**Life Saver Blood Bank**

Software Requirements Specification

Version-1.5

Date: June 29, 2020

Mohan Pratapa

Venkata Hareesh Bhuma

Raghunandan Kumar Naishadam

Sumanth Gorantla

Venkat Prudhvi Dommaraju

Submitted in partial fulfilment

Of the requirements of

CSIS 44-691 Graduate Directed Project 1

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

# Document Approval

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

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

**Table of Contents**

**Table of Contents Page Number**

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

3.5.1. Performance

3.5.2. Reliability

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

1. Design

4.1. ER diagram

4.2. GUI

1. Analysis Models

5.1. Data Flow Diagram

5.2. Sequence Diagram

1. Technical Manual
2. End-user Manual

## Introduction

**1.1 Purpose:**

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

**1.2 Scope:**

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

**1.4 References:**

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

## 2. General Description

**2.1 Product functions:**

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

**3. Specific Requirements**

**3. Specific Requirements**

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

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

**3.1 External Interface Requirements**

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

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

**3.1.3 Software Interfaces (programming Lang)**

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

**3.2 Functional Requirements:**

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

**Donor Functionalities**:

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

**Accepter Functionalities:**

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

**Admin Functionalities:**

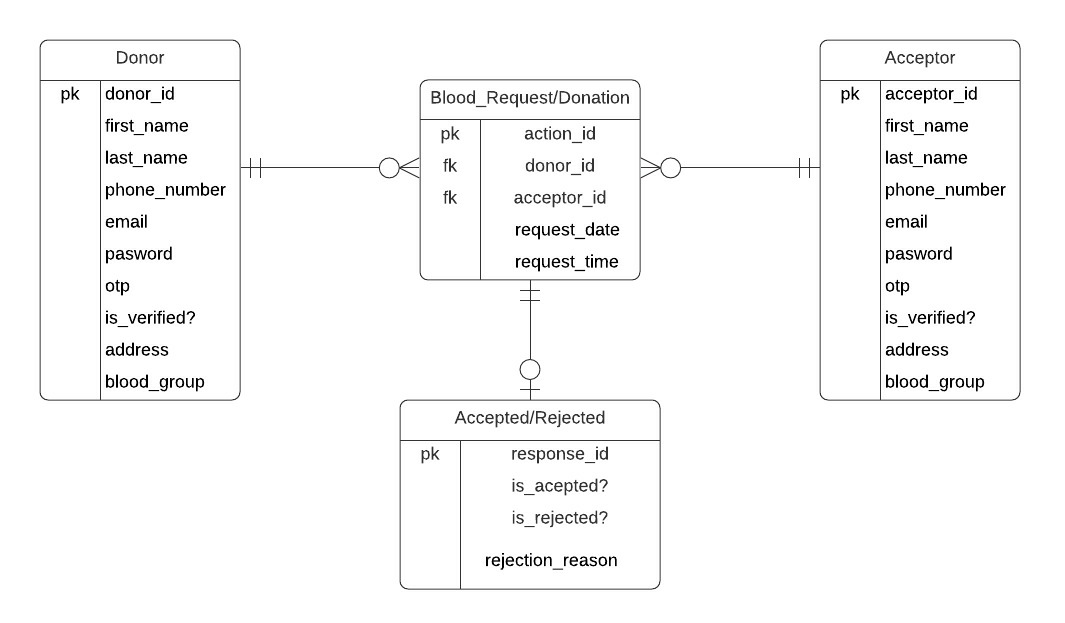
* **Manage Users:** The Manage Users screen will add, change, and delete the users. Additionally, Admin users can search for users, change user details, and manage what users have permission to see and do in the app.

