

PROJECT REPORT ON

# **LIFE MATE**

Submitted by

**MOHAMMED FAZIL KP**

Under the guidance of

**Shahinsha**



**FUTURA LABS**

ENABLE • TRANSFORM • ACCELERATE

*Futura Labs Kochi, Kerala, India*

*PH: 9946325888,*

<https://thefuturalabs.com>

# CONTENTS

SL.NO.	CONTENTS	PAGE NO.
01	INTRODUCTION	03
02	CONCEPTUAL MODEL	04
03	DATABASE DESIGN	08
04	BIBLIOGRAPHY	10
05	APPENDIX	11
06	ANNEXURE	12

# INTRODUCTION

## ➤ ABSTRACT

The application proposed here aims at improving to get blood requirement. Through this, a user-friendly interaction between admins and users can be make possible.

The project includes four modules: - Admin and User. This is intended to be create by comprising features such as Request for Blood, Blood Sources, Request Updates, Manage Users, Manage Requests, etc.

The front end of the application is planned to develop with Dart Programming language with flutter framework and the back end is set on Firebase as database.

## ➤ MODULES

The complete project is divided into two modules. And the modularization is based on the type of the users inthe system. The different modules based on the type of the users of the system are:

➤ ADMIN

➤ USER

### USER

- Login
- Sign Up
- User Profile
- Register
- History
- User Account
- Request Updates
- Blood Source

### ADMIN

- Login
- Admin Profile
- Manage Requests
- Manage Users
- History
- Admin Account

# CONCEPTUAL MODELS

## ➤ REQUIREMENT MODELING DATA FLOW DIAGRAM

Data Flow Diagram (DFD) is used to define the flow of the system and its resources such as information. Data flow diagrams are the way of expressing system requirements in a graphical manner. DFD represents one of the most ingenious tools used for structured analysis. A DFD is also known as a bubble chart. It has the purpose of clarifying system requirements identifying major transformations that will become programs in system design. In the normal convention, logical DFD can be completed using only 5 notations.



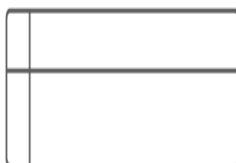
Represents source/destination data.



Represents data flow.

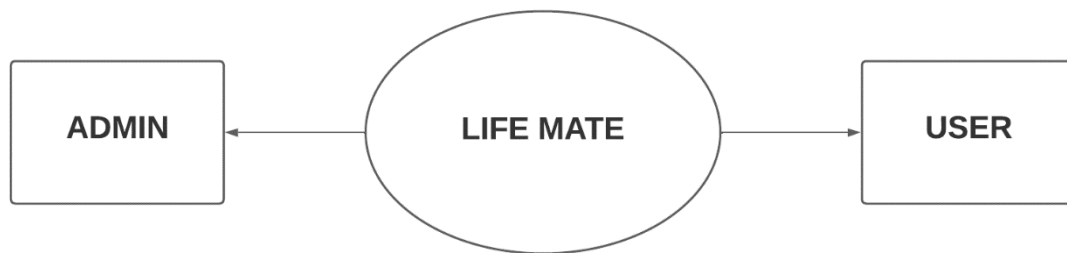


Represents a process that transforms incoming data into outgoing flow.

