MobiQ Application Map:

Breakdown of components and their functional description

Components (Java classes)	Type	Functional description
LocationHttpService MondayQHttpService ThursdayQHttpService CallLogHttpService SmsLogHttpService DeviceInfoHttpService	Service components that extend IntentService	• Run in the background when called by Intent. • Acquires wake lock using PowerManager API • Checks for network availability using **ConnectionManager**. • If network is available read file holding app data. • If file contains data creates a **FileUploader** instance. • Use **FileUploader** to upload data in file to database URL. • Post current data to database URL. • If network is unavailable save current data to file.
ConnectionManager	Java Class	• Uses ConnectivityManager API to get network information and connectivity status. • Implements method to display network state. • Utilized by all the above Http service components.
FileUploader	Java Class	• Uses a FileInputStream to read file contents into an array of bytes. • Writes the contents of the array into a DataOutputStream object. • Creates a URL object and opens Http connection. • Uses the DataOutputStream object to send file contents to a remote URL via the Http connection.
MobiQBroadcastReciever	Broadcast Receiver	• Listens for BOOT_COMPLETED intent and periodic intents from AlarmManager. • If device is being reboot (BOOT_COMPLETED intent received) check if app was in an active state before shut down. • If app was in active state restart AlarmManager object. Otherwise, maintain alarm sleep state. • Else If intent is received from AlarmManager object, do the following: ○ Use **FilesChecker** object to check for app data, upload data to database URL if there is a connection. ○ Check the difference between the current time and the last event(s) reference time(s). ○ If 20 minutes elapsed, invoke **LocationHttpService** to acquire new GPS information and send/save to file. ○ If 60 minutes elapsed, invoke **CallLogHttpService** to get new calls

		information for sending. o If 120 minutes elapsed, invoke **SmsLogHttpService** to get new calls information for sending. o If the day is Monday and time is 5pm, send intent to **MondayDialog** activity to alert user for weekly questionnaire. o If the day is Thursday and time is 5pm, send intent to **ThursdayDialog** activity to alert user for weekly questionnaire. • Implements methods to cancel and reset the AlarmManager object. • If the AlarmManager object is set/reset, it sends intents to the Broadcast receiver periodically (every 10 minutes). • When the AlarmManager object is cancelled, no intents will be received except the BOOT_COMPLETED intent. When the AlarmManager object is reset, it continues to receive the periodic intents and operate as described above.
FilesChecker	Java Class	• Implements methods to check connections and data file contents. • If there is any valid data in the file, and a connection is detected it uses a **FileUploader** object to upload the given file to a specified remote URL.
MondayDialog	Activity	• Uses an AlertDialog to alert user for the Monday weekly survey. • Calls a startSurvey method if user presses OK button on AlertDialog. • Within the startSurvey method, a **ResetResults** object is used to clear the weekly survey results cache. • Checks the state of the previously answered questions, if all ‘Ever used’ questions were answered yes, invoke the **WeeklyActivity** screen via an intent. • If not all ‘Ever used’ questions were answered yes invoke either **EverActivity** or **EverOthersActivity** screen via an intent depending on the answers supplied to the ‘Ever used’ questions in the previous week.
EverActivity	Activity	• A screen that pops up with the ‘Ever used’ questions when invoked via an intent. • Only those questions *never asked before* will be displayed on the screen. • Questions are answered by selecting from grouped radio buttons.

		• When the user hits the 'submit button', all the necessary consistency checks are made and the results are cached. • If (from checking the cache) all given answers are 'no' the **CameraActivity** is called to take a picture (OPTIONAL). The screen is then closed. • If some answers were 'yes' and/or the user currently answered yes to 'any other drugs' question, the **EverOthersActivity** screen is called by sending an intent. The screen is then closed. • If (from checking the cache) the user has previously answered all yes to 'having ever used other drugs' the **WeeklyActivity** Screen is called instead and the current screen is closed.
EverOthersActivity	Activity	• A screen that pops up with the 'Ever used other' questions when invoked via an intent. • Only those questions on 'any other drugs' *never asked before* will be displayed on the screen. • Questions are answered by selecting from grouped radio buttons. • When the user hits the 'submit button', all the necessary consistency checks are made and the results are cached. • Sends an intent to invoke the **WeeklyActivity** screen.
WeeklyActivity	Activity	• A screen that pops up with questions about the substances use 'in the last week' when invoked via an intent. • It checks the cache to determine which questions to ask, corresponding to the users previous answers on substances ever used. • When the user hits the 'submit button', all the necessary consistency checks are made and the results are cached. • Calls **MondayQHttpService** to store/send the final questionnaire answers. • The **CameraActivity** is called to take a picture (OPTIONAL). The screen is then closed.
ResetResults	Java Class	• Contains a method that resets all the app cache contents to initial state. • Contains a method that clears the weekly survey cache contents. • Implements a method to cancel the AlarmManager object periodic operation.
ThursdayDialog	Activity	• Uses an AlertDialog to alert user for the Monday weekly survey. • Calls a startSurvey method if user presses OK button on AlertDialog. • Within the startSurvey method, a **ResetResults** object is used to clear the weekly survey results

