

Blood Bank Management System

(BBMS)



Department of Computer Science and Engineering
Bangladesh University of Business and Technology

Dhaka-1216

July 2021

Blood Bank Management System (BBMS)

A report

*submitted to the department of Computer Science and Engineering
in partial fulfillment of the requirements
for the degree of*

Bachelor of Science in Computer Science and Engineering

By

Momin Ali (ID: 17182103011)

Arifur Rahman Kawser (ID: 17182103012)

Shakil Ahmed (ID: 17182103010)

Supervised by

Md. Anwar Hussen Wadud

Lecturer

Department of Computer Science and Engineering (CSE)
Bangladesh University of Business and Technology (BUBT)
Mirpur-2, Dhaka-1216, Bangladesh

ABSTRACT

Blood transfusion safety is a relevant and significant public health issue in the Sultanate of Oman. Since most blood banks are still in a paper-based system, various disadvantages are experienced by various stakeholders, which endanger the lives of patients and deter the healthcare system. As such, the researchers aimed to design, develop, and implement an online blood bank management system apps (OBMMS). This web-based application allows hospitals in Bangladesh to make inventories of their blood bags online, subsequently, allowing each hospital to check the availability of blood bags anytime. The researchers designed and administered questionnaires that assess the perceptions of various stakeholders in both manual-based and OBBMS. Based on the findings and results, it was found out that these stakeholders perceived the online blood bank management system as much better than the manual system. Therefore, with the use of online blood bank management system, blood transfusion process is safe and secured. Threats on improper blood donor documentation, or misplaced records will be totally eradicated. Also, processes involving recording about blood donors, blood bag collection, storage, and inventory will be systematized and organized, hence, improving the healthcare management for blood banks.

DECLARATION

We declare that this thesis and the work presented in it are our own and has been generated by us as the result of our own original research

We confirm that:

- This Work is done wholly or mainly while in candidature for a research degree at this University;
- This thesis work has not been previously submitted for any degree at this university or any other educational institutes;
- We have quoted from the work of others; the source is always given. With the exception of such quotations, this thesis is entire our own work;

Shakil Ahmed
ID: 17182103010

Momin Ali
ID: 17182103011

Arifur Rahman
Kawser
ID: 17182103012

CERTIFICATE

This is to certify that Momin Ali, Arifur Rahman Kawser and Shakil Ahmed students of B.Sc. in CSE have completed their thesis work titled “Blood Bank Management System” satisfactorily in partial fulfillment for the requirement of B.Sc.in CSE. Bangladesh University of Business and Technology in the year 2021.

Shakil Ahmed
ID: 17182103010

Momin Ali
ID: 17182103011

Arifur Rahman
Kawser
ID: 17182103012

Thesis Supervisor
(Md. Anwar Hussen Wadud)

Lecturer

Department of Computer Science and Engineering (CSE)

Bangladesh University of Business and Technology (BUBT)

DEDICATION

Dedicated to our parents for all their love and inspiration.

ACKNOWLEDGEMENTS

First of all, we are thankful and expressing our gratefulness to Almighty Allah who offers us His divine blessing, patience, mental and physical strength to complete this project work.

We are deeply indebted to our project supervisor Md. Anwar Hussen Wadud, Lecturer, Department of Computer Science and Engineering (CSE), Bangladesh University of Business and Technology (BUBT). His scholarly guidance, important suggestions, work for going through our drafts and correcting them, and generating courage from the beginning to the end of the research work has made the completion of this thesis possible.

We would like to express our deep gratitude to our Teacher Md. Anwar Hussen Wadud, Lecturer, Department of Computer Science and Engineering (CSE), Bangladesh University of Business and Technology (BUBT). It was fantastic to get help from him and without his support it will be tough for us to reach the accurate goal.

A very special gratitude goes out to all our friends for their support and help to implement our works. The discussions with them on various topics of our works have been very helpful for us to enrich our knowledge and conception regarding the work.

Last but not the least; we are highly grateful to our parents and family members for supporting us spiritually throughout writing this thesis and our life in general.

APPROVAL

This Thesis “**Blood Bank Management System**” Submitted by **Momin Ali, Arifur Rahman Kawser** and **Shakil Ahmed** ID NO: **17182103011, 17182103012** and **17182103010** Department of Computer Science and Engineering (CSE), Bangladesh University of Business and Technology (BUBT) under the supervision of Md. Anwar Hussen Wadud, Lecturer, Department of Computer Science and Engineering has been accepted as satisfactory for the partial fulfillment of the requirement for the degree of Bachelor of Science (B.Sc. Eng.) in Computer Science and Engineering and approved as to its style and contents.

Supervisor:

Md. Anwar Hussen Wadud

Lecturer
Department of Computer Science and Engineering (CSE)
Bangladesh University of Business and Technology (BUBT)
Mirpur-2, Dhaka-1216, Bangladesh

Chairman:

Prof. M. Ameer Ali

Professor and Chairman
Department of Computer Science and Engineering (CSE)
Bangladesh University of Business and Technology (BUBT)
Mirpur-2, Dhaka-1216, Bangladesh

© Copyright by Momin Ali (17182103011), Arifur Rahman Kawser (17182103012), and Shakil Ahmed (17182103010)

All Right Reserved

Abbreviations

Synonyms and Acronyms

Descriptions

RAM

Random Access Memory

Table of Content

	Page no
Abstract	ii
Declaration	iii
Certification	iv
Acknowledgement	v
Dedication	vi
Approval	vii
Copyright	
Chapter 01: Introduction	15-16
1.1 Introduction	15
1.2 Aims and Objectives	16
1.3 Conclusion	16
Chapter 02 : Developing Environment of Android	17-21
2.1 Introduction	17
2.2 Build Developing Environment of Android	18
2.3 The Design Principle of Android Application	19
2.4 Function and Structure Design of Android System	20
2.5 Conclusion	21
Chapter 3 : Analysis	22-25
3.1 Introduction	22
3.2 Economic Feasibility	22
3.3 Technical Feasibility	23
3.4 Social Feasibility	23
3.5 Software Specification	24
3.6 Conclusion	25

Chapter 4 : System Design	26-33
4.1 Introduction	26
4.2 Introduction of AppStarting module in the project	26
4.3 Introduction of engineering program structure	26
4.4 Design of our apps features	27
4.5 Profile Creating	27
4.6 Donor registration	28
4.7 List of blood group	29
4.8 Find Donor	30
4.9 Achievement	31
4.10 Finding nearest Hospital	32
4.11 Navigation menu	33
 Chapter 5 : Code Analysis and Conclusions	 34-83
5.1 Introduction	34
5.2 Code	34-82
5.3 Conclusion	83
5.4 Future Works	83
List of Tables	ix
List of Figures	x

List of Figures

	Page no
Figure-1.1: Profile Creating	27
Figure-1.2: Donor Registration	28
Figure-1.3: List of Blood Group	29
Figure-1.4: Find Donor	30
Figure-1.5: Achievement	31
Figure-1.6: Finding nearest Hospital	32
Figure-1.7: Navigation menu	33

List of Tables

N/A

Chapter 01

Introduction

1.1 Introduction

Blood Bank Management System (BBMS) is an Application system that can assist the information of blood bags during its handling in the blood bank. With this system, the user of this system can key in the result of a blood test that has been conducted to each of the blood bags received by the blood bank.

A blood donation is a process whereby a person voluntarily has blood drawn to be used for future transfusions when in need at hospitals for treatment procedures that require them.

Donation may be of whole blood (blood drawn directly from the body) or of specific components of the blood; such as red blood cells, white blood cells, plasma, and platelets. Blood banks often participate in the process of collecting blood and other procedures such as managing stocks, approving blood requests and updating donation information. The inspiration of this project is to improve blood banks in Bangladesh and to develop a blood bank information system which focuses on making an online system that is accessible for both donors and administrators. Donors can directly receive information regarding their previous blood donations, including their blood results and donation history, in order to easily schedule their next donations. They can also update the personal information through the system, without having to contact the blood bank registry information if necessary. The administrator is also responsible for responding to the hospital's blood requests and checking the stocks in the blood bank's inventory

1.2 Aim and Objectives

The main objective of the Blood Bank Management System is to manage the details of Blood ,Donor, Blood Group, Blood Bank, Stock and Location of Hospitals. It manages all the information about Blood , Blood Cell, Stock , Donor, Looking-for blood . The project is totally built at the administrative end and thus only the administrator & users are guaranteed the access. The aim and objectives of the Blood Bank Management System are as follows:

1. To develop a system that provides functions to support donors to view and manage their information conveniently.
2. To maintain records of blood donors, blood donation information and bloodstocks in a centralized database system.
3. To inform donors of their blood result after their donation.
4. To support searching, matching and requesting for blood convenient for administrators.

1.3 Conclusion

In this project We tried to implement the Centralized Blood Bank Management System. This project is built on salesforce and can serve many advantages to the Users. This software is efficient in maintaining the donor's details and can easily perform operations on blood donation's records. This software also reduces the workload of the blood bank management to know how much blood is available and also keep the records of how many times a donor donates blood by this application.

Chapter 02

Developing Environment of Android

2.1 Introduction

Android is an open source code mobile phone operating system that was released by Google in November 2007. Its appearance has broken the traditional closed mobile phone operating system. Anyone can modify the mobile phone operating system as well as function according to personal preference, which is also the most attractive merit of Android. Blood Bank Management System in this article is an application software based on Google Android.

Android's application on mobile terminals also completely broke the traditional understanding of the mobile terminals. And appreciating music is one of the best ways to relieve pressure in stressful modern society life. Therefore, many kinds of Blood Bank Management Systems are also developed. However, a lot of Blood Bank Management Systems devote themselves to fancy appearance and function, while causing resources wasting to the user's mobile phone, such as large required memory and CPU, which brings a lot of inconvenience as multiple programs running at the same time. For the most ordinary users, many functions are useless.

Blood Bank Management System based on Android applications are popular in the market at the present. The complete development of the Android operating system gives developers a nice platform, which can learn the popular computer technology combining with learned knowledge, and master the latest knowledge, enrich oneself, and enjoy entertainment.

2.2 Build Developing Environment of Android

The applications of Android need not be based on android environment. The following is the configuration requirement and installation steps of Android Development Environment.

The Required Software Of The Developing Environment

Operating System: Windows 10

Software: Android SDK (Software Development Kit), ADT(Android Development Tool)

IDE environment: Android Studio

JDK : Java Runtime Environment virtual machine, Java Development Kit (JDK)

Installation Steps Of The Developing Environment

Step1: install the Java virtual machine JDK version-6

Step2: install Android Studio;

download address: <https://developer.android.com/studio?hl=id>

Step3: install the Android SDK: first download the Android SDK

Download Address: <http://developer-android-com/sdk/index-html>

Step 4: Install Android Git plug-in, run Android Studio and select help - > install new software selected.

Input SDK tools path in the SDK location: D:\android\software\androidSDK-Windows and click OK.

The Android environment is set up successfully.

