NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC
Accredited by NAAC with 'A' Grade & Accredited by NBA

A MINI PROJECT REPORT

for

Mini Project in Mobile Application Development (20CSE77A)

on

Life360+ : Extending Healthcare to your Fingertips

Submitted by

RAHUL MUSALIYATH DINESH
USN: 1NH18CS738, Sem-Sec: 7-D

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Academic Year: 2021-22

NEW HORIZON COLLEGE OF ENGINEERING

Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC
Accredited by NAAC with 'A' Grade & Accredited by NBA

CERTIFICATE

This is to certify that the mini project work titled

Life360+ : Extending

Healthcare to your Fingertips

Submitted in partial fulfillment of the degree of Bachelor of Engineering in Computer Science and Engineering by

RAHUL MUSALIYATH DINESH
USN: 1NH18CS738

DURING

ODD SEMESTER 2021-2022

for

Course: Mini-project in Mobile Application Development - 20CSE77A

Signature of Reviewer

Signature of HOD

SEMESTER END EXAMINATION

Name of the Examiner	*Signature with date*
1. _____	_____
2. _____	_____

PLAGIARISM CERTIFICATE

ORIGINALITY REPORT

9%	%	9%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Reto Meier, Ian Lake. "Professional Android®", Wiley, 2018 Publication	2%
2	Ana C. R. Paiva, Joao M. E. P. Gouveia, Jean-David Elizabeth, Marcio E. Delamaro. "Testing When Mobile Apps Go to Background and Come Back to Foreground", 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2019 Publication	1%
3	Manikandan B, Kishore R, Saran Kumar K, Suriya S, Vivek K V. "Enhanced Security for ATM Machine with OTP and Facial Recognition Features", International Research Journal of Multidisciplinary Technovation, 2019 Publication	1%
4	Hanoi University Publication	1%
5	Android Recipes, 2012.	

ABSTRACT

Access to an affordable and working healthcare system is one of the aspects by which a country can be classified upon. Today, healthcare in India has become of the fastest growing sectors with respect to both employment and revenue.

But as this sector continues to grow, the poor and downtrodden sections of our society are being left behind. They have to make do with the crumbling public healthcare system which by all means would mostly fail due to lack of adequate infrastructure and doctors.

The COVID 19 pandemic has also taught the world about the importance of blood/plasma banks that have played a key role in this battle against the virus. Plasma banks which collect the plasma of recovered patients have played a pivotal role in helping patients affected by the virus, to recover from it.

This mini-project solves the above-mentioned problems through the development of a mobile application. This mobile application would maintain a database of the users and other relevant information and would have a graphical user interface that would be simple and easy to use. The entire app has been developed in Android Studio and uses the Android Studio IDE for running the application.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

I have great pleasure in expressing gratitude to **Dr. Mohan Manghnani**, Chairman of New Horizon Educational Institutions for providing necessary infrastructure and creating good environment.

I take this opportunity to express my profound gratitude to **Dr. Manjunatha,** Principal, New Horizon College of Engineering, for his constant support and encouragement.

I am grateful to **Dr. Amarjeet Singh,** Professor and Dean-Academics, NHCE for his unfailing encouragement and suggestions, given to me in the course of my project work.

I would also like to thank **Dr. B. Rajalakshmi**, Professor and Head, Department of Computer Science and Engineering, for her constant support.

I also express my gratitude to **Ms. Uma N.**, Senior Assistant Professor, Department of Computer Science and Engineering, my project guide, for constantly monitoring the development of the project and setting up precise deadlines. Her valuable suggestions were the motivating factors in completing the work.

RAHUL MUSALIYATH DINESH

USN: 1NH18CS738

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 ABOUT ANDROID

Android is an **open source** and **Linux-based Operating System** for mobile devices such as smartphones and tablet computers. Android was **developed by the *Open Handset Alliance***, led by Google, and other companies.

Android offers a **unified approach to application development** for mobile devices which means developers need only to develop for Android. These applications then should be able to **run on different devices powered by Android**.

It is an open-source software stack that includes

- operating system
- middleware
- key mobile applications
- set of API libraries for writing applications that can shape the look, feel, and function of the devices on which they run

Modern mobile devices have become powerful tools that incorporate

- Touchscreens
- Cameras
- media players
- Global Positioning System (GPS) receivers
- Near Field Communications (NFC) hardware

In Android, native and third-party applications are written with the same APIs and executed on the same run time. These APIs feature include:

- hardware access
- video recording
- location-based services
- support for background services
- map-based activities
- relational databases
- inter-application communication
- Bluetooth
- NFC
- 2D and 3D graphics

Android has powerful APIs, a thriving developer community, excellent documentation, no development or distribution costs. • Android is an ecosystem made up of a combination of three components:

- A free, open-source operating system for embedded devices
- An open-source development platform for creating applications
- Devices, particularly mobile phones, that run the Android operating system and the applications created for it

1.2 ANDROID ARCHITECTURE

The Android software stack is, put simply, a Linux kernel and a collection of C/C++ libraries exposed through an application framework that provides services for, and management of, the run time and applications.

The Android software stack is composed of the following elements :

1. **Linux kernel**: Linux Kernel is heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime.
    - The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc.
    - The features of Linux kernel are:
        - **Security**: The Linux kernel handles the security between the application and the system.
        - **Memory Management**: It efficiently handles the memory management thereby providing the freedom to develop our apps.
        - **Process Management**: It manages the process well, allocates resources to processes whenever they need them.
        - **Network Stack**: It effectively handles the network communication.
        - **Driver Model**: It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.
2. **Libraries**: The Platform Libraries includes various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide a support for android development.
    - Media library provides support to play and record an audio and video formats.
    - Surface manager responsible for managing access to the display subsystem.

- SGL and OpenGL both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
- SQLite provides database support and FreeType provides font support.
- Web-Kit This open source web browser engine provides all the functionality to display web content and to simplify page loading.
- SSL (Secure Sockets Layer) is security technology to establish an encrypted link between a web server and a web browser.

3. **Android run time**:
   - The run time is what makes an Android phone an Android phone rather than a mobile Linux implementation. It includes:
     - **Core libraries**: The core Android libraries provide most of the functionality available in the core Java libraries, as well as the Android specific libraries.
     - **Dalvik VM**: Dalvik is a register-based Virtual Machine that's been optimized to ensure that a device can run multiple instances efficiently. It relies on the Linux kernel for threading and low-level memory management.
   - Android Runtime environment is one of the most important part of Android.
   - It contains components like core libraries and the Dalvik virtual machine(DVM).
   - Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.
   - Like Java Virtual Machine (JVM), **Dalvik Virtual Machine (DVM)** is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently.
   - It depends on the layer Linux kernel for threading and low-level memory management.
   - The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.

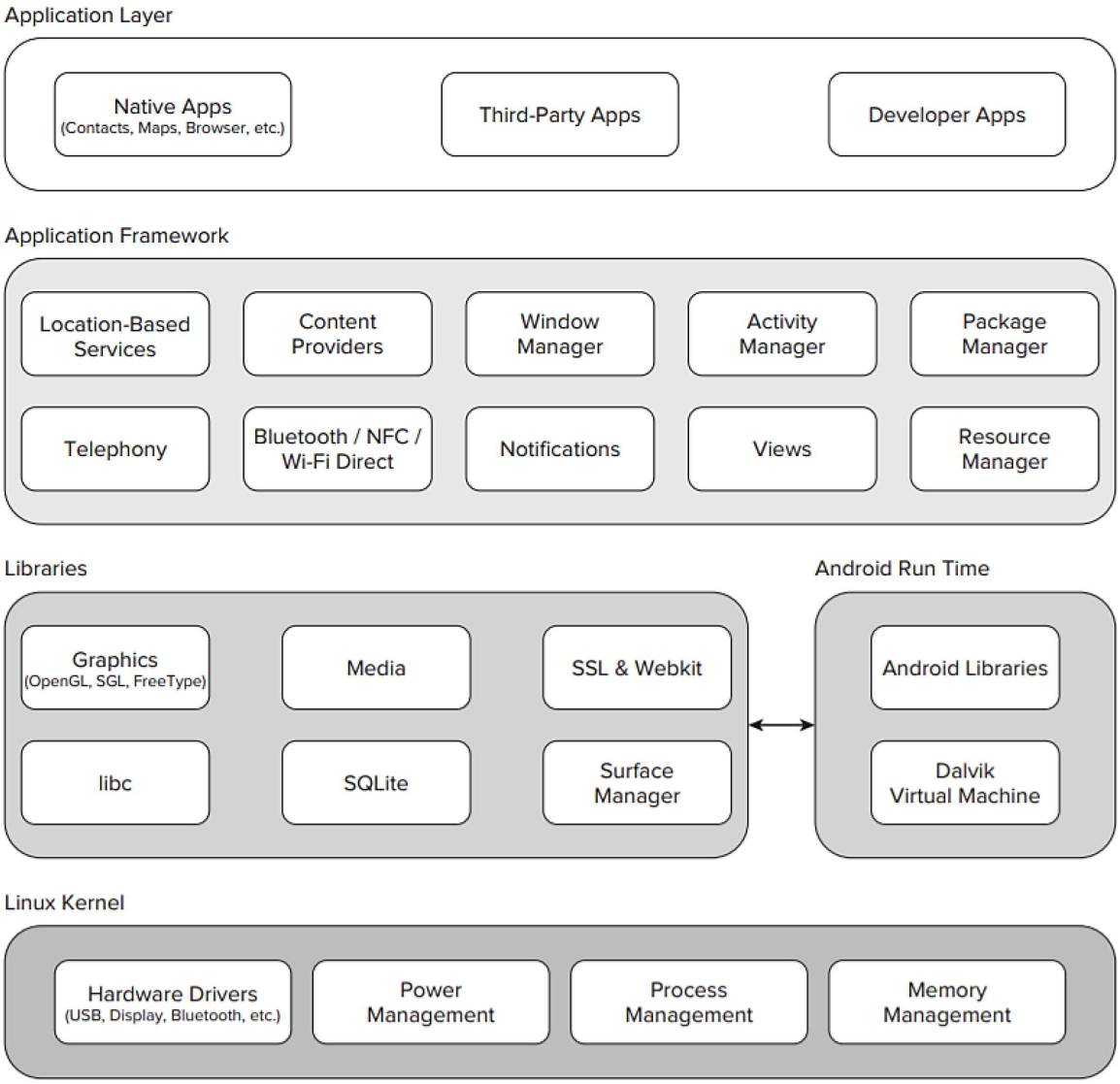

Figure 1.1: Android Architecture

4. **Application framework**:
   - It provides the classes used to create Android apps. It also provides a generic abstraction for hardware access and manages the user interface (UI) and application resources.
   - Application Framework provides several important classes which are used to create an Android application.
   - It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources.

- Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.
- It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite.

5. **Application layer**:
   - All applications, both native and third-party, are built on the application layer by means of the same API libraries.
   - The application layer runs within the Android run time, using the classes and services made available from the application framework.

1.3 ABOUT THE MINI-PROJECT

In India, access to free healthcare remains a big issue especially in the rural parts of our country where the many of the downtrodden citizens of our country live. Healthcare when available is also quite expensive and this has impacted the poor of our country the most.

Another issue that is common in our country is the lack of timely access to blood and plasma banks which have resulted in many lives being lost. The COVID-19 pandemic has once again shown the importance of maintaining these banks as plasma from recovered patients has proved to be effective in treating those who have been infected. This mini-project aims to solve the aforementioned problem.

1.4 OBJECTIVES OF THE MINI-PROJECT

Once developed, the application will provide services to

- Register users and maintain a database with their required information
- Register Blood Banks/Hospitals to record blood/plasma available with them
- Help those users who require immediate access to blood and plasma

- Allow users to volunteer for any emergency blood donation requests.
- Find out the nearest donor when a user has an emergency request (within a city)

1.5 ADVANTAGES OF THE MINI-PROJECT

The mini-project aims to provide a simple and secure platform to register users and maintain a database of their information. It also would be able to register hospitals and blood banks and maintain a database with relevant data. It would be able to provide users who require immediate access to blood and plasma. While registering, users would also have the option of signing up for blood donation. It would also help users who are patients to connect with the nearest donors/hospital or blood banks.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 MOBILE REQUIREMENTS

The following are the mobile specifications required for the implementation of this miniproject:

- Operating system: Android 4.2 or above
- CPU speed: Quad Core 1.2GHz or faster
- Storage: 8 GB or more
- RAM: 2 GB or more

2.2 COMPUTER REQUIREMENTS

The following are the development computer's specifications required for the implementation of this miniproject:

- Operating system : Microsoft Windows 8 or Microsoft Windows 10
- Android Studio version: Arctic Fox | 2020.3.1
- Android Studio SDK: Android 11 (API level 30)
- Android Studio Emulator: 30.4.5 (February 23, 2021)
- RAM: 8 GB or above
- Screen Resolution: 1280 x 800 minimum screen resolution
- Disk Space: 8 GB of available disk space minimum
- CPU: x86_64 (64 bit) CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor

CHAPTER 3

ANALYSIS AND DESIGN

3.1 PSEUDOCODE

The implementation of the mini-project is done using the Java programming language using the Android Studio Integrated Development Environment. Various concepts that form the foundations of Android Programming like Intents, Views and Activities have been employed extensively to realize the mini-project.

