

****

**Hospital Management System**

**Website Report**

**Table of Contents**

1. **Introduction**
   * Overview of project
   * Technologies Used
2. **Website Features**
   * Home Page
   * Login/Registration
   * Admin Controls
   * Notifications
   * Appointments Booking
3. **Backend Architecture**
   * Express.js and API Design
   * User Authentication
   * Database Schema
4. **User Portal**
   * Login
   * Can Apply for Doctor
   * Book an appointment
   * View all appointment
5. **Doctor**
   * Approve Appointments
   * Can view all appointments
6. **Security Measures**
   * JWT Tokens
   * Encryption
7. **Conclusion**
   * Summary
   * Future Enhancement

**Overview Of The Project**

Our Hospital Management System (HMS) is a comprehensive web-based platform designed to streamline administrative processes and enhance patient care within healthcare facilities. Built on the MERN stack (MongoDB, Express.js, React.js, Node.js), our system offers a user-friendly interface and robust functionality to meet the diverse needs of hospitals, clinics, and healthcare providers.

**Technologies Used** : **(MERN Stack)**

The MERN stack is popular for several reasons, and its advantages include:

1. Full-stack JavaScript: One of the key advantages of the MERN stack is that it allows developers to use JavaScript across the entire stack, from the front end to the back end. This means that developers can use the same language and ecosystem throughout the development process, which can lead to increased productivity and code maintainability.

2. React for dynamic UIs: React.js, as part of the MERN stack, enables developers to build dynamic and interactive user interfaces. React's component-based architecture makes it easy to create reusable UI components, which can help streamline development and improve code organization.

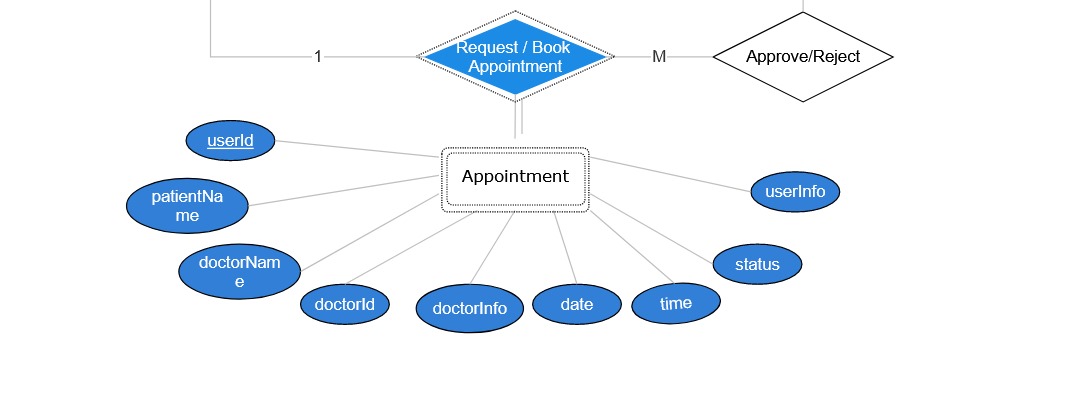
3. Node.js for server-side development: Node.js provides a lightweight and efficient platform for building server-side applications. With Node.js, developers can use JavaScript on the server side, allowing for easy sharing of code and expertise between the front end and back end development teams.

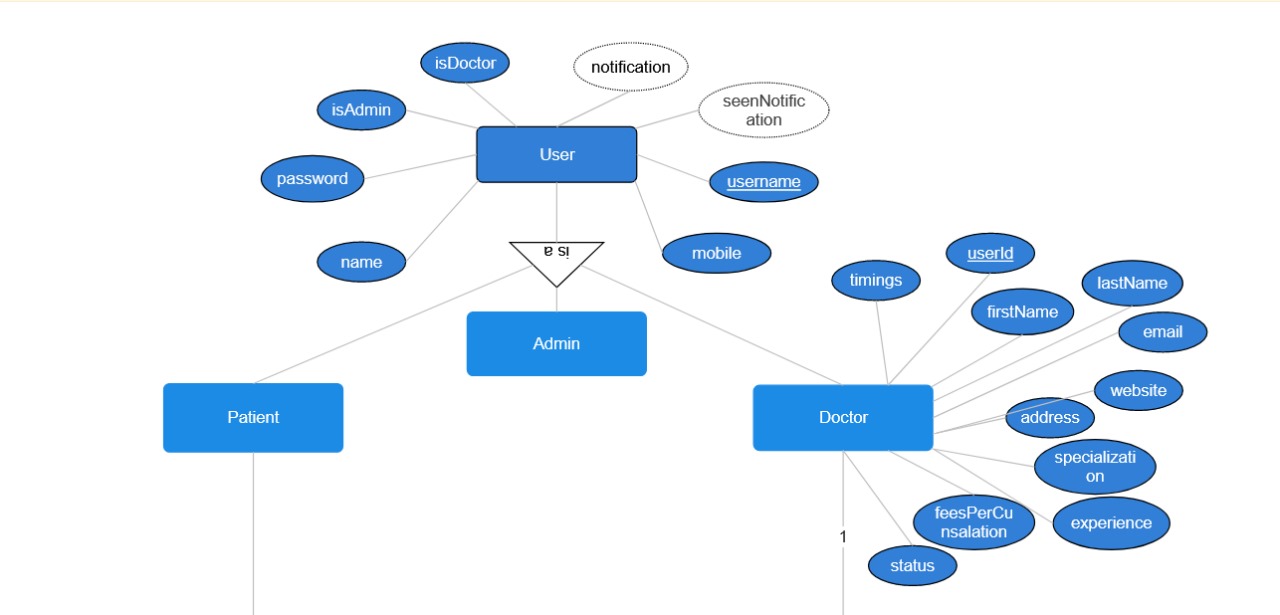
4. Express.js for server-side routing and middleware: Express.js, a minimalist web framework for Node.js, simplifies the process of building server-side applications and APIs. It provides a robust set of features for routing, middleware, and HTTP utilities, making it easy to handle requests and responses in a concise and efficient manner.

5. MongoDB for flexible data storage: MongoDB, a NoSQL database, offers flexibility and scalability for storing and querying data. Its document-based data model allows developers to store data in JSON-like documents, which can easily map to JavaScript objects used in the rest of the MERN stack. MongoDB's scalability and flexibility make it well-suited for handling large volumes of data and adapting to evolving application requirements.

