**TELEMEDICINE : EMPOWERING EMERGENCY DERMATOLOGY CARE WITH A CONSULT MOBILE APPLICATION**

**Abstract**:

In these task, the modules of client and specialist in these application, the client register and login in to the application. the client login with legitimate verification, in these application the client catch picture of skin and transfer to specialist, the specialist register and login in to the application, in client side the specialist list view and the client select the specialist to send pictures with his information. The specialist get and add the aftereffect of his sickness. The specialist send the installment add up to the client, the client once pay the sum to the specialist the specialist confirm and send the remedy report in these application. at long last the client get the record to download

**TABLE OF CONTENTS**

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 2.3.2 | Module Diagram |  |
| 4.2 | Activity Diagram for Anonymous database Management |  |
| 4.3 | Use case Diagram for Anonymous database Management |  |
| 4.4 | Data flow diagram for Anonymous database Management |  |
| 4.5 | Sequence diagram for Anonymous database Management |  |
| 4.6 | Collaboration diagram for Anonymous database Management. |  |
| 4.7 | Class diagram for Anonymous database Management |  |
| 4.8 | Architecture Diagram for Anonymous database Management |  |
| 4.9 | Object Diagram |  |
| 4.10 | State Diagram |  |
| 4.11 | Component Diagram |  |
| 4.12 | E-R Diagram |  |

**LIST OF SYMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class B  Class A  Class A  Class B | Associations represents static relationships between classes. Roles representsthe way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processs. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | F inal state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Usecase |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which is a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or acion. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**CHAPTER 1:**

**1.1 INTRODUCTION:**

In these endeavor, the modules of client and master in these application, the client register and login in to the application. the client login with legitimate check, in these application the client get picture of skin and move to prepared proficient, the master register and login in to the application, in client side the master overview view and the client select the master to send pictures with his information. The master get and add the possible result of his burden. The master send the piece add up to the client, the client once pay the total to the master the master assert and send the fix report in these application. At last the client get the record to download

**SCOPE**:

The scope of this application encompasses a range of functionalities designed to facilitate efficient interactions between clients and dermatological specialists. It includes secure user registration and login processes for both parties, ensuring client legitimacy through verification steps. Clients can capture and upload images of their skin conditions, providing additional context, while specialists can view these submissions and offer detailed diagnoses along with tailored feedback. A secure payment system will allow clients to pay specialists easily, with a confirmation process to ensure transactions are completed. Additionally, specialists can create and send remedy reports that clients can download for future reference. The application will feature a user-friendly interface, analytics for tracking interactions and treatment outcomes, a feedback mechanism for client ratings, and support for multiple languages to cater to a diverse user base. This comprehensive scope aims to enhance the overall experience of dermatological care for both clients and specialists.

**OBJECTIVE:**

The objective of this application is to create a streamlined and user-friendly platform that connects clients with dermatological specialists, enabling efficient communication and care. By allowing clients to securely register, upload images of their skin conditions, and select specialists, the application facilitates prompt and accurate diagnoses. It aims to simplify payment processes and ensure clients receive timely remedy reports, enhancing the overall experience of seeking dermatological advice. Ultimately, the application seeks to improve access to specialized care, promote effective treatment management, and foster a supportive environment for clients navigating their skin health concerns.

**1.2.1 LITERATURE SURVEY:**

# TITLE: Designing a High-Sensitivity Microscale Triple-Band Biosensor Based on Terahertz MTMs to Provide a Perfect Absorber for Non-Melanoma Skin Cancer Diagnostic

**AUTHOR :** Musa N. Hamza , Mohammad Tariqul Islam , Senior Member, IEEE, Slawomir Koziel , Fellow, IEEE, Muhamad A. Hamad , Iftikhar ud Din , Ali Farmani , Sunil Lavadiya , and Mohammad Alibakhshikenari

**YEAR:** 2024

**DESCRIPTION:**

Non-melanoma skin cancer (NMSC) is among the most prevalent forms of cancer originating in the top layer of the skin, with basal cell carcinoma (BCC) and squamous cell carcinoma (SCC) being its primary categories. While both types are highly treatable, the success of treatment hinges on early diagnosis. Earlystage NMSC detection can be achieved through clinical examination, typically involving visual inspection. An alternative, albeit invasive, method is a skin biopsy. Microwave imaging has gained prominence for non-invasive early detection of various cancers, leveraging distinct dielectric properties of healthy and malignant tissues to discriminate tumors and categorize them as benign or malignant. Recent studies demonstrate the potential of terahertz (THz) spectroscopy for detecting biomarkers by aligning electromagnetic wave frequencies in the low THz range (0.1 to 10 THz) with resonant frequencies of biomolecules, such as proteins. This study proposes an innovative microscale biosensor designed to operate in the THz range for the high-sensitivity and efficient diagnosis of non-melanoma skin cancer. By incorporating meticulously designed metamaterial layers, the sensor’s absorption properties can be controlled, a critical aspect for discriminating between normal and NMSC-affected skin. In particular, the interaction between skin and THz waves, influenced by dielectric properties and unique vibrational resonances of molecules within tissue, is crucial for wave propagation and scattering. Extensive numerical studies showcased the suitability of the proposed biosensor for NMSC diagnosis, illustrated through specific case studies. These findings hold the potential to pave the way for further development of non-invasive microwave-imaging-based techniques for detecting NMSC and other types of skin cancer.

**TITLE**: Prosthetic Limb Attachment via Electromagnetic Attraction Through a Closed Skin Envelope

**AUTHOR :** Will Flanagan , Kai Becraft, Haley Warren, Alexandra I. Stavrakis , Nicholas M. Bernthal , Thomas J. Hardin, and Tyler R. Clites

**YEAR:** 2024

**DESCRIPTION:**

Objective: Current socket-based methods of prosthetic limb attachment are responsible for many of the dominant problems reported by persons with amputation. In this work, we introduce a new paradigm for attachment via electromagnetic attraction between a bone-anchored ferromagnetic implant and an external electromagnet. Our objective was to develop a design framework for electromagnetic attachment, and to evaluate this framework in the context of transfemoral amputation. Methods: We first used inverse dynamics to calculate the forces required to suspend a knee-ankle-foot prosthesis during gait. We then conducted cadaveric dissections to inform implant geometry and design a surgical methodology for covering the implant. We also developed an in silico framework to investigate how electromagnet design affects system performance. Simulations were validated against benchtop testing of a custom-built electromagnet. Results: The physical electromagnet matched simulations, with a root-meansquare percentage error of 4.2% between measured and predicted forces. Using this electromagnet, we estimate that suspension of a prosthesis during gait would require 33 W of average power. After 200 and 1000 steps of simulated walking, the temperature at the skin would increase 2.3 °C and 15.4 °C relative to ambient, respectively. Conclusion: Our design framework produced an implant and electromagnet that could feasibly suspend a knee-ankle-foot prosthesis during short walking bouts. Future work will focus on optimization of this system to reduce heating during longer bouts. Significance: This work demonstrates the initial feasibility of an electromagnetic prosthetic attachment paradigm that has the potential to increase comfort and improve residual limb health for persons with amputation.

**TITLE**: Murine Skin Dosimetry Under Millimeter Wave Exposure

**AUTHOR : SERAFEIM IAKOVIDIS , SIMONA LEONARDI, EMILIANO FRATINI , SIMONETTA PAZZAGLIA, MARIATERESA MANCUSO, AND THEODOROS SAMARAS**

**YEAR:** 2024

**DESCRIPTION:**

The upper part of the frequency spectrum (millimeter waves, MMW) applied by modern communications technologies (5G and beyond), makes skin the dominantly exposed tissue to electromagnetic fields. In this work, a methodology for murine skin dosimetry evaluation is presented, intended to contribute to animal studies with mice exposed to MMW radiation, in particular 27.5 GHz. A stratified skin model is proposed and the variations of the skin layers’ thicknesses during a hair cycle are measured in mice. The variations of skin layers’ dielectric properties due to age, based on the changes of total body water, are also evaluated. The impact of these variations in dosimetric metrics (i.e., mean absorbed power density, APD, and power loss) within each layer is assessed and found to be significant. Changes in the skin layers’ thicknesses throughout a hair cycle considerably affect the APD, resulting in a two-fold increase, compared to changes in the dielectric properties due to aging or due to hair presence inside the skin.

**TITLE** : Large-Scale Vision-Based Tactile Sensing for Robot Links: Design, Modeling, and Evaluation

**AUTHOR** : Lac Van Duong and Van Anh Ho

**YEAR:** 2021

**DESCRIPTION :**

The sense of touch allows individuals to physically interact with and better perceive their environment. Touch is even more crucial for robots, as robots equipped with thorough tactile sensation can more safely interact with their surroundings, including humans. This article describes a recently developed large-scale tactile sensing system for a robotic link, called TacLINK, which can be assembled to form a whole-body tactile sensing robot arm. The proposed system is an elongated structure comprising a rigid transparent bone covered by continuous artificial soft skin. The soft skin of TacLINK not only provides tactile force feedback but can change its form and stiffness by inflation at low pressure. Upon contact with the surrounding environment, TacLINK perceives tactile information through the three-dimensional (3-D) deformation of its skin, resulting from the tracking of an array of markers on its inner wall by a stereo camera located at both ends of the transparent bone. A finite element model (FEM) was formulated to describe the relationship between applied forces and the displacements of markers, allowing detailed tactile information, including contact geometry and distribution of applied forces, to be derived simultaneously, regardless of the number of contacts. TacLINK is scalable in size, durable in operation, and low in cost, as well as being a high-performance system, that can be widely exploited in the design of robotic arms, prosthetic arms, and humanoid robots, etc. This article presents the design, modeling, calibration, implementation, and evaluation of the system.