2.3 The Design Principle of Android Application

Twice the result with half the effort will get if an overall study of the principle is done before the design and follow them in the operation. The principle of software design mainly includes the following points:

Reliability

The reliability of the software design must be determined. The reliability of the software system refers to the ability to avoid fault occurring in the process of system running, as well as the ability to remedy troubles once the fault occurs.

Reusability

Look for commonness of similar codes, and come up with new methods abstractly and reasonably. Pay Attention To The Generic Design.

Understandability

The understandability of software not only requires a clear and readable document, but the simplified structure of software itself, which requires the designer to possess keen insight and creativity, and know well about the design objects.

Simple Program

To keep the program simple and clear, good programmers can use simple programs to solve complex problems.

Testability

Testability means that the created system has a proper data collection to conduct a comprehensive test of the entire system.

The Open-Closed Principle

Module Is Extensible But Cannot Be Modified. Thatistosay,extensions open the existing code in order to adapt to the new requirements. While modify is closed to the categories. Once The Design Is Completed,the categories cannot be modified.

2.4 Function and Structure Design of Android System

This system adopts the modularized program design,and system function is correspondingly divided into function modules,the main modules include:

- User Databases
- User Login & Sign up
- Donor Details
- Finding blood group easily
- Finding Nearby Hospitals
- Achievements & Rewards

2.5 Conclusion

In this project We tried to implement the Centralized Blood Bank Management System. This project is built on salesforce and can serve many advantages to the Users. This software is efficient in maintaining the donor's details and can easily perform operations on blood donation's records. This software also reduces the workload of the blood bank management to know how much blood is available and also keep the records of how many times a donor donates blood by this application.

Chapter 03

Analysis

3.1 Introduction

A blood bank is a center where blood gathered as a result of blood donation is stored and preserved for later use in blood transfusion. The term "blood bank" typically refers to a division of a hospital where the storage of blood product occurs and where proper testing is performed (to reduce the risk of transfusion related adverse events). However, it sometimes refers to a collection center, and some hospitals also perform collection. Blood banking includes tasks related to blood collection, processing, testing, separation, and storage.

3.2 Economic Feasibility

The feasibility study is performed to determine whether the proposed system is viable considering the Technical, Operational and Economical factors. After going through a feasibility study we can have a clear-cut view of the system's benefits and drawbacks. Here we discuss Economic Feasibility study. To design Blood Bank Management System as long as a computer has the Android development and the application development of Android is free. In addition, the Blood Bank Management System is a basic need for the public. The information that functions are necessary form all the consumers , which functions are needed for some people, and which features are seldom easy to understand. And a lot of research is eliminated, thus saving the spending. Therefore, the whole process of development doesn't need to spend money who is economic feasibility.

3.3 Technical Feasibility

To develop a blood bank management system app which meets the basic requirements, a deep understanding of JAVA language, the Android system architecture, application of framework and other technical knowledge are needed. (framework is the core of the application, and rules that all the programmers participating in the development must abide by). Based on 10 related technology information and resources for Android on the market, and equipped with technical personnel of technology and the spirit of willing to learn, the technology is feasible.

3.4 Social Feasibility

With the rapid development of the mobile phone market, all kinds of apps are widely circulated on the Internet. These resources seem ordinary, but have gradually become an indispensable part of people's lives, which resulted in the development of blood bank management system. But a lot of apps are devoted to fancy appearance, strong function causing a lot of wasted resources to the user's mobile phone and bringing a lot of inconvenience to the user as multitasking operation is needed. Some functions are useless to ordinary people. Powerful apps are a good thing, but a lot of functions are actually useless for most users. Aimed at these problems, developing a multiplied blood bank management system which owns the features of User Databases, User Login & Sign up, Donor Details, Finding blood group easily, Finding NearBy Hospitals, Achievements & Rewards.

3.5 Software Specification

There were few software and tools used to develop this project.

Firestore Database

Firestore is a platform developed by Google for creating mobile and web applications. It was originally an independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development. The Firestore Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in realtime. Cloud Firestore enables you to store, sync and query app data at global scale.

Firestore Authentication

Firestore Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.

Google Maps API

Google Maps APIs are prepackaged pieces of code that let you quickly and easily include maps on your websites or in your mobile apps – and then add extra functions to your applications. They're available for Android, iOS and web browsers, and as HTTP web services that let you pass information between systems. The Google Maps API is one of those clever bits of Google technology that helps you take the power of Google Maps and put it directly on your own site. It lets you add relevant content that is useful to your visitors and customise the look and feel of the map to fit with the style of your site.

Android Studio IDE

Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools. A unified environment where you can develop for all Android devices. Apply Changes to push code and resource changes to your running app without restarting your app.

Android Operating System version 4.0 or higher

The Android operating system is a mobile operating system that was developed by Google (GOOGLE) to be primarily used for touchscreen devices, cell phones, and tablets. Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen.

3.6 Conclusion

In this project We tried to implement the Centralized Blood Bank Management System. This project is built on salesforce and can serve many advantages to the Users. This software is efficient in maintaining the donor's details and can easily perform operations on blood donation's records. This software also reduces the workload of the blood bank management to know how much blood is available and also keep the records of how many times a donor donates blood by this application.

Chapter 04

System Design

4.1 Introduction

In this section, the App Starting module of the medicine remainder in the project is introduced, as well as the Android engineering program structure, etc.

4.2 Introduction of AppStarting module in the project

Any AppStarting needs AndroidManifest. XML file to start. And any new project content will automatically generate an Android Manifest. XML file. Configuration files are the core of the whole program, which contains the Android SDK version, and the default Activity in program running. The system will automatically be looking for a logo in Android Manifest to react to the corresponding operation when any component of the program triggers events. To define the system, the first thing is launching the Activity: Android Activity. There are properties such as action and category in < intent - filter >. Most of these are the default values of the system. Setting the action and category realizes the switch between different Activities. When any components of the program are about to be used, the declaration must be in the Android Manifest.Xml files. To be clear that authorities must be illustrated as the statement of the provider. Each component has a lot of attributes; the program will define different attributes according to different needs.

4.3 Introduction of engineering program structure

The basic structure content of Android project includes: the SRC (source code), gen (constant that Android system automatically generates), res (resource file), and the layout of file and pictures in the main storage program interface.

4.4 Design of our apps features

4.5 Profile Creating

Creating profile for users: Registration with name, blood group, contact number and other information.

AirtelJairte... grameenphone 74% 1:05PM

← Profile

Name: Abdul Khalek

Sex: Male ▼

Blood Group: O+ ▼

Contact No: 01234567890

Address: Tangail

Division: Dhaka ▼

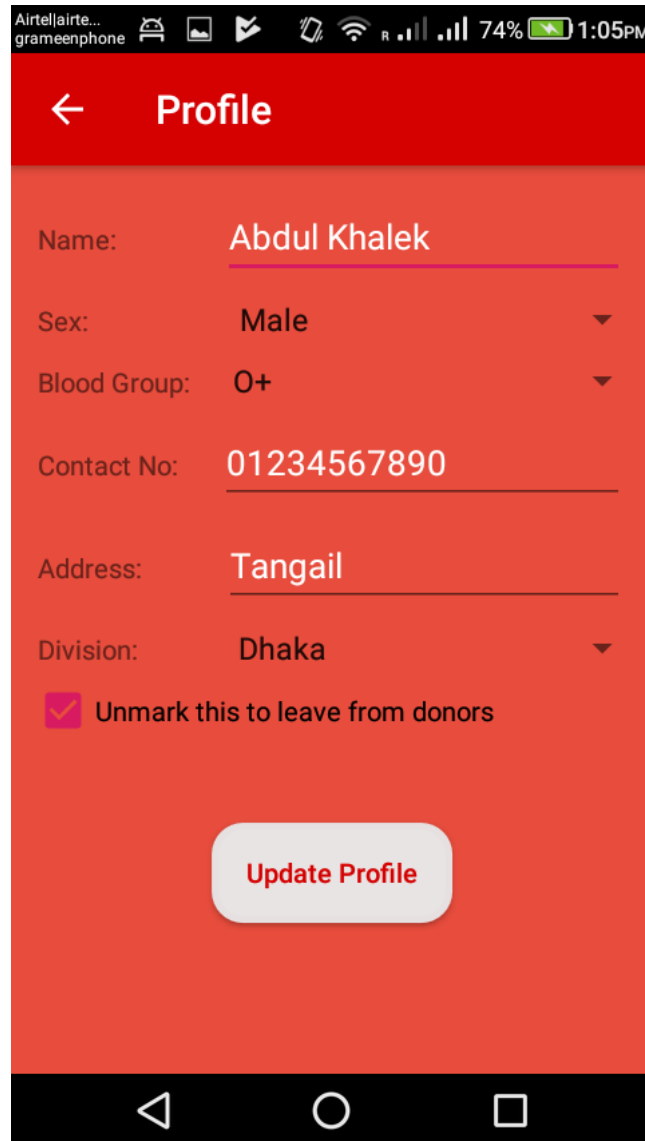
☒ Unmark this to leave from donors

Update Profile

Figure 1.1: Profile Creating

4.6 Donor registration

Donor management-donor registration, managing donor databases, recording their physical and medical statistics.



The screenshot shows a mobile application interface for donor registration. The status bar at the top indicates the carrier is 'Airtel', the phone is 'grameenphone', and the battery is at 74%. The app's header is red with a back arrow and the title 'Profile'. The form fields are as follows:

Field	Value
Name:	Abdul Khalek
Sex:	Male
Blood Group:	O+
Contact No:	01234567890
Address:	Tangail
Division:	Dhaka

Below the form fields is a checkbox labeled 'Unmark this to leave from donors' which is currently checked. At the bottom of the form is a button labeled 'Update Profile'.

Figure 1.2 Donor Registration

4.7 List of blood group

List of requested blood groups for patients. Every donor is added to this list.

← Profile

Name: Abdul Khalek

Sex: Male

Blood Group: O+

Contact No: 01234567890

Address: Tangail

Division: Dhaka

☒ Unmark this to leave from donors

Update Profile

Figure 1.3 List of Blood Group

4.8 Find Donor

Query donor by blood group & division.

