**Life Saver Blood Bank**

Software Requirements Specification

Version-1.6

Date: August 23, 2020

Mohan Pratapa

Venkata Hareesh Bhuma

Raghunandan Kumar Naishadam

Sumanth Gorantla

Venkat Prudhvi Dommaraju

Submitted in partial fulfilment

Of the requirements of

CSIS 44-691 Graduate Directed Project 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 6/10/2020 | Version 1.1 | Mohan Pratapa, Raghunandan Naishadam | Updated requirements and added ER Diagram |
| 6/11/2020 | Version 1.2 | Raghunandan Naishadam,  Venkat Prudhvi Dommaraju | Updated document with prototypes |
| 6/16/2020 | Version 1.3 | Bhuma Venkata Hareesh, Sumanth Gorantla | Updated Donor, Acceptor and  Admin Functionalities. |
| 6/29/2020 | Version 1.4 | Mohan Pratapa,  Hareesh Bhuma,  Raghunandan Naishadam,  Venkat Prudhvi,  Sumanth Gorantla | Update the document with the technical manual explaining the overview of the code and its functionality. |
| 6/30/2020 | Version 1.5 | Mohan Pratapa,  Hareesh Bhuma,  Raghunandan Naishadam,  Venkat Prudhvi,  Sumanth Gorantla | Update the document with the End-user manual explaining how to use the application with screenshots. |
| 8/23/2020 | Version 1.6 | Mohan Pratapa,  Hareesh Bhuma,  Raghunandan Naishadam,  Venkat Prudhvi,  Sumanth Gorantla | Added Project Objectives |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

**Table of Contents**

1. Introduction
   1. Purpose
   2. Scope
   3. Definitions, Acronyms, and Abbreviations
   4. References
   5. Overview
2. General Description
   1. Product Perspective
   2. Product Functions
   3. User Characteristics
   4. General Constraints
   5. Assumptions and Dependencies
3. Specific Requirements
   1. External Interface Requirements
      1. User Interfaces
      2. Hardware Interfaces
      3. Software Interfaces
      4. Communications Interface
   2. Functional Requirements
   3. Use Cases
   4. Class/Objects
   5. Non-Functional Requirements

3.5.1. Performance

3.5.2. Reliability

* + 1. Availability
    2. Security
    3. Portability
  1. Inverse Requirements
  2. Design Constraints
  3. Logical Database Requirements
  4. Other Requirements
  5. Prototypes (for complete project)
  6. Use Case Diagrams

1. Design

4.1. ER diagram

4.2. GUI

1. Analysis Models

5.1. Data Flow Diagram

5.2. Sequence Diagram

1. Technical Manual
2. End-user Manual

## Introduction

**1.1 Purpose:**

The purpose of our application is to let the users who need blood in emergencies can log in to the Life Saver Blood Bank application and can request the donors for blood. Once the request is placed by the users based on the availability of the blood group.

**Objectives:**

* Finding a blood donor is a challenging issue in almost every country, this project will help the users who need it in an emergency.
* To allow the probable recipients to make search and match the volunteer donors, and request the blood.
* To provide efficient donor and bloodstock management functions to the blood bank by recording the donor and blood details.
* To provide synchronized and centralized donor and bloodstock database.

**1.2 Scope:**

A new user opens the application and registers using his/her details. If the user is already registered, he/she needs to login using his/her credentials. When the emergency occurs, for all users i.e. existing and new users the application displays the available donors where a request can be made online. The request is fixed once the user clicks the submit button. Later he/she can view the requests which were placed by him/her. Users can go through his/her profile in settings where he/she can edit his/her credentials including his/her mobile number. Once the user request is submitted then the blood donor can accept based on his/her availability.

**1.4 References:**

1. <https://www.ncbb.org/>

## 2. General Description

**2.1 Product functions:**

Initially to use the app user should signup and create an account using an email id or phone number. Then the user has to login into the application using the credentials. Based on the emergency, all the donors will be displayed. By clicking on the required blood group, the user can see all the details of that particular donor. Users can select a particular blood group and request the blood. Then the request will be sent to the donor where they can either accept or reject the request. Once the request is accepted, the user’s request is confirmed.

**3. Specific Requirements**

**3. Specific Requirements**

Life Saver Blood Bank is an application where the users can get responses from the donors online. Here, Donors are the clients and the application is the link to make the requests for users. So, the main purpose of the application is to make the users find the blood in case of an emergency with ease.

The application is built according to progressive enhancement principles. Responsive design for varying screen size, Version compatibility for devices, more touch flexibility, and less keyboard usage in the application are the features of the application. The other main feature is the application and data security.

**3.1 External Interface Requirements**

**3.1.2 Hardware Interfaces (computers and servers)**

* This application must be able to install on any android device with a version greater than or equal to 5.0
* Android devices must have Internet services.
* The user should give access to the Internet once the app gets installed on his/her android device.

**3.1.3 Software Interfaces (programming Lang)**

* Android Studio has to be installed on all the Laptops.
* Java has to be installed on all the Laptops.
* Java will be used as a programming language to develop the application.

**3.2 Functional Requirements:**

* This app will help the users to find blood in case of an emergency.
* This app will have Users and Donors.

**Donor Functionalities**:

* **Login (sign up, sign in, password retrieval, logout)**: Donor can use his/her email address and password to log into the website. If the Donor password is not working, request a new password by entering the email address in the password retrieval form.
* **Profile Management:** Profile management saves your personal information such as password, donor blood group in a set of files.
* **Request management:** Request management allows catalog items to be requested and fulfilled based on the defined flows.
* **Notification:** The notification API lets our app send notifications that are displayed outside the page and it will send information to the donor even if the application is idle or in the background.
* **FAQ (email and textbox): The** FAQ page includes a series of questions that are commonly asked by Donors

**Accepter Functionalities:**

* **Login (sign up, sign in, password retrieval, logout):** Acceptor can use his/her email address and password to log into the website. If the Acceptor password is not working, request a new password by entering the email address in the password retrieval form.
* **Profile Management:** Profile management saves your personal information such as password, acceptor blood group in a set of files.
* **Search:** The search page will be useful to find compatible blood donors who can receive a blood request.
* **View donors:** View donor’s page will display all the donors' information whose blood group same as the acceptors.
* **FAQ (email and textbox): The** FAQ page includes a series of questions that are commonly asked by Acceptors.
* **Live chat:** Live chat allows acceptors to communicate with donors in real-time rather than having to speak with them on the phone, acceptors on the app can have a live interaction with donors in a chatbox.
* **Send request:** Send request page allows acceptors can send the request to the donors whose blood group same as the acceptors.
* **Accept/deny response:** When acceptors receive a request in the Accept/deny response page, acceptors can respond to the request by clicking on the accept or decline checkbox. If the user accepts the request then the donor will receive a meeting invitation to his/her registered mail address.
* **Notification:** The notification API lets our app send notifications that are displayed outside the page and it will send information to the acceptors even if the application is idle or in the background.
* **Scheduling (time, transportation):** By using the scheduling page donors can schedule an appointment with the acceptors.
* **Gift option:** By using the Gift option page acceptor send a gift message or greeting card to the donor.

**Admin Functionalities:**

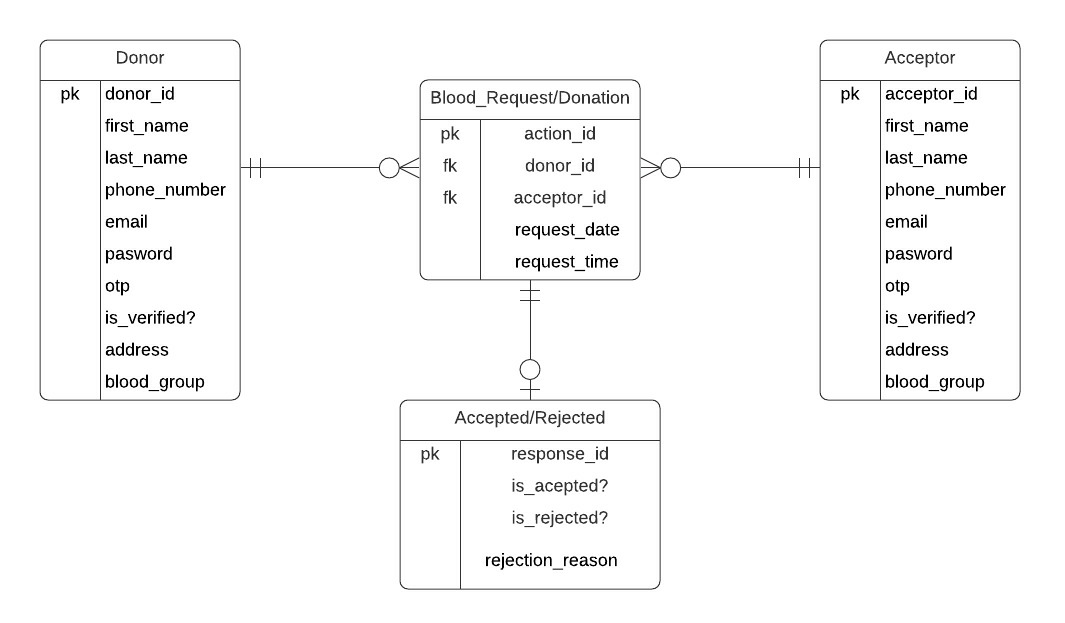
* **Manage Users:** The Manage Users screen will add, change, and delete the users. Additionally, Admin users can search for users, change user details, and manage what users have permission to see and do in the app.

**3.5.1 Performance:**

* This app should be user-friendly.
* This app should have less response time.
* This app should not crash in any situation.

**4. Design**

**4.1 ER Diagram:**

****

**Description:**

If the user creates his account make an action such as requesting and accepting blood donation requests, he should either be donor or recipient. One acceptor can make as many requests as required and so as the donor. So we established mandatory one to optional many relationships between Blood\_request/donation, donor, and acceptor entities. If the donor accepts/rejects the requests then there should be a donation request, so the relationship is mandatory one to optional between Blood\_Request/Donation to Accepted/rejected entities.