**TITLE** : A Wide Range Transmission Line Based Linear Displacement Sensor

**AUTHOR** : Ademola A. Mustapha,Omar S. Hassan, Temesgen Ataro, Mohammed Saif ur Rahman, and Mohamed A. Abou-Khousa, Senior

**YEAR:** 2023

**DESCRIPTION :**

Accurate displacement measurement is critical for many applications. Recently, a variety of radio-frequency-based linear displacement sensing techniques were introduced. However, the application of the previously proposed techniques was either not fully demonstrated towards realizing the sensor and/or they were limited in measurement range to less than 4 cm. This paper introduces a fully characterized wide-range radio-frequency-based contactless linear displacement sensor. The sensor consists of a short-circuited 50 Ω microstrip line and a movable current-sensing resonant probe. The probe accurately measures the sinusoidal magnetic field distribution along the short-circuited transmission line from a certain distance above it. The measured normalized field values are mapped to displacement using the inverse of the sinusoidal function in the post-processing stage. The proposed technique is comprehensively validated using simulations and measurements of a compact sensor prototype operating at 727.5 MHz. Furthermore, the merits of the proposed sensor compared to the widely-accepted linear variable differential transformer (LVDT) displacement transducer are highlighted herein. The metrological characterization of the proposed sensor shows that it offers a very wide dynamic range of 68 mm with a standard deviation of the estimation error of less than 0.09 mm (0.13% of the full range). It is also demonstrated that the proposed sensor outperforms the commercial LVDT transducer in terms of overall displacement measurement accuracy. In general, the proposed sensor is scalable and has a theoretical dynamic range of λ/2 making it suitable for a wide range of applications.

|  |  |
| --- | --- |
| **Existing System** | **Proposed System** |
| Although many soft robotic skins have been introduced, their use has been hindered due to practical limitations, such as difficulties in manufacturing, poor accessibility, and cost inefficiency. To solve this, we present a low-cost, easy-to-build soft robotic skin utilizing air-pressure sensors and 3D-printed pads. | the pick expert in these application. the expert view in these application. transfer illness name and solution archive in these application they add how much charges to client. the client view the expenses and pay the sum in these application, the expert affirm the installment, they after installment , the solution view to the client, the past records view and organize in these application |
| **Techniques**:  PHRI METHOD | **Techniques:**  Realtime database, authentication -firebase, firebase API |
| **Demerits**:  Efficiency is low  Not user friendly  Robotic structure of arrangement in sensor connect to devices by automation of cost is high  Latency is low | **Merits:**  Efficiency is high  User friendly  Cost efficient  Latency is high |

**CHAPTER 2**

**2.1 GENERAL:**

**2.2 METHODOLOGIES**

Methodologies is the process of analyzing the principles or procedure of a Progressive Anonymous Database management system.

**2.2.1 MODULES:**

**MODULE NAMES:**

* USER
* DOCTOR

**2.2.2 MODULE DESCRIPTION:**

1. **USER REGISTER AND LOGIN:**

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database user will transfer the amount to provider. They registered with datas the verification of email once verified they able to login in to the application

1. **DOCTOR REGISTER AND LOGIN:**

This is the first module in our project, here symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database user will transfer the amount to provider. They registered with datas the verification of email once verified they able to login in to the application

1. **USER APPLICATION:**

The user apply the form of application of the patient details with the location of datas with capture live image from android mobile device to upload with other datas in these application to share to particular doctor.

1. **DOCTOR ACCEPT:**

If accept the application form of the user with view the images in these application. The doctor update the disease datas in these application

**5.UPLOAD PRESCRIPTION FILE:**

In they upload the prescription of doctor to user of patient of particular datas of management in these application to database.

**6.PAYMENT TO DOCTOR:**

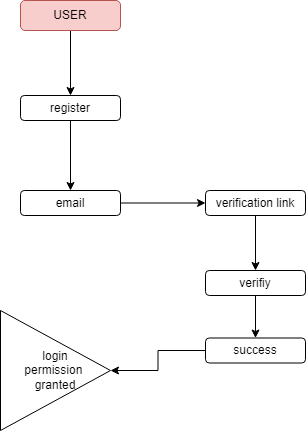
In these module, the user pay the amount of doctor requested amount and gpay number, they successfully payed details pass to doctor of in these application.

**7.DOWNLOAD PRESCRIPTION:**

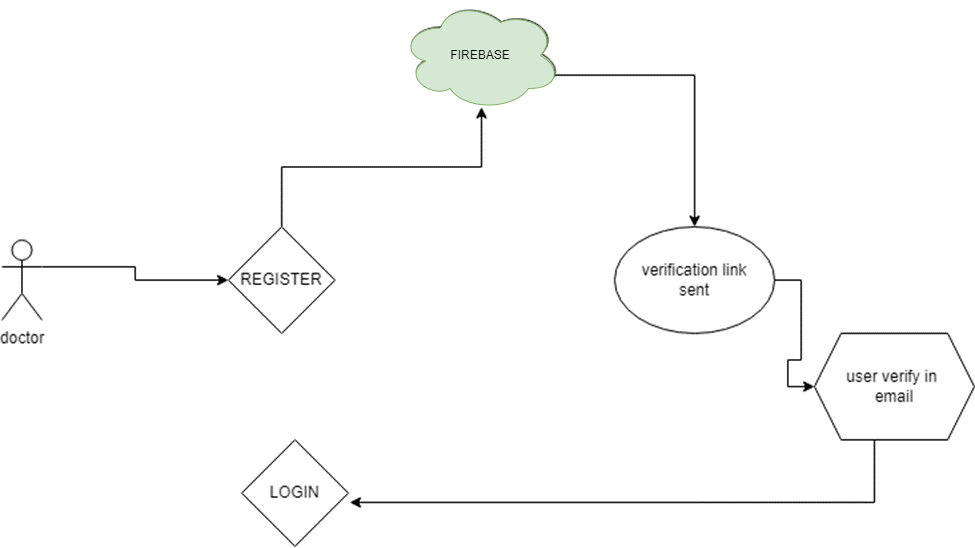
After successfully payed, they automatically view the download option and the user download the prescription documents.

**MODULE DIAGRAM:**

**USER:**

****

**DOCTOR:**

****

**CHAPTER 3:**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

These are the requirements for doing the project. Without using these tools and software’s we can’t do the project. So we have two requirements to do the project. They are

1. Hardware Requirements.

2. Software Requirements.

**MINIMUM SYSTEM REQUIREMENTS**

**HARDWARE REQUIREMENTS**

* PROCESSOR : GREATER THAN I3
* RAM : 4GB DD RAM
* HARD DISK : 1TB

**SOFTWARE REQUIREMENTS**

* FRAMEWORK : FLUTTER
* OPERATING SYSTEM : WINDOWS 10
* IDE : ANDROID STUDIO
* DATABASE : FIREBASE

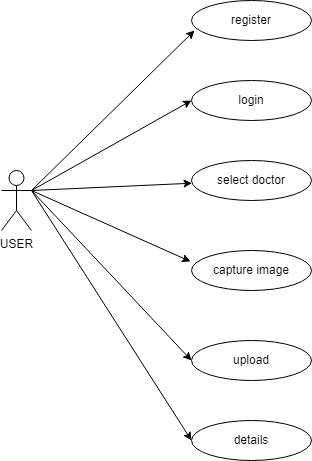
**CHAPTER 4**

**DESIGN ENGINEERING**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modeling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

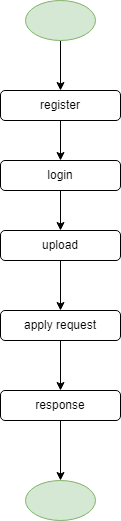
**4.1.1 USE-CASE DIGRAM:**



**EXPLANATION:**

The use case diagram is the main building block of [object oriented](http://en.wikipedia.org/wiki/Object_oriented) modeling. It is used both for general [conceptual modeling](http://en.wikipedia.org/wiki/Conceptual_model) of the systematic of the application, and for detailed modeling translating the models into [programming code](http://en.wikipedia.org/wiki/Programming_code). For this in our component diagram first propose a data In this proposed method we are using Hash-Solomon Code Algorithm to encrypt the data.

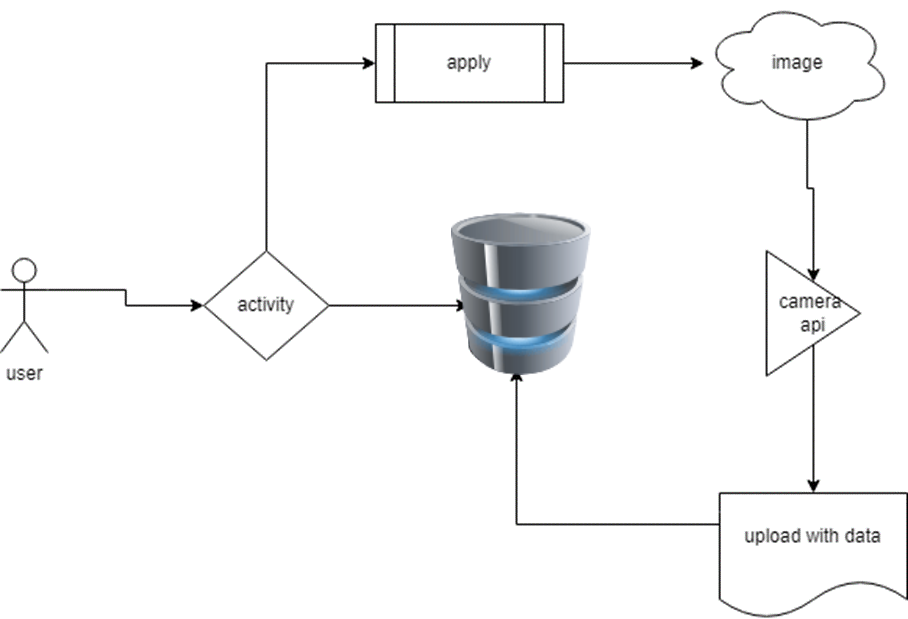
**4.1.2 STATE DIAGRAM**

****

**EXPLANATION:**

State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics. In our state diagram first propose a . For this in our component diagram first propose a data In this proposed method we are using Hash-Solomon Code Algorithm to encrypt the data.

