INDEX

1. INTRODUCTION

1.1 Project Description :

The Online Health Checkup Booking Portal is a MERN (MongoDB, Express.js, React.js, Node.js) project designed to provide users with a convenient and efficient platform for booking health checkup appointments. The portal allows users to register and login to their accounts, where they can browse available doctors, book appointments, and receive notifications regarding their appointments. Additionally, the project includes an administrative panel that enables the approval of doctor requests and facilitates the management of the system.

1.2 Problem Statement :

The traditional process of booking health checkup appointments can be time-consuming and inconvenient for both users and healthcare providers. Users often face difficulties in finding suitable doctors, booking appointments, and receiving timely notifications. On the other hand, healthcare providers may struggle with managing appointment requests and efficiently communicating with their patients. These challenges necessitate the development of an automated and user-friendly system to streamline the appointment booking process.

1.3 Existing System :

The current system for health checkup appointment booking often relies on manual processes, such as phone calls or in-person visits to healthcare facilities. Users need to physically contact the healthcare provider, inquire about available time slots, and then schedule their appointments. This system lacks automation, making it prone to errors, delays, and miscommunication. It also lacks an organized platform for users to view and manage their appointments.

1.4 Limitation of Existing System :

The limitations of the existing system include:

- **Time-consuming process:** Users need to invest significant time in finding suitable doctors and scheduling appointments.
- **Lack of real-time availability:** Users may not have access to up-to-date information about available time slots.
- **Limited communication:** Users often rely on phone calls or face-to-face interactions, leading to potential miscommunication.
- **Manual approval process:** Healthcare providers need to manually approve appointment requests, leading to delays and potential administrative errors.
- **Absence of user-friendly interface:** The existing system may lack an intuitive interface for users to view and manage their appointments.

1.5 Proposed System :

The proposed Online Health Checkup Booking Portal aims to address the limitations of the existing system by providing an automated and user-friendly platform. The system will feature a comprehensive database of doctors, enabling users to browse and select suitable healthcare providers based on their specialization, availability, and patient reviews. Users will be able to register and login to their accounts, where they can easily book appointments with their chosen doctors.

Furthermore, the project includes an administrative panel that allows the system administrator to approve doctor requests and convert regular user accounts into doctor accounts. This feature ensures that only verified healthcare professionals can accept appointment requests. The proposed system will also implement a notification system, enabling users and doctors to receive timely updates about their appointments through their respective dashboards.

1.6 Objectives :

The main objectives of the Online Health Checkup Booking Portal are as follows:

- Provide users with a user-friendly platform for booking health checkup appointments.
- Enable users to browse and select suitable doctors based on specialization, availability, and reviews.
- Automate the appointment booking process, eliminating the need for manual intervention.
- Implement a notification system to keep users and doctors informed about appointment updates.
- Allow the system administrator to approve doctor requests and manage the system effectively.

1.7 Benefits of Proposed System :

The proposed system offers several benefits over the existing system, including:

- Time-saving: Users can easily find and book appointments with suitable doctors through a streamlined process.
- Real-time availability: Users have access to up-to-date information about doctor availability, ensuring more accurate scheduling.
- Improved communication: Users and doctors can receive notifications regarding their appointments, reducing the chances of miscommunication.
- Efficient administrative management: The system administrator can effectively handle doctor requests and oversee the entire system.
- Enhanced user experience: The user-friendly interface and automated processes enhance the overall experience of users and doctors.

2. SYSTEM REQUIREMENTS

2.1 Hardware Specifications :

- **Server:** You will need a reliable server to host your web application. The server should have sufficient processing power, memory, and storage capacity to handle the expected traffic and data storage requirements. Consider using a cloud hosting service like AWS, Google Cloud, or Azure, or a dedicated server if you prefer.
- **Database Server:** You will need a database server to store and retrieve your application's data. You can use MongoDB as the database for your MERN stack project. Ensure that the database server meets the recommended system requirements for MongoDB.
- **Network Infrastructure:** A stable and secure network infrastructure is essential to ensure smooth communication between the server and the client's devices. Make sure you have a reliable internet connection and proper network configurations in place.

2.2 Software Specifications :

- **Operating System:** Choose an operating system compatible with the MERN stack. You can use any popular operating system such as Linux (e.g., Ubuntu, CentOS), Windows, or macOS, depending on your familiarity and preferences.
- **Node.js:** Install the latest stable version of Node.js on the server. Node.js is the runtime environment for executing JavaScript on the server-side. Ensure that the Node.js version is compatible with the MERN stack framework you'll be using.
- **MongoDB:** Install and configure MongoDB on the server. MongoDB is a NoSQL database that works seamlessly with Node.js and is commonly used in MERN stack applications.
- **Web Browser:** Ensure that your web application is compatible with popular web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. Consider testing your application's compatibility with different browser versions to provide a consistent user experience.

2.3 Tools Survey :

- **Text Editors/IDEs:** Choose a text editor or integrated development environment (IDE) for writing your code. Some popular options include Visual Studio Code, Sublime Text, Atom, or WebStorm. Select the one that suits your preferences and offers useful features for web development.
- **Express.js:** Express is a popular web application framework for Node.js. It provides a robust set of features for building web APIs and handling HTTP requests and responses.
- **React.js:** React is a JavaScript library for building user interfaces. It allows you to create reusable UI components and efficiently update the UI when data changes. React is a key component of the MERN stack, responsible for rendering the frontend of your application.
- **Package Manager:** NPM (Node Package Manager) comes bundled with Node.js and is used to install and manage packages and dependencies for your MERN project. It allows you to easily integrate third-party libraries or frameworks into your application.
- **Axios:** Axios is a widely used HTTP client for making asynchronous requests from the frontend to the backend. It simplifies the process of sending and handling API requests and responses.
- **React Router:** React Router is a library for handling routing in React applications. It allows you to define routes and navigation between different pages or components within your application.
- **Bootstrap or Material-UI:** These are popular frontend frameworks that provide pre-designed UI components and CSS styles. They can help you build responsive and visually appealing user interfaces more efficiently.
- **JWT (JSON Web Tokens):** JWT is a standard for securely transmitting information between parties as a JSON object. It is commonly used for authentication and authorization purposes in web applications.

3. SYSTEM DESIGN

3.1 Modules

3.1.1 User Module:

- **User Registration:** Allows users to create an account by providing their personal information.
- **User Login:** Enables users to securely log into their accounts using their credentials.
- **Doctor Search:** Provides a search functionality for users to browse and find doctors based on specialization, location, and availability.
- **Appointment Booking:** Allows users to select a doctor, choose an available time slot, and book appointments for health checkups.
- **Notification System:** Sends notifications to users regarding appointment confirmations, reminders, and any updates or changes.

3.1.2 Doctor Module:

- **Doctor Request:** Allows doctors to submit a request to become a registered doctor on the platform by providing their credentials and qualifications.
- **Doctor Profile Management:** Enables doctors to update and manage their professional information, including specialization, availability, and clinic details.
- **Appointment Management:** Provides doctors with a dashboard to view their upcoming appointments, reschedule if necessary, and mark appointments as completed.
- **Notification System:** Sends notifications to doctors regarding new appointment requests, changes in appointments, and any important updates.

3.1.3 Admin Module:

- **Doctor Approval:** Enables the system administrator to review and approve doctor requests, ensuring only verified healthcare professionals are listed on the platform.
- **User and Doctor Management:** Provides administrative capabilities to manage user and doctor accounts, including account activation, deactivation, or modification.

3.2 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the flow of data in a system. It is used to visualize and analyze the flow of information within a system and identify potential bottlenecks, inefficiencies, or security risks. A DFD typically consists of a set of symbols and shapes that represent various components in a system and the relationships between them.

DFDs are useful for a wide range of applications, including systems analysis, software design, process improvement, and project management. They are often used in the early stages of software development to define the requirements for a system and to identify potential problems before they occur.

3.2.1 Advantages of Data Flow Diagrams:

- **Clarity:** DFDs provide a clear visual representation of the flow of data in a system, making it easier to understand and communicate to others.
- **Systematic Approach:** DFDs help to identify the different components of a system and the relationships between them, making it easier to analyze and improve the system.
- **Improved Accuracy:** DFDs help to ensure that data is accurately represented in a system, reducing the risk of errors or discrepancies.
- **Early Identification of Problems:** DFDs can be used to identify potential bottlenecks, inefficiencies, or security risks in a system before they occur, allowing for early resolution.
- **Better Understanding of System Requirements:** DFDs help to define the requirements for a system and ensure that all stakeholders have a clear understanding of what the system is expected to do.

3.2.2 Disadvantages of Data Flow Diagrams:

- **Complexity:** For complex systems, creating a detailed and accurate DFD can be time consuming and require a high level of expertise.
- **Limited Flexibility:** Once a DFD has been created, it can be difficult to change or modify, making it less flexible than other design tools.
- **Limited Representation of Real-World Situations:** DFDs are a simplified representation of a system and may not accurately reflect real-world situations.
- **Limited Representation of Dynamic Systems:** DFDs are designed to represent static systems and may not accurately represent dynamic systems that change over time.
- **Limited Representation of Non-Data Aspects:** DFDs are focused on the flow of data and may not accurately represent other important aspects of a system, such as user interfaces or security requirements.

3.2.3 Components of Data Flow Diagrams:

	Process	A process shows a transformation or manipulation of data flows within the system.
	External Entity	An external entity is a source or destination of a data flow which is outside the area of study. Only those entities which originate or receive data are represented on a business process diagram.
	Data Store	A data store is a holding place for information within the system. It is represented by an open-ended narrow rectangle.
	Data Flow	A data flow shows the flow of information from its source to its destination. A data flow is represented by a line, with arrowheads showing the direction of flow.

Fig. No. 3.1

3.2.4 Data Flow Diagrams Levels

Context Level (Zero level) DFD:

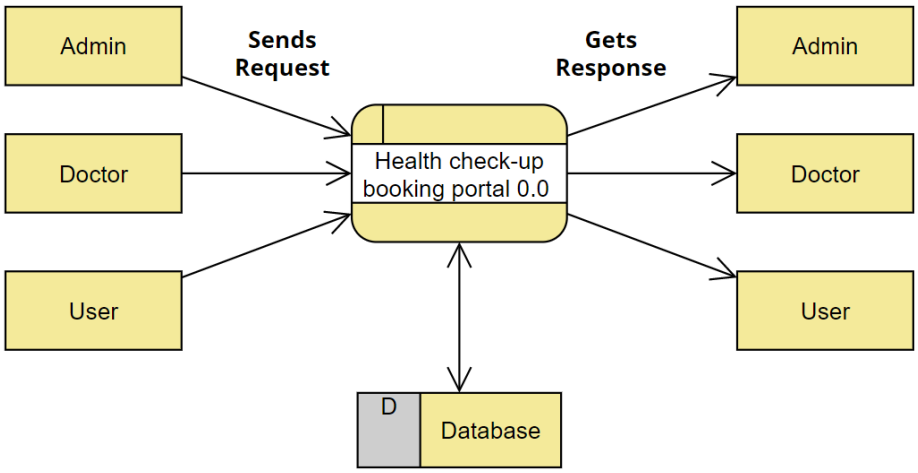

Fig. No. 3.2

First Level DFD:

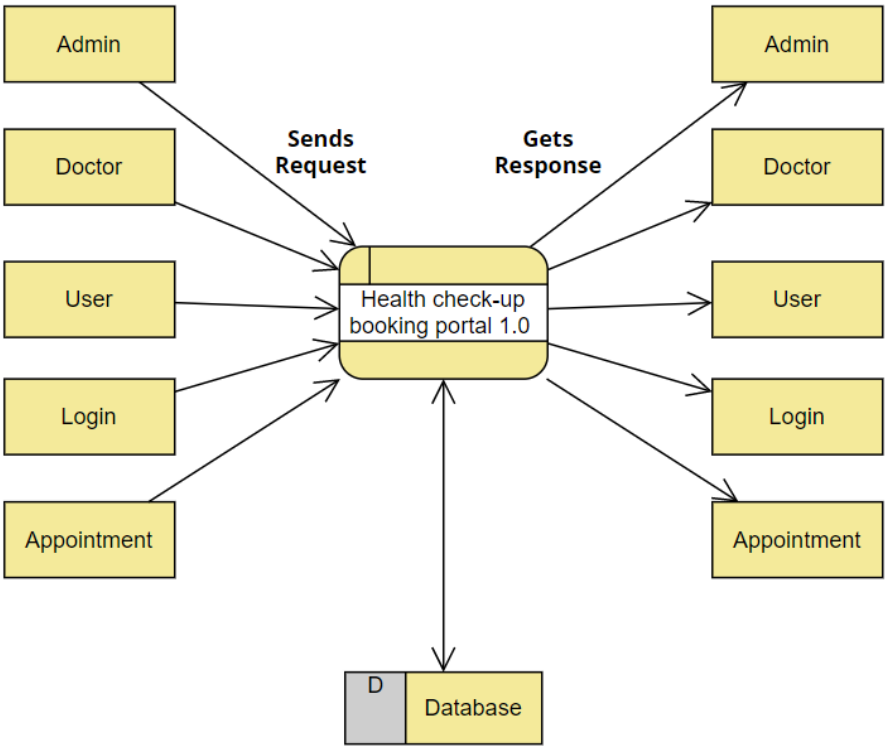

Fig. No. 3.3

Second Level DFD:

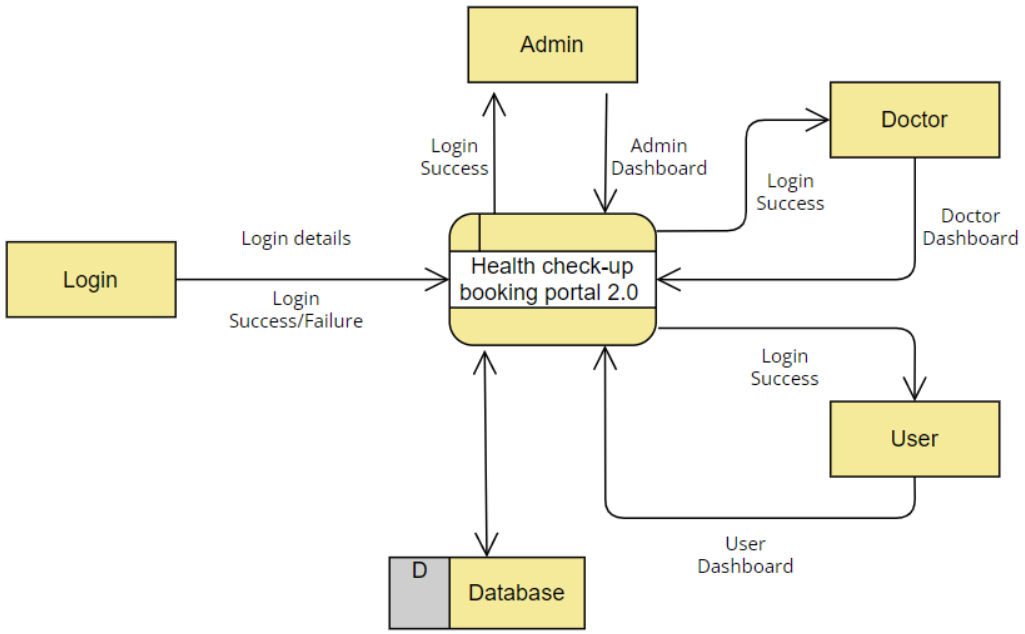

Fig. No. 3.4

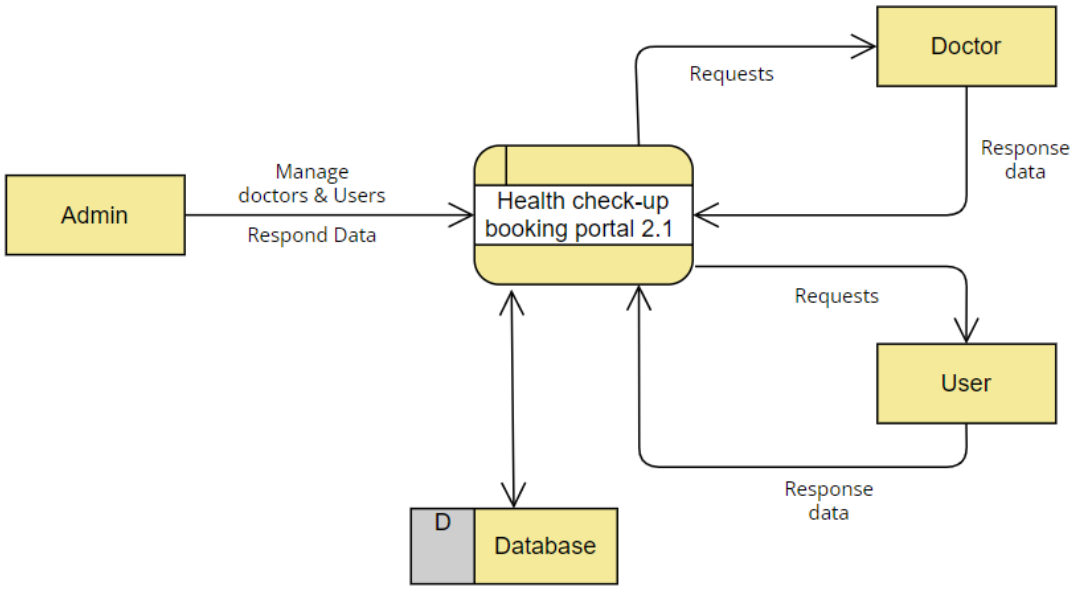

Fig. No. 3.5

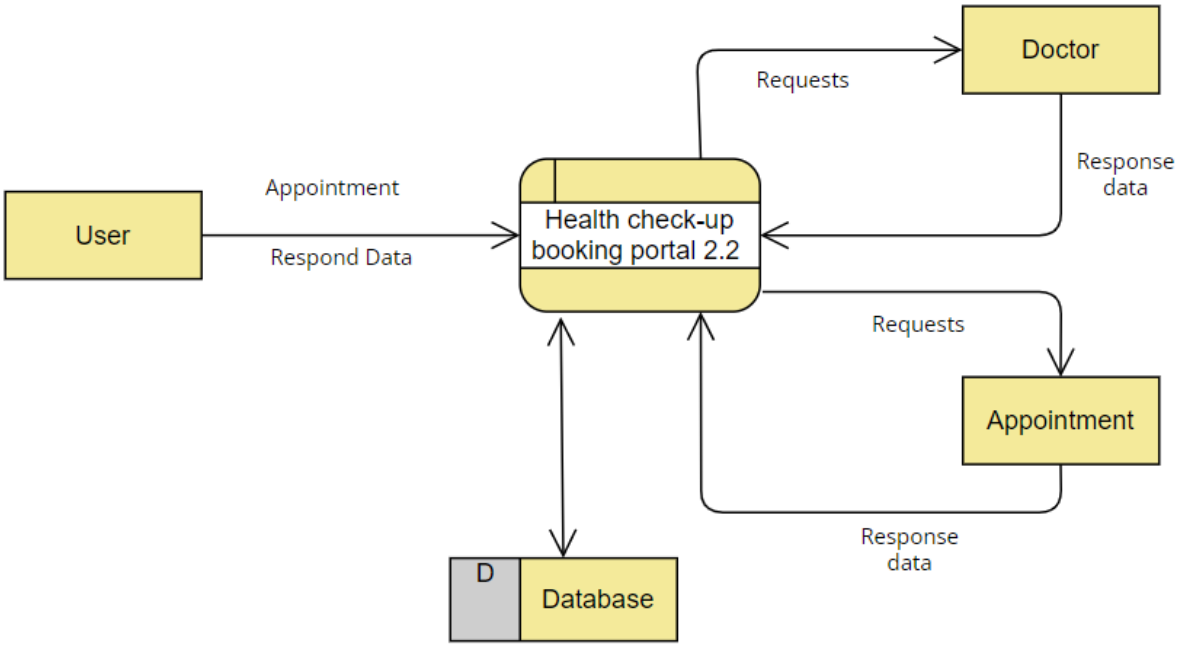

Fig. No. 3.6

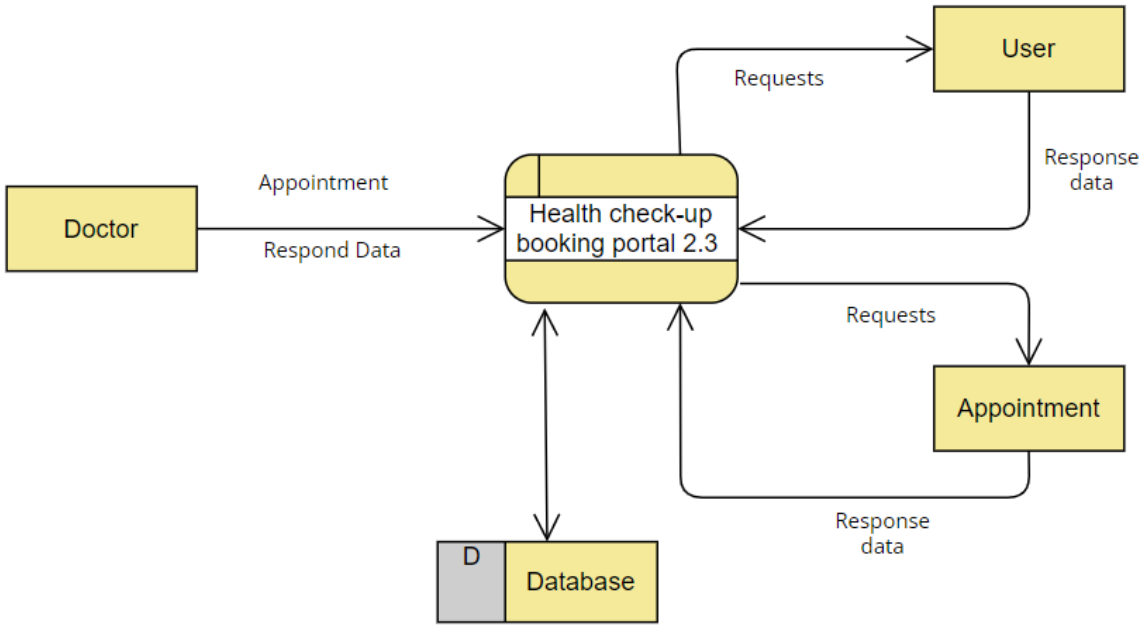

Fig. No. 3.7

3.3 ER-Diagram

An Entity Relationship Diagram (ERD) is a graphical representation of entities and their relationships to each other, used in database design. An entity can be a real-world object, such as a customer or an order, with attributes described by columns in a database table. Relationships between entities are depicted as lines connecting them, with a diamond shape representing the relationship itself and identifying the cardinality (e.g. one-to-one, one-to-many, many-to-many). ERDs help visualize the structure of data and communicate design decisions to stakeholders.

Entities: The representation of real-world objects or concepts as graphical symbols, with their properties defined as attributes.

Relationships: The connections between entities, depicted as lines connecting the entities, with a diamond shape representing the relationship itself and identifying the cardinality.

3.3.1 Advantages of Entity Relationship Diagram (ERD):

- **Clear visualization:** ERDs provide a clear and concise visual representation of entities and their relationships, making it easier for stakeholders to understand the data structure and design decisions.
- **Database design aid:** ERDs help database designers to identify entities, attributes, and relationships before creating tables, thus reducing the risk of design errors.
- **Improved data integrity:** ERDs help ensure data integrity by highlighting relationships between entities and enforcing rules, such as referential integrity, through the use of primary and foreign keys.
- **Facilitated communication:** ERDs can be used to communicate design decisions to stakeholders and facilitate discussion about the data structure.
- **Ease of maintenance:** ERDs make it easier to maintain the database as it grows and evolves, as changes to the data structure can be easily communicated and made through updating the diagram.

3.3.2 Disadvantages of Entity Relationship Diagram (ERD):

- **Complexity:** ERDs can become complex, especially for large and sophisticated databases, making it difficult to understand the relationships and design decisions.
- **Limited functionality:** ERDs only provide a high-level overview of the data structure and do not include detailed information such as data types and constraints.
- **Time-consuming:** Creating an ERD can be time-consuming, especially for large and complex databases, as it requires a thorough understanding of the data requirements and design decisions.
- **Difficulty in making changes:** Making changes to the data structure once the ERD is complete can be challenging, as it requires updates to the diagram and the corresponding database design.

3.3.3 Entity Relationship Diagram (ERD):

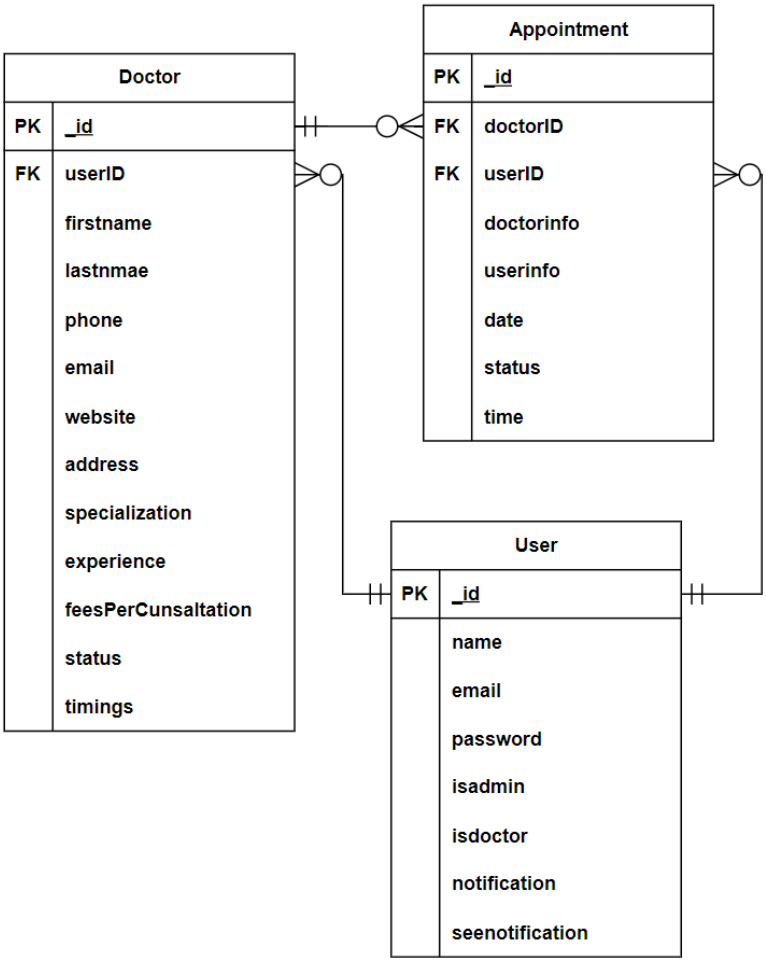

Fig. No. 3.8

3.4 Database Design

Database Name - Healthcheckup

1. **User Collection**

	Field name	**Datatype**	**Null**
PK	_id	Objectid	No
	name	string	No
	email	string	No
	password	string	No
	isadmin	Boolean	No
	isdoctor	Boolean	No
	notification	Array	No
	seennotification	Array	No

Table. No. 3.1

2. **Doctor Collection**

	Field name	**Datatype**	**Null**
PK	_id	Objectid	No
FK	userID	string	No
	firstname	string	No
	lastname	string	No
	phone	String	No
	email	String	No
	website	String	No
	address	String	No
	specialization	String	No
	experience	String	No
	feesPerCunsaltation	Int32	No
	status	String	No
	timings	Array	No

Table. No. 3.2

3. Appointment Collection

	Field name	Datatype	Null
PK	_id	Objectid	No
FK	userId	string	No
FK	doctorId	string	No
	userInfo	string	No
	date	string	No
	status	string	No
	time	string	No

Table. No. 3.3

4. CODING

4.1 Front-End

4.1.1 Client/src/components/– Folder

1 About.js

```
import React from "react";
import Navbar from "./Navbar";
function About() {
  return (
    <Navbar>
      <div className="container">
        <h1 className="text-center mt-4">About Us</h1>
        <p>
          Welcome to the Online Health Check-up Booking Portal! We are dedicated
          to providing convenient and accessible health check-up services to
          individuals. Our platform allows you to book appointments for a
          variety of check-ups, including general health check-ups, specialized
          check-ups, and dental check-ups.
        </p>
        <p>
          Our team of experienced healthcare professionals ensures that you
          receive the best care and attention during your check-up appointments.
          We strive to make the booking process seamless and efficient, allowing
          you to schedule your appointments with ease.
        </p>
        <p>
          With our Online Health Check-up Booking Portal, you can take control
          of your health by staying proactive and maintaining regular check-ups.
          We believe that prevention is better than cure, and our platform
          empowers you to prioritize your well-being.
        </p>
        <p>
          Thank you for choosing our platform for your health check-up needs. We
```

```
        look forward to serving you and helping you maintain a healthy
        lifestyle.
       </p>
     </div>{" "}
   </Navbar>
  );
}
export default About;
```