The following is the code snippet used in the mini-project to create the activity called MainActivity, which is the activity that is launched first.

```
package com.example.life360plus;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getSupportActionBar().hide();
    }
```

```
    public void goToLoginActivity(View v){
        Intent loginActivity = new
            Intent(getApplicationContext(),LoginActivity.class);
        startActivity(loginActivity);
    }

    public void goToRegistrationPage(View v){
        Intent loginActivity = new
           Intent(getApplicationContext(),SelectRegType.class);
        startActivity(loginActivity);
    }
}
```

The following is the code snippet of the XML file that is referenced in the above Java code. It is used to create the layout of the activity MainActivity.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#D2F1F1"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="347dp"
        android:layout_height="211dp"
        android:layout_marginTop="160dp"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/header" />

    <Button
        android:id="@+id/loginButton"
        android:layout_width="750px"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:backgroundTint="#447eed"
        android:onClick="goToLoginActivity"
        android:text="Login"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />

    <Button
        android:id="@+id/registerButton"
        android:layout_width="750px"
        android:layout_height="wrap_content"
        android:layout_marginTop="25dp"
        android:backgroundTint="#447eed"
        android:onClick="goToRegistrationPage"
        android:text="Register"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/loginButton"
       />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Hence, the Java code provides the backend implantation while the XML file is used to create the frontend which will be interacting with the user of the application. The onCreate() function is used to create the layout of the Activity by referencing the XML file. It also loads any previous state if any. Additionally, here, it also hides the support action bar to provide a better UI experience.

The functions goToRegistrationPage() and goToLoginActivity() are assigned to the buttons having id registerButton and loginButton using the android:onClick attribute in the respective button elements located in the XML file. The XML file uses the Constraint Layout as the base layout to organize the various elements.

3.2 FLOWCHART

The flowchart representation of the mini-project's implementation, provided in the next page, gives an overview of the various activities used and how the user is redirected from one activity to another using Intents.

The activity, MainActivity is launched when the user taps on the app's icon located in the app drawer. This activity presents the user with 2 buttons which will redirect the user either to the Login activity or the Registration Type selection activity.

The Login activity will have 2 buttons. One will be used for authenticating the credentials entered by the user, while the other will be used for navigating the user to the Forgot Password activity. The Registration Type selection activity will redirect the user either to the Individual user / Organization user registration activity.

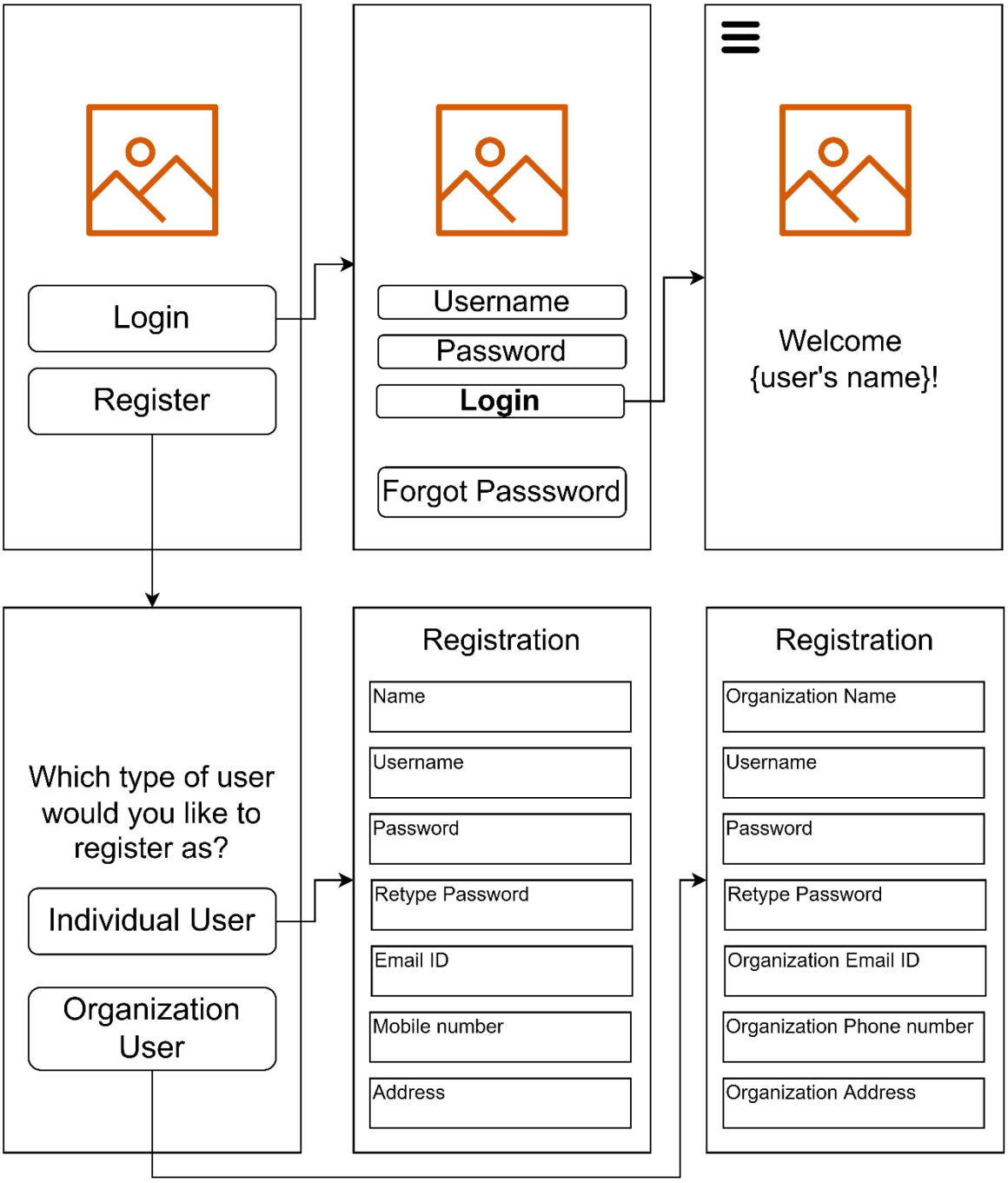

Figure 3.1: User flow diagram

The above user-flow diagram shows a subset of the the logical flow of the mini-project as per the user's actions. As there are over 15 activities used for implementing the mini project, it is not feasible to show all the activities in a single diagram.

CHAPTER 4

IMPLEMENTATION

4.1 ANDROID CONCEPTS USED

Several Android Programming concepts were used for the implementation of this mini-project. A few are listed below:

1. **Activities**
   - An Android activity is one screen of the Android app's user interface. In that way an Android activity is very similar to windows in a desktop application.
   - An Android app may contain one or more activities, meaning one or more screens. The Android app starts by showing the main activity, and from there the app may make it possible to open additional activities.
   - The implementation of the MainActivity was provided in the previous section.
2. **Views**
   - A View is a simple building block of a user interface. It is a small rectangular box that can be TextView, EditText, or even a button.
   - It occupies the area on the screen in a rectangular area and is responsible for drawing and event handling.
   - View is a superclass of all the graphical user interface components.
   - Types of Views:
     - TextView
     - EditText
     - Button
     - Image Button
     - Date Picker
     - RadioButton
     - CheckBox buttons
     - Image View

- This mini project has used several of the above Views, including but not limited to: TextView, EditText, Button and RadioButton.

3. Layouts

- Layout basically refers to the arrangement of elements on a page these elements are likely to be images, texts or styles. These are a part of Android Jetpack.
- They define the structure of android user interface in the app, like in an activity. All elements in the layout are built with the help of Views and ViewGroups.
- These layouts can have various widgets like buttons, labels, textboxes, and many others.
- Types of Layouts in Android
    - Linear Layout
    - Relative Layout
    - Constraint Layout
    - Table Layout
    - Frame Layout
    - List View
    - Grid View
    - Absolute Layout
    - WebView
    - ScrollView
- This mini project has used several of the above layouts, including but not limited to: Linear Layout, Constraint Layout, Table Layout and Scroll View.

4. Toasts

- A Toast is a feedback message. It takes a very little space for displaying while overall activity is interactive and visible to the user.
- It disappears after a few seconds. It disappears automatically. If user wants permanent visible message, notifications can be used.
- Another type of Toast is custom Toast, in which images can be used instead of a simple message.

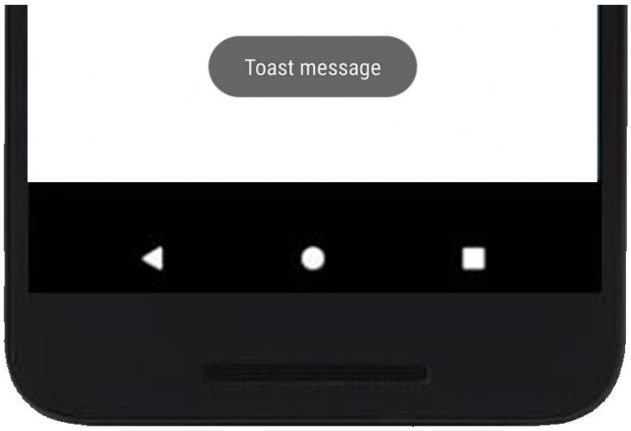

Figure 4.1: A custom Toast

- Custom toasts have been been used several times in the implementation of this mini-project. The following is the syntax of a custom toast:

```
public static Toast makeText(Context context, CharSequence
text, int duration)
```

The following method of the Toast class makes the toast containing a given text that would be displayed for the specified duration. The duration is specified using 2 constants provided in the Toast class:

Table 4.1: Duration Constants of Toast Class

Constant	Description
public static final int LENGTH_LONG	displays for long duration of time.
public static final int LENGTH_SHORT	displays for short duration of time.

The following method of the Toast class displays the custom Toast: `public void show()`. The following is an example of an implementation of custom Toast:

```
Toast.makeText(getApplicationContext(),"Hello NHCE",
Toast.LENGTH_SHORT).show();
```

5. Recycler View

- It is used to display a scrolling list of elements based on large data sets (or data that frequently changes. Ex: Gmail Inbox, Amazon Search Results).

- The RecyclerView fills itself with views provided by a layout manager. The views in the list are represented by view holder objects
- These objects are instances of a class you define by extending RecyclerView.ViewHolder
- You supply the data and define how each item looks, and the RecyclerView library dynamically creates the elements when they're needed.

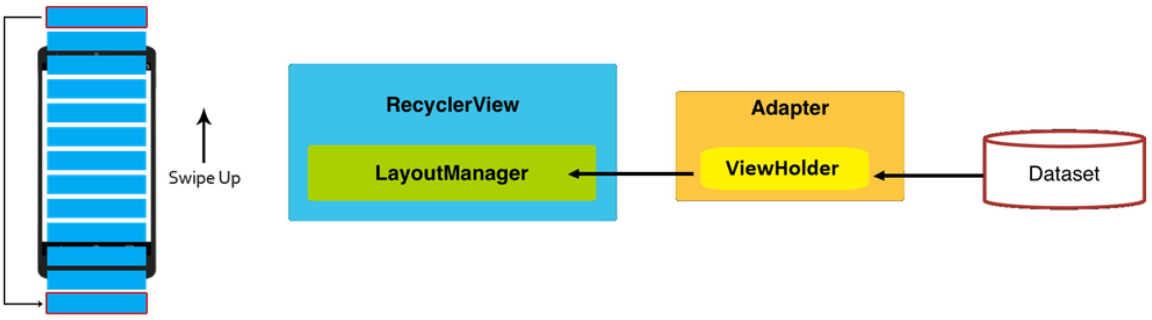

Figure 4.2: Recycler View

- As the name implies, RecyclerView *recycles* those individual elements. When an item scrolls off the screen, RecyclerView doesn't destroy its view.
- The RecyclerView has been used in the mini-project to display the list of near-by emergency donors, blood banks and plasma banks.

6. Adapters

- In Android, an Adapter is a bridge between UI component and data source that helps us to fill data in UI component.
- It holds the data and send the data to an Adapter view then view can takes the data from the adapter view and shows the data on different views like as ListView, GridView, Spinner etc.
- To fill data in a list or a grid we need to implement Adapter. Adapters acts like a bridge between UI component and data source.
- Here data source is the source from where we get the data and UI components are list or grid items in which we want to display that data.

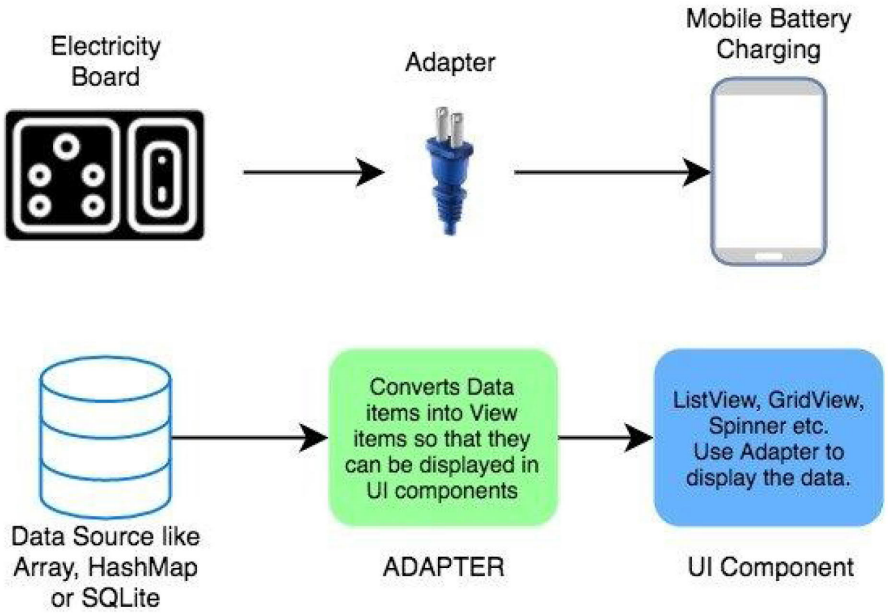

Figure 4.3: Adapter

7. Intents

- An intent is used to perform an action on the screen.
- It is mostly used to start activity, send broadcast receiver,start services and send message between two activities.
- There are two intents available in android as Implicit Intents and Explicit Intents.
- Implicit intents are used for inter-application communication whereas explicit intents are used for intra-application communication.
- Intents can also pass information between two activities by using methods like putExtra(). The following is a code snippet for an explicit intent started in Activity1 to transfer the flow to Activity 2.

```
Intent i = new Intent(Main_Activity.this, Activity2.class);
i.putExtra("uname", username);
i.putExtra("email", emailid);
startActivity(i);
```

- To retrieve the information sent along with the intent, we have to use the following code snippet which should be placed inside the onCreate() function of Activity2.

```
Intent thisActivity = getIntent();
String userName = thisActivity.getStringExtra("uname");
String emailID = thisActivity.getStringExtra("email");
```

- Intents have been widely used for navigation control purposes, i.e., for transferring the user from one activity to another when a button is clicked.