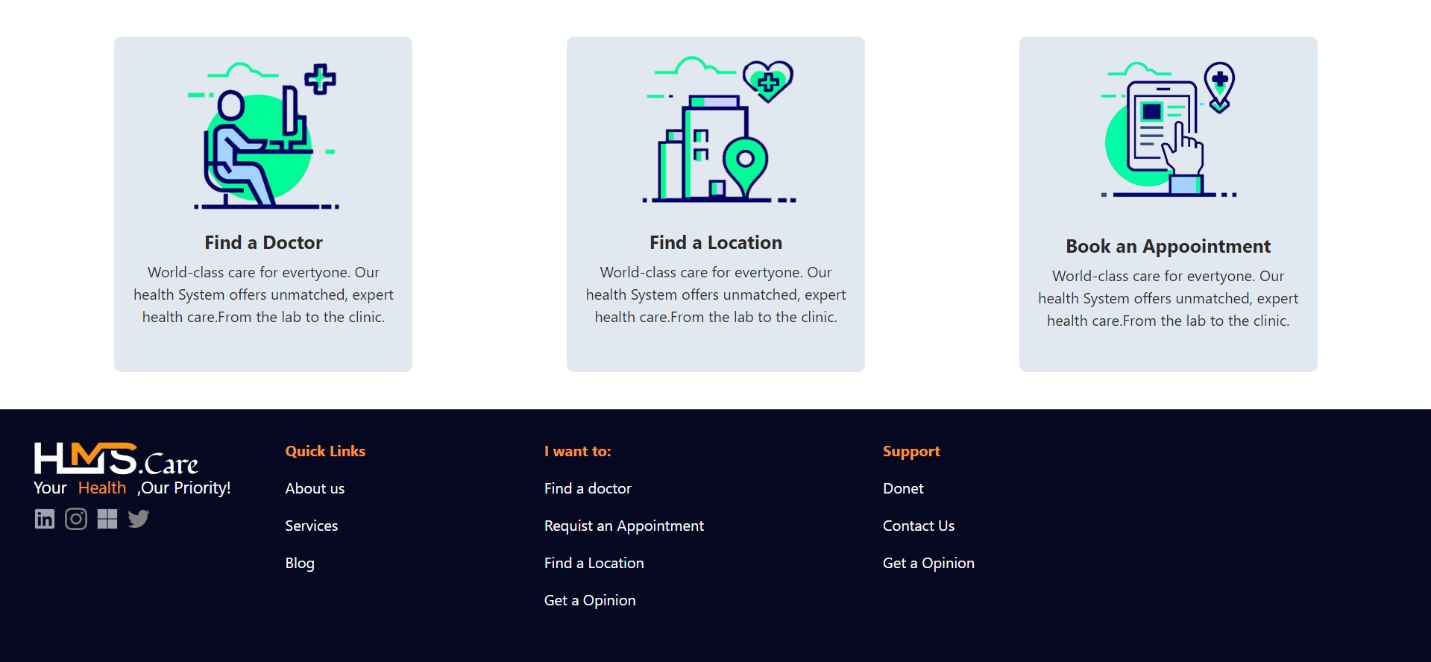
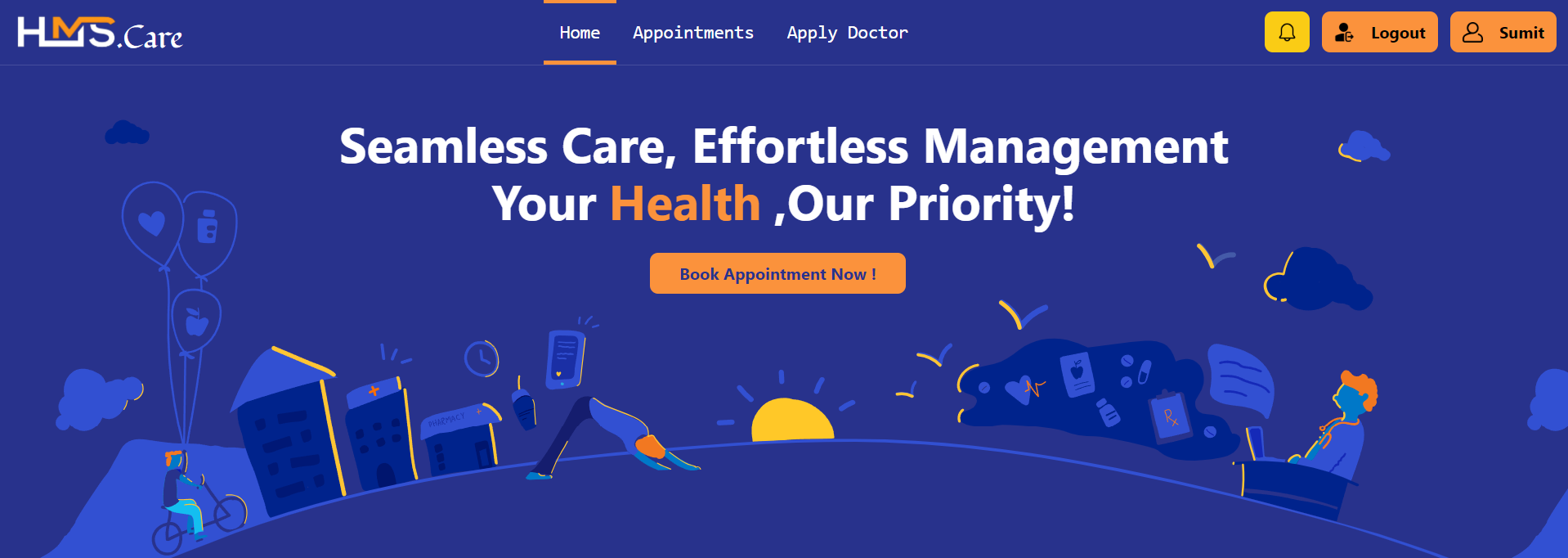
6. Community and ecosystem: The MERN stack benefits from a large and active community of developers, which means there are plenty of resources, tutorials, and libraries available to help developers get started and solve problems. Additionally, the MERN stack leverages the rich ecosystem of tools and packages available in the JavaScript ecosystem, further enhancing its appeal for developers.

Overall, the MERN stack offers a powerful and efficient solution for building modern web applications, with advantages such as full-stack JavaScript development, dynamic UIs with React.js, server-side development with Node.js and Express.js, and flexible data storage with MongoDB.