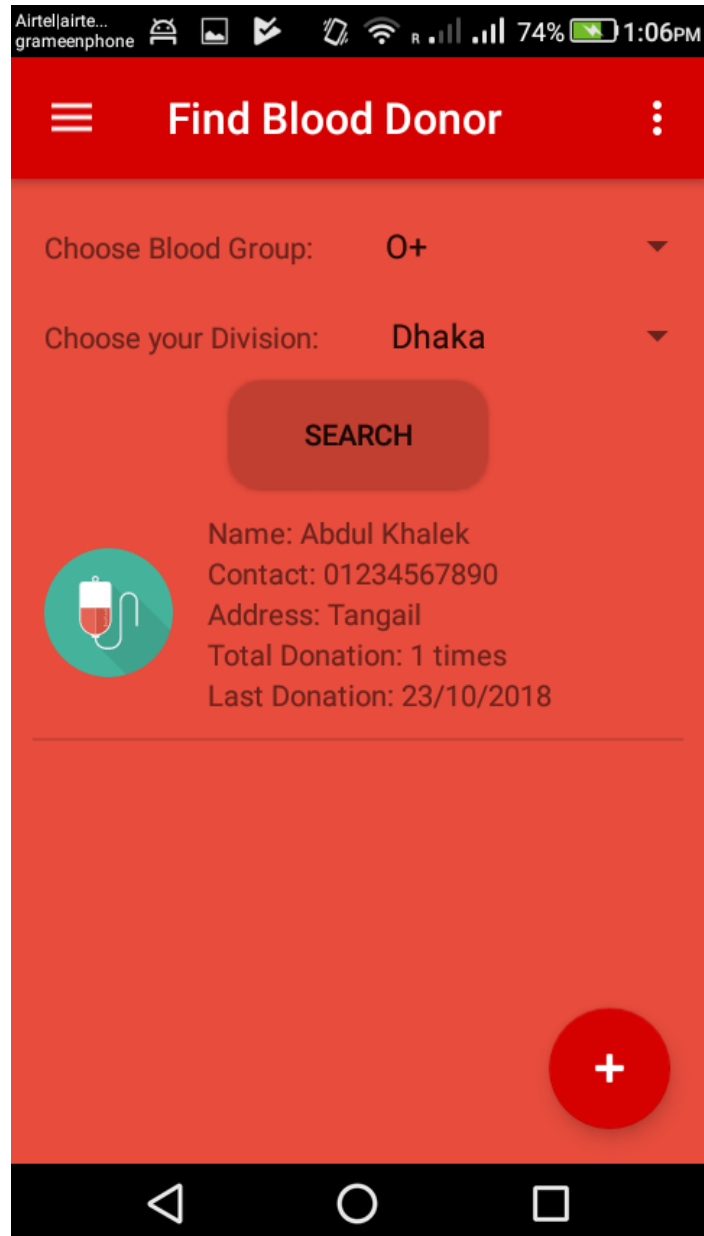


Figure 1.4 Find Donor

4.9 Achievement

There is a database table named “Achievement” : where a donor can count the next donation date, Total Donation also the last donation date.

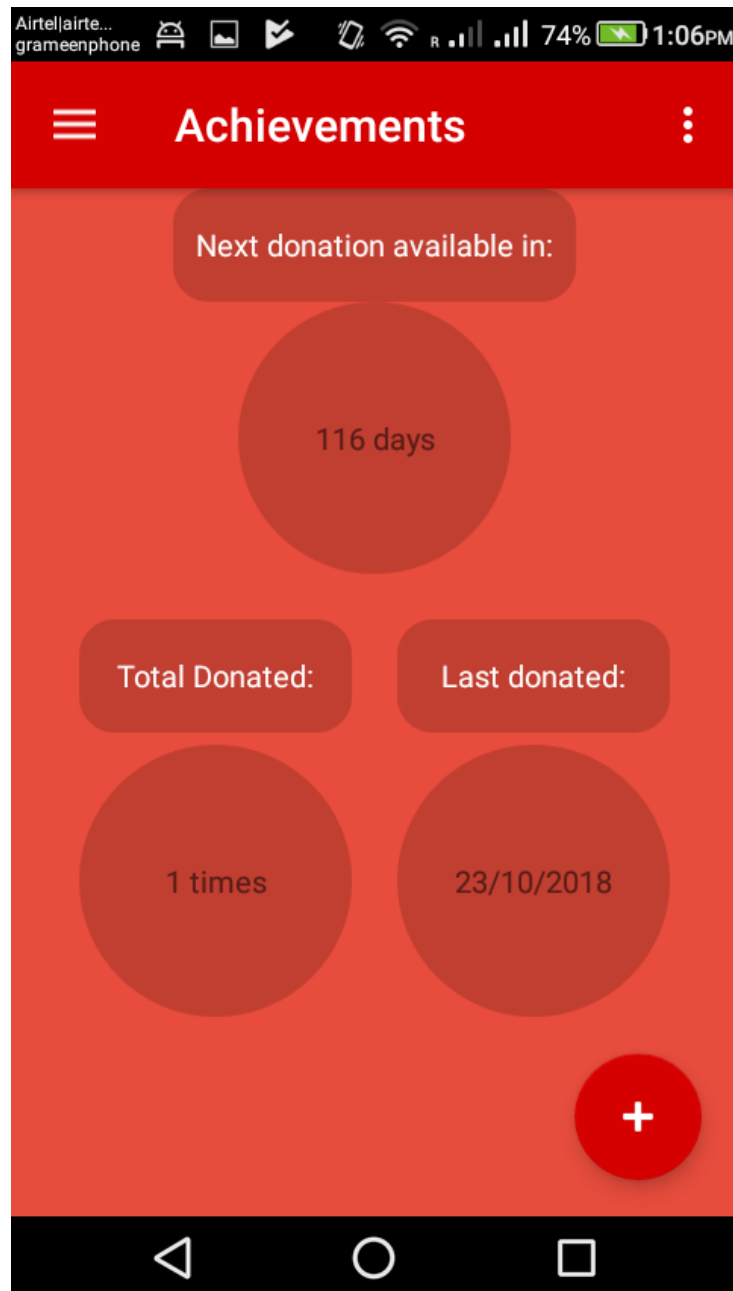


Figure 1.5 Achievement

4.10 Finding nearest Hospital

An interested Donor can donate blood without blood request. For this the donor has to use “The Nearest Hospital” Module. In this module Donor can find the nearest hospitals from his/her current location.

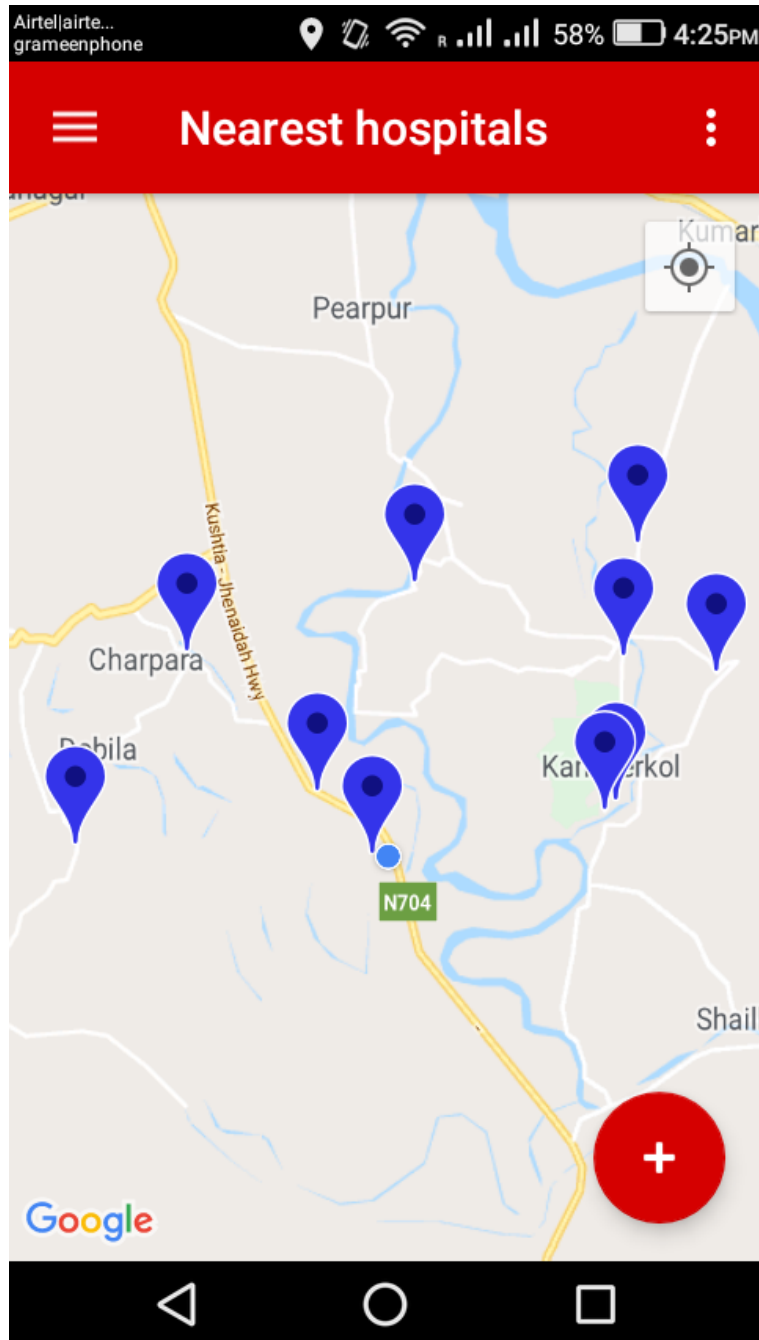


Figure 1.6 Finding nearest hospital

4.11 Navigation menu

In its App, There is a navigation button where users can easily shift one module to another module and also can logout for closing the app.

