



**Islington college**  
(इस्लिङ्टन कलेज)

**Module Code & Module Title**  
**CS6P05NI Final Year Project**

**Assessment Weightage & Type**

**25% FYP Interim Report**

**Semester**

**2021 Autumn**

**Project Title: Can-Dect**

**Student Name: Prince Panjiyar**

**London Met ID: 19033703**

**College ID: np01cp4s200049**

**Internal Supervisor: Sunil Raut Kshetri**

**External Supervisor: Ashutosh Chauhan**

**Assignment Due Date: 15<sup>th</sup> December, 2021**

**Assignment Submission Date: 15<sup>th</sup> December, 2021**

**Word Count (Where Required): 3440**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.*

## Abstract

*This report exhibits the development progress of Can-Dect project. The Can-Dect is a hospital management system that manages the appointment as well as provides doctor's assistance with the detection of breast cancer based on AI model with the help of histological slides images. This project aims to reduce the gathering of patient flow and un-mannered process of booking appointment that creates conflict and problem to both staff and patient. It provides an online medium to manage the hospital. It also helps doctors to provide medication to patient without calling the patient to the hospital.*

## Contents

Introduction: .....	1
Problem scenario:.....	1
Project as a solution: .....	2
Aims and Objectives:.....	2
Project Features: .....	3
Mobile Application: .....	3
Desktop application:.....	3
Web-Application: .....	3
Report Structure: .....	4
Background: .....	4
Development: .....	4
Analysis of the progress:.....	4
Future work: .....	4
Background/Literature review:.....	5
Technologies Used: .....	5
Other similar apps:.....	6
Medi-Flow: .....	6
Cogent Health: .....	8
Analysis of Similar Project with my Project: .....	9
Development: .....	10
Methodology: .....	10
Considered Methodology: .....	10
Selected Methodology:.....	13
Gantt Chart: .....	16
Use Case Diagram: .....	17
High Level of Use Case: .....	18
System Architecture:.....	22
Mobile UI Design: .....	23
Desktop Application Development: .....	29
Mobile Application Development:.....	35
Analysis of Progress:.....	47
Progress Table: .....	47

Progress Review:.....	48
Current scenario of Progress: .....	48
Action Plan: .....	48
Future Work: .....	49
Bibliography .....	50
Appendix: .....	51
UI design: .....	51
Desktop Application Code: .....	59
Login Page: .....	59
Home Page: .....	65
Menu Section: .....	76
Appointment Page:.....	79
Report Page: .....	85
Medicine Page: .....	95
Notify Page: .....	104
Mobile Application Code: .....	112
Login Page .....	112
Signup Page: .....	119
Verify Email Page:.....	125
User Details Page: .....	130
Home Page: .....	139
Report Page: .....	166
Medicine Page: .....	170
Booking Detail Page:.....	178
Change Password Page: .....	184
User Data Details Page:.....	190
Forget Password Page:.....	197

## Table of Figures:

Figure 1 Medi-Flow.....	7
Figure 2 Cogent Health .....	8
Figure 3 Waterfall Model .....	11
Figure 4 Kanban Board .....	15
Figure 5 Can-Dect Gantt Chart .....	16
Figure 6 Use Case Diagram.....	17
Figure 7 System Architecture .....	22

## Introduction:

Breast cancer is a disorder in which the cells of the breast proliferate uncontrollably. There are various types of breast cancer. The type of breast cancer is determined by which cells in the breast develop into cancer. Breast cancer can develop in a variety of locations within the breast. A breast is composed of three major components: lobules, ducts, and connective tissue. Invasive ductal carcinoma is the most frequent type of breast cancer. The cancer cells start in the ducts and subsequently spread to other regions of the breast tissue. Invasive cancer cells can also spread to other parts of the body, a process known as metastasis. Invasive lobular carcinoma Cancer cells develop in the lobules and subsequently move to nearby breast tissues. These invasive cancer cells have the potential to spread to other places of the body as well (CDC, 2021).

Can-Dect is a hospital management system based on artificial intelligence for breast cancer screening. It allows clients to schedule appointments for the many services offered by the hospital and assists doctors in detecting breast cancer using image processing and CNN on histological slides or data from histological slides.

## Problem scenario:

Breast cancer has steadily climbed over the last ten years, and it is now the most common cancer in Nepalese women as well. The age-adjusted rate can reach up to 25.8 per 100,000 women, with a death rate of 12.7 per 100,000 women. Breast cancer accounts for 9.6 percent of all cancer patients, according to the Globocan 2020 report. The biggest issue is that patients have to fly to big cities for cancer tests without a valid appointment, which causes a crowd in the hospital and forces them to spend more time in cities, which is expensive. Lack of well-trained and highly qualified doctors to identify cancer cells results in erroneous identification of cancer and a panic scenario for patients, as well as late discovery causing life-threatening dangers. Along with the increase in cases, the crowd for proper appointment and checkup of the diseases at hospital has been shifted to unmannered way of creating crowd for the ticket and this increases the chances of COVID as well.

### Project as a solution:

The app allows patients to schedule an appointment and travel for the test on the scheduled date. It also enables the patient to receive results and medical prescriptions directly from the doctor via the app. It assists doctors in reviewing cancer histology slides, and the AI system assists them in correctly detecting the malignancy. The app also allows the doctor to send notifications to patients regarding their follow-up appointments as well as any new information about their illness.

### Aims and Objectives:

The app's primary goal is to assist in the proper management of patient flow in the hospital. It enables faraway patients to make appointments online at their leisure and provides better cancer detection results. It also enables doctors to treat patients appropriately based on their current status and to deliver timely updates on their condition and prescribed medications. The following are the primary goals:

#### Smart goal:

- Identify, manage, and develop the app's core scope.
- Choosing a methodology and a plan to carry out professional and well-mannered app development.
- Look for a suitable data set for training the machine learning model to appropriately identify cancer.

#### Personal goals:

- To be well-versed in flutter application programming;
- To be well-versed in the MySQL database system and database design.
- To have frequently knowledge of Django, python and machine learning for the development of backend and machine learning model to predict highly accurate breast cancer.
- Proper knowledge of API integration with flutter is required.

System objectives:

- To properly manage and keep log of patient's details.
- Manage proper flow of appointment with the doctor.

Project Features:

Mobile Application:

- Made for Patients to sign up and login.
- Can book appointment selecting their preferable date and time.
- Can check the reports and medication.
- Receives notification from the doctor about their progress.
- Can check their bookings as well as payment made by them.

Desktop application:

- Doctors can Login.
- Can create report as well as medication according to the reports result.
- Can notify patients about their health progress.
- Can check their appointments.
- Can Use AI to detect breast cancer.

Web-Application:

- Admin Dashboard
- Can add Doctor and services with price.
- Can check daily appointment details.



## Report Structure:

### Background:

The background of the project provides a broad view of the need, requirement and features of the app to provide better facility and solve the domain problem.

### Development:

Development explains how the project is developed and the pace with proper planning to successfully make progress in the development of the system. It explains how the whole project is split into pieces to categorize the project under different section and make development in a systematic and mannered way.

### Analysis of the progress:

It deals with the overview of the progress made in the development of the project with respect to Gantt chart time-graph. It helps to analyze the need of time management and increase in process of development when lagging behind the timeline.

### Future work:

It deals with the works remaining to be done with the time and the pending works that are lagging behind the timeline. It also deals with proper management and plan making to complete the project on time with the required features promised.

## Background/Literature review:

The hospital system software includes features that integrate and simplify healthcare workers' work as well as their contact with patients. There is always a diverse range of features that may be incorporated into the system. Furthermore, and most importantly, they are designed to simplify numerous operations that fulfill the demands of all users. The hospital management system feature set is focused on creating a positive experience for patients, employees, and hospital administrators. Although their expectations may appear to differ, they are nonetheless covered by components of the hospital information system. Quality and security are still the most important factors in the medical sector. It is also noted for making frequent and quick modifications to increase the efficiency of medical services and patient happiness.

It may do a variety of jobs depending on the features of the hospital management system software. It aids in the development and implementation of policies, ensuring staff communication and coordination, automating routine processes, designing patient-oriented workflows, advertising services, managing human and financial resources, and ensuring an unbroken supply chain. The components of a hospital information system can be selected and integrated to produce a general system that fits the demands and norms of the healthcare business, as well as quality standards. Security is one of the most important criteria of the clinic management system. All medical records must be safeguarded and only available to those who have been granted access. In order to preserve sensitive data, convenient and informative interfaces should correspond to their duties and responsibilities (Existek, 2021).

## Technologies Used:

For the development of this project, Flutter is used to create mobile application (for users) and desktop-based application (for doctors) and HTML, CSS for web-based application (for admin to conduct CRUD operations). Django is used as backend and its rest framework is used as API to connect backend with the mobile and desktop-based application whereas template rendering of Django is used for web-based application.

Figma is used for the basic design of the application. CNN algorithm is used for the detection of breast cancer and will be accessible by the doctors only. MySQL database is used to store data of the management system. IDE used are Android Studio for the emulator purpose, VS Code for flutter development and PyCharm Professional for the Django backend development. Kaggle dataset of breast cancer histological slides image for the training of AI to detect cancer.

Other similar apps:

Medi-Flow:

It is a Nepal based hospital management system that has features of searching hospital and book their appointment for different kind of medical services with online payment system.

- It has similar way of booking the Appointment for different services provided by the hospitals.
- Online report generation system.
- Online medication provided by the doctor.

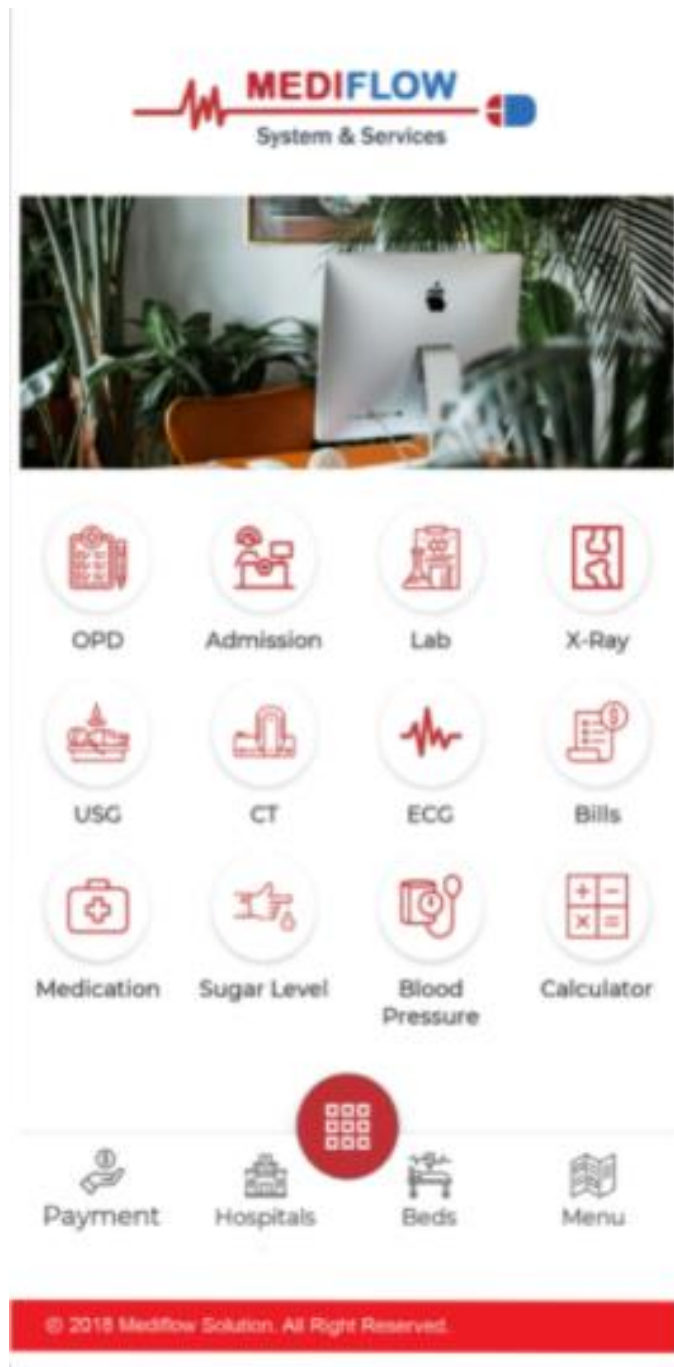


Figure 1 Medi-Flow

### Cogent Health:

Cogent Health Pvt Ltd., a sister organization of ISO certified F1 Soft International Pvt. Ltd. has a history of more than 10 years and works in the field of health informatics. Cogent Health is a popular EMR (Electronic Medical Records) SISH (Hospital Integrated Service) with integrated modules for CDSS (Clinical Decision Support System), CPOE (Computerized Physician Order Entry), and Data Mining. Software) has been developed. In addition, SISH specializes in academic research and statistical programs for the integration of healthcare systems and financial technologies.

- Tracks the reports and provide medicines based on the report by the doctor.
- Appointment through e-payment system.

The screenshot displays the Cogent Health mobile application interface. At the top, there is a red header bar with a back arrow, the text "Cogent Health", and a menu icon. Below the header, the "Patient Information" section shows "Full Name: NANU NEPALI", "Age: 37 year", "Sex: Female", and "Bed No: ortho-213". A table with two columns, "EXAMINATIONS" and "MEDICINES", follows. The "EXAMINATIONS" column lists "Diastolic BP", "O2 Saturation", "Pulse Rate", "Respiratory Rate", "Systolic BP", and "Temperature(F)". The "MEDICINES" column shows "0" for each row. A red "SUBMIT" button is at the bottom of the form. The status bar at the top right shows the time as 2:35.

EXAMINATIONS	MEDICINES
Diastolic BP	0
O2 Saturation	0
Pulse Rate	0
Respiratory Rate	0
Systolic BP	0
Temperature(F)	0

Figure 2 Cogent Health

### Analysis of Similar Project with my Project:

Medi-Flow and Cogent Health are the currently well-known hospital management system working in Nepal. They provide similar services like online appointment booking, online report and medication. They store and keep track on the patient's condition and provides updated medications. Both the app manages and maintains the internal records of the hospital. Both app keeps record of the payments done by the patients and manages the total in-flow of money from patient to the hospital.

In comparison with the project being developed, the project as similar way of booking the appointment but with the option of selecting the doctor specialized and available at the hospital. The app also displays the price and lets users to select their preferable date and time for the appointment with the options of e-payment. The app provides system to let doctors notify the patients about their current scenario and progress of recovery and provide updated medication to the patient. The billing records as well as managed appointment records are stored and managed with the help of the app. It also provides assistance to the doctors to provide a clear idea and detection of the breast cancer by analysis through AI predictions based on pre-trained models. The app also provides doctors to view their appointment schedule and allows to update the report of the patient with the required medicines. The app allows patients to download their report as well. The admin dashboard allows to add the Doctor and the services provided by the hospital. The admin is allowed to check all the appointments made and the remove the doctor from the system. The mobile app is made for the patients to perform simple tasks like appointment whereas desktop app is made for the doctors to update and check the reports of the patients and use AI for the breast cancer detection in the histological slides of lobules, duct or connective tissues. The web app is made for the admin to track the uses of the app by the doctors. The mobile app verifies the user by validating the email with a 4-digit verification code and after validating the email it allows users to book appointment and properly use the mobile app.

Thus, all these app manages and control the in-flow of patient to the hospital in a well mannered arranged and systematic way and eradicates the over gathering of the patients into the premises of hospital for their prescribed appointments.

## Development:

### Methodology:

#### Considered Methodology:

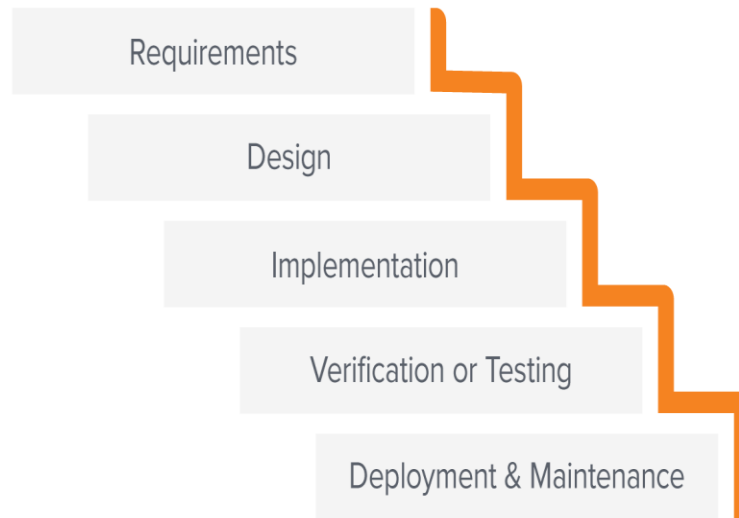
##### 1. Waterfall Methodology:

The Waterfall technique, also known as the Waterfall model, is a sequential development process that flows like a waterfall through all stages of a project (analysis, design, development, and testing), with each step entirely winding up before moving on to the next phase. The Waterfall method's success is determined by the amount and quality of work done on the front end, which includes documenting everything in advance, including the user interface, user stories, and all feature variants and outcomes. With the bulk of the research completed ahead of time, estimations of the time required for each need are more accurate, allowing for a more predictable release date (workfront., 2021).

#### Disadvantages:

- With this historical approach, projects may take longer to complete than with an iterative strategy, such as the Agile technique.
- Clients frequently don't completely understand what they want up front, which opens the door to demands for revisions and new features later in the process, when they're more difficult to implement.
- Clients are excluded from the design and implementation phases.

## The Waterfall Method



*Figure 3 Waterfall Model*



## 2. Prototyping Model:

The prototyping model is a systems development process in which a prototype is produced, tested, and then tweaked until an acceptable output is obtained from which the entire system or product may be developed. This technique works well in situations where not all of the project needs are known in advance.

### Disadvantages:

The primary downside of this technique is that it is more expensive in terms of both time and money when compared to other development methods, such as the spiral or Waterfall model. Because prototypes are often abandoned, some businesses may not see the benefit in using this method. Inviting client feedback so early in the development process may also generate issues. One issue is that there may be an overwhelming number of modification requests that are difficult to satisfy (Lewis, 2019).

### Selected Methodology:

#### 3. Kanban:

The Kanban Method is a technique for creating, managing, and improving flow systems for knowledge work. Additionally, the strategy enables firms to begin with their present workflow and achieve evolutionary transformation. They can accomplish this by visualizing their work flow, limiting work in progress (WIP), and stopping starting and starting again. The Kanban Method is named after the use of kanban – visual signaling systems – to control work in progress for intangible work items. Kanban can be utilized in any knowledge work scenario, but it is especially useful when work arrives in an unpredictable manner and/or when you want to deploy work as soon as it is ready, rather than waiting for other work items to finish. To properly track and build the software, the Kanban board comprises six major components.

- Visual Signals: The first and most crucial stage in adopting and executing the Kanban Method is the use of visual signals. You must represent the process phases that you presently use to provide your work or services - either on a physical board or an electronic Kanban Board.
- Columns are organized into three categories: 'To-Do,' 'Doing,' and 'Done.' As the development process progresses, tasks are dragged from one column to the next, from the leftmost column (future tasks) to the rightmost column (finished tasks).
- Work In Progress (WIP) defines the maximum number of requirements that the product team can handle successfully at each step of their Kanban process. WIP limitations can be an effective technique for teams to control their work and finish assignments more rapidly.
- Manage Flow: A Kanban system supports in flow management by displaying the many phases of the workflow as well as the current condition of work in each stage. Depending on how well the workflow is

built and WIP Limitations are set, you will see either a fluid flow within WIP limits or work piling up when something becomes blocked and begins to hold up capacity. All of this determines how quickly work progresses from the start to the end of the workflow.

- **Make Process Policies Explicit:** As part of visualizing your process, it is appropriate to define and show explicitly your policies (process rules or guidelines) for how you accomplish the tasks you do. You provide a common foundation for all participants to understand how to conduct any sort of work in the system by creating explicit process guidelines. Policies can be defined at three levels: the board, the swim lane, and the column. They can be a checklist of processes to be followed for each work item type, entry-exit criteria for each column, or anything else that helps team members manage the flow of work on the board more effectively (Digite, 2021).
- **Implement Feedback loops:** Feedback loops helps to address changes that needs to be done. The Lifecycle section describes the feedback loops used in Kanban.

Benefits that Kanban provides are:

- increased adaptability
- Continuous enhancement
- Enhanced collaboration
- Employee engagement
- Streamlined workflow
- improved inventory management
- enhanced quality control

### Reasons to choose Kanban:

- Kanban boards, both physical and digital, assists in visualizing work.
- Kanban is simple to implement; simply begin with what you have.
- Limiting your work in progress allows you to become more efficient.
- Kanban concepts and practices provide an evolutionary path to agility without interrupting existing operations.

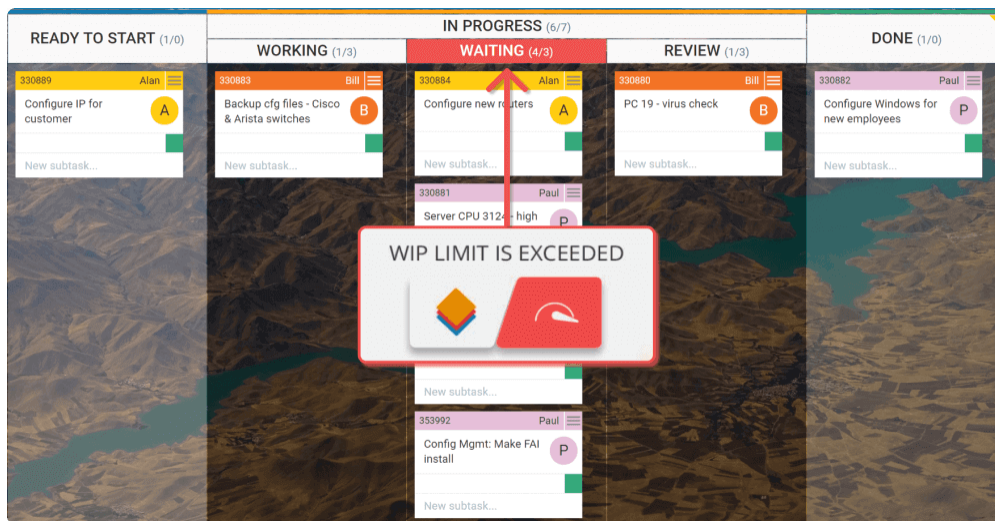


Figure 4 Kanban Board

## Gantt Chart:

A Gantt chart, which is widely used in project management, is one of the most popular and practical methods of displaying activities (tasks or events) against time. A list of the activities is shown on the left side of the chart, and a time scale is shown at the top. Each activity is represented as a bar, with the location and length of the bar reflecting the activity's start, duration, and finish dates. Gantt charts are often used to track project timelines. It is helpful to be able to provide extra information about the project's many tasks or stages, such as how the activities connect to one another, how far each task has gone, what resources are being utilized for each work, and so on (Gantt, 2021).