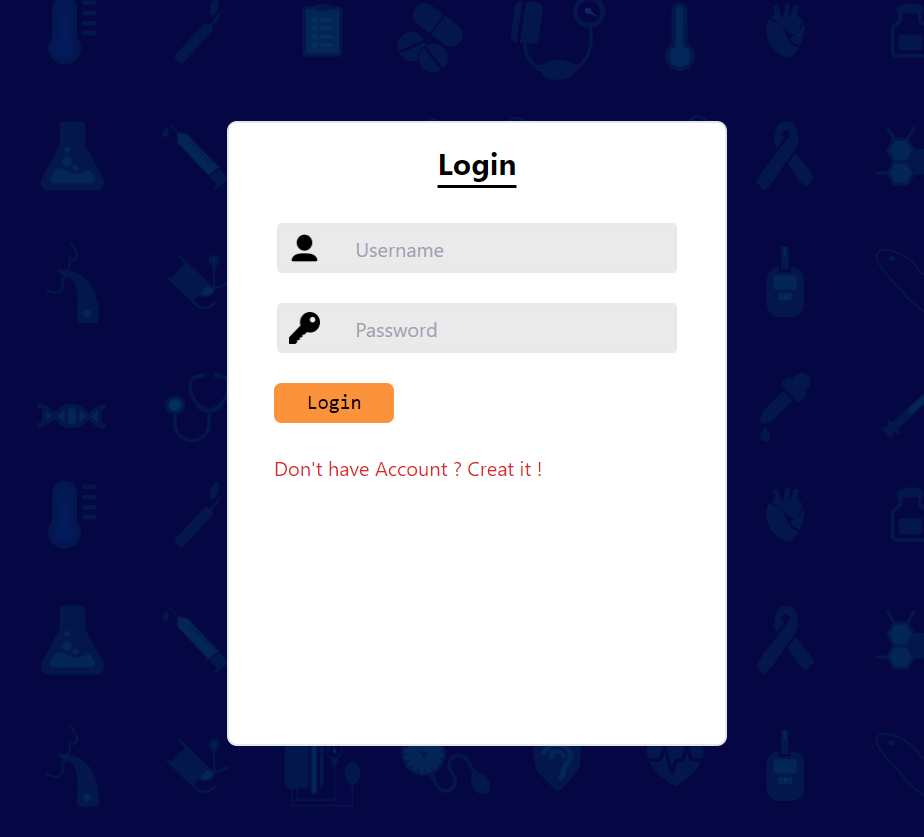
 **: ER Diagram :**



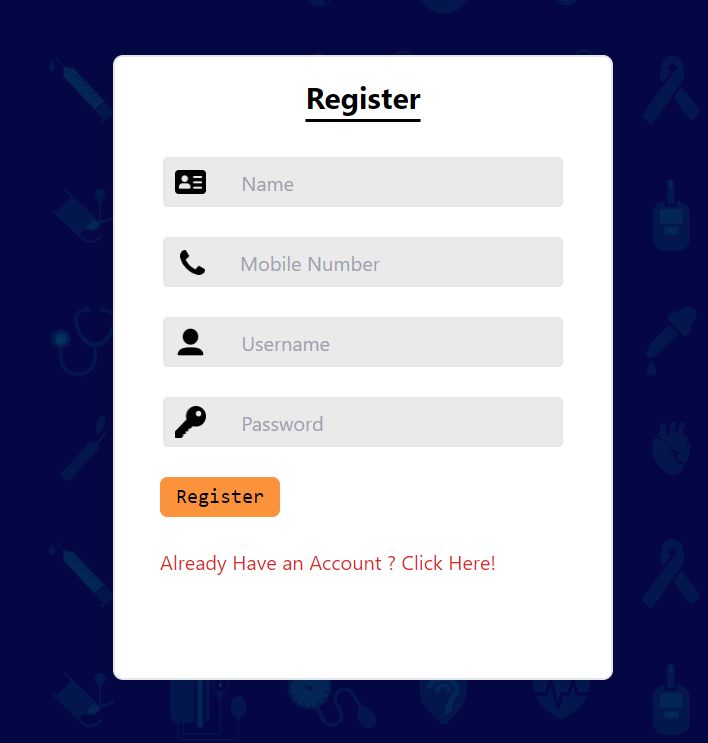
**Website Features**

**Home Page**

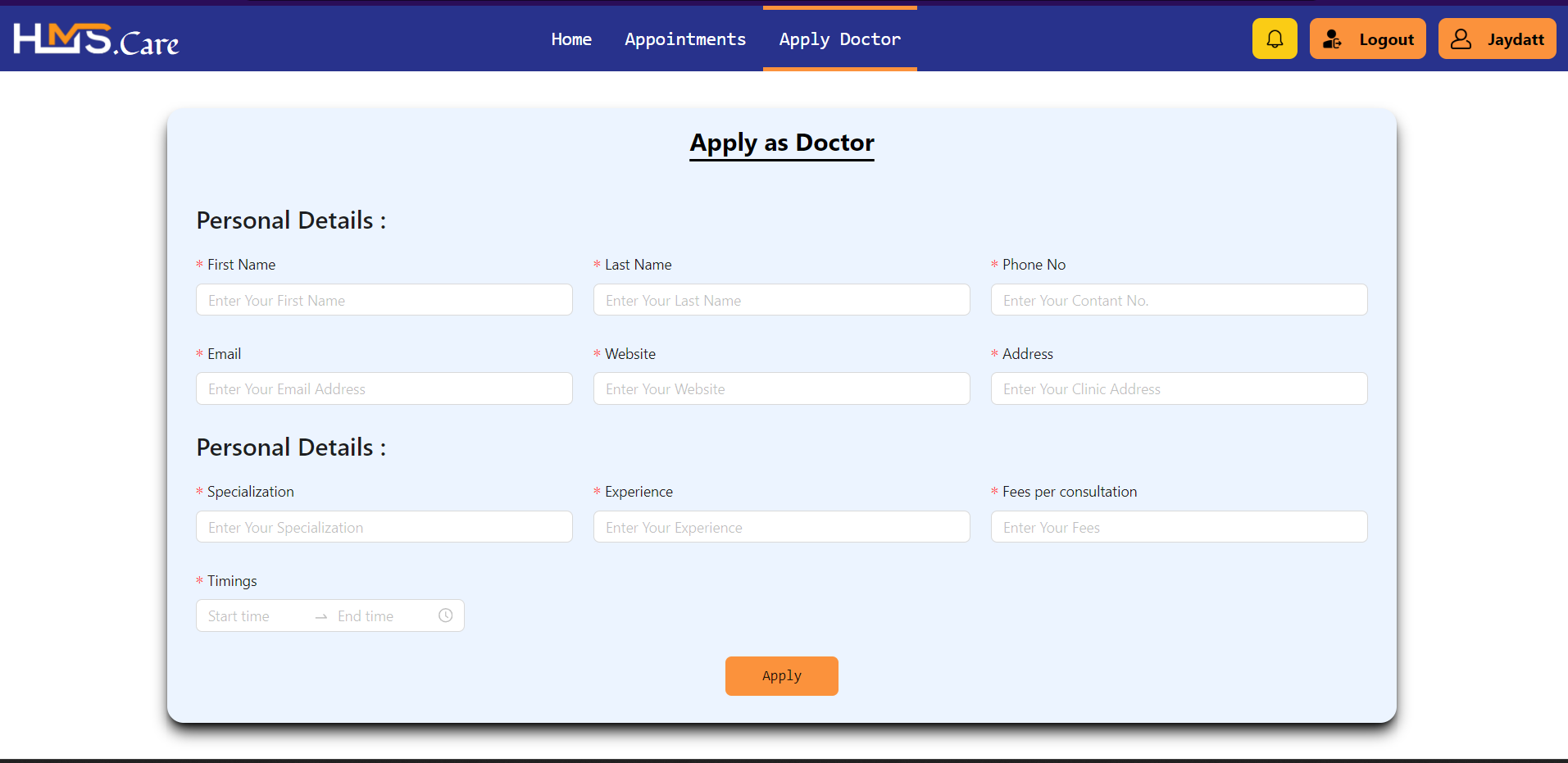
**Log in Page**

****

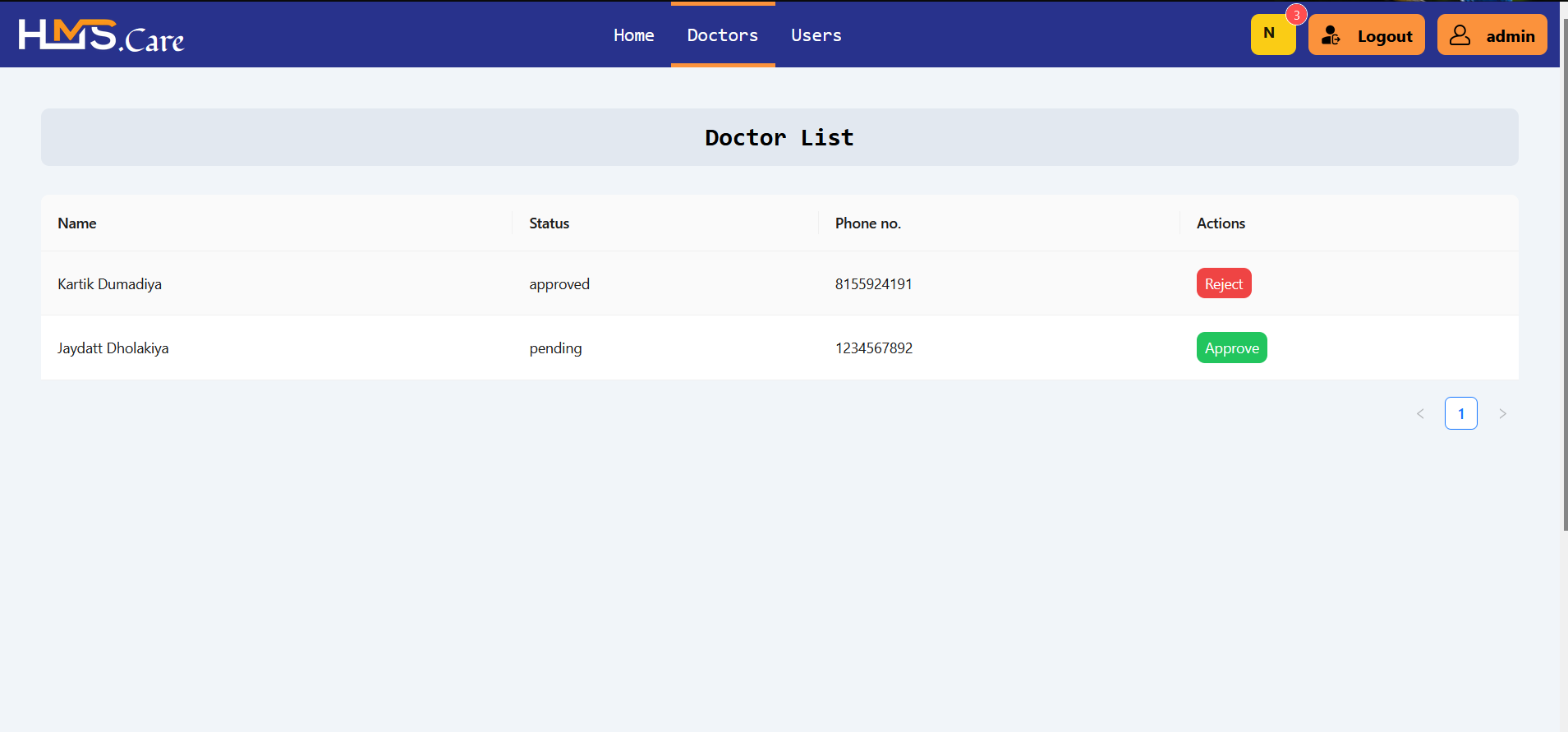
