_TABLE +
                            " (" + APP_URI + " varchar(255), " +
                            APP_NAME + " varchar(255)," +
                            APP_INSTALLED + " int DEFAULT 0," +
                            IS_SYSTEM_APP + " int DEFAULT 0, " +
                            APP_ICON_PATH + " varchar(255), " +
                            APP_INSTALL_TIME + " long DEFAULT 0, " +
                            APP_HAS_RUN + " int DEFAULT 0, " +
                            "PRIMARY KEY (" + APP_URI + ")" +
                            ")";

            String CREATE_DOWNLOAD_QUEUE_TABLE="CREATE TABLE " +
DOWNLOAD_QUEUE_TABLE +
                            " (" + APP_URI + " varchar(255)," +
                            "PRIMARY KEY (" + APP_URI + ")" +
                            ")";

            db.execSQL(CREATE_TRACKER_TABLE);
            db.execSQL(CREATE_DOWNLOAD_QUEUE_TABLE);
    }
```

```
    @Override
    public void onUpgrade(SQLiteDatabase db, int arg1, int arg2) {
  // Drop older table if existed
  db.execSQL("DROP TABLE IF EXISTS " + APP_INFO_TABLE);
  // Create tables again
  onCreate(db);
    }


    public void addApp(App app) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(APP_URI, app.getUri());
        values.put(APP_NAME, app.getName());
        values.put(APP_INSTALLED, (app.isInstalled()) ? 1 : 0);
        values.put(IS_SYSTEM_APP, (app.isSystemApp()) ? 1 : 0);
        values.put(APP_ICON_PATH, (app.getIconPath()));
        values.put(APP_INSTALL_TIME, (app.getInstallTime()));
        db.insertWithOnConflict(APP_INFO_TABLE, null, values,
SQLiteDatabase.CONFLICT_REPLACE);
    }

    public void addApps(List<App> apps) {
        for (App app : apps) {
            addApp(app);
        }
    }

    public void updateApp(App app) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(APP_URI, app.getUri());
        values.put(APP_NAME, app.getName());
        values.put(APP_INSTALLED, (app.isInstalled()) ? 1 : 0);
        printApps();
        db.updateWithOnConflict(
                APP_INFO_TABLE,
                values,
                APP_URI + "= ?",
                new String[] {app.getUri()},
                SQLiteDatabase.CONFLICT_IGNORE);
```

```
}

public void deleteApp(App app) {
        SQLiteDatabase db = this.getWritableDatabase();
  db.delete(APP_INFO_TABLE, APP_URI + " = ?",
        new String[] { app.getUri() });
}

/* for testing */
public void dropAndCreateTable() {
        String query = "DROP TABLE IF EXISTS " + APP_INFO_TABLE;
        SQLiteDatabase db = this.getWritableDatabase();
        db.execSQL(query);
        onCreate(db);
}

public void printApps() {
        List<App> apps = getApps();
        for (App app : apps) {
                Logger.d(TAG, app.getUri() + " " + app.isInstalled());
        }
}

public List<App> getApps() {
        List<App> apps = new ArrayList<App>();
        String query = "SELECT * FROM " + APP_INFO_TABLE;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(query, null);
        if (cursor.moveToFirst()) {
                while (!cursor.isAfterLast()) {
                        App app = new App(
                                        cursor.getString(0),
                                        cursor.getString(1),
                                        (cursor.getInt(2) == 1) ? true : false
                                );
                        apps.add(app);
                        cursor.moveToNext();
                }
        }
        return apps;
}
```

```
	public Cursor getAppsForDownloadInCursor() {
		String query = "SELECT *, " + APP_URI + " as _id FROM " + APP_INFO_TABLE + " WHERE
installed < 1";
		SQLiteDatabase db = this.getReadableDatabase();
		Cursor c = db.rawQuery(query, null);
		return c;
	}

	public List<App> getAppsForDownload() {
		List<App> apps = new ArrayList<App>();
		String query = "SELECT * FROM " + APP_INFO_TABLE + " WHERE installed < 1";
		SQLiteDatabase db = this.getReadableDatabase();
		Cursor cursor = db.rawQuery(query, null);
		if (cursor.moveToFirst()) {
			while (!cursor.isAfterLast()) {
				App app = new App(
							cursor.getString(0),
							cursor.getString(1),
							(cursor.getInt(2) == 1) ? true : false
						);
				apps.add(app);
				cursor.moveToNext();
			}
		}
		return apps;
	}

	public App getApp(String appUri) {
		SQLiteDatabase db = this.getReadableDatabase();
		Cursor cursor = db.query(
				APP_INFO_TABLE,
				new String[] {APP_URI, APP_NAME, APP_INSTALLED,
APP_INSTALL_TIME},
				APP_URI + "=?",
				new String[] {appUri},
				null,
				null,
				null);
		App app = new App();
		if (cursor.moveToFirst()) {
				app = new App(
							cursor.getString(0),
```

```
                                        cursor.getString(1),
                                        (cursor.getInt(2) == 1) ? true : false
                                );
                }
        return app;
}

