Doctor Appointment Booking Development with MERN Project Documentation

1. Introduction

- **Project Title:** Doctor Appointment Booking Web App
- **Team Members:** Divya Priya S(Team leader), Adlin Gracia (Team Member), Christy Titus(Team Member), Helan K (Team Member).

2. Project Overview

- **Purpose:**

  The Doctor Appointment Web Application using (Mongo DB, Node.js, React.js, Express.js) aims to provide a seamless platform for users to search,
  compare, and book flights. It ensures a user-friendly interface, secure payment processing, and personalized flight recommendations. The system is designed to handle scalability, offer real-time updates, and simplify the booking process with

- **Features (For Patients):**

  Doctor Search and Filtering:

  Search for doctors by name, specialty, or location and filters you the best doctors nearby who are relatable and available.

  Appointment Booking:

  Easy selection of date, time, and doctors.

  Patient Dashboard:

  View, reschedule, or cancel appointments.

  Medical Records Upload:

  Securely upload prescriptions or reports.

  Payment Gateway:

  Easy booking process with secure online payment options.
  Support for multiple payment methods (credit/debit cards, wallets, etc.).

  Notifications:

  Automated reminders for appointments.

  Telemedicine:

  Option for virtual consultations via video call.

(For Doctors/ Healthcare providers):

Availability Management:

Manage schedules and time slots.

Appointment Tracking:

View upcoming and past appointments.

Patient records:

Access detailed patient history and records.

General Features:

Admin Panel:

Manage users, doctors, and appointments.

Role-Based Access:

Different interfaces for patients and providers.

Multi language support:

Cater to diverse demographics.

User authentication:

Secure login for patients and doctors.

3. Architecture

1. Front-End (Client Layer)

- **Technology: React.js**
- **Responsibilities:**
  - o Provides an interactive and responsive user interface.
  - o Handles user interactions such as searching doctors, viewing details, appointment booking, and payments.
  - o Communicates with the backend API for data retrieval and updates.
  - o Implements state management using tools like React Context API or Redux.

2. Back-End (Application Layer)

- **Technology**: Node.js with Express.js
- **Responsibilities**:
  - o Exposes RESTful APIs for communication between the front-end and the database.
  - o Implements business logic, such as search filters, booking workflows, and payment validation.

- Handles user authentication and authorization using JSON Web
- Tokens (JWT). Manages flight scheduling, pricing algorithms, and cancellation policies.

- **Database:**

  - **Technology: MongoDB**
  - **Responsibilities:**
    - Stores and manages application data, including:
      - Flight schedules and details. User
      - information and booking history.
      - Payment records and transaction logs.
    - Implements indexing for fast search and retrieval of flight data.

4. Setup Instructions

Prerequisites

1. Install Node.js (LTS version recommended). 2. Install MongoDB and ensure it is running locally or have access to a cloud database (e.g., MongoDB Atlas).
3. Install Git.
4. Ensure you have a modern web browser (e.g., Google Chrome).
5. (Optional) Install a package manager like Yarn if preferred over npm.

1. Clone the Repository

```
bash
Copy code
 git clone
```

2. Install Backend Dependencies

```
bash
Copy code
npm install express joi jsonwebtoken moment mongoose morgan
nodemon zxcvbn dotenv colors bcryptjs
```

3. Install Frontend Dependencies

```
bash
Copy code
cd client
npm i react-router-dom react-redux axios antd @reduxjs/toolkit
react-bootstrap moment
```

4. Configure Environment Variables

Backend:

DB_URL = mongodb+srv://<user>:

<pass>url.mongodb.net/database

JWT_SECRET = A_Secret_Value

PORT = 4000

Frontend:

- In the frontend folder, create a .env file with:
  REACT_APP_API_URL=http://localhost:5000

5 . Access the App

- Open your web browser and navigate to:
  http://localhost:3000 (Front-end)
  http://localhost:5000 (Back-end API)

5. Folder Structure

1. Client (React Frontend)

The React frontend is organized as follows:

```
client/
├── public/
│   ├── index.html
│   ├── favicon.ico
│   ├── manifest.json
│   └── assets/
├── src/
│   ├── components/
│   ├── pages/
│   ├── context/
│   ├── hooks/
│   ├── services/
│   ├── styles/
│   ├── App.js
│   ├── index.js
│   └── .env
├── package.json
└── README.md
```

- The components/ folder contains reusable UI elements like buttons, forms, or headers.
- The pages/ folder includes specific pages, such as HomePage.js, BookingPage.js, and Paymentgate.js.
- API calls and utilities are abstracted into the utils/ folder.
- The context/ folder manages global states, such as the Flight booking website or authentication state.

2. Server:

The Node.js backend is structured as follows:

```
server/
├── src/
│   ├── config/
│   ├── controllers/
│   ├── middlewares/
│   ├── models/
│   ├── routes/
│   ├── utils/
│   ├── server.js
│   ├── .env
├── package.json
├── README.md
└── tests/
```

6. Running the Application

Start MongoDB:

Start MongoDB locally or ensure your MongoDB Atlas instance is running.

Start Backend:

From the backend folder

npm start

Start Frontend:

From the frontend folder

npm start

7. API Documentation

1. User Authentication

1.1 Register a User

Method: POST

Endpoint: /api/auth/register
Description: Registers a new user.