**Sign in Page**

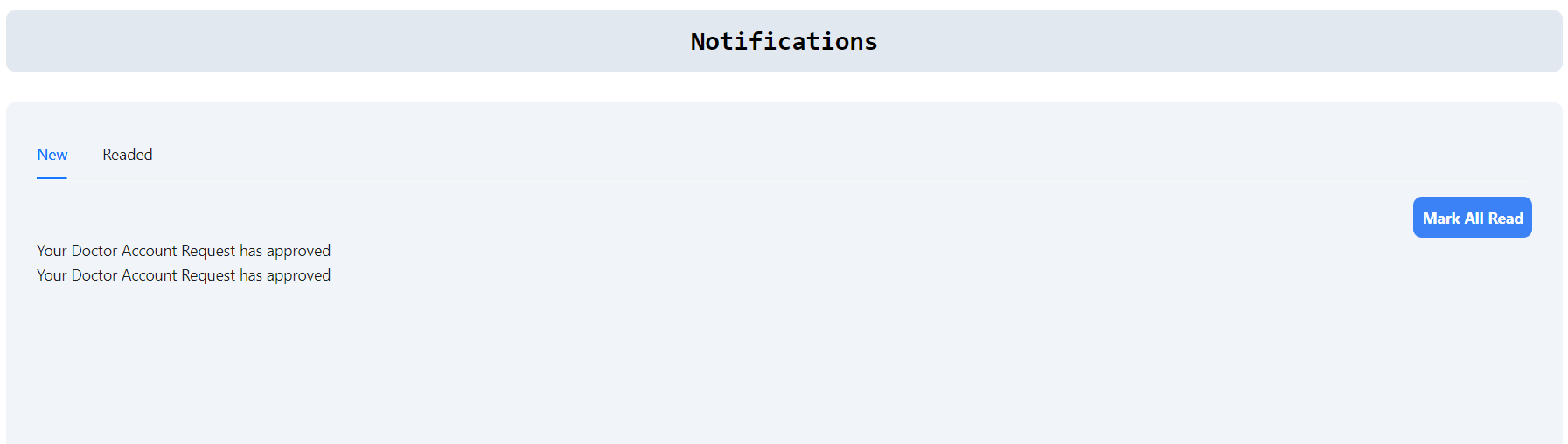
****

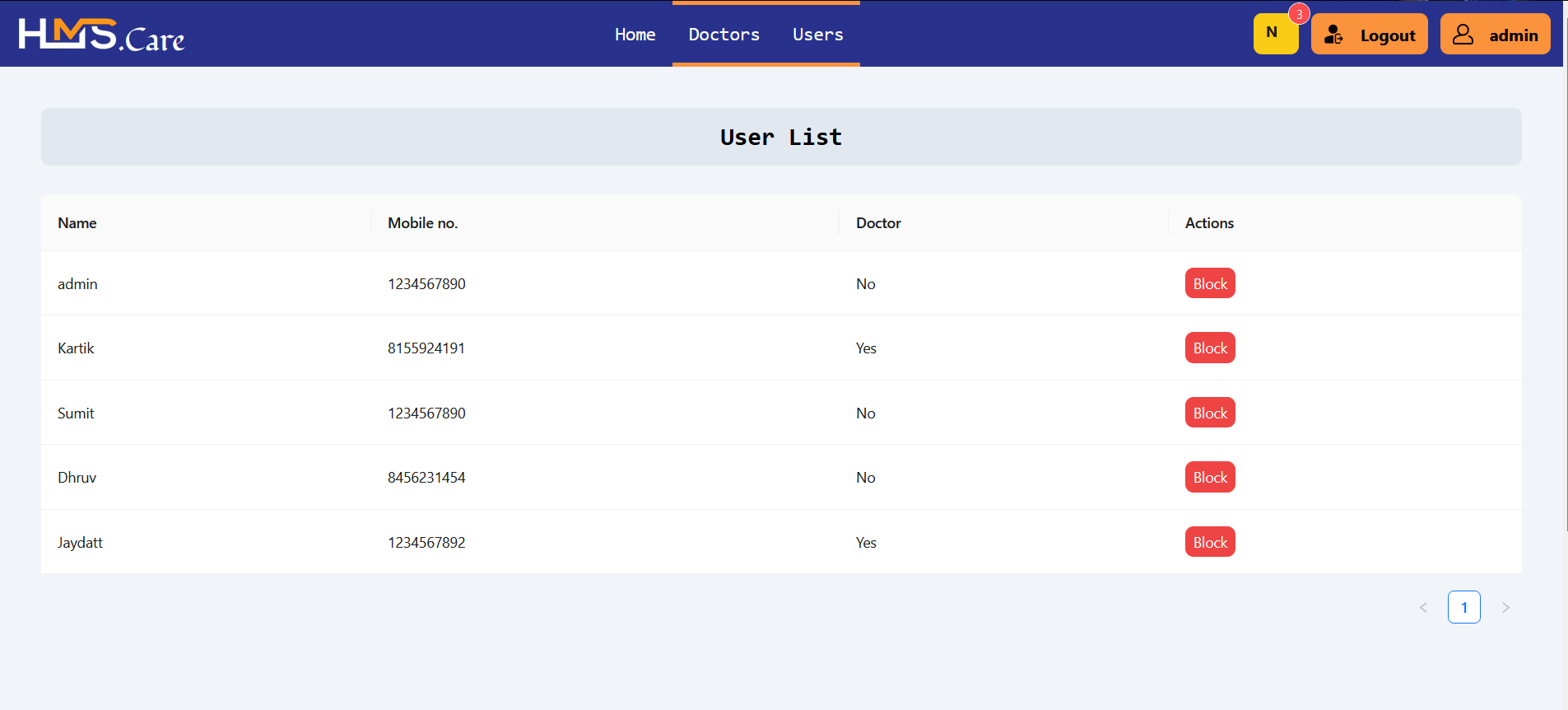
**Applying of Doctor:**

****

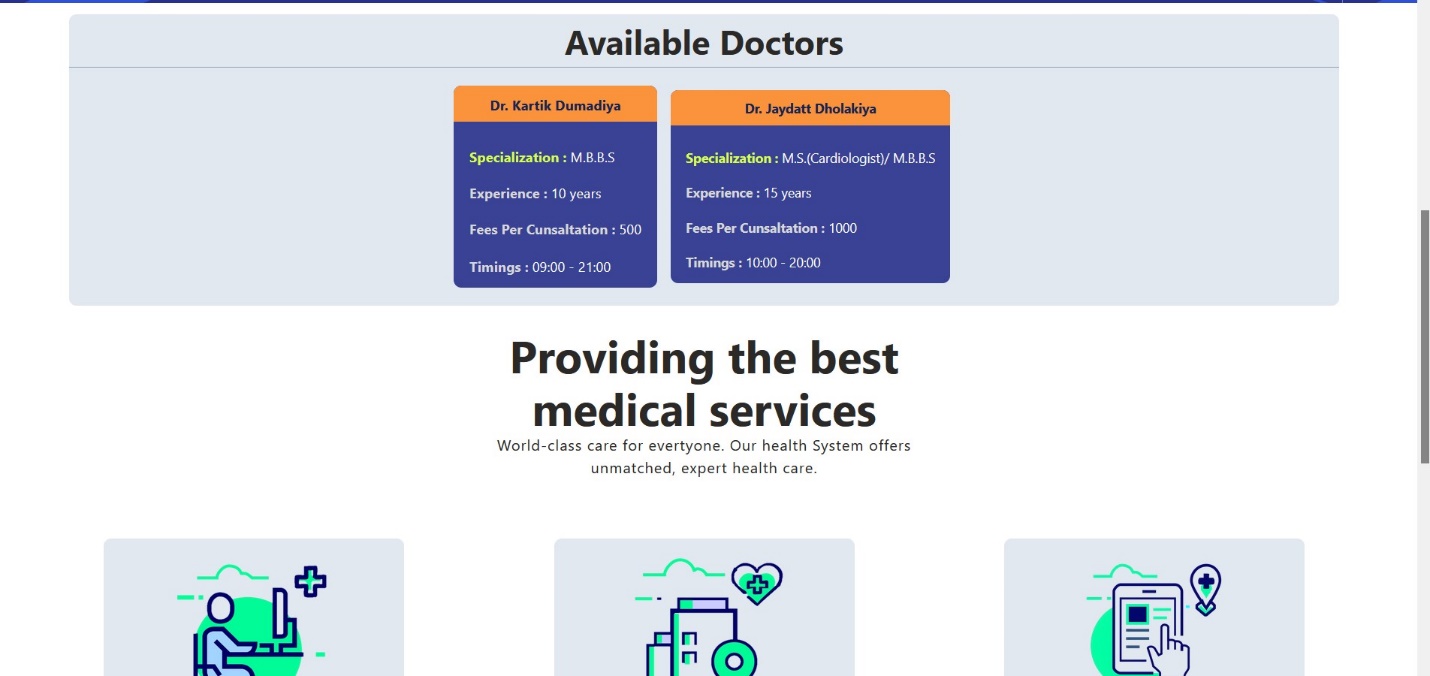
**Admin Notification Page And Approval Portal:**