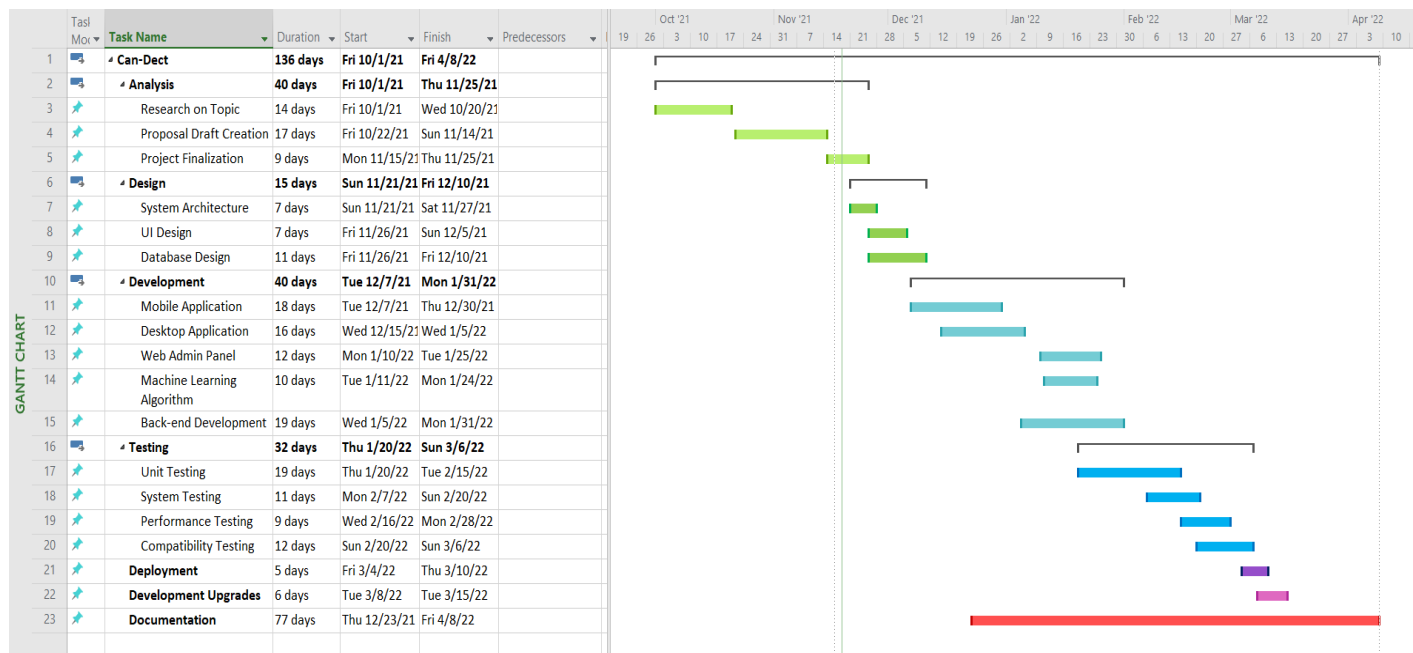


Figure 5 Can-Dect Gantt Chart

### Use Case Diagram:

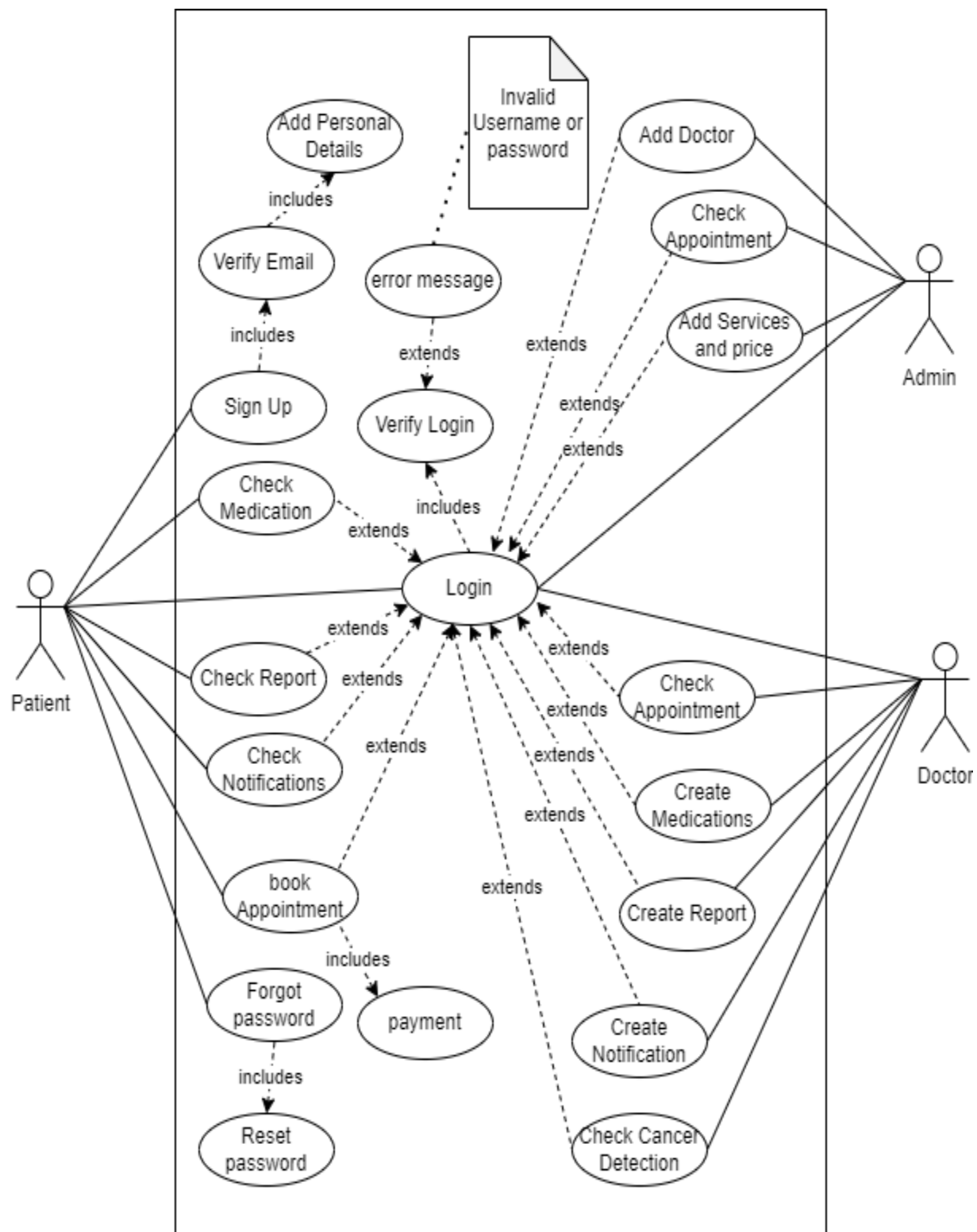


Figure 6 Use Case Diagram

## High Level of Use Case:

**Sign Up:****Use Case:** Sign Up**Actors:** Patient

**Description:** The patients provide their email and password. Afterwards, a verification code is sent to email address provided to verify the email and after verification the personal information of the patients are taken.

**Login:****Use Case:** Login**Actors:** Patient, Doctor, Admin

**Description:** The actors provide their credentials and the system authenticates the credentials and login the actors. If the credentials are wrong, it displays error message.

**Check Medication:****Use Case:** Check Medication**Actors:** Patient

**Description:** The patients check the medication provided for various diseases as per their checkup done.

**Book Appointment:****Use Case:** Book Appointment**Actors:** Patient

**Description:** The patient selects the service and doctor along with date and time as their choice and proceeds to payment in order to book appointment.

**Check Notification:**

**Use Case:** Check Notification

**Actors:** Patient

**Description:** The patient checks notifications received from the doctor in order to know their recovery progress and follow up details.

**Check Report:**

**Use Case:** Check Report

**Actors:** Patient

**Description:** The patient checks the report provided by the doctor on checked diseases and the patient can download the report as well.

**Forget Password:**

**Use Case:** Forget Password

**Actors:** Patient

**Description:** The patient receives a verification code on the provided email address if the email address is added to the system and after verification the password is allowed to be changed.

**Add Doctor:**

**Use Case:** Add Doctor

**Actors:** Admin

**Description:** The admin provides all the information about doctor with their credentials to add doctor to the system.



**Add Services and Price:**

**Use Case:** Add Services and Price

**Actors:** Admin

**Description:** The admin adds services and price to the system along with doctors specialized to the field to allow patients to book appointment for the services.

**Check Appointment:**

**Use Case:** Check Appointment

**Actors:** Admin

**Description:** The admin checks all the appointment that are booked for the current date.

**Check Appointment:**

**Use Case:** Check Appointment

**Actors:** Doctor

**Description:** The doctor checks all his appointments to schedule his tasks.

**Create Medication:**

**Use Case:** Create Medication

**Actors:** Doctor

**Description:** The doctor creates medication for the patients with respect to their report of diseases.

**Create Report:**

**Use Case:** Create Report

**Actors:** Doctor

**Description:** The doctor creates report for the patients with respect to outcomes of the tests of diseases.

**Create Notification:**

**Use Case:** Create Notification

**Actors:** Doctor

**Description:** The doctor creates notification for the patients with respect to the recovery made by the patients for follow up or update.

**Check Cancer Detection:**

**Use Case:** Check cancer Detection

**Actors:** Doctor

**Description:** The doctor appends the data or images to the AI model and get a prediction of having cancer or not.

## System Architecture:

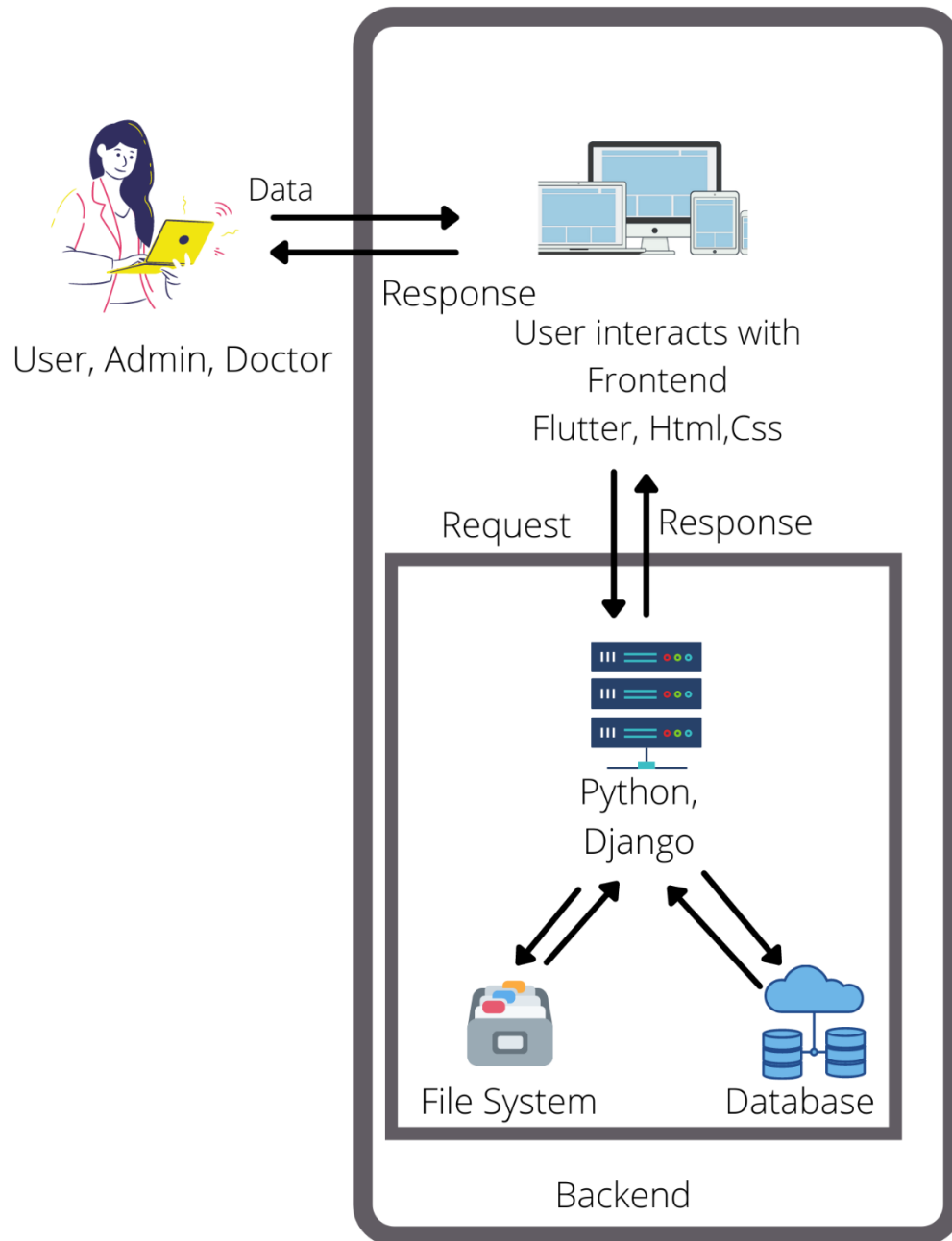



Figure 7 System Architecture

## Mobile UI Design:



Can-Dect


New to Can-Dect? [Sign up](#)

----- OR -----


Email

Email Address

Assistive text  
Password

Password 

Assistive text [Forget Password?](#)

 LOGIN

## Can-Dect Sign Up

1 of 2 Steps

Email


Email Address

Assistive text

Password

Password


Assistive text



Re-Password

Confirm Password

Assistive text




Next >


## Verify Email

Please Check Email for the 4 Digit verification Code

Code

Code does not match!

 [Resend Code!](#)

 [Verify](#)

