**FULL STACK DEVELOPMENT WITH MERN**

**Project Documentation**

**1. Introduction**

|  |  |
| --- | --- |
| **Project Title** | DocSpot: Seamless Appointment Booking for Health |
| **Team ID** | LTVIP2026TMIDS35442 |
| **Team Size** | 4 Members |

**Team Members and Roles**

| **Name** | **Role** | **Responsibilities** |
| --- | --- | --- |
| **Prathi Adithya Durga Sri Krishna Kumar** | Full Stack Developer | Project coordination, Server architecture, State management, Database design, Documentation |
| **Botla Siva Satyanarayana** | Full Stack Developer | UI/UX design, React components, Responsive Design, API development |
| **Satya Srithadi** | Backend Developer | API development, API integration, Authentication, System Testing |
| **Perabathula Satya Sai Manikanta** | Frontend Developer | Frontend-backend integration, Component development, Bug fixing, Deployment support |

**2. Project Overview**

**Purpose**

DocSpot is a web-based doctor appointment booking platform that bridges the gap between patients and healthcare providers. The purpose is to eliminate the frustration of traditional phone-based appointment booking by providing a seamless online solution where patients can find doctors, check availability, book appointments, and manage their healthcare journey—all from the comfort of their homes.

**Goals**

* Provide patients with 24/7 access to doctor appointment booking
* Enable doctors to manage their schedules digitally and reduce administrative burden
* Give admins tools to verify doctors and monitor platform activity
* Create a trustworthy healthcare ecosystem with verified doctor profiles

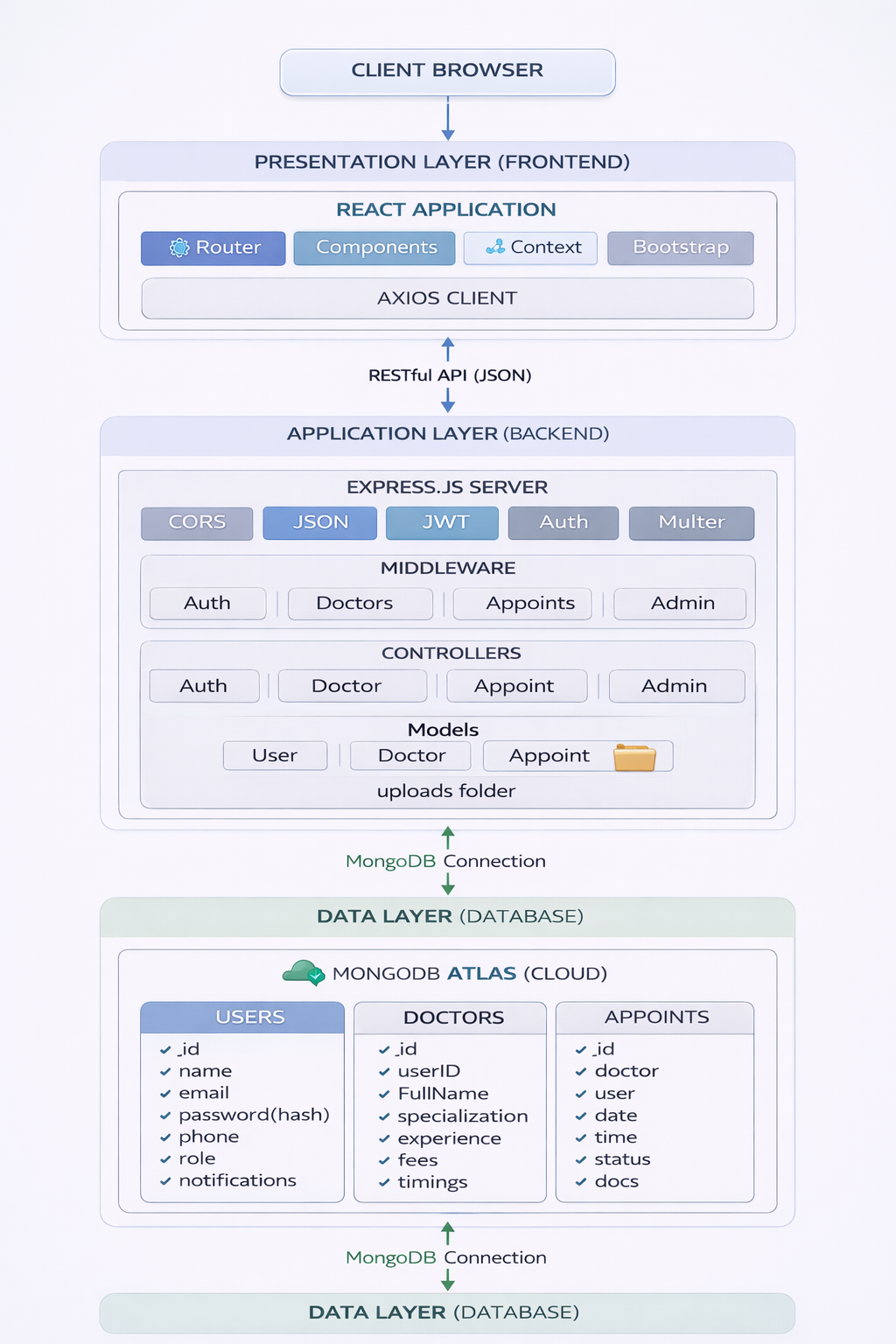
**Key Features**

| **Feature** | **Description** |
| --- | --- |
| **User Authentication** | Secure registration/login with JWT and password encryption |
| **Role-Based Access** | Separate dashboards for Patients, Doctors, and Admins |
| **Doctor Discovery** | Browse and filter doctors by specialization |
| **Doctor Profiles** | View qualifications, experience, consultation fees |
| **Appointment Booking** | Select date/time, add reason, upload medical documents |
| **Appointment Management** | Track status (pending/approved/rejected/completed/cancelled) |
| **Cancel/Reschedule** | Patients can cancel appointments from dashboard |
| **Doctor Dashboard** | View and manage appointment requests with approve/reject options |
| **Doctor Profile Management** | Update qualifications, fees, and day-wise timings |
| **Admin Dashboard** | Platform statistics and monitoring |
| **Doctor Verification** | Admin approval workflow for new doctor registrations |
| **User Management** | Admin can view all patients and doctors |
| **Appointment Oversight** | Admin can view and filter all appointments |

**3. Architecture**

**System Architecture Overview**

DocSpot follows a **three-tier client-server architecture** with clear separation between presentation, application, and data layers.

****

**Frontend Architecture (React)**

The frontend is built using React.js with a component-based architecture:

| **Layer** | **Technology** | **Purpose** |
| --- | --- | --- |
| **UI Library** | React.js 18.2.0 | Component-based user interface |
| **Routing** | React Router 6 | Client-side navigation and protected routes |
| **State Management** | Context API | Global state for authentication and notifications |
| **HTTP Client** | Axios 1.6 | API communication with token interception |
| **Styling** | Bootstrap 5.3, React-Bootstrap 2.10 | Responsive UI components |
| **Icons** | React Icons | Icon library for UI elements |

**Key Frontend Components:**

* **Authentication Components**: Login, Register forms with validation
* **Patient Components**: DoctorsList, DoctorProfile, BookAppointment, MyAppointments
* **Doctor Components**: DoctorDashboard, DoctorAppointments, DoctorProfileEdit
* **Admin Components**: AdminDashboard, ManageDoctors, ManageUsers, ManageAppointments
* **Layout Components**: Navbar, Footer, ProtectedRoute

**Backend Architecture (Node.js/Express)**

The backend follows MVC (Model-View-Controller) pattern:

| Layer | Technology | Purpose |
| --- | --- | --- |
| **Runtime** | Node.js 22.18 | JavaScript runtime environment |
| **Framework** | Express.js 4.18 | Web application framework |
| **Authentication** | JWT, Bcryptjs | Token-based auth and password hashing |
| **File Upload** | Multer 1.4 | Handle multipart form data for document uploads |
| **Environment** | Dotenv 16.3 | Environment variable management |
| **CORS** | CORS 2.8 | Cross-origin resource sharing |

**API Structure:**

* **Routes**: Define endpoints and HTTP methods
* **Controllers**: Business logic for each entity
* **Middleware**: Authentication, file upload, error handling
* **Models**: Mongoose schemas for data validation

**Database Architecture (MongoDB)**

| Aspect | Specification |
| --- | --- |
| **Database** | MongoDB Atlas (Cloud) |
| **ODM** | Mongoose 7.5 |
| **Collections** | users, doctors, appointments |