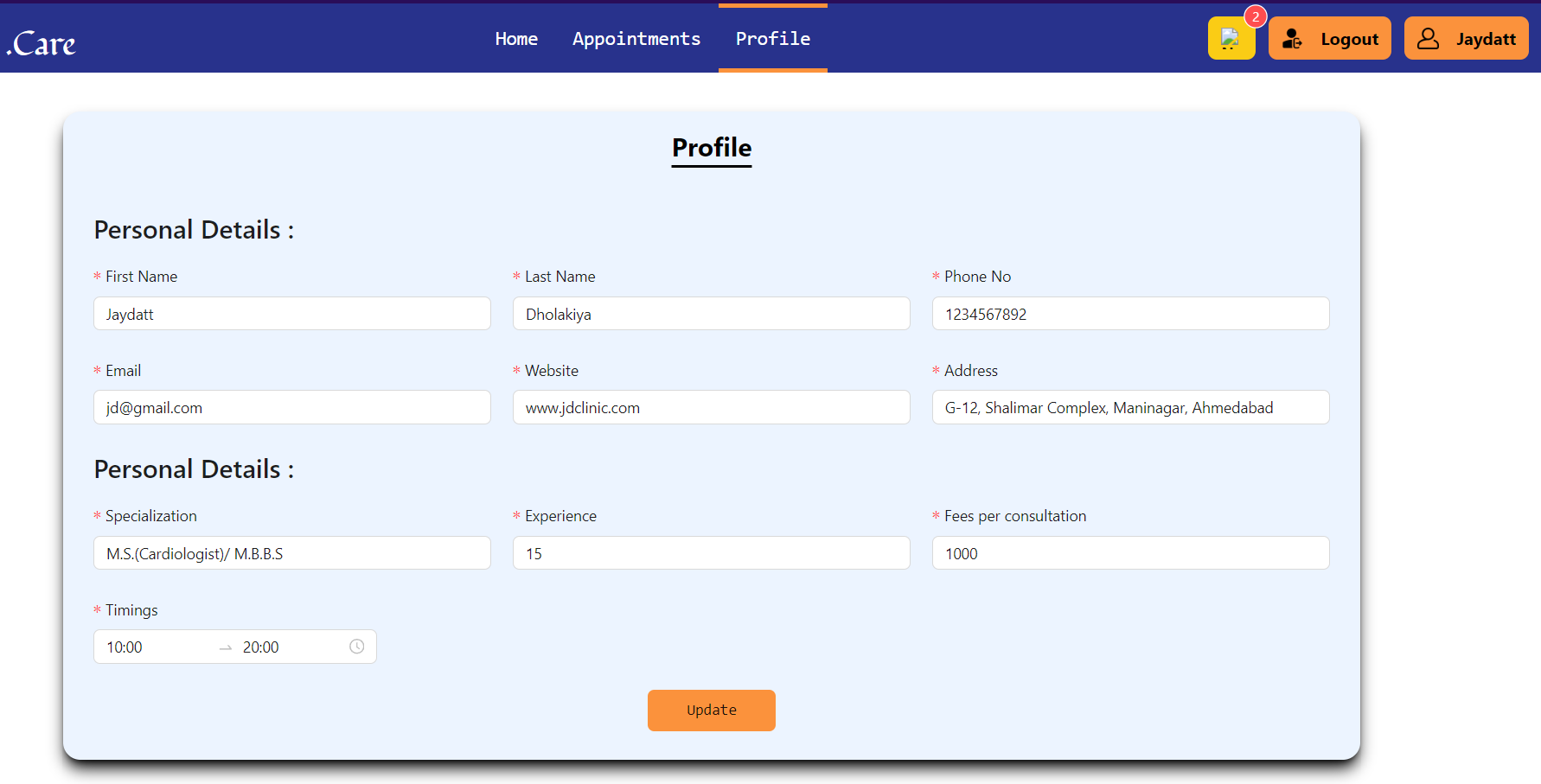
****

**Doctor’s Approval Notification:**

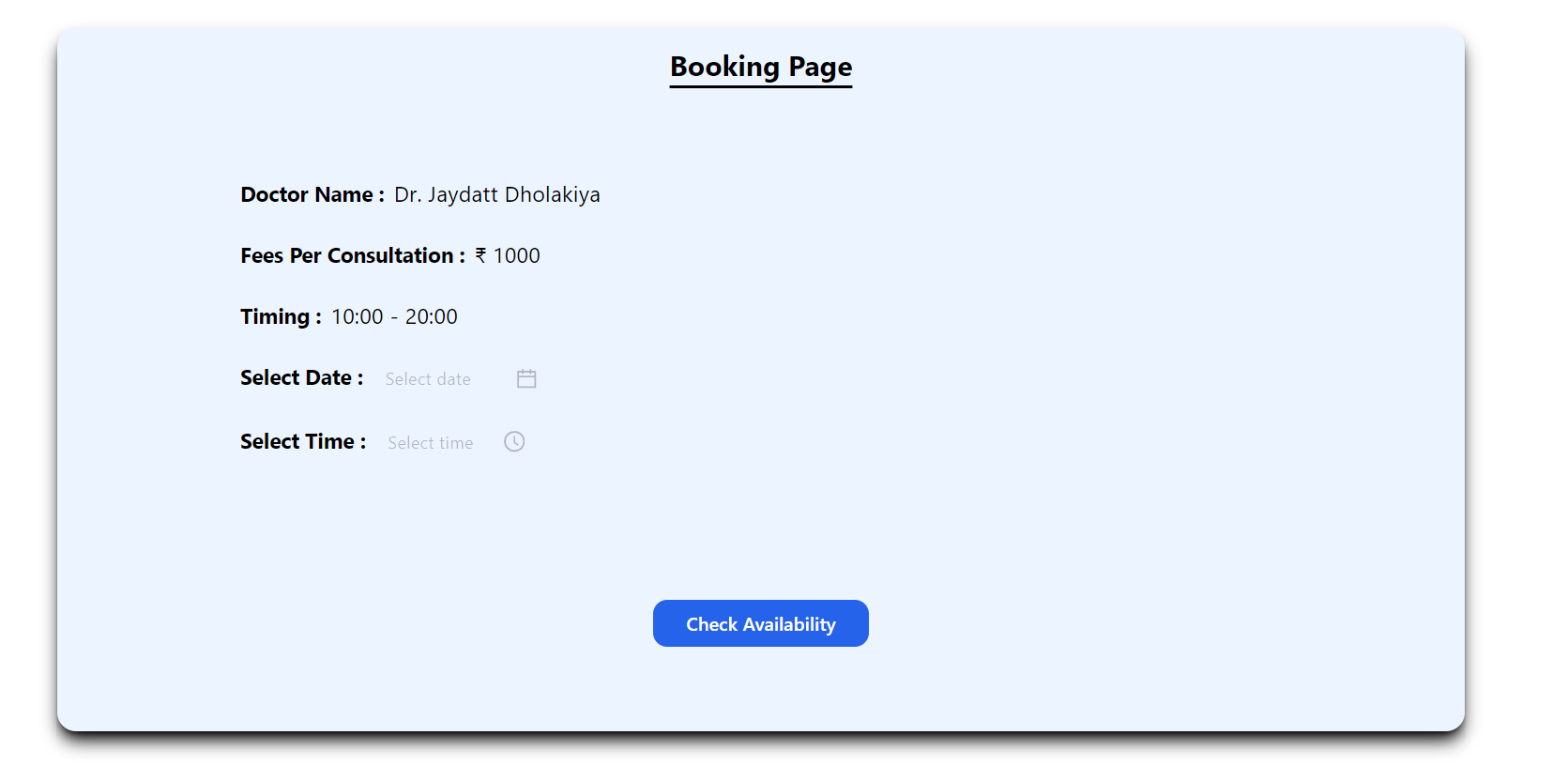
**User List:**

**Displaying Available Doctors:**

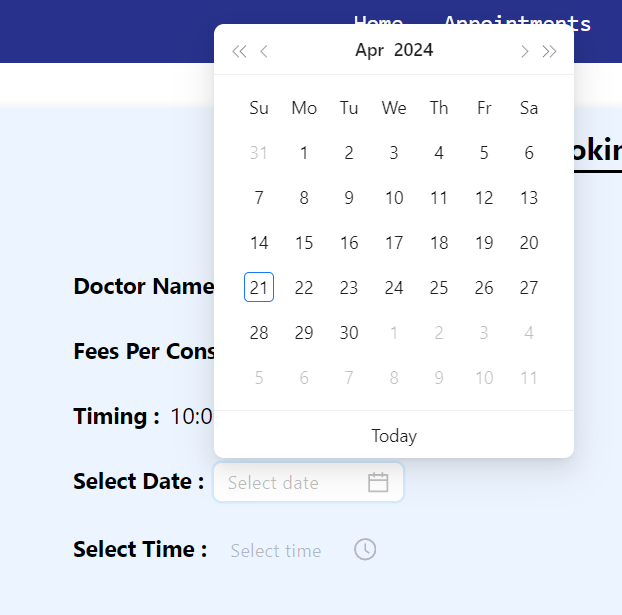
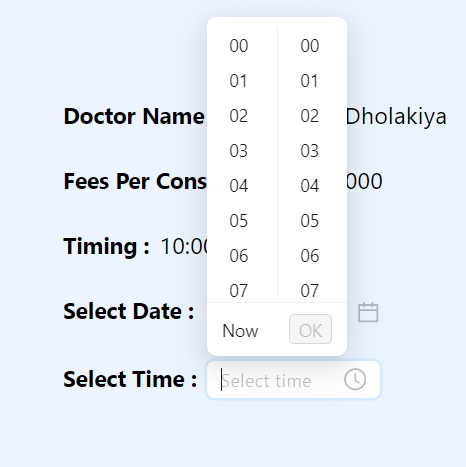