**4.2 GUI:**

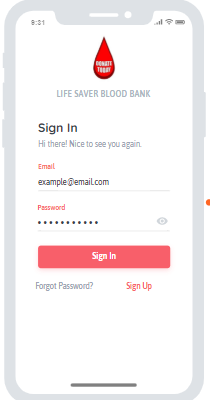
**1. Launch Screen:**

This is the launch screen of our application. It will be displayed whenever the application is launched.

****

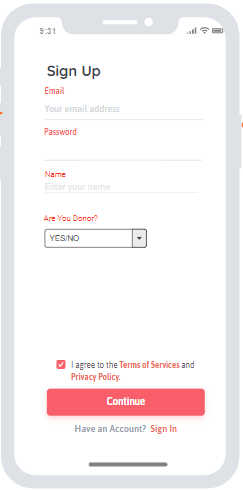
1. **Sign-in Screen:**

This is the Sign-in screen where the user has to log in with their respective credentials.

****

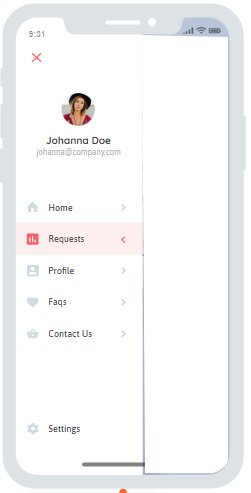
1. **Sign Up Screen:**

This screen is used for registration activity. Users and Donors will register here.

****

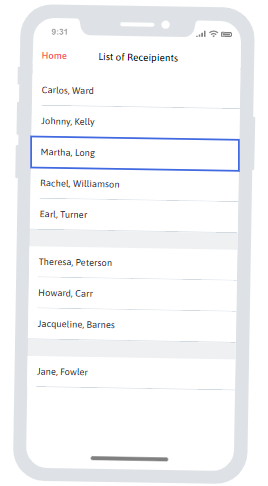
1. **Donor Home Screen:**

This is the screen where the complete details of the donors will be available and can view the requests done by the users.

****

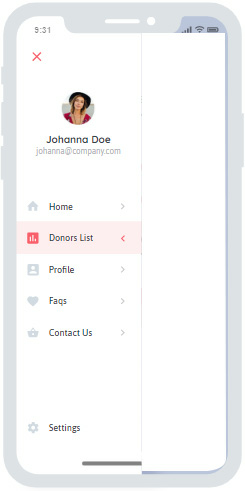
1. **Requests Screen:**

This is the screen where all the requests done by the users will be available.

****

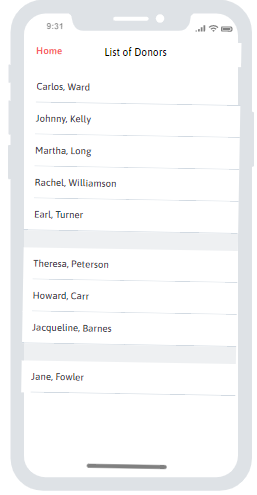
1. **Acceptor Home Screen:**

In this screen, the details of the acceptors can be able to view the list of donors for a particular blood group.

****

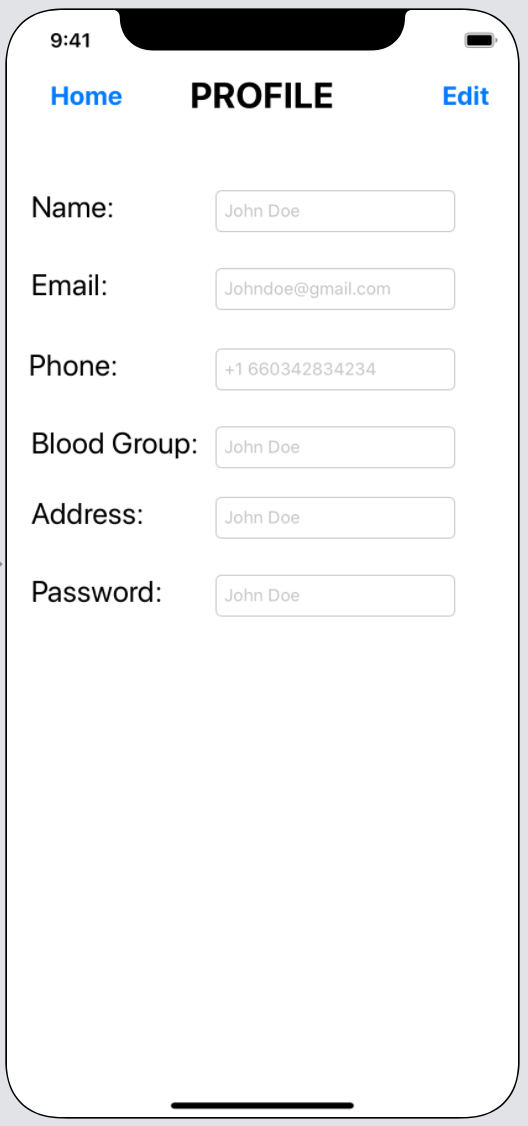
1. **Donor list Screen:**

The complete list of donors who are willing to donate blood can be seen here.

****

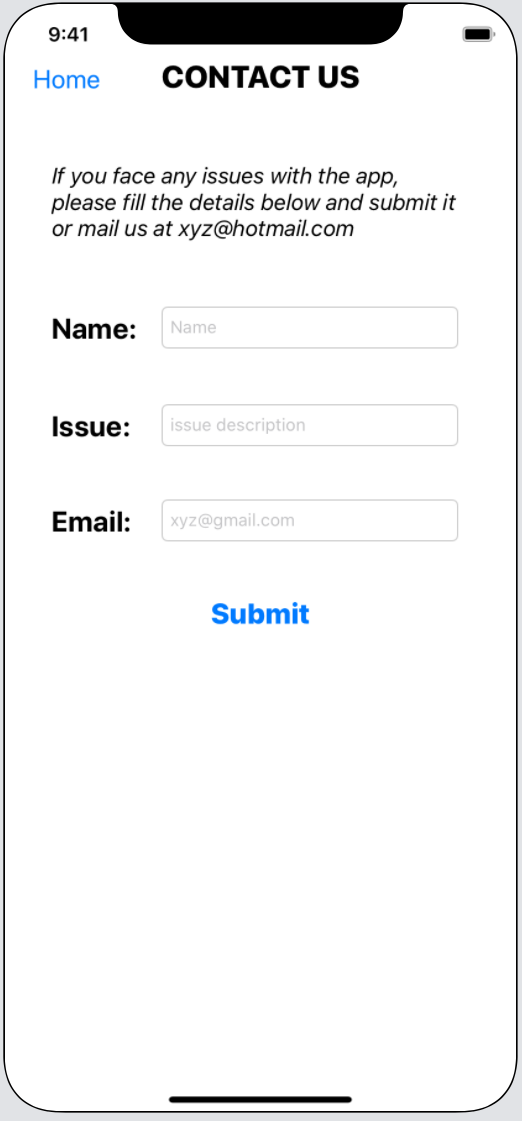
1. **Profile Screen:**

The users can edit their profile settings in the Profile screen.

****

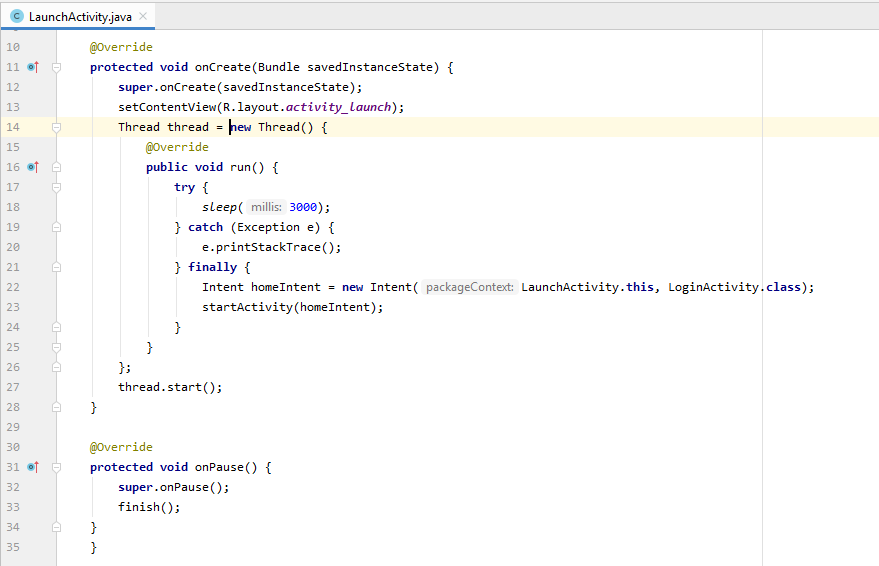
1. **Contact Us Screen:**

If the users face any issues with the app, they can file their issues in the Contact Us screen.