8. Async Tasks

- An AsyncTask (Asynchronous Task) allows us to run the instruction in the background and then synchronize again with our main thread.
- AsyncTask must be subclassed to be used. The subclass will override at least one method doInBackground(), and most often will override onPostExecute().
- AsyncTask class is used to do background operations that will update the UI.
- Async Tasks have been used in the mini-project for the purpose of sending emails.
- Used mainly for short operations. The following diagram illustrates the workflow of an Async Task.

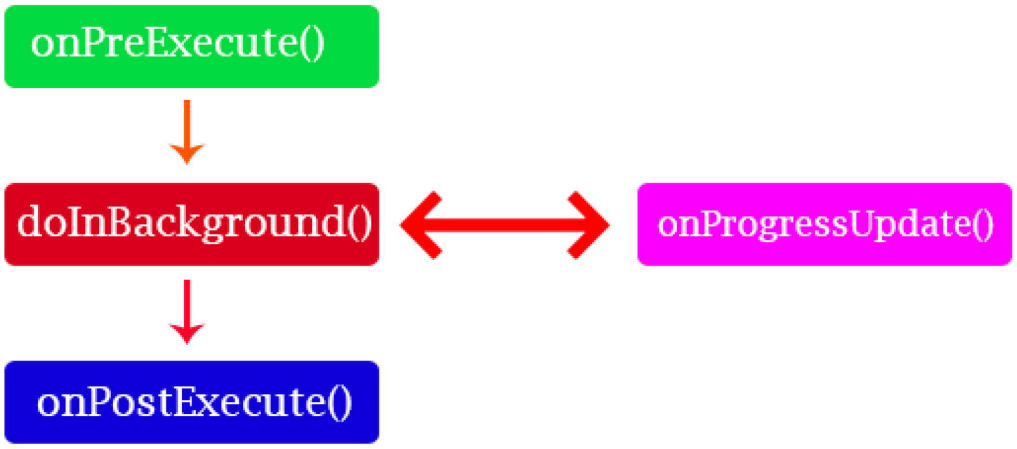

Figure 4.4: Async Task

4.2 FUNCTIONALITY

The mini-project aims to provide a platform for individual users like you and me to form a community of voluntary emergency donors, blood banks and plasma banks. To implement this, two types of user categories were created:

Table 4.2: Types of Users

User Type	Description
Individual user	Users who are independent and not related to any organization
Organization user	Users who represent organizations like Hospitals, Blood Banks and Plasma Banks

Hence the information collected for these two types of users would be different. The following table shows the various data collected from each type of user:

Table 4.3: Information collected from the Users

Individual User	Organizational User
• Name	• Name
• Username	• Username
• Password	• Password
• Email address	• Email address
• Phone number	• Phone number
• Address	• Address
• City	• City ○ Bangalore ○ Chennai ○ Kochi ○ Mysore ○ Mumbai ○ New Delhi ○ Kolkata
• Blood Group ○ A+ ○ B+ ○ O+ ○ AB+ ○ A- ○ B- ○ O- ○ AB-	
• Gender ○ Male ○ Female	• Type of Organization ○ Hospital ○ Blood Bank ○ Plasma Bank
• Whether user would like to register themselves as an Emergency Donor ○ Yes ○ No	

Both categories of users are verified by sending a One Time Password (OTP) to the email address provided by them. This ensures that only genuine users are registered with verifiable email IDs.

Upon logging in, an 'Individual' type user would see the landing page with a welcome message with their name. A side navigation bar is used for listing the various activities that they can do, which include:

Table 4.4: Individual User Features

Feature	Description
Edit Profile	To update the user's contact information. However, the user would not be able to change their username and Email ID.
Find Near-by Blood banks	Here, the users would be able to lookup organizations with the user's city that provide a certain blood group given by the user or other compatible blood groups. The contact information about the organization is displayed using a Recycler View. Information regarding other blood groups that are compatible with the blood group specified by the user is also displayed.
Find Near-by Plasma banks	This feature is similar to the above feature but here the user can lookup organizations that provide a specified plasma group or compatible plasma groups and find their contact information.
Find Near-by Blood donors	Here, the users would be able to lookup the information of users within the Life360+ community who reside within the city of the user and have voluntarily signed up as emergency blood donors to service an emergency request. The information of users having the same or compatible blood groups are displayed.
Logout	The user can logout of their session.

Upon logging in, an 'Organization' type user would see the landing page with a welcome message with their name. A side navigation bar is used for listing the various activities that they can do, which include:

Table 4.5: Organization User Features

Feature	Description
Edit Profile	To update the user's contact information. However, the user would not be able to change their username and Email ID.
Edit Blood Inventory	This option is only available to users who represent organizations that are either Hospitals or Blood Banks. Here, the organization can update the quantity of blood belonging to various blood groups. The information entered
Edit Plasma Inventory	This option is only available to users who represent organizations that are either Hospitals or Plasma Banks. Here, the organization can update the quantity of plasma belonging to various plasma groups. The information entered here is updated in real-time so that 'Individual' users can look up this information.
Logout	The user can logout of their session.

Additionally, an OTP mechanism is used in 2 scenarios. The first one being when the user is registering themselves. An OTP is used for confirming the veracity of the user. The second one being, if a user forgets their password, an option to reset their password securely using an OTP mechanism is available in the Login activity. The user enters their username. The application checks if the username exists. If it exists, an email containing the OTP to reset the password is sent to the registered Email ID. If the OTP, matches the user would be able to create a new password for their account.

4.3 FIREBASE CONNECTIVITY

Firebase is a Backend-as-a-Service, and it is a real-time database which is basically designed for mobile applications. It can be used for providing database connectivity for several platforms like Android, iOS, Web, or Unity.

Firebase evolved from Envolve. Envolve is a prior startup founded by James Tamplin and Andrew Lee in 2011. Envolve provided developers an API which allowed the integration of online chat functionality into their websites. After releasing the chat service, it found that the Envlove was being used to pass application data, which were not chat messages. Developers used Envolve to sync application to separate the real-time architecture and the chat system which powered it. In September 2011, Tamplin and Lee founded firebase as a separate company. It was finally launched to the public in April 2012.

Firebase Real-time Database was the first product of firebase. It is an API which syncs application data across Android, iOS, and Web devices. It gets stored on Firebase's cloud. The firebase real-time database helps developers to build real-time, collaborative applications.

To store data into Firebase, we first have to add the Firebase into our android application. Before adding Firebase into our android application, we need to be aware of some prerequisites.

These requisites are as follows:

- It is required that we should have a Google account to work with Firebase.
- We also require a device or emulator, which is running on Android 4.0 or later.
- We also require the latest version of Android Studio, i.e., version 1.5 or higher.
- The app must use Gradle 4.1 or later.

There are two methods through which we can add Firebase to our application:

- One is through the Firebase assistant, which is an easy-to-use wizard inside Android Studio. It will connect our existing project or create a new one for us and automatically install any necessary Gradle dependencies. It requires some additional configuration.
- The second option is to manually add Firebase to our Android application via Firebase console. It is the recommended way of adding Firebase to our application.

This mini-project has used Firebase's Real Time Database to provide database connectivity. All Firebase Realtime Database data is stored as JSON objects. We can think of the database as a cloud-hosted JSON tree. Unlike a SQL database, there are no tables or records. When we add data to the JSON tree, it becomes a node in the existing JSON structure with an associated key. Here, we use the username as the key for each node in the "users" tree.

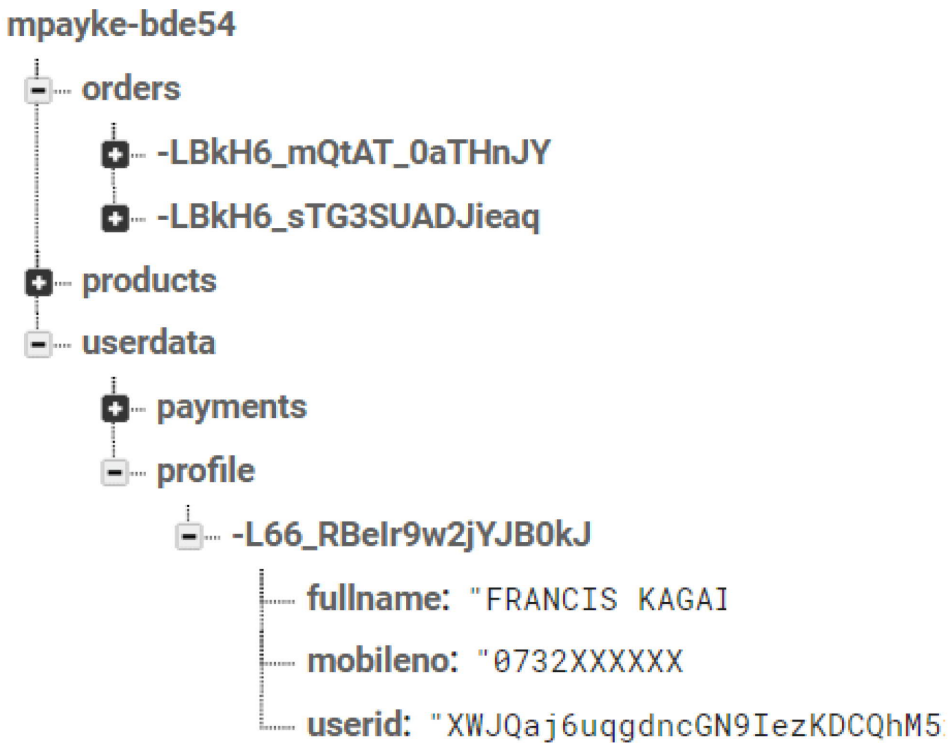

Figure 4.5: Structure of Data in Firebase Realtime Database

For storing the data of an 'Organization' type user, we use the following JSON structure:

```
{
"users" : {
    "username" : {
            "address" : "",
            "bg" : "",
            "city" : "",
            "emailid" : "",
```

```
            "id" : "",
            "mobile" : "",
            "name" : "",
            "orgType" : "",
            "password" : "",
            "userType" : "",
            "username" : ""
    }
}
```

For storing the data of an 'Individual' type user, we use the following JSON structure:

```
{
"users" : {
    "username" : {
            "address" : "",
            "bg" : "",
            "city" : "",
            "eDonor" : "",
            "emailid" : "",
            "gender" : "",
            "id" : "",
            "mobile" : "",
            "name" : "",
            "password" : "",
            "userType" : "",
            "username" : ""
    }
}
```

CHAPTER 5

SAMPLE OUTPUT

5.1 LAUNCH AND TYPE OF REGISTRATION SELECTION ACTIVITY

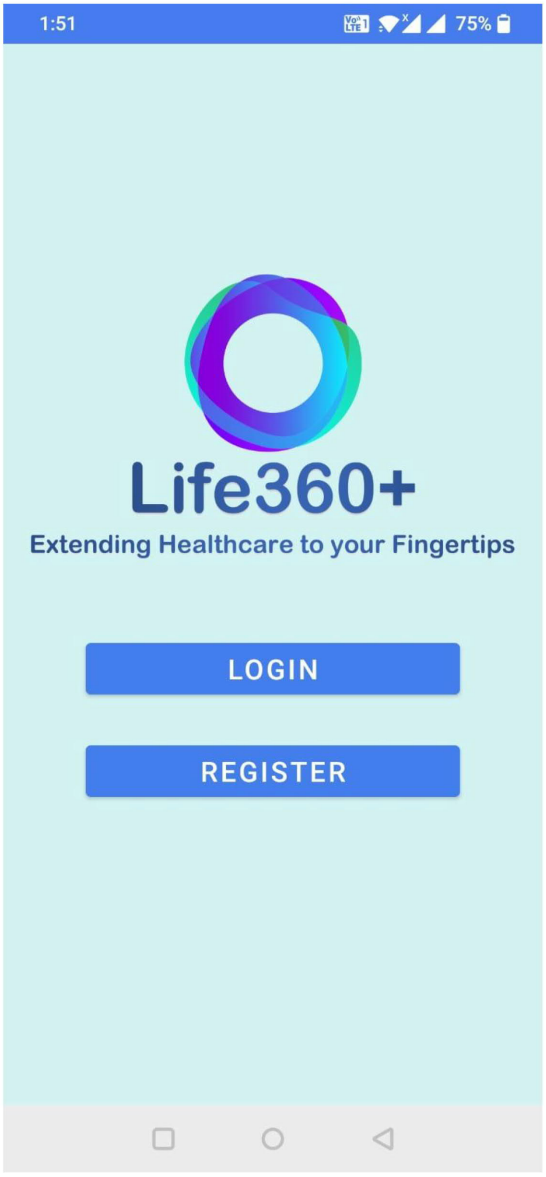

Figure 5.1: Launch Screen

Figure 5.2: Type of User Registration Selection Activity

The above two figures show the Launch screen, which is the Main Activity and the Type of User Registration Selection activity.