## Can-Dect Sign Up

2 of 2 Steps

Name

Full Name

Assistive text

Phone

Phone Number

Assistive text

Address

Address

Assistive text

Age

Age

Assistive text

Sex

Male

Assistive text

Done

## Can-Dect

Welcome User,



Appointment



Report



Prescribed  
Medicine



Notifications



Bookings



Payments



## Can-Dect

Book Appointment,

Purpose

Select one

Doctor

Doctor's Name

Date

--/--/----

Time

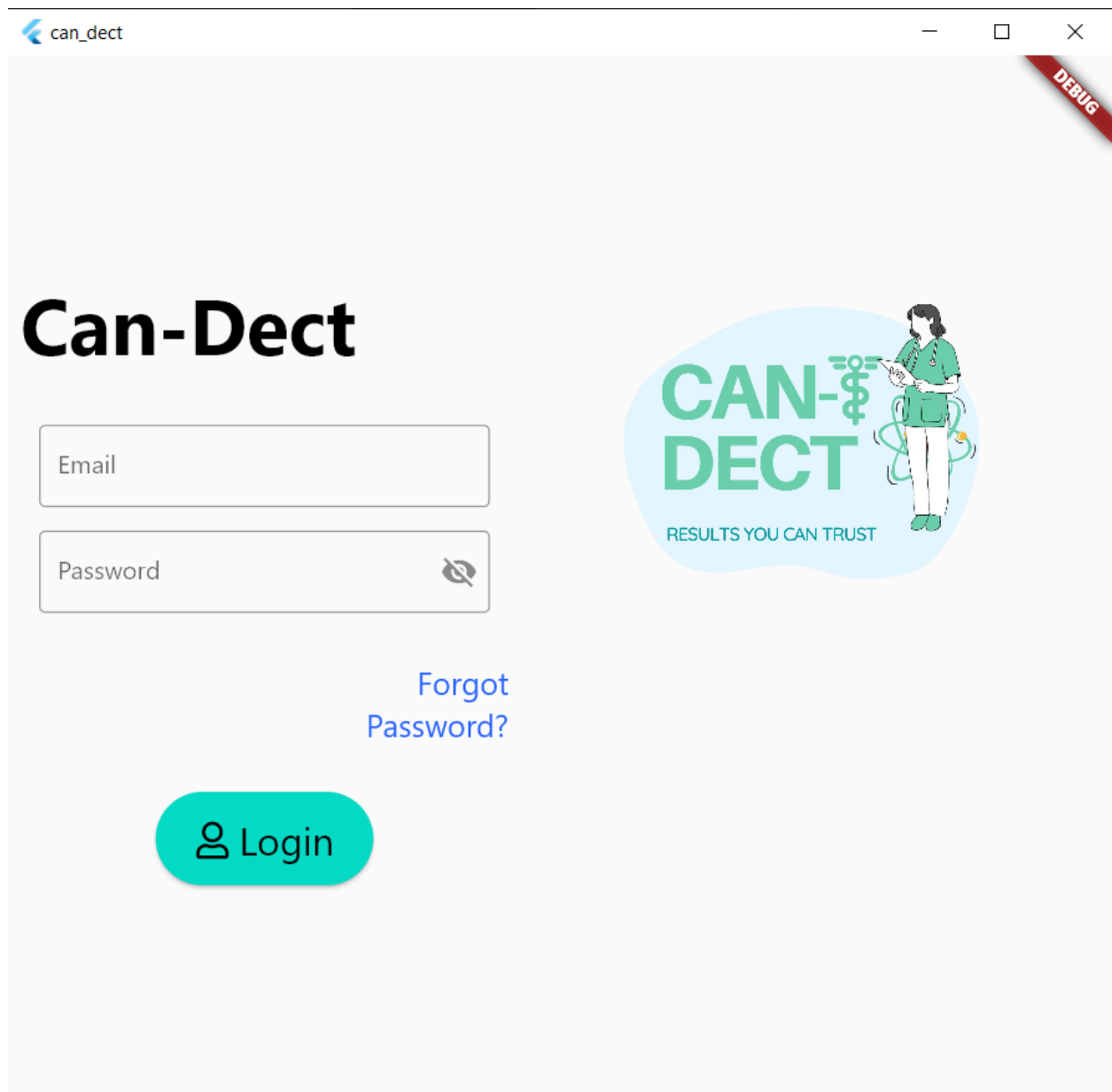
12:30

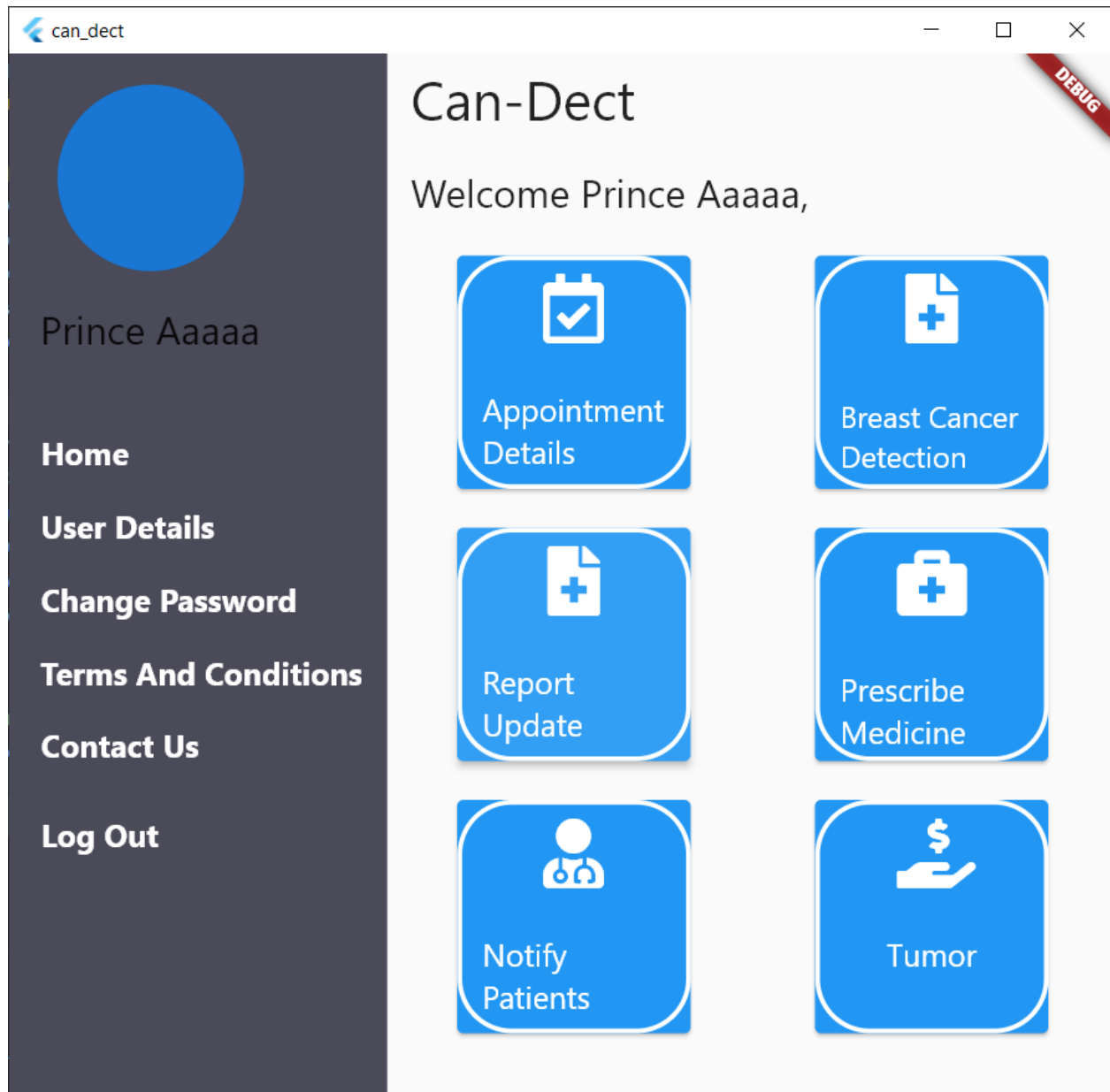
Appintment Fee

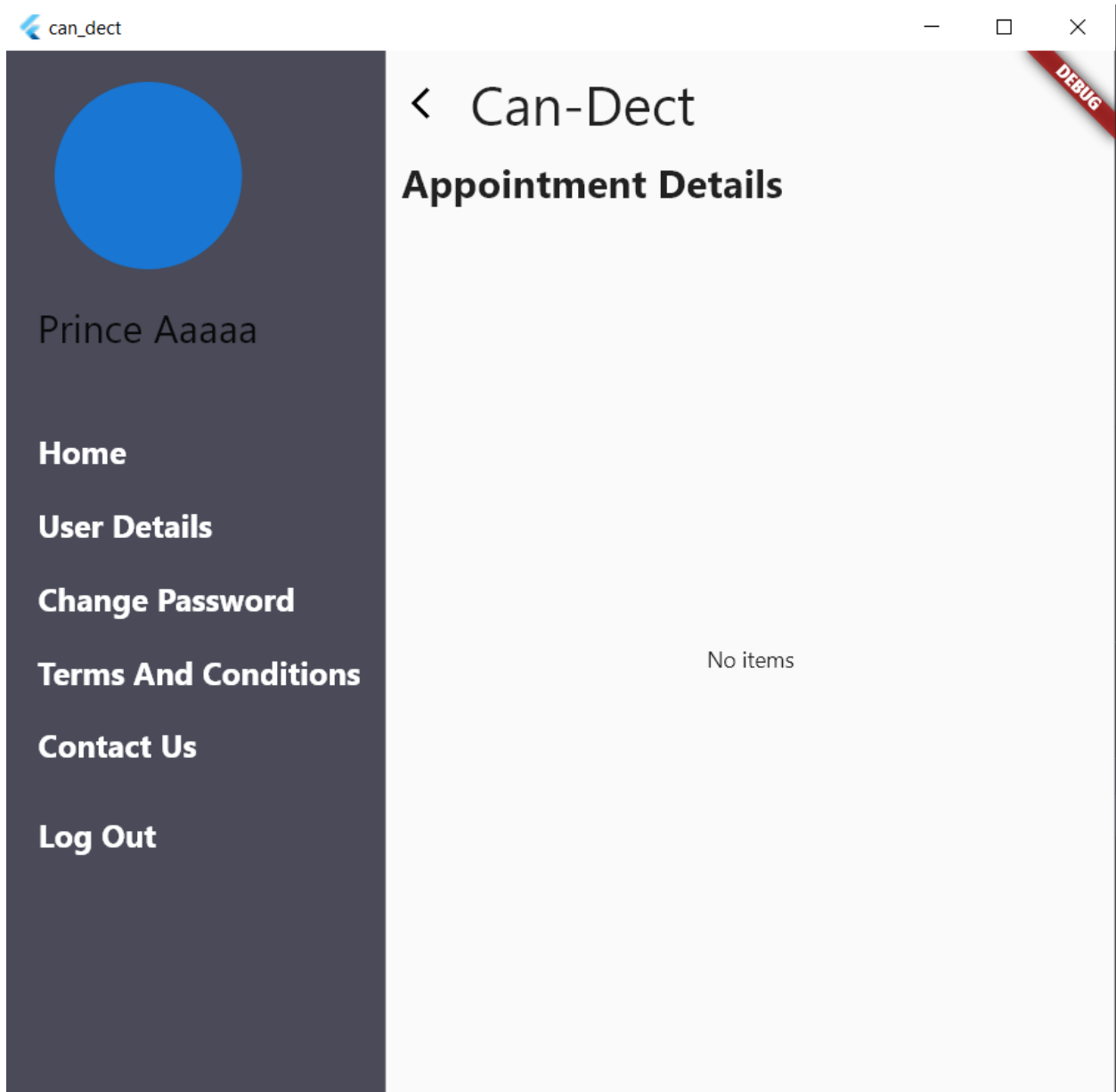
Rs 1000

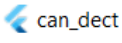
Proceed >


## Desktop Application Development:







can\_dect



Prince Aaaaa

**Home**

**User Details**

**Change Password**

**Terms And Conditions**

**Contact Us**

**Log Out**

## < Can-Dect

Create Report,

Patient

---

Diseases

Status  
Status ▾

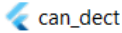
Additional Details


0/50

Upload Report **Upload**

**Create**

DEBUG

can\_dect



Prince Aaaaa

**Home**

**User Details**

**Change Password**

**Terms And Conditions**

**Contact Us**

**Log Out**

< Can-Dect

Create Medicine,

Patient

---

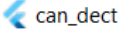
Disease


Medicine

0/50

Notify

DEBUG

 can\_dect



Prince Aaaaa

**Home**

**User Details**

**Change Password**

**Terms And Conditions**

**Contact Us**

**Log Out**

## < Can-Dect

Create Notification,

Patient

---

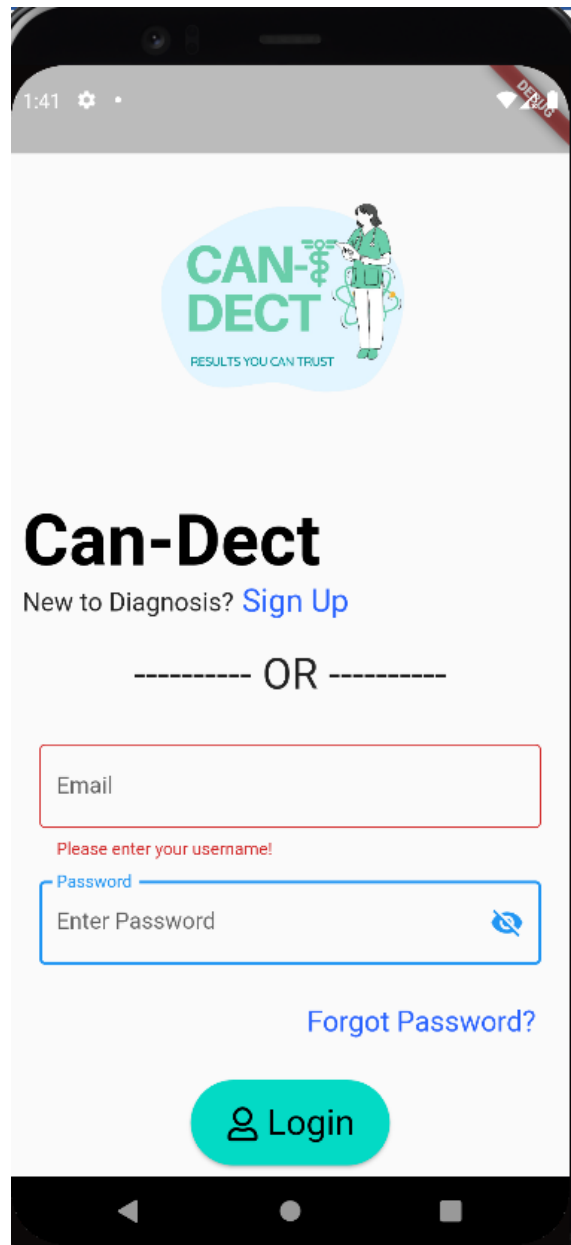
Message

0/50

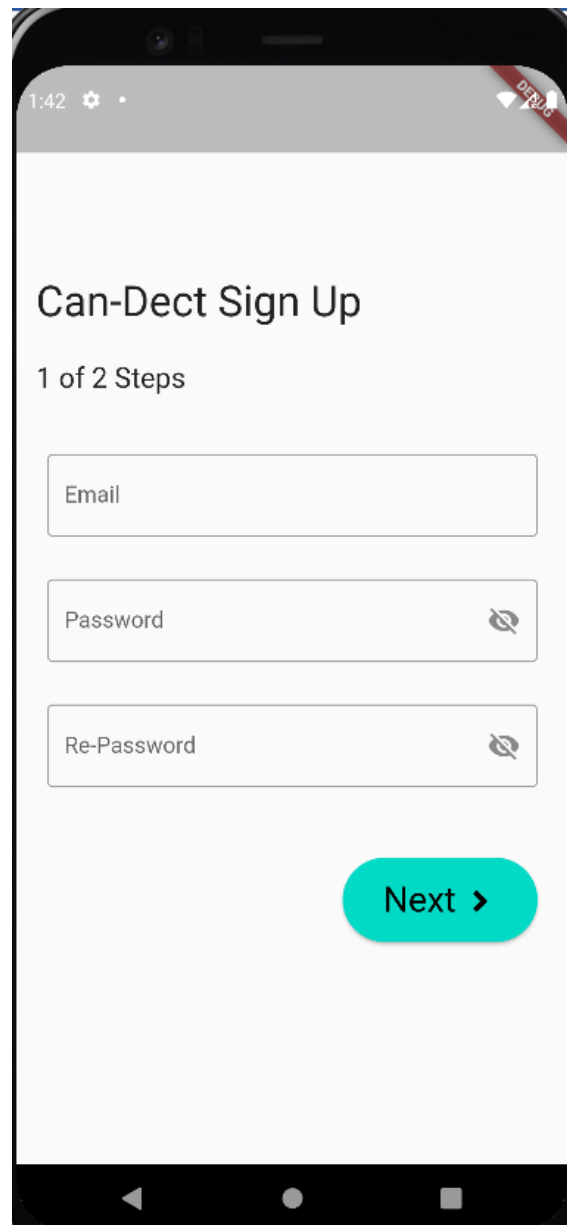
**Notify**

DEBUG

## Mobile Application Development:







1:42

Can-Dect Sign Up

1 of 2 Steps

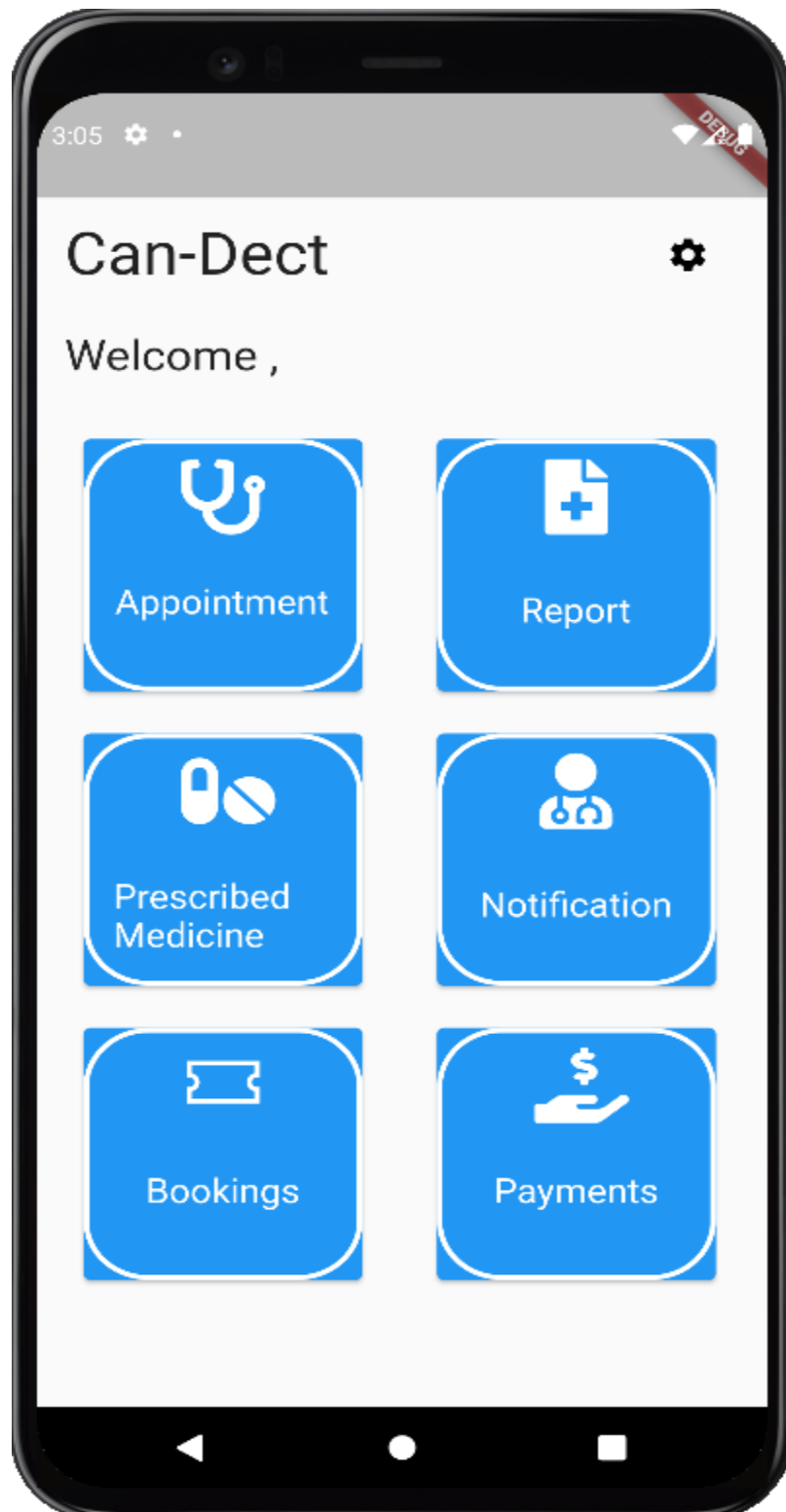
Email

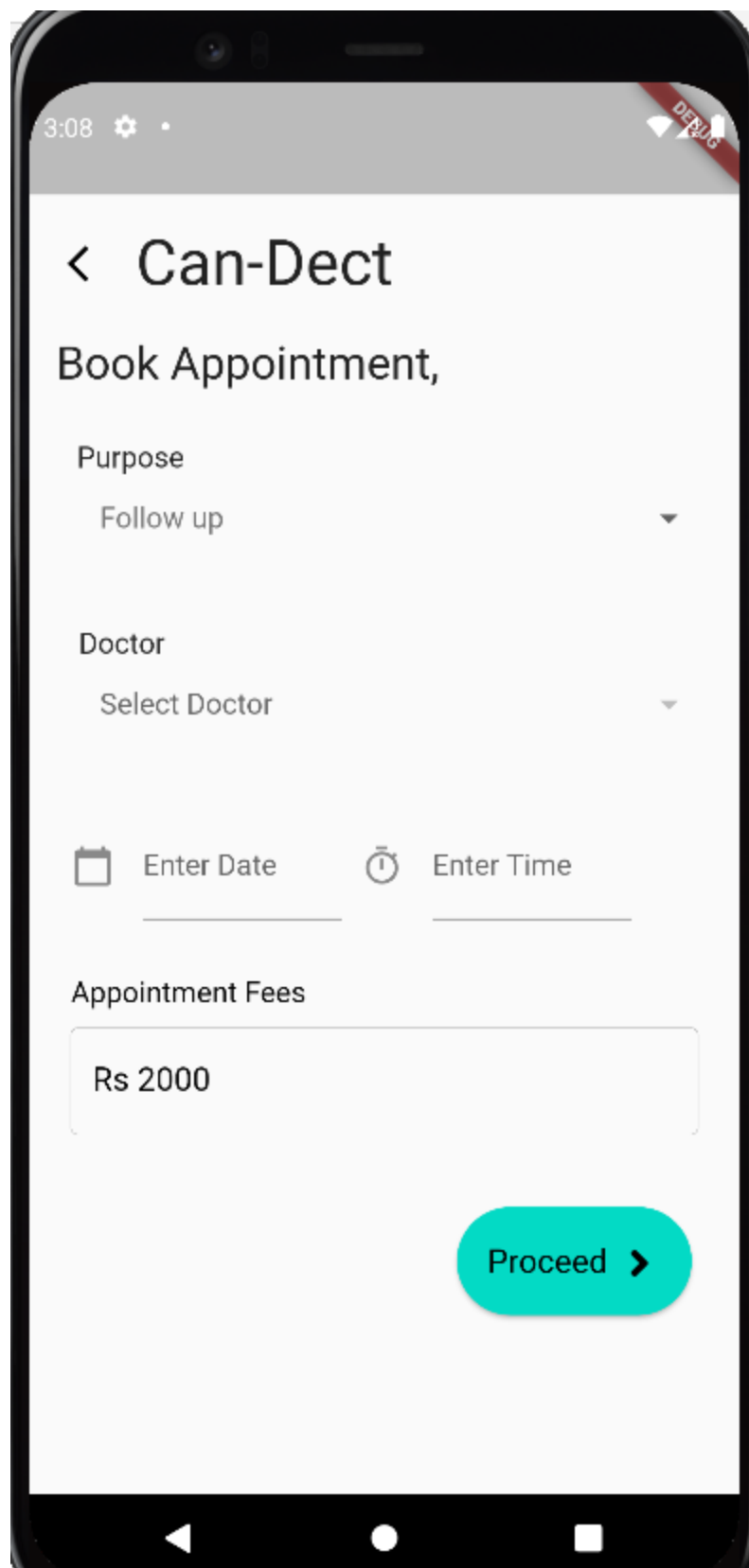
Password

Re-Password

Next >

The image shows a mobile application interface for a sign-up process. At the top, the status bar displays the time 1:42 and various icons. The app's title bar shows the time 09:46. The main heading is 'Can-Dect Sign Up', followed by '1 of 2 Steps'. There are three input fields: 'Email', 'Password', and 'Re-Password'. The 'Password' and 'Re-Password' fields have an eye icon to toggle visibility. A teal 'Next >' button is at the bottom right. The bottom of the screen shows the Android navigation bar.





The screenshot shows a mobile application interface for booking an appointment. At the top, the status bar displays the time 3:08, a settings gear icon, and a battery level indicator. The app's title bar shows a back arrow and the text "Can-Dect". Below the title, the main heading is "Book Appointment,". The form consists of several sections: a "Purpose" section with a dropdown menu currently showing "Follow up"; a "Doctor" section with a dropdown menu currently showing "Select Doctor"; a date and time selection section with icons for a calendar and a clock, and input fields labeled "Enter Date" and "Enter Time"; and an "Appointment Fees" section with a text input field containing "Rs 2000". At the bottom right, there is a prominent teal button labeled "Proceed" with a right-pointing arrow. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.

