**DocSpot: Online Doctor Appointment Booking System**

**Project Documentation**

**Team Details:**

**Team ID** : LTVIP2025TMID54356

22PA1A05G5 – Thumu Vasthavi Sreya

22Pa1A45B5 - Velthurla Venkata Ramya

23Pa1A04H3 - Vasadi Yesu Karthik

23Pa5A0431 - Varikallu Dharani

**Introduction:**

DocSpot is a full-stack MERN (MongoDB, Express.js, React, Node.js) web application designed to modernize and streamline the healthcare appointment booking process. In today's fast-paced digital age, convenience and accessibility in healthcare services are crucial. DocSpot aims to bridge the gap between patients and doctors by providing a platform where users can easily search for medical professionals, schedule appointments, and manage their bookings online. The application includes distinct roles such as Patients, Doctors, and Admins, each having tailored functionalities and dashboards.

This project showcases a comprehensive implementation of modern web development practices and tools, emphasizing role-based access control, real-time user interactions, and scalable architecture.

**Project Goals and Objectives:**

The primary goals of the DocSpot project are:

* To create a seamless user experience for booking doctor appointments.
* To provide a scalable and maintainable full-stack application.
* To ensure secure and efficient management of user data.
* To implement real-world features such as notifications, admin approval systems, and appointment scheduling.

**System Architecture and Technology Stack:**

DocSpot follows a client-server architecture where the frontend (client) communicates with the backend (server) through RESTful APIs. Here's a detailed look at the technologies used:

**Frontend:**

* React.js for UI development
* Redux Toolkit for state management
* Material UI for design components
* Formik and Yup for form handling and validation
* Axios for HTTP requests
* React Phone Input and Moment.js for specialized input and formatting

**Backend:**

* Node.js with Express.js framework
* MongoDB for NoSQL database management
* Mongoose as the ODM for MongoDB
* JSON Web Tokens (JWT) for secure authentication

**Other Tools:**

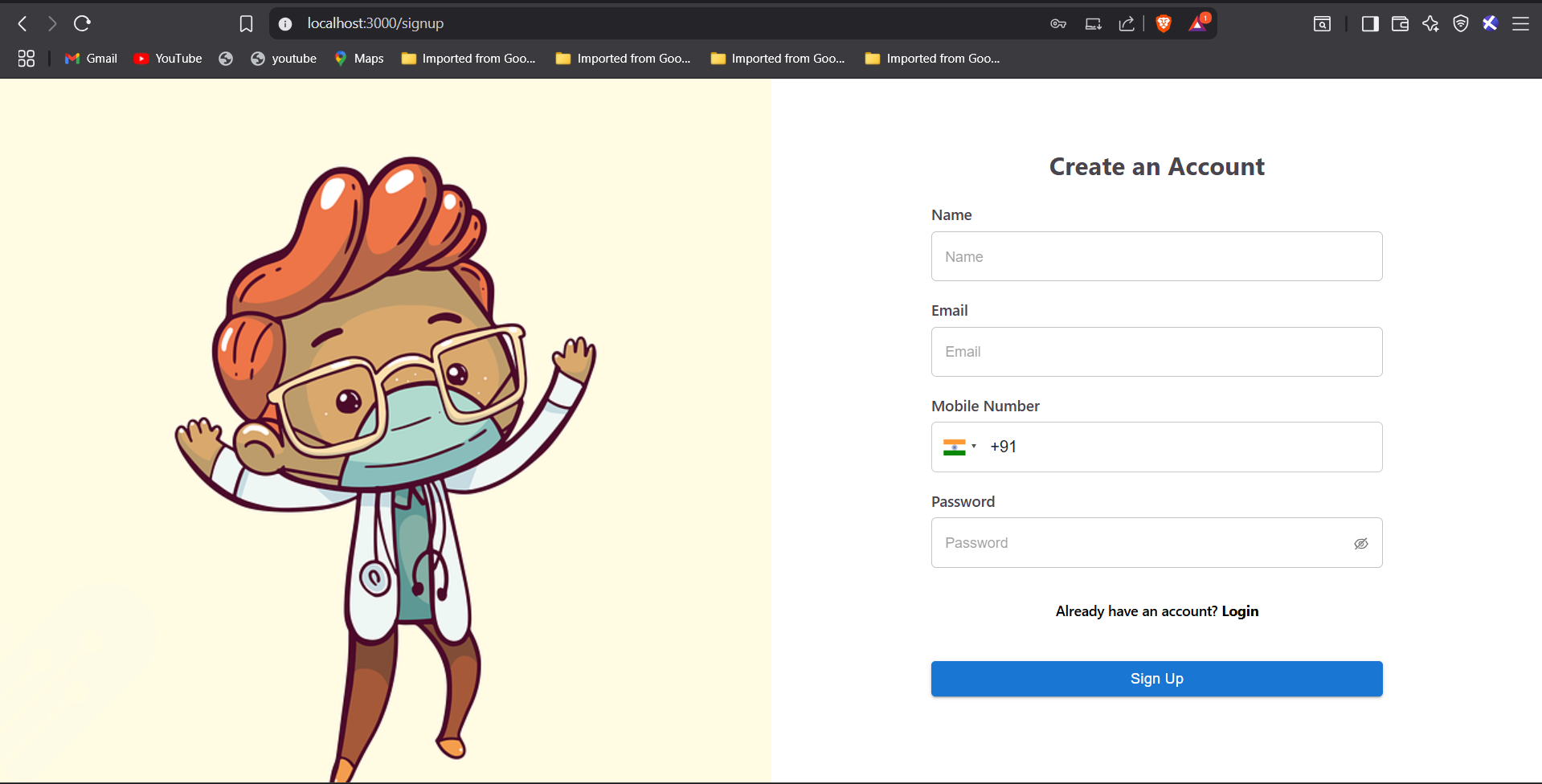
* Postman for API testing
* ESLint for code quality
* Git for version control