5.2 INDIVIDUAL USER REGISTRATION ACTIVITY

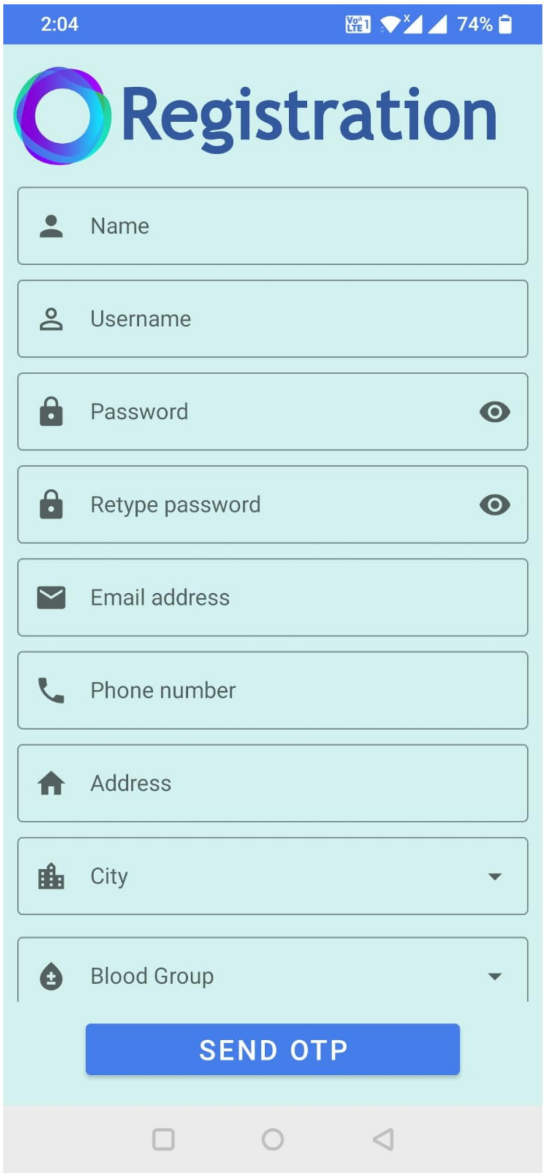

Figure 5.3: Individual User Registration Activity - Empty Form

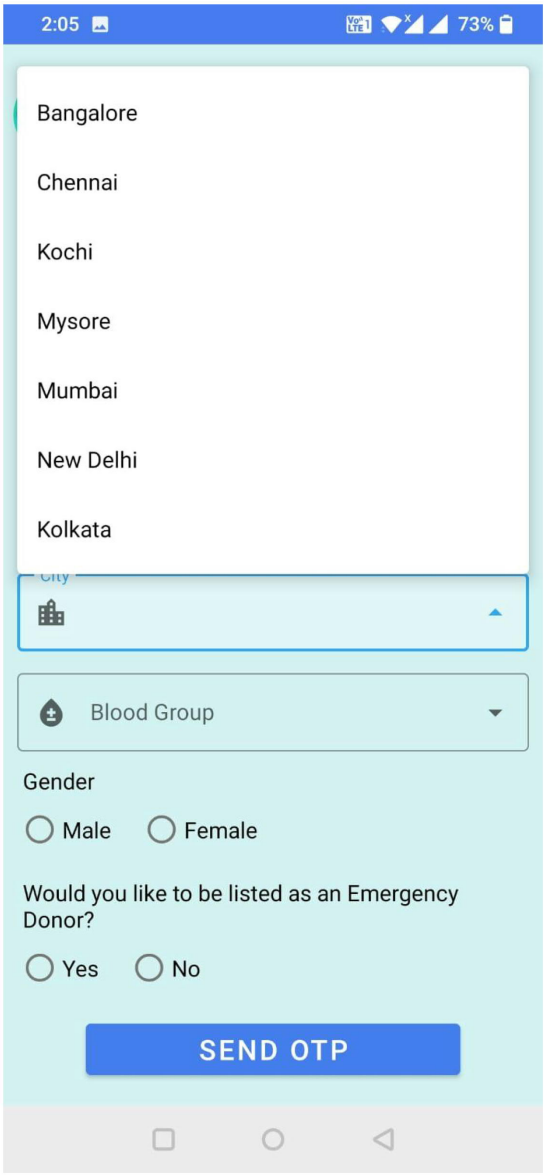

Figure 5.4: Individual User Registration Activity - City Dropdown

The above two figures show the unfilled 'Individual User' registration activity. All the fields have been placed in a Scroll View so that the user can access all the fields. Once the user clicks on 'Send OTP' button, the app verifies the data entered. If errors are found, an appropriate Toast is raised.

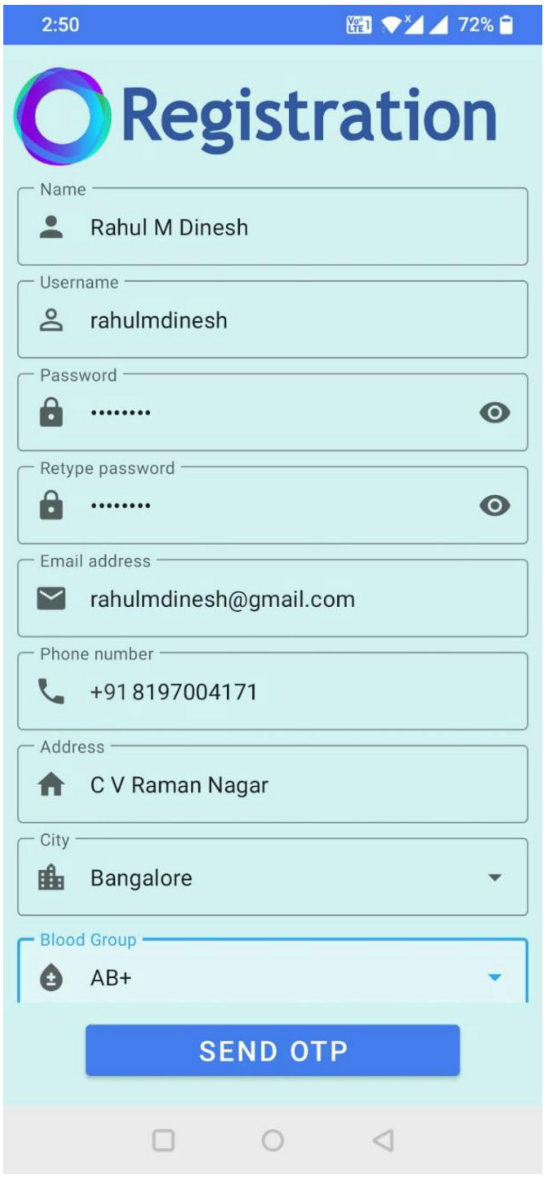

Figure 5.5: Individual User Registration Activity - Filled Form

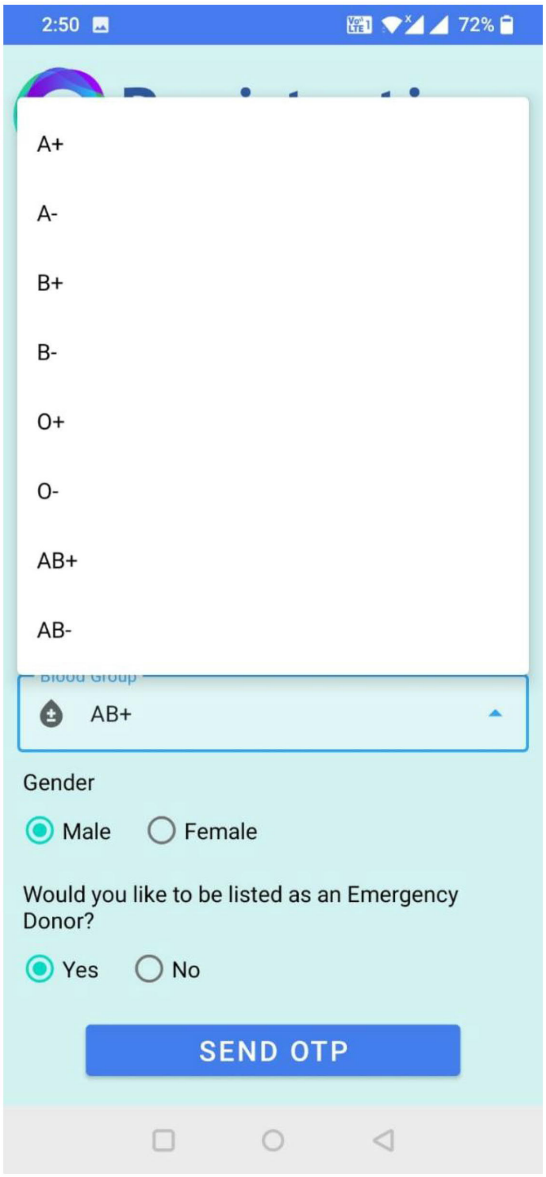

Figure 5.6: Individual User Registration Activity - Blood Group Dropdown

The above two figures show the filled 'Individual User' registration activity. If the username is already taken or if the passwords do not match or if any field is empty then appropriate Toast messages are displayed. If no such error exists, an OTP is sent to the entered Email ID to verify the user's identity and all the fields are frozen so that user will not be able to make any changes to the entered data.

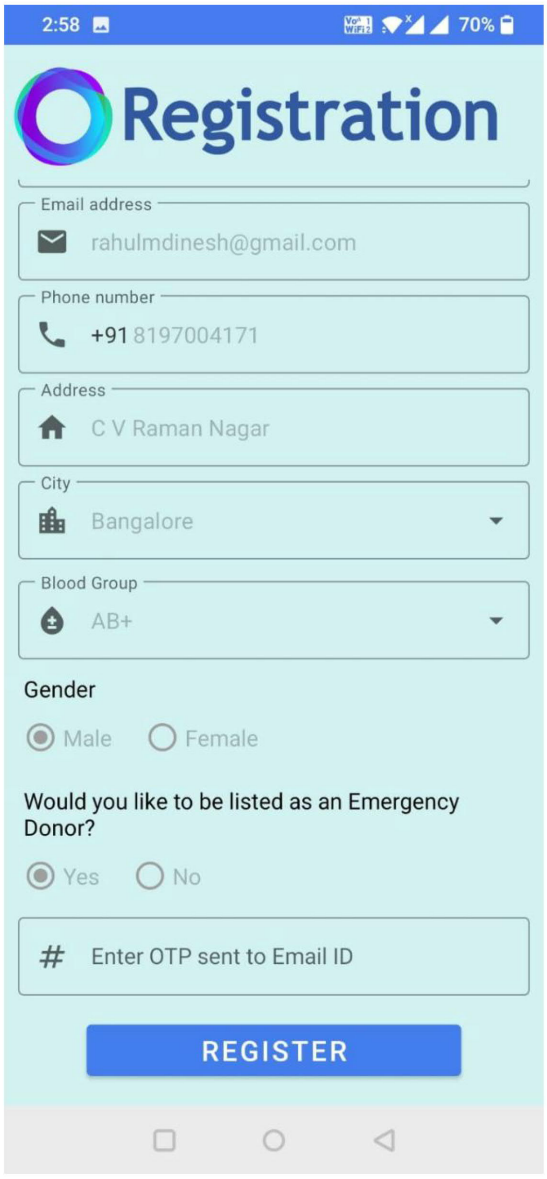

Figure 5.7: Individual User Registration Activity - Fields Frozen

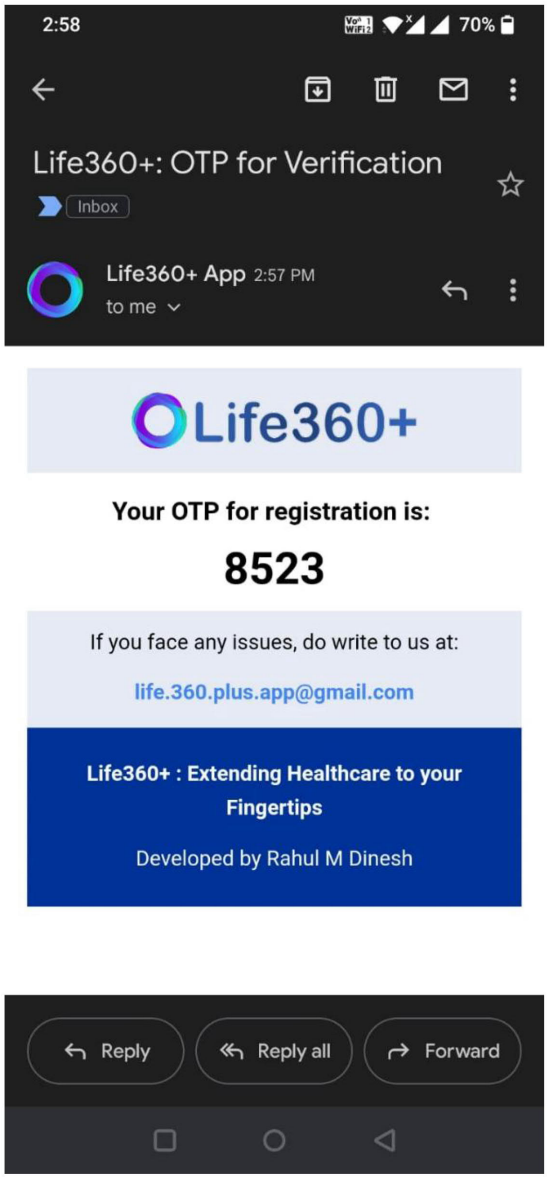

Figure 5.8: OTP sent to Email ID

The above two figures show the filled fields of the 'Individual User' registration activity being frozen after the OTP is sent to the given Email ID. The email sent is also shown which contains the OTP to confirm the user's identity. Upon entering the correct OTP, the user would be successfully registered and a welcome mail would also be sent to the user's email containing the information provided by them. If an incorrect OTP is entered, an appropriate Toast is made to alert the user.

