**Clinic Management System Project Report**

**1. Project Description**

The **Web Clinic Management System** is a modern healthcare management application designed using **Jakarta REST Web Services**. This system leverages RESTful architecture to streamline clinic operations, offering efficient management of clinic information through well-defined HTTP endpoints. It supports multiple response formats, including HTML and JSON, ensuring compatibility with diverse client applications.

The system includes core web resource classes such as **WebDoctorResource**, **WebPatientResource**, and **WebClinicResource**, which handle key functionalities like managing doctors, patients, and clinic-wide operations. Complementing these are client-side applications like **ClientDoctorManagement**, **ClientPatientManagement**, and **ClientClinicManagement**, which interact with the RESTful services to provide a seamless and dynamic user experience.

By integrating these components, the **Web Clinic Management System** delivers a robust platform for efficient clinic operations, highlighting the power and flexibility of RESTful web services in modern healthcare solutions.

**Core Functionalities**

* **Login & Role-Based Dashboards**:
  + Implemented user roles: Admin, Doctor, Patient.
  + Implemented user roles for specific Doctor and Patient.
  + Each role redirects to its dedicated dashboard after login.
* **Doctor Management**:
  + View and manage doctor records through endpoints and HTML interfaces.
  + Add, search, and delete doctor records.
* **Patient Management**:
  + Add, search, update, and delete patient records.
  + Schedule and reschedule appointments.
  + Filter patients by assigned doctor and appointment dates.
* **Prescription Management:**
  + Record and manage prescriptions linked to specific patients.
  + View prescriptions through RESTful endpoints.
  + Generate detailed prescription reports for analysis.
* **Statistics and Reporting**:
  + Analyze clinic data, such as:
    - Number of doctors and average consultation fees by specialization.
    - Patient age-group analysis (children, adults, seniors).
    - Total clinic revenue.
  + Add prescription to specific patients
* **Dynamic Interfaces**:

1. **HTML Forms**: Interactive forms for adding, updating, and managing doctor and patient records including prescription details.
2. **Role-Based Dashboards**: Admin, doctor, patient, individual doctors and patients are directed to their respective dashboards after login, ensuring tailored access to functionalities.
3. **Improved Validation**: Enhanced form validation using JavaScript ensures accurate data entry and user-friendly error messages.
4. **Styled UI Components**: Modern and responsive design elements improve user experience.

* **Data Management**:

1. **Real-Time File Updates**: Changes to doctor and patient records are immediately reflected in Doctors.in and Patients.in files.
2. **Efficient Retrieval**: Use of HashMap structures for quick data retrieval and manipulation.

**Functionality**: Endpoints to support role-based data access and management, such as:

* + Admin-exclusive endpoints for creating and deleting records.
  + Doctor-specific endpoints for viewing assigned patients and appointments.
  + Patient-specific endpoints for booking and managing appointments.
  + Individual Doctor and Patient endpoints for booking and managing data

1. **Unified Management**: Streamlined endpoints for managing appointments and patient records across user roles.

* **Security Enhancements**:

1. **Role-Based Login**: Pre-defined credentials ensure secure and differentiated access for admin, doctor, and patient roles.
2. **Session Management**: Logout functionality invalidates user sessions to enhance security and prevent unauthorized access.

* **Appointment Scheduling**:

1. **Role-Specific Access**:

* **Patients**: Can book or reschedule their appointments dynamically.
* **Doctors**: View upcoming appointments and manage availability.
* **Admin**: Oversee all appointment schedules and manage conflicts.

1. **Dynamic Forms**: Forms automatically populate available doctors and slots, ensuring accurate and convenient scheduling.

* **Advanced Search**:
  1. **Patients**:
  + Search by appointment date and doctor ID for their own records.
  1. **Doctors**:
  + Search for patients by specialization and appointment date.
  1. **Admin**:
  + Perform global searches across all doctor and patient data.
  1. **Individual Doctor**:
  + View Upcoming appointments and add prescription
  1. **Individual Patient:**
* View user info and assigned doctor
* **Revenue Tracking**:

Calculate and display revenue details specific to:

* Individual doctors based on their consultation fees and patient appointments.
* Total clinic revenue for the admin dashboard.

**2. Business Impact**

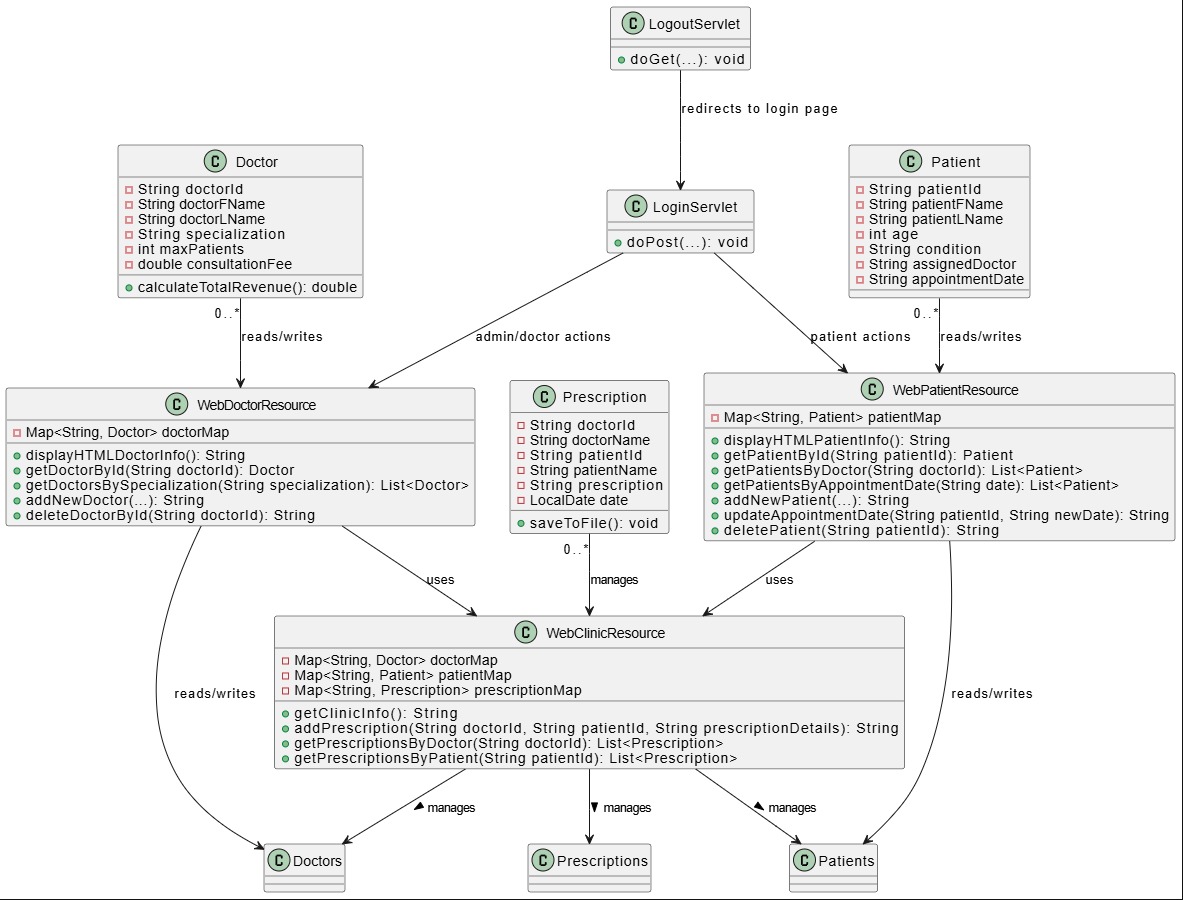
* **Operational Benefits**:
  + Instant access to doctor and patient information
  + Real-time revenue tracking
  + Automated statistical analysis
  + Monitoring resources by specialization
  + Streamlined appointment management.
  + Centralized doctor and patient data storage.
  + Automated statistical insights for informed decision-making.
* **User Experience**:
  + Role-based dashboards for simplified navigation.
  + Accessible UI for patients to manage their appointments.
* **Information Management**
  + Centralized data storage
  + Automated report generation
  + Multiple data viewing formats (HTML/JSON)
  + Age-group based patient tracking
* **Financial Benefits**

1. **Revenue Monitoring**
   * Automated revenue calculations per doctor
   * Overall clinic revenue tracking
   * Specialization-based fee information
   * Financial data visualization

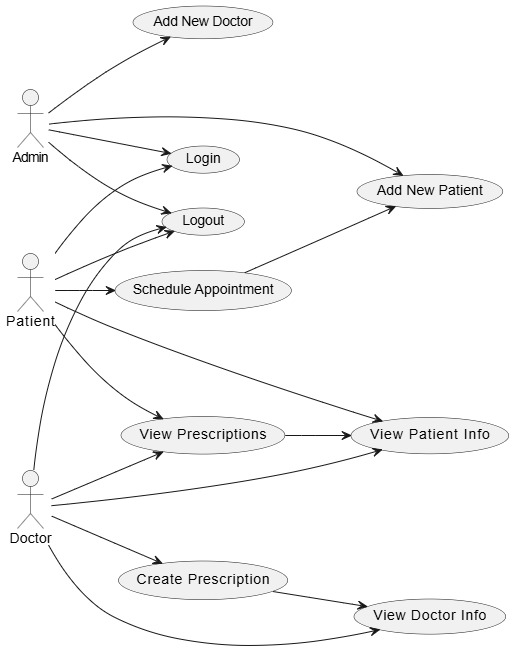
**3. Design Architecture**

**Class Design**

* + Doctor and Patient classes with additional fields.
  + Utilization of WebDoctorResource and WebPatientResource classes for REST API endpoints.
  + Prescription classes with additional fields.
  + Utilization of WebDoctorResource and WebPatientResource classes for REST API endpoints.
* **UML Diagram**



* **Use Case Diagram**



* **File Management**:
  + Robust file handling mechanisms for error-free data management.
* **Web UI**:
  + New HTML forms for managing appointments and searching records.

**Web Service Design**

* **RESTful Design (Resource-Based URLs)**

**Doctors Resource**

* /WebDoctor/doctors  
  Retrieve complete doctor information (HTML/JSON).
* /WebDoctor/doctor/{id}  
  Access individual doctor details (JSON).
* /WebDoctor/doctorSpecialization/{specialization}  
  Retrieve a list of doctors by specialization (JSON).
* /WebDoctor/addDoctor  
  Add a new doctor (HTML).
* /WebDoctor/deleteDoctor?doctorId={id}  
  Delete a doctor by ID (HTML).

**Patients Resource**

* /WebPatient/patients  
  Retrieve complete patient information (HTML/JSON).
* /WebPatient/patient/{id}  
  Access individual patient details (JSON).
* /WebPatient/doctorPatients/{doctorId}  
  Retrieve a list of patients assigned to a specific doctor (JSON).
* /WebPatient/appointmentDate/{date}  
  Retrieve a list of patients with appointments on a specific date (JSON).
* /WebPatient/addPatient  
  Add a new patient (HTML).
* /WebPatient/updateAppointmentDate?patientId={id}&newDate={date}  
  Update the appointment date for a specific patient (HTML).
* /WebPatient/deletePatient?patientId={id}  
  Delete a patient by ID (HTML).
* /WebPatient/patientSearch?doctorId={doctorId}&appointmentDate={date}  
  Retrieve upcoming appointments for a doctor starting from a specific date (Text).

**Clinic Resource**

* /WebClinic/info  
  Retrieve clinic statistics, including counts of doctors and patients by criteria (HTML).
* / WebClinic /{patientId}: Retrieve all prescriptions for a given patient.
* /WebClinic /{doctorId}: Retrieve all prescriptions issued by a specific doctor.
* /WebClinic /addprescriptions: Add a new prescription to the system.

**Client-Side Implementation**

**ClientDoctorManagementREST**

This Java client uses the Jersey Client framework to interact with the WebDoctorManagement REST service. The client supports the following functionalities:

1. **Retrieve and Display All Doctors (HTML Format)**  
   The client sends a GET request to /WebDoctor/doctors and retrieves a list of all doctors in HTML format for display.
2. **Search Doctor by ID (JSON Format)**  
   The client sends a GET request to /WebDoctor/doctor/{id} where {id} is the doctor’s unique identifier. It fetches the doctor's details in JSON format.
3. **Search Doctors by Specialization (JSON Format)**  
   The client sends a GET request to /WebDoctor/doctorSpecialization/{specialization} to retrieve all doctors matching the provided specialization.
4. **Add a New Doctor**  
   The client sends a POST request to /WebDoctor/addDoctor with form parameters such as doctor ID, name, specialization, max patients, and consultation fee. The response confirms the addition in HTML format.
5. **Delete a Doctor by ID**  
   The client sends a DELETE request to /WebDoctor/deleteDoctor with a query parameter doctorId. The response indicates success or failure in HTML format.
6. **Continuous Interaction**  
   The client provides an interactive console where users can perform operations repeatedly until they choose to exit.

**ClientPatientManagementREST**

This Java client interacts with the WebPatientManagement REST service and provides the following functionalities:

1. **Retrieve and Display All Patients (HTML Format)**  
   Sends a GET request to /WebPatient/patients to retrieve a list of all patients in HTML format.
2. **Search Patient by ID (JSON Format)**  
   Sends a GET request to /WebPatient/patient/{id} where {id} is the patient’s unique identifier. Returns the patient's details in JSON format.
3. **Search Patients by Doctor ID (JSON Format)**  
   Sends a GET request to /WebPatient/doctorPatients/{doctorId} to retrieve a list of all patients assigned to a specific doctor.
4. **Search Patients by Appointment Date (JSON Format)**  
   Sends a GET request to /WebPatient/appointmentDate/{date} where {date} is the desired appointment date. Returns a list of patients with appointments on that date.
5. **Book a New Appointment**  
   Sends a POST request to /WebPatient/addPatient with form parameters such as name, age, condition, assigned doctor, and appointment date. The response confirms booking success in HTML format.
6. **Reschedule an Appointment**Sends a PUT request to /WebPatient/updateAppointmentDate with query parameters patientId and newDate to update an appointment date for a patient.
7. **Delete Patient by ID**  
   Sends a DELETE request to /WebPatient/deletePatient with the patientId as a query parameter. The response indicates success or failure in HTML format.

**clinicManagementREST**

This RESTful web service handles the core functionalities of clinic management, offering various endpoints for managing doctors, patients, appointments, and prescriptions. The service is built using Jakarta REST Web Services and deployed on GlassFish Server

1. **Retrieve and Display Clinic Stats**

Sends a GET request to /WebClinc/info to retrieve a list of all patients and Doctors including total revenue in HTML format.

1. **Add Prescription by PatientID and Doctors**

Sends a POST request to /WebClinic/addPrescription to submit a new prescription for a patient by a specific doctor.

**Continuous Interaction**  
Like the Doctor client, this client provides an interactive console for users to perform multiple operations until they choose to exit.

**4. Data Structures Implementation**

**Primary Data Structures**

* **HashMap<String, Doctor> doctorMap:** Efficient doctor information retrieval
* **HashMap<String, Patient> patientMap**: Efficient patient information retrieval

**Data Processing Structures**

* **Java Streams:** For data filtering, statistical calculations, and sorting
* **StringBuilder:** For dynamic HTML generation and response construction
* **Collections Framework:** For list generation, sorting, and data grouping

**Key Features**

* **Data Loading:** File-based data initialization with error handling
* **Data Processing:** Stream-based filtering and statistical calculations
* **Response Generation:** Dynamic HTML and JSON response formatting

**5. Future Enhancements**

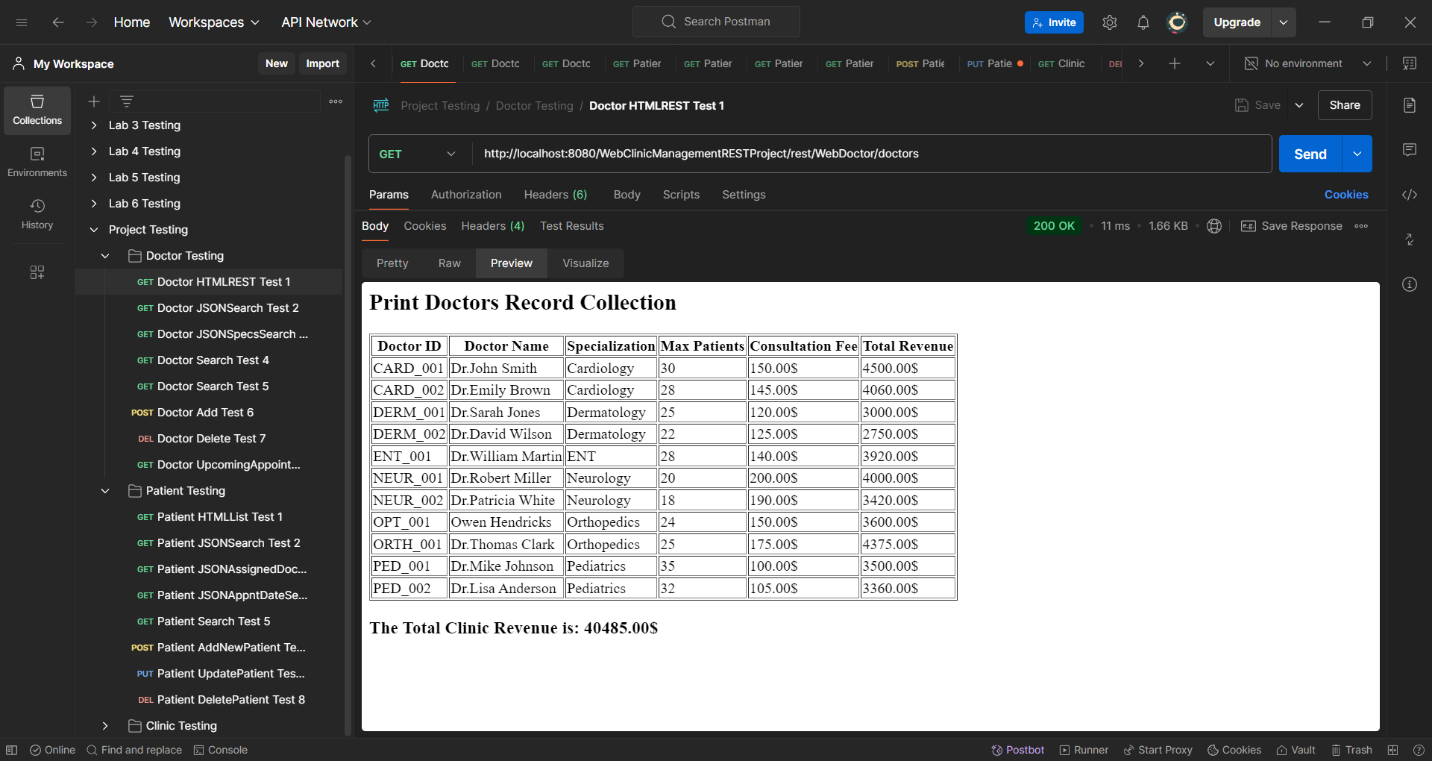
* **Advanced Security**:
  + Implement hashed passwords for secure login.
  + Add session timeouts and multi-factor authentication.
* **Data Visualization**:
  + Integrate chart libraries for graphical representation of statistics.
* **Scalability**:
  + Move to a database like MySQL or PostgreSQL for large-scale data handling.
* **Medication Management**:
  + Allow doctors to prescribe medications, track prescriptions, and manage refills for patients. Patients could also receive reminders for medication intake.
* **Advanced Reporting:**
  + Data visualization and downloadable reports
* **Feedback System:**
  + Patient ratings for service improvement
    1. **Conclusion**

