**Activity: Main or List of Rides**

This is the first activity that appears when the user logs in. The Main Activity, or Ride List, provides navigation to the rest of the app, and displays the user’s ride history. The Activity was built using Android’s Navigation Drawer layout, which uses multiple layout files, and an .xml file for the navigation drawer’s content.

The primary feature of the Activity itself is to display the current user’s history of the rides they’ve participated in. The rides are shown in a custom ListView that displays a series of layouts showing the trips destination, the driver’s profile picture, the departure time, available number of seats, and the number of chat messages sent. Under the ListView are two buttons that allow the user to launch Google Maps to find a nearby ride, or to create a new ride respectively.

It also creates a navigation drawer which provides links to other Activities within the app.

**Manifest Details**

This activity is implemented with MainActivity.java. Like most of the Activities, its screenOrientation is restricted to “portrait.”

The Activity uses the theme”@style/AppTheme.NoActionBar”, which can be found in the res/values/styles.xml.

The Intent filter provides standard access when accessed by name.

**Layout Files**

Most of Android Studio’s Activity Templates are put together with multiple layout files, that reference *other* layout files, and it can be confusing to work with. If you’re new to developing Android apps, it is recommended to research the desired template before using it. But in most cases, sometimes using an Empty or Basic Activity is enough.

activity\_main.xml

* Creates the actual navigation drawer, by retrieving the drawer’s entries from activity\_main\_drawer, and retrieving the drawer’s header.
* nav\_view (NavigationView) – the navigation drawer that pulls data from another layout within the project files
  + menu: @menu/activity\_main\_drawer – displays menu items
* <include> -- Includes are used to display layouts in different files, referenced in the layout property.
  + Layout: @layout/app\_bar\_main

nav\_header.main.xml

* The header of the navigation drawer.
* This holds the user’s profile picture, name, and email address, and all three values are stored in a Linear Layout, and are accessed in the Activity’s code.

app\_bar\_main.xml

* Holds a toolbar that shows the application’s title, and provides a link to the apps conent.
* Toolbar – the Activity’s toolbar. It’s necessity is…questionable. In every class I have a block of code that creates a toobar and adds a back button, but the app crashes on this Activity because a toolbar already exists. It’s probably safe to delete this…
* <include> -- an include with a link to the Activity’s actual content.
  + Layout: @layout/content\_main

content\_main.xml

* This is the section that contains all the views that actually need to be edited within the layout.
* noRidesHeader/Footer (TextView) Contains a message that the user is not a part of any rides.
  + Visibility: INVISIBLE – Both views are hidden by default. If the user’s list of rides are empty, MainActivity.java will set visibility to VISIBLE.
* findRideButton (Button)
  + onClick: startFindRide – Start the Find Ride Activity
* createRideButton (Button)
  + onClick: startCreateRide – start the Create Ride Activity
* rideListView (ListView) – displays multiple item\_ridelist layouts in a list. This layout is set from MainActivity.java
  + divider: @android:color/transparent – ListViews show dividers between each element by default. This makes the dividers invisible.

item\_ridelist.xml

* The Ridelist Item is a layout that gives a trip’s summarized details. It displays the origin and destination, the driver’s profile image, the address of both locations, the number of seats in the driver’s vehicles, and the number of messages that were passed along. To display all of the user’s rides in content\_main’s ListView, MainActivity.java will actually use a CustomListAdapter to achieve this. None of the item’s functionality is provided within the layout. Click events are handled in the code behind.

**Resource Files**

activity\_main\_drawer.xml

* The navigation drawer’s actual menu. Each item is added in the form of a Menu Item.
* Menu Items are named through their “title” property, and they are referenced in the code behind through their “id” property.

|  |  |
| --- | --- |
|  |  |

**Class File**

MainActivity.java implements the Main Activity, or Ride List. It includes an internal class for setting up RideList Items.

Variables

* -prefs: SharedPreferences
  + Holds a reference to the devices preferences, which stores the user’s Id and access token
* -userId: String
  + Holds the user’s id number has a string to append to a request’s url.

Methods and Internal Classes

* \*onCreate(savedInstanceState): void
  + Creates the Activity’s layout, initializes the ‘prefs’ and ‘userId’, initializes the Activity’s toolbar, and creates a toggle button to open and close the Navigation Drawer. Lastly, it calls the method to assign user information onto the navigation drawer’s header, and to retrieve all trips the user has taken part in.
* -setupNavigation(): void
  + Adds the navigation drawer’s header, then calls the method to retrieve the user’s information from the server, passing the header as a parameter.
* -getUserInformation(v): void
  + Initializes the header’s profileImage, user\_nameTextView, and emailTextView, and sends a JSON Object GET request to display the user’s profile picture, first and last names, and email address.
  + **Note:** When referring to the user’s login, variables and Views would use “username.” In this case, it is called “user**\_**name” in references to the user’s *actual full name*. This is the only instance of the app where this happens.
* -getAllRides(): void
  + Displays all of the user’s trips in the Activity’s ListView.
  + A request is sent using a JSON Array GET request.
  + First an ArrayList that accepts HashMaps is created. Next, the request loops through all of the JSON Objects returned from the server. Each iteration creates a new HashMap, where both the key and value pair are String objects, and assign the trip’s values. At the end of the iteration, the HashMap is added to the ArrayList.
  + However, if no trips were returned from the server, the ListView’s visibility is set to INVISIBILE, while the “No Rides” message is set to VISIBLE. Otherwise, the request creates a new instance of the ListAdapter class to populate the ListView, passing the ArrayList into the constructor.
  + If there is an error with the request, the application is closed, and a Toast message appears.
* +startFindRide(view): void
  + Opens the Find Ride Activity
* +startCreateRide(view): void
  + Starts the Create Ride Activity
* +onBackPressed(): void
  + Gives two different functions to the Android’s back button
  + The first: If the navigation drawer is open, simply close the navigation drawer
  + The second: but if the navigation drawer is closed, the app will create an Intent to restart the Main Activity, but it will also send a Boolean Extra named “Exit”. When the Main Activity’s onCreate() function is called again, it will check if the “Exit” Extra exists, and it will close the application without going back to the Login Activity.
* +onNavigationItemSelected(item): Boolean
  + Checks the value of the Menu Item tapped in the navigation drawer, and opens the corresponding activity. This contains link to the Profile, Messages, and Terms and Conditions. It can also log out the user by clearing the data saved on the phone, and reopening the Login Activity
* +clearUserData(): void
  + Clears the user’s access token and userId stored on the phone, and stops the TripLocationService if it was active.
  + The commented code also clears the username, password, and token expiration date, but these are currently unused since the app no longer stores user information for automatic logins.
* -RideAdapter
  + This internal class creates Ridelist Items to display in the Main Activity’s ListView.
  + –rides: ArrayList<HashMap>
    - Holds a copy of the ArrayList sent from getAllRides()
  + RideAdapter(context, rides)
    - Assigns rides to this.rides
  + +getView(position, convertView, parent): View
    - Inflates a single item\_ridelist.xml layout
    - Retrieves a HashMap’s containing one ride’s details from the ArrayList
    - Assigns data to the layout’s textViews
    - Creates two onClick listeners:
      * rideDetails – The portion of the layout that includes the trip’s addresses and departure time. When this is clicked, it creates an Intent to launch the Ride Details Activity, passing the tripId as an Extra.
      * messageButton – This shows the number of messages a trip has. When this section is tapped, the app starts the Single Message, or Single Thread Activity, passing the tripId as an extra.
    - Finally, this method returns the finished layout, and moves on to the next.