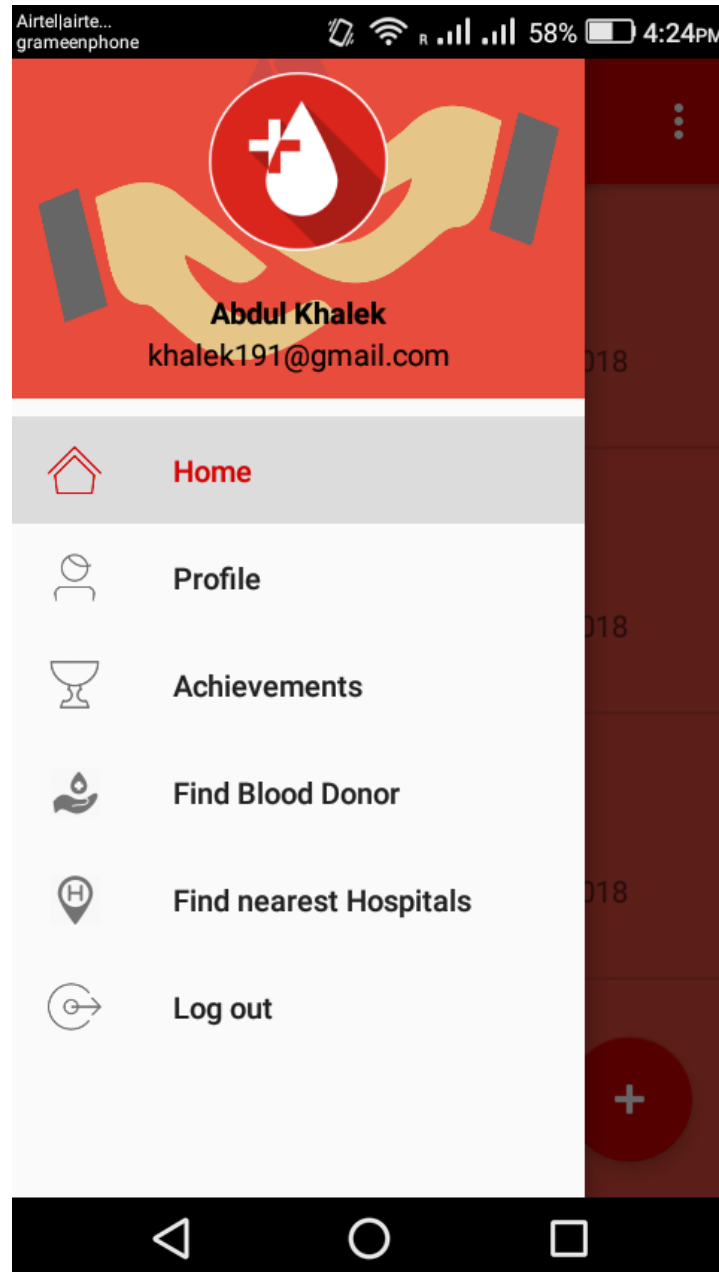


Figure 1.7 Navigation Menu

Chapter 05

Code Analysis and Conclusions

5.1 Introduction

In this chapter we discuss about code of our project

5.2 Code

Dashboard

```
package com.android.iunoob.bloodbank.activities;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.FloatingActionButton;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.View;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;

import com.android.iunoob.bloodbank.R;
import com.android.iunoob.bloodbank.fragments.AboutUs;
import com.android.iunoob.bloodbank.fragments.AchievementsView;
import com.android.iunoob.bloodbank.fragments.BloodInfo;
import com.android.iunoob.bloodbank.fragments.HomeView;
import com.android.iunoob.bloodbank.fragments.NearByHospitalActivity;
import com.android.iunoob.bloodbank.fragments.SearchDonorFragment;
import com.android.iunoob.bloodbank.viewmodels.UserData;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
```

```

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;

import static com.android.iunoob.bloodbank.R.id.home;

public class Dashboard extends AppCompatActivity
    implements NavigationView.OnNavigationItemSelectedListener {

    private FirebaseAuth mAuth;
    private TextView getUsername;
    private TextView getEmail;
    private FirebaseDatabase user_db;
    private FirebaseUser cur_user;
    private DatabaseReference userdb_ref;

    private ProgressDialog pd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dashboard);

        pd = new ProgressDialog(this);
        pd.setMessage("Loading...");
        pd.setCancelable(true);
        pd.setCanceledOnTouchOutside(false);

        mAuth = FirebaseAuth.getInstance();
        user_db = FirebaseDatabase.getInstance();
        cur_user = mAuth.getCurrentUser();
        userdb_ref = user_db.getReference("users");

        getEmail = findViewById(R.id.UserEmailView);
        getUsername = findViewById(R.id.UserNameView);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton)
findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(Dashboard.this, PostActivity.class));
            }
        });

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);

```

```

navigationView.setNavigationItemSelectedListener(this);
View header = navigationView.getHeaderView(0);

getEmail = (TextView) header.findViewById(R.id.UserEmailView);
getName = (TextView) header.findViewById(R.id.UserNameView);

Query singleuser = userdb_ref.child(cur_user.getUid());
pd.show();
singleuser.addListenerForSingleValueEvent(new ValueEventListener() {

    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        //pd.show();
        String name = dataSnapshot.getValue(UserData.class).getName();

        getName.setText(name);
        getEmail.setText(cur_user.getEmail());

        pd.dismiss();
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
        Log.d("User", databaseError.getMessage());
    }
});

if(savedInstanceState == null)
{

getSupportFragmentManager().beginTransaction().replace(R.id.fragmentcontainer,
new HomeView()).commit();
        navigationView.getMenu().getItem(0).setChecked(true);

    }

}

@Override
public void onBackPressed() {
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is
    present.
    getMenuInflater().inflate(R.menu.dashboard, menu);
    return true;
}

```

```

    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        if (id == R.id.donateinfo) {

            getSupportFragmentManager().beginTransaction().replace(R.id.fragmentcontainer,
            new BloodInfo()).commit();
        }
        if (id == R.id.devinfo) {

            getSupportFragmentManager().beginTransaction().replace(R.id.fragmentcontainer,
            new AboutUs()).commit();
        }

        return super.onOptionsItemSelected(item);
    }

    @SuppressWarnings("StatementWithEmptyBody")
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();

        if (id == home) {

            getSupportFragmentManager().beginTransaction().replace(R.id.fragmentcontainer,
            new HomeView()).commit();

        } else if (id == R.id.userprofile) {
            startActivity(new Intent(getApplicationContext(),
            ProfileActivity.class));
        }
        else if (id == R.id.user_achiev) {

            getSupportFragmentManager().beginTransaction().replace(R.id.fragmentcontainer,
            new AchievementsView()).commit();

        }
        else if (id == R.id.logout) {
            mAuth.signOut();
            Intent intent = new Intent(getApplicationContext(),
            LoginActivity.class);
            startActivity(intent);
        }
        else if (id == R.id.blood_storage){

            getSupportFragmentManager().beginTransaction().replace(R.id.fragmentcontainer,
            new SearchDonorFragment()).commit();
        }
    }

```

```

        } else if (id == R.id.nearby_hospital) {

getSupportFragmentManager().beginTransaction().replace(R.id.fragmentcontainer,
new NearbyHospitalActivity()).commit();

        }

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }

    @Override
    protected void onStart() {
        super.onStart();
        FirebaseUser currentUser = mAuth.getCurrentUser();
        if(currentUser == null)
        {
            Intent intent = new Intent(getApplicationContext(),
LoginActivity.class);
            startActivity(intent);
            finish();
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        FirebaseUser currentUser = mAuth.getCurrentUser();
        if(currentUser == null)
        {
            Intent intent = new Intent(getApplicationContext(),
LoginActivity.class);
            startActivity(intent);
            finish();
        }
    }
}

```

Logein Activities

```

package com.android.iunoob.bloodbank.activities;

import android.app.ProgressDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.android.iunoob.bloodbank.R;

```



```

Toast.makeText(getApplicationContext(),
                                "Authentication Failed",
                                Toast.LENGTH_LONG).show();
                                Log.v("error",
task.getException().getMessage());
                                } else {
                                    Intent intent = new
Intent(getApplicationContext(), Dashboard.class);
                                    startActivity(intent);
                                    finish();
                                }
                                pd.dismiss();
                            }
                        });
                    }
                    else
                    {
                        Toast.makeText(getApplicationContext(), "Please fill
all the field.", Toast.LENGTH_LONG).show();
                    }

                } catch (Exception e)
                {
                    e.printStackTrace();
                }
            }
        });

        signup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(),
ProfileActivity.class);
                startActivity(intent);
            }
        });

        resetpass.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(),
RestorePassword.class);
                startActivity(intent);
            }
        });
    }
}

```


Post Activities

```
package com.android.iunoob.bloodbank.activities;

import android.app.ProgressDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import com.android.iunoob.bloodbank.R;
import com.android.iunoob.bloodbank.viewmodels.UserData;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;

import java.util.Calendar;

public class PostActivity extends AppCompatActivity {

    ProgressDialog pd;

    EditText text1, text2;
    Spinner spinner1, spinner2;
    Button btnpost;

    FirebaseDatabase fdb;
    DatabaseReference db_ref;
    FirebaseAuth mAuth;

    Calendar cal;
    String uid;
    String Time, Date;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_post);

        pd = new ProgressDialog(this);
        pd.setMessage("Loading...");
        pd.setCancelable(true);
        pd.setCanceledOnTouchOutside(false);
    }
}
```

```

getSupportActionBar().setTitle("Post Blood Request");
getSupportActionBar().setDisplayHomeAsUpEnabled(true);

text1 = findViewById(R.id.getMobile);
text2 = findViewById(R.id.getLocation);

spinner1 = findViewById(R.id.SpinnerBlood);
spinner2 = findViewById(R.id.SpinnerDivision);

btnpost = findViewById(R.id.postbtn);

cal = Calendar.getInstance();

int day = cal.get(Calendar.DAY_OF_MONTH);
int month = cal.get(Calendar.MONTH);
int year = cal.get(Calendar.YEAR);
int hour = cal.get(Calendar.HOUR);
int min = cal.get(Calendar.MINUTE);
month+=1;
Time = "";
Date = "";
String ampm="AM";

if(cal.get(Calendar.AM_PM) ==1)
{
    ampm = "PM";
}

if(hour<10)
{
    Time += "0";
}
Time += hour;
Time += ":";

if(min<10) {
    Time += "0";
}

Time +=min;
Time += (" "+ampm);

Date = day+"/"+month+"/"+year;

FirebaseUser cur_user = mAuth.getInstance().getCurrentUser();

if(cur_user == null)
{
    startActivity(new Intent(PostActivity.this, LoginActivity.class));
} else {
    uid = cur_user.getId();
}

mAuth = FirebaseAuth.getInstance();
fdb = FirebaseDatabase.getInstance();
db_ref = fdb.getReference("posts");

```

```

try {
    btnpost.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            pd.show();
            final Query findname = fdb.getReference("users").child(uid);

            if(text1.getText().length() == 0)
            {
                Toast.makeText(getApplicationContext(), "Enter your
contact number!",
                                Toast.LENGTH_LONG).show();
            }
            else if(text2.getText().length() == 0)
            {
                Toast.makeText(getApplicationContext(), "Enter your
location!",
                                Toast.LENGTH_LONG).show();
            }
            else {
                findname.addListenerForSingleValueEvent(new
ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {

                        if (dataSnapshot.exists()) {

                            db_ref.child(uid).child("Name").setValue(dataSnapshot.getValue(UserData.class).
getName());

                            db_ref.child(uid).child("Contact").setValue(text1.getText().toString());

                            db_ref.child(uid).child("Address").setValue(text2.getText().toString());

                            db_ref.child(uid).child("Division").setValue(spinner2.getSelectedItem().toStrin
g());

                            db_ref.child(uid).child("BloodGroup").setValue(spinner1.getSelectedItem().toStr
ing());

                            db_ref.child(uid).child("Time").setValue(Time);

                            db_ref.child(uid).child("Date").setValue(Date);
                                Toast.makeText(PostActivity.this, "Your post
has been created successfully",
                                                Toast.LENGTH_LONG).show();
                                startActivity(new Intent(PostActivity.this,
Dashboard.class));

                        } else {
                            Toast.makeText(getApplicationContext(),
"Database error occured.",
                                                Toast.LENGTH_LONG).show();
                        }
                    }
                });
            }
        }
    });
}

```

```

        }

        @Override
        public void onCancelled(@NonNull DatabaseError
databaseError) {
            Log.d("User", databaseError.getMessage());
        }
    });
}

});
}
catch (Exception e)
{
    e.printStackTrace();
}
pd.dismiss();
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            onBackPressed();
            return true;
    }

    return super.onOptionsItemSelected(item);
}
}

