Android Final Project Documentation

BLOOD DRIVE

*Harika Bandaru & ADARSH BODINENI*

2017

Contents

[1.Navigation Drawer 2](#_Toc500523401)

[2.Fragment 5](#_Toc500523402)

[3.Activities 7](#_Toc500523403)

[4.RecyclerView 9](#_Toc500523404)

[5.View Pager 11](#_Toc500523405)

[6. Screen Orientation Changes 15](#_Toc500523406)

[7.Host your data in Cloud 15](#_Toc500523407)

[7.1 Key-capabilities of FireBase: 15](#_Toc500523408)

[8. Data 17](#_Toc500523409)

[8.1 Real API data (Facebook Graph API) 17](#_Toc500523410)

[8.2 Library usage (OKHttp; urllib; firebase; Retrofit) 17](#_Toc500523411)

[9 Security to Cloud data 17](#_Toc500523412)

[10. System 18](#_Toc500523413)

[10.1 Service 18](#_Toc500523414)

[11. Coordinator Layout 20](#_Toc500523415)

[12. Custom Views 22](#_Toc500523416)

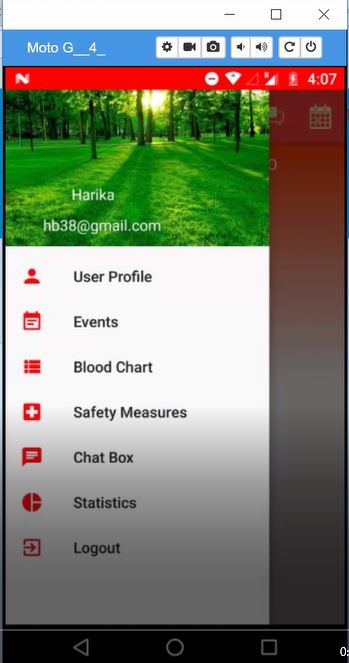
[13. Camera and Gallery 24](#_Toc500523417)

[14. Flipbook Style 27](#_Toc500523418)

[15. Maps 28](#_Toc500523419)

[16.Multiple Things on Action Bar 30](#_Toc500523420)

# 1.Navigation Drawer

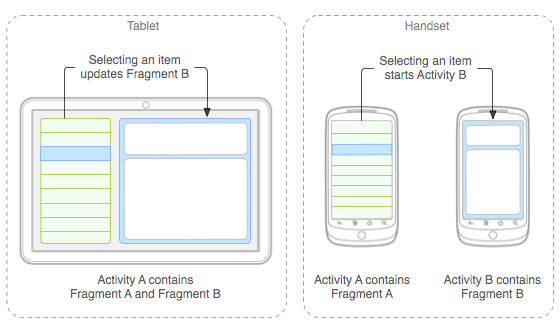
* The navigation drawer is a panel that displays the app’s main navigation options on the left edge of the screen.
* It is hidden most of the time, but is revealed when the user swipes a finger from the left edge of the screen or, while at the top level of the app, the user touches the app icon in the action bar.
* User interface which is the layout file is “Drawer Layout” as the root view of your layout. Drawer layout contains two child’s: fragment and a Frame Layout



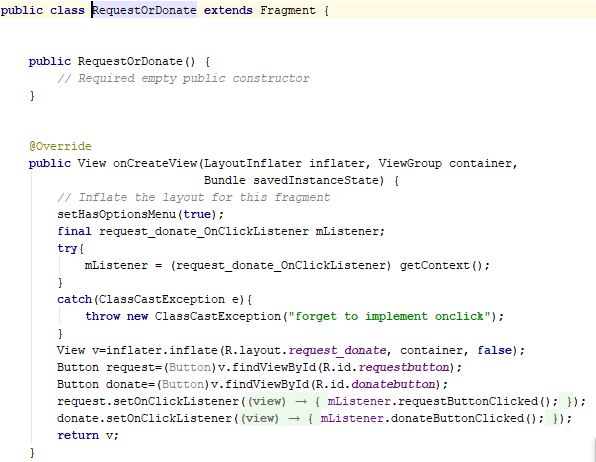


* Main content populated at runtime using the fragment and a navigation view for navigation drawer.
* Initialize the drawer list and have to handle navigation clicks by redirecting to particular activity using intents and to fragments using transaction manager calling the newinstance method of fragment
* Listen for close and open events for navigation drawer.
* Uses navigation drawer adapter to populate the navigation drawer list.

# 2.Fragment

* Fragments provides the reusability capability and populate multiple layouts in a single activity.
* Fragments must be embedded in an activity although fragments have it’s own life cycle it will be affected by the life cycle of activity.
* Fragments lies in the view group of activity’s view hierarchy.
* onCreate contains essential information about fragments current state can retain whenever necessary.
* onCreateView() called for the first time to lay the layout for the fragment.
* A fragment should have a default constructor that is used by the android OS to populate the fragment after recreating it.
* Performing transaction from activity to fragment using fragmentManager and beginning the transaction.
* To view the activity layout uses “getActivity()” instance.

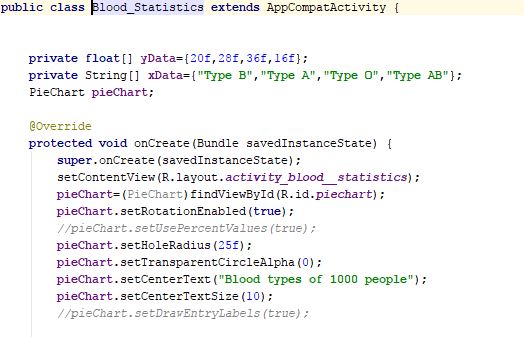




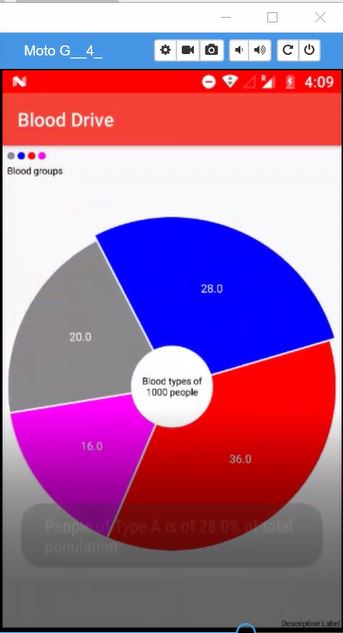


# 3.Activities

* Activities are the fundamental building blocks of applications on the android platform.
* Operation changes smoothly without distributing the user Interface
* User data is not lost during activity transaction.
* Activity has its own lifecycle and even the lifecycle of fragment depends on this.
* On launching an app “Launcher” activity will be launched which will be declared in the manifest file.