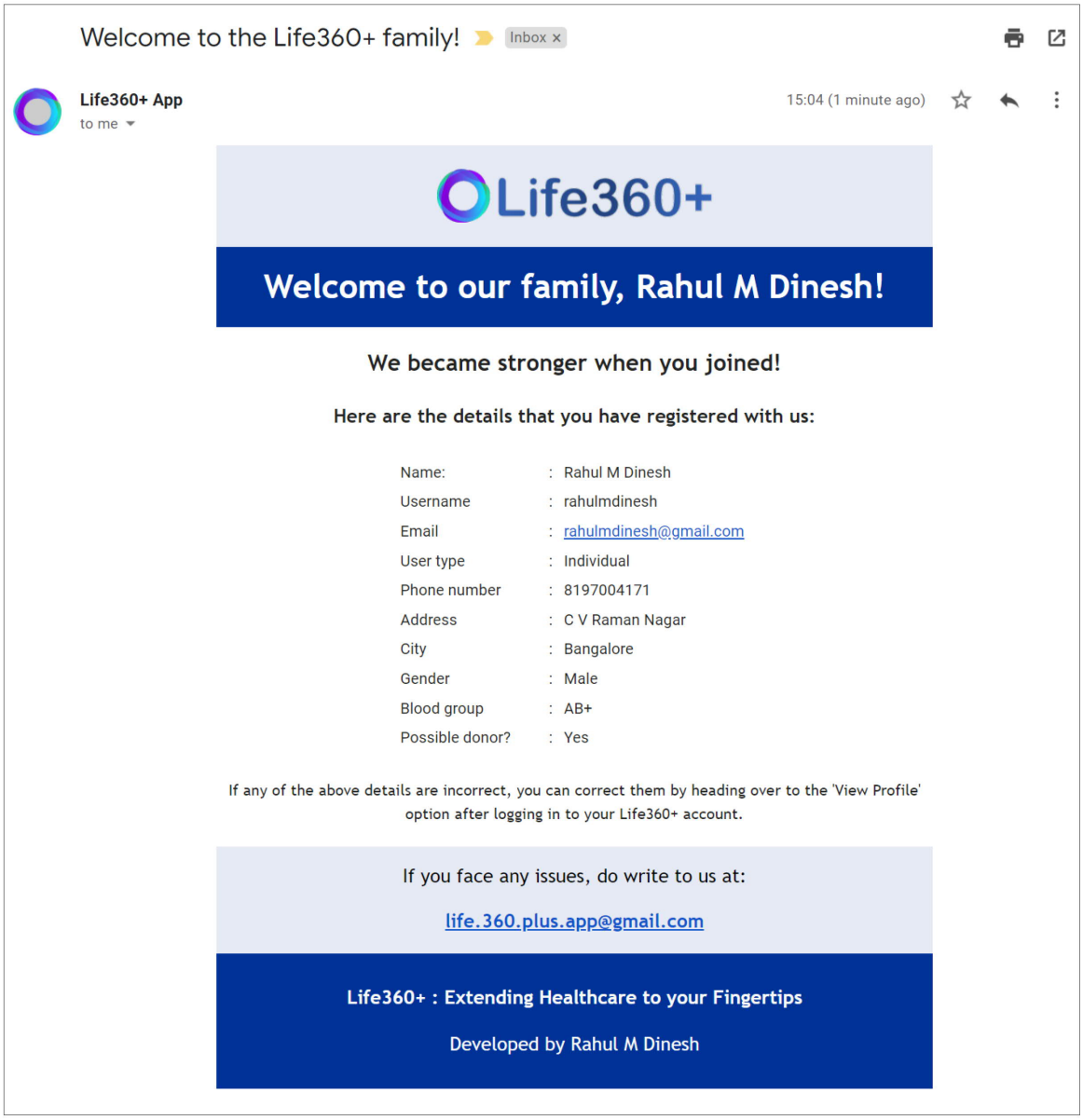

Figure 5.9: Welcome Email Sent to the Individual User

5.3 ORGANIZATION USER REGISTRATION ACTIVITY

The 'Organization' type users are also registered in a similar manner. The only difference with the 'Individual' type users' registration is that, the data required from the user is different. Also a slightly different welcome mail is sent after the OTP is verified.

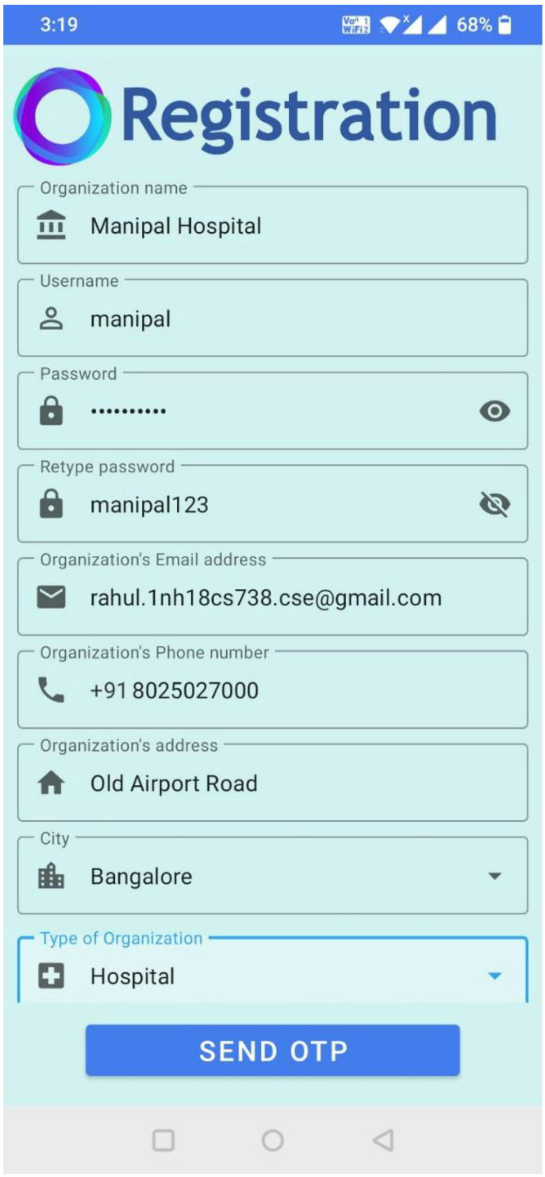

Figure 5.10: Organization User Registration Activity

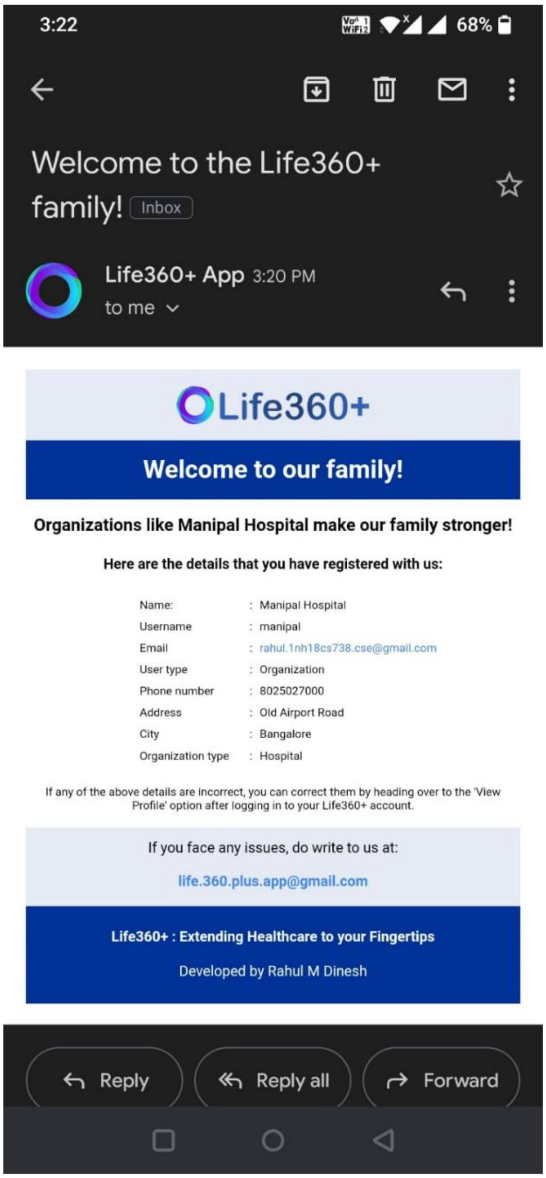

Figure 5.11: Welcome Email sent to the Organization type User

5.4 FORGOT PASSWORD ACTIVITY

An option to reset the user's password in the scenario that they forgot it, is provided in the Login Activity. This button transfers the user to the Forgot Password activity, where the username is asked. Once it is verified that the user exists, then an OTP is sent to the user's registered email. If the OTP matches, the user will be able to reset their password.

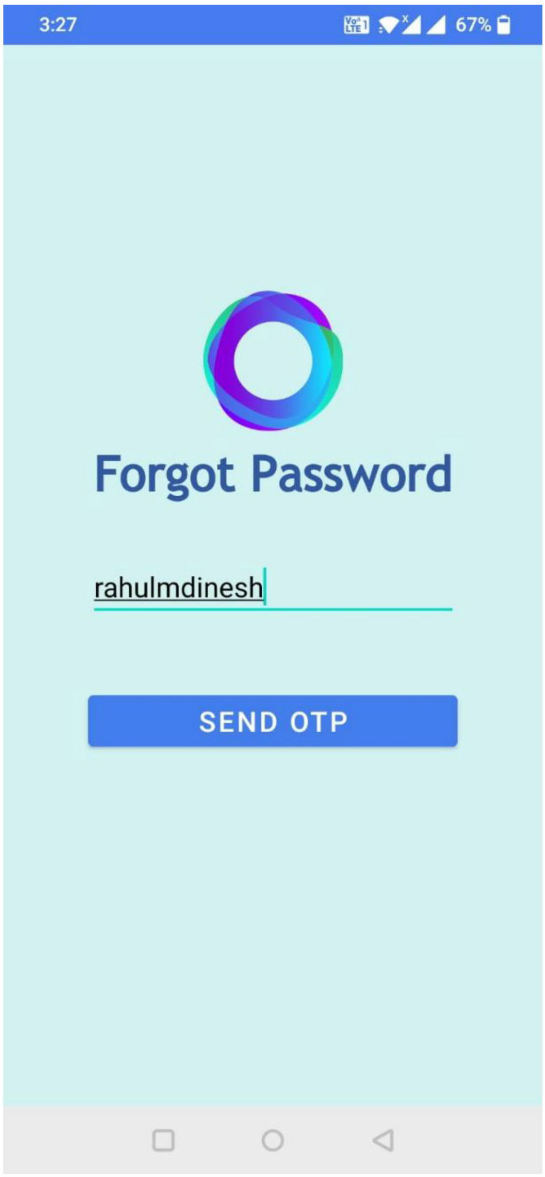

Figure 5.12: Forgot Password Activity - Entering Username

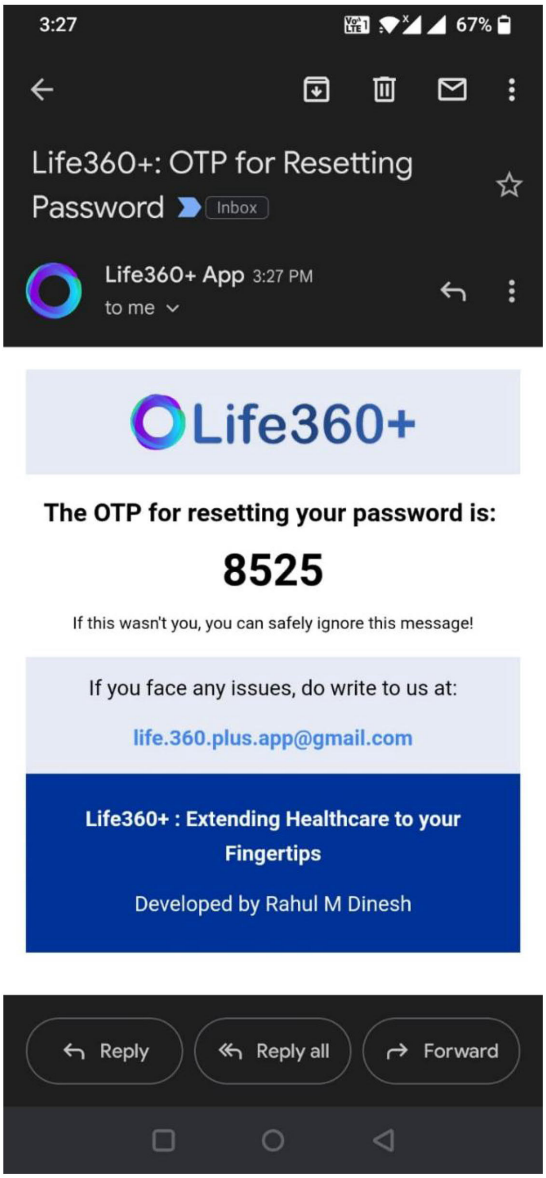

Figure 5.13: Email Sent to the user's registered Email ID containing OTP

If the OTP entered by the user does not match the OTP generated by the app, then an appropriate Toast is made to alert the users that the OTP does not match. Once, the OTP matches, the user is provided with two fields to enter and re-enter their password. If the entered and re-entered passwords do not match, an appropriate Toast is made to alert the user that the passwords do not match. After the password is reset, the user is redirected back to the Login activity.

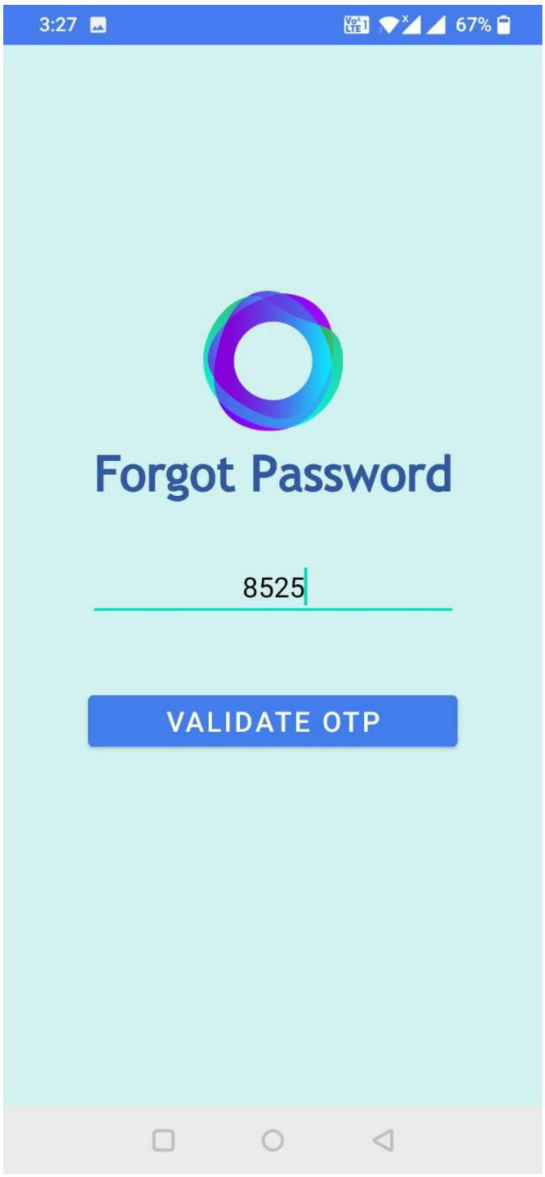

Figure 5.14: Forgot Password Activity - Entering OTP

Figure 5.15: Forgot Password Activity - Entering a new password

5.5 LOGIN ACTIVITY

After registering, the user can login in to their account using the Login activity. Here, the users' username and password are required. If there exists an entry in the database with that matches the credentials entered by the user, the user is allowed to access to their account and is redirected to the user landing activity. Else, a toast is raised to alert the user alerting them that the credentials are invalid.