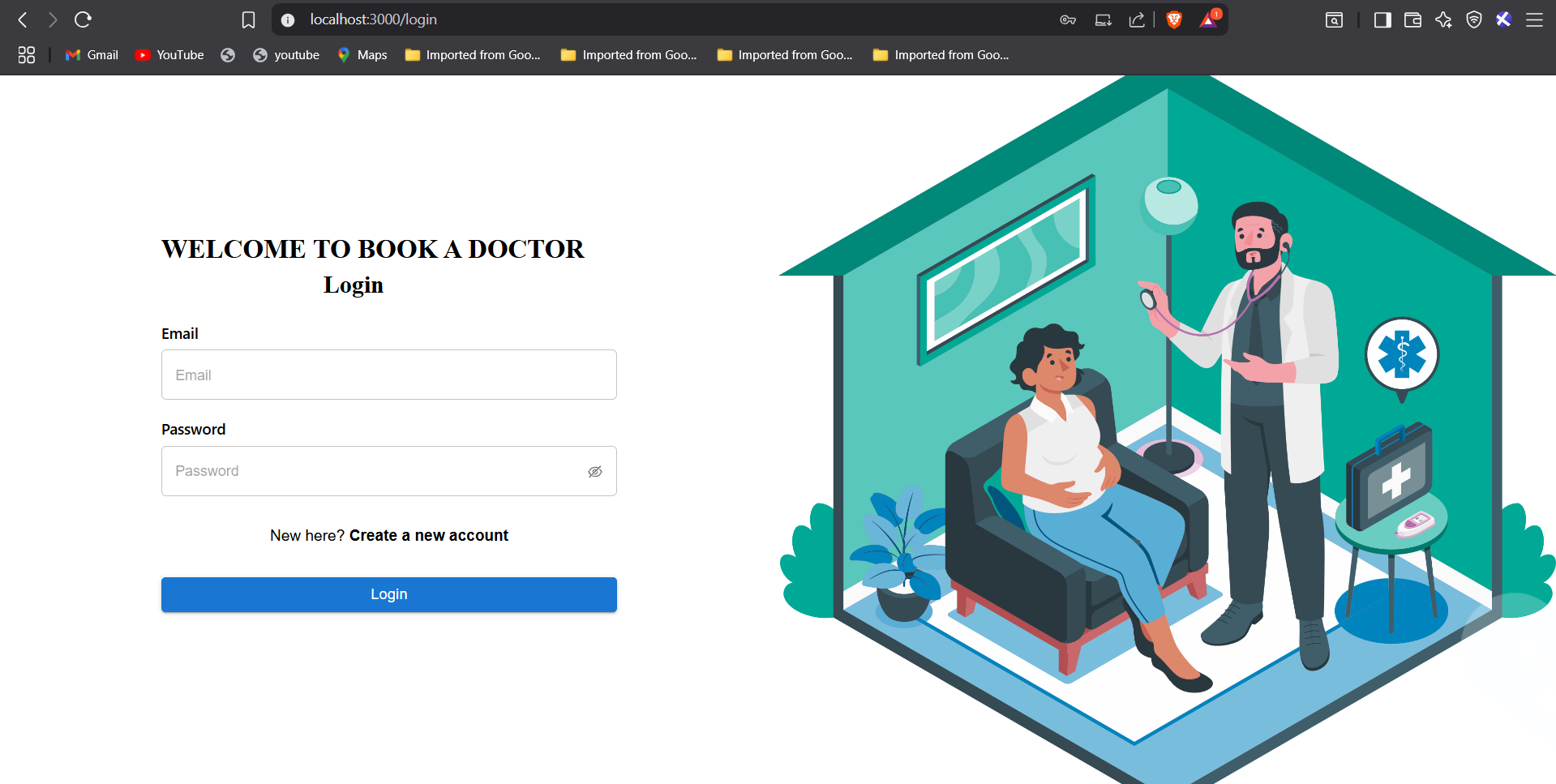
**Features and Functionalities:**

DocSpot offers a variety of features across three user roles:

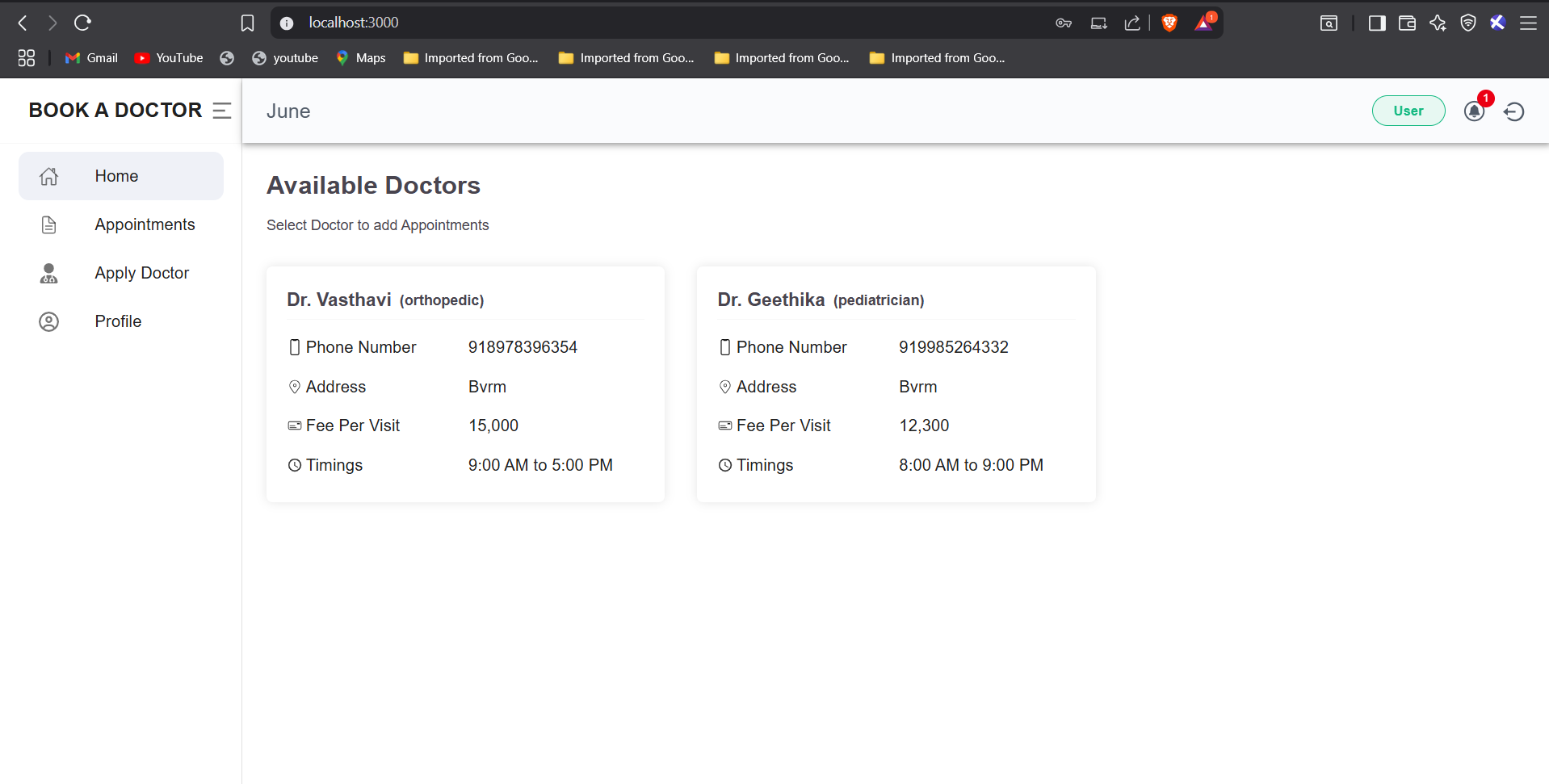
**Patient Functionality:**

* Register and log in

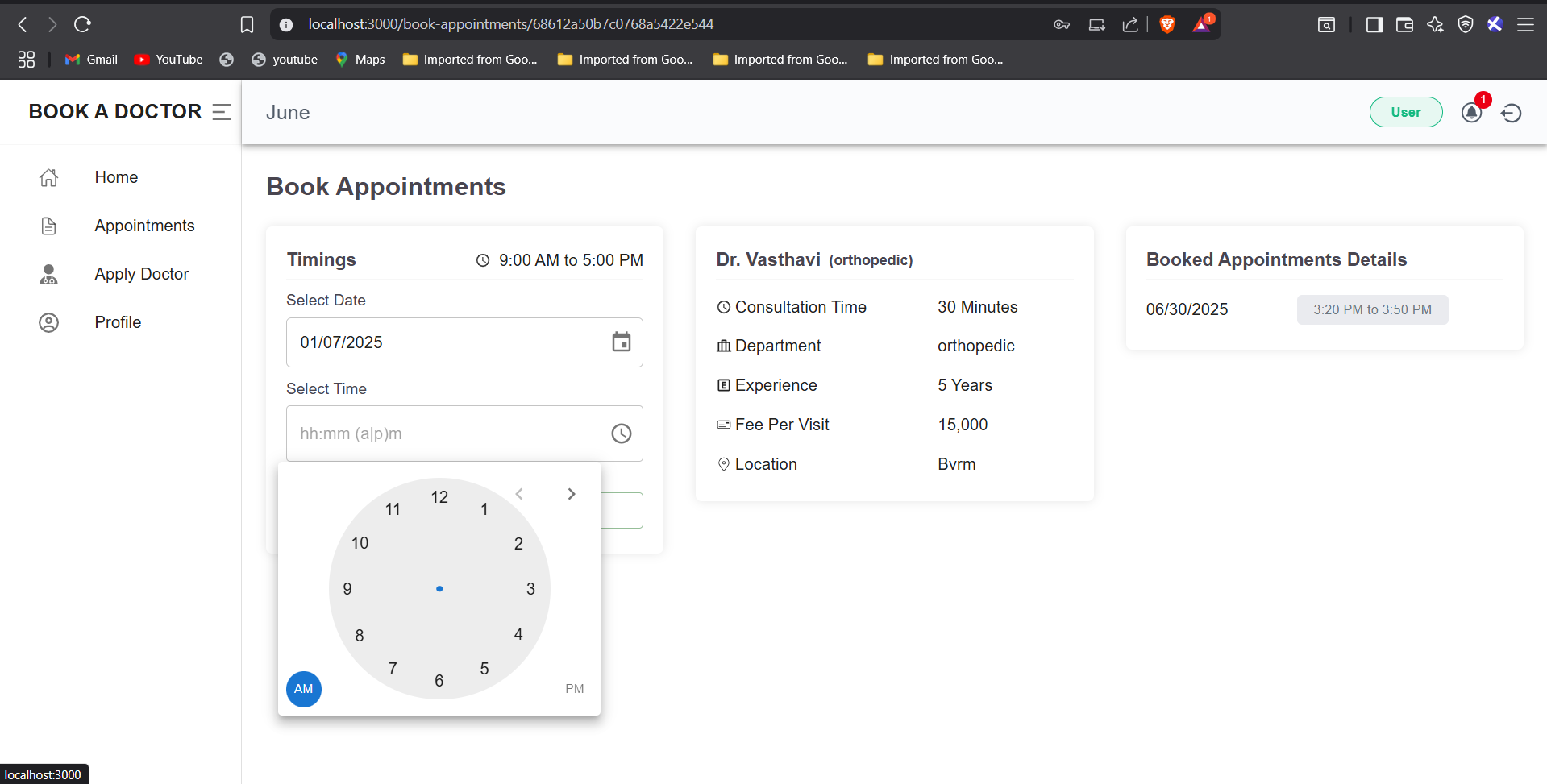