****

1. **Technical Manual**

**Launch Activity:**



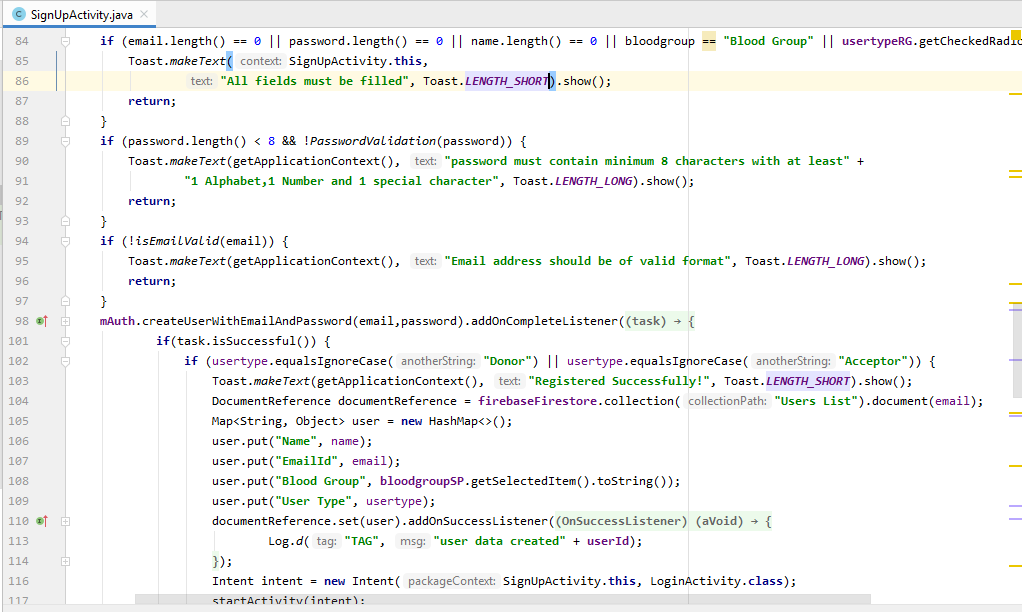
A thread is created in launch activity to display a splash screen (launch screen) for a fixed amount of time when the app starts. We might choose to show the splash screen for 3 seconds. However, if we want to show the splash screen for a variable amount of time (e.g. app startup time) we can specify it in the “try block”. After completion of the chosen time, it will be navigated to login activity.

**Login Activity:**



We have created an instance of FirebaseFirestore and defined an onClick method. In this method, we will retrieve email and password which is already stored in the database by using getText () and toString () methods. Once we click on sign in it will hit the database and verify the username and password, if it matches the task will successful. Once the task is successful it will pop up with a message "Logged in successfully!", else it will display "Entered Email or password does not match".

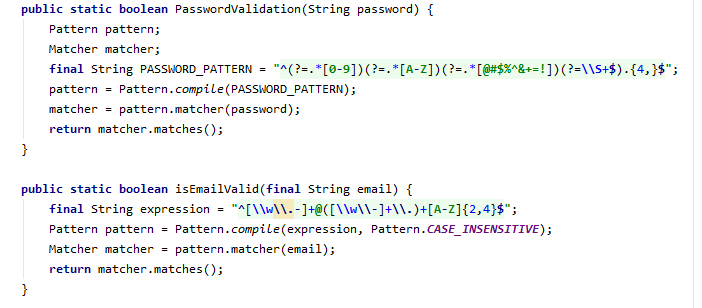
**Signup Activity:**



* The above code is for validation of text fields, spinner data, radio buttons while registering and storing all the user entered details into the database.
* A method i.e., createUserWithEmailAndPassword() performs whole functionality. A document reference is created and a collection called “Users List” is created in which all the data is stored using a hash map. An intent is created for navigation to login activity after a successful registration.

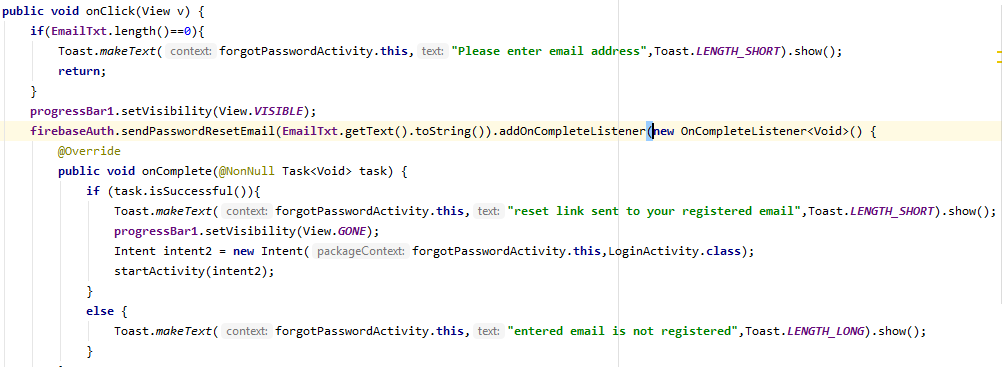


The above code is for adding blood groups in spinner which is a list.



The above code is for validation of the password and email address. Regular expressions are used to define a specific pattern while registering email and password. These methods return a Boolean value whether the password or email matches the pattern (password requirements) or not.

**Forgot Password:**



Above code snippet have two methods for handling forgot password.

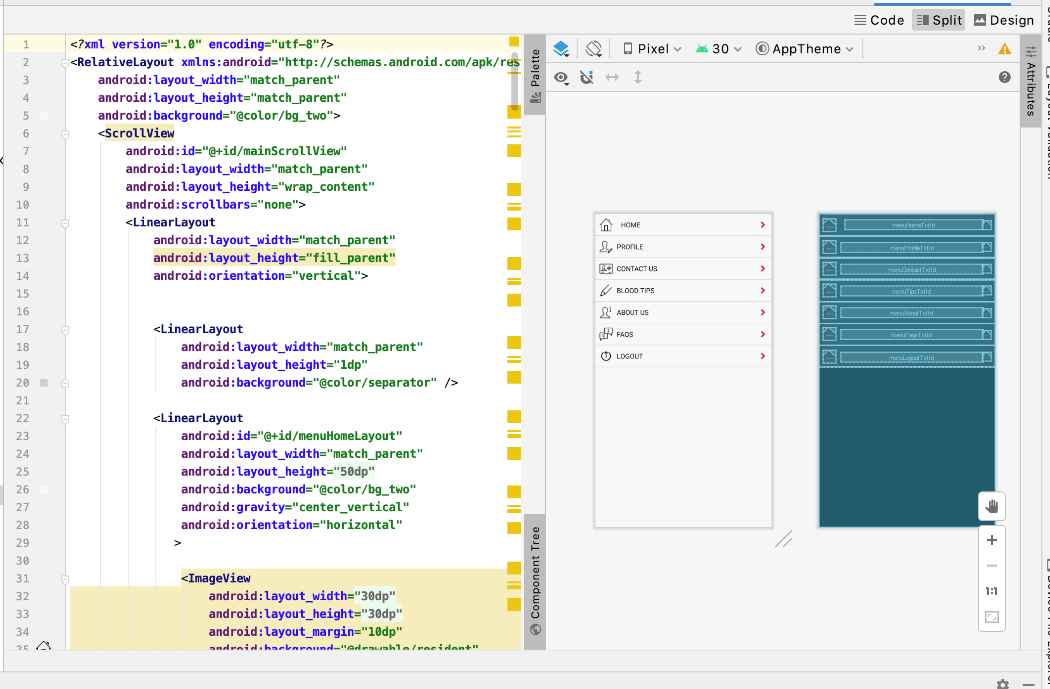
One method (onClick (View v)) which takes view as a parameter and handles validating registered email address. If a user forgets his password while logging in, then the application gives a chance to retrieve it. When the user press the forgot password on the sign-in page, the forgot password page is shown. On this page user is supposed to enter his registered email-id, first, it checks whether the user has entered an email address or not, if not, then a toast message will appear which says 'please enter email address'.

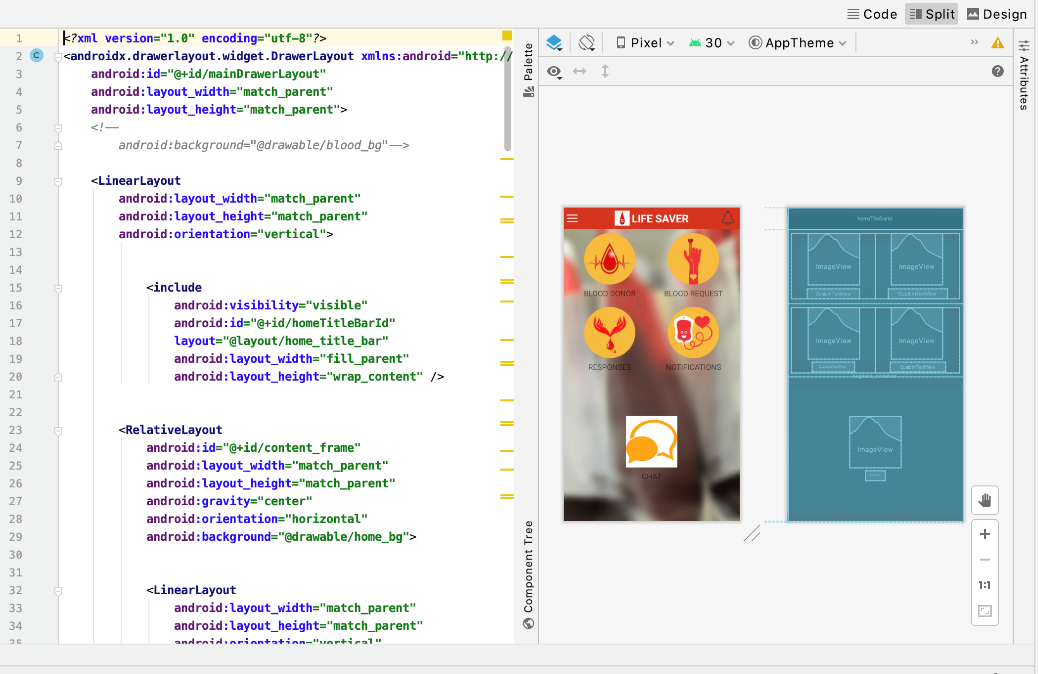
The second method (onComplete (task)), takes the task as a parameter. If the task is complete i.e., Once the registered email address is entered password reset link will be sent to the registered email and a toast message will be shown as ‘reset link sent to your registered email’. If the email address is not registered it shows a popup as ‘entered email is not registered’.

Once the user presses on reset link it will redirect him to reset the password page where the user needs to enter a new password. Once the password reset, the firebase will update the password in the database, and the user will be able to sign in to the application.