**Database Schema**

**Users Collection**

{

\_id: ObjectId,

name: String,

email: String (unique),

password: String (hashed),

phone: String,

isDoctor: Boolean,

type: String (enum: user/doctor/admin),

notification: Array,

createdAt: Date

}

**Doctors Collection**

{

\_id: ObjectId,

userId: ObjectId (ref: Users),

fullName: String,

email: String,

phone: String,

specialization: String,

experience: Number,

fees: Number,

address: Object,

qualifications: Array,

timings: Object,

status: String (enum: pending/approved/rejected),

rating: Number,

totalReviews: Number,

createdAt: Date

}

**Appointments Collection**

{

\_id: ObjectId,

doctorId: ObjectId (ref: Doctors),

userId: ObjectId (ref: Users),

doctorInfo: Object,

userInfo: Object,

date: Date,

timeSlot: Object,

documents: Array,

status: String (enum: pending/approved/rejected/completed/cancelled),

reason: String,

createdAt: Date

}

**4. Setup Instructions**

**Prerequisites**

| Software | Version | Purpose |
| --- | --- | --- |
| **Node.js** | v18.x or higher | JavaScript runtime |
| **npm** | v9.x or higher | Package manager |
| **MongoDB Atlas** | - | Cloud database (or local MongoDB) |
| **Git** | - | Version control |
| **Web Browser** | Chrome/Firefox/Edge | Running the application |

**Installation:**

**Step 1: Clone the Repository**

git clone <repository-url>

cd DocSpot-App

**Step 2: Backend Setup**

cd backend

npm install

Create a .env file in the backend folder:

env

PORT=5000

MONGODB\_URI=mongodb+srv://username:password@cluster.mongodb.net/docspot

JWT\_SECRET=your\_jwt\_secret\_key\_here

NODE\_ENV=development

**Step 3: Frontend Setup**

cd ../frontend

npm install

Create a .env file in the frontend folder:

Env

REACT\_APP\_API\_URL=http://localhost:5000/api

**Step 4: Database Setup**

1. Create a MongoDB Atlas account at <https://www.mongodb.com/atlas>
2. Create a new cluster (free tier)
3. Create a database named docspot
4. Create collections: users, doctors, appointments
5. Get your connection string and update the .env file

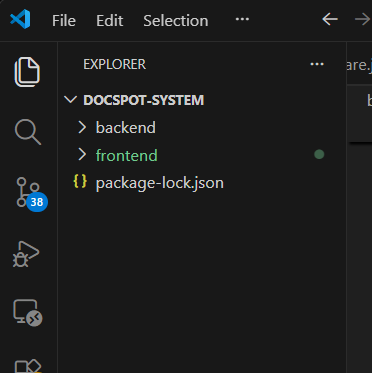
**Step 5: Seed Sample Data (Optional)**

cd backend

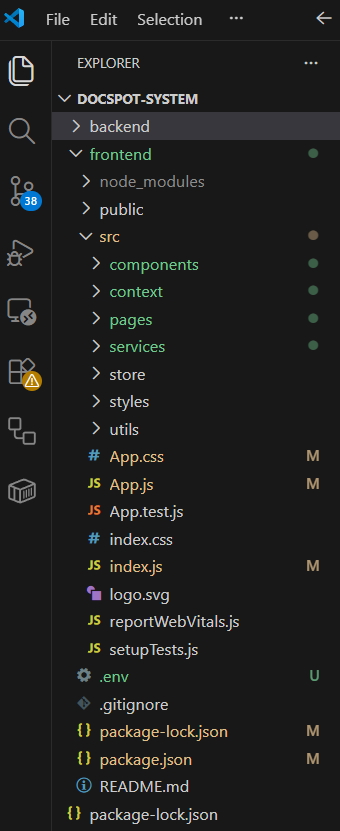
npm run seed

**5. Folder Structure**

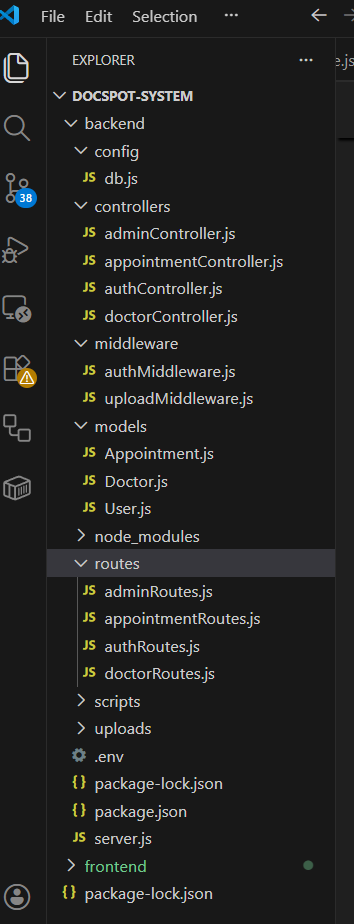
**Project Root:**

****

**Frontend Structure:**

****

**Backend Structure:**

****

**6. Running the Application**

**Start MongoDB (if using local)**

# Start MongoDB service

net start MongoDB

**Start Backend Server**

cd backend

npm run dev

Backend runs at: [http://localhost:5000](http://localhost:5000/)

**Start Frontend Server**

cd frontend

npm start

Frontend runs at: [http://localhost:3000](http://localhost:3000/)

**Access the Application**

* Open browser and navigate to: [http://localhost:3000](http://localhost:3000/)
* Default credentials (after seeding):
  + Admin: admin@docspot.com / Admin@123
  + Doctor: sarah.johnson@doctor.com / password123
  + Patient: john.smith@example.com / password123

**7. API Documentation**

**Base URL:** <http://localhost:5000/api>

**Authentication Endpoints**

| **Method** | **Endpoint** | **Description** | **Auth Required** |
| --- | --- | --- | --- |
| POST | /auth/register | Register new user | No |
| POST | /auth/login | Login user | No |
| GET | /auth/profile | Get user profile | Yes |
| PUT | /auth/profile | Update profile | Yes |
| GET | /auth/notifications | Get notifications | Yes |

**Doctor Endpoints**

| **Method** | **Endpoint** | **Description** | **Auth Required** |
| --- | --- | --- | --- |
| GET | /doctors | Get all approved doctors | No |
| GET | /doctors/:id | Get doctor by ID | No |
| POST | /doctors/apply | Apply as doctor | Yes |
| PUT | /doctors/profile | Update doctor profile | Yes (Doctor) |
| GET | /doctors/appointments/list | Get doctor's appointments | Yes (Doctor) |
| PUT | /doctors/appointments/:id | Update appointment status | Yes (Doctor) |

**Appointment Endpoints**

| Method | Endpoint | Description | Auth Required |
| --- | --- | --- | --- |
| POST | /appointments | Book appointment | Yes |
| GET | /appointments/my-appointments | Get user appointments | Yes |
| PUT | /appointments/:id/cancel | Cancel appointment | Yes |
| PUT | /appointments/:id/reschedule | Reschedule appointment | Yes |

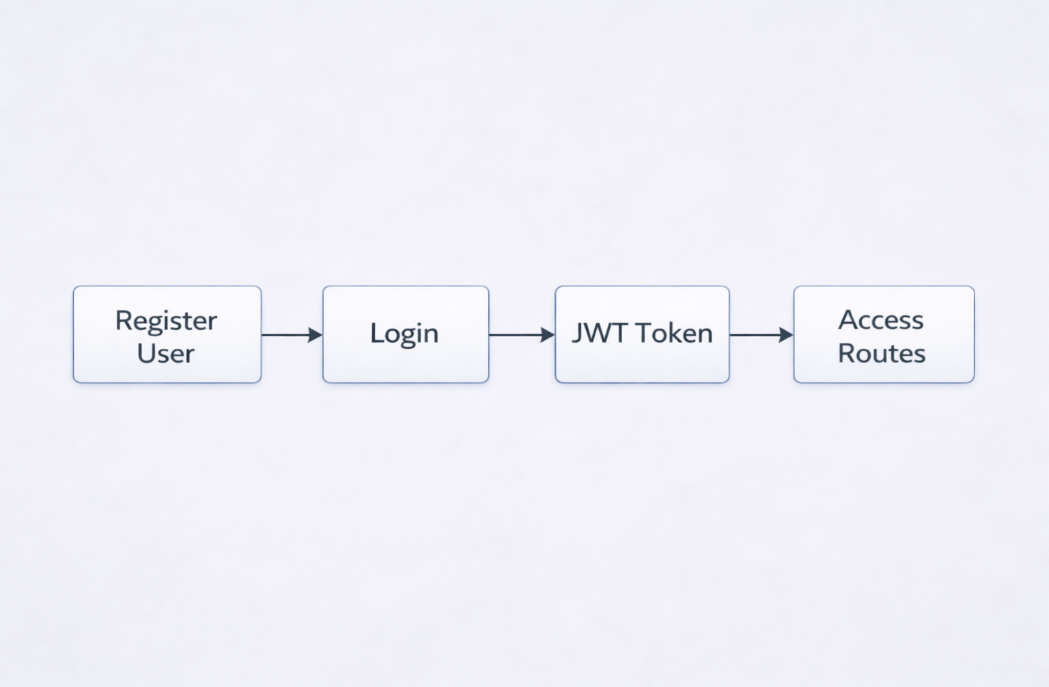
**Admin Endpoints**

| Method | Endpoint | Description | Auth Required |
| --- | --- | --- | --- |
| GET | /admin/users | Get all users | Yes (Admin) |
| GET | /admin/doctors | Get all doctors | Yes (Admin) |
| PUT | /admin/doctors/:id/status | Update doctor status | Yes (Admin) |
| GET | /admin/appointments | Get all appointments | Yes (Admin) |
| GET | /admin/stats | Get dashboard statistics | Yes (Admin) |

**8. Authentication**

**Authentication Flow**

DocSpot uses **JWT (JSON Web Token)** based authentication with role-based access control.