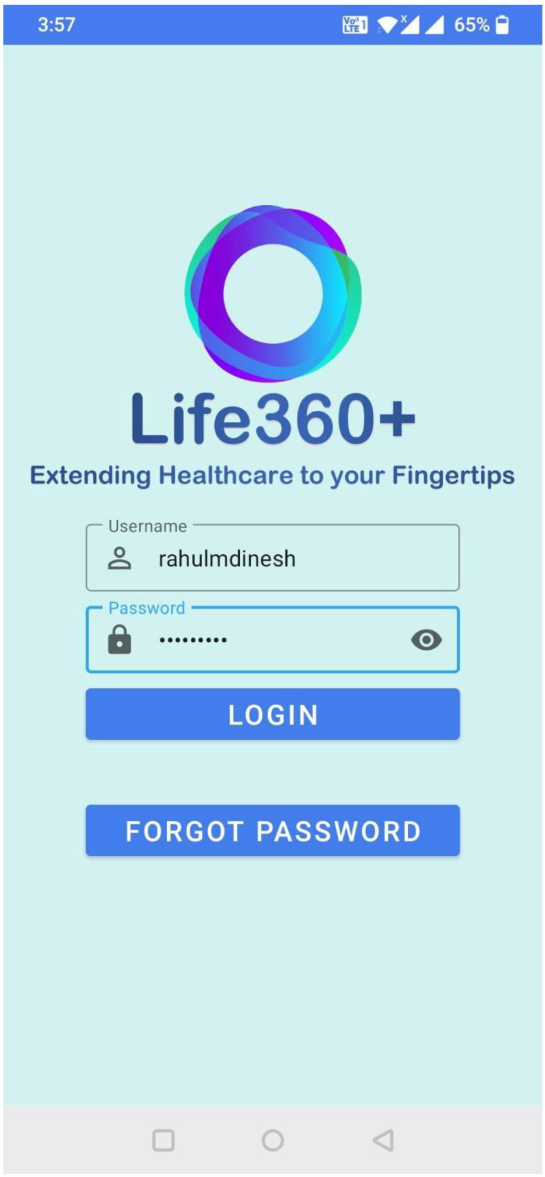

Figure 5.16: Login Activity

Figure 5.17: Toast raised for invalid credentials in the Login Activity

5.6 INDIVIDUAL USER - LANDING ACTIVITY

Once an 'Individual' type user successfully logins-in, they would be redirected to the Individual user landing activity. This activity can be considered as the 'Home' page. It contains a welcome page, with a side navigation bar containing the various features that the user can use.

Figure 5.18: Individual User: Landing Activity's welcome message

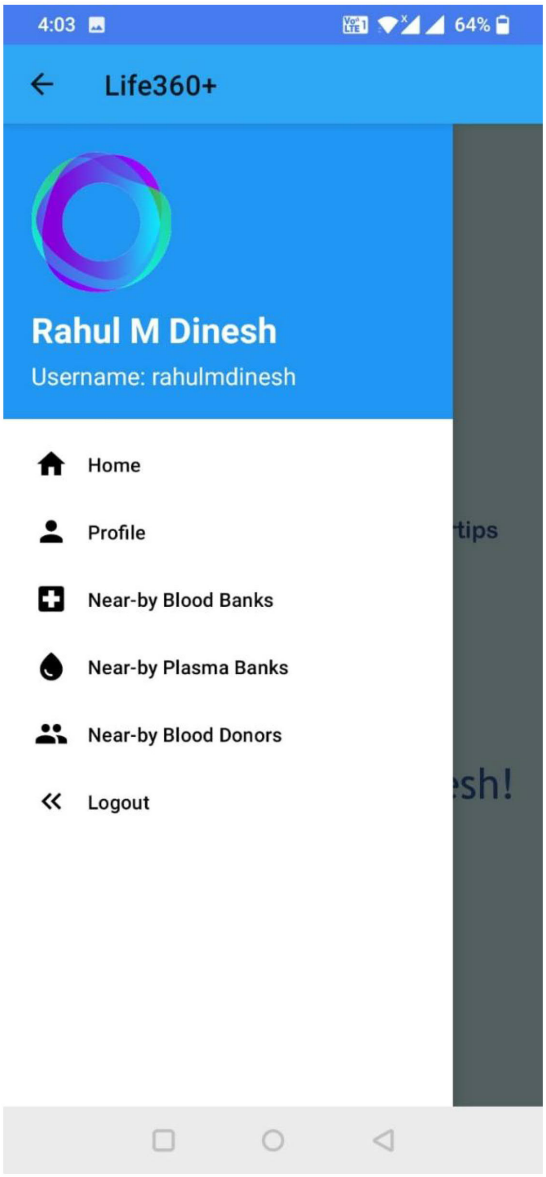

Figure 5.19: Individual User: Side Navigation Bar

The side navigation bar displays the name of the user currently logged in and the user's username. It then lists out the activities that the user can do like updating their profile, viewing near-by blood banks, viewing near-by plasma banks, viewing near-by blood donors who have volunteered for emergency blood donation requests as well as a feature to log out from the current session.

5.7 INDIVIDUAL USER - UPDATING PROFILE

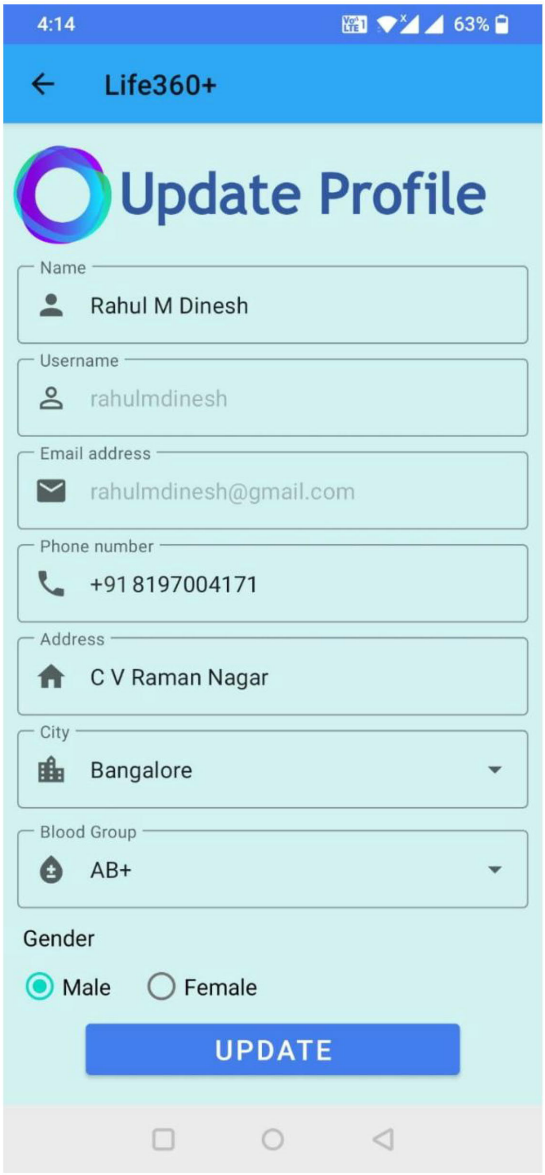

Figure 5.20: Individual User: Before Updating Profile

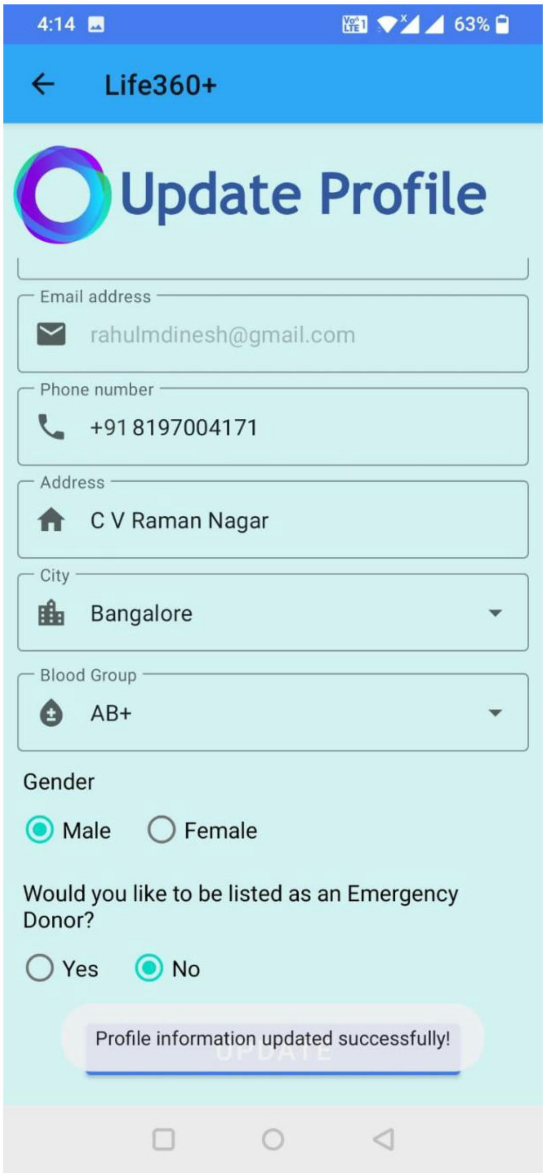

Figure 5.21: Individual User: After Updating Profile

The user can update any profile information except the username and email ID. This is seen by the greying out of the username and password fields. Users typically would use this feature to toggle the 'Emergency Donor' radio button.

5.8 INDIVIDUAL USER - VIEWING NEAR-BY BLOOD BANKS

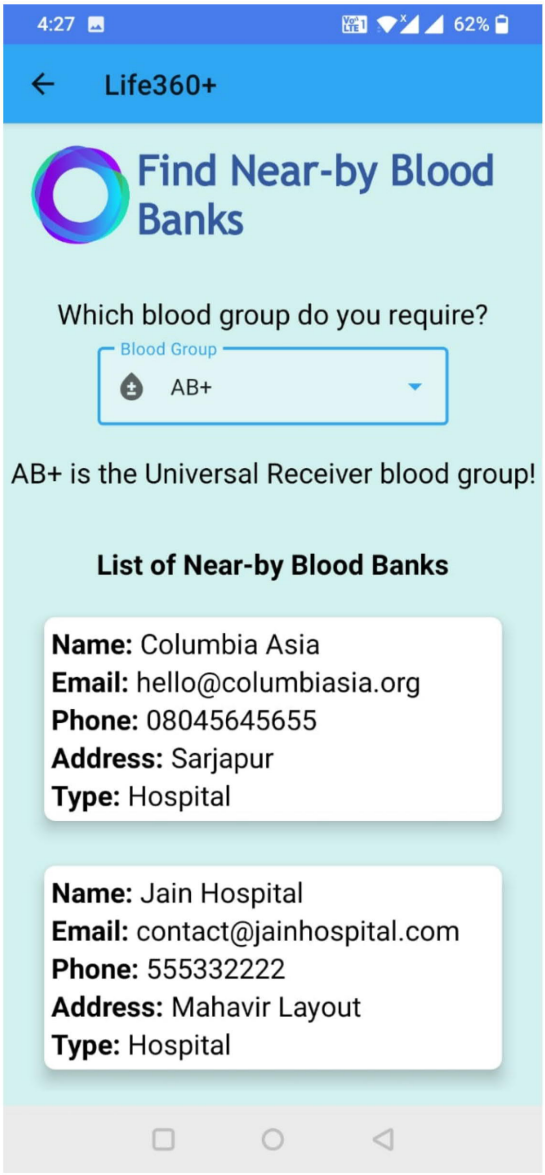

Figure 5.22: Individual User: Finding Near-by Blood Banks - 1

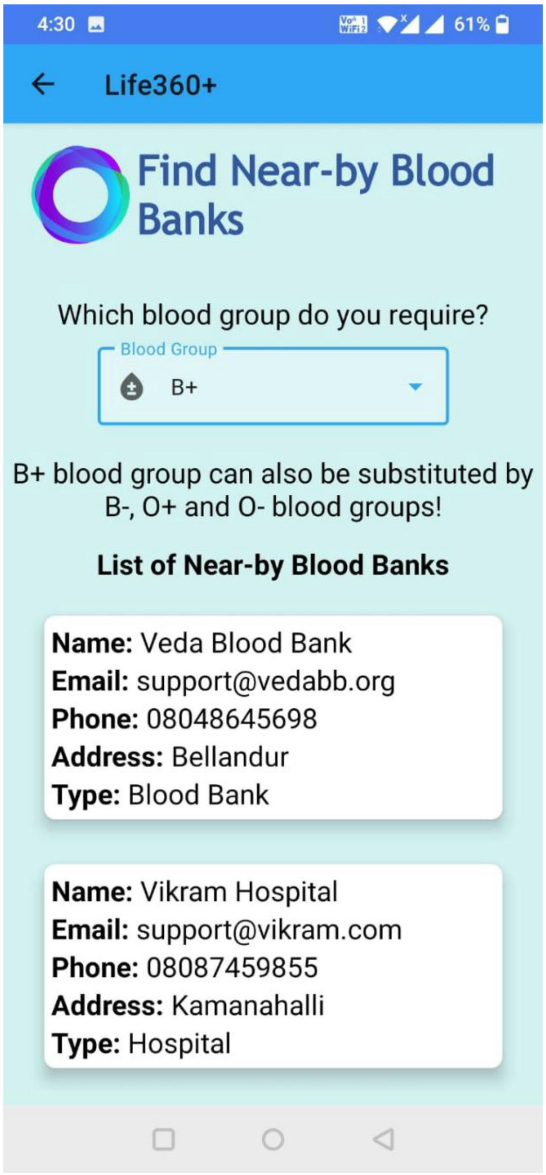

Figure 5.23: Individual User: Finding Near-by Blood Banks - 2

The user can look-up the hospitals and blood banks in user's city that have stock of the specified blood group given by the user. The user taps the drop down and selects the blood group of their requirement. The Recycler View is immediately populated with the information of the matching hospitals and blood banks.