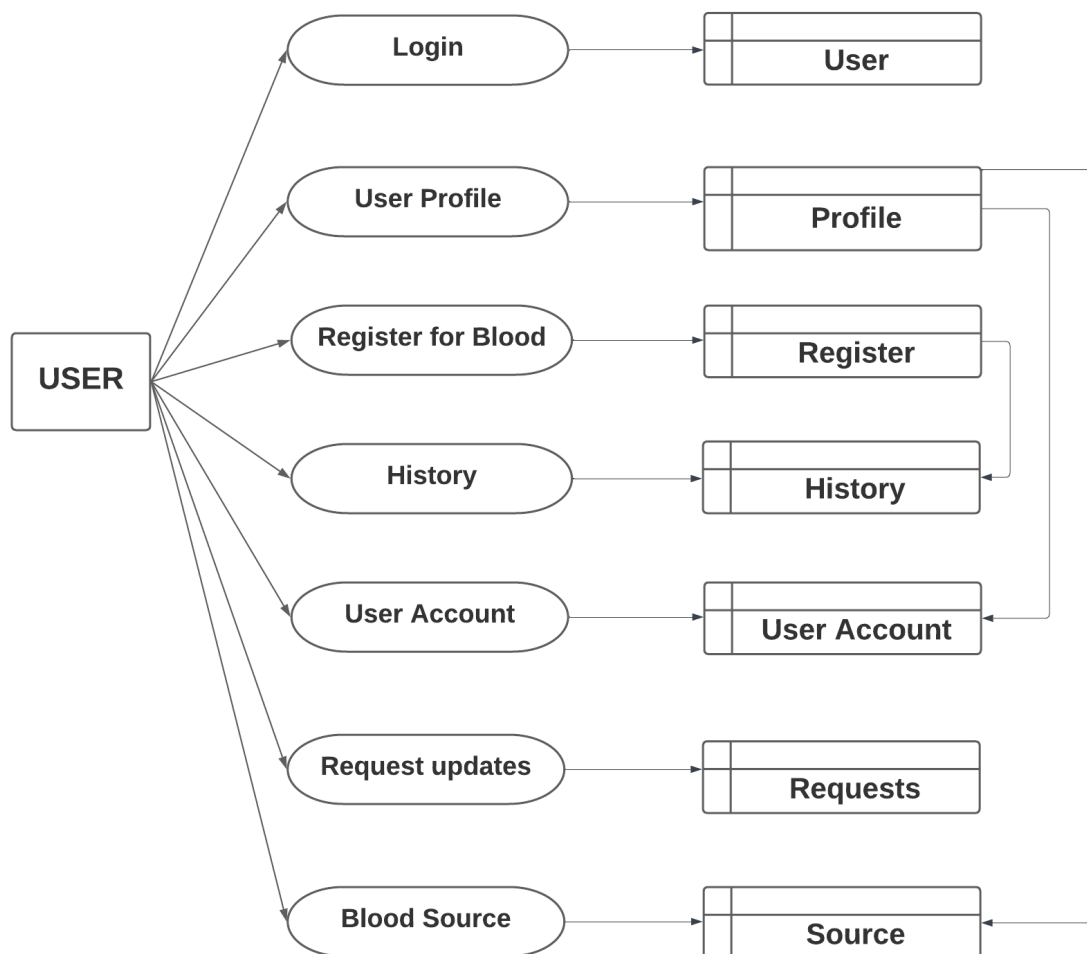


Represents data storage/internal storage.

