**PROJECT REPORT**

**On**

**INSTA CARE**

**COMPUTER SCIENCE AND ENGINEERING**

**B.E. Batch-2021**

**in**

**MAY-2024**

**Submitted By:** Abhinn Singh Bisht (2110990056)

Abhishek Thakur (2110990070)

Aditi Rana (2110990080)

Akshat Chandel (2110990124)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CHITKARA UNIVERSITY**

**PUNJAB**

**CANDIDATE’S DECLARATION**

We, Abhishek Thakur (2110990070) , Aditi Rana (2110990080), Aditya Thakur (2110990101), Akshat Chandel (2110990124)**,** B.E.-2021 of the Chitkara University, Punjab hereby declare that the Project Report entitled **“Insta Care”** is an original work and data provided in the study is authentic to the best of our knowledge. This report has not been submitted to any other Institute for the award of any other course.

| **Sign. of Student 1** | **Sign. of Student 2** | **Sign. of Student 3** |
| --- | --- | --- |
| Name of the Student  Abhishek Thakur | Name of the Student  Aditi Rana | Name of the Student  Abhinn Singh Bisht |
| ID No: 2110990070 | ID No: 2110990080 | ID No: 2110990056 |

| **Sign. of Student 4** |
| --- |
| Name of the Student  Akshat Chandel |
| ID No: 2110990124 |

**1. ABSTRACT/KEYWORDS**

**1.1 Abstract**

Small clinics face numerous challenges in managing appointments and increasing income due to inefficient scheduling, high no-show rates, limited online presence, and lack of marketing resources. This project addresses these issues by developing a web application using the MERN stack, aimed at streamlining appointment scheduling, reducing no-shows, enhancing online visibility, and providing marketing tools. This report details the project's objectives, software and hardware requirements, database design, program structure, implementation, testing, limitations, conclusions, and future scope.

### 1.1 Keywords

Small clinics, appointment scheduling, no-shows, online presence, marketing resources, MERN stack, web application.

**2. INTRODUCTION TO THE PROJECT**

**2.1 Background**

Small clinics often rely on outdated, manual processes to manage appointments, leading to inefficiencies and lost revenue. These clinics also struggle with high no-show rates, limited online presence, and inadequate marketing capabilities, which hinder their ability to grow and compete with larger healthcare providers. This project aims to develop a web application to address these challenges, providing a digital solution to improve appointment management, reduce no-shows, enhance online visibility, and support marketing efforts.

**2.2 Problem Statement**

Small clinics face several challenges that hinder their ability to efficiently manage appointments and increase their income:

* Inefficient Appointment Scheduling: Manual processes are error-prone, time-consuming, and inefficient.
* High Rate of No-Shows: No-shows and last-minute cancellations result in wasted time and lost revenue.
* Limited Online Presence: Lack of online visibility makes it difficult to attract new patients.
* Lack of Marketing Resources: Limited budgets and expertise restrict effective marketing efforts.

**3. SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATION**

**3.1 Methods**

The project utilizes the MERN stack (MongoDB, Express.js, React, Node.js) to develop the web application. This choice provides a robust, scalable, and efficient platform for building modern web applications.

*Why MERN Stack?*

* **MongoDB**: A NoSQL database that stores data in flexible, JSON-like documents. Its schema-less nature allows for easy modifications as the project evolves.
* **Express.js**: A minimal and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications. It simplifies the development of server-side logic.
* **React**: A JavaScript library for building user interfaces, particularly single-page applications where data needs to be dynamically updated. React’s component-based architecture allows for reusable UI components.
* **Node.js**: A JavaScript runtime built on Chrome's V8 JavaScript engine. It is used to build scalable network applications, particularly suited for building fast and scalable server-side applications.

### 3.2 Programming/Working Environment

The working environment for developing this project includes the following tools and technologies:

***Frontend***

* **React.js**: Used for building the user interface. React’s virtual DOM and component-based architecture make it efficient and easy to maintain.

***Backend***

* **Node.js**: Provides the runtime environment for executing server-side code.
* **Express.js**: A web application framework for Node.js that simplifies routing and middleware management.
* **JWT (JSON Web Tokens)**: Used for handling authentication and securing API routes.

***Database***

* **MongoDB**: A NoSQL database used to store user data, appointment information, and clinic details. Its flexibility and scalability make it suitable for this project.

***Development Tools***

* **Visual Studio Code**: A popular code editor with extensions for JavaScript, Node.js, and MongoDB.
* **Git**: Version control system to track changes and collaborate with other developers.
* **npm (Node Package Manager)**: Used to manage project dependencies.

### 3.3 Requirements to Run the Application

#### Hardware Requirements

* **Development Machine**:
  + At least 8GB RAM for smooth development and running of development tools.
  + A modern multi-core CPU to handle development and testing environments efficiently.
  + Adequate storage for project files, database storage, and version control repositories.
* **Web Server**:
  + A server to host the application, with sufficient resources to handle expected traffic.
  + Cloud-based options like AWS, DigitalOcean, or Heroku can be used for scalability.

#### Software Requirements

* **Node.js and npm**:
  + **Node.js**: Required to run the server-side code. Download and install from the official Node.js website.
  + **npm**: Comes with Node.js, used to install and manage project dependencies.
* **MongoDB**:
  + Download and install MongoDB from the official MongoDB website or use a cloud-based service like MongoDB Atlas for easy management and scalability.
* **Web Browser**:
  + Any modern web browser (e.g., Google Chrome, Mozilla Firefox) to access and test the web application.
* **Internet Connection**:
  + Required for installing dependencies, accessing external libraries, and using online services such as cloud databases.

By ensuring these hardware and software requirements, the development and deployment of the web application can be executed efficiently, leading to a robust and scalable solution for small clinics.

**4. DATABASE ANALYZING, DESIGN, AND IMPLEMENTATION**

### Database Design

* **Collections**: Users, Appointments, Clinics, Notifications
* **Key Fields**:
  + Users: userId, name, email, password, role (patient/doctor)
  + Appointments: appointmentId, userId, clinicId, date, time, status
  + Clinics: clinicId, name, location, contactInfo
  + Notifications: notificationId, userId, message, date

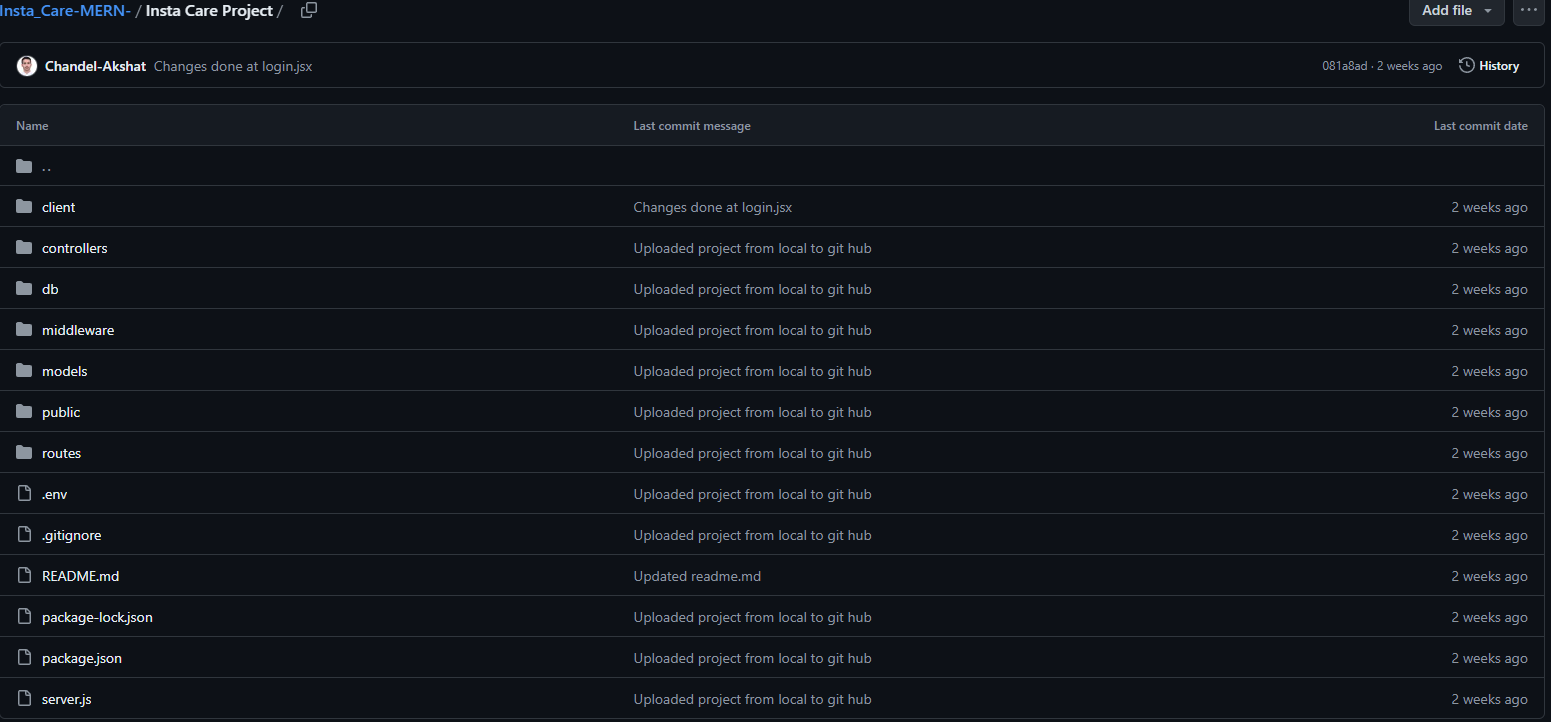
### Implementation

* MongoDB is used to store and manage data related to users, appointments, clinics, and notifications.
* Mongoose is utilized to interact with the MongoDB database in a structured manner.