Request Body:

json

Copy code

```
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "password": "securepassword"
}
```

Response:

json

Copy code

```
{
  "success": true,
  "message": "User registered successfully."
}
```

POST /api/auth/login

- Description: Logs in an existing user and provides a JWT token.
- Request Method: POST
- Request Body:

json

Copy code

```
{
  "email": "john.doe@example.com",
  "password": "password123"
}
```

  - email: (string) The user's email.
  - password: (string) The user's password.

- Response:

200 OK:

json

Copy code

```
{
  "token": "your_jwt_token_here"
}
```

400 Bad Request: Invalid credentials.

json

Copy code

```
{
  "error": "Invalid email or password."
}
```

1.2 Login a User

Method: POST
Endpoint: /api/auth/login
Description: Authenticates a user and returns a JWT token.

Request Body:

```
{
  "email": "john.doe@example.com",
```

"password": "securepassword"

}

Response:

{

"success": true,

"token": "eyJhbGciOiJIUzI1NiIsInR5..."

}

2. Doctors

2.1 Search Doctors

Method: GET
Endpoint: GET /api/doctors?
specialty=cardiology&location=newyork

Query Parameters:

- specialty - The specialty of the doctor (e.g., cardiology).
- location - The location of the doctor (e.g., New York).
- name (optional) - The name of the doctor.

Example Request:

http

Copy code

GET /api/flights/search?from=JFK&to=LAX&departureDate=2024-11-20&passengers=2

Response:

json

Copy code

[

{

"success": true,

"message": "Doctors fetched successfully",

"data": [

{

"id": "d1",
"name": "Dr. Sarah Johnson",
"specialty": "Cardiology",
"location": "New York",
"experience": 10,
"rating": 4.8,
"availability": [
{
"date": "2024-11-17",
"timeSlots": ["10:00 AM", "12:00 PM", "3:00 PM"]
},
{
"date": "2024-11-18",
"timeSlots": ["11:00 AM", "1:00 PM", "4:00 PM"]
}
]
},
{
"id": "d2",
"name": "Dr. Michael Brown",
"specialty": "Cardiology",
"location": "New York",
"experience": 15,
"rating": 4.6,
"availability": [
{
"date": "2024-11-17",
"timeSlots": ["9:00 AM", "2:00 PM", "5:00 PM"]
},
{
"date": "2024-11-19",
"timeSlots": ["10:00 AM", "12:00 PM", "3:0
0 PM"]
}
]
}
]
}

2.2 Get Doctor Details

Method: GET Endpoint: GET /api/doctor/:id

Path Parameter: id - The unique ID of the doctor (e.g., d1).

{
"id": "d1",
"name": "Dr. Sarah Johnson",
"specialty": "Cardiology",
"location": "New York",
"experience": 10,
"rating": 4.8,
"consultationFee": 150,
"contact": {
"email": "sarah.johnson@example.com",
"phone": "+1-234-567-8901"
},
"clinic": {
"name": "Heart Care Clinic",
"address": "123 Main Street, New York, NY",
"timings": "9:00 AM - 6:00 PM"
}

3. Booking

3.1 Book a Appoinment

Method: POST
Endpoint: /api/bookings
Description: Create a booking for doctor's appointment

Request Body:

json

Copy code

{
{
"doctorId": "d1",
"patientId": "p123",
"date": "2024-11-17",
"timeSlot": "10:00 AM",
"symptoms": "Chest pain and shortness of breath",
"paymentStatus": "Pending"
}

Response:
 json
Copy code
{ {
  "success": true,
  "message": "Appointment booked successfully",
  "data": { "appointmentId": "a456",
    "doctor": { "id": "d1","name": "Dr. Sarah Johnson",
     "specialty": "Cardiology"
    },
    "patient": { "id": "p123", "name": "John Doe"
    },

}

} 3.2 Get User Bookings

Method: GET
Endpoint: /api/bookings/user/:userId
Description: Retrieves all bookings made by a specific user.

Request:

{

{

  "userId": "u123",

  "doctorId": "d1",

  "appointmentDate": "2024-11-17",

  "timeSlot": "10:00 AM",

  "symptoms": "Headache and dizziness",

  "contactDetails": {

    "phone": "+1-234-567-8901",

    "email": "user@example.com"

  }

}
Response:

json

Copy code

[

{

"success": true,

"message": "Appointment booked successfully",

"data": { "appointmentId": "a789", "user": {

"id": "u123","name": "John Doe","email" "user@example.com"

},

"doctor": {"id": "d1","name": "Dr. Sarah Johnson","specialty": "Cardiology"

},

"appointmentDate": "2024-11-17",

"timeSlot": "10:00 AM",

"symptoms": "Headache and dizziness",

"contactDetails": {"phone": "+1-234-567-8901", "email": "user@example.com"

},

"status": "Confirmed",

"paymentStatus": "Pending"

}

}

}]

4. Admin

4.1 Add a Doctor

Method: POST
Endpoint: /api/admin/doctors
Description: Adds a new user to the system (admin only).