/* TODO: Make app not string/boolean */
public void setHasRun(String appUri, boolean hasRun) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(APP_URI, appUri);
        values.put(APP_HAS_RUN, (hasRun) ? 1 : 0);
        db.updateWithOnConflict(
                        APP_INFO_TABLE,
                        values,
                        APP_URI + "= ?",
                        new String[] {appUri},
                        SQLiteDatabase.CONFLICT_IGNORE);
        //db.close();
}

public void setInstalled(App app) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(APP_URI, app.getUri());
        values.put(APP_INSTALLED, (app.isInstalled()) ? 1 : 0);
        db.updateWithOnConflict(
                        APP_INFO_TABLE,
                        values,
                        APP_URI + "= ?",
                        new String[] {app.getUri()},
                        SQLiteDatabase.CONFLICT_IGNORE);
        //db.close();
}

public boolean moreAppsToInstall() {
        String query = " select count(*) as count from " + APP_INFO_TABLE +
                        " where installed < 1 OR installed is NULL";
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(query, null);
        if (cursor.moveToFirst()) {
```

```
                    if (cursor.getInt(0) > 0) return true;
                }
                return false;
        }

        public void addToDownloadQueue(String uri) {
                SQLiteDatabase db = this.getWritableDatabase();
                ContentValues values = new ContentValues();
                values.put(APP_URI, uri);
                Logger.d(TAG, "adding: " + uri);
                db.insertWithOnConflict(DOWNLOAD_QUEUE_TABLE, null, values,
SQLiteDatabase.CONFLICT_REPLACE);
        }

        public void removeFromDownloadQueue(String uri) {
                SQLiteDatabase db = this.getWritableDatabase();
                db.delete(DOWNLOAD_QUEUE_TABLE, APP_URI + " = ?", new String[]{ uri });
        }

        public ArrayList<String> getDownloadQueue() {
                ArrayList<String> appQueue = new ArrayList<String>();
                String query = "SELECT * FROM " + DOWNLOAD_QUEUE_TABLE;
                SQLiteDatabase db = this.getReadableDatabase();
                Cursor cursor = db.rawQuery(query, null);
                if (cursor.moveToFirst()) {
                        while (!cursor.isAfterLast()) {
                                Logger.d(TAG, cursor.getString(0));
                                appQueue.add(cursor.getString(0));
                                cursor.moveToNext();
                        }
                }
                return appQueue;
        }

        public void createTestData() {
                dropAndCreateTable();
                String query = "INSERT INTO " + APP_INFO_TABLE +
                                " SELECT 'com.twitter.android' as " + APP_URI + ", 'twitter' as " +
APP_NAME + ", 0 as " + APP_INSTALLED +
                                " UNION SELECT 'com.evernote','evernote', 0 " +
                                " UNION SELECT 'com.alphonso.pulse','pulse', 0";
                SQLiteDatabase db = this.getWritableDatabase();
```

```
                db.execSQL(query);
                //db.close();
        }
}
```

Filename: DownloadAdapter.java

```
package com.verti.prototype.utils;

import android.content.Context;
import android.database.Cursor;
import android.support.v4.widget.SimpleCursorAdapter;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.TextView;

import com.verti.prototype.R;

public class DownloadAdapter extends SimpleCursorAdapter {
        private Cursor items;
        private Context mContext;
        private RemoteDrawableManager mDrawableManager;
        private DatabaseHandler mDbh;

        public DownloadAdapter(Context context, int layout, Cursor cursor,
                        String[] from, int[] to) {
                super(context, layout, cursor, from, to, 0);
                items = cursor;
                mContext = context;
                mDrawableManager = new RemoteDrawableManager(context);
                mDbh = DatabaseHandler.getInstance(context);
        }

        @Override
  public View getView(int position, View convertView, ViewGroup parent) {
    if(convertView == null)
      convertView = View.inflate(mContext,R.layout.download_row, null);
    View row = convertView;
```

```
    items.moveToPosition(position);
    final String uri = items.getString(0);
    final String appName = items.getString(1);
    final String iconUri = items.getString(4);
    TextView mAppLaunchText = (TextView) convertView.findViewById(R.id.download_app_name);
    mAppLaunchText.setText((appName.isEmpty()) ? uri : appName);
    mDrawableManager.fetchDrawableOnThread(iconUri, mAppLaunchText);
    final CheckBox checkBox = (CheckBox) convertView.findViewById(R.id.download_this);
    checkBox.setOnCheckedChangeListener(new OnCheckedChangeListener() {
                        @Override
                        public void onCheckedChanged(CompoundButton button, boolean isChecked) {
                                if (isChecked) {
                                        mDbh.addToDownloadQueue(uri);
                                }
                                else mDbh.removeFromDownloadQueue(uri);
                        }

    });
    decideCheckStatus(checkBox, uri);
    return row;
  }

        private void decideCheckStatus(CheckBox checkBox, String uri) {
                if (mDbh.getDownloadQueue().contains(uri)) checkBox.setSelected(true);

        }

}
```

Filename: FindRunningApps.java

```
package com.verti.prototype.utils;

import java.util.List;
import java.util.Timer;
import java.util.TimerTask;

import android.app.ActivityManager;
import android.app.ActivityManager.RunningAppProcessInfo;
import android.content.Context;
```

```
public class FindRunningApps extends Thread {
  private final static String TAG = "FindRunningApps";
        private Context context;
  private DatabaseHandler dbh;
  public FindRunningApps(Context context) {
        this.context=context;
        this.setName(TAG);
  }