**5. PROGRAM’S STRUCTURE ANALYZING AND GUI CONSTRUCTING**

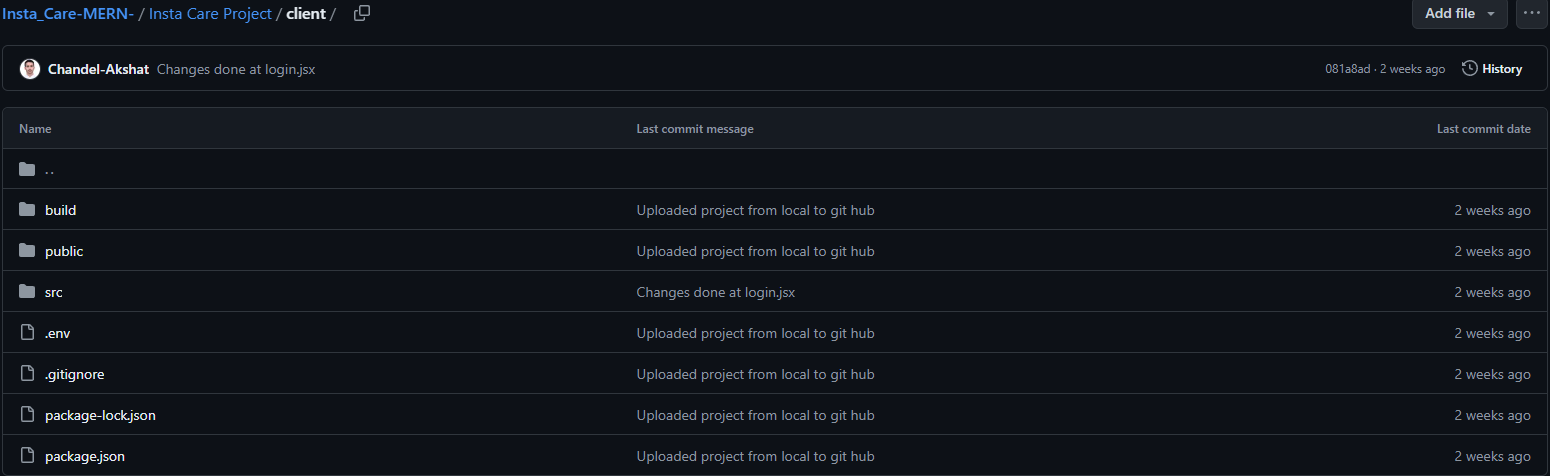
**(PROJECT SNAPSHOTS)**

**Program Structure**

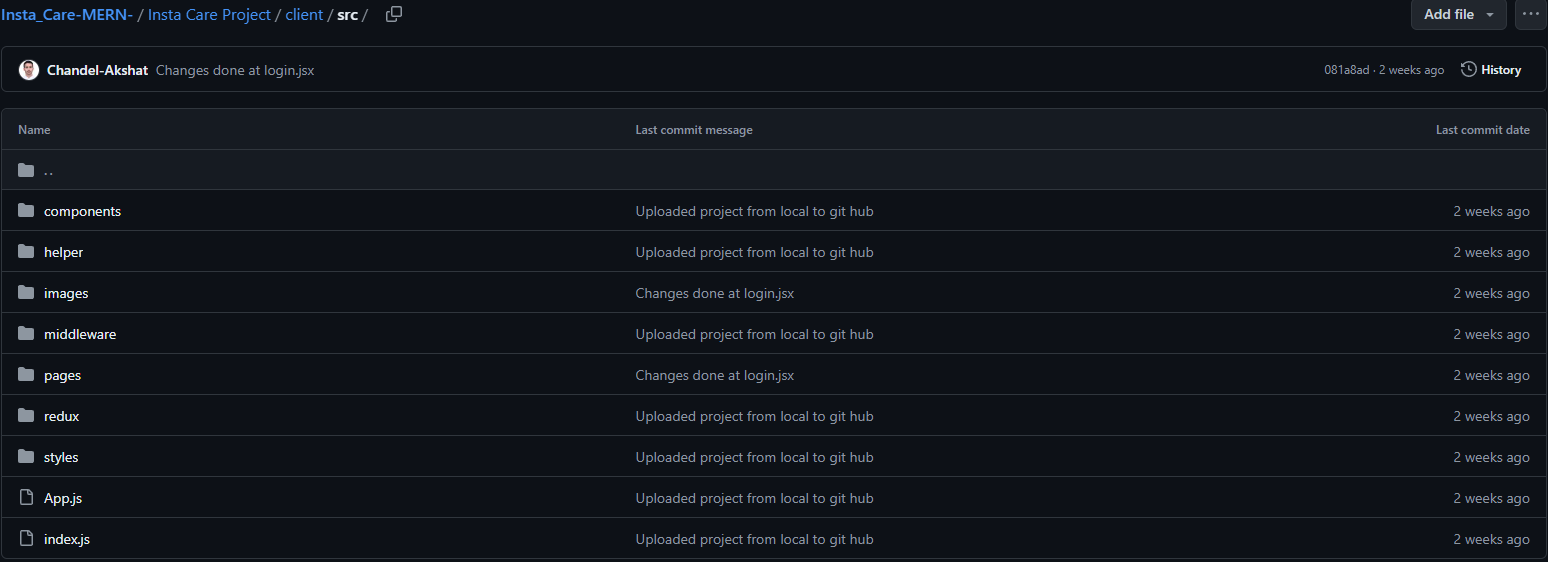
****

* **Frontend**: React components for login, registration, appointment scheduling, clinic management, and notification management.