**3.5.1 Performance:**

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

**4. Design**

**4.1 ER Diagram:**

****

**Description:**

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

**4.2 GUI:**

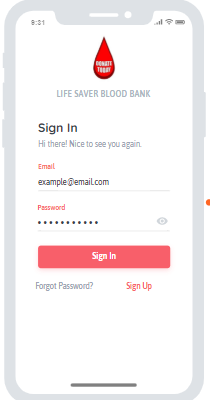
**1. Launch Screen:**

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

****

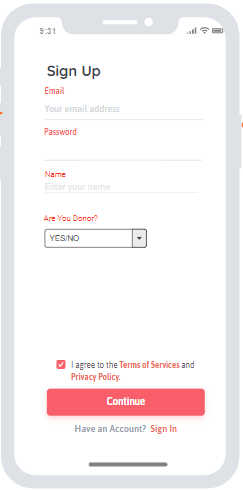
1. **Sign in Screen:**

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

****

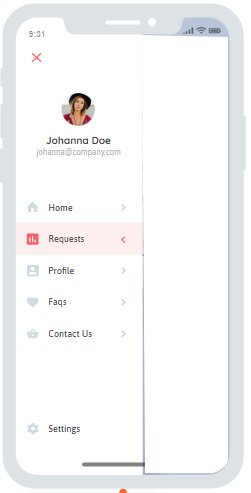
1. **Sign Up Screen:**

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

****

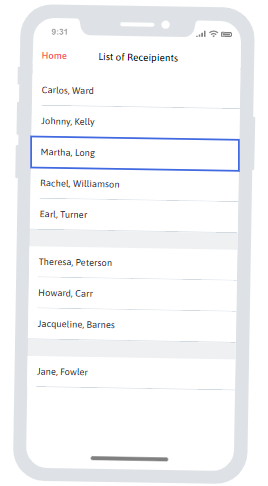
1. **Donor Home Screen:**

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

****

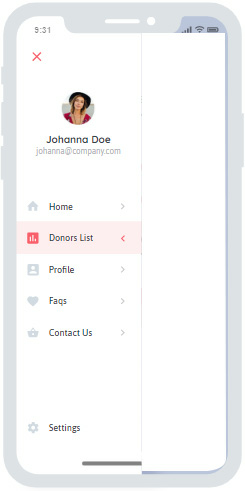
1. **Requests Screen:**

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

****

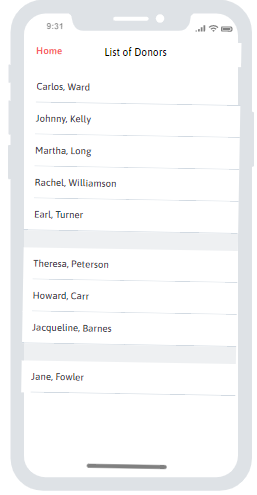
1. **Acceptor Home Screen:**

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

