Tanta University

Faculty of Computers & Information

DOCLIN APP

Graduation Project

Student's names

1. إسراء رضا عبدالله محمد الجز
2. أيمن عبدالقادر محمد على جاب الله
3. عبدالرحمن محمد محمد عبدالسلام عياد
4. عبدالرحمن هانى مؤمن محمد الشاذلى
5. عدى حاتم مصطفى مصطفى عثمان
6. مروه محمد عبدالمنعم منصور السودانى
7. يوسف محمد عبدالفتاح عبدالعزيز عبدالمنعم

Under supervisor

Prof. Dr.

Omnia Elbarbary

July 2023

ACKNOWLEDGEMENT

This Project is our cordial effort and our supervisor's initiative and constant motivation. But First of all we would like to be grateful to the Allah, who gives us the effort to work on this project for the last two semester. Special thanks goes to our honorable Supervisor **Dr. Omnia Elbarbary** for this enormous support, inspiration and helpful criticism in this project. We are very grateful to her for giving us the opportunity to work with her.

We must thank and acknowledge our faculty. We also want to thank our beloved classmates and other students of the university.

Last but not least we thank our respectable parents for educating us with aspect from both arts and science for their unconditional support and encouragement that give us patience and ambitions to reach to this level.

ABSTRACT

We are going to create an easy, faster and smooth appointment application between doctor and patient. By using this application people can easily get to know about the timing of doctor's counselling period and make their meeting whenever they want.

Both doctors and patients can register themselves which is monitored by application. Doctors can sign up by giving necessary details like Name, Qualifications, Specializations, and Work History etc.

The doctors can login using their username and password and check for any appointment requests by patients. If the appointment is available, an appointment is sent to the patient about the same. They can also prescribe medicines after consultation and view feedback given by the patient.

The patient must also be a registered user and he can select the particular doctor who want to book an appointment with him in this application.

This application has the advantage of recording the diagnosis, the necessary medications, and the patient's private papers, such as analyzes and x-rays, through the doctor, and keeping them as a patient medical history.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

Preface

DOCLIN APP

Figure 1 logo App

Version 1

This documentation is oriented to customers,

Manager, Developer and Tester

Chapter 1: Introduction

1.1 project idea

In our daily life we face a lot of problems. Disease is one of most common issues for a person's life. If anybody is ill and wants to visit a doctor for checkup, he or she needs to visit the clinic and waits until the doctor is available. The patient also waits in a queue while getting appointment. If the doctor cancels the appointment for some emergency reasons then the patient is not able to know about the cancelation of the appointment unless or until he or she visits the clinics, it's necessary to get a consultation with Doctors whenever we got affected with various diseases.

Vision of this project is to create doctor patient handling management application that will help patients to book doctor appointment and fulfil their prospects. In this system doctors are allowed to manage their booking slots in online, they specify the time of section and sets working hours. Patients can make their appointment to book empty slots too,

This is the application of reservation for counselling by patients name. This application manages different kinds of doctors at a time.

1.2 scope of problem:

As we said, patients face some difficulties to appointment with doctors and other problems. So, we can recapitulate some problem here:

1-Difficulty scheduling appointments for doctor and patients.

2- Patients having to wait longer.

3-patients can forget the appointment

4- Loss of analyzes, x-rays and prescriptions for the patient

5-pantients or doctors can face problem in their account

1.3 proposed solution

According to these problem, our system offers solutions that will help patients and doctors

For patients:

1- Our application provide ease and comfort to patients while taking appointment from doctors and it also resolves the problems that the patients has to face while making an appointment.

2- Our application that is used to remind the patients of their appointment with doctor.

3- There is a medical history for each patient, with his tests and the doses he took.

4-There are a message between support team and them (doctor and patient) to solve their problems.

5- There is huge collection of doctor information.

For doctors:

1- No need any assistant for appointment.

2- Easily access to history of medication of a patient.

3- Can know acceptability by rating of patient.

4- Seeing patients' appointments has the freedom to show whether to accept or pending.

1.4 Objectives

1-Helping people to search for doctors and get appointment is our main objectives.

2- User can search doctors which can make sure to find specific doctor an easy task.

3-Doctors and patients can check patient previous medical history for better checkup.

4-Make the process of booking so easily.

5-Providing easy way for doctors to manage their appointments schedule.

1.5 Software development tools:

We have used some software tools and platform for development is describe below:

We used visual studio

For frontend development:

1-React native

2- React hooks

3- Redux toolkit

4-javascript

5-modern JavaScript ES 6

For backend development:

1-my Sql

2-php

For UI/UX design:

Adobe xd

For system analysis diagram:

Wondershare EdrawMax

Chapter2: Theoretical considerations

2.1 Requirement collection and analysis:

Requirement collection and analysis is very crucial part of any project. Without analysis, collecting data or a good planning, a project will never complete properly. When we are developing on a project, a delivery time of the project has already given.

Analysis and requirement collection was our big challenge, when we start thinking about this project. After start analysis we figure out some significant features that boost our project.

There are two types of requirements:

1-user requirement definition

2-system requirement specification

2.2 User requirement:

2.2.1 Functional requirement:

1-The system should enable patients and doctors to register.

2-The system should enable patients and doctors to log in.

3-The system should enable patients and doctors to log out.

4- The system should enable patients and doctors to show the medical history of patients.

5- The system should allow Patients and Doctors to view and modify their profile.

6- The system should allow Patients and doctors to make enquiries

7- The system should allow Doctors to set their available time.

8- The system should allow Doctors and patients to view appointments.

9-The system should allow Patients to search for available doctors.

10-The system should allow Patients to choose the time which doctor is available.

11- The system should allow Patients to book appointment.

12- The system allows the patient to make a medical history, whether private or public.

13- The system should enable support team to reply enquiries.

2.2.2 Non Functional requirement:

1-**performance:**

- The system should have a good response time
- The system must run without error while operating with a huge set of data.

2-**security:**

- All external communications between the system's data server and clients must be more secure.

3-**usability:**

- The system should have a well-formed graphical user interfaces
- The system should be user friendly.

4-**Reliability**

- The system should have a very low failure rate.
- The system should be available when requested for service by users.

2.3 System Requirements:

2.3.1 Functional requirement:

1- The system should enable patients and doctors to register.

- In case of patient collect user information (Name, phone, Email, Password, confirm password, gender, blood type, weight, height, age)
- In case of doctor collect information (Name, Email, password, confirm password, department, Experience, location, qualification, phone, work days, start time, end time, section time)
- Check information is valid:
  - Password is not empty
  - Password and confirm password is the same
  - Email hasn't been used before.
  - If the information is valid add user to database.

2- The system should enable patients and doctors to login.

- The user should enter their email and password.
- The information that given is valid.
- Access should be granted or denied

3- The system should enable patients and doctors to log out.

- User log out when user click on log out button.

4- The system should enable patients and doctors to show the history medical of patients.

- There is a medical history of patients through which they can know the patient's condition in addition to a set of analyzes and medicines for each patient.
- Patient can make his medical history private or public
- The doctor and patient can show this result.

5- The system should allow Patients and Doctors to view and modify their profile.

- They enter new information(Change password, change profile photo ,change user name)
- Replace new information with old information in database.

6- The system should allow Patients and doctors to make enquiries

- They enter their enquiries to contact us
- They should write a comment that the issue that face them.
- Support team will contact with them to solve it

7- The system should allow Doctors to set their available time.

- The doctor enter the date and time he will be available.

8-The system should allow Patients to search for available doctors.

- From their department the user can show the doctors that are available.
- They can search by name, Specialization and rating.

9- The system should allow Patients to book appointment.

- The patient shall select the department of doctor
- The system shall display the available time of the particular doctor to be booked.
- There are more than one payment method for booking such as Visa Card or cash.
- The system shall send a confirmation when appointment is made.

10- The system should allow Patients to choose the time which doctor is available.

- When doctor set the time which he available
- The patient can choose time and book appointment.

11- The system should enable administrator to reply enquiries

- The administrator should be able to read the enquiries.
- After the administrator writes the reply it should be sent to user.

2.3.2 Non Functional requirement:

1-**performance:**

- The system must have a good response time.
- The load time for the user interface should take less than 30 seconds.
- It should be able to respond to multiple numbers of people at the same time.
- The log in information should be verified within 30 seconds.

2- security:

- All external communications between the system's data server and clients must be more secure:
    - The access permissions for system data may only be changed by the system's data administrator: The system's administrator should be the only one with the authority to enable access to the system data

3-usability:

- The system should have a well-formed graphical user interfaces:
  - The system should have a well- structured easy to understand manual to guide its user.

- The system should be user friendly:
  - The user must understand what the system does.
  - The user must feel satisfied with the system

4- Reliability:

- The system should have a very low failure rate:
  - The failure rate should be kept as minimal as possible.

2.4 Flow chart:

2.4.1Flow chart of all patients:

As we can see, how all application work from first point (Register) to all processes in application, and how patients can pass from every point or step to other by providing specific condition to each step.

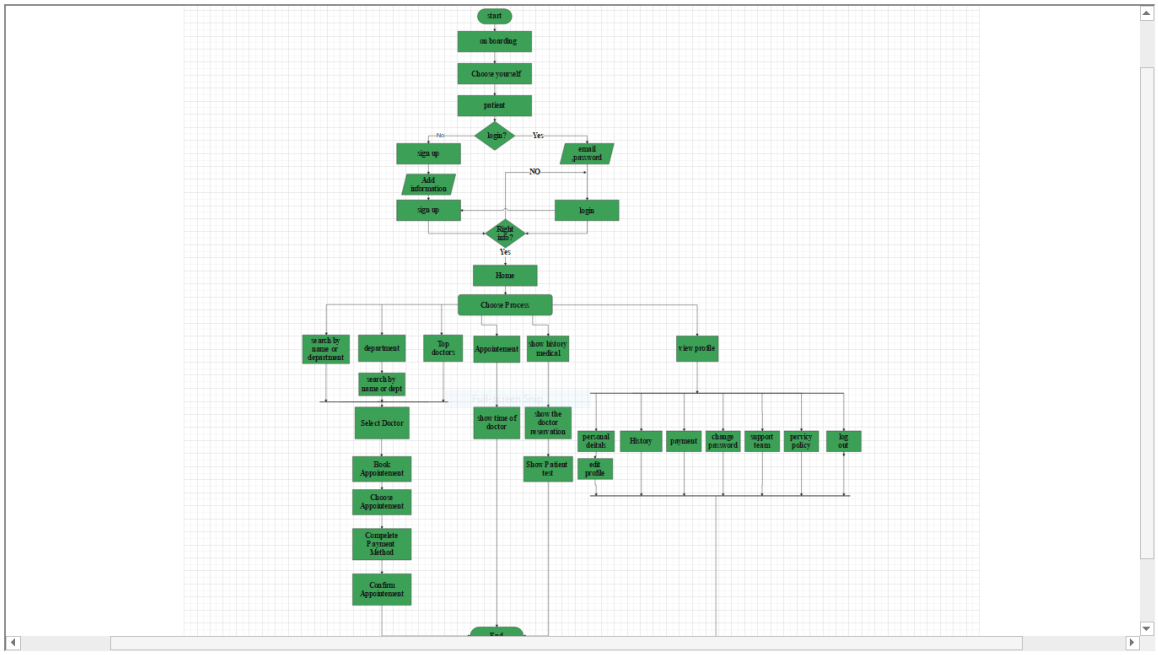


Figure 2 flow chart of patient

2.4.2Flow chart of all doctors:

This describe how doctors flow from the first point to end to make each process in the application.

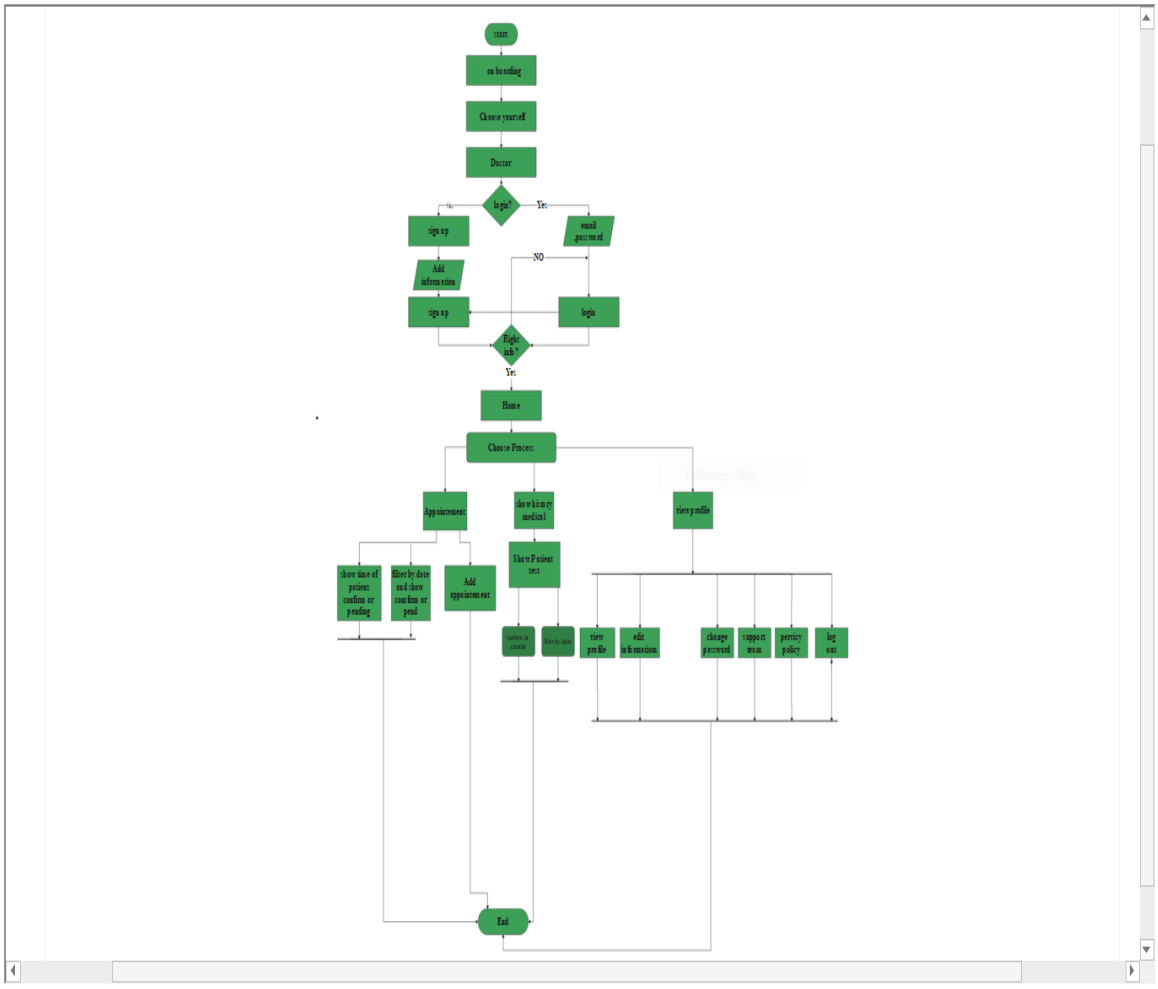


Figure 3 flow doctor

2.5 Use case diagram:

A use case diagram at its simplest is a representation of user's interaction with the system that shows the relationship between the users.

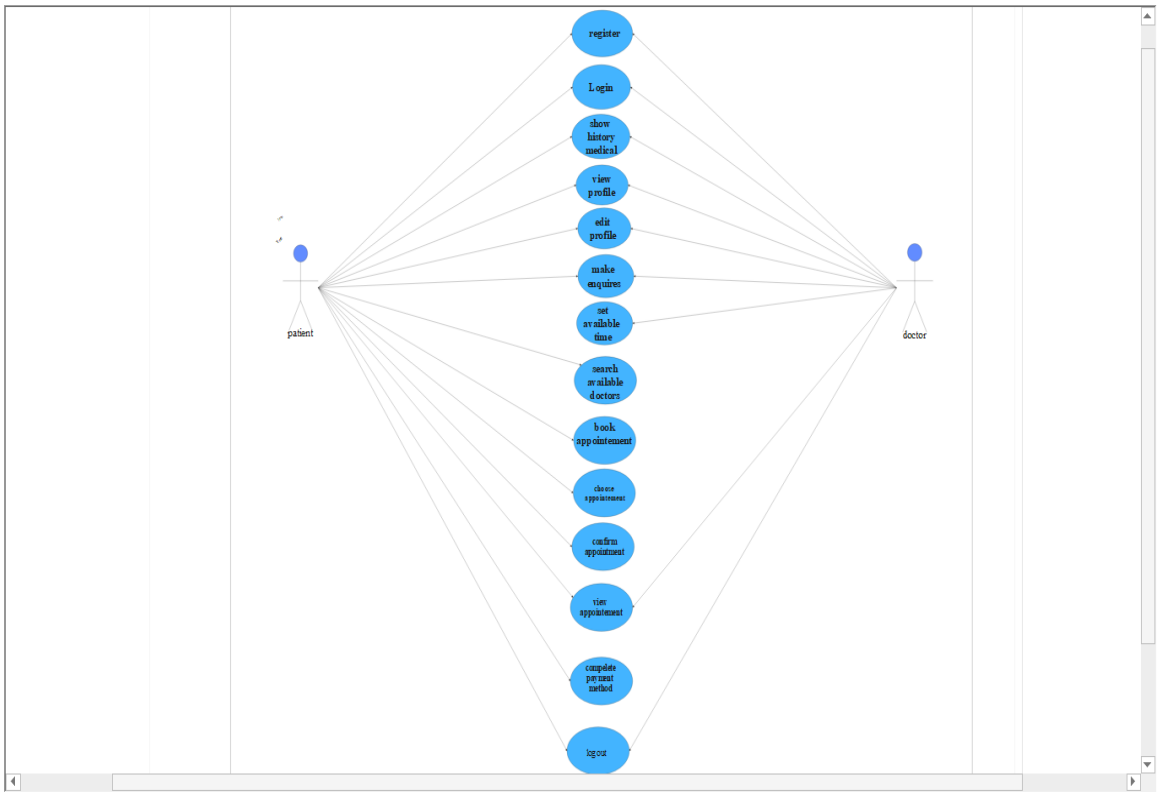


Figure 4 use case diagram

2.6 Use Case Tables:

2.6.1 Register use case table:

Use case name:	Register
Number:	1
Actors:	Doctor, Patient
Actor's Goal:	To create an account that can book appointment for patient and for doctor to able manage appointment.
Preconditions:	The system displays the menu for doctor or patient to enter personal information (New email, New name, phone ,password, confirm password,gender,age)
Flow of events:	1-Doctor/patient open application and choose who doctor or patient 2- Application shows user a sign up page to get some basic information about user such as name,email,password confirm password, gender 3-patient or doctor enter information and press submit. 4-system display a message that registration is successful.
Post conditions:	System redirect user to log in
Exception:	1-User enter invalid information such as user name or password. 2- System shows a warning message that username or password entered was wrong and asked for entering information again.