**4.1.3 ACTIVITY DIAGRAM:**

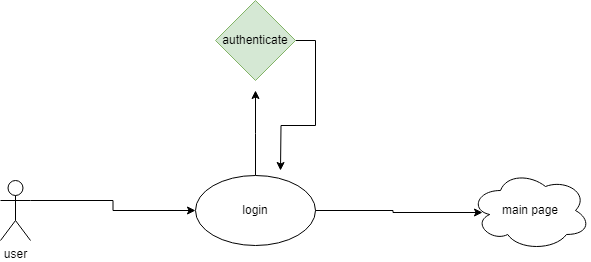


Activity diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. UML, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. UML activity diagrams could potentially model the internal logic of a complex operation. In many ways UML activity diagrams are the object-oriented equivalent of flow charts and data flow diagrams (DFDs)from structural development.

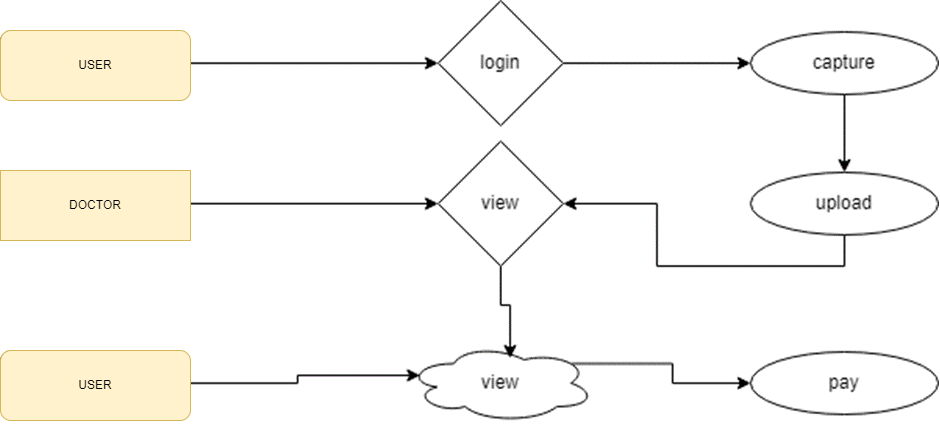
**4.1.7 DATAFLOW DIAGRAM**

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system. It differs from the flowchart as it shows the data flow instead of the control flow of the program. A data flow diagram can also be used for the visualization of data processing. The DFD is designed to show how a system is divided into smaller portions and to highlight the flow of data between those parts.

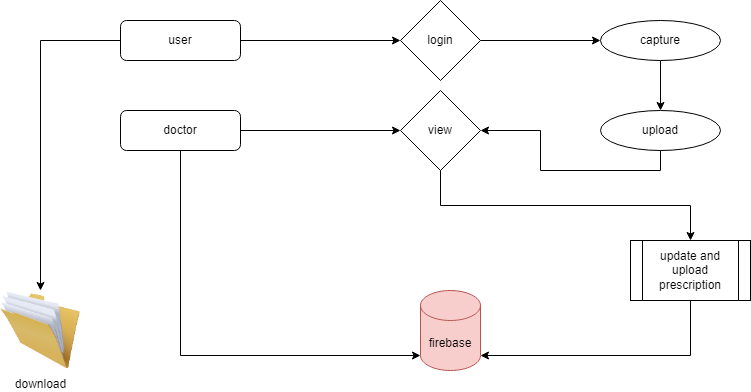
**LEVEL0:**

****

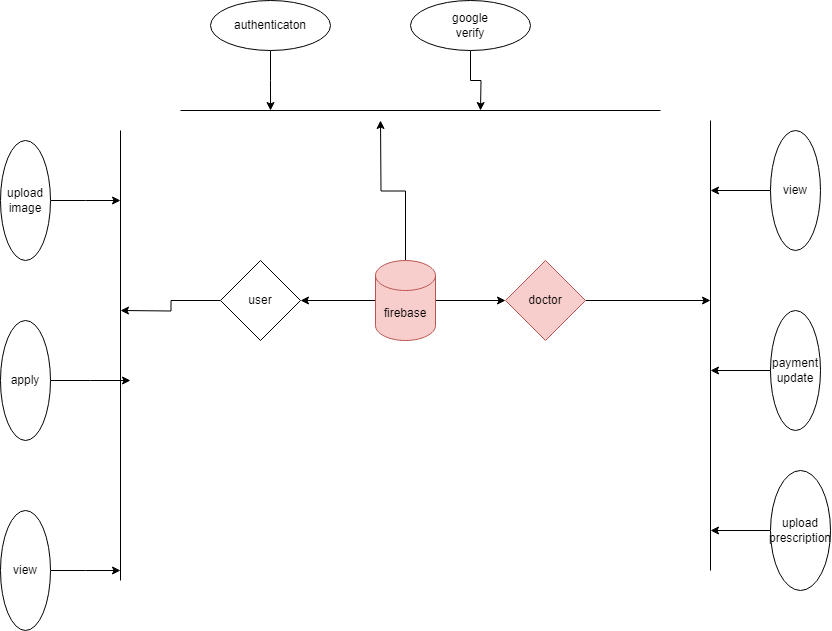
**LEVEL1:**

****

**LEVEL2:**

****

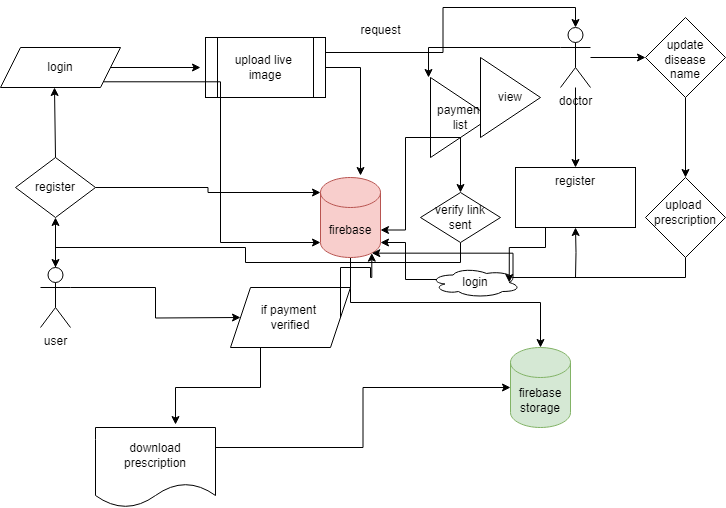
**4.1.8 ER-Diagram:**

****

**EXPLANATION:**

An entity is represented as rectangle in an ER diagram. For example: In the following ER diagram we have two entities Student and College and these two entities have many to one relationship as many students study in a single college. We will read more about relationships later, for now focus on entities.

**4.1.9 System Architecture:**

****

**EXPLANATION:**

The systems architect establishes the basic structure of the system, we propose a Hash code Solomon algorithm and we can put a small part of data in local machine and fog server in order to protect the privacy. Moreover, based on computational intelligence, this algorithm can compute the distribution proportion stored in cloud, fog, and local machine, respectively. Through the theoretical safety analysis and experimental evaluation, the feasibility of our scheme has been validated, which is really a powerful supplement to existing cloud storage scheme

**CHAPTER 5**

**DEVELOPMENT TOOLS**

**5.3 Layout:**

Earlier in client- server computing, each application had its own client program and it worked as a user interface and need to be installed on each user's personal computer. Most web applications use HTML/XHTML that are mostly supported by all the browsers and web pages are displayed to the client as static documents.

A web page can merely displays static content and it also lets the user navigate through the content, but a web application provides a more interactive experience.

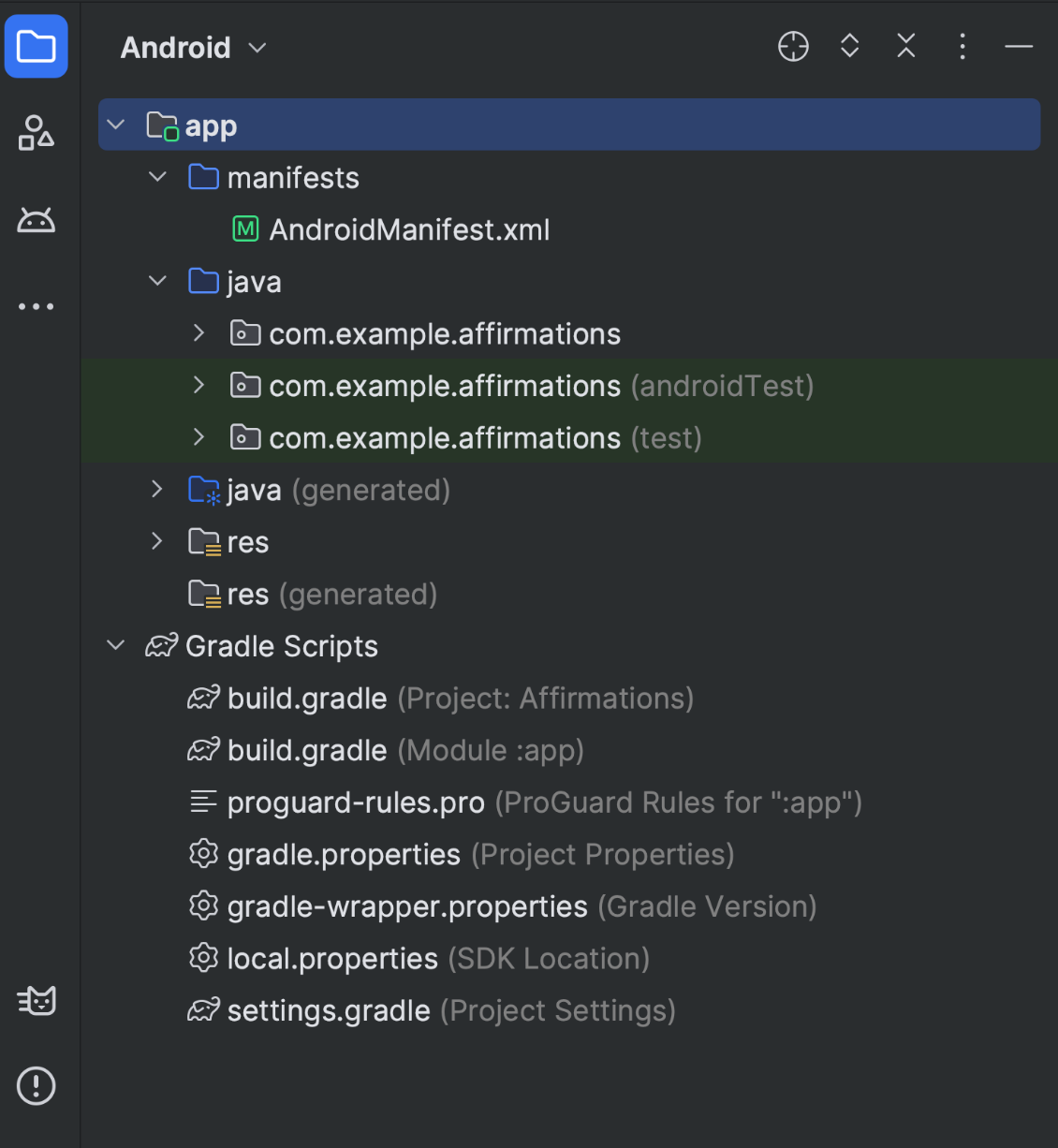
Any computer running Servlets or JSP needs to have a container. A container is nothing but a piece of software responsible for loading, executing and unloading the Servlets and JSP. While servlets can be used to extend the functionality of any Java- enabled server.