5.9 INDIVIDUAL USER - VIEWING NEAR-BY PLASMA BANKS

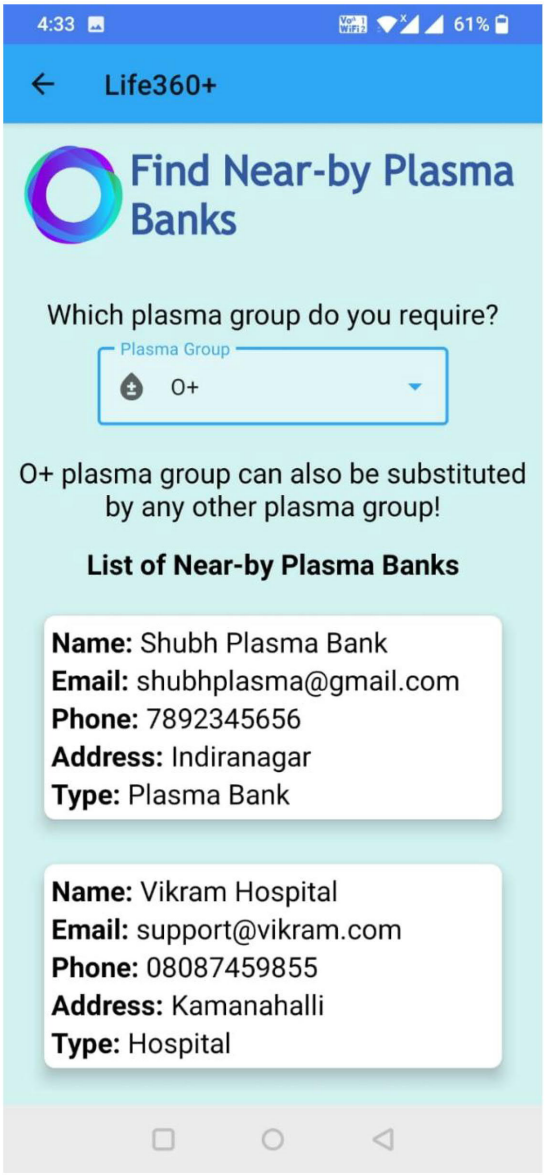

Figure 5.24: Individual User: Finding Near-by Plasma Banks - 1

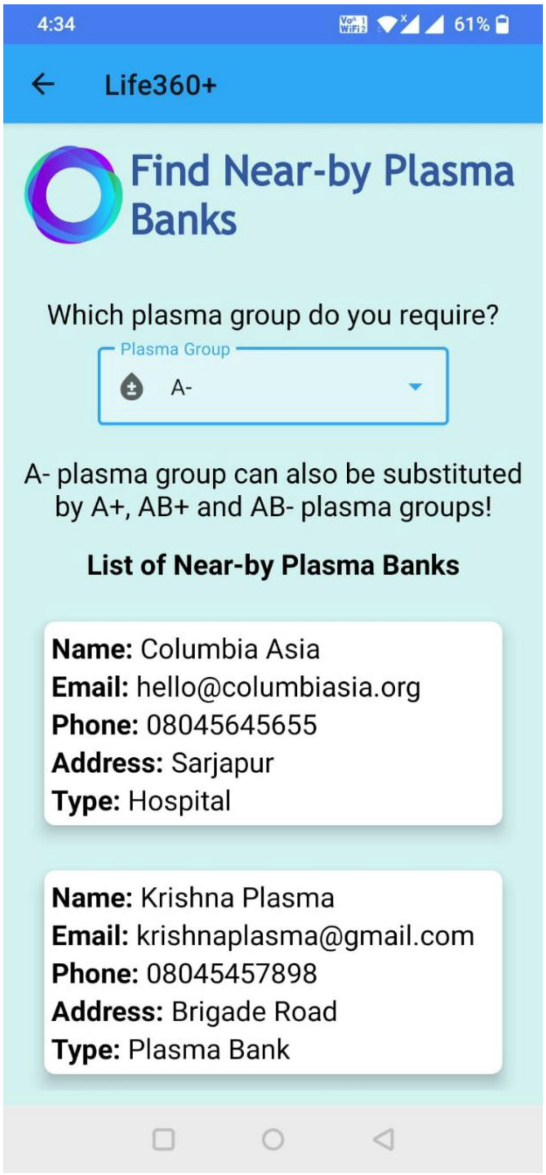

Figure 5.25: Individual User: Finding Near-by Plasma Banks - 2

The user can look-up the hospitals and plasma banks in user's city that have stock of the specified plasma group given by the user. The user taps the drop down and selects the plasma group of their requirement. The Recycler View is immediately populated with the information of the matching hospitals and plasma banks.

5.10 INDIVIDUAL USER - VIEWING NEAR-BY BLOOD DONORS

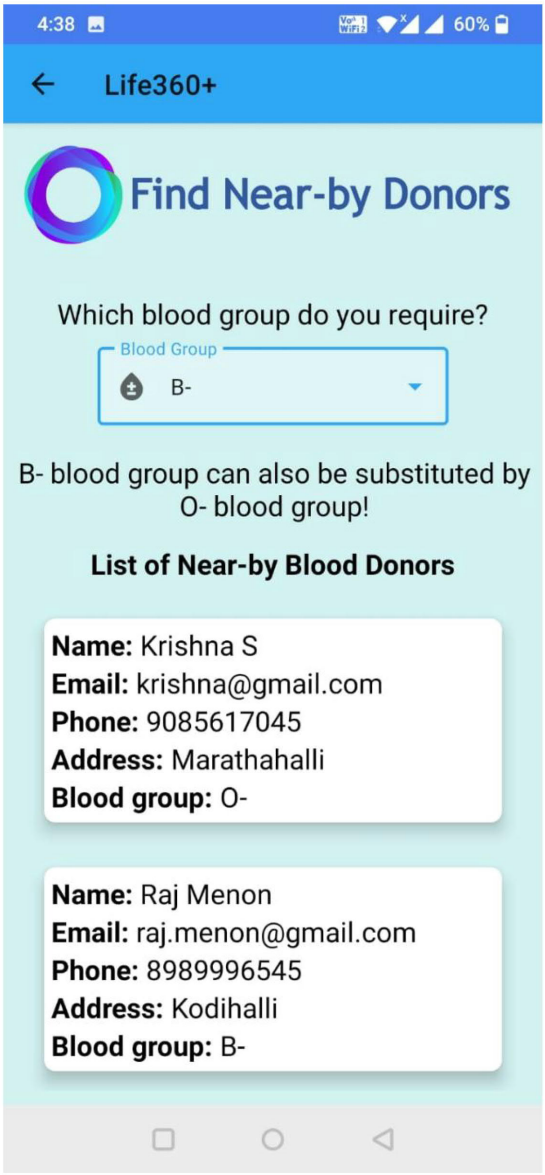

Figure 5.26: Individual User: Finding Near-by Blood Donors - 1

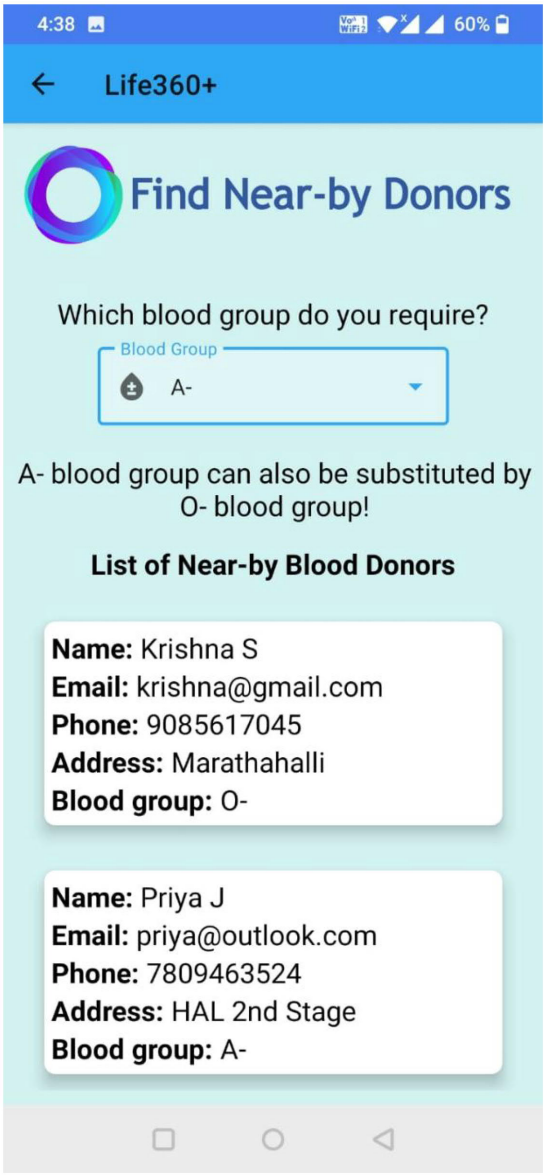

Figure 5.27: Individual User: Finding Near-by Blood Donors - 2

The user can look-up the emergency blood donors in user's city whose blood group matches or is compatible with the specified blood group given by the user. The user taps the drop down and selects the blood group of their requirement. The Recycler View is immediately populated with the information of the matching donors.

5.11 ORGANIZATION USER - LANDING ACTIVITY

Figure 5.28: Organization User: Landing Activity's welcome message

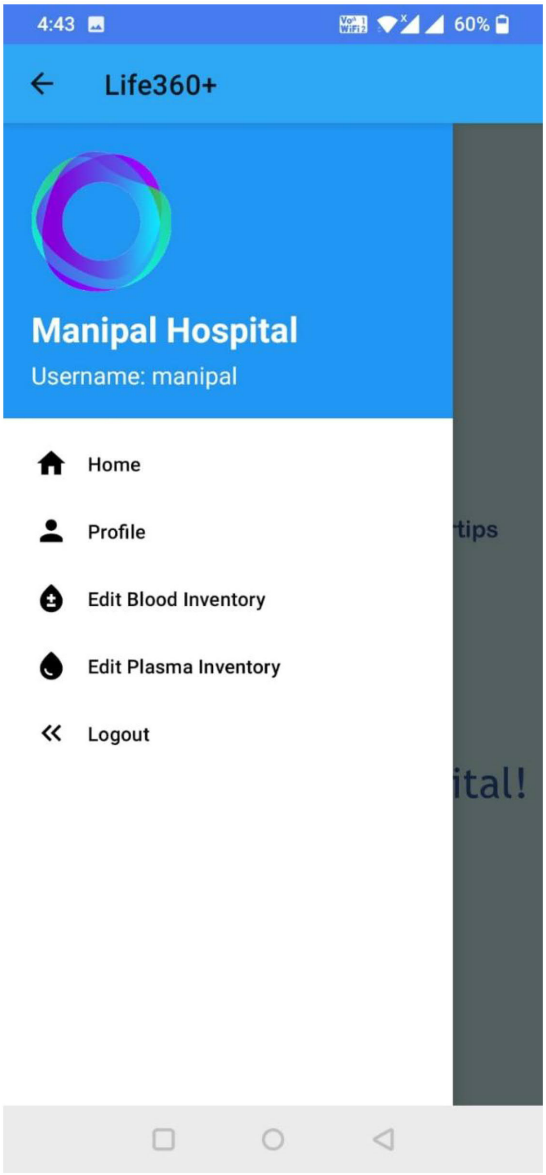

Figure 5.29: Organization User: Side Navigation Bar

Once an 'Organization' type user successfully logins-in, they would be redirected to the Organization user landing activity. This activity can be considered as the 'Home' page. It contains a welcome page, with a side navigation bar containing the various features that the user can use.