        @Override
        public void run() {
                Timer timer = new Timer();
                TimerTask task = new TimerTask() {
            public void run() {
                        ActivityManager activityManager = (ActivityManager)
context.getSystemService(Context.ACTIVITY_SERVICE);
                        List<RunningAppProcessInfo> runningApps =
activityManager.getRunningAppProcesses();
                        for (RunningAppProcessInfo runningApp : runningApps ) {
                                dbh = DatabaseHandler.getInstance(context);
                                dbh.setHasRun(runningApp.processName, true);
                        }
              }
            };

            timer.schedule(task, 100, 30000);
        }

}
```

Filename: Logger.java

```
package com.verti.prototype.utils;

import android.util.Log;

public class Logger {
        public static void d(String tag, String message) {
                Log.d(tag, message);
        }
}
```

Filename: RemoteDrawableManager.java

```
package com.verti.prototype.utils;

/*
Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements.  See the NOTICE file
distributed with this work for additional information
regarding copyright ownership.  The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License.  You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied.  See the License for the
specific language governing permissions and limitations
under the License.
*/
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.util.HashMap;
import java.util.Map;

import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;

import android.content.Context;
import android.graphics.drawable.Drawable;
import android.os.Handler;
import android.os.Message;
import android.widget.TextView;

public class RemoteDrawableManager {
  private final Map<String, Drawable> drawableMap;
```

```
public RemoteDrawableManager(Context context) {
  drawableMap = new HashMap<String, Drawable>();
}

public Drawable fetchDrawable(String urlString) {
  if (drawableMap.containsKey(urlString)) {
    return drawableMap.get(urlString);
  }

  Logger.d(this.getClass().getSimpleName(), "image url:" + urlString);
  try {
    InputStream is = fetch(urlString);
    Drawable drawable = Drawable.createFromStream(is, "src");


    if (drawable != null) {
      drawableMap.put(urlString, drawable);
      Logger.d(this.getClass().getSimpleName(), "got a thumbnail drawable: " + drawable.getBounds()
+ ", "
          + drawable.getIntrinsicHeight() + "," + drawable.getIntrinsicWidth() + ", "
          + drawable.getMinimumHeight() + "," + drawable.getMinimumWidth());
    } else {

    }

    return drawable;
  } catch (MalformedURLException e) {

    return null;
  } catch (IOException e) {

    return null;
  }
}

public void fetchDrawableOnThread(final String urlString, final TextView textView) {

      if (drawableMap.containsKey(urlString)) {
        textView.setCompoundDrawablesWithIntrinsicBounds( drawableMap.get(urlString), null,
null, null );
      //imageView.setImageDrawable(drawableMap.get(urlString));
```

```
    }

    final Handler handler = new Handler() {
      @Override
      public void handleMessage(Message message) {
            textView.setCompoundDrawablesWithIntrinsicBounds( (Drawable) message.obj, null, null,
null );
      }
    };

    Thread thread = new Thread() {
      @Override
      public void run() {
        //TODO : set imageView to a "pending" image
        Drawable drawable = fetchDrawable(urlString);
        Message message = handler.obtainMessage(1, drawable);
        handler.sendMessage(message);
      }
    };
    thread.start();
  }