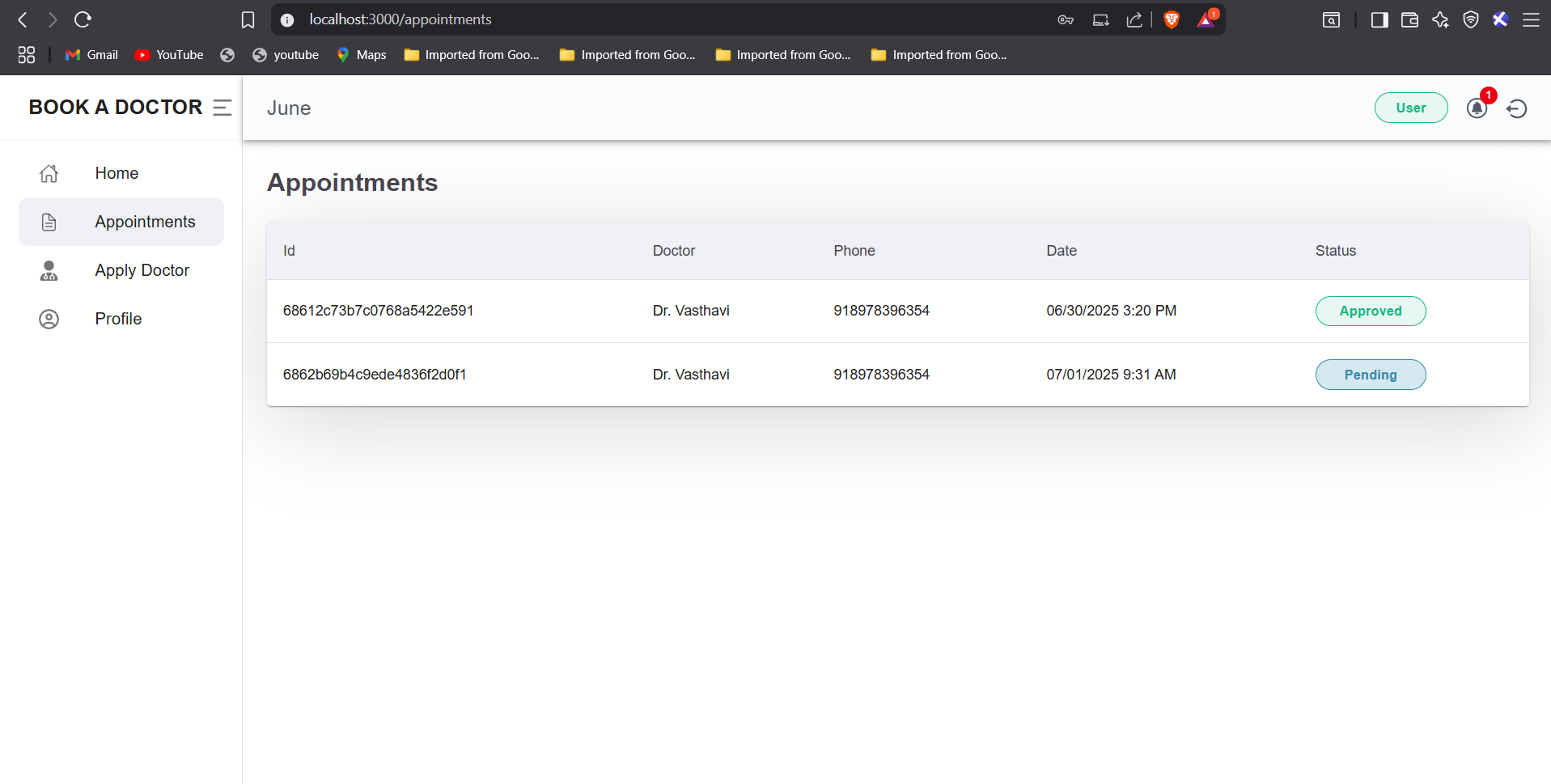
* Search and view doctors by specialization



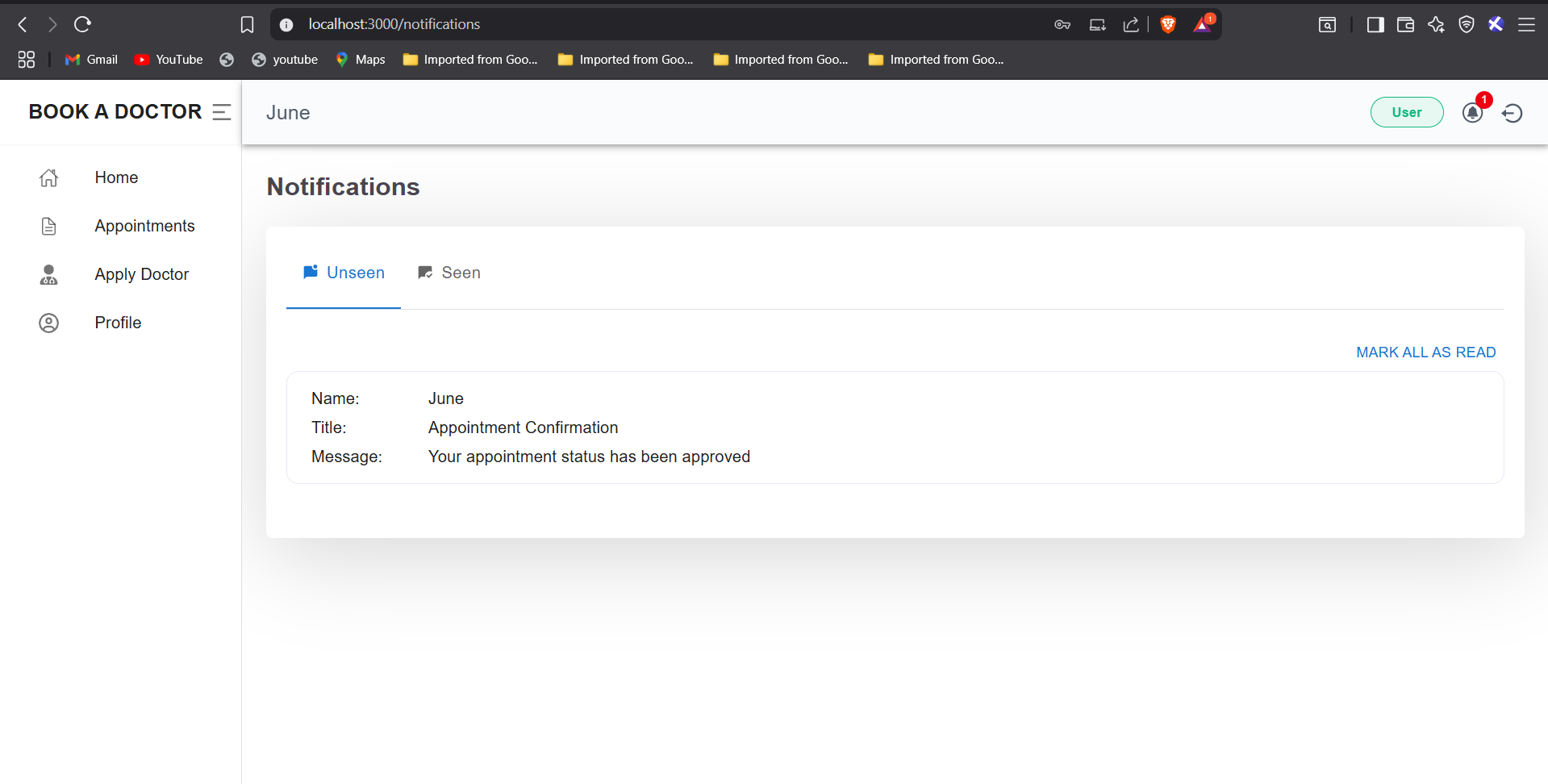
* Book appointments with available doctors