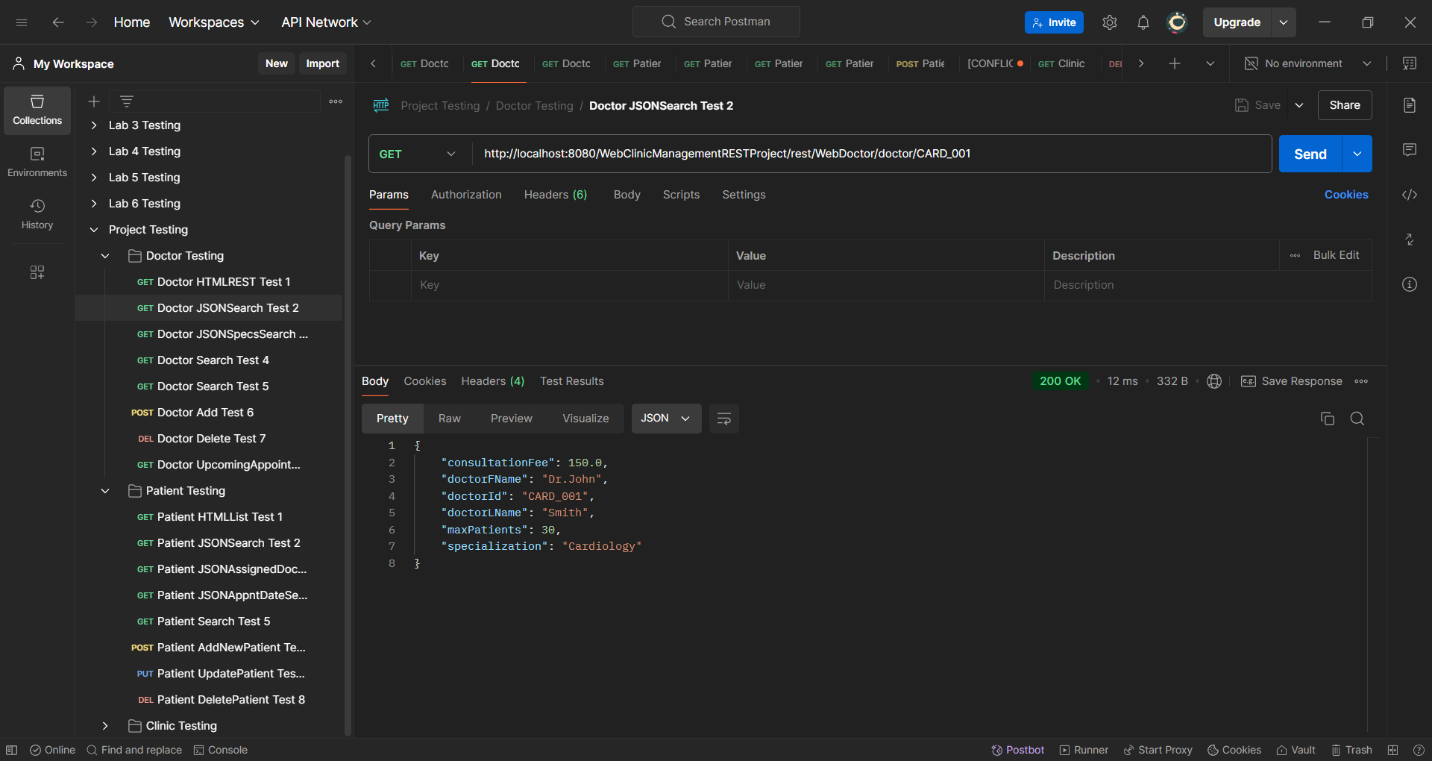
The **Web Clinic Management System** highlights how RESTful architecture can transform clinic operations, greatly improving data accessibility and operational efficiency. By integrating role-based dashboards, advanced RESTful APIs, and dynamic client-side applications, the system serves as a comprehensive platform for managing clinical workflows. Key features such as streamlined doctor and patient management, efficient appointment scheduling and rescheduling, and insightful statistical reporting enable better decision-making and operational optimization.

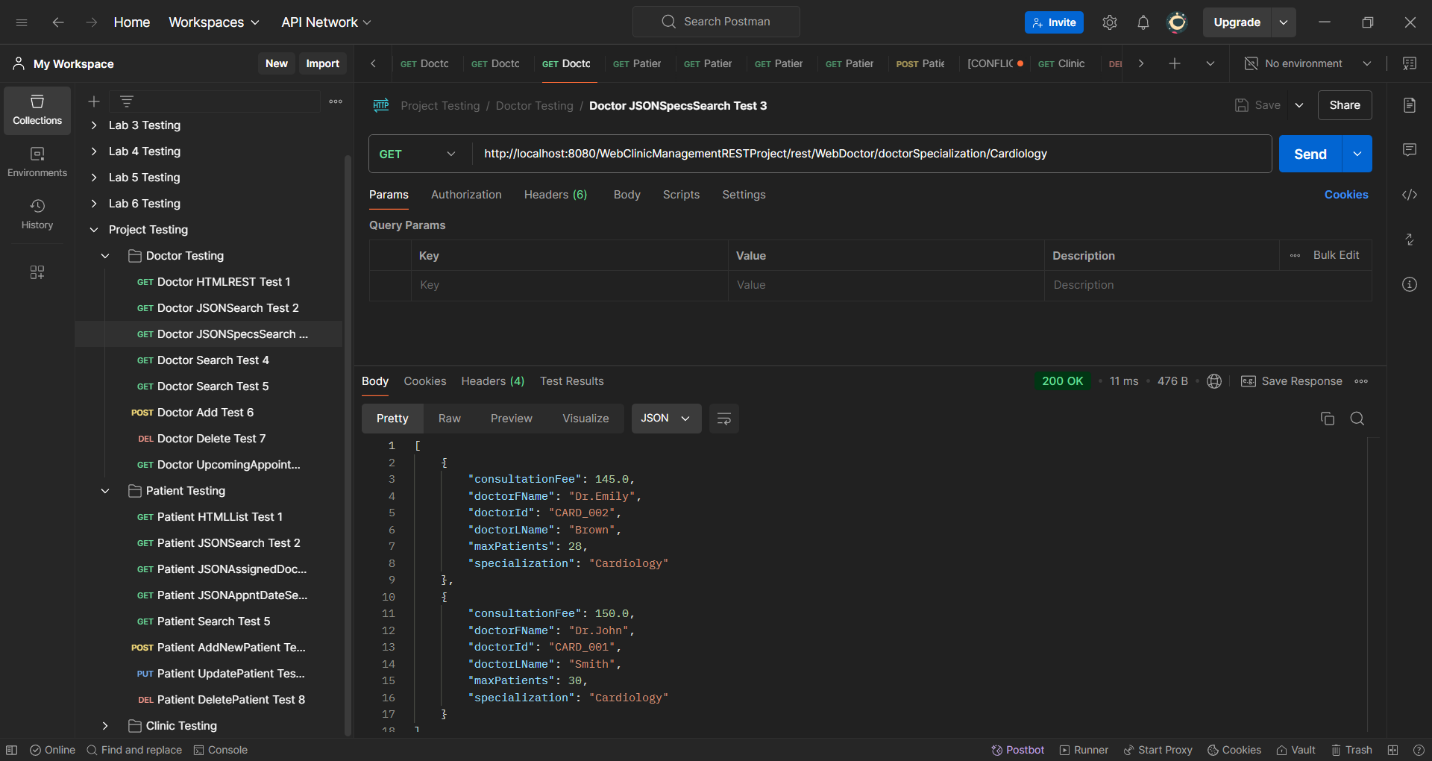
This project underscores the significance of leveraging RESTful services and client-side technologies for effective data management and intuitive user experiences. The use of dynamic HTML forms and JavaScript validation ensures a seamless interface, while real-time file updates and optimized data retrieval using HashMaps enhance system responsiveness and reliability. Security features like session-based logins provide a strong foundation for secure and controlled access.

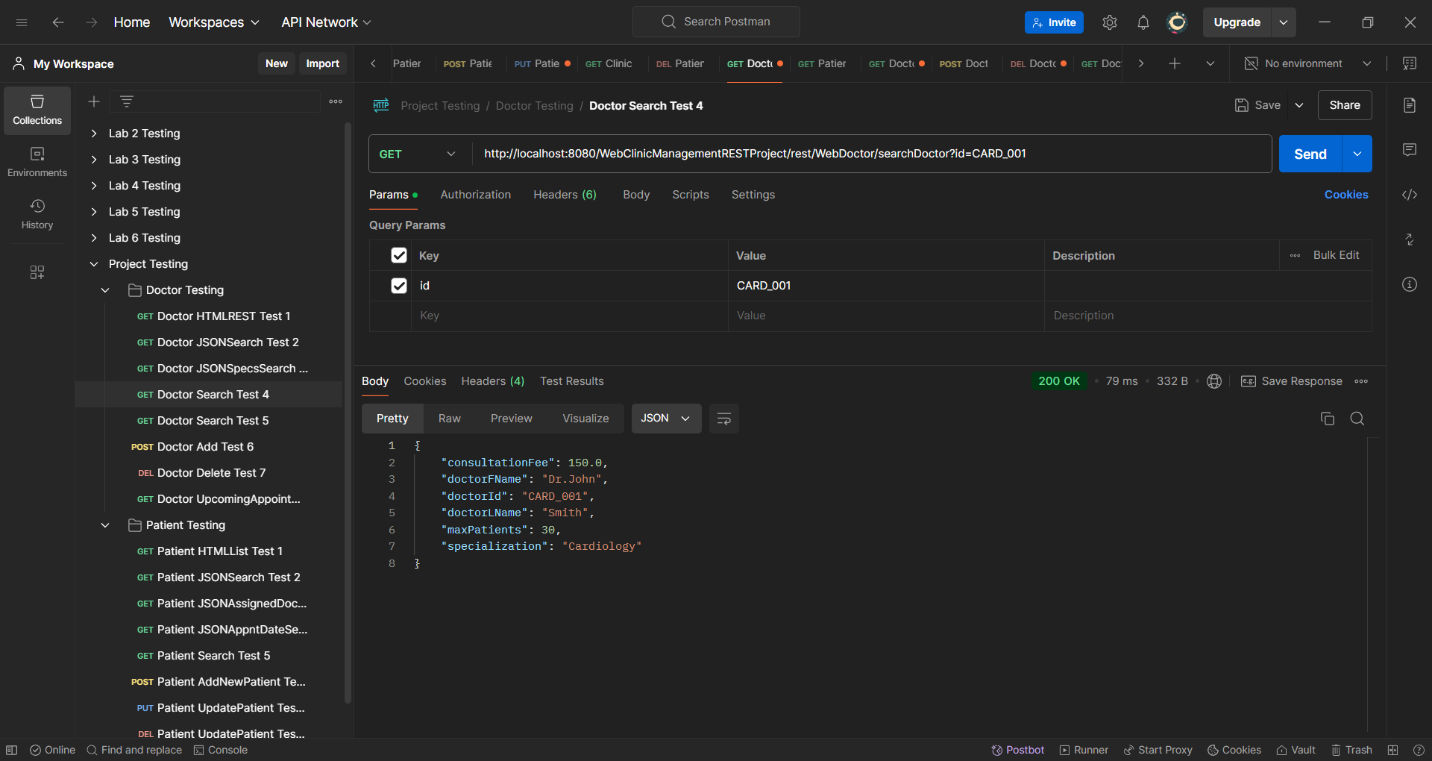
Looking ahead, enhancements such as secure password hashing, advanced data visualization, and database integration will ensure the system remains scalable, secure, and adaptable to evolving needs. The **Web Clinic Management System** exemplifies a holistic approach to modern software design and implementation in healthcare management, setting a strong foundation for future advancements.