**Implementation Details**

| **Component** | **Technology** | **Purpose** |
| --- | --- | --- |
| **Password Hashing** | bcryptjs | Secure password storage with salt rounds |
| **Token Generation** | jsonwebtoken | Create JWT tokens on successful login |
| **Token Storage** | localStorage | Store token client-side |
| **Token Validation** | JWT Middleware | Verify token on protected routes |
| **Role-Based Access** | Custom Middleware | Restrict access based on user type |

**Authentication Middleware**

// authMiddleware.js

const protect = async (req, res, next) => {

const token = req.headers.authorization?.split(' ')[1];

if (!token) {

return res.status(401).json({ message: 'Not authorized' });

}

try {

const decoded = jwt.verify(token, process.env.JWT\_SECRET);

req.user = await User.findById(decoded.id).select('-password');

next();

} catch (error) {

res.status(401).json({ message: 'Invalid token' });

}

};

// Role-based middleware

const admin = (req, res, next) => {

if (req.user && req.user.type === 'admin') {

next();

} else {

res.status(403).json({ message: 'Admin access required' });

}

};

**Protected Routes in Frontend**

// ProtectedRoute component

const ProtectedRoute = ({ children, requiredRole }) => {

const { user } = useAuth();

if (!user) return <Navigate to="/login" />;

if (requiredRole === 'admin' && user.type !== 'admin') {

return <Navigate to="/dashboard" />;

}

return children;

};

**9. User Interface**

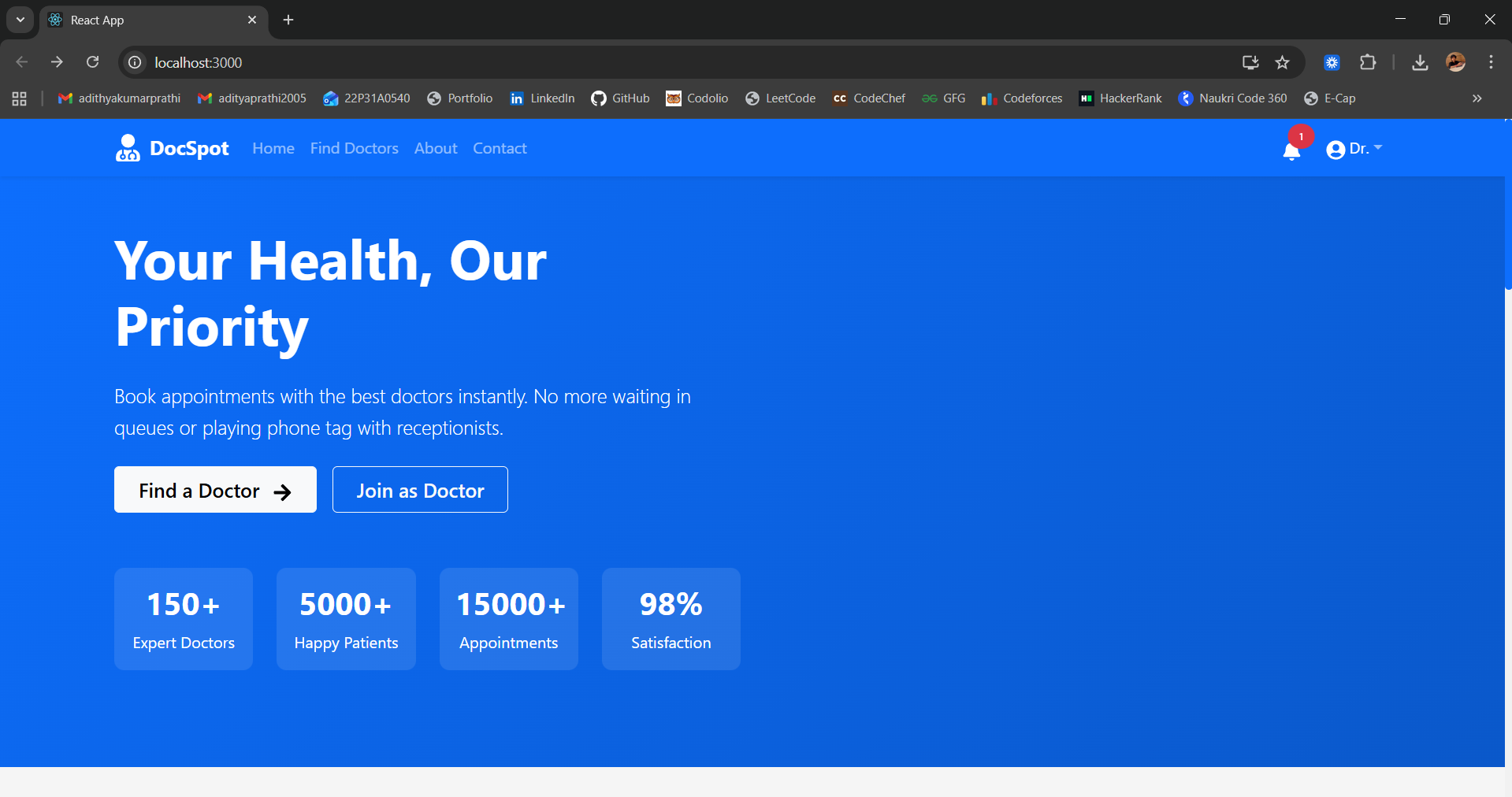
**Key UI Components**

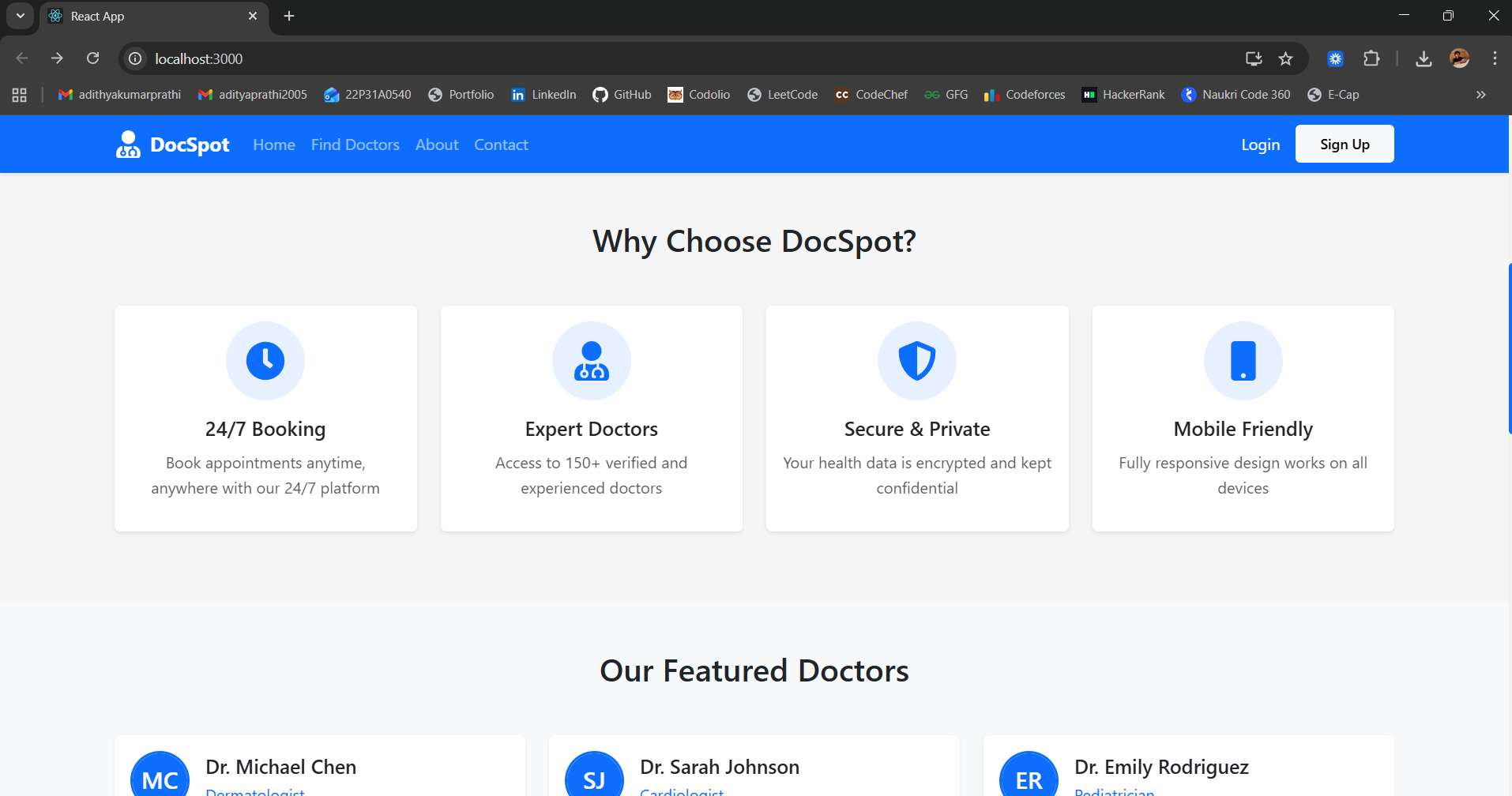
| **Component** | **Description** |
| --- | --- |
| **Navbar** | Responsive navigation with role-based menu items |
| **Hero Section** | Landing page with statistics and call-to-action |
| **Doctor Cards** | Doctor information with hover animations |
| **KPI Cards** | Statistics cards with gradient backgrounds |
| **Status Badges** | Color-coded appointment status indicators |
| **Modals** | Confirmation dialogs for cancel/reschedule |
| **Forms** | Validated forms with clear error messages |
| **Tables** | Responsive tables with hover effects |