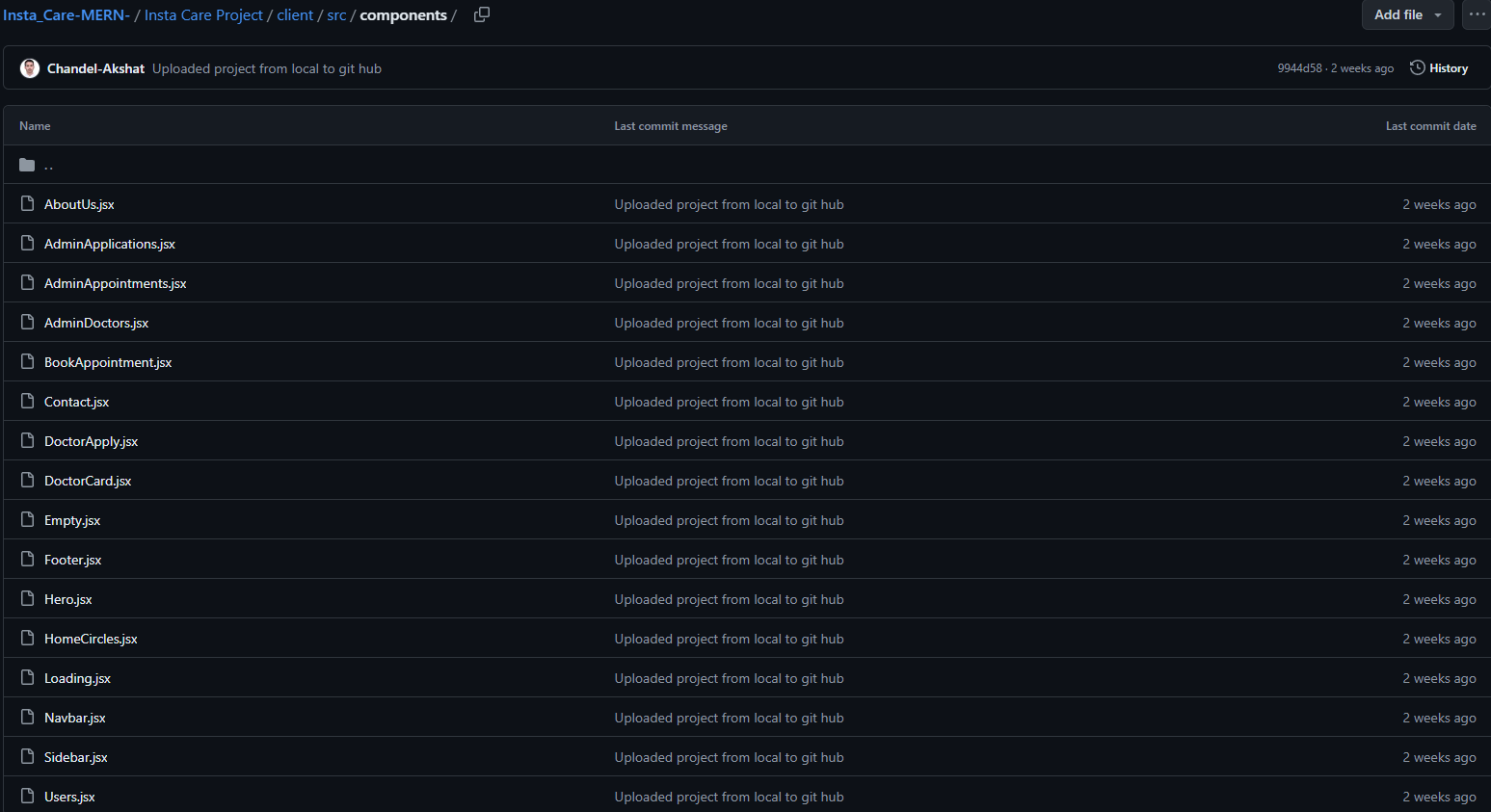
*Project 🡪 Client(Frontend)*



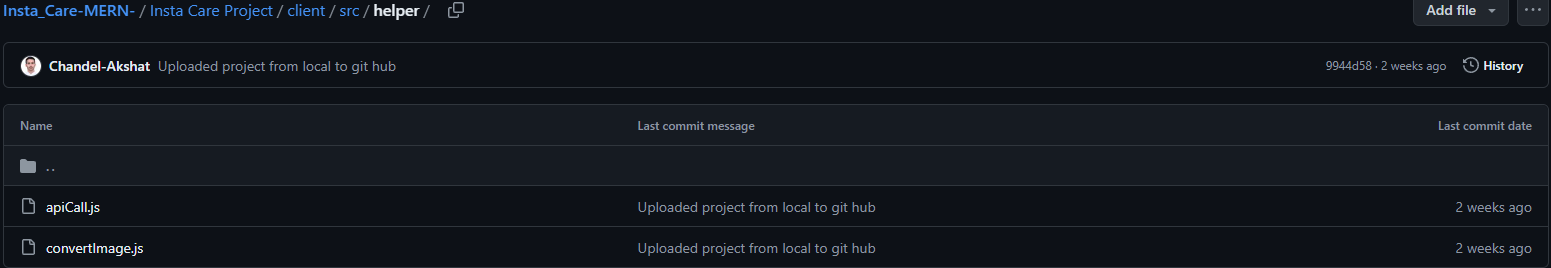
*Project 🡪 Client 🡪 Src*

**

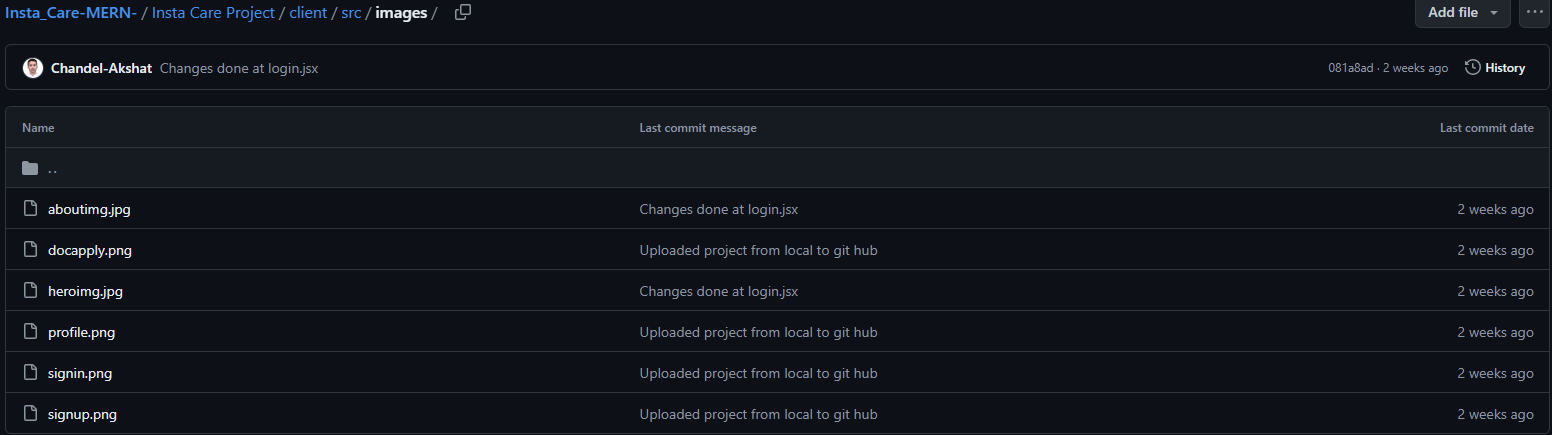
*Project 🡪 Client 🡪 Src 🡪 Components*

**

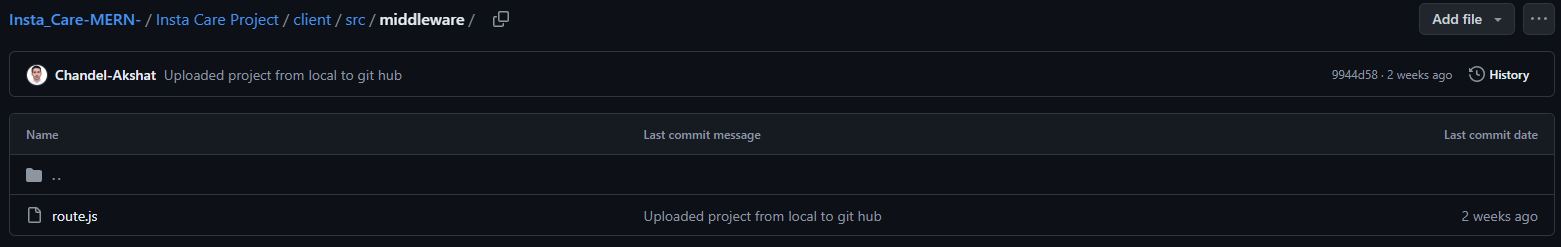
*Project 🡪 Client 🡪 Src 🡪 Helper*

**

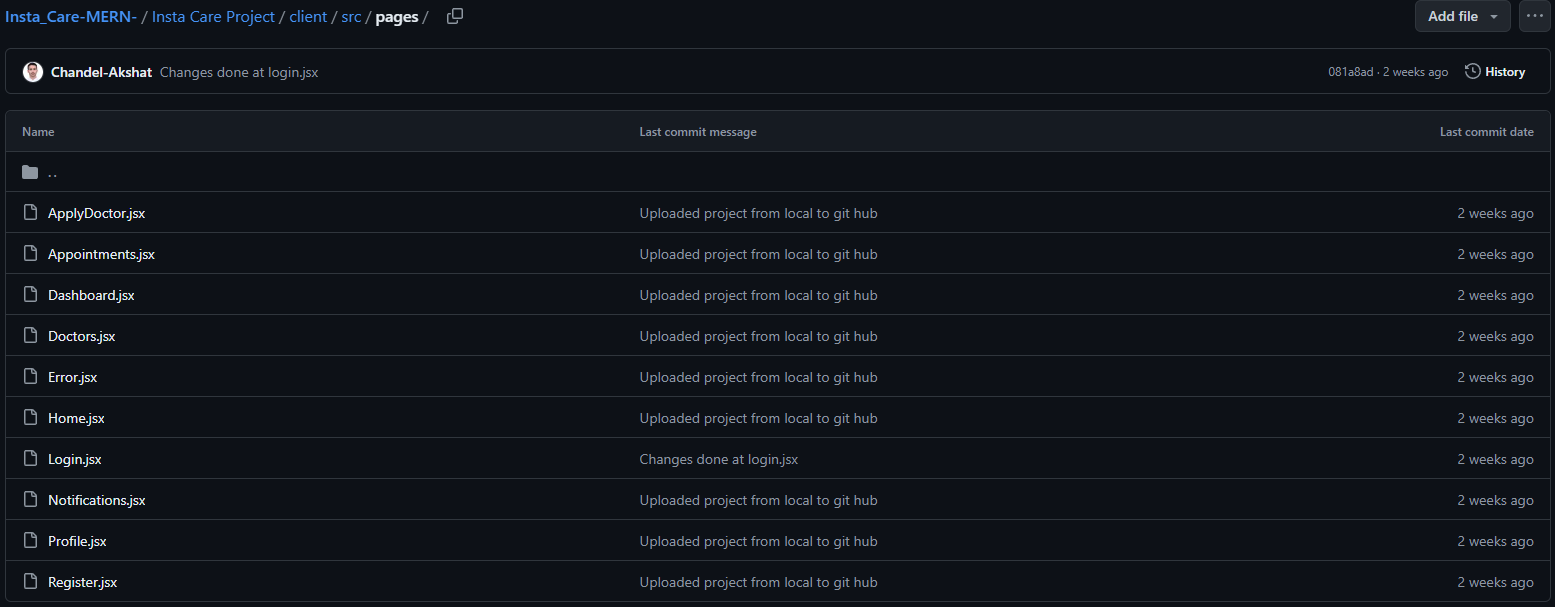
*Project 🡪 Client 🡪 Src 🡪 Images*

**

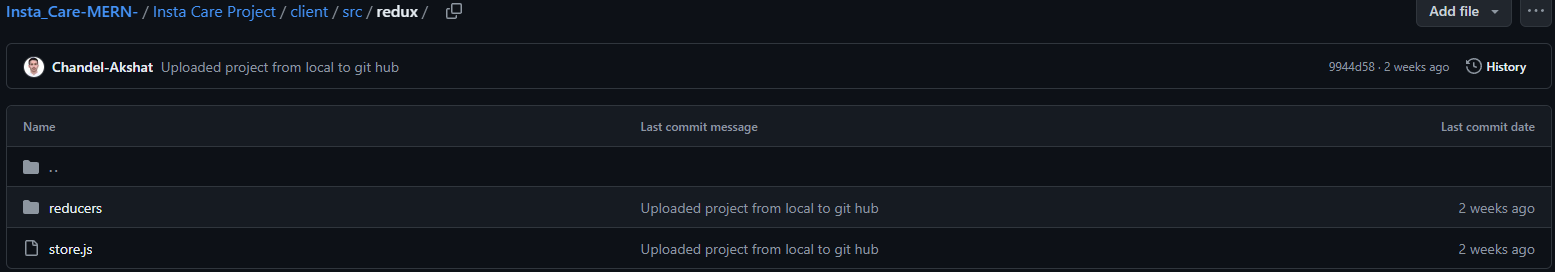