		cache. • Calls the **WeeklySportsClubActivity** screen via an intent.
WeeklySportsClubActivity	Activity	• A screen that pops up when invoked via an intent with 3 questions asking whether sports, clubs and non-club activities were done in the last week. • Answers are given via grouped radio buttons. • When the user hits the 'submit button', all the necessary consistency checks are made and the results are cached. • If all 3 answers are 'no' then an intent is sent to invoke the **CameraActivity** (OPTIONAL). **ThursdayQHttpService** is called to upload/store the answers. The activity is then closed. • If the user answered yes to sport then **SportsListActivity** screen is called via intent. • If the user answered no to sport but yes to club activities then **ClubsListActivity** screen is called via intent. • If the user answered no to sport and club activity but yes to non-club activities then **NonClubsListActivity** screen is called via intent.
SportsListActivity	Activity	• Creates a checkboxed list of sport activities pulled from a String array stored in the string.xml resource file. • When the user hits the 'submit button', all the necessary consistency checks are made and the results are cached. • Notes whether 'other' sport is selected and bundles this in intent which is sent to invoke the **SportsRegularActivity** screen.
SportsRegularActivity	Activity	• A pop up screen that contains questions on how many times the user did the selected sports activity. Grouped radio buttons are used to receive user response. • When the user hits the 'submit button', all the necessary consistency checks are made and the results are cached. • If 'other' sport was selected in the previous **SportsListActivity** screen an intent is sent to call the **OtherSportActivity** screen. • Else if 'other' sport was not selected in the previous **SportsListActivity** screen an intent is sent to call the **ClubsListActivity** screen. • Else if the user previously answered no to club activities but yes to non-club activities an intent is sent to call the **NonClubsListActivity** screen. • If none of the above is true, then the **CameraActivity** is called to take a picture

		(OPTIONAL). **ThursdayQHttpService** is called to upload/store the answers. The activity is then closed.
OtherSportActivity	Activity	• Contains an edit text box for the user to enter the name of one other sport activity not listed in the **SportsListActivity** screen. • When the user hits the 'submit button', all the necessary consistency checks are made and the results are cached. • If the user previously answered yes to club activities an intent is sent to call the **ClubsListActivity** screen • Else if the user previously answered no to club activities but yes to non-club activities an intent is sent to call the **NonClubsListActivity** screen. • If none of the above is true, then the **CameraActivity** is called to take a picture (OPTIONAL). **ThursdayQHttpService** is called to upload/store the answers. The activity is then closed.
ClubsListActivity	Activity	• Creates a checkboxed list of club activities pulled from a String array stored in the string.xml resource file. • When the user hits the 'submit button', all the necessary consistency checks are made and the results are cached. • Notes whether 'other' club activity is selected and bundles this in intent which is sent to invoke the **ClubsRegularActivity** screen.
ClubsRegularActivity	Activity	• A pop up screen that contains questions on how many times the user did the selected club activity. Grouped radio buttons are used to receive user response. • When the user hits the 'submit button', all the necessary consistency checks are made and the results are cached. • If 'other' club activity was selected in the previous **ClubsListActivity** screen an intent is sent to call the **OtherClubActivity** screen. • Else if 'other' club activity was not selected in the previous **ClubsListActivity** screen and the user previously answered no to non-club activities, **ThursdayQHttpService** is called to upload/store the answers. **CameraActivity** is called to take a picture (OPTIONAL).The activity is then closed. • Else if the user previously answered yes to non-club activities an intent is sent to call the **NonClubsListActivity** screen.
OtherClubActivity	Activity	• Contains an edit text box for the user to enter the name of one other club activity not listed in the **ClubsListActivity** screen.