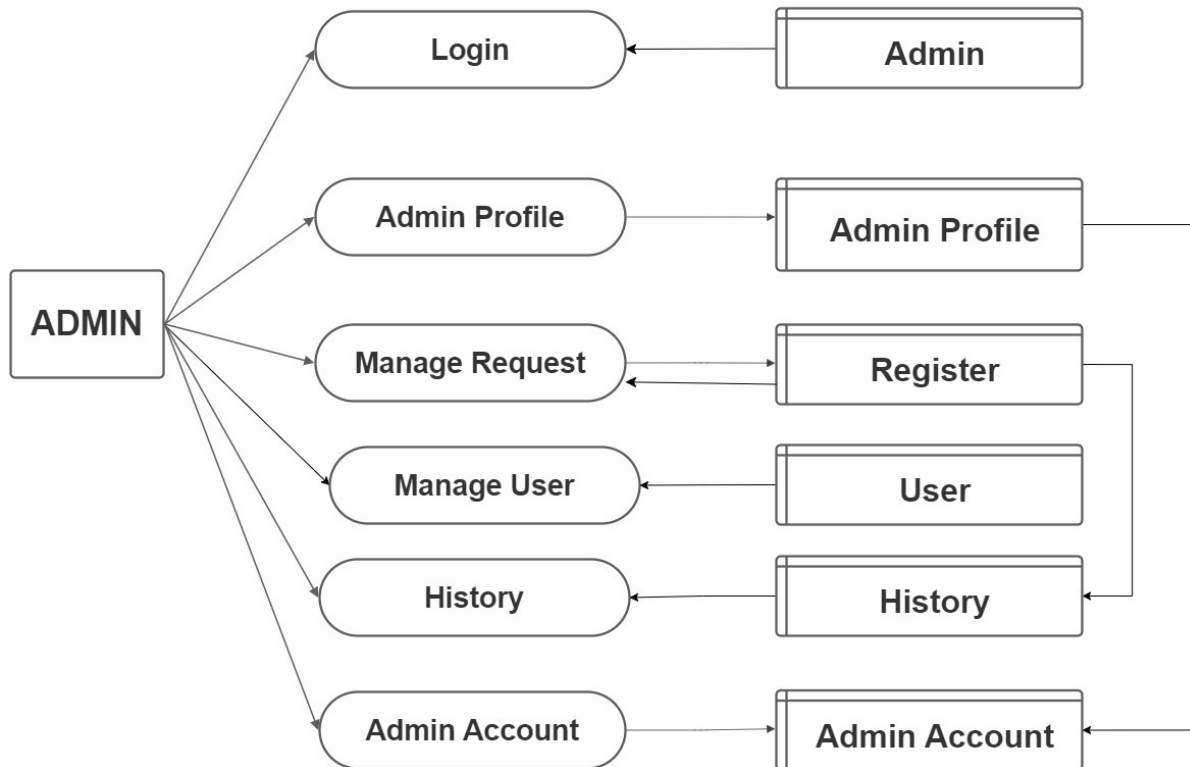
➤ **LEVEL 0 DFD**



➤ **LEVEL 1 DFD - USER**



### ➤ LEVEL 1 DFD - ADMIN



### 3.3.1 FUNCTIONS

The system after careful analysis has been identified to be presented with the following modules:

#### ➤ USER

- **Login** – User can login to the application by entering username and password.
- **User Profile** – User can upload their profile details.
- **Register for Blood** – User can send blood request to admin.
- **History** – User can view all their progress and updates.
- **User Account** – User can view their profile.
- **Request Updates** – User can view updates provided by admin.
- **Blood Source** – Users can view all profiles to match their requirements.

## ➤ **ADMIN**

- **Login** – Admin can login to the application by entering panel code and password.
- **Admin Profile** – Admin can upload their profile details.
- **Manage Requests** – Admin can view and manage request.
- **Manage Users** – Admin can view and manage all users.
- **History** – Admin can view all their progress and updates.
- **Admin Account** – Admin can view their profile.

# **DATABASE DESIGN**

A database is a collection of records. The main objective of database design is to provide effective auxiliary storage without any applications and to contribute to the overall efficiency of the computer program. components of the whole system. The organization of data in the database aims to achieve the following objectives.

- Controlled redundancy
- Ease of learning in use
- Data independence
- More information in low cost
- Accuracy and integrity
- Recovery from failures
- Privacy and security
- Performance

The design should be done in a way the information stored in the database can be retrieved quickly whenever necessary. The general theme behind a database is to handle information as an interfered whole. A database is a collection of interrelated data stored with minimum redundancy to serve users quickly and efficiently. Database design runs parallel without application design. As we collect information about what is to be done, we will obviously collect information about data needed to enter,stored messages and printed reports. The designing of the database is done with utmost care and security during the designing phase of the system. Special care was taken to develop a minimum number of databases for the maximum efficiency of the system.



LifeMate - Authentication - Firebase

console.firebase.google.com/u/0/project/lifemate-9147d/authentication/users

Google Gmail YouTube Maps Resume Builder | Re...