**Home page UI:**

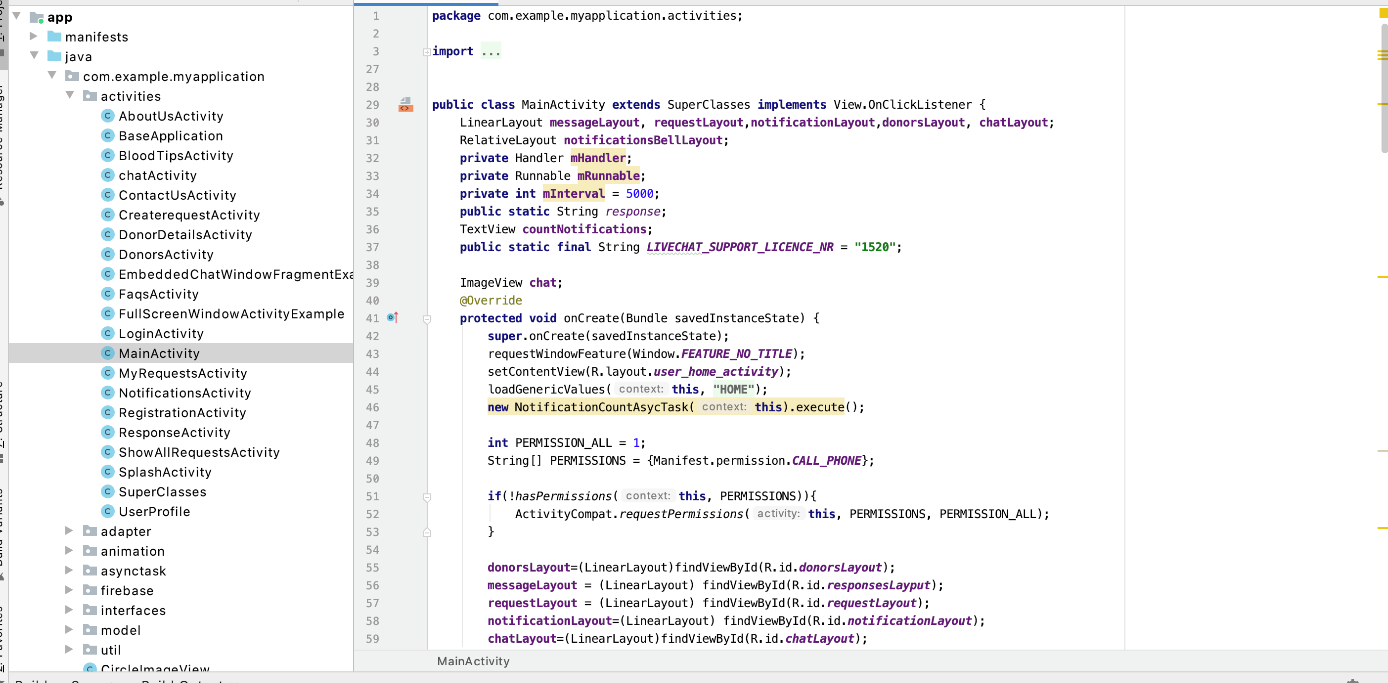
We have created image views and a linear layout for each menu item. We have added a menu layout for each task. With these menu layouts, users can navigate to respective tasks.





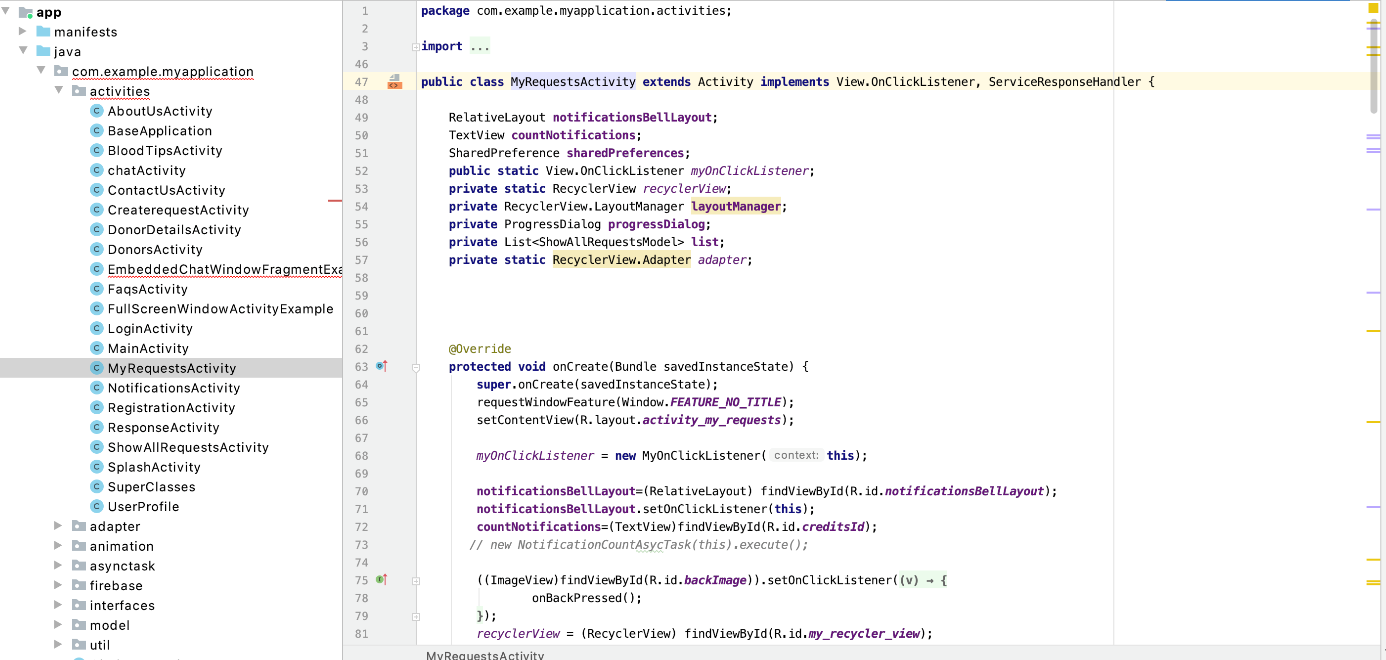
**Home page functionality:**

We have added functionalities where users can navigate to respective tabs. We have implemented on click listeners so that user can navigate. This class takes id as input and does navigation to the tab.

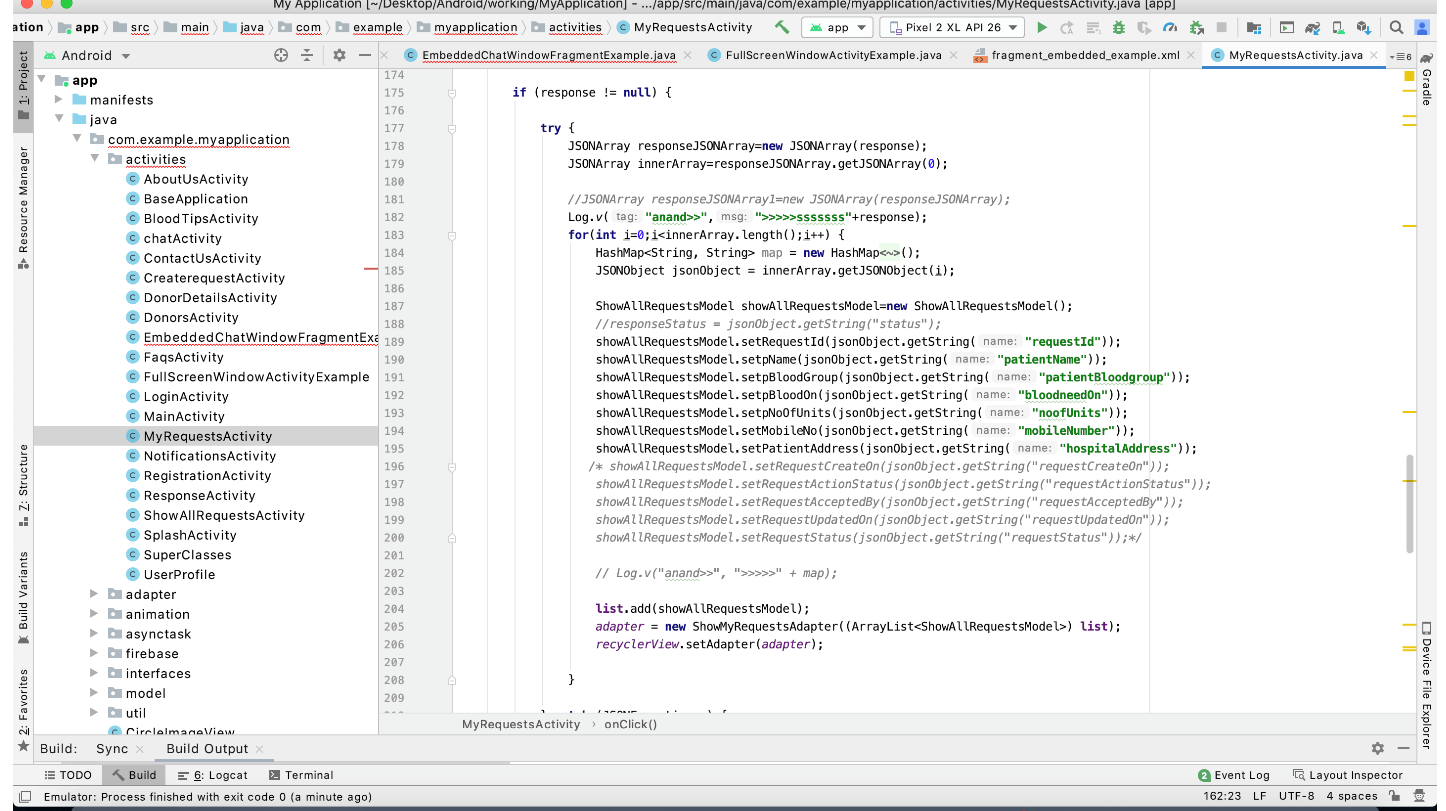


**Blood Request Functionality:**

We have my request activity where the user can create new blood requests. This class takes inputs such as name, blood type, etc as inputs and creates the requests.