```

Profile Activities:

```

package com.android.iunoob.bloodbank.activities;

import android.app.ProgressDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;

```

```

import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import com.android.iunoob.bloodbank.R;
import com.android.iunoob.bloodbank.viewmodels.UserData;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;

public class ProfileActivity extends AppCompatActivity {

    private EditText inputemail, inputpassword, retypePassword, fullName,
address, contact;
    private FirebaseAuth mAuth;
    private Button btnSignup;
    private ProgressDialog pd;
    private Spinner gender, bloodgroup, division;

    private boolean isUpdate = false;

    private DatabaseReference db_ref, donor_ref;
    private FirebaseDatabase db_User;
    private CheckBox isDonor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        pd = new ProgressDialog(this);
        pd.setMessage("Loading...");
        pd.setCancelable(true);
        pd.setCanceledOnTouchOutside(false);
        pd.show();
        setContentView(R.layout.activity_profile);

        db_User = FirebaseDatabase.getInstance();
        db_ref = db_User.getReference("users");
        donor_ref = db_User.getReference("donors");
        mAuth = FirebaseAuth.getInstance();

        inputemail = findViewById(R.id.input_userEmail);
        inputpassword = findViewById(R.id.input_password);
        retypePassword = findViewById(R.id.input_password_confirm);
        fullName = findViewById(R.id.input_fullName);
        gender = findViewById(R.id.gender);
        address = findViewById(R.id.inputAddress);
        division = findViewById(R.id.inputDivision);
        bloodgroup = findViewById(R.id.inputBloodGroup);
        contact = findViewById(R.id.inputMobile);
        isDonor = findViewById(R.id.checkbox);
    }

```

```

btnSignup = findViewById(R.id.button_register);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);

if (mAuth.getCurrentUser() != null) {

    inputemail.setVisibility(View.GONE);
    inputpassword.setVisibility(View.GONE);
    retypePassword.setVisibility(View.GONE);
    btnSignup.setText("Update Profile");
    pd.dismiss();
    /// getSupportActionBar().setTitle("Profile");
    getSupportActionBar().setTitle("Profile");
    findViewById(R.id.image_logo).setVisibility(View.GONE);
    isUpdate = true;

    Query Profile = db_ref.child(mAuth.getCurrentUser().getUid());
    Profile.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            UserData userData = dataSnapshot.getValue(UserData.class);

            if (userData != null) {
                pd.show();
                fullName.setText(userData.getName());
                gender.setSelection(userData.getGender());
                address.setText(userData.getAddress());
                contact.setText(userData.getContact());
                bloodgroup.setSelection(userData.getBloodGroup());
                division.setSelection(userData.getDivision());
                Query donor =
donor_ref.child(division.getSelectedItem().toString())
                    .child(bloodgroup.getSelectedItem().toString())
                    .child(mAuth.getCurrentUser().getUid());

                donor.addListenerForSingleValueEvent(new
ValueEventListener() {

                    @Override
                    public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {

                        if (dataSnapshot.exists())
                        {
                            isDonor.setChecked(true);
                            isDonor.setText("Unmark this to leave from
donors");
                        }
                        else
                        {
                            Toast.makeText(ProfileActivity.this, "Your
are not a donor! Be a donor and save life by donating blood.",
                                Toast.LENGTH_LONG).show();
                        }
                        pd.dismiss();
                    }
                });
            }
        }
    });
}

```

```

        }

        @Override
        public void onCancelled(@NonNull DatabaseError
databaseError) {
            Log.d("User", databaseError.getMessage());
        }
    });
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {
    Log.d("User", databaseError.getMessage());
}
});

} else pd.dismiss();
btnSignup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final String email = inputemail.getText().toString();
        final String password = inputpassword.getText().toString();
        final String confirmPassword =
retypePassword.getText().toString();
        final String Name = fullName.getText().toString();
        final int Gender = gender.getSelectedItemId();
        final String Contact = contact.getText().toString();
        final int BloodGroup = bloodgroup.getSelectedItemId();
        final String Address = address.getText().toString();
        final int Division = division.getSelectedItemId();
        final String blood = bloodgroup.getSelectedItem().toString();
        final String div = division.getSelectedItem().toString();

        try {

            if (Name.length() <= 2) {
                ShowError("Name");
                fullName.requestFocusFromTouch();
            } else if (Contact.length() < 11) {
                ShowError("Contact Number");
                contact.requestFocusFromTouch();
            } else if (Address.length() <= 2) {
                ShowError("Address");
                address.requestFocusFromTouch();
            } else {
                if (!isUpdate) {
                    if (email.length() == 0) {
                        ShowError("Email ID");
                        inputemail.requestFocusFromTouch();
                    } else if (password.length() <= 5) {
                        ShowError("Password");
                        inputpassword.requestFocusFromTouch();
                    }
                }
            }
        }
    }
});

```

```

        } else if (password.compareTo(ConfirmPassword) != 0)
        {
            Toast.makeText(ProfileActivity.this, "Password
did not match!", Toast.LENGTH_LONG)
                .show();
            retypePassword.requestFocusFromTouch();
        } else {
            pd.show();
            mAuth.createUserWithEmailAndPassword(email,
password)

            .addOnCompleteListener(ProfileActivity.this, new
OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull
Task<AuthResult> task) {

                    if (!task.isSuccessful()) {

                        Toast.makeText(ProfileActivity.this, "Registration failed! try agian.",
                        Toast.LENGTH_LONG)
                            .show();
                        Log.v("error",
task.getException().getMessage());
                    } else {
                        String id =
mAuth.getCurrentUser().getUid();

                        db_ref.child(id).child("Name").setValue(Name);

                        db_ref.child(id).child("Gender").setValue(Gender);

                        db_ref.child(id).child("Contact").setValue(Contact);

                        db_ref.child(id).child("BloodGroup").setValue(BloodGroup);

                        db_ref.child(id).child("Address").setValue(Address);

                        db_ref.child(id).child("Division").setValue(Division);

                        if(isDonor.isChecked())
                        {

                            donor_ref.child(div).child(blood).child(id).child("UID").setValue(id).toString(
);

                            donor_ref.child(div).child(blood).child(id).child("LastDonate").setValue("Don't
donate yet!");

                            donor_ref.child(div).child(blood).child(id).child("TotalDonate").setValue(0);

                            donor_ref.child(div).child(blood).child(id).child("Name").setValue(Name);

                            donor_ref.child(div).child(blood).child(id).child("Contact").setValue(Contact);

                            donor_ref.child(div).child(blood).child(id).child("Address").setValue(Address);

```



```

    }

    Toast.makeText(getApplicationContext(), "Welcome, your account has been
    created!", Toast.LENGTH_LONG)

                                .show();
    Intent intent = new
Intent(ProfileActivity.this, Dashboard.class);

                                startActivity(intent);

                                finish();
    }
    pd.dismiss();
}

});
}

} else {

    String id = mAuth.getCurrentUser().getUid();
    db_ref.child(id).child("Name").setValue(Name);
    db_ref.child(id).child("Gender").setValue(Gender);
    db_ref.child(id).child("Contact").setValue(Contact);

    db_ref.child(id).child("BloodGroup").setValue(BloodGroup);
    db_ref.child(id).child("Address").setValue(Address);

    db_ref.child(id).child("Division").setValue(Division);

    if (isDonor.isChecked())
    {

        donor_ref.child(div).child(blood).child(id).child("UID").setValue(id).toString(
        );

        donor_ref.child(div).child(blood).child(id).child("LastDonate").setValue("Don't
        donate yet!");

        donor_ref.child(div).child(blood).child(id).child("TotalDonate").setValue(0);

        donor_ref.child(div).child(blood).child(id).child("Name").setValue(Name);

        donor_ref.child(div).child(blood).child(id).child("Contact").setValue(Contact);

        donor_ref.child(div).child(blood).child(id).child("Address").setValue(Address);

        }
        else
        {

            donor_ref.child(div).child(blood).child(id).removeValue();

        }
    }
}

```

```

        Toast.makeText(getApplicationContext(), "Your
account has been updated!", Toast.LENGTH_LONG)
            .show();
        Intent intent = new Intent(ProfileActivity.this,
Dashboard.class);
        startActivity(intent);
        finish();
    }
    pd.dismiss();

    }
} catch (Exception e) {
    e.printStackTrace();
}

    }
});
}

private void ShowError(String error) {

    Toast.makeText(ProfileActivity.this, "Please, Enter a valid "+error,
        Toast.LENGTH_LONG).show();
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            onBackPressed();
            return true;
    }

    return super.onOptionsItemSelected(item);
}
}
}

```

Restore Password:

```

package com.android.iunoob.bloodbank.activities;

import android.app.ProgressDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;

```

```

import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.android.iunoob.bloodbank.R;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class RestorePassword extends AppCompatActivity {

    Button resetbtn;
    EditText useremail;
    private FirebaseAuth mAuth;
    private ProgressDialog pd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_restore_password);

        mAuth = FirebaseAuth.getInstance();
        pd = new ProgressDialog(this);
        pd.setMessage("Loading...");
        pd.setCancelable(true);
        pd.setCanceledOnTouchOutside(false);

        useremail = findViewById(R.id.resetUsingEmail);

        findViewById(R.id.resetPassbtn).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

                FirebaseUser user = mAuth.getCurrentUser();

                final String email = useremail.getText().toString();

                if(TextUtils.isEmpty(email))
                {
                    useremail.setError("Email required!");
                }
                else
                {
                    pd.show();
                    mAuth.sendPasswordResetEmail(email)
                        .addOnCompleteListener(new
OnCompleteListener<Void>() {
                            @Override
                            public void onComplete(@NonNull Task<Void> task)
                            {
                                if(task.isSuccessful())
                                {

```

```

                                Toast.makeText(getApplicationContext(),
"We have sent an email to "+" '"+ email +"'. Please check your email.",
Toast.LENGTH_LONG)
                                .show();
                                startActivity(new
Intent(getApplicationContext(), LoginActivity.class));
                                //useremail.setText(null);
                                }
                                else
                                {
                                Toast.makeText(getApplicationContext(),
"Sorry, There is something went wrong. please try again some time later.",
Toast.LENGTH_LONG)
                                .show();
                                useremail.setText(null);
                                }
                                pd.dismiss();
                                }
                                });
                                }
                                }
                                });
                                }
                                }
}

```

Splash Activities:

```

package com.android.iunoob.bloodbank.activities;

import android.content.Intent;
import android.graphics.drawable.AnimationDrawable;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ImageView;

import com.android.iunoob.bloodbank.R;

public class SplashActivity extends AppCompatActivity {

    protected ImageView imageView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
    }
}

```

```

        imageView = findViewById(R.id.load_image);

        AnimationDrawable animationDrawable = (AnimationDrawable)
imageView.getDrawable();
        animationDrawable.start();

        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(getApplicationContext(),
LoginActivity.class);
                startActivity(intent);
                finish();
            }
        }, 2000);
    }
}