*Project 🡪 Client 🡪 Src 🡪 Middleware*

**

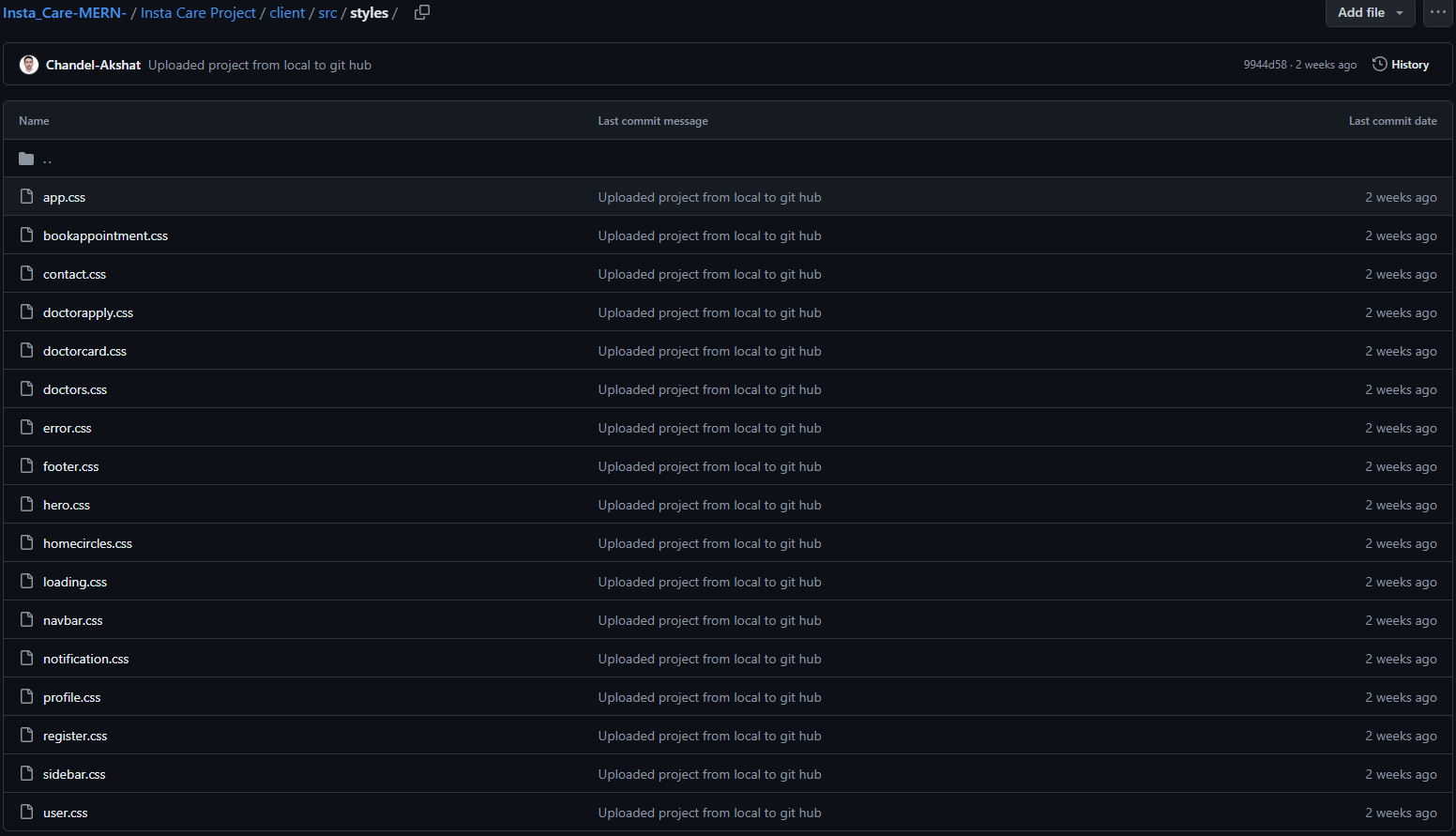
*Project 🡪 Client 🡪 Src 🡪 Pages*

**

*Project 🡪 Client 🡪 Src 🡪 Redux*

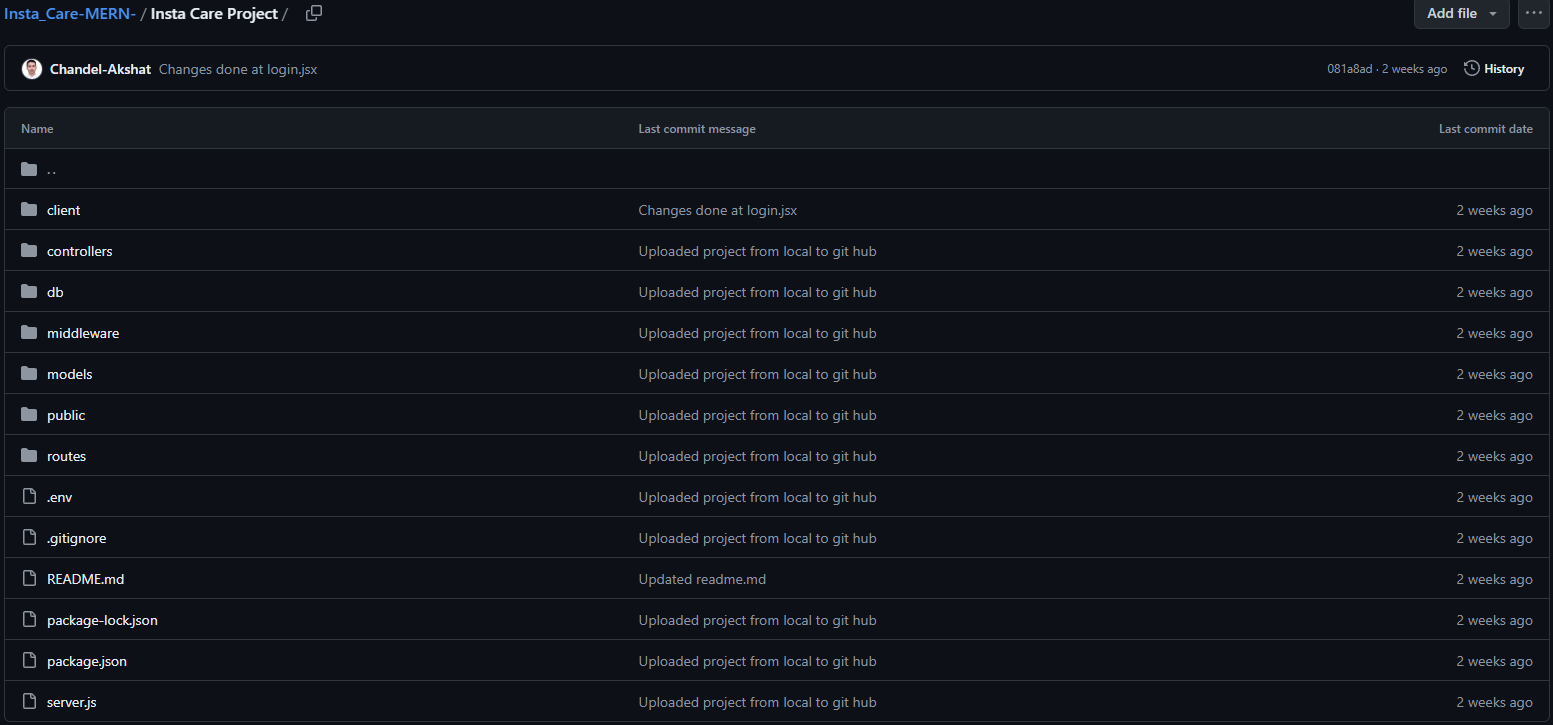
**

*Project 🡪 Client 🡪 Src 🡪 Styles*

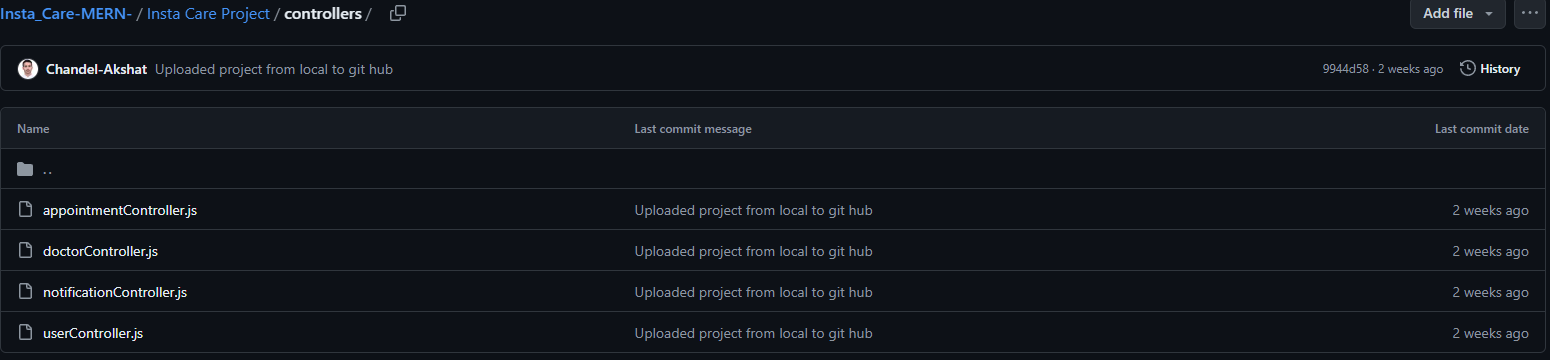
**

* **Backend:** Express routes for handling API requests, authentication, appointment management, and notification services.