2 Contact.js

```
import React from "react";
import Navbar from "./Navbar";
function Contact() {
  return (
    <Navbar>
      <div className="container">
        <h2 className="text-center mt-4">Contact Us</h2>
        <p>
          For any queries or assistance, please feel free to reach out to us.
          You can contact our customer support team at:
        </p>
        <p className="email">support@healthcheckup.com</p>
        <p>
          We aim to respond to all inquiries within 24 hours. Whether you have
          questions about our services, need help with an appointment, or
          require any other information, our team is here to assist you.
        </p>
        <p>
          When contacting us via email, please ensure to provide the following
          details to help us serve you better:
        </p>
        <ul>
          <li>Your full name</li>
          <li>Contact number</li>
```

```
        <li>Appointment details (if applicable)</li>
        <li>Details of your inquiry or issue</li>
       </ul>
       <p>Example format for emailing us:</p>
       <pre>
        <code>
         Subject: [Your Name] - [Inquiry/Issue]
         <br />
         <br />
         Full Name: [Your Name]
         <br />
         Contact Number: [Your Contact Number]
         <br />
         Appointment Details (if applicable): [Appointment Details]
         <br />
         <br />
         [Details of your inquiry or issue]
        </code>
       </pre>
       <p>
        We appreciate your feedback and strive to provide you with the best
        possible customer service. Thank you for choosing our Online Health
        Check-up Booking Portal!
       </p>
      </div>{" "}
    </Navbar>
  );
}
export default Contact;
```

3 Footer.js

```
import React from "react";
import { NavLink } from "react-router-dom";
import "./../../styles/LayoutStyles.css";
```

```
const Footer = () => {
  return (
    <div className="footer">
      <h6 className="text-center text-white">
        Copyrights &copy;
        {new Date().getFullYear()} <span className="text-danger">|</span> Online
        Health Checkup Booking Portal <span className="text-danger">|</span> All
        Right Reserved
      </h6>
      <p className="text-center mt3">
        <NavLink to="/about">About Us</NavLink>|
        <NavLink to="/contact">Contact Us</NavLink>|
        <NavLink to="/policy">Privacy & Policy</NavLink>|
        <NavLink to="/terms">Terms & Conditions</NavLink>
      </p>
    </div>
  );
};
export default Footer;
```

4 Header.js

```
import React from "react";
import { NavLink, Link } from "react-router-dom";
import logo from "./../../images/healthcheckuplogo.png";
function Header() {
  const token = localStorage.getItem("token");
  return (
    <>
      <nav className="navbar navbar-expand-lg navbar-dark bg-info ">
        <button
          className="navbar-toggler"
          type="button"
          data-bs-toggle="collapse"
          data-bs-target="#navbarTogglerDemo01"
```

```
  aria-controls="navbarTogglerDemo01"
  aria-expanded="false"
  aria-label="Toggle navigation"
 >
  <span className="navbar-toggler-icon" />
 </button>
 <div className="collapse navbar-collapse" id="navbarTogglerDemo01">
  <Link to="/" className="navbar-brand p-3">
   <img src={logo} alt="logo" style={{ width: "auto", height: 60 }} />
      ONLINE HEALTH CHECKUP BOOKING PORTAL
  </Link>
  <ul className="navbar-nav ms-auto mt-2 mt-lg-0">
   <li className="nav-item">
    <NavLink to="/" className="nav-link text-light">
     Home
    </NavLink>
   </li>
   {token ? (
    <>
     {" "}
     <li className="nav-item">
      <NavLink to="/dashboard" className="nav-link text-light">
       Dashboard
      </NavLink>
     </li>
     <li className="nav-item">
      <NavLink
       to="/dashboard/apply-doctor"
       className="nav-link text-light"
      >
       Apply Doctor
      </NavLink>
     </li>
    </>
```

```
          ) : (
            <>
              {" "}
              <li className="nav-item">
                <NavLink to="/register" className="nav-link text-light">
                  Register
                </NavLink>
              </li>
              <li className="nav-item">
                <NavLink to="/login" className="nav-link text-light">
                  Login
                </NavLink>
              </li>
            </>
          )}
        </ul>
      </div>
    </nav>
  </>
 );
}
export default Header;
```

5 Navbar.js

```
import React from "react";
import Header from "./Header";
import Footer from "./Footer";
const Navbar = ({ children }) => {
 return (
  <div>
    <Header />
    <main style={{ minHeight: "70vh" }}>{children}</main>
    <Footer />
  </div>
```

```
  );
};
export default Navbar;
```

6 Pagenotfound.js

```
import React from "react";
import Navbar from "./Navbar";
import { Link } from "react-router-dom";
function Pagenotfound() {
  return (
    <Navbar>
      <center>
        <section className="page_404">
          <div className="container">
            <div className="row">
              <div className="col-sm-12 ">
                <div className="col-sm-10 col-sm-offset-1  text-center">
                  <div className="four_zero_four_bg">
                    <h1 className="text-center ">404 Page Not Found</h1>
                  </div>
                  <div className="contant_box_404">
                    <h3 className="h2">Look like you're lost</h3>
                    <p>The page you are looking for is not availble!</p>
                    <Link to="/" className="link_404">
                      Go to Home
                    </Link>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </section>
      </center>
    </Navbar>
```

```
  );
}
export default Pagenotfound;
```

7 Policy.js

```
import React from "react";
import Navbar from "./Navbar";
function Policy() {
  return (
    <Navbar>
      <div className="container">
        <h1 className="text-center mt-4">Privacy Policy</h1>
        <p>
          At the Online Health Check-up Booking Portal, we take your privacy
          seriously. This Privacy Policy outlines how we collect, use, and
          protect your personal information when you use our services.
        </p>
        <h2>Information We Collect</h2>
        <p>
          When you use our Online Health Check-up Booking Portal, we may collect
          the following information:
        </p>
        <ul>
          <li>
            Personal information such as your name, email address, and contact
            number.
          </li>
          <li>
            Demographic information such as your age, gender, and location.
          </li>
          <li>
            Information related to your health check-up appointments and medical
            history.
          </li>
```

```
</ul>
<h2>How We Use Your Information</h2>
<p>We use the collected information to:</p>
<ul>
  <li>
    Facilitate the booking and management of your health check-up
    appointments.
  </li>
  <li>
    Provide personalized recommendations and suggestions based on your
    health needs.
  </li>
  <li>Improve our services and enhance the user experience.</li>
  <li>
    Respond to your inquiries, provide support, and communicate
    important updates.
  </li>
  <li>
    Comply with legal obligations and protect the rights and safety of
    users.
  </li>
</ul>
<h2>Data Security</h2>
<p>
  We implement industry-standard security measures to protect your
  personal information from unauthorized access, alteration, or
  disclosure. However, please be aware that no method of transmission
  over the internet or electronic storage is completely secure.
</p>
<h2>Third-Party Services</h2>
<p>
  We may utilize third-party services to facilitate certain functions of
  the Online Health Check-up Booking Portal, such as payment processing
  and analytics. These third-party services have their own privacy
```

```
        policies and practices, and we encourage you to review them.
      </p>
      <h2>Updates to the Privacy Policy</h2>
      <p>
        We may update this Privacy Policy from time to time. Any changes will
        be reflected on this page, and we encourage you to review the Privacy
        Policy periodically.
      </p>
      <h2>Contact Us</h2>
      <p>
        If you have any questions or concerns regarding our Privacy Policy or
        the handling of your personal information, please contact us at
        support@healthcheckup.com.
      </p>
    </div>{" "}
  </Navbar>
 );
}
export default Policy;
```

8 Terms.js

```
import React from "react";
import Navbar from "./Navbar";
function Terms() {
 return (
  <Navbar>
   <div className="container">
    <h1 className="text-center mt-4">Terms and Conditions</h1>
    <h2>Introduction</h2>
    <p>
     These terms and conditions govern your use of the Online Health
     Check-up Booking Portal. By accessing or using our services, you agree
     to comply with these terms and conditions. If you do not agree with
     any part of these terms, you should not use our platform.
```

```
</p>
<h2>Use of the Platform</h2>
<p>
  The Online Health Check-up Booking Portal is intended for
  informational and booking purposes only. You must be at least 18 years
  old to use our services. The information provided on the platform is
  not a substitute for professional medical advice, diagnosis, or
  treatment.
</p>
<h2>Booking and Appointments</h2>
<p>
  When booking an appointment through our platform, you agree to provide
  accurate and up-to-date information. The availability of appointments
  is subject to change and is not guaranteed. We reserve the right to
  cancel or reschedule appointments as necessary.
</p>
<h2>Intellectual Property</h2>
<p>
  All intellectual property rights related to the Online Health Check-up
  Booking Portal, including but not limited to trademarks, logos, and
  content, belong to us or our licensors. You may not use, reproduce, or
  distribute any of our intellectual property without prior written
  consent.
</p>
<h2>Limitation of Liability</h2>
<p>
  We strive to provide accurate and reliable information, but we make no
  warranties or representations regarding the completeness, accuracy, or
  reliability of the information on our platform. We are not liable for
  any direct, indirect, incidental, or consequential damages arising
  from the use or inability to use our services.
</p>
<h2>Changes to the Terms and Conditions</h2>
<p>
```

```
        We reserve the right to modify or update these terms and conditions at
        any time. Any changes will be effective upon posting on this page. It
        is your responsibility to review the terms periodically. Your
        continued use of the platform after any modifications constitutes
        acceptance of the updated terms and conditions.
      </p>
      <h2>Contact Us</h2>
      <p>
        If you have any questions or concerns regarding these terms and
        conditions, please contact us at support@healthcheckup.com.
      </p>
    </div>{" "}
  </Navbar>
 );
}
export default Terms;
```

9 DoctorList.js

```
import React from "react";
import { useNavigate } from "react-router-dom";
const DoctorList = ({ doctor }) => {
 const navigate = useNavigate();
 const getRandomStars = () => {
  const minStars = 3;
  const maxStars = 5;
  const numStars =
   Math.floor(Math.random() * (maxStars - minStars + 1)) + minStars;
  return "★".repeat(numStars);
 };
 return (
  <>
   <div
    className="card m-2"
```

```
      style={{ cursor: "pointer" }}
      onClick={() =>
        navigate(`/dashboard/doctor/book-appointment/${doctor._id}`)
      }
    >
      <div className="card-header">
        Dr. {doctor.firstName} {doctor.lastName}
      </div>
      <div className="card-body">
        <p>
          <b>Specialization</b> {doctor.specialization}
        </p>
        <p>
          <b>Experience</b> {doctor.experience}
        </p>
        <p>
          <b>Fees Per Cunsaltation</b> {doctor.feesPerCunsaltation}
        </p>
        <p>
          <b>Timings</b> {doctor.timings[0]} - {doctor.timings[1]}
        </p>
        <p>
          <b>Rate:</b>
          {getRandomStars()}
        </p>
      </div>
    </div>
  </>
 );
};
export default DoctorList;
```

10 Layout.js

```
import React from "react";
```

```
import "../styles/LayoutStyles.css";
import { adminMenu, userMenu } from "./../Data/data";
import { Link, useLocation, useNavigate } from "react-router-dom";
import { useSelector } from "react-redux";
import { Badge, message } from "antd";
const Layout = ({ children }) => {
  const { user } = useSelector((state) => state.user);
  const location = useLocation();
  const navigate = useNavigate();
  // logout funtion
  const handleLogout = () => {
    localStorage.clear();
    message.success("Logout Successfully");
    navigate("/login");
  };
  // ========== doctor menu ==========
  const doctorMenu = [
    {
      name: "Dashboard",
      path: "/dashboard",
      icon: "fa-solid fa-house",
    },
    {
      name: "Appointments",
      path: "/dashboard/doctor-appointments",
      icon: "fa-solid fa-list",
    },

    {
      name: "Profile",
      path: `/dashboard/doctor/profile/${user?._id}`,
      icon: "fa-solid fa-user",
    },
  ];
```

```
// ========== doctor menu ==============
// redering menu list
const SidebarMenu = user?.isAdmin
  ? adminMenu
  : user?.isDoctor
  ? doctorMenu
  : userMenu;
return (
  <>
    <div className="main">
      <div className="layout">
        <div className="sidebar">
          <div className="logo">
            <h6 className="text-light">HEALTH CHECKUP BOOKING</h6>
            <hr />
          </div>
          <div className="menu">
            {SidebarMenu.map((menu) => {
              const isActive = location.pathname === menu.path;
              return (
                <>
                  <div className={`menu-item ${isActive && "active"}`}>
                    <i className={menu.icon}></i>
                    <Link to={menu.path}>{menu.name}</Link>
                  </div>
                </>
              );
            })}
            <div className={`menu-item `} onClick={handleLogout}>
              <i className="fa-solid fa-right-from-bracket"></i>
              <Link to="/login">Logout</Link>
            </div>
          </div>
        </div>
```

```
        <div className="content">
          <div className="header">
            <div className="header-content" style={{ cursor: "pointer" }}>
              <Badge
                count={user && user.notifcation.length}
                onClick={() => {
                  navigate("/dashboard/notification");
                }}
              >
                <i class="fa-solid fa-bell"></i>
              </Badge>
              <Link to="/dashboard/profile">{user?.name}</Link>
            </div>
          </div>
          <div className="body">{children}</div>
        </div>
      </div>
    </>
  );
};
export default Layout;
```

11 ProtectedRoute.js

```
import React, { useEffect } from "react";
import { Navigate } from "react-router-dom";
import axios from "axios";
import { useSelector, useDispatch } from "react-redux";
import { hideLoading, showLoading } from "../redux/features/alertSlice";
import { setUser } from "../redux/features/userSlice";
export default function ProtectedRoute({ children }) {
  const dispatch = useDispatch();
  const { user } = useSelector((state) => state.user);
```

```
//get user
//eslint-disable-next-line
const getUser = async () => {
  try {
    dispatch(showLoading());
    const res = await axios.post(
      "/api/v1/user/getUserData",
      { token: localStorage.getItem("token") },
      {
        headers: {
          Authorization: `Bearer ${localStorage.getItem("token")}`,
        },
      }
    );
    dispatch(hideLoading());
    if (res.data.success) {
      dispatch(setUser(res.data.data));
    } else {
      localStorage.clear();
      <Navigate to="/login" />;
    }
  } catch (error) {
    localStorage.clear();
    dispatch(hideLoading());
    console.log(error);
  }
};
useEffect(() => {
  if (!user) {
    getUser();
```

```
    }
  }, [user, getUser]);
  if (localStorage.getItem("token")) {
    return children;
  } else {
    return <Navigate to="/login" />;
  }
}
```

12 PublicRoute.js

```
import React from "react";
import { Navigate } from "react-router-dom";
export default function PublicRoute({ children }) {
  if (localStorage.getItem("token")) {
    return <Navigate to="/dashboard" />;
  } else {
    return children;
  }
}
```

13 spinner.js

```
import React from "react";
const Spinner = () => {
  return (
    <div class="d-flex justify-content-center spinner">
      <div class="spinner-border" role="status">
        <span class="visually-hidden">Loading...</span>
      </div>
    </div>
  );
```

```
};
export default Spinner;
```

14 Data.js

```
export const userMenu = [
  {
    name: "Dashboard",
    path: "/dashboard",
    icon: "fa-solid fa-house",
  },
  {
    name: "Appointments",
    path: "/dashboard/appointments",
    icon: "fa-solid fa-list",
  },
  {
    name: "Home",
    path: "/",
    icon: "fa-solid fa-home",
  },

  // {
  //   name: "Apply Doctor",
  //   path: "/dashboard/apply-doctor",
  //   icon: "fa-solid fa-user-doctor",
  // },
  // {
  //   name: "Profile",
  //   path: "/dashboard/profile",
  //   icon: "fa-solid fa-user",
  // },
];
// admin menu
```

```
export const adminMenu = [
  {
    name: "Dashboard",
    path: "/dashboard",
    icon: "fa-solid fa-house",
  },
  {
    name: "Doctors",
    path: "/dashboard/admin/doctors",
    icon: "fa-solid fa-user-doctor",
  },
  {
    name: "Users",
    path: "/dashboard/admin/users",
    icon: "fa-solid fa-user",
  },
];
```

4.1.2 Client/src/pages/admin – Folder

1 Doctors.js

```
import React, { useState, useEffect } from "react";
import Layout from "./../../components/Layout";
import axios from "axios";
import { message, Table } from "antd";
const Doctors = () => {
  const [doctors, setDoctors] = useState([]);
  //getUsers
  const getDoctors = async () => {
    try {
      const res = await axios.get("/api/v1/admin/getAllDoctors", {
        headers: {
          Authorization: `Bearer ${localStorage.getItem("token")}`,
        },
```

```
    });
    if (res.data.success) {
      setDoctors(res.data.data);
    }
  } catch (error) {
    console.log(error);
  }
};
// handle account
const handleAccountStatus = async (record, status) => {
  try {
    const res = await axios.post(
      "/api/v1/admin/changeAccountStatus",
      { doctorId: record._id, userId: record.userId, status: status },
      {
        headers: {
          Authorization: `Bearer ${localStorage.getItem("token")}`,
        },
      }
    );
    if (res.data.success) {
      message.success(res.data.message);
      window.location.reload();
    }
  } catch (error) {
    message.error("Something Went Wrong");
  }
};
useEffect(() => {
  getDoctors();
}, []);
const columns = [
  {
    title: "Name",
```

```
    dataIndex: "name",
    render: (text, record) => (
      <span>
        {record.firstName} {record.lastName}
      </span>
    ),
  },
  {
    title: "Status",
    dataIndex: "status",
  },
  {
    title: "phone",
    dataIndex: "phone",
  },
  {
    title: "Actions",
    dataIndex: "actions",
    render: (text, record) => (
      <div className="d-flex">
        {record.status === "pending" ? (
          <button
            className="btn btn-success"
            onClick={() => handleAccountStatus(record, "approved")}
          >
            Approve
          </button>
        ) : (
          <button className="btn btn-danger">Reject</button>
        )}
      </div>
    ),
  },
];
```

```
  return (
    <Layout>
      <h1 className="text-center bg-dark text-light">All Doctors</h1>
      <Table columns={columns} dataSource={doctors} />
    </Layout>
  );
};
export default Doctors;
```

2 Profile.js

```
import React, { useEffect, useState } from "react";
import Layout from "./../../components/Layout";
import axios from "axios";
import { useParams, useNavigate } from "react-router-dom";
import { Col, Form, Input, Row, TimePicker, message } from "antd";
import { useSelector, useDispatch } from "react-redux";
import { showLoading, hideLoading } from "../../redux/features/alertSlice";
import moment from "moment";
const Profile = () => {
  const { user } = useSelector((state) => state.user);
  const [doctor, setDoctor] = useState(null);
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const params = useParams();
  // update doc ==========
  //handle form
  const handleFinish = async (values) => {
    try {
      dispatch(showLoading());
      const res = await axios.post(
        "/api/v1/doctor/updateProfile",
        {
          ...values,
          userId: user._id,
```

```
        timings: [
          moment(values.timings[0]).format("HH:mm"),
          moment(values.timings[1]).format("HH:mm"),
        ],
      },
      {
        headers: {
          Authorization: `Bearer ${localStorage.getItem("token")}`,
        },
      }
    );
    dispatch(hideLoading());
    if (res.data.success) {
      message.success(res.data.message);
      navigate("/");
    } else {
      message.error(res.data.success);
    }
  } catch (error) {
    dispatch(hideLoading());
    console.log(error);
    message.error("Somthing Went Wrrong ");
  }
};
// update doc ==========
//getDOc Details
const getDoctorInfo = async () => {
  try {
    const res = await axios.post(
      "/api/v1/doctor/getDoctorInfo",
      { userId: params.id },
      {
        headers: {
          Authorization: `Bearer ${localStorage.getItem("token")}`,
```

```
        },
       }
     );
     if (res.data.success) {
       setDoctor(res.data.data);
     }
   } catch (error) {
     console.log(error);
   }
 };
 useEffect(() => {
   getDoctorInfo();
   //eslint-disable-next-line
 }, []);
 return (
   <Layout>
     <h1>Manage Profile</h1>
     {doctor && (
       <Form
         layout="vertical"
         onFinish={handleFinish}
         className="m-3"
         initialValues={{
           ...doctor,
           timings: [
             moment(doctor.timings[0], "HH:mm"),
             moment(doctor.timings[1], "HH:mm"),
           ],
         }}
       >
         <h4 classNAme="">Personal Details : </h4>
         <Row gutter={20}>
           <Col xs={24} md={24} lg={8}>
             <Form.Item
```

```
      label="First Name"
      name="firstName"
      required
      rules={[{ required: true }]}
    >
      <Input type="text" placeholder="your first name" />
    </Form.Item>
  </Col>
  <Col xs={24} md={24} lg={8}>
    <Form.Item
      label="Last Name"
      name="lastName"
      required
      rules={[{ required: true }]}
    >
      <Input type="text" placeholder="your last name" />
    </Form.Item>
  </Col>
  <Col xs={24} md={24} lg={8}>
    <Form.Item
      label="Phone No"
      name="phone"
      required
      rules={[{ required: true }]}
    >
      <Input type="text" placeholder="your contact no" />
    </Form.Item>
  </Col>
  <Col xs={24} md={24} lg={8}>
    <Form.Item
      label="Email"
      name="email"
      required
      rules={[{ required: true }]}
```

```
      >
        <Input type="email" placeholder="your email address" />
      </Form.Item>
    </Col>
    <Col xs={24} md={24} lg={8}>
      <Form.Item label="Website" name="website">
        <Input type="text" placeholder="your website" />
      </Form.Item>
    </Col>
    <Col xs={24} md={24} lg={8}>
      <Form.Item
        label="Address"
        name="address"
        required
        rules={[{ required: true }]}
      >
        <Input type="text" placeholder="your clinic address" />
      </Form.Item>
    </Col>
  </Row>
  <h4>Professional Details :</h4>
  <Row gutter={20}>
    <Col xs={24} md={24} lg={8}>
      <Form.Item
        label="Specialization"
        name="specialization"
        required
        rules={[{ required: true }]}
      >
        <Input type="text" placeholder="your specialization" />
      </Form.Item>
    </Col>
    <Col xs={24} md={24} lg={8}>
      <Form.Item
```

```
            label="Experience"
            name="experience"
            required
            rules={[{ required: true }]}
          >
            <Input type="text" placeholder="your experience" />
          </Form.Item>
        </Col>
        <Col xs={24} md={24} lg={8}>
          <Form.Item
            label="Fees Per Cunsaltation"
            name="feesPerCunsaltation"
            required
            rules={[{ required: true }]}
          >
            <Input type="text" placeholder="your contact no" />
          </Form.Item>
        </Col>
        <Col xs={24} md={24} lg={8}>
          <Form.Item label="Timings" name="timings" required>
            <TimePicker.RangePicker format="HH:mm" />
          </Form.Item>
        </Col>
        <Col xs={24} md={24} lg={8}></Col>
        <Col xs={24} md={24} lg={8}>
          <button className="btn btn-primary form-btn" type="submit">
            Update
          </button>
        </Col>
      </Row>
    </Form>
  )}
 </Layout>
);
```

```
};
export default Profile;
```

3 ApplyDoctor.js

```
import React from "react";
import Layout from "./../components/Layout";
import { Col, Form, Input, Row, TimePicker, message } from "antd";
import { useSelector, useDispatch } from "react-redux";
import { useNavigate } from "react-router-dom";
import { showLoading, hideLoading } from "../redux/features/alertSlice";
import axios from "axios";
import moment from "moment";
const ApplyDoctor = () => {
  const { user } = useSelector((state) => state.user);
  const dispatch = useDispatch();
  const navigate = useNavigate();
  //handle form
  const handleFinish = async (values) => {
    try {
      dispatch(showLoading());
      const res = await axios.post(
        "/api/v1/user/apply-doctor",
        {
          ...values,
          userId: user._id,
          timings: [
            moment(values.timings[0]).format("HH:mm"),
            moment(values.timings[1]).format("HH:mm"),
          ],
        },
        {
          headers: {
            Authorization: `Bearer ${localStorage.getItem("token")}`,
          },
```

```
      }
    );
    dispatch(hideLoading());
    if (res.data.success) {
      message.success(res.data.message);
      navigate("/");
    } else {
      message.error(res.data.message);
    }
  } catch (error) {
    dispatch(hideLoading());
    console.log(error);
    message.error("Somthing Went Wrrong ");
  }
};
return (
  <Layout>
    <h1 className="text-center bg-dark text-light">Apply Doctor</h1>
    <Form layout="vertical" onFinish={handleFinish} className="m-3">
      <h4 className="">Personal Details : </h4>
      <Row gutter={20}>
        <Col xs={24} md={24} lg={8}>
          <Form.Item
            label="First Name"
            name="firstName"
            required
            rules={[{ required: true }]}
          >
            <Input type="text" placeholder="your first name" />
          </Form.Item>
        </Col>
        <Col xs={24} md={24} lg={8}>
          <Form.Item
            label="Last Name"
```

```
    name="lastName"
    required
    rules={[{ required: true }]}
   >
    <Input type="text" placeholder="your last name" />
   </Form.Item>
  </Col>
  <Col xs={24} md={24} lg={8}>
   <Form.Item
    label="Phone No"
    name="phone"
    required
    rules={[{ required: true }]}
   >
    <Input type="text" placeholder="your contact no" />
   </Form.Item>
  </Col>
  <Col xs={24} md={24} lg={8}>
   <Form.Item
    label="Email"
    name="email"
    required
    rules={[{ required: true }]}
   >
    <Input type="email" placeholder="your email address" />
   </Form.Item>
  </Col>
  <Col xs={24} md={24} lg={8}>
   <Form.Item label="Website" name="website">
    <Input type="text" placeholder="your website" />
   </Form.Item>
  </Col>
  <Col xs={24} md={24} lg={8}>
   <Form.Item
```

```
        label="Address"
        name="address"
        required
        rules={[{ required: true }]}
      >
        <Input type="text" placeholder="your clinic address" />
      </Form.Item>
    </Col>
  </Row>
  <h4>Professional Details :</h4>
  <Row gutter={20}>
    <Col xs={24} md={24} lg={8}>
      <Form.Item
        label="Specialization"
        name="specialization"
        required
        rules={[{ required: true }]}
      >
        <Input type="text" placeholder="your specialization" />
      </Form.Item>
    </Col>
    <Col xs={24} md={24} lg={8}>
      <Form.Item
        label="Experience"
        name="experience"
        required
        rules={[{ required: true }]}
      >
        <Input type="text" placeholder="your experience" />
      </Form.Item>
    </Col>
    <Col xs={24} md={24} lg={8}>
      <Form.Item
        label="Fees Per Cunsaltation"
```

```
            name="feesPerCunsaltation"
            required
            rules={[{ required: true }]}
           >
            <Input type="text" placeholder="your contact no" />
           </Form.Item>
          </Col>
          <Col xs={24} md={24} lg={8}>
           <Form.Item label="Timings" name="timings" required>
            <TimePicker.RangePicker format="HH:mm" />
           </Form.Item>
          </Col>
          <Col xs={24} md={24} lg={8}></Col>
          <Col xs={24} md={24} lg={8}>
           <button className="btn btn-primary form-btn" type="submit">
            Submit
           </button>
          </Col>
         </Row>
        </Form>
       </Layout>
  );
};
export default ApplyDoctor;
```

4 Appointment.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import Layout from "./../components/Layout";
import moment from "moment";
import { Table } from "antd";
const Appointments = () => {
  const [appointments, setAppointments] = useState([]);
  const getAppointments = async () => {
```

```
  try {
    const res = await axios.get("/api/v1/user/user-appointments", {
      headers: {
        Authorization: `Bearer ${localStorage.getItem("token")}`,
      },
    });
    if (res.data.success) {
      setAppointments(res.data.data);
    }
  } catch (error) {
    console.log(error);
  }
};
useEffect(() => {
  getAppointments();
}, []);
const columns = [
  {
    title: "ID",
    dataIndex: "_id",
  },
  {
    title: "Date & Time",
    dataIndex: "date",
    render: (text, record) => (
      <span>
        {moment(record.date).format("DD-MM-YYYY")}  
        {moment(record.time).format("HH:mm")}
      </span>
    ),
  },
  {
    title: "Status",
    dataIndex: "status",
```

```
    },
  ];
  return (
    <Layout>
      <h1 className="text-center bg-dark text-light">Appoinmtnets Lists</h1>
      <Table columns={columns} dataSource={appointments} />
    </Layout>
  );
};
export default Appointments;
```

5 BookingPage.js