Table 1 Register use case Table

2.6.2 Login use case table

Use case name:	Login
Number:	2
Actors:	Doctors, patients.
Actor's Goal:	To log into account.
Preconditions:	They register before.
Flow of events:	1- Doctor or patient opens application and choose Doctor or patient 2- Application shows user a login page to enter email and password. 3- User enters the information and click on login button. 4- Application displays "login successful"
Post conditions:	The application should redirect user to homepage
Exception:	1- User enter invalid username or password. 2- System shows a warning message that username or password entered was wrong and asked for entering information again

Table 2 Login use case table

2.6.3 Show medical history

Use case name:	Show medical history
Number:	3
Actors:	Doctors, Patients
Actor’s Goal:	Recording the diagnosis, the necessary medications, and the patient's private papers, such as analyzes and x-rays, through the doctor, and keeping them as a patient history.
Preconditions:	1-the patient should go to doctor previously 2-the doctor should upload the all information that belongs to patient. 3- they should register an login.
Flow of events:	1-doctor and patient should login 2-appliction redirect them to home page 3- The doctor or patient click on history medical button. 4- show the doctor who booked with him
Post conditions:	Doctor and patient can show the patient medical history.
Exception:	Doctor can’t upload the information that belongs to patient. Then the doctor or patient can’t see the history medical The doctor or patient can’t register

Table 3 Show history use case table

2.6.4 View profile and edit profile

Use case name:	View Profile and edit profile
Number:	4,5
Actors:	Doctors, Patients
Actor's Goal:	To show profile and change personal information
Preconditions:	The user must log in to the system
Flow of events:	1- User clicks on account page 2- System displays user's personal account page. 3- User selects option edit information. 4- The system displays form to user. User enters new information saves the made changes by clicking on the Save button
Post conditions:	The system should display a message to ensure information updated
Exception:	1-when you want to change password you must enter the old password if you forget it You can't change it.

Table 4 View and modify profile

2.6.5 Make enquires

Use case name:	Make enquires
Number:	6
Actors:	Doctors, Patients
Actor's Goal:	To solve the problem that face them
Preconditions:	The user must log in to the system
Flow of events:	1- User clicks on account page 2- System displays user's personal account page. 3- User selects option support team 4- The system displays message to user. User enters the problem that face him in a message clicking on the submit button
Post conditions:	The system should redirect a message to whatsApp to inform him that problem has been sent.
Exception:	The message is empty and system request to enter message again.

Table 5 make enquires

2.6.6 Set available time

Use case name:	Set available time
Number:	7
Actors:	Doctors
Actor's Goal:	To set available time slot in schedule
Preconditions:	The doctor must log in to the system
Flow of events:	Doctor clicks on add appointment button. 2- doctor enter date and time that he available on it.
Post conditions:	The system should display doctor's available time.
Exception:	Doctor can't register before.

Table 6 set avaliable time

2.6.7Search doctor

Use case name:	Search doctor
Number:	8
Actors:	patients
Actor's Goal:	To search for a particular doctor
Preconditions:	User must create an account and logged in
Flow of events:	Patient can search by enter the name of doctor or by choose department and choose doctor Or selected by top doctor.
Post conditions:	The system should display doctor's information.
Exception:	Patient can't register before. Doctor isn't in system.

Table 7 search doctor

2.6.8 Book appointment

Use case name:	Book appointment
Number:	9
Actors:	patients
Actor's Goal:	To make online doctor appointment
Preconditions:	User must create an account and logged in
Flow of events:	1- patient search and select doctor 2-click on the button of book appointment 3-choose the time that doctor is available on it . 4-compelete payment method. 5-confirm appointment.
Post conditions:	The system should display message that time is booked.
Exception:	Patient can't register before.

Table 8 Book appointment

2.7 Sequence Diagram:

Sequence Diagrams are interaction diagrams that detail how operations are carried out.

They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when

2.7.1sequence diagram on Registration:

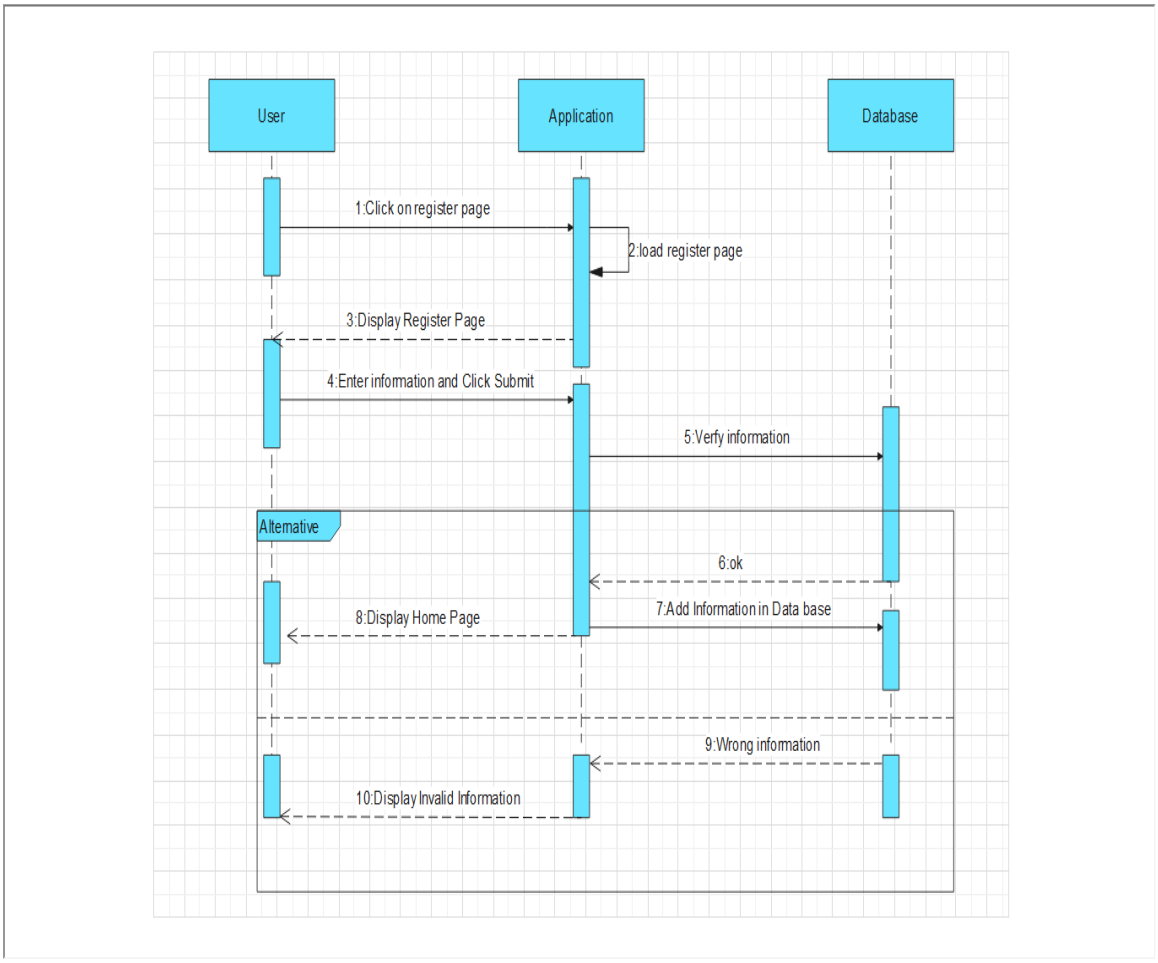


Figure 5 Sequence Diagram on Register

2.7.2sequence diagram on Login:

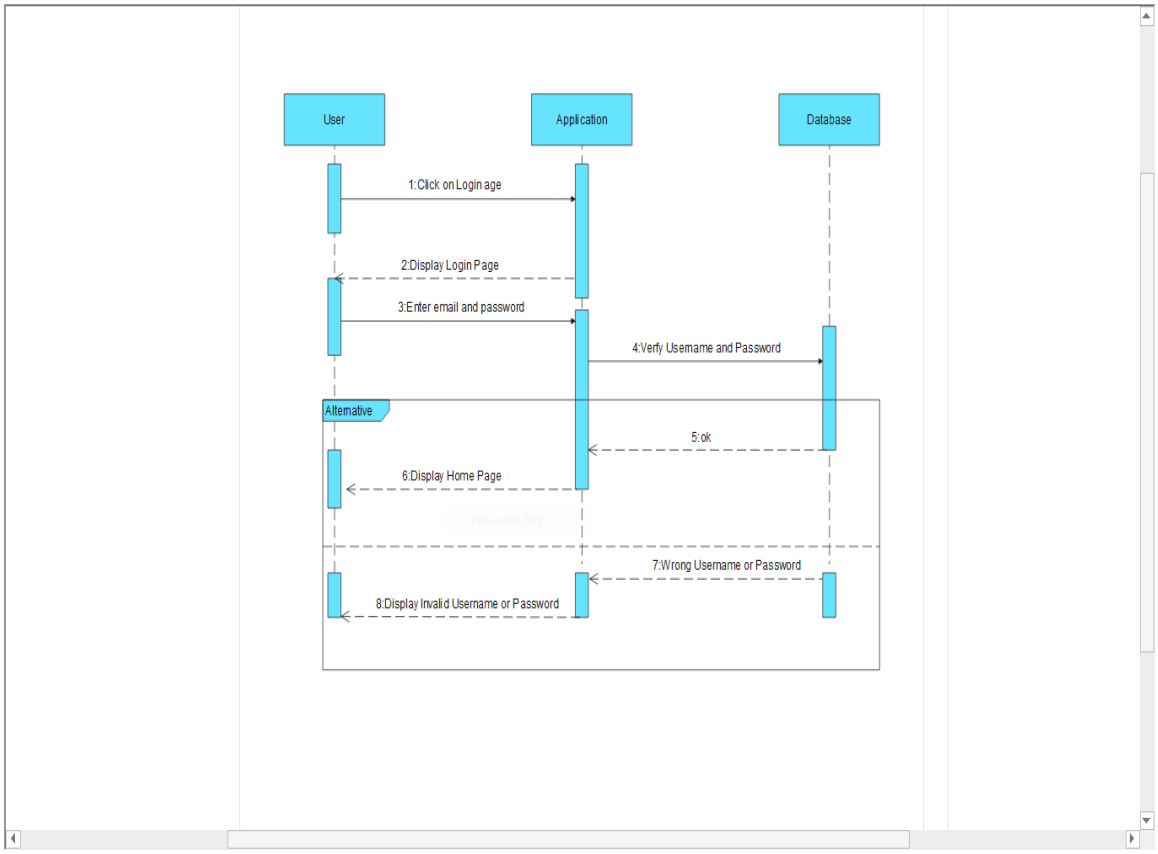


Figure 6Sequence Diagram on Login

2.7.3sequence diagram on search:

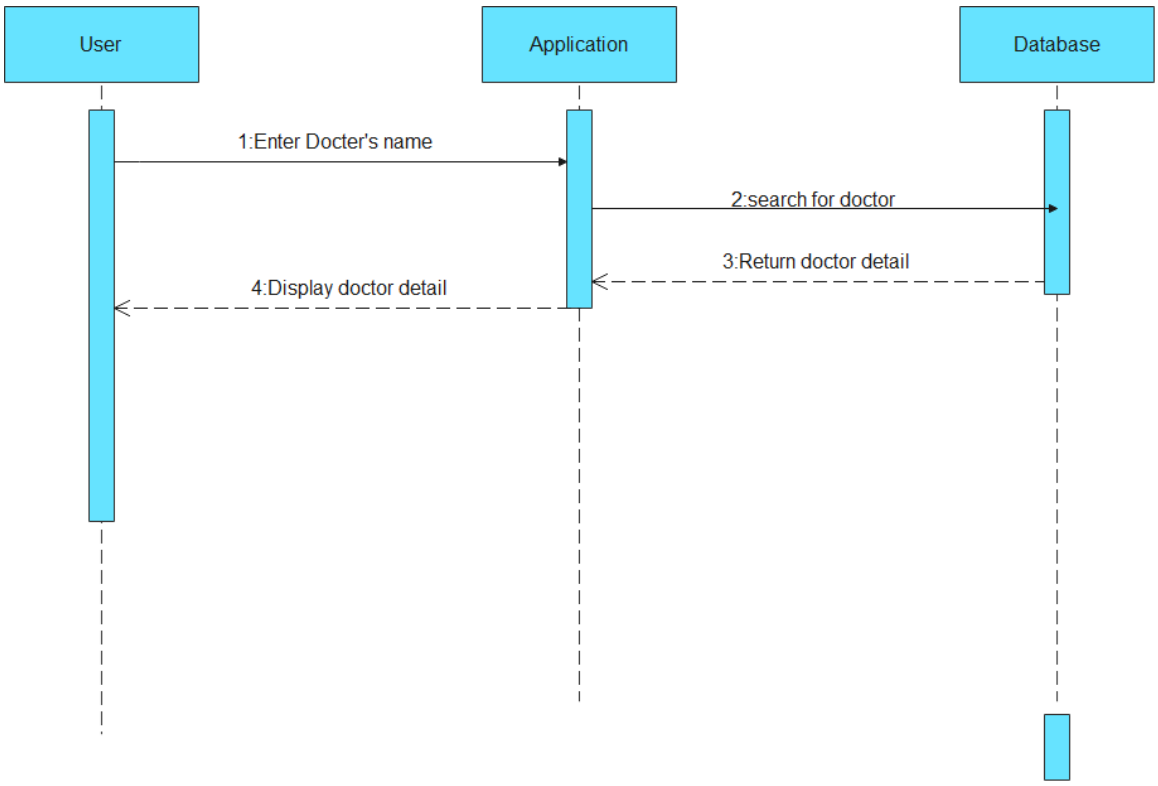


Figure 7sequence diagram on search

2.7.4sequence diagram on booking appointment:

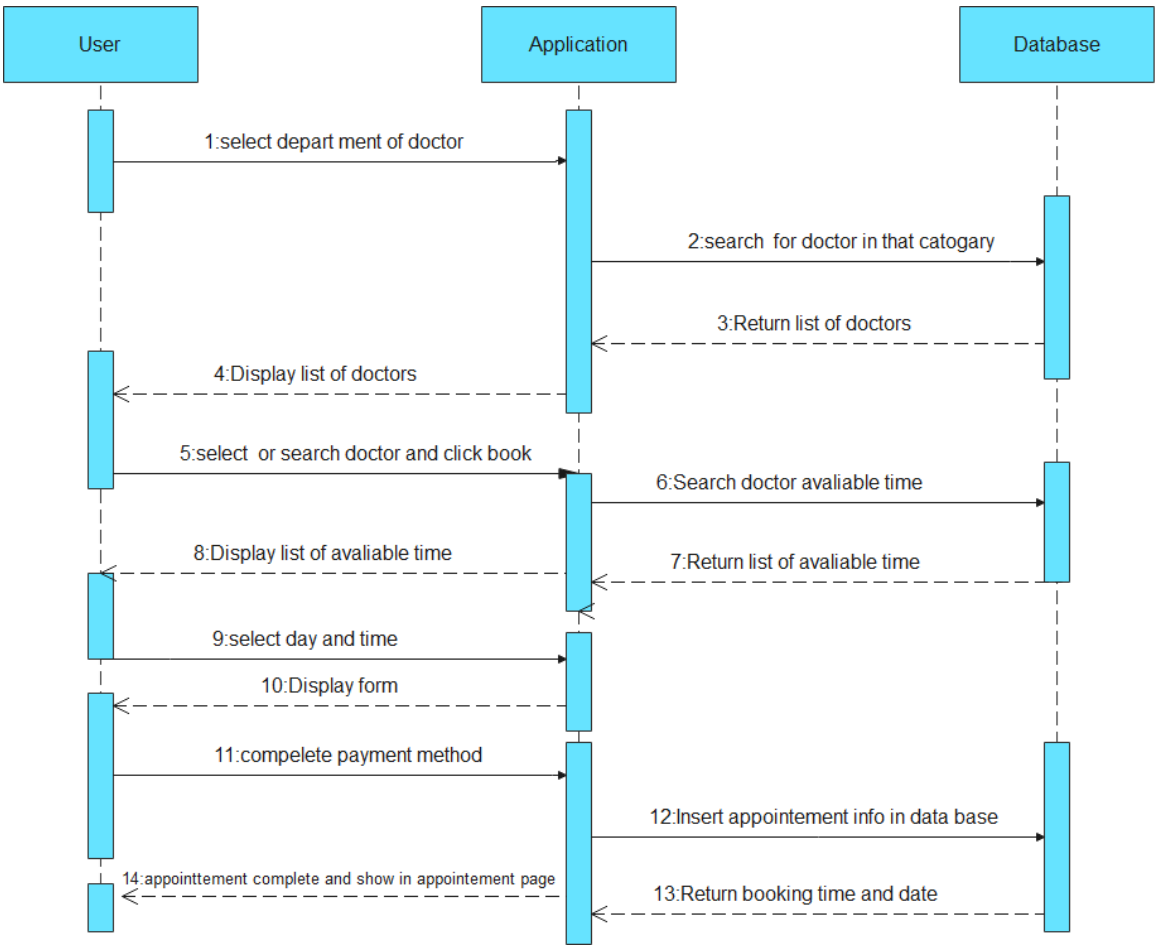


Figure 8 sequence diagram on booking appointment

2.7.5sequence diagram on Add appointment:

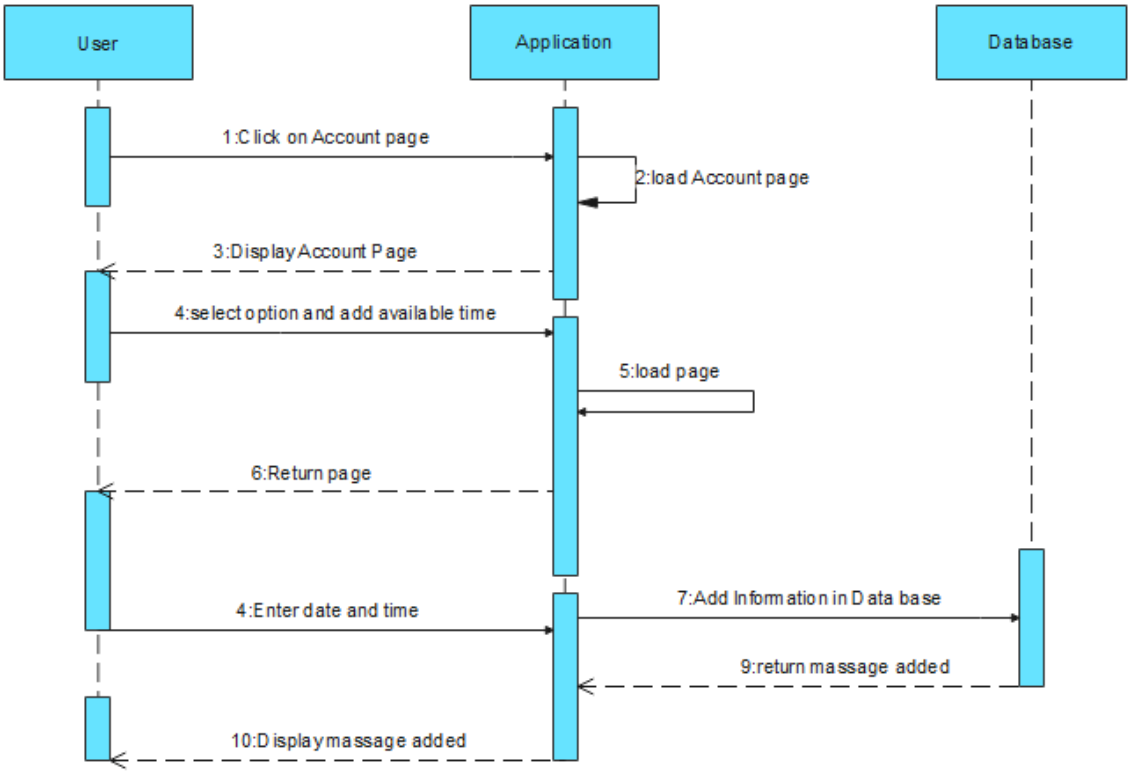


Figure 9 sequence diagram of Add appointment

Chapter 3: Results and Discussions

3.1 System Design Definition:

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

3.2 Types of system design:

1-Logical Design

2-Physical Design

3.2.1 Logical design:

Logical design is graphical representation of system showing the system's processes and data flow into and out of processes we use logical design to document information system because we represent the nature of system what tasks the system doing.