Response Body: json Copy code {

"success": true,

"message": "User registered successfully",

"data": {

"userId": "u123",

"name": "John Doe",

"email": "johndoe@example.com",

"phone": "+1-234-567-8901",

"role": "doctor"

}}

4.2 Delete a Doctor

Method: DELETE
Endpoint: /api/admin/doctors/:id
Description: Deletes a doctor/user from the system (admin only).

Path Parameter:

- id: doctors ID

Response:

json

Copy code

```
{
  "success": true, "message": " User deleted
  successfully."
}
```

8. Authentication

1. User Registration

New users can create an account by providing their name, email, and a secure password.

The system ensures:

- Validation of user details.
- Prevention of duplicate email registrations.
- Secure storage of passwords using encryption techniques.

2. User Login

- Registered users can log in by providing their email and password. The system authenticates the user and, upon success, generates a **JWT token** that allows the user to access protected features.

3. Authentication Middleware

To secure backend endpoints, middleware is implemented to verify the JWT token included
in the user's request.

- If the token is valid, the user can proceed.
- If the token is invalid or missing, access is denied with an error message.

4. Token Management

The JWT token includes user-specific data such as user ID and role. Tokens have a limited validity period to enhance security. Users must log in again when the token expires.

5. Password Security

Passwords are never stored in plain text. They are encrypted before storage to protect user data, even in the event of a breach. During login, the system compares the encrypted form of the entered password with the stored one.