```
import React, { useState, useEffect } from "react";
import Layout from "../components/Layout";
import { useParams } from "react-router-dom";
import axios from "axios";
import { DatePicker, message, TimePicker } from "antd";
import moment from "moment";
import { useDispatch, useSelector } from "react-redux";
import { showLoading, hideLoading } from "../redux/features/alertSlice";
const BookingPage = () => {
  const { user } = useSelector((state) => state.user);
  const params = useParams();
  const [doctors, setDoctors] = useState([]);
  const [date, setDate] = useState("");
  const [time, setTime] = useState();
  const [isAvailable, setIsAvailable] = useState(false);
  const dispatch = useDispatch();
  // login user data
  const getUserData = async () => {
    try {
      const res = await axios.post(
        "/api/v1/doctor/getDoctorById",
        { doctorId: params.doctorId },
```

```
      {
        headers: {
          Authorization: "Bearer " + localStorage.getItem("token"),
        },
      }
    );
    if (res.data.success) {
      setDoctors(res.data.data);
    }
  } catch (error) {
    console.log(error);
  }
};
// ============ handle availiblity
const handleAvailability = async () => {
  try {
    dispatch(showLoading());
    const res = await axios.post(
      "/api/v1/user/booking-availbility",
      { doctorId: params.doctorId, date, time },
      {
        headers: {
          Authorization: `Bearer ${localStorage.getItem("token")}`,
        },
      }
    );
    dispatch(hideLoading());
    if (res.data.success) {
      setIsAvailable(true);
      console.log(isAvailable);
      message.success(res.data.message);
    } else {
      message.error(res.data.message);
    }
```

```
  } catch (error) {
    dispatch(hideLoading());
    console.log(error);
  }
};
// =============== booking func
const handleBooking = async () => {
  try {
    setIsAvailable(true);
    if (!date && !time) {
      return alert("Date & Time Required");
    }
    dispatch(showLoading());
    const res = await axios.post(
      "/api/v1/user/book-appointment",
      {
        doctorId: params.doctorId,
        userId: user._id,
        doctorInfo: doctors,
        userInfo: user,
        date: date,
        time: time,
      },
      {
        headers: {
          Authorization: `Bearer ${localStorage.getItem("token")}`,
        },
      }
    );
    dispatch(hideLoading());
    if (res.data.success) {
      message.success(res.data.message);
    }
  } catch (error) {
```

```
    dispatch(hideLoading());
    console.log(error);
  }
};
useEffect(() => {
  getUserData();
  //eslint-disable-next-line
}, []);
return (
  <Layout>
    <h3 className="text-center bg-dark text-light">Booking Page</h3>
    <div className="container m-2">
      {doctors && (
        <div className="text-center">
          <div className="d-flex justify-content-evenly">
            <h5 className="text-light">
              Dr.{doctors.firstName} {doctors.lastName}
            </h5>
            <h5 className="text-light">
              Fees : {doctors.feesPerCunsaltation}
            </h5>
            <h5 className="text-light">
              Timings : {doctors.timings && doctors.timings[0]} -{" "}
              {doctors.timings && doctors.timings[1]}{" "}
            </h5>
          </div>
          <div className="d-flex flex-column w-50">
            <DatePicker
              aria-required={"true"}
              className="mt-3"
              format="DD-MM-YYYY"
              onChange={(value) => {
                setDate(moment(value).format("DD-MM-YYYY"));
              }}
```

```
          />
          <TimePicker
            aria-required={"true"}
            format="HH:mm"
            className="mt-3"
            onChange={(value) => {
              setTime(moment(value).format("HH:mm"));
            }}
          />
          <button
            className="btn btn-primary mt-2"
            onClick={handleAvailability}
          >
            Check Availability
          </button>
          <button className="btn btn-dark mt-2" onClick={handleBooking}>
            Book Now
          </button>
        </div>
      </div>
    )}
  </div>
 </Layout>
 );
};
export default BookingPage;
```

6 Home.js

```
import React from "react";
import Navbar from "../components/HomeLayout/Navbar";
import generalCheckupsImage from "./../images/genrealcheckup.jpg";
import specializedCheckupsImage from "./../images/SpecializedCheck-ups.jpg";
// import dentalCheckupsImage from "./../images/DentalCheck-ups.jpg";
import insuranceImage from "./../images/insurance.jpg";
```

```
// import healthTipsImage from "./../images/health-tips.jpg";
const Home = () => {
  const sections = [
    {
      title: "General Check-ups",
      imageUrl: generalCheckupsImage,
      description: "Book an appointment for general health check-ups.",
    },
    {
      title: "Specialized Check-ups",
      imageUrl: specializedCheckupsImage,
      description: "Book an appointment for specialized health check-ups.",
    },
  ];
  const insurance = [
    {
      title: "Insurance",
      imageUrl: insuranceImage,
      description: "Learn about our insurance coverage options.",
    },
  ];
  return (
    <Navbar>
      <div className="container">
        <h1 className="mt-5 mb-4">
          Welcome to the Online Health Check-up Booking Portal!
        </h1>
        <div className="row">
          {sections.map((section) => (
            <div key={section.title} className="col-md-4">
              <div className="card mb-4">
                <img
                  height={250}
                  src={section.imageUrl}
```

```
            alt={section.title}
            className="card-img-top"
          />
          <div className="card-body">
            <h2 className="card-title">{section.title}</h2>
            <p className="card-text">{section.description}</p>
            <a href="/login" className="btn btn-info text-light">
              Book Now
            </a>
          </div>
        </div>
      </div>
    ))}
    {insurance.map((section) => (
      <div key={section.title} className="col-md-4">
        <div className="card mb-4">
          <img
            height={250}
            src={section.imageUrl}
            alt={section.title}
            className="card-img-top"
          />
          <div className="card-body">
            <h2 className="card-title">{section.title}</h2>
            <p className="card-text">{section.description}</p>
            <a href="/insurance" className="btn btn-info text-light">
              Learn More
            </a>
          </div>
        </div>
      </div>
    ))}
  </div>
</div>
```

```
    </Navbar>
  );
};
export default Home;
```

7 HomePage.js

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import Layout from "./../components/Layout";
import { Row } from "antd";
import DoctorList from "../components/DoctorList";
const HomePage = () => {
  const [doctors, setDoctors] = useState([]);
  // login user data
  const getUserData = async () => {
    try {
      const res = await axios.get(
        "/api/v1/user/getAllDoctors",
        {
          headers: {
            Authorization: "Bearer " + localStorage.getItem("token"),
          },
        }
      );
      if (res.data.success) {
        setDoctors(res.data.data);
      }
    } catch (error) {
      console.log(error);
    }
  };
```

```
  useEffect(() => {
    getUserData();
  }, []);
  return (
    <Layout>
      <h1 className="text-center bg-dark text-light">Dashboard</h1>
      <h4 className="text-center bg-light m-2">
        Select the doctor based on your check-up and doctor's specialization
      </h4>
      <Row>
        {doctors && doctors.map((doctor) => <DoctorList doctor={doctor} />)}
      </Row>
    </Layout>
  );
};
export default HomePage;
```

8 Insurance.js

```
import React from "react";
import Navbar from "../components/HomeLayout/Navbar";
import insuranceImage from "./../images/insurance.jpg";
const Insurance = () => {
  return (
    <Navbar>
      <div className="container">
        <h1 className="mt-5 mb-4 text-center">Insurance Coverage</h1>
        <div className="row">
          <div className="col-md-6">
            <img
              height={300}
```

```
    src={insuranceImage}
    alt="Insurance"
    className="img-fluid mb-4"
   />
  </div>
  <div className="col-md-6">
   <h2 className="mb-3">Why Choose Our Insurance Coverage?</h2>
   <p>
    At our Online Health Check-up Booking Portal, we offer
    comprehensive insurance coverage for your medical needs. Here are
    some reasons to choose our insurance:
   </p>
   <ul className="list-group">
    <li className="list-group-item">
     <i className="bi bi-check-circle-fill text-success me-2"></i>
     Flexible plans tailored to your requirements
    </li>
    <li className="list-group-item">
     <i className="bi bi-check-circle-fill text-success me-2"></i>
     Extensive network of healthcare providers
    </li>
    <li className="list-group-item">
     <i className="bi bi-check-circle-fill text-success me-2"></i>
     Easy claims process for hassle-free reimbursement
    </li>
    <li className="list-group-item">
     <i className="bi bi-check-circle-fill text-success me-2"></i>
     24/7 customer support for any assistance needed
    </li>
    <li className="list-group-item">
```

```
            <i className="bi bi-check-circle-fill text-success me-2"></i>
            Competitive premiums and discounts
          </li>
        </ul>
      </div>
    </div>
    <center>
      {" "}
      To Enquire more details about insurance or to claim any insurance
      please contact <b>insurance@healthcheckup.com</b>{" "}
    </center>
  </div>
 </Navbar>
 );
};
export default Insurance;
```

9 Login.js

```
import React from "react";
import "../styles/RegiserStyles.css";
import { Form, Input, message } from "antd";
import { useDispatch } from "react-redux";
import { showLoading, hideLoading } from "../redux/features/alertSlice";
import { Link, useNavigate } from "react-router-dom";
import axios from "axios";
import Navbar from "../components/HomeLayout/Navbar";
const Login = () => {
  const navigate = useNavigate();
  const dispatch = useDispatch();
  //form handler
```

```
const onfinishHandler = async (values) => {
  try {
    dispatch(showLoading());
    const res = await axios.post("/api/v1/user/login", values);
    window.location.reload();
    dispatch(hideLoading());
    if (res.data.success) {
      localStorage.setItem("token", res.data.token);
      message.success("Login Successfully");
      navigate("/dashboard");
    } else {
      message.error(res.data.message);
    }
  } catch (error) {
    dispatch(hideLoading());
    console.log(error);
    message.error("something went wrong");
  }
};
return (
  <Navbar>
    <div className="form-container image1">
      <Form
        layout="vertical"
        onFinish={onfinishHandler}
        className="register-form bg-info "
      >
        <h3 className="text-center">Login</h3>
        <Form.Item label="Email" name="email">
          <Input placeholder="Enter Your Email" type="email" required />
```

```
        </Form.Item>
        <Form.Item label="Password" name="password">
          <Input placeholder="Enter Your Password" type="password" required />
        </Form.Item>
        Not Yet Registered?
        <Link to="/register" className="text-light m-2">
          Register
        </Link>
        <br />
        Forget Password ?
        <Link to="/forget-password" className="text-light m-2">
          Reset
        </Link>
        <br />
        <br />
        <button className="btn btn-primary" type="submit">
          Login
        </button>
      </Form>
    </div>
  </Navbar>
 );
};
export default Login;
```

10 NotificationPage.js

```
import React from "react";
import Layout from "./../components/Layout";
import { message, Tabs } from "antd";
import { useSelector, useDispatch } from "react-redux";
```

```
import { showLoading, hideLoading } from "../redux/features/alertSlice";
import { useNavigate } from "react-router-dom";
import axios from "axios";
const NotificationPage = () => {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const { user } = useSelector((state) => state.user);
  //   handle read notification
  const handleMarkAllRead = async () => {
    try {
      dispatch(showLoading());
      const res = await axios.post(
        "/api/v1/user/get-all-notification",
        {
          userId: user._id,
        },
        {
          headers: {
            Authorization: `Bearer ${localStorage.getItem("token")}`,
          },
        }
      );
      dispatch(hideLoading());
      if (res.data.success) {
        message.success(res.data.message);
      } else {
        message.error(res.data.message);
      }
    } catch (error) {
      dispatch(hideLoading());
```

```
    console.log(error);
    message.error("somthing went wrong");
  }
};
// delete notifications
const handleDeleteAllRead = async () => {
  try {
    dispatch(showLoading());
    const res = await axios.post(
      "/api/v1/user/delete-all-notification",
      { userId: user._id },
      {
        headers: {
          Authorization: `Bearer ${localStorage.getItem("token")}`,
        },
      }
    );
    dispatch(hideLoading());
    if (res.data.success) {
      message.success(res.data.message);
    } else {
      message.error(res.data.message);
    }
  } catch (error) {
    dispatch(hideLoading());
    console.log(error);
    message.error("Somthing Went Wrong In Ntifications");
  }
};
return (
```

```
<Layout>
  <h4 className="p-3 text-center">Notification Page</h4>
  <Tabs>
    <Tabs.TabPane tab="unRead" key={0}>
      <div className="d-flex justify-content-end">
        <h4 className="p-2" onClick={handleMarkAllRead}>
          Mark All Read
        </h4>
      </div>
      {user?.notifcation.map((notificationMgs) => (
        <div className="card" style={{ cursor: "pointer" }}>
          <div
            className="card-text"
            onClick={() => navigate(notificationMgs.onClickPath)}
          >
            {notificationMgs.message}
          </div>
        </div>
      ))}
    </Tabs.TabPane>
    <Tabs.TabPane tab="Read" key={1}>
      <div className="d-flex justify-content-end">
        <h4
          className="p-2 text-primary"
          style={{ cursor: "pointer" }}
          onClick={handleDeleteAllRead}
        >
          Delete All Read
        </h4>
      </div>
```

```
        {user?.seennotification.map((notificationMgs) => (
          <div className="card" style={{ cursor: "pointer" }}>
            <div
              className="card-text"
              onClick={() => navigate(notificationMgs.onClickPath)}
            >
              {notificationMgs.message}
            </div>
          </div>
        ))}
      </Tabs.TabPane>
    </Tabs>
  </Layout>
 );
};
export default NotificationPage;
```

11 Register.js

```
import React from "react";
import "../styles/RegiserStyles.css";
import { Form, Input, message } from "antd";
import axios from "axios";
import { Link, useNavigate } from "react-router-dom";
import { useDispatch } from "react-redux";
import { showLoading, hideLoading } from "../redux/features/alertSlice";
import Navbar from "../components/HomeLayout/Navbar";
import image from "./../images/genrealcheckup.jpg";
const Register = () => {
  const navigate = useNavigate();
  const dispatch = useDispatch();
```

```
//form handler
const onfinishHandler = async (values) => {
  try {
    dispatch(showLoading());
    const res = await axios.post("/api/v1/user/register", values);
    dispatch(hideLoading());
    if (res.data.success) {
      message.success("Register Successfully!");
      navigate("/login");
    } else {
      message.error(res.data.message);
    }
  } catch (error) {
    dispatch(hideLoading());
    console.log(error);
    message.error("Something Went Wrong");
  }
};
return (
  <>
    <Navbar>
      <div className="form-container image">
        <Form
          layout="vertical"
          onFinish={onfinishHandler}
          className="register-form bg-info"
        >
          <h3 className="text-center">Register</h3>
          <Form.Item label="Name" name="name">
            <Input placeholder="Enter Your Name" type="text" required />
```

```
          </Form.Item>
          <Form.Item label="Email" name="email">
            <Input placeholder="Enter Your Email" type="email" required />
          </Form.Item>
          <Form.Item label="Number" name="number">
            <Input placeholder="Enter Your Number" type="text" required />
          </Form.Item>
          <Form.Item label="Password" name="password">
            <Input
              placeholder="Enter Your Password"
              type="password"
              required
            />
          </Form.Item>
          Already user ?
          <Link to="/login" className="m-2 text-light">
            login
          </Link>
          <br />
          <br />
          <button className="btn btn-primary" type="submit">
            Register
          </button>
        </Form>
      </div>
    </Navbar>
  </>
 );
};
export default Register;
```

4.1.3 Client/src/redux/features – Folder

1 Alertslice.js

```
import { createSlice } from "@reduxjs/toolkit";
export const alertSlice = createSlice({
  name: "alerts",
  initialState: {
    loading: false,
  },
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    hideLoading: (state) => {
      state.loading = false;
    },
  },
});
export const { showLoading, hideLoading } = alertSlice.actions;
```

2 userslice.js

```
import { createSlice } from "@reduxjs/toolkit";
export const userSlice = createSlice({
  name: "user",
  initialState: {
    user: null,
  },
  reducers: {
    setUser: (state, action) => {
      state.user = action.payload;
```

```
    },
  },
});
export const { setUser } = userSlice.actions;
```

3 Store.js

```
import { configureStore } from "@reduxjs/toolkit";
import { alertSlice } from "./features/alertSlice";
import { userSlice } from "./features/userSlice";
export default configureStore({
  reducer: {
    alerts: alertSlice.reducer,
    user: userSlice.reducer,
  },
});
```

4 App.js

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import HomePage from "./pages/HomePage";
import Login from "./pages/Login";
import Register from "./pages/Register";
import { useSelector } from "react-redux";
import Spinner from "./components/Spinner";
import ProtectedRoute from "./components/ProtectedRoute";
import PublicRoute from "./components/PublicRoute";
import ApplyDoctor from "./pages/ApplyDoctor";
import NotificationPage from "./pages/NotificationPage";
import Users from "./pages/admin/Users";
import Doctors from "./pages/admin/Doctors";
import Profile from "./pages/doctor/Profile";
```

```
import BookingPage from "./pages/BookingPage";
import Appointments from "./pages/Appointments";
import DoctorAppointments from "./pages/doctor/DoctorAppointments";
import Home from "./pages/Home";
import About from "./components/HomeLayout/About";
import Contact from "./components/HomeLayout/Contact";
import Terms from "./components/HomeLayout/Terms";
import Policy from "./components/HomeLayout/Policy";
import Pagenotfound from "./components/HomeLayout/Pagenotfound";
import Forget from "./pages/Forget";
import Insurance from "./pages/Insurance";
function App() {
 const { loading } = useSelector((state) => state.alerts);
 return (
  <>
   <BrowserRouter>
    {loading ? (
     <Spinner />
    ) : (
     <Routes>
      <Route
       path="/dashboard/apply-doctor"
       element={
        <ProtectedRoute>
         <ApplyDoctor />
        </ProtectedRoute>
       }
      />
      <Route
```

```
path="/dashboard/admin/users"
            element={
              <ProtectedRoute>
                <Users />
              </ProtectedRoute>
            }
          />
          <Route
            path="/dashboard/admin/doctors"
            element={
              <ProtectedRoute>
                <Doctors />
              </ProtectedRoute>
            }
          />
          <Route
            path="/dashboard/doctor/profile/:id"
            element={
              <ProtectedRoute>
                <Profile />
              </ProtectedRoute>
            }
          />
          <Route
            path="/dashboard/doctor/book-appointment/:doctorId"
            element={
              <ProtectedRoute>
                <BookingPage />
              </ProtectedRoute>
```

```
}
      />
      <Route
        path="/dashboard/notification"
        element={
          <ProtectedRoute>
            <NotificationPage />
          </ProtectedRoute>
        }
      />
      <Route
        path="/login"
        element={
          <PublicRoute>
            <Login />
          </PublicRoute>
        }
      />
      <Route
        path="/forget-password"
        element={
          <PublicRoute>
            <Forget />
          </PublicRoute>
        }
      />

      <Route
        path="/register"
```

```
element={
          <PublicRoute>
            <Register />
          </PublicRoute>
        }
      />
      <Route
        path="/dashboard/appointments"
        element={
          <ProtectedRoute>
            <Appointments />
          </ProtectedRoute>
        }
      />
      <Route
        path="/dashboard/doctor-appointments"
        element={
          <ProtectedRoute>
            <DoctorAppointments />
          </ProtectedRoute>
        }
      />
      <Route
        path="/dashboard"
        element={
          <ProtectedRoute>
            <HomePage />
          </ProtectedRoute>
        }
```

```
          />
          <Route path="/insurance" element={<Insurance />} />
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
          <Route path="/contact" element={<Contact />} />
          <Route path="/terms" element={<Terms />} />
          <Route path="/policy" element={<Policy />} />
          <Route path="*" element={<Pagenotfound />} />
        </Routes>
      )}
    </BrowserRouter>
  </>
 );
}

export default App;
```

4.2 Back-End :

4.2.1 Config/db.js

```
const mongoose = require("mongoose");
const colors = require("colors");
const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGO_URL);
    console.log(`Mongodb connected ${mongoose.connection.host}`.bgGreen.white);
  } catch (error) {
    console.log(`Mongodb Server Issue ${error}`.bgRed.white);
  }
};
module.exports = connectDB;
```

4.2.2 controllers – Folder

1 Adminctrl.js

```
const doctorModel = require("../models/doctorModel");
const userModel = require("../models/userModels");
const getAllUsersController = async (req, res) => {
  try {
    const users = await userModel.find({});
    res.status(200).send({
      success: true,
      message: "users data list",
      data: users,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
```

```
      success: false,
      message: "erorr while fetching users",
      error,
    });
  }
};
const getAllDoctorsController = async (req, res) => {
  try {
    const doctors = await doctorModel.find({});
    res.status(200).send({
      success: true,
      message: "Doctors Data list",
      data: doctors,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "error while getting doctors data",
      error,
    });
  }
};
// doctor account status
const changeAccountStatusController = async (req, res) => {
  try {
    const { doctorId, status } = req.body;
    const doctor = await doctorModel.findByIdAndUpdate(doctorId, { status });
    const user = await userModel.findOne({ _id: doctor.userId });
    const notifcation = user.notifcation;
```

```
    notifcation.push({
      type: "doctor-account-request-updated",
      message: `Your Doctor Account Request Has ${status} `,
      onClickPath: "/notification",
    });
    user.isDoctor = status === "approved" ? true : false;
    await user.save();
    res.status(201).send({
      success: true,
      message: "Account Status Updated",
      data: doctor,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Eror in Account Status",
      error,
    });
  }
};
module.exports = {
  getAllDoctorsController,
  getAllUsersController,
  changeAccountStatusController,
};
```

2 Docotrctrl.js

```
const appointmentModel = require("../models/appointmentModel");
const doctorModel = require("../models/doctorModel");
```

```
const userModel = require("../models/userModels");
const getDoctorInfoController = async (req, res) => {
  try {
    const doctor = await doctorModel.findOne({ userId: req.body.userId });
    res.status(200).send({
      success: true,
      message: "doctor data fetch success",
      data: doctor,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error in Fetching Doctor Details",
    });
  }
};
// update doc profile
const updateProfileController = async (req, res) => {
  try {
    const doctor = await doctorModel.findOneAndUpdate(
      { userId: req.body.userId },
      req.body
    );
    res.status(201).send({
      success: true,
      message: "Doctor Profile Updated",
      data: doctor,
    });
```

```
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "Doctor Profile Update issue",
      error,
    });
  }
};
//get single docotor
const getDoctorByIdController = async (req, res) => {
  try {
    const doctor = await doctorModel.findOne({ _id: req.body.doctorId });
    res.status(200).send({
      success: true,
      message: "Sigle Doc Info Fetched",
      data: doctor,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Erro in Single docot info",
    });
  }
};
const doctorAppointmentsController = async (req, res) => {
  try {
    const doctor = await doctorModel.findOne({ userId: req.body.userId });
```

```
    const appointments = await appointmentModel.find({
      doctorId: doctor._id,
    });
    res.status(200).send({
      success: true,
      message: "Doctor Appointments fetch Successfully",
      data: appointments,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error in Doc Appointments",
    });
  }
};
const updateStatusController = async (req, res) => {
  try {
    const { appointmentsId, status } = req.body;
    const appointments = await appointmentModel.findByIdAndUpdate(
      appointmentsId,
      { status }
    );
    const user = await userModel.findOne({ _id: appointments.userId });
    const notifcation = user.notifcation;
    notifcation.push({
      type: "status-updated",
      message: `your appointment has been updated ${status}`,
      onCLickPath: "/doctor-appointments",
```

```
    });
    await user.save();
    res.status(200).send({
      success: true,
      message: "Appointment Status Updated",
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error In Update Status",
    });
  }
};
module.exports = {
  getDoctorInfoController,
  updateProfileController,
  getDoctorByIdController,
  doctorAppointmentsController,
  updateStatusController,
};
```

3 Userctrl.js

```
const userModel = require("../models/userModels");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");
const doctorModel = require("../models/doctorModel");
const appointmentModel = require("../models/appointmentModel");
const moment = require("moment");
```

```
//register callback
const registerController = async (req, res) => {
  try {
    const exisitingUser = await userModel.findOne({ email: req.body.email });
    if (exisitingUser) {
      return res
        .status(200)
        .send({ message: "User Already Exist", success: false });
    }
    function validateGmail(email) {
      // Regular expression pattern to match Gmail address
      var gmailRegex = /^[a-zA-Z0-9._%+-]+@gmail\.com$/;
      // Test if the email matches the Gmail pattern
      if (gmailRegex.test(email)) {
        return true; // Valid Gmail address
      }
      return false; // Invalid Gmail address
    }
    function validatePassword(password) {
      // Regular expression patterns for different password requirements
      var uppercaseRegex = /[A-Z]/;
      var lowercaseRegex = /[a-z]/;
      var numberRegex = /[0-9]/;
      var symbolRegex = /[!@#$%^&*()]/;
      // Check if the password meets the length requirement
      if (password.length < 8 || password.length > 10) {
        return false; // Invalid password length
      }
      // Test if the password meets all the requirements
      if (
```

```
    uppercaseRegex.test(password) &&
    lowercaseRegex.test(password) &&
    numberRegex.test(password) &&
    symbolRegex.test(password)
  ) {
    return true; // Valid password
  }
  return false; // Invalid password
}
if (req.body.number.length !== 10) {
  return res
    .status(200)
    .send({ message: "Enter 10 Digits Number only", success: false });
}
if (validateGmail(req.body.email) == false) {
  return res
    .status(200)
    .send({ message: "Enter Correct Gmail", success: false });
}
if (validatePassword(req.body.password) == false) {
  return res
    .status(200)
    .send({ message: "Enter an valid password", success: false });
}
const password = req.body.password;
const salt = await bcrypt.genSalt(10);
const hashedPassword = await bcrypt.hash(password, salt);
req.body.password = hashedPassword;
const newUser = new userModel(req.body);
await newUser.save();
```

```
    res.status(201).send({ message: "Register Sucessfully", success: true });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: `Register Controller ${error.message}`,
    });
  }
};
// login callback
const loginController = async (req, res) => {
  try {
    const user = await userModel.findOne({ email: req.body.email });
    if (!user) {
      return res
        .status(200)
        .send({ message: "user not found", success: false });
    }
    const isMatch = await bcrypt.compare(req.body.password, user.password);
    if (!isMatch) {
      return res
        .status(200)
        .send({ message: "Invalid Email or Password", success: false });
    }
    const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
      expiresIn: "1d",
    });
    res.status(200).send({ message: "Login Success", success: true, token });
  } catch (error) {
    console.log(error);
```

```
    res.status(500).send({ message: `Error in Login CTRL ${error.message}` });
  }
};
const authController = async (req, res) => {
  try {
    const user = await userModel.findById({ _id: req.body.userId });
    user.password = undefined;
    if (!user) {
      return res.status(200).send({
        message: "User not found",
        success: false,
      });
    } else {
      res.status(200).send({
        success: true,
        data: user,
      });
    }
  } catch (error) {
    console.log(error);
    res.status(500).send({
      message: "auth error",
      success: false,
      error,
    });
  }
};
// Apply Doctor CTRL
const applyDoctorController = async (req, res) => {
  try {
```

```
    const newDoctor = await doctorModel({ ...req.body, status: "pending" });
    await newDoctor.save();
    const adminUser = await userModel.findOne({ isAdmin: true });
    const notifcation = adminUser.notifcation;
    notifcation.push({
      type: "apply-doctor-request",
      message: `${newDoctor.firstName} ${newDoctor.lastName} Has Applied For A Doctor
Account`,
      data: {
        doctorId: newDoctor._id,
        name: newDoctor.firstName + " " + newDoctor.lastName,
        onClickPath: "/admin/docotrs",
      },
    });
    await userModel.findByIdAndUpdate(adminUser._id, { notifcation });
    res.status(201).send({
      success: true,
      message: "Doctor Account Applied Successfully",
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error WHile Applying For Doctotr",
    });
  }
};
//notification ctrl
const getAllNotificationController = async (req, res) => {
  try {
```

```
    const user = await userModel.findOne({ _id: req.body.userId });
    const seennotification = user.seennotification;
    const notifcation = user.notifcation;
    seennotification.push(...notifcation);
    user.notifcation = [];
    user.seennotification = notifcation;
    const updatedUser = await user.save();
    res.status(200).send({
      success: true,
      message: "all notification marked as read",
      data: updatedUser,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      message: "Error in notification",
      success: false,
      error,
    });
  }
};
// delete notifications
const deleteAllNotificationController = async (req, res) => {
  try {
    const user = await userModel.findOne({ _id: req.body.userId });
    user.notifcation = [];
    user.seennotification = [];
    const updatedUser = await user.save();
    updatedUser.password = undefined;
    res.status(200).send({
```