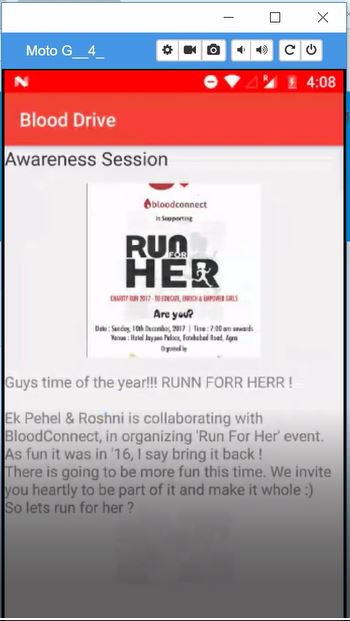




# 4.RecyclerView

* RecyclerView overcomes the drawbacks of listViews.
* Will use recyclerview with recycle adapter provided by the firebase to populate data to recycle viewholder.
* We can do animations in recyclerview while populating the data.



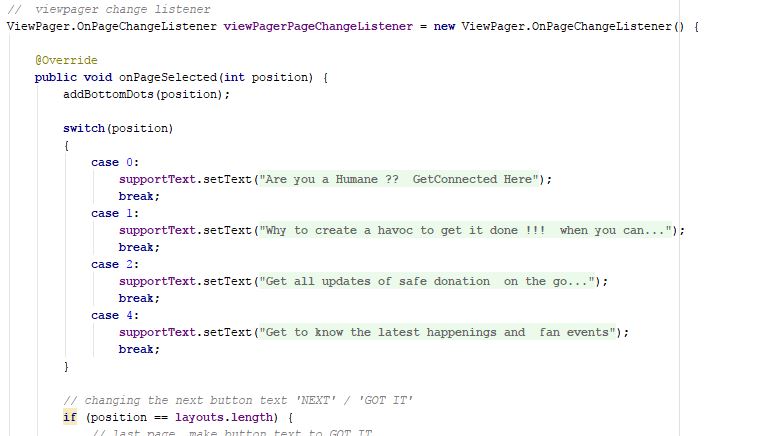


# 5.View Pager

* Flip the pages by recycling the holder of the page layout.
* Page adapter is used to fill in the data for the holder provided by the viewPager.
* Two adapters are provided for the viewPager.They are FragmentPagerAdapter and FragmentStatePagerAdapter.
* Views which are annotated with the [ViewPager.DecorView](https://developer.android.com/reference/android/support/v4/view/ViewPager.DecorView.html) annotation are treated as part of the view pagers 'decor'. Each decor view's position can be controlled via its android:layout\_gravity attribute
* Depending on build SDK we can set the decor values.
* Used Viewpager with viewpager adapter in the introduction of the app.









# 6. Screen Orientation Changes

Will provide only portrait mode to the application to reduce the configuration issues that keep changing from SDK version to SDK version and some issues will be created during runtime.

To make app work efficiently will lock the app’s configuration change to potrait.

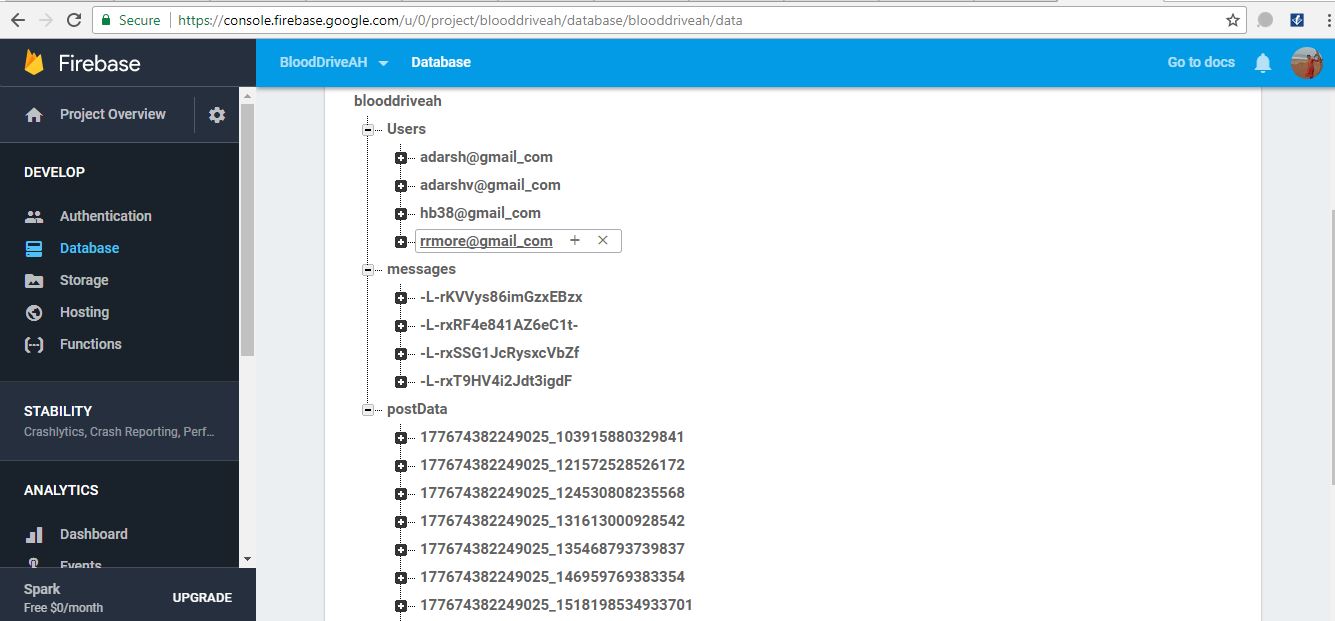
* There are two ways to do it :
* setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_POTRAIT);
* android:screenOrientation="portrait"