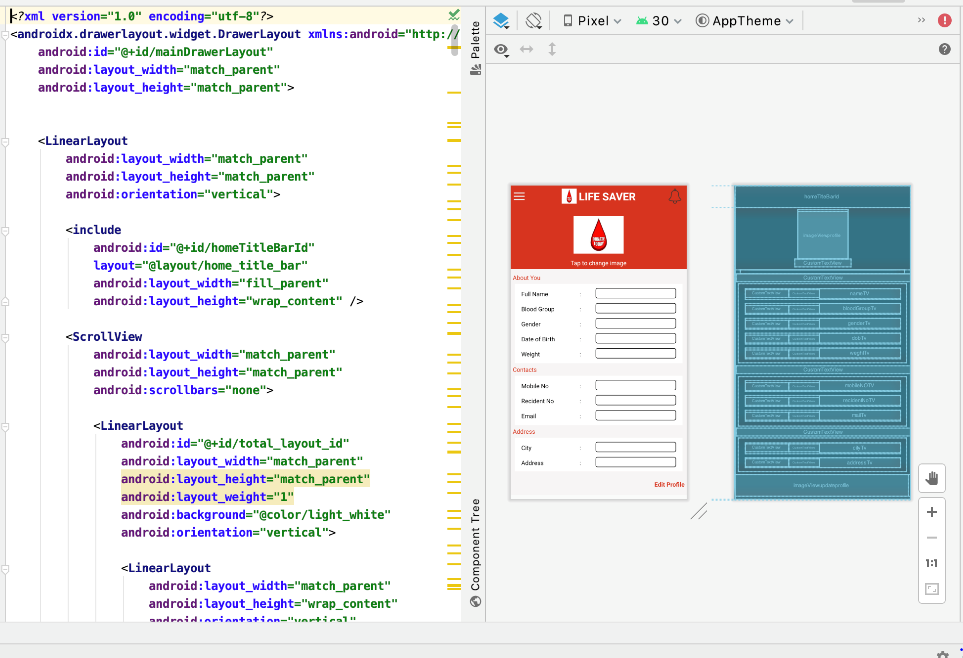


The below method fetches the data from the database i.e. if requests already exist then it will show those details else it will not show anything.



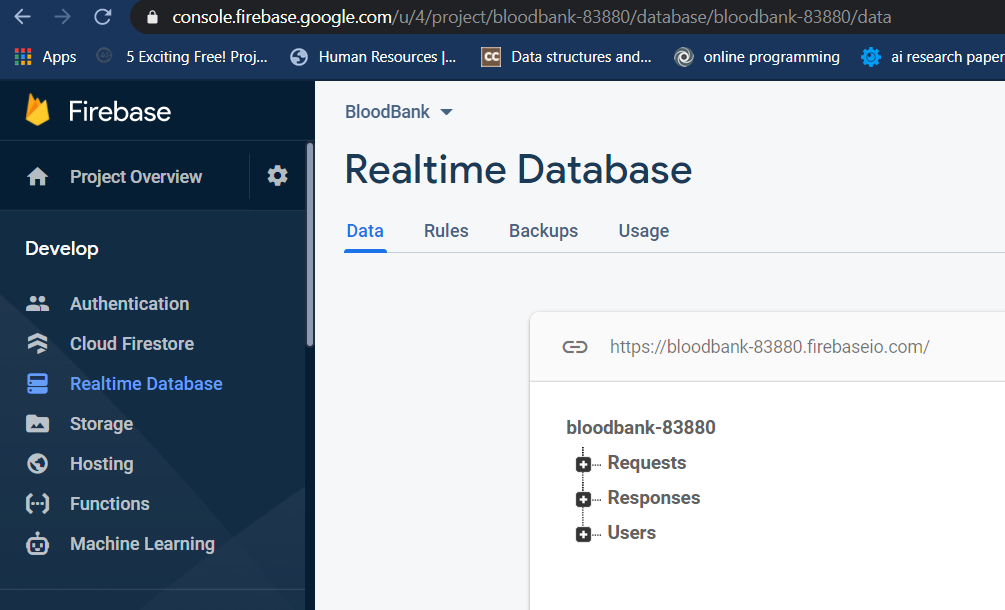
**Profile Page UI :**

We have added image view and text fields where the user can enter his details, we have added an edit profile button so the user can update his profile. We have regular labels where user can what has to be entered in the text fields.



**Database Functionality:**

All the databases will be stored in NoSQL format in the firebase. So the user has to create a google account to create the database. If the user creates his account make an action such as requesting and accepting blood donation requests, he should either be donor or recipient. One acceptor can make as many requests as required and so as the donor. So we established mandatory one to optional many relationships between Blood\_request/donation, donor, and acceptor entities. If the donor accepts/rejects the requests then there should be a donation request, so the relationship is mandatory one to optional between Blood\_Request/Donation to Accepted/rejected entities.



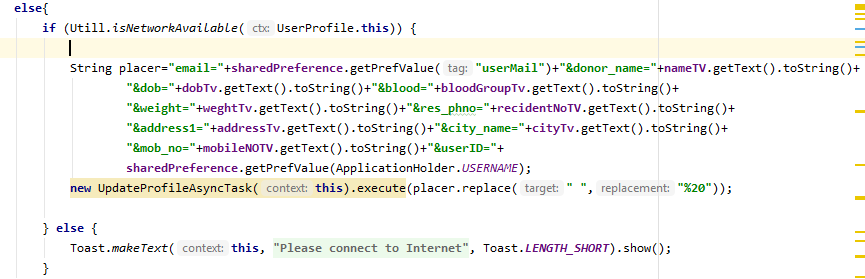
**Response Functionality:**



The above code is for fetching data from the database which is the responses data and displaying it for the user. We have parameters called snapshot and previous child names in our method. If a snapshot exists then with the id names the data is fetched from the database.

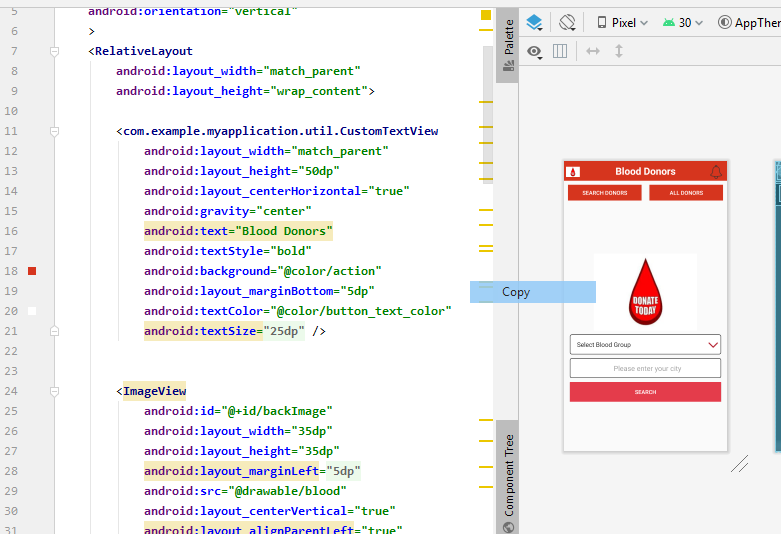
**Profile functionality:**

The following is the piece of code for updating user details in the application. The updated details will be updated in the database too.



**Blood Donor UI:**

The following is the piece of code for blood donor UI. This UI includes a search bar to search for blood donors based on city and blood group. It also has a tab which displays all the donors in the application.



**Search Functionality:**



The above is the block of code which is responsible for search functionality. I have used a method called showspecifiedDonors() where I filter donors using blood group and their city. If they both matches particular donor list will be displayed.

**Blood Request UI:**

<?xml version="1.0" encoding="utf-8"?>

|  |  |
| --- | --- |
|  | <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="match\_parent" |
|  | android:orientation="vertical"> |
|  | <!--android:background="@drawable/login\_bg"-->  (....) |
|  | <com.example.myapplication.util.CustomButton |
|  | android:id="@+id/request" |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_marginBottom="20dp" |
|  | android:layout\_marginTop="10dp" |
|  | android:background="@color/action" |
|  | android:text="Request" |
|  | android:textColor="#fff" /> |
|  | </LinearLayout> |
|  | </ScrollView> |
|  |  |
|  |  |
|  | </LinearLayout> |

In the Blood Requests UI, we have two buttons one of them is “MY REQUESTS” and the other button is “NEW REQUEST” which has their respective functionalities.

**Notifications:**

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

setContentView(R.layout.activity\_notification);

myOnClickListener = new MyOnClickListener(this);

sharedPreferences = getSharedPreferences("AuthenticationPref", Context.MODE\_PRIVATE);

((ImageView) findViewById(R.id.backImage)).setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

onBackPressed();

}

});

recyclerView = (RecyclerView) findViewById(R.id.my\_recycler\_view);

recyclerView.setHasFixedSize(true);

layoutManager = new LinearLayoutManager(this);

recyclerView.setLayoutManager(layoutManager);

recyclerView.setItemAnimator(new DefaultItemAnimator());

(….)

public static void responseRequests(List<NotificationModel> list) {

data = (ArrayList) list;

adapter = new NotificationsAdpter(data);

recyclerView.setAdapter(adapter);

}

}

The above code is a sample code for the notification functionality in our application here we are using recycler view to display all the notifications to the user and we have another method for the response request where if the user accepts the request it is stored as a response in the database.

**About Us UI:**

<com.example.myapplication.util.CustomTextView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:id="@+id/aboutUsTxt"

android:text="About Us"

android:textColor="#A90113"

android:textSize="20dp"

android:layout\_marginTop="10dp"

android:visibility="gone"

android:gravity="center\_horizontal"/>

(....)