To represent logical design we use different diagram like Entity-Relationship Diagram (E-R Diagram).

3.2.1.1 Entity-Relationship Diagram:

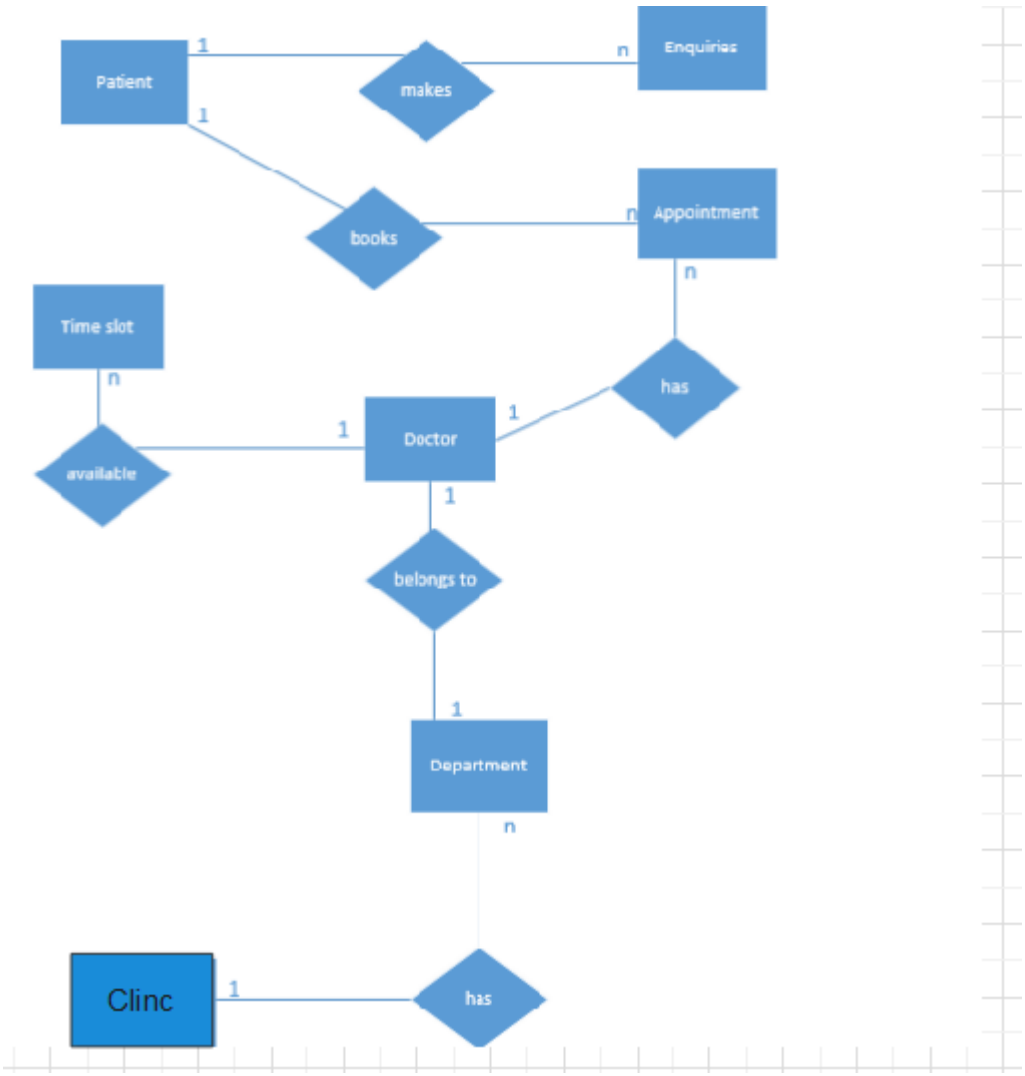


Figure 10 ER Diagram

3.2.2 Physical design:

Describes the actual input and output processes of the system, and focuses on how data is entered into a system, verified, processed, and displayed as output.

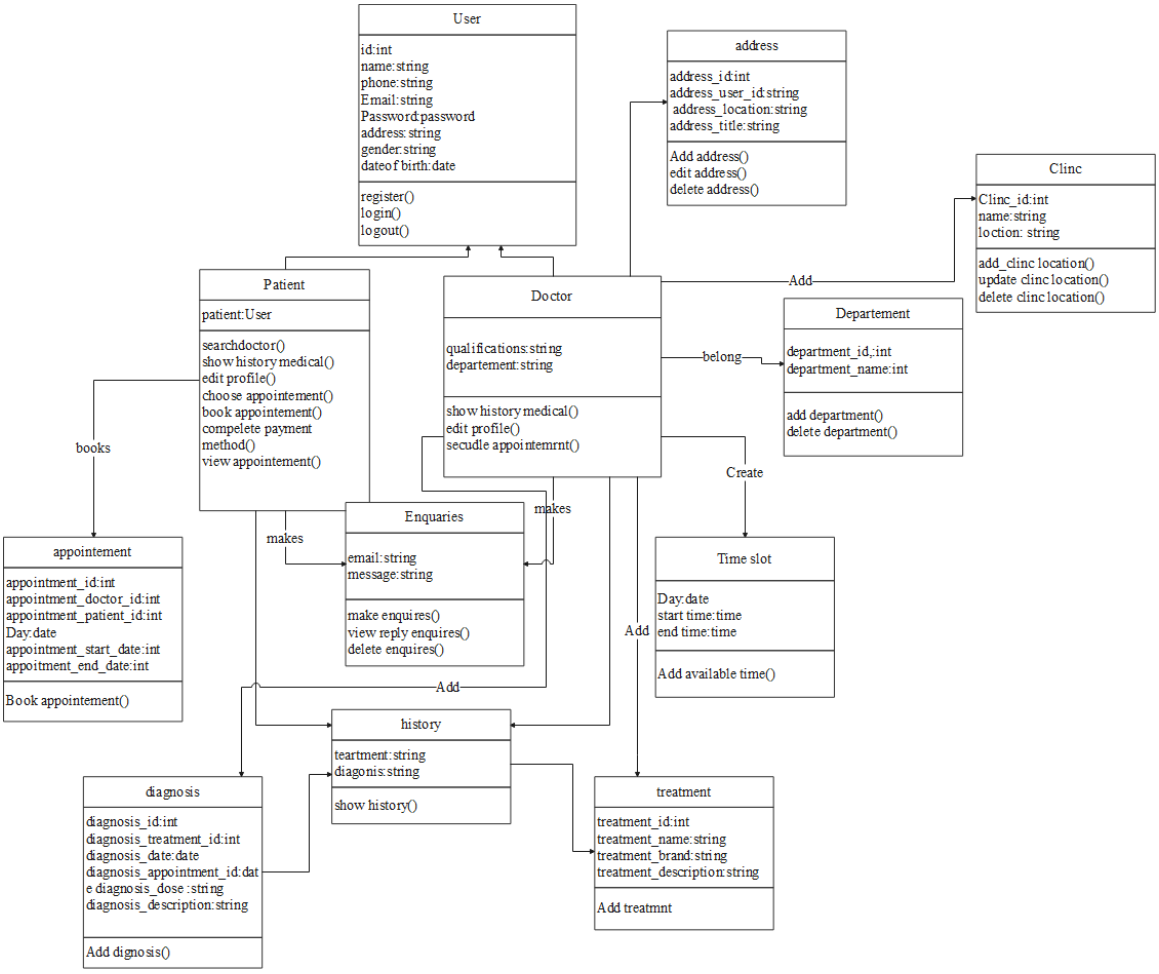


Figure 11 Class Diagram

3.3 User Interface (UI) Definition:

User interface (UI) design is the process of making interfaces screens in software or computerized devices with a focus on looks or style. And creating it easy to use and pleasurable interface to interact with it.

3.4 User Design screen:

1- Splash screen
2- Introslider screen
3-Register screen
4-medical sheet screen
5-login screen
6-forget password screen
7-verfication code screen
8- Reset password screen
9-Home screen
10-search screen
11-Doctor profile details screen
12-Book appointment
13-payment screen
14-appointement screen
15-History screen
16- Patient profile screen
17- Personal details screen.
18-change password screen
19-support team screen
20- privacy policy screen

3.4.1 Splash and Introslider screen:

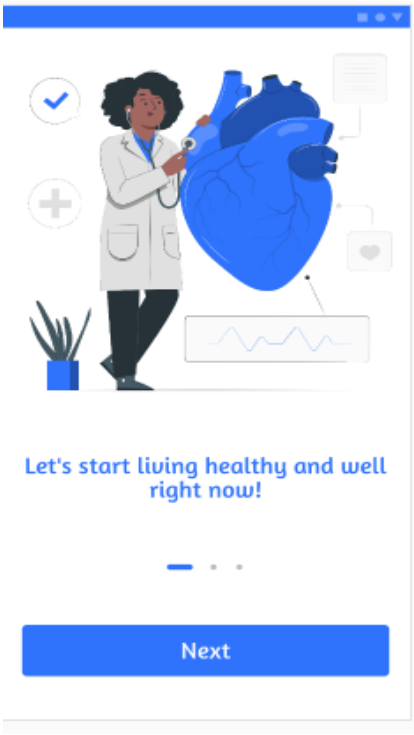
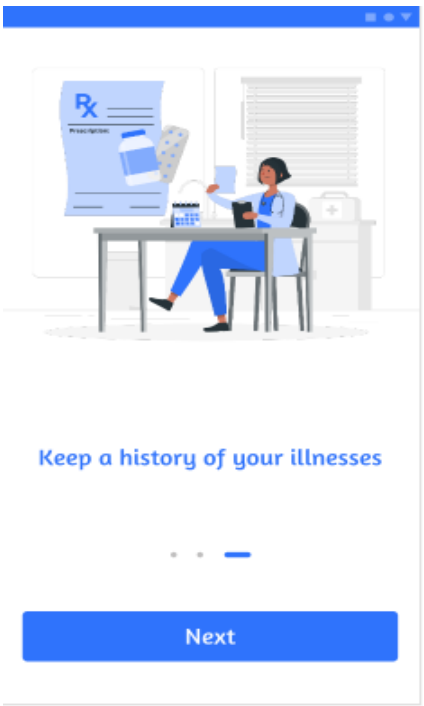
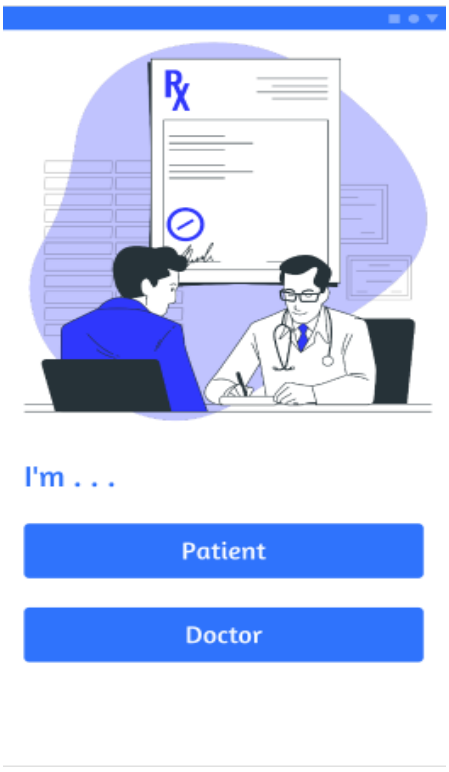