```
      success: true,
      message: "Notifications Deleted successfully",
      data: updatedUser,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      message: "unable to delete all notifications",
      error,
    });
  }
};
//GET ALL DOC
const getAllDocotrsController = async (req, res) => {
  try {
    const doctors = await doctorModel.find({ status: "approved" });
    res.status(200).send({
      success: true,
      message: "Docots Lists Fetched Successfully",
      data: doctors,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Errro WHile Fetching DOcotr",
    });
  }
```

```
};
//BOOK APPOINTMENT
const bookeAppointmnetController = async (req, res) => {
  try {
    req.body.date = moment(req.body.date, "DD-MM-YYYY").toISOString();
    req.body.time = moment(req.body.time, "HH:mm").toISOString();
    req.body.status = "pending";
    const newAppointment = new appointmentModel(req.body);
    await newAppointment.save();
    const user = await userModel.findOne({ _id: req.body.doctorInfo.userId });
    user.notifcation.push({
      type: "New-appointment-request",
      message: `A nEw Appointment Request from ${req.body.userInfo.name}`,
      onCLickPath: "/user/appointments",
    });
    await user.save();
    res.status(200).send({
      success: true,
      message: "Appointment Book succesfully",
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error While Booking Appointment",
    });
  }
};
// booking bookingAvailabilityController
```

```
const bookingAvailabilityController = async (req, res) => {
  try {
    const date = moment(req.body.date, "DD-MM-YY").toISOString();
    const fromTime = moment(req.body.time, "HH:mm")
      .subtract(1, "hours")
      .toISOString();
    const toTime = moment(req.body.time, "HH:mm").add(1, "hours").toISOString();
    const doctorId = req.body.doctorId;
    const appointments = await appointmentModel.find({
      doctorId,
      date,
      time: {
        $gte: fromTime,
        $lte: toTime,
      },
    });
    if (appointments.length > 0) {
      return res.status(200).send({
        message: "Appointments not Availibale at this time",
        success: false,
      });
    } else {
      return res.status(200).send({
        success: true,
        message: "Appointments available",
      });
    }
  } catch (error) {
    console.log(error);
    res.status(500).send({
```

```
      success: false,
      error,
      message: "Error In Booking",
    });
  }
};
const userAppointmentsController = async (req, res) => {
  try {
    const appointments = await appointmentModel.find({
      userId: req.body.userId,
    });
    res.status(200).send({
      success: true,
      message: "Users Appointments Fetch SUccessfully",
      data: appointments,
    });
  } catch (error) {
    console.log(error);
    res.status(500).send({
      success: false,
      error,
      message: "Error In User Appointments",
    });
  }
};
module.exports = {
  loginController,
  registerController,
  authController,
  applyDoctorController,
```

```
  getAllNotificationController,
  deleteAllNotificationController,
  getAllDocotrsController,
  bookeAppointmnetController,
  bookingAvailabilityController,
  userAppointmentsController,
};
```

4.2.4 Middleware – Folder

1 Middleware.js

```
const JWT = require("jsonwebtoken");
module.exports = async (req, res, next) => {
  try {
    const token = req.headers["authorization"].split(" ")[1];
    JWT.verify(token, process.env.JWT_SECRET, (err, decode) => {
      if (err) {
        return res.status(200).send({
          message: "Auth Fialed",
          success: false,
        });
      } else {
        req.body.userId = decode.id;
        next();
      }
    });
  } catch (error) {
    console.log(error);
    res.status(401).send({
      message: "Auth Failed",
      success: false,
```

```
    });
  }
};
```

4.2.5 Models – Folder

1 AppointmrntModel.js

```
const mongoose = require("mongoose");
const appointmentSchema = new mongoose.Schema(
  {
    userId: {
      type: String,
      required: true,
    },
    doctorId: {
      type: String,
      required: true,
    },
    doctorInfo: {
      type: String,
      required: true,
    },
    userInfo: {
      type: String,
      required: true,
    },
    date: {
      type: String,
      required: true,
    },
    status: {
```

```
      type: String,
      required: true,
      default: "pending",
    },
    time: {
      type: String,
      required: true,
    },
  },
  { timestamps: true }
);
const appointmentModel = mongoose.model("appointments", appointmentSchema);
module.exports = appointmentModel;
```

2 DoctorModel.js

```
const mongoose = require("mongoose");
const doctorSchema = new mongoose.Schema(
  {
    userId: {
      type: String,
    },
    firstName: {
      type: String,
      required: [true, "first name is required"],
    },
    lastName: {
      type: String,
      required: [true, "last name is required"],
    },
    phone: {
```

```
  type: String,
  required: [true, "phone no is required"],
 },
 email: {
  type: String,
  required: [true, "email is required"],
 },
 website: {
  type: String,
 },
 address: {
  type: String,
  required: [true, "address is required"],
 },
 specialization: {
  type: String,
  required: [true, "specialization is require"],
 },
 experience: {
  type: String,
  required: [true, "experience is required"],
 },
 feesPerCunsaltation: {
  type: Number,
  required: [true, "fee is required"],
 },
 status: {
  type: String,
  default: "pending",
 },
```

```
    timings: {
      type: Object,
      required: [true, "work timing is required"],
    },
  },
  { timestamps: true }
);
const doctorModel = mongoose.model("doctors", doctorSchema);
module.exports = doctorModel;
```

3 UserModel.js

```
const mongoose = require("mongoose");
const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, "Name is require"],
  },
  email: {
    type: String,
    required: [true, "Email is require"],
  },
  number: {
    type: String,
    required: [true, "Number is require"],
  },
  password: {
    type: String,
    required: [true, "Password is require"],
  },
  isAdmin: {
```

```
    type: Boolean,
    default: false,
  },
  isDoctor: {
    type: Boolean,
    default: false,
  },
  notifcation: {
    type: Array,
    default: [],
  },
  seennotification: {
    type: Array,
    default: [],
  },
});
const userModel = mongoose.model("users", userSchema);
module.exports = userModel;
```

4.2.6 Route – Folder

1 AdminRoute.js

```
const express = require("express");
const {
  getAllUsersController,
  getAllDoctorsController,
  changeAccountStatusController,
} = require("../controllers/adminCtrl");
const authMiddleware = require("../middlewares/authMiddleware");
const router = express.Router();
//GET METHOD || USERS
```

```
router.get("/getAllUsers", authMiddleware, getAllUsersController);
//GET METHOD || DOCTORS
router.get("/getAllDoctors", authMiddleware, getAllDoctorsController);
//POST ACCOUNT STATUS
router.post(
  "/changeAccountStatus",
  authMiddleware,
  changeAccountStatusController
);
module.exports = router;
```

2 DoctorRoute.js

```
const express = require("express");
const {
  getDoctorInfoController,
  updateProfileController,
  getDoctorByIdController,
  doctorAppointmentsController,
  updateStatusController,
} = require("../controllers/doctorCtrl");
const authMiddleware = require("../middlewares/authMiddleware");
const router = express.Router();
//POST SINGLE DOC INFO
router.post("/getDoctorInfo", authMiddleware, getDoctorInfoController);
//POST UPDATE PROFILE
router.post("/updateProfile", authMiddleware, updateProfileController);
//POST  GET SINGLE DOC INFO
router.post("/getDoctorById", authMiddleware, getDoctorByIdController);
//GET Appointments
router.get(
```

```
  "/doctor-appointments",
  authMiddleware,
  doctorAppointmentsController
);
//POST Update Status
router.post("/update-status", authMiddleware, updateStatusController);
module.exports = router;
```

3 UserRoute.js

```
const express = require("express");
const {
  loginController,
  registerController,
  authController,
  applyDoctorController,
  getAllNotificationController,
  deleteAllNotificationController,
  getAllDocotrsController,
  bookeAppointmnetController,
  bookingAvailabilityController,
  userAppointmentsController,
} = require("../controllers/userCtrl");
const authMiddleware = require("../middlewares/authMiddleware");
//router onject
const router = express.Router();
//routes
//LOGIN || POST
router.post("/login", loginController);
//REGISTER || POST
router.post("/register", registerController);
```

```
//Auth || POST
router.post("/getUserData", authMiddleware, authController);


//APply Doctor || POST
router.post("/apply-doctor", authMiddleware, applyDoctorController);
//Notifiaction  Doctor || POST
router.post(
  "/get-all-notification",
  authMiddleware,
  getAllNotificationController
);
//Notifiaction  Doctor || POST
router.post(
  "/delete-all-notification",
  authMiddleware,
  deleteAllNotificationController
);
//GET ALL DOC
router.get("/getAllDoctors", authMiddleware, getAllDocotrsController);
//BOOK APPOINTMENT
router.post("/book-appointment", authMiddleware, bookeAppointmnetController);
//Booking Avliability
router.post(
  "/booking-availbility",
  authMiddleware,
  bookingAvailabilityController
);
//Appointments List
router.get("/user-appointments", authMiddleware, userAppointmentsController);
module.exports = router;
```

4 server.js