```

Blood Request Adapter:

```

package com.android.iunoob.bloodbank.adapters;

import android.graphics.Color;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.android.iunoob.bloodbank.R;
import com.android.iunoob.bloodbank.viewmodels.CustomUserData;

import java.util.List;

public class BloodRequestAdapter extends
RecyclerView.Adapter<BloodRequestAdapter.PostHolder> {

    private List<CustomUserData> postLists;

    public class PostHolder extends RecyclerView.ViewHolder
    {
        TextView Name, bloodgroup, Address, contact, posted;

        public PostHolder(@NonNull View itemView) {

```

```

        super(itemView);

        Name = itemView.findViewById(R.id.reqstUser);
        contact = itemView.findViewById(R.id.targetCN);
        bloodgroup = itemView.findViewById(R.id.targetBG);
        Address = itemView.findViewById(R.id.reqstLocation);
        posted = itemView.findViewById(R.id.posted);
    }
}

public BloodRequestAdapter(List<CustomUserData> postLists)
{
    this.postLists = postLists;
}

@Override
public PostHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
    View listitem = LayoutInflater.from(viewGroup.getContext())
        .inflate(R.layout.request_list_item, viewGroup, false);

    return new PostHolder(listitem);
}

@Override
public void onBindViewHolder(PostHolder postHolder, int i) {
    if(i%2==0)
    {
        postHolder.itemView.setBackgroundColor(Color.parseColor("#C13F31"));
    }
    else
    {
        postHolder.itemView.setBackgroundColor(Color.parseColor("#FFFFFF"));
    }

    CustomUserData customUserData = postLists.get(i);
    postHolder.Name.setText("Posted by: "+customUserData.getName());
    postHolder.Address.setText("From: "+customUserData.getAddress()+"",
"+customUserData.getDivision());
    postHolder.bloodgroup.setText("Needs "+customUserData.getBloodGroup());
    postHolder.posted.setText("Posted on: "+customUserData.getTime()+"",
"+customUserData.getDate());
    postHolder.contact.setText(customUserData.getContact());
}

@Override
public int getItemCount() {
    return postLists.size();
}
}

```

Search Donor Adapter:

```
package com.android.iunoob.bloodbank.adapters;

import android.graphics.Color;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.android.iunoob.bloodbank.R;
import com.android.iunoob.bloodbank.viewmodels.DonorData;

import java.util.List;

public class SearchDonorAdapter extends
RecyclerView.Adapter<SearchDonorAdapter.PostHolder> {

    private List<DonorData> postLists;

    public class PostHolder extends RecyclerView.ViewHolder
    {
        TextView Name, Address, contact, posted, totaldonate;

        public PostHolder(@NonNull View itemView) {
            super(itemView);

            Name = itemView.findViewById(R.id.donorName);
            contact = itemView.findViewById(R.id.donorContact);
            totaldonate = itemView.findViewById(R.id.totaldonate);
            Address = itemView.findViewById(R.id.donorAddress);
            posted = itemView.findViewById(R.id.lastdonate);
        }
    }

    public SearchDonorAdapter(List<DonorData> postLists)
    {
        this.postLists = postLists;
    }

    @Override
    public PostHolder onCreateViewHolder(ViewGroup viewGroup, int i) {

        View listitem = LayoutInflater.from(viewGroup.getContext())
            .inflate(R.layout.search_donor_item, viewGroup, false);
```

```

        return new PostHolder(listitem);
    }

    @Override
    public void onBindViewHolder(PostHolder postHolder, int i) {

        if(i%2==0)
        {
            postHolder.itemView.setBackgroundColor(Color.parseColor("#C13F31"));
        }
        else
        {
            postHolder.itemView.setBackgroundColor(Color.parseColor("#FFFFFF"));
        }
        DonorData donorData = postLists.get(i);
        postHolder.Name.setText("Name: "+donorData.getName());
        postHolder.contact.setText(donorData.getContact());
        postHolder.Address.setText("Address: "+donorData.getAddress());
        postHolder.totaldonate.setText("Total Donation:
"+donorData.getTotalDonate()+" times");
        postHolder.posted.setText("Last Donation: "+donorData.getLastDonate());

    }

    @Override
    public int getItemCount() {
        return postLists.size();
    }
}

```

About us:

```

package com.android.iunoob.bloodbank.fragments;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.text.util.Linkify;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

```



```

import com.android.iunoob.bloodbank.R;

public class AboutUs extends Fragment {
    private TextView textView;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
        ViewGroup container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.aboutus, container, false);
        getActivity().setTitle("About us");

        textView = view.findViewById(R.id.txtv);
        Linkify.addLinks(textView, Linkify.ALL);
        return view;
    }
}

```

Achievements View:

```

package com.android.iunoob.bloodbank.fragments;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.android.iunoob.bloodbank.R;
import com.android.iunoob.bloodbank.activities Dashboard;
import com.android.iunoob.bloodbank.viewmodels.DonorData;
import com.android.iunoob.bloodbank.viewmodels.UserData;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;

```

```

import com.google.firebase.database.ValueEventListener;

import java.util.Calendar;
import java.util.TimeZone;

public class AchievementsView extends Fragment {

    private int cur_day, cur_month, cur_year, day, month, year, totday;
    private Calendar calendar;
    private ProgressDialog pd;
    DatabaseReference db_ref, user_ref;
    FirebaseAuth mAuth;

    private TextView totalDonate, lastDonate, nextDonate, donateInfo;

    private String[] bloodgroup, divisionlist;
    private String lastDate;

    private View view;
    private Button yes;
    private LinearLayout yesno;

    public AchievementsView() {

    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        view = inflater.inflate(R.layout.user_achievement_fragment, container,
false);

        pd = new ProgressDialog(getActivity());
        pd.setMessage("Loading...");
        pd.setCancelable(true);
        pd.setCanceledOnTouchOutside(false);
        bloodgroup = getResources().getStringArray(R.array.Blood_Group);
        divisionlist = getResources().getStringArray(R.array.division_list);
        lastDonate = view.findViewById(R.id.setLastDonate);
        totalDonate = view.findViewById(R.id.settotalDonate);
        donateInfo = view.findViewById(R.id.donateInfo);

        getActivity().setTitle("Achievements");
        mAuth = FirebaseAuth.getInstance();
        lastDate = "";

        db_ref = FirebaseDatabase.getInstance().getReference("donors");
        user_ref = FirebaseDatabase.getInstance().getReference("users");

        Query userQ = user_ref.child(mAuth.getCurrentUser().getUid());

        try {
            pd.show();
            userQ.addListenerForSingleValueEvent(new ValueEventListener() {

```

```

@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

    if(dataSnapshot.exists())
    {
        final UserData userData =
dataSnapshot.getValue(UserData.class);
        final int getdiv = userData.getDivision();
        final int getbg = userData.getBloodGroup();
        final Query donorQ = db_ref.child(divisionlist[getdiv])
            .child(bloodgroup[getbg])
            .child(mAuth.getCurrentUser().getUid());

        donorQ.addListenerForSingleValueEvent(new
ValueEventListener() {

            @Override
            public void onDataChange(@NonNull final DataSnapshot
dataSnapshot) {

                if(dataSnapshot.exists())
                {
                    final DonorData donorData =
dataSnapshot.getValue(DonorData.class);

                    totalDonate.setText(donorData.getTotalDonate()+" times");
                    if(donorData.getTotalDonate() == 0) {
                        lastDate = "01/01/2001";
                        lastDonate.setText("Do not donate
yet!");
                    }
                    else {
                        lastDate = donorData.getLastDonate();

                    lastDonate.setText(donorData.getLastDonate());
                    }

                    totday = 0;
                    nextDonate =

view.findViewById(R.id.nextDonate);
                    yesno = view.findViewById(R.id.yesnolayout);
                    if(lastDate.length() != 0) {

                        int cnt = 0;
                        int tot = 0;
                        for (int i = 0; i < lastDate.length();

i++) {

                            if (cnt == 0 && lastDate.charAt(i)

                                day = tot;
                                tot=0;
                                cnt+=1;

                            } else if (cnt == 1 &&

lastDate.charAt(i) == '/') {

                                cnt+=1;
                                month = tot;
                                tot=0;

```

```

        (lastDate.charAt(i) - '0');

Calendar.getInstance(TimeZone.getDefault());
calendar.get(Calendar.DAY_OF_MONTH);
calendar.get(Calendar.MONTH)+1;

        } else tot = tot * 10 +
    }
    year = tot;
    calendar =
    cur_day =
    cur_month =
    cur_year = calendar.get(Calendar.YEAR);

    if(day>cur_day) {
        cur_day += 30;
        cur_month -= 1;
    }
    today += (cur_day - day);

    if(month>cur_month)
    {
        cur_month+=12;
        cur_year -=1;
    }
    today += ((cur_month - month)*30);

    today += ((cur_year - year)*365);

    try
    {
        if(today>120)
        {
            donateInfo.setText("Have you

donated today?");

nextDonate.setVisibility(View.GONE);

yesno.setVisibility(View.VISIBLE);

view.findViewById(R.id.btnYes);

calendar.get(Calendar.DAY_OF_MONTH);

calendar.get(Calendar.MONTH)+1;

calendar.get(Calendar.YEAR);

View.OnClickListener() {

{

db_ref.child(divisionlist[getdiv])

.child(bloodgroup[getbg])

        yes =

        cur_day =

        cur_month =

        cur_year =

        yes.setOnClickListener(new

            @Override
            public void onClick(View v)

```

```

        .child(mAuth.getCurrentUser().getUid())

        .child("LastDonate").setValue(cur_day+"/"+cur_month+"/"+cur_year);

        db_ref.child(divisionlist[getdiv])

        .child(bloodgroup[getbg])

        .child(mAuth.getCurrentUser().getUid())

        .child("TotalDonate").setValue(donorData.getTotalDonate()+1);
        startActivity(new
Intent(getActivity(), Dashboard.class));

    }
    });
}
else
{
    donateInfo.setText("Next
donation available in:");
    yesno.setVisibility(View.GONE);

    nextDonate.setVisibility(View.VISIBLE);
    nextDonate.setText((120-today)+" days");
}
} catch (Exception e)
{
    e.printStackTrace();
}

}

else
{
    LinearLayout linearLayout =
view.findViewById(R.id.donorAchiev);
    linearLayout.setVisibility(View.GONE);
    TextView tv =
view.findViewById(R.id.ShowInof);
    tv.setVisibility(View.VISIBLE);
    Toast.makeText(getActivity(), "Update your
profile to be a donor first.", Toast.LENGTH_LONG)
        .show();
}
pd.dismiss();
}

@Override
public void onCancelled(@NonNull DatabaseError
databaseError) {

}

```

```

        });

        }
        else
        {
            Toast.makeText(getActivity(), "You are not a
user."+divisionlist[0]+" "+bloodgroup[0], Toast.LENGTH_LONG)
                .show();
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

        Log.d("User", databaseError.getMessage());
    }

});

} catch (Exception e)
{
    e.printStackTrace();
}

return view;
}
}