****

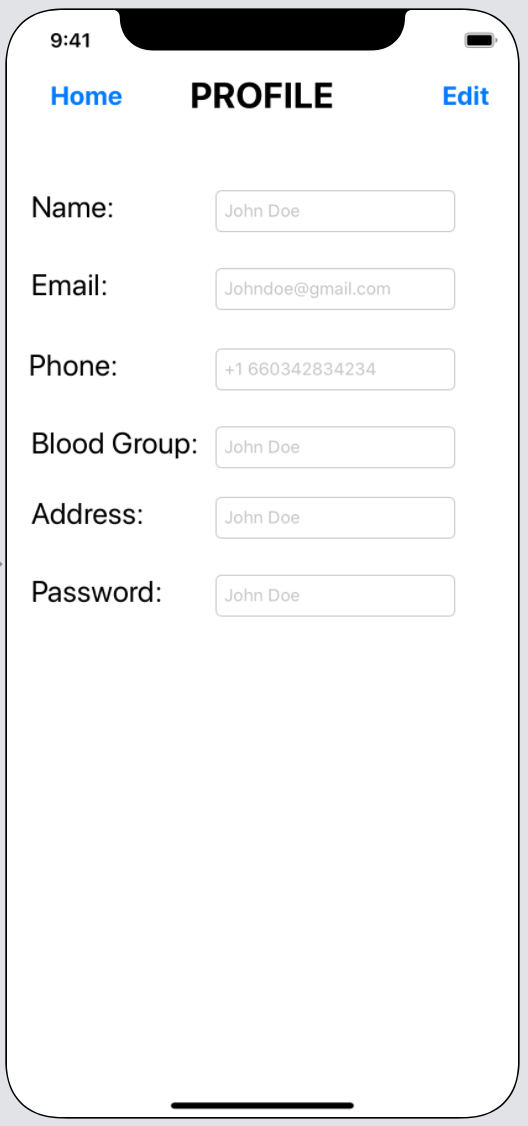
1. **Donor list Screen:**

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

****

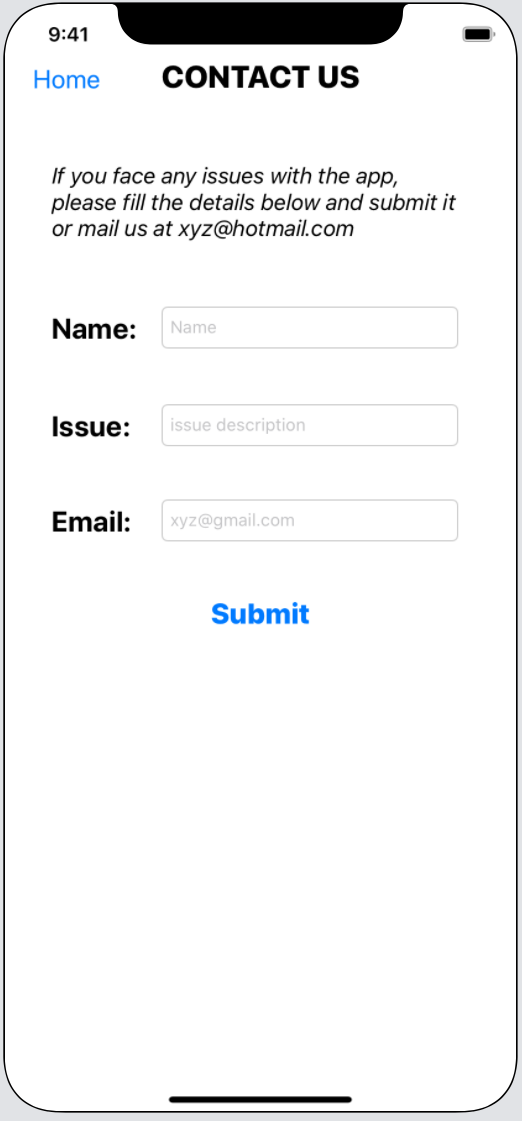
1. **Profile Screen:**

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

****

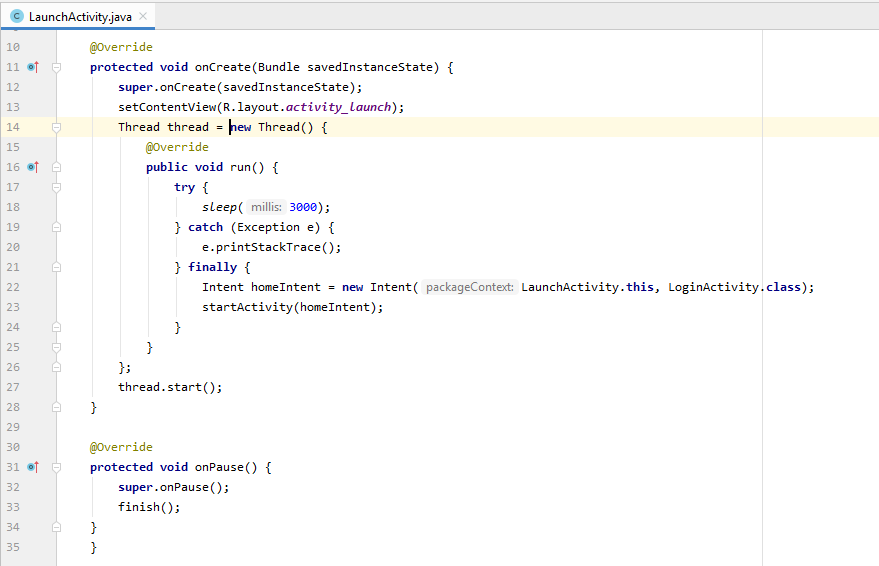
1. **Contact Us Screen:**

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

****

1. **Technical Manual**

**Launch Activity:**



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

**Login Activity:**

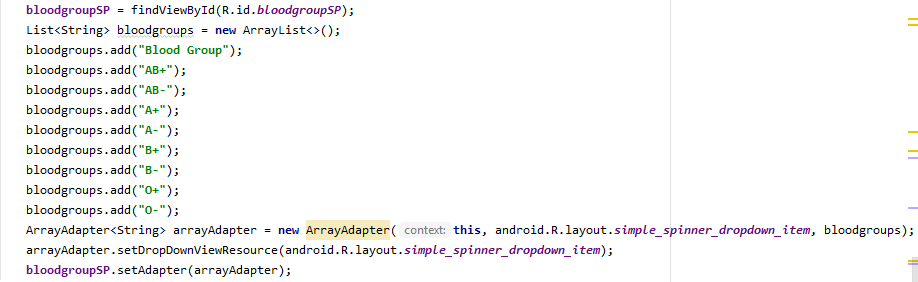


We have created an instance of FirebaseFirestore and defined an onClick method. In this method, we will retrieve email and password which is already stored in the database by using getText () and toString () methods. Once we click on sign in it will hit the database and verify the username and password,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

