Android App Design

As any other android applications, the application consists of activities which are standalone and complete independent from one another. (To clarify terminology, an activity is a screen that displays on the android device in use.) The application also consists of several auxiliary classes which assist in the functionalities of the main activities (i.e. the activities).

The WitsCabs android application has five activities in total, which three out of the five activities are the core activities of the application. Each of the five activities are standalone as stated previously, and thus each of the five activities constitute to different and specific functionalities of the application. Each of the activities are linked to a single .xml file which helps to display the visual aspect of the activity (the GUI) on the device.

The different (main) activities along with their functionalities are described below:

* StartUpActivity: this activity is linked to the startupactivity.xml. This specific activity is launched when the application is launched. The StartUpActivity displays an image along with text that reads “Welcome to WitsCabs!”, and automatically times out after 7 seconds to start up the next activity which is the login page for the application.
* loginPage: this activity is linked to the loginlayout.xml and is launched automatically after the StartUpActivity. This activity allows the user (i.e. the driver) to login to the application by inserting his/her username and password in the provided text boxes followed by clicking the login button, and also provides a button which the user can click to register as a driver at WitsCabs. Depending on which button is clicked the application will lead to a different activity. If the login button is clicked, the application will then lead to the dashboard of the application and if the register button is clicked, the application will lead to the register activity.
* signUpPage: this activity is linked to the signuplayout.xml and is launched when the “Register” button is clicked in the loginPage activity. This activity consists of several text boxes which allows the user to insert his/her information into. Note that these text boxes have ”hints” which indicates to the user what information is required to be entered by the user at each text box. At the of the application there is a button named “Register” to officially register the user, and once clicked and if successful the application will lead back to the loginPage activity and if unsuccessful the application will stay on the same activity and display a not registered message via a toast.
* DashBoard: this activity is linked to the dashboardlayout.xml and is launched when the user is successfully logged in, as well as this activity is the “main activity” after logging in. In this activity is where the user can change his availability status, there is an analogue clock that displays the time, a text view section which displays the details of the passenger that has been allocated to the specific driver, and there is a “Go To Maps” button which once clicked the application leads to the MapsActivity.

Note: all of the above activities extends from the Activity superclass.

* MapsActivity: this activity is linked to the activity\_maps.xml and is launched when the “Go To Maps” button is clicked in the dashboard. The main purpose of the activity is to give directions to the driver to get to the passenger, and to get the passenger to his/her destination. This activity makes use of the google maps API and google directions API, along with the auxiliary predefined modules by google. These modules include DirectionFinder determines the route of the trip required, DirectionFinderListener which listens when directions are requested, Duration calculates and stores the duration of the current trip, Route stores the route.

Auxiliary classes

* AsyncClassSignUp and AsyncClassLogin: initiates the communication with the backend server in order to send the required registration and login information respectively to the server, and receives a response from the server. Based on the response, certain decisions are made in the signUpPage and loginPage respectively. Please note that the IP address when declaring the Socket has to be altered based on the machine that the backend server is being run on.
* JSON\_Handler: converts an array of strings to a JSON string in order to send data to the server using a JSON string format.

Important notes:

The actual application is not fully as required due time constraints and issues with the google API, however basic functionality of the app is operating.