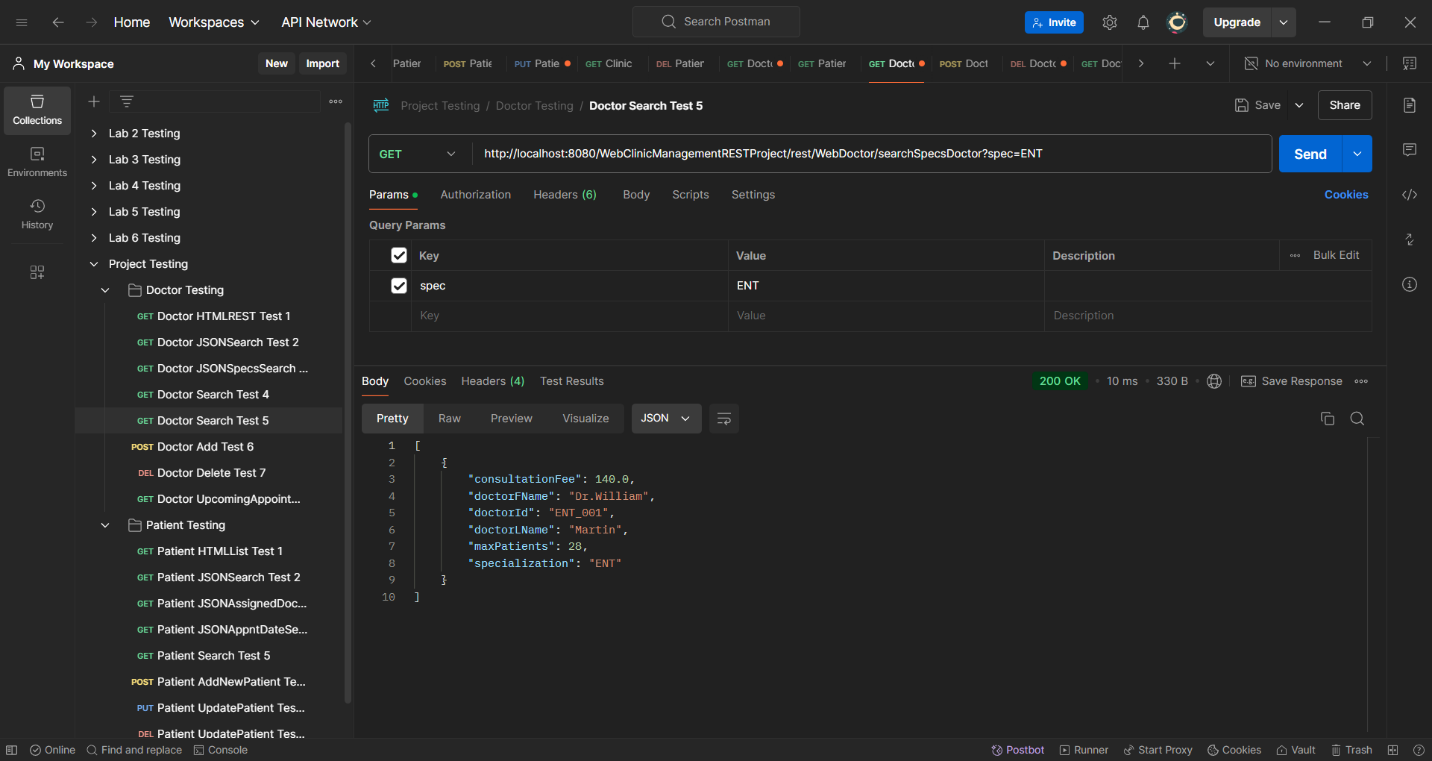
**SCREENSHOTS**

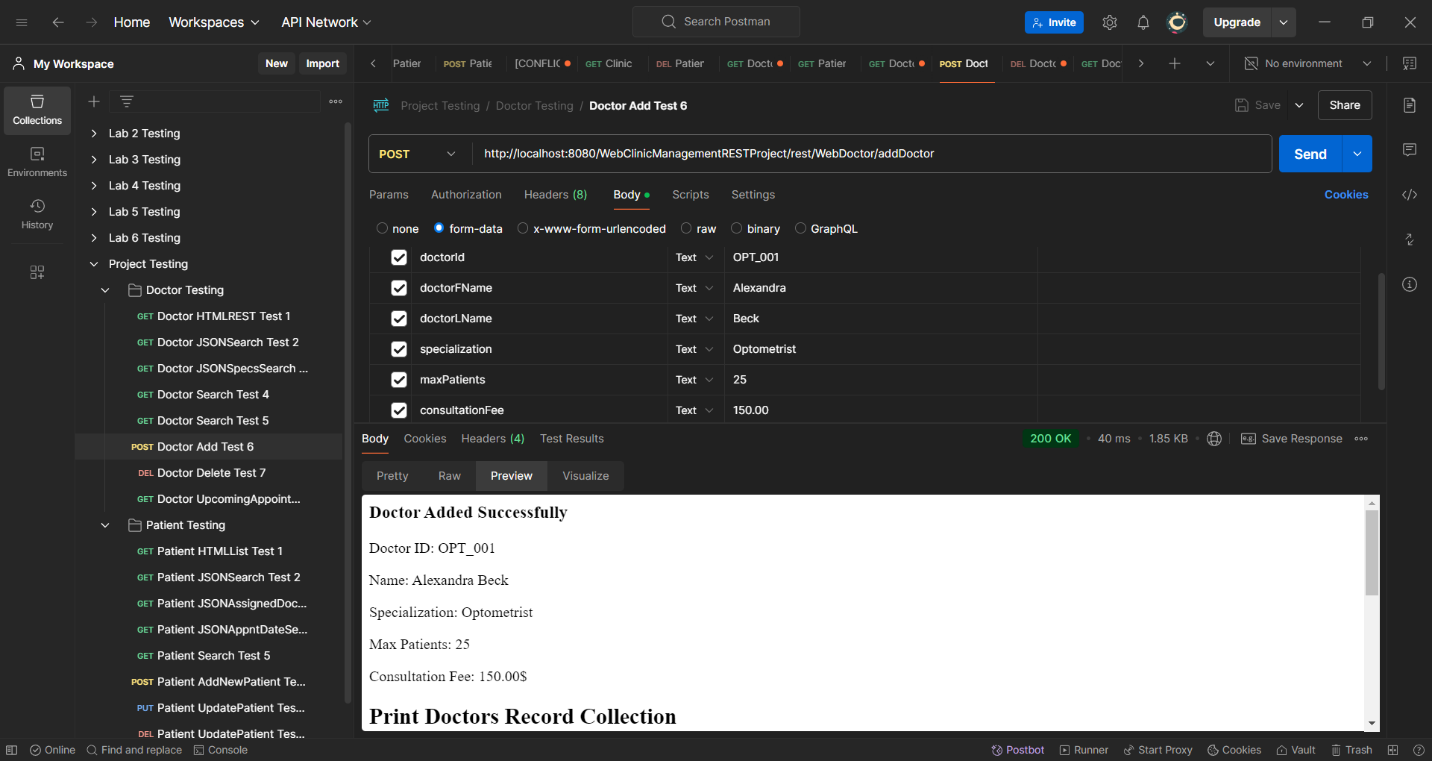
****

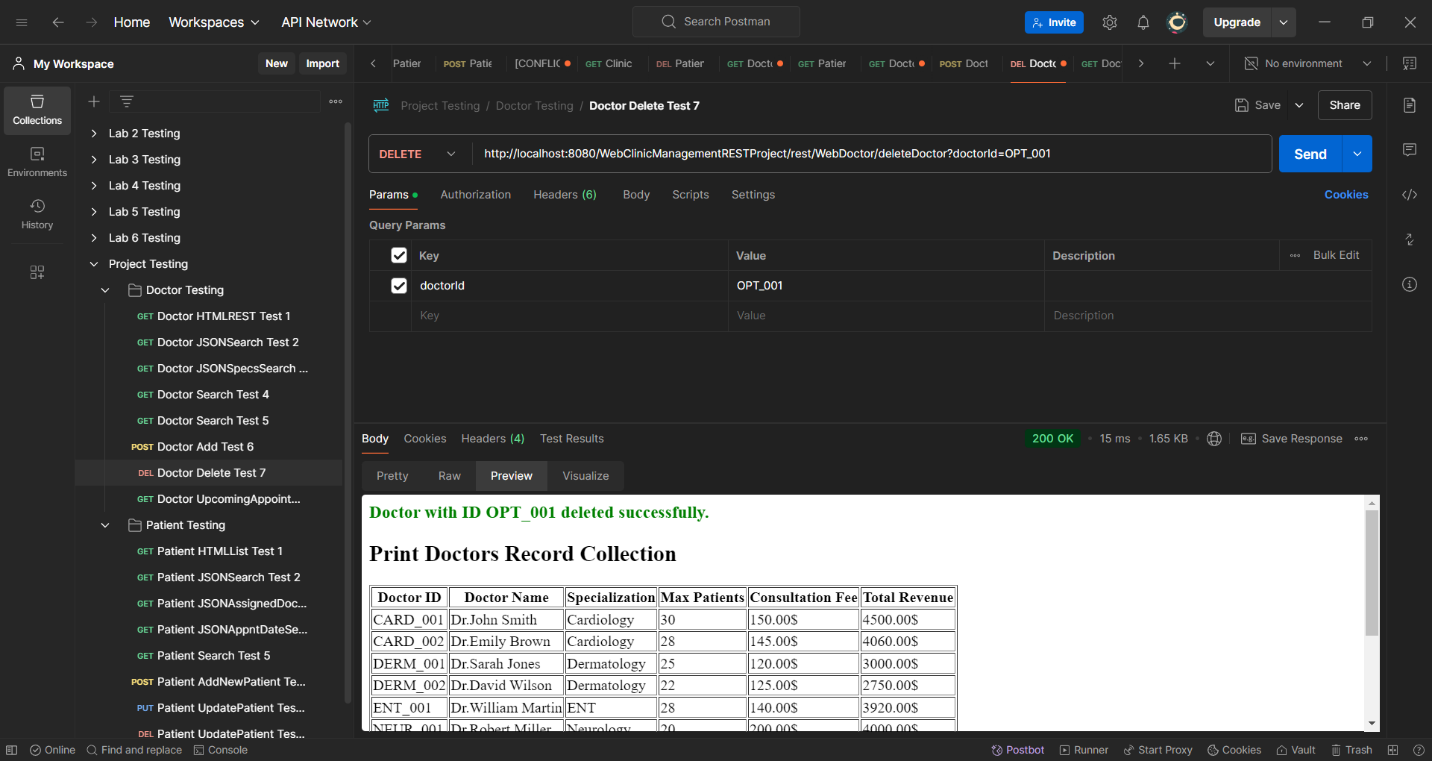
****

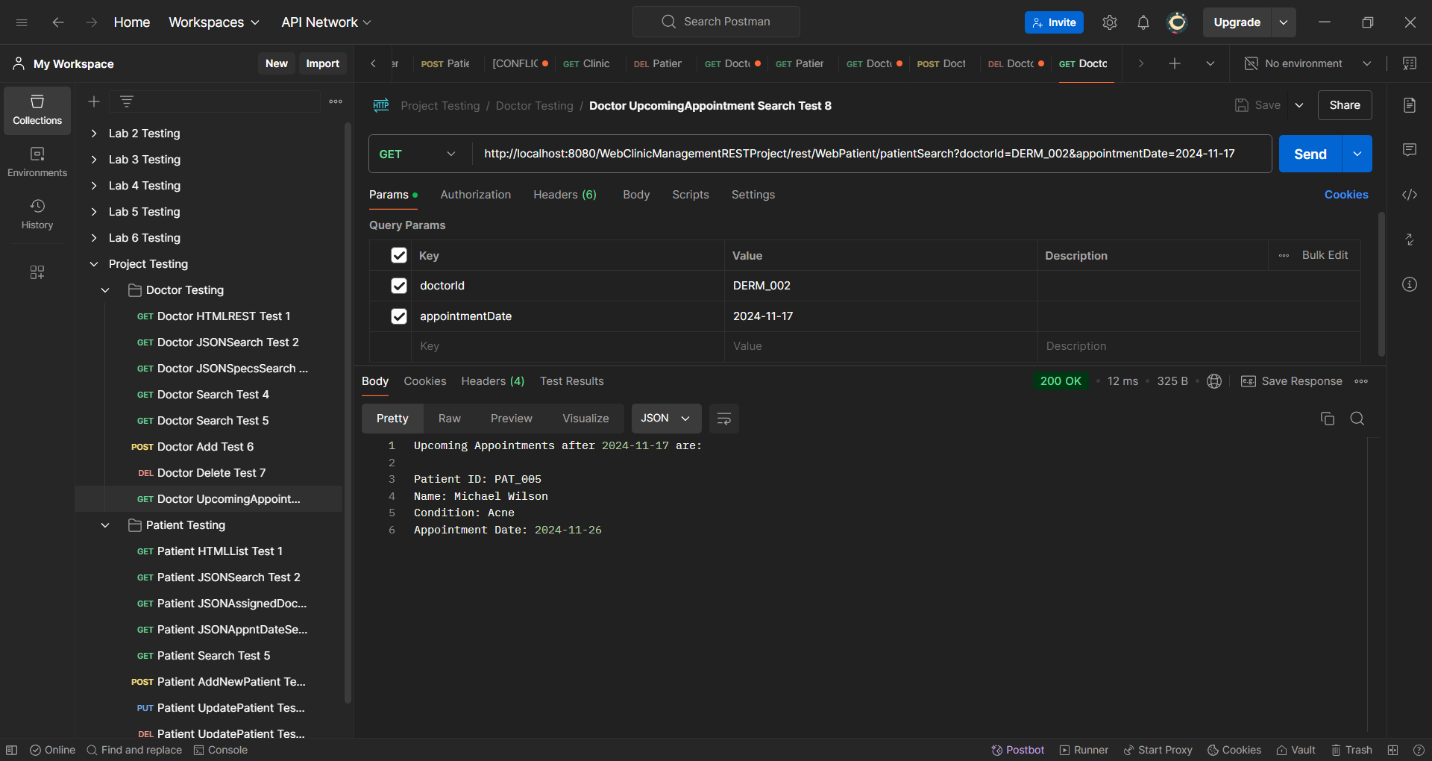
****

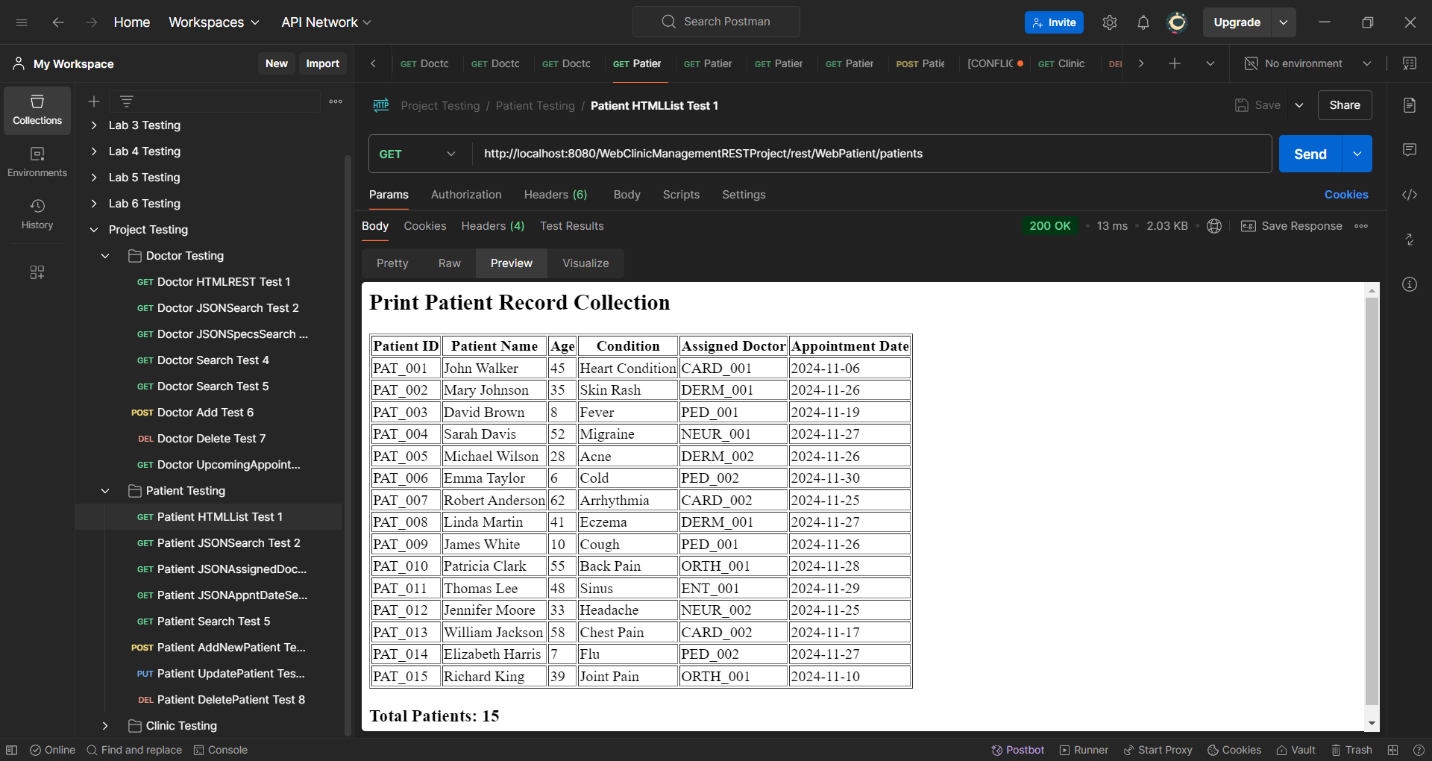