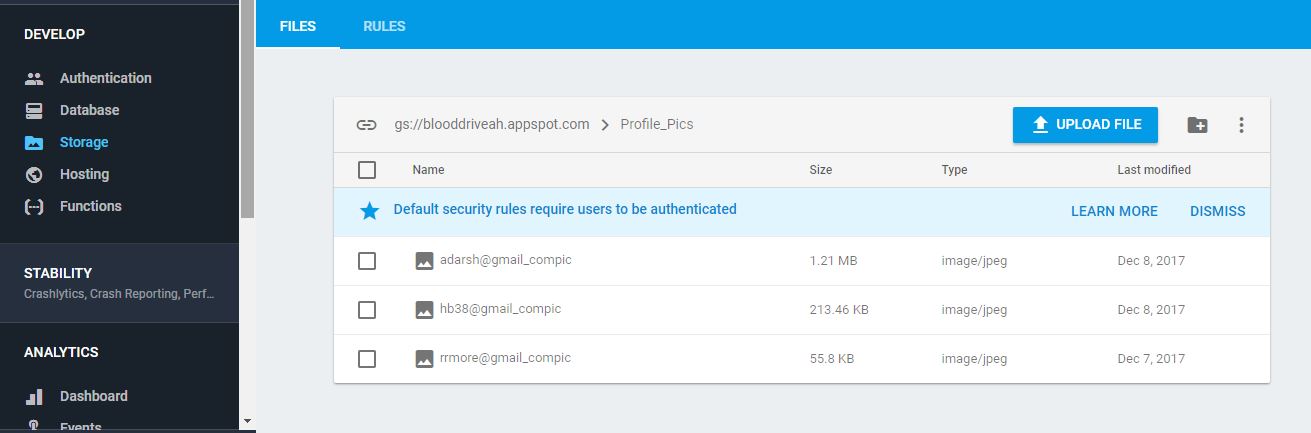
# 7.Host your data in Cloud

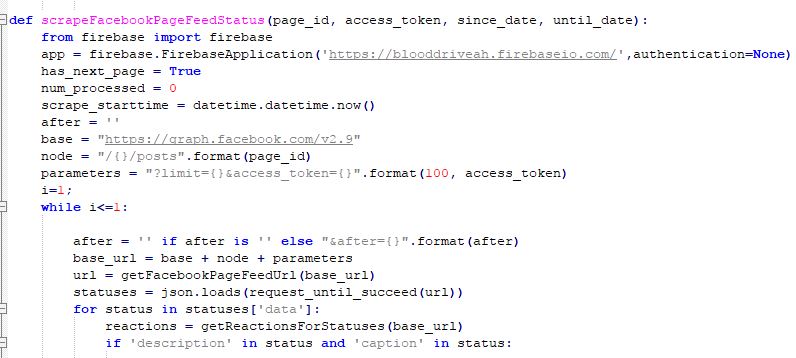
* Google provides powerful tool for cloud storage, nosql “key-value” based database and user authentication with security rules that helps developer to chase the functionalities of application quickly.
* In this app we used cloud storage to store the profile picture of the user’s.
* Used **cloud database** to store the user details.
* Implemented a real time **group messaging** functionality using the firebase real time messages.
* Implemented to store the data retrieved using “**facebook graph API**” into the firebase to reduce the latency to fetch data from API calls.
* The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client.
* **Authentication** feature provided by the Firebase with email-password and gmail

## 7.1 Key-capabilities of FireBase:

* Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds.
* Firebase apps remain responsive even when offline because the Firebase Realtime Database SDK persists your data to disk. Once connectivity is reestablished, the client device receives any changes it missed, synchronizing it with the current server state.
* The Firebase Realtime Database can be accessed directly from a mobile device or web browser; there’s no need for an application server. **Security and data validation** are available through the Firebase Realtime Database Security Rules, expression-based rules that are executed when data is read or written.

**Database**

**Storage**



# 8. Data

## 8.1 Real API data (Facebook Graph API)

* The Graph API is the primary way to get data out of, and put data into, Facebook's platform. It's a low-level HTTP-based API that you can use to programmatically query data, post new stories, manage ads, upload photos, and perform a variety of other tasks that an app might implement.
* The graph composed of :
  + **nodes** - basically "things" such as a User, a Photo, a Page, a Comment
  + **8edges** - the connections between those "things", such as a Page's Photos, or a Photo's Comments
  + **fields** - info about those "things", such as a person's birthday, or the name of a Page
* Construct query like graph.facebook.com/version/node/edge/fields/access-token
* Choose the option **Get User Access Token**.

## 8.2 Library usage (OKHttp; urllib; firebase; Retrofit)

**Used python-script to fetch data:**

* For smooth app usage and to reduce the latency on the user side used python script to fetch data and store that in firebase database.
* Used library **urllib** similar to retrofit and more powerful even the retrofit uses this under the hood.
* Used firebase library to populate data fetched into the firebase database.
* Calculated the reactions by considering different kind of reactions introduced by the facebook like “**like', 'love', 'wow', 'haha', 'sad', 'angry**”

# 9 Security to Cloud data

* Used Firebase Realtime Database Security Rules to secure your data.



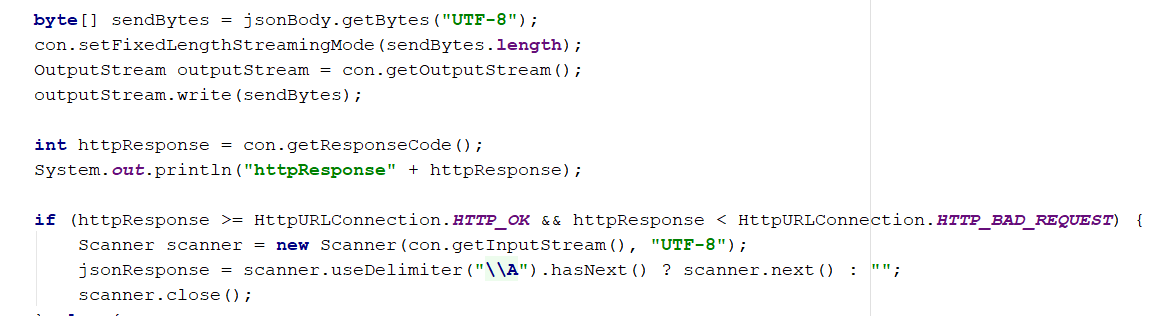
# 10. System

## 10.1 Service

* We have used Async tasks for sending notifications to other users.
* The service have been used because these tasks are long running tasks. There might be situation where more number of users might register in the application. To send notifications to all of them, the async task service is required.
* The one signal API is used to push notifications from one user to multiple users.
* This is achieved by registering each user with a tag in one signal.