```
const express = require("express");
const colors = require("colors");
const moragan = require("morgan");
const dotenv = require("dotenv");
const connectDB = require("./config/db");
//dotenv conig
dotenv.config();
//mongodb connection
connectDB();
//rest obejct
const app = express();
//middlewares
app.use(express.json());
app.use(moragan("dev"));
//routes
app.use("/api/v1/user", require("./routes/userRoutes"));
app.use("/api/v1/admin", require("./routes/adminRoutes"));
app.use("/api/v1/doctor", require("./routes/doctorRoutes"));
//port
const port = process.env.PORT || 8080;
//listen port
app.listen(port, () => {
  console.log(`Server Running  on port ${process.env.PORT}`.bgCyan.white);
});
```

5. IMPLEMENTATION

5.1 Screenshots

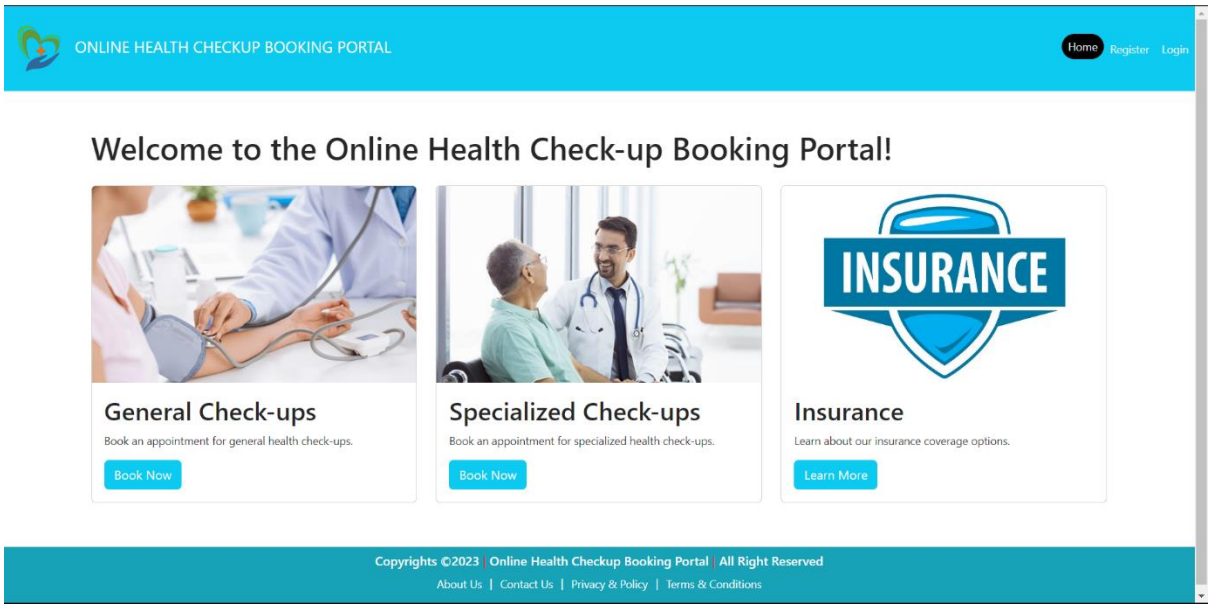

Fig. No. 5.1: Home Page

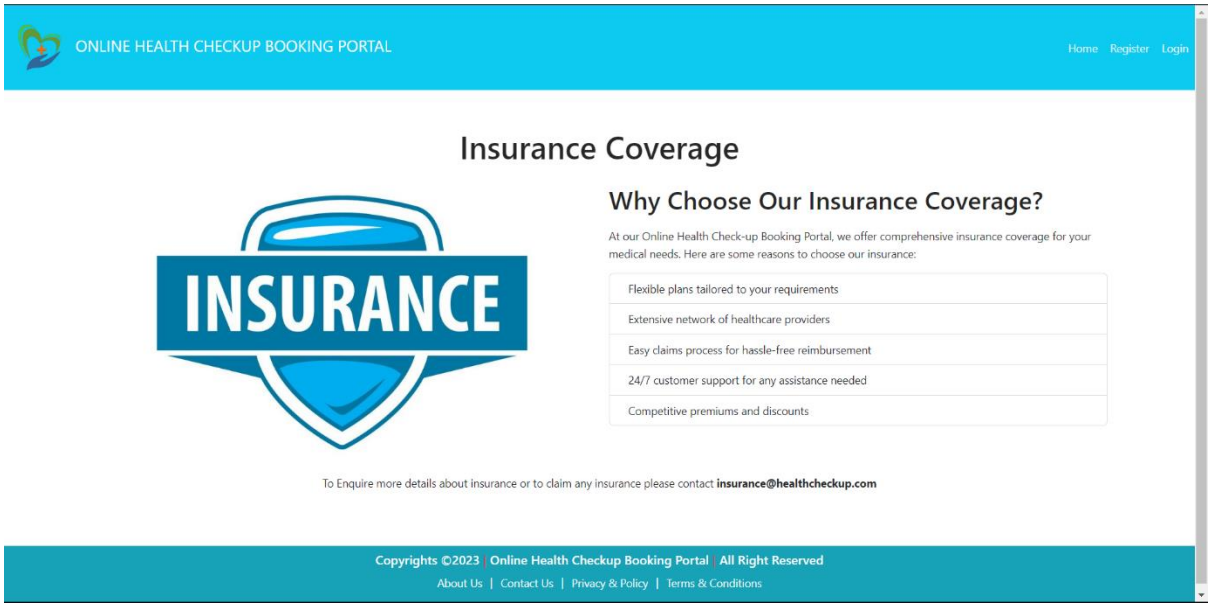

Fig. No. 5.2: Insurance Page

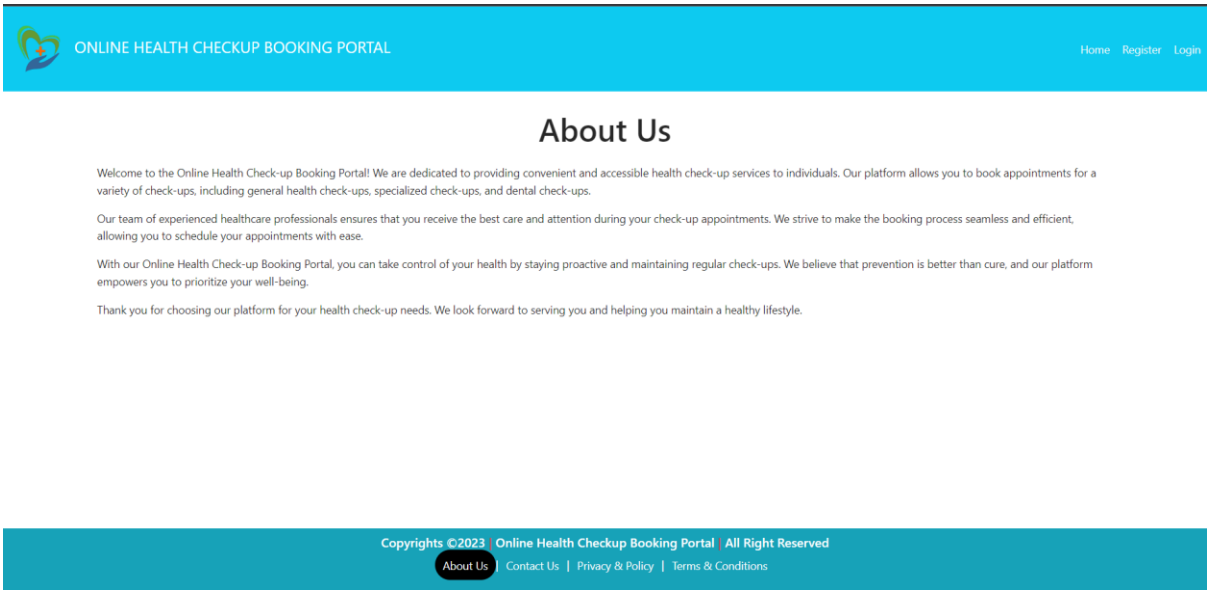

Fig. No. 5.3: About us Page

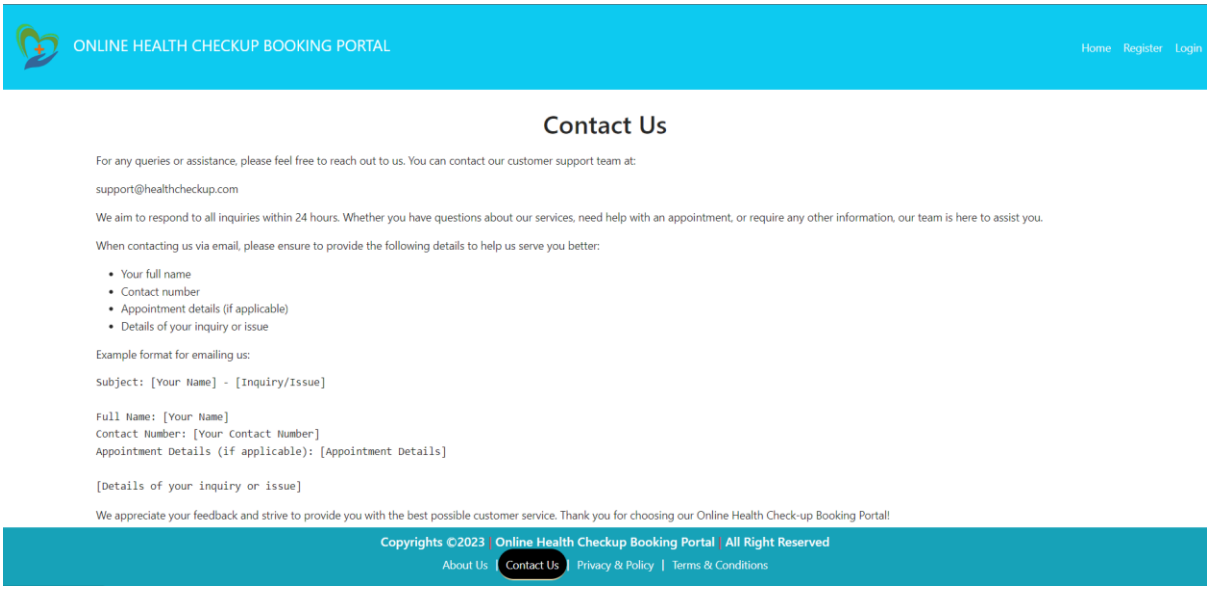

Fig. No. 5.4: Contact us Page

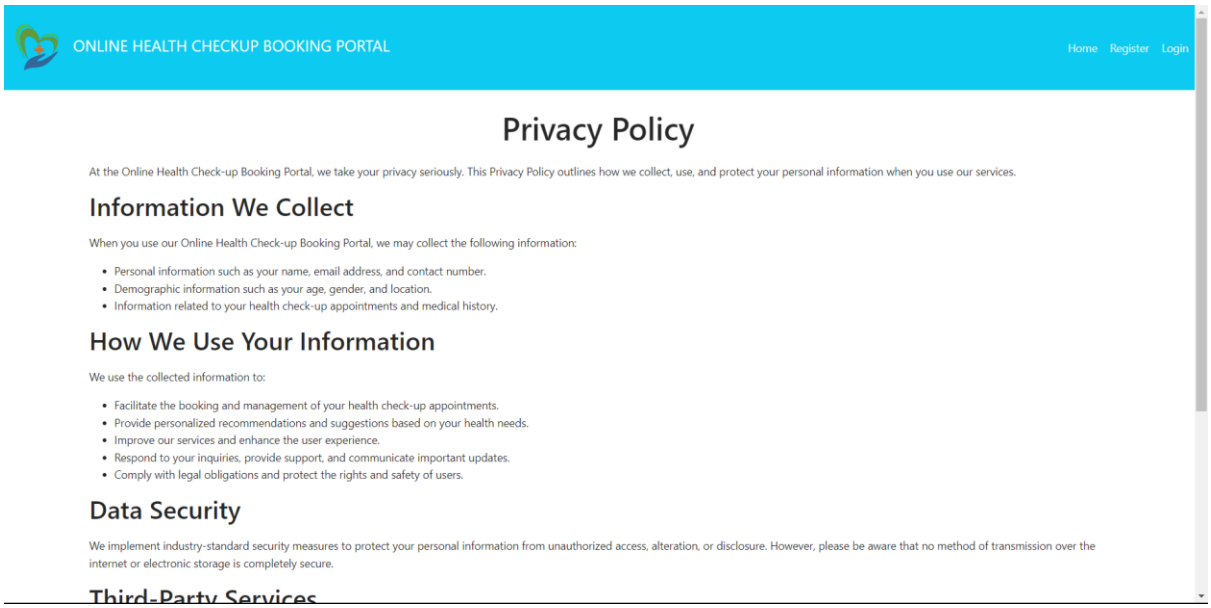

Fig. No. 5.5: Privacy Policy Page

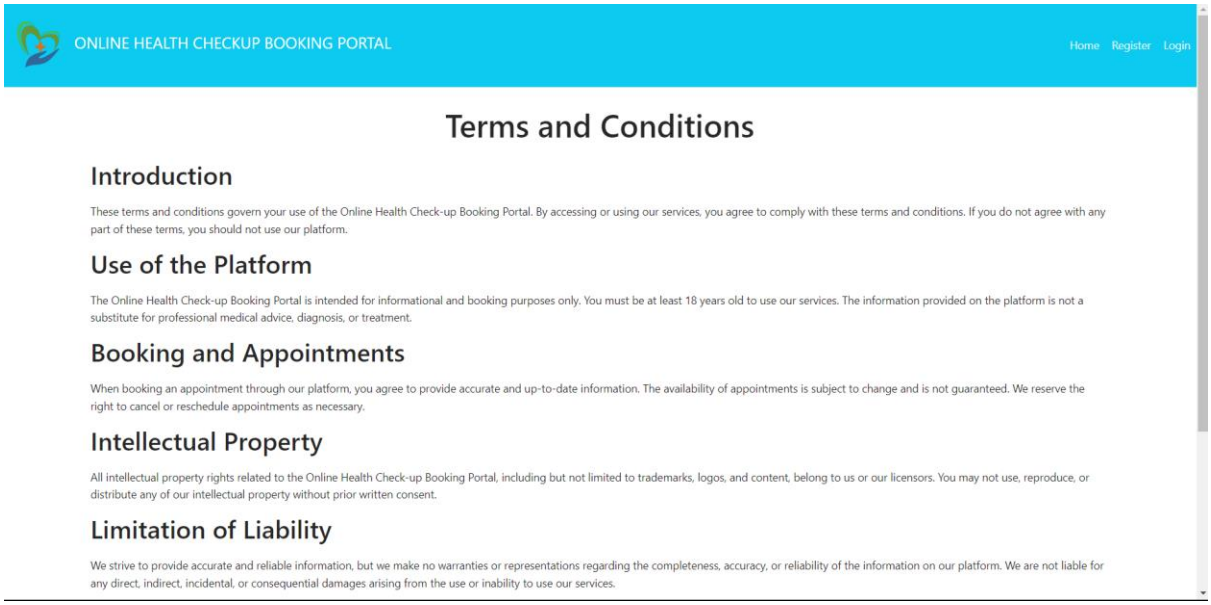

Fig. No. 5.6: Terms and Conditions Page

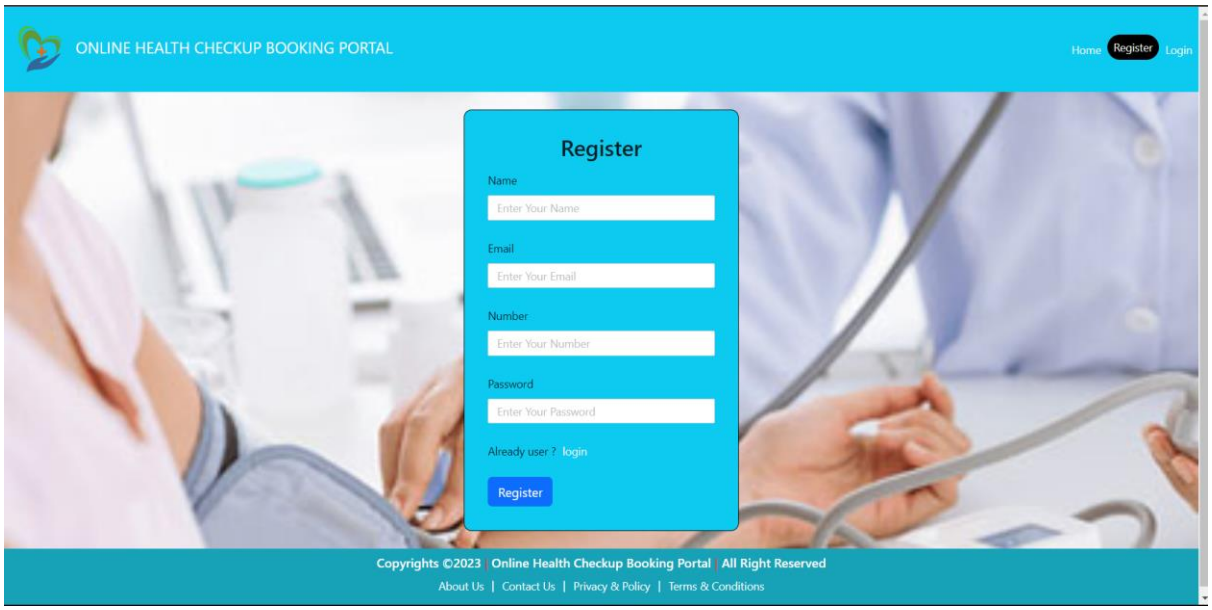

Fig. No. 5.7: Register Page

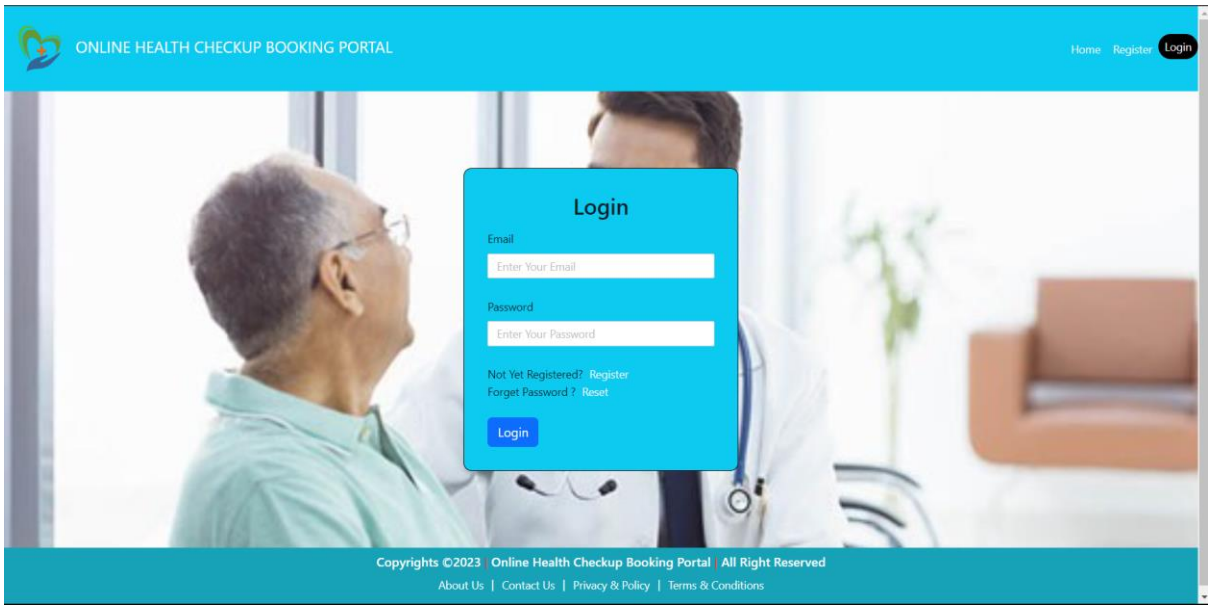

Fig. No. 5.8: Login Page

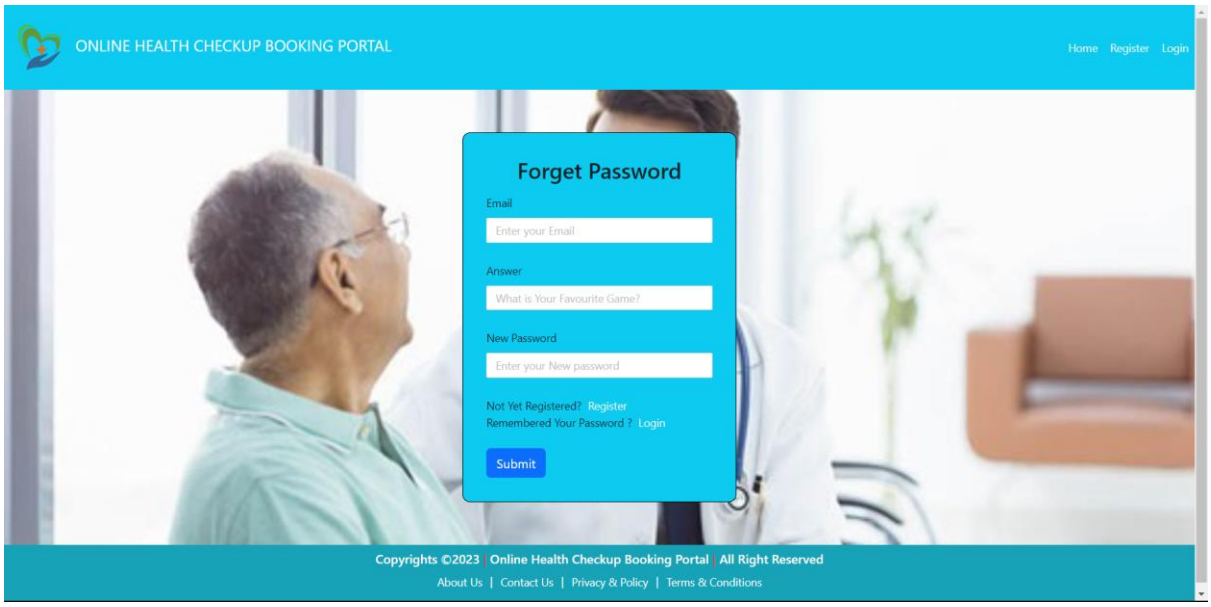

Fig. No. 5.9: Forget Password Page

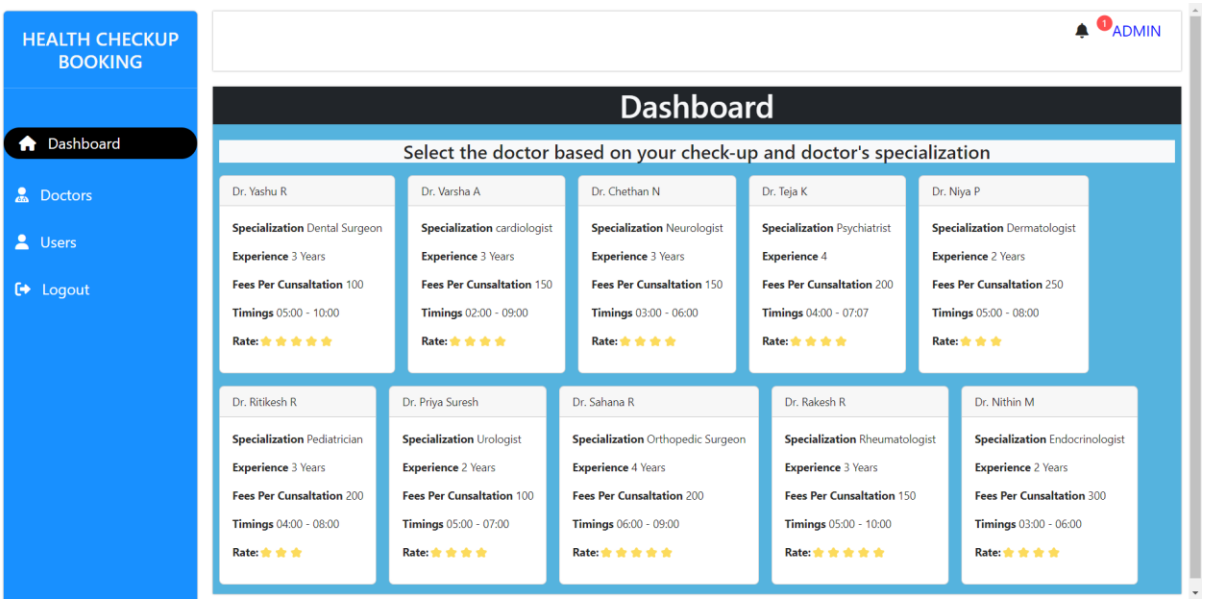

Fig. No. 5.10: Admin's Dashboard Page

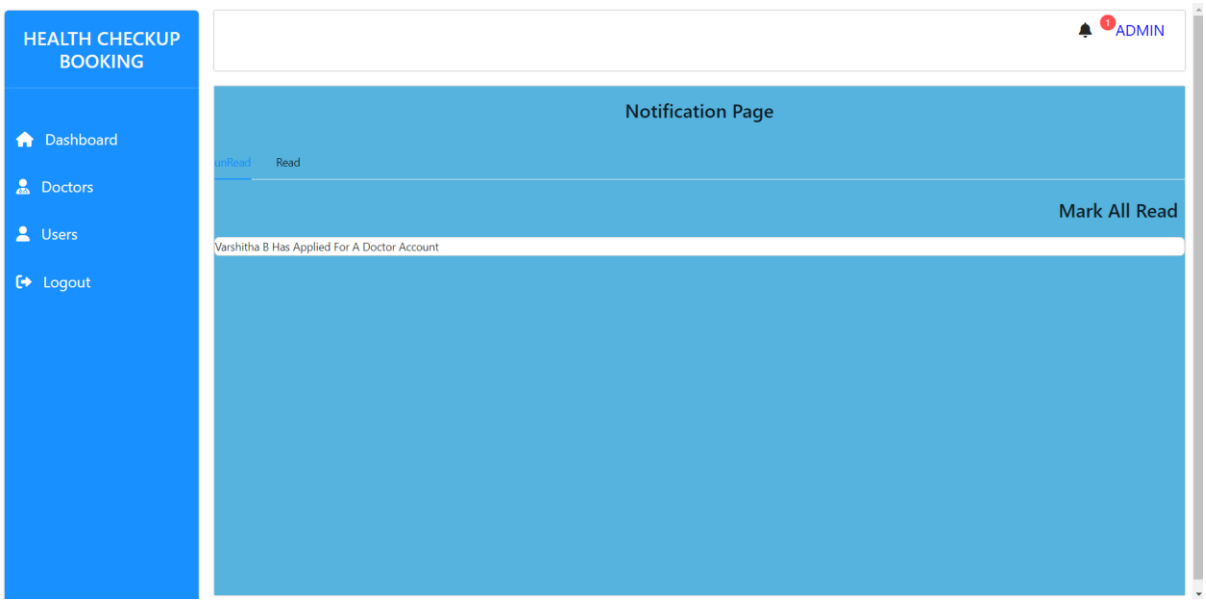

Fig. No. 5.11: Notification Page

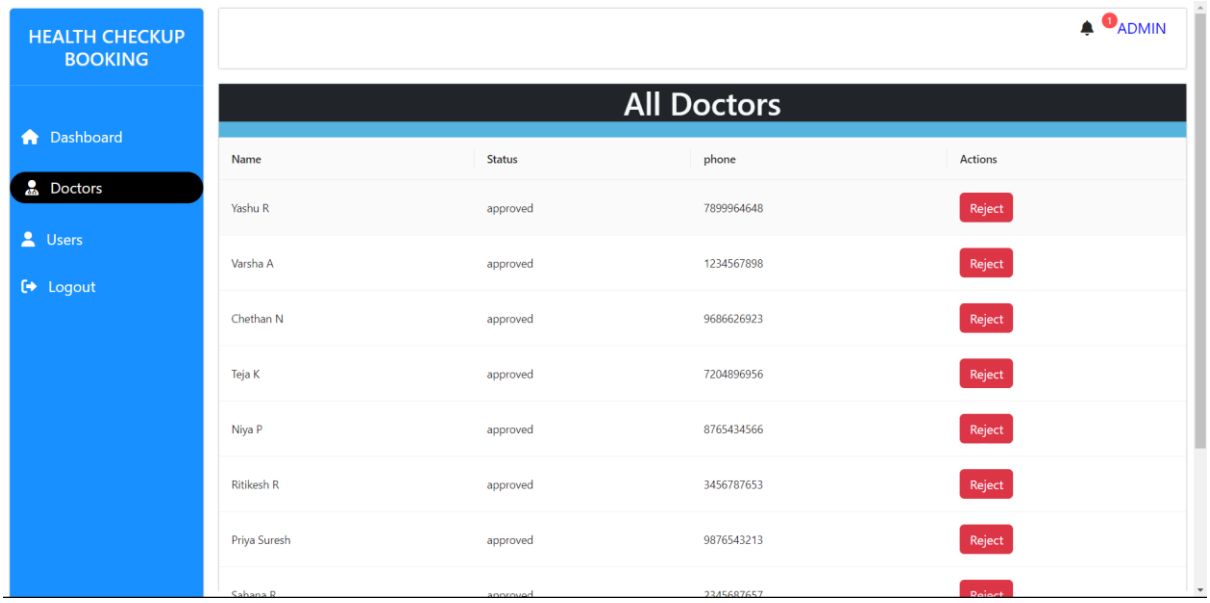

Fig. No. 5.12: Admin's Doctors Page

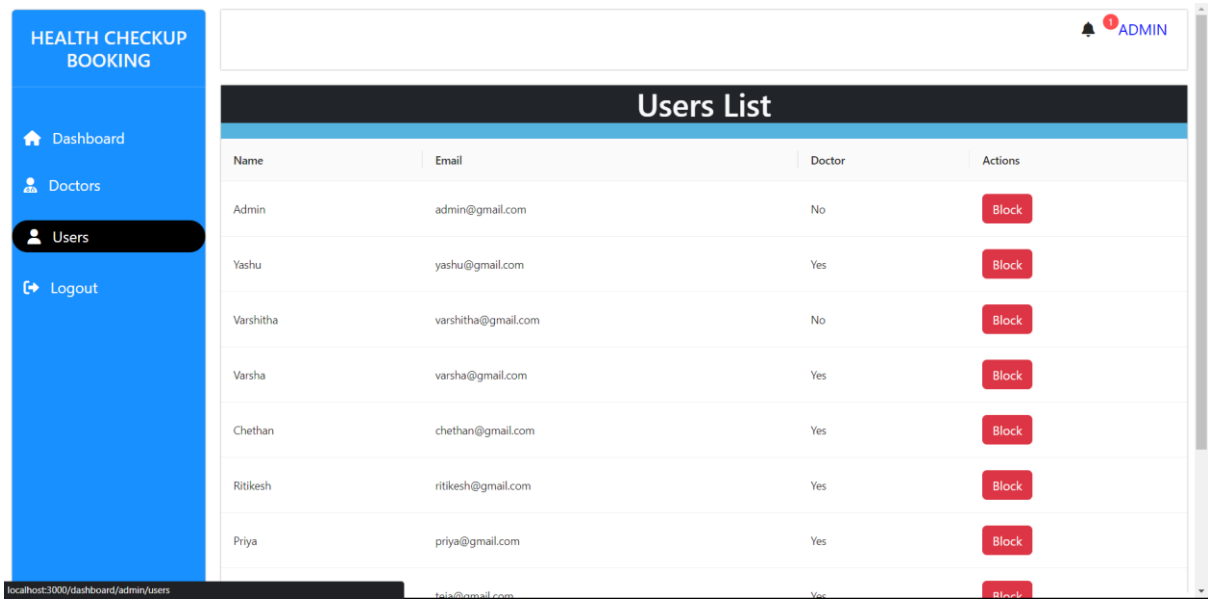

Fig. No. 5.13: Admin's Users Page

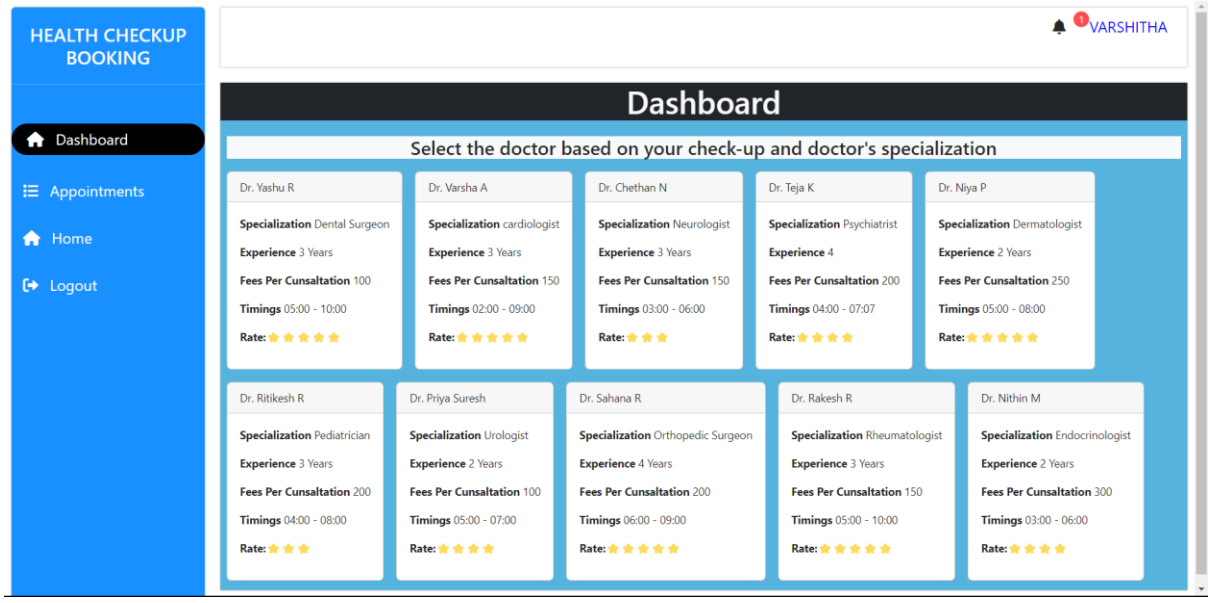

Fig. No. 5.14: User's Dashboard Page

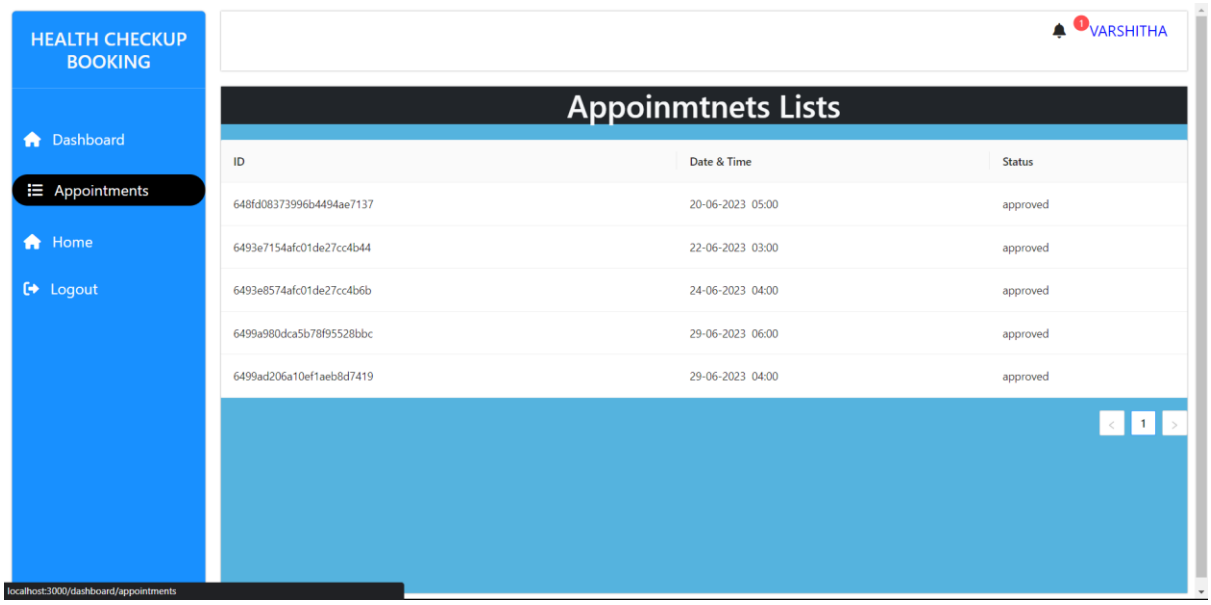

Fig. No. 5.15: User's Appointments Page

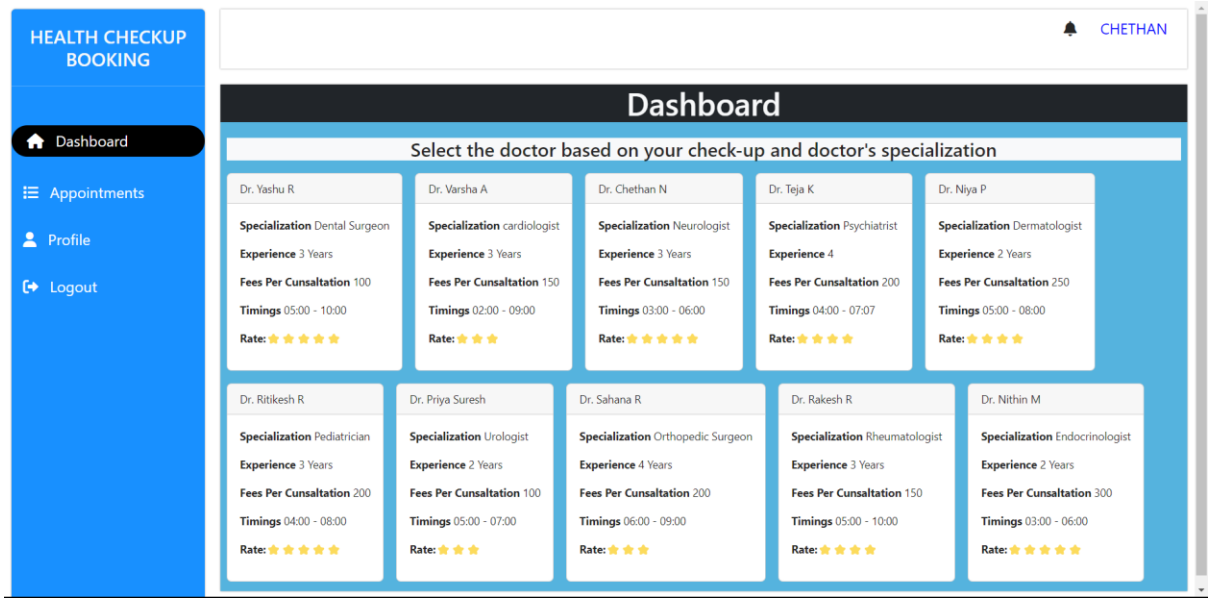

Fig. No. 5.16: Doctor's Dashboard Page

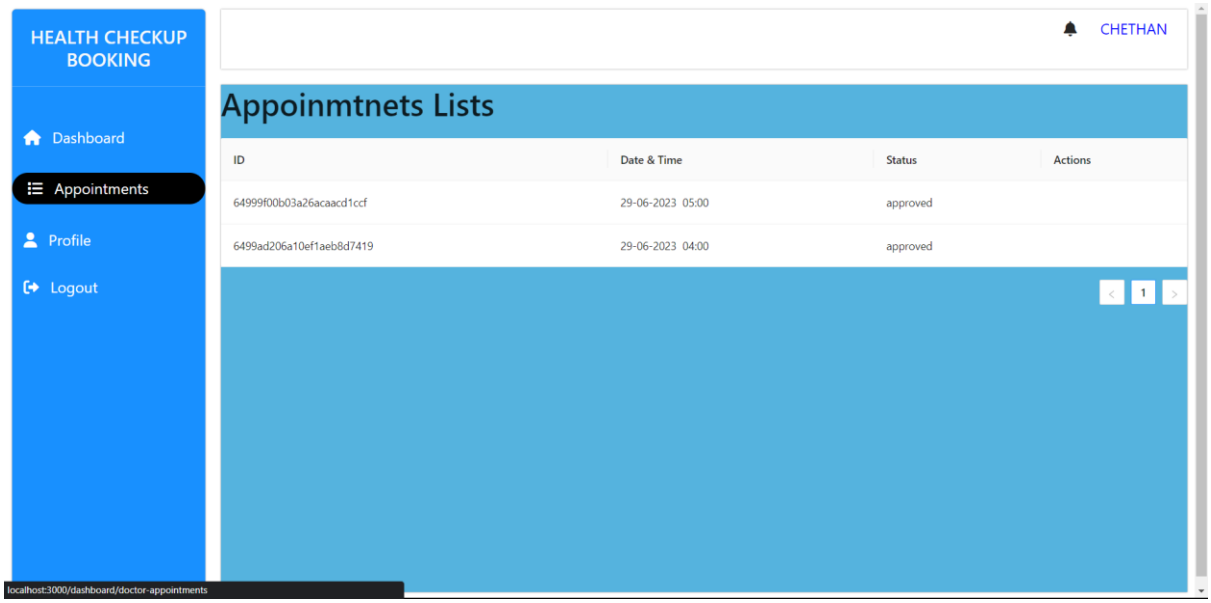

Fig. No. 5.17: Doctor's Appointments Page

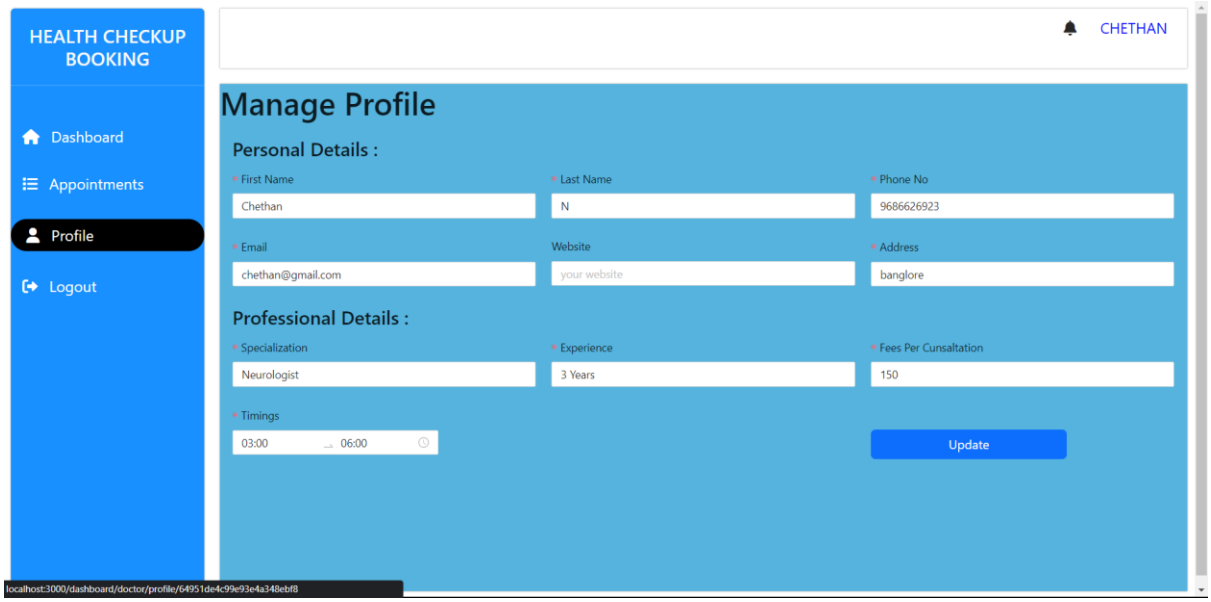

Fig. No. 5.18: Doctor's Profile Page

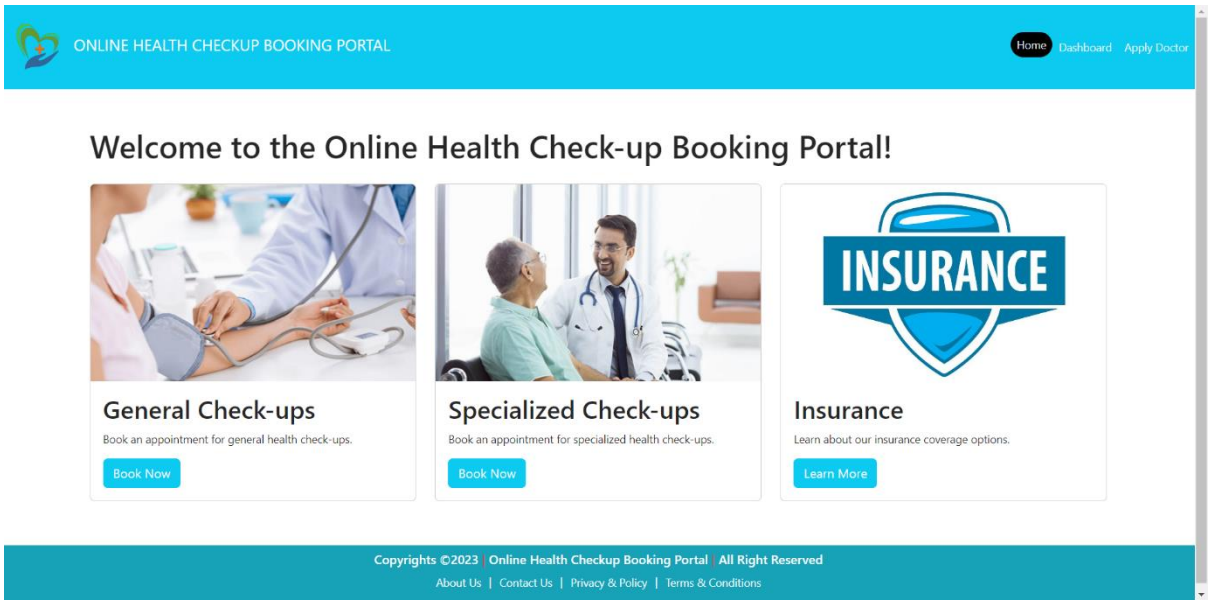

Fig. No. 5.19: Home Page After Login

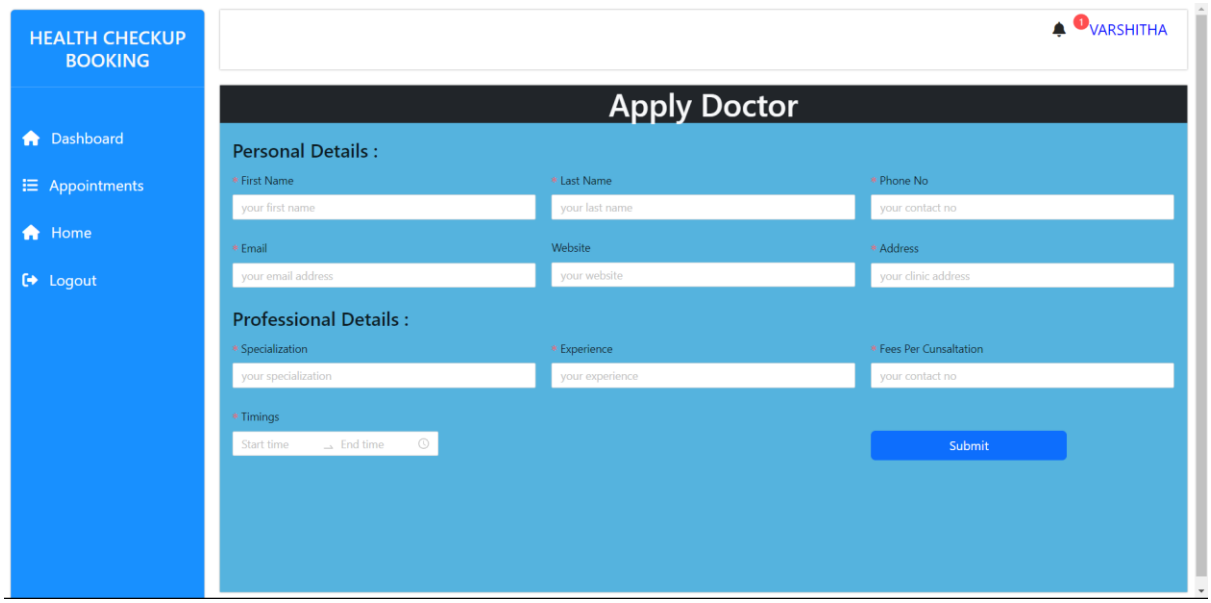

Fig. No. 5.20: Apply for Doctor Page

6. TESTING

In general, testing is finding out how well something works. In terms of human beings, testing tells what level of knowledge or skill has been acquired. In computer hardware and software development, testing is used at key checkpoints in the overall process to determine whether objectives are being met.

6.1 System Testing

System testing is a type of software testing that evaluates the complete and fully integrated system to verify that it meets the specified requirements and works as intended. System testing is performed after integration testing and focuses on verifying the system's behavior in a real-world scenario, including performance, security, and other non-functional requirements. The objective of system testing is to validate that the system meets the business and user requirements, and to identify and fix any defects or issues before the system is released to the end-users. System testing can be performed manually or with the help of automated tools, and typically involves testing the system with realistic data and user scenarios to simulate real-world usage. System testing is an important step in the software development process as it ensures that the system meets the quality standards and is ready for deployment.