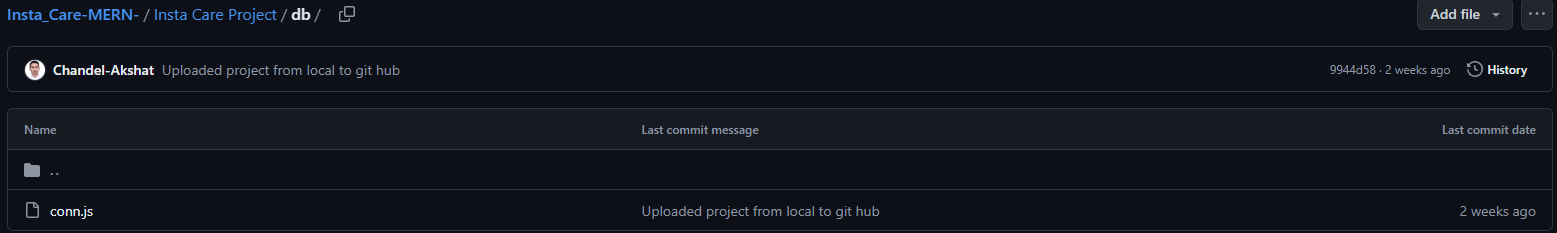
*Project*

**

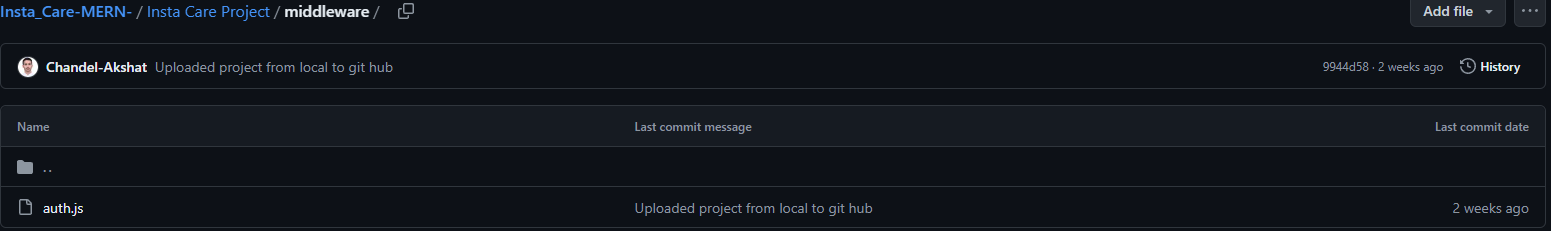
*Project 🡪 Controllers*

**

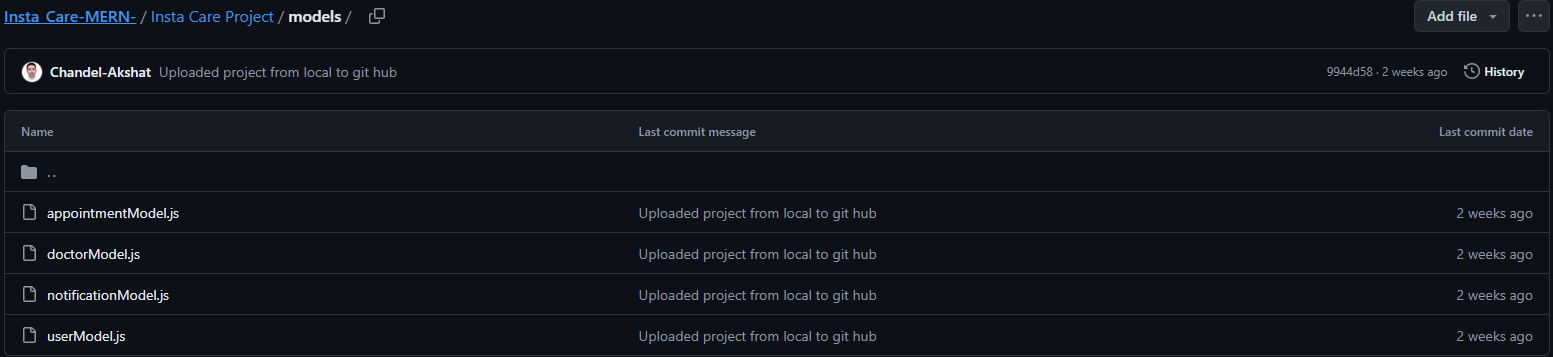
*Project 🡪 Db*

**

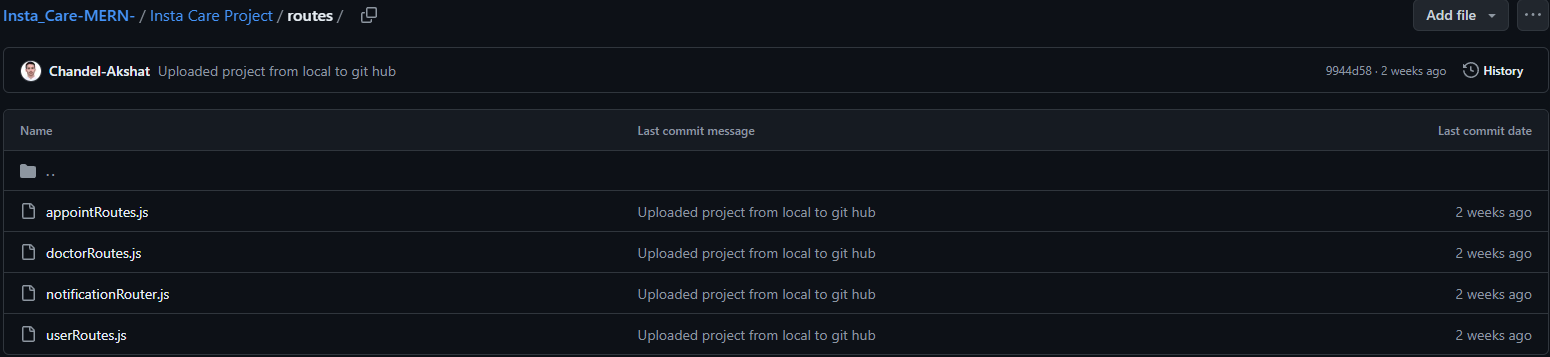
*Project 🡪 Middleware*

**

*Project 🡪 Models*

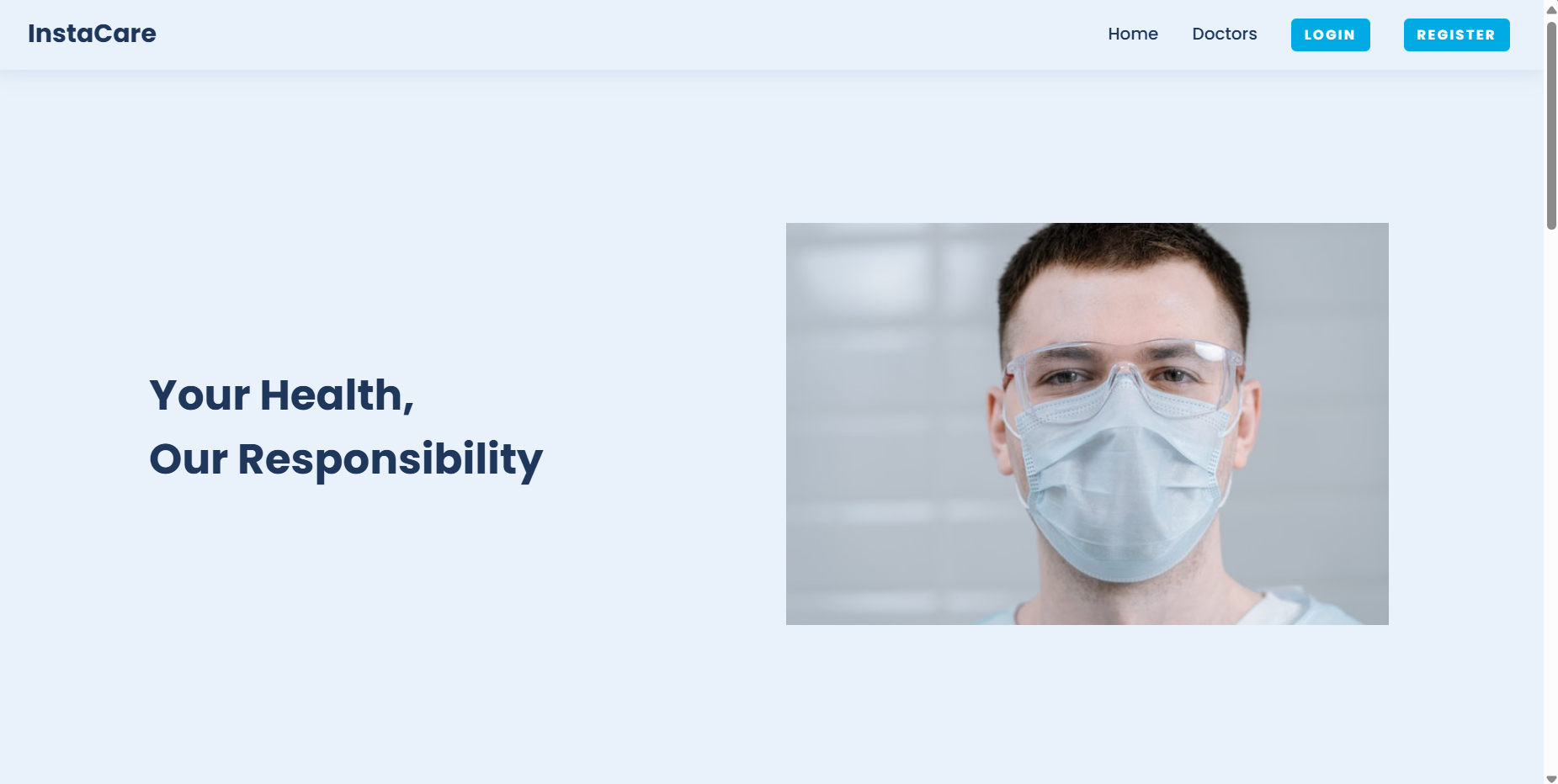
**

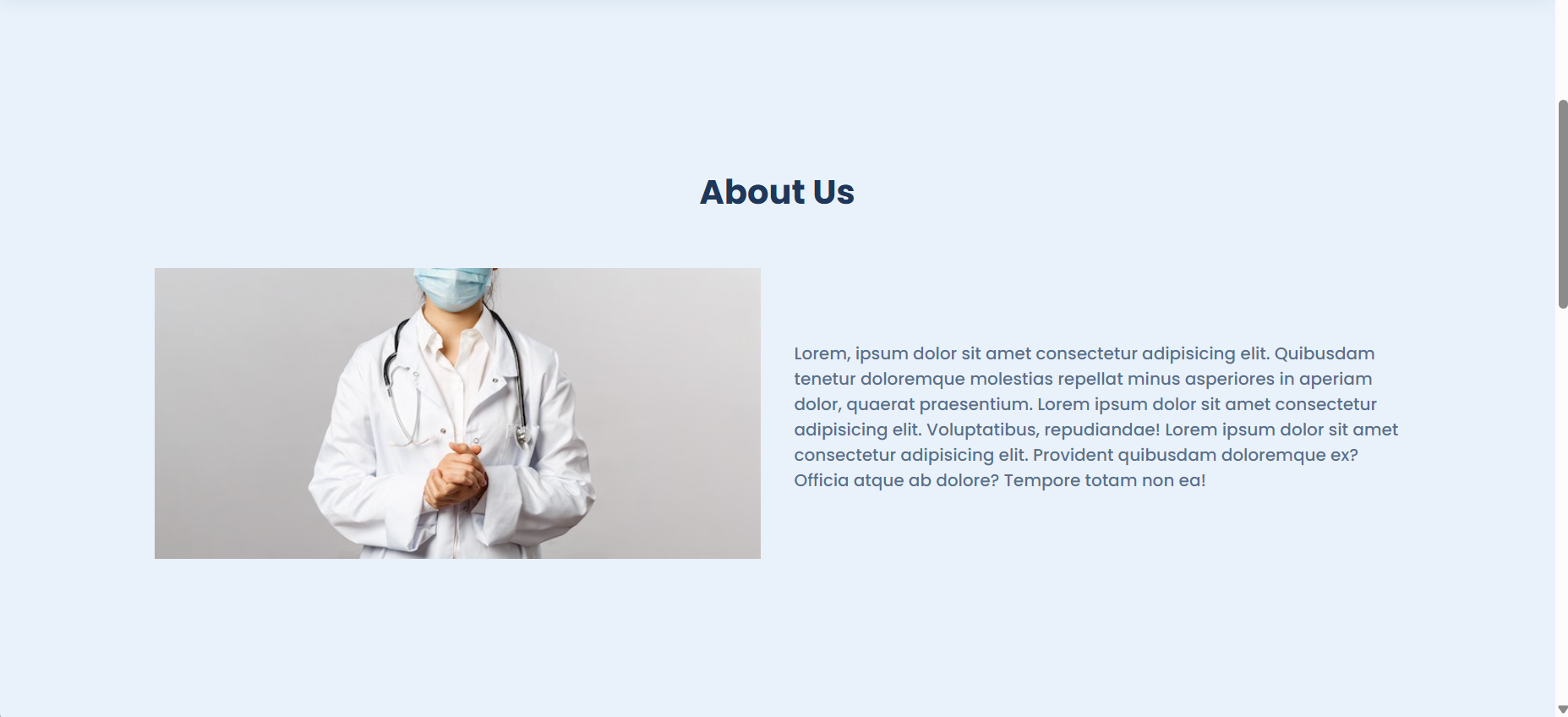
*Project 🡪 Routes*

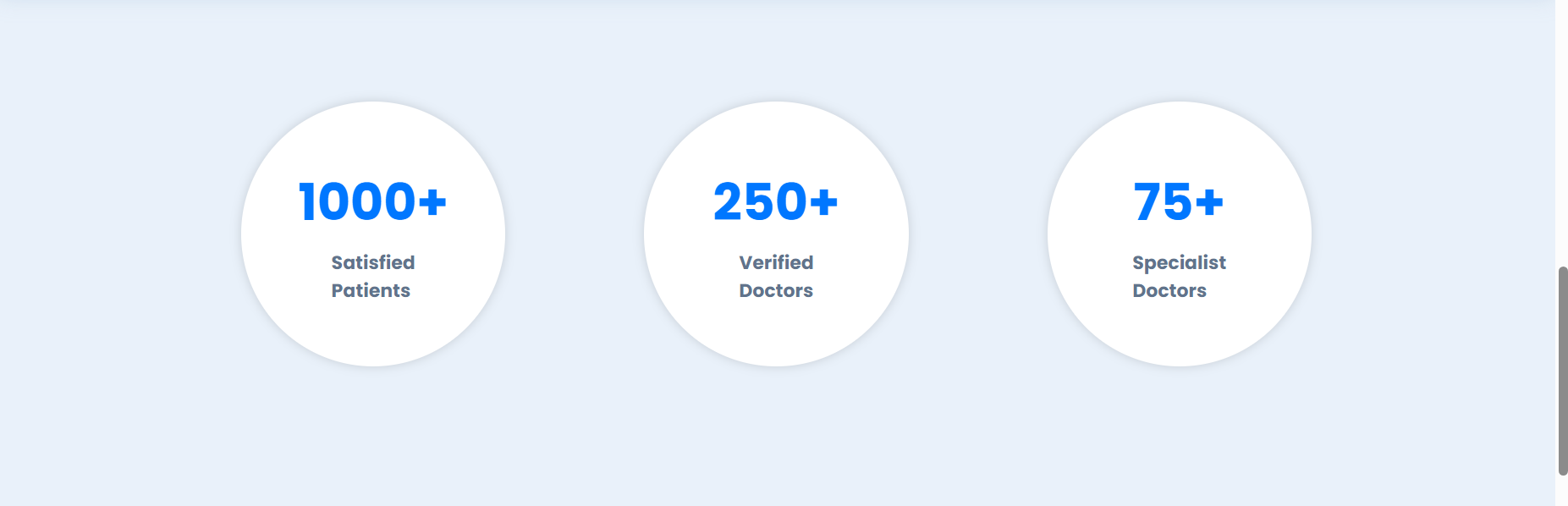
**

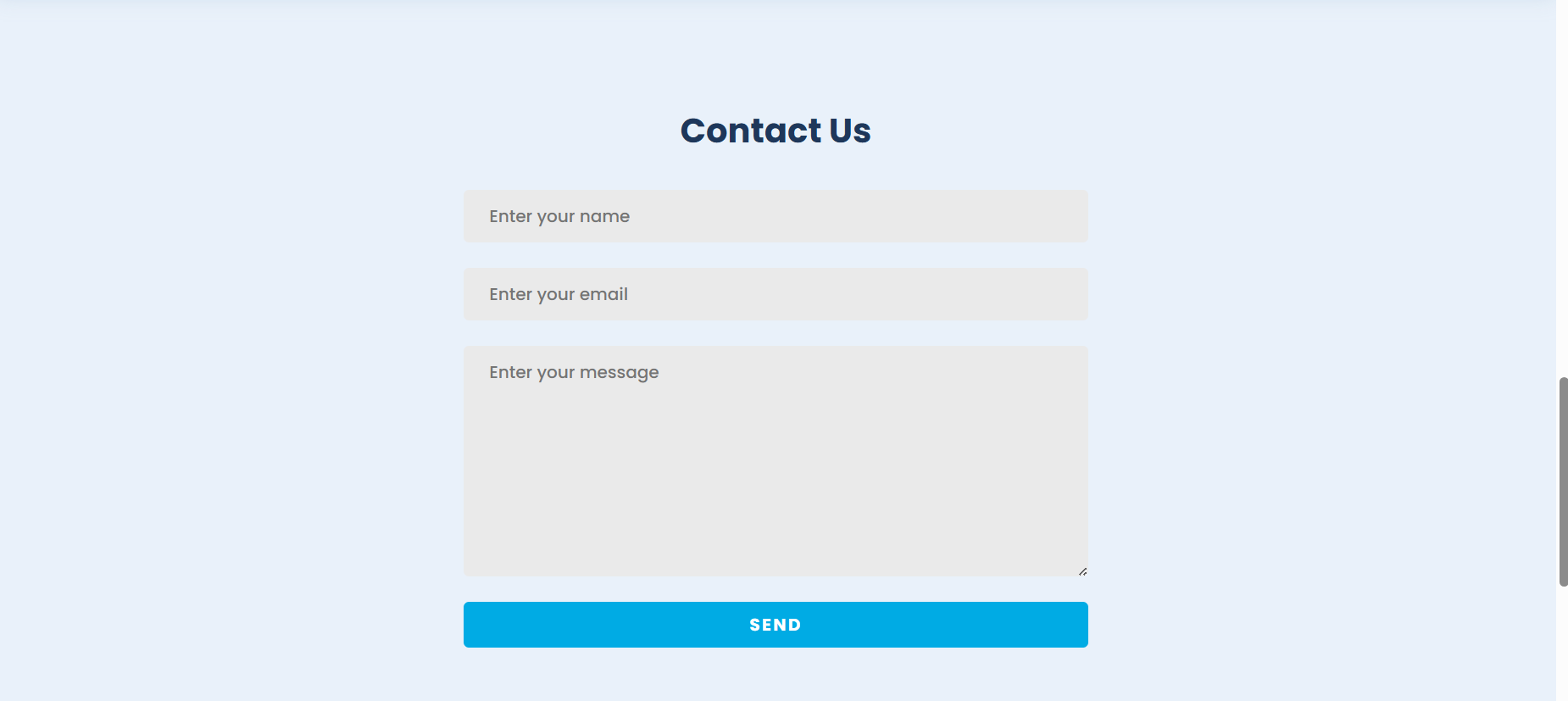
**GUI Snapshots**

* **Home Page:** Landing page of the website.