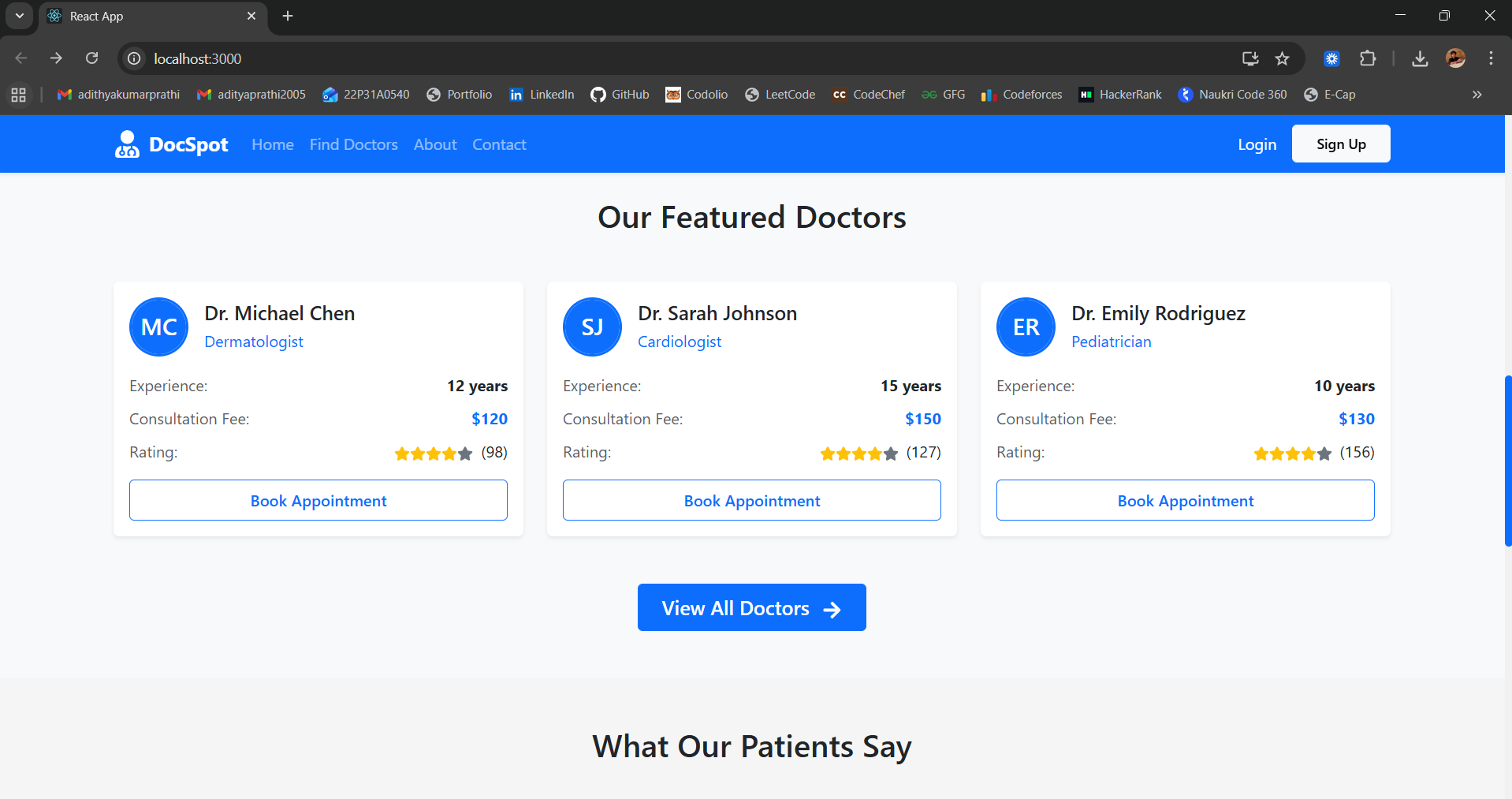
**Responsive Design**

* Mobile-first approach using Bootstrap grid
* Collapsible navigation on smaller screens
* Cards stack vertically on mobile devices
* Touch-friendly buttons and form elements

**Screenshots:**







**10. Testing**

**Testing Strategy**

| Test Type | Description | Tools |
| --- | --- | --- |
| **Unit Testing** | Test individual components and functions | Jest (planned) |
| **Integration Testing** | Test API endpoints and database interactions | Postman |
| **UI Testing** | Test user interface and interactions | Manual testing |
| **Cross-browser Testing** | Test on multiple browsers | Chrome, Firefox, Edge |
| **Responsive Testing** | Test on multiple screen sizes | Chrome DevTools |

**Test Coverage**

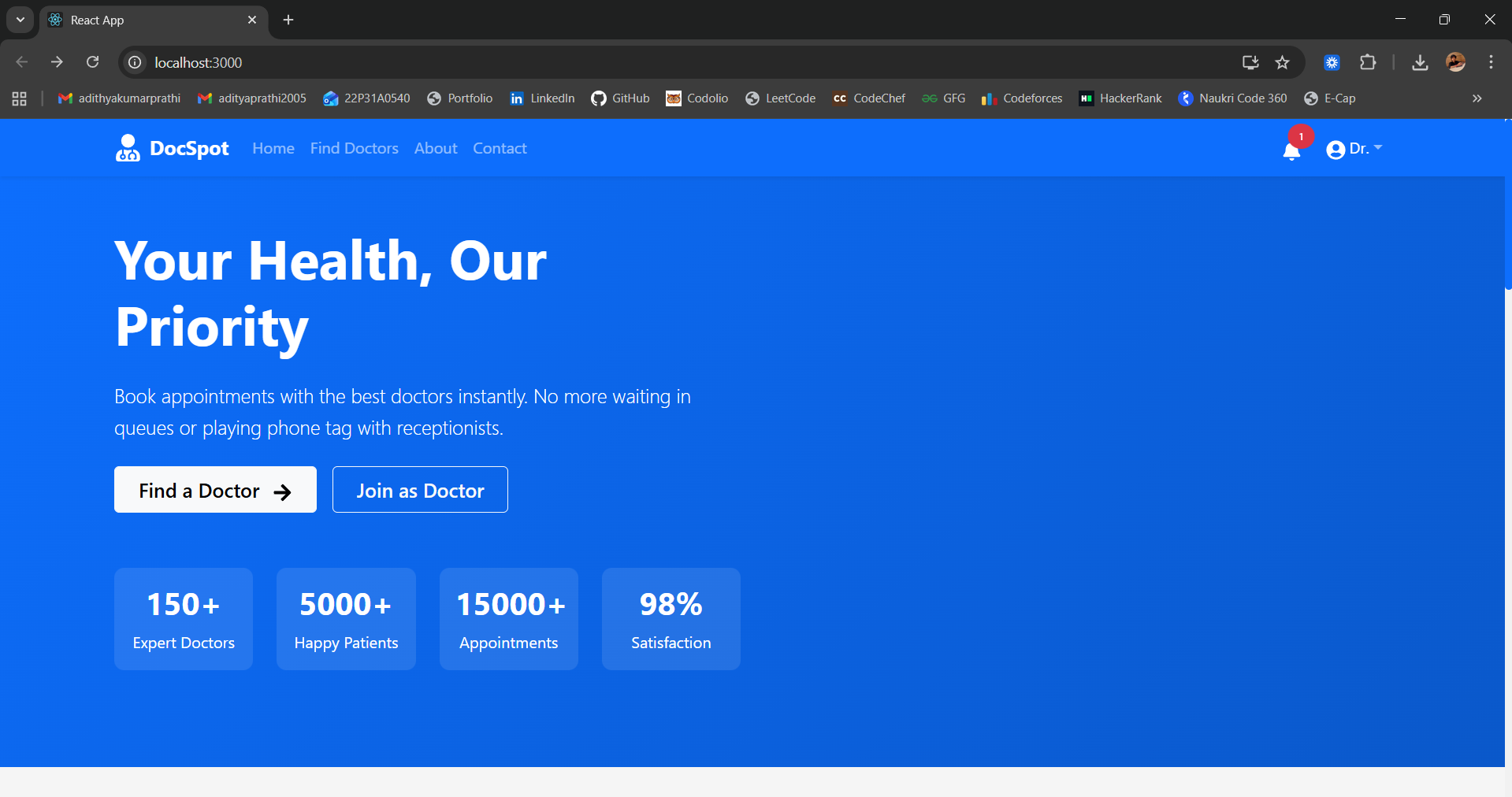
| **Module** | **Test Cases** | **Pass Rate** |
| --- | --- | --- |
| Authentication | 6 | 100% |
| Patient Features | 9 | 100% |
| Doctor Features | 6 | 100% |
| Admin Features | 5 | 100% |
| UI/Responsive | 2 | 100% |
| **Total** | **28** | **100%** |