<include

android:id="@+id/menuBarInclude"

android:layout\_marginTop="50dp"

layout="@layout/menulayout"

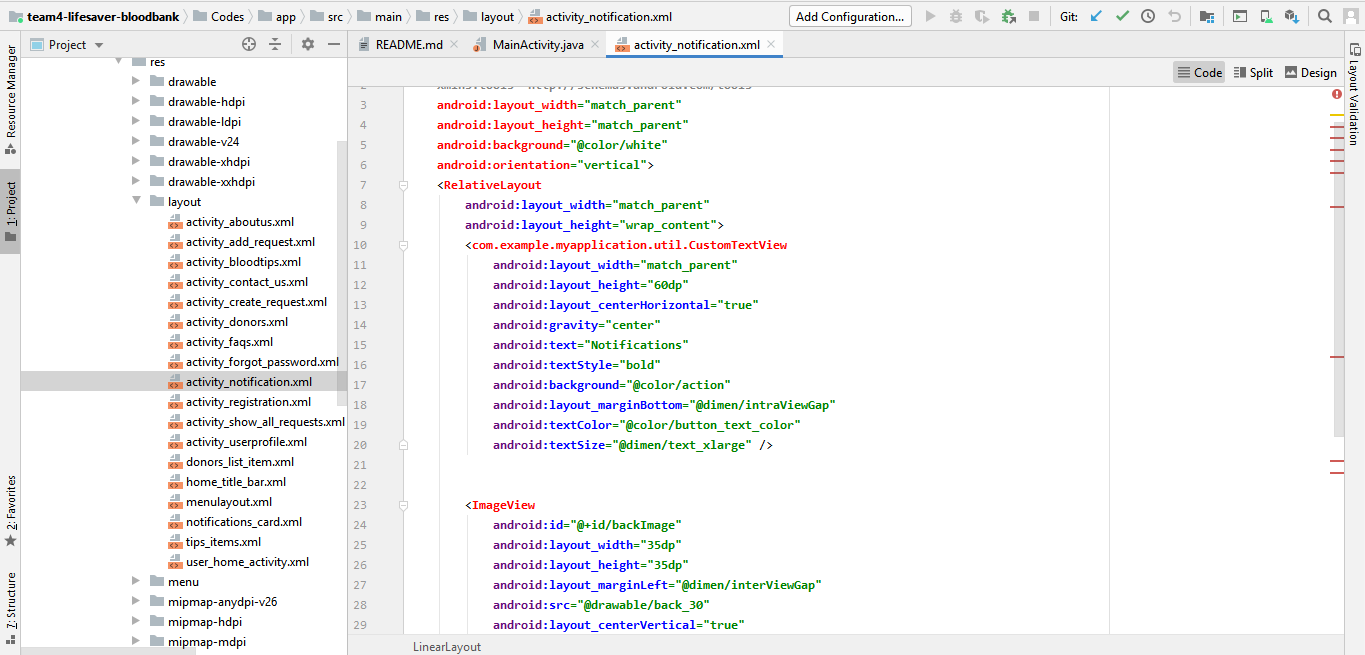
android:layout\_width="fill\_parent"

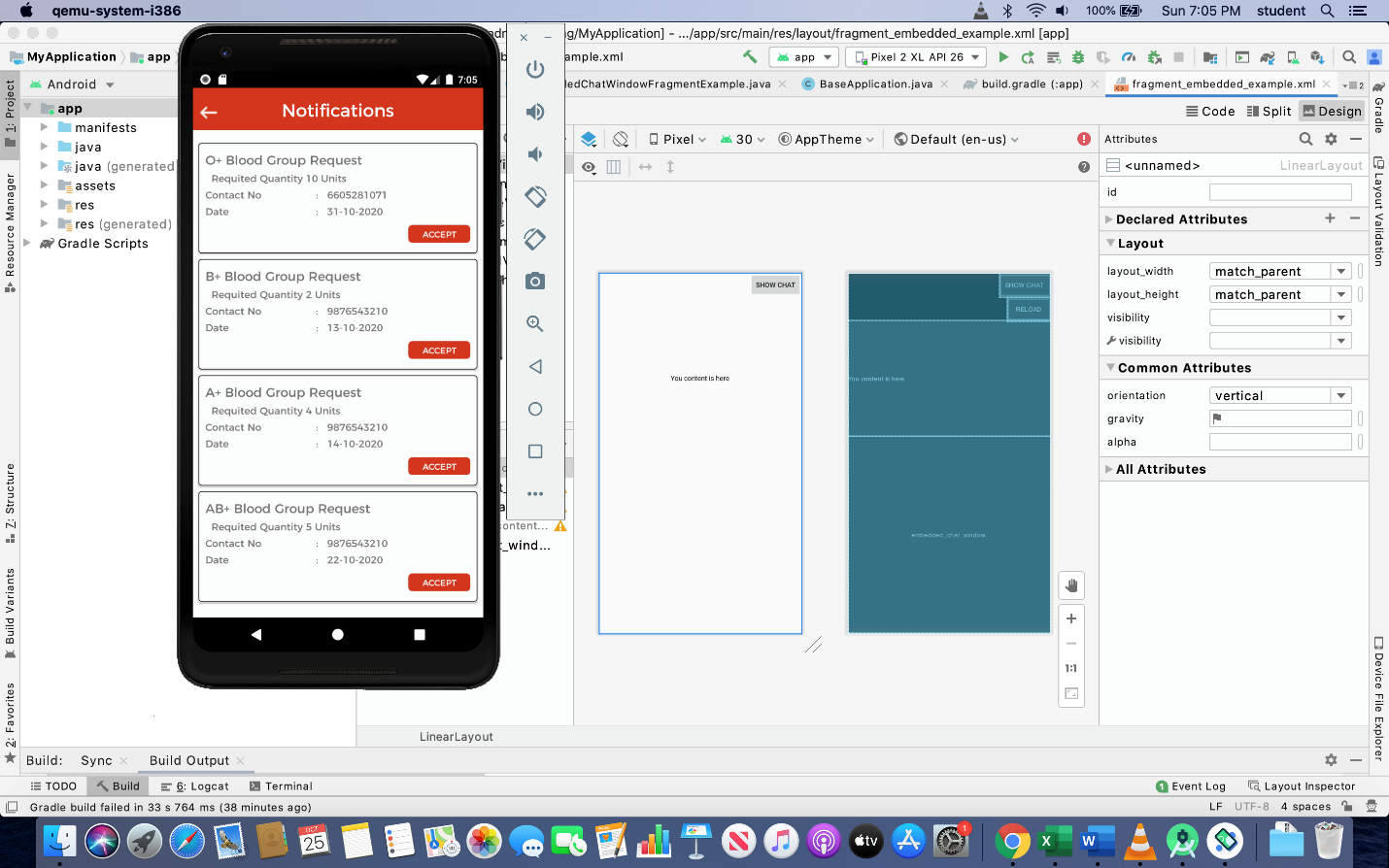
android:layout\_height="fill\_parent"/>

</LinearLayout>

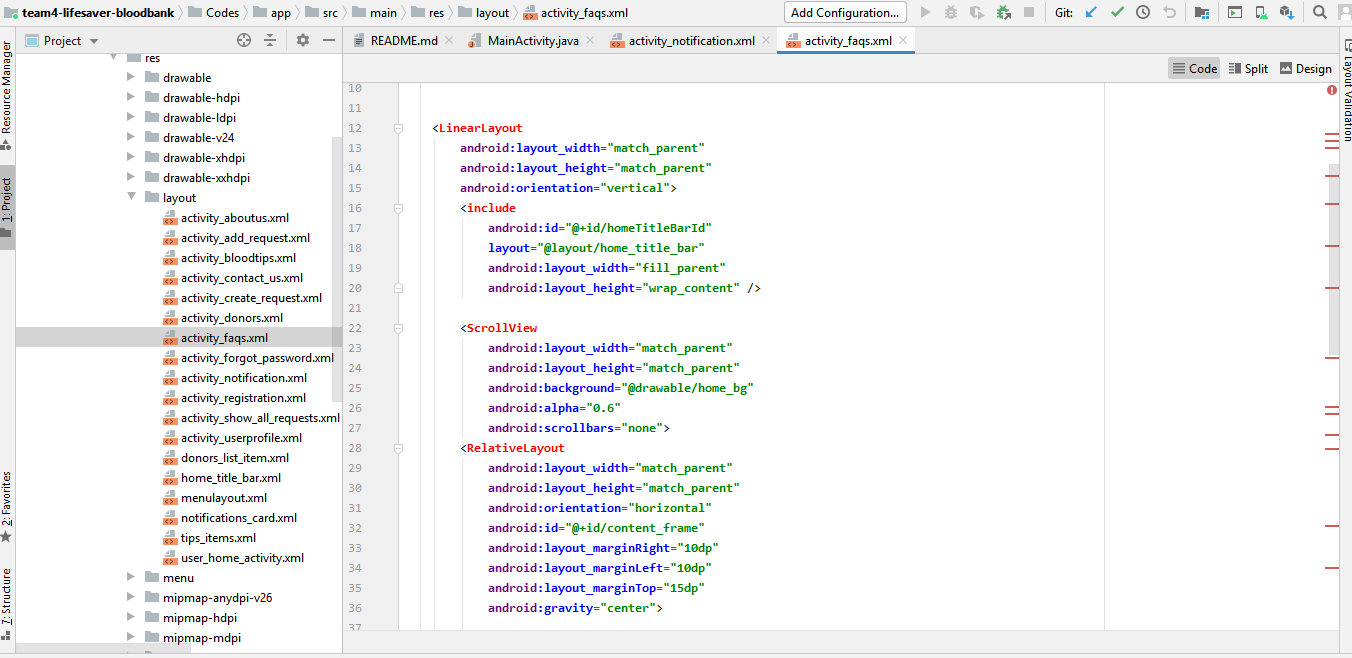
On the about us page, we have the description of our application.