* View personal appointment history

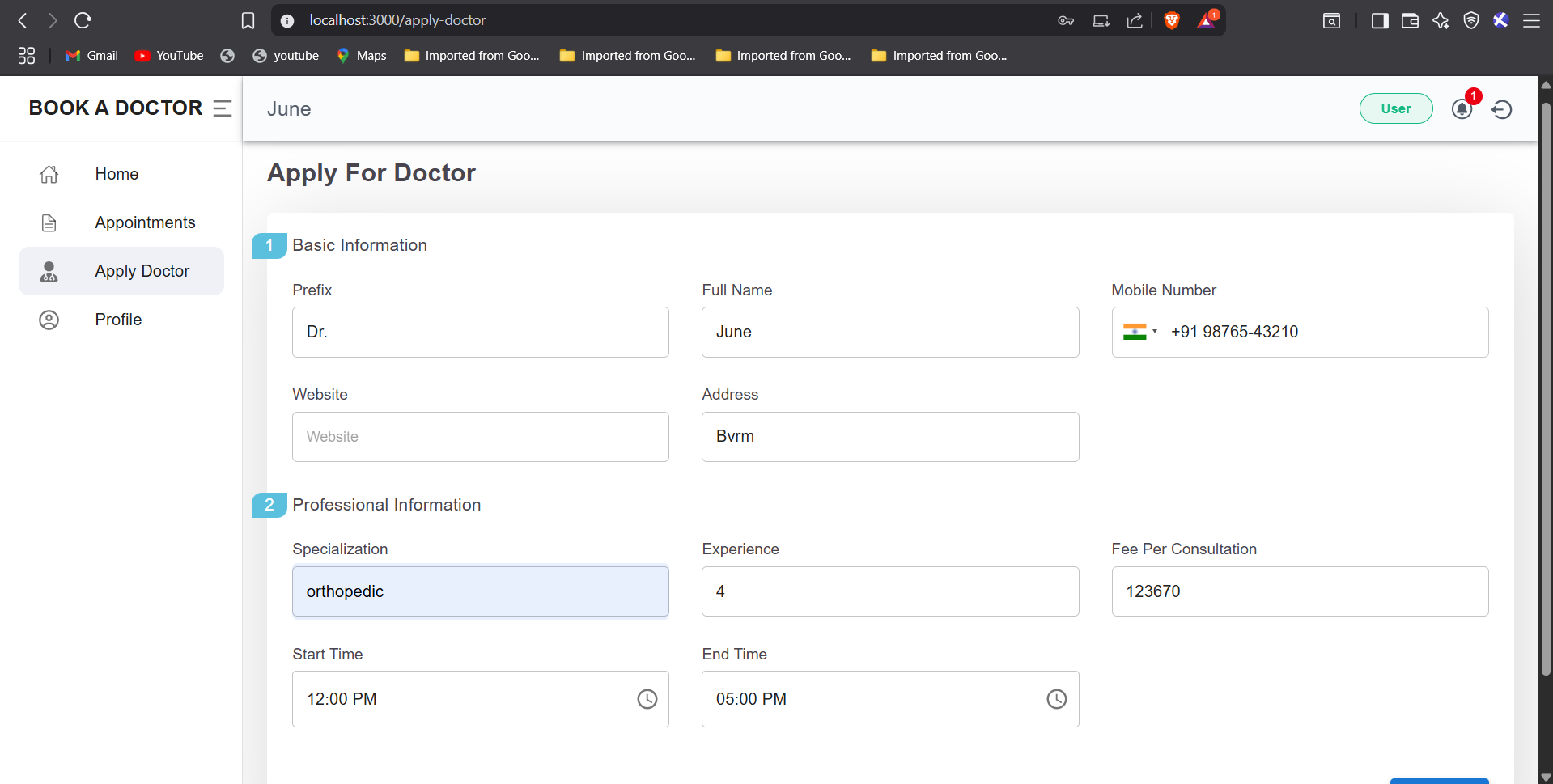


* Receive and manage notifications

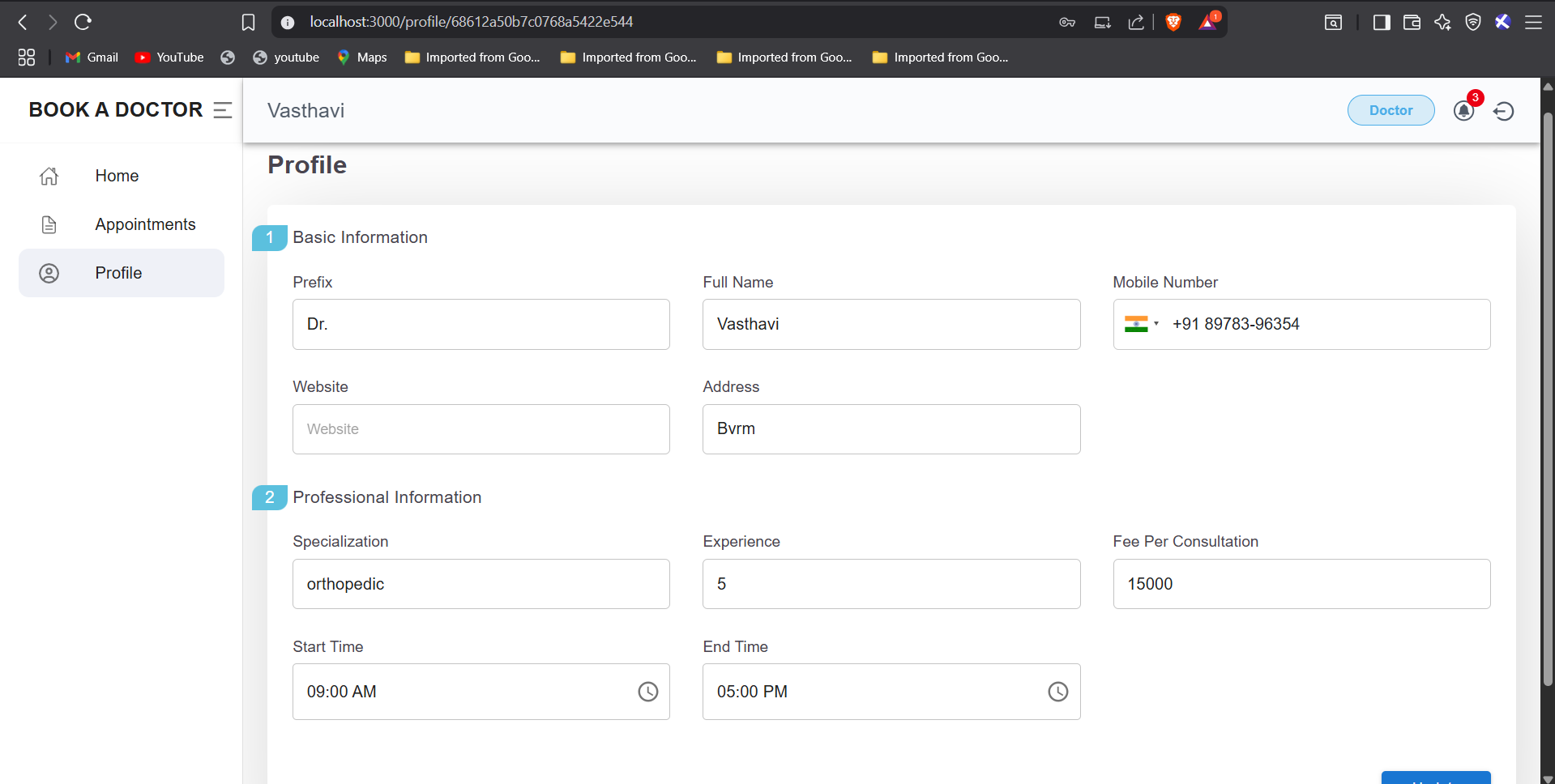