  private InputStream fetch(String urlString) throws MalformedURLException, IOException {
    DefaultHttpClient httpClient = new DefaultHttpClient();
    HttpGet request = new HttpGet(urlString);
    HttpResponse response = httpClient.execute(request);
    return response.getEntity().getContent();
  }
}
```

Claims

1. A system for displaying an advertisement, the system comprising:

a mobile computing device comprising a processor and a memory;

an application module coupled to the processor, the application module constructed and arranged to identify at least one application of the mobile computing device and store a database including at least one listing of the at least one application in the memory;

a queue module coupled to the processor, the queue module constructed and arranged to build a queue of the at least one application from the at least one listing in the database; and

a listener module coupled to the processor, the listener module constructed and arranged to:

identify at least one broadcast event generated by the processor and associated with the at least one installed application, and

display at least one advertisement in response to the at least one broadcast event.

2. The system of claim 1 or any claim herein, further comprising a main module coupled to the processor, the main module constructed and arranged to manage the application module, the queue module, the listener module, a rules module coupled to the processor, or any combination thereof.

3. The system of claim 1 or any claim herein, wherein the queue module is constructed and arranged to build the queue by determining whether the at least one application should be displayed in an interface.

4. The system of claim 3 or any claim herein, wherein the determination is based on whether the at least one application has been run.

5. The system of claim 3 or any claim herein, wherein the determination is based on whether the at least one application has been installed.

6. The system of claim 1 or any claim herein, further comprising a rules module coupled to the processor, the rules module constructed and arranged to evaluate the at least one broadcast event against at least one rule to determine whether the at least one advertisement can be displayed, to determine a characteristic of the at least one advertisement, or a combination thereof.