		• When the user hits the ‘submit button’, all the necessary consistency checks are made and the results are cached. • If the user previously answered yes to non-club activities an intent is sent to call the **NonClubsListActivity** screen • Else If the user previously answered no to non-club activities, the **CameraActivity** is called to take a picture (OPTIONAL). **ThursdayQHttpService** is called to upload/store the answers. The activity is then closed.
NonClubsListActivity	Activity	• Creates a checkboxed list of non-club activities pulled from a String array stored in the string.xml resource file. • When the user hits the ‘submit button’, all the necessary consistency checks are made and the results are cached. • Notes whether ‘other’ non-club activity is selected and bundles this in intent which is sent to invoke the **NonClubsRegularActivity** screen.
NonClubsRegularActivity	Activity	• A pop up screen that contains questions on how many times the user did the selected non club activity. Grouped radio buttons are used to receive user response. • When the user hits the ‘submit button’, all the necessary consistency checks are made and the results are cached. • If ‘other’ non club activity was selected in the previous **NonClubsListActivity** screen an intent is sent to call the **OtherNonClubActivity** screen. • Else **ThursdayQHttpService** is called to upload/store the answers. **CameraActivity** is called to take a picture (OPTIONAL).The activity is then closed.
OtherNonClubActivity	Activity	• Contains an edit text box for the user to enter the name of one other club activity not listed in the **NonClubsListActivity** screen. • When the user hits the ‘submit button’, all the necessary consistency checks are made and the results are cached. • The **CameraActivity** is called to take a picture (OPTIONAL). **ThursdayQHttpService** is called to upload/store the answers. The activity is then closed.
UserInterfaceActivity	Activity	• This is the only **Launcher** activity within the app and through which the user can interact with the app to select operation mode, take pictures, and enter comments/problems etc. • If the app is reset, a **WelcomeActivity** screen is displayed on tapping the launch icon and the **UserInterfaceActivity** is immediately exited.

WelcomeActivity	Activity	• The first welcome screen the user sees on installation. • It flashes a welcome for a few seconds. • It uses an **InfoGetter** object to collect device information such as imei, OS name, OS version, OS release, model etc. and stores this in a cache. • It initializes the AlarmManager object (i.e. activates the app operation), initializes most of the apps global variables stored in the cache. • A GPS LocationListener object is activated to request location updates every 15 minutes. • The DeviceInfoHttpService is called via intent to store/upload the collected device data to the remote database.
ReportProblemActivity	Activity	• A screen that presents an input text box to the user to report any problems they may be encountering with the app's operation. • It is only accessible to the user via the **UserInterfaceActivity**. • When the submit button is hit, the comments along with device imei, and current time are uploaded to the remote database, or stored on file if there is no connection.
CameraActivity	Activity	• Requests services of the camera hardware to take a picture. • Enables storage on external SD card directory. • Calls **FileUploader** to upload the captured image file and an optional file containing caption/comments in a separate non UI thread.
ModeSelectActivity	Activity	• A screen that is accessible through the presents the **UserInterfaceActivity** with options to select Admin mode of operation or user mode. • Selecting the user mode returns the app back to the **UserInterfaceActivity.** • Selecting the Admin mode button brings up a login screen i.e. **AdminLoginActivity** where an admin passcode can be entered to login as a privileged user.
AdminLoginActivity	Activity	• Login screen for enabling privileged mode of operation. • If login is successful, the **MainActivity** screen will be brought up. • The screen is exited via an exit button and this also completely exits from the app.
MainActivity	Activity	• This screen mainly contains buttons for privileged mode demonstration of the app's capabilities. • It contains, Start Survey, Reset Mobi-Q, Suspend, Resume, Survey Results, Exit Admin Mode, and View Database buttons.

		• App can be reset, suspended, or resumed from this screen.
ShowResults	Activity	• Pulls current survey answers from the cache for display when the Survey Results button on the **MainActivity** is hit.
CallMonitor	Broadcast Receiver	• Class for monitoring outgoing calls. • Listens for outgoing calls and determines destination number, call time, and calculates call duration. • Stores call data to file along with device imei.
SmsRecoder	Java Class	• Uses ContentResolver queries to get SMS log information. • Determines change in number of outgoing SMS messages from cached records. • If there are new messages since the last check, collect the SMS data (destination number, time) and append imei. Store these on the SMS data file for uploading to remote database.
InfoGetter	Java Class	• Returns device id, OS name, version, api, model, cpu, model, release etc. of the Android device.