**Doctors Can Update Their Profile:**

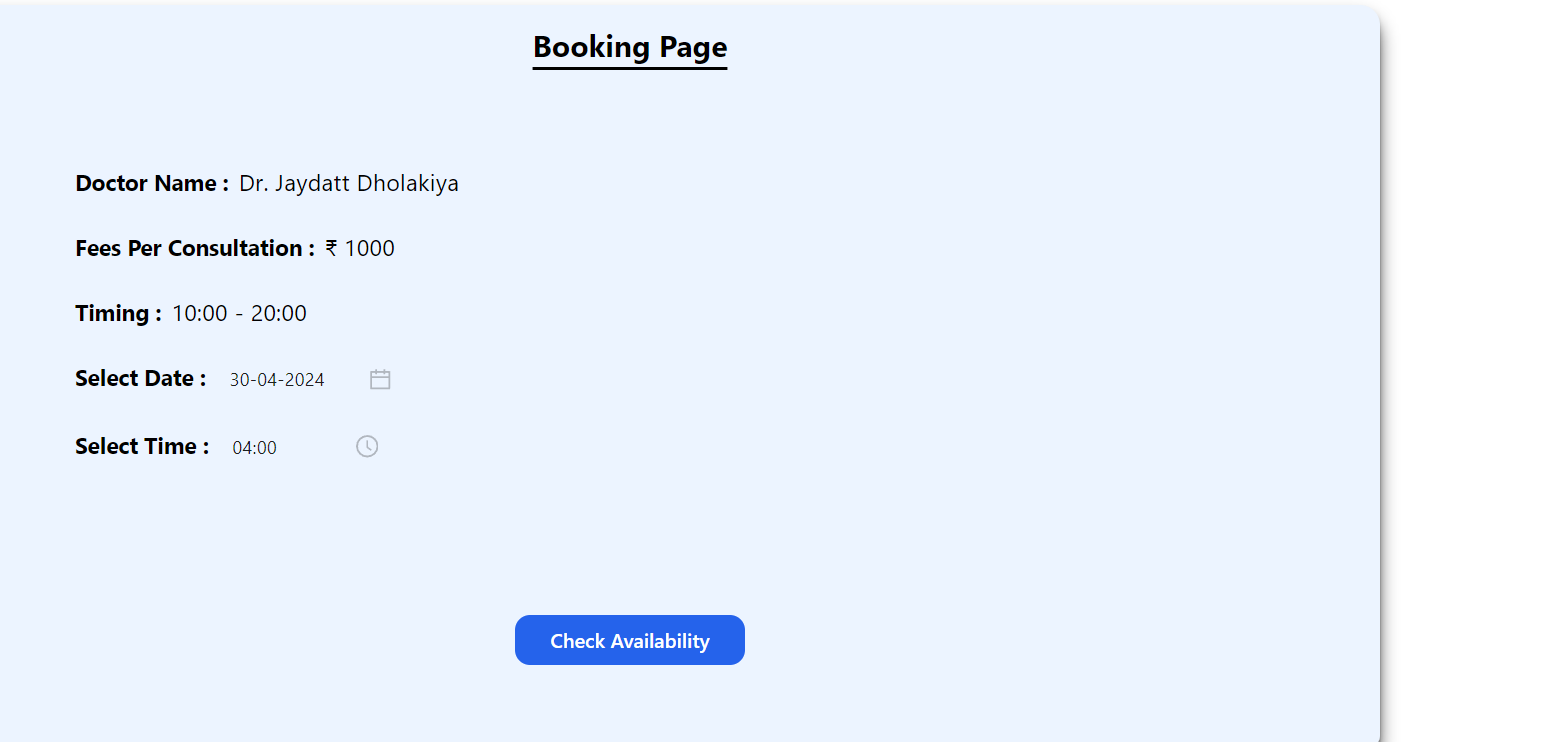
**Appointments Page:**

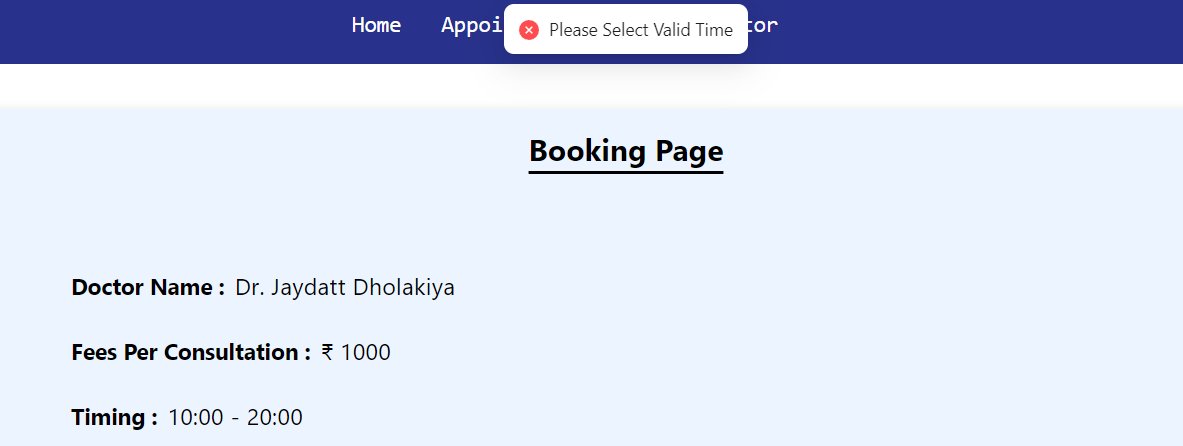
****

**Selecting Appropriate Time and Date:**

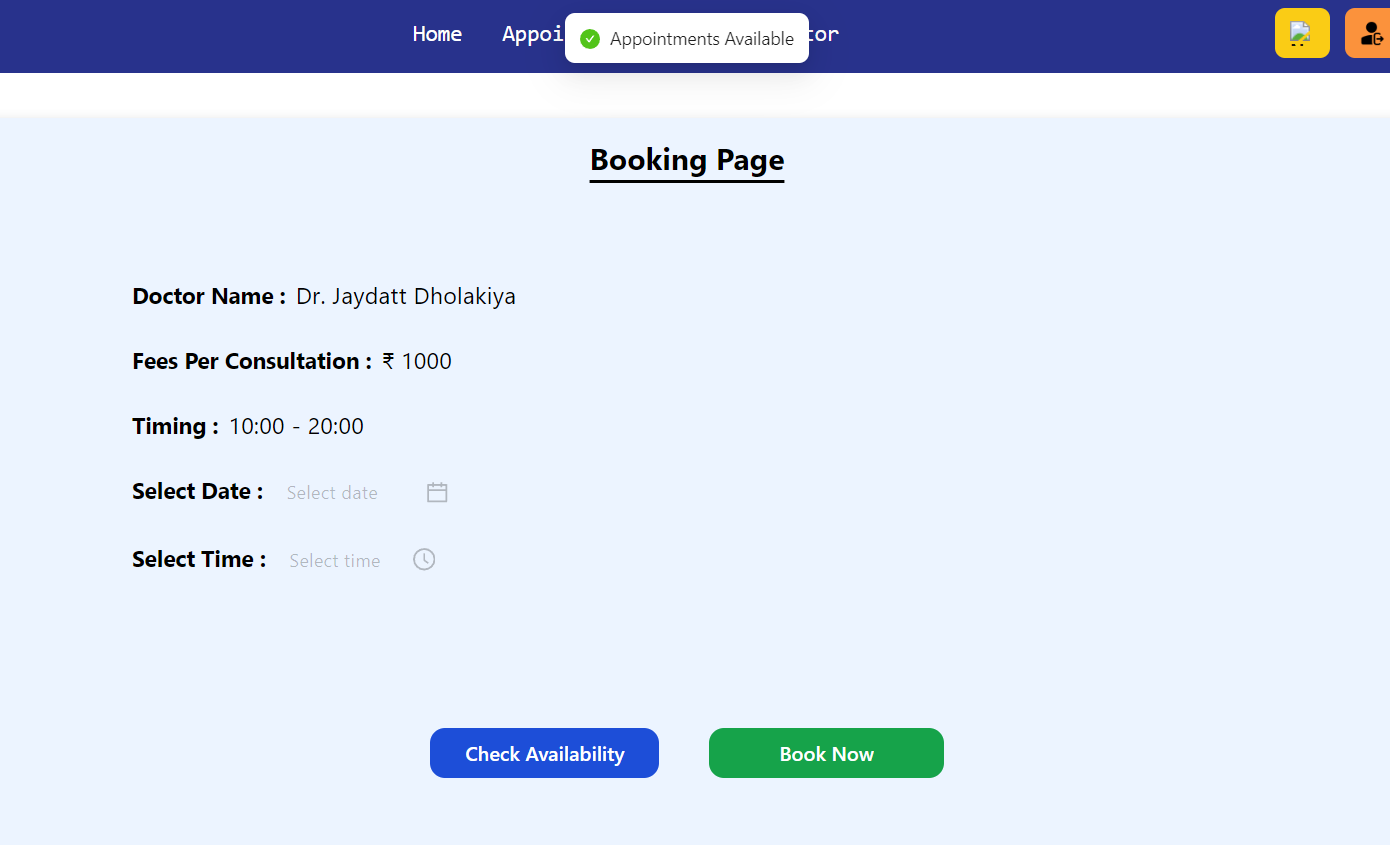