**Firebase**

Project Overview

Project shortcuts

- Authentication
- Firestore Database
- Storage

Product categories

Build

Release and monitor

Analytics

Engage

All products

Customise your navigation

You can now focus your console experience by customising your navigation

Learn more Got it

Spark

No cost \$0/month Upgrade

**LifeMate**

## Authentication

Users Sign-in method Templates Usage Settings Extensions

Search by email address, phone number or user UID

Add user

Identifier	Providers	Created	Signed in	User UID
hellokp@gmail.com		4 Oct 2023	4 Oct 2023	TwDAWPbYbExKm2vwlagoH0HCJ...
1234@gmail.com		1 Oct 2023	1 Oct 2023	koQdaj8oJxMeTskRCieK33yCXZB3
mfazilkp10@gmail.com		1 Oct 2023	6 Oct 2023	F8yvTmWp1RP9te9tEnrxM2ytokS2
mfazilkp@gmail.com		1 Oct 2023	1 Oct 2023	7gbKUIUVQbZqCB0Nb0bsjMwIOE2
email@gmail.com		1 Oct 2023	1 Oct 2023	sVesL8jbMkMnLSsN0pxBqqqPBT...
fazilkp7@gmail.com		1 Oct 2023	1 Oct 2023	4DyvNesoCXe8qeMYoybAjoypmM...
hahaha@gmail.com		1 Oct 2023	1 Oct 2023	qlg9GfXVae3qRvJ1Iij7HMMaJ3
hello@gmail.com		1 Oct 2023	1 Oct 2023	fKizos3przSYTBzhCdsclAkrBGk2
fazilkp@gmail.com		1 Oct 2023	1 Oct 2023	SWN4AJc0neypuH5nhSnaCb6Vr...

Rows per page 50 1 - 9 of 9

29°C Partly sunny

LifeMate - Cloud Firestore - Firebase

console.firebase.google.com/u/0/project/lifemate-9147d/firestore/data/~2Fusers-2F4DyvNesoCXe8qeMYoybAjoypmMJ3

Google Gmail YouTube Maps Resume Builder | Re...

**Firebase**

Project Overview

Project shortcuts

- Authentication
- Firestore Database
- Storage

Product categories

Build

Release and monitor

Analytics

Engage

All products

Customise your navigation

You can now focus your console experience by customising your navigation

Learn more Got it

Spark

No cost \$0/month Upgrade

**LifeMate**

## Cloud Firestore

Data Rules Indexes Usage Extensions

Panel view Query builder

More in Google Cloud

users > 4DyvNesoCXe8...

(default) users 4DyvNesoCXe8qeMYoybAjoypmMJ3

+ Start collection

admin

adminDonar

adminLogin

adminProfile

userProfile

userRegister

users

+ Add document

4DyvNesoCXe8qeMYoybAjoypmMJ3

7gbKUIUVQbZqCB0Nb0bsjMwIOE2

F8yvTmWp1RP9te9tEnrxM2ytokS2

TwDAWPbYbExKm2vwlagoH0HCJGz1

koQdaj8oJxMeTskRCieK33yCXZB3

qlg9GfXVae3qRvJ1Iij7HMMaJ3

sVesL8jbMkMnLSsN0pxBqqqPBT22

yh4a2Zc0IXbuC8tEZ1RWuvBENeQ2

+ Start collection

+ Add field

email: "fazilkp7@gmail.com"

password: "123456"

uid: "4DyvNesoCXe8qeMYoybAjoypmMJ3"

userName: "ahahaha"