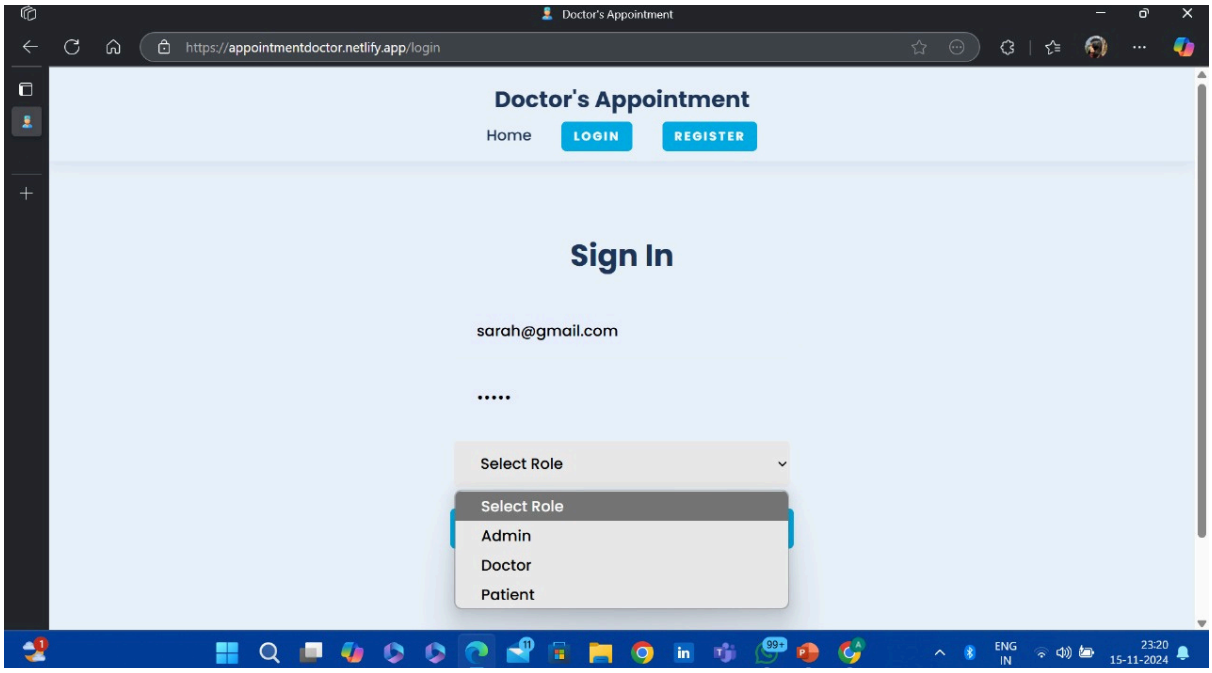
6. Protected Routes

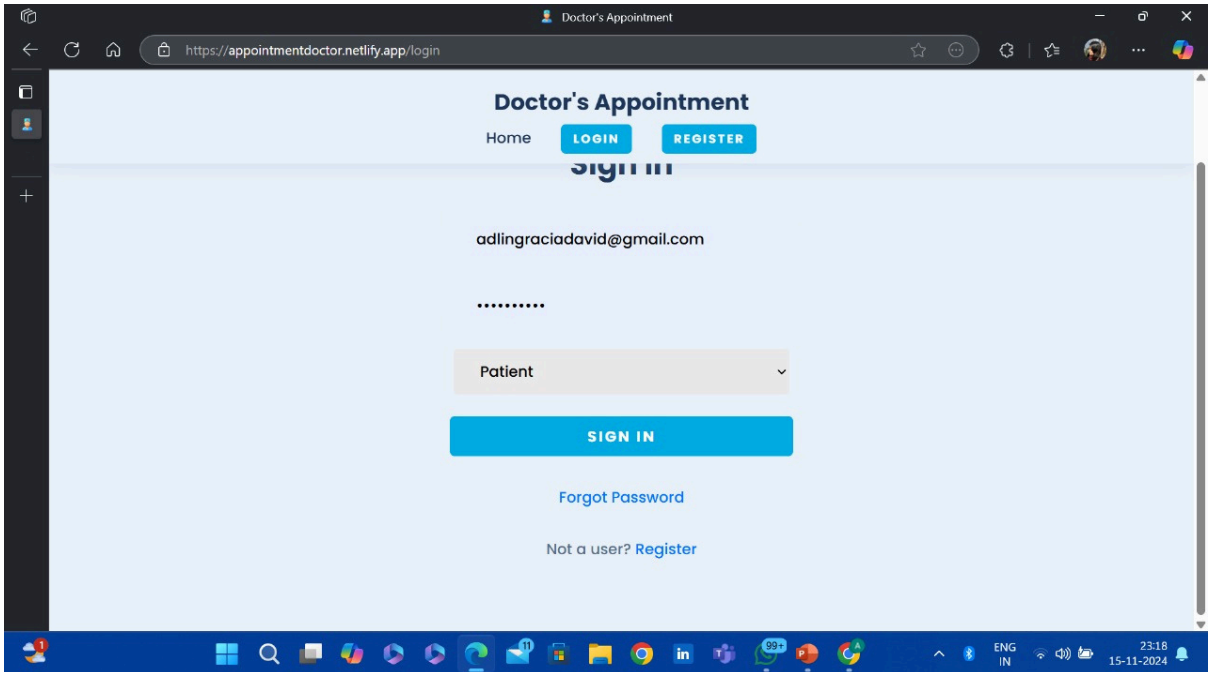
Certain functionalities, like viewing bookings or managing flights, require users to be authenticated. These routes check for the presence of a valid token before granting access.