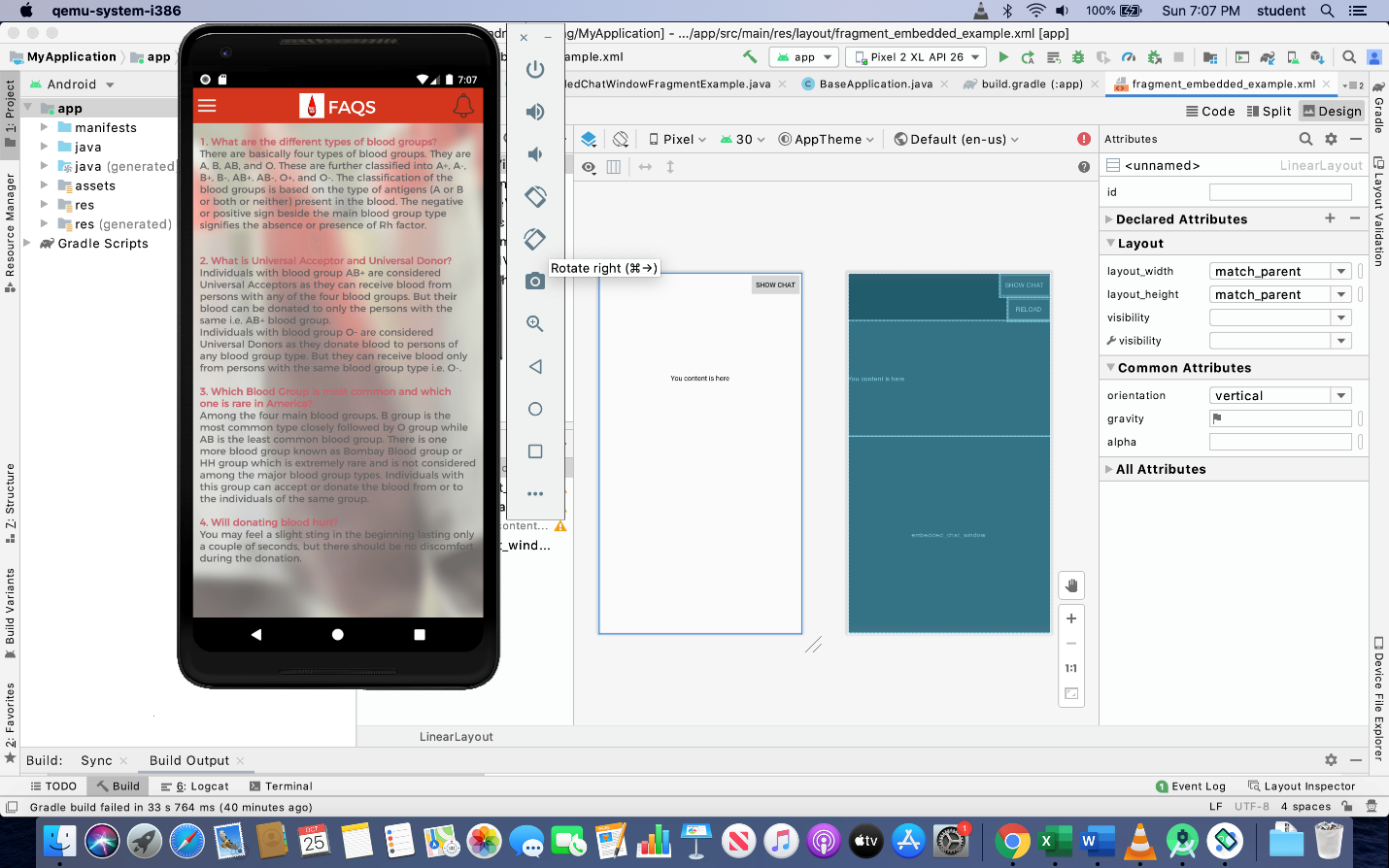
**Notification UI**: The following is the piece of code and screenshot for Notification UI. This UI includes all the details of the requests that are placed by the users who are in need of blood. The details include date, contact information and quantity needed.





**FAQ’S UI**: The following is the piece of code and screenshot for FAQ UI. This UI includes all the information regarding the frequently asked questions regarding the blood groups, donations and injuries.





1. **End-user Manual**

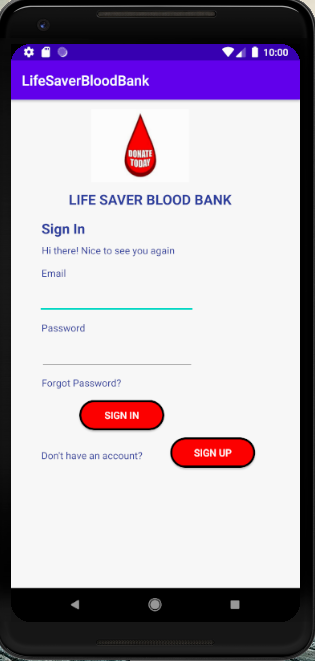
**Launch Page:**

Whenever the user opens the application the following image will be displayed first which is the launch screen.



**Login Page:**

If the user is an existing user, then he needs to enter his registered email address and respective password. Firebase will validate these credentials, if both are matched, then the home page will be shown having a pop-up message as "Logged in successfully!", else it will display "Entered Email or password does not match". This page will also have the option to change passwords and signup as well.

****

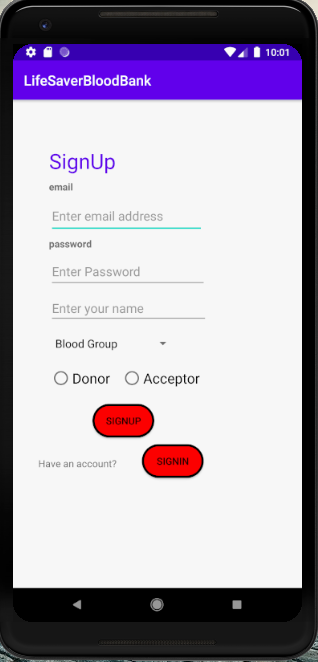
**Forgot Password Page:**

If a user forgets his password while logging in, then the application gives a chance to retrieve it. When the user press forgot password on the sign-in page, the forgot password page is shown. On this page, the user is supposed to enter his registered email-id. If the user presses the send button without entering the email address, then a toast message will appear which says 'please enter email address', else password reset link will be sent to the registered email address. Once the user presses on reset link it will redirect him to reset the password page where the user needs to enter a new password. Once the password reset, the firebase will update the password in the database, and the user will be able to sign in to the application.



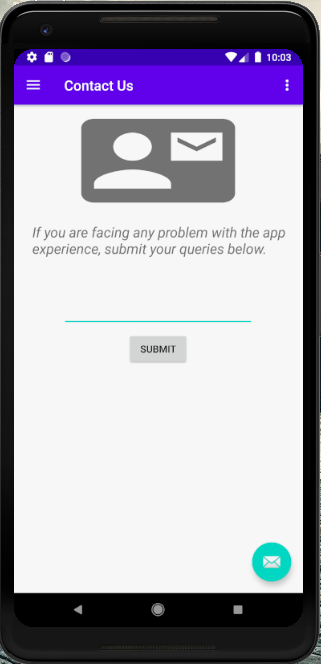
**Sign Up Page:**

If a user is new to the application he/she needs to sign up to access the application. On the signup page, certain details are needed to submit the form. This application requires an email address, password, name, blood group, and user type (Donor/Acceptor). Every field is mandatory on this page or else it will display a message called “All fields must be filled”. Email and password have a specific format i.e., email should end with “@gmial.com”, the password must contain a minimum of 8 characters with at least one alphabet, one number, and one special character. After entering all the details user has to click on the sign-up button. If all entered details are valid then a message called “Registered Successfully” and navigates to the login page or else it will display “Could not register”.



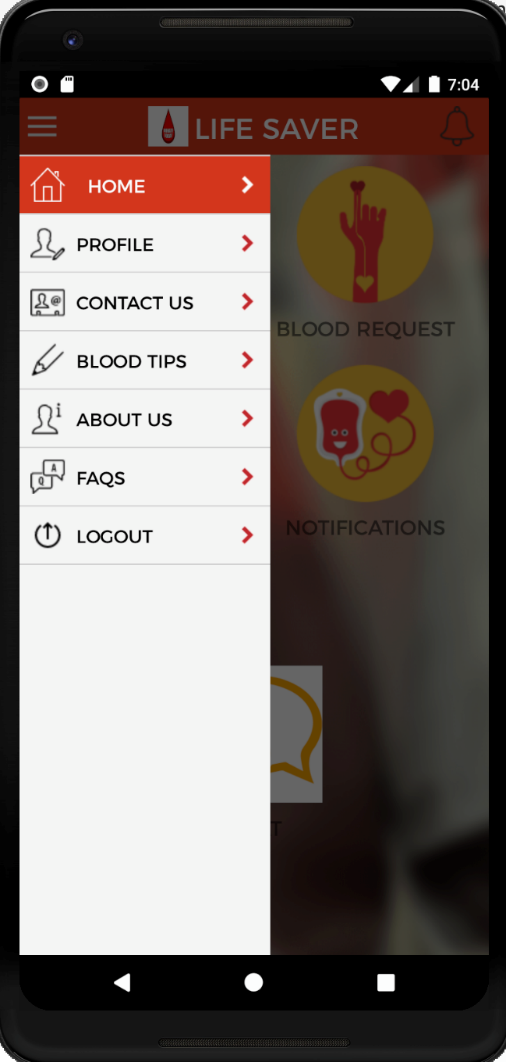
**Contact Us Page:**

This is the contact us page where user can submit their queries about the application. If the submit button is clicked without entering any text it will display a message called “Please enter the query which needs to be addressed”. If anything is entered and the submit button is clicked then those queries will be stored in the database with a pop-up message called “Your queries are submitted successfully”.