29°C Partly sunny

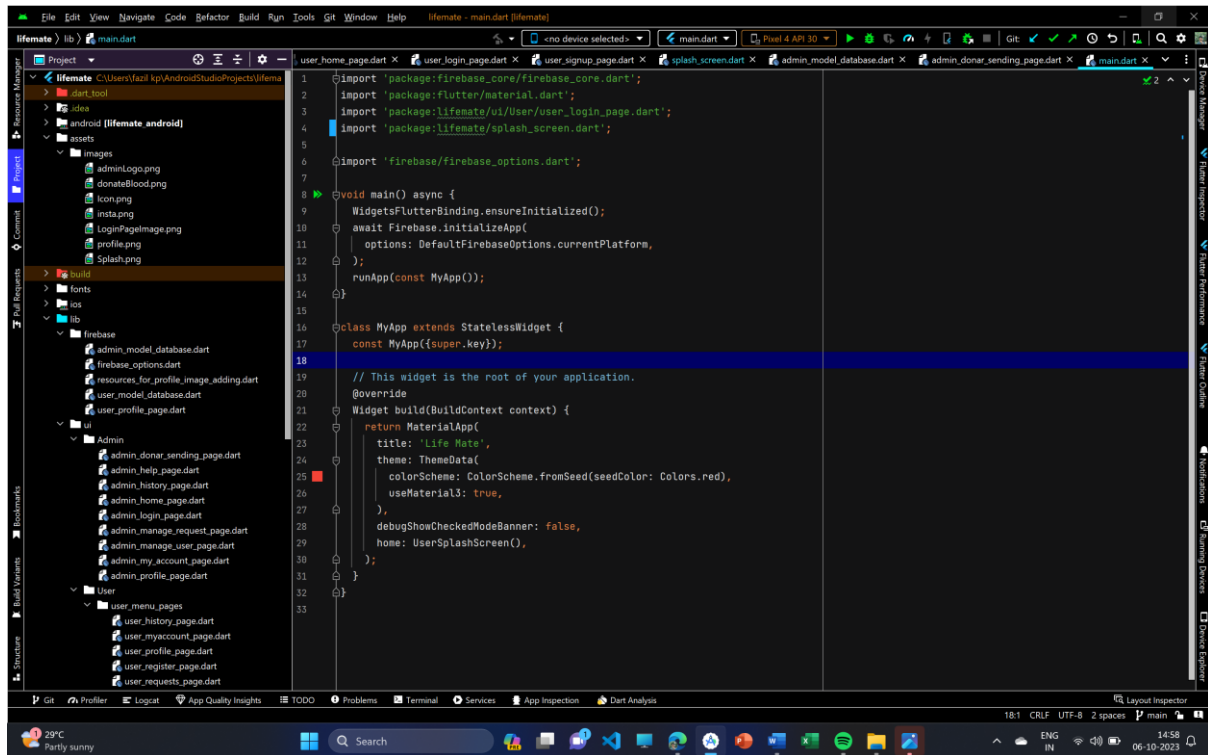
# BIBLIOGRAPHY

## WEBSITES

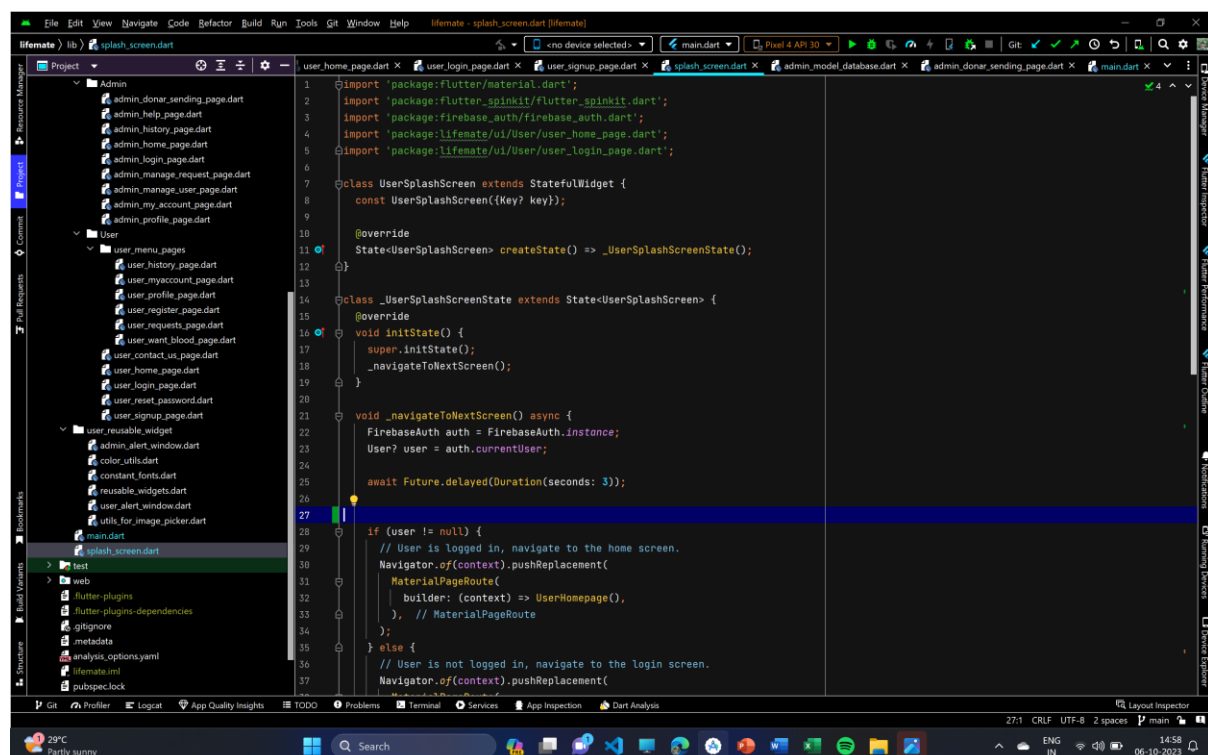
1. Flutter documentation: <https://flutter.dev/docs>
2. Firebase documentation: <https://firebase.google.com/docs>
3. FlutterFire library for Flutter and Firebase integration: <https://firebase.flutter.dev/>
4. Flutter and Firebase tutorial by Fireship: <https://fireship.io/lessons/flutter-firebase-app-course/>
5. Flutter and Firebase tutorial by The Net Ninja:  
<https://www.youtube.com/watch?v=1gDhl4leEzA&list=PL4cUxeGkcC9j--TKIdkb3ISfRbJeJYQwC>
6. Flutter and Firebase tutorial by Reso Coder:  
<https://resocoder.com/category/tutorials/flutter/firebase/>
7. Flutter and Firebase tutorial by Andrea Bizzotto:  
<https://www.andreabizzotto.com/course/flutter-firebase/>
8. Flutter and Firebase tutorial by Flutter Explained: <https://flutter-explained.dev/docs/firebase/firestore/>