They are mostly used to extend web servers, and are efficient replacement for CGI scripts. CGI was one of the earliest and most prominent server side dynamic content solutions, so before going forward it is very important to know the difference between CGI and the Servlets.

**5.4 WIDGETS:**

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. It is derived from Standard Generalized Markup Language(SGML). Basically, the XML tags are not predefined in XML. We need to implement and define the tags in XML. XML tags define the data and used to store and organize data. It’s easily scalable and simple to develop. In Android, the XML is used to implement UI-related data, and it’s a lightweight markup language that doesn’t make layout heavy. XML only contains tags, while implementing they need to be just invoked.

## Project structure



**Figure 1.** Project files in Android project view.

Each project in Android Studio contains one or more modules with source code files and resource files. The types of modules include:

* Android app modules
* Library modules
* Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files. All the build files are visible at the top level, under **Gradle Scripts**.

Each app module contains the following folders:

* **manifests**: Contains the AndroidManifest.xml file.

For more information, see [Projects overview](https://developer.android.com/studio/projects).

## Gradle build system

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the [Android Gradle plugin](https://developer.android.com/build/releases/gradle-plugin). This build system runs as an integrated tool from the Android Studio menu and independently from the command line. You can use the features of the build system to do the following:

* Customize, configure, and extend the build process.
* Create multiple APKs for your app with different features, using the same project and modules.
* Reuse code and resources across source sets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core source files.

Android Studio build files are named build.gradle.kts if you use [Kotlin](https://kotlinlang.org/) (recommended) or build.gradle if you use [Groovy](http://groovy-lang.org/). They are plain text files that use the Kotlin or Groovy syntax to configure the build with elements provided by the Android Gradle plugin. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

**Note:** We might reference either the **build.gradle.kts** or **build.gradle** file alone in the documentation, but they're conceptually interchangeable. For example if you see **build.gradle.kts** but you use the Groovy DSL to configure your build, you can think of it as the **build.gradle** file (and the other way around).

To learn more about the build system and how to configure your build, see [Configure your build](https://developer.android.com/build).

### Build variants

The build system can help you create different versions of the same app from a single project. This is useful when you have both a free version and a paid version of your app or if you want to distribute multiple APKs for different device configurations on Google Play.

For more information about configuring build variants, see [Configure build variants](https://developer.android.com/build/build-variants).

**FLUTTER:**

Flutter is an open-source UI software development toolkit created by Google. It is used to build natively compiled applications for mobile, web, and desktop from a single codebase. Flutter was first introduced in 2015 and has gained popularity for its ability to enable developers to create high-performance, visually appealing applications with a flexible and expressive user interface.

Here are key aspects of Flutter:

1. **Single Codebase:**
   * Flutter allows developers to write code once and deploy it on multiple platforms, including iOS, Android, web, and desktop. This helps in reducing development time and effort.
2. **Dart Programming Language:**
   * Flutter apps are primarily written in the Dart programming language. Dart is a modern, object-oriented language developed by Google. It is designed for building web, mobile, and desktop applications.
3. **Widgets:**
   * Flutter uses a widget-based architecture for building user interfaces. Everything in Flutter is a widget, including structural elements like buttons and layout elements like rows and columns. Widgets are reusable and can be combined to create complex UIs.
4. **Hot Reload:**
   * One of Flutter's most praised features is hot reload, which allows developers to instantly see the effect of code changes during development without restarting the entire application. This accelerates the development process and makes it easier to experiment with UI changes.
5. **Rich Set of Pre-designed Widgets:**
   * Flutter comes with a rich set of pre-designed widgets for common UI elements, making it easier for developers to create consistent and attractive interfaces. Custom widgets can also be created to suit specific design requirements.
6. **Material Design and Cupertino Styles:**
   * Flutter provides widgets that implement the Material Design guidelines for Android apps and the Cupertino style for iOS apps. This allows developers to create platform-specific designs while sharing most of the code.
7. **Performance:**
   * Flutter compiles to native ARM code, providing high performance on both iOS and Android. The framework also includes a GPU-accelerated rendering engine, making it suitable for graphics-intensive applications.
8. **Integration with Native Features:**
   * Flutter allows developers to integrate with native features and APIs of the underlying operating systems, providing access to device-specific capabilities.
9. **Community and Ecosystem:**
   * Flutter has a growing and active community of developers who contribute to its ecosystem. There is a wide range of third-party packages and plugins available through Dart's package manager (pub.dev) that can be used to extend Flutter's functionality.
10. **Web and Desktop Support:**
    * Flutter extends its platform support beyond mobile devices. It includes experimental support for building web applications and desktop applications for Windows, macOS, and Linux.

**5.5 Conclusion**

xml and layout are gaining rapid acceptance as means to provide dynamic content on the Internet. With full access to the Java platform, running from the server in a secure manner, the application possibilities are almost limitless. When JSPs are used with Enterprise JavaBeans technology, e-commerce and database resources can be further enhanced to meet an enterprise's needs for web applications providing secure transactions in an open platform. J2EE technology as a whole makes it easy to develop, deploy and use web server applications instead of mingling with other technologies such as CGI and ASP. There are many tools for facilitating quick web software development and to easily convert existing server-side technologies to JSP and Servlets.

**CHAPTER 6 :**

**IMPLEMENTATION:**

import 'package:firebase\_core/firebase\_core.dart';

import 'package:flutter/material.dart';

import 'package:telemedicine/widgets/auth\_gate.dart';

import 'package:telemedicine/widgets/auth\_gate2.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp();

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: SplashScreen(),

);

}

}

class SplashScreen extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

body: Stack(

children: [

// Background Image

Image.asset(

'assets/images/wppp.jpg', // Replace with your image asset path

width: double.infinity,

height: double.infinity,

fit: BoxFit.cover,

),

// Fading Container

Container(

decoration: BoxDecoration(

gradient: LinearGradient(

begin: Alignment.topCenter,

end: Alignment.bottomCenter,

colors: [

Colors.transparent, // Fade from transparent

Colors.black.withOpacity(0.8), // to slightly opaque

],

),

),

child: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text(

'Time Health...',

style: TextStyle(

fontSize: 35,

color: Colors.indigo,

fontWeight: FontWeight.w800, // Make it bold

fontStyle: FontStyle.italic, // Add italic style

// Add additional styling as needed

),

),

SizedBox(height: 30),

ElevatedButton(

onPressed: () {

Navigator.pushReplacement(

context,

MaterialPageRoute(builder: (context) => AuthGate()),

);

},

child: Text('Continue as User'),

),

SizedBox(height: 10),

ElevatedButton(

onPressed: () {

Navigator.pushReplacement(

context,

MaterialPageRoute(builder: (context) => AuthGate2()),

);

},

child: Text('Continue as Doctor'),

),

],

),

),

),

],

),

);

}

}

class CustomerScreen extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Customer Page'),

),

body: Center(

child: Text('Welcome, Customer!'),

),

);

}

}

class AdminScreen extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Admin Page'),

),

body: Center(

child: Text('Welcome, Admin!'),

),

);

}

}

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:firebase\_core/firebase\_core.dart';

import 'package:firebase\_database/firebase\_database.dart';

import 'finallistpage.dart';

import 'fulllistviewpage.dart';

void main() async {

WidgetsFlutterBinding.ensureInitialized();

await Firebase.initializeApp();

runApp(vii());

}

class vii extends StatefulWidget {

const vii({super.key,});

@override

State<vii> createState() => \_usrequState();

}

class da{

final String cname;

final String caddress;

final String cmobile;

final String clocation;

final String ckey;

final String cnkey;

final String date;

final String members;

final String naddress;

final String ngoname;

final String nkey;

final String nmobile;

final String nlocation;

da(this.cname, this.date,this.caddress,this.cmobile,this.clocation,

this.ngoname, this.cnkey,this.nkey,this.nmobile,this.nlocation,this.members,

this.ckey,this.naddress

);

}

class \_usrequState extends State<vii> {

String authh = " ";

final DatabaseReference \_databaseReference =

FirebaseDatabase.instance.reference().child('shedule');

List<da> dataList = [];

@override

void initState() {

super.initState();

final FirebaseAuth \_auth = FirebaseAuth.instance;

User? user = \_auth.currentUser;

String? userId = user?.uid;

if (userId != null) {

setState(() {

authh = userId;

});

// Construct the path in the Realtime Database using the userId

// Fetch data from the Realtime Database

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

body: \_buildListViewWithDivider(),

);

}

Widget \_buildListViewWithDivider() {

return StreamBuilder(

stream: \_databaseReference

.orderByChild('pkey')

.equalTo(authh)