****

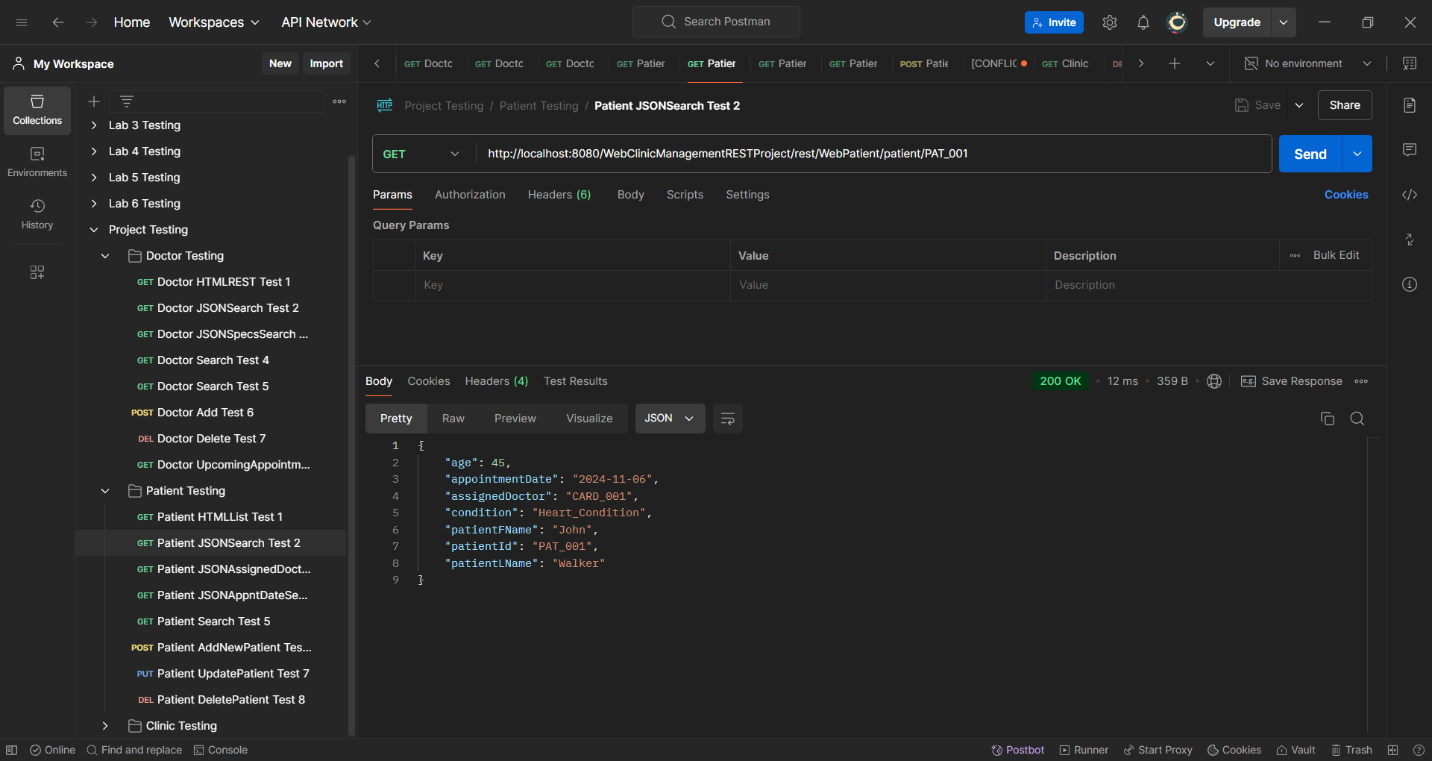
****

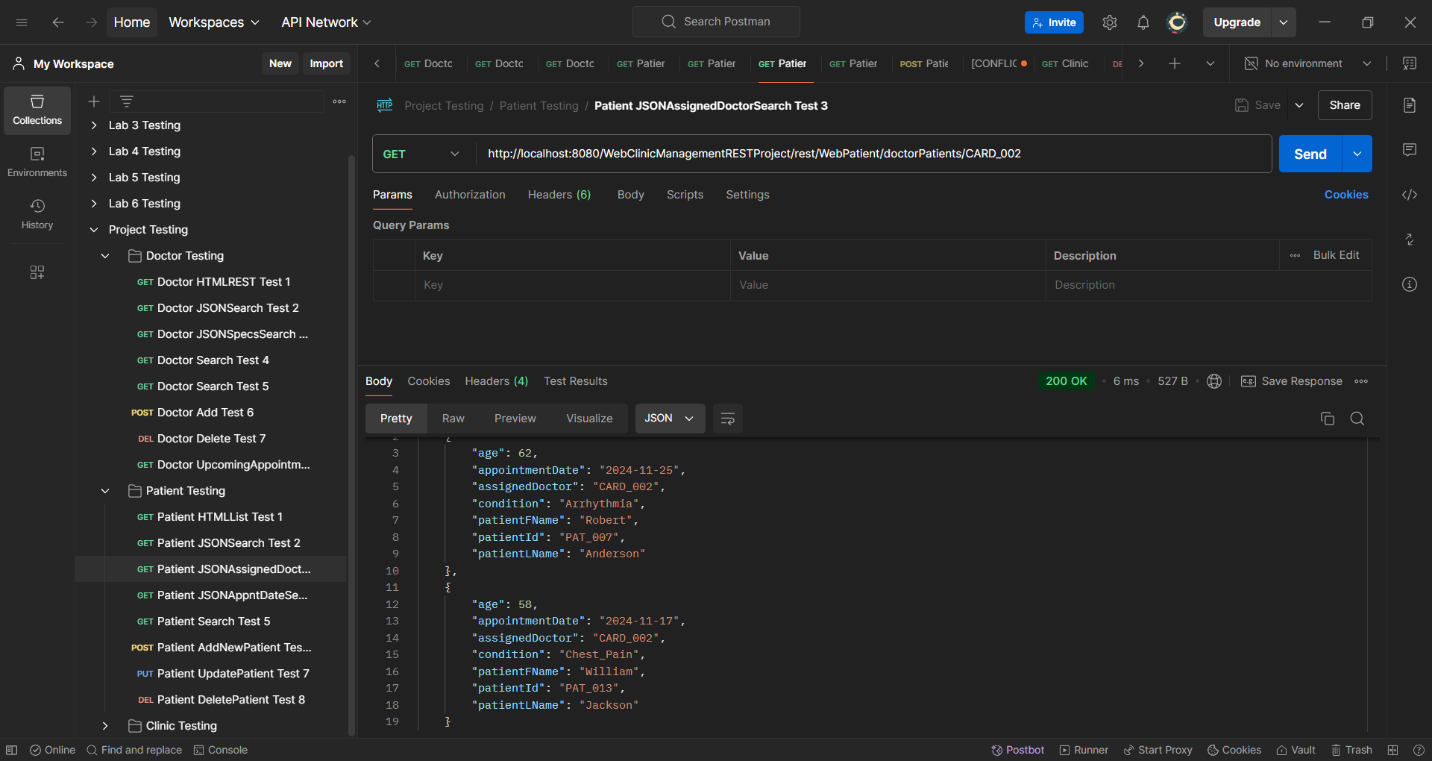
****

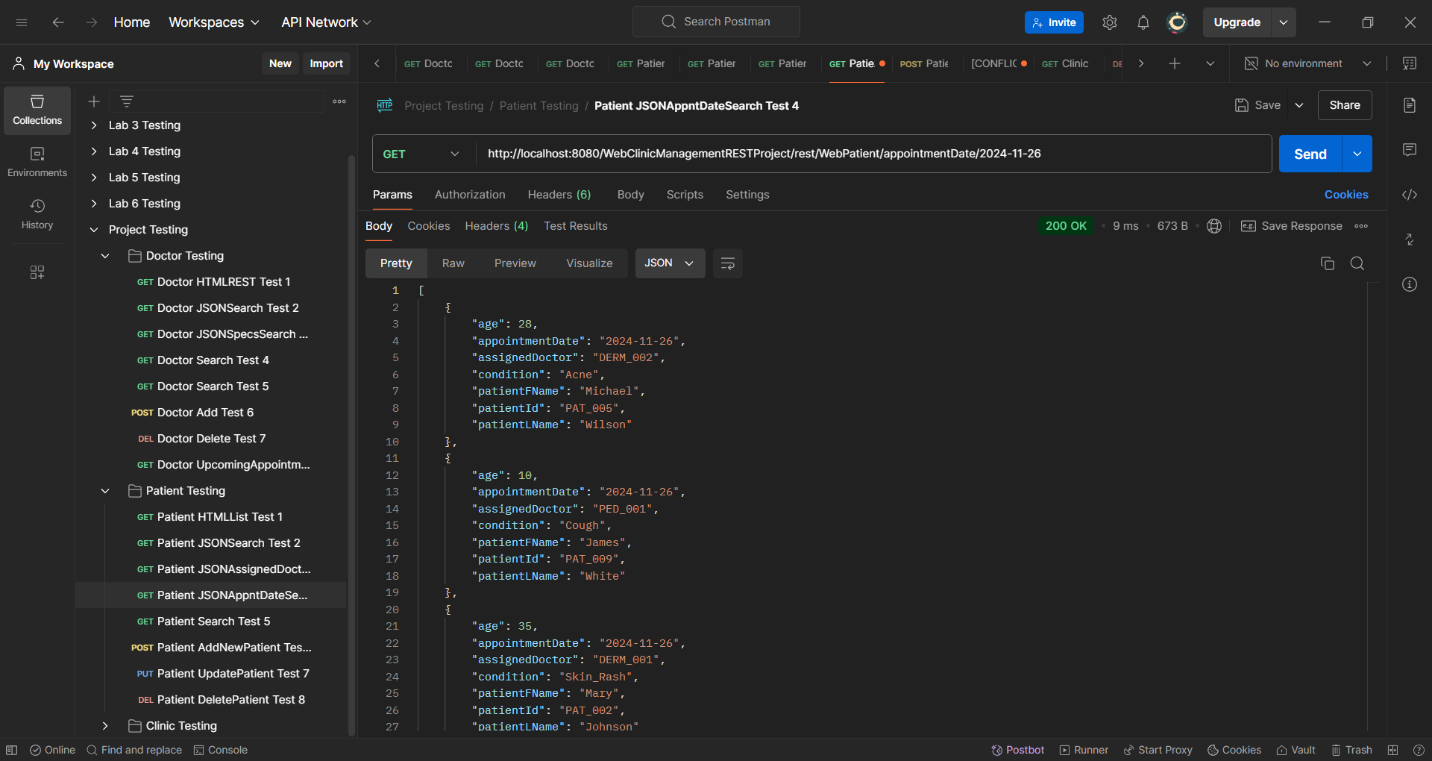
****

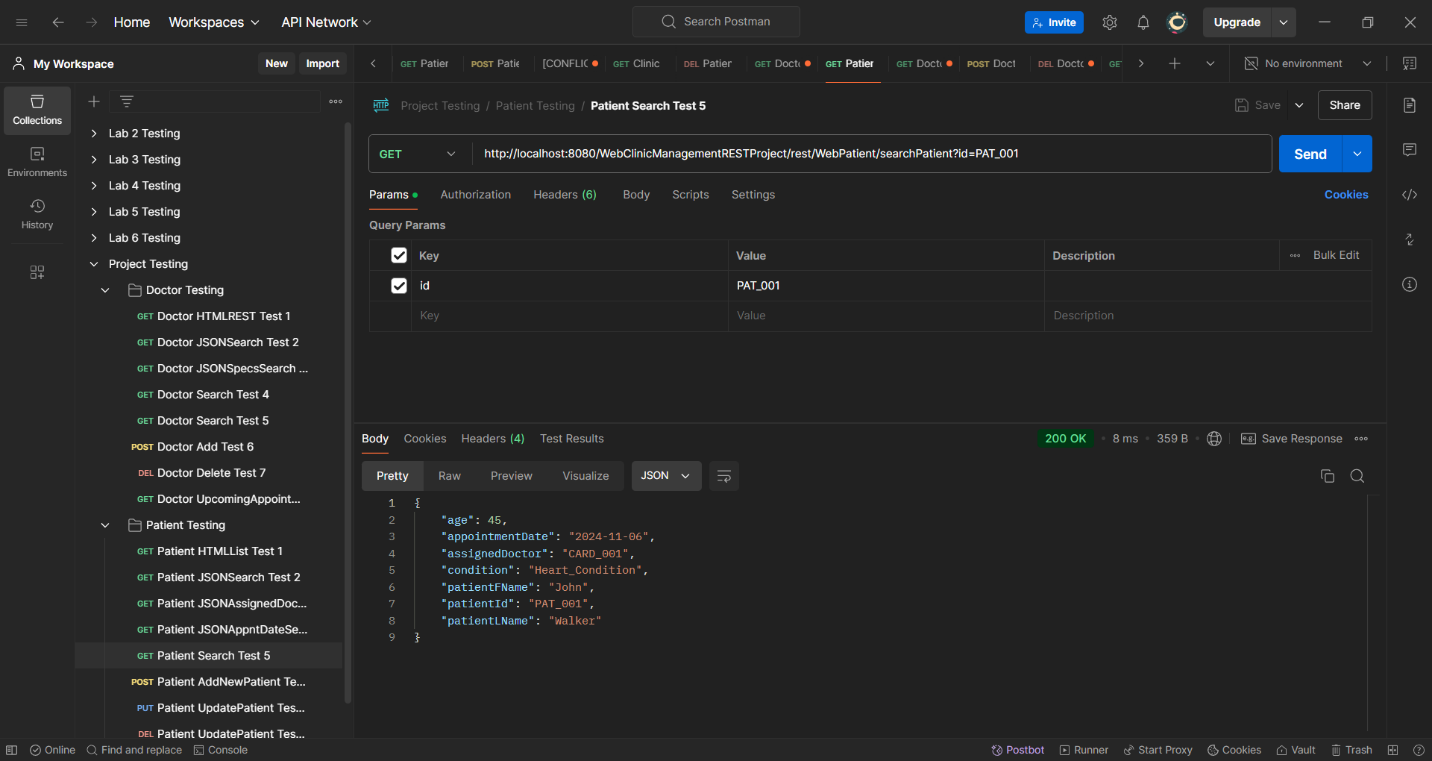
****

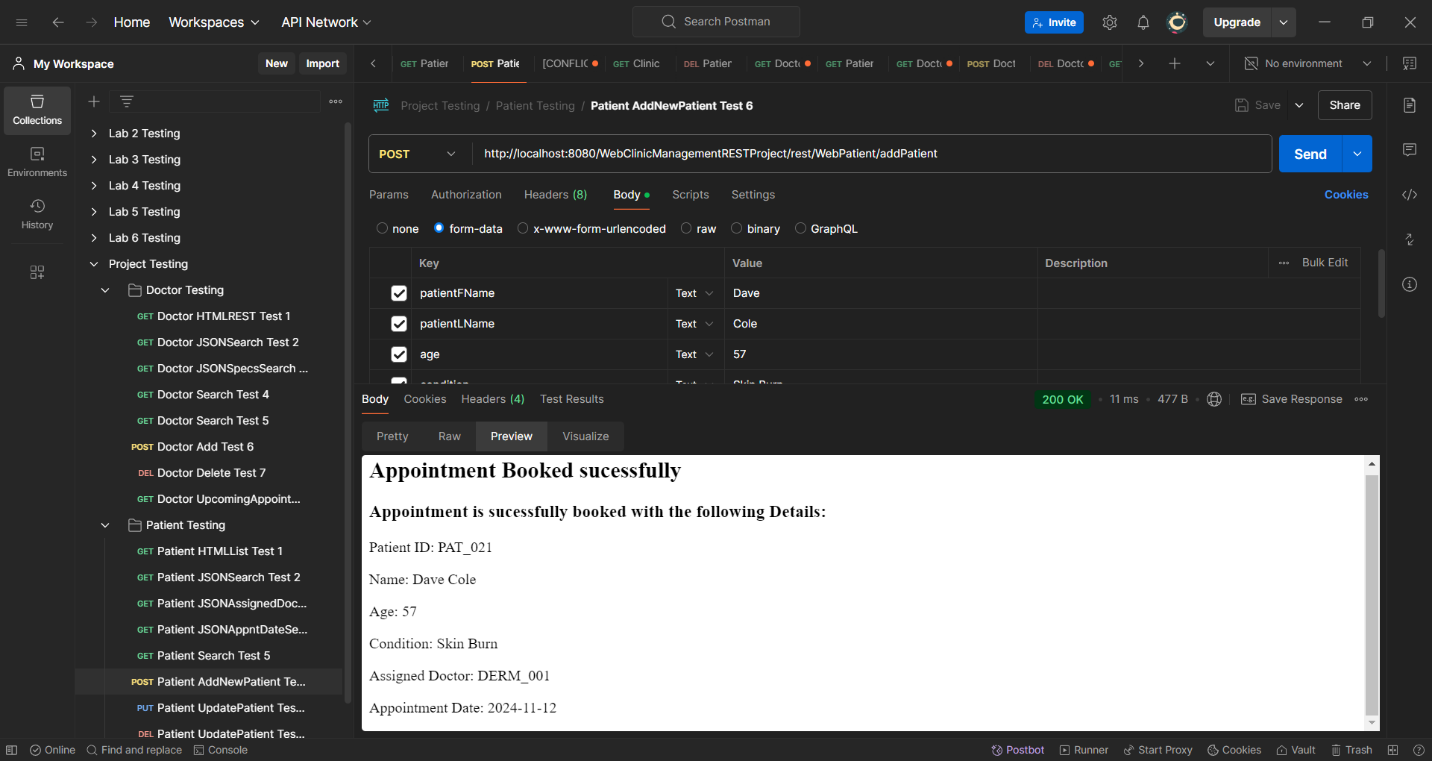