5.12 ORGANIZATION USER - UPDATING PROFILE

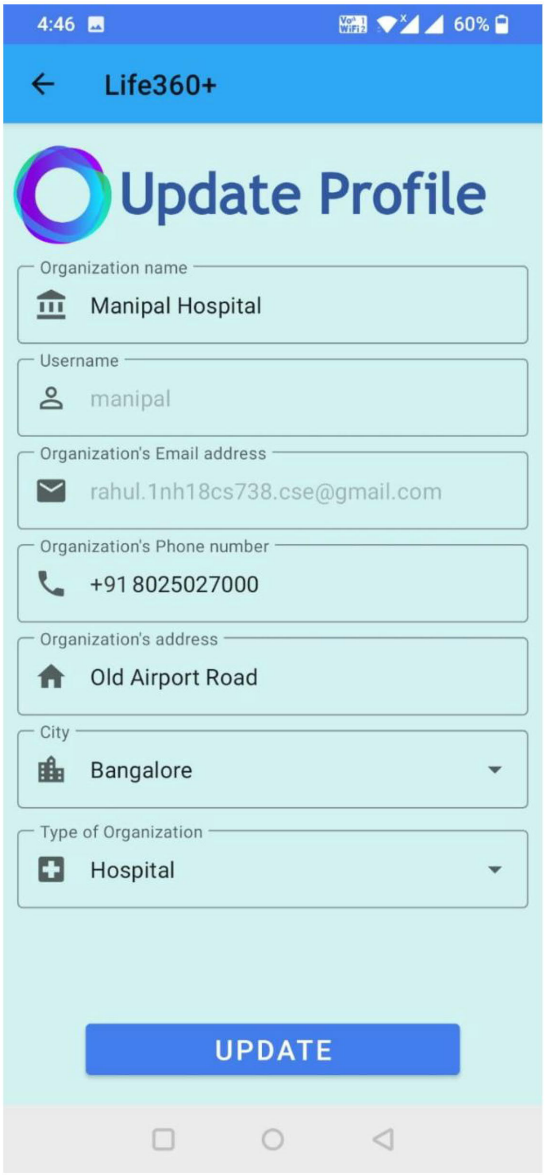

Figure 5.30: Organization User: Before Updating Profile

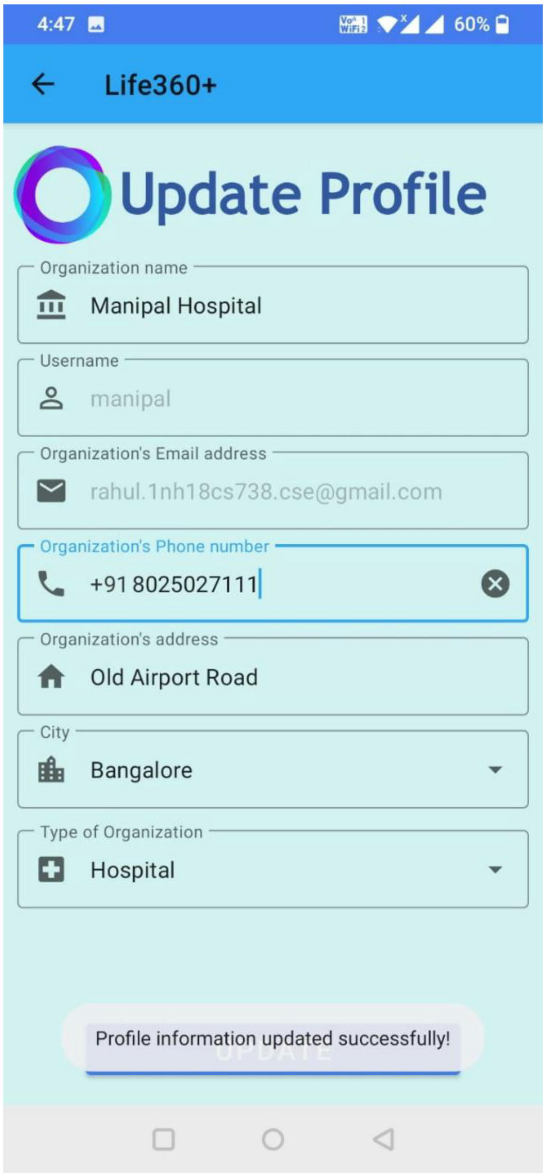

Figure 5.31: Organization User: After Updating Profile

The Organization type user can update any profile information except the username and email ID. This is seen by the greying out of the username and password fields. Users typically would use this feature to update the phone number which would be used by the Individual users to contact the organization.

5.13 ORGANIZATION USER - UPDATING BLOOD INVENTORY

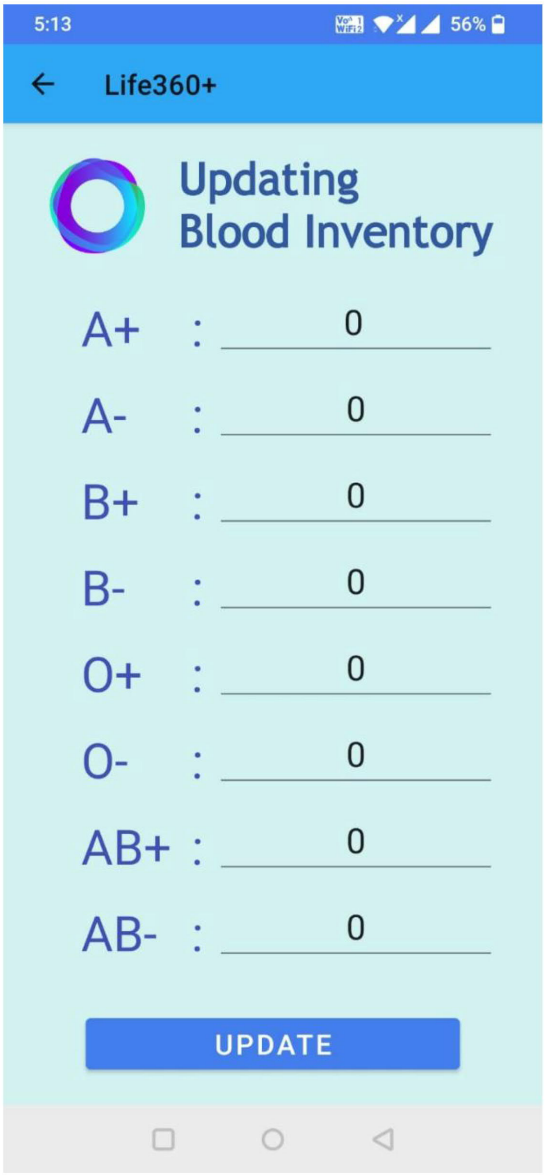

Figure 5.32: Organization User: Blood Inventory Activity

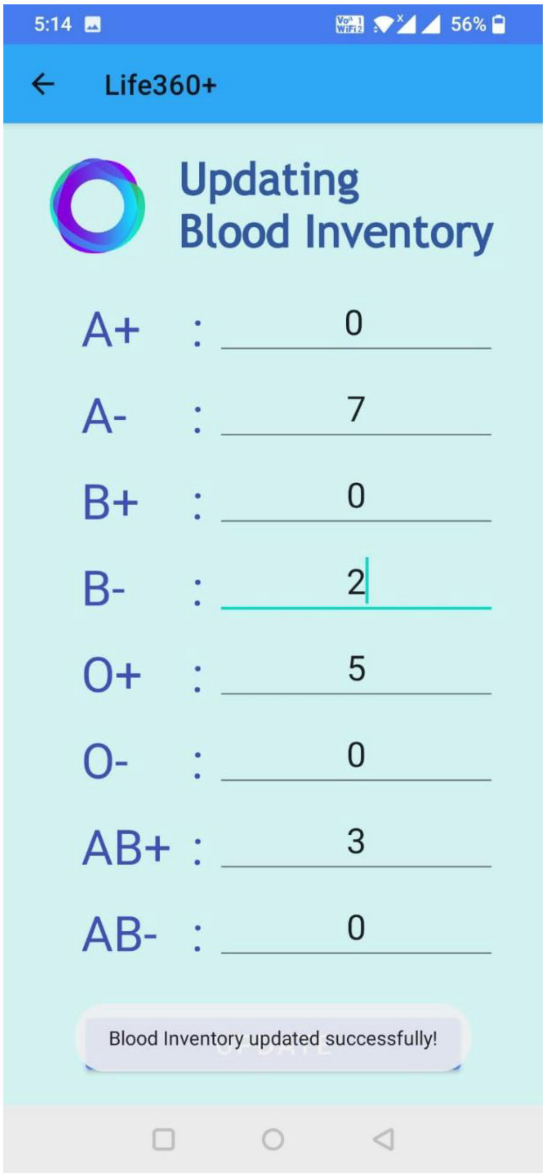

Figure 5.33: Organization User: Updating Blood Inventory

The Organization type user can update their organization's plasma inventory. This feature is only available to organizations that are either Hospitals or Blood Banks. Once an organization (hospital/blood bank) is created, a Blood Inventory entry is also made with initial values of all quantities set to zero. Later, the user can update them, as shown above.

5.14 ORGANIZATION USER - UPDATING PLASMA INVENTORY

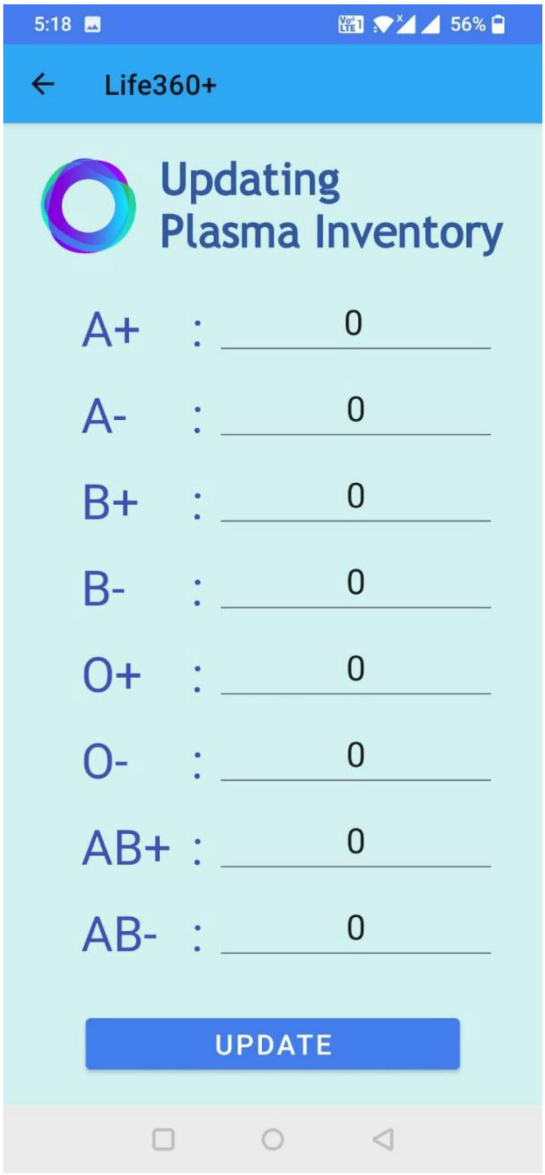

Figure 5.34: Organization User: Plasma Inventory Activity

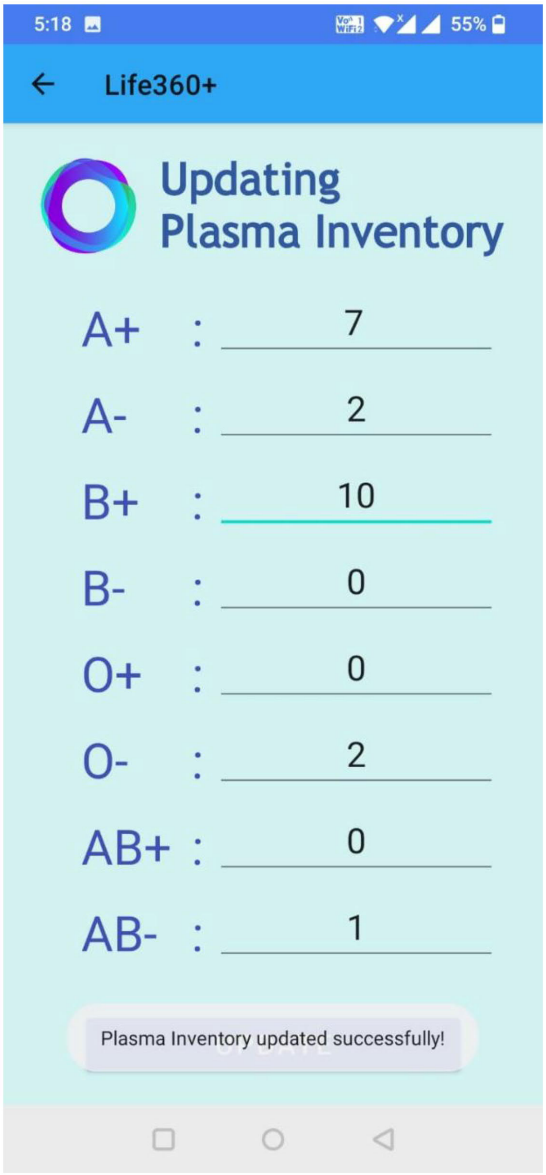

Figure 5.35: Organization User: Updating Plasma Inventory

The Organization type user can update their organization's plasma inventory. This feature is only available to organizations that are either Hospitals or Plasma Banks. Once an organization (hospital/plasma bank) is created, a Plasma Inventory entry is also made with initial values of all quantities set to zero. Later, the user can update them, as shown above.

CHAPTER 6

CONCLUSION

The mini project has successfully accomplished the goals it had set out in the objectives and design sections of this report. The Android application developed in this mini project has been able to provide a simple and easy to use graphical user interface that can potentially save lives.

Separate set of activities have been made for the both type of users, i.e. individual users and organization users.

The individual user type UI has successfully implemented several modules. An individual user can opt to volunteer as a potential blood donor at the time of registration. Once logged in, the user can update his/her profile information which would be shown during a donor search by other users. The user can additionally search for near-by blood banks, plasma banks and donors who have been registered on the application.

The organization user type UI has also successfully implemented several modules. Organizations like hospitals, blood and plasma banks can now update their blood inventory / plasma inventory on a real time basis and changes will automatically be reflected to any individual users who are searching for near-by blood / plasma.

The application has also appropriately exploited the power of Android programming through the use of its various features, views and layouts. By using a cloud-based database like Firebase's Realtime Database, users can have real-time data continuously available.

In the future, I aim to extend this mini project by having adding new features that would enhance the user experience like providing the ability for individual users to make appointments with doctors and hospitals among other possible features.

REFERENCES

[1] https://developer.android.com/docs

[2] https://firebase.google.com/docs

[3] https://www.javatpoint.com/

[4] https://geeksforgeeks.org/

[5] https://www.tutorialspoint.com/

[6] https://data-flair.training/

[7] https://www.section.io/

[8] https://www.edureka.co/

[9] https://abhiandroid.com/

[10] https://medium.com/

[11] https://material.io/

[12] https://stackoverflow.com/