.onValue,

builder: (context, snapshot) {

if (snapshot.connectionState == ConnectionState.waiting) {

return Center(

child: CircularProgressIndicator(),

);

} else if (snapshot.hasError) {

return Center(

child: Text('Error loading requests.'),

);

} else if (!snapshot.hasData || snapshot.data?.snapshot.value == null) {

return Center(

child: Text('No Requests.'),

);

} else {

Map<String, dynamic> data =

Map<String, dynamic>.from(snapshot.data!.snapshot.value as Map);

List<String> itemIds = data.keys.toList();

return ListView.builder(

itemCount: itemIds.length \* 2 - 1, // Add dividers

itemBuilder: (context, index) {

if (index.isOdd) {

// Divider

return Divider();

} else {

// Item

int itemIndex = index ~/ 2;

String itemId = itemIds[itemIndex];

String mname = data[itemId]['mname']?.toString() ?? '';

String pname = data[itemId]['pname']?.toString() ?? '';

// List<String> dateArray = date.split(' ');

// Now dateArray contains the parts of the date split by space

// print('First part of the date: ${dateArray[0]}');

// String datefinal=dateArray[0];

String dosage = data[itemId]['dosage']?.toString() ?? '';

String category = data[itemId]['category']?.toString() ?? '';

String pkey = data[itemId]['pkey']?.toString() ?? '';

return Card(

elevation: 5,

margin: EdgeInsets.all(20),

color: Colors.blueGrey, // Set the card color here

child: Padding(

padding: EdgeInsets.all(10),

child: ListTile(

title: Text(

pname,

style: TextStyle(

color: Colors.black45, // Set title text color here

fontWeight: FontWeight.bold,fontSize: 18,

),

),

subtitle: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text(

'Medicine name: $mname',

style: TextStyle(color: Colors

.white), // Set date text color here

),

Text(

'Category: $category',

style: TextStyle(color: Colors

.white), // Set time text color here

),

Text(

'Dosage: $dosage',

style: TextStyle(color: Colors

.white), // Set address text color here

),

],

),

leading: ClipRRect(

borderRadius: BorderRadius.circular(30),

child: Image.asset(

'assets/icons/doctor.png',

width: 60,

height: 60,

fit: BoxFit.cover,

),

),

trailing: ElevatedButton(

onPressed: () {

// Navigator.push(

// context,

// MaterialPageRoute(

// builder: (context) =>

// FullListPage(

// dname: dname,

// address: address,

// location: location,

// des: des,

// hosname: hosname,

// pkey: pkey,

// udkey: udkey,

// mobile: mobile,

// status1: status1,

// status2: status2,

// experience: experience,

// specialist: specialist,

//

// ),

// ),

// );

},

style: ButtonStyle(

backgroundColor:

MaterialStateProperty.all<Color>(Colors.green),

),

child: Text(

'View',

style: TextStyle(color: CupertinoColors.white),

),

),

),

),

);

}

},

);

}

},

);

}

}import 'dart:io';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:firebase\_database/firebase\_database.dart';

import 'package:firebase\_storage/firebase\_storage.dart';

import 'package:flutter/material.dart';

import 'package:open\_file/open\_file.dart';

import 'package:path\_provider/path\_provider.dart';

import 'package:telemedicine/user/response.dart';

import 'package:url\_launcher/url\_launcher.dart';

class innerlist extends StatefulWidget {

final String dname;

final String udkey;

final String result;

final String status1;

final String status2;

final String des;

final String gpay;

final String amount;

final String disease;

final String specialist;

innerlist({

required this.dname,

required this.udkey,

required this.result,

required this.status1,

required this.status2,

required this.des,

required this.gpay,

required this.amount,

required this.disease,

required this.specialist,

});

@override

\_innerlistState createState() => \_innerlistState();

}

class \_innerlistState extends State<innerlist> {

late TextEditingController \_paymentController;

late TextEditingController \_extraController;

@override

void initState() {

super.initState();

\_paymentController = TextEditingController(

text: widget.amount); // Initialize the controller with amount

\_extraController = TextEditingController(

text: widget.gpay); // Initialize the controller with amount

}

Future<void> \_downloadFile() async {

try {

Reference storageRef = FirebaseStorage.instance

.ref()

.child('prescriptions')

.child(widget.udkey!);

// Get the download URL of the file

String downloadURL = await storageRef.getDownloadURL();

// Create a temporary directory to save the downloaded PDF

Directory tempDir = await getTemporaryDirectory();

File tempFile = File('${tempDir.path}/downloaded\_file.pdf');

// Download the PDF file to the temporary directory

await storageRef.writeToFile(tempFile);

// Open the downloaded PDF file

OpenFile.open(tempFile.path);

} catch (e) {

print('Error downloading file: $e');

}

}

Future<void> pay() async {

try {

FirebaseAuth \_auth = FirebaseAuth.instance;

User? user = \_auth.currentUser;

String? userId = user?.uid;

// Get a reference to the Firebase Realtime Database

DatabaseReference \_database = FirebaseDatabase.instance.reference();

// Update data in the Realtime Database for the specified udkey

await \_database.child('Prescription').child(widget.udkey!).update({

"status2": 'payed',

// Add other fields you want to update

});

ScaffoldMessenger.of(context).showSnackBar(

SnackBar(

content: Center(

child: Text(

'Payed Successfully',

style: TextStyle(

color: Colors.white,

fontSize: 16.0,

fontWeight: FontWeight.bold,

),

),

),

behavior: SnackBarBehavior.floating,

backgroundColor: Colors.green, // Use a color to indicate successful update

elevation: 4.0,

),

);

Navigator.push(

context,

MaterialPageRoute(

builder: (context) =>

response(

// Pass other fields as needed

),

),

);

setState(() {});

} catch (e) {

print('Error updating data: $e');

// Handle error updating data

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Details'),

),

backgroundColor: Colors.grey[300],

body:

Padding(

padding: const EdgeInsets.all(16.0),

child: Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: [

Text('Doctor Name: ${widget.dname},',style: TextStyle(fontSize: 20),),

Text('Result: ${widget.result}',style: TextStyle(fontSize: 20),),

Text('Status1: ${widget.status1}',style: TextStyle(fontSize: 20),),

Text('Status2: ${widget.status2}',style: TextStyle(fontSize: 20),),

Text('Description: ${widget.des}',style: TextStyle(fontSize: 20),),

Text('GPay: ${widget.gpay}',style: TextStyle(fontSize: 20),),

Text('Amount: ${widget.amount}',style: TextStyle(fontSize: 20),),

Text('Disease: ${widget.disease}',style: TextStyle(fontSize: 20),),

Text('Specialist: ${widget.specialist}',style: TextStyle(fontSize: 20),),

// Conditional rendering of the Pay button

SizedBox(height: 30,),

if (widget.status2 == 'request')

Container(

width: double.infinity,

child: ElevatedButton(

onPressed: () {

\_showPaymentDialog(context, widget.amount);

},

style: ElevatedButton.styleFrom(

primary: Colors.blue,

),

child: Text(

'Pay',

style: TextStyle(fontSize: 16, color: Colors.white),

),

),

),

],

),

),

// Align the download prescription button at the bottom center

floatingActionButton: widget.status2 == 'payed'

? SizedBox(

width: 380.0,

child: FloatingActionButton(

onPressed:

// Implement your download prescription logic here

\_downloadFile,

tooltip: 'Download Prescription',

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Icon(Icons.file\_download),

Text('Download Prescription'),

],

),

),

)

: Container(),

);

}

// Function to show the payment dialog

void \_showPaymentDialog(BuildContext context, String initialAmount) {

showDialog(

context: context,

builder: (BuildContext context) {

return AlertDialog(

title: Text('Payment Form'),

content: Column(

crossAxisAlignment: CrossAxisAlignment.start,

mainAxisSize: MainAxisSize.min,

children: [

TextFormField(

controller: \_paymentController,

decoration: InputDecoration(

labelText: 'Payment Amount',

),

),

SizedBox(height: 16),

TextFormField(

controller: \_extraController,

decoration: InputDecoration(

labelText: 'Gpay number',

),

),

],

),

actions: [

TextButton(

onPressed: () {

Navigator.pop(context); // Close the dialog

},

child: Text('Cancel'),

),

ElevatedButton(

onPressed: () {

// Implement your payment logic here

// Access the payment amount and extra information

String enteredAmount = \_paymentController.text;

String enteredExtra = \_extraController.text;

print('Entered Amount: $enteredAmount');

print('Entered Extra Information: $enteredExtra');

pay();