****

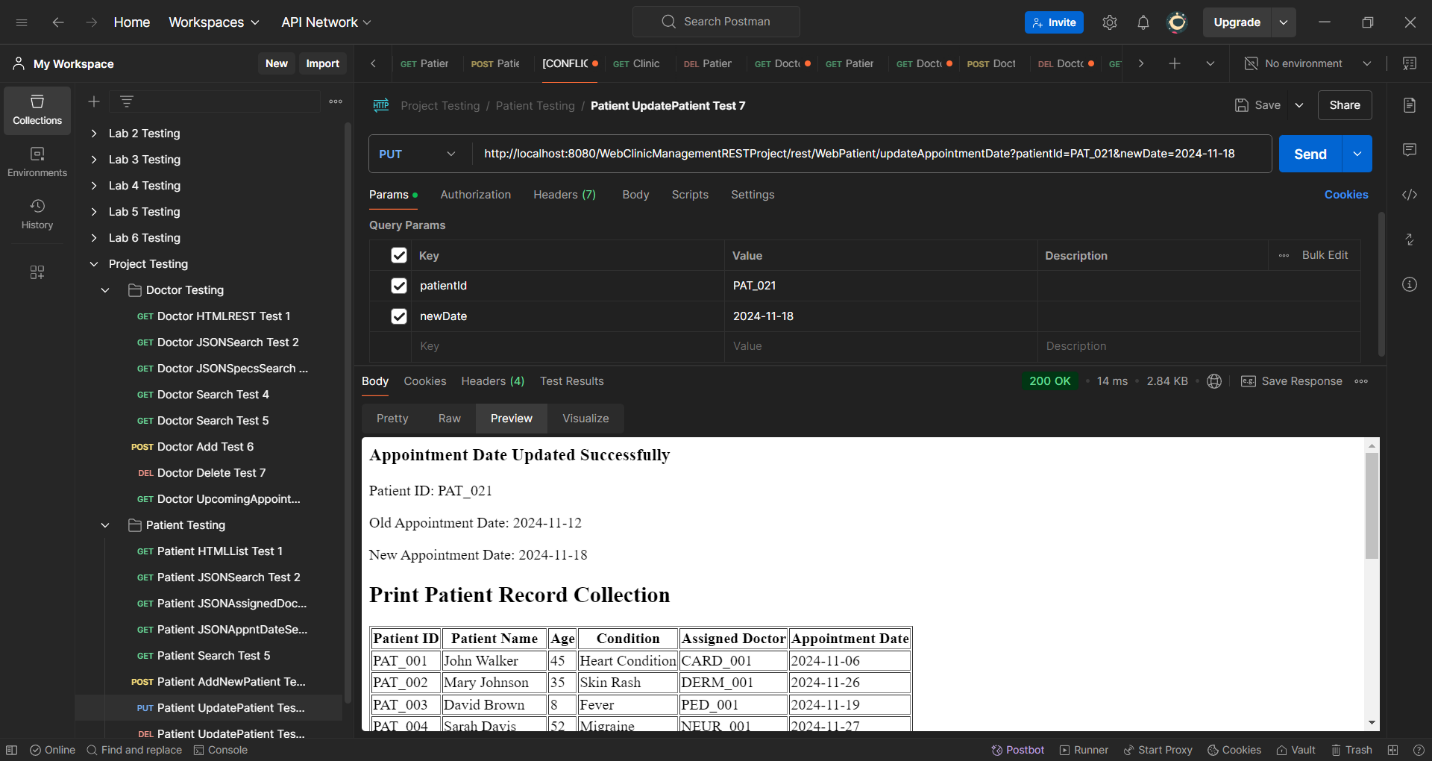
****

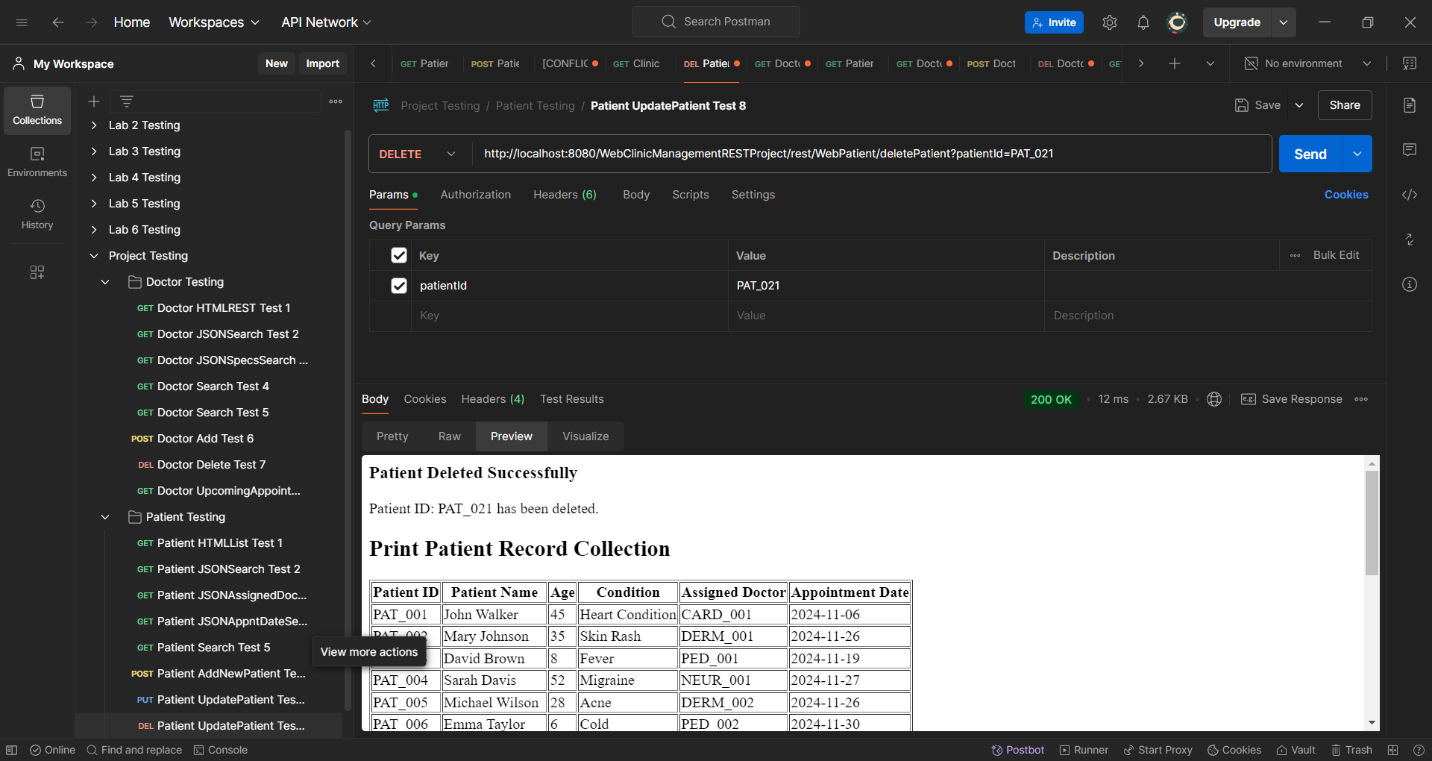
****

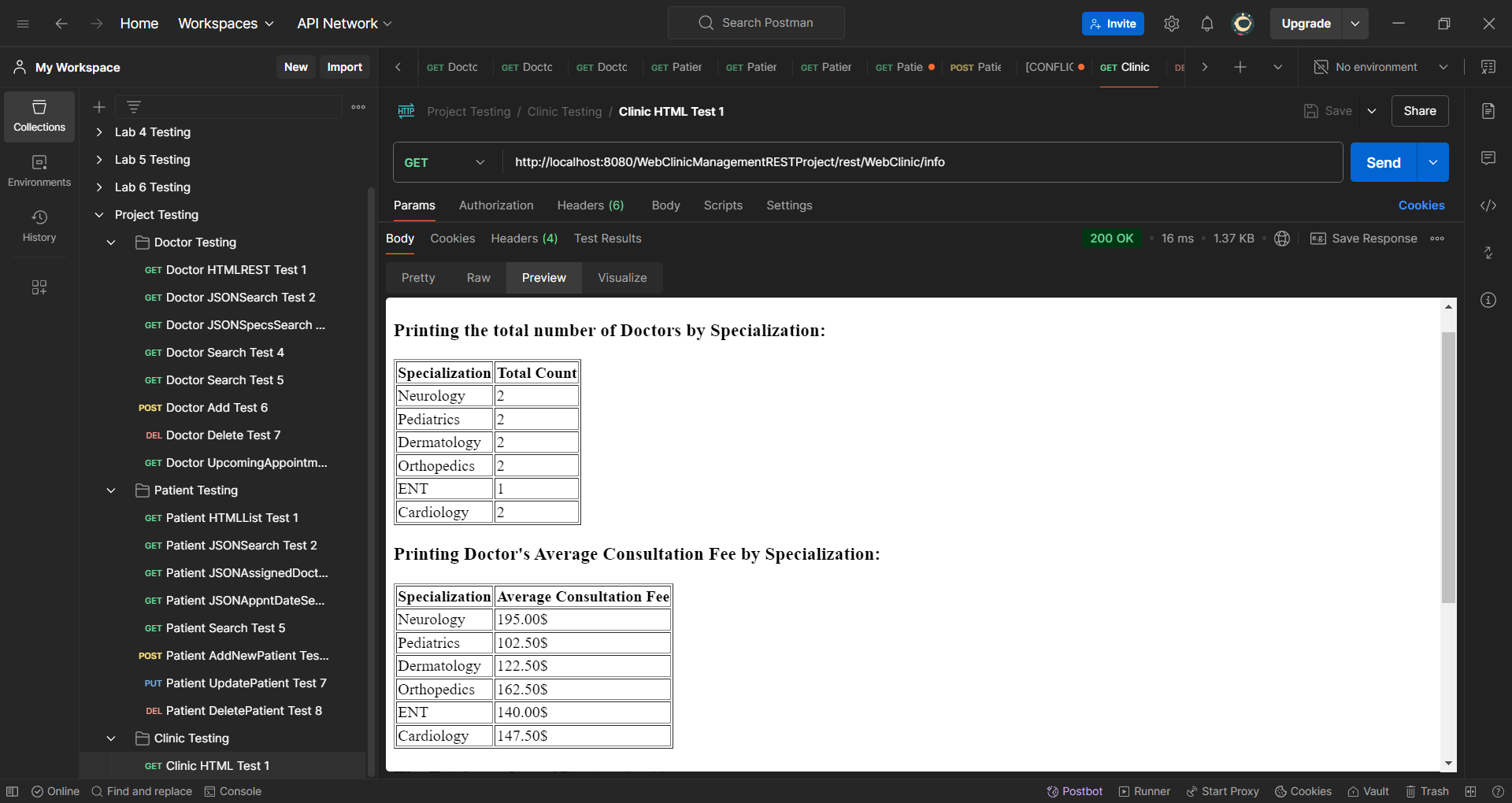
****

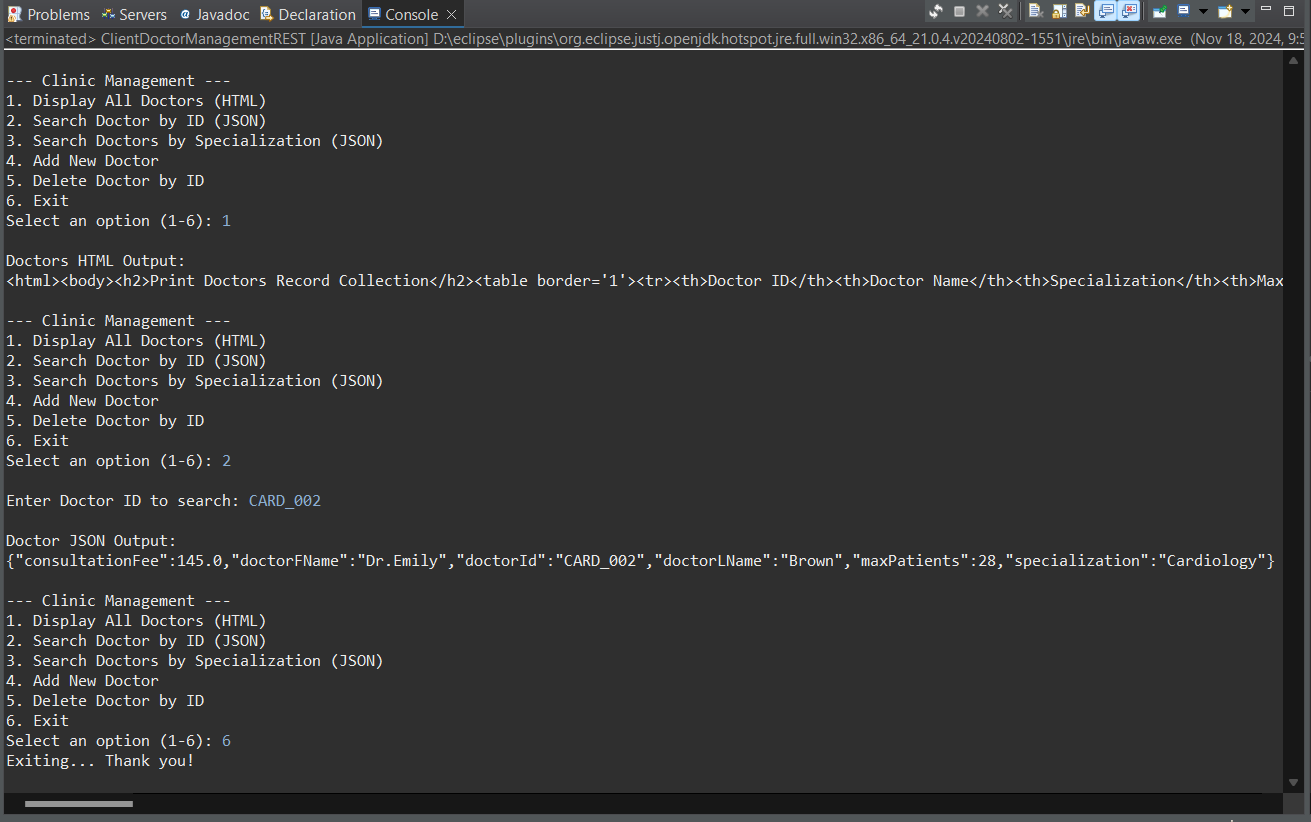
****

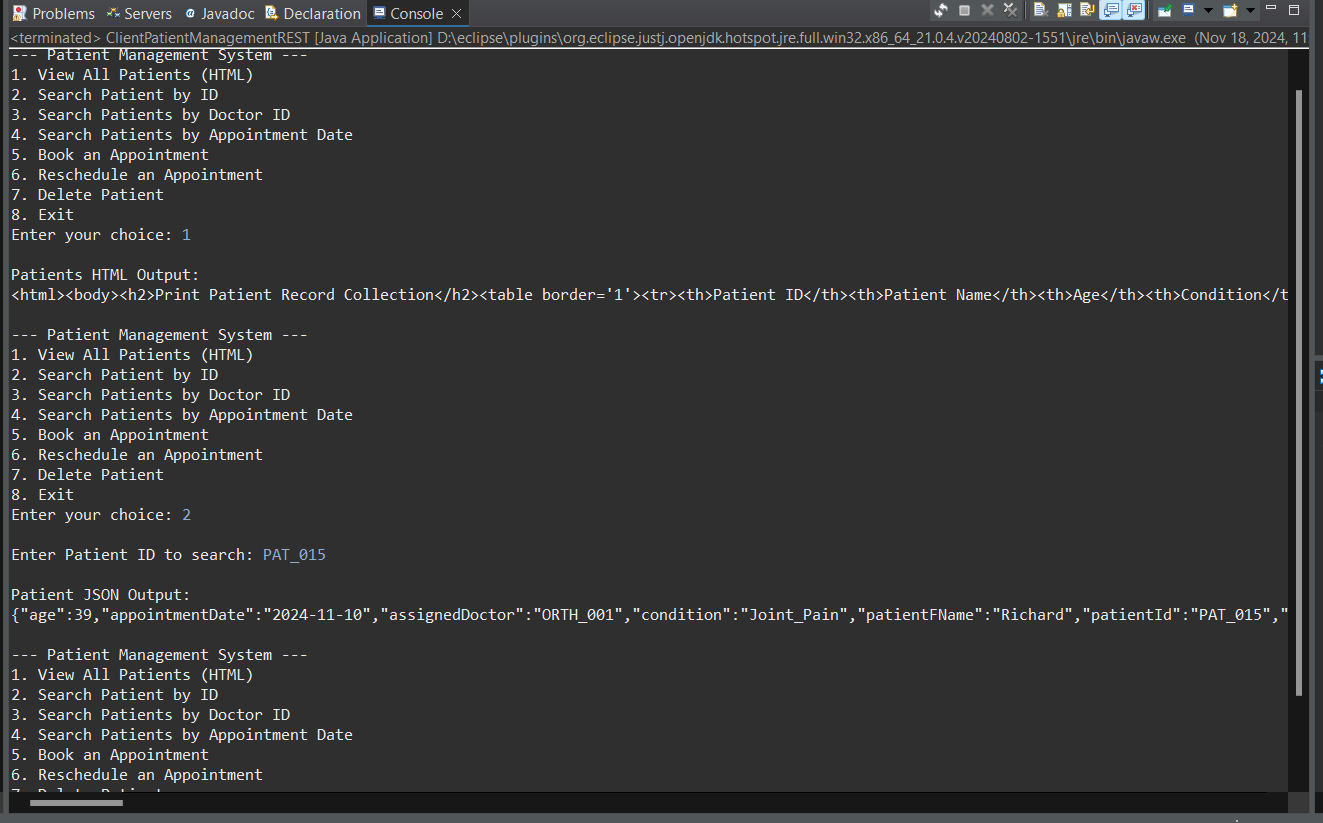
****

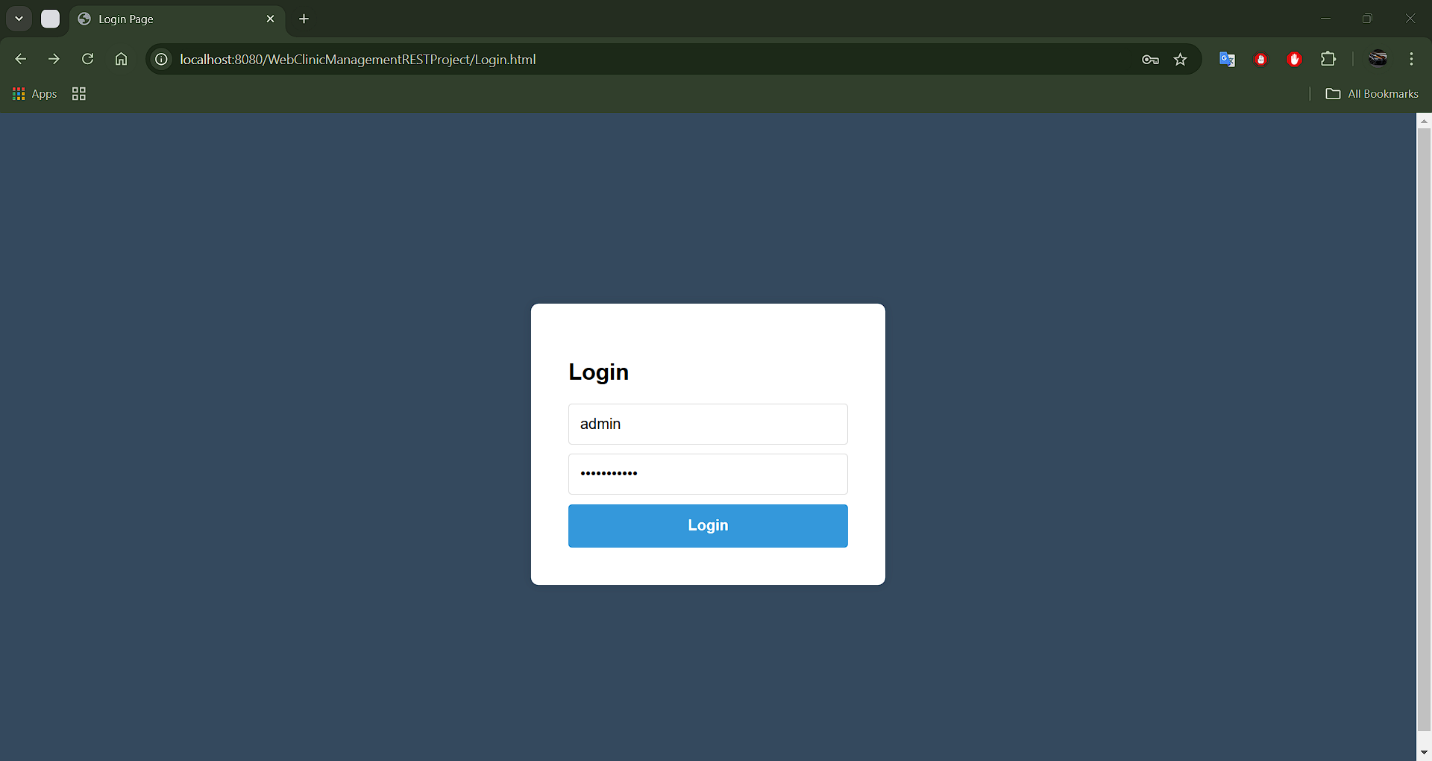