6.2 Integration Testing

Integration testing is a type of software testing where individual units or components of the software are combined and tested as a group to verify the interactions between them. The purpose of integration testing is to validate that the interfaces between the components work correctly, and that the components function as intended when integrated into the larger system. Integration testing is performed after unit testing and focuses on verifying the interaction and communication between components, as well as the overall functionality of the system as a whole. Integration testing can be performed manually or with the help of automated tools and typically involves testing the system with realistic data and user scenarios to simulate real-world usage. Integration testing is an important step in the software development process as it helps to identify and resolve any issues with the integration of components before the system is tested at the system level.

6.3 Unit Testing

Unit testing is a type of software testing where individual units or components of the software are tested in isolation from the rest of the system. The purpose of unit testing is to validate that each unit of the system performs as intended and meets its functional and design specifications. Unit tests are typically automated and are run frequently during the development process to catch any issues early on. This type of testing is performed by developers and focuses on small, isolated pieces of code, such as individual functions or methods. Unit testing helps to identify and fix bugs early in the development process, which can save time and resources later on in the software development cycle. Additionally, unit tests serve as a documentation of the expected behavior of the system, making it easier for developers to maintain and update the code over time.

6.4 Test Cases

Test Case – 01 (Register Form)

SI NO.	Description	Expected Result	Actual Result	Status
01	Register Form	All Valid Details	Empty Name	Fail
		All Valid Details	Empty / Invalid Number	Fail
		All Valid Details	Empty / Invalid Email	Fail
		All Valid Details	Empty / Invalid Password	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.1

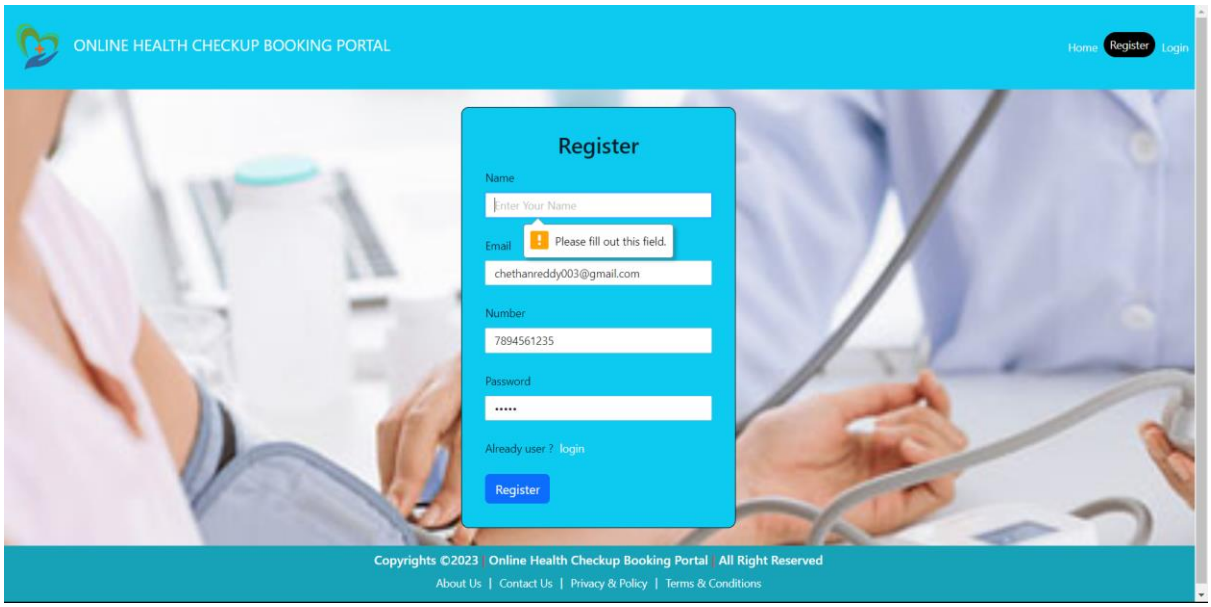

Fig. No. 6.1: Empty Name

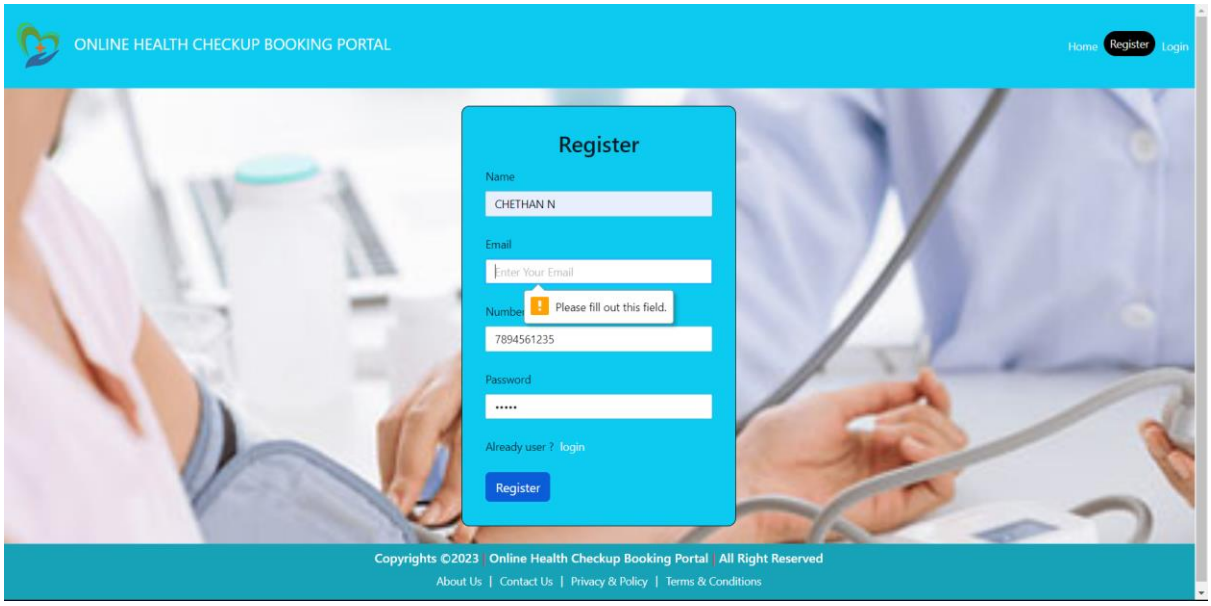

Fig. No. 6.2: Empty Email

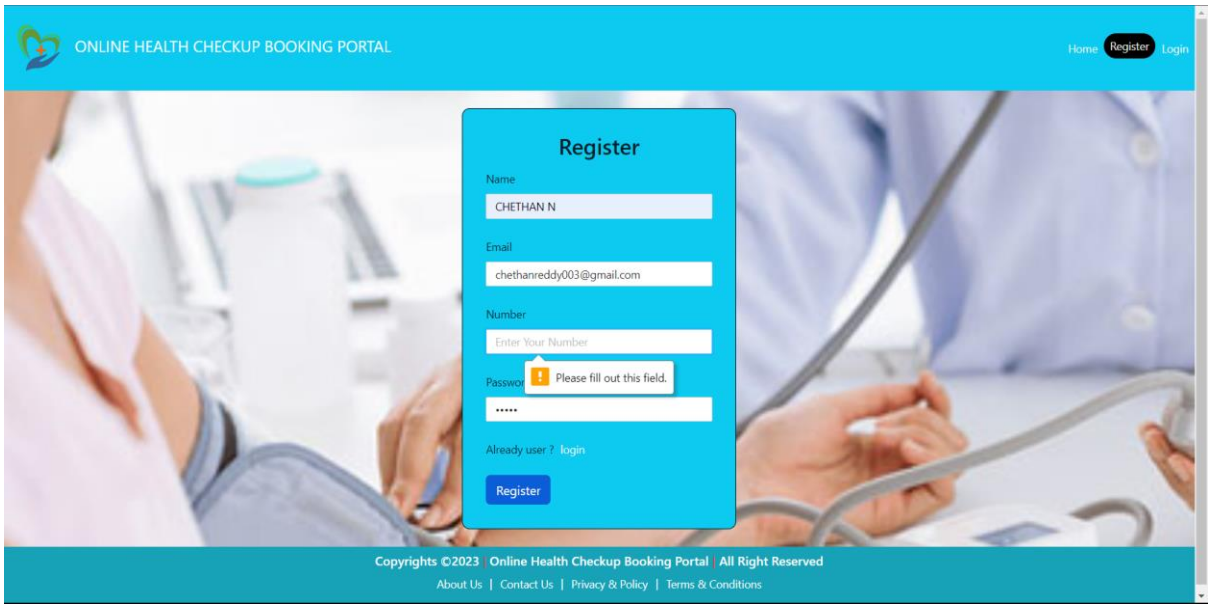

Fig. No. 6.3: Empty Number

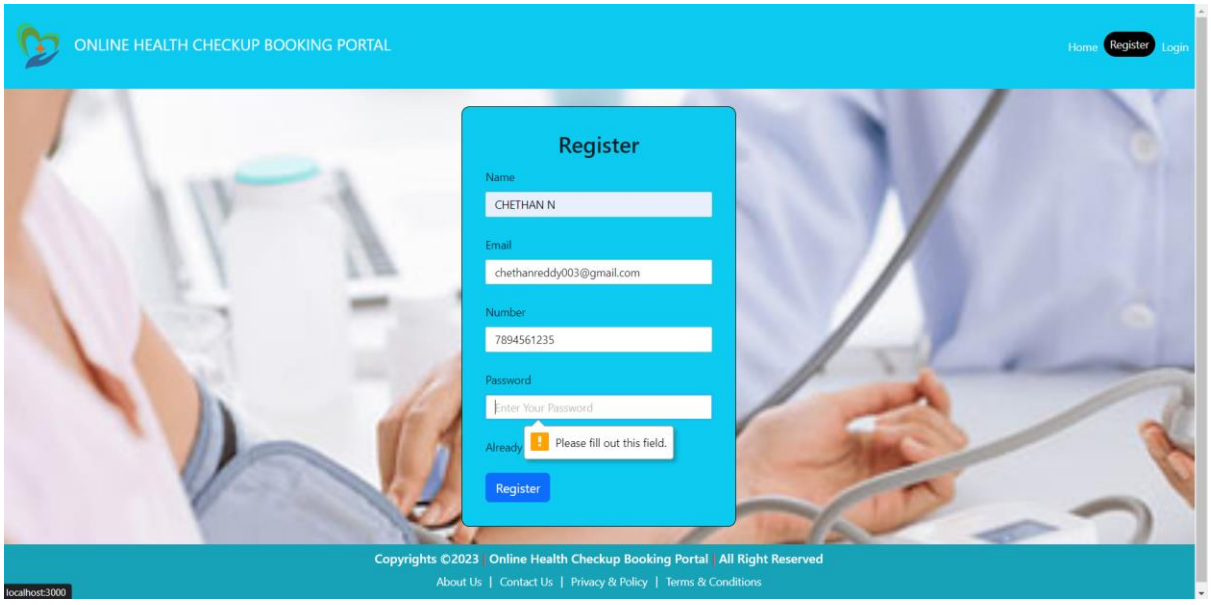

Fig. No. 6.4: Empty Password

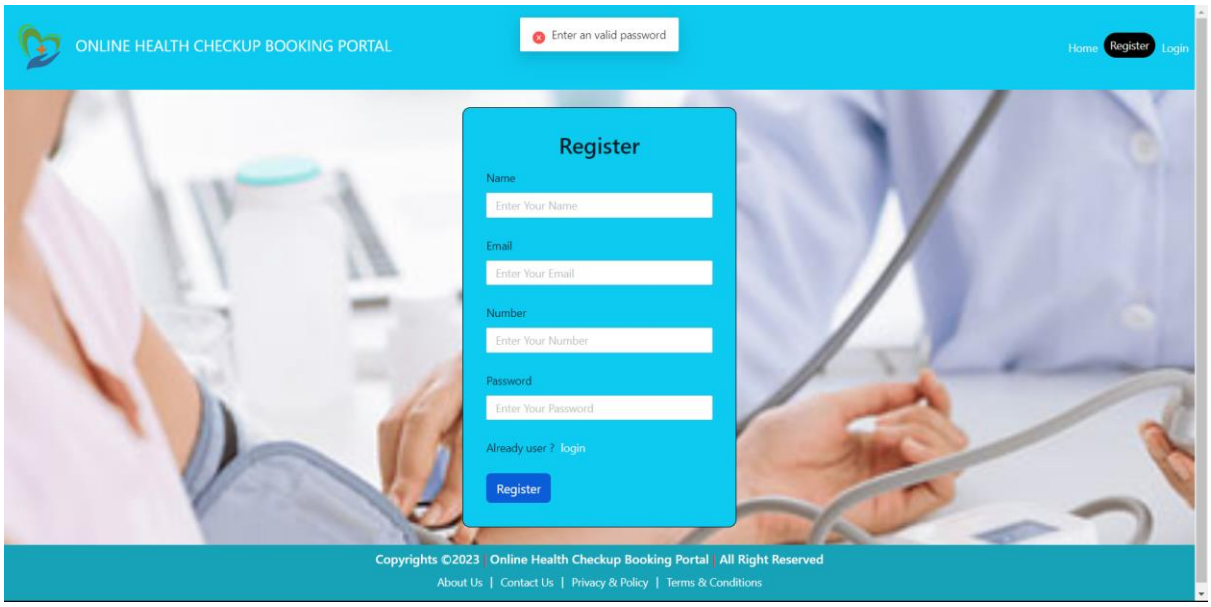

Fig. No. 6.5: Invalid Password

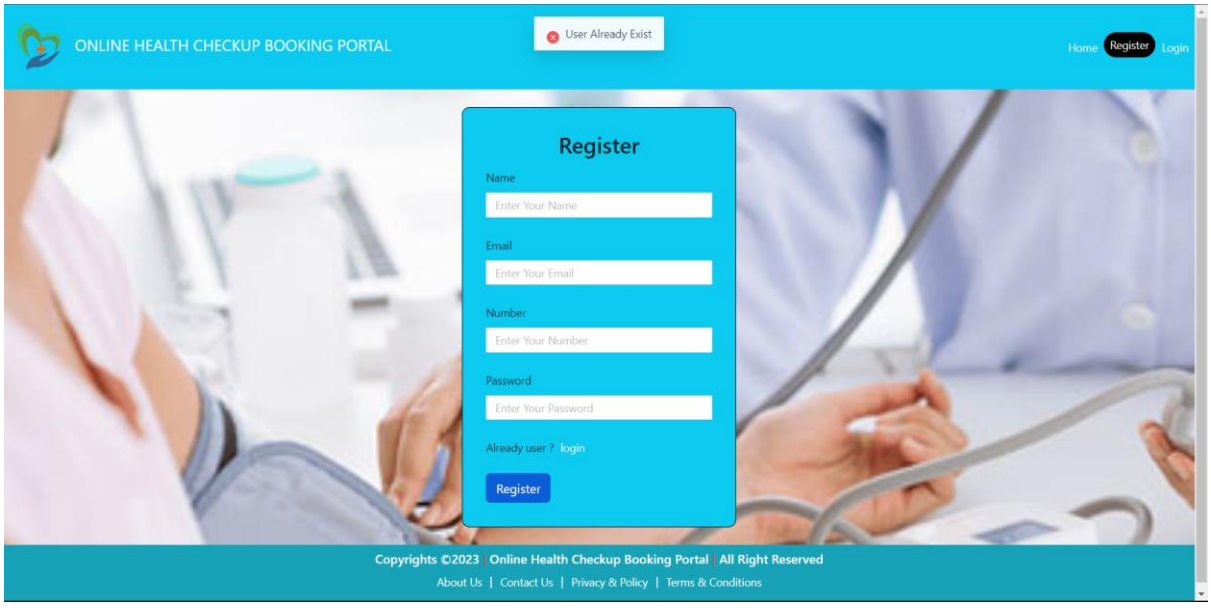

Fig. No. 6.6: User Already Exist

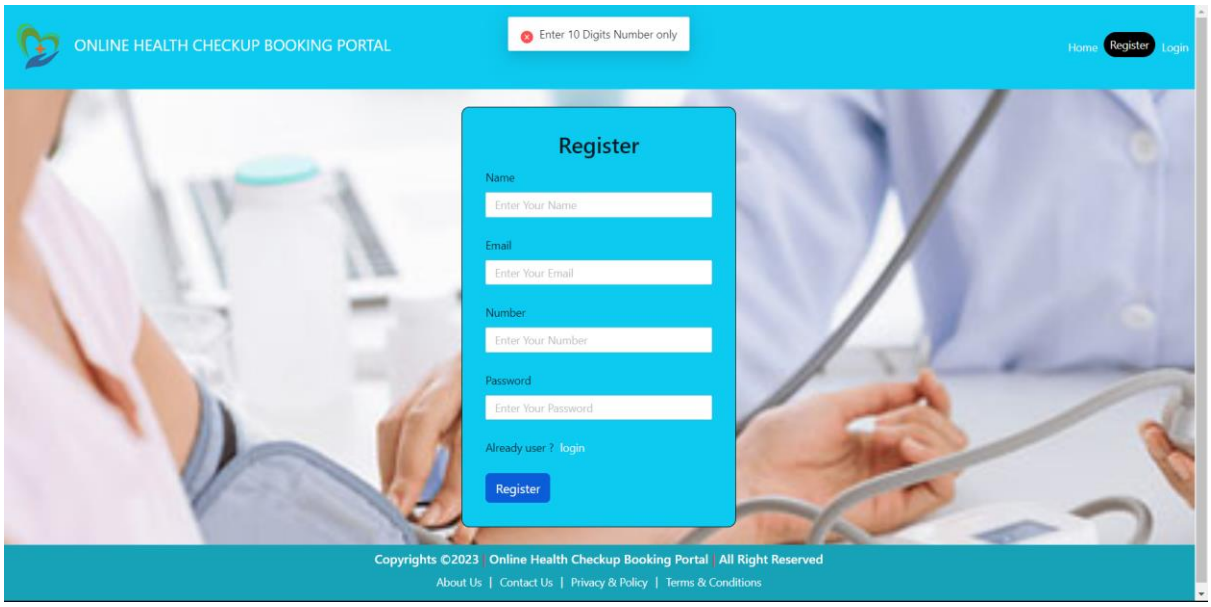

Fig. No. 6.7: Invalid Number

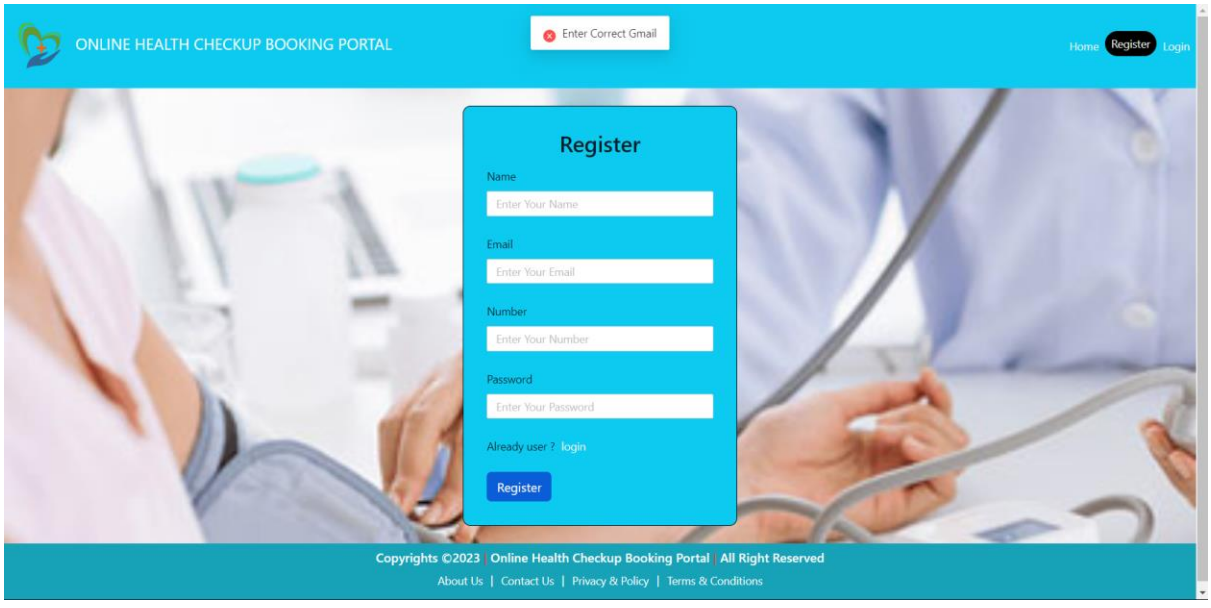

Fig. No. 6.8: Invalid Email

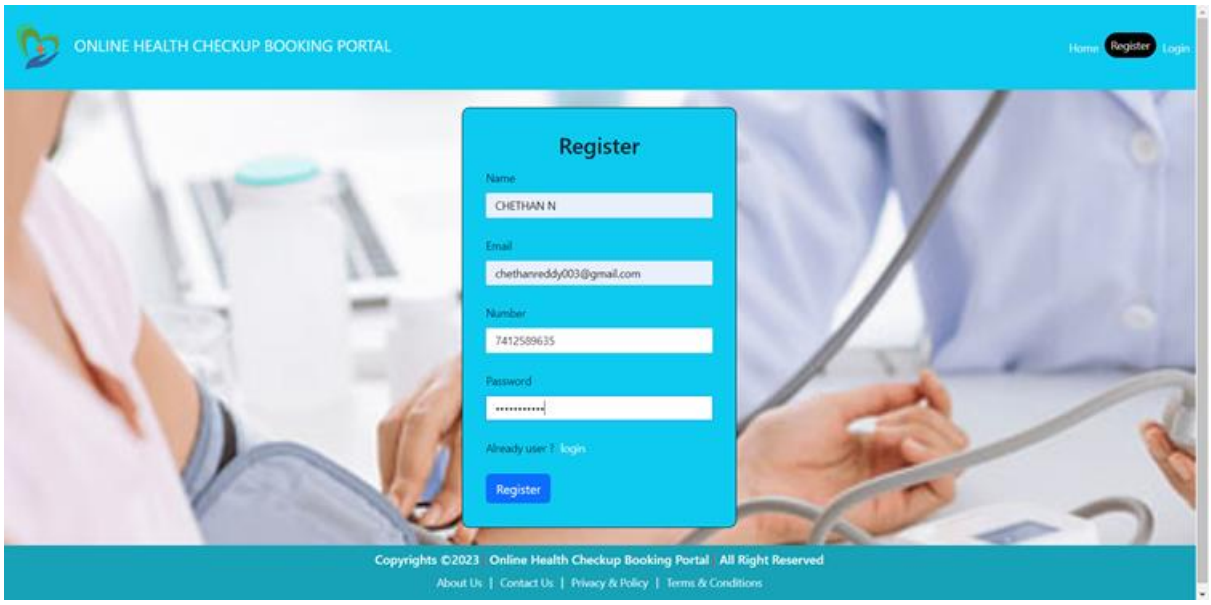

Fig. No. 6.9: All Valid Details

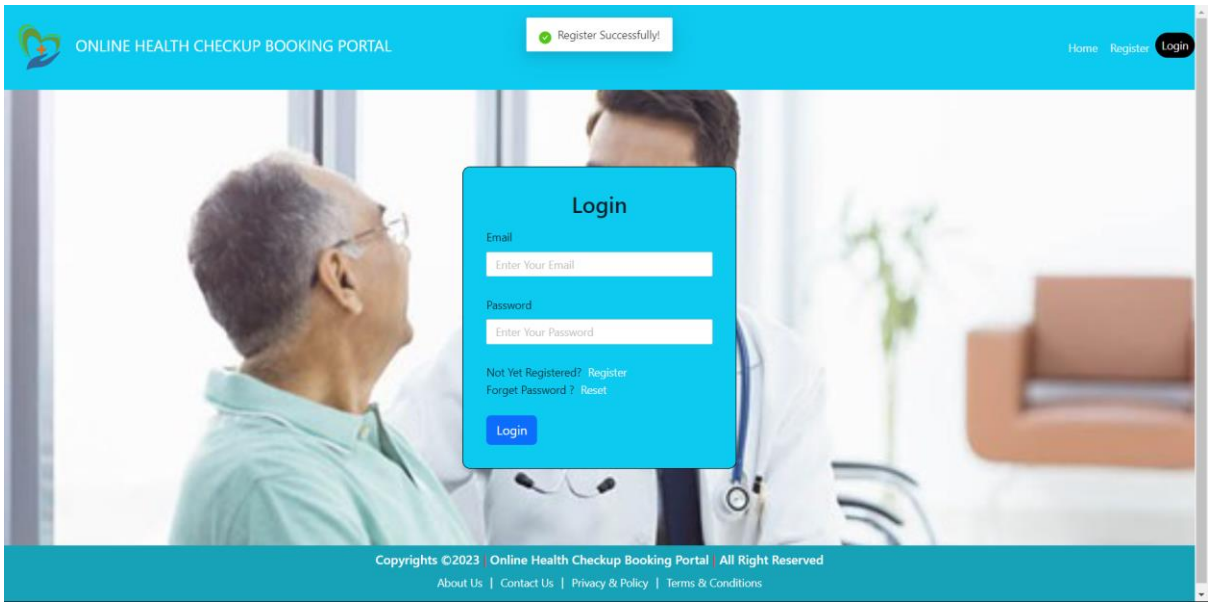

Fig. No. 6.10: Register Success