Navigator.pop(context); // Close the dialog

},

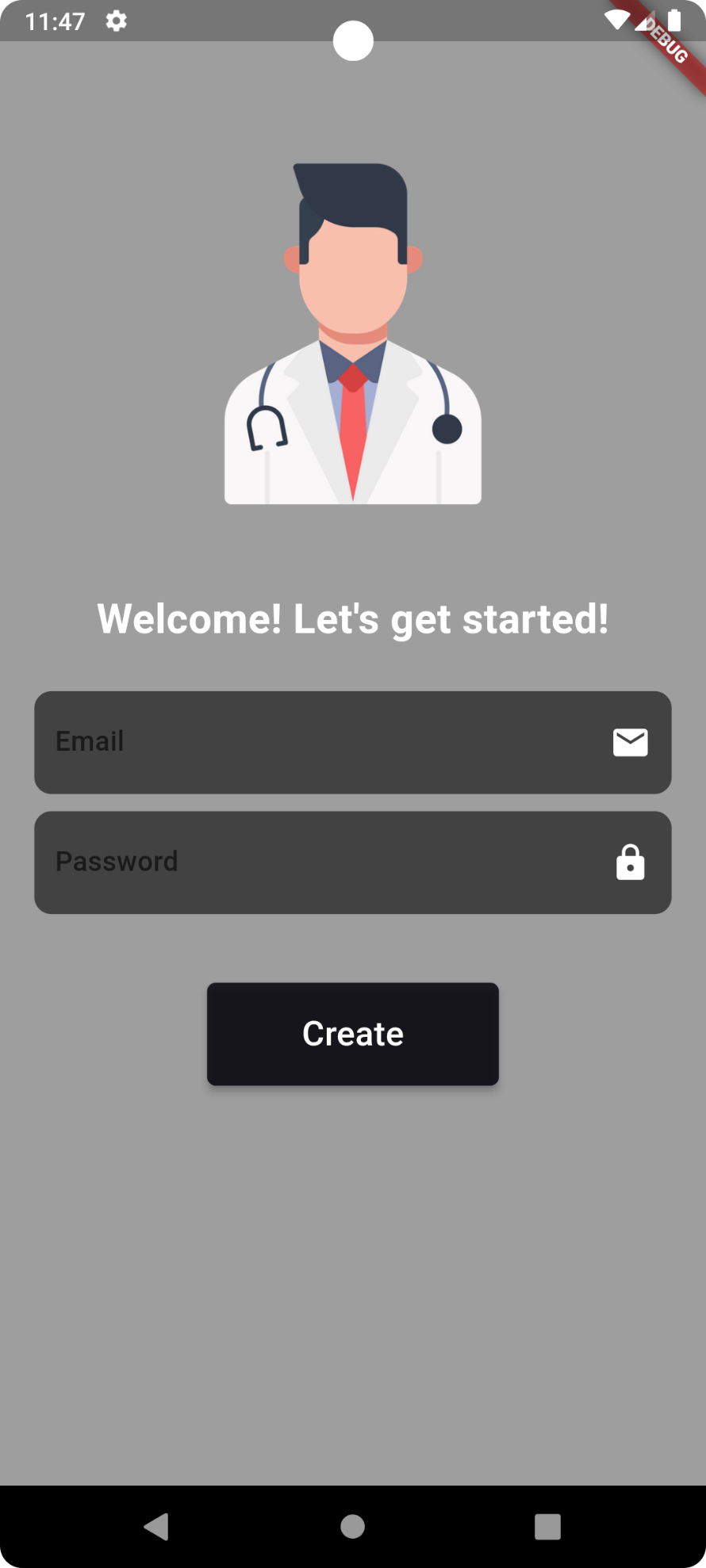
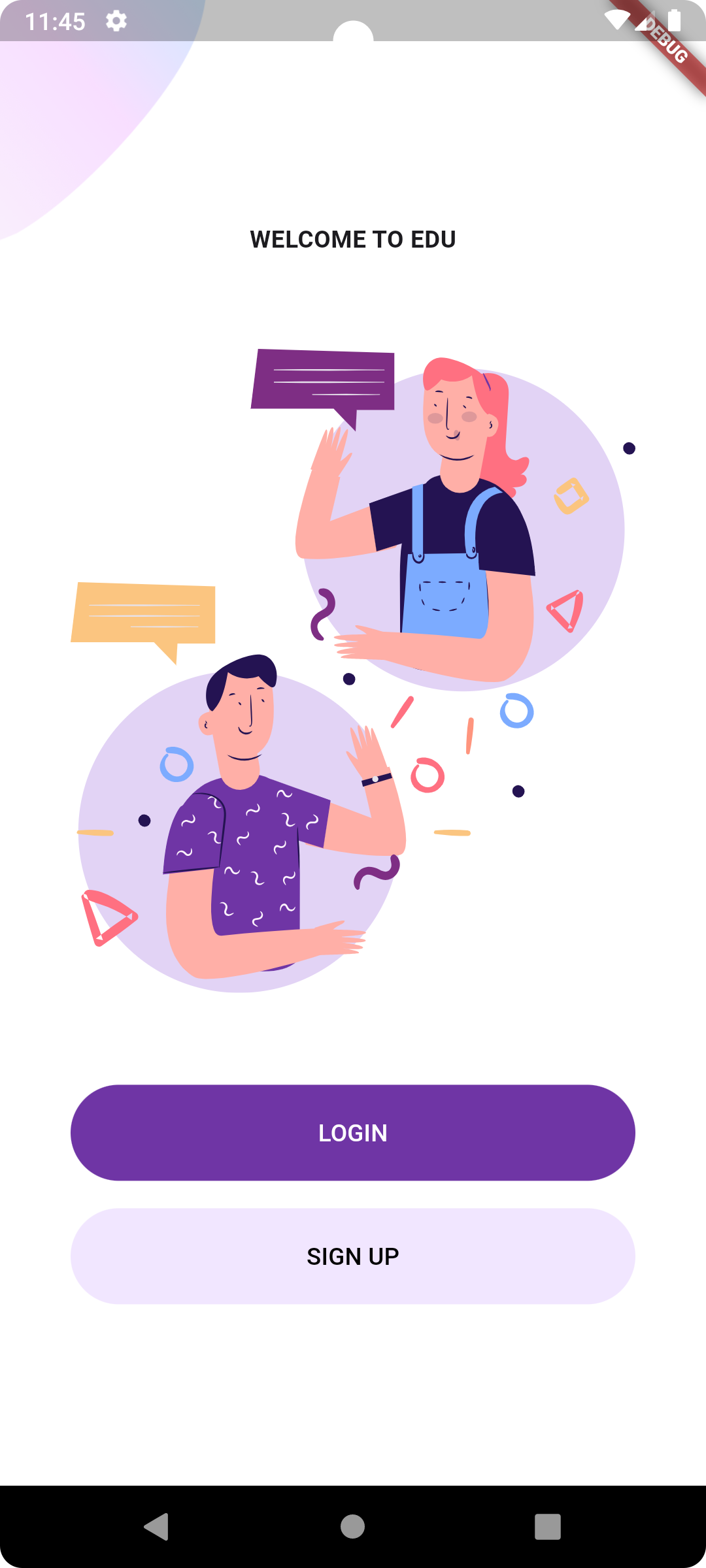
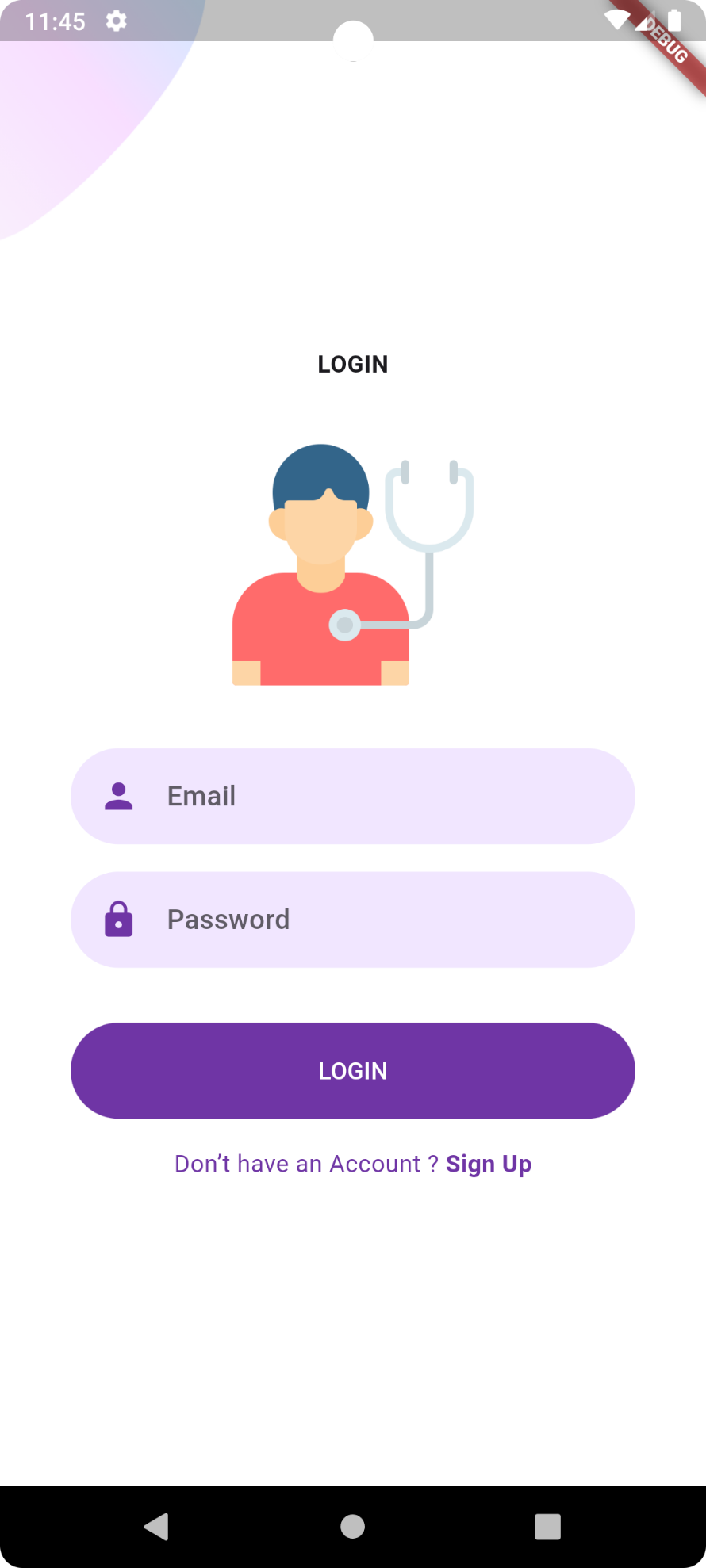