3:08

< Can-Dect

Book Appointment,

Purpose

Follow up

Doctor

Select Doctor

Enter Date

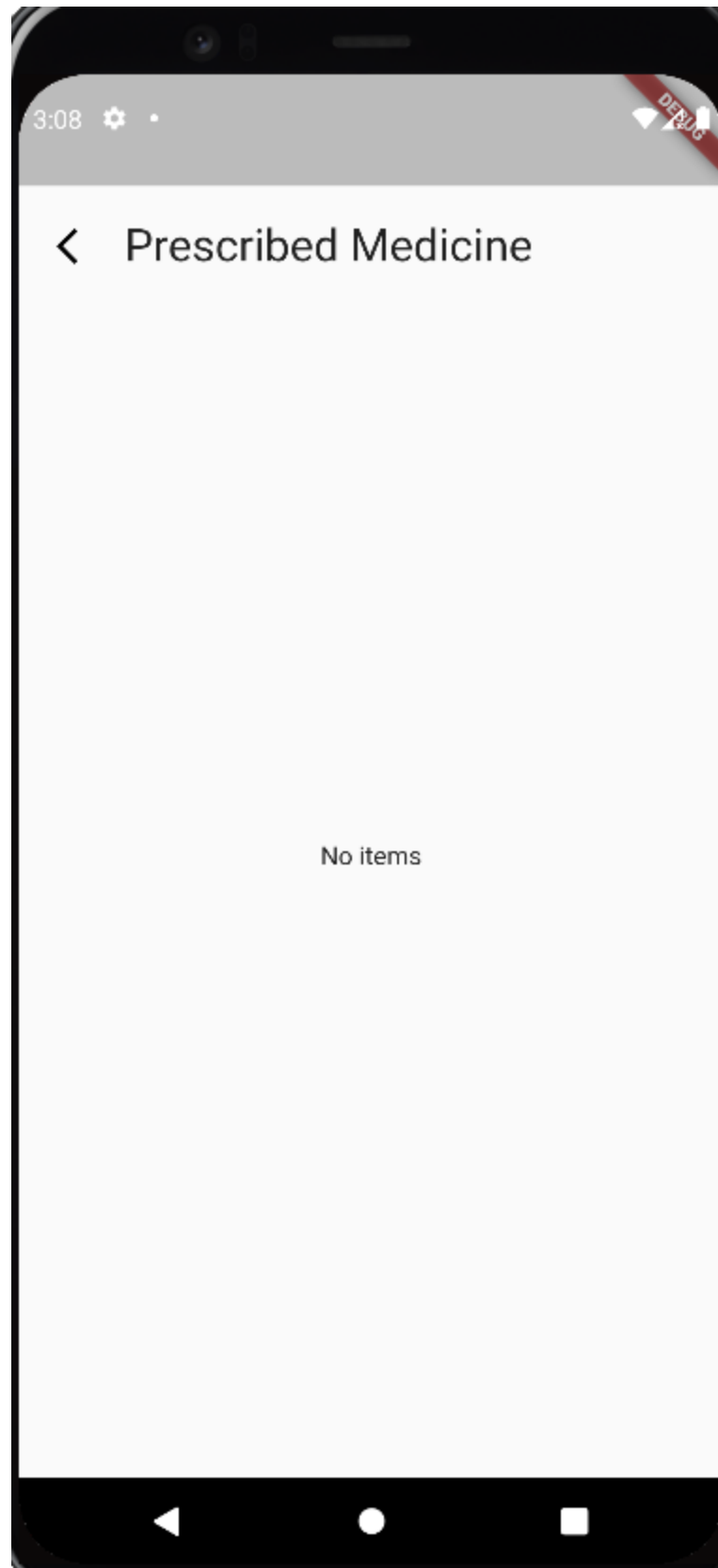
Enter Time

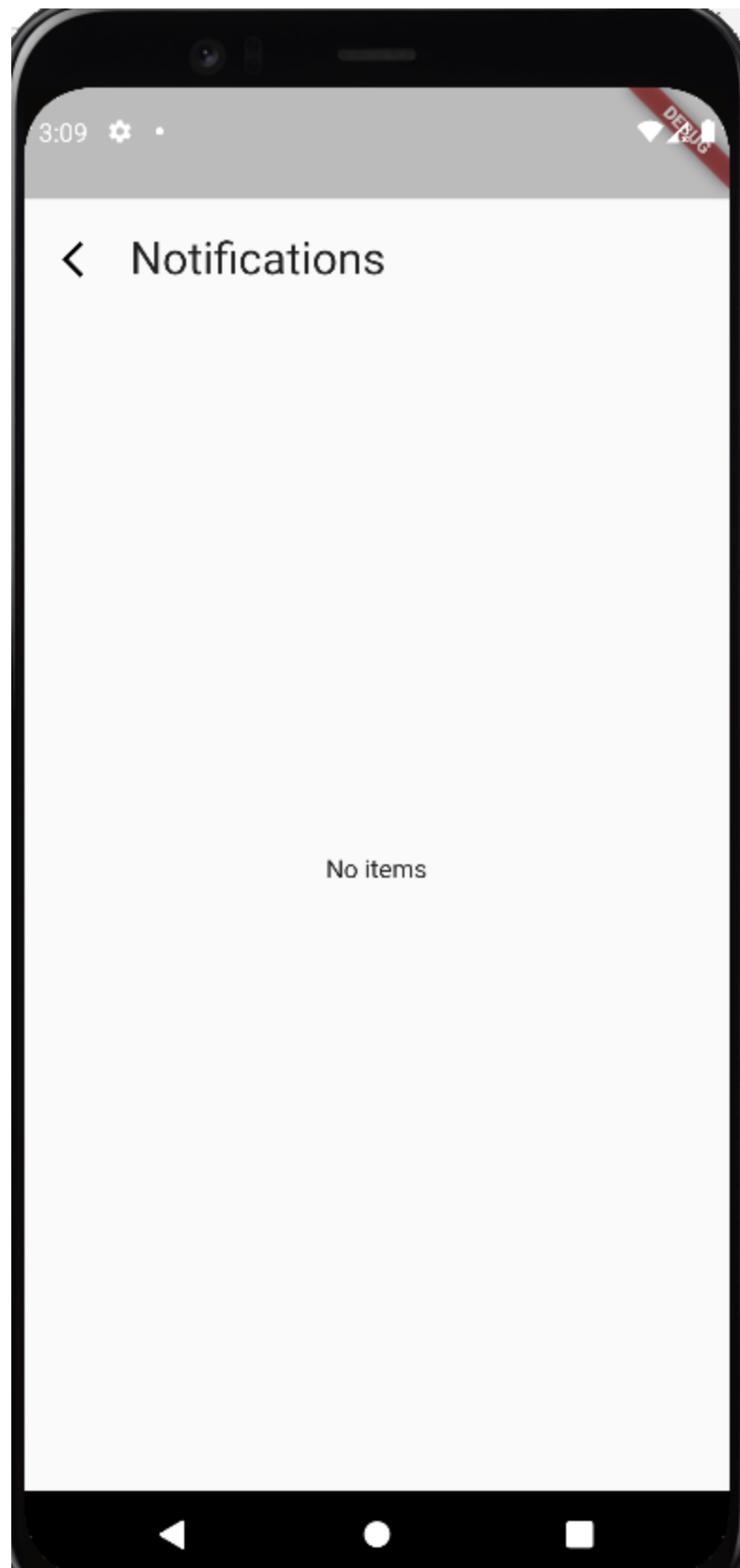
Appointment Fees

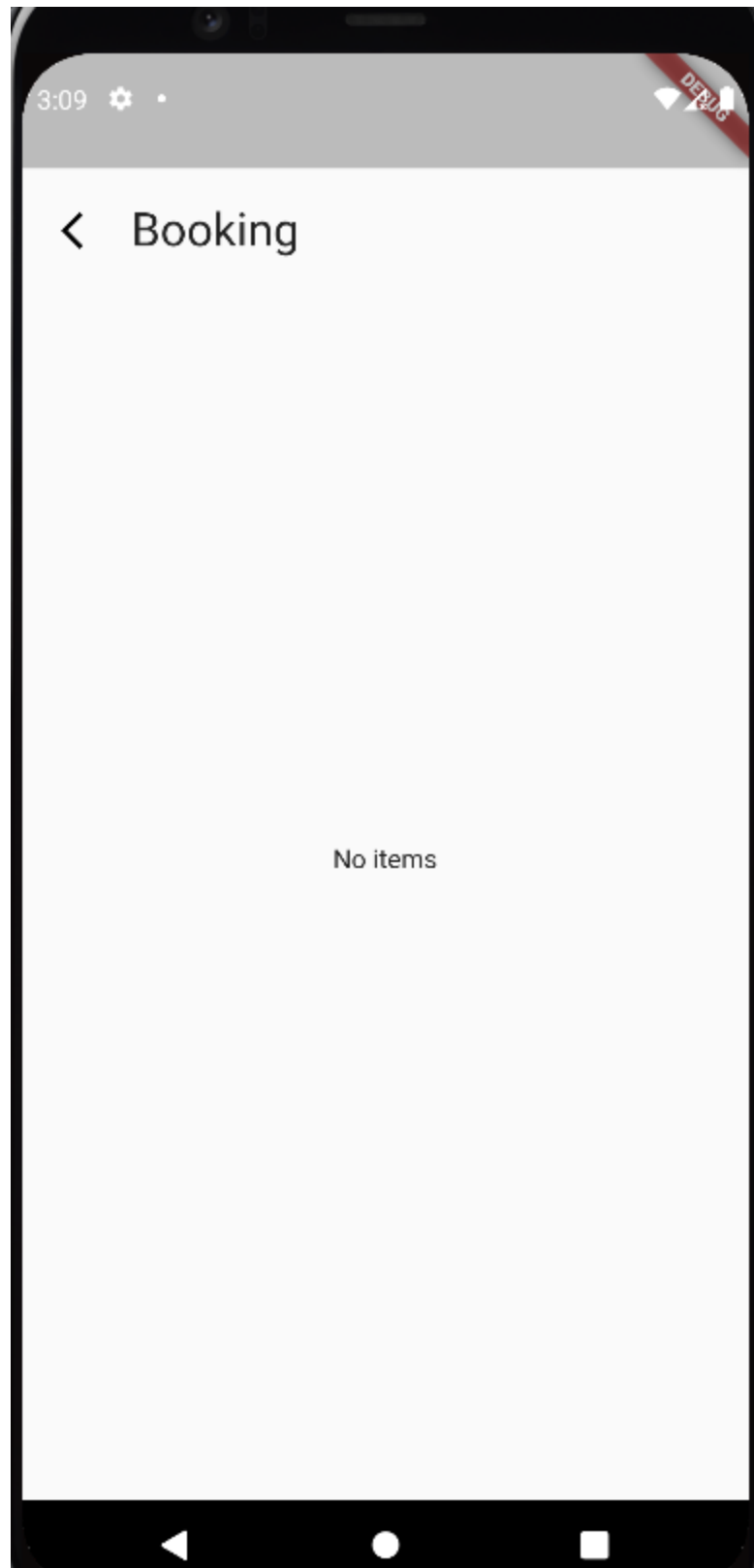
Rs 2000

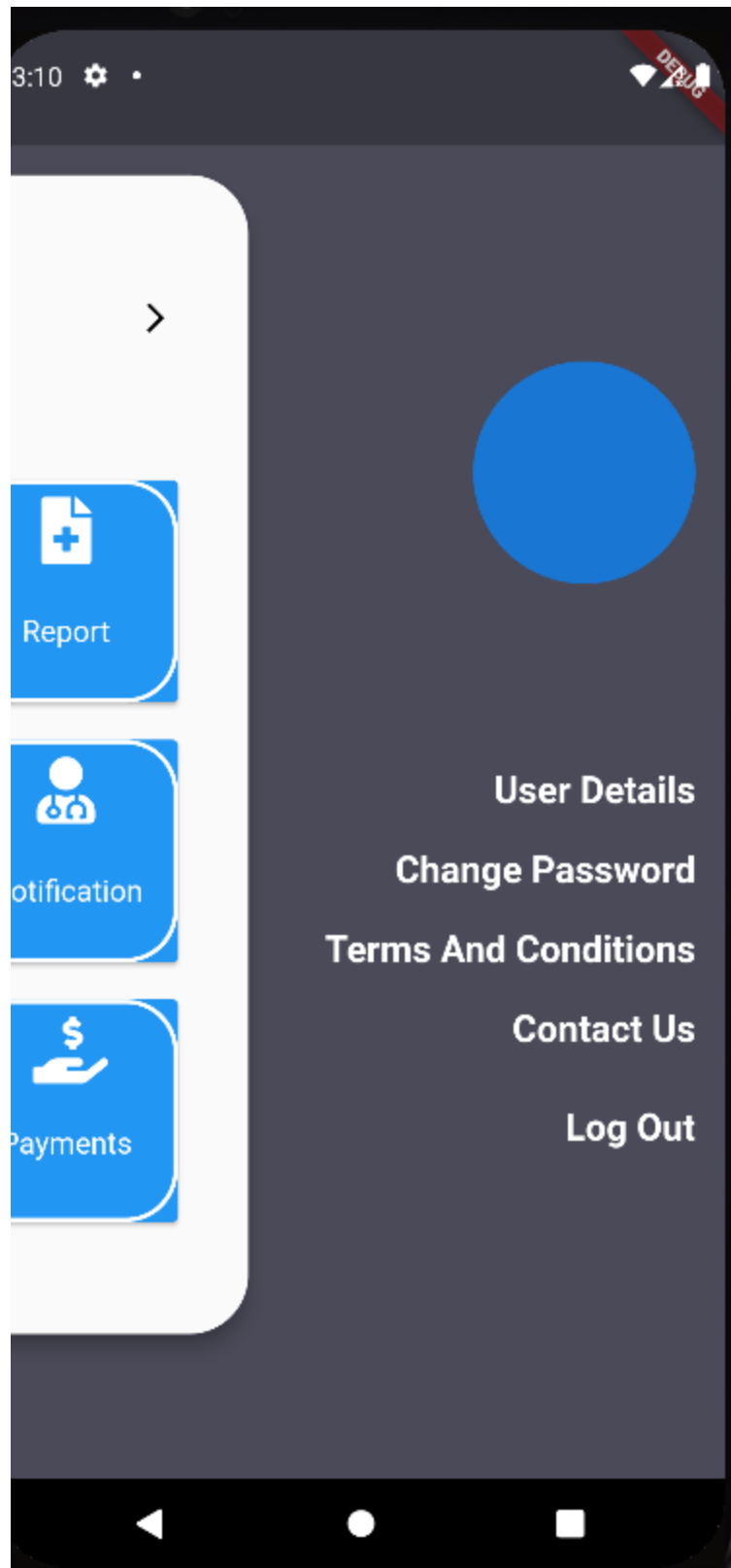
Proceed >













3:10

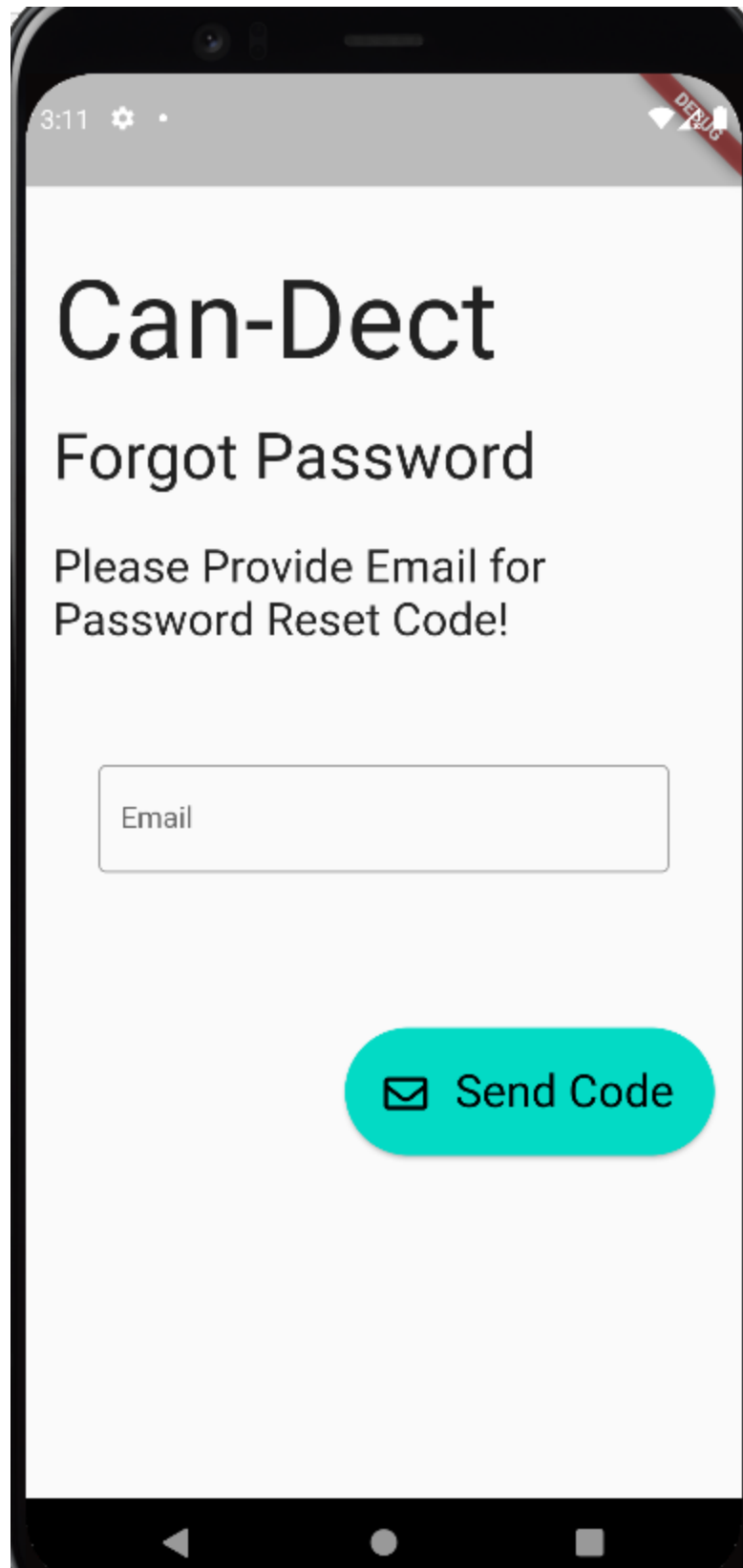
<

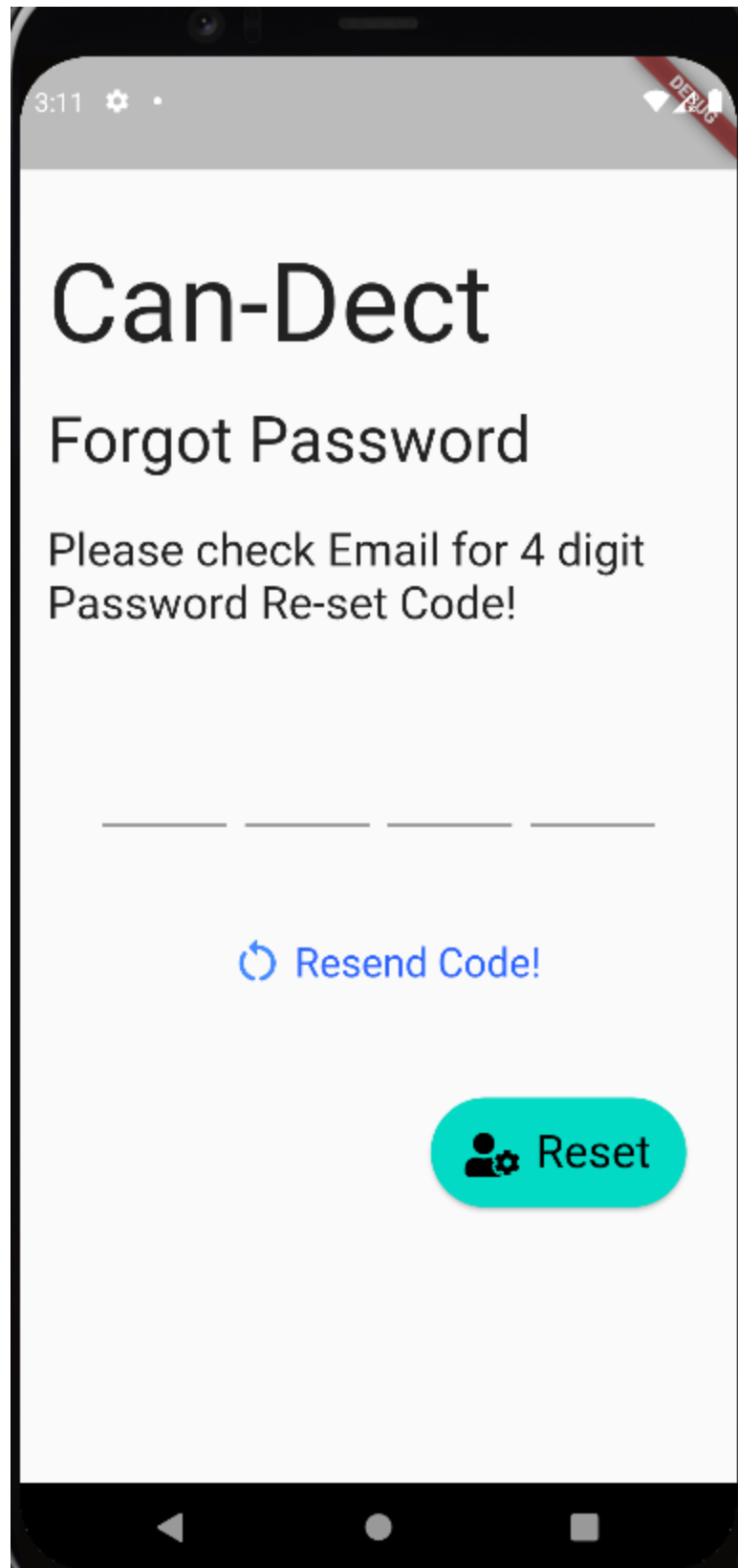
Old Password

New Password

Re-Password

Confirm ✓





### Analysis of Progress:

This section analyses the progress made on the project in comparison with the Gantt chart. This section assists to determine the project's current progress and know whether the progress is going as planned or is lagging behind and steps to bring it on track again. The progress made on the project in comparison with the Gantt chart is going as per plan. The Database design is lagging behind. The following things that were completed as described in the Gantt chart are:

#### Progress Table:

S.N.	Tasks	Status	Progress
1	Topic Selection	Completed	100%
2	Research on similar Projects	Completed	100%
3	Finalize Proposal	Completed	100%
4	System Architecture	Completed	100%
5	UI Design	Completed	90%
6	Database Design	Incomplete	0%
7	Mobile Application	Partially Completed	90%
8	Desktop Application	Partially Completed	80%
9	Admin Dash-Board	Incomplete	0%
10	Backend	Partially Completed	70%
11	AI Model training	Incomplete	0%
12	Testing	Incomplete	0%
13	Refinement	Incomplete	0%
14	Final FYP Report	Incomplete	0%

### Progress Review:

#### Current scenario of Progress:

In the initial phase, the topic finalization as well as proposal for the project was submitted so as to finalize the project topic for the development. The System Architecture as well as UI design for the project application was made and according to it the development of the mobile application as well as desktop application has been developed. The research and study on Flutter has been done to complete the application as well as backend with the help of Django framework has been started to be developed. The development work along with debugging is going simultaneously so as to boost the project completion.

Up to now, the development of the project is about 60% completed. The rate of completion of the project is going faster than planned.

#### Action Plan:

The tasks are being completed ahead of the planning date as well as documentation and testing of each component is going on simultaneously which will lead to completion of the project on time.

## Future Work:

This section contains the project's remaining work that needs to be completed according to the deadline made in the Gantt Chart.

- Web Admin panel:

For the development of web Admin panel, I would start developing the frontend with the help of HTML, CSS along with-it Django template rendering will be used to render pages according to data flow through frontend and backend.

- Database Design:

The database design for the proper storage of data and manipulate the data from database is a must need. For it, proper research of attributes and entities are necessary.

- Testing:

After the completion of the project, proper testing of work flow of all the components is a must need to finally conclude the completion of the project.

- Final Documentation:

At last, the documentation of working of the project along with the testing needs to be documented so as to finally show evidence of the development and working of the project.

## Bibliography

CDC, 2021. *CDC.* [Online]  
Available at: [https://www.cdc.gov/cancer/breast/basic\\_info/what-is-breast-cancer.htm](https://www.cdc.gov/cancer/breast/basic_info/what-is-breast-cancer.htm)  
[Accessed 5 December 2021].

Digite, 2021. *Digite.* [Online]  
Available at: <https://www.digite.com/kanban/what-is-kanban/>  
[Accessed 15 December 2021].

Existek, 2021. *Existek.* [Online]  
Available at: <https://existek.com/blog/hospital-managment-system/>  
[Accessed 15 December 2021].

Gantt, 2021. *Gantt.* [Online]  
Available at: <https://www.gantt.com/>  
[Accessed 14 December 2021].

Lewis, S., 2019. *TechTarget.* [Online]  
Available at: <https://searchcio.techtarget.com/definition/Prototyping-Model>  
[Accessed 14 December 2021].

workfront., 2021. *workfront..* [Online]  
Available at: <https://www.workfront.com/project-management/methodologies/waterfall>  
[Accessed 14 December 2021].

## Appendix:

UI design:

## Can-Dect

Book Appointment,

Purpose

Select one

Doctor

Doctor's Name

Date

--/--/----

Time

12:30

Appintment Fee

Rs 1000

Proceed >



## Can-Dect

### Check Report,

Report Name

Patient Name: Abcd Awsd

Diseases: Cancer

Status: Positive

Report Name

Patient Name: Abcd Awsd

Diseases: TB

Status: Negative

Report Name

Patient Name: Abcd Awsd

Diseases: BP

Status: High

## Can-Dect

### Prescribed Medicine

Diseases Name

Sinex - 2 times a day

Paracetamol - 1 times a day

Diseases Name

Sinex - 2 times a day

Paracetamol - 1 times a day

## Can-Dect

### Notifications,

Your Reports have come.

Appointment has been booked.

You have a follow up checkup on  
Sunday.

## Can-Dect

Bookings,

Purpose:  
Booking Date:  
Token Number:  
Time:  
Room No:

Purpose:  
Booking Date:  
Token Number:  
Time:  
Room No:

## Can-Dect


### Payment Details,

Purpose: Cancer Diagnosis

Amount: 15000

Payment Method: Cash

Payment Status: Received



Hari Prasad Shah

Phone

9865400000

Assistive text

Address


Anamnagar


Assistive text

Email

hariprasad@gmail.com

Assistive text

Save 



Hari Prasad Shah

Assistive text

Old Password

Password

Assistive text

New-Password

New Password

Assistive text

Re-Password

Confirm Password

Assistive text

Confirm

## Desktop Application Code:

## Login Page:

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:http/http.dart' as http;

import 'homes.dart';

// ignore: camel_case_types
class Loginpage extends StatefulWidget {
  const Loginpage({Key? key}) : super(key: key);

  @override
  _LoginpageState createState() => _LoginpageState();
}

// ignore: camel_case_types
class _LoginpageState extends State<Loginpage> {
  final username = TextEditingController();
  final password = TextEditingController();
  bool isWro = false;
  bool emptyuser = false;
  bool emptypw = false;
  bool pwseen = false;
  late final int data;

  login(String username, String password) async {
    final http.Response response = await http.post(
      Uri.parse('http://127.0.0.1:8000/doctorapi/v1/doctorLogin/'),
      headers: {"Content-Type": "application/json"},
      body: json.encode(
        <String, dynamic>{
          'email': username,
          'password': password,
        },
      ),
    );
    if (response.statusCode == 200) {
      var result = json.decode(response.body);
      if (result == "not successful") {
        isWro = true;
      } else {
```



```

        isWro = false;
        setState(() {
          data = result['id'];
        });
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => homepage(
              email: data,
            ),
          ),
        );
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: <Widget>[
          SizedBox(
            child: Row(
              children: [
                Column(
                  children: <Widget>[
                    Material(
                      child: SingleChildScrollView(
                        scrollDirection: Axis.vertical,
                        child: SizedBox(
                          height: MediaQuery.of(context).size.height,
                          width: 330,
                          child: Padding(
                            padding: const EdgeInsets.all(8.0),
                            child: Column(
                              crossAxisAlignment: CrossAxisAlignment.start,
                              mainAxisAlignment: MainAxisAlignment.center,
                              children: [
                                const Text(
                                  "Can-Dect",
                                  style: TextStyle(
                                    color: Colors.black,
                                    fontSize: 50,
                                    fontWeight: FontWeight.bold,

```

```

    ),
  ),
  const SizedBox(
    height: 20,
  ),
  Padding(
    padding: const EdgeInsets.all(12.0),
    child: Column(
      children: [
        TextField(
          controller: username,
          obscureText: false,
          onChanged: (val) {
            emptyuser = false;
            isWro = false;
          },
          style:
            const TextStyle(color: Colors.blue),
          decoration: InputDecoration(
            border: const OutlineInputBorder(
              borderSide: BorderSide(
                color: Colors.blueAccent),
            ),
            labelText: 'Email',
            hintText: 'Enter Email',
            errorText: emptyuser
              ? 'Please enter your username!'
              : null,
          ),
        ),
        const SizedBox(
          height: 15,
        ),
        TextField(
          controller: password,
          obscureText: pwseen ? false : true,
          onChanged: (val) {
            emptypw = false;
            isWro = false;
          },
          decoration: InputDecoration(
            border: const OutlineInputBorder(
              borderSide:
                BorderSide(color: Colors.blue),

```

```

    ),
    labelText: 'Password',
    hintText: 'Enter Password',
    errorText: emptypw
      ? 'Please enter your password!'
      : null,
    suffixIcon: IconButton(
      icon: pwseen
        ? const Icon(Icons.visibility)
        : const Icon(
            Icons.visibility_off),
      onPressed: () {
        setState(() {
          if (pwseen == true) {
            pwseen = false;
          } else {
            pwseen = true;
          }
        });
      },
    ),
  ),
  isWro
    ? const Text(
        'Username or password is invalid!',
        style: TextStyle(
          color: Colors.red,
          fontSize: 14),
      )
    : const Text(''),
  ],
),
),
Padding(
  padding: const EdgeInsets.only(left: 200.0),
  child: Column(
    children: [
      GestureDetector(
        onTap: () => Navigator.pushNamed(
          context, 'forgetemail'),
        child: Text(
          'Forgot Password?',
          style: TextStyle(

```

```

        color: Colors.blueAccent[700],
        fontSize: 20),
        textAlign: TextAlign.end,
    ),
    ),
  ],
),
),
const SizedBox(
  height: 20,
),
Center(
  child: Padding(
    padding: const EdgeInsets.only(
      top: 10.0,
    ),
  ),
  child: ElevatedButton(
    onPressed: () {
      if (username.text.isEmpty) {
        setState(() {
          emptyuser = true;
        });
      } else if (password.text.isEmpty) {
        setState(() {
          emptypw = true;
        });
      } else {
        final String txtusername =
          username.text;
        final String txtpassword =
          password.text;
        login(txtusername, txtpassword);
      }
    },
    style: ElevatedButton.styleFrom(
      primary: const Color.fromRGBO(
        3, 218, 197, 1),
      fixedSize: const Size(140, 60),
      shape: RoundedRectangleBorder(
        borderRadius:
          BorderRadius.circular(50),
      ),
    ),
  ),
  child: Row(

```



```

    ),
  ],
),
);
}
}

```

### Home Page:

```

// ignore_for_file: import_of_legacy_library_into_null_safe, camel_case_types

import 'dart:convert';
import 'package:can_dect/Pages/appoint_details.dart';
import 'package:can_dect/Pages/medicine.dart';
import 'package:can_dect/Pages/menu.dart';
import 'package:can_dect/Pages/notify.dart';
import 'package:can_dect/Pages/update_report.dart';
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:line_icons/line_icons.dart';
import 'package:http/http.dart' as http;
import 'package:text_tools/text_tools.dart';

const Color backgroundColor = Color(0xFF4A4A58);

class homepage extends StatefulWidget {
  final int email;
  const homepage({Key? key, required this.email}) : super(key: key);

  @override
  _homepageState createState() => _homepageState();
}

class _homepageState extends State<homepage>
  with SingleTickerProviderStateMixin {
  bool isCollapsed = true;
  late double screenWidth, screenHeight;

  late final int data;
  List userss = [];
  late String username;

  @override
  void initState() {
    username = '';
  }

```

```

    _user(widget.email);

    super.initState();
  }

  _user(int email) async {
    final http.Response response = await http.get(
      Uri.parse(
        'http://127.0.0.1:8000/doctorapi/v1/doctordetail/$email/?format=json'),
      headers: {"Content-Type": "application/json"},
    );
    if (response.statusCode == 200) {
      var users = json.decode(response.body);
      setState(() {
        usersss.add(users);
        username =
          TextTools.toUppercaseFirstLetter(text: usersss[0]['FirstName']) +
            " " +
            TextTools.toUppercaseFirstLetter(text: usersss[0]['LastName']);
      });
    }
  }

  @override
  void dispose() {
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    Size size = MediaQuery.of(context).size;
    screenHeight = size.height;
    screenWidth = size.width;

    return Scaffold(
      backgroundColor: backgroundColor,
      body: Stack(
        children: <Widget>[
          SizedBox(
            child: Row(
              children: [
                menu(username: username, email: widget.email),
                Column(
                  children: <Widget>[

```

```
Material(  
  child: SingleChildScrollView(  
    scrollDirection: Axis.vertical,  
    physics: const ClampingScrollPhysics(),  
    child: Column(  
      crossAxisAlignment: CrossAxisAlignment.start,  
      children: <Widget>[  
        SizedBox(  
          height: 670,  
          width: 467,  
          child: Padding(  
            padding: const EdgeInsets.only(  
              top: 5.0, right: 10.0, left: 15, bottom: 5),  
            child: Column(  
              mainAxisAlignment: MainAxisAlignment.start,  
              crossAxisAlignment: CrossAxisAlignment.start,  
              children: [  
                Row(  
                  children: const [  
                    Text(  
                      'Can-Dect',  
                      style: TextStyle(  
                        fontSize: 35,  
                        fontWeight: FontWeight.normal,  
                      ),  
                    ),  
                  ],  
                ),  
                const SizedBox(  
                  height: 20,  
                ),  
                Text(  
                  'Welcome $username,',  
                  style: const TextStyle(  
                    fontSize: 25,  
                    fontWeight: FontWeight.normal,  
                  ),  
                ),  
                const SizedBox(  
                  height: 15,  
                ),  
                Padding(  
                  padding: const EdgeInsets.only(  
                    top: 10.0,
```



```

        left: 30.0,
      ),
      child: Column(
        children: [
          Row(
            children: [
              Container(
                height: 150,
                width: 150,
                foregroundDecoration:
                  BoxDecoration(
                    borderRadius:
                      BorderRadius.circular(
                        35.0),
                    border: Border.all(
                      color: Colors.white,
                      width: 3.0),
                  ),
              ElevatedButton(
                onPressed: () => {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) =>
                        Appointdetails(
                          doctorid: userss[0]
                            ['id'],username:
username,email: widget.email,
                    ),
                  ),
                ),
              ),
            ],
          ),
        ],
      ),
      child: Padding(
        padding:
          const EdgeInsets.only(
            top: 10.0),
        child: Column(
          children: const [
            Icon(
              FontAwesomeIcons
                .calendarCheck,
              size: 45,
            ),
            SizedBox(

```

```

        height: 30,
      ),
      Text(
        'Appointment Details',
        style: TextStyle(
          fontSize: 20,
          fontWeight:
            FontWeight
              .normal,
        ),
      ),
    ],
  ),
),
const SizedBox(
  width: 80,
),
Container(
  height: 150,
  width: 150,
  foregroundDecoration:
    BoxDecoration(
      borderRadius:
        BorderRadius.circular(
          35.0),
      border: Border.all(
        color: Colors.white,
        width: 3.0),
    ),
  child: ElevatedButton(
    onPressed: () => {
      // Navigator.push(
      //   context,
      //   MaterialPageRoute(
      //     builder: (context) =>
      //       Reportpage(
      //         doctorid:
usersss[0]['id'],
      //       ),
      //     ),
      //   ),
    },
  ),
),

```

Detection',

```

        child: Padding(
          padding:
            const EdgeInsets.only(
              top: 10.0),
          child: Column(
            children: const [
              Icon(
                FontAwesomeIcons
                  .fileMedical,
                size: 45,
              ),
              SizedBox(
                height: 35,
              ),
              Text(
                'Breast Cancer
Detection',

                style: TextStyle(
                  fontSize: 19,
                  fontWeight:
                    FontWeight
                      .normal,
                ),
              ),
            ],
          ),
        ),
      ),
    ),
  ],
),
const SizedBox(
  height: 25,
),
Row(
  children: [
    Container(
      height: 150,
      width: 150,
      foregroundDecoration:
        BoxDecoration(
          borderRadius:
            BorderRadius.circular(
              35.0),

```

```

border: Border.all(
  color: Colors.white,
  width: 3.0),
),
child: ElevatedButton(
  onPressed: () => {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) =>
          CreateReport(username
: username,email: widget.email,)),
    ),
  ),
),
},
child: Padding(
  padding:
    const EdgeInsets.only(
      top: 10.0),
  child: Column(
    children: const [
      Icon(
        FontAwesomeIcons
          .fileMedical,
        size: 45,
      ),
      SizedBox(
        height: 30,
      ),
      Text(
        'Report Update',
        style: TextStyle(
          fontSize: 20,
          fontWeight:
            FontWeight
              .normal,
        ),
      ),
    ],
  ),
),
),
),
const SizedBox(

```

```

width: 80,
),
Container(
  height: 150,
  width: 150,
  foregroundDecoration:
    BoxDecoration(
      borderRadius:
        BorderRadius.circular(
          35.0),
      border: Border.all(
        color: Colors.white,
        width: 3.0),
    ),
  child: ElevatedButton(
    onPressed: () => {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) =>
            CreateMedicine(
              doctorid: userss[0]
                ['id'],username:
username, email: widget.email,
            ),
          ),
        ),
      ),
    ),
  ),
  child: Padding(
    padding:
      const EdgeInsets.only(
        top: 10.0),
    child: Column(
      children: const [
        Icon(
          FontAwesomeIcons
            .briefcaseMedical,
          size: 45,
        ),
        SizedBox(
          height: 35,
        ),
        Text(
          'Prescribe Medicine',

```

```

        style: TextStyle(
          fontSize: 20,
          fontWeight:
            FontWeight
              .normal,
        ),
      ),
    ],
  ),
),
const SizedBox(
  height: 25,
),
Row(
  children: [
    Container(
      height: 150,
      width: 150,
      foregroundDecoration:
        BoxDecoration(
          borderRadius:
            BorderRadius.circular(
              35.0),
          border: Border.all(
            color: Colors.white,
            width: 3.0),
        ),
      child: ElevatedButton(
        onPressed: () => {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) =>
                CreateNotification(
                  doctorid: userss[0]
                    ['id'],username:
username,email: widget.email,

```

```

    },
    child: Padding(
      padding:
        const EdgeInsets.only(
          top: 10.0),
      child: Column(
        children: const [
          Icon(
            FontAwesomeIcons
              .userMd,
            size: 45,
          ),
          SizedBox(
            height: 30,
          ),
          Text(
            'Notify Patients',
            style: TextStyle(
              fontSize: 20,
              fontWeight:
                FontWeight
                  .normal,
            ),
          ),
        ],
      ),
    ),
  ),
  const SizedBox(
    width: 80,
  ),
  Container(
    height: 150,
    width: 150,
    foregroundDecoration:
      BoxDecoration(
        borderRadius:
          BorderRadius.circular(
            35.0),
        border: Border.all(
          color: Colors.white,
          width: 3.0),
      ),
  ),

```





```

    ),
  ),
],
),
],
),
),
],
),
);
}
}

```

### Menu Section:

```

import 'package:can_dect/Pages/homes.dart';
import 'package:flutter/material.dart';

class menu extends StatefulWidget {
  String username;
  int email;
  menu({
    Key? key,
    required this.username,
    required this.email,
  }) : super(key: key);

  @override
  _menuState createState() => _menuState();
}

class _menuState extends State<menu> {
  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Padding(
          padding: const EdgeInsets.only(
            right: 16.0,
            top: 20.0,
            left: 20,
          ),
          child: Align(
            alignment: Alignment.centerRight,

```

```
child: Column(  
  mainAxisAlignment: MainAxisAlignment.min,  
  mainAxisAlignment: MainAxisAlignment.spaceAround,  
  crossAxisAlignment: CrossAxisAlignment.start,  
  children: <Widget>[  
    Column(  
      mainAxisAlignment: MainAxisAlignment.end,  
      crossAxisAlignment: CrossAxisAlignment.center,  
      children: [  
        const CircleAvatar(  
          radius: 60.0,  
          foregroundImage:  
            ExactAssetImage('lib/images/download.jpg'),  
        ),  
        const SizedBox(  
          height: 20,  
        ),  
        Text(  
          widget.username,  
          style: const TextStyle(  
            fontSize: 25,  
            fontWeight: FontWeight.normal,  
          ),  
        ),  
      ],  
    ),  
    const SizedBox(  
      height: 50,  
    ),  
    GestureDetector(  
      onTap: () => Navigator.push(  
        context,  
        MaterialPageRoute(  
          builder: (context) => homepage(  
            email: widget.email,  
          ),  
        ),  
      ),  
    ),  
    child: const Text(  
      'Home',  
      style: TextStyle(  
        color: Colors.white,  
        fontSize: 20,  
        fontWeight: FontWeight.bold),  
    ),  
  ],  
),
```

```
        textAlign: TextAlign.end,
      ),
    ),
    const SizedBox(height: 20),
    GestureDetector(
      onTap: () => Navigator.pushNamed(context, 'userdetails'),
      child: const Text(
        'User Details',
        style: TextStyle(
          color: Colors.white,
          fontSize: 20,
          fontWeight: FontWeight.bold),
        textAlign: TextAlign.end,
      ),
    ),
    const SizedBox(height: 20),
    GestureDetector(
      onTap: () => Navigator.pushNamed(context, 'change'),
      child: const Text(
        'Change Password',
        style: TextStyle(
          color: Colors.white,
          fontSize: 20,
          fontWeight: FontWeight.bold),
        textAlign: TextAlign.end,
      ),
    ),
    const SizedBox(height: 20),
    GestureDetector(
      onTap: () => Navigator.pushNamed(context, 'forgetemail'),
      child: const Text(
        'Terms And Conditions',
        style: TextStyle(
          color: Colors.white,
          fontSize: 20,
          fontWeight: FontWeight.bold),
        textAlign: TextAlign.end,
      ),
    ),
    const SizedBox(height: 20),
    GestureDetector(
      onTap: () => Navigator.pushNamed(context, 'forgetemail'),
      child: const Text(
        'Contact Us',
```