Figure 12 splash and Introslider screens





3.4.1 Register, medical sheet and login screen:

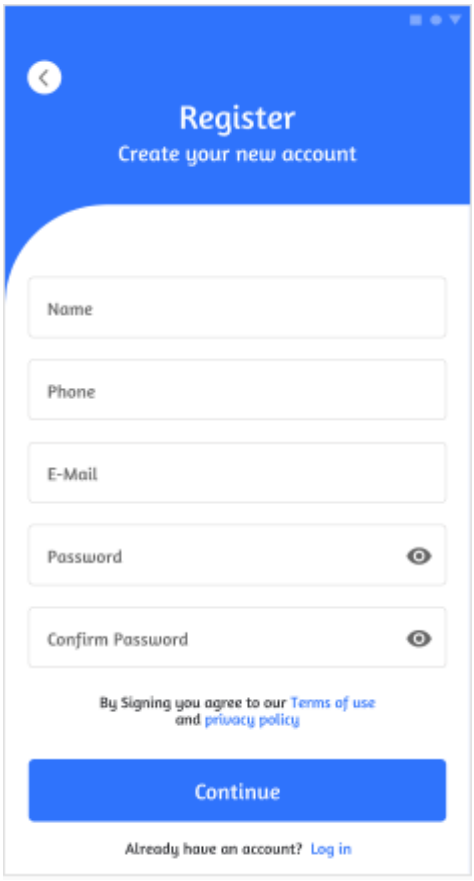


Figure 13 Register screen

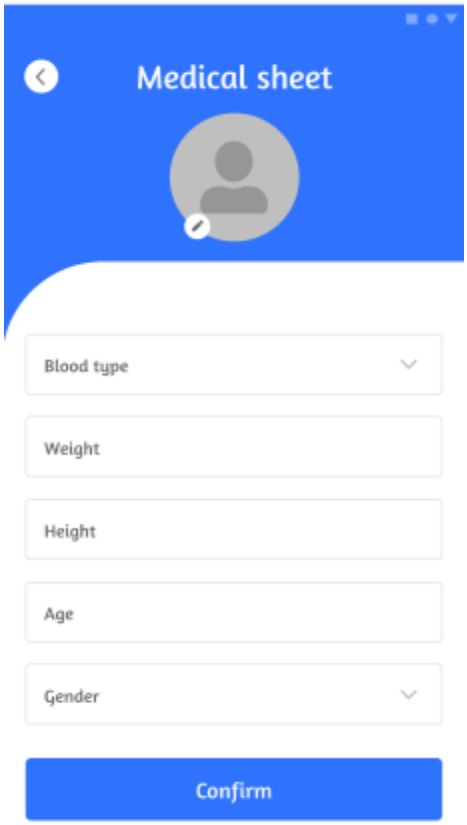


Figure 14 Medical sheet screen

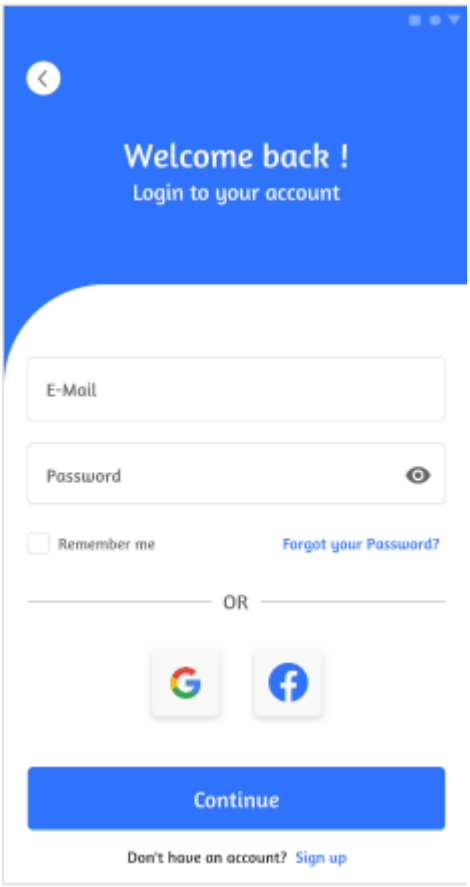


Figure 15 login screen

3.4.2 Forget, verification and reset password screens:

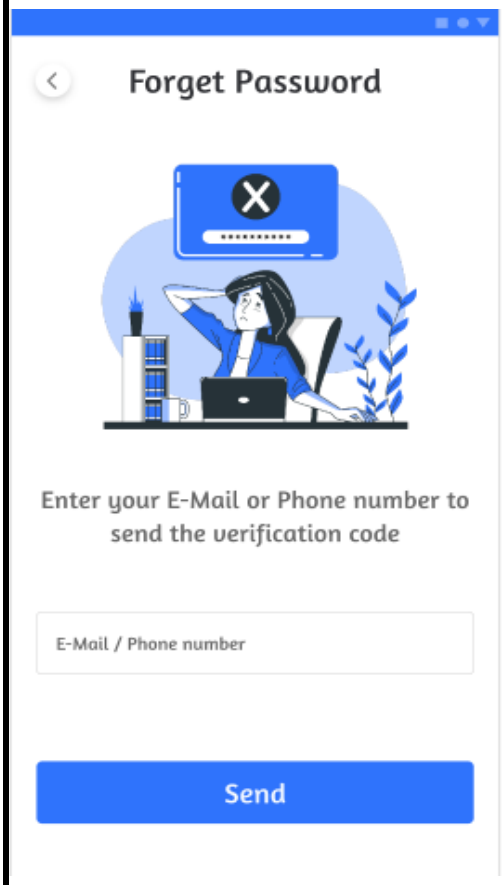


Figure 16 Forget Password

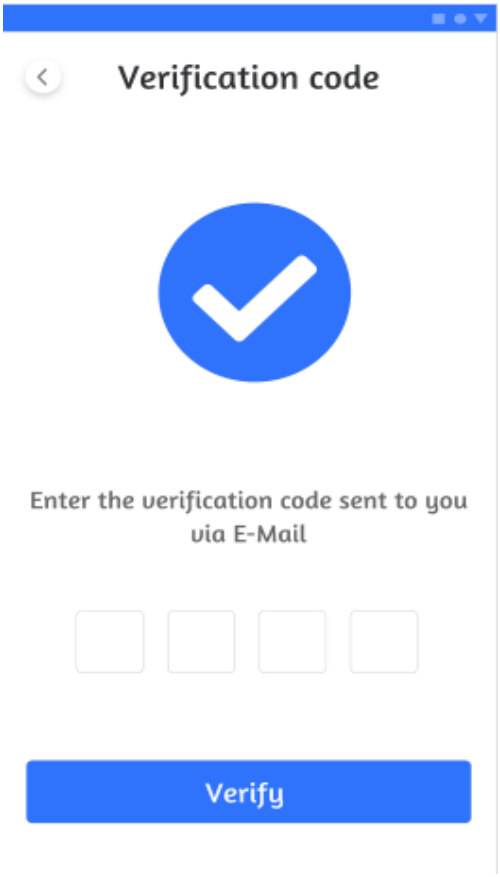


Figure 17 Verification code

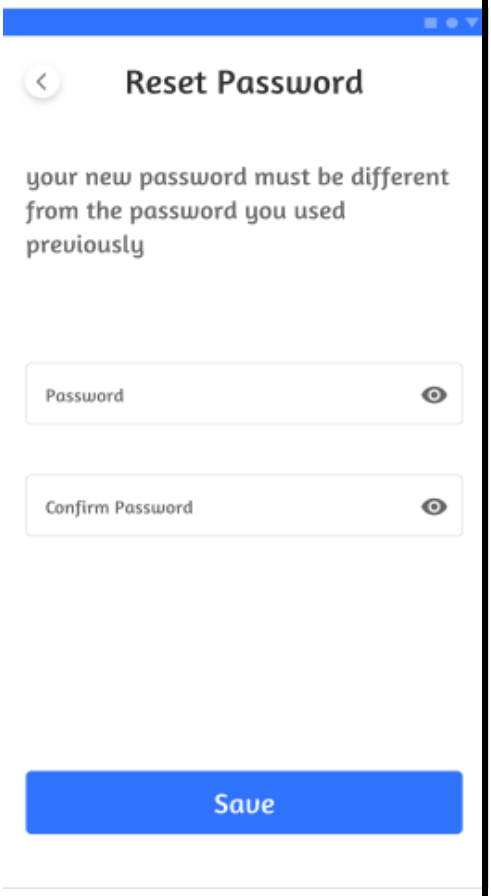


Figure 18Reset password

3.4.3 Home, search and appointment screens:

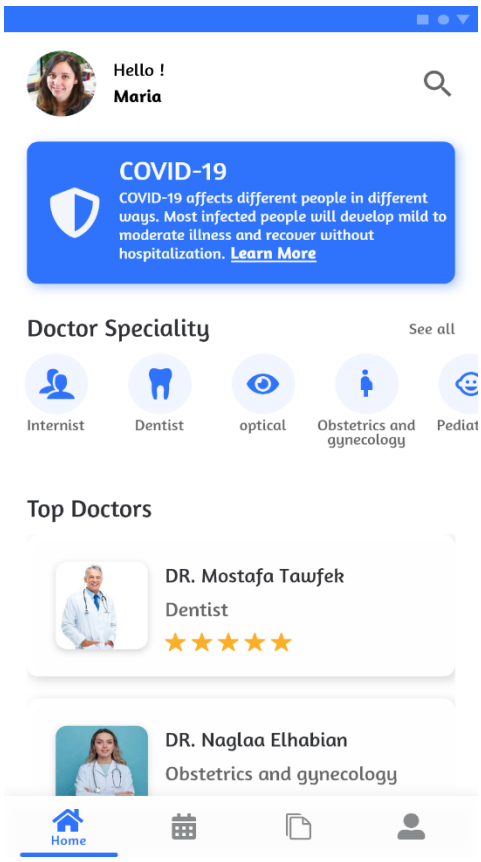


Figure 19 Home screen

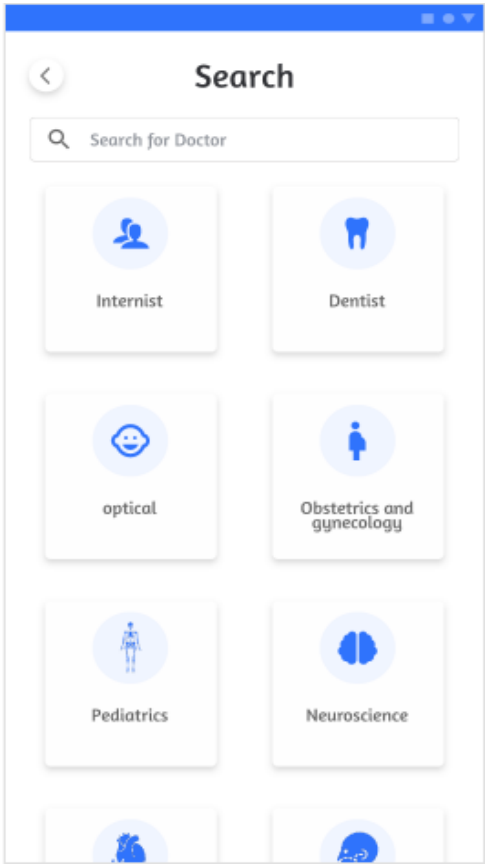


Figure 21 Search screen

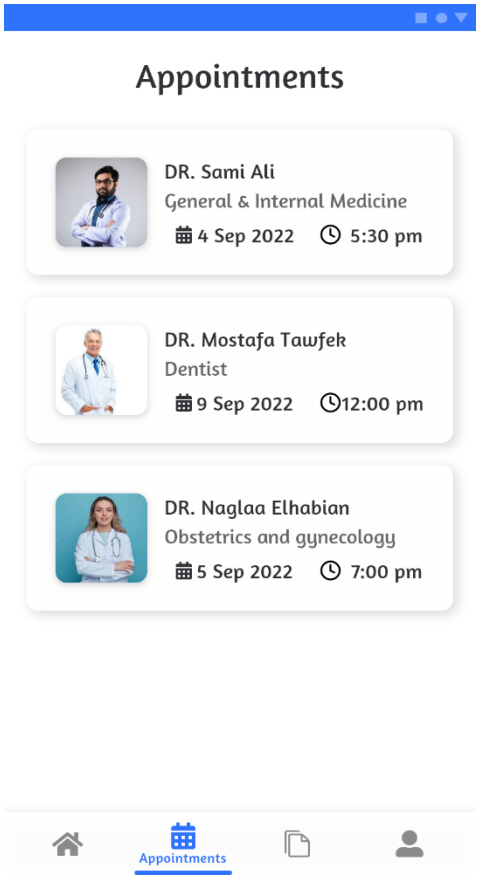


Figure 20 Appointments screen

3.4.4 Book appointment screens:

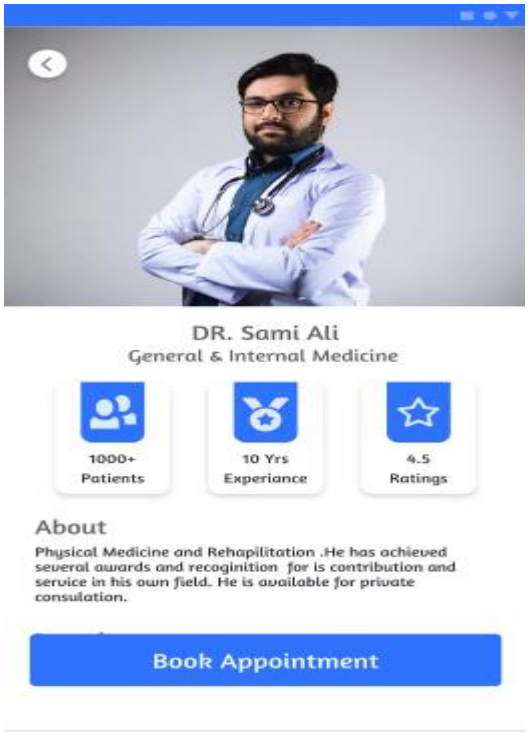
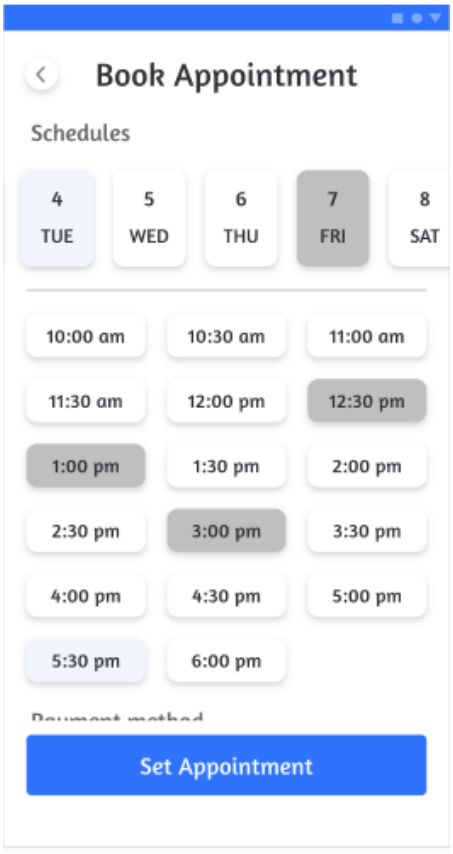
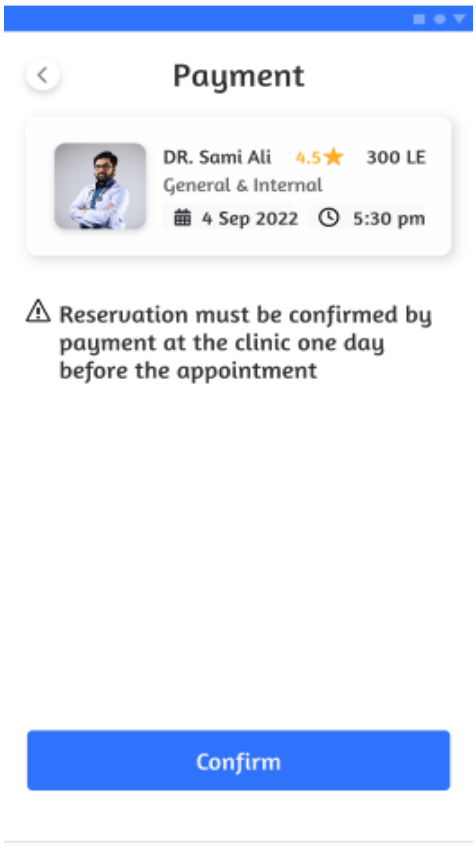
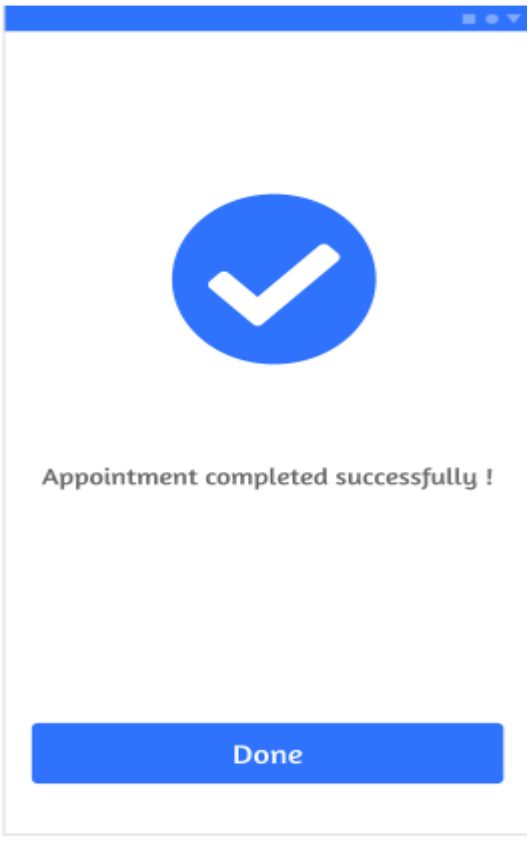


Figure 22 Book appointments screens







3.4.5 History screens:

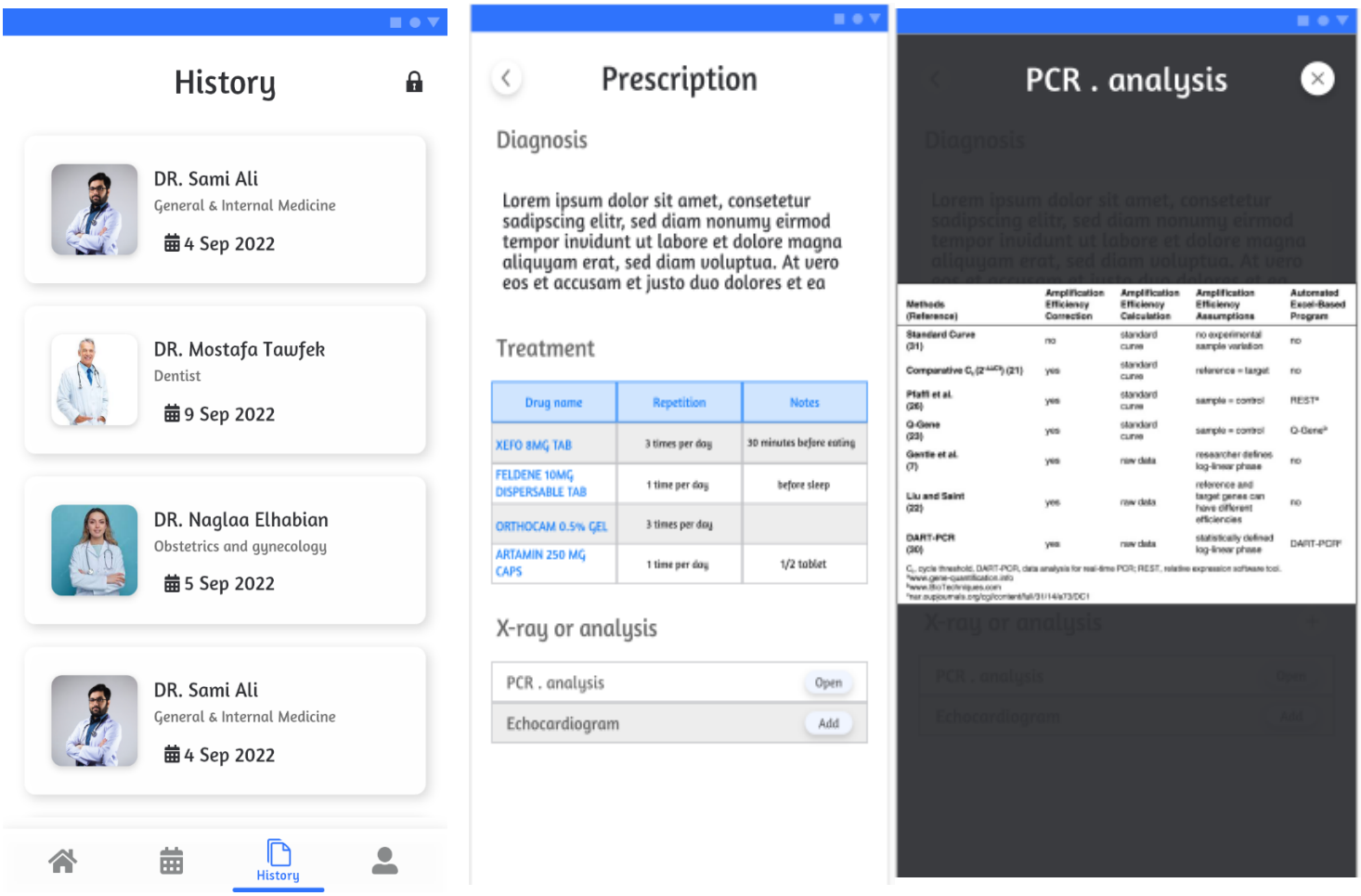
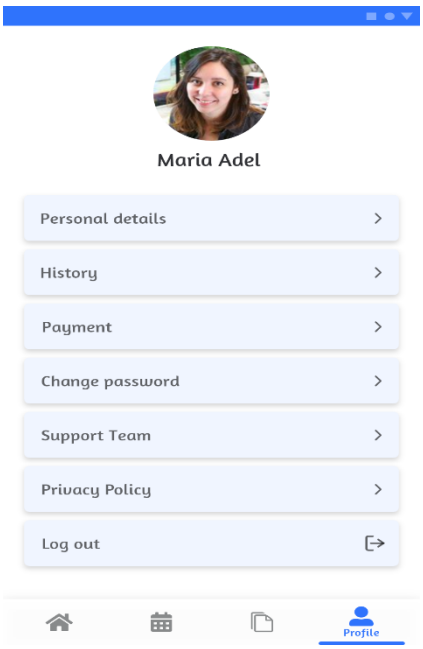
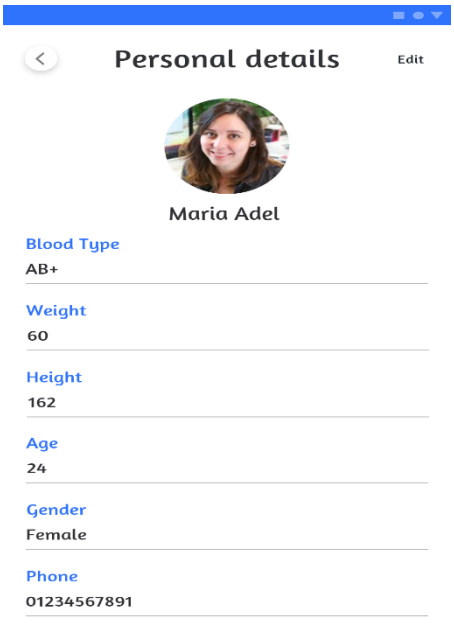
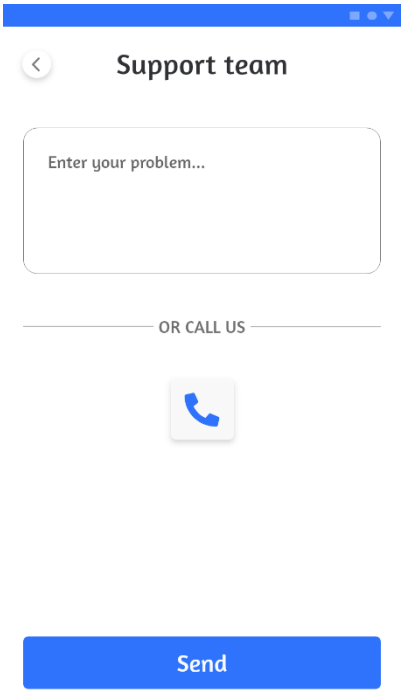