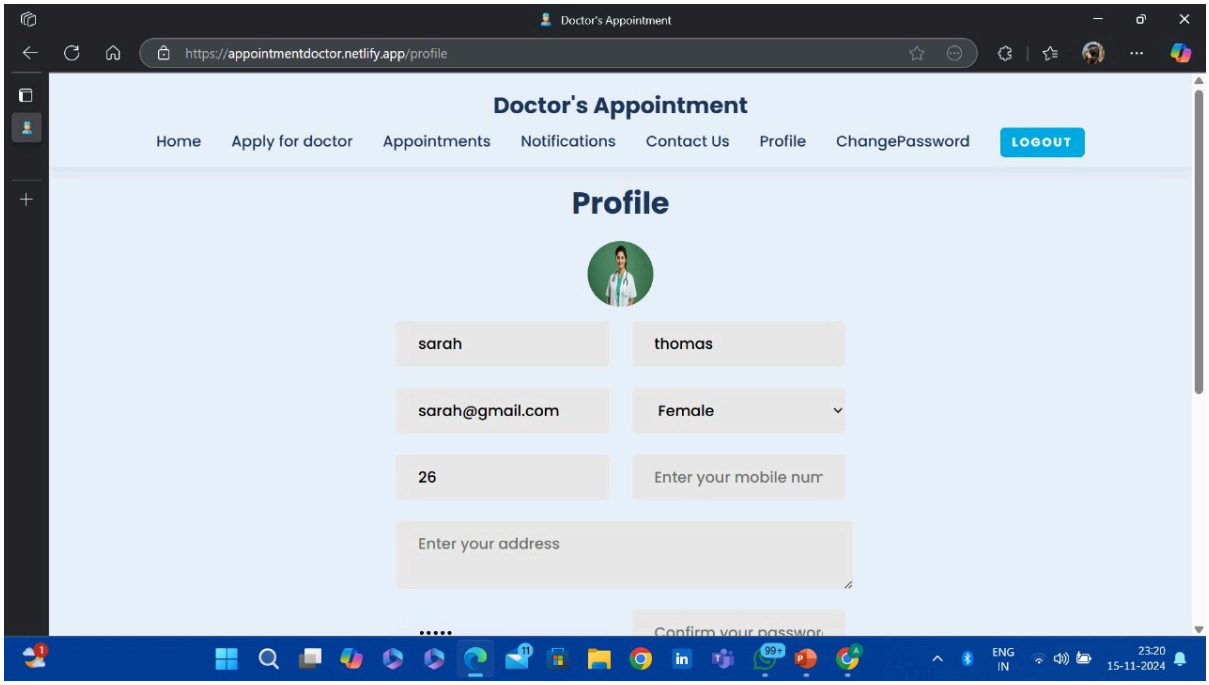
7. Security Practices

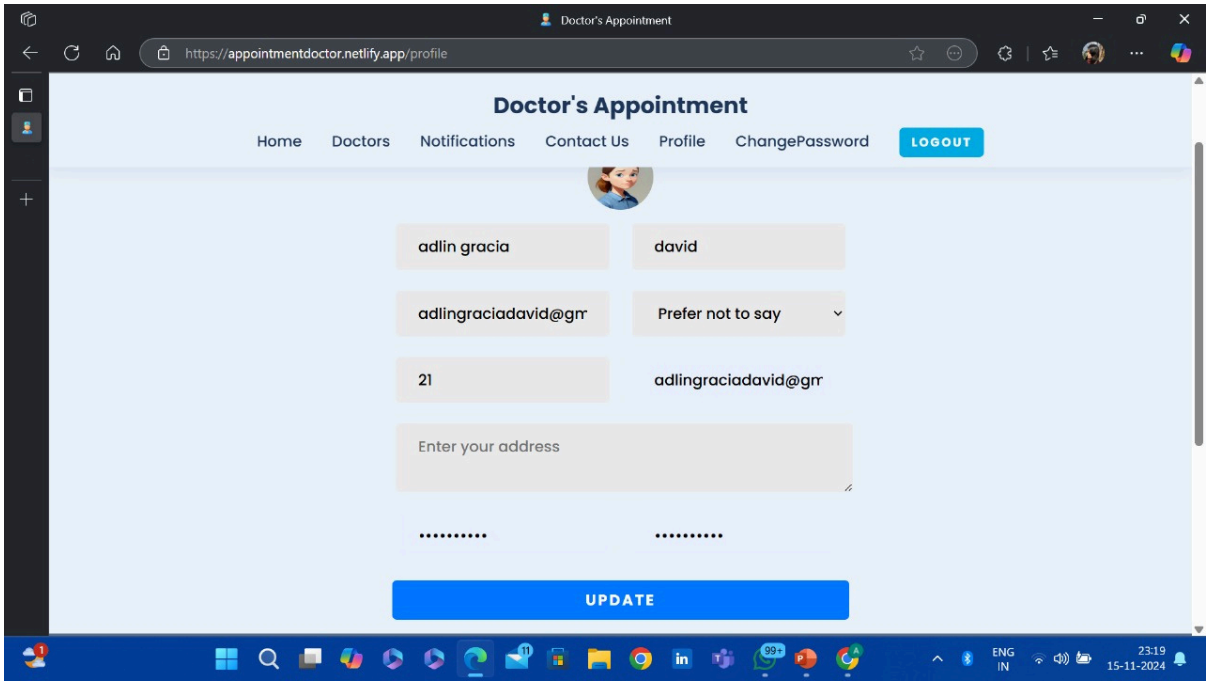
- Tokens are passed securely via headers in API requests.
- HTTPS is used to encrypt data transmission.
- Authentication errors provide generic responses to avoid exposing sensitive information.