```

        style: TextStyle(
          color: Colors.white,
          fontSize: 20,
          fontWeight: FontWeight.bold),
        textAlign: TextAlign.end,
      ),
    ),
    const SizedBox(height: 30),
    GestureDetector(
      onTap: () => Navigator.pushNamed(context, 'login'),
      child: const Text(
        'Log Out',
        style: TextStyle(
          color: Colors.white,
          fontSize: 20,
          fontWeight: FontWeight.bold),
        textAlign: TextAlign.end,
      ),
    ),
  ],
),
),
],
);
}
}

```

### Appointment Page:

```

import 'dart:convert';

import 'package:can_dect/Pages/menu.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:text_tools/text_tools.dart';

import 'package:can_dect/api/appointdata.dart';

const Color backgroundColor = Color(0xFF4A4A58);

class Appointdetails extends StatefulWidget {
  final int doctorid;
  String username;

```

```

    int email;
    Appointdetails({
      Key? key,
      required this.doctorid,
      required this.username,
      required this.email,
    }) : super(key: key);

    @override
    _AppointdetailsState createState() => _AppointdetailsState();
  }

class _AppointdetailsState extends State<Appointdetails> {
  List<AppointDetails> appoint = [];

  @override
  void initState() {
    _appointData(widget.doctorid);
    super.initState();
  }

  _appointData(int userID) async {
    appoint = [];
    final http.Response response = await http.get(
      Uri.parse(
        'http://127.0.0.1:8000/appointapi/v1/viewdoctorAppoint/$userID/?format=
json'),
      headers: {"Content-Type": "application/json"},
    );
    if (response.statusCode == 200) {
      List data1 = json.decode(response.body);
      for (var element in data1) {
        appoint.add(AppointDetails.fromMap(element));
      }
      setState(() {});
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: backgroundColor,
      body: Stack(
        children: <Widget>[

```

```

    SizedBox(
      child: Row(
        children: [
          menu(username: widget.username, email: widget.email),
          Column(
            children: <Widget>[
              Material(
                child: SingleChildScrollView(
                  scrollDirection: Axis.vertical,
                  child: SizedBox(
                    height: 670,
                    width: 467,
                    child: Padding(
                      padding: const EdgeInsets.all(10.0),
                      child: Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: [
                          Row(
                            children: [
                              InkWell(
                                child: const Icon(
                                  Icons.arrow_back_ios_new_outlined,
                                  color: Colors.black),
                                onTap: () {
                                  setState(() {
                                    Navigator.pop(context);
                                  });
                                },
                              ),
                              const SizedBox(
                                width: 20,
                              ),
                              const Text(
                                'Can-Dect',
                                style: TextStyle(
                                  fontSize: 35,
                                  fontWeight: FontWeight.normal,
                                ),
                              ),
                            ],
                          ),
                        ],
                      ),
                    ),
                  ),
                ),
              const SizedBox(
                height: 10,

```

```

    ),
    const Text(
      'Appointment Details',
      style: TextStyle(
        fontSize: 25,
        fontWeight: FontWeight.bold,
      ),
    ),
  ),
  const SizedBox(height: 20,),
  RefreshIndicator(
    onRefresh: () async {
      _appointData(widget.doctorid);
    },
    child: SizedBox(
      height: 540,
      child: appoint.isNotEmpty
        ? ListView.separated(
            itemCount: appoint.length,
            itemBuilder: (BuildContext context,
              int index) {
              return Container(
                decoration: BoxDecoration(
                  borderRadius:
                    BorderRadius.circular(15),
                  border: Border.all(
                    color: Colors.black,
                    width: 3,
                  ),
                ),
                child: ListTile(
                  title: Row(
                    crossAxisAlignment:
                      CrossAxisAlignment
                        .start,
                    mainAxisAlignment:
                      MainAxisAlignment
                        .spaceBetween,
                    children: [
                      Text(
                        'Patient: ' +
                          TextTools.toUppercase
FirstLetter(
                                text: appoint[
                                  index]

```

```

                .patient
                .FirstName) +
            " " +
            TextTools.toUpperCase(
FirstLetter(
                text: appoint[
                    index]
                    .patient
                    .LastName),
                style: const TextStyle(
                    color: Colors.black,
                    fontSize: 17,
                ),
            ),
            const SizedBox(
                width: 10,
            ),
            Column(
                children: [
                    Text(
                        appoint[index].date,
                        style:
                            const TextStyle(
                                color:
                                    Colors.black,
                                fontSize: 15,
                            ),
                    ),
                    Text(
                        appoint[index]
                            .times,
                        style:
                            const TextStyle(
                                color:
                                    Colors.black,
                                fontSize: 15,
                            ),
                    ),
                ],
            ),
        ],
    ),
    subtitle: Padding(
        padding:

```





## Report Page:

```
// ignore_for_file: import_of_legacy_library_into_null_safe

import 'dart:convert';
import 'dart:io';

import 'package:can_dect/api/userselectiondata.dart';
import 'package:dropdown_formfield/dropdown_formfield.dart';
import 'package:file_picker/file_picker.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:text_tools/text_tools.dart';

import 'package:can_dect/Pages/menu.dart';

const Color backgroundColor = Color(0xFF4A4A58);

class CreateReport extends StatefulWidget {
  String username;
  int email;
  CreateReport({
    Key? key,
    required this.username,
    required this.email,
  }) : super(key: key);

  @override
  _CreateReportState createState() => _CreateReportState();
}

class _CreateReportState extends State<CreateReport> {
  String _myActivity = '';
  String myValue = '';
  var show = '';
  final diseases = TextEditingController();
  final details = TextEditingController();
  late String tempValue;
  bool patientSelection = false;
  bool emptypatient = false;
  bool statusSelect = false;
  bool emptyadd = false;
  bool emptydisease = false;
  bool emptydate = false;
  List<Userselection> patientdata = [];

```

```

static String _displayStringForOption(Userselection option) =>
    TextTools.toUppercaseFirstLetter(text: option.FirstName) +
    " " +
    TextTools.toUppercaseFirstLetter(text: option.LastName);
List<File> files = [];

@override
void initState() {
    files = [];
    tempValue = '';
    _selectPatient();
    _myActivity = '';
    super.initState();
}

_selectPatient() async {
    patientdata = [];
    final http.Response response = await http.get(
        Uri.parse('http://127.0.0.1:8000/userapi/v1/viewuserReport/?format=json'),
        headers: {"Content-Type": "application/json"},
    );
    if (response.statusCode == 200) {
        List data = json.decode(response.body);
        for (var element in data) {
            patientdata.add(Userselection.fromMap(element));
        }
        setState(() {});
    }
}

Future createReport(int patient, String diseases, String status,
    String additional, List<File> reportPdf) async {
    print(reportPdf);
    final http.Response response = await http.post(
        Uri.parse('http://127.0.0.1:8000/reportapi/v1/report/'),
        headers: {"Content-Type": "application/json"},
        body: json.encode(
            <String, dynamic>{
                'patient_id': patient,
                'status': status,
                'diseases': diseases,
                'adddetails': additional,
                'pdf': reportPdf,
            },
        ),
    );
}

```

```

    ),
  );
  if (response.statusCode == 200) {
    var result = json.decode(response.body).toString();
    if (result == "Successful") {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => CreateReport(
            username: widget.username,
            email: widget.email,
          ),
        ),
      );
    } else {}
  }
}

onclick() async {
  FilePickerResult? result = await FilePicker.platform.pickFiles(
    allowMultiple: true,
    type: FileType.custom,
    allowedExtensions: ['pdf'],
  );

  if (result != null) {
    files = result.paths.map((path) => File(path!)).toList();
  } else {}
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: backgroundColor,
    body: Stack(
      children: <Widget>[
        SizedBox(
          child: Row(
            children: [
              menu(username: widget.username, email: widget.email),
              Column(
                children: <Widget>[
                  Material(
                    child: SingleChildScrollView(

```

```
scrollDirection: Axis.vertical,
child: SizedBox(
  height: 670,
  width: 467,
  child: Center(
    child: Padding(
      padding: const EdgeInsets.only(
        top: 10.0, right: 15.0, left: 15),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Row(
            children: [
              InkWell(
                child: const Icon(
                  Icons.arrow_back_ios_new_outlined,
                  color: Colors.black),
                onTap: () {
                  setState(() {
                    Navigator.pop(context);
                  });
                },
              ),
              const SizedBox(
                width: 20,
              ),
              const Text(
                'Can-Dect',
                style: TextStyle(
                  fontSize: 35,
                  fontWeight: FontWeight.normal,
                ),
              ),
            ],
          ),
          const SizedBox(
            height: 20,
          ),
          const Text(
            'Create Report, ',
            style: TextStyle(
              fontSize: 25,
              fontWeight: FontWeight.normal,
```

```

    ),
  ),
  const SizedBox(
    height: 20,
  ),
  SizedBox(
    child: Padding(
      padding: const EdgeInsets.all(8.0),
      child: Column(
        mainAxisAlignment:
          MainAxisAlignment.start,
        children: [
          Column(
            crossAxisAlignment:
              CrossAxisAlignment.start,
            children: [
              const Text(
                'Patient',
                style: TextStyle(
                  fontSize: 16,
                  fontWeight: FontWeight.normal,
                ),
              ),
              SizedBox(
                width: MediaQuery.of(context)
                  .size
                  .width,
                child:
                  Autocomplete<Userselection>(
                    displayStringForOption:
                      _displayStringForOption,
                    optionsBuilder:
                      (TextEditingValue
                        textEditingValue) {
                        if (textEditingValue.text ==
                          '') {
                          return const Iterable<
                            Userselection>.empty();
                        }
                        return patientdata.where(
                          (Userselection option) {
                            return option
                              .toString()
                              .contains(

```

```

                                                                    textEditingValue
                                                                    .text
                                                                    .toLowerCase())
;

    });
  },
  onSelected: (Userselection
    selection) {
    myValue =
      selection.id.toString();
    // print(
    //       'You just selected
    ${selection.useremail.email}');
  },
),
),
emptypatient
  ? const SizedBox(
    height: 10,
    child: Text(
      'Please select a
Patient!',
      style: TextStyle(
        color: Colors.red,
        fontSize: 16,
      ),
    ),
  )
  : const Text(''),
],
),
const SizedBox(
  height: 20,
),
TextField(
  obscureText: false,
  enabled: true,
  controller: diseases,
  decoration: InputDecoration(
    border: const OutlineInputBorder(
      borderSide: BorderSide(
        color: Colors.blueAccent),
    ),
  ),
  labelText: 'Diseases',

```

```

        errorText: emptydisease
          ? 'Please add a Disease!'
          : null,
        labelStyle: const TextStyle(
          color: Colors.black,
          fontWeight: FontWeight.normal,
          fontSize: 16,
        ),
      ),
    ),
  ),
  const SizedBox(
    height: 20,
  ),
  DropdownFormField(
    titleText: 'Status',
    hintText: 'Status',
    value: _myActivity,
    errorText: statusSelect
      ? 'Please select a Status'
      : null,
    onSave: (value) {
      setState(() {
        _myActivity = value;
        statusSelect = false;
      });
    },
    onChanged: (value) {
      setState(() {
        _myActivity = value;
        statusSelect = false;
      });
    },
    dataSource: const [
      {
        "display": "Positive",
        "value": "Positive",
      },
      {
        "display": "Negative",
        "value": "Negative",
      },
      {
        "display": "Normal",
        "value": "Normal",
      },
    ],
  ),

```



```

    },
    {
      "display": "High",
      "value": "High",
    },
    {
      "display": "Low",
      "value": "Low",
    },
  ],
  textField: 'display',
  valueField: 'value',
),
const SizedBox(
  height: 20,
),
SizedBox(
  height: 100,
  child: TextField(
    obscureText: false,
    enabled: true,
    controller: details,
    maxLength: 50,
    decoration: const InputDecoration(
      border: OutlineInputBorder(
        borderSide: BorderSide(
          color: Colors.blueAccent),
      ),
      labelText: 'Additional Details',
      labelStyle: TextStyle(
        color: Colors.black,
        fontWeight: FontWeight.normal,
        fontSize: 16,
      ),
    ),
  ),
),
),
const SizedBox(
  height: 10,
),
Row(
  children: [
    const Text('Upload Report'),
    const SizedBox(

```

```

        width: 15,
      ),
      ElevatedButton(
        onPressed: () {
          onclick();
        },
        style: ElevatedButton.styleFrom(
          primary: const Color.fromRGBO(
            3, 218, 197, 1),
          fixedSize:
            const Size(130, 60),
          shape: RoundedRectangleBorder(
            borderRadius:
              BorderRadius.circular(
                50),
          ),
        ),
        child: const Text(
          "Upload",
          style: TextStyle(
            fontSize: 18,
            color: Colors.black,
            fontWeight:
              FontWeight.normal),
        ),
      ),
    ],
  ),
  const SizedBox(
    height: 10,
  ),
  Padding(
    padding: const EdgeInsets.only(
      top: 5.0,
      left: 210,
    ),
    child: ElevatedButton(
      onPressed: () {
        if (myValue.isEmpty) {
          setState(() {
            emptypatient = true;
          });
        } else if (_myActivity
          .isEmpty) {

```

```

        setState(() {
          statusSelect = true;
        });
      } else if (diseases
        .text.isEmpty) {
        setState(() {
          emptydisease = true;
        });
      } else {
        setState(() {
          int patients =
            int.parse(myValue);
          String status = _myActivity;
          String dise = diseases.text;
          String additional =
            details.text;
          createReport(
            patients,
            dise,
            status,
            additional,
            files);
        });
      }
    },
    style: ElevatedButton.styleFrom(
      primary: const Color.fromRGBO(
        3, 218, 197, 1),
      fixedSize: const Size(130, 60),
      shape: RoundedRectangleBorder(
        borderRadius:
          BorderRadius.circular(50),
      ),
    ),
    child: Row(
      mainAxisAlignment:
        MainAxisAlignment.center,
      mainAxisAlignment: MainAxisAlignment.min,
      children: const [
        Text(
          "Create",
          style: TextStyle(
            fontSize: 18,
            color: Colors.black,

```



```

final int doctorid;
String username;
int email;
CreateMedicine({
  Key? key,
  required this.doctorid,
  required this.username,
  required this.email,
}) : super(key: key);

@override
_CreateMedicineState createState() => _CreateMedicineState();
}

class _CreateMedicineState extends State<CreateMedicine> {
  String myValue = '';
  var show = '';
  final diseases = TextEditingController();
  final details = TextEditingController();
  late String tempValue;
  bool patientSelection = false;
  bool emptypatient = false;
  bool emptydetail = false;
  bool emptydis = false;
  List<Userselection> patientdata = [];
  static String _displayStringForOption(Userselection option) =>
    TextTools.toUppercaseFirstLetter(text: option.FirstName) +
    " " +
    TextTools.toUppercaseFirstLetter(text: option.LastName);

  @override
  void initState() {
    tempValue = '';
    _selectPatient();
    super.initState();
  }

  _selectPatient() async {
    patientdata = [];
    final http.Response response = await http.get(
      Uri.parse('http://127.0.0.1:8000/userapi/v1/viewuserReport/?format=json'),
      headers: {"Content-Type": "application/json"},
    );
    if (response.statusCode == 200) {

```

```

        List data = json.decode(response.body);
        for (var element in data) {
            patientdata.add(Userselection.fromMap(element));
        }
        setState(() {});
    }
}

Future createMed(
    int patient,
    String dises,
    String meddetails,
) async {
    final http.Response response = await http.post(
        Uri.parse('http://127.0.0.1:8000/medicineapi/v1/medicine/'),
        headers: {"Content-Type": "application/json"},
        body: json.encode(
            <String, dynamic>{
                'doctor_id': widget.doctorid,
                'patient_id': patient,
                'diseases': dises,
                'medicine': meddetails,
            },
        ),
    );
    if (response.statusCode == 200) {
        var result = json.decode(response.body).toString();
        if (result == "Successful") {
            Navigator.push(
                context,
                MaterialPageRoute(
                    builder: (context) => CreateMedicine(
                        doctorid: widget.doctorid,
                        username: widget.username,
                        email: widget.email,
                    ),
                ),
            );
        } else {}
    }
}

@override
Widget build(BuildContext context) {

```

```

return Scaffold(
  backgroundColor: backgroundColor,
  body: Stack(
    children: <Widget>[
      SizedBox(
        child: Row(
          children: [
            menu(username: widget.username, email: widget.email),
            Column(
              children: <Widget>[
                Material(
                  child: SingleChildScrollView(
                    scrollDirection: Axis.vertical,
                    child: SizedBox(
                      height: 670,
                      width: 467,
                      child: Center(
                        child: Padding(
                          padding: const EdgeInsets.only(
                            top: 10.0, right: 15.0, left: 15, bottom: 5),
                          child: Column(
                            mainAxisAlignment: MainAxisAlignment.start,
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                              Row(
                                children: [
                                  InkWell(
                                    child: const Icon(
                                      Icons.arrow_back_ios_new_outlined,
                                      color: Colors.black),
                                    onTap: () {
                                      setState(() {
                                        Navigator.pop(context);
                                      });
                                    },
                                  ),
                                  const SizedBox(
                                    width: 20,
                                  ),
                                  const Text(
                                    'Can-Dect',
                                    style: TextStyle(
                                      fontSize: 35,
                                      fontWeight: FontWeight.normal,

```

```

    ),
  ),
],
),
const SizedBox(
  height: 20,
),
const Text(
  'Create Medicine, ',
  style: TextStyle(
    fontSize: 25,
    fontWeight: FontWeight.normal,
  ),
),
const SizedBox(
  height: 20,
),
SizedBox(
  child: Padding(
    padding: const EdgeInsets.all(8.0),
    child: Column(
      mainAxisAlignment:
        MainAxisAlignment.start,
      children: [
        Column(
          crossAxisAlignment:
            CrossAxisAlignment.start,
          children: [
            const Text(
              'Patient',
              style: TextStyle(
                fontSize: 16,
                fontWeight: FontWeight.normal,
              ),
            ),
            SizedBox(
              width: MediaQuery.of(context)
                .size
                .width,
              child:
                Autocomplete<Userselection>(
                  displayStringForOption:
                    _displayStringForOption,
                  optionsBuilder:

```



```

        (TextEditingValue
          textEditingValue) {
      if (textEditingValue.text ==
        '') {
        return const Iterable<
          Userselection>.empty();
      }
      return patientdata.where(
        (Userselection option) {
          return option
            .toString()
            .contains(
              textEditingValue
                .text
                .toLowerCase())
        });
    },
    onSelected: (Userselection
      selection) {
      myValue =
        selection.id.toString();
      // print(
      //   'You just selected
    ${selection.useremail.email}');
    },
  ),
  emptypatient
    ? const SizedBox(
      height: 10,
      child: Text(
        'Please select a
Patient!',
        style: TextStyle(
          color: Colors.red,
          fontSize: 16,
        ),
      ),
    )
    : const Text(''),
  ],
),
const SizedBox(

```

```
        height: 20,
      ),
      TextField(
        obscureText: false,
        enabled: true,
        controller: diseases,
        decoration: InputDecoration(
          border: const OutlineInputBorder(
            borderSide: BorderSide(
              color: Colors.blueAccent),
          ),
          labelText: 'Disease',
          labelStyle: const TextStyle(
            color: Colors.black,
            fontWeight: FontWeight.normal,
            fontSize: 16,
          ),
          errorText: emptydis
            ? 'Please provide a Disease!'
            : null,
        ),
      ),
      const SizedBox(
        height: 20,
      ),
      SizedBox(
        height: 90,
        child: TextField(
          obscureText: false,
          enabled: true,
          controller: details,
          maxLength: 50,
          decoration: InputDecoration(
            border:
              const OutlineInputBorder(
                borderSide: BorderSide(
                  color: Colors.blueAccent),
              ),
            labelText: 'Medicine',
            labelStyle: const TextStyle(
              color: Colors.black,
              fontWeight: FontWeight.normal,
              fontSize: 16,
            ),
          ),
        ),
      ),
    ),
  ),
),
```

Details!'

```

        errorText: emptydetail
        ? 'Please provide Medicine
        : null,
    ),
  ),
),
const SizedBox(
  height: 20,
),
Padding(
  padding: const EdgeInsets.only(
    top: 10,
  ),
  child: ElevatedButton(
    onPressed: () {
      if (myValue.isEmpty) {
        emptypatient = true;
      } else if (diseases
        .text.isEmpty) {
        emptydis = true;
      } else if (details
        .text.isEmpty) {
        emptydetail = true;
      } else {
        setState(() {
          int patients =
            int.parse(myValue);
          String meddetail =
            details.text;
          String dises =
            diseases.text;

          createMed(patients, dises,
            meddetail);
        });
      }
    },
    style: ElevatedButton.styleFrom(
      primary: const Color.fromRGBO(
        3, 218, 197, 1),
      fixedSize: const Size(130, 60),
      shape: RoundedRectangleBorder(
        borderRadius:

```



## Notify Page:

```
// ignore_for_file: import_of_legacy_library_into_null_safe

import 'dart:convert';

import 'package:can_dect/api/userselectiondata.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:text_tools/text_tools.dart';

import 'package:can_dect/Pages/menu.dart';

Color backgroundColor = Color(0xFF4A4A58);

class CreateNotification extends StatefulWidget {
  final int doctorid;
  String username;
  int email;
  CreateNotification({
    Key? key,
    required this.doctorid,
    required this.username,
    required this.email,
  }) : super(key: key);

  @override
  _CreateNotificationState createState() => _CreateNotificationState();
}

class _CreateNotificationState extends State<CreateNotification> {
  String myValue = '';
  var show = '';
  final details = TextEditingController();
  late String tempValue;
  bool patientSelection = false;
  bool emptypatient = false;
  bool emptydetail = false;

  List<Userselection> patientdata = [];
  static String _displayStringForOption(Userselection option) =>
    TextTools.toUppercaseFirstLetter(text: option.FirstName) +
    " " +
    TextTools.toUppercaseFirstLetter(text: option.LastName);
}
```

```

@override
void initState() {
  _selectPatient();
  super.initState();
}

Future createNotify(
  int patient,
  String notifydetails,
) async {
  final http.Response response = await http.post(
    Uri.parse('http://127.0.0.1:8000/notifyapi/v1/notify/'),
    headers: {"Content-Type": "application/json"},
    body: json.encode(
      <String, dynamic>{
        'doctor_id': widget.doctorid,
        'patient_id': patient,
        'message': notifydetails,
      },
    ),
  );
  if (response.statusCode == 200) {
    var result = json.decode(response.body).toString();
    if (result == "Successful") {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => CreateNotification(
            doctorid: widget.doctorid,
            email: widget.email,
            username: widget.username,
          ),
        ),
      );
    } else {}
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: backgroundColor,
    body: Stack(
      children: <Widget>[

```

```

SizedBox(
  child: Row(
    children: [
      menu(username: widget.username, email: widget.email),
      Column(
        children: <Widget>[
          Material(
            child: SingleChildScrollView(
              scrollDirection: Axis.vertical,
              child: SizedBox(
                height: 670,
                width: 467,
                child: Center(
                  child: Padding(
                    padding: const EdgeInsets.only(
                      top: 10.0, right: 15.0, left: 15, bottom: 5),
                    child: Column(
                      mainAxisAlignment: MainAxisAlignment.start,
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        Row(
                          children: [
                            InkWell(
                              child: const Icon(
                                Icons.arrow_back_ios_new_outlined,
                                color: Colors.black),
                              onTap: () {
                                setState(() {
                                  Navigator.pop(context);
                                });
                              },
                            ),
                            const SizedBox(
                              width: 20,
                            ),
                            const Text(
                              'Can-Dect',
                              style: TextStyle(
                                fontSize: 35,
                                fontWeight: FontWeight.normal,
                              ),
                            ),
                          ],
                        ),
                      ],
                    ),
                  ),
                ),
              ),
            ),
          ),
        ],
      ),
    ],
  ),
)

```

```

const SizedBox(
  height: 20,
),
const Text(
  'Create Notification, ',
  style: TextStyle(
    fontSize: 25,
    fontWeight: FontWeight.normal,
  ),
),
const SizedBox(
  height: 20,
),
SizedBox(
  child: Padding(
    padding: const EdgeInsets.all(8.0),
    child: Column(
      mainAxisAlignment:
        MainAxisAlignment.start,
      children: [
        Column(
          crossAxisAlignment:
CrossAxisAlignment.start,

          children: [
            const Padding(
              padding: EdgeInsets.only(
                right: 0.0),
              child: Text(
                'Patient',
                style: TextStyle(
                  fontSize: 16,
                  fontWeight:
                    FontWeight.normal,
                ),
              ),
            ),
          ],
        ),
        SizedBox(
          width: MediaQuery.of(context)
            .size
            .width,
          child:
            Autocomplete<Userselection>(
              displayStringForOption:
                _displayStringForOption,

```



```

optionsBuilder:
  (TextEditingValue
    textEditingValue) {
    if (textEditingValue.text ==
      '') {
      return const Iterable<
        Userselection>.empty();
    }
    return patientdata.where(
      (Userselection option) {
        return option
          .toString()
          .contains(
            textEditingValue
              .text
              .toLowerCase())
      });
  },
onSelected: (Userselection
  selection) {
  myValue =
    selection.id.toString();
  // print(
  //   'You just selected
  ${selection.useremail.email}');
  },
),
),
emptypatient
  ? const SizedBox(
    height: 10,
    child: Text(
      'Please select a
Patient!',
      style: TextStyle(
        color: Colors.red,
        fontSize: 16,
      ),
    ),
  )
: const Text(''),
],
),

```

```

const SizedBox(
  height: 20,
),
SizedBox(

  child: TextField(
    obscureText: false,
    enabled: true,
    controller: details,
    maxLength: 50,
    decoration: InputDecoration(
      border:
        const OutlineInputBorder(
          borderSide: BorderSide(
            color: Colors.blueAccent),
        ),
      labelText: 'Message',
      labelStyle: const TextStyle(
        color: Colors.black,
        fontWeight: FontWeight.normal,
        fontSize: 16,
      ),
      errorText: emptydetail
        ? 'Please provide a Message!'
        : null,
    ),
  ),
),
const SizedBox(
  height: 20,
),
Padding(
  padding: const EdgeInsets.only(
    top: 20.0,
    left: 80,
  ),
  child: ElevatedButton(
    onPressed: () {
      if (myValue.isEmpty) {
        setState(() {
          emptypatient = true;
        });
      } else if (details
        .text.isEmpty) {

```