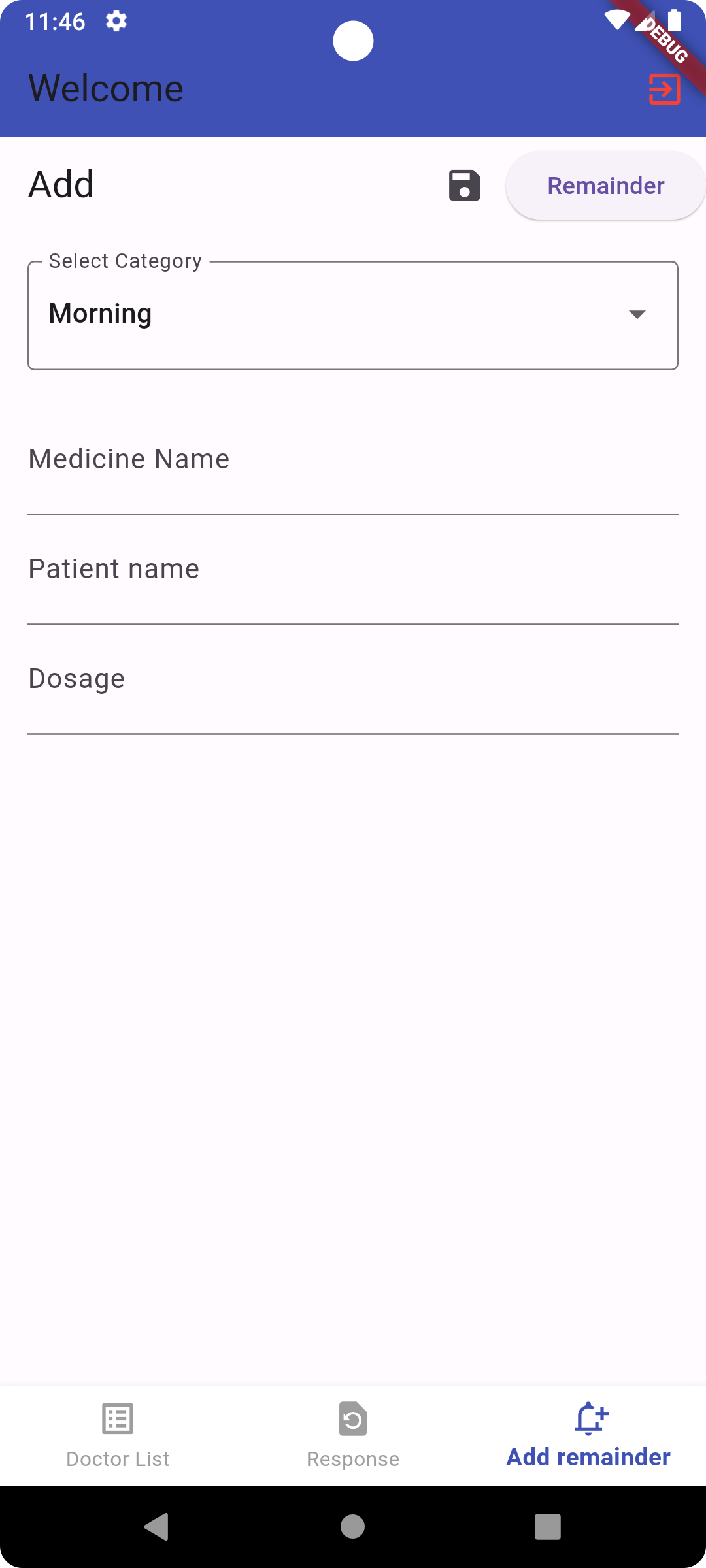
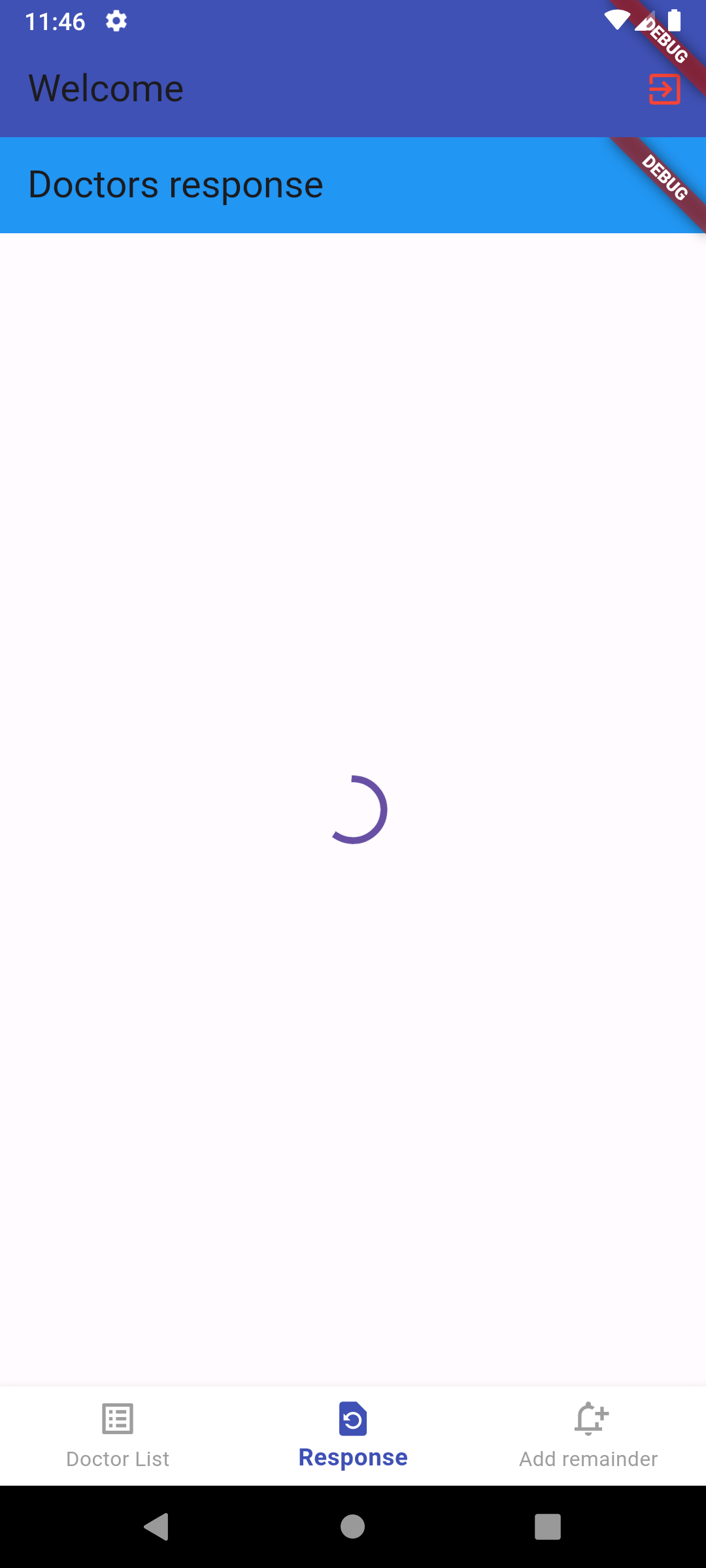
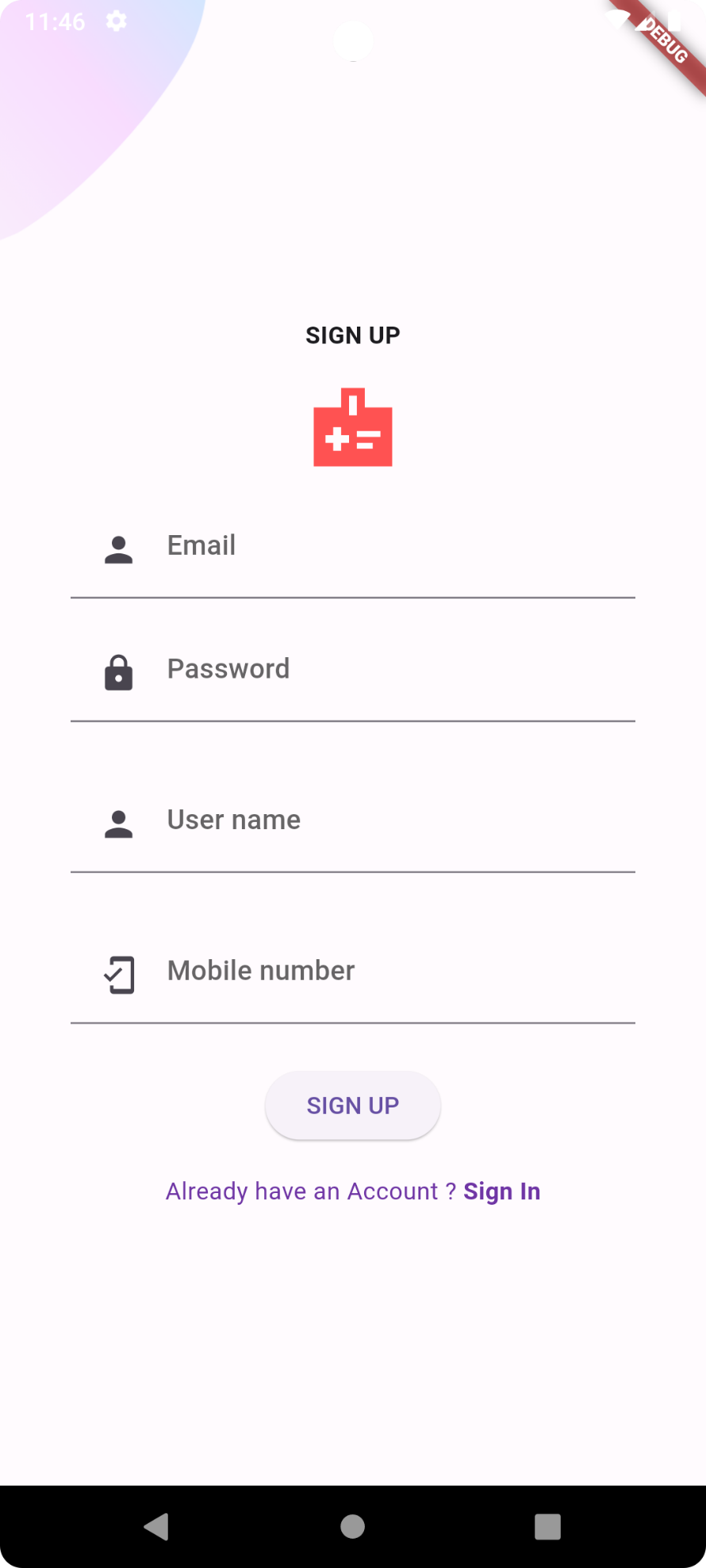
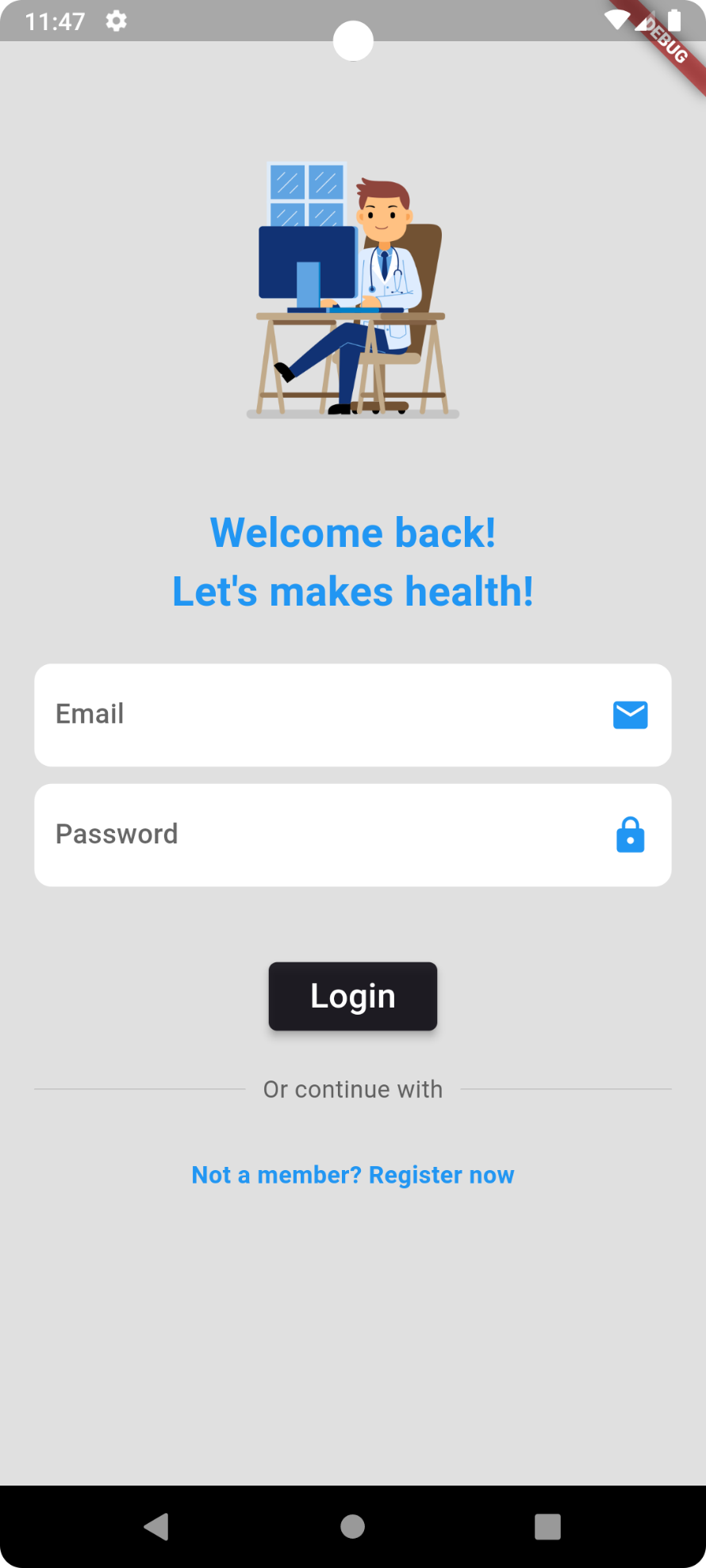
child: Text('Pay'),