Test Case – 02 (Login Form)

SI NO.	Description	Expected Result	Actual Result	Status
02	Login Form	All Valid Details	Empty / Invalid Email	Fail
		All Valid Details	Empty / Invalid Password	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.2

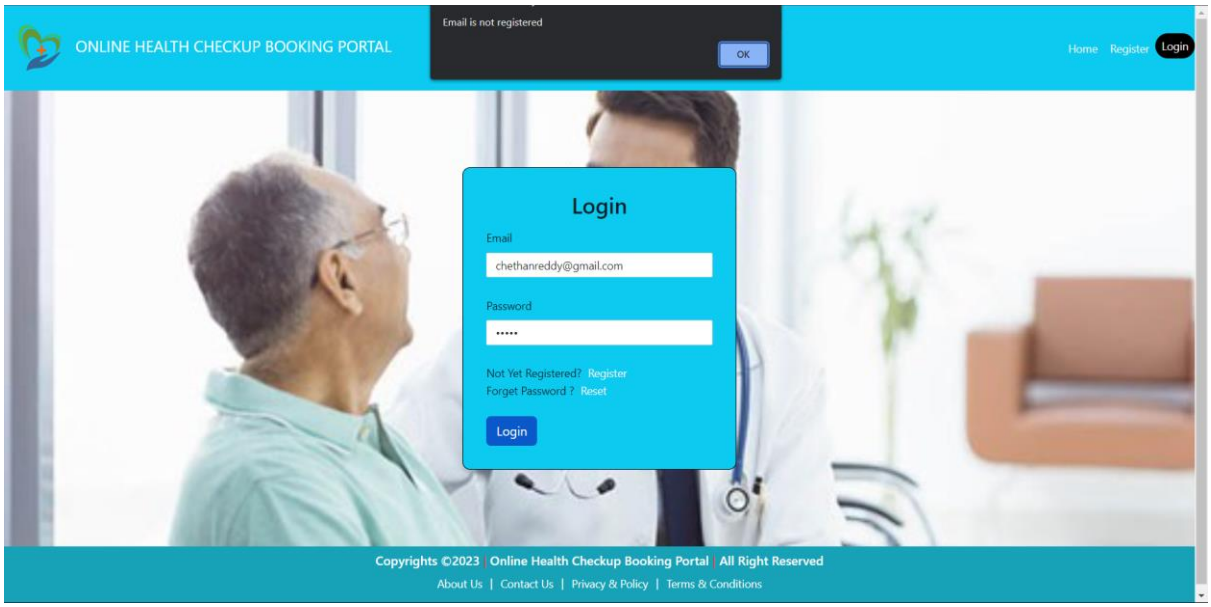

Fig. No. 6.11: Invalid Email

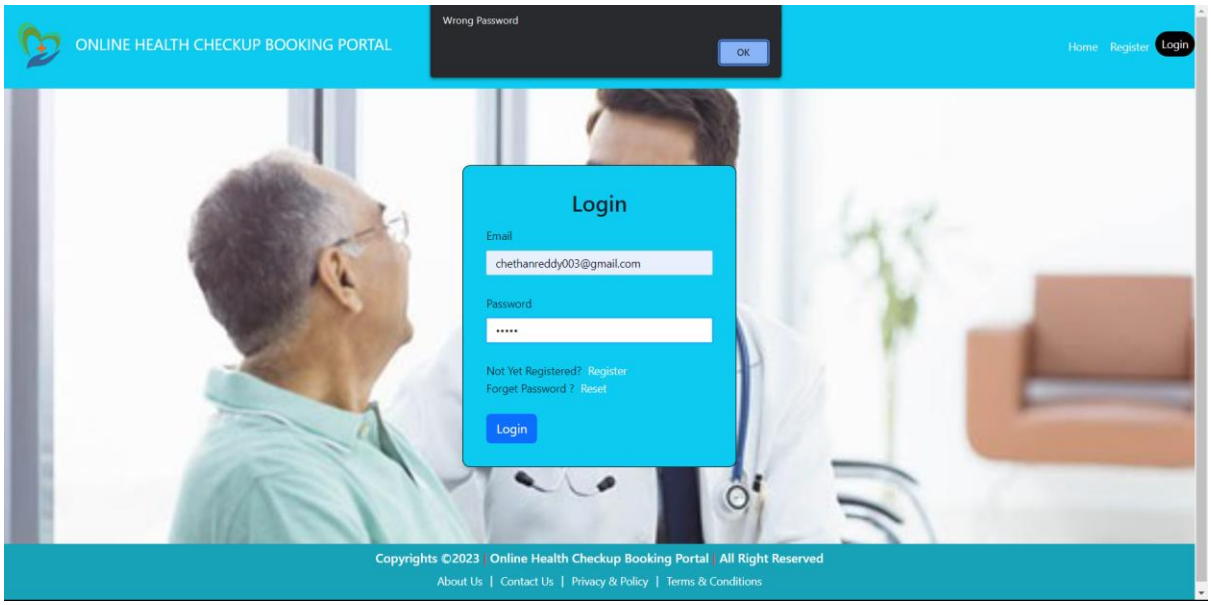

Fig. No. 6.12: Invalid Password

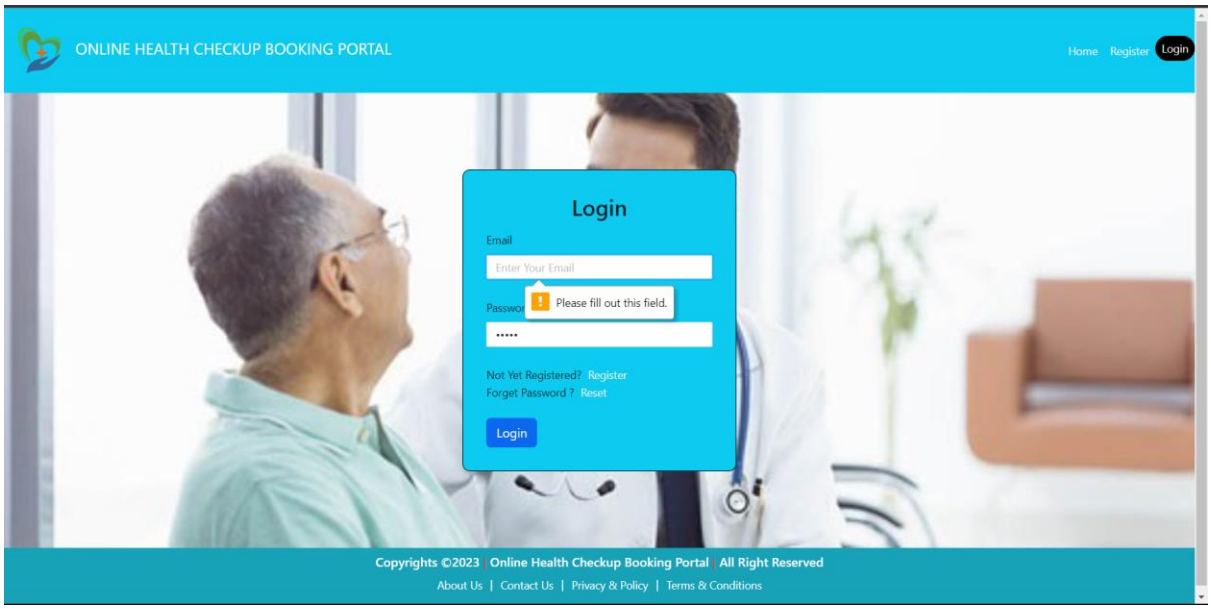

Fig. No. 6.13: Empty Email

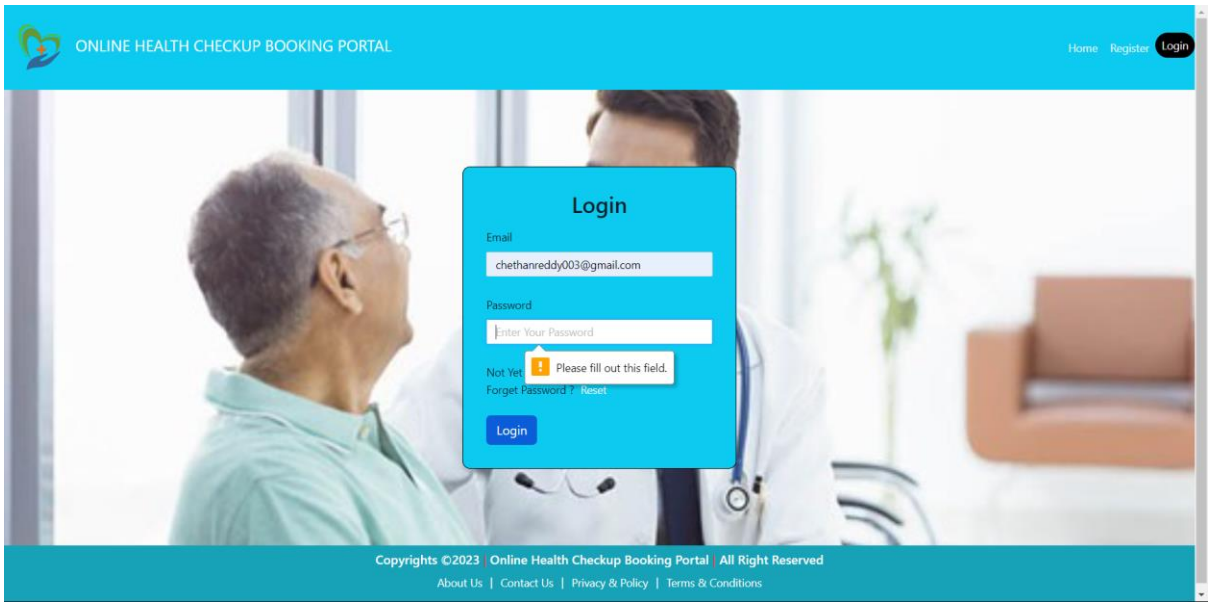

Fig. No. 6.14: Empty Password

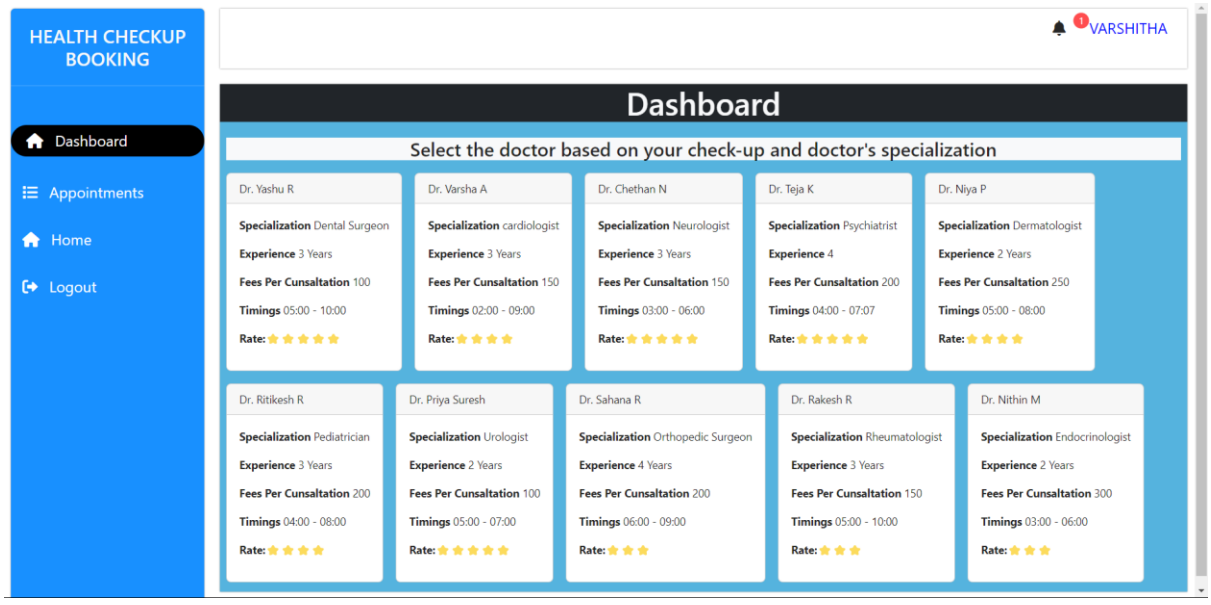

Fig. No. 6.15: Login Success

Test Case – 03 (Forget Password)

SI NO.	Description	Expected Result	Actual Result	Status
03	Forget Password	All Valid Details	Empty / Invalid Email	Fail
		All Valid Details	Empty / Invalid Number	Fail
		All Valid Details	Empty / Invalid New password	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.3

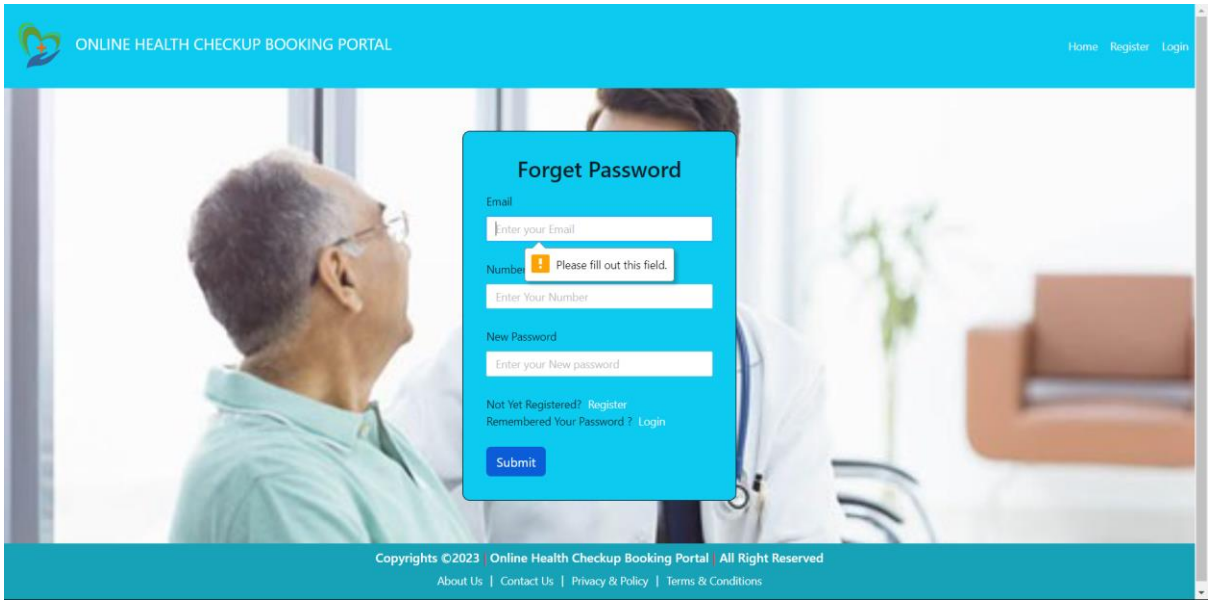

Fig. No. 6.16: Empty Email

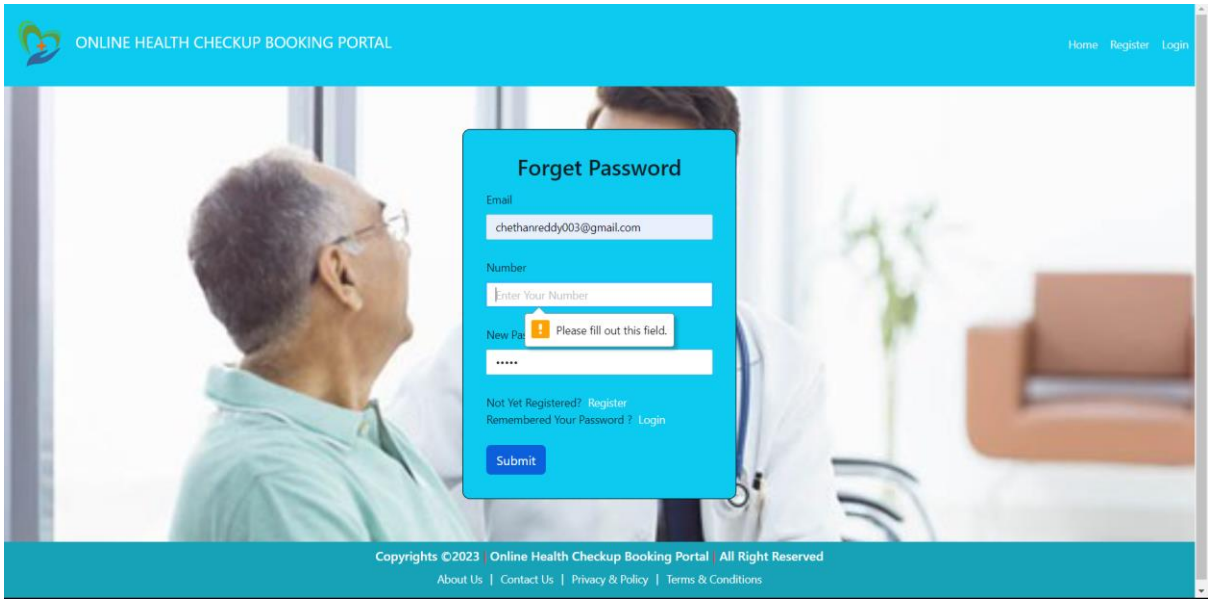

Fig. No. 6.17: Empty Number

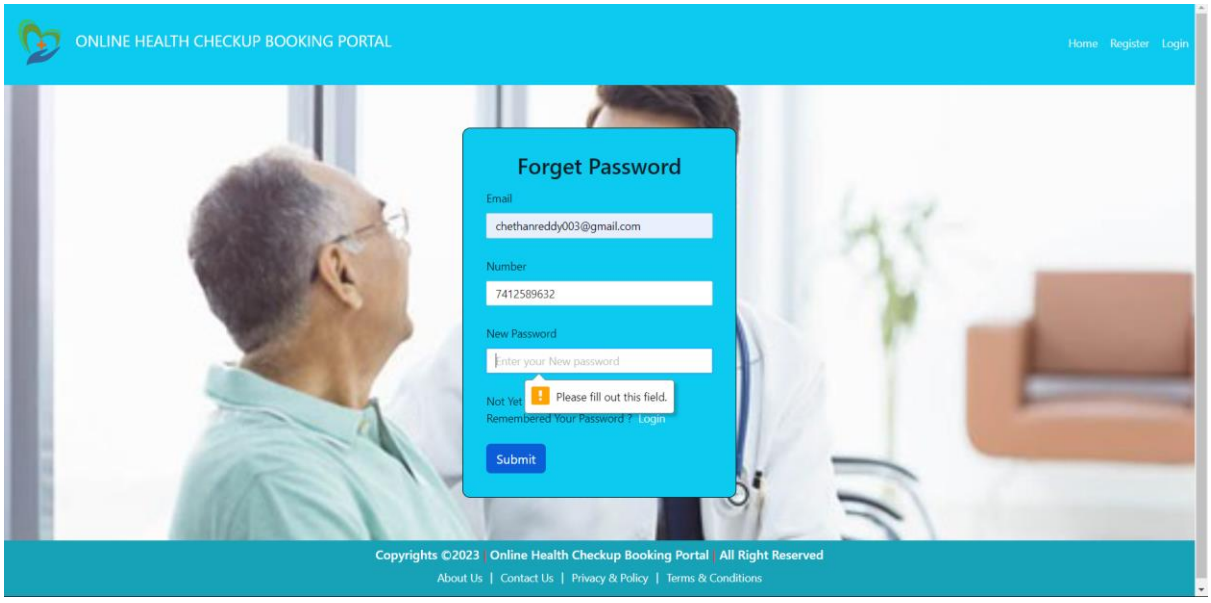

Fig. No. 6.18: Empty New Password

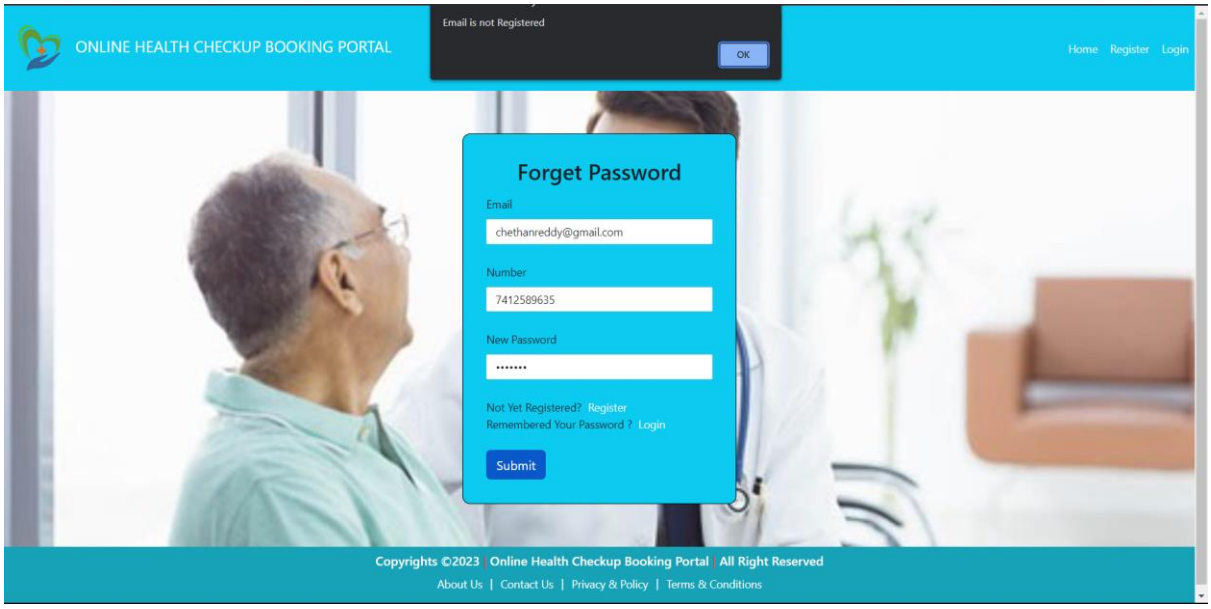

Fig. No. 6.19: Invalid Email

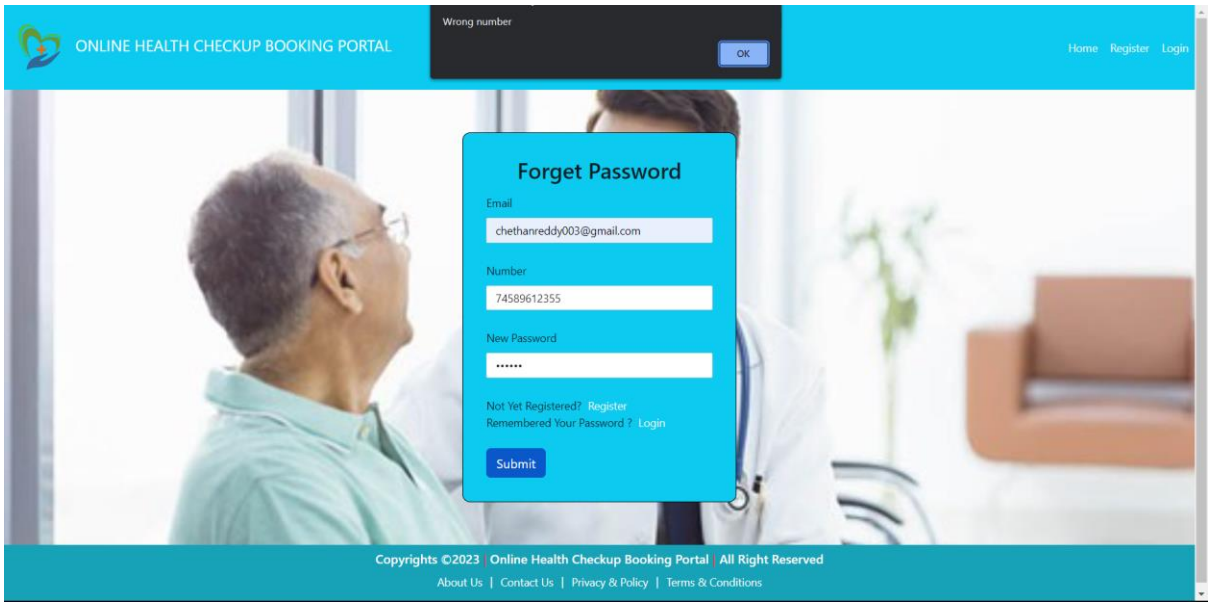

Fig. No. 6.20: Invalid Number

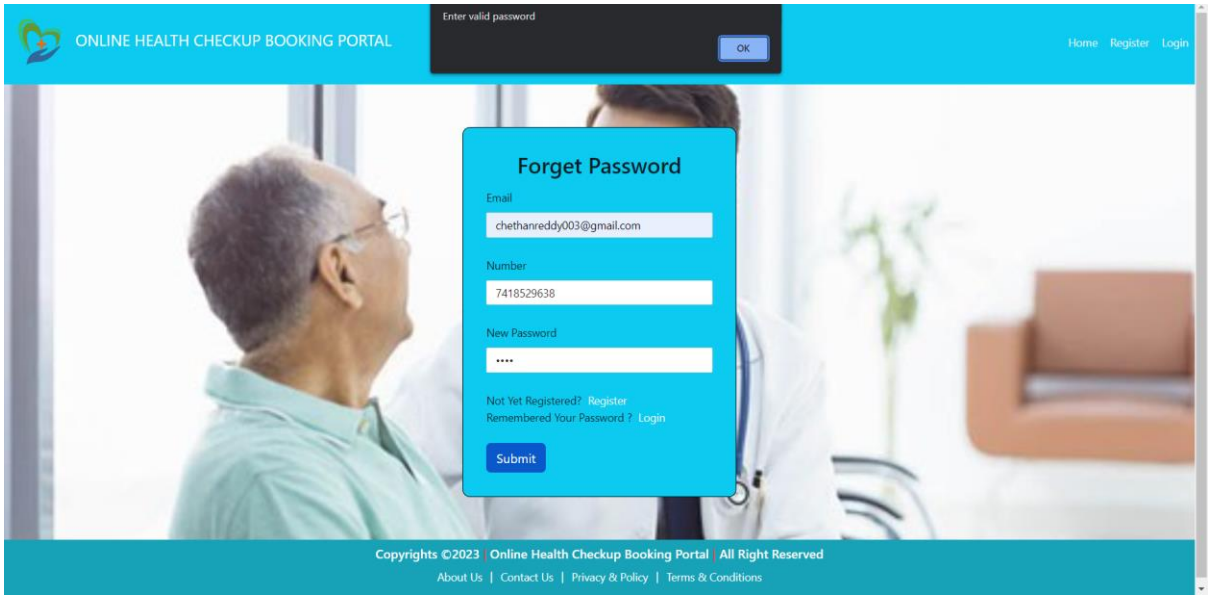

Fig. No. 6.21: Invalid New Password

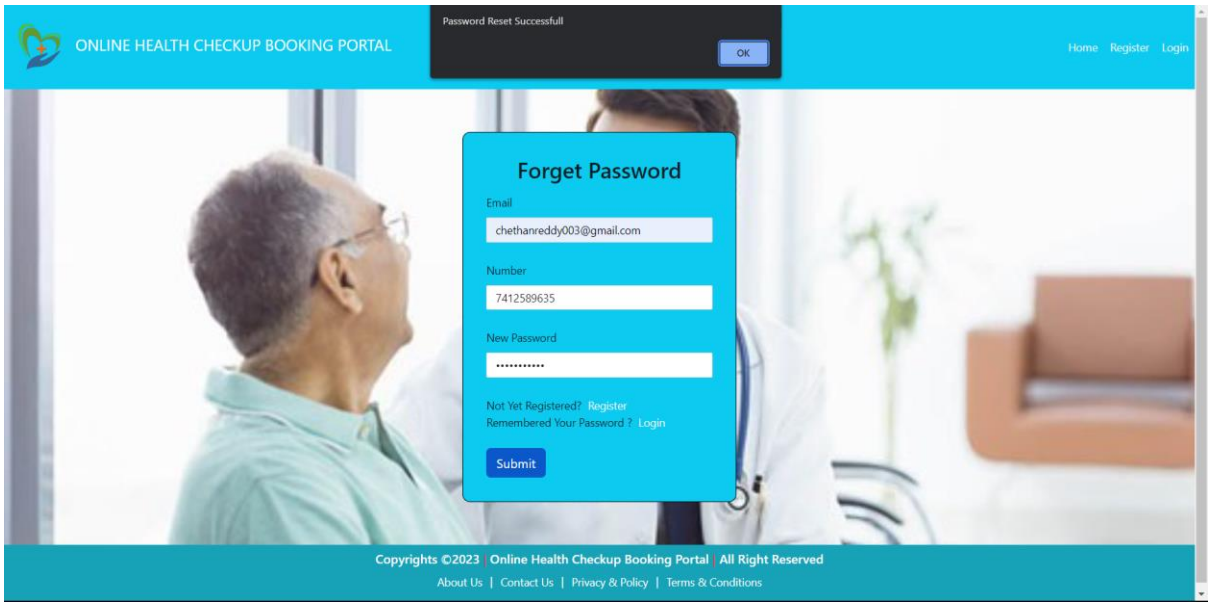

Fig. No. 6.22: Password Reset Successful

Test Case – 04 (Check Availability)

SI NO.	Description	Expected Result	Actual Result	Status
04	Check Availability	All Valid Details	Empty / Invalid Date	Fail
		All Valid Details	Empty / Invalid Time	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.4

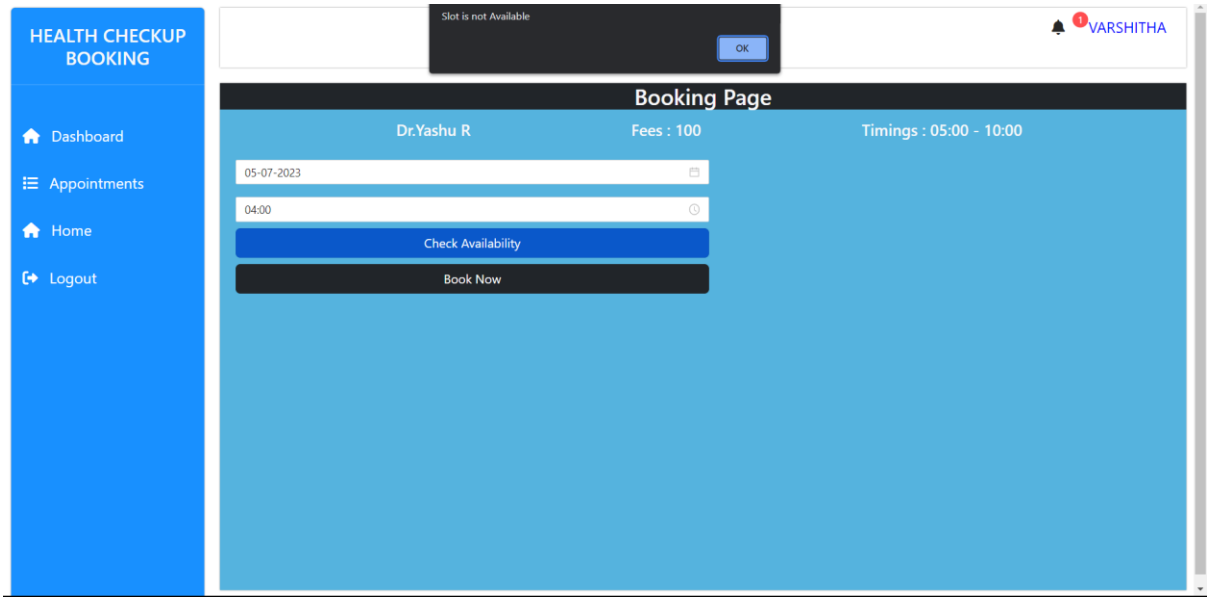

Fig. No. 6.23: Slot not available

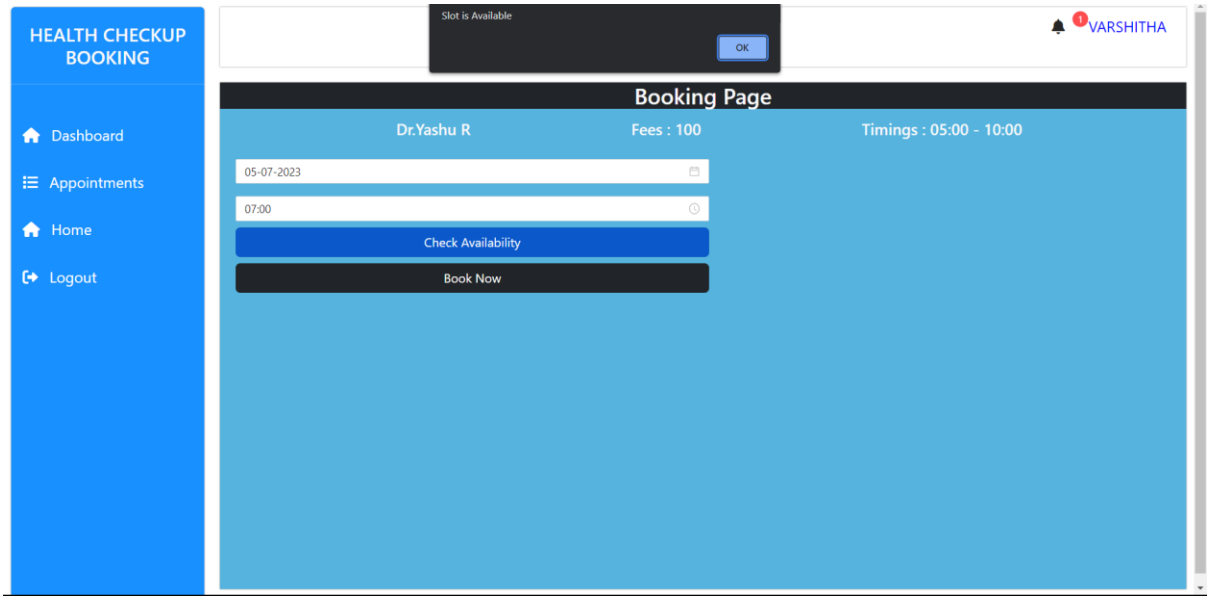

Fig. No. 6.24: Slot available

Test Case – 05 (Appointment Booking)

SI NO.	Description	Expected Result	Actual Result	Status
05	Appointment Booking	All Valid Details	Empty date	Fail
		All Valid Details	Empty Time	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.5

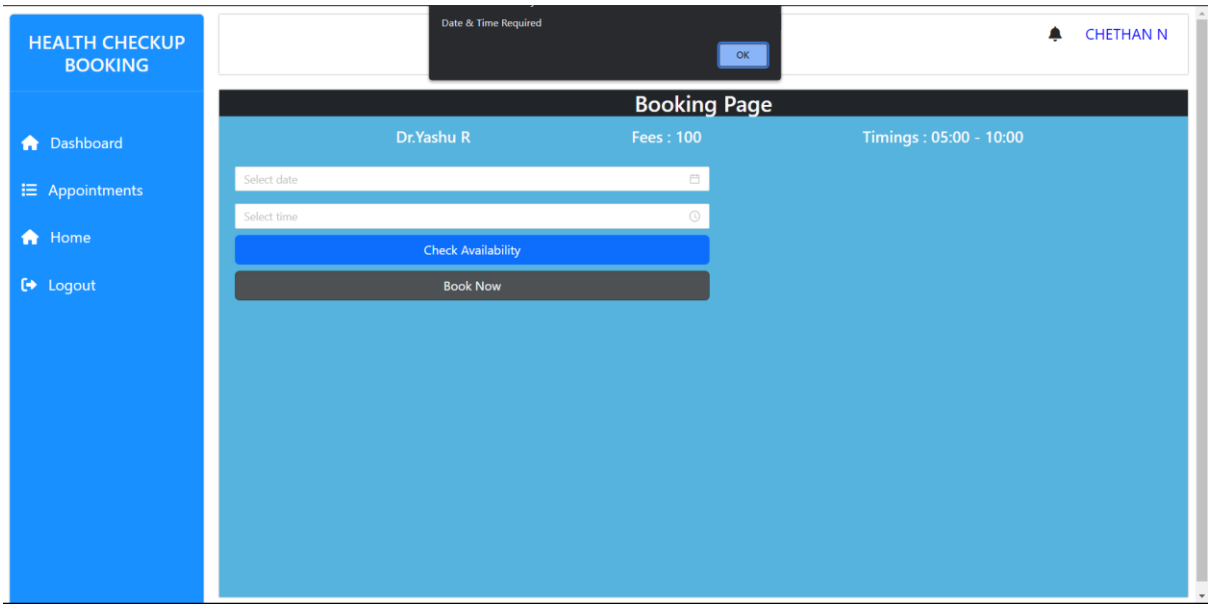

Fig. No. 6.25: Empty Date / Time

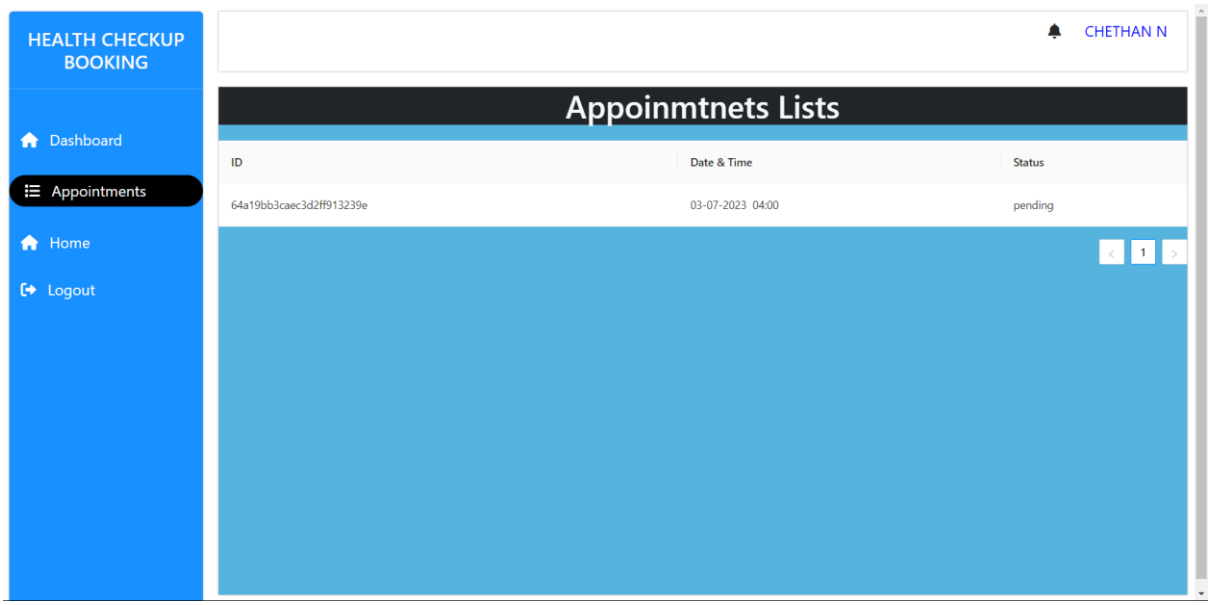

Fig. No. 6.26: Appointment Booking Success

Test Case – 06 (Appointment status)

SI NO.	Description	Expected Result	Actual Result	Status
06	Appointment status	All Valid Details	Approve	Pass
		All Valid Details	Reject	Pass

Table No. 6.6

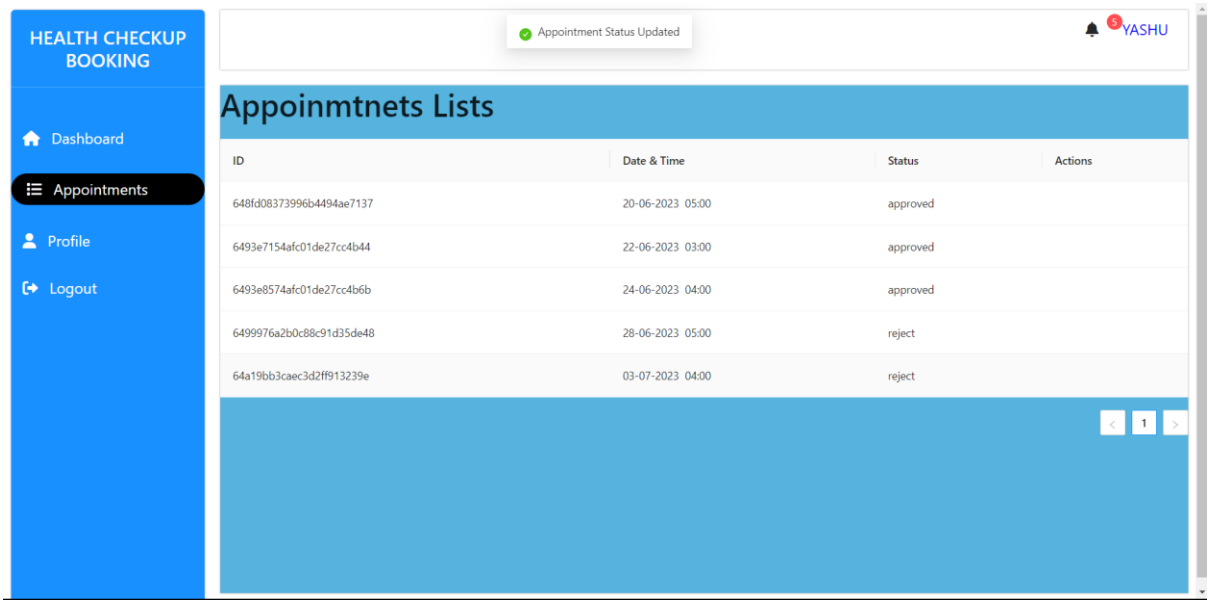

Fig. No. 6.27: Appointment Reject

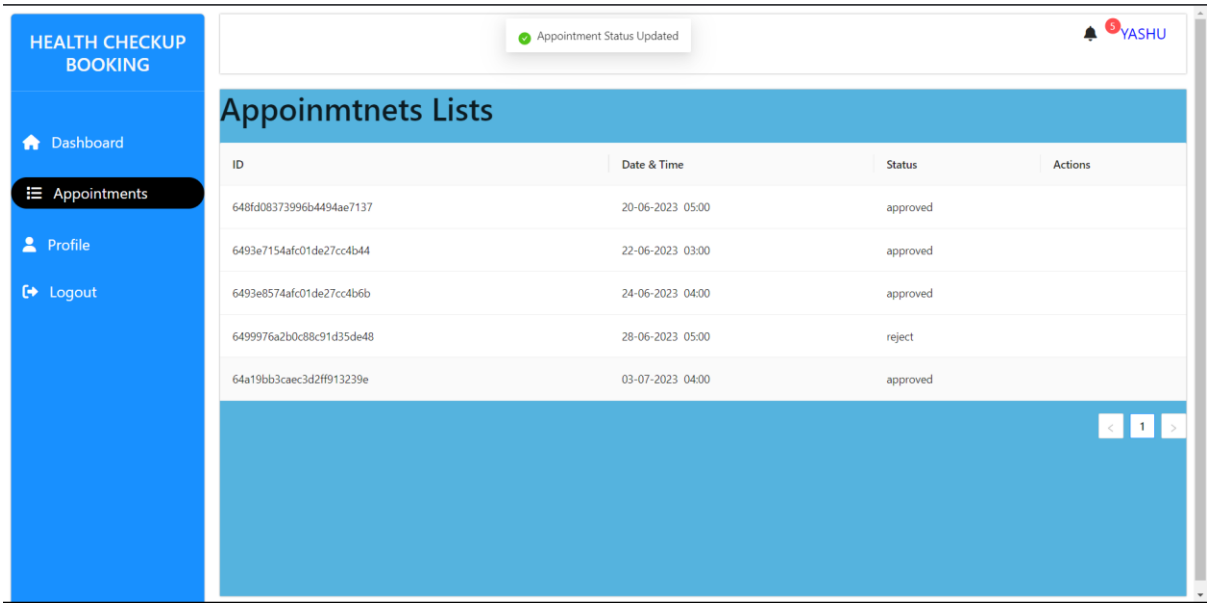

Fig. No. 6.28: Appointment Approved

Test Case – 07 (Doctor Profile)

SI NO.	Description	Expected Result	Actual Result	Status
07	Doctor Profile	All Valid Details	Empty / Invalid First Name	Fail
		All Valid Details	Empty / Invalid Last Name	Fail
		All Valid Details	Empty / Invalid Number	Fail
		All Valid Details	Empty / Invalid Email	Fail
		All Valid Details	Empty / Invalid Address	Fail
		All Valid Details	Empty / Invalid Number	Fail
		All Valid Details	Empty / Invalid Specialization	Fail
		All Valid Details	Empty / Invalid Experience	Fail
		All Valid Details	Empty / Invalid fee	Fail
		All Valid Details	Empty / Invalid Timings	Fail
		All Valid Details	All Valid Details	Pass

Table No. 6.7