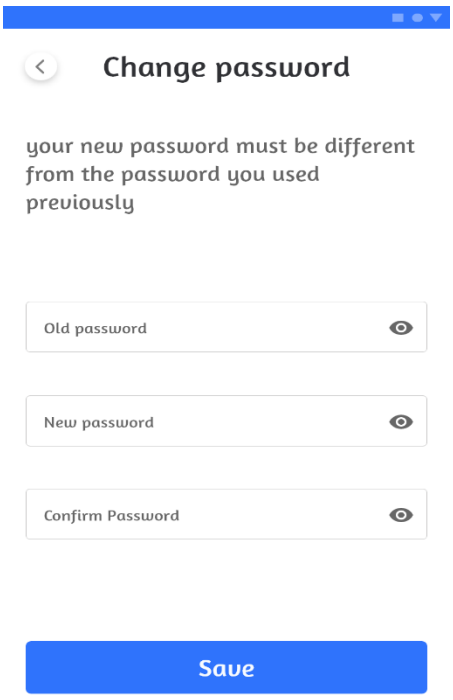
Figure 23 History Screen

3.4.6 profile screen, personal details and support team ,change password:



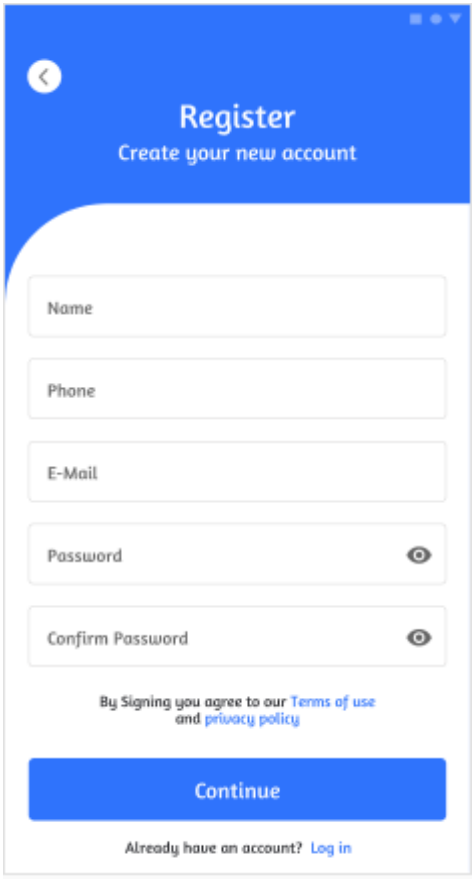


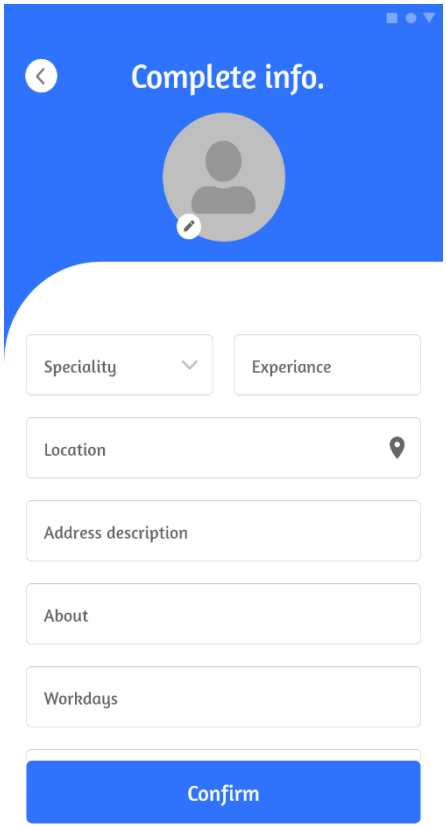


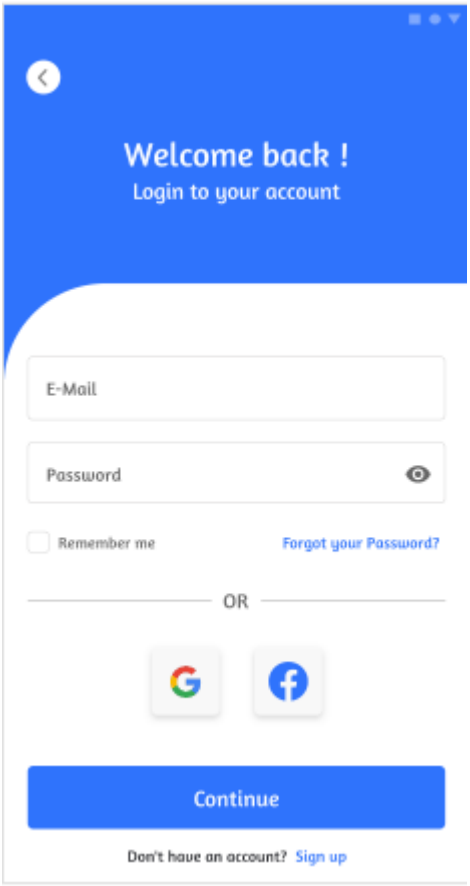


3.5 Doctor Design screens

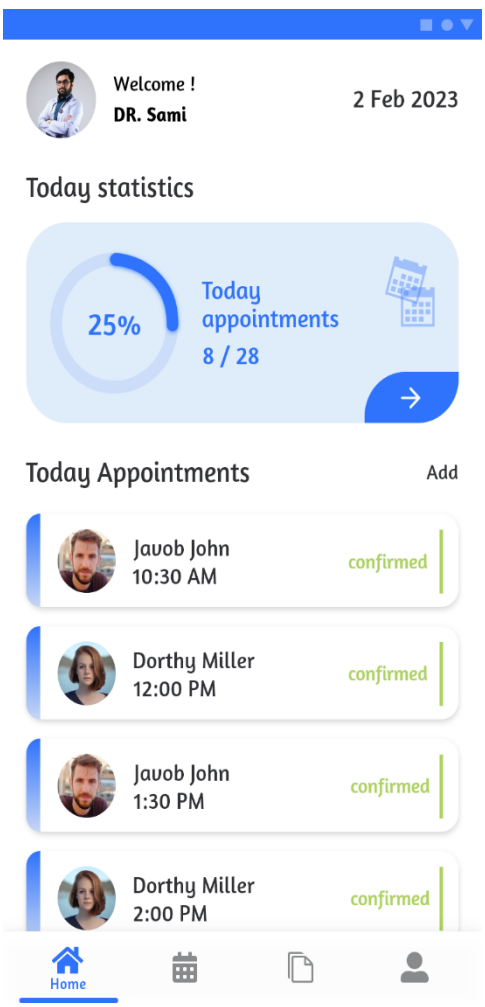
3.5.1 Register, complete information and login screen:

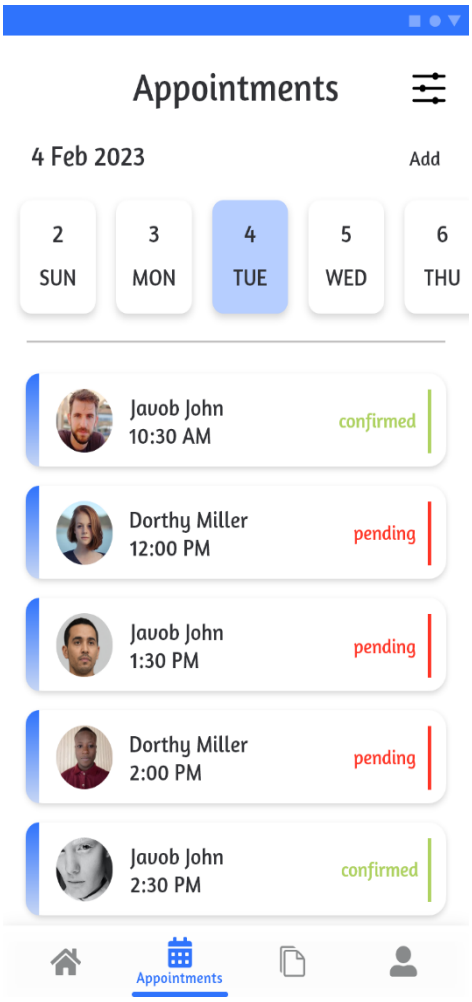


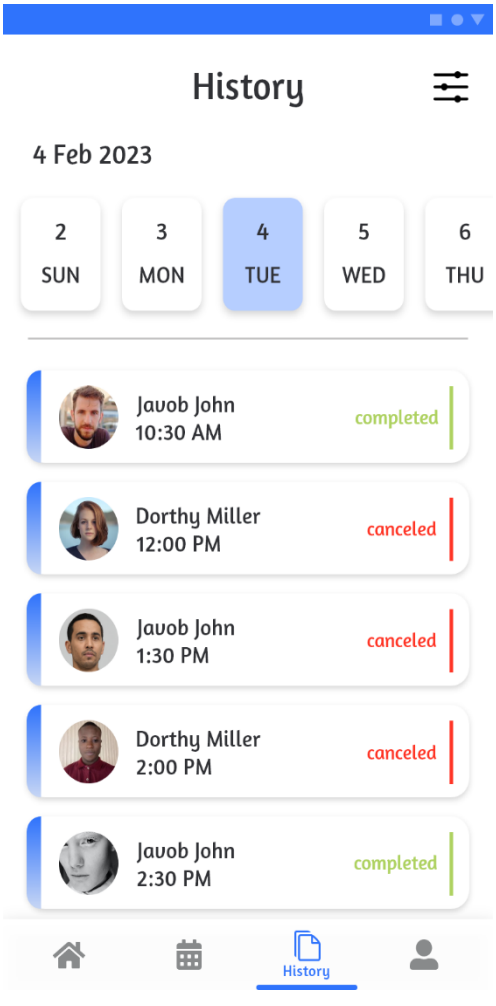




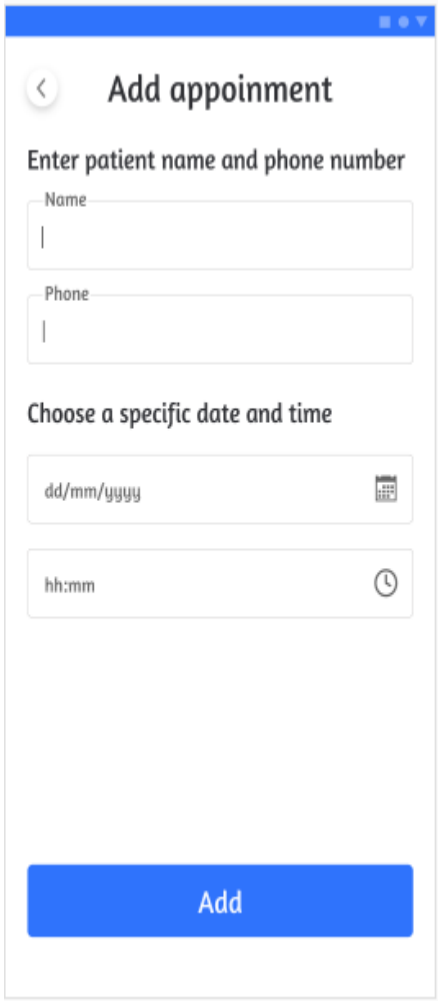
3.5.2 Home, and appointment, History screens:

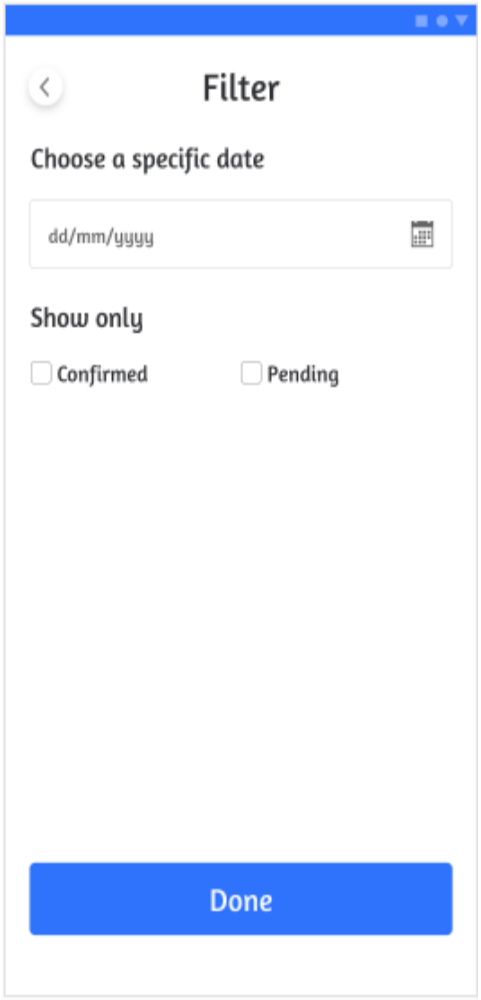


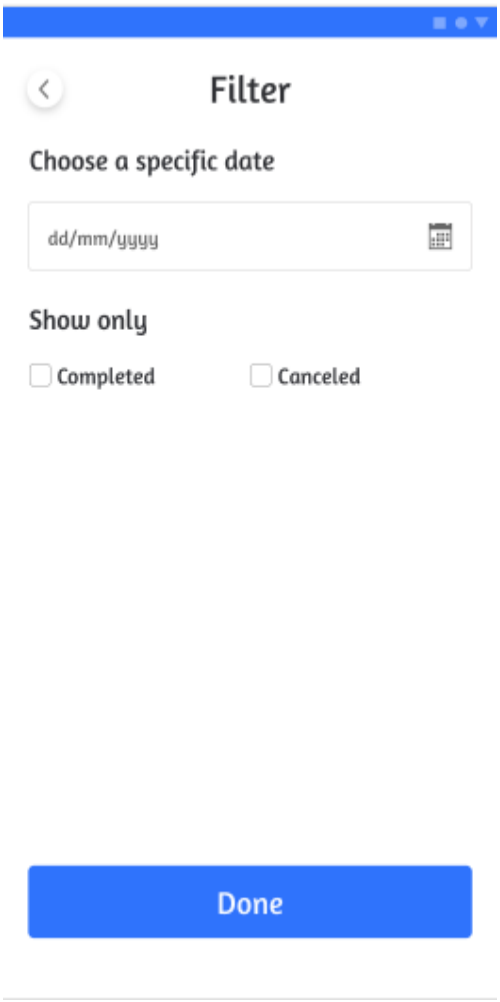




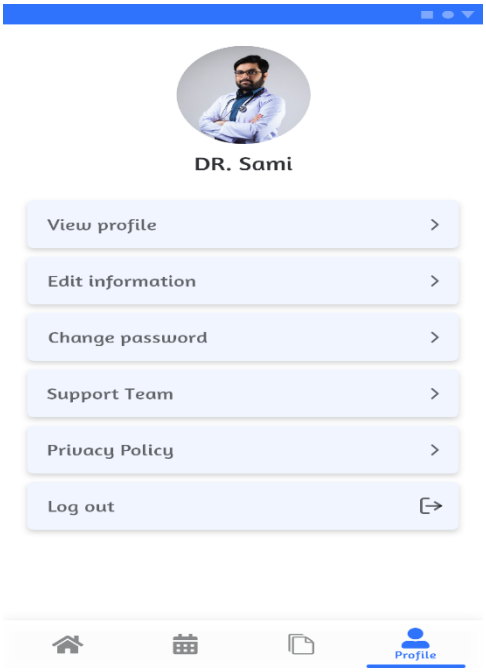
3.5.3 Add, Filter Appointment and Filter History:

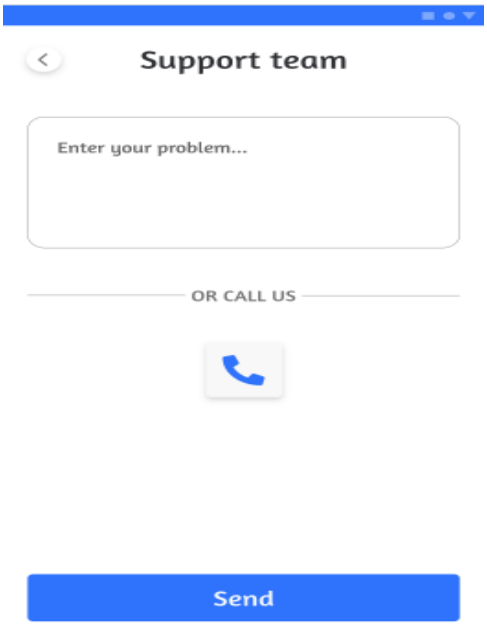


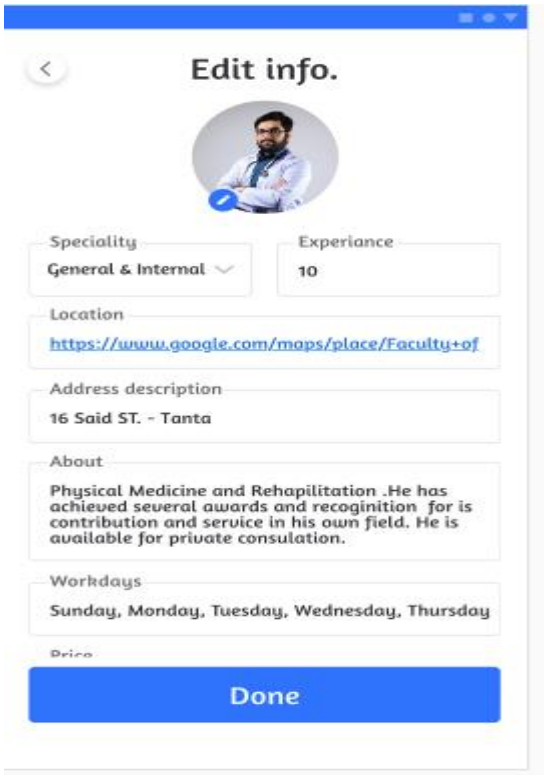


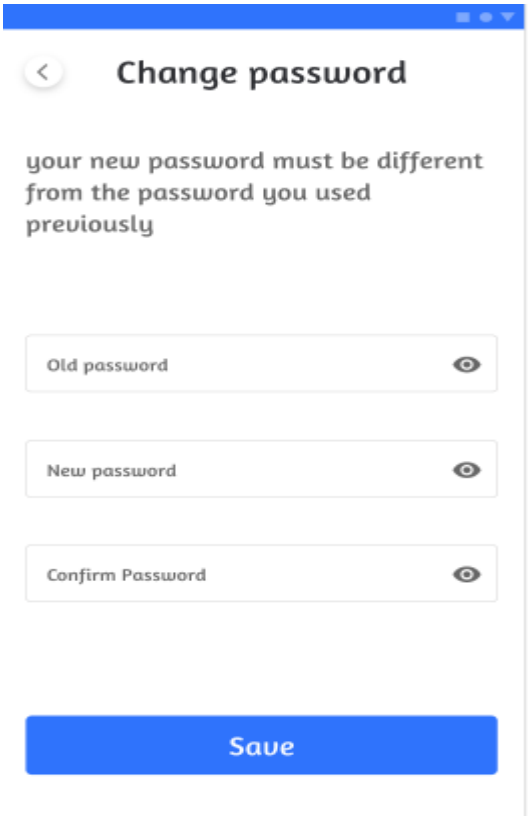


3.5.3 profile, edit profile, change password and Support Team:

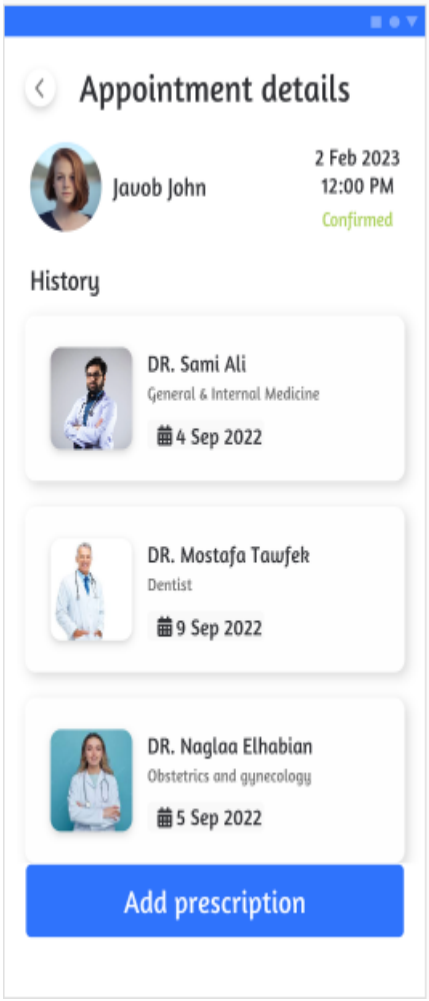


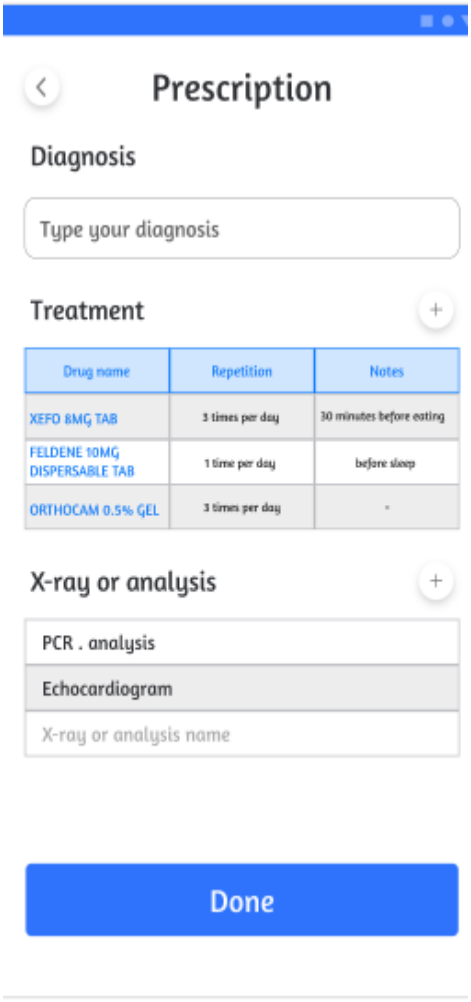


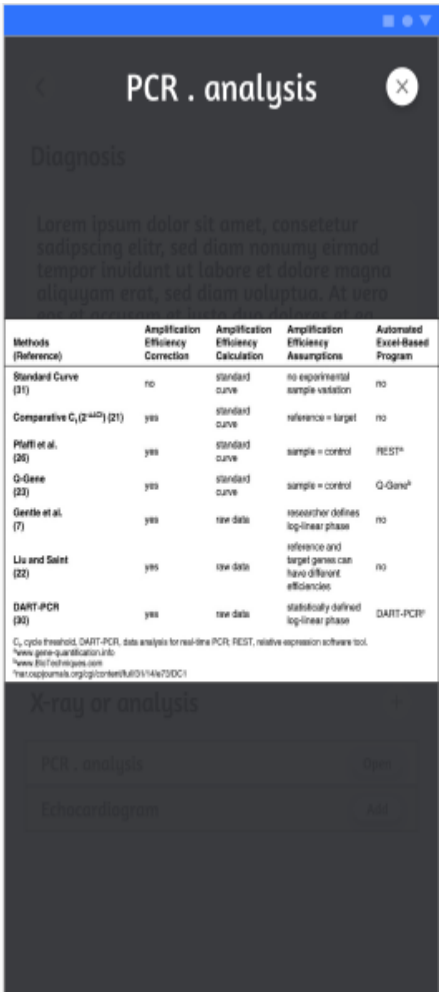




3.5.3Appointement details, Add prescription and prescription :







Chapter 4: practical work

4.1 System Implementation Definition:

System implementation is the process of carrying out, and executing of plan, model, design, and specification to build a complete system that performs a specific task and achieves system goals, by using technical tools.

4.2 Front-End development:

Front-end development primarily focuses on user experience. Using the related coding and design techniques, you as front-end developers build the elements of an application that are directly accessed by end-users with a goal of rendering the entire interface elegant, easy to use, fast, and secure, fostering user engagement and interaction

And building components that interact with users. Examples are the user interface, buttons, user-entered data, websites, and user experience (UX) features.

The major trend in front-end development in recent years is the growth of applications for mobile and smart devices, with users accessing applications from a growing number of devices with different screen sizes and interaction options.

4.2.1 Tools and techniques used in Front-End:

1-**React-native**

React Native is a JavaScript framework for building native mobile apps for android an ios with use the same code. It uses the React framework and offers large amount of inbuilt components and APIs.

It only works in the Front End part and can work with any back End where there is no difference. Applications communicate on the server according to the HTTP web protocol, which is a separate method from the programming logic, that is, what connects the front end and the back end is only the data that is transmitted over the network and is in the format JSON which can be handled in both Front & Back End

2-**React hook**

React Hooks enables the functional components to attach the local state to it, so that you can use React functionality without using a class component.

Advantages of react hooks:

- readable
- easy Code
- Writing a Functional component with state
- Performance boost up with Functional Component.
- Writing complex components became easier.

3- **Redux toolkit**

Redux Toolkit is a set of tools you can use for state management in React in place of Redux. The Redux team created it. Redux Toolkit offers a standardized approach to building redux code and comes with libraries and tools that make it simpler to create scalable, maintainable, and effective redux code. It used to store data and make application response performance are good.

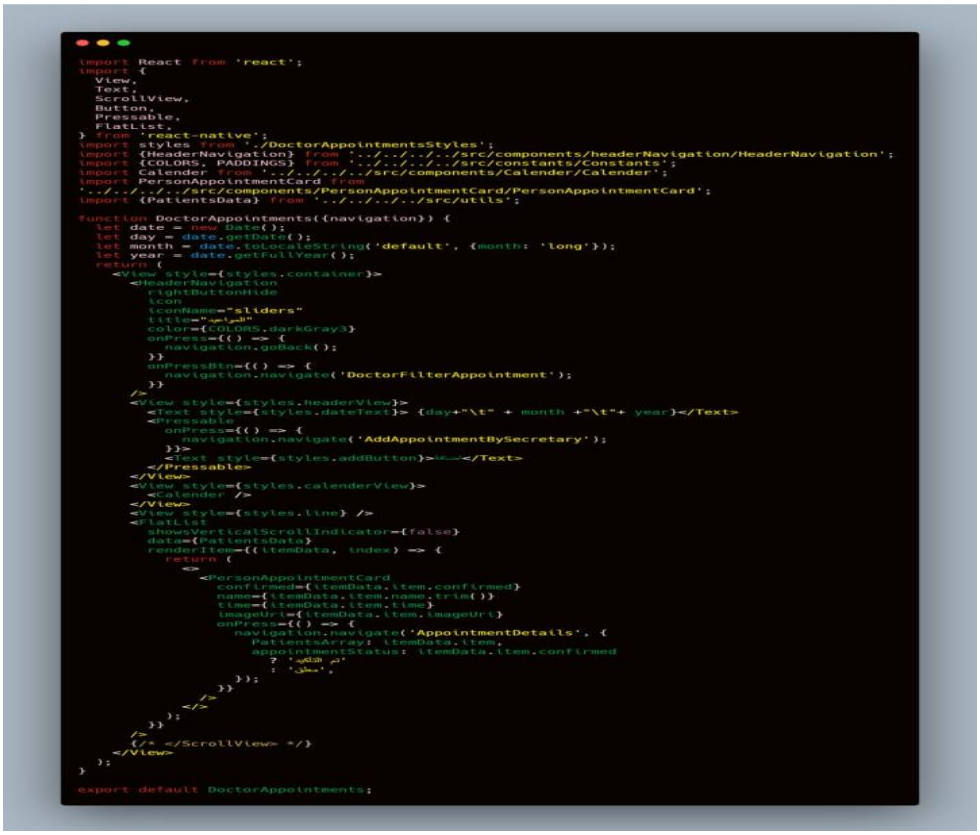
Redux Toolkit combines the reducer, action creator, and action concepts from Redux into a single concept called slices. This reduces the number of concepts that developers need to learn along with the amount of code developers .need to write, especially for apps using TypeScript

4-**javascript and modern JavaScript Es6**

JavaScript programming language is text-based and can be used on both client and server-side. It controls multimedia within web pages and allows them to become interactive. JavaScript empowers a developer to do many things like adding animation to images or updating content automatically on a page.

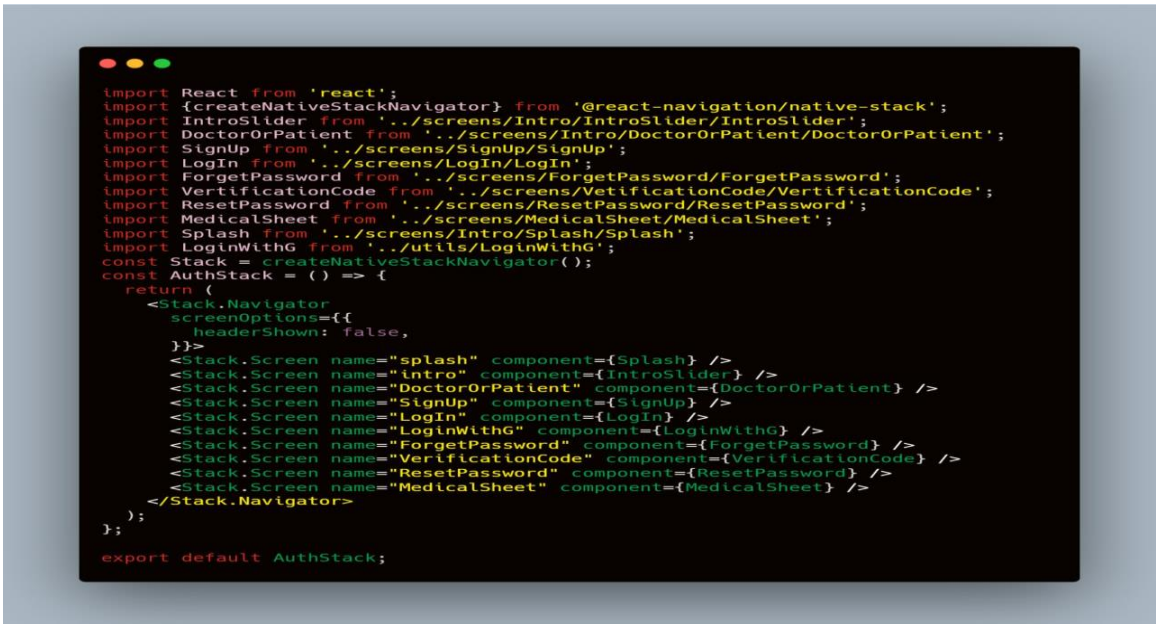
JavaScript ES6 brings new syntax and new awesome features to make your code more modern and more readable. It allows you to write less code and do more. ES6 introduces us to many great features like arrow functions, template strings, class destruction, Modules… and more.

4.2.2 Front-end Implementation:



```
import React from 'react';
import {
  View,
  Text,
  ScrollView,
  Button,
  Pressable,
  FlatList,
} from 'react-native';
import styles from '../DoctorAppointmentsStyles';
import {HeaderNavigation} from '../../../../src/components/headerNavigation/HeaderNavigation';
import {COLORS, PADDINGS} from '../../../../src/constants/Constants';
import Calender from '../../../../src/components/Calender/Calender';
import PersonAppointmentCard from
'../../../../src/components/PersonAppointmentCard/PersonAppointmentCard';
import {PatientsData} from '../../../../src/utils';

function DoctorAppointments({navigation}) {
  let date = new Date();
  let day = date.getDate();
  let month = date.toLocaleString('default', {month: 'long'});
  let year = date.getFullYear();
  return (
    <View style={styles.container}>
      <HeaderNavigation
        rightButtonHide
        icon
        iconName="sliders"
        title="مواعيدي"
        color={COLORS.darkGray3}
        onPress={() => {
          navigation.goBack();
        }}
        onPressBtn={() => {
          navigation.navigate('DoctorFilterAppointment');
        }}
      />
      <View style={styles.headerView}>
        <Text style={styles.dateText}> {day+"\t" + month +"\t"+ year}</Text>
        <Pressable
          onPress={() => {
            navigation.navigate('AddAppointmentBySecretary');
          }}>
          <Text style={styles.addButton}>إضافة</Text>
        </Pressable>
      </View>
      <View style={styles.calenderView}>
        <Calender />
      </View>
      <View style={styles.line} />
      <FlatList
        showsVerticalScrollIndicator={false}
        data={PatientsData}
        renderItem={({itemData, index}) => {
          return (
            <>
              <PersonAppointmentCard
                confirmed={itemData.item.confirmed}
                name={itemData.item.name.trim()}
                time={itemData.item.time}
                imageUrl={itemData.item.imageUrl}
                onPress={() => {
                  navigation.navigate('AppointmentDetails', {
                    PatientsArray: itemData.item,
                    appointmentStatus: itemData.item.confirmed
                      ? 'تم الكشف'
                      : 'معلق',
                  });
                }}
              />
            </>
          );
        }}
      />
      {/* </ScrollView> */}
    </View>
  );
}
export default DoctorAppointments;
```

Figure 25 Doctor appointement



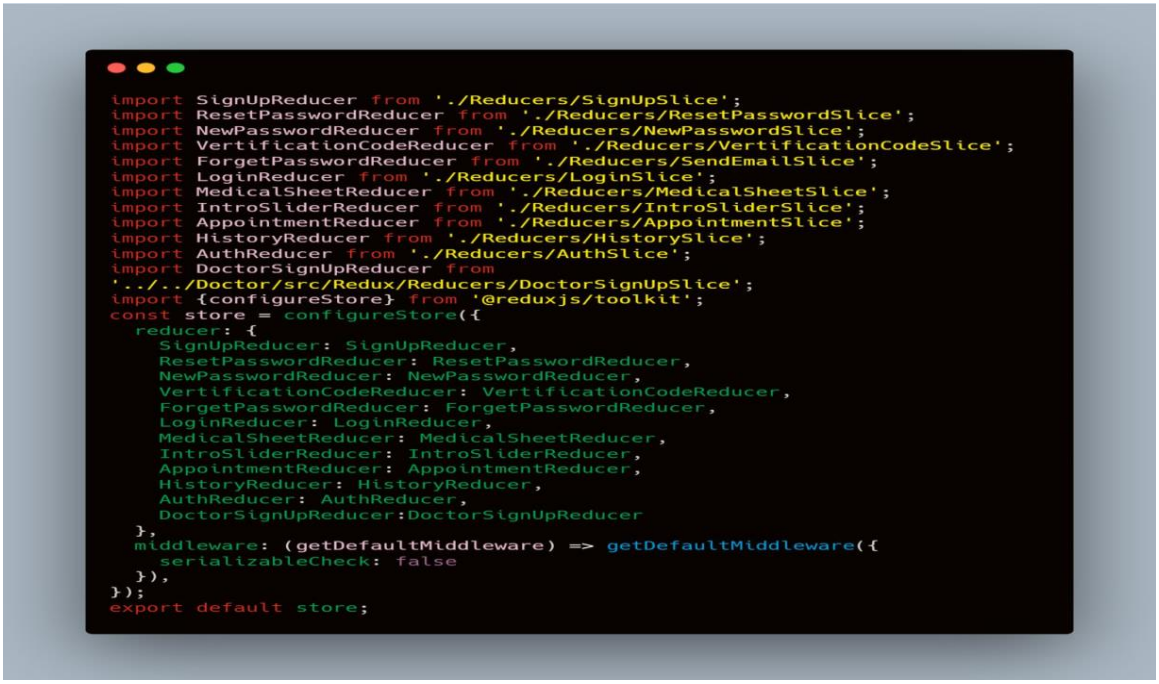
```
import React from 'react';
import {createNativeStackNavigator} from '@react-navigation/native-stack';
import IntroSlider from '../screens/Intro/IntroSlider/IntroSlider';
import DoctorOrPatient from '../screens/Intro/DoctorOrPatient/DoctorOrPatient';
import SignUp from '../screens/SignUp/SignUp';
import LogIn from '../screens/LogIn/LogIn';
import ForgetPassword from '../screens/ForgetPassword/ForgetPassword';
import VertificationCode from '../screens/VetificationCode/VertificationCode';
import ResetPassword from '../screens/ResetPassword/ResetPassword';
import MedicalSheet from '../screens/MedicalSheet/MedicalSheet';
import Splash from '../screens/Intro/Splash/Splash';
import LoginWithG from '../utils/LoginWithG';
const Stack = createNativeStackNavigator();
const AuthStack = () => {
  return (
    <Stack.Navigator
      screenOptions={{
        headerShown: false,
      }}>
      <Stack.Screen name="splash" component={Splash} />
      <Stack.Screen name="intro" component={IntroSlider} />
      <Stack.Screen name="DoctorOrPatient" component={DoctorOrPatient} />
      <Stack.Screen name="SignUp" component={SignUp} />
      <Stack.Screen name="LogIn" component={LogIn} />
      <Stack.Screen name="LoginWithG" component={LoginWithG} />
      <Stack.Screen name="ForgetPassword" component={ForgetPassword} />
      <Stack.Screen name="VerificationCode" component={VerificationCode} />
      <Stack.Screen name="ResetPassword" component={ResetPassword} />
      <Stack.Screen name="MedicalSheet" component={MedicalSheet} />
    </Stack.Navigator>
  );
};

export default AuthStack;
```