7. The system of claim 6 or any claim herein, wherein the rule comprises whether it is permissible to display information full screen on the device, whether the advertisement may be displayed by interrupting control of the device, whether the advertisement may be displayed while the device is being used as a phone, whether the advertisement may be displayed while the device is in a screen lock mode, or any combination thereof.

8. The system of claim 1 or any claim herein, wherein the broadcast event is a PACKAGE_INSTALL.

9. The system of claim 1 or any claim herein, wherein the broadcast event is an operating system broadcast intent

10. The system of claim 1 or any claim herein, wherein a user interaction with the at least one advertisement is stored in the memory.

11. The system of claim 1 or any claim herein, wherein a user interaction with the at least one advertisement is stored in a database of a remote computer.

12. The system of claim 1 or any claim herein, wherein at least a portion of the at least one advertisement is received at the device from a remote computer via a network.

13. The system of claim 1 or any claim herein, wherein the at least one advertisement comprises one or more of the following functionalities:

functionality to visit a website;

functionality to download at least one new application;

functionality to operate the at least one application with one tap on a screen of the device;

functionality to download at least one new application with one tap on a screen of the device;

functionality to visit a website with one tap on a screen of the device;

reminder functionality; and

a graphical user interface.

14. The system of claim 1 or any claim herein, wherein the at least one advertisement comprises a text advertisement, a graphic advertisement, a video advertisement, an audio advertisement, or a combination thereof.

15. The system of claim 1 or any claim herein, wherein the at least one advertisement is contextually relevant to the at least one application.

16. The system of claim 15 or any claim herein, wherein the contextually relevant advertisement promotes a product that shares at least one characteristic with the at least one application.

17. The system of claim 16 or any claim herein, wherein the product is a second application that has a same type as the at least one application.

