CS3301 Practical

Student ID:160016114

Date: 26th February 2019

Overview:

The aim of this practical was to develop a component system, by developing a small-scale component-based application. The practical involved designing and implementing an Android Pseudo-Uber Client-side application, where custom components had to be designed and public components had to be used to make a functioning application. The aim of this application is to make use of different components to complete basic specifications, such as making user accounts on the application, displaying a map, where users can search for a location, make requests of trips, from one place to another, simulating drivers on the map as a part of requesting trips, and displaying a summary page displaying elements such as rating the journey, and displaying the fare. The framework used to build this Client-side application is Android Studio.

Design:

The application is divided into multiple sections, one being the startup section, then a login and registration section, the main maps page, and finally a concluding section. The component types used in this program are Activities and Content providers. Content providers grant permissions in this application for instance, to get the users' current location.

In the "MainActivity", there exists a display of the introductory page, which states the title of the app and a button titled "Continue", which directs the user to the login and registration page.

The next part of the application as mentioned, is the login and registration page, which is described inside the "DriverLoginActivity" Activity, where the user of the application could do either by just entering an email address and a password. The application will throw an error should there be an already existing user trying to register with the same credentials or a new user trying to login, without registering first. The login credentials of every user, that makes an account on the application are stored on the database Firebase, which makes sure, that once the user registers, they can login the next time they want to use the application. After the user login or registration is successful, the user is directed to the main page of the application, where the "MapsActivity" comes into play.

Inside the "MapsActivity", in the "onCreate", there are multiple variables made to define the "textviews", the buttons present in the activity and the relative layouts in the "activity_maps.xml" file. An "onClickListener" exists for every button that is present in "OnCreate" which defines the tasks that occur as soon as the button is tapped on. A map is displayed along with a search bar and a logout button, where the "onMapReady" override method is called inside the "MapsActivity". As soon as the user searches for a valid location, the "initialise" method which is present inside the "onMapReady" method is called, which further calls the geolocate method as soon as the user presses enter on the keyboard. The operation variable is set to 0, so that there is no movement of the car marker, and the "getRouteToMLocation" method is called, to draw the polylines, from the user's location to the location that has been searched for by the user. The camera moves to the location searched for so the user can press the icon at the top right to redirect themselves back to their own location, to select a driver. Finally, a layout appears on the maps page itself, which states that the customer is now able to pick a driver for the journey, by tapping on each available driver, and selecting whichever driver they prefer for their journey. All the drivers are given a description, which includes the name of the driver, the name of the car they drive and their rating. A separate class is made that stores a constructor so as to define the drivers' credentials. Whenever there is a need for a new driver, this class is accessed to be make a new driver giving that driver all the required credentials. Apart from the textual description, which appears on the layout, where one can confirm driver, every driver is also assigned an image. As the cars displayed are fundamentally markers on the map. A "setOnMarkerClickListener" is made, which displays the selected driver marker's description along with the image of the driver. At the beginning of this marker listener, a boolean condition is set which makes it mandatory for the user to search for a location first, only then is it possible for them to select a driver for the journey. This boolean condition is set to true at the end of the function "geLocation", which is used to search for a location on the map.

The onMapReady also where the description of all five drivers is added to an array list 'credentials', so that the right driver's description is displayed from this arraylist in the marker listener. The cars icon might be placed in such a way that they overlap each other, however, so it might seem that there are less than 5 cars but that is not the case. If the user taps on the cars, then they could read the description for each of their drivers. Once the user selects a driver, there is a button at the bottom right of this layout, titled 'Confirm Driver' which when tapped on, sets the operation to 1, which means that this time the selected car marker is going to move from its original position, to the user's location once the "getRouteToMLocation" method is called. The distance from the customer's location to the destination is stated along with the duration of the journey. The polylines drawn from the user's location to the final destination are first erased and a route from the driver's location to the user's location is drawn as the override method "onRoutingSuccess" is executed when "getRouteToMLocation" method is called. Once the driver arrives the user's location, the text on the layout changes, indicating to the user that the driver has arrived. A button titled "I'm in the car!" appears on the bottom left of the layout, which the user can tap on to begin their journey. Once this button is tapped, the car marker starts to move along the polylines that have now been redrawn from the user's location to the final destination, that the user searched for, and confirmed the driver for. The layout, journey description layout also disappears as soon as the said button is tapped.

As soon as the car marker reaches the destination, a final relative layout called Arrival is displayed which just states that the user has arrived at the destination and gives the user an opportunity to rate their journey and review the fare, and finally exit the page by clicking on the done button. The fare is a simple calculation performed in the "getFare" method which takes the distance and the duration as parameters. There are local variables which store the values for base fare, cost per kilometer and cost per minute. The total fare is equal to the base fare added to the product of the distance and the cost per kilometer, which is further added into the product of cost per minute and the total duration of the journey. Since the values that were being generated using this algorithm were extremely huge, the base fare, the cost per kilometer and the cost per minute are extremely small.

Evaluation:

This implementation meets all the requirements of the basic specification. Users can log in or register in the app, request trip from their position to any legitimate location. The users have options to select from a list of five drivers. The drivers are shown to drive from their location to the pick-up location of the user and then shown to drive from the pick-up location to the destination. After journey completion, the user has an option to rate the journey and also view their fare. This application also uses the various attributes present in the xml file to make the application features more aesthetic and attractive to the user's eye. The extensions added in this application are tracking drivers' real time location.

Conclusion:

This application does not work correctly in some instances, for example, the driver markers don't appear when once the user logs out of the application and logs back in. The app needs to be closed and then launched again for it to work. Apart from this, the application serves a suitable example of a Pseudo Uber client side application. After working on this practical, I have learnt a lot of important aspects about android application development, in android studio. Given more time, I would have implemented more extensions and made my application look more aesthetic, and easy to use.