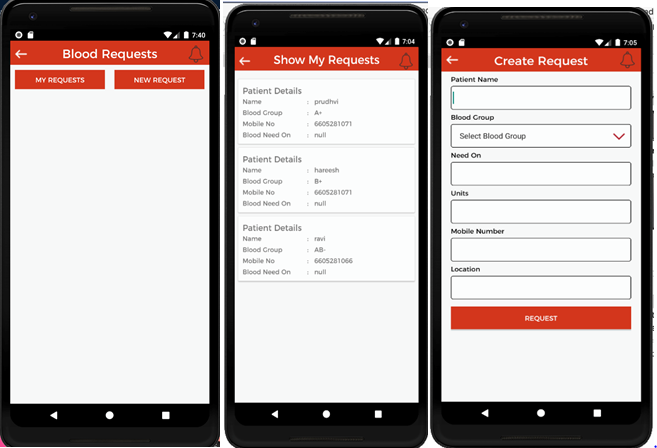
**Home Page:**

This the main activity after login which has a navigation bar with all the options which can be accessed by the user. It also has a home page UI where we have several icons that can be accessed to direct to respective pages. On the home page, we have blood donors, blood requests, responses, notifications, chat which can be used to access respective pages.

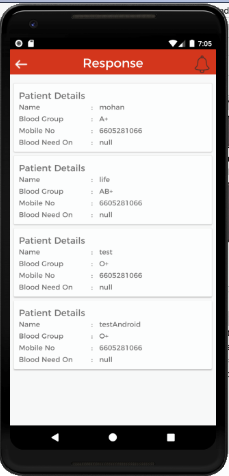
**Blood Request:**

On the blood request page, users will have options to create new blood requests and view existing requests. When the user presses on my request button, then from firebase database data will be fetched and will be shown respective requests. When the user presses on New Request, the user will have the option to create a new request where the user has to enter all the patient details.



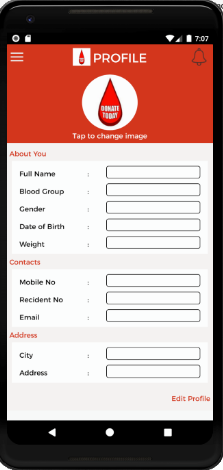
**Responses page:**

On this page, all the responses to the blood requests will be displayed. An “Accept” button will be available for the user in the notifications. Whenever the user clicks on it that response is stored in the database. This response page will have those details fetched from the database and displayed in a recycler view.



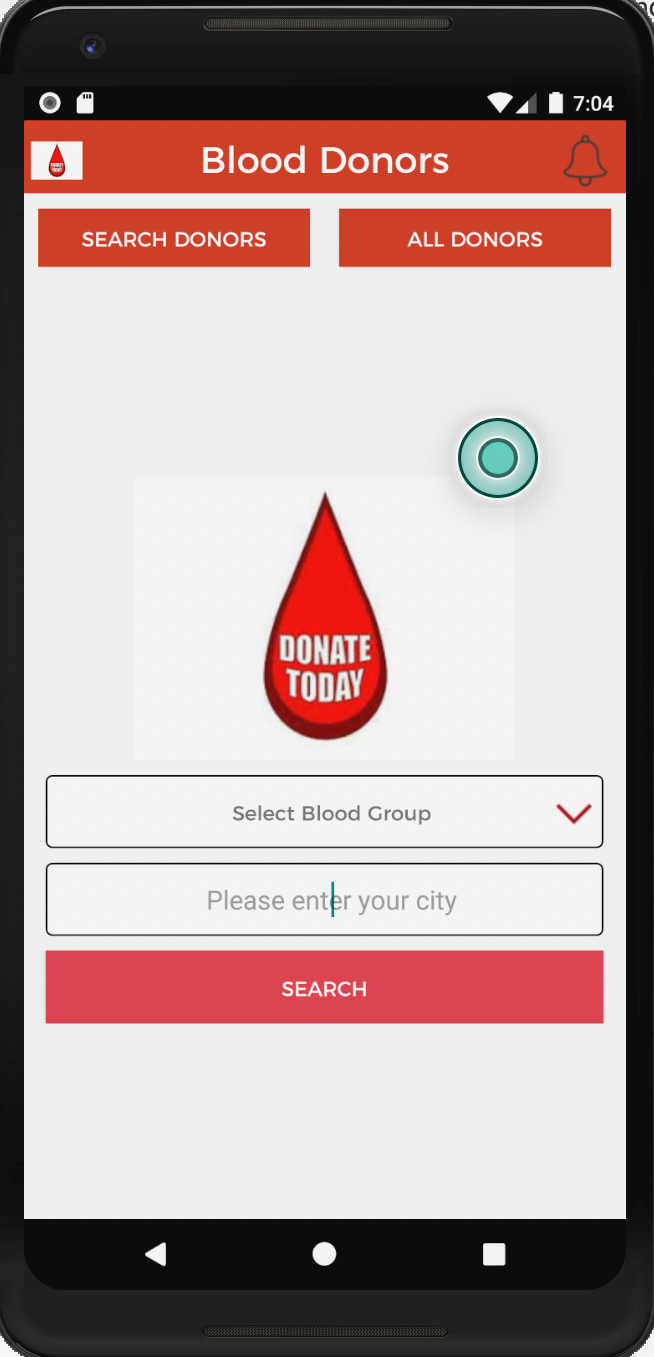
**Profile page:**

On this page, the user has an option to edit his/her profile details such as name, mobile number, address, email id, etc. Whenever the user clicks on the edit profile button which is on the bottom right side of the page he/she can edit the details. Once it is done they have to click on the update button so that their new details will be updated in the database.



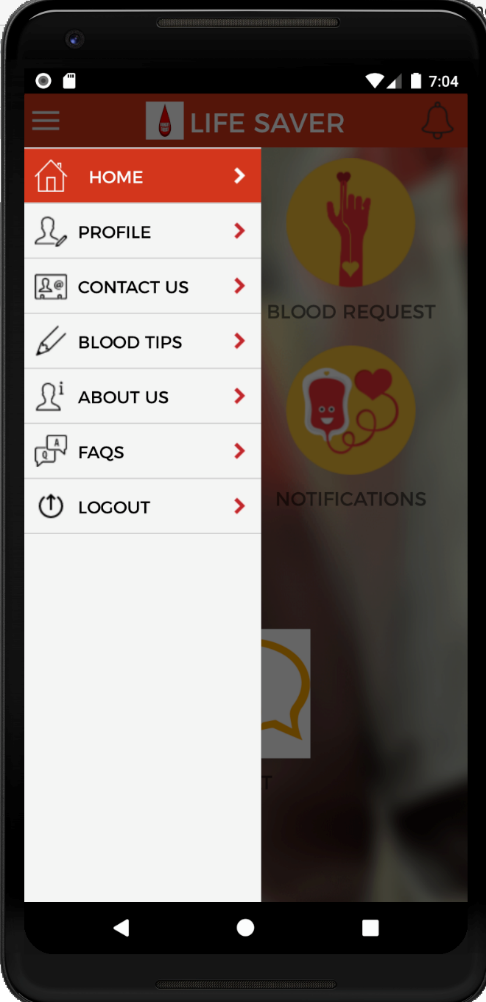
**Search Donor:**

In this section search, the donor activity function is implemented, which gives the users the ability to search for available blood. Anyone can search for the required group of blood by providing their blood group and the city. The resulted donors list will be displayed in the form of a recycler view.



**Logout:**

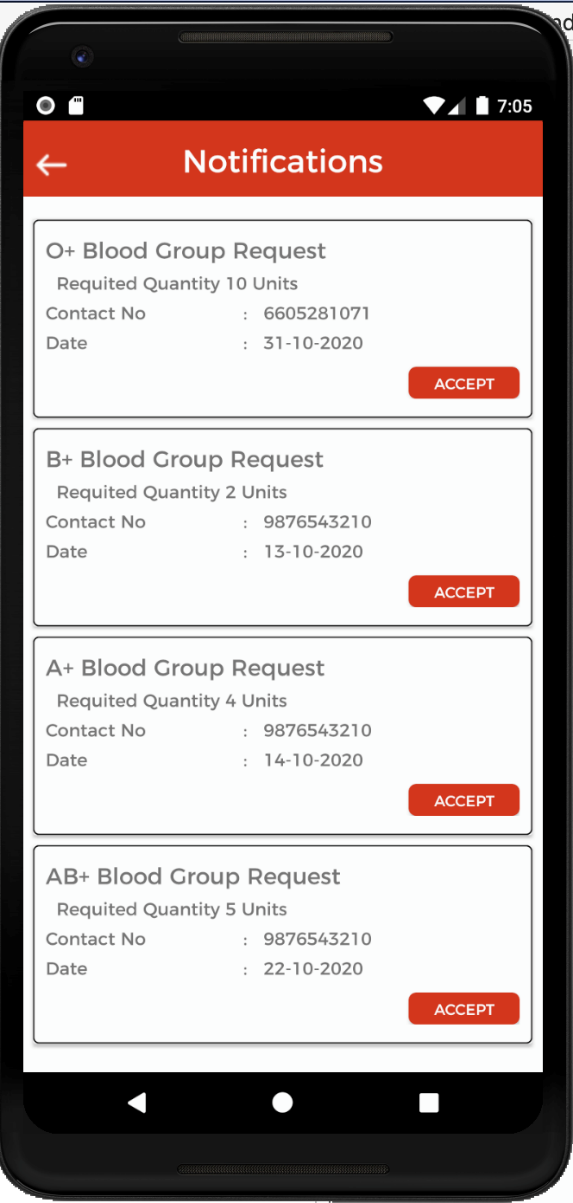
Depending on your device, you can sign out of the Android App by tapping the 'three vertical dots' icon at the top-left corner of the Home screens. When you do so, you should see a menu pop-up with a "LOGOUT" button.



**Notifications:**

After clicking on the notifications icon on the home page we will be redirected to the notifications page where the users request blood based on the blood group, the request contains blood group, the quantity of blood required, the Date on which the blood is required, and contact number.

These requests are stored in the recycler view. If the user accepts the request then the remaining users cannot see that notification on the notification page. After that, the response is stored in the database.

****

**Blood Donor functionality:**

After clicking on the All donors button on the donor page,we will be redirected to the donors list page where a particular user can find the information regarding the details of the donors that are available. The details of the donor here are saved in the recycler view which has name, blood group, phone number and the location.

Below is the screen shot of the blood donors functionality.