Figure 24Auth stack navigation

```
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';
import { setLoggedIn, setLoggedOut } from "../../Redux/Reducers/AuthSlice"
import AsyncStorage from '@react-native-async-storage/async-storage';
import { USER_TOKEN, USER_DATA } from '../../constants/Constants';
import axios from 'axios';
import Axios from '../../utils/axios';

const initState = {
  userInfo: null,
  //userToken,
  isLoading: false,
  error: null,
};
export const loginUser = createAsyncThunk(
  "Login/loginUser ",
  async (args, thunkAPI) => {
    const { rejectWithValue, dispatch } = thunkAPI
    try {
      await Axios
        ({
          method: "POST",
          url: "/general/login.php",
          data: args
        }).then((res) => {
          if (res.status == 200) {
            if (res.data.user_token) {
              AsyncStorage.setItem(USER_TOKEN,
JSON.stringify(res.data.user_token))
              AsyncStorage.setItem(USER_DATA, JSON.stringify(res.data))
              dispatch(setUserInfo(res.data))
              dispatch(setLoggedIn())
            } else {
              console.log(res.data);
            }
          } else {
            alert("حدث خطأ اثناء الاتصال بالخادم من فضلك حاول مجددا")
          }
        }).catch((err) => {
          console.log(err)
        })
    } catch (error) {
      console.log(rejectWithValue(error.message))
      return rejectWithValue(error.message);
    }
  })
const LoginSlice = createSlice({
  name: 'Login',
  initialState: initState,
  reducers: {
    setUserInfo: (state, action) => {
      state.userInfo = action.payload;
    }
  },
  extraReducers: (builder) => {
    builder.addCase(loginUser.pending, (state, action) => {
      state.isLoading = true;
      state.error = null;
    }),
      builder.addCase(loginUser.fulfilled, (state, action) => {
        state.isLoading = false;
        // state.userToken=action.payload.userToken;
      }),
      builder.addCase(loginUser.rejected, (state, action) => {
        state.isLoading = false;
        state.error = action.payload;
      }),
      builder.addCase(setLoggedOut, (state, action) => {
        state.userInfo = null;
      })
  }

});
export default LoginSlice.reducer;
export const { setUserInfo } = LoginSlice.actions;
```

Figure 26 login slice in redux

```
import {
  addDays,
  eachDayOfInterval,
  eachWeekOfInterval,
  format,
  subDays,
} from 'date-fns';
import React, {useState} from 'react';
import {StyleSheet, Text, View, TouchableOpacity,ScrollView} from 'react-
native';
import PagerView from 'react-native-pager-view';
import {RFValue} from 'react-native-responsive-fontsize';
import {FONTS, COLORS} from '../../constants/Constants';

const dates = eachWeekOfInterval(
  {
    start: subDays(new Date(), 14),
    end: addDays(new Date(), 14),
  },
  {
    weekStartsOn: 1,
  },
).reduce((acc, cur) => {
  const allDays = eachDayOfInterval({
    start: cur,
    end: addDays(cur, 4),
  });
  acc.push(allDays);
  return acc;
}, []);
function Calender() {
  const [chosenDay, setChosenDay] = useState(null);
  return (
    <PagerView style={styles.container}>
      {dates.map((week, index) => {
        return (
          <View key={index} style={styles.calenderView}>
            {week.map((day, idx) => {
              const dayName = format(day, 'eee');
              return (
                <TouchableOpacity
                  onPress={() => setChosenDay(idx)}
                  key={idx}
                  style={[
                    styles.dateCard,
                    idx === chosenDay
                      ? {backgroundColor: COLORS.lightBlue}
                      : null,
                  ]}>
                  <Text style={styles.dayText}>{day.getDate()}</Text>
                  <Text style={styles.dayText}>{dayName}</Text>
                </TouchableOpacity>
              );
            })}
          </View>
        );
      })}
    </PagerView>
  );
}

const styles = StyleSheet.create({
  container: {
    height: RFValue(85),
    width: '100%',
    alignItems: 'center',
  },
  calenderView: {
    flexDirection: 'row',
    justifyContent: 'space-around',
  },
  dateCard: {
    alignSelf: 'center',
    alignItems: 'center',
    justifyContent: 'center',
    width: RFValue(55),
    height: RFValue(70),
    backgroundColor: '#fff',
    elevation: 2,
    marginEnd: RFValue(2),
    borderRadius: RFValue(10),
  },
  dayText: {
    fontSize: FONTS.h8,
    fontWeight: 'bold',
  },
});

export default Calender;
```



```
import SignUpReducer from './Reducers/SignUpSlice';
import ResetPasswordReducer from './Reducers/ResetPasswordSlice';
import NewPasswordReducer from './Reducers/NewPasswordSlice';
import VertificationCodeReducer from './Reducers/VertificationCodeSlice';
import ForgetPasswordReducer from './Reducers/SendEmailSlice';
import LoginReducer from './Reducers/LoginSlice';
import MedicalSheetReducer from './Reducers/MedicalSheetSlice';
import IntroSliderReducer from './Reducers/IntroSliderSlice';
import AppointmentReducer from './Reducers/AppointmentSlice';
import HistoryReducer from './Reducers/HistorySlice';
import AuthReducer from './Reducers/AuthSlice';
import DoctorSignUpReducer from
'../../Doctor/src/Redux/Reducers/DoctorSignUpSlice';
import {configureStore} from '@reduxjs/toolkit';
const store = configureStore({
  reducer: {
    SignUpReducer: SignUpReducer,
    ResetPasswordReducer: ResetPasswordReducer,
    NewPasswordReducer: NewPasswordReducer,
    VertificationCodeReducer: VertificationCodeReducer,
    ForgetPasswordReducer: ForgetPasswordReducer,
    LoginReducer: LoginReducer,
    MedicalSheetReducer: MedicalSheetReducer,
    IntroSliderReducer: IntroSliderReducer,
    AppointmentReducer: AppointmentReducer,
    HistoryReducer: HistoryReducer,
    AuthReducer: AuthReducer,
    DoctorSignUpReducer:DoctorSignUpReducer
  },
  middleware: (getDefaultMiddleware) => getDefaultMiddleware({
    serializableCheck: false
  }),
});
export default store;
```

Figure 27 store in redux

4.3 Back-End development:

Backend development refers to the technical aspects of the website/application. It is also called server-side development, which is the part of the website which is not visible to users. The primary aim of backend development is to store and organize data and ensure everything on the client-side works perfectly.

4.3.1 Tools and techniques used in Back-end:

there are many packages that you can choose among to start build the back end of your website or web application. For example, you can use the programming language PHP and the database system MySQL to start build the back end of Application, or you can use the programming language Python with the framework Django and the database system MySQL for the back end of your application. The back end of the application was built using the package of PHP and the database system MySQL

1-Mysql

We have used “MySQL” for designing our database. MySQL is an open source relational database management system. It runs as a server and allows multiple users to manage and create numerous databases. It is a central component in the LAMP stack of open source web application software that is used to create websites. LAMP stands for Linux, Apache, MySQL, and PHP

2-php

PHP can be used to develop dynamic and interactive web pages, applications, and ecommerce platforms. PHP is well suited to various web tasks, from generating dynamic web pages, sending emails, and collecting web forms to sending and receiving cookies. As PHP can store, delete, and modify information in a database, it becomes a strong foundation for creating web applications, including content management systems (CMS), custom online databases, ecommerce websites, gaming applications, and community portals.

4.4 Back-end Implementation:

4.4.1 Login, Send otp:

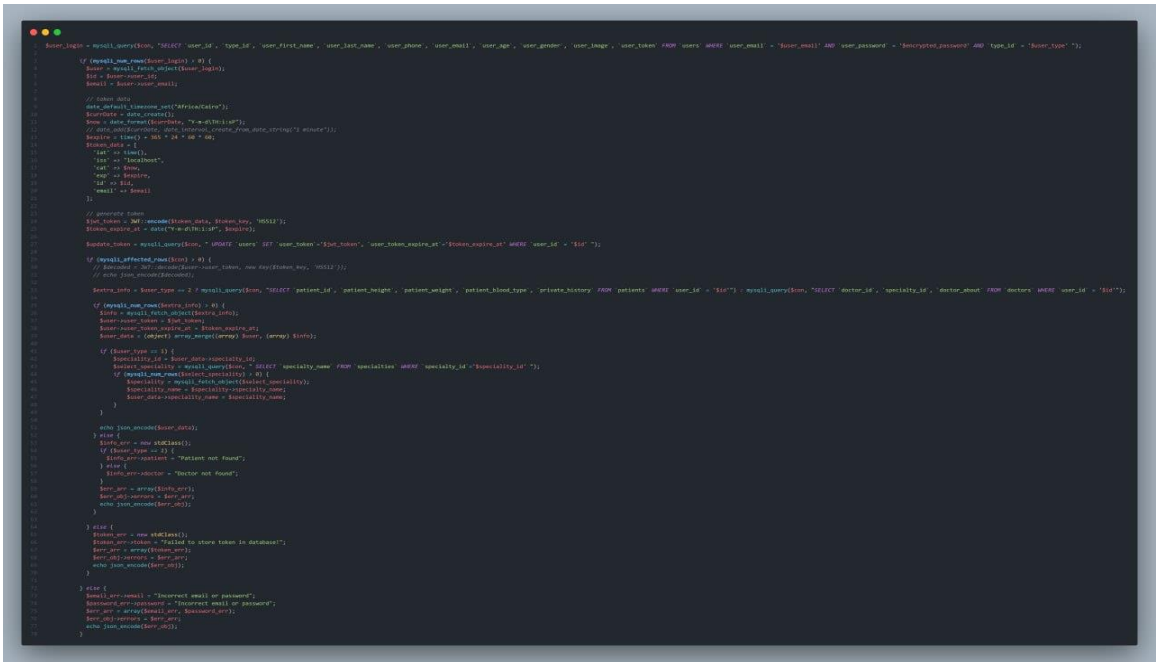


Figure 29 login in patient and doctors

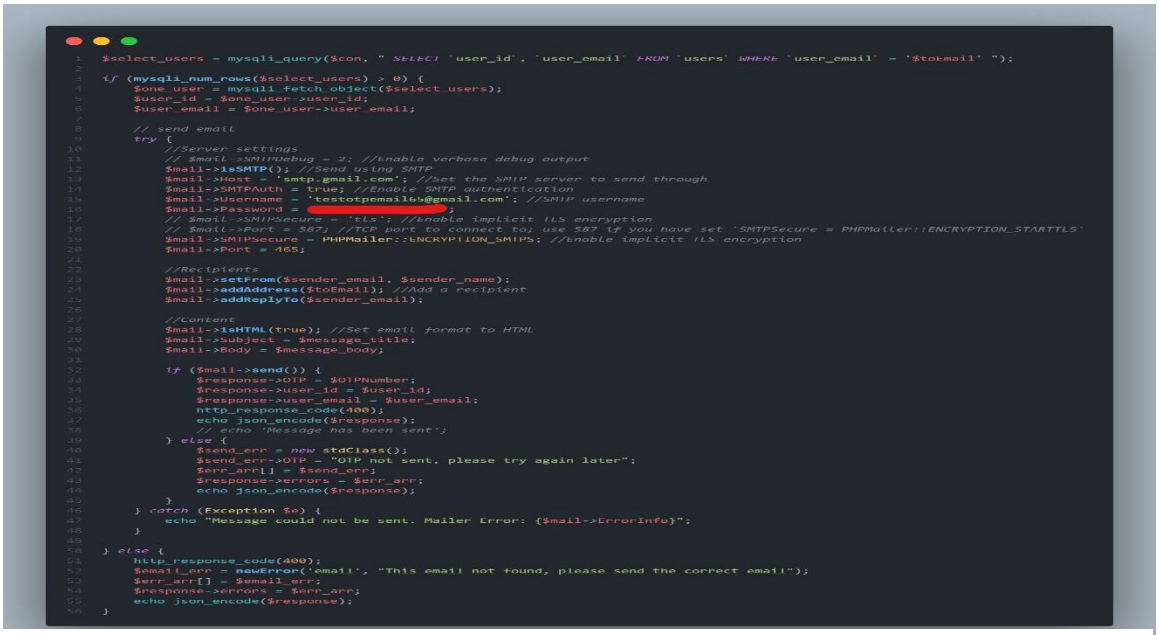
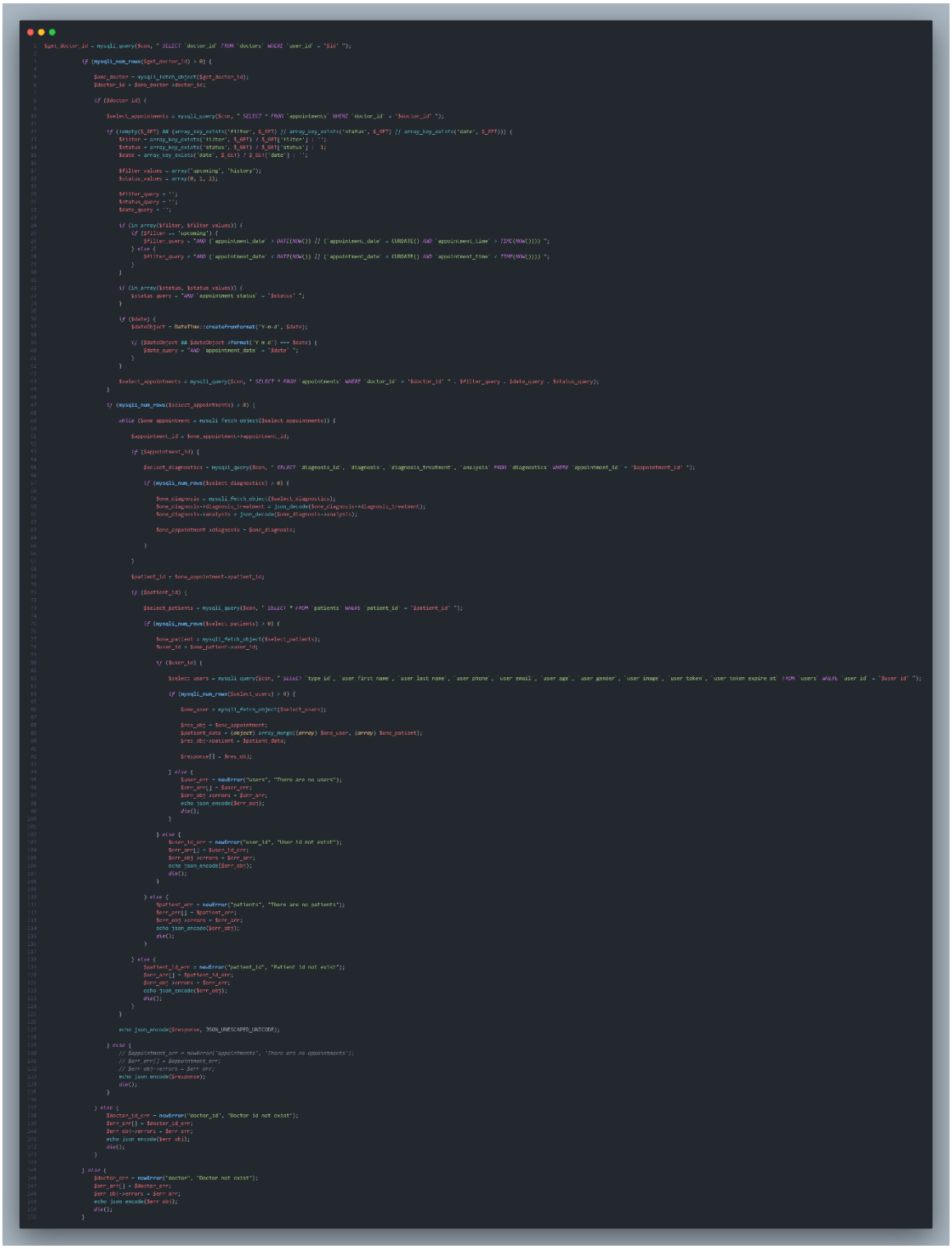