****

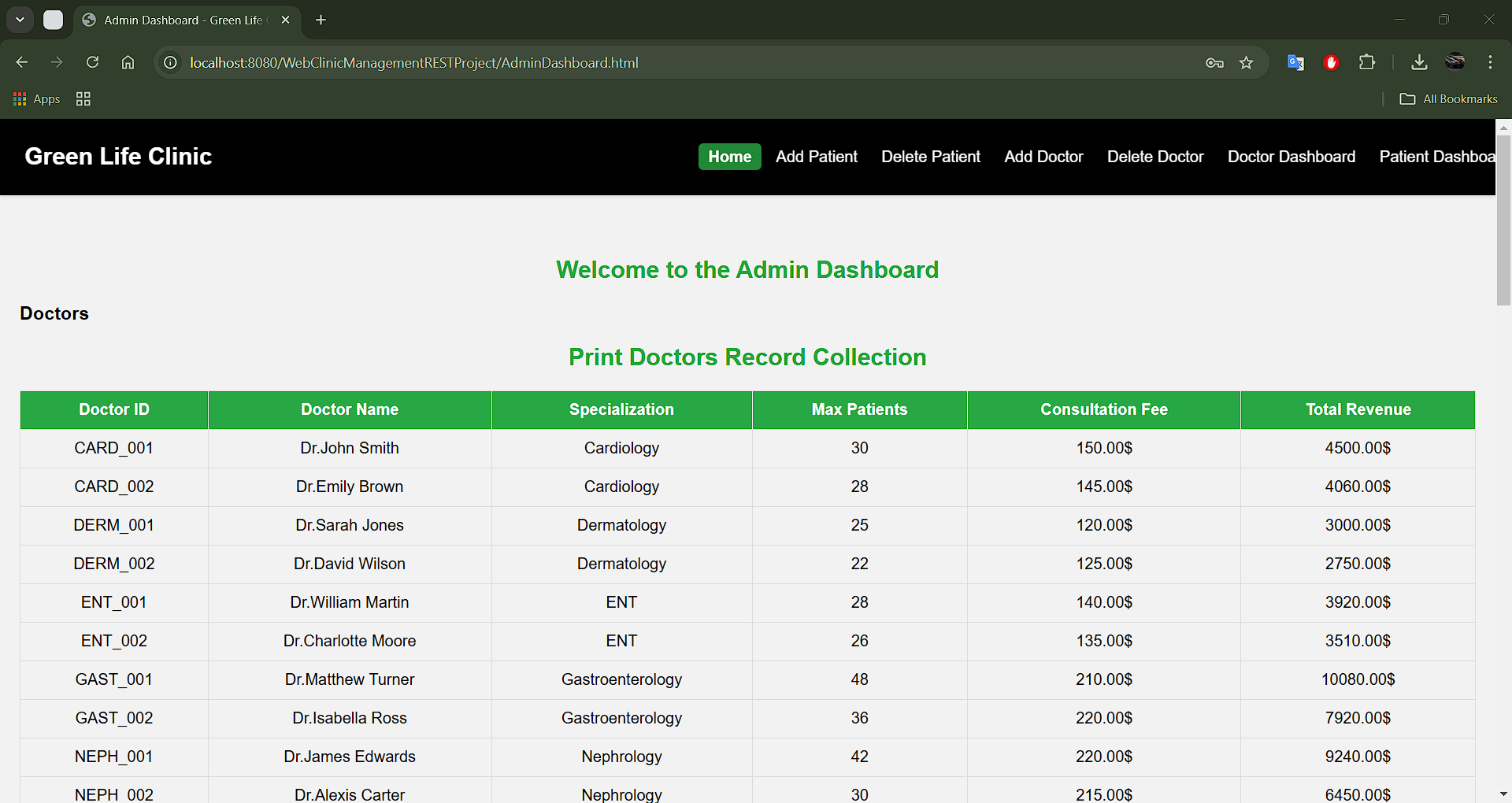
****

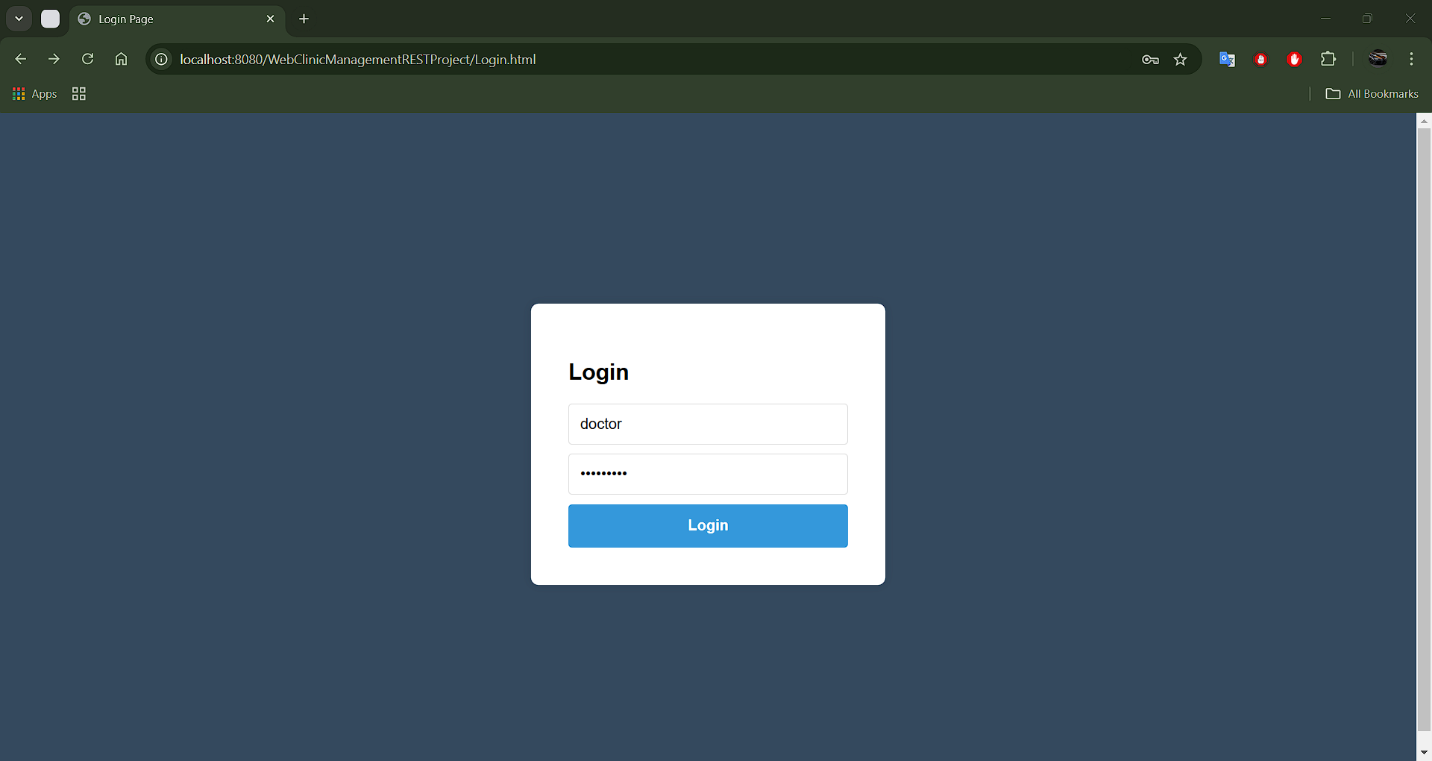
****

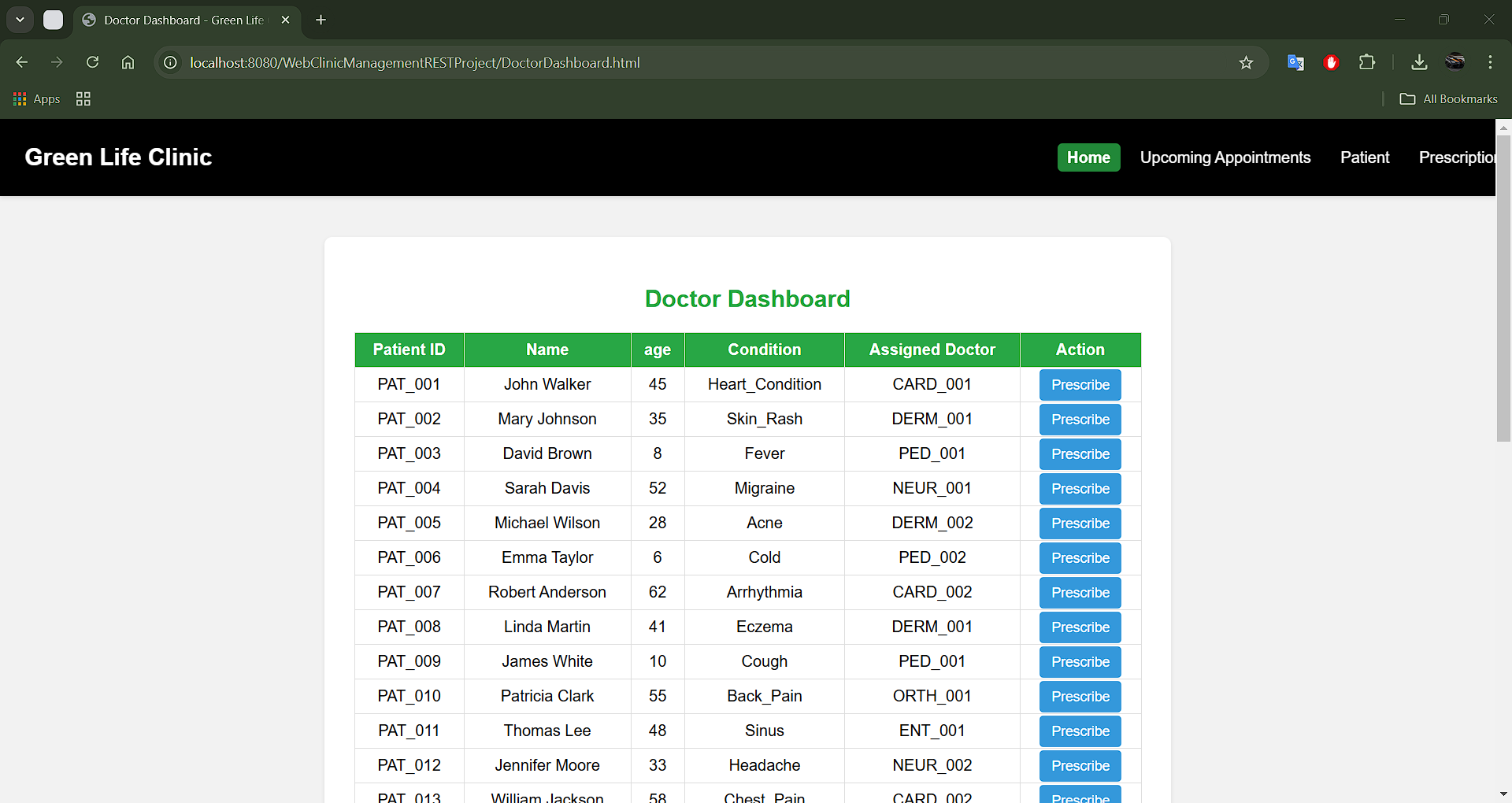
****

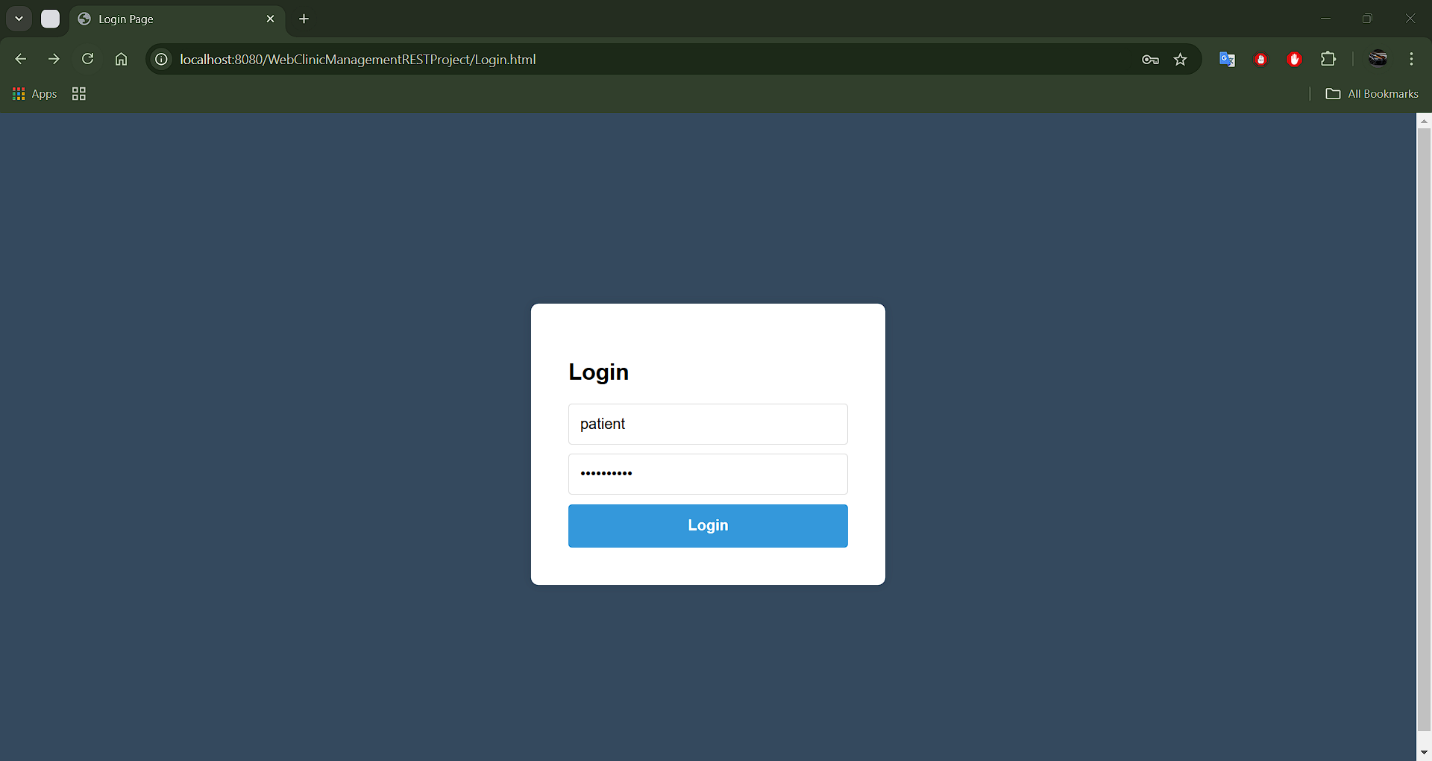
****

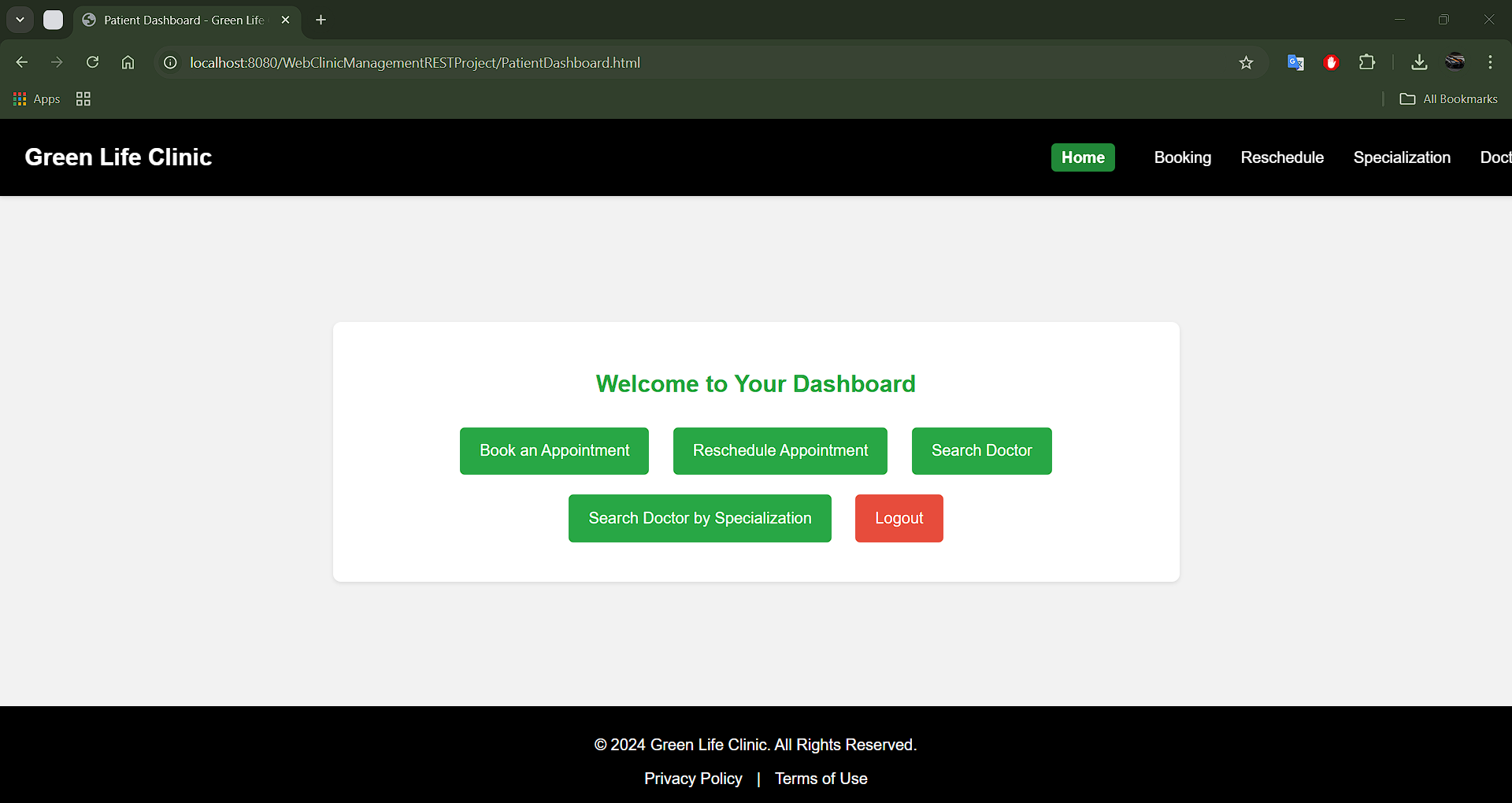