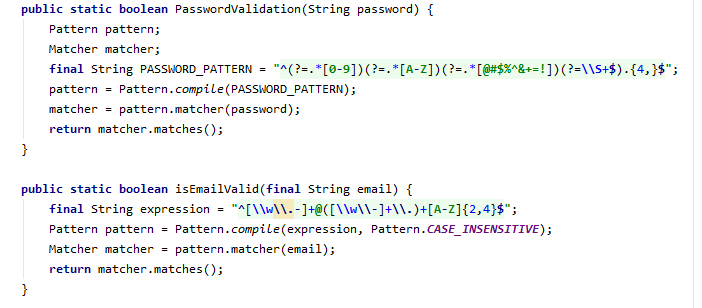
**Signup Activity:**



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

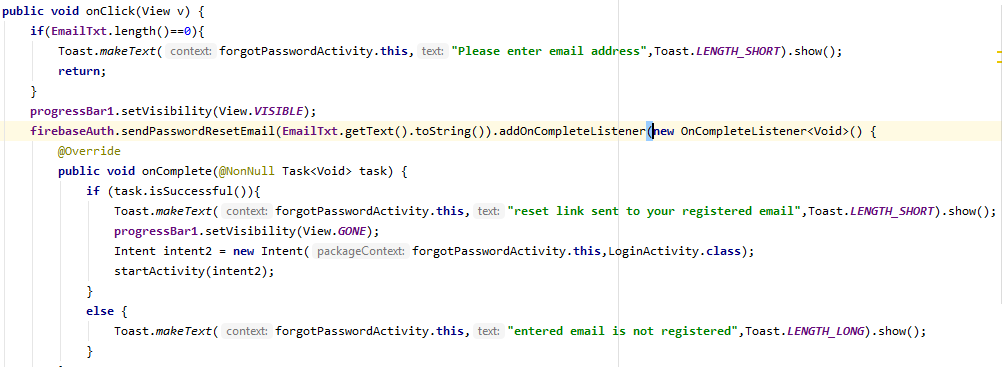


The above code is for adding blood groups in spinner which is a list.



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

**Forgot Password:**



Above code snippet have two methods for handling forgot password.

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

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

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

1. **End-user Manual**

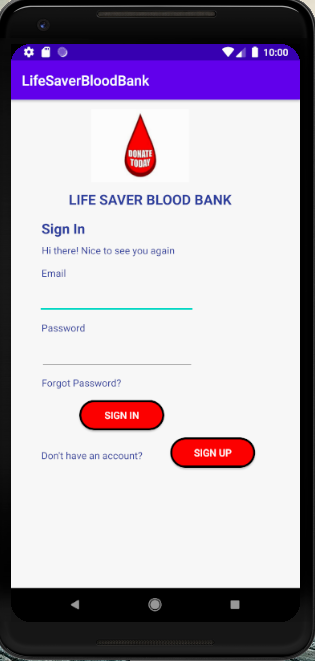
**Launch Page:**

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



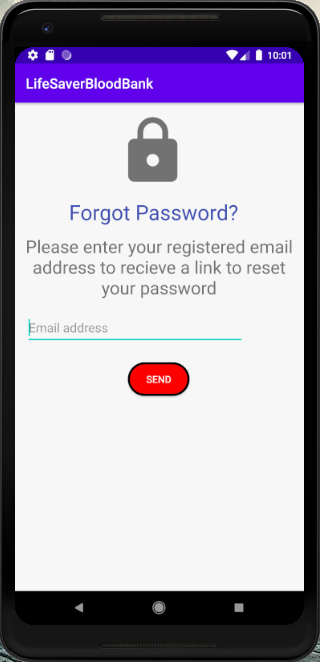
**Login Page:**

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

****

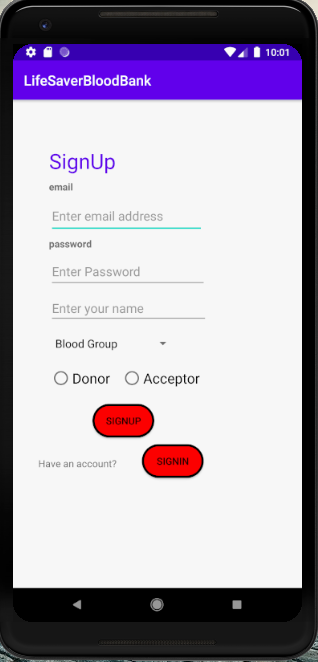
**Forgot Password Page:**

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



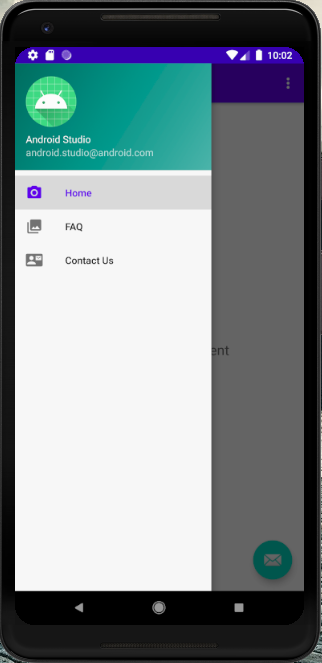
**Sign Up Page:**

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



**Home Page:**

This the main activity after login which have a navigation bar with all the options which can be accessed by the user.

****

**Contact Us Page:**

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