),

**CHAPTER 7**

**SNAPSHOTS:**





**CHAPTER 8**

**FIREBASE:**

**Firebase** is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google’s infrastructure.

Firebase is categorized as a [NoSQL](https://www.educative.io/edpresso/whats-the-difference-betweensql-and-nosql) database program, which stores data in JSON-like documents.



In Firebase, a document is a set of key-value pairs defined by a schema. A group of documents makes up a collection.

## Key Features

### 1. Authentication

It supports authentication using passwords, phone numbers, Google, Facebook, Twitter, and more. The Firebase Authentication (SDK) can be used to manually integrate one or more sign-in methods into an app.

### 2. Realtime database

Data is synced across all clients in realtime and remains available even when an app goes offline.

### 3. Hosting

Firebase Hosting provides fast hosting for a web app; content is cached into content delivery networks worldwide.

### 4. Test lab

The application is tested on virtual and physical devices located in Google’s data centers.

### 5. Notifications

Notifications can be sent with firebase with no additional coding.

Users can get started with firebase for free; more details can be found on the [official website](https://firebase.google.com/).

**SOFTWARE TESTING:**

**8.1. FEASIBILITY STUDY**

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of the existing business or proposed venture, opportunities and threats as presented by the environment, the resources required to carry through, and ultimately the prospects for success.

In its simplest term, the two criteria to judge feasibility are cost required and value to be attained. As such, a well-designed feasibility study should provide a historical background of the business or project, description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, feasibility studies precede technical development and project implementation.

They are 3 types of Feasibility

• Economical feasibility

• Technical feasibility

• Operational feasibility

**8.1.1. ECONOMICAL FEASIBILITY**

The assessment is based on an outline design of system requirements in terms of Input, Processes, Output, Fields, Programs, and Procedures. This can be quantified in terms of volumes of data, trends, frequency of updating, etc. in order to estimate whether the new system will perform adequately or not.

**8.1.2. TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources.

**8.1.3. OPERATIONAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

**8.2. SYSTEM TESTING**

The software, which has been developed, has to be tested to prove its validity. Testing is considered to be the least creative phase of the whole cycle of system design. In the real sense it is the phase, which helps to bring out the creativity of the other phases makes it shine.

**8.2.1. VARIOUS LEVELS OF TESTING**

1. White Box Testing

2. Black Box Testing

3. Unit Testing

4. Functional Testing

5. Performance Testing

6. Integration Testing

7. Validation Testing

8. System Testing

9. Output Testing

10. User Acceptance Testing

**8.2.1.1. WHITE BOX TESTING**

White-box testing, sometimes called glass-box, is a test case design method that uses the control structure of the procedural design to derive test cases. Using White Box testing methods, we can derive test cases that

• Guarantee that all independent paths within a module have been exercised at least once

• Exercise all logical decisions on their true and false sides.

• Execute all loops at their boundaries and within their operational bounds.

• Exercise internal data structures to assure their validity.

**8.2.1.2. BLACK BOX TESTING**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

**8.2.1.3. UNIT TESTING**

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process.

Unit testing is software verification and validation method in which the individual units of source code are tested fit for use. A unit is the smallest testable part of an application. In this testing, each class is tested to be working satisfactorily.

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration.

**8.2.1.4. FUNCTIONAL TESTING**

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional Testing usually describes what the system does. Functional testing differs from system testing in that functional testing "verifies a program by checking it against ... design document(s) or specification(s)", while system testing "validate a program by checking it against the published user or system requirements" (Kane, Falk, Nguyen 1999, p. 52). Functional testing typically involves five steps. The identification of functions that the software is expected to perform

1. The creation of input data based on the function's specifications

2. The determination of output based on the function's specifications

3. The execution of the test case

4. The comparison of actual and expected outputs.

**8.2.1.5. PERFORMANCE TESTING**

In general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Performance testing is a subset of performance engineering, an emerging computer science practice which strives to build performance into the implementation, design and architecture of a system.

**8.2.1.6. INTEGRATION TESTING**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when put together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready. All the errors found in the system are corrected for the next phase.

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctlyfor example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing.

**8.2.1.7. VALIDATION TESTING**

Verification and Validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it full fills its intended purpose. These are critical components of a quality management system such as ISO 9000. The words "verification" and "validation" are sometimes preceded with "Independent" (or IV&V), indicating that the verification and validation is to be performed by a disinterested third party.

It is sometimes said that validation can be expressed by the query "Are you building the right thing?" and verification by "Are you building it right?". In practice, the usage of these terms varies. Sometimes they are even used interchangeably.

**8.2.1.8. SYSTEM TESTING**

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behaviour and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification.

**8.2.1.9. OUTPUT TESTING**

After performing the validation testing, next step is output testing of the proposed system since no system could be useful if it does not produce the required output generated or considered in to two ways. One is on screen and another is printed format. The output comes as the specified requirements by the user. Hence output testing does not result in any correction in the system.

**8.2.1.10. USER ACCEPTANCE TESTING**

User acceptance of a system is the factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required.

• Input screen design.

• Output screen design.

• Online message to guide user.

• Format of the ad-hoc reports and other outputs.

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using the test data. While testing the system by using test data errors are again uncovered and correct.

**9.3 FUTURE ENHANCEMENTS:**

Feature enhancements for this application aim to significantly improve the user experience and service delivery for both clients and specialists. Key improvements include the integration of teleconsultation capabilities, allowing real-time video interactions that personalize care. Advanced data security measures, such as encryption and multi-factor authentication, will be implemented to protect sensitive health information. The user interface will be refined for better navigation, complemented by a notification system to keep clients informed about appointments and treatment updates. Additionally, a feedback and rating system will empower clients to share their experiences, fostering accountability among specialists. An analytics dashboard will provide specialists with insights into client interactions and treatment outcomes, facilitating continuous improvement. Multi-language support will ensure accessibility for a diverse user base, while educational resources will be added to help clients better understand their skin health. Together, these enhancements will create a more effective and supportive platform for dermatological care.

**CONCLUSION:**

The client prepared to login in these application. in essential page the game plan of ppg assessment using camera programming connection point the working principle of spot a finger in camera remember for streak light flash and take review to assessment of picture treatment of red pixels of circulation system to recognize the derived heart beat. the another communication is live acknowledgment of heart beat. the circulatory system of ecg is by and large show and scrutinizing show live scrutinizing of circulation system assessment. they show the ordinary truly check out at results with worth of area.