****

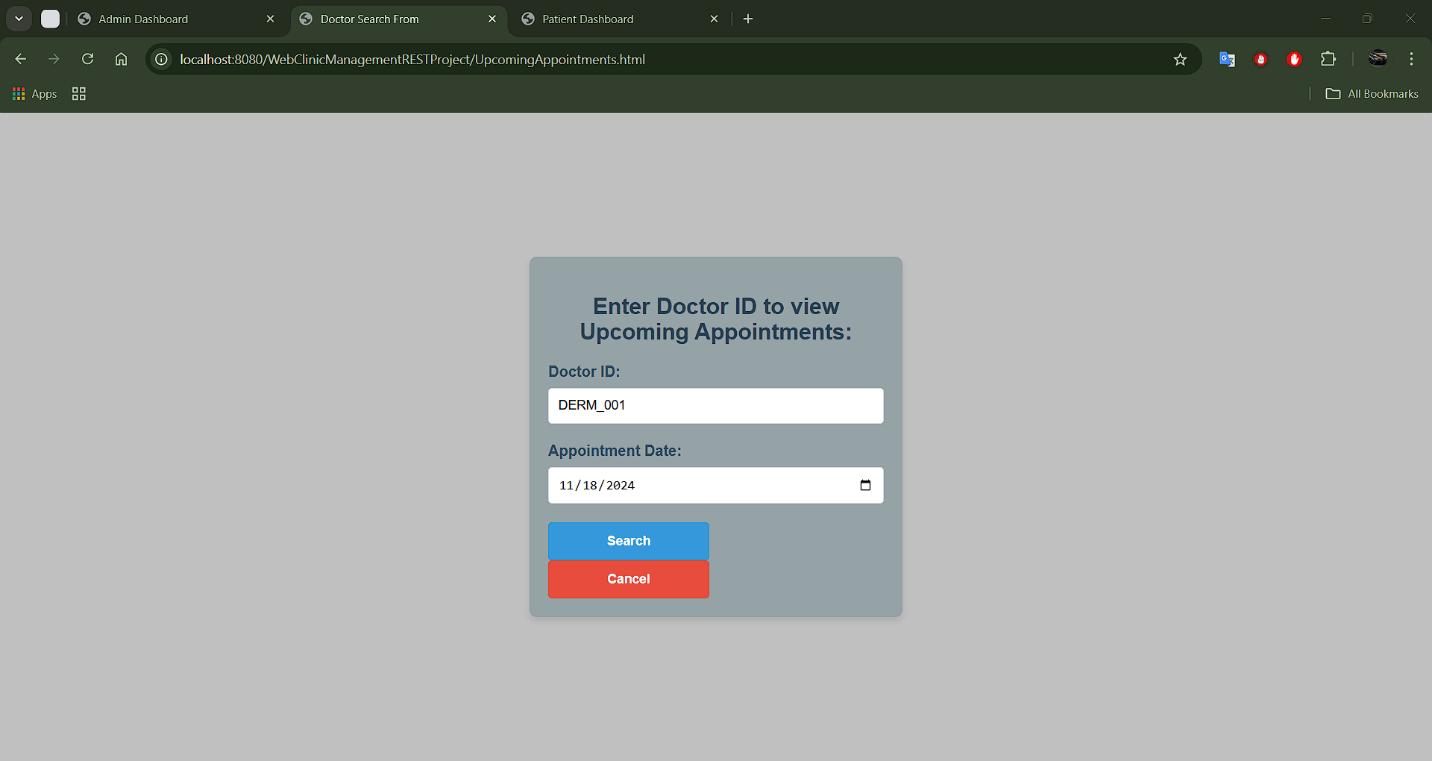
****

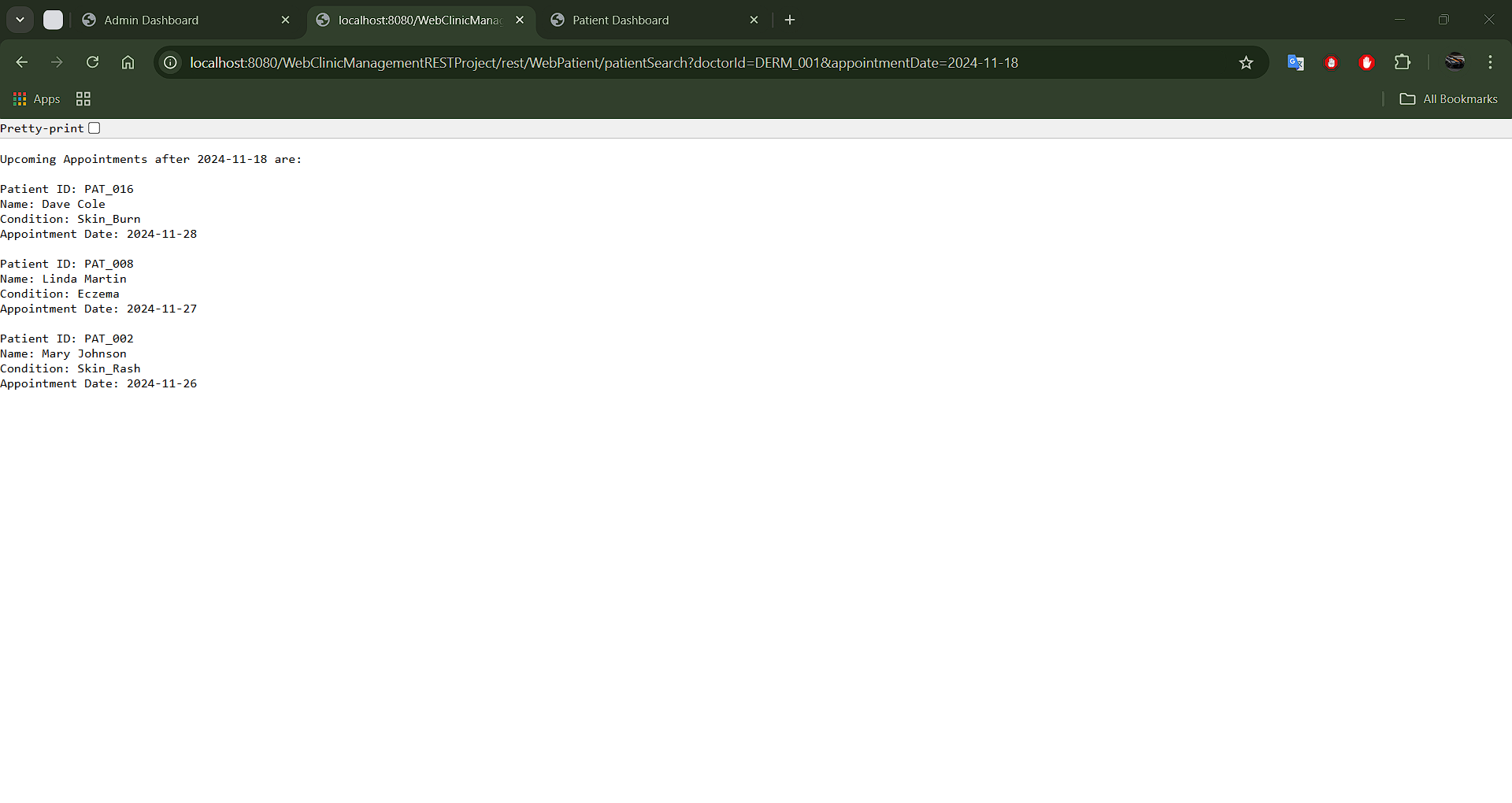
****

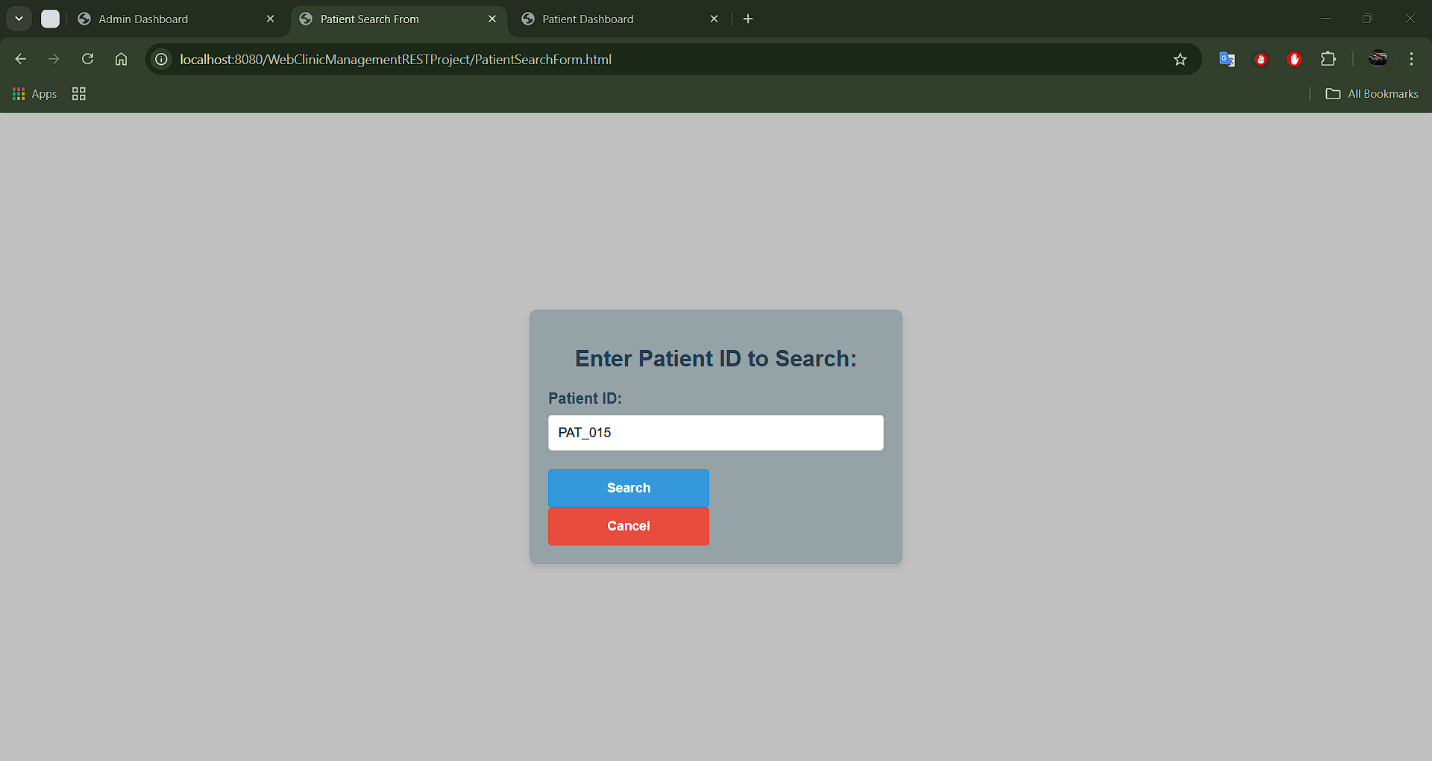
****

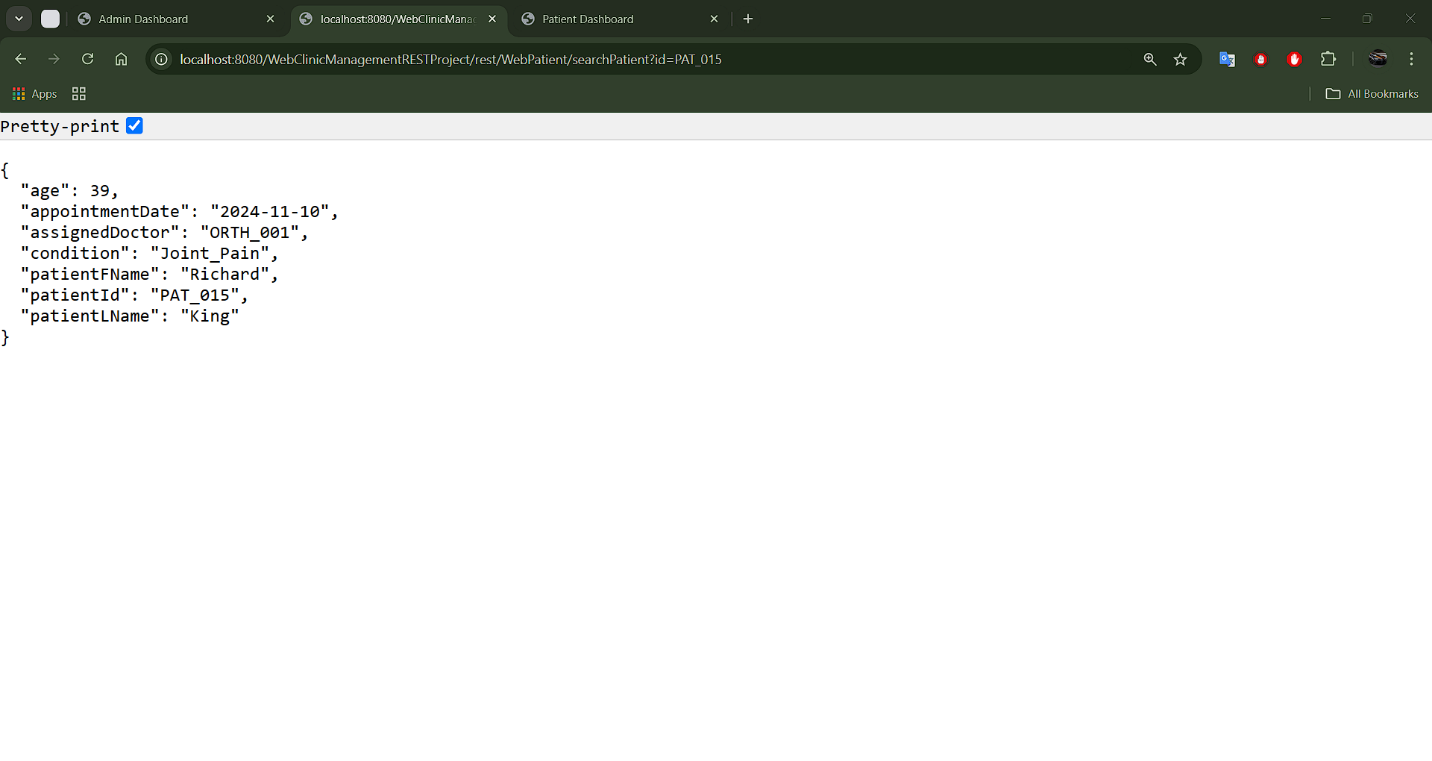
****

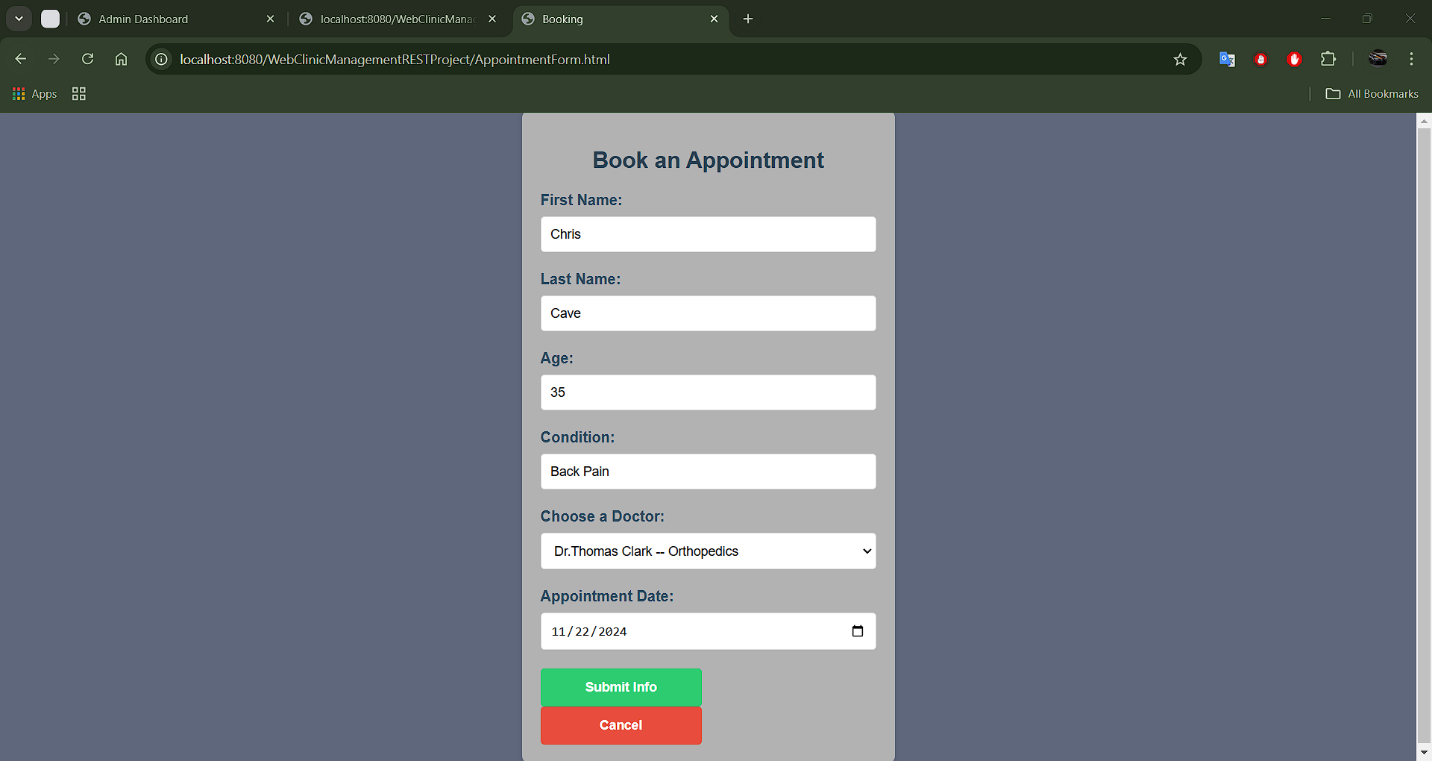