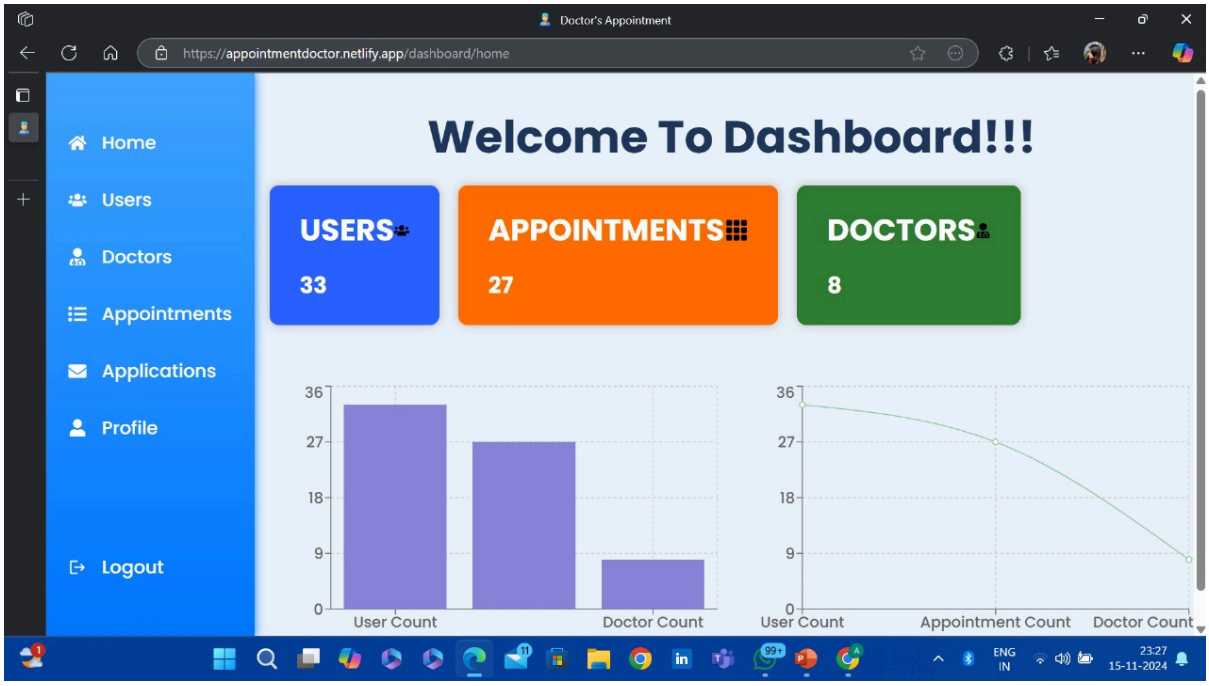
9. User Interface

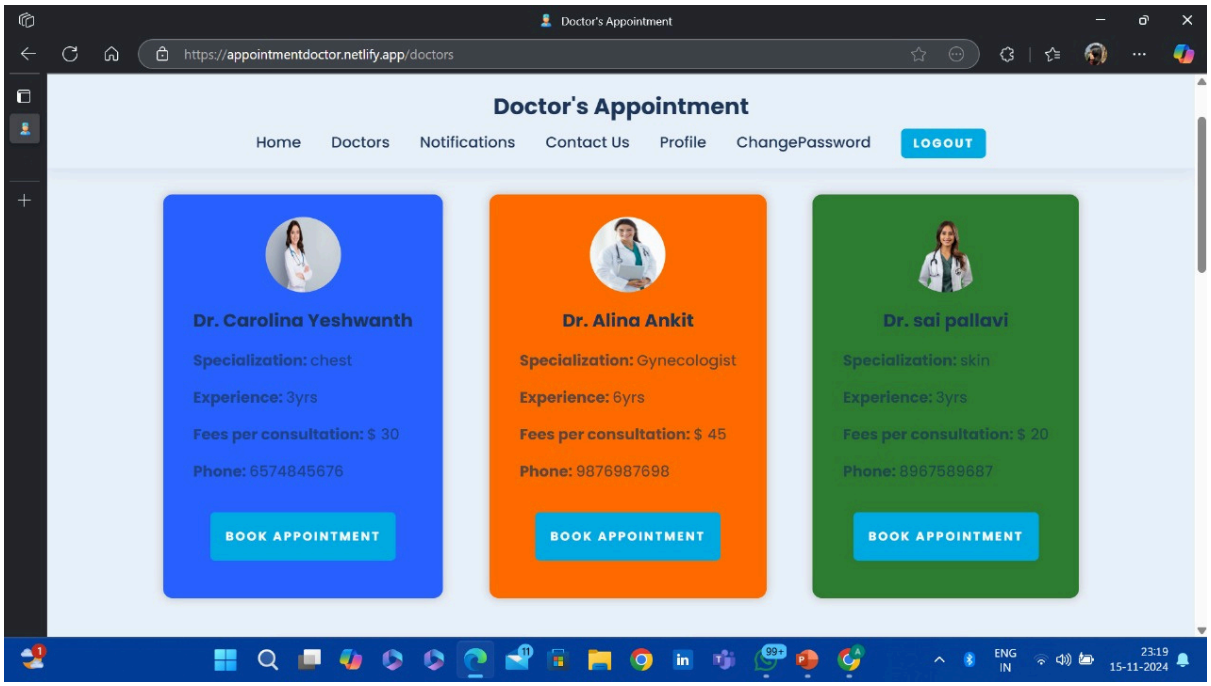


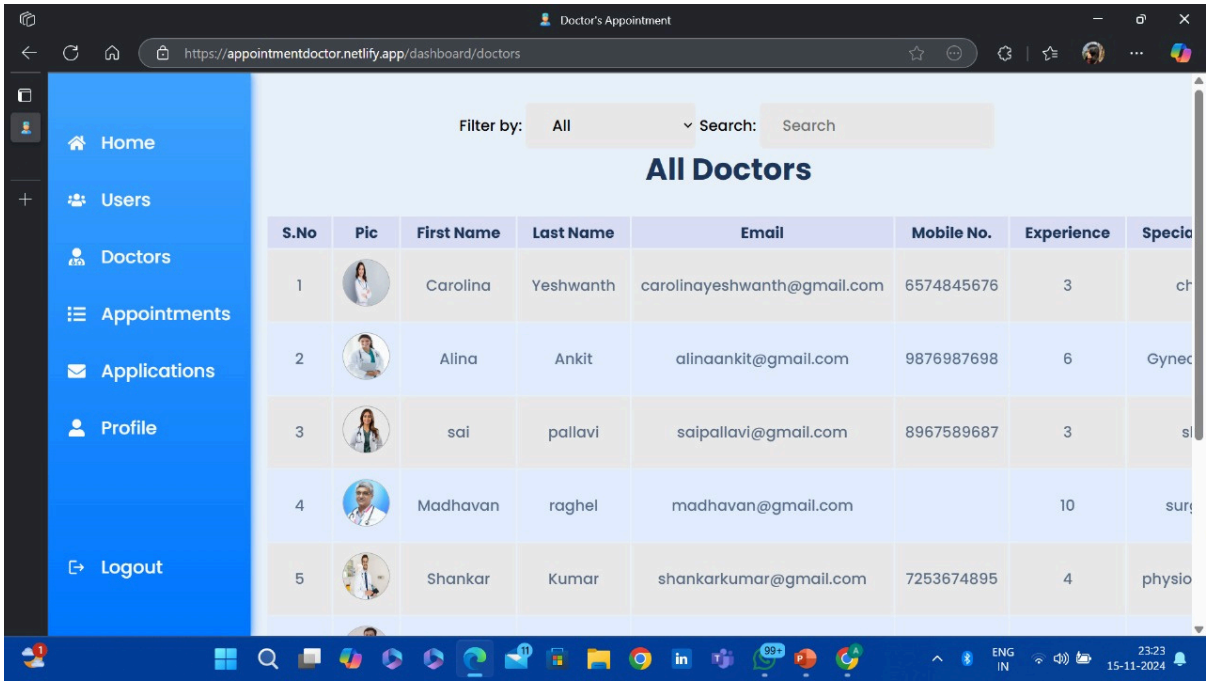

Doctor's Appointment
https://appointmentdoctor.netlify.app/login
Doctor's Appointment
Home
LOGIN
REGISTER
adlingraciadavid@gmail.com
Patient
SIGN IN
Forgot Password
Not a user? Register