****

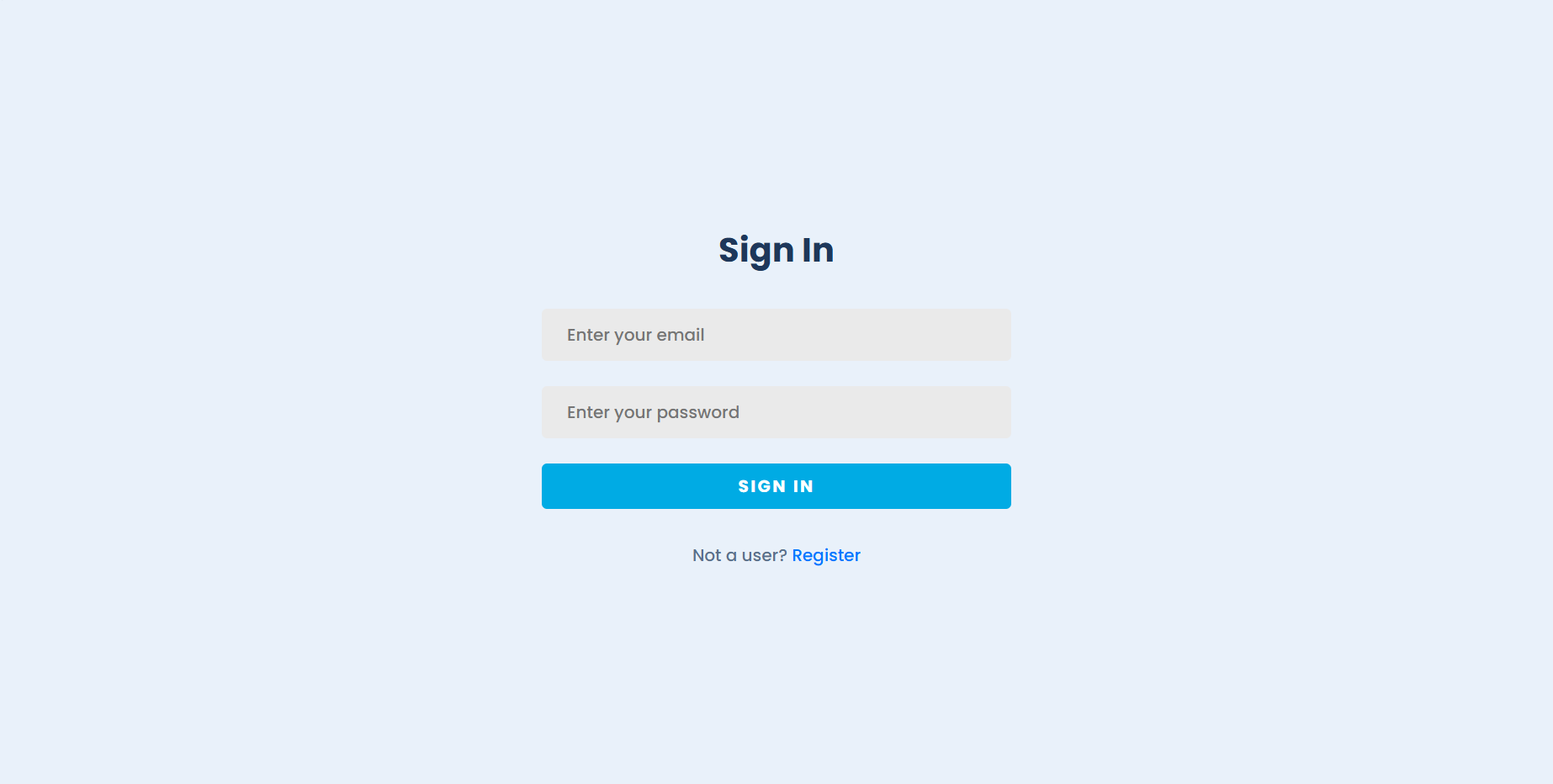
****

****

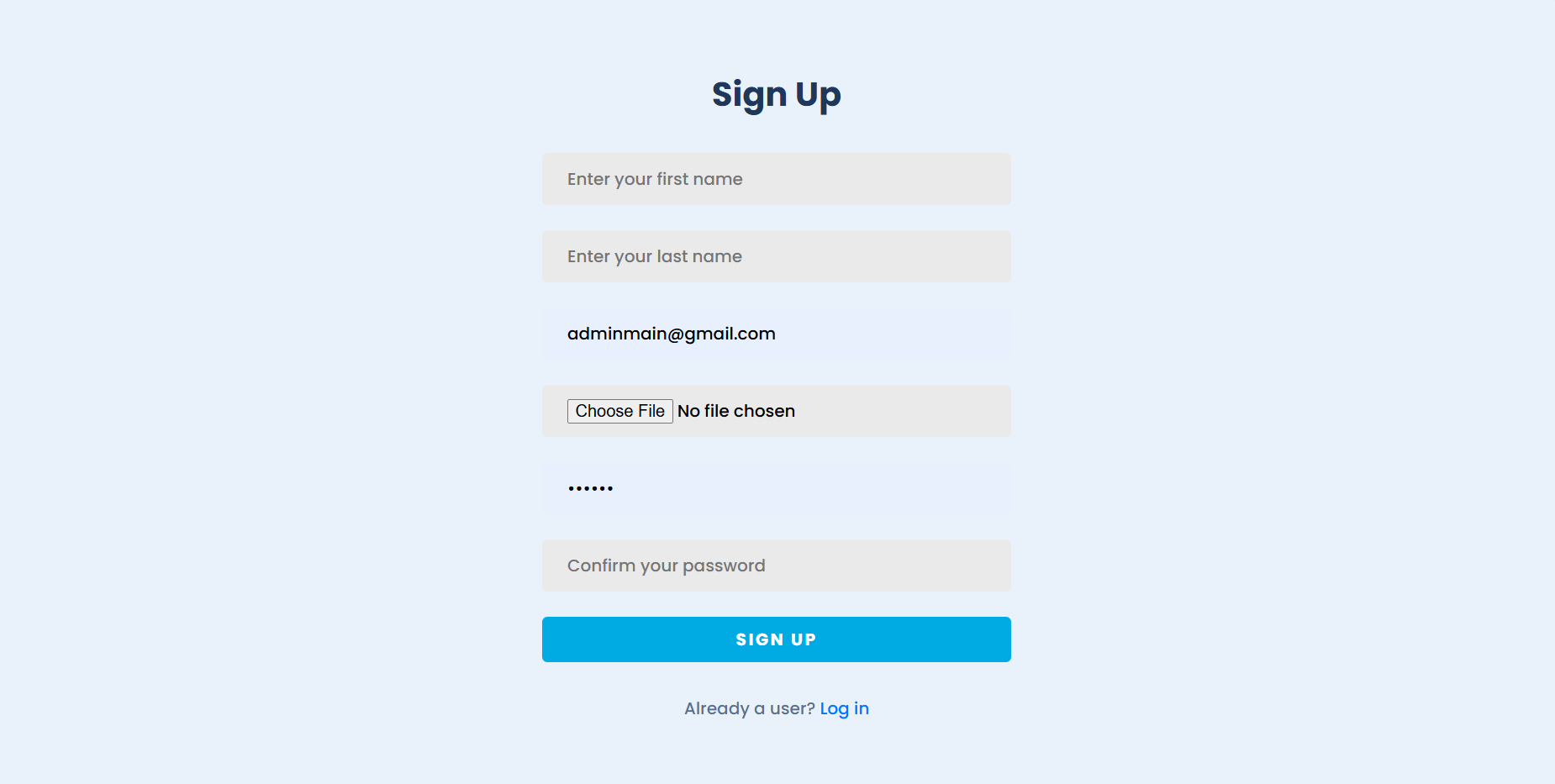
****

****

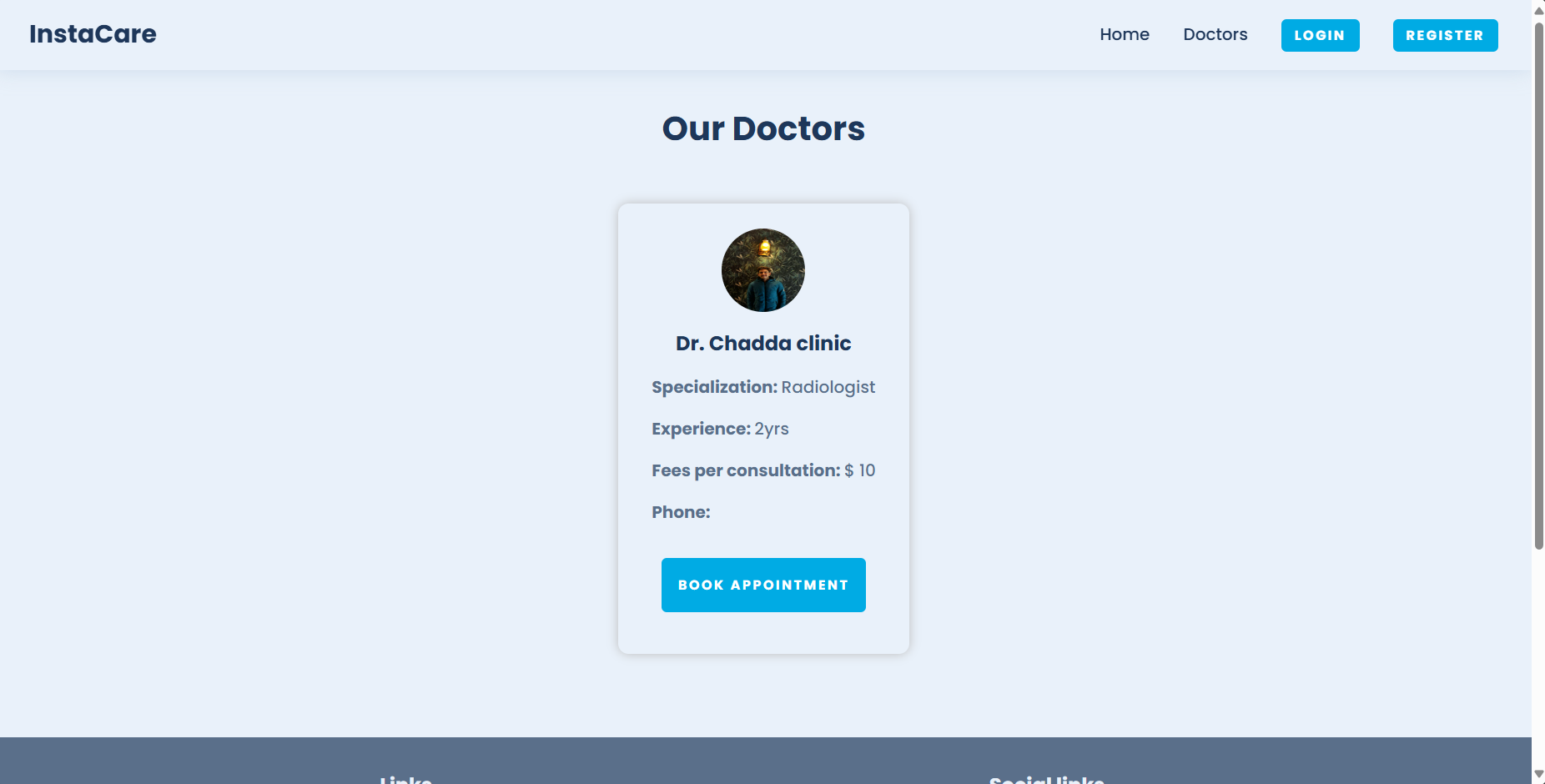
* **Sign-In Page:** Allow users to log in to the system.