18. The system of claim 1 or any claim herein, wherein the mobile computing device is an Android device.

19. A method for displaying an advertisement on a mobile computing device comprising a processor and a memory, the method comprising:

Identifying, with an application module coupled to the processor, at least one application of the mobile computing device;

storing, with the application module, a database including at least one listing of the at least one application in the memory;

building, with a queue module coupled to the processor, a queue of the at least one application from the at least one listing in the database;

identifying, with a listener module coupled to the processor, at least one broadcast event generated by the processor and associated with the at least one installed application; and

displaying, with the listener module, at least one advertisement in response to the at least one broadcast event.

20. The method of claim 19 or any claim herein, further comprising managing, with a main module coupled to the processor, the application module, the queue module, the listener module, a rules module coupled to the processor, or any combination thereof.

21. The method of claim 19 or any claim herein, wherein building the queue comprises determining whether the at least one application should be displayed in an interface.

22. The method of claim 22 or any claim herein, wherein the determining is based on whether the at least one application has been run.

23. The method of claim 22 or any claim herein, wherein the determining is based on whether the at least one application has been installed.

24. The method of claim 19 or any claim herein, further comprising evaluating, with a rules module coupled to the processor, the at least one broadcast event against

at least one rule to determine whether the at least one advertisement can be displayed, to determine a characteristic of the at least one advertisement, or a combination thereof.

25. The method of claim 24 or any claim herein, wherein the rule comprises whether it is permissible to display information full screen on the device, whether the advertisement may be displayed by interrupting control of the device, whether the advertisement may be displayed while the device is being used as a phone, whether the advertisement may be displayed while the device is in a screen lock mode, or any combination thereof.