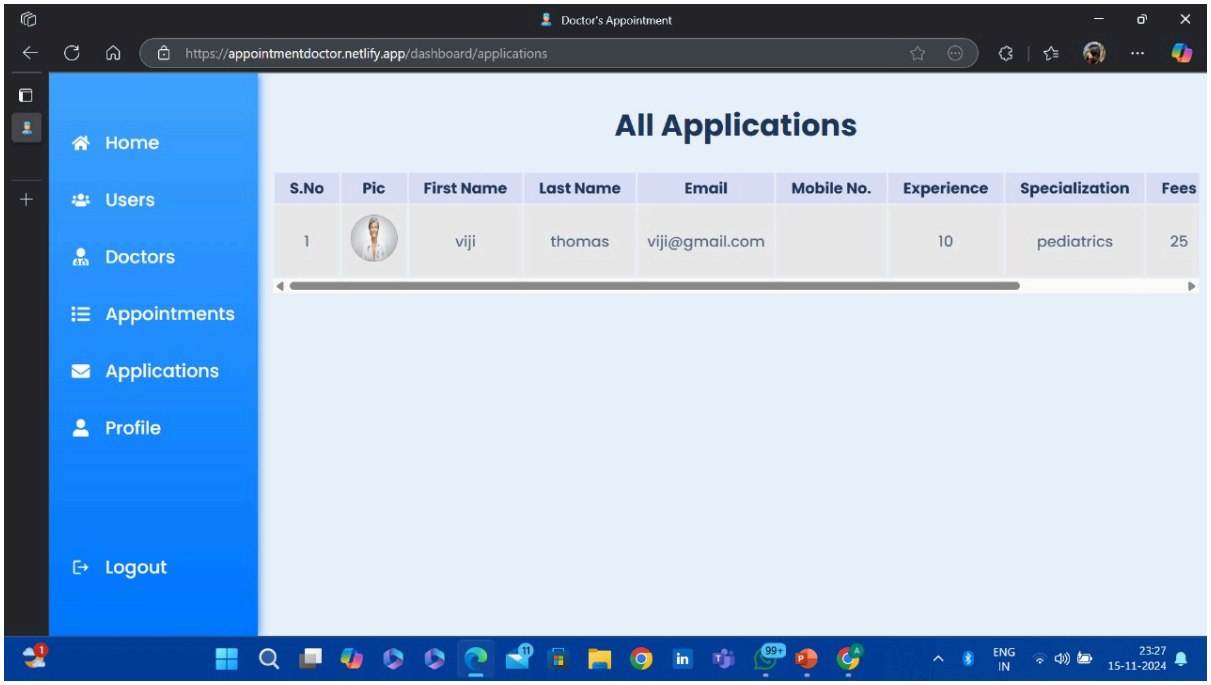

Doctor's Appointment
https://appointmentdoctor.netlify.app/profile
Doctor's Appointment
Home
Apply for doctor
Appointments
Notifications
Contact Us
Profile
ChangePassword
LOGOUT
Profile
sarah
thomas
sarah@gmail.com
Female
26
Enter your mobile num
Enter your address