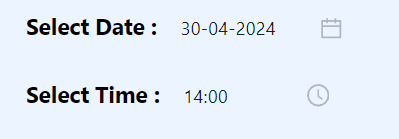
****

**Checking Of Valid Time (Check Availability):**

****

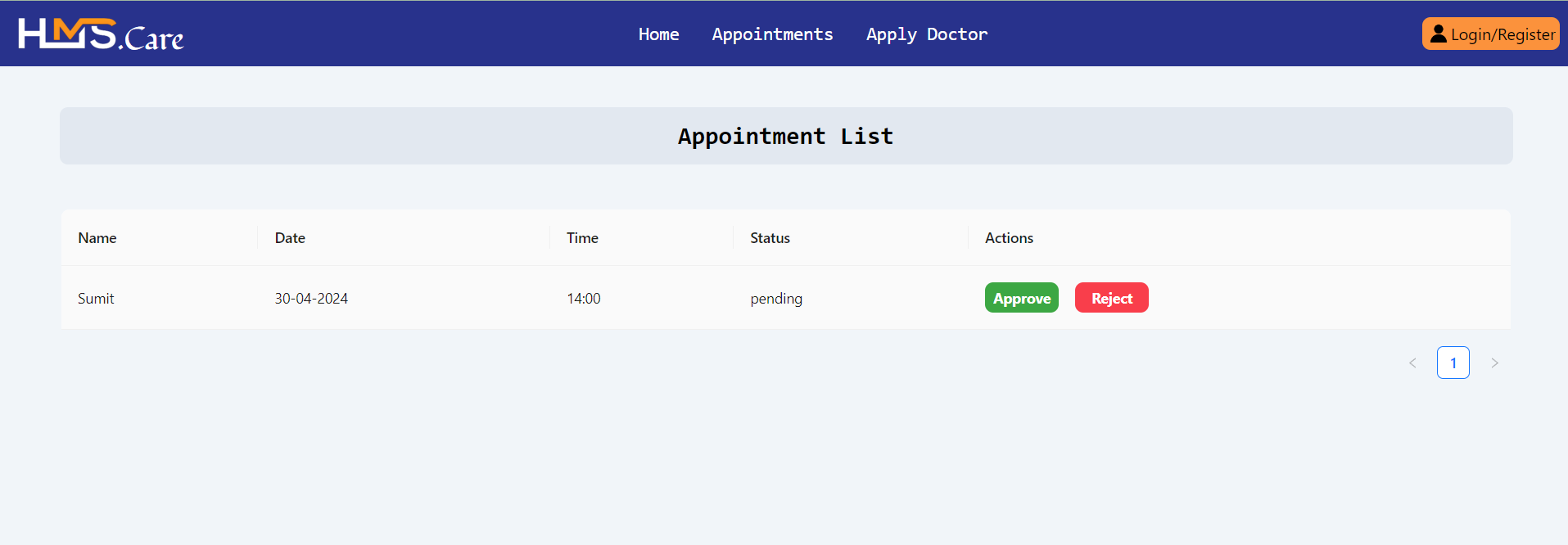


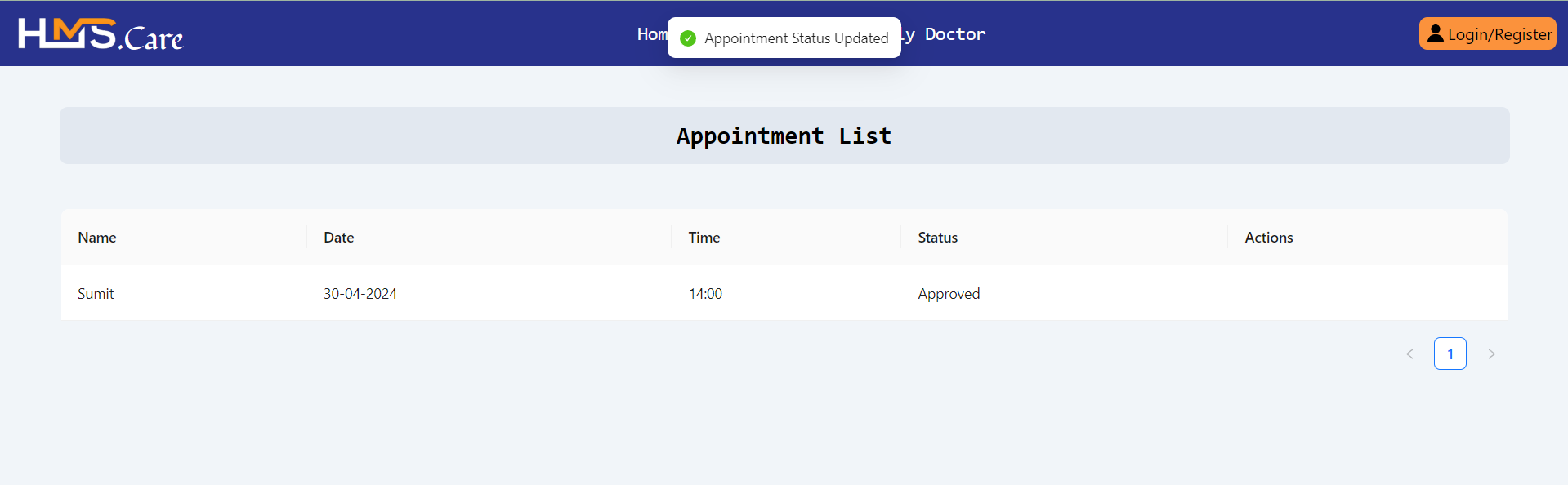
**If Appointment Is Available**:



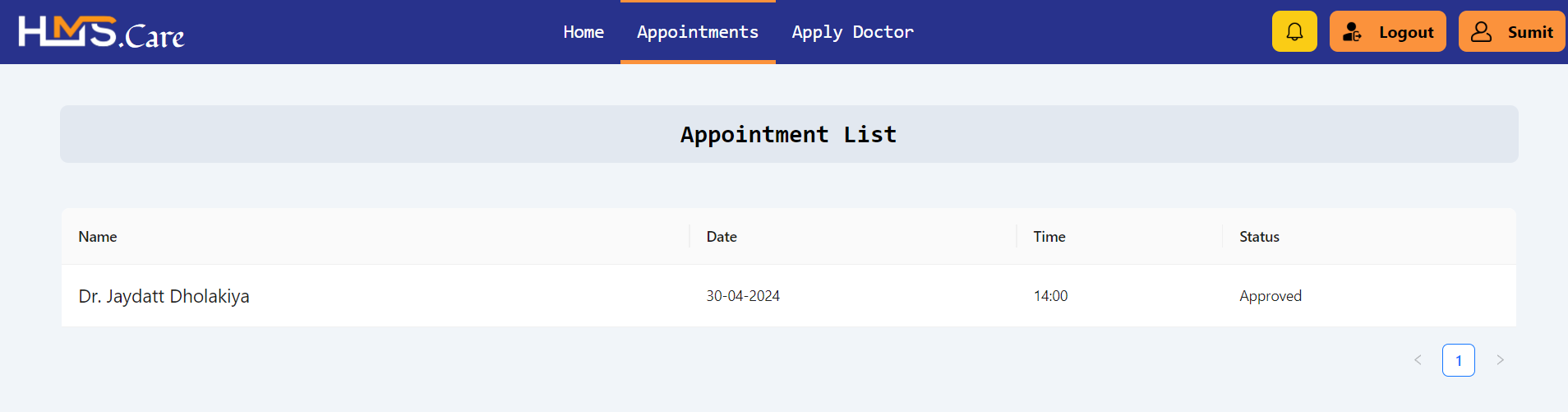
**After Requesting Appointment :**

**Doctor’s Side:**

****

****

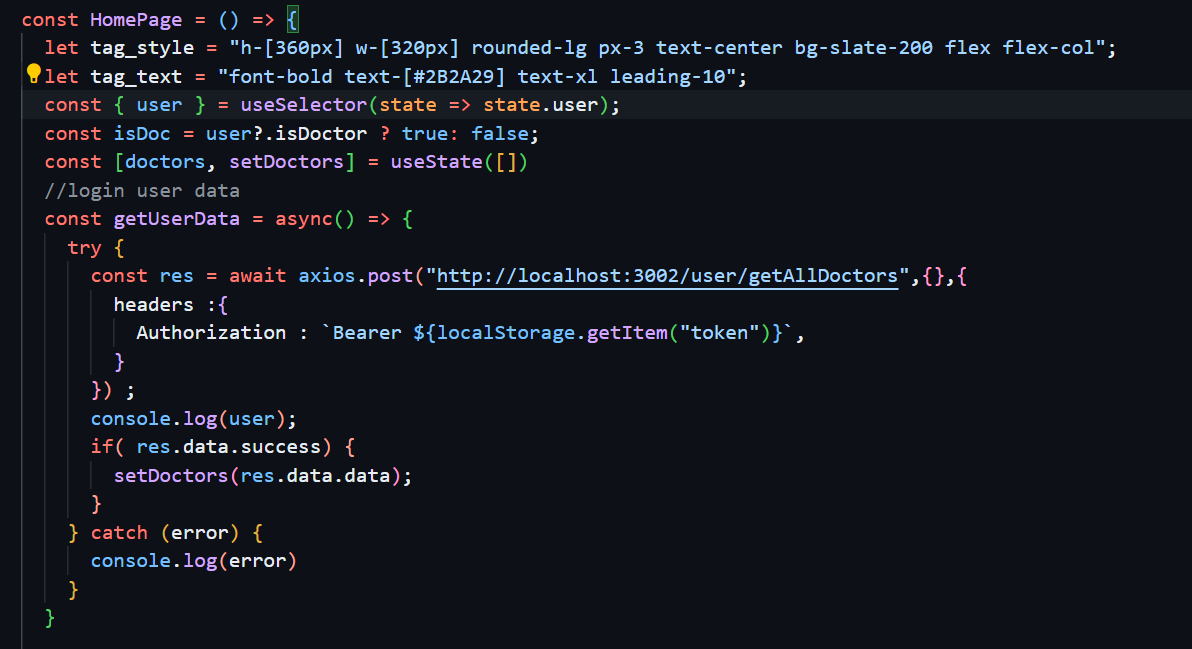
**Patient Side:**

****

**Security Measures**

JWT Tokens:

* Implemented JSON Web Token (JWT) authentication to secure the website and ensure that only authenticated users can access protected routes and resources.
* JWT tokens are issued upon successful user authentication and contain encoded user information that includes user account number.
* Each subsequent request to the server includes the JWT token in the authorization header, allowing the server to verify the user's identity and permissions.
* Enhances security by preventing unauthorized access to sensitive data and functionalities, reducing the risk of data breaches and unauthorized actions.
* Facilitates seamless user authentication and authorization processes, providing a reliable mechanism for maintaining user sessions and protecting user privacy.
* Client Side



* Server Side

import JWT from "jsonwebtoken";

export default 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 Failed",

                    success : false

                })

            }else {

                // console.log(decode);

                req.body.userId = decode.id

                next();

            }

        })

    } catch (error) {

        console.log(error)

        res.status(401).send({

            message: 'Auth Failed',

            success : false

        })

    }

};

**Results of Authentications:**



Encryption:

* Employed bcrypt.js, a cryptographic library, to securely hash and encrypt sensitive user data such as Passwords.
* Utilized bcrypt's one-way hashing algorithm to convert plain text Passwords into irreversible hash values before storing them in the database.
* Ensured that even if the database is compromised, attackers cannot reverse-engineer the hashed Passwords to obtain the original values.

**Conclusion**

Our Hospital Management System represents a pivotal advancement in healthcare administration and patient care. By harnessing the power of modern technology and the flexibility of the MERN stack, we have created a robust platform that streamlines operations, enhances efficiency, and improves outcomes for both healthcare providers and patients. With user-friendly interfaces, comprehensive features, and a commitment to continuous innovation, our system stands at the forefront of digital healthcare solutions. As we continue to evolve and incorporate feedback from users and stakeholders, we remain dedicated to revolutionizing healthcare delivery and making a positive impact on the lives of individuals and communities worldwide.

**Future Enhancements**

1. **Enhanced Appointment Booking:**
   * Implement features like appointment reminders through email or SMS.
   * Enable patients to cancel or reschedule appointments online.
2. **Electronic Health Records (EHR) Management:**
   * Develop a module to store and manage patient health records securely.
   * Include features for doctors to update patient records, view medical history, and add prescriptions.
3. **Billing and Invoicing System:**
   * Integrate a billing system to generate invoices for services provided.
   * Include features for tracking payments, generating receipts, and managing billing cycles.
4. **Telemedicine Integration:**
   * Implement video conferencing capabilities for remote consultations between doctors and patients.
   * Ensure compliance with privacy and security regulations for telemedicine.
5. **Medication Management:**
   * Develop a module for managing medication prescriptions and refills.
   * Include features for tracking medication inventory and generating alerts for low stock.
6. **Real-Time Dashboard and Analytics:**
   * Create a dashboard for administrators to monitor key metrics such as appointment bookings, patient demographics, and resource utilization.
   * Implement analytics tools to identify trends, optimize resource allocation, and improve operational efficiency.
7. **Mobile Application:**
   * Develop a mobile app for patients to access the system on their smartphones.
   * Include features like appointment booking, medication reminders, and access to medical records.

**References**

* <https://docpulse.com/>
* [Vite | Next Generation Frontend Tooling (vitejs.dev)](https://vitejs.dev/)
* [Express - Node.js web application framework (expressjs.com)](https://expressjs.com/)
* [Documentation | Framer for Developers](https://www.framer.com/motion/)
* [Getting Started | Axios Docs (axios-http.com)](https://axios-http.com/docs/intro)
* [mongodb.com](https://www.mongodb.com/)
* [youtube.com - CodeWithHarry](https://www.youtube.com/@CodeWithHarry)
* [Nodemailer :: Nodemailer](https://nodemailer.com/)