```

        setState(() {
          emptydetail = true;
        });
      } else {
        setState(() {
          emptydetail = false;
          emptypatient = false;
          int patients =
            int.parse(myValue);
          String notifydetail =
            details.text;
          createNotify(
            patients, notifydetail);
        });
      }
    },
    style: ElevatedButton.styleFrom(
      primary: const Color.fromRGBO(
        3, 218, 197, 1),
      fixedSize: const Size(130, 60),
      shape: RoundedRectangleBorder(
        borderRadius:
          BorderRadius.circular(50),
      ),
    ),
    child: Row(
      mainAxisAlignment:
        MainAxisAlignment.center,
      mainAxisSize: MainAxisSize.min,
      children: const [
        Text(
          "Notify",
          style: TextStyle(
            fontSize: 18,
            color: Colors.black,
            fontWeight:
              FontWeight.normal),
        ),
      ],
    ),
  ),
],
),
),
),
),
],
),
),

```

```

    ),
    ),
    ],
    ),
    ),
    ),
    ),
    ),
    ),
    ),
    ],
    ),
    ],
    ),
    ),
    ],
    ),
    );
}

_selectPatient() async {
  patientdata = [];
  final http.Response response = await http.get(
    Uri.parse('http://127.0.0.1:8000/userapi/v1/viewuserReport/?format=json'),
    headers: {"Content-Type": "application/json"},
  );
  if (response.statusCode == 200) {
    List data = json.decode(response.body);
    for (var element in data) {
      patientdata.add(Userselection.fromMap(element));
    }
    setState(() {});
  }
}
}

```

## Mobile Application Code:

## Login Page

```
import 'dart:convert';
import 'package:ecom/pages/Homepage.dart';
import 'package:ecom/pages/Signuppage.dart';
import 'package:ecom/pages/verify.dart';
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:http/http.dart' as http;

import 'Signup.dart';

// ignore: camel_case_types
class Loginpage extends StatefulWidget {
  @override
  _LoginpageState createState() => _LoginpageState();
}

// ignore: camel_case_types
class _LoginpageState extends State<Loginpage> {
  final username = TextEditingController();
  final password = TextEditingController();
  bool isWro = false;
  bool emptyuser = false;
  bool emptypw = false;
  bool pwseen = false;

  Future login(String username, String password) async {
    final http.Response response = await http.post(
      Uri.parse('http://10.0.2.2:8000/userapi/v1/userlogin/'),
      headers: {"Content-Type": "application/json"},
      body: json.encode(
        <String, dynamic>{
          'email': username,
          'password': password,
        },
      ),
    );
    if (response.statusCode == 200) {
      var result = json.decode(response.body);
      print(result);
      if (result == "invalid") {
        setState(() {
```

```

        isWro = true;
    });
} else if ((result["is_Verified"]).toString() == "false") {
    setState(() {
        isWro = false;
    });
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => Verify(
                email: username,
            ),
        ),
    );
} else if ((result["has_data"]).toString() == "false") {
    setState(() {
        isWro = false;
    });
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => Signupp(
                email: username,
            ),
        ),
    );
} else {
    setState(() {
        isWro = false;
    });
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => homepage(
                email: username,
            ),
        ),
    );
}
}
}

@override
Widget build(BuildContext context) {

```

```

return Scaffold(
  body: Stack(
    children: [
      SingleChildScrollView(
        scrollDirection: Axis.vertical,
        child: Container(
          height: MediaQuery.of(context).size.height,
          width: MediaQuery.of(context).size.width,
          child: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              mainAxisAlignment: MainAxisAlignment.end,
              children: [
                Column(
                  children: [
                    Center(
                      child: Image.asset(
                        'lib/images/can.png',
                        width: MediaQuery.of(context).size.width,
                        height: 280,
                      ),
                    ),
                  ],
                ),
                Text(
                  "Can-Dect",
                  style: TextStyle(
                    color: Colors.black,
                    fontSize: 50,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                Row(
                  children: [
                    Text(
                      "New to Can-Dect?",
                      style: TextStyle(fontSize: 18),
                    ),
                    GestureDetector(
                      onTap: () => Navigator.pushNamed(context, 'Signup'),
                      child: Text(
                        ' Sign Up',
                        style: TextStyle(
                          color: Colors.blueAccent[700], fontSize: 22),
                      ),
                    ),
                  ],
                ),
              ],
            ),
          ),
        ),
      ),
    ],
  ),
);

```

```

    ),
  ),
],
),
PreferredSize(
  height: 20,
),
Center(
  child: Text(
    "----- OR -----",
    style: TextStyle(fontSize: 30),
  ),
),
PreferredSize(
  height: 20,
),
Padding(
  padding: const EdgeInsets.all(12.0),
  child: Column(
    children: [
      TextField(
        controller: username,
        obscureText: false,
        onChanged: (val) {
          emptyuser = false;
          isWro = false;
        },
        style: TextStyle(color: Colors.blue),
        decoration: InputDecoration(
          border: OutlineInputBorder(
            borderSide:
              new BorderSide(color: Colors.blueAccent),
          ),
          labelText: 'Email',
          hintText: 'Enter Email',
          errorText: emptyuser
            ? 'Please enter your username!'
            : null,
        ),
      ),
    ],
  ),
PreferredSize(
  height: 15,
),
TextField(

```



```

        controller: password,
        obscureText: pwseen ? false : true,
        onChanged: (val) {
            emptypw = false;
            isWro = false;
        },
        decoration: InputDecoration(
            border: OutlineInputBorder(
                borderSide: new BorderSide(color: Colors.blue),
            ),
            labelText: 'Password',
            hintText: 'Enter Password',
            errorText: emptypw
                ? 'Please enter your password!'
                : null,
            suffixIcon: IconButton(
                icon: pwseen
                    ? const Icon(Icons.visibility)
                    : const Icon(Icons.visibility_off),
                onPressed: () {
                    setState(() {
                        if (pwseen == true) {
                            pwseen = false;
                        } else {
                            pwseen = true;
                        }
                    });
                },
            ),
        ),
    ),
    isWro
        ? Text(
            'Username or password is invalid!',
            style: TextStyle(
                color: Colors.red, fontSize: 14),
        )
        : Text(''),
    ],
),
),
Padding(
    padding: const EdgeInsets.only(left: 200.0),
    child: Column(

```

```

        children: [
          GestureDetector(
            onTap: () =>
              Navigator.pushNamed(context, 'forgetemail'),
            child: Text(
              'Forgot Password?',
              style: TextStyle(
                color: Colors.blueAccent[700], fontSize: 20),
              textAlign: TextAlign.end,
            ),
          ),
        ],
      ),
    ),
    SizedBox(
      height: 20,
    ),
    Center(
      child: Padding(
        padding: const EdgeInsets.only(
          top: 10.0,
        ),
        child: ElevatedButton(
          onPressed: () {
            if (username.text.isEmpty) {
              setState(() {
                emptyuser = true;
              });
            } else if (password.text.isEmpty) {
              setState(() {
                emptypw = true;
              });
            } else {
              setState(() {
                final String txtusername = username.text;
                final String txtpassword = password.text;
                login(txtusername, txtpassword);
              });
            }
          },
          style: ElevatedButton.styleFrom(
            primary: Color.fromRGB(3, 218, 197, 1),
            fixedSize: Size(140, 60),
            shape: RoundedRectangleBorder(

```

```
borderRadius: BorderRadius.circular(50),
    ),
  ),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.center,
    mainAxisSize: MainAxisSize.min,
    children: [
      Icon(
        FontAwesomeIcons.user,
        color: Colors.black,
      ),
      SizedBox(
        width: 5,
      ),
      Text(
        "Login",
        style: TextStyle(
          fontSize: 25,
          color: Colors.black,
          fontWeight: FontWeight.normal,
        ),
      ),
    ],
  ),
],
),
),
),
),
),
],
),
),
),
),
],
),
);
}
```

## Signup Page:

```

import 'dart:convert';

import 'package:ecom/pages/verify.dart';
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:http/http.dart' as http;

class Signup extends StatefulWidget {
  const Signup({
    Key? key,
  }) : super(key: key);

  @override
  _SignupState createState() => _SignupState();
}

class _SignupState extends State<Signup> {
  final emailco = TextEditingController();
  final personNameco = TextEditingController();
  final passwordco = TextEditingController();
  final rePassword = TextEditingController();
  bool pass = true;
  bool empresent = false;
  bool emptypw = false;
  bool emptyem = false;
  bool lesspw = false;
  bool invaemail = false;
  bool pwseen = false;
  bool pwseen1 = false;

  Future createuser(String email, String password) async {
    final http.Response response = await http.post(
      Uri.parse('http://10.0.2.2:8000/userapi/v1/Signup/'),
      headers: {"Content-Type": "application/json"},
      body: json.encode(
        <String, dynamic>{
          'email': email,
          'password': password,
        },
      ),
    );

    if (response.statusCode == 200) {

```

```

var result = json.decode(response.body);
if (result == "Successful") {
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => Verify(
        email: email,
      ),
    ),
  );
} else if (result == "Email exists") {
  empresent = true;
} else {
  invaemail = true;
}
} else {}
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Stack(
      children: [
        SingleChildScrollView(
          scrollDirection: Axis.vertical,
          child: Container(
            height: MediaQuery.of(context).size.height,
            width: MediaQuery.of(context).size.width,
            child: Padding(
              padding: const EdgeInsets.all(14.0),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    'Can-Dect Sign Up',
                    style: TextStyle(
                      fontSize: 30,
                      fontWeight: FontWeight.normal,
                    ),
                  ),
                  SizedBox(
                    height: 25,
                  ),
                ],
              ),
            ),
          ),
        ),
      ],
    ),
  );
}

```

[illegible]

```
TextField(
    obscureText: pwseen ? false : true,
    controller: passwordco,
    keyboardType: TextInputType.visiblePassword,
    style: TextStyle(color: Colors.blue),
    onChanged: (val) {
        emptypw = false;
        lesspw = false;
    },
    decoration: InputDecoration(
        border: OutlineInputBorder(
            borderSide:
                new BorderSide(color: Colors.blueAccent),
        ),
        labelText: 'Password',
        hintText: 'Enter Password',
        errorText: emptypw
            ? 'Please enter 6 character long password!'
            : lesspw
                ? 'Password must be 6 character long!'
                : null,
        suffixIcon: Container(
            child: IconButton(
                icon: pwseen
                    ? const Icon(Icons.visibility)
                    : const Icon(Icons.visibility_off),
                onPressed: () {
                    setState(() {
                        if (pwseen == true) {
                            pwseen = false;
                        } else {
                            pwseen = true;
                        }
                    });
                },
            ),
        ),
    ),
),
 SizedBox(
    height: 30,
),
 TextField(
    obscureText: pwseen1 ? false : true,
```

```

        controller: rePassword,
        style: TextStyle(color: Colors.blue),
        onChanged: (val) {
            pass = true;
        },
        decoration: InputDecoration(
            border: OutlineInputBorder(
                borderSide:
                    new BorderSide(color: Colors.blueAccent),
            ),
            labelText: 'Re-Password',
            hintText: 'Confirm Password',
            errorText:
                pass ? null : "Password doesn't match!",
            suffixIcon: IconButton(
                icon: pwseen1
                    ? const Icon(Icons.visibility)
                    : const Icon(Icons.visibility_off),
                onPressed: () {
                    setState(() {
                        if (pwseen1 == true) {
                            pwseen1 = false;
                        } else {
                            pwseen1 = true;
                        }
                    });
                },
            ),
        ),
    ),
    SizedBox(
        height: 40,
    ),
    Padding(
        padding: const EdgeInsets.only(
            top: 10.0,
            left: 210,
        ),
        child: ElevatedButton(
            onPressed: () {
                if (emailco.text == '') {
                    setState(() {
                        emptyem = true;
                    });
                }
            },
        ),
    ),

```



```

    } else if (passwordco.text.isEmpty) {
        setState(() {
            emptypw = true;
        });
    } else {
        if (passwordco.text.length < 6) {
            setState(() {
                lesspw = true;
            });
        } else {
            if (passwordco.text == rePassword.text) {
                setState(
                    () {
                        final String textemail =
                            emailco.text;

                        final String textpassword =
                            passwordco.text;
                        pass = true;
                        createuser(textemail, textpassword);
                    },
                );
            } else {
                setState(() {
                    pass = false;
                });
            }
        }
    }
},
style: ElevatedButton.styleFrom(
    primary: Color.fromRGBO(3, 218, 197, 1),
    fixedSize: Size(140, 60),
    shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(50),
    ),
),
child: Row(
    mainAxisAlignment: MainAxisAlignment.center,
    mainAxisSize: MainAxisSize.min,
    children: [
        Text(
            "Next",
            style: TextStyle(

```

```
fontSize: 25,  
color: Colors.black,  
fontWeight: FontWeight.normal),  
),  
SizedBox(  
width: 5,  
),  
Icon(  
FontAwesomeIcons.angleRight,  
color: Colors.black,  
),  
],  
),  
),  
),  
],  
),  
),  
),  
],  
),  
),  
),  
],  
),  
),  
],  
),  
);  
}  
}
```

## Verify Email Page:

```
// ignore_for_file: import_of_legacy_library_into_null_safe

import 'dart:convert';
import 'package:ecom/pages/Signup.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:flutter_pin_code_fields/flutter_pin_code_fields.dart';

class Verify extends StatefulWidget {
  final String email;
  const Verify({Key? key, required this.email}) : super(key: key);
}
```

```

    @override
    _VerifyState createState() => _VerifyState();
}

class _VerifyState extends State<Verify> {
    final codec = TextEditingController();
    bool matchcode = false;
    bool emptycode = false;
    int codde = 0;

    @override
    void initState() {
        sendcode(widget.email);
        super.initState();
    }

    sendcode(String email) async {
        final http.Response response = await http.post(
            Uri.parse('http://10.0.2.2:8000/userapi/v1/verify/'),
            headers: {"Content-Type": "application/json"},
            body: json.encode(
                <String, dynamic>{
                    'email': email,
                },
            ),
        );
        if (response.statusCode == 200) {
            setState(() {
                codde = json.decode(response.body);
            });
        }
    }

    sendVerify(String email) async {
        final http.Response response1 = await http.put(
            Uri.parse(
                'http://10.0.2.2:8000/userapi/v1/getverify/$email/?format=json'),
            headers: {"Content-Type": "application/json"},
        );
        if (response1.statusCode == 200) {
            var cod = json.decode(response1.body);
            if (cod == 'verified') {
                Navigator.push(
                    context,

```

```
        MaterialPageRoute(
          builder: (context) => Signupp(email: widget.email)),
        );
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      children: [
        Scaffold(
          body: Padding(
            padding: const EdgeInsets.all(15.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                Text(
                  'Verify Email',
                  style: TextStyle(fontSize: 35),
                ),
                SizedBox(
                  height: 25,
                ),
                Text(
                  'Please check Email for 4 digit Verification Code!',
                  style: TextStyle(fontSize: 25),
                ),
                SizedBox(
                  height: 50,
                ),
                Container(
                  child: Padding(
                    padding: const EdgeInsets.all(25.0),
                    child: PinCodeFields(
                      length: 4,
                      animationDuration: const Duration(milliseconds: 200),
                      animationCurve: Curves.easeInOut,
                      switchInAnimationCurve: Curves.easeIn,
                      switchOutAnimationCurve: Curves.easeOut,
                      animation: Animations.SlideInDown,
                      controller: codec,
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      ],
    );
  }
}
```

```

    ),
  ),
),
matchcode
  ? SizedBox(
    height: 20,
    child: Text(
      "Code doesn't match",
      style: TextStyle(fontSize: 18, color: Colors.red),
    ),
  ),
  : Text(''),
SizedBox(
  height: 20,
),
Column(
  children: [
    Padding(
      padding: const EdgeInsets.only(left: 100),
      child: Row(
        crossAxisAlignment: CrossAxisAlignment.end,
        children: [
          Icon(
            Icons.restart_alt_rounded,
            color: Colors.blueAccent,
            size: 30,
          ),
          GestureDetector(
            onTap: () => sendcode(widget.email),
            child: Text(
              ' Resend Code!',
              style: TextStyle(
                color: Colors.blueAccent[700], fontSize: 22),
            ),
          ),
          SizedBox(
            height: 25,
          ),
        ],
      ),
    ),
  ],
),
),
],
),
SizedBox(

```

```

        height: 100,
      ),
    ],
  ),
),
floatingActionButton: Padding(
  padding: const EdgeInsets.only(
    bottom: 70,
    right: 20,
  ),
  child: SizedBox(
    width: 130,
    height: 53,
    child: FloatingActionButton.extended(
      onPressed: () {
        if (codec.text.isNotEmpty) {
          setState(() {
            emptycode = false;
            if (int.parse(codec.text) == code) {
              setState(() {
                emptycode = false;
                matchcode = false;
                sendVerify(widget.email);
              });
            } else {
              setState(() {
                matchcode = true;
              });
            }
          });
        } else {
          setState(() {
            emptycode = true;
          });
        }
      },
      label: Row(
        children: <Widget>[
          Icon(
            Icons.verified,
            color: Colors.black,
            size: 30,
          ),
          SizedBox(

```

```

        width: 10,
      ),
      Text(
        "Verify ",
        style: TextStyle(
          fontSize: 25,
          fontWeight: FontWeight.normal,
          color: Colors.black),
      ),
    ],
  ),
  backgroundColor: Color.fromRGBO(3, 218, 197, 1),
),
),
),
),
],
);
}
}

```

### User Details Page:

```

// ignore_for_file: import_of_legacy_library_into_null_safe

import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:dropdown_formfield/dropdown_formfield.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:http/http.dart' as http;

class Signupp extends StatefulWidget {
  final String email;
  const Signupp({
    Key? key,
    required this.email,
  }) : super(key: key);

  @override
  _SignuppState createState() => _SignuppState();
}

class _SignuppState extends State<Signupp> {
  String _myActivity = '';

```

```

String _myActivityResult = '';
final formKey = new GlobalKey<FormState>();
final phonec = TextEditingController();
final fname = TextEditingController();
final lname = TextEditingController();
final agec = TextEditingController();
final addressc = TextEditingController();
bool presentphone = false;
bool emptyphone = false;
bool emptyage = false;
bool emptyfname = false;
bool emptylname = false;
bool emptyaddress = false;
late int data;

void initState() {
  _userdata(widget.email);
  _myActivity = '';
  _myActivityResult = '';
  super.initState();
}

_userdata(String email) async {
  final http.Response response = await http.get(
    Uri.parse(
      'http://10.0.2.2:8000/userapi/v1/viewSignupDetail/$email/?format=json'
    ),
    headers: {"Content-Type": "application/json"},
  );
  if (response.statusCode == 200) {
    final data1 = json.decode(response.body);
    setState(() {
      data = data1['id'];
    });
  }
}

_saveForm() {
  var form = formKey.currentState;
  if (form!.validate()) {
    form.save();
    setState(() {
      _myActivityResult = _myActivity;
    });
  }
}

```



```

    }
  }

Future addUser(int xemail, String textfname, String textlname,
    String textgender, String textAddre, int intage, int intphn) async {
  final http.Response response = await http.post(
    Uri.parse('http://10.0.2.2:8000/userapi/v1/detail/'),
    headers: {"Content-Type": "application/json"},
    body: json.encode(
      <String, dynamic>{
        'useremail_id': xemail,
        'firstname': textfname,
        'lastname': textlname,
        'age': intage,
        'phone': intphn,
        'gender': textgender,
        'address': textAddre,
      },
    ),
  );

  if (response.statusCode == 200) {
    var result = json.decode(response.body);
    if (result == "present phone") {
      presentphone = true;
    } else {
      presentphone = false;
      Navigator.pushNamed(context, 'login');
    }
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Stack(
      children: [
        SingleChildScrollView(
          scrollDirection: Axis.vertical,
          child: Container(
            height: MediaQuery.of(context).size.height,
            width: MediaQuery.of(context).size.width,
            child: Padding(
              padding: const EdgeInsets.all(14.0),

```

```
child: Column(  
  mainAxisAlignment: MainAxisAlignment.center,  
  crossAxisAlignment: CrossAxisAlignment.start,  
  children: [  
    Text(  
      'Can-Dect Sign Up',  
      style: TextStyle(  
        fontSize: 30,  
        fontWeight: FontWeight.normal,  
      ),  
    ),  
    SizedBox(  
      height: 25,  
    ),  
    Text(  
      '2 of 2 Steps',  
      style: TextStyle(  
        fontSize: 20,  
        fontWeight: FontWeight.normal,  
      ),  
    ),  
    SizedBox(  
      height: 35,  
    ),  
    Container(  
      child: Padding(  
        padding: const EdgeInsets.all(8.0),  
        child: Column(  
          mainAxisAlignment: MainAxisAlignment.end,  
          children: [  
            Row(  
              children: [  
                SizedBox(  
                  width: 168,  
                  child: TextField(  
                    obscureText: false,  
                    controller: fname,  
                    onChanged: (val) {  
                      emptyfname = false;  
                    },  
                    style: TextStyle(color: Colors.blue),  
                    decoration: InputDecoration(  
                      border: OutlineInputBorder(  
                        borderSide: new BorderSide(  

```

```

        color: Colors.blueAccent),
    ),
    labelText: 'First Name',
    hintText: 'Name',
    errorText: emptyfname
        ? 'Please provide your Name!'
        : null,
    ),
    ),
    ),
    SizedBox(
        width: 10,
    ),
    SizedBox(
        width: 168,
        child: TextField(
            obscureText: false,
            controller: lname,
            onChanged: (val) {
                emptylname = false;
            },
            style: TextStyle(color: Colors.blue),
            decoration: InputDecoration(
                border: OutlineInputBorder(
                    borderSide: new BorderSide(
                        color: Colors.blueAccent),
                ),
            ),
            labelText: 'Last Name',
            hintText: 'Family Name',
            errorText: emptylname
                ? 'Please provide your Family Name'
                : null,
        ),
    ),
    ),
    ),
    ],
    ),
    SizedBox(
        height: 30,
    ),
    TextField(
        obscureText: false,
        controller: phonec,
        onChanged: (val) {

```

[illegible]

```

    ),
    Row(
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        SizedBox(
          width: 140,
          child: TextField(
            obscureText: false,
            controller: agec,
            onChanged: (val) {
              emptyage = false;
            },
            keyboardType: TextInputType.number,
            style: TextStyle(color: Colors.blue),
            decoration: InputDecoration(
              border: OutlineInputBorder(
                borderSide: new BorderSide(
                  color: Colors.blueAccent),
              ),
              labelText: 'Age',
              hintText: 'Age',
              errorText: emptyphone
                ? 'Please provide your age!'
                : null,
            ),
          ),
        ),
        SizedBox(
          width: 10,
        ),
        SizedBox(
          width: 160,
          height: 100,
          child: DropdownFormField(
            titleText: 'Gender',
            hintText: 'Gender',
            value: _myActivity,
            errorText: 'Please select a gender',
            onSave: (value) {
              setState(() {
                _myActivity = value;
              });
            },
            onChanged: (value) {

```

```

        setState(() {
          _myActivity = value;

        });
      },
      dataSource: [
        {
          "display": "Male",
          "value": "Male",
        },
        {
          "display": "Female",
          "value": "Female",
        },
        {
          "display": "Others",
          "value": "Others",
        },
      ],
      textField: 'display',
      valueField: 'value',
    ),
  ),
],
),
 SizedBox(
  height: 40,
),
 Padding(
  padding:
    const EdgeInsets.only(top: 10.0, left: 210),
  child: ElevatedButton(
    onPressed: () {
      if (fname.text.isEmpty) {
        setState(() {
          emptyfname = true;
        });
      } else if (lname.text.isEmpty) {
        setState(() {
          emptylname = true;
        });
      } else if (agec.text.isEmpty) {
        setState(() {
          emptyage = true;

```

```

    ));
} else if (addressc.text.isEmpty) {
    setState(() {
        emptyaddress = true;
    });
} else if (phonec.text.isEmpty) {
    setState(() {
        emptyphone = true;
    });
} else {
    setState(
        () {
            final String textfname = fname.text;
            final String textlname = lname.text;
            final String textgender =
                _myActivityResult;
            final String textAddre = addressc.text;
            final int intage = int.parse(agec.text);
            final int intphn =
                int.parse(phonec.text);

            addUser(
                data,
                textfname,
                textlname,
                textgender,
                textAddre,
                intage,
                intphn,
            );
        },
    );
}
},
style: ElevatedButton.styleFrom(
    primary: Color.fromRGBO(3, 218, 197, 1),
    fixedSize: Size(140, 60),
    shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(50),
    ),
),
child: Row(
    mainAxisAlignment: MainAxisAlignment.center,
    mainAxisAlignment: MainAxisAlignment.min,

```

```
children: [
    Icon(
      FontAwesomeIcons.smile,
      color: Colors.yellow,
    ),
    SizedBox(
      width: 5,
    ),
    Text(
      "Done",
      style: TextStyle(
        fontSize: 25,
        color: Colors.black,
        fontWeight: FontWeight.normal),
    ),
  ],
),
],
),
),
),
],
),
),
],
),
),
),
],
),
);
}
```

Home Page:

```
// ignore_for_file: import_of_legacy_library_into_null_safe, camel_case_types

import 'dart:convert';
import 'package:ecom/api/notifydetails.dart';
import 'package:ecom/pages/Appointment.dart';
import 'package:ecom/pages/Userdetail.dart';
import 'package:ecom/pages/bookings.dart';
import 'package:ecom/pages/changepass.dart';
```



```
import 'package:ecom/pages/medicine.dart';
import 'package:ecom/pages/notifications.dart';
import 'package:ecom/pages/report.dart';
import 'package:text_tools/text_tools.dart';
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:line_icons/line_icons.dart';
import 'package:http/http.dart' as http;

final Color backgroundColor = Color(0xFF4A4A58);

class homepage extends StatefulWidget {
  final String email;
  const homepage({Key? key, required this.email}) : super(key: key);

  @override
  _homepageState createState() => _homepageState();
}

class _homepageState extends State<homepage>
  with SingleTickerProviderStateMixin {
  bool isCollapsed = true;
  late double screenWidth, screenHeight;
  final Duration duration = const Duration(milliseconds: 300);
  late AnimationController _controller;
  late Animation<double> _scaleAnimation;
  late Animation<double> _menuScaleAnimation;
  late Animation<Offset> _slideAnimation;
  late final int data;
  List userss = [];
  late String username;