****

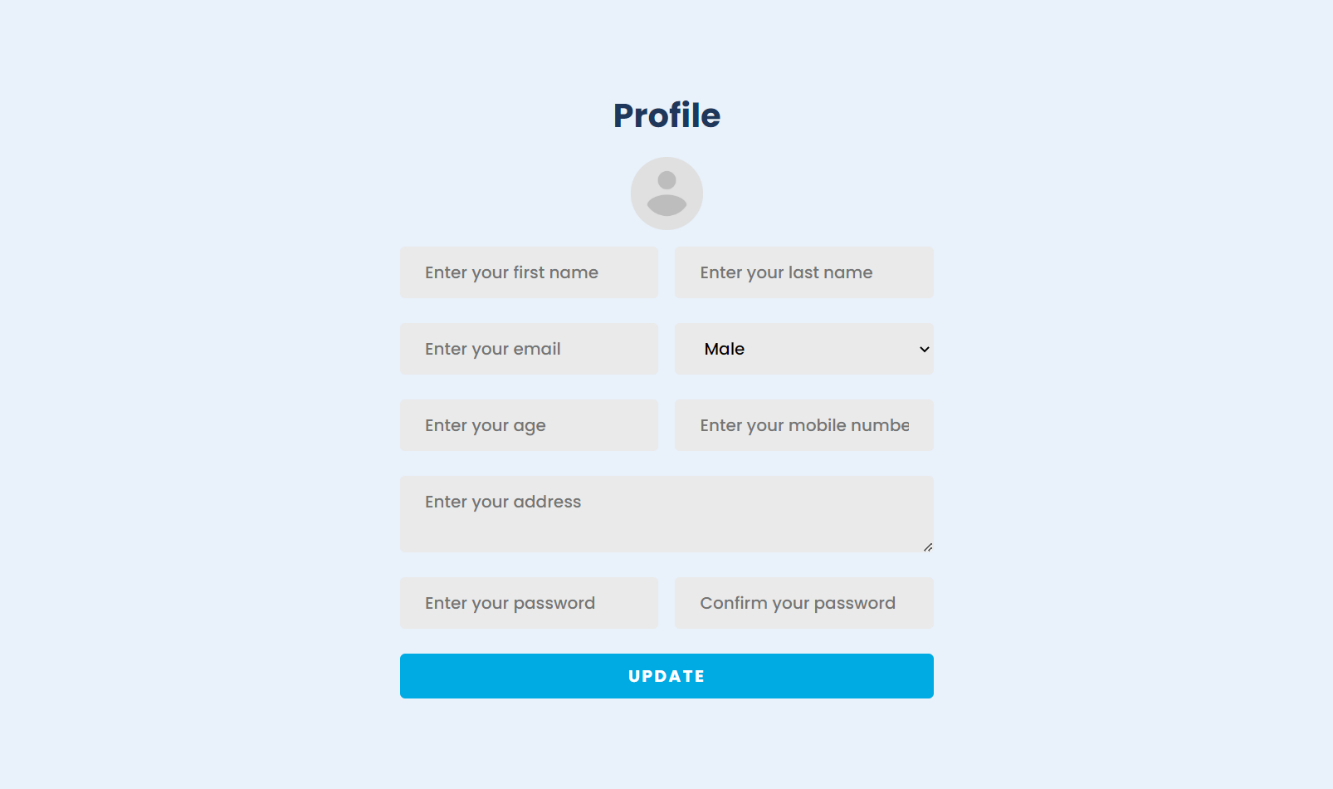
* **Sign-Up Page:** To register new users to the site.

****

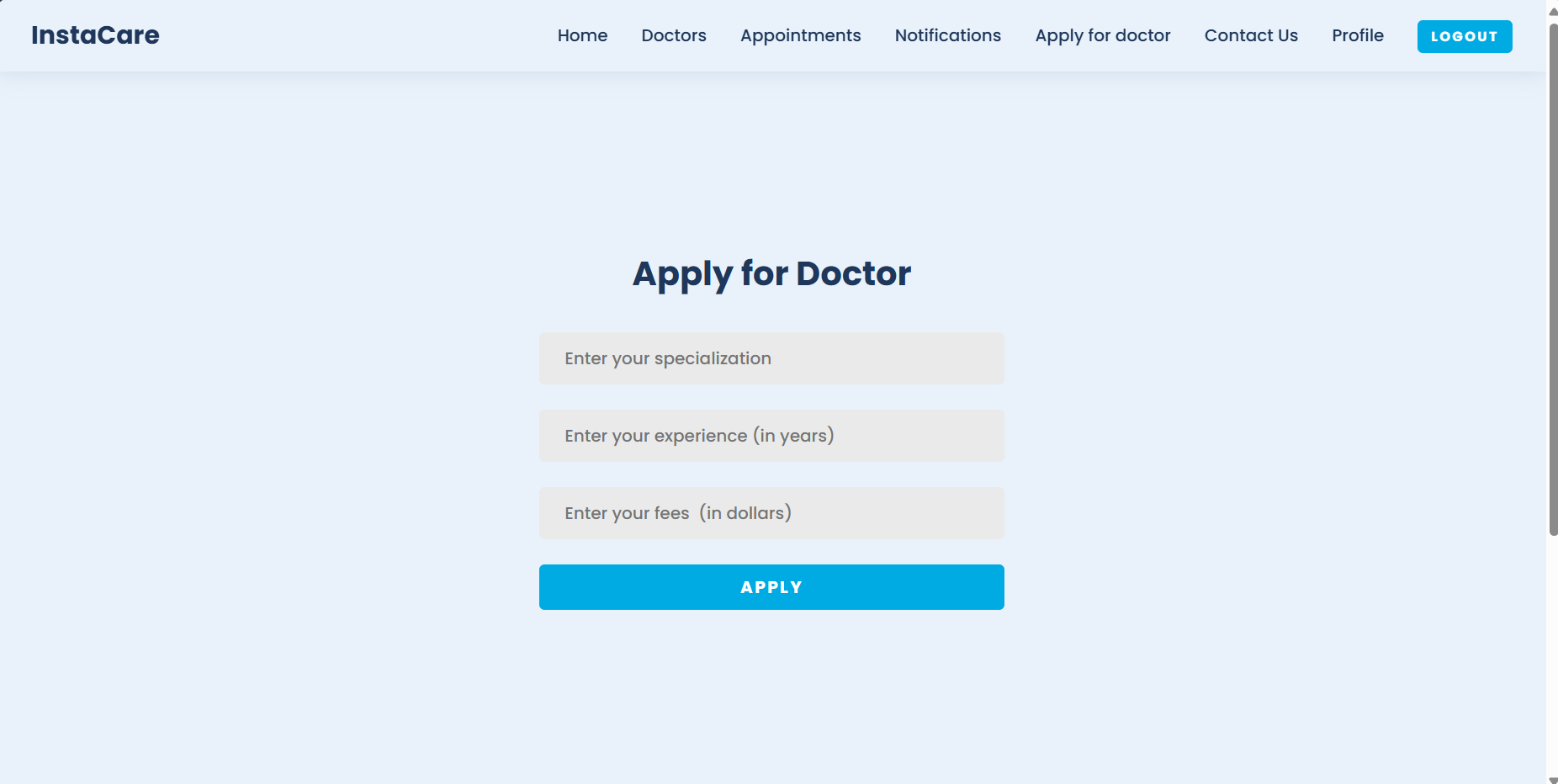
* **All Doctors:** Shows currently registered doctors with our website.

****

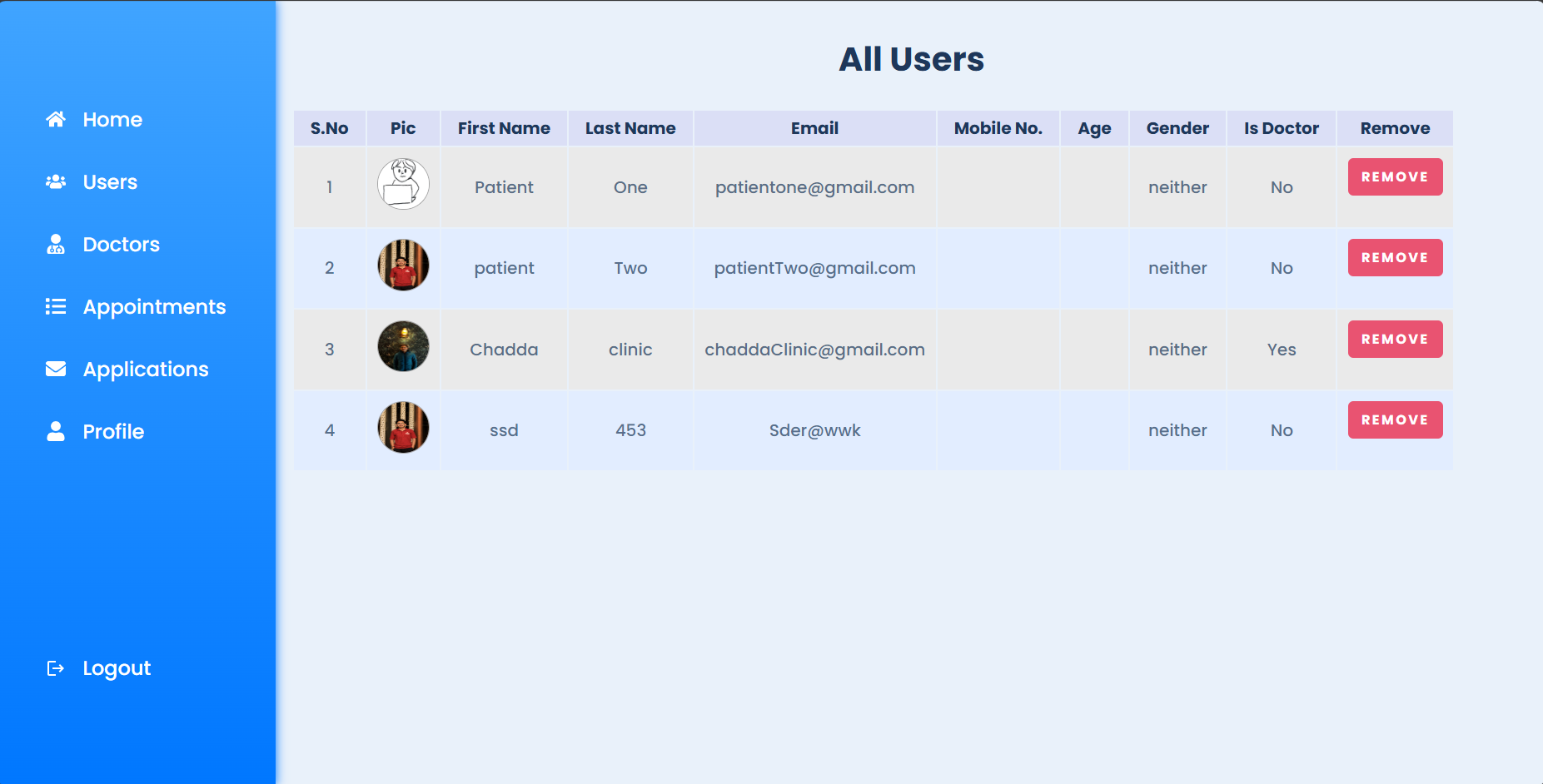
* **Profile Page:** Here you can make changes to the profile of current users.