```

Blood Info:

```

package com.android.iunoob.bloodbank.fragments;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.android.iunoob.bloodbank.R;

public class BloodInfo extends Fragment {

    @Nullable

```

```

        @Override
        public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
        ViewGroup container, @Nullable Bundle savedInstanceState) {

            View view = inflater.inflate(R.layout.bloodinfo, container, false);
            getActivity().setTitle("Information");
            return view;
        }
    }
}

```

Home View:

```

package com.android.iunoob.bloodbank.fragments;

import android.app.ProgressDialog;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v7.widget.DefaultItemAnimator;
import android.support.v7.widget.DividerItemDecoration;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.android.iunoob.bloodbank.R;
import com.android.iunoob.bloodbank.adapters.BloodRequestAdapter;
import com.android.iunoob.bloodbank.viewmodels.CustomUserData;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

public class HomeView extends Fragment {

```

```

private View view;
private RecyclerView recentPosts;

private DatabaseReference donor_ref;
private FirebaseAuth mAuth;
private BloodRequestAdapter restAdapter;
private List<CustomUserData> postLists;
private ProgressDialog pd;

public HomeView() {

}

@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {

    view = inflater.inflate(R.layout.home_view_fragment, container, false);
    recentPosts = (RecyclerView) view.findViewById(R.id.recyleposts);

    recentPosts.setLayoutManager(new LinearLayoutManager(getContext()));

    donor_ref = FirebaseDatabase.getInstance().getReference();
    postLists = new ArrayList<>();

    pd = new ProgressDialog(getActivity());
    pd.setMessage("Loading...");
    pd.setCancelable(true);
    pd.setCanceledOnTouchOutside(false);

    mAuth = FirebaseAuth.getInstance();
    getActivity().setTitle("Blood Point");

    restAdapter = new BloodRequestAdapter(postLists);
    RecyclerView.LayoutManager pmLayout = new
LinearLayoutManager(getContext());
    recentPosts.setLayoutManager(pmLayout);
    recentPosts.setItemAnimator(new DefaultItemAnimator());
    recentPosts.addItemDecoration(new DividerItemDecoration(getActivity(),
LinearLayoutManager.VERTICAL));
    recentPosts.setAdapter(restAdapter);

    AddPosts();
    return view;

}

private void AddPosts()
{
    Query allposts = donor_ref.child("posts");
    pd.show();
    allposts.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            if(dataSnapshot.exists()) {

```



```

        for (DataSnapshot singlepost : dataSnapshot.getChildren()) {
            CustomUserData customUserData =
singlepost.getValue(CustomUserData.class);
            postLists.add(customUserData);
            restAdapter.notifyDataSetChanged();
        }
        pd.dismiss();
    }
    else
    {
        Toast.makeText(getActivity(), "Database is empty now!",
            Toast.LENGTH_LONG).show();
    }
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {

    Log.d("User", databaseError.getMessage());

}

});

}

@Override
public void onResume() {
    super.onResume();
}

@Override
public void onStop() {
    super.onStop();
}

@Override
public void onPause() {
    super.onPause();
}

@Override
public void onDestroy() {
    super.onDestroy();
}
}

```

Nearby Hospitals:

```

package com.android.iunoob.bloodbank.fragments;

```

```

import android.Manifest;
import android.content.pm.PackageManager;
import android.location.Location;
import android.os.Build;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.support.v4.content.ContextCompat;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.android.iunoob.bloodbank.R;
import com.android.iunoob.bloodbank.viewmodels.GetNearbyPlacesData;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.location.LocationSettingsRequest;
import com.google.android.gms.location.SettingsClient;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnSuccessListener;

public class NearByHospitalActivity extends Fragment implements
    OnMapReadyCallback, GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener, LocationListener {

    private GoogleMap mMap;
    View view;
    private GoogleApiClient client;
    private LocationRequest locationRequest;
    Location lastlocation;
    private Marker currentLocationmMarker = null;
    private static final int Permission_Request = 99;
    int PROXIMITY_RADIUS = 10000;
    private FusedLocationProviderClient fusedLocationProviderClient;

    @Nullable
    @Override

```

```

    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
    ViewGroup container, @Nullable Bundle savedInstanceState) {

        view = inflater.inflate(R.layout.near_by_hospitals, container, false);
        getActivity().setTitle("Nearest hospitals");

        return view;
    }

    @Override
    public void onViewCreated(@NonNull final View view, @Nullable Bundle
    savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        fusedLocationProviderClient =
        LocationServices.getFusedLocationProviderClient(getActivity());

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            checkLocationPermission();
        }

        SupportMapFragment mapFragment = (SupportMapFragment)
        getChildFragmentManager().findFragmentById(R.id.gMap);
        if (mapFragment != null) {
            mapFragment.getMapAsync(this);
        }
        else
        {
            Toast.makeText(getActivity(), "MapFragment is null, why?",
            Toast.LENGTH_LONG).show();
        }

    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
    permissions, @NonNull int[] grantResults) {
        switch (requestCode) {
            case Permission_Request:
                if (grantResults.length > 0 && grantResults[0] ==
                PackageManager.PERMISSION_GRANTED) {
                    if (ContextCompat.checkSelfPermission(getActivity(),
                    Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED)
                    {
                        if (client == null) {
                            buildGoogleApiClient();
                        }
                        mMap.setMyLocationEnabled(true);
                        mMap.setTrafficEnabled(true);
                    }
                } else {
                    Toast.makeText(getActivity(), "Permission Denied",
                    Toast.LENGTH_LONG).show();
                }
            }
        }
    }

```

```

@Override
public void onMapReady(GoogleMap googleMap) {

    mMap = googleMap;

    if (ContextCompat.checkSelfPermission(getActivity(),
Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED)
    {
        buildGoogleApiClient();
        mMap.setMyLocationEnabled(true);
        mMap.setTrafficEnabled(true);
    }

}

protected synchronized void buildGoogleApiClient() {
    client = new
GoogleApiClient.Builder(getActivity().getApplicationContext()).addConnectionCal
lbacks(this).addOnConnectionFailedListener(this).addApi(LocationServices.API).b
uild();
    client.connect();
}

@Override
public void onConnected(@Nullable Bundle bundle) {
    locationRequest = new LocationRequest();
    locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

    LocationSettingsRequest.Builder builder = new
LocationSettingsRequest.Builder();
    builder.addLocationRequest(locationRequest);
    LocationSettingsRequest locationSettingsRequest = builder.build();

    SettingsClient settingsClient =
LocationServices.getSettingsClient(getActivity());
    settingsClient.checkLocationSettings(locationSettingsRequest);

    if (ActivityCompat.checkSelfPermission(getActivity(),
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED
        && ActivityCompat.checkSelfPermission(getActivity(),
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        Toast.makeText(getActivity(), "You need to enable permissions to
display location !", Toast.LENGTH_SHORT).show();
    }

    fusedLocationProviderClient.getLastLocation().addOnSuccessListener(getActivity(
), new OnSuccessListener<Location>() {
        @Override
        public void onSuccess(Location location) {
            ShowHospitals(location.getLatitude(), location.getLongitude());
        }
    }
}

```

```

    });

}

    private String getUrl(double latitude, double longitude, String nearbyPlace)
{
    StringBuilder googlePlaceUrl = new
StringBuilder("https://maps.googleapis.com/maps/api/place/nearbysearch/json?")
;

googlePlaceUrl.append("location=").append(latitude).append(",").append(longitud
e);

    googlePlaceUrl.append("&radius=").append(PROXIMITY_RADIUS);
    googlePlaceUrl.append("&type=").append(nearbyPlace);
    googlePlaceUrl.append("&sensor=true");
    googlePlaceUrl.append("&key=" +
"AIzaSyAmeQ8IwQWBcmFLRpKARu7LM1TlShQKmfq");

    Log.d("NearbyHospitalActivity", "url = " + googlePlaceUrl.toString());

    return googlePlaceUrl.toString();
}

    public boolean checkLocationPermission() {
        if (ContextCompat.checkSelfPermission(getActivity(),
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED)
        {

            if
(ActivityCompat.shouldShowRequestPermissionRationale(getActivity(),
Manifest.permission.ACCESS_FINE_LOCATION)) {
                ActivityCompat.requestPermissions(getActivity(), new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, PERMISSION_REQUEST);
            } else {
                ActivityCompat.requestPermissions(getActivity(), new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, PERMISSION_REQUEST);
            }
            return false;
        } else
            return true;
    }

    @Override
    public void onConnectionSuspended(int i) {

    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }

    @Override
    public void onLocationChanged(Location location) {

```

```

        lastlocation = location;
        LatLng latLng = new LatLng(location.getLatitude() ,
location.getLongitude());
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(latLng);
        markerOptions.title("Current Location");

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactor
y.HUE_BLUE));
        currentLocationmMarker = mMap.addMarker(markerOptions);
        mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mMap.animateCamera(CameraUpdateFactory.zoomBy(0));

    }

    @Override
    public void onStart() {
        super.onStart();

        if(client!=null)
        {
            client.connect();
        }
    }

    public void ShowHospitals(double latitude, double longitude)
    {
        mMap.clear();
        Object dataTransfer[] = new Object[2];
        GetNearbyPlacesData getNearbyPlacesData = new GetNearbyPlacesData();
        String url = getUrl(latitude, longitude, "hospital");
        dataTransfer[0] = mMap;
        dataTransfer[1] = url;

        getNearbyPlacesData.execute(dataTransfer);
        Toast.makeText(getApplicationContext(), "Showing Nearby Hospitals",
Toast.LENGTH_SHORT).show();
    }
}

```

Search Donor Fragments:

```

package com.android.iunoob.bloodbank.fragments;

import android.app.ProgressDialog;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v7.widget.DefaultItemAnimator;

```

```

import android.support.v7.widget.DividerItemDecoration;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.Toast;

import com.android.iunoob.bloodbank.R;
import com.android.iunoob.bloodbank.adapters.SearchDonorAdapter;
import com.android.iunoob.bloodbank.viewmodels.DonorData;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

public class SearchDonorFragment extends Fragment {

    private View view;

    FirebaseAuth mAuth;
    FirebaseUser fuser;
    FirebaseDatabase fdb;
    DatabaseReference db_ref, user_ref;

    Spinner bloodgroup, division;
    Button btnsearch;
    ProgressDialog pd;
    List<DonorData> donorItem;
    private RecyclerView recyclerView;

    private SearchDonorAdapter sdadapter;

    public SearchDonorFragment() {

    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
    ViewGroup container, @Nullable Bundle savedInstanceState) {

        view = inflater.inflate(R.layout.search_donor_fragment, container,
false);

        pd = new ProgressDialog(getActivity());