**Doctor Functionality:**

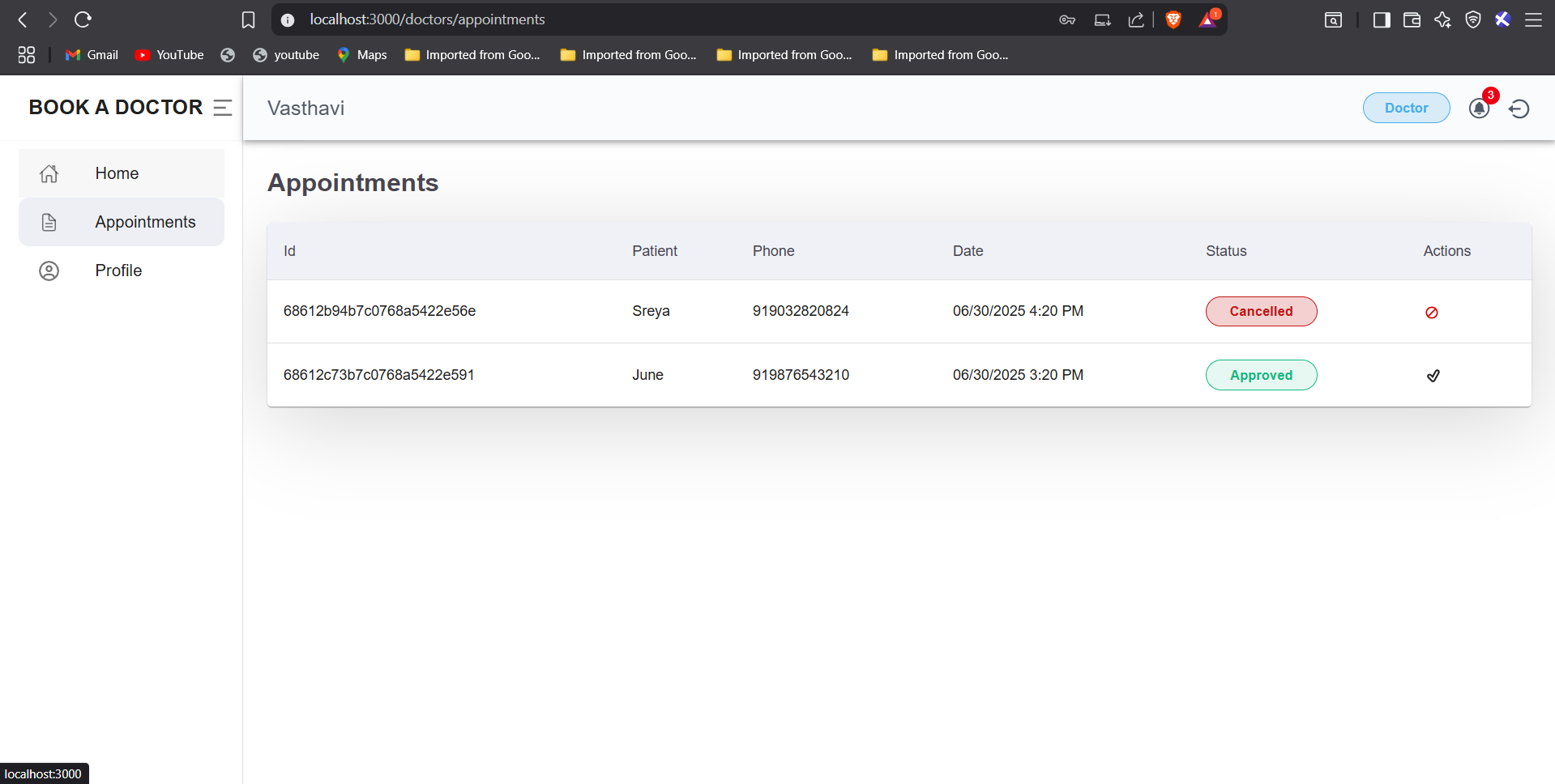
* Register and submit application for approval