# CODE

## APPENDIX



```
1 import 'package:firebase_core/firebase_core.dart';
2 import 'package:flutter/material.dart';
3 import 'package:lifemate/ui/User/user_login_page.dart';
4 import 'package:lifemate/splash_screen.dart';
5
6 import 'firebase/firebase_options.dart';
7
8 void main() async {
9   WidgetsFlutterBinding.ensureInitialized();
10  await Firebase.initializeApp(
11    options: DefaultFirebaseOptions.currentPlatform,
12  );
13  runApp(const MyApp());
14
15  class MyApp extends StatelessWidget {
16    const MyApp({super.key});
17
18    // This widget is the root of your application.
19    @override
20    Widget build(BuildContext context) {
21      return MaterialApp(
22        title: 'Life Mate',
23        theme: ThemeData(
24          colorScheme: ColorScheme.fromSeed(seedColor: Colors.red),
25          useMaterial3: true,
26        ),
27        debugShowCheckedModeBanner: false,
28        home: UserSplashScreen(),
29      );
30    }
31  }
32 }
```

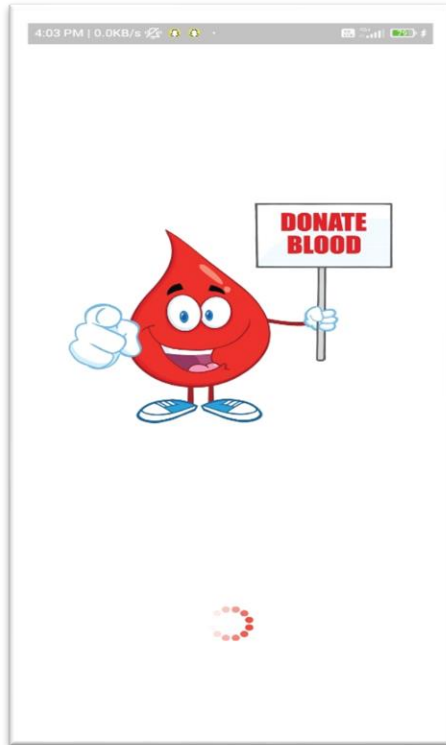


```
1 import 'package:flutter/material.dart';
2 import 'package:flutter_spinkit/flutter_spinkit.dart';
3 import 'package:firebase_auth/firebase_auth.dart';
4 import 'package:lifemate/ui/User/user_home_page.dart';
5 import 'package:lifemate/ui/User/user_login_page.dart';
6
7 class UserSplashScreen extends StatefulWidget {
8   const UserSplashScreen({Key? key});
9
10  @override
11  State<UserSplashScreen> createState() => _UserSplashScreenState();
12
13  class _UserSplashScreenState extends State<UserSplashScreen> {
14    @override
15    void initState() {
16      super.initState();
17      _navigateToNextScreen();
18    }
19
20    void _navigateToNextScreen() async {
21      FirebaseAuth auth = FirebaseAuth.instance;
22      User? user = auth.currentUser;
23
24      await Future.delayed(Duration(seconds: 3));
25
26      if (user != null) {
27        // User is logged in, navigate to the home screen.
28        Navigator.of(context).pushReplacement(
29          MaterialPageRoute(
30            builder: (context) => UserHomePage(),
31          ), // MaterialPageRoute
32        );
33      } else {
34        // User is not logged in, navigate to the login screen.
35        Navigator.of(context).pushReplacement(
36          MaterialPageRoute(
37            builder: (context) => UserLoginPage(),
38          ), // MaterialPageRoute
39        );
40      }
41    }
42  }
43 }
```