Figure 28 send otp

4.4.2 Appointments:



```
$get_doctor_id = mysqli_query($con, " SELECT `doctor_id` FROM `doctors` WHERE `user_id` = '$id' ");

        if (mysqli_num_rows($get_doctor_id) > 0) {

            $one_doctor = mysqli_fetch_object($get_doctor_id);
            $doctor_id = $one_doctor->doctor_id;

            if ($doctor_id) {

                $select_appointments = mysqli_query($con, " SELECT * FROM `appointments` WHERE `doctor_id` = '$doctor_id' ");

                if (!empty($_GET) && (array_key_exists('filter', $_GET) || array_key_exists('status', $_GET) || array_key_exists('date', $_GET))) {
                    $filter = array_key_exists('filter', $_GET) ? $_GET['filter'] : '';
                    $status = array_key_exists('status', $_GET) ? $_GET['status'] : 1;
                    $date = array_key_exists('date', $_GET) ? $_GET['date'] : '';

                    $filter_values = array('upcoming', 'history');
                    $status_values = array(0, 1, 2);

                    $filter_query = '';
                    $status_query = '';
                    $date_query = '';

                    if (in_array($filter, $filter_values)) {
                        if ($filter == 'upcoming') {
                            $filter_query = "AND (`appointment_date` > DATE(NOW()) || (`appointment_date` = CURDATE() AND `appointment_time` > TIME(NOW())))";
                        } else {
                            $filter_query = "AND (`appointment_date` < DATE(NOW()) || (`appointment_date` = CURDATE() AND `appointment_time` < TIME(NOW())))";
                        }
                    }

                    if (in_array($status, $status_values)) {
                        $status_query = "AND `appointment_status` = '$status' ";
                    }

                    if ($date) {
                        $dateObject = DateTime::createFromFormat('Y-m-d', $date);

                        if ($dateObject && $dateObject->format('Y-m-d') === $date) {
                            $date_query = "AND `appointment_date` = '$date' ";
                        }
                    }

                    $select_appointments = mysqli_query($con, " SELECT * FROM `appointments` WHERE `doctor_id` = '$doctor_id' " . $filter_query . $date_query . $status_query);
                }

                if (mysqli_num_rows($select_appointments) > 0) {

                    while ($one_appointment = mysqli_fetch_object($select_appointments)) {

                        $appointment_id = $one_appointment->appointment_id;

                        if ($appointment_id) {

                            $select_diagnostics = mysqli_query($con, " SELECT `diagnosis_id`, `diagnosis`, `diagnosis_treatment`, `analysis` FROM `diagnostics` WHERE `appointment_id` = '$appointment_id' ");

                            if (mysqli_num_rows($select_diagnostics) > 0) {

                                $one_diagnosis = mysqli_fetch_object($select_diagnostics);
                                $one_diagnosis->diagnosis_treatment = json_decode($one_diagnosis->diagnosis_treatment);
                                $one_diagnosis->analysis = json_decode($one_diagnosis->analysis);

                                $one_appointment->diagnosis = $one_diagnosis;

                            }

                        }

                        $patient_id = $one_appointment->patient_id;

                        if ($patient_id) {

                            $select_patients = mysqli_query($con, " SELECT * FROM `patients` WHERE `patient_id` = '$patient_id' ");

                            if (mysqli_num_rows($select_patients) > 0) {

                                $one_patient = mysqli_fetch_object($select_patients);
                                $user_id = $one_patient->user_id;

                                if ($user_id) {

                                    $select_users = mysqli_query($con, " SELECT `type_id`, `user_first_name`, `user_last_name`, `user_phone`, `user_email`, `user_age`, `user_gender`, `user_image`, `user_token`, `user_token_expire_at` FROM `users` WHERE `user_id` = '$user_id' ");

                                    if (mysqli_num_rows($select_users) > 0) {

                                        $one_user = mysqli_fetch_object($select_users);

                                        $res_obj = $one_appointment;
                                        $patient_data = (object) array_merge((array) $one_user, (array) $one_patient);
                                        $res_obj->patient = $patient_data;

                                        $response[] = $res_obj;

                                    } else {
                                        $user_err = newError("users", "There are no users");
                                        $err_arr[] = $user_err;
                                        $err_obj->errors = $err_arr;
                                        echo json_encode($err_obj);
                                        die();
                                    }

                                } else {
                                    $user_id_err = newError("user_id", "User id not exist");
                                    $err_arr[] = $user_id_err;
                                    $err_obj->errors = $err_arr;
                                    echo json_encode($err_obj);
                                    die();
                                }

                            } else {
                                $patient_err = newError("patients", "There are no patients");
                                $err_arr[] = $patient_err;
                                $err_obj->errors = $err_arr;
                                echo json_encode($err_obj);
                                die();
                            }

                        } else {
                            $patient_id_err = newError("patient_id", "Patient id not exist");
                            $err_arr[] = $patient_id_err;
                            $err_obj->errors = $err_arr;
                            echo json_encode($err_obj);
                            die();
                        }
                    }

                    echo json_encode($response, JSON_UNESCAPED_UNICODE);

                } else {
                    // $appointment_err = newError("appointments", "There are no appointments");
                    // $err_arr[] = $appointment_err;
                    // $err_obj->errors = $err_arr;
                    echo json_encode($response);
                    die();
                }

            } else {
                $doctor_id_err = newError("doctor_id", "Doctor id not exist");
                $err_arr[] = $doctor_id_err;
                $err_obj->errors = $err_arr;
                echo json_encode($err_obj);
                die();
            }

        } else {
            $doctor_err = newError("doctor", "Doctor not exist");
            $err_arr[] = $doctor_err;
            $err_obj->errors = $err_arr;
            echo json_encode($err_obj);
            die();
        }
```

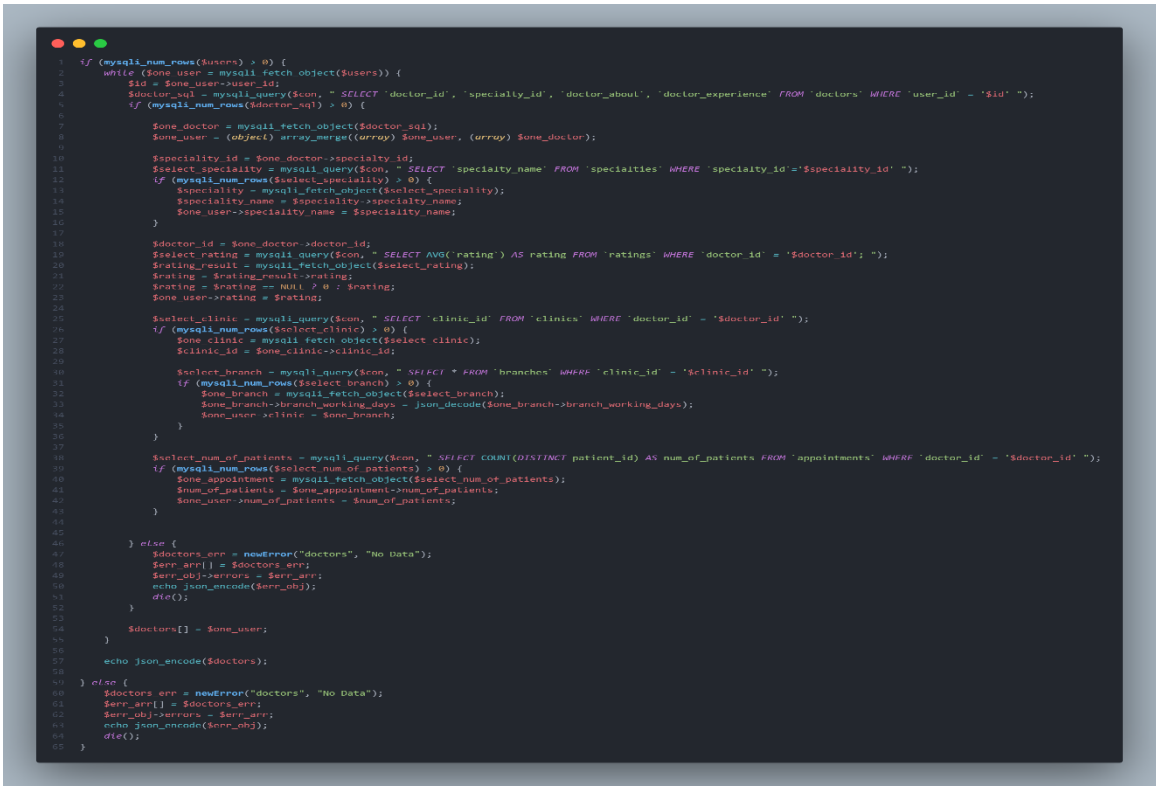
4.4.2 Add, book and update appointment:

Figure 34 Add appointment

Figure 33 Book appointment

Figure 32 update appointment

4.4.3 Doctors, patients and rate doctors:

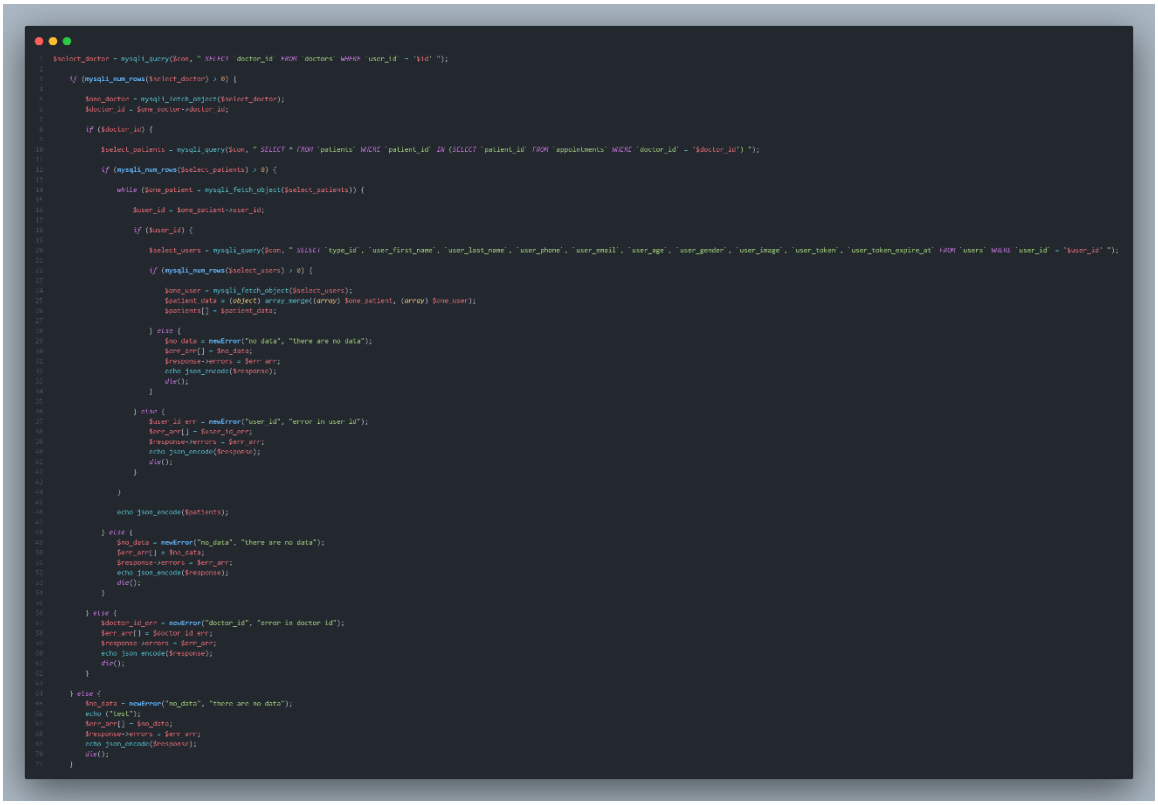


```php
if (mysqli_num_rows($users) > 0) {
    while ($one_user = mysqli_fetch_object($users)) {
        $id = $one_user->user_id;
        $doctor_sql = mysqli_query($con, " SELECT `doctor_id`, `specialty_id`, `doctor_about`, `doctor_experience` FROM `doctors` WHERE `user_id` = '$id' ");
        if (mysqli_num_rows($doctor_sql) > 0) {

            $one_doctor = mysqli_fetch_object($doctor_sql);
            $one_user = (object) array_merge((array) $one_user, (array) $one_doctor);

            $speciality_id = $one_doctor->specialty_id;
            $select_speciality = mysqli_query($con, " SELECT `specialty_name` FROM `specialties` WHERE `specialty_id`='$speciality_id' ");
            if (mysqli_num_rows($select_speciality) > 0) {
                $speciality = mysqli_fetch_object($select_speciality);
                $speciality_name = $speciality->specialty_name;
                $one_user->speciality_name = $speciality_name;
            }

            $doctor_id = $one_doctor->doctor_id;
            $select_rating = mysqli_query($con, " SELECT AVG(`rating`) AS rating FROM `ratings` WHERE `doctor_id` = '$doctor_id'; ");
            $rating_result = mysqli_fetch_object($select_rating);
            $rating = $rating_result->rating;
            $rating = $rating == NULL ? 0 : $rating;
            $one_user->rating = $rating;

            $select_clinic = mysqli_query($con, " SELECT `clinic_id` FROM `clinics` WHERE `doctor_id` = '$doctor_id' ");
            if (mysqli_num_rows($select_clinic) > 0) {
                $one_clinic = mysqli_fetch_object($select_clinic);
                $clinic_id = $one_clinic->clinic_id;

                $select_branch = mysqli_query($con, " SELECT * FROM `branches` WHERE `clinic_id` = '$clinic_id' ");
                if (mysqli_num_rows($select_branch) > 0) {
                    $one_branch = mysqli_fetch_object($select_branch);
                    $one_branch->branch_working_days = json_decode($one_branch->branch_working_days);
                    $one_user->clinic = $one_branch;
                }
            }

            $select_num_of_patients = mysqli_query($con, " SELECT COUNT(DISTINCT patient_id) AS num_of_patients FROM `appointments` WHERE `doctor_id` = '$doctor_id' ");
            if (mysqli_num_rows($select_num_of_patients) > 0) {
                $one_appointment = mysqli_fetch_object($select_num_of_patients);
                $num_of_patients = $one_appointment->num_of_patients;
                $one_user->num_of_patients = $num_of_patients;
            }


        } else {
            $doctors_err = newError("doctors", "No Data");
            $err_arr[] = $doctors_err;
            $err_obj->errors = $err_arr;
            echo json_encode($err_obj);
            die();
        }

        $doctors[] = $one_user;
    }

    echo json_encode($doctors);

} else {
    $doctors_err = newError("doctors", "No Data");
    $err_arr[] = $doctors_err;
    $err_obj->errors = $err_arr;
    echo json_encode($err_obj);
    die();
}
```

Figure 35 doctors

```php
if (!is_numeric($rating) || $rating <= 0) {

        $rating_err = new stdClass();
        $rating_err->rating = "Invalid rating";

        array_push($err_arr, $rating_err);
        $err_obj->errors = $err_arr;
        // echo json_encode($err_obj);

    }

    if (!is_numeric($doctor_id) || $doctor_id <= 0) {

        $doctor_id_err = new stdClass();
        $doctor_id_err->doctor_id = "Invalid doctor_id";

        array_push($err_arr, $doctor_id_err);
        $err_obj->errors = $err_arr;
        // echo json_encode($err_obj);

    }

    if (!is_numeric($patient_id) || $patient_id <= 0) {

        $patient_id_err = new stdClass();
        $patient_id_err->patient_id = "Invalid patient_id";

        array_push($err_arr, $patient_id_err);
        $err_obj->errors = $err_arr;
        // echo json_encode($err_obj);

    }

    $select_rating = mysqli_query($con, " SELECT `doctor_id`, `patient_id` FROM `ratings` WHERE `doctor_id` = '$doctor_id' AND `patient_id` = '$patient_id' ");
    if (mysqli_num_rows($select_rating) > 0) {
        $same_id_err = new stdClass();
        $same_id_err->same_id = "patient cannot rate the same doctor two times!";

        array_push($err_arr, $same_id_err);
        $err_obj->errors = $err_arr;
    }

    if (!empty($err_arr)) {
        http_response_code(400);
        echo json_encode($err_obj);
        die();
    } else {

        // added users table
        $insert_rating = mysqli_query($con, "INSERT INTO `ratings`( `rating`, `doctor_id`, `patient_id`, `rating_review`) VALUES ('$rating','$doctor_id','$patient_id', '$rating_review')");

        if (mysqli_affected_rows($con) > 0) {
            echo json_encode("successfully rate doctor");
        } else {
            $sql_err = new stdClass();
            $sql_err->rate_doctor = "Faild rate doctor";
            $err_obj->errors = $sql_err;
            array_push($err_arr, $err_obj);
            http_response_code(500);
            echo json_encode($err_obj);
            die();

        }
    }
```

Figure 36 rate doctors



```php
$select_doctor = mysqli_query($con, " SELECT `doctor_id` FROM `doctors` WHERE `user_id` = '$id' ");

    if (mysqli_num_rows($select_doctor) > 0) {

        $one_doctor = mysqli_fetch_object($select_doctor);
        $doctor_id = $one_doctor->doctor_id;

        if ($doctor_id) {

            $select_patients = mysqli_query($con, " SELECT * FROM `patients` WHERE `patient_id` IN (SELECT `patient_id` FROM `appointments` WHERE `doctor_id` = '$doctor_id') ");

            if (mysqli_num_rows($select_patients) > 0) {

                while ($one_patient = mysqli_fetch_object($select_patients)) {

                    $user_id = $one_patient->user_id;

                    if ($user_id) {

                        $select_users = mysqli_query($con, " SELECT `type_id`, `user_first_name`, `user_last_name`, `user_phone`, `user_email`, `user_age`, `user_gender`, `user_image`, `user_token`, `user_token_expire_at` FROM `users` WHERE `user_id` = '$user_id' ");

                        if (mysqli_num_rows($select_users) > 0) {

                            $one_user = mysqli_fetch_object($select_users);
                            $patient_data = (object) array_merge((array) $one_patient, (array) $one_user);
                            $patients[] = $patient_data;

                        } else {
                            $no_data = newError("no data", "there are no data");
                            $err_arr[] = $no_data;
                            $response->errors = $err_arr;
                            echo json_encode($response);
                            die();
                        }

                    } else {
                        $user_id_err = newError("user_id", "error in user id");
                        $err_arr[] = $user_id_err;
                        $response->errors = $err_arr;
                        echo json_encode($response);
                        die();
                    }

                }

                echo json_encode($patients);

            } else {
                $no_data = newError("no_data", "there are no data");
                $err_arr[] = $no_data;
                $response->errors = $err_arr;
                echo json_encode($response);
                die();
            }

        } else {
            $doctor_id_err = newError("doctor_id", "error in doctor id");
            $err_arr[] = $doctor_id_err;
            $response->errors = $err_arr;
            echo json_encode($response);
            die();
        }

    } else {
        $no_data = newError("no_data", "there are no data");
        echo ("test");
        $err_arr[] = $no_data;
        $response->errors = $err_arr;
        echo json_encode($response);
        die();
    }
```

Figure 37 Patients

Chapter 5: Conclusions &Future Scope

5.1 Discussion and Conclusion:

Doctor and patient appointment system is a very exciting topic to work.

After going through the work, we faced many challenging tasks. Day by day healthcare system become an important part of our society, so we have decided to build this system.

We researched so many system that showed us the direction how to develop our system. We interact with the people that what type of problem they facing. They were very happy to take this system as it is give them some relief in modern age

Despite everything we achieved, we faced many challenges to finish this project.

In conclusion we implemented all the functionalities stated in the requirement, all were tested and it showed that they are all working properly with the exception of the mail functionality which works perfectly fine on server.

Also we implemented the system requirement quality like responsiveness so on different devices the system adapts to suit it.

5.2 Scope of Further Development:

Online system is always a changeable system. It develops day by day, getting better and better to easier for peoples.

This could be a revolutionary mobile application that may help relationship between doctor and patient.

We believe we can make this system more advanced in future, Advance features and User interface will be updated in future. Our system is already user friendly but we will try to make this system more user friendly in future.

References

1- React-native

https://www.udemy.com/course/react-native-m

https://www.udemy.com/course/react-native-the-practical-guide

2- Redux and Redux toolkit

https://www.youtube.com/playlist?list=PLejc1JbD4ZFREfrBoSl8tjAPZOY6HNqZv

https://www.youtube.com/playlist?list=PLejc1JbD4ZFQFvS469VXyCPO_py_kvVD5

3-modern javascript

https://www.youtube.com/playlist?list=PLejc1JbD4ZFRUB2SGwp5nSUKe44atnTSM

4-system analysis diagrams

https://www.youtube.com/watch?v=nOWDVKyTAus&list=PLYkGENsF6IN-j2P8dOUAy5NgNlqFrheU5

5-ui/ux

https://www.udemy.com/course/learn-adobe-xd-for-ui-ux-/design-with-a-design-project

https://www.udemy.com/course/the-complete-mobile-app-/design-course-uiux

https://www.udemy.com/course/learn-adobe-xd-ui-ux-design-from-zero-to-hero/learn/lecture/30794618?start=18#overview

https://youtube.com/playlist?list=PLjzhiGLyugKzxD2WKrI0riNZ9E6HoZYkH

6-Mysql

https://www.w3schools.com/sql/default.asp

7-PHP

https://www.w3schools.com/php/default.asp

https://www.php.net/manual/en