26. The method of claim 19 or any claim herein, wherein the broadcast event is a PACKAGE_INSTALL.

27. The method of claim 19 or any claim herein, wherein the broadcast event is an operating system broadcast intent

28. The method of claim 19 or any claim herein, further comprising storing a user interaction with the at least one advertisement in the memory.

29. The method of claim 19 or any claim herein, further comprising storing a user interaction with the at least one advertisement in a database of a remote computer.

30. The method of claim 19 or any claim herein, further comprising receiving at least a portion of the at least one advertisement at the device from a remote computer via a network.

31. The method of claim 19 or any claim herein, wherein the at least one advertisement comprises one or more of the following functionalities:

functionality to visit a website;

functionality to download at least one new application;

functionality to operate the at least one application with one tap on a screen of the device;

functionality to download at least one new application with one tap on a screen of the device;

functionality to visit a website with one tap on a screen of the device;

reminder functionality; and

a graphical user interface.

32. The method of claim 19 or any claim herein, wherein the at least one advertisement comprises a text advertisement, a graphic advertisement, a video advertisement, an audio advertisement, or a combination thereof.

33. The method of claim 19 or any claim herein, wherein the at least one advertisement is contextually relevant to the at least one application.

34. The method of claim 33 or any claim herein, wherein the contextually relevant advertisement promotes a product that shares at least one characteristic with the at least one application.

35. The method of claim 34 or any claim herein, wherein the product is a second application that has a same type as the at least one application.

36. The method of claim 19 or any claim herein, wherein the mobile computing device is an Android device.

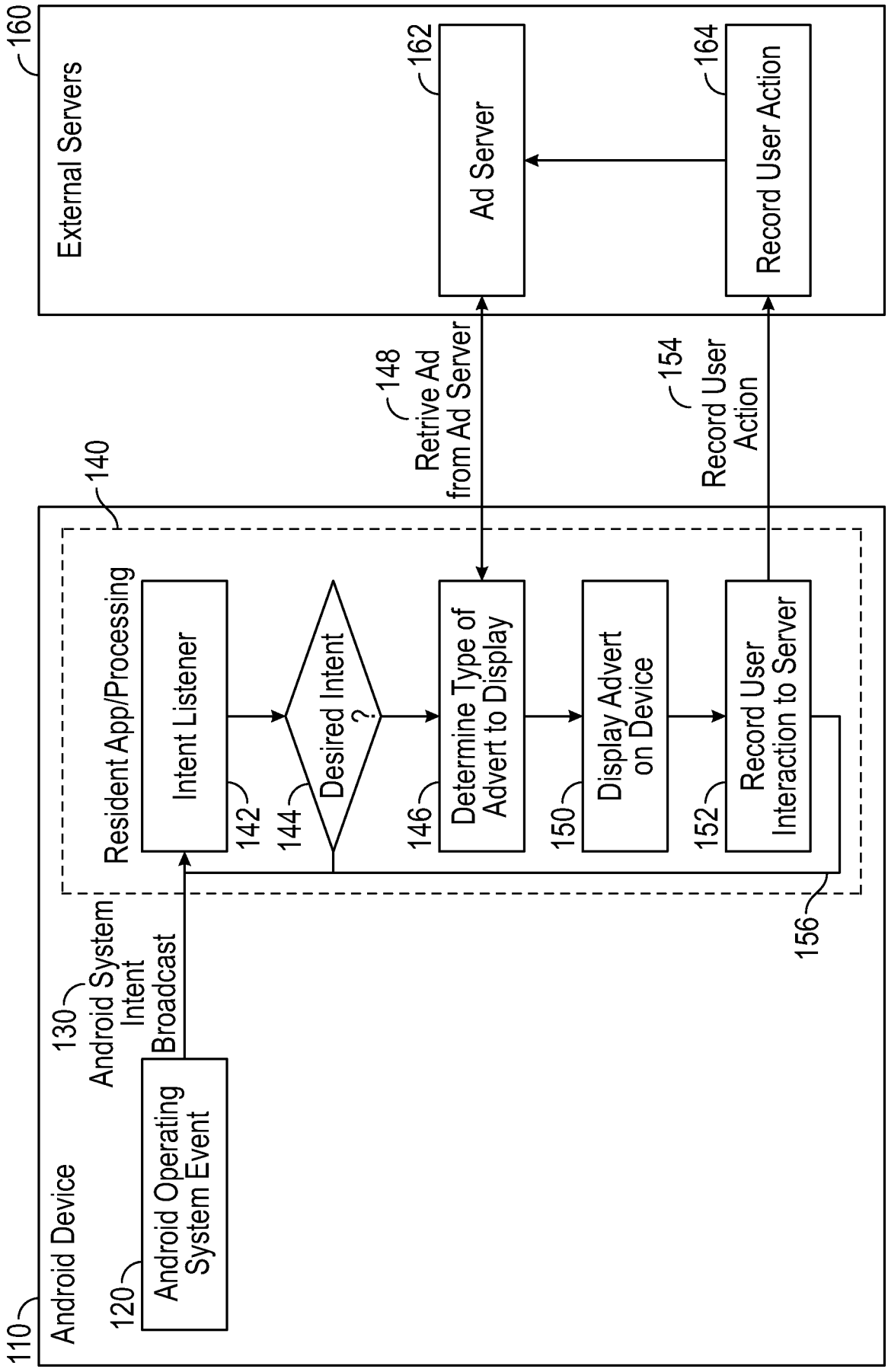

160
External Servers
162
Ad Server
164
Record User Action
148
Retrive Ad from Ad Server
154
Record User Action
140
Resident App/Processing
Intent Listener
142
Desired Intent ?
144
Determine Type of Advert to Display
146
Display Advert on Device
150
Record User Interaction to Server
152