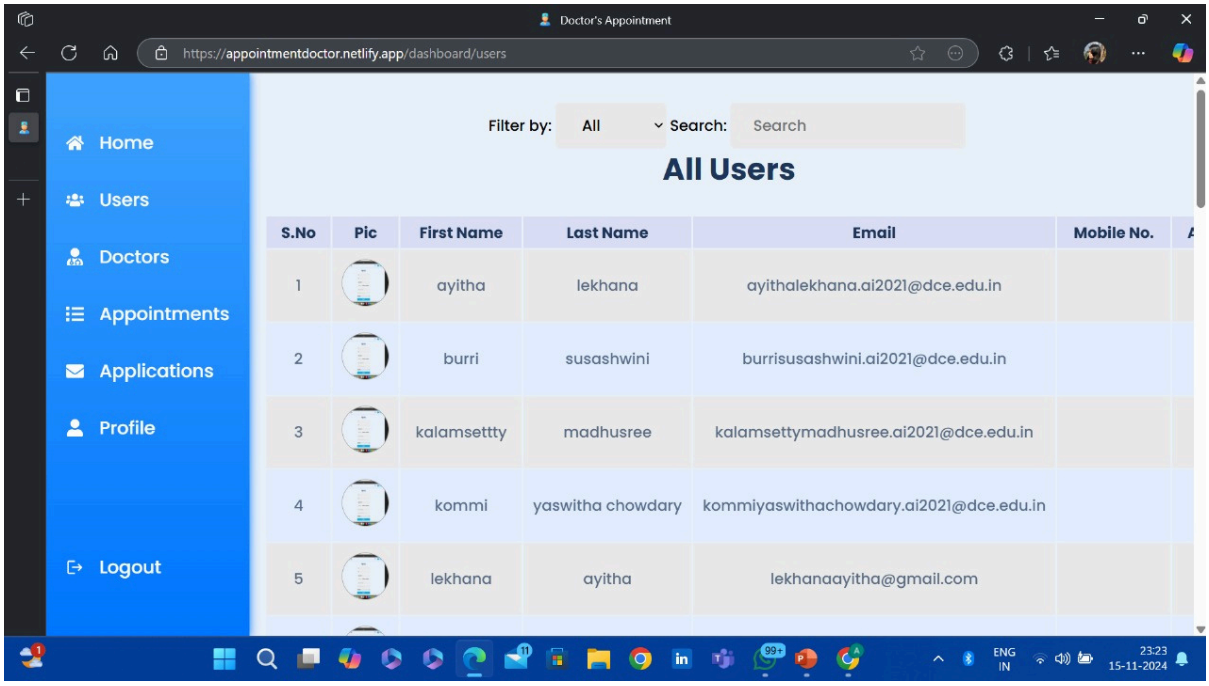

Doctor's Appointment
https://appointmentdoctor.netlify.app/profile
Doctor's Appointment
Home
Doctors
Notifications
Contact Us
Profile
ChangePassword
LOGOUT
adlin gracia
david
adlingraciadavid@gm
Prefer not to say
21
adlingraciadavid@gm
Enter your address
UPDATE


Doctor's Appointment
https://appointmentdoctor.netlify.app/dashboard/home
Home
Users
Doctors
Appointments
Applications
Profile
Logout
Welcome To Dashboard!!!
USERS
33
APPOINTMENTS
27
DOCTORS
8
36
27
18
9
0
User Count
Doctor Count
Appointment Count


Doctor's Appointment
https://appointmentdoctor.netlify.app/doctors
Doctor's Appointment
Home
Doctors
Notifications
Contact Us
Profile
ChangePassword
LOGOUT
Dr. Carolina Yeshwanth
Specialization: chest
Experience: 3yrs
Fees per consultation: $ 30
Phone: 6574845676
BOOK APPOINTMENT
Dr. Alina Ankit
Specialization: Gynecologist
Experience: 6yrs
Fees per consultation: $ 45
Phone: 9876987698
BOOK APPOINTMENT
Dr. sai pallavi
Specialization: skin
Experience: 3yrs
Fees per consultation: $ 20
Phone: 8967589687
BOOK APPOINTMENT
ENG
IN
23:19
15-11-2024


Doctor's Appointment
https://appointmentdoctor.netlify.app/dashboard/doctors
Home
Users
Doctors
Appointments
Applications
Profile
Logout
Filter by:
All
Search:
Search
All Doctors
S.No
Pic
First Name
Last Name
Email
Mobile No.
Experience
Specia
1
Carolina
Yeshwanth
carolinayeshwanth@gmail.com
6574845676
3
2
Alina
Ankit
alinaankit@gmail.com
9876987698
6
3
sai
pallavi
saipallavi@gmail.com
8967589687
3
4
Madhavan
raghel
madhavan@gmail.com
10
5
Shankar
Kumar
shankarkumar@gmail.com
7253674895
4
ENG
IN
23:23
15-11-2024


Doctor's Appointment
https://appointmentdoctor.netlify.app/dashboard/applications
Home
Users
Doctors
Appointments
Applications
Profile
Logout
All Applications
S.No
Pic
First Name
Last Name
Email
Mobile No.
Experience
Specialization
Fees
1
viji
thomas
viji@gmail.com
10
pediatrics
25


Doctor's Appointment
https://appointmentdoctor.netlify.app/dashboard/users
Home
Users
Doctors
Appointments
Applications
Profile
Logout
Filter by:
All
Search:
Search
All Users
S.No
Pic
First Name
Last Name
Email
Mobile No.
1
ayitha
lekhana
ayithalekhana.ai2021@dce.edu.in
2
burri
susashwini
burrisusashwini.ai2021@dce.edu.in
3
kalamsettty
madhusree
kalamsettymadhusree.ai2021@dce.edu.in
4
kommi
yaswitha chowdary
kommiyaswithachowdary.ai2021@dce.edu.in
5
lekhana
ayitha
lekhanaayitha@gmail.com

11. Screenshot or Demo:

https://drive.google.com/file/d/1FZ9XmVKr2UPNkrgzPca7KITvg1bRecT8/view?usp=sharing

12. Future Enhancements

To continually improve the functionality, user experience, and scalability of the Flight Ticket Booking , the following enhancements are planned:

1. Mobile Application

- Develop native mobile applications for **iOS** and **Android** using technologies like **React Native** or **Flutter**.
- Include push notifications for order updates, promotional offers, and reminders for reordering frequently purchased items.

2. Subscription and Loyalty Programs

- Introduce a subscription model for regular customers
- Implement a loyalty rewards system:
  - Earn points for every purchase.
  - Redeem points for discounts or free booking.

3. Voice Search and Assistant Integration

- Add voice search functionality to enable users to find products using voice commands.
- Integrate with virtual assistants like **Google Assistant** or **Alexa** for hands-free appointment booking and tracking.

4. Enhanced Analytics for Admins

- Develop a dashboard with advanced analytics to help administrators track:
  - Sales trends.
  - Inventory levels.
  - Customer behavior and preferences.
- Use predictive analytics to forecast demand and optimize inventory management