**Bug Tracking**

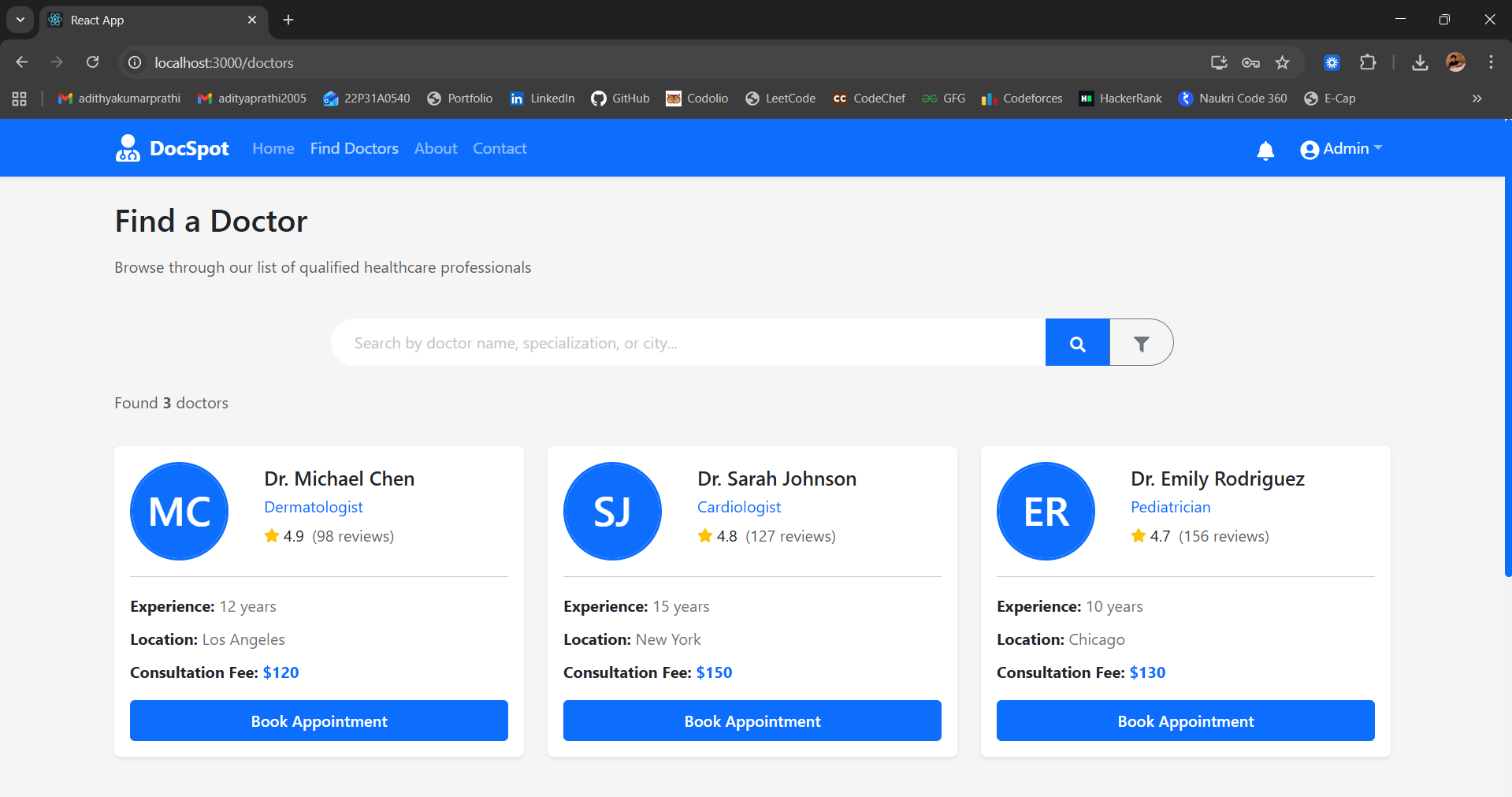
* Total bugs identified during UAT: 7
* Bugs fixed: 7
* Bug fix rate: 100%