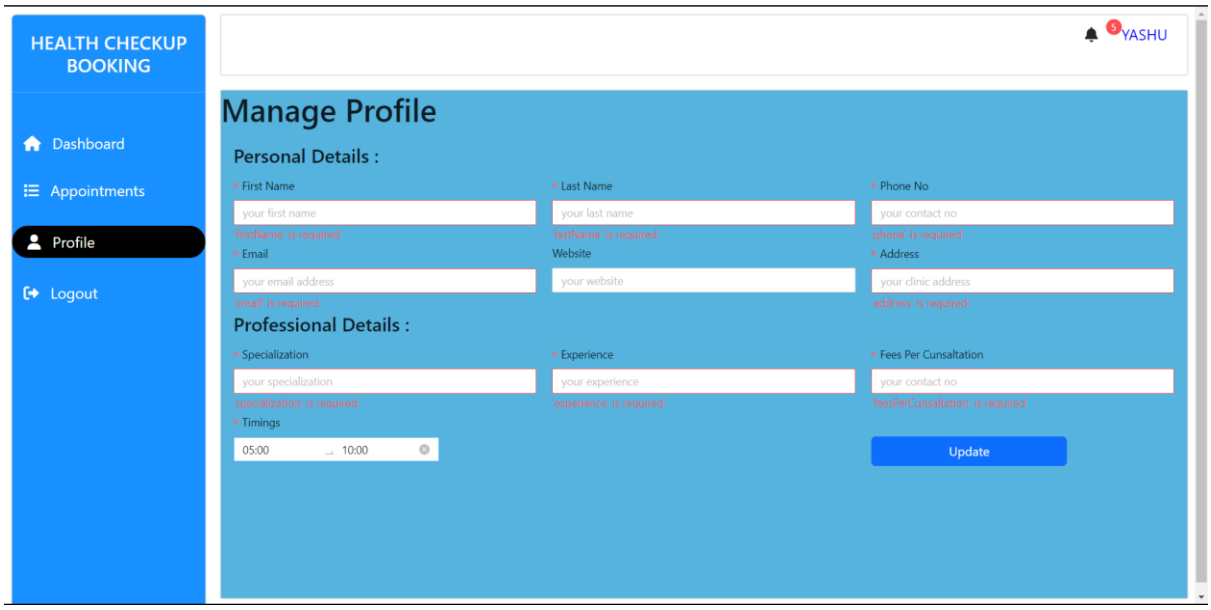

Fig. No. 6.29: Empty Fields

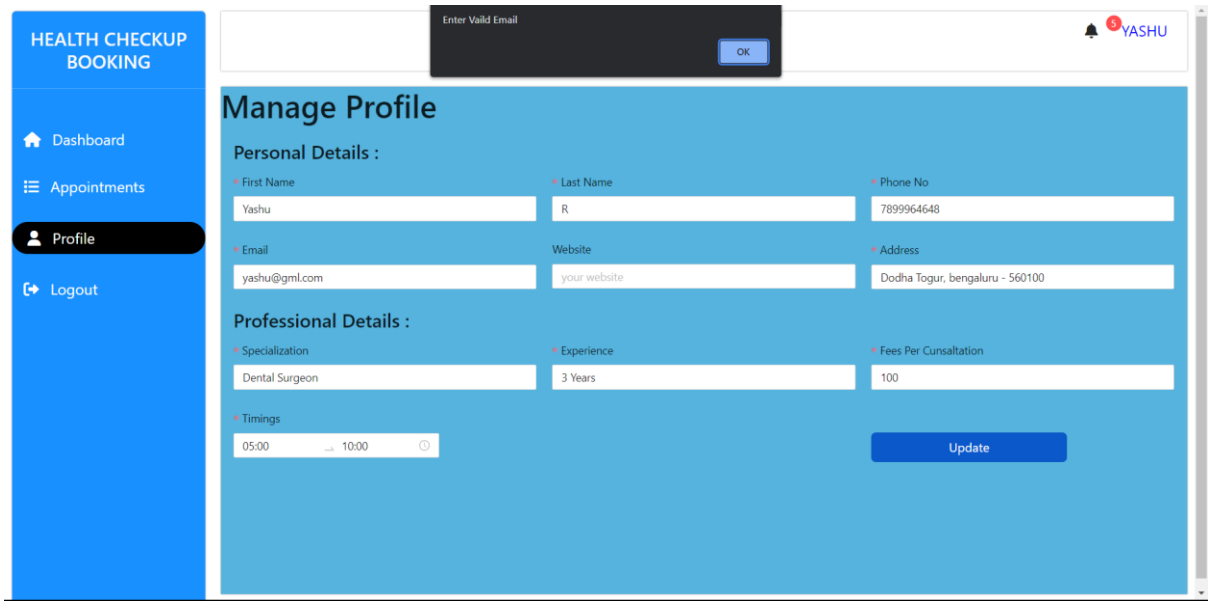

Fig. No. 6.30: Invalid Email

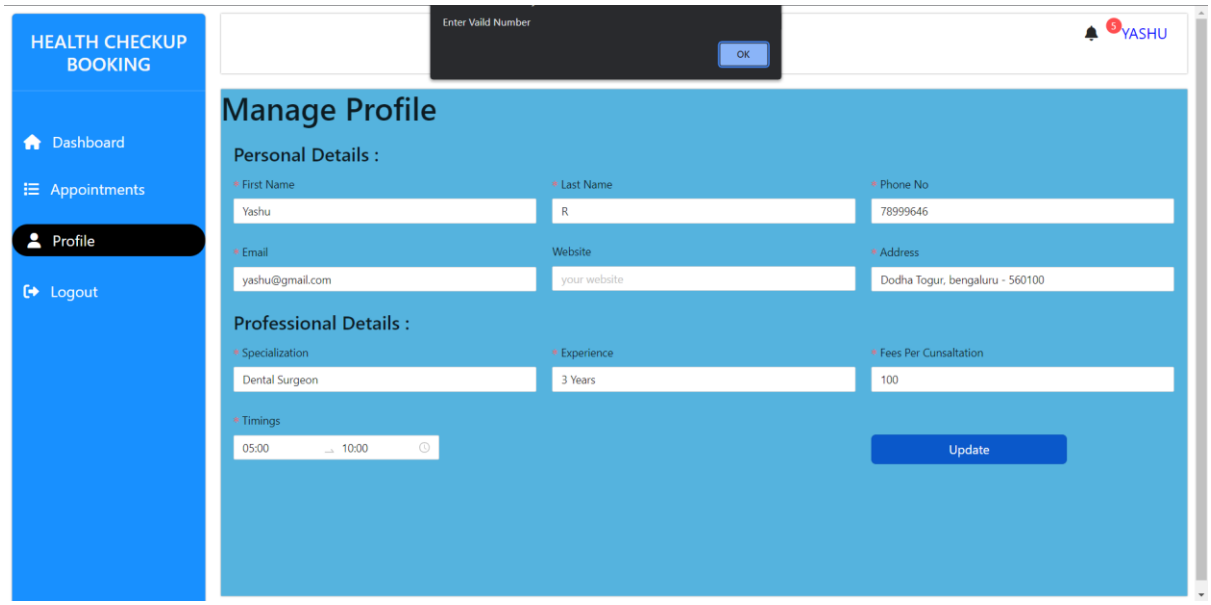

Fig. No. 6.31: Invalid Number

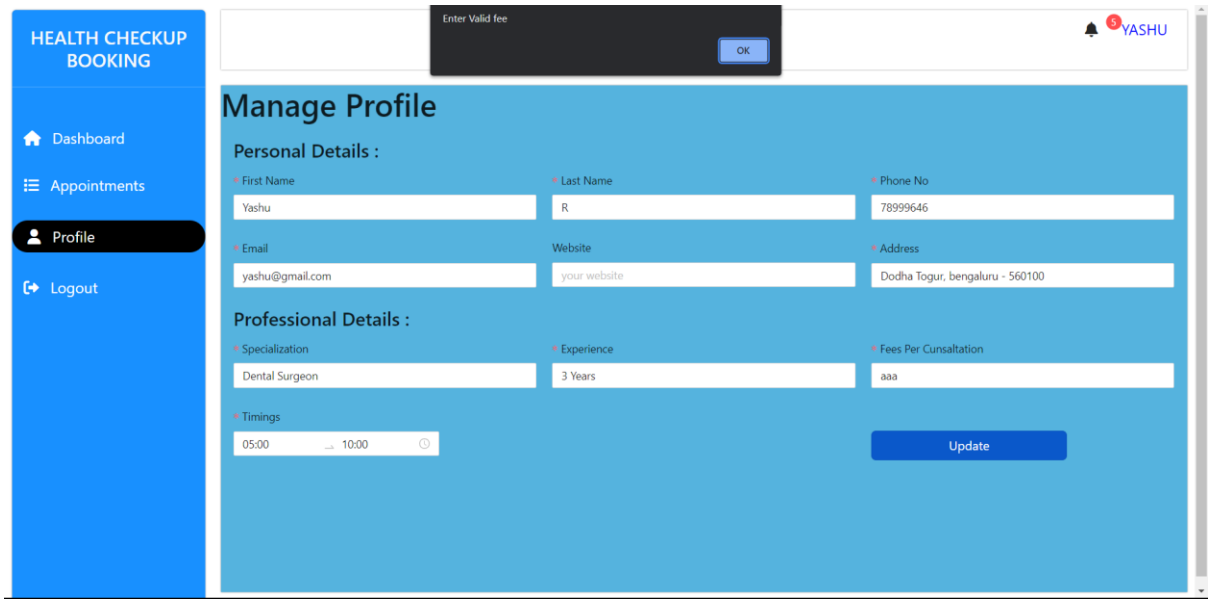

Fig. No. 6.32: Invalid Fee

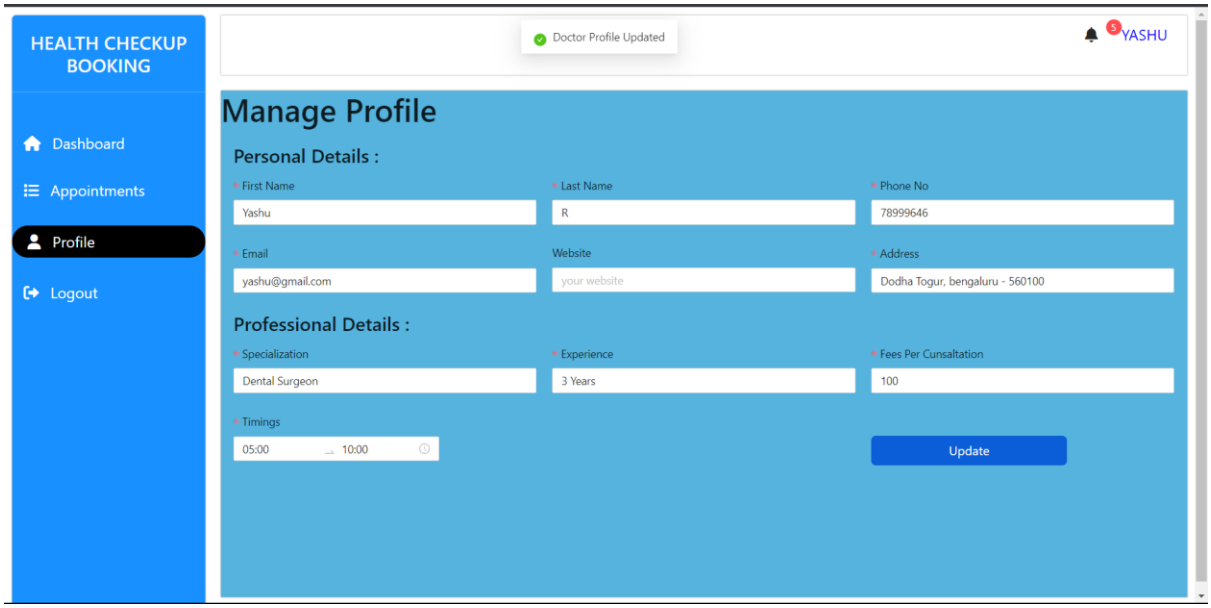

Fig. No. 6.33: Profile Update Success

7. CONLUSION

In conclusion, the development of the Online Health Checkup Booking Portal using the MERN stack has successfully addressed the limitations of the existing manual appointment booking system. The proposed system provides users with a user-friendly platform for booking health checkup appointments, automating the process and improving efficiency. It offers real-time availability of doctors, streamlined appointment booking, and a notification system for timely updates. Additionally, the system includes an administrative module for managing doctor requests and overseeing the entire system effectively.

The Online Health Checkup Booking Portal has numerous benefits, such as saving time for users, improving communication between users and doctors, and enhancing the overall user experience. The system streamlines the appointment booking process, reduces manual intervention, and provides an intuitive interface for users to manage their appointments. It also offers doctors a platform to manage their appointments, communicate with patients, and generate prescriptions and reports digitally.

8. FUTURE ENHANCEMENT

While the Online Health Checkup Booking Portal provides significant advantages, there are several potential areas for future enhancements to further improve the system:

- **Integration with Electronic Health Records (EHR):** Integrating the system with EHR systems would allow doctors to access and update patient medical records seamlessly.
- **Advanced Analytics and Reporting:** Enhancing the reporting and analytics capabilities of the system would enable administrators to gain insights into appointment statistics, patient feedback, and doctor performance.
- **Telemedicine Integration:** Integrating telemedicine functionality would allow users to have virtual consultations with doctors, expanding the reach and convenience of the system. Users could have remote appointments, receive medical advice, and even undergo virtual examinations.
- **Mobile Application Development:** Developing a mobile application for the Online Health Checkup Booking Portal would increase accessibility and convenience for users.
- **Integration with Payment Gateways:** Integrating secure payment gateways would allow users to make online payments for their appointments, simplifying the payment process and reducing manual efforts.

These future enhancements would further elevate the Online Health Checkup Booking Portal, providing more comprehensive healthcare services, improved patient-doctor interactions, and enhanced system functionality.

9. REFERENCES

1) https://youtube.com/playlist?list=PLuHGmgpyHfRw0wBGN8knxsJsMi74r34zw
2) https://github.com/techinfo-youtube/Doctor-appointmnet-system-mern-project
3) https://www.npmjs.com/
4) https://nodejs.org/api/https.html
5) https://legacy.reactjs.org/docs/getting-started.html
6) https://stackoverflow.com/