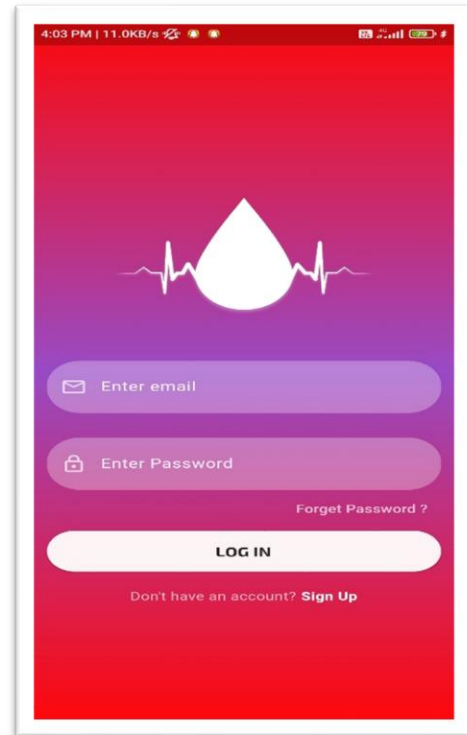
## ANNEXURE

### ➤ APP – FOR USERS

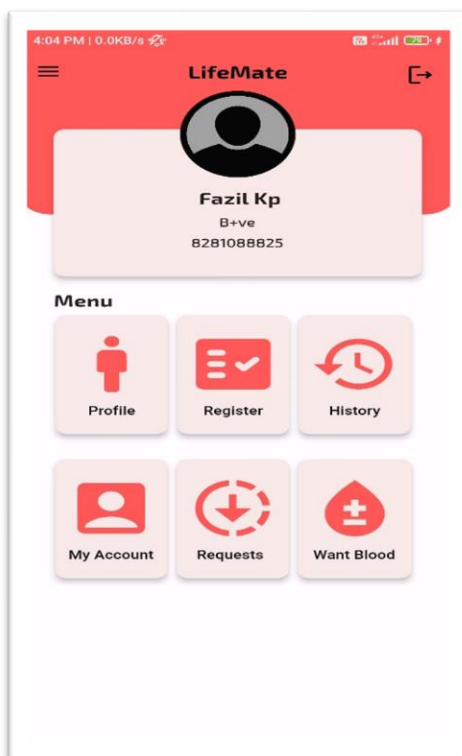
#### SPLASH SCREEN



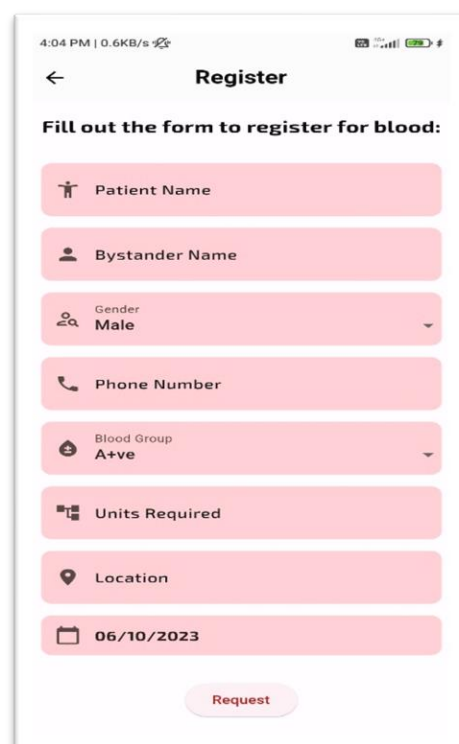
#### LOGIN PAGE



#### HOME PAGE

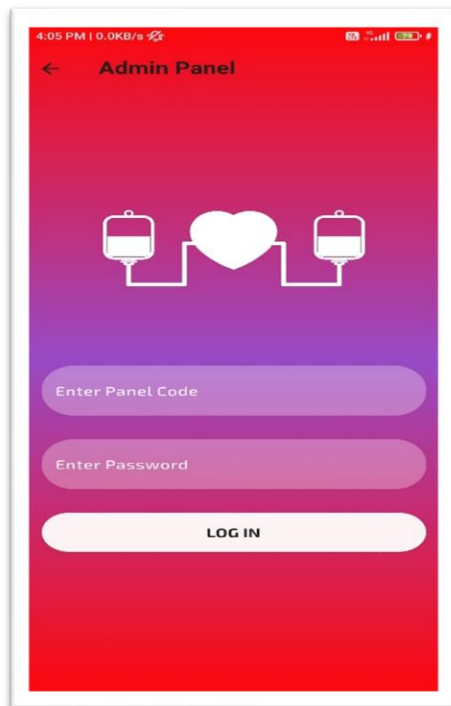