```

```

pd.setMessage("Loading...");
pd.setCancelable(true);
pd.setCanceledOnTouchOutside(false);

 mAuth = FirebaseAuth.getInstance();
 fuser = mAuth.getCurrentUser();
 fdb = FirebaseDatabase.getInstance();
 db_ref = fdb.getReference("donors");

 bloodgroup = view.findViewById(R.id.btngetBloodGroup);
 division = view.findViewById(R.id.btngetDivison);
 btnsearch = view.findViewById(R.id.btnSearch);

 getActivity().setTitle("Find Blood Donor");

 btnsearch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        pd.show();
        donorItem = new ArrayList<>();
        donorItem.clear();
        sdadapter = new SearchDonorAdapter(donorItem);
        recyclerView = (RecyclerView)
view.findViewById(R.id.showDonorList);
        recyclerView.setLayoutManager(new
LinearLayoutManager(getContext()));
        RecyclerView.LayoutManager searchdonor = new
LinearLayoutManager(getContext());
        recyclerView.setLayoutManager(searchdonor);
        recyclerView.setItemAnimator(new DefaultItemAnimator());
        recyclerView.addItemDecoration(new
DividerItemDecoration(getActivity(), LinearLayoutManager.VERTICAL));
        recyclerView.setAdapter(sdadapter);
        Query qpath =
db_ref.child(division.getSelectedItem().toString())
            .child(bloodgroup.getSelectedItem().toString());
        qpath.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot)
            {
                if(dataSnapshot.exists())
                {
                    for(DataSnapshot singleitem :
dataSnapshot.getChildren())
                    {
                        DonorData donorData =
singleitem.getValue(DonorData.class);
                        donorItem.add(donorData);
                        sdadapter.notifyDataSetChanged();
                    }
                }
            }
            else
            {

```



```

        Toast.makeText(getActivity(), "Database is empty
now!",
                        Toast.LENGTH_LONG).show();
    }
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError)
{
    Log.d("User", databaseError.getMessage());
}
});
pd.dismiss();
}
});
return view;
}
}

```

Customer Data:

```

package com.android.iunoob.bloodbank.viewmodels;

import java.io.Serializable;

public class CustomUserData implements Serializable {
    private String Address, Division, Contact;
    private String Name, BloodGroup;
    private String Time, Date;

    public CustomUserData() {

    }

    public CustomUserData(String address, String division, String contact,
String name, String bloodGroup, String time, String date) {
        Address = address;
        Division = division;
        Contact = contact;
        Name = name;
        BloodGroup = bloodGroup;
        Time = time;
        Date = date;
    }
}

```

```
public String getAddress() {  
    return Address;  
}  
  
public void setAddress(String address) {  
    this.Address = address;  
}  
  
public String getDivision() {  
    return Division;  
}  
  
public void setDivision(String division) {  
    this.Division = division;  
}  
  
public String getContact() {  
    return Contact;  
}  
  
public void setContact(String contact) {  
    this.Contact = contact;  
}  
  
public String getName() {  
    return Name;  
}  
  
public void setName(String name) {  
    this.Name = name;  
}  
  
public String getBloodGroup() {  
    return BloodGroup;  
}  
  
public void setBloodGroup(String bloodGroup) {  
    this.BloodGroup = bloodGroup;  
}  
  
public String getTime() {  
    return Time;  
}  
  
public void setTime(String time) {  
    this.Time = time;  
}  
  
public String getDate() {  
    return Date;  
}  
  
public void setDate(String date) {  
    this.Date = date;  
}  
}
```

Data Parser:

```
package com.android.iunoob.bloodbank.viewmodels;

import android.util.Log;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class DataParser {

    private HashMap<String, String> getPlace(JSONObject googlePlaceJson)
    {
        HashMap<String, String> googlePlaceMap = new HashMap<>();
        String placeName = "--NA--";
        String vicinity= "--NA--";
        String latitude= "";
        String longitude="";
        String reference="";

        Log.d("DataParser","jsonobject =" +googlePlaceJson.toString());

        try {
            if (!googlePlaceJson.isNull("name")) {
                placeName = googlePlaceJson.getString("name");
            }
            if (!googlePlaceJson.isNull("vicinity")) {
                vicinity = googlePlaceJson.getString("vicinity");
            }

            latitude =
googlePlaceJson.getJSONObject("geometry").getJSONObject("location").getString("
lat");
            longitude =
googlePlaceJson.getJSONObject("geometry").getJSONObject("location").getString("
lng");

            reference = googlePlaceJson.getString("reference");

            googlePlaceMap.put("place_name", placeName);
            googlePlaceMap.put("vicinity", vicinity);
```

```

        googlePlaceMap.put("lat", latitude);
        googlePlaceMap.put("lng", longitude);
        googlePlaceMap.put("reference", reference);

    }
    catch (JSONException e) {
        e.printStackTrace();
    }
    return googlePlaceMap;
}

private List<HashMap<String, String>>getPlaces(JSONArray jsonArray)
{
    int count = jsonArray.length();
    List<HashMap<String, String>> placelist = new ArrayList<>();
    HashMap<String, String> placeMap = null;

    for(int i = 0; i<count;i++)
    {
        try {
            placeMap = getPlace((JSONObject) jsonArray.get(i));
            placelist.add(placeMap);
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
    return placelist;
}

public List<HashMap<String, String>> parse(String jsonData)
{
    JSONArray jsonArray = null;
    JSONObject jsonObject;

    Log.d("json data", jsonData);

    try {
        jsonObject = new JSONObject(jsonData);
        jsonArray = jsonObject.getJSONArray("results");
    } catch (JSONException e) {
        e.printStackTrace();
    }
    return getPlaces(jsonArray);
}
}

```

Donor Data:

```
package com.android.iunoob.bloodbank.viewmodels;

public class DonorData {

    private int TotalDonate;
    private String LastDonate, Name, Contact, UID, Address;

    public DonorData() {

    }

    public DonorData(int totalDonate, String lastDonate, String Name, String
Contact, String Address, String UID) {
        this.TotalDonate = totalDonate;
        this.LastDonate = lastDonate;
        this.Name = Name;
        this.Contact = Contact;
        this.UID = UID;
        this.Address = Address;
    }

    public int getTotalDonate() {
        return TotalDonate;
    }

    public void setTotalDonate(int totalDonate) {
        this.TotalDonate = totalDonate;
    }

    public String getLastDonate() {
        return LastDonate;
    }

    public void setLastDonate(String lastDonate) {
        this.LastDonate = lastDonate;
    }

    public String getUID() {
        return UID;
    }

    public void setUID(String UID) {
        this.UID = UID;
    }

    public String getName() {
        return Name;
    }

    public void setDonorName(String donorName) {
        this.Name = Name;
    }

    public String getContact() {
        return Contact;
    }
}
```

```

    public void setContact(String donorContact) {
        this.Contact = Contact;
    }

    public String getAddress() {
        return Address;
    }

    public void setAddress(String address) {
        Address = address;
    }
}

```

Download URL:

```

package com.android.iunoob.bloodbank.viewmodels;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

public class DownloadUrl {

    public String readUrl(String myUrl) throws IOException
    {
        String data = "";
        InputStream inputStream = null;
        HttpURLConnection urlConnection = null;

        try
        {
            URL url = new URL(myUrl);
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.connect();

            inputStream = urlConnection.getInputStream();
            BufferedReader br = new BufferedReader(new
InputStreamReader(inputStream));
            StringBuffer sb = new StringBuffer();
            String line="";
            while ((line = br.readLine()) != null)
            {
                sb.append(line);
            }

```

```

        data = sb.toString();
        br.close();

    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    finally {
        inputStream.close();
        urlConnection.disconnect();
    }

    return data;
}
}

```

Get Nearby Place Data:

```

package com.android.iunoob.bloodbank.viewmodels;

import android.os.AsyncTask;
import android.util.Log;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

import java.io.IOException;
import java.util.HashMap;
import java.util.List;

public class GetNearbyPlacesData extends AsyncTask<Object, String, String> {

    String googlePlacesData;
    GoogleMap mMap;
    String url;

    @Override
    protected String doInBackground(Object... objects) {
        mMap = (GoogleMap) objects[0];
        url = (String) objects[1];

        DownloadUrl downloadUrl = new DownloadUrl();
        try {

```

```

        googlePlacesData = downloadUrl.readUrl(url);
    } catch (IOException e) {
        e.printStackTrace();
    }

    return googlePlacesData;
}

@Override
protected void onPostExecute(String s) {

    List<HashMap<String, String>> nearbyPlaceList;
    DataParser parser = new DataParser();
    nearbyPlaceList = parser.parse(s);

    showNearbyPlaces(nearbyPlaceList);
}

private void showNearbyPlaces(List<HashMap<String, String>> nearbyplaces)
{
    for(int i=0; i<nearbyplaces.size(); i++)
    {
        MarkerOptions markerOptions = new MarkerOptions();
        HashMap<String, String> googlePlace = nearbyplaces.get(i);

        String PlaceName = googlePlace.get("place_name");
        String vicinity = googlePlace.get("vicinity");
        double lat = Double.parseDouble(googlePlace.get("lat"));
        double lng = Double.parseDouble(googlePlace.get("lng"));

        LatLng latLng = new LatLng(lat, lng);
        markerOptions.position(latLng);
        markerOptions.title(PlaceName+" "+vicinity);

markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactor
y.HUE_BLUE));

        mMap.addMarker(markerOptions);
        mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mMap.animateCamera(CameraUpdateFactory.zoomBy(10));
    }
}
}

```

User Data:

```
package com.android.iunoob.bloodbank.viewmodels;
```



```
public class UserData {

    private String Name, Email, Contact, Address;
    private int Gender, BloodGroup, Division;

    public UserData() {

    }

    public String getContact() {
        return Contact;
    }

    public void setContact(String contact) {
        Contact = contact;
    }

    public String getAddress() {
        return Address;
    }

    public void setAddress(String address) {
        this.Address = address;
    }

    public int getDivision() {
        return Division;
    }

    public void setDivision(int division) {
        this.Division = division;
    }

    public String getName() {
        return Name;
    }

    public int getBloodGroup() {
        return BloodGroup;
    }

    public void setBloodGroup(int bloodGroup) {
        this.BloodGroup = bloodGroup;
    }

    public String getEmail() {
        return Email;
    }

    public int getGender() {
        return Gender;
    }

    public void setName(String name) { this.Name = name; }

    public void setEmail(String email) {
```

```
        this.Email = email;
    }

    public void setGender(int gender) {
        this.Gender = gender;
    }
}
```

5.3 Conclusions

The proposed system provides an Android based application which is extremely useful at Emergency Services i.e. at the time of Blood Donation, insertion, etc. this method provides a more robust way to communicate with Blood Donors. The system provides a more robust way to communicate with blood banks. It's also ready to maintain reports like stock, blood requirements, etc. It's easy to keep up the records through a database of the registered Donor's. It also provides

us knowledge about the most recent technology utilized in developing android based applications.

5.4 Future work

Some of the future scopes that can be done to this system are:

1. Add chat option for donor and user.
2. Create a website based on this apps.
3. Realtime location direction of patient to donor.
4. Health checkup information update of donor.