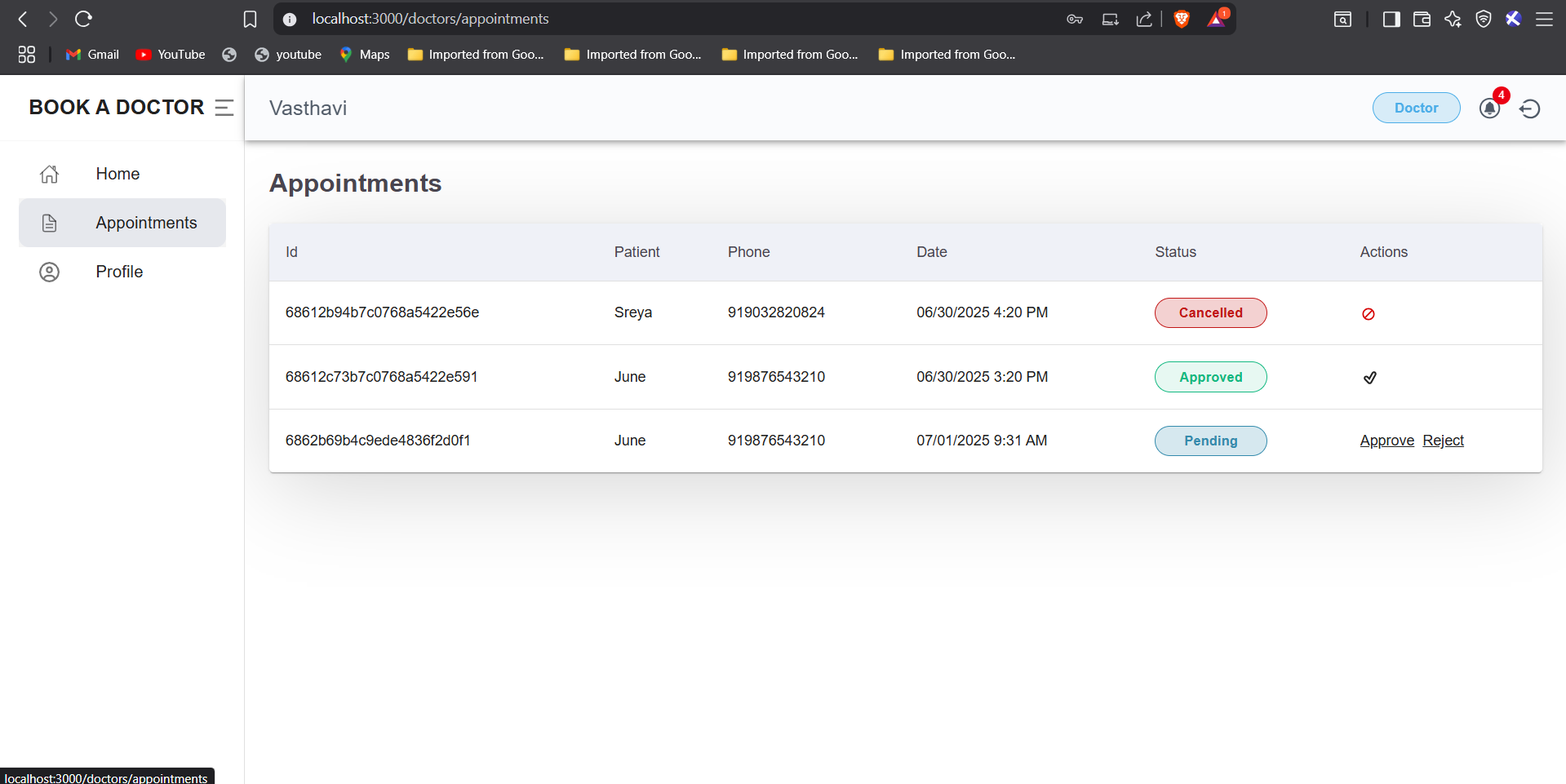
* Manage profile and availability schedule