****

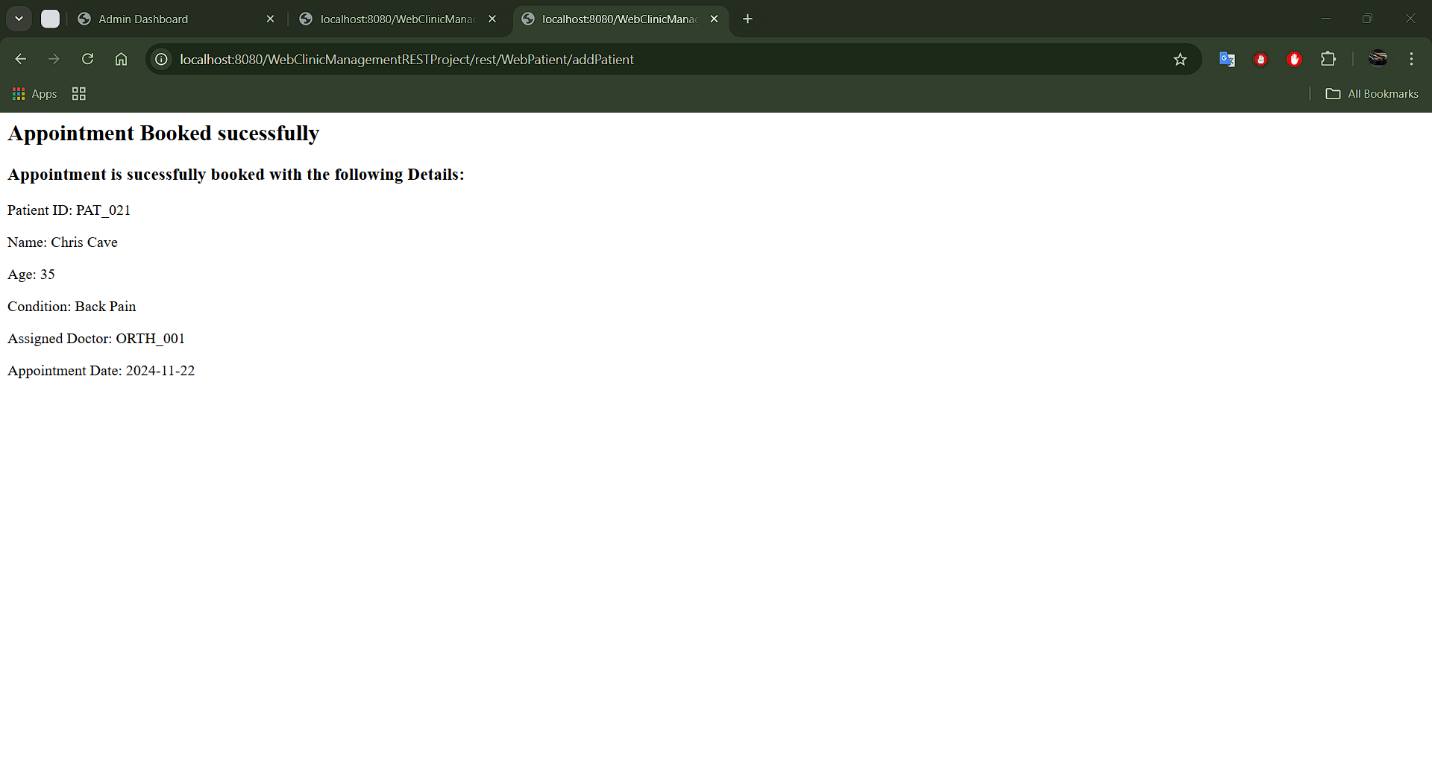
****

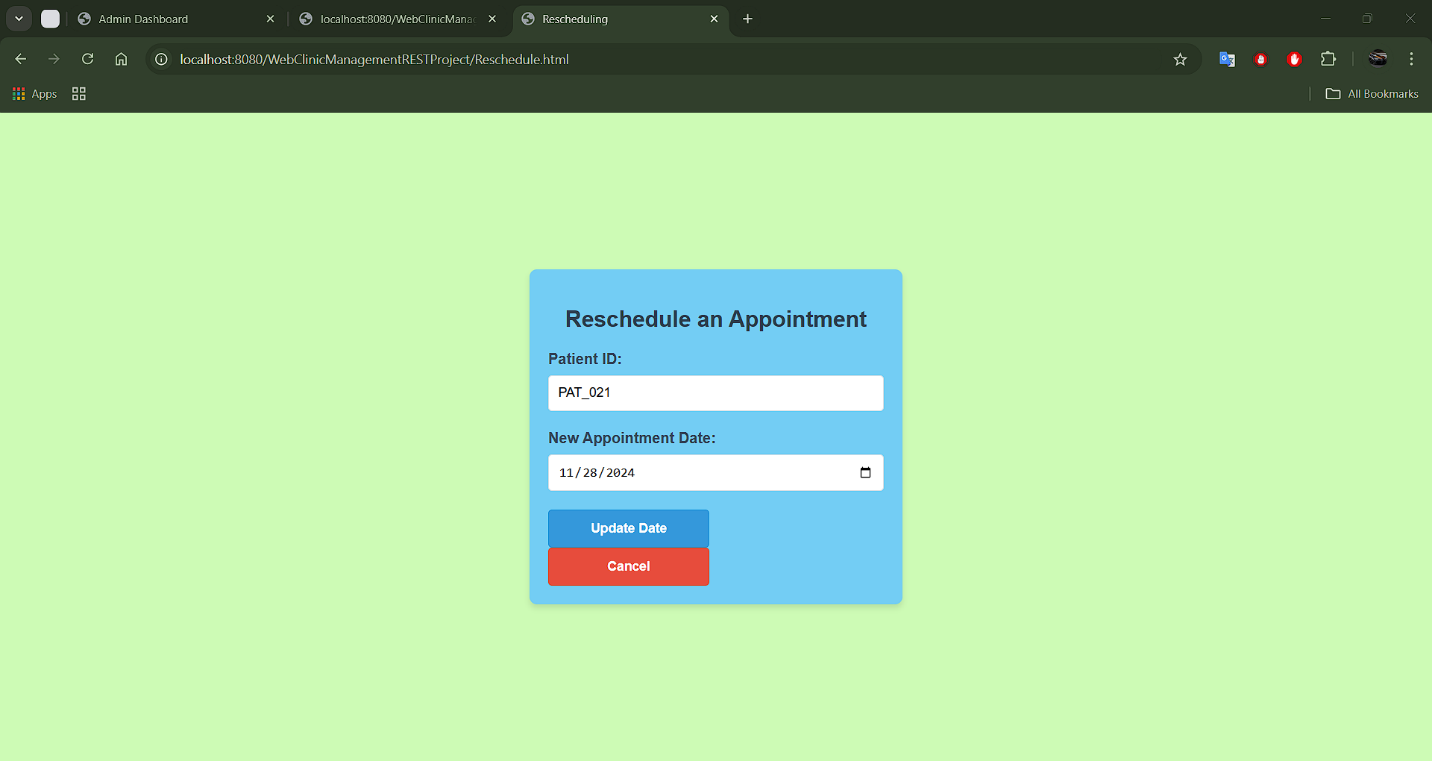
****

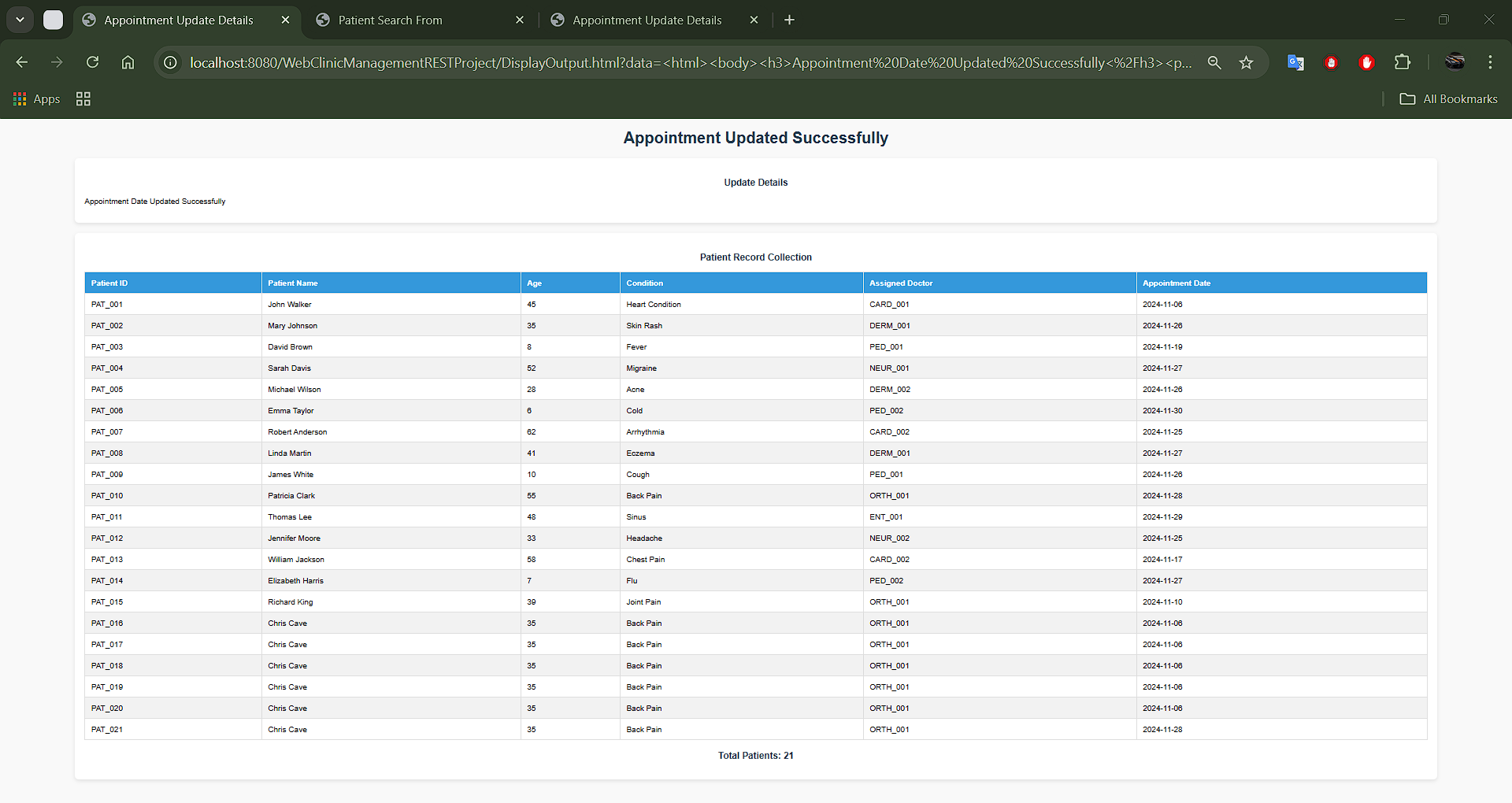
****

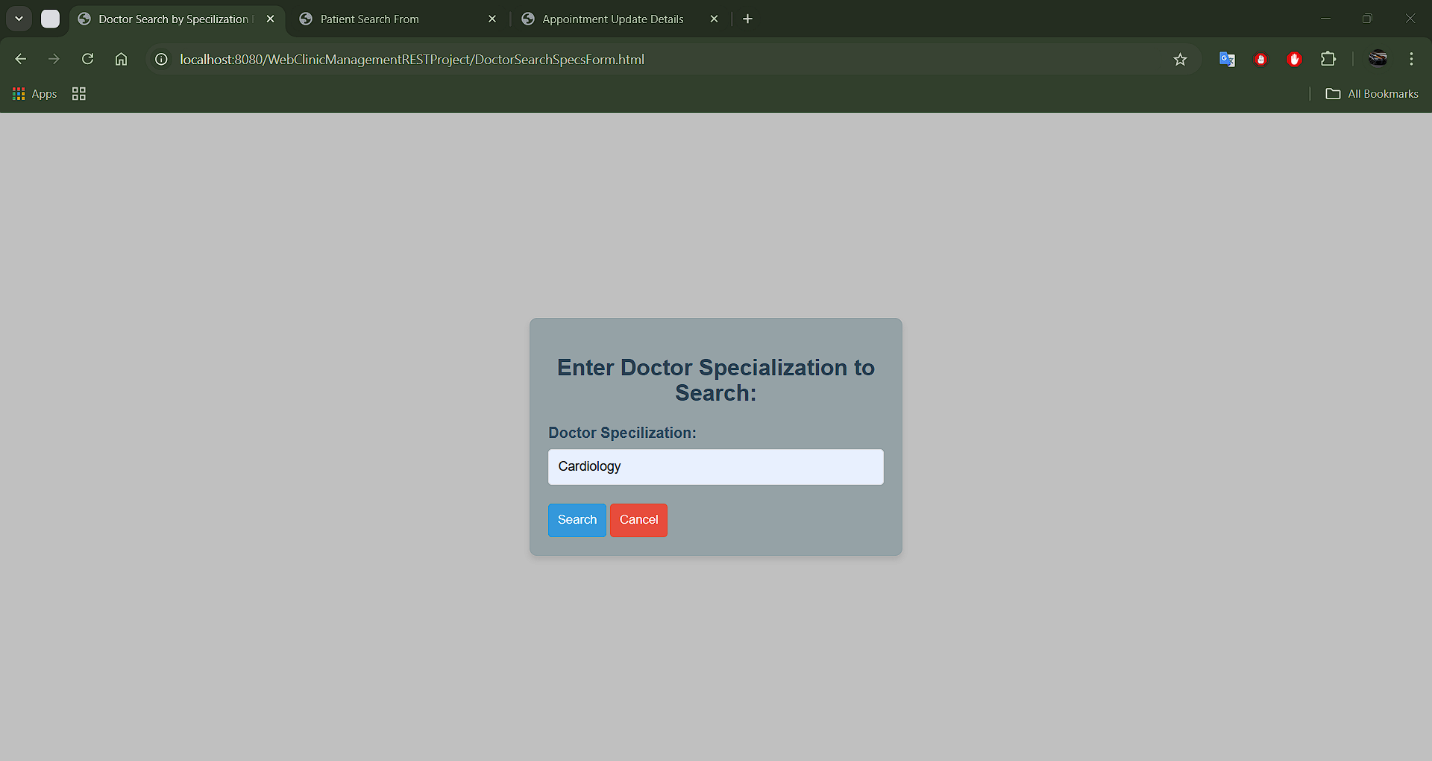
****

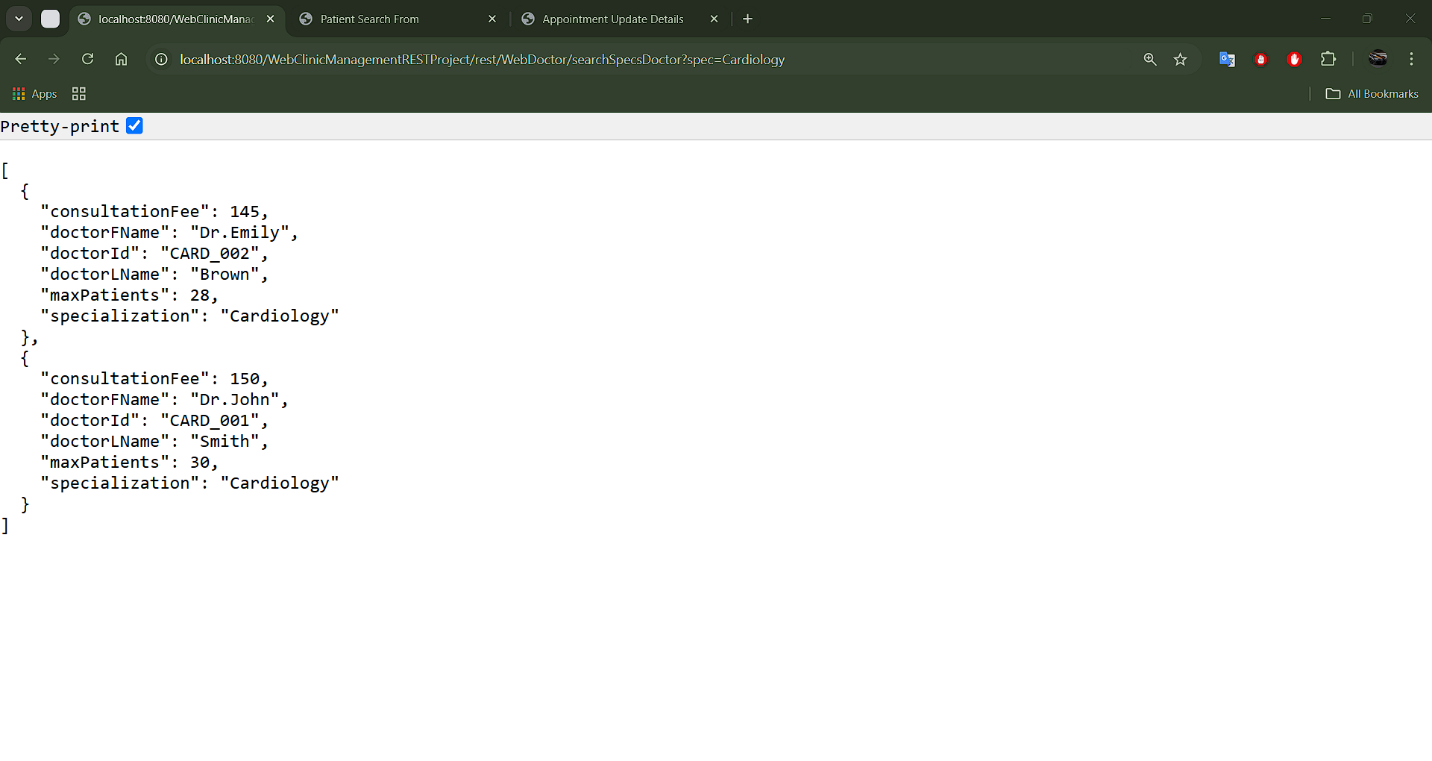
****