  @override
  void initState() {
    username = '';
    _user(widget.email);
    _controller = AnimationController(vsync: this, duration: duration);
    _scaleAnimation = Tween<double>(begin: 1, end: 0.8).animate(_controller);
    _menuScaleAnimation =
      Tween<double>(begin: 0.5, end: 1).animate(_controller);
    _slideAnimation = Tween<Offset>(begin: Offset(-1, 0), end: Offset(0, 0))
      .animate(_controller);
    super.initState();
  }
}
```

```

_user(String email) async {
  final http.Response response = await http.get(
    Uri.parse(
      'http://10.0.2.2:8000/userapi/v1/viewSignupDetail/$email/?format=json')
  ),
  headers: {"Content-Type": "application/json"},
  );
  if (response.statusCode == 200) {
    final data1 = json.decode(response.body);
    setState(() {
      data = data1['id'];
    });

    final http.Response response1 = await http.get(
      Uri.parse(
        'http://10.0.2.2:8000/userapi/v1/viewUserdetail/$data/?format=json'),
      headers: {"Content-Type": "application/json"},
    );

    if (response1.statusCode == 200) {
      var users = json.decode(response1.body);
      setState(() {
        userss.add(users);
        username =
          TextTools.toUppercaseFirstLetter(text: userss[0]['FirstName']) +
            " " +
            TextTools.toUppercaseFirstLetter(text: userss[0]['LastName']);
      });
    }
  }
}

_logout() async {
  final http.Response response = await http.post(
    Uri.parse('http://10.0.2.2:8000/userapi/v1/userlogout/?format=json'),
    headers: {"Content-Type": "application/json"},
  );
  if (response.statusCode == 200) {
    final data1 = json.decode(response.body);
    if (data1 == 'Logged Out') {
      Navigator.pushNamed(context, 'login');
    }
  }
}

```

```

}

@override
void dispose() {
  _controller.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  Size size = MediaQuery.of(context).size;
  screenHeight = size.height;
  screenWidth = size.width;

  return Scaffold(
    backgroundColor: backgroundColor,
    body: Stack(
      children: <Widget>[
        menu(context),
        dashboard(context),
      ],
    ),
  );
}

Widget menu(context) {
  return SlideTransition(
    position: _slideAnimation,
    child: ScaleTransition(
      scale: _menuScaleAnimation,
      child: Padding(
        padding: const EdgeInsets.only(right: 16.0),
        child: Align(
          alignment: Alignment.centerRight,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.min,
            mainAxisSize: MainAxisSize.min,
            crossAxisAlignment: CrossAxisAlignment.spaceAround,
            children: <Widget>[
              Column(
                mainAxisAlignment: MainAxisAlignment.start,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: [
                  CircleAvatar(

```

```
        radius: 60.0,
        foregroundImage:
          ExactAssetImage('lib/images/download.jpg'),
      ),
      SizedBox(
        height: 20,
      ),
      Text(
        '$username',
        style: TextStyle(
          fontSize: 25,
          fontWeight: FontWeight.normal,
        ),
      ),
    ],
  ),
  SizedBox(
    height: 50,
  ),
  GestureDetector(
    onTap: () => Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => Userdetails(
          username: username,
          email: data,
        ),
      ),
    ),
    child: Text(
      'User Details',
      style: TextStyle(
        color: Colors.white,
        fontSize: 20,
        fontWeight: FontWeight.bold),
      textAlign: TextAlign.end,
    ),
  ),
  SizedBox(height: 20),
  GestureDetector(
    onTap: () => Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => ChangePassword(
```

```
username: username,email: widget.email,  
    ),  
  ),  
),  
child: Text(  
  'Change Password',  
  style: TextStyle(  
    color: Colors.white,  
    fontSize: 20,  
    fontWeight: FontWeight.bold),  
  textAlign: TextAlign.end,  
),  
),  
SizedBox(height: 20),  
GestureDetector(  
  onTap: () => Navigator.pushNamed(context, 'forgetemail'),  
  child: Text(  
    'Terms And Conditions',  
    style: TextStyle(  
      color: Colors.white,  
      fontSize: 20,  
      fontWeight: FontWeight.bold),  
    textAlign: TextAlign.end,  
  ),  
),  
SizedBox(height: 20),  
GestureDetector(  
  onTap: () => Navigator.pushNamed(context, 'forgetemail'),  
  child: Text(  
    'Contact Us',  
    style: TextStyle(  
      color: Colors.white,  
      fontSize: 20,  
      fontWeight: FontWeight.bold),  
    textAlign: TextAlign.end,  
  ),  
),  
SizedBox(height: 30),  
GestureDetector(  
  onTap: () => _logout(),  
  child: Text(  
    'Log Out',  
    style: TextStyle(  
      color: Colors.white,
```

```

        fontSize: 20,
        fontWeight: FontWeight.bold),
        textAlign: TextAlign.end,
      ),
    ),
  ],
),
),
),
),
);
}

Widget dashboard(context) {
  return AnimatedPositioned(
    duration: duration,
    top: 0,
    bottom: 0,
    left: isCollapsed ? 0 : -0.6 * screenWidth,
    right: isCollapsed ? 0 : 0.55 * screenWidth,
    child: ScaleTransition(
      scale: _scaleAnimation,
      child: Material(
        animationDuration: duration,
        borderRadius:
          isCollapsed ? null : BorderRadius.all(Radius.circular(40)),
        elevation: 8,
        child: SingleChildScrollView(
          scrollDirection: Axis.vertical,
          physics: ClampingScrollPhysics(),
          child: Container(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: <Widget>[
                Container(
                  height: MediaQuery.of(context).size.height,
                  width: MediaQuery.of(context).size.width,
                  child: Container(
                    child: Padding(
                      padding: const EdgeInsets.only(
                        top: 25.0, right: 10.0, left: 15, bottom: 15),
                      child: Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        crossAxisAlignment: CrossAxisAlignment.start,

```

```

children: [
  Row(
    children: [
      Text(
        'Can-Dect',
        style: TextStyle(
          fontSize: 35,
          fontWeight: FontWeight.normal,
        ),
      ),
      SizedBox(
        width: 180,
      ),
      InkWell(
        child: Icon(
          isCollapsed
            ? Icons.settings
            : Icons.arrow_forward_ios_outlined,
          color: Colors.black),
        onTap: () {
          setState(() {
            if (isCollapsed)
              _controller.forward();
            else
              _controller.reverse();

            isCollapsed = !isCollapsed;
          });
        },
      ),
    ],
  ),
  SizedBox(
    height: 25,
  ),
  Text(
    'Welcome $username,',
    style: TextStyle(
      fontSize: 25,
      fontWeight: FontWeight.normal,
    ),
  ),
  SizedBox(
    height: 25,
  ),

```

```

),
Padding(
  padding: const EdgeInsets.all(10.0),
  child: Column(
    children: [
      Row(
        children: [
          Container(
            height: 150,
            width: 150,
            foregroundDecoration: BoxDecoration(
              borderRadius:
                BorderRadius.circular(35.0),
              border: Border.all(
                color: Colors.white, width: 3.0),
            ),
          ),
          ElevatedButton(
            onPressed: () => {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) =>
                    Appointment(
                      userID: userss[0]['id'],
                    ),
                ),
              ),
            ),
          ),
        ],
      ),
      child: Padding(
        padding: const EdgeInsets.only(
          top: 10.0),
        child: Column(
          children: [
            Icon(
              FontAwesomeIcons.stethoscope,
              size: 45,
            ),
            SizedBox(
              height: 30,
            ),
            Text(
              'Appointment',
              style: TextStyle(
                fontSize: 20,

```





[illegible]

```

        top: 10.0),
      child: Column(
        children: [
          Icon(
            FontAwesomeIcons.pills,
            size: 45,
          ),
          SizedBox(
            height: 30,
          ),
          Text(
            'Prescribed Medicine',
            style: TextStyle(
              fontSize: 20,
              fontWeight:
                FontWeight.normal,
            ),
          ),
        ],
      ),
    ),
  ),
),
),
),
SizedBox(
  width: 40,
),
Container(
  height: 150,
  width: 150,
  foregroundDecoration: BoxDecoration(
    borderRadius:
      BorderRadius.circular(35.0),
    border: Border.all(
      color: Colors.white, width: 3.0),
  ),
  child: ElevatedButton(
    onPressed: () => {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) =>
            NotifyDetails(
              userID: usersss[0]['id'],
            ),

```

```
        ),
      ),
    },
    child: Padding(
      padding: const EdgeInsets.only(
        top: 10.0),
      child: Column(
        children: [
          Icon(
            FontAwesomeIcons.userMd,
            size: 45,
          ),
          SizedBox(
            height: 35,
          ),
          Text(
            'Notification',
            style: TextStyle(
              fontSize: 20,
              fontWeight:
                FontWeight.normal,
            ),
          ),
        ],
      ),
    ),
  ),
],
),
SizedBox(
  height: 25,
),
Row(
  children: [
    Container(
      height: 150,
      width: 150,
      foregroundDecoration: BoxDecoration(
        borderRadius:
          BorderRadius.circular(35.0),
        border: Border.all(
          color: Colors.white, width: 3.0),
      ),
    ),
```

```

child: ElevatedButton(
  onPressed: () => {
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) =>
          Bookingpage(
            userID: userss[0]['id'],
          ),
        ),
      ),
    ),
  },
  child: Padding(
    padding: const EdgeInsets.only(
      top: 10.0),
    child: Column(
      children: [
        Icon(
          LineIcons.alternateTicket,
          size: 45,
        ),
        SizedBox(
          height: 30,
        ),
        Text(
          'Bookings',
          style: TextStyle(
            fontSize: 20,
            fontWeight:
              FontWeight.normal,
          ),
        ),
      ],
    ),
  ),
),
SizedBox(
  width: 40,
),
Container(
  height: 150,
  width: 150,
  foregroundDecoration: BoxDecoration(

```

```
        borderRadius:
          BorderRadius.circular(35.0),
        border: Border.all(
          color: Colors.white, width: 3.0),
      ),
      child: ElevatedButton(
        onPressed: () => {
          Navigator.pushNamed(
            context, 'userdetails')
        },
        child: Padding(
          padding: const EdgeInsets.only(
            top: 10.0),
          child: Column(
            children: [
              Icon(
                FontAwesomeIcons
                  .handHoldingUsd,
                size: 45,
              ),
              SizedBox(
                height: 30,
              ),
              Text(
                'Payments',
                style: TextStyle(
                  fontSize: 20,
                  fontWeight:
                    FontWeight.normal,
                ),
              ),
            ],
          ),
        ),
      ),
    ],
  ),
],
),
),
```

```

    ),
  ),
],
),
),
),
),
),
),
);
}
}

```

Appointment Page:

```

// ignore_for_file: import_of_legacy_library_into_null_safe
import 'dart:convert';
import 'package:ecom/pages/payment.dart';
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:intl/intl.dart';
import 'package:http/http.dart' as http;
import 'package:text_tools/text_tools.dart';
import 'package:time_picker_widget/time_picker_widget.dart';

class Appointment extends StatefulWidget {
  final int userID;
  const Appointment({Key? key, required this.userID}) : super(key: key);

  @override
  _AppointmentState createState() => _AppointmentState();
}

class _AppointmentState extends State<Appointment> {
  String myValue = '';
  var show = '';
  var show2 = '';
  String price = '';
  late String tempValue;
  late String tempDoctor;
  bool purposeSelection = false;
  bool canSelect = false;
  bool doctorSelect = false;
  bool emptypurpose = false;

```

```

bool emptydoctor = false;
bool emptydate = false;
bool emptytime = false;
String _myActivity1 = '';
final formKey = new GlobalKey<FormState>();
TextEditingController dateinput = TextEditingController();
TextEditingController timeinput = TextEditingController();
bool sametime = false;
late final int data;
List servicedata = [];
List doctordata = [];
List<int> _availableHours = [9,10,11, 12,13,14,15,16,17,18];
List<int> _availableMinutes = [0, 10,20, 30,40, 50];
late String selectedTime;

@override
void initState() {
  selectedTime = '';
  tempDoctor = '';
  tempValue = '';
  this._selection();
  timeinput.text = '';
  dateinput.text = ''; //set the initial value of text field
  super.initState();
}

Future AppointmentDetail(int purpose, int doctor, String date, String time,
  String price, int user) async {
  final http.Response response = await http.post(
    Uri.parse('http://10.0.2.2:8000/appointapi/v1/appoint/'),
    headers: {"Content-Type": "application/json"},
    body: json.encode(
      <String, dynamic>{
        'purpose_id': purpose,
        'doctor_id': doctor,
        'date': date,
        'time': time,
        'price': price,
        'user_id': user,
      },
    ),
  );
  if (response.statusCode == 200) {
    var result = json.decode(response.body).toString();
  }
}

```



```

    if (result == "Successful") {
      sametime = false;
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => PaymentPage(),
        ),
      );
    } else {
      sametime = true;
    }
  } else {}
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Stack(
      children: [
        SingleChildScrollView(
          scrollDirection: Axis.vertical,
          child: Container(
            height: MediaQuery.of(context).size.height,
            width: MediaQuery.of(context).size.width,
            child: Center(
              child: Padding(
                padding: const EdgeInsets.only(
                  top: 80.0, right: 15.0, left: 15, bottom: 5),
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.start,
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Row(
                      children: [
                        InkWell(
                          child: const Icon(Icons.arrow_back_ios_new_outlined,
                            color: Colors.black),
                          onTap: () {
                            setState(() {
                              Navigator.pop(context);
                            });
                          },
                        ),
                        const SizedBox(

```

```

        width: 20,
      ),
      const Text(
        'Can-Dect',
        style: TextStyle(
          fontSize: 35,
          fontWeight: FontWeight.normal,
        ),
      ),
    ],
  ),
  SizedBox(
    height: 20,
  ),
  Text(
    'Book Appointment, ',
    style: TextStyle(
      fontSize: 25,
      fontWeight: FontWeight.normal,
    ),
  ),
  SizedBox(
    height: 20,
  ),
  Container(
    child: Padding(
      padding: const EdgeInsets.all(8.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
          Column(
            children: [
              Padding(
                padding:
                  const EdgeInsets.only(right: 280.0),
                child: Text(
                  'Purpose',
                  style: TextStyle(
                    fontSize: 16,
                    fontWeight: FontWeight.normal,
                  ),
                ),
              ),
            ],
          ),
          SizedBox(

```

```

width: MediaQuery.of(context).size.width,
child: DropdownButtonHideUnderline(
  child: ButtonTheme(
    alignedDropdown: true,
    child: DropdownButton<String>(
      value: purposeSelection
        ? show
        : tempValue,
      style: TextStyle(
        color: Colors.black54,
        fontSize: 16,
      ),
      hint: Text('Please select a purpose'),
      onChanged: (newVal) {
        setState(() {
          myValue = newVal.toString();
          show = myValue;
          purposeSelection = true;
          emptypurpose = false;
          doctorSelect = false;
          canSelect = true;
          _doctordata(int.parse(myValue));
          for (var i = 0;
            i < servicedata.length;
            i++) {
            if (servicedata[i]['id'] ==
              int.parse(myValue)) {
              setState(() {
                price = (servicedata[i]
                  ['price'])
                  .toString();
              });
            }
          }
        });
      },
      items: servicedata.map((item) {
        return new DropdownMenuItem(
          child: new Text(item['services']),
          value: item['id'].toString(),
        );
      }).toList(),
    ),
  ),
),

```

```
),  
    ),  
    emptypurpose  
      ? SizedBox(  
        height: 10,  
        child: Text(  
          'Please select a purpose!',  
          style: TextStyle(  
            color: Colors.red,  
            fontSize: 16,  
          ),  
        ),  
      ),  
    )  
    : Text(''),  
  ],  
),  
SizedBox(  
  height: 20,  
),  
Column(  
  children: [  
    Padding(  
      padding:  
        const EdgeInsets.only(right: 290.0),  
      child: Text(  
        'Doctor',  
        style: TextStyle(  
          fontSize: 16,  
          fontWeight: FontWeight.normal,  
        ),  
      ),  
    ),  
  ],  
),  
SizedBox(  
  width: MediaQuery.of(context).size.width,  
  child: DropdownButtonHideUnderline(  
    child: ButtonTheme(  
      alignedDropdown: true,  
      child: DropdownButton<String>( value:  
        doctorSelect ? show2 : tempDoctor,  
        style: TextStyle(  
          color: Colors.black54,  
          fontSize: 16,  
        ),  
      ),
```

```

        hint: Text('Select Doctor'),
        onChanged: (newVal) {
          setState(() {
            emptydoctor = false;
            _myActivity1 = newVal.toString();
            show2 = _myActivity1;
            print(_myActivity1);
          });
        },
        items: doctordata.map((item) {
          return new DropdownMenuItem(
            child: new Text(
              TextTools
                .toUppercaseFirstLetter(
                  text: item[
                    'FirstName']) +
                " " +
              TextTools
                .toUppercaseFirstLetter(
                  text: item[
                    'LastName']),
            value: item['id'].toString(),
          );
        }).toList(),
      ),
    ),
  ),
),
emptydoctor
  ? SizedBox(
    height: 10,
    child: Text(
      'Please select a Doctor!',
      style: TextStyle(
        color: Colors.red,
        fontSize: 16,
      ),
    ),
  )
: Text(''),
],
),
SizedBox(

```

```

        height: 20,
      ),
      Row(
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          SizedBox(
            width: 150,
            height: 70,
            child: TextField(
              controller:
                dateinput, //editing controller of this
TextField
              decoration: InputDecoration(
                icon: Icon(Icons
field
                  .calendar_today), //icon of text

                labelText:
                  "Enter Date", //label text of field
                errorText: emptydate
                  ? "Please select a date!"
                  : null,
              ),
            ),
            readOnly:
              true, //set it true, so that user will
not able to edit text

            onTap: () async {
              DateTime? pickedDate =
                await showDatePicker(
                  context: context,
                  initialDate: DateTime.now(),
                  firstDate: DateTime.now()
                    lastDate: DateTime(2101));
              emptydate = false;
              if (pickedDate != null) {

                String formattedDate =
                  DateFormat('yyyy-MM-dd')
                    .format(pickedDate);

                setState(() {
                  dateinput.text = formattedDate;

                  print(dateinput.text);

```

```
});  
    }  
  },  
),  
),  
SizedBox(  
  width: 10,  
),  
SizedBox(  
  height: 70,  
  width: 150,  
  child: TextField(  
    controller:  
      timeinput, //editing controller of this  
  
TextFiled  
    decoration: InputDecoration(  
      icon: Icon(  
        Icons.timer), //icon of text field  
      labelText:  
        "Enter Time", //label text of field  
      errorText: emptytime  
        ? 'Please select a time!'  
        : sametime  
          ? 'Time not available!'  
          : null,  
    ),  
    readOnly:  
      true, //set it true, so that user will  
not able to edit text  
    onTap: ()  
=>{  
      showCustomTimePicker(  
        context: context,  
onFailValidation: (context) =>  
print('Unavailable selection'),  
initialTime: TimeOfDay(  
  hour: _availableHours.first,  
  minute: _availableMinutes.first),  
selectableTimePredicate: (time) =>  
  _availableHours.indexOf(time!.hour)  
!= -1 &&  
    _availableMinutes.indexOf(time.minu  
te) != -1).then((time) =>  
      setState(() {
```

```

time!.format(context);
                                timeinput.text =
                                emptytime = false;
                                },
                                ),
                                ),
                                },
                                ),
                                )
                                ],
                                ),
                                SizedBox(
                                  height: 20,
                                ),
                                Column(
                                  crossAxisAlignment: CrossAxisAlignment.start,
                                  children: [
                                    Text(
                                      'Appointment Fees',
                                      style: TextStyle(
                                        color: Colors.black,
                                        fontSize: 16,
                                        fontWeight: FontWeight.normal,
                                      ),
                                    ),
                                    SizedBox(
                                      height: 10,
                                    ),
                                    TextField(
                                      obscureText: false,
                                      enabled: false,
                                      decoration: InputDecoration(
                                        border: OutlineInputBorder(
                                          borderSide: new BorderSide(
                                            color: Colors.blueAccent),
                                        ),
                                        labelText: 'Rs $price',
                                        labelStyle: TextStyle(
                                          color: Colors.black,
                                          fontWeight: FontWeight.normal,
                                          fontSize: 18,
                                        ),
                                      ),
                                    ),
                                  ],
                                ),
                                ],

```



```

),
  SizedBox(
    height: 20,
  ),
  Padding(
    padding: const EdgeInsets.only(
      top: 20.0,
      left: 210,
    ),
    child: ElevatedButton(
      onPressed: () {
        if (myValue.isEmpty) {
          emptypurpose = true;
        } else if (_myActivity1.isEmpty) {
          emptydoctor = true;
        } else if (dateinput.text.isEmpty) {
          emptydate = true;
        } else if (timeinput.text.isEmpty) {
          emptytime = true;
        } else {
          setState(() {
            int purpose = int.parse(myValue);
            int doctor = int.parse(_myActivity1);
            String date = dateinput.text;
            String time = timeinput.text;
            AppointmentDetail(purpose, doctor, date,
              time, price, widget.userID);
          });
        }
      },
      style: ElevatedButton.styleFrom(
        primary: Color.fromRGBO(3, 218, 197, 1),
        fixedSize: Size(130, 60),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(50),
        ),
      ),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.center,
        mainAxisSize: MainAxisSize.min,
        children: [
          Text(
            "Proceed",
            style: TextStyle(

```

```

        fontSize: 18,
        color: Colors.black,
        fontWeight: FontWeight.normal),
    ),
    SizedBox(
      width: 5,
    ),
    Icon(
      FontAwesomeIcons.angleRight,
      color: Colors.black,
    ),
  ],
),
),
),
],
),
),
),
],
),
),
),
],
),
),
),
],
),
);
}

_selection() async {
  final http.Response response = await http.get(
    Uri.parse('http://10.0.2.2:8000/service/v1/viewservice/?format=json'),
    headers: {"Content-Type": "application/json"},
  );
  if (response.statusCode == 200) {
    var data = json.decode(response.body);
    setState(() {
      servicedata = data;
      tempValue = (data[0]['id']).toString();
      price = (data[0]['price']).toString();
    });
  }
}
```

```

_doctordata(int serviceid) async {
  final http.Response response = await http.get(
    Uri.parse(
      'http://10.0.2.2:8000/doctorapi/v1/doctorappointdetail/$serviceid/?format=json'),
    headers: {"Content-Type": "application/json"},
  );
  if (response.statusCode == 200) {
    var data1 = json.decode(response.body);
    setState(() {
      doctordata = data1;
      tempDoctor = (doctordata[0]['id']).toString();
    });
  }
}

```

### Report Page:

```

import 'dart:convert';

import 'package:ecom/api/reportdata.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class Reportpage extends StatefulWidget {
  final int userID;
  const Reportpage({Key? key, required this.userID}) : super(key: key);

  @override
  _ReportpageState createState() => _ReportpageState();
}

class _ReportpageState extends State<Reportpage> {
  List<Reportdata> report = [];

  void initState() {
    _reportData(widget.userID);
    super.initState();
  }

  _reportData(int userID) async {
    report = [];

```

```

final http.Response response = await http.get(
  Uri.parse(
    'http://10.0.2.2:8000/reportapi/v1/viewReport/$userID/?format=json'),
  headers: {"Content-Type": "application/json"},
);
if (response.statusCode == 200) {
  List data1 = json.decode(response.body);
  data1.forEach((element) {
    report.add(Reportdata.fromMap(element));
  });
  setState(() {});
}
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Stack(children: [
      SingleChildScrollView(
        scrollDirection: Axis.vertical,
        child: Container(
          height: MediaQuery.of(context).size.height,
          width: MediaQuery.of(context).size.width,
          child: Padding(
            padding: const EdgeInsets.all(15.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Padding(
                  padding: const EdgeInsets.only(top: 60.0),
                  child:
                    Row(
                      children: [
                        InkWell(
                          child: const Icon(Icons.arrow_back_ios_new_outlined,
                            color: Colors.black),
                          onTap: () {
                            setState(() {
                              Navigator.pop(context);
                            });
                          },
                        ),
                        const SizedBox(

```

```
width: 20,  
),  
const Text(  
  'Report',  
  style: TextStyle(  
    fontSize: 25,  
    fontWeight: FontWeight.normal,  
  ),  
),  
],  
),  
),  
RefreshIndicator(  
  onRefresh: () async {  
    _reportData(widget.userID);  
  },  
child: Container(  
  height: 650,  
  child: report.length > 0  
    ? ListView.separated(  
      itemCount: report.length,  
      itemBuilder: (BuildContext context, int index) {  
        return Container(  
          decoration: BoxDecoration(  
            borderRadius: BorderRadius.circular(15),  
            border: Border.all(  
              color: Colors.black,  
              width: 3,  
            ),  
          ),  
          child: ListTile(  
            title: Row(  
              crossAxisAlignment:  
                CrossAxisAlignment.start,  
              mainAxisAlignment:  
                MainAxisAlignment.spaceBetween,  
              children: [  
                Text(  
                  'Diseases: ' + report[index].Disease,  
                  style: TextStyle(  
                    color: Colors.black,  
                    fontSize: 17,  
                  ),  
                ),  
              ],  
            ),  
          ),  
        );  
      },  
    ),  
  ),  
),
```

```

        SizedBox(
          width: 10,
        ),
        Text(
          report[index].Date,
          style: TextStyle(
            color: Colors.black,
            fontSize: 15,
          ),
        ),
      ],
    ),
    subtitle: Padding(
      padding: const EdgeInsets.only(top: 15.0),
      child: Column(
        crossAxisAlignment:
          CrossAxisAlignment.start,
        children: [
          Text(
            'Status: ' + report[index].Status,
            style: TextStyle(
              color: Colors.black,
              fontSize: 22,
            ),
          ),
          Text(
            'Additional Details: ',
            style: TextStyle(
              color: Colors.black,
              fontSize: 22,
            ),
          ),
          Padding(
            padding: const
EdgeInsets.only(left:30.0),

            child: Text(
              report[index].AdditionalDetails,
              style: TextStyle(
                color: Colors.black,
                fontSize: 18,
              ),
            ),
          ),
          Text(

```