**11. Screenshots or Demo**

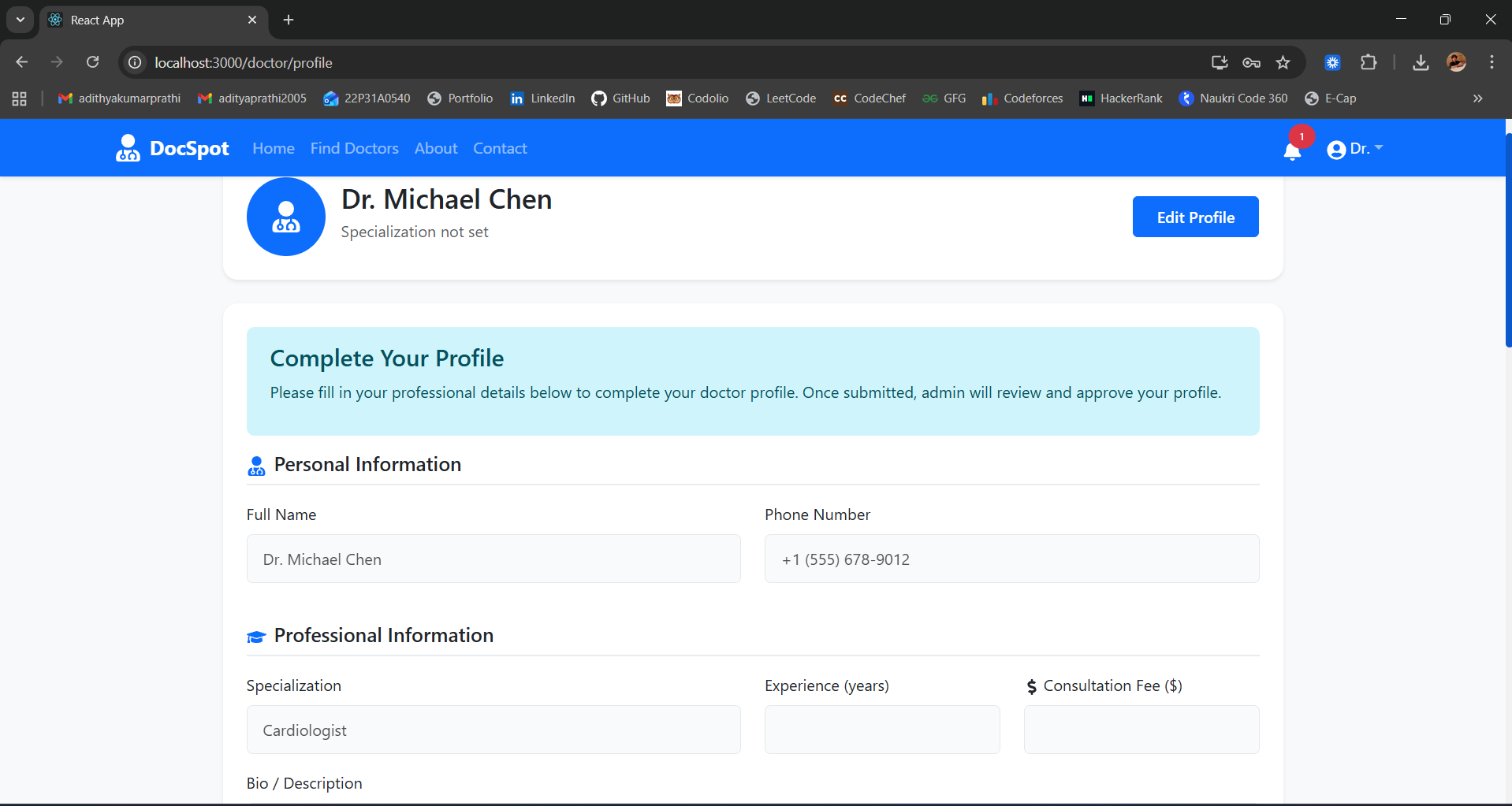
**Landing Page**



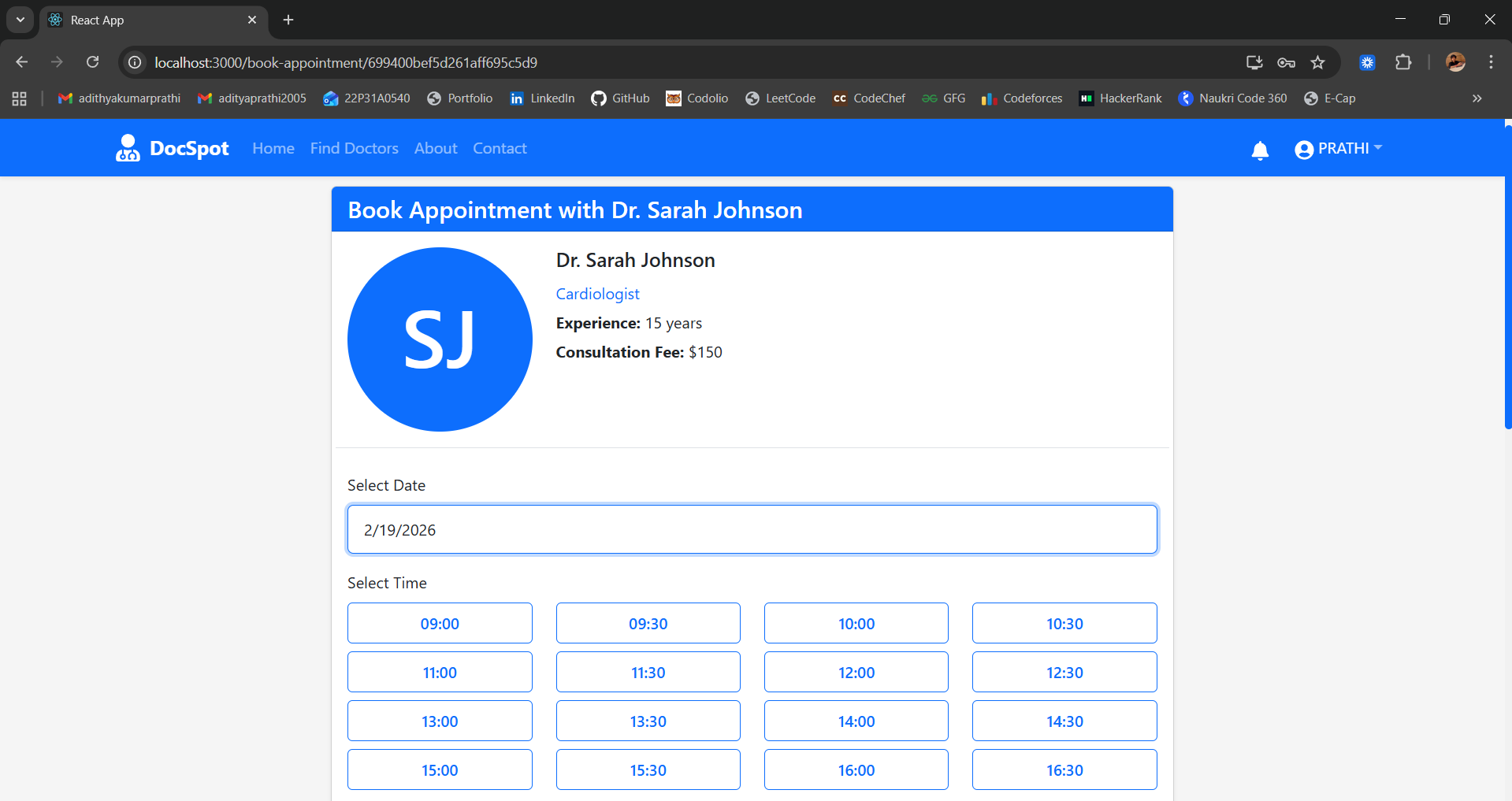
**Doctors List Page**

****

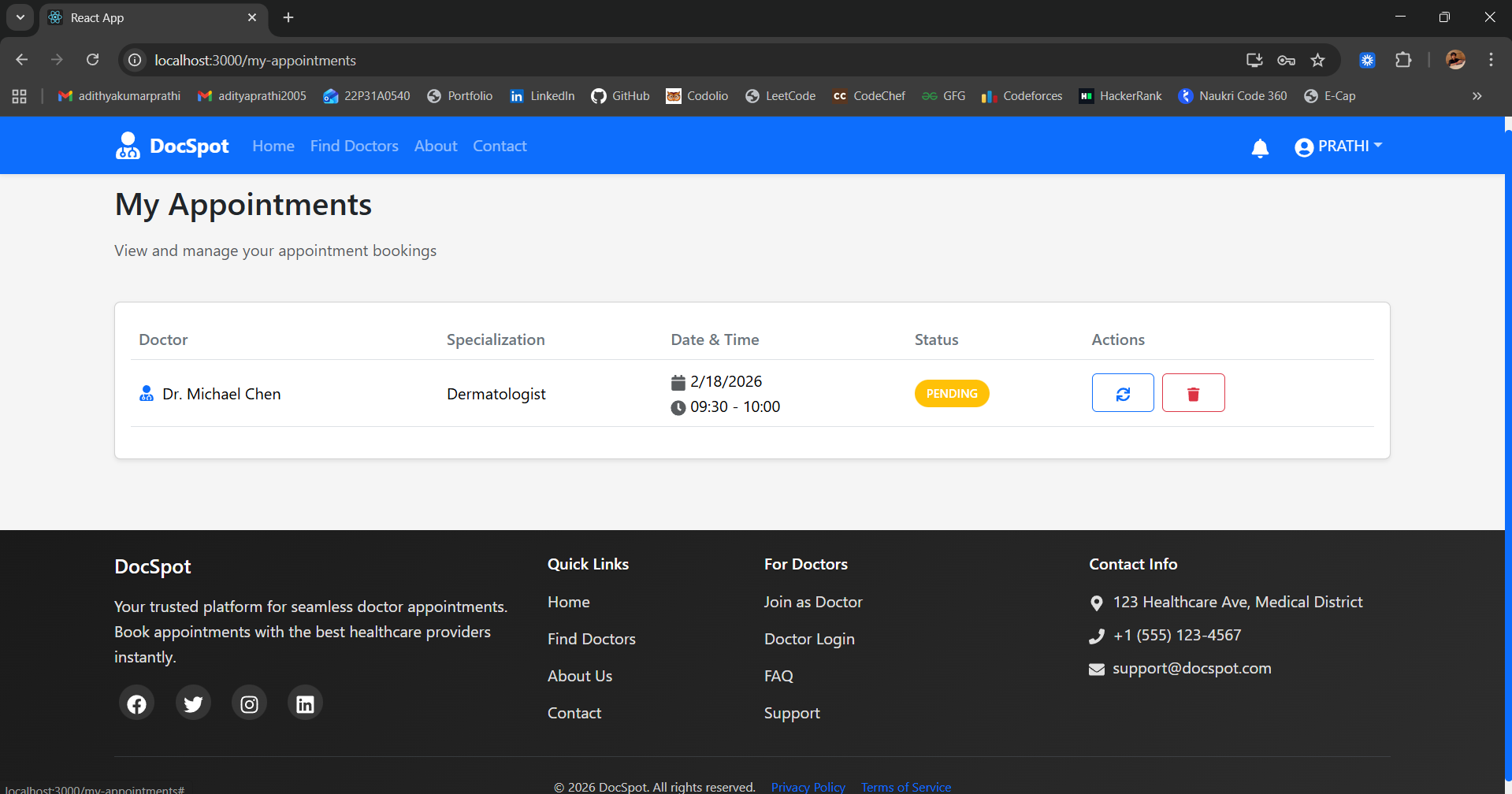
**Doctor Profile**



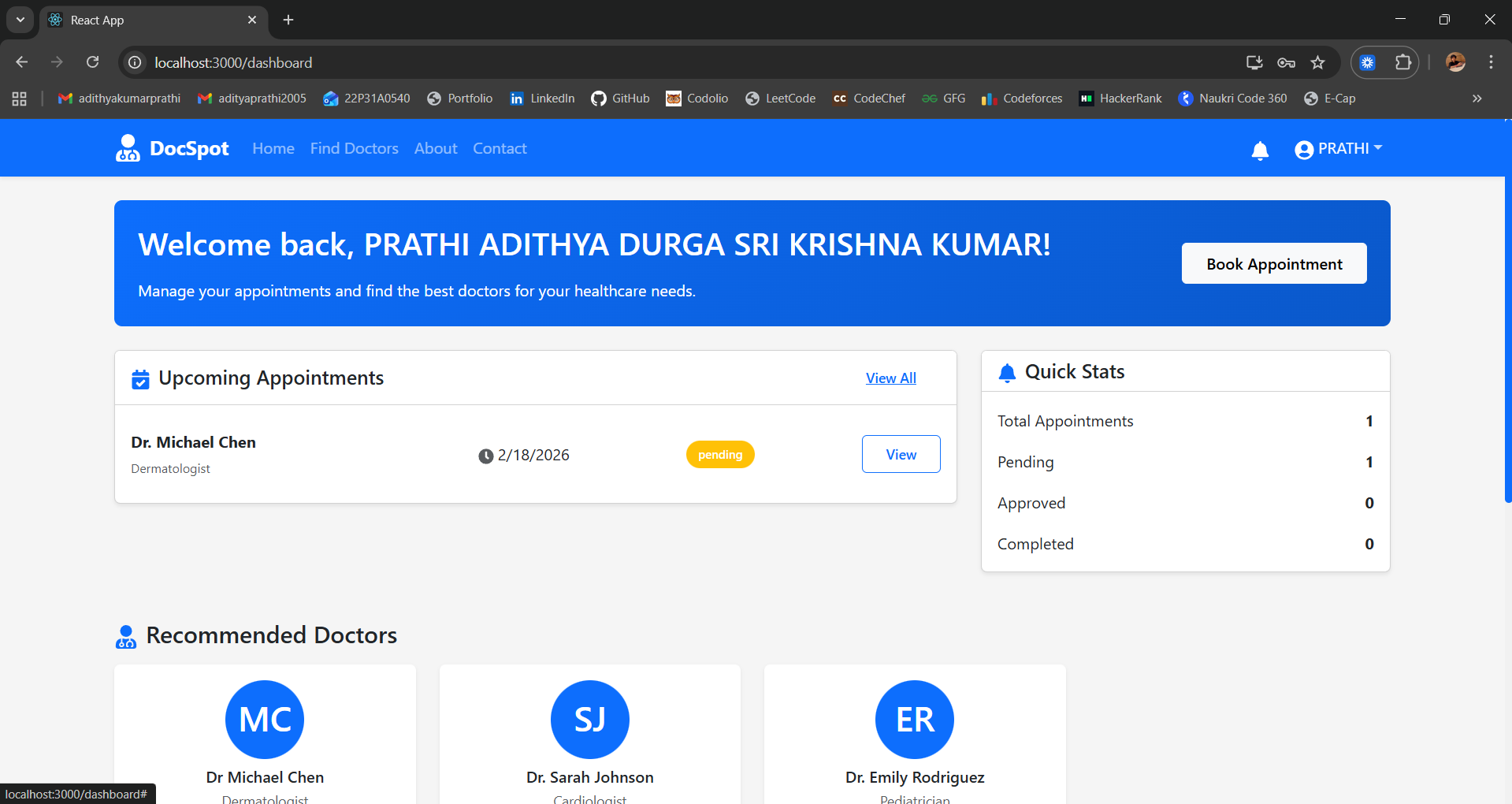
**Appointment Booking**

**

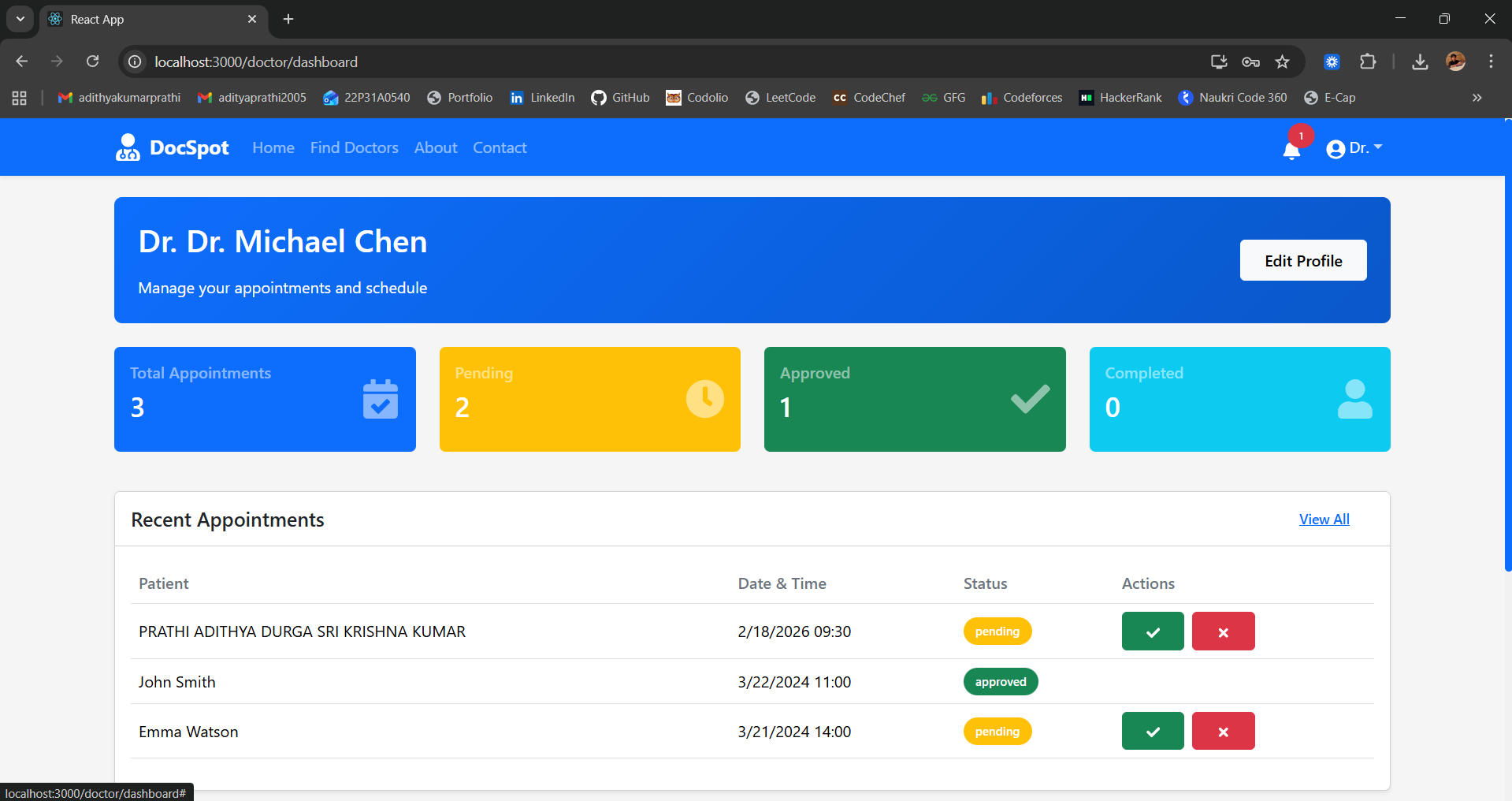
**My Appointments**



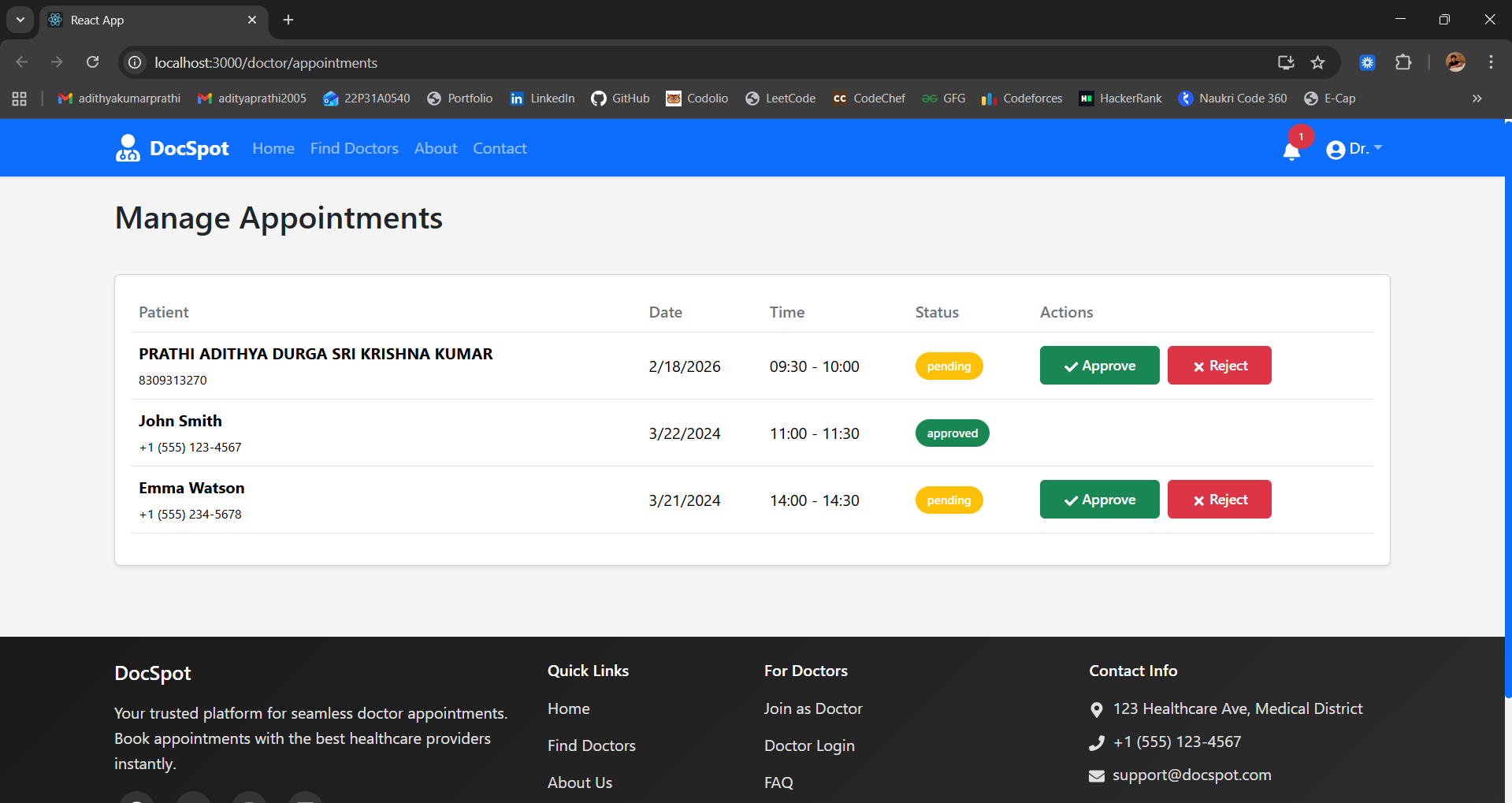
**Patient Dashboard**

**

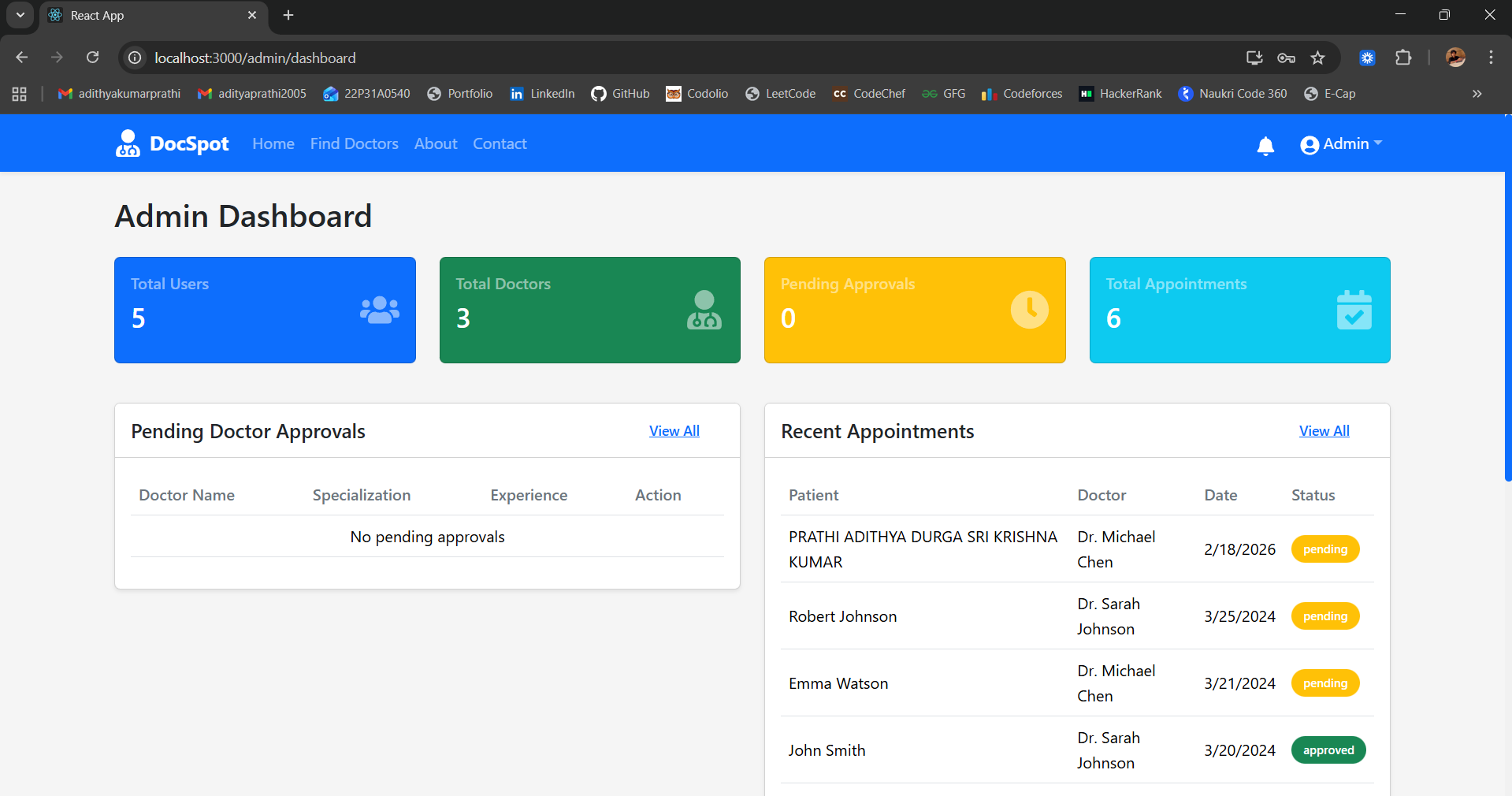
**Doctor Dashboard**

**

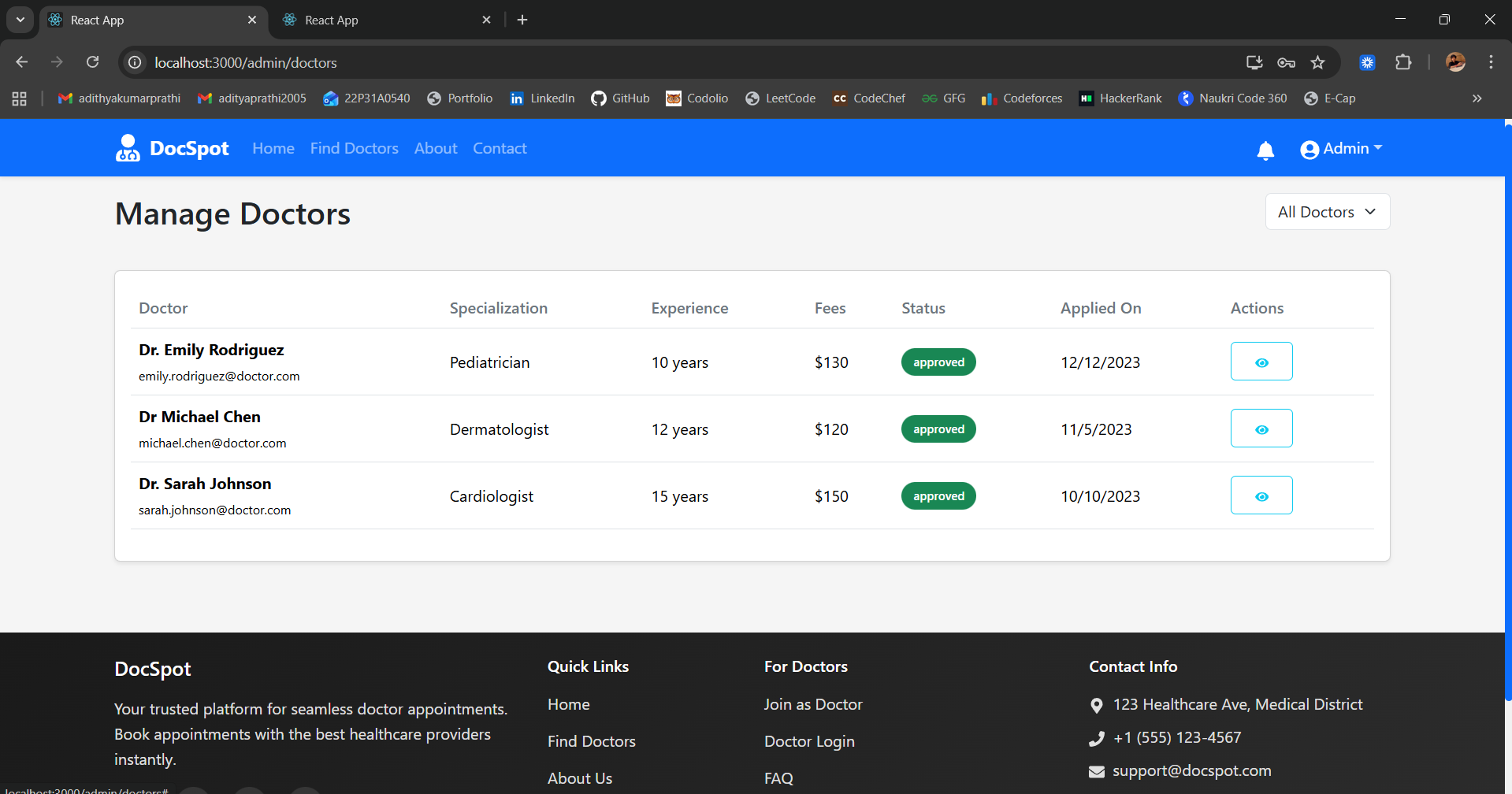
**Doctor Appointments Management**

**

**Admin Dashboard**

**

**Admin Doctor Approval**



**Video Demo:** <https://drive.google.com/file/d/18imnZNXOX-wAFloZGO-t3SNZsxP70Rbl/view?usp=drive_link>

**12. Known Issues**

| Issue ID | Description | Status | Workaround |
| --- | --- | --- | --- |
| **N/A** | No known issues in current version | - | - |

**13. Future Enhancements**

| **Enhancement** | **Description** | **Priority** |
| --- | --- | --- |
| **Email Notifications** | Send email confirmations and reminders | High |
| **SMS Reminders** | Send SMS reminders for appointments | Medium |
| **Payment Integration** | Online payment for consultation fees | Medium |
| **Video Consultation** | Telemedicine integration | Medium |
| **Ratings & Reviews** | Patients can rate doctors after appointment | Medium |
| **Calendar Sync** | Sync appointments with Google/Outlook calendar | Low |
| **Mobile App** | Native mobile application | Low |
| **Multi-language Support** | Support for multiple languages | Low |