156
130
Android System Intent Broadcast
Android Operating System Event
120
Android Device
110

FIG. 1

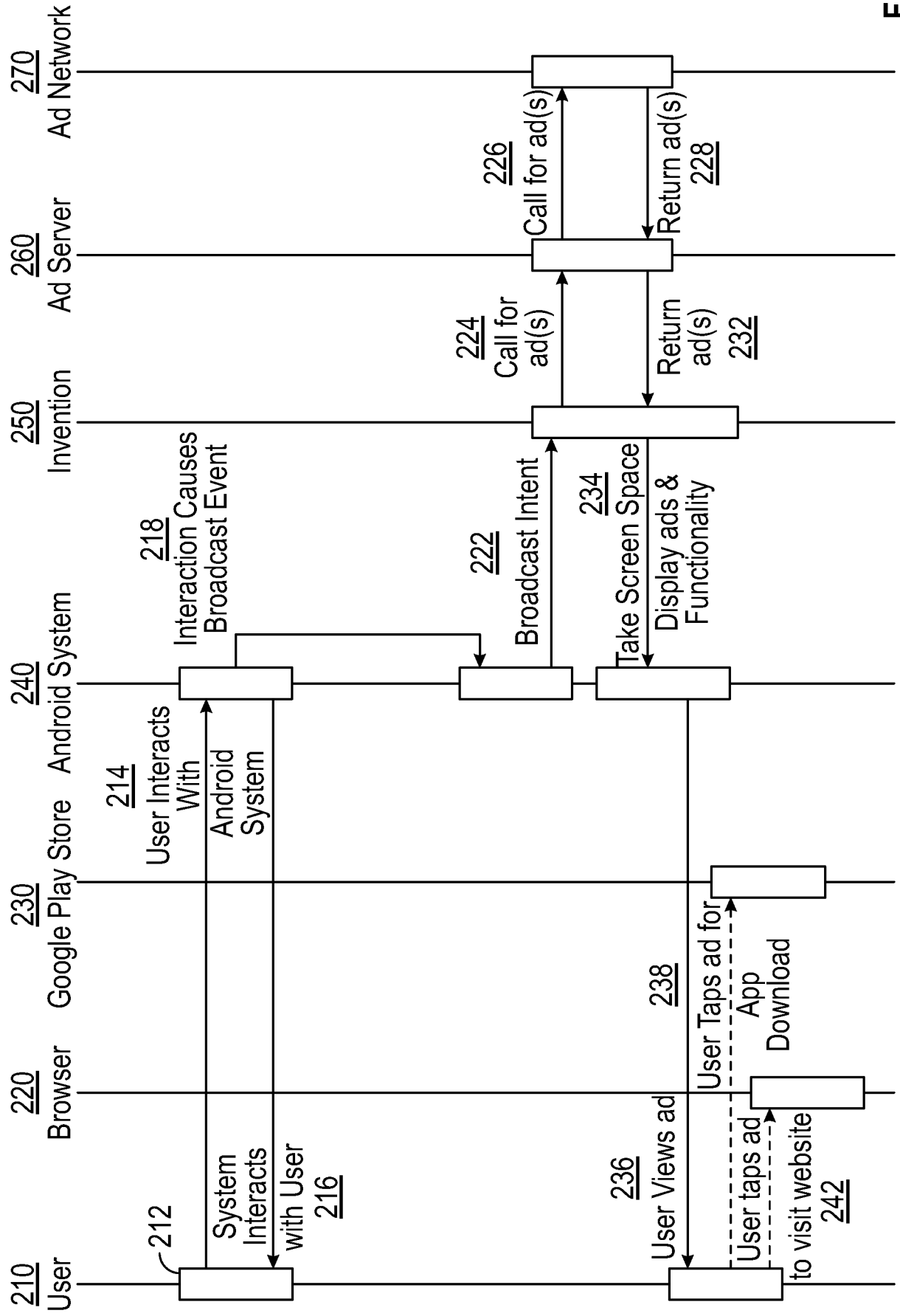

210 User
220 Browser
230 Google Play Store
240 Android System
250 Invention
260 Ad Server
270 Ad Network
212
214 User Interacts With Android System
216 System Interacts with User
218 Interaction Causes Broadcast Event
222 Broadcast Intent
224 Call for ad(s)
226 Call for ad(s)
228 Return ad(s)
232 Return ad(s)
234 Take Screen Space
Display ads & Functionality
236 User Views ad
238 User Taps ad for App Download
242 User taps ad to visit website

FIG. 2

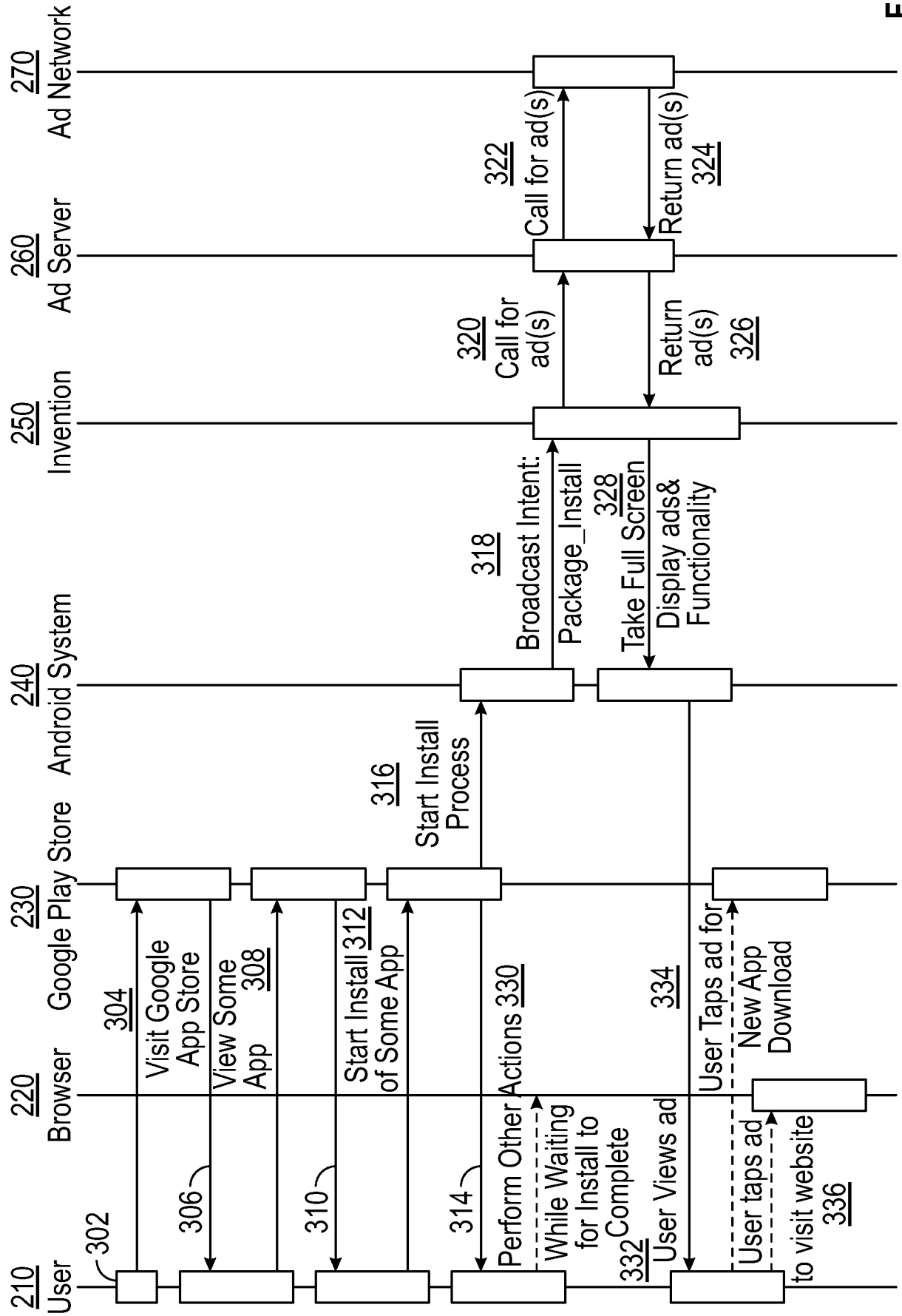

210 User
220 Browser
230 Google Play Store
240 Android System
250 Invention
260 Ad Server
270 Ad Network
302
304 Visit Google App Store
306
308 View Some App
310
312 Start Install of Some App
314
316 Start Install Process
318 Broadcast Intent: Package_Install
320 Call for ad(s)
322 Call for ad(s)
324 Return ad(s)
326 Return ad(s)
328 Take Full Screen Display ads& Functionality
330 Perform Other Actions While Waiting for Install to Complete
332
334 User Views ad
User Taps ad for New App Download
User taps ad to visit website
336

FIG. 3

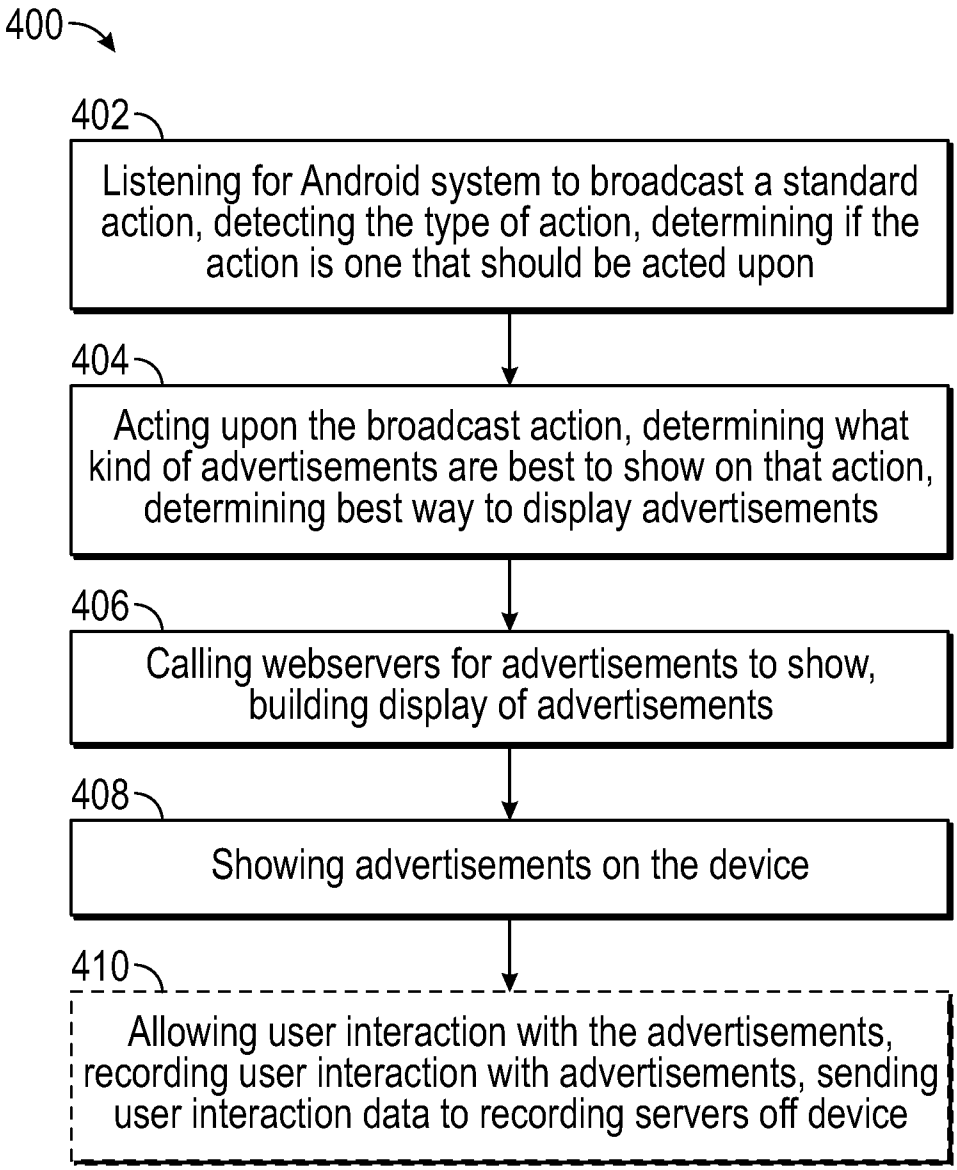

400
402
Listening for Android system to broadcast a standard action, detecting the type of action, determining if the action is one that should be acted upon
404
Acting upon the broadcast action, determining what kind of advertisements are best to show on that action, determining best way to display advertisements
406
Calling webservers for advertisements to show, building display of advertisements
408
Showing advertisements on the device
410
Allowing user interaction with the advertisements, recording user interaction with advertisements, sending user interaction data to recording servers off device

FIG. 4

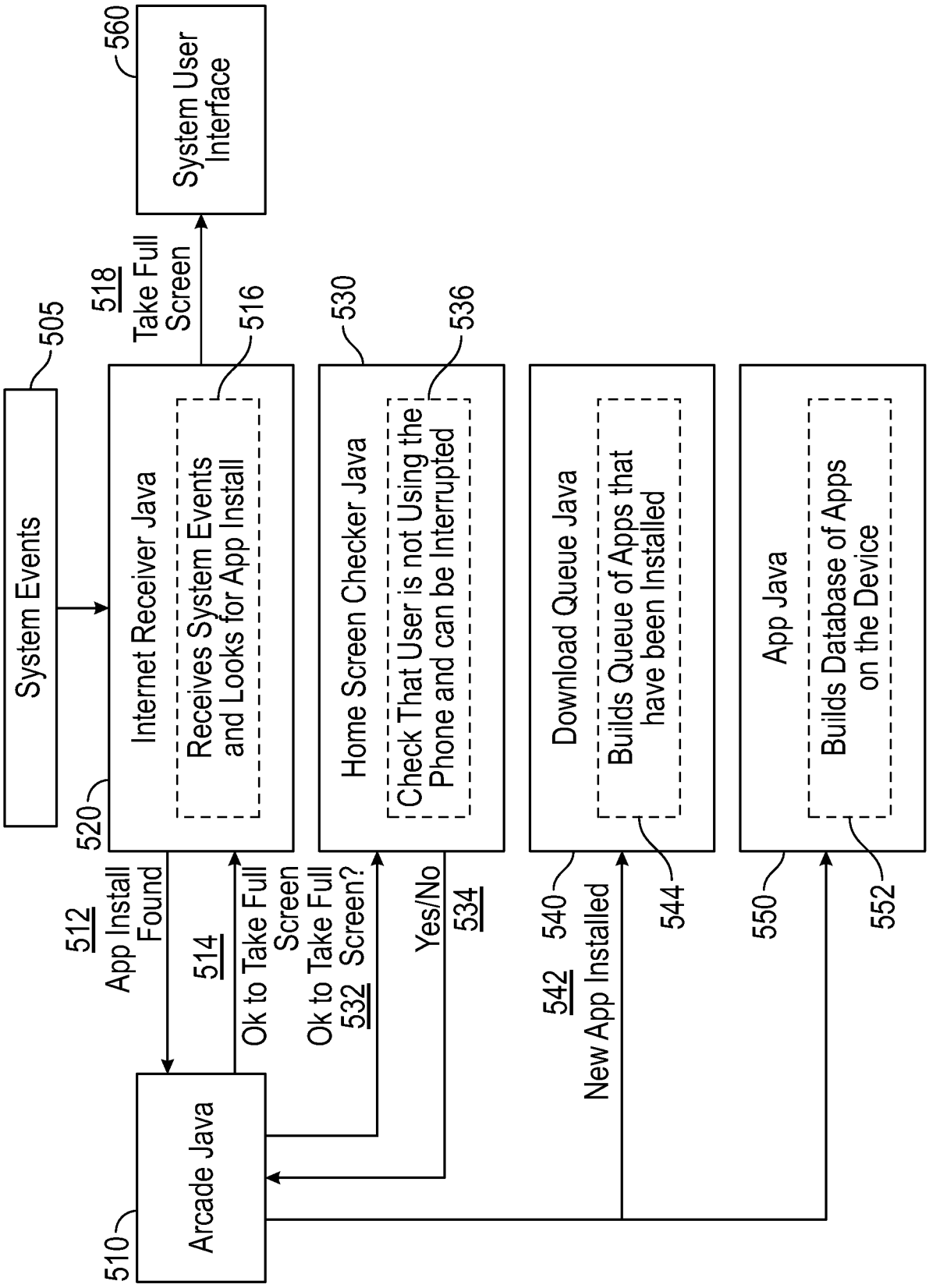

System Events
505
Internet Receiver Java
Receives System Events and Looks for App Install
520
516
518
Take Full Screen
System User Interface
560
Home Screen Checker Java
Check That User is not Using the Phone and can be Interrupted
530
536
Download Queue Java
Builds Queue of Apps that have been Installed
App Java
Builds Database of Apps on the Device
Arcade Java
510
512
App Install Found
514
Ok to Take Full Screen
532
Ok to Take Full Screen?
Yes/No
534
540
542
New App Installed
544
550
552

FIG. 5