#### REGISTER PAGE

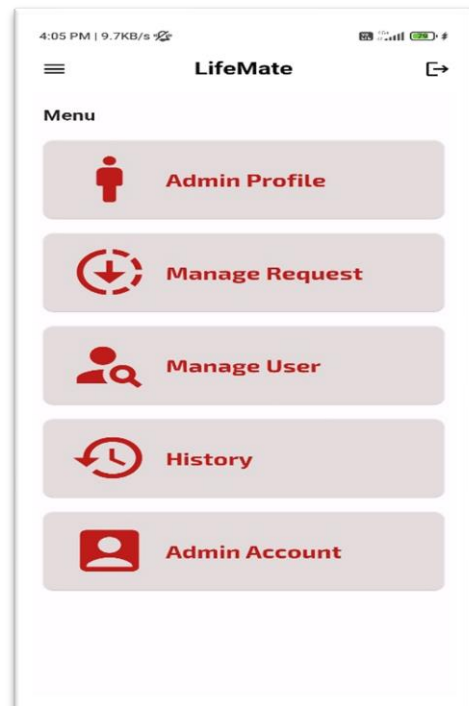


## ➤ APP – FOR ADMIN

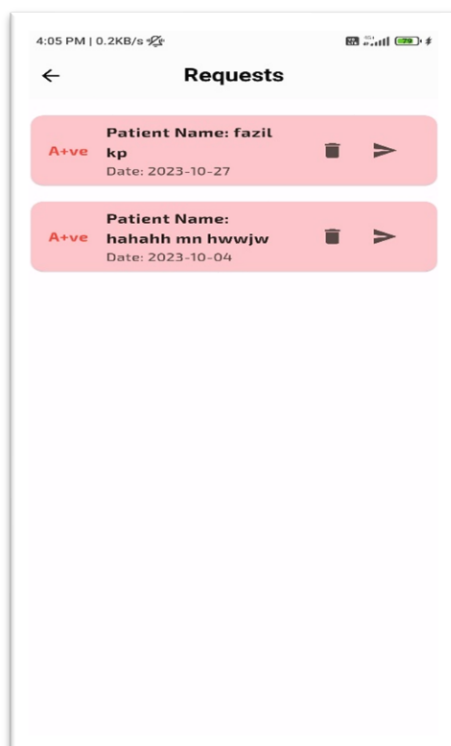
### LOGIN PAGE



### HOME PAGE



### REQUESTS PAGE



### USER LIST