```

        'Report: ' + report[index].ReportPDF,
        style: TextStyle(
          color: Colors.black,
          fontSize: 22,
        ),
      ),
    ],
  ),
),
);
},
separatorBuilder:
  (BuildContext context, int index) =>
    const Divider(),
)
: const Center(child: Text('No items')),
),
),
],
),
),
),
),
),
],
));
}
}

```

## Medicine Page:

```
import 'dart:convert';

import 'package:ecom/api/medicinedata.dart';
import 'package:ecom/pages/Homepage.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class MedicineDetails extends StatefulWidget {
  final int userID;
  const MedicineDetails({Key? key, required this.userID}) : super(key: key);

  @override
  _MedicineDetailsState createState() => _MedicineDetailsState();
}
```

```

class _MedicineDetailsState extends State<MedicineDetails> {
  List<Medicines> medicine = [];

  void initState() {
    _medicineData(widget.userID);
    super.initState();
  }

  _medicineData(int userID) async {
    medicine = [];
    final http.Response response = await http.get(
      Uri.parse(
        'http://10.0.2.2:8000/medicineapi/v1/viewMedicine/$userID/?format=json'
      ),
      headers: {"Content-Type": "application/json"},
    );
    if (response.statusCode == 200) {
      List data1 = json.decode(response.body);
      data1.forEach((element) {
        medicine.add(Medicines.fromMap(element));
        print(element['id']);
      });
      setState(() {});
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(children: [
        SingleChildScrollView(
          scrollDirection: Axis.vertical,
          child: Container(
            height: MediaQuery.of(context).size.height,
            width: MediaQuery.of(context).size.width,
            child: Padding(
              padding: const EdgeInsets.all(15.0),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Padding(
                    padding: const EdgeInsets.only(top: 60.0),

```



```

        child: Row(
          children: [
            InkWell(
              child: const Icon(Icons.arrow_back_ios_new_outlined,
                color: Colors.black),
              onTap: () {
                setState(() {
                  Navigator.pop(context);
                });
              },
            ),
            const SizedBox(
              width: 20,
            ),
            const Text(
              'Prescribed Medicine',
              style: TextStyle(
                fontSize: 25,
                fontWeight: FontWeight.normal,
              ),
            ),
          ],
        ),
      ),
      RefreshIndicator(
        onRefresh: () async {
          _medicineData(widget.userID);
        },
        child: Container(
          height: 650,
          child: medicine.length > 0
            ? ListView.separated(
                itemCount: medicine.length,
                itemBuilder: (BuildContext context, int index) {
                  return Container(
                    decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(15),
                      border: Border.all(
                        color: Colors.black,
                        width: 3,
                      ),
                    ),
                    child: ListTile(
                      title: Row(

```

```

        crossAxisAlignment:
            CrossAxisAlignment.start,
        mainAxisAlignment:
            MainAxisAlignment.spaceBetween,
        children: [
            Text(
                'Diseases: ' + medicine[index].Disease,
                style: TextStyle(
                    color: Colors.black,
                    fontSize: 17,
                ),
            ),
            SizedBox(
                width: 10,
            ),
            Text(
                medicine[index].Date,
                style: TextStyle(
                    color: Colors.black,
                    fontSize: 15,
                ),
            ),
        ],
    ),
    subtitle: Padding(
        padding: const EdgeInsets.only(top: 15.0),
        child: Text(
            medicine[index].Medicine,
            style: TextStyle(
                color: Colors.black,
                fontSize: 22,
            ),
        ),
    ),
);
},
separatorBuilder:
    (BuildContext context, int index) =>
        const Divider(),
)
: const Center(child: Text('No items')),
),
),

```

```

        ],
      ),
    ),
  ),
),
])));
}
}

```

### Notification Page:

```

import 'dart:convert';

import 'package:ecom/api/notifydetails.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:text_tools/text_tools.dart';

class NotifyDetails extends StatefulWidget {
  final int userID;
  const NotifyDetails({Key? key, required this.userID}) : super(key: key);

  @override
  _NotifyDetailsState createState() => _NotifyDetailsState();
}

class _NotifyDetailsState extends State<NotifyDetails> {
  List<NotifyDetail> notify = [];

  void initState() {
    _NotifyData(widget.userID);
    super.initState();
  }

  _NotifyData(int userID) async {
    notify = [];
    final http.Response response = await http.get(
      Uri.parse(
        'http://10.0.2.2:8000/notifyapi/v1/viewNotify/$userID/?format=json'),
      headers: {"Content-Type": "application/json"},
    );
    if (response.statusCode == 200) {
      List data1 = json.decode(response.body);
    }
  }
}

```

```

        data1.forEach((element) {
            notify.add(NotifyDetail.fromMap(element));
            print(notify);
        });
        setState(() {});
    }
}

@override
Widget build(BuildContext context) {
    return Scaffold(
        body: Stack(children: [
            SingleChildScrollView(
                scrollDirection: Axis.vertical,
                child: Container(
                    height: MediaQuery.of(context).size.height,
                    width: MediaQuery.of(context).size.width,
                    child: Padding(
                        padding: const EdgeInsets.all(15.0),
                        child: Column(
                            mainAxisAlignment: MainAxisAlignment.center,
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                                Padding(
                                    padding: const EdgeInsets.only(top: 60.0),
                                    child: Row(
                                        children: [
                                            InkWell(
                                                child: const Icon(Icons.arrow_back_ios_new_outlined,
                                                    color: Colors.black),
                                                onTap: () {
                                                    setState(() {
                                                        Navigator.pop(context);
                                                    });
                                                },
                                            ),
                                            const SizedBox(
                                                width: 20,
                                            ),
                                            const Text(
                                                'Notifications',
                                                style: TextStyle(
                                                    fontSize: 25,
                                                    fontWeight: FontWeight.normal,

```

```

    ),
  ),
],
),
),
RefreshIndicator(
  onRefresh: () async {
    _NotifyData(widget.userID);
  },
  child: Container(
    height: 650,
    child: notify.length > 0
      ? ListView.separated(
        itemCount: notify.length,
        itemBuilder: (BuildContext context, int index) {
          return Container(
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(15),
              border: Border.all(
                color: Colors.black,
                width: 3,
              ),
            ),
          ),
        child: ListTile(
          title: Row(
            crossAxisAlignment: CrossAxisAlignment.start,
            mainAxisAlignment:
              MainAxisAlignment.end,
            children: [
              Text(
                notify[index].Date,
                style: TextStyle(
                  color: Colors.black,
                  fontSize: 15,
                ),
              ),
            ],
          ),
          subtitle: Padding(
            padding: const EdgeInsets.only(top: 10.0),
            child: Column(
              crossAxisAlignment:
                CrossAxisAlignment.start,
              children: [

```

```

        Text(
          notify[index].Message,
          style: TextStyle(
            color: Colors.black,
            fontSize: 24,
          ),
        ),
        SizedBox(height: 5,),
        Column(
          crossAxisAlignment:
CrossAxisAlignment.start,
          mainAxisAlignment:
MainAxisAlignment.end,
          children: [
            Text(
              'From: Dr.' +
                TextTools
                  .toUppercaseFirstLetter(
                    text: notify[index]
                      .Doctor_Name
                      .FirstName) +
                " " +
                TextTools
                  .toUppercaseFirstLetter(
                    text: notify[index]
                      .Doctor_Name
                      .LastName),
              style: TextStyle(
                color: Colors.black,
                fontSize: 16,
              ),
            ),
          ],
        ),
      ],
    ),
  ),
);
},
separatorBuilder:
  (BuildContext context, int index) =>
    const Divider(),
)

```

```

        : const Center(child: Text('No items')),
      ),
    ),
  ],
),
),
),
),
),
]),
]);
}
}

```

### Booking Detail Page:

```

// ignore_for_file: import_of_legacy_library_into_null_safe

import 'dart:convert';
import 'package:ecom/api/bookingDetail.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'package:intl/intl.dart';
import 'package:text_tools/text_tools.dart';

class Bookingpage extends StatefulWidget {
  final int userID;
  const Bookingpage({Key? key, required this.userID}) : super(key: key);

  @override
  _BookingpageState createState() => _BookingpageState();
}

class _BookingpageState extends State<Bookingpage> {
  List<BookingDetails> booking = [];
  late String purposename;
  late String doctorname;
  bool canCancel = false;

  void initState() {
    _bookingsData(widget.userID);
    super.initState();
  }

  _bookingsData(int userID) async {
    booking = [];

```

```

final http.Response response = await http.get(
  Uri.parse(
    'http://10.0.2.2:8000/appointapi/v1/viewAppoint/$userID/?format=json'),
  headers: {"Content-Type": "application/json"},
);
if (response.statusCode == 200) {
  List data1 = json.decode(response.body);
  data1.forEach(
    (element) {
      booking.add(BookingDetails.fromMap(element));
    },
  );
  setState(() {});
}
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Stack(
      children: [
        SingleChildScrollView(
          scrollDirection: Axis.vertical,
          child: Container(
            height: MediaQuery.of(context).size.height,
            width: MediaQuery.of(context).size.width,
            child: Padding(
              padding: const EdgeInsets.all(15.0),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Padding(
                    padding: const EdgeInsets.only(top: 60.0),
                    child: Row(
                      children: [
                        InkWell(
                          child: const Icon(Icons.arrow_back_ios_new_outlined,
                            color: Colors.black),
                          onTap: () {
                            setState(() {
                              Navigator.pop(context);

```



```

        });
      },
    ),
    const SizedBox(
      width: 20,
    ),
    const Text(
      'Booking',
      style: TextStyle(
        fontSize: 25,
        fontWeight: FontWeight.normal,
      ),
    ),
  ],
),
),
RefreshIndicator(
  onRefresh: () async {
    _bookingsData(widget.userID);
  },
  child: Container(
    height: 650,
    child: booking.length > 0
      ? ListView.separated(
          itemCount: booking.length,
          itemBuilder: (BuildContext context, int index) {
            return Container(
              decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(15),
                border: Border.all(
                  color: Colors.black,
                  width: 3,
                ),
            ),
            child: ListTile(
              title: Row(
                crossAxisAlignment:
                  CrossAxisAlignment.start,
                mainAxisAlignment:
                  MainAxisAlignment.spaceBetween,
                children: [
                  Text(
                    'Booking Ticket',
                    style: TextStyle(

```

```

        color: Colors.black,
        fontSize: 17,
      ),
    ),
    SizedBox(
      width: 10,
    ),
    Column(
      children: [
        Text(
          booking[index].date.toString(),
          style: TextStyle(
            color: Colors.black,
            fontSize: 15,
          ),
        ),
        Text(
          booking[index].times,
          style: TextStyle(
            color: Colors.black,
            fontSize: 15,
          ),
        ),
      ],
    ),
  ],
),
subtitle: Padding(
  padding:
    const EdgeInsets.only(top: 15.0),
  child: Column(
    crossAxisAlignment:
      CrossAxisAlignment.start,
    children: [
      Text(
        'Purpose: ' +
          booking[index]
            .purpose
            .services,
        style: TextStyle(
          color: Colors.black,
          fontSize: 22,
        ),
      ),
    ],
  ),
),

```

```

Text(
  'Doctor: ' +
    TextTools
      .toUppercaseFirstLetter(
        text: booking[index]
          .Doctor
          .FirstName) +
    " " +
    TextTools
      .toUppercaseFirstLetter(
        text: booking[index]
          .Doctor
          .LastName),
  style: TextStyle(
    color: Colors.black,
    fontSize: 22,
  ),
),
Text(
  'Price: ' +
    (booking[index].price)
      .toString(),
  style: TextStyle(
    color: Colors.black,
    fontSize: 22,
  ),
),
// FutureBuilder<Object>(
//   future: checkCancel(index),
//   builder: (context, snapshot) {
//     return
Padding(
  padding: const EdgeInsets.only(
    left: 160.0, bottom: 5),
  child: canCancel
    ? ElevatedButton(
      onPressed: () {},
      style: ElevatedButton
        .styleFrom(
          primary: Color.fromRGBO(
            3, 218, 197, 1),
          fixedSize:
            Size(158, 60),
          shape:

```

```

RoundedRectangleBorder
r(
    borderRadius:
        BorderRadius
            .circular(50),
    ),
    ),
    child: Text(
        "Cancel Booking",
        style: TextStyle(
            fontSize: 18,
            color: Colors.black,
            fontWeight:
                FontWeight
                    .normal),
    ),
    )
    : null,
    ),
    // },
    // ),
    ],
    ),
    ),
    ),
    ),
    );
    },
    separatorBuilder:
        (BuildContext context, int index) =>
            const Divider(),
    )
    : const Center(child: Text('No items')),
    ),
    ),
    ],
    ),
    ),
    ),
    ),
    ),
    ],
    ),
    );
}
}

```

## Change Password Page:

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:http/http.dart' as http;

class ChangePassword extends StatefulWidget {
  String username;
  String email;
  ChangePassword({
    Key? key,
    required this.username,
    required this.email,
  }) : super(key: key);

  @override
  _ChangePasswordState createState() => _ChangePasswordState();
}

class _ChangePasswordState extends State<ChangePassword> {
  bool pwseen = false;
  bool pwseen1 = false;
  bool pwseen2 = false;
  final oldpw = TextEditingController();
  final newpw = TextEditingController();
  final repw = TextEditingController();
  bool incorrect = false;
  bool issame = false;

  updatepw(String email, String oldpw, String newpw) async {
    final http.Response response = await http.post(
      Uri.parse(
        'http://10.0.2.2:8000/userapi/v1/updateUserpw/$email/?format=json'),
      headers: {"Content-Type": "application/json"},
      body: json.encode(
        <String, dynamic>{
          'old': oldpw,
          'new': newpw,
        },
      ),
    );
  }
```

```
if (response.statusCode == 200) {
  var result = json.decode(response.body).toString();
  if (result == "Password Updated") {
    Navigator.pop(context);
    setState(() {
      incorrect = false;
    });
  } else {
    setState(() {
      incorrect = true;
    });
  }
} else {}
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Stack(
      children: [
        SingleChildScrollView(
          scrollDirection: Axis.vertical,
          child: Container(
            height: MediaQuery.of(context).size.height,
            width: MediaQuery.of(context).size.width,
            child: Padding(
              padding: const EdgeInsets.all(14.0),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Column(
                    children: [
                      Column(
                        children: [
                          Padding(
                            padding: const EdgeInsets.only(right: 350.0),
                            child: InkWell(
                              child: const Icon(
                                Icons.arrow_back_ios_new_outlined,
                                color: Colors.black,
                              ),
                              onTap: () {
                                setState(() {
                                  Navigator.pop(context);
                                });
                              },
                            ),
                          const Text(
                            "Go Back",
                            style: TextStyle(
                              color: Colors.black,
                              fontSize: 16,
                            ),
                          ),
                        ],
                      ),
                    ],
                  ),
                ],
              ),
            ),
          ),
        ),
      ],
    ),
  );
}
```

```

        });
      },
    ),
  ],
),
const SizedBox(
  width: 20,
),
Column(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    CircleAvatar(
      radius: 60.0,
      foregroundImage:
        ExactAssetImage('lib/images/download.jpg'),
    ),
    SizedBox(
      height: 20,
    ),
    Text(
      widget.username,
      style: TextStyle(
        fontSize: 25,
        fontWeight: FontWeight.normal,
      ),
    ),
  ],
),
],
),
SizedBox(
  height: 35,
),
Container(
  child: Padding(
    padding: const EdgeInsets.all(8.0),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.end,
      children: [
        TextField(
          controller: oldpw,
          obscureText: pwseen ? false : true,
          decoration: InputDecoration(

```

[illegible]



[illegible]

```

        : Text(''),
      SizedBox(
        height: 40,
      ),
      Padding(
        padding: const EdgeInsets.only(
          top: 10.0,
          left: 180,
        ),
        child: ElevatedButton(
          onPressed: () {
            final String newpass = newpw.text;
            final String oldpass = oldpw.text;
            final String repass = repw.text;
            if (newpass == repass) {
              setState(() {
                issame = false;
                updatepw(widget.email, oldpass, newpass);
              });
            } else {
              setState(() {
                issame = true;
              });
            }
          },
          style: ElevatedButton.styleFrom(
            primary: Color.fromRGBO(3, 218, 197, 1),
            fixedSize: Size(300, 60),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(50),
            ),
          ),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            mainAxisSize: MainAxisSize.min,
            children: [
              Text(
                "Confirm",
                style: TextStyle(
                  fontSize: 25,
                  color: Colors.black,
                  fontWeight: FontWeight.normal),
              ),
              SizedBox(

```

```

width: 5,
),
Icon(
  FontAwesomeIcons.checkCircle,
  color: Colors.black,
),
],
),
),
),
),
],
),
),
),
),
],
),
),
),
),
],
),
),
);
}
}

```

### User Data Details Page:

```

import 'dart:convert';

import 'package:ecom/api/detail.dart';
import 'package:ecom/api/signdata.dart';
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:http/http.dart' as http;

class Userdetails extends StatefulWidget {
  String username;
  int email;
  Userdetails({
    Key? key,
    required this.username,
    required this.email,
  }) : super(key: key);
}

```

```

@override
_UserdetailsState createState() => _UserdetailsState();
}

class _UserdetailsState extends State<Userdetails> {
  late String dataaddress;
  late int dataphn;
  late int dataage;
  bool emptyage = false;
  bool emptyaddress = false;
  bool emptyphone = false;
  TextEditingController phone = TextEditingController();
  TextEditingController address = TextEditingController();
  TextEditingController age = TextEditingController();
  List<userDetails> detailss = [];

  void initState() {
    dataaddress = ' ';
    dataphn = 0;
    dataage = 0;
    _users(widget.email);
    super.initState();
  }

  _users(int email) async {
    detailss = [];
    final http.Response response = await http.get(
      Uri.parse(
        'http://10.0.2.2:8000/userapi/v1/viewUserdetail/$email/?format=json'),
      headers: {"Content-Type": "application/json"},
    );
    if (response.statusCode == 200) {
      var data1 = json.decode(response.body);
      print(data1);
      setState(() {
        dataaddress = data1['address'];
        dataage = data1['age'];
        dataphn = data1['phone'];
      });
    } else {}
  }

  _userdetails(int email, int age, int phone, String address) async {
    final http.Response response = await http.post(

```

```

Uri.parse(
  'http://10.0.2.2:8000/userapi/v1/updateUserdetail/$email/?format=json')
',
headers: {"Content-Type": "application/json"},
body: json.encode(
  <String, dynamic>{
    'age': age,
    'phone': phone,
    'address': address,
  },
),
);
if (response.statusCode == 200) {
  var result = json.decode(response.body).toString();
  if (result == "Updated") {
    Navigator.pop(context);
  }
} else {}
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Stack(
      children: [
        SingleChildScrollView(
          scrollDirection: Axis.vertical,
          child: Container(
            height: MediaQuery.of(context).size.height,
            width: MediaQuery.of(context).size.width,
            child: Padding(
              padding: const EdgeInsets.all(14.0),
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Column(
                    children: [
                      Column(
                        children: [
                          Padding(
                            padding: const EdgeInsets.only(right: 350.0),
                            child: InkWell(
                              child: const Icon(
                                Icons.arrow_back_ios_new_outlined,

```

```

        color: Colors.black,
      ),
      onTap: () {
        setState(() {
          Navigator.pop(context);
        });
      },
    ),
  ),
],
),
PreferredSize(
  height: 10,
),
Column(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    CircleAvatar(
      radius: 60.0,
      foregroundImage:
        ExactAssetImage('lib/images/download.jpg'),
    ),
    SizedBox(
      height: 20,
    ),
    Text(
      widget.username,
      style: TextStyle(
        fontSize: 25,
        fontWeight: FontWeight.normal,
      ),
    ),
  ],
),
],
),
PreferredSize(
  height: 35,
),
Container(
  child: Padding(
    padding: const EdgeInsets.all(8.0),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.end,

```

```
children: [
  TextFormField(
    initialValue: dataphn.toString(),
    controller: phone,
    obscureText: false,
    keyboardType: TextInputType.number,
    style: TextStyle(color: Colors.blue),
    decoration: InputDecoration(
      border: OutlineInputBorder(
        borderSide:
          new BorderSide(color: Colors.blueAccent),
      ),
      labelText: 'Phone',
      hintText: 'Phone Number',
      errorText:
        emptyphone ? 'Please Fill the box' : '',
    ),
  ),
  SizedBox(
    height: 30,
  ),
  TextFormField(
    initialValue: dataaddress,
    controller: address,
    obscureText: false,
    style: TextStyle(color: Colors.blue),
    decoration: InputDecoration(
      border: OutlineInputBorder(
        borderSide:
          new BorderSide(color: Colors.blueAccent),
      ),
      labelText: 'Address',
      hintText: 'Address',
      errorText:
        emptyaddress ? 'Please Fill the box' : '',
    ),
  ),
  SizedBox(
    height: 30,
  ),
  TextFormField(
    initialValue: dataage.toString(),
    obscureText: false,
    keyboardType: TextInputType.number,
```

```
style: TextStyle(color: Colors.blue),  
decoration: InputDecoration(  
    border: OutlineInputBorder(  
        borderSide:  
            new BorderSide(color: Colors.blueAccent),  
    ),  
    labelText: 'Age',  
    hintText: 'Age',  
    errorText:  
        emptyage ? 'Please Fill the box' : '',  
),  
,  
,  
),  
SizeBox(  
    height: 40,  
,  
),  
Padding(  
    padding: const EdgeInsets.only(  
        top: 10.0,  
        left: 210,  
    ),  
child: ElevatedButton(  
    onPressed: () {  
        if (phone.text.isEmpty) {  
            setState(() {  
                emptyphone = true;  
            });  
        } else if (address.text.isEmpty) {  
            setState(() {  
                emptyaddress = true;  
            });  
        } else if (age.text.isEmpty) {  
            setState(() {  
                emptyage = true;  
            });  
        } else {  
            setState(() {  
                emptyaddress = false;  
                emptyage = false;  
                emptyphone = false;  
                final String address = address.text;  
                final int phn = int.parse(phone.text);  
                final int ag = int.parse(age.text);  
                _userdetails(  
                    widget.email, ag, phn, address);
```



```
});
    }
  },
  style: ElevatedButton.styleFrom(
    primary: Color.fromRGBO(3, 218, 197, 1),
    fixedSize: Size(140, 60),
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(50),
    ),
  ),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.center,
    mainAxisSize: MainAxisSize.min,
    children: [
      Text(
        "Save",
        style: TextStyle(
          fontSize: 25,
          color: Colors.black,
          fontWeight: FontWeight.normal),
      ),
      SizedBox(
        width: 10,
      ),
      Icon(
        FontAwesomeIcons.checkCircle,
        color: Colors.black,
      ),
    ],
  ),
),
),
),
),
),
),
),
),
),
),
),
);
```

```
}
}
```

### Forget Password Page:

```
// ignore_for_file: import_of_legacy_library_into_null_safe
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:flutter_pin_code_fields/flutter_pin_code_fields.dart';

class Forgetpassword extends StatefulWidget {
  const Forgetpassword({Key? key}) : super(key: key);

  @override
  _ForgetpasswordState createState() => _ForgetpasswordState();
}

class _ForgetpasswordState extends State<Forgetpassword> {
  final codec = TextEditingController();

  onAdd(){
    final int intcode = int.parse(codec.text);
    // widget.client
    // .post(Uri.parse('http://10.0.2.2:8000/userapi/v1/verify/'), body: {
    //   'code': intcode,
    // });
    Navigator.pushNamed(context, 'signupp');
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          SingleChildScrollView(
            scrollDirection: Axis.vertical,
            child: Container(
              height: MediaQuery.of(context).size.height,
              width: MediaQuery.of(context).size.width,
              child: Center(
                child: Padding(
                  padding: const EdgeInsets.only(
                    top: 100.0,
                    right: 15.0,
                    left: 15,

```

```
bottom: 5,  
),  
child: Column(  
  crossAxisAlignment: CrossAxisAlignment.start,  
  mainAxisAlignment: MainAxisAlignment.start,  
  children: [  
    Text(  
      "Can-Dect",  
      style: TextStyle(  
        fontSize: 60,  
        fontWeight: FontWeight.normal,  
      ),  
    ),  
    SizedBox(  
      height: 20,  
    ),  
    Text(  
      'Forgot Password',  
      style: TextStyle(fontSize: 35),  
    ),  
    SizedBox(  
      height: 25,  
    ),  
    Text(  
      'Please check Email for 4 digit Password Re-set Code!',  
      style: TextStyle(fontSize: 25),  
    ),  
    SizedBox(  
      height: 40,  
    ),  
    Container(  
      child: Padding(  
        padding: const EdgeInsets.all(25.0),  
        child: PinCodeFields(  
length: 4,  
animationDuration: const Duration(milliseconds: 200),  
animationCurve: Curves.easeInOut,  
switchInAnimationCurve: Curves.easeIn,  
switchOutAnimationCurve: Curves.easeOut,  
animation: Animations.SlideInDown,  
controller: codec,  
          ),  
        ),  
      ),
```

```

    SizedBox(
      height: 20,
    ),
    Column(
      children: [
        Padding(
          padding: const EdgeInsets.only(left: 100),
          child: Row(
            crossAxisAlignment: CrossAxisAlignment.end,
            children: [
              Icon(
                Icons.restart_alt_rounded,
                color: Colors.blueAccent,
                size: 30,
              ),
              GestureDetector(
                onTap: () => Navigator.pushNamed(context, ''),
                child: Text(
                  ' Resend Code!',
                  style: TextStyle(
                    color: Colors.blueAccent[700],
                    fontSize: 22),
                ),
              ),
              SizedBox(
                height: 25,
              ),
            ],
          ),
        ),
      ],
    ),
    SizedBox(
      height: 50,
    ),
    Padding(
      padding: const EdgeInsets.only(
        top: 10.0,
        left: 210,
      ),
      child: ElevatedButton(
        onPressed: () {
          Navigator.pushNamed(context, 'homepage');
        },
      ),
    ),
  ),
)

```

```

        style: ElevatedButton.styleFrom(
          primary: Color.fromRGBO(3, 218, 197, 1),
          fixedSize: Size(140, 60),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(50),
          ),
        ),
      ),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.center,
        mainAxisSize: MainAxisSize.min,
        children: [
          Icon(
            FontAwesomeIcons.userCog,
            color: Colors.black,
          ),
          SizedBox(
            width: 15,
          ),
          Text(
            "Reset",
            style: TextStyle(
              fontSize: 25,
              color: Colors.black,
              fontWeight: FontWeight.normal),
          ),
        ],
      ),
    ),
  ),
),
],
);
}

```