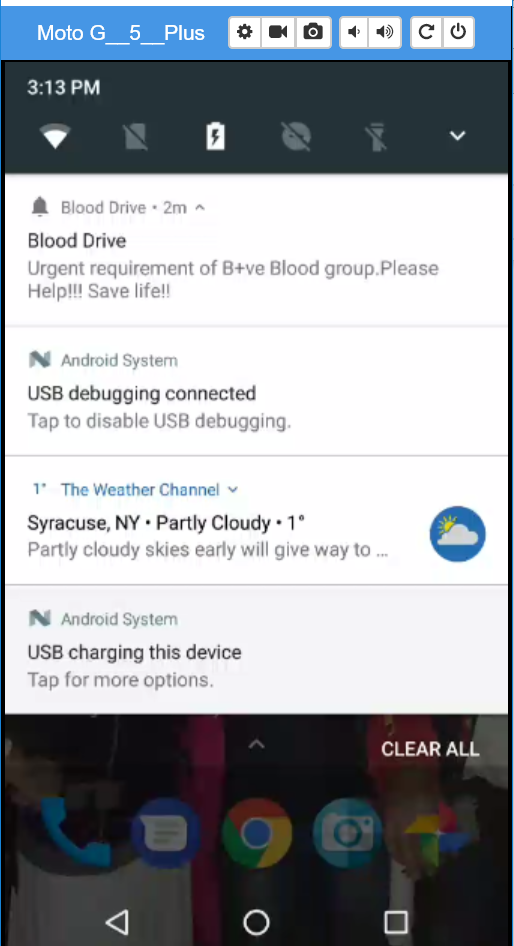
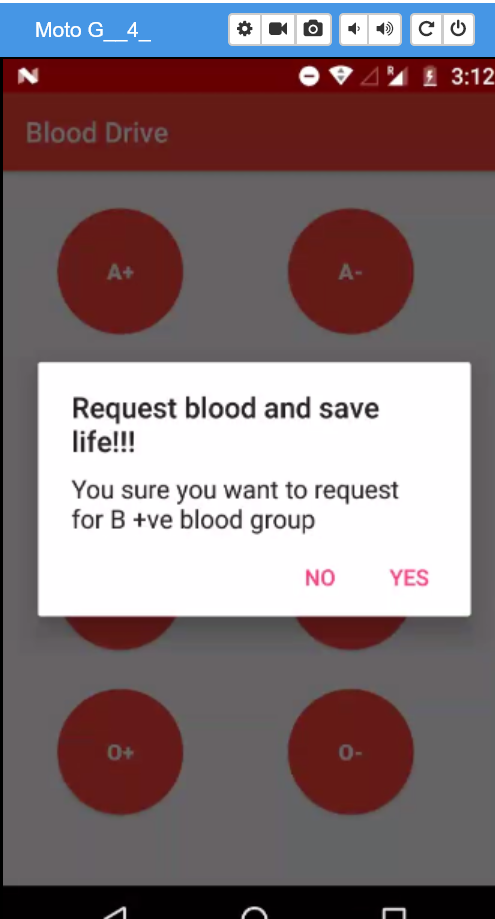
##### Push notifications are a communication channel built into every mobile device sold today.

****

****

**Screenshot:**

Sending notification requesting blood from one phone(user) and receiving the notification on other phone(user).

****

# 11. Coordinator Layout

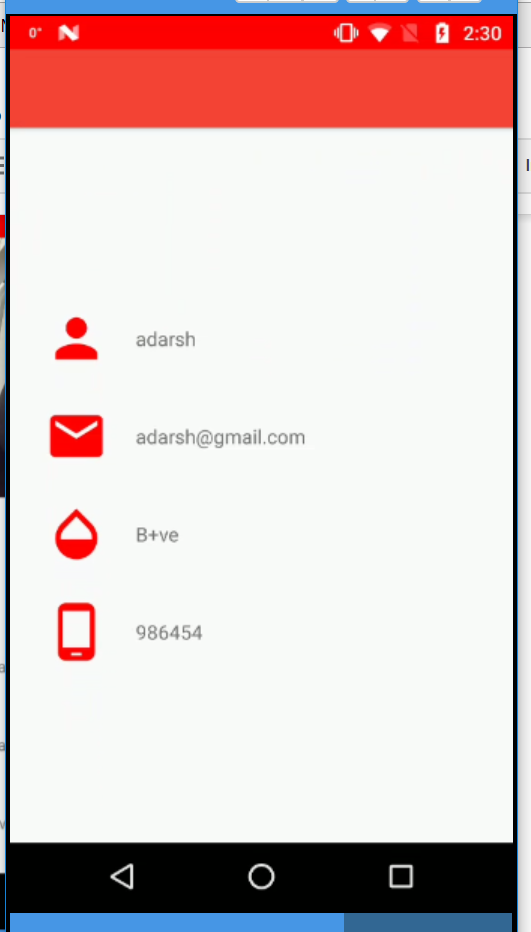
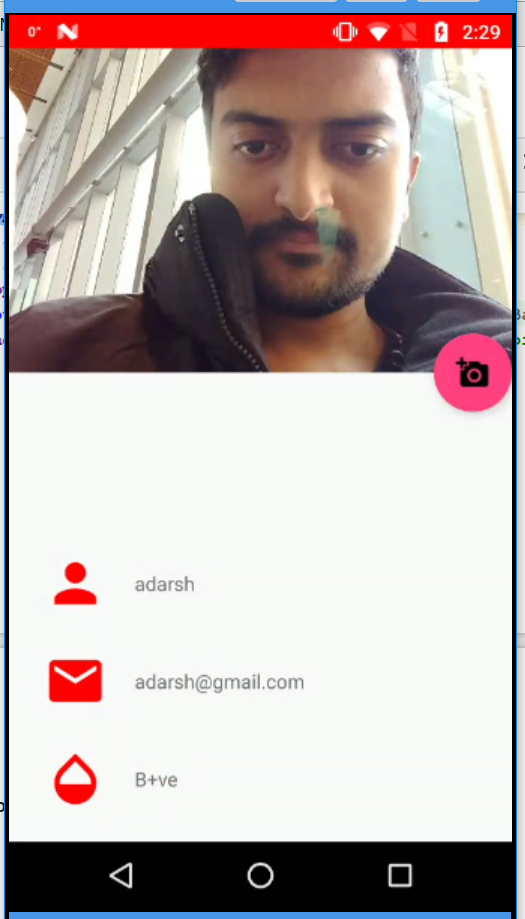
* We have used coordinator layout in the User profile page.
* We have used collapsing toolbar layout inside this coordinator layout . The user profile image will contract or expand which is embedded in this collapsing toolbar layout.
* The other frame of the coordinator layout is filled with a nested scroll view fragment that contains user profile details.