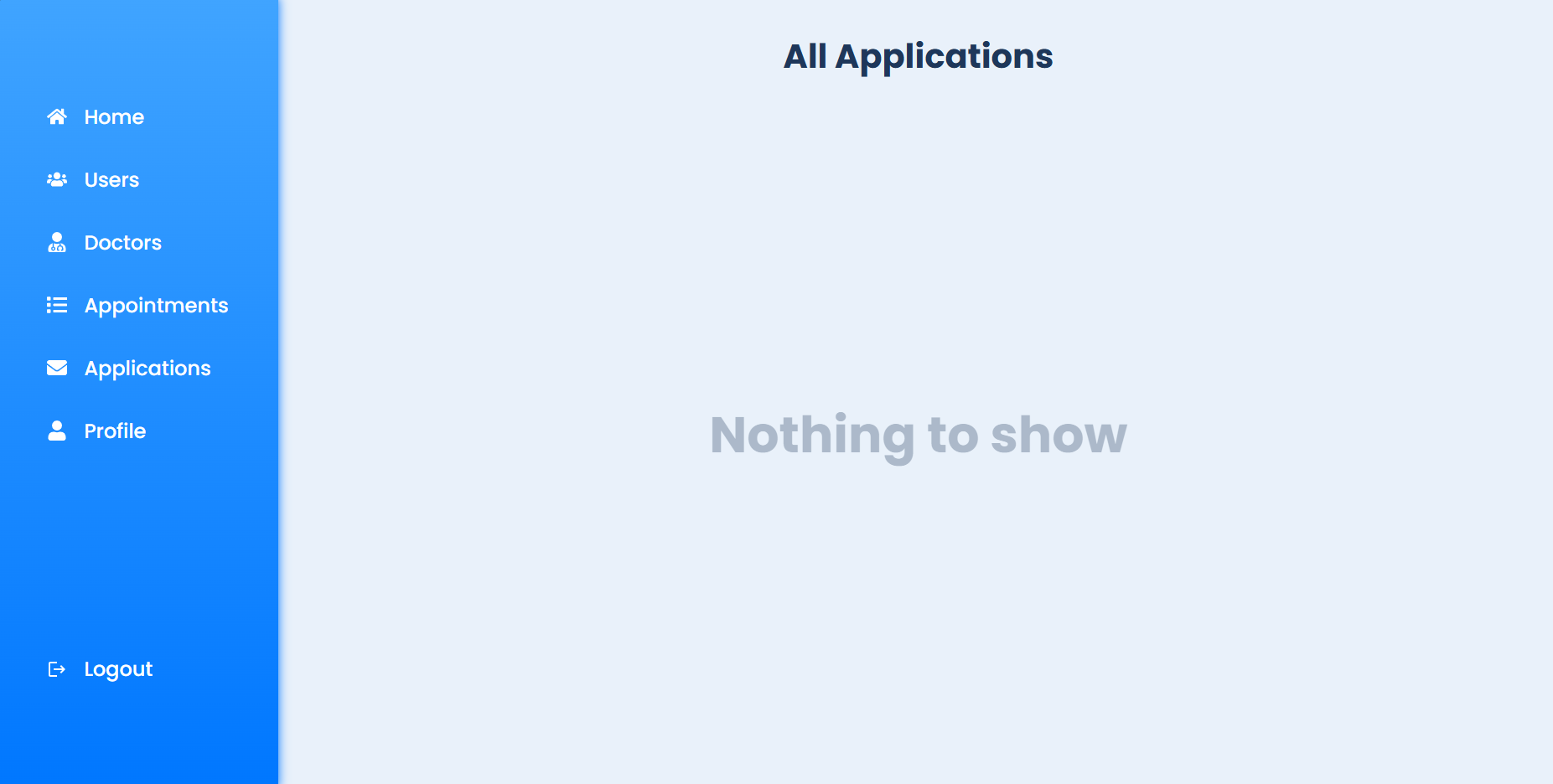
* **Apply Doctor Page:** Users can apply to become a doctor and now admin can approve it.

****

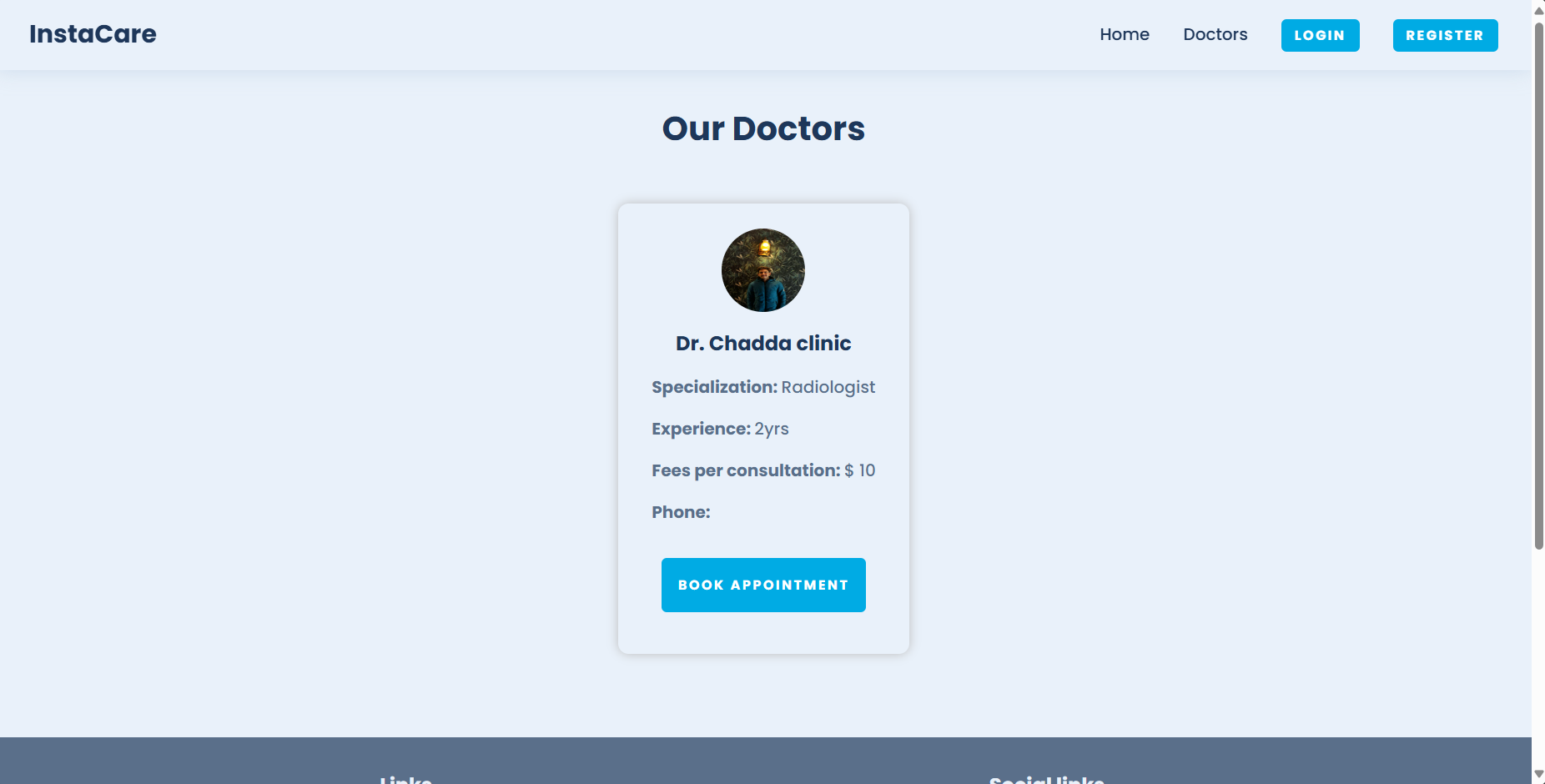
* **Admin all users Dashboard:** Shows the admin number of users and doctors. Help admin to do CRUD operations in the database.

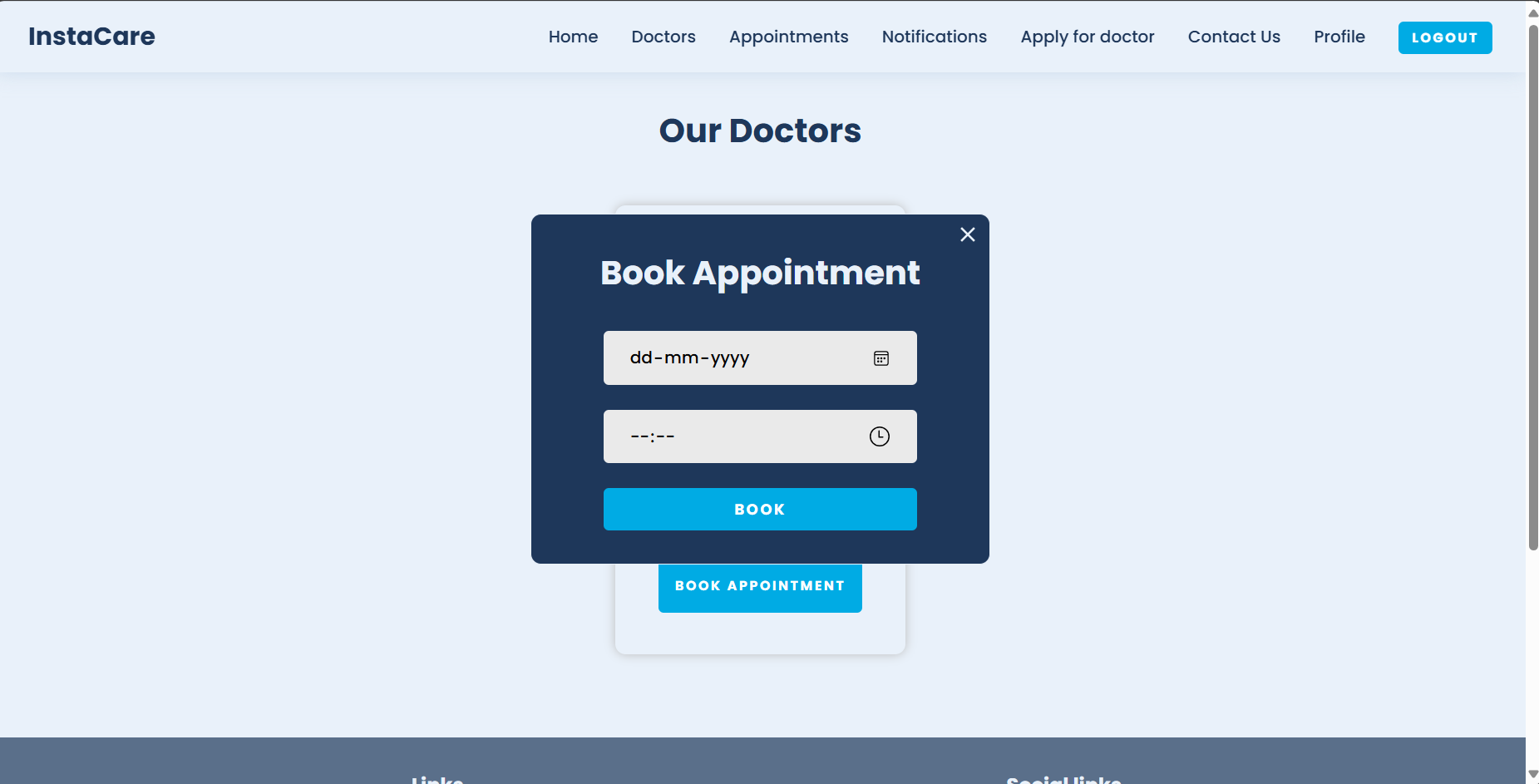
****

* **Admin all Applications:** Shows appointments booked by patients.



* **Book Appointment Page:** here you book an appointment with the preferable doctor you want.

****

****

**6. CODE IMPLEMENTATION AND DATABASE CONNECTIONS**

### Code Implementation

* **Frontend**: React components communicate with the backend via Axios for API requests.
* **Backend**: Express routes handle CRUD operations for users, appointments, and notifications.

### Database Connections

* **Mongoose**: Used to connect and interact with the MongoDB database.
* **Configuration**: Database connection settings are defined in a configuration file.

**7. CONCLUSION**

This project successfully addresses the key challenges faced by small clinics by providing a digital solution for appointment management, reducing no-show rates, enhancing online presence, and supporting marketing efforts. The web application developed using the MERN stack is robust, scalable, and user-friendly, offering a significant improvement over manual processes.

**8. FUTURE SCOPE**

### Potential Enhancements

* **Automated Reminders**: Implementing SMS and email reminders for appointments.
* **Advanced Analytics**: Providing insights into clinic performance and patient behaviour.
* **Telehealth Integration**: Adding support for virtual consultations.
* **Marketing Tools**: Developing features for automated marketing campaigns.

**9. BIBLIOGRAPHY/REFERENCES**

* MERN Stack Documentation
* MongoDB Documentation
* Express.js Documentation
* React Documentation
* Node.js Documentation
* Industry articles on clinic management and healthcare technology