* View booked appointments

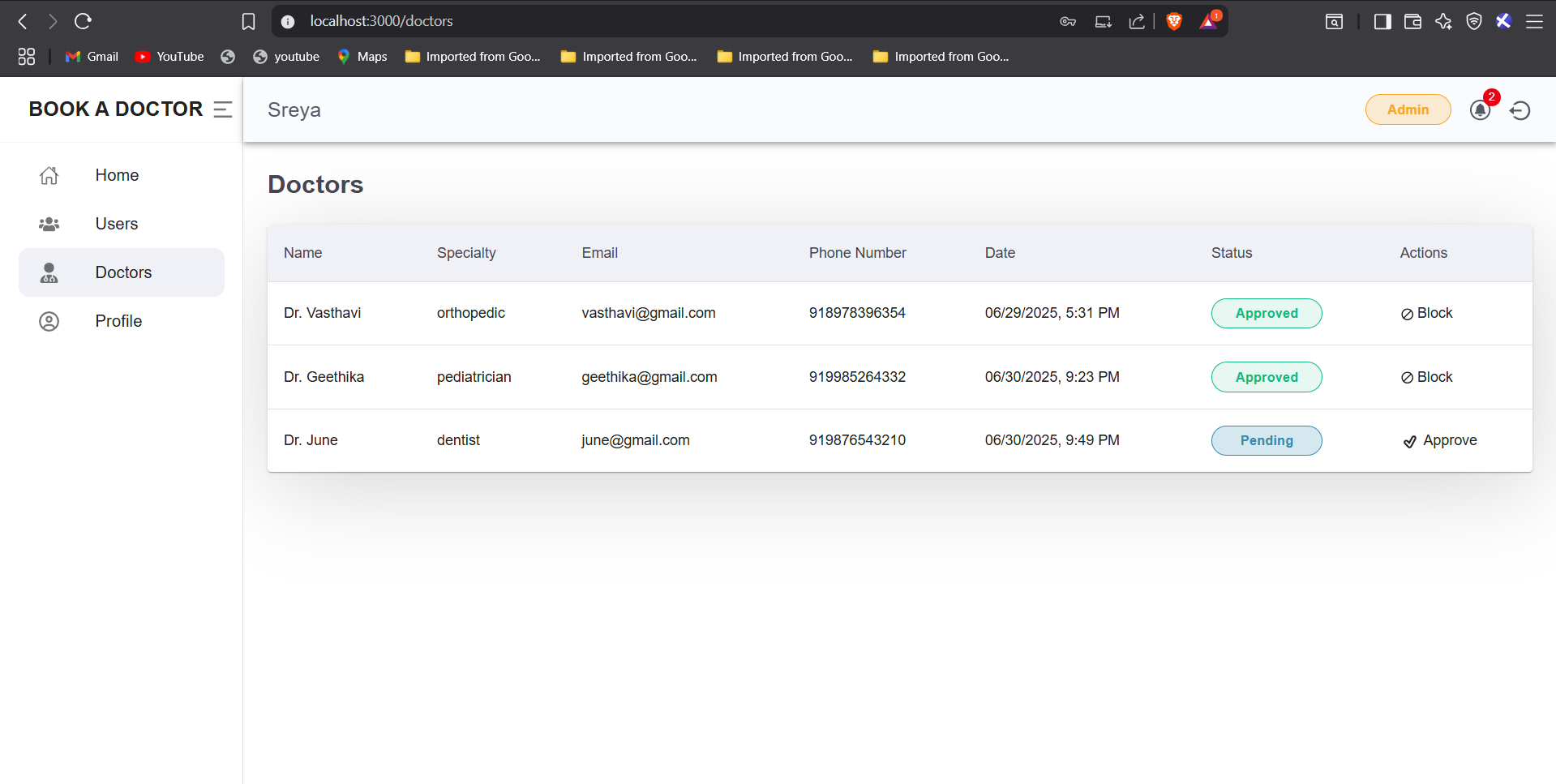


* Approve or reject patient bookings- Access dashboard for appointment analytics

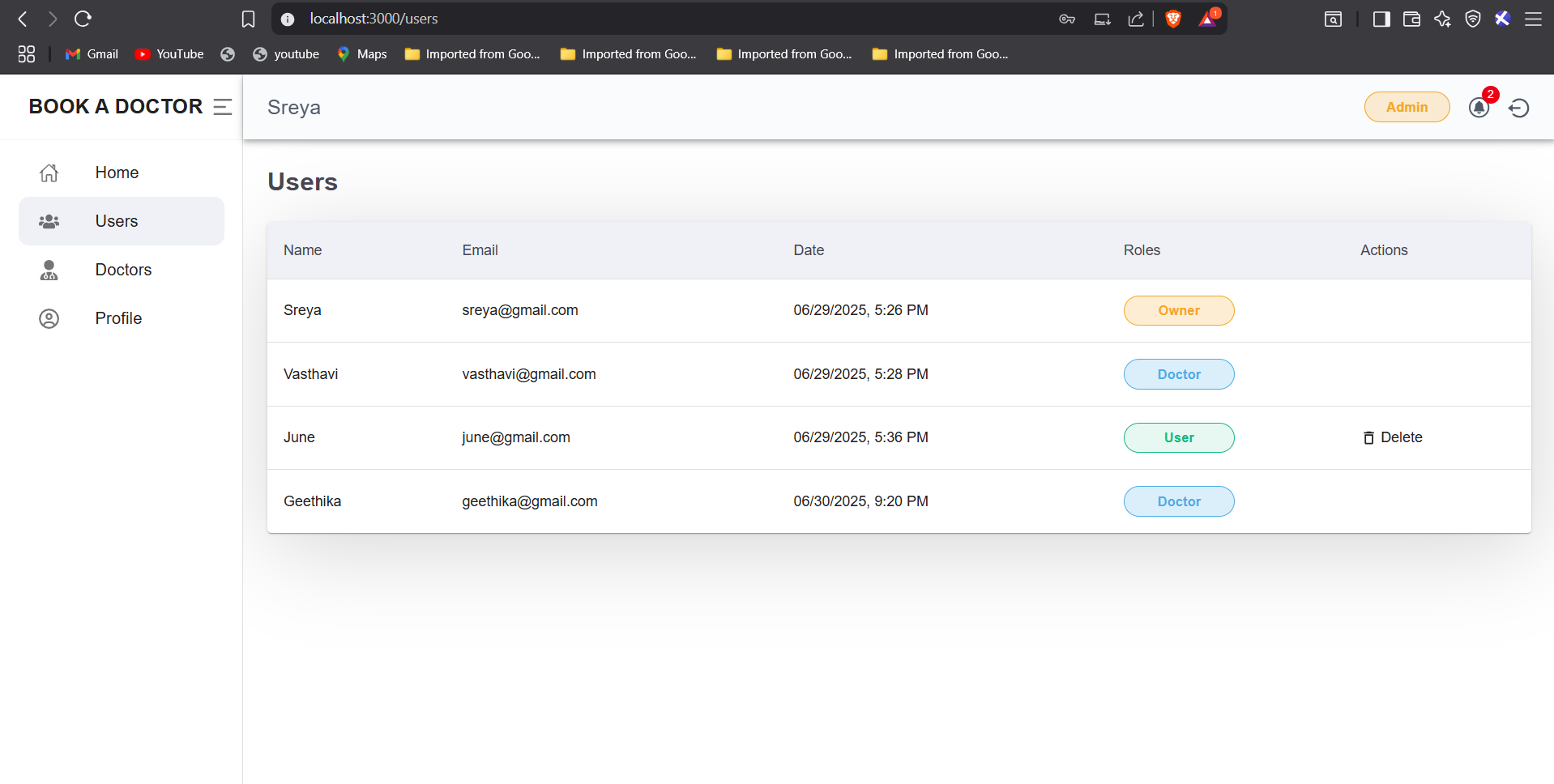


**Admin Functionality:**

* Approve or reject doctor applications

****

* View and manage all users and doctors

****

* Monitor system appointments- Delete users or doctors when necessary

**Security & Validation:**

* All routes are protected based on user roles
* JWT tokens ensure secure access
* Form validation prevents invalid data submission

**API Overview:**

The backend exposes various RESTful APIs, grouped by module:

**Authentication APIs:**

* POST /users/signup - User registration
* POST /users/login - User login

**Doctor APIs:**

* POST /doctors/signup - Doctor application
* GET /doctors/approved-doctors - List of approved doctors
* PUT /doctors/:id - Update doctor profile
* POST /doctors/check-booking-availability - Check schedule

**Appointment APIs:**

* POST /users/book-appointment - Book a new appointment
* GET /users/user-appointments/:id - Get user's appointments
* GET /doctors/appointments/:id - Get doctor's appointments
* POST /doctors/change-appointment-status - Modify appointment status

**Notification APIs:**

* POST /users/mark-all-notification-as-seen
* POST /users/delete-all-notifications

**Admin APIs:**

* GET /users - View all users
* DELETE /users/:id - Delete user

**Application Flow and User Interface:**

DocSpot has a responsive and intuitive user interface that adjusts to desktop and mobile devices. The flow of the application can be summarized as follows:

- A new user can register as a patient or a doctor.

- Doctors must be approved by the admin before their profiles are visible.

- Patients can view a list of approved doctors and book appointments.

- Doctors manage appointment requests and update their availability.

- Admins have full control over user and doctor data.

Each dashboard is customized for the user's role, with access restricted to relevant data and features.

**Testing and Quality Assurance:**

Testing is performed both manually and through automated tools:

**Frontend:**

Jest and React Testing Library for unit and integration testing

**Backend:**

Postman used to test API endpoints

**Code Quality:**

ESLint and Prettier are used for maintaining clean code

**Challenges and Resolutions:**

During development, several challenges were encountered:

* Role-Based Access Control: Ensuring only the correct users could access specific routes was implementedusing middleware functions.
* Doctor Approval Workflow: A mechanism had to be developed for admin approval, which required new APIsand frontend logic.
* Real-Time Notifications: Though currently implemented via manual refresh, future updates aim to integrateWebSockets for real-time updates.

**Future Enhancements:**

DocSpot is a feature-rich application with potential for further development. Planned features include:

* **Video Consultations**: Using WebRTC for real-time video calls between patients and doctors.
* **Payment Integration:** Allowing users to pay for consultations through Stripe or Razorpay.
* **Email & SMS Notifications:** Automated reminders for appointments.
* **Real-Time Notifications:** Implementing Socket.IO for live updates.
* **Password Reset Feature:** Enabling users to recover their accounts securely.

**Conclusion:**

DocSpot successfully demonstrates the implementation of a real-world full-stack web application that addresses a common problem in healthcare accessibility. It combines modern technologies and frameworks to deliver a robust, scalable, and user-friendly platform. With continued development and deployment, DocSpot has the potential to be used in real-life clinical or telemedicine environments.

This project reflects practical experience in designing, building, and deploying a MERN stack application, with full lifecycle development including requirement analysis, architecture, implementation, testing, and planning for future growth.