****

**The user profile details are displayed in nested scroll view.**

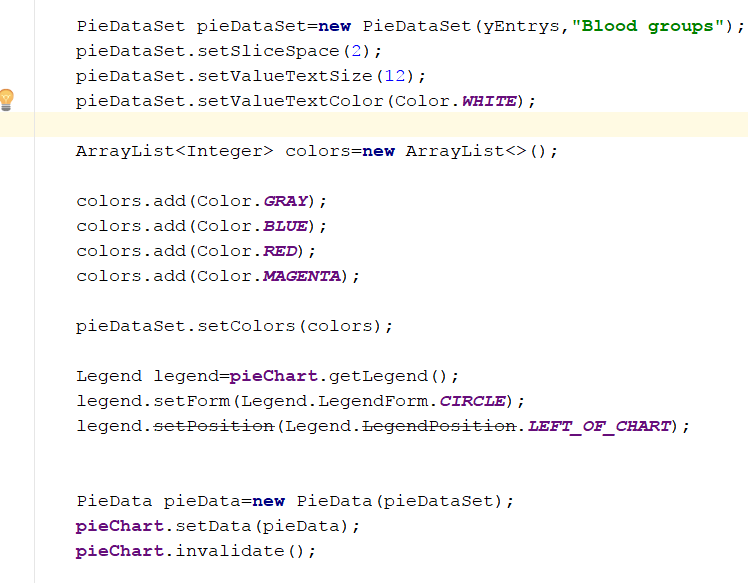
****

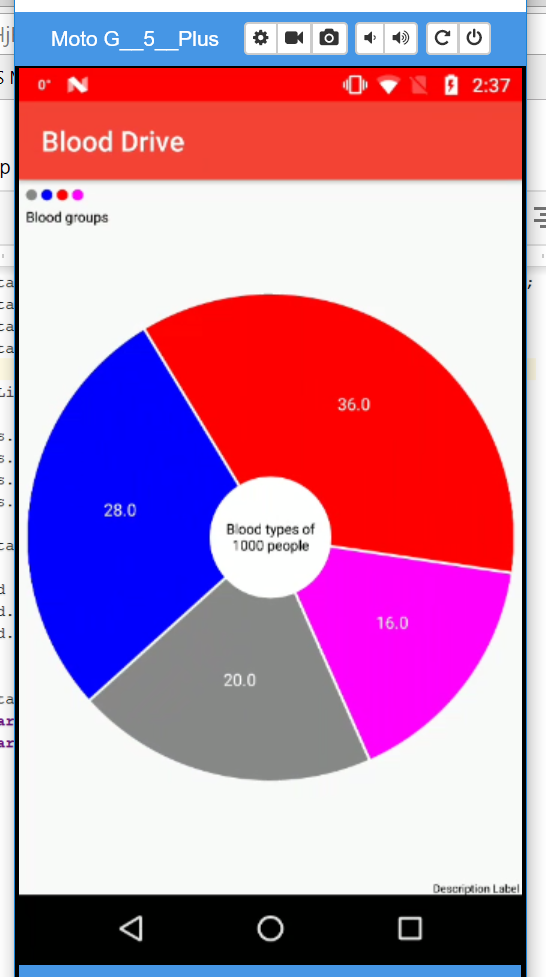
**The screen before and after collapsing.**

****

# 12. Custom Views

* We have used custom view pie chart to display the percentage of people with each blood group.
* The pie chart is designed in such a way that it can be rotated either way around 360 degrees.
* Each slice of pie chart is set with different colour.
* We have used legend to get data from each slice of the chart and displayed on the left side top screen.

****

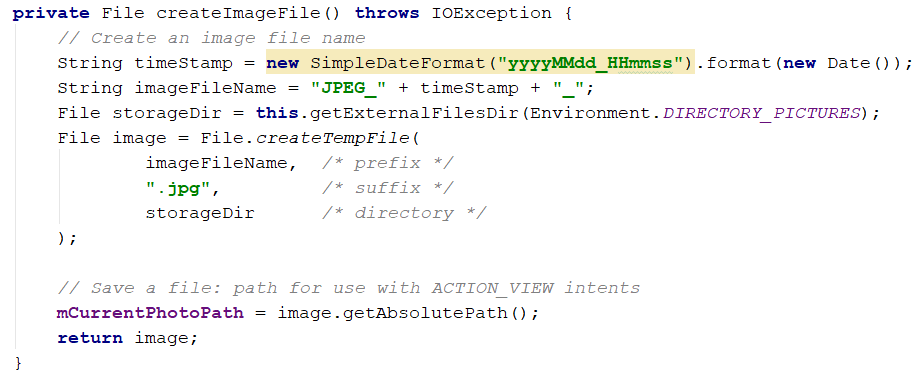
****

# 13. Camera and Gallery

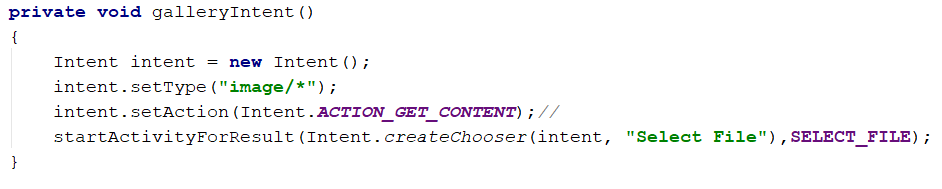
* We have used camera and Gallery to update profile picture of the user.
* The user on clicking on the floating button in the  user profile page, he will get an option to select either “take a photo” or  “choose from library”.
* In “Take a photo” option , if the user is using the camera for first time, the application will ask for permissions to use the camera. Once he takes the picture, the picture will be stored in firebase database and loaded into the imageview.
* In “Choose from library” option, the user can choose the image from gallery and can upload it. The image will store in firebase and displayed on the image view.

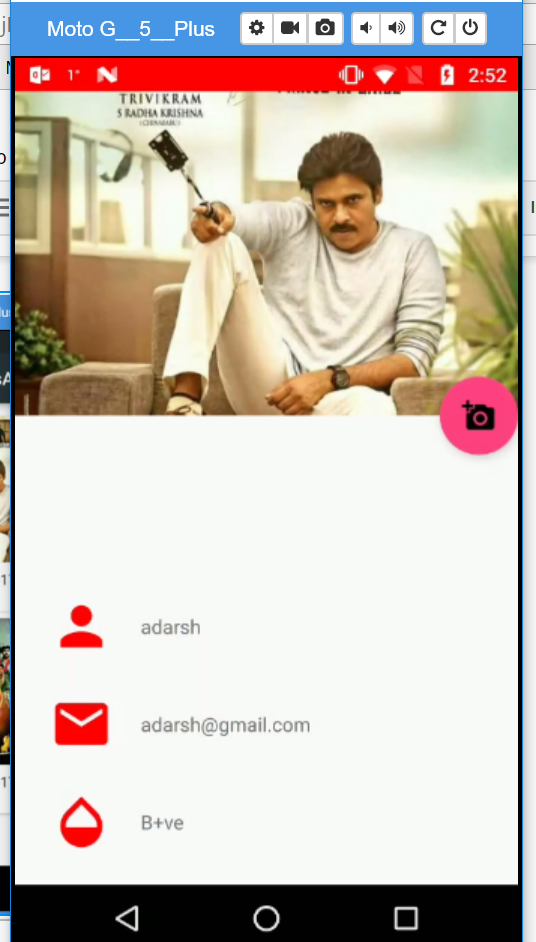
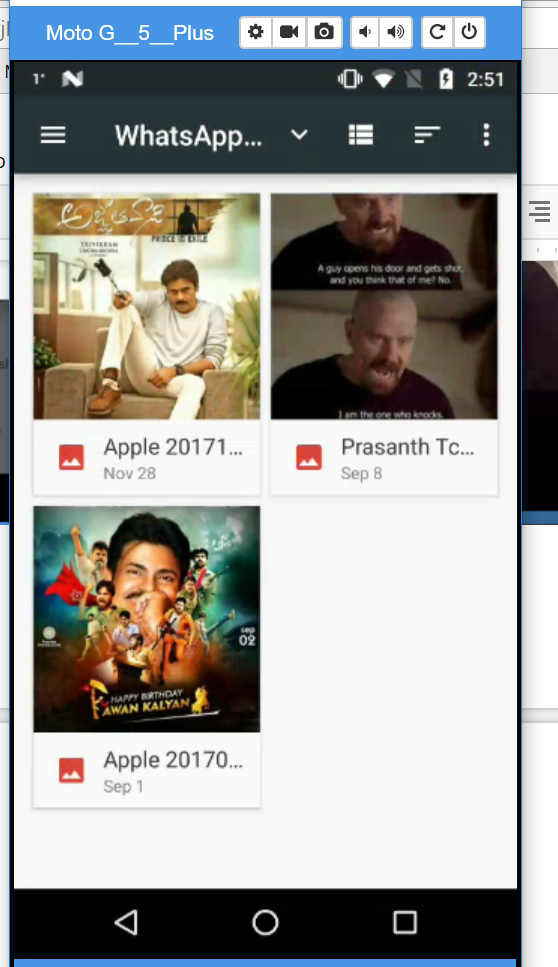
The task**s** are accomplished in the following way:

Creating image file

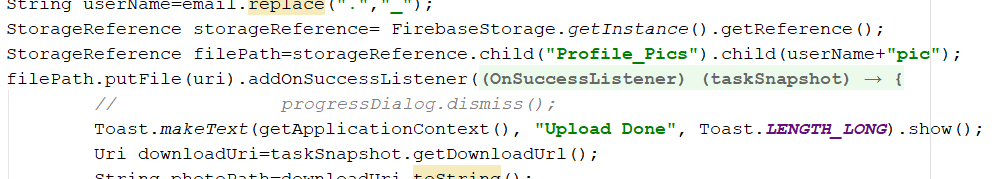
****

**Gallery Intent**

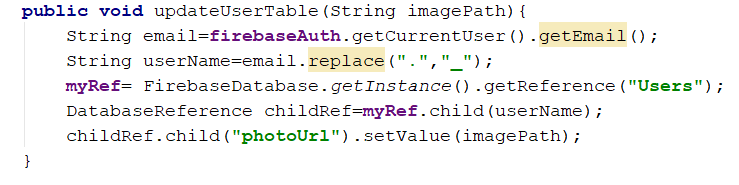
****

****

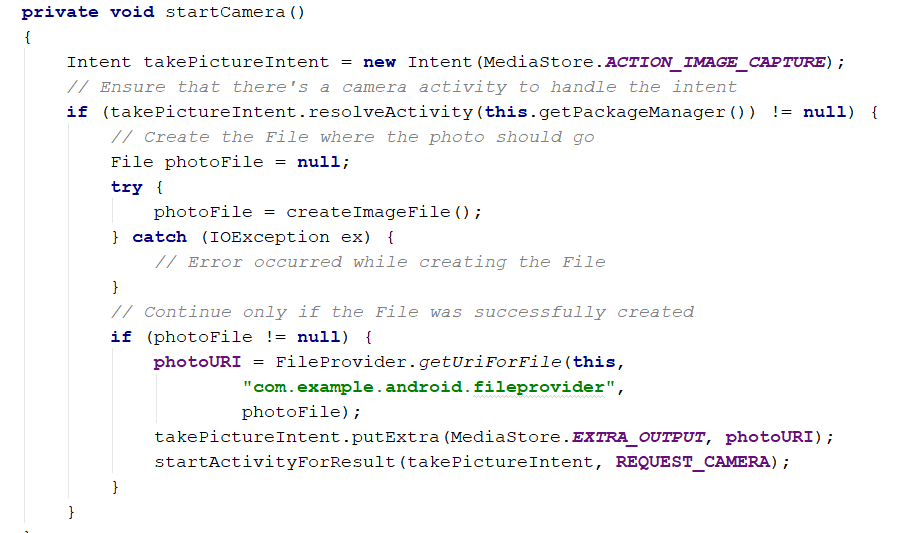
**Storing in firebase storage**

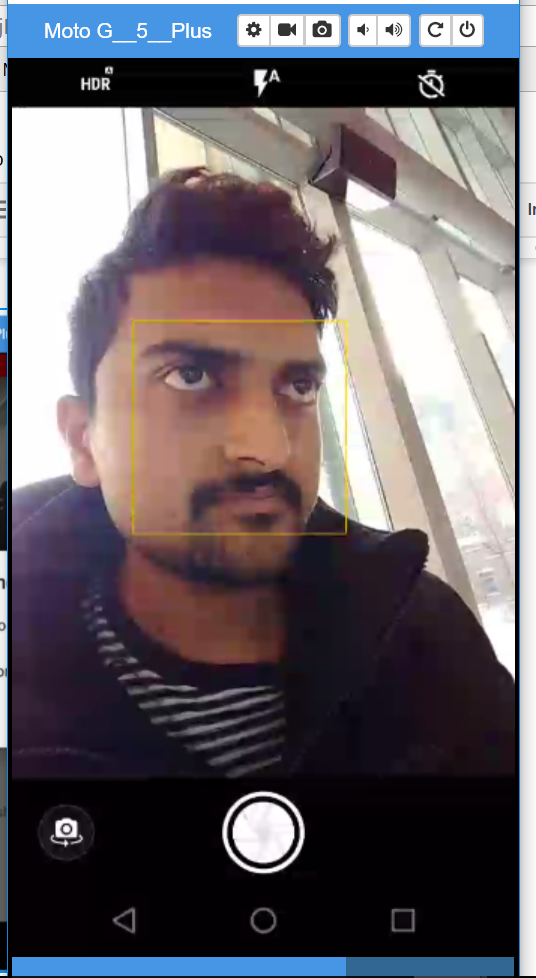
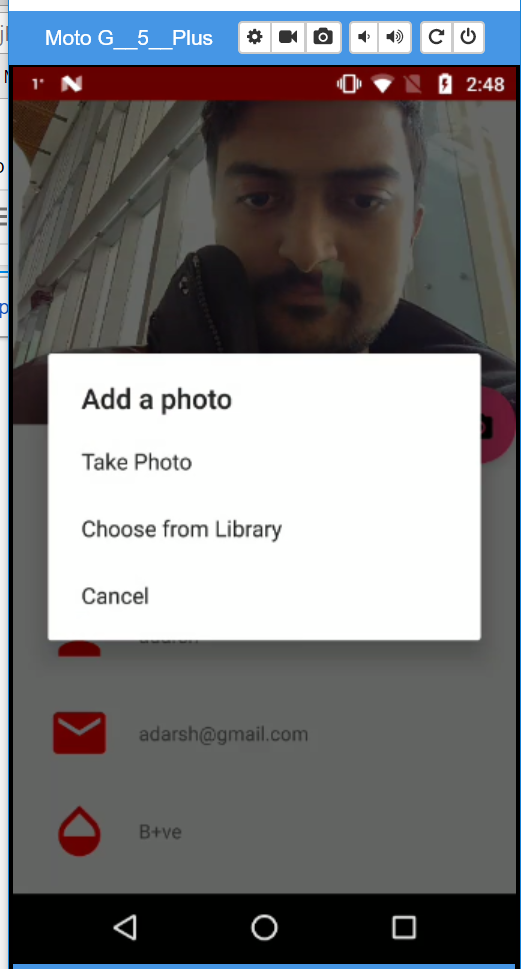
****

Updating user table

****

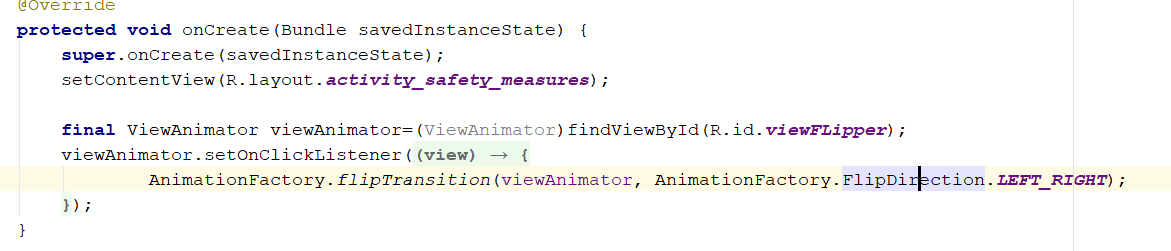
Camera Intent:

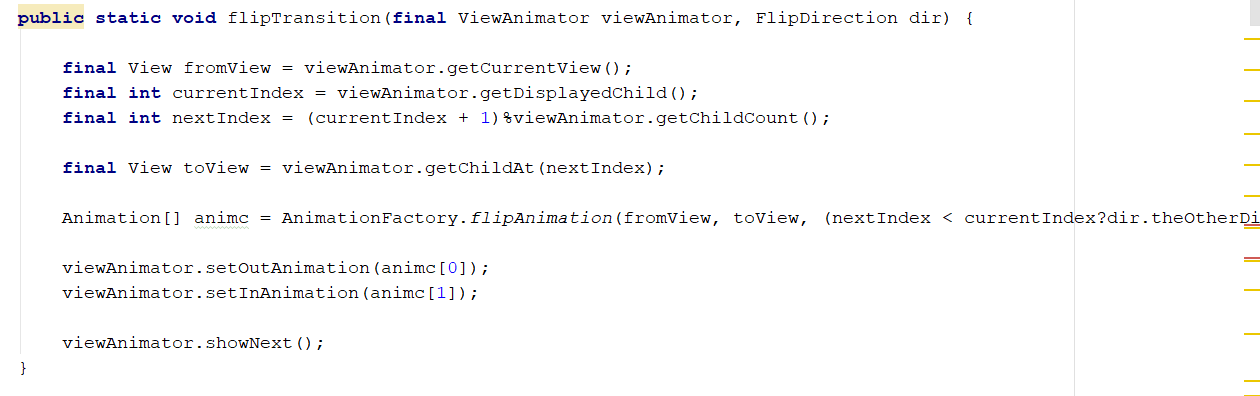
****

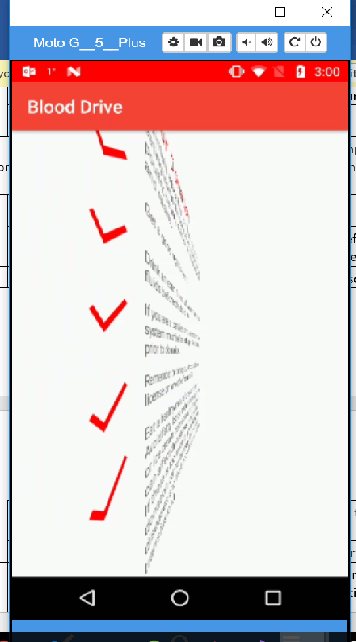
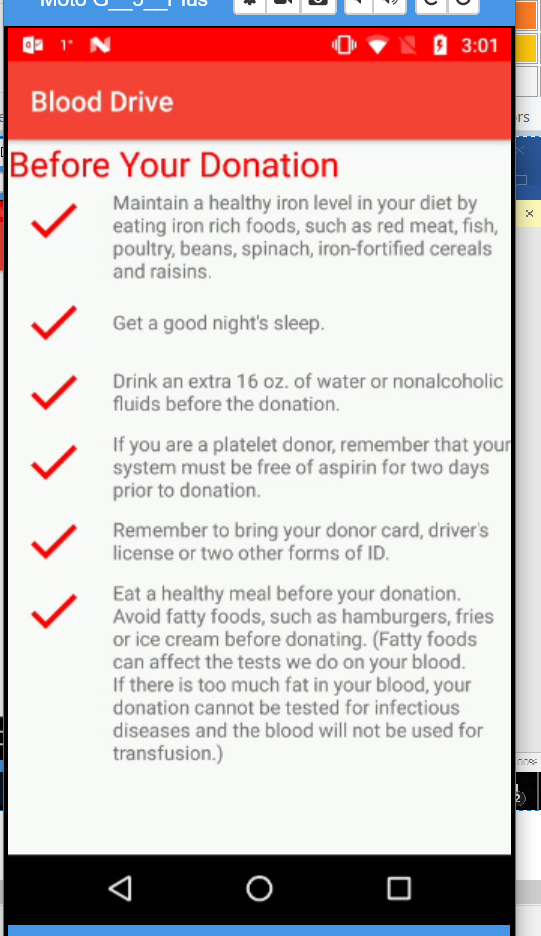
****Screenshots:

# 14. Flipbook Style

* The flipbook style has been implemented between two fragments in the safety measures activity.
* The two fragments are placed in the view flipper and on click on a fragment, the other fragment will be loaded with flipbook animation.





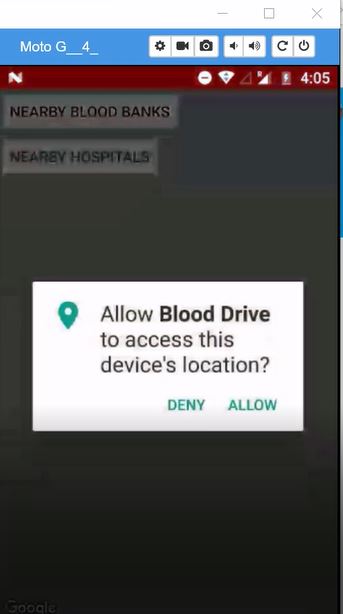
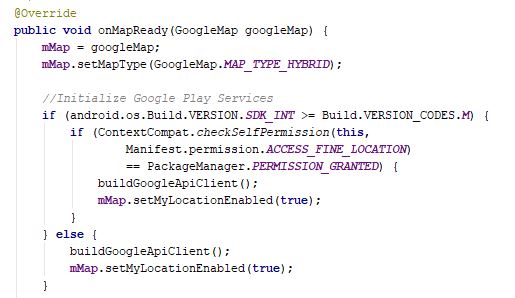


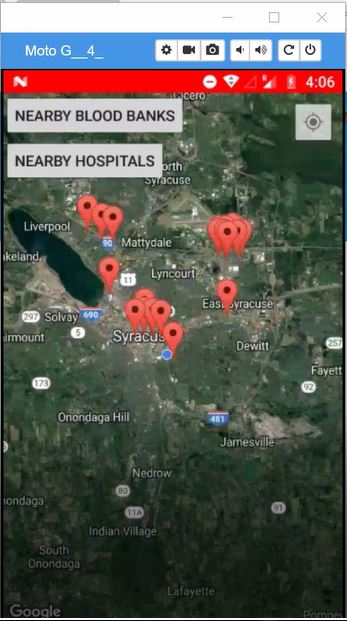
# 15. Maps

* Implemented to fetch the nearby hospitals and blood banks using the API provided by Google **“MAPs API” and “Retrofit library”.**
* Add user permission for internet access.
* Will build retrofit library to fetch data will add markers to the fetched positions depending on the search.
* Functions that need to be taken care are: onMapReady (); onConnected; onConnectionSuspended; onLocationChanged.
* buildGoogleApiClient() is used to build GoogleApiClient on top.

GoogleApiClient provides following functionalities:

1. GoogleApiClient.Builder is used to configure client.
2. .addConnectionCallbacks provides callbacks that are called when client connected or disconnected.
3. .addOnConnectionFailedListener covers scenarios of failed attempt of connect client to service.
4. addApi adds the LocationServices API endpoint from Google Play Services.
5. mGoogleApiClient.connect(): A client must be connected before excecuting any operation.





# 16.Multiple Things on Action Bar

* Inserted multiple things on action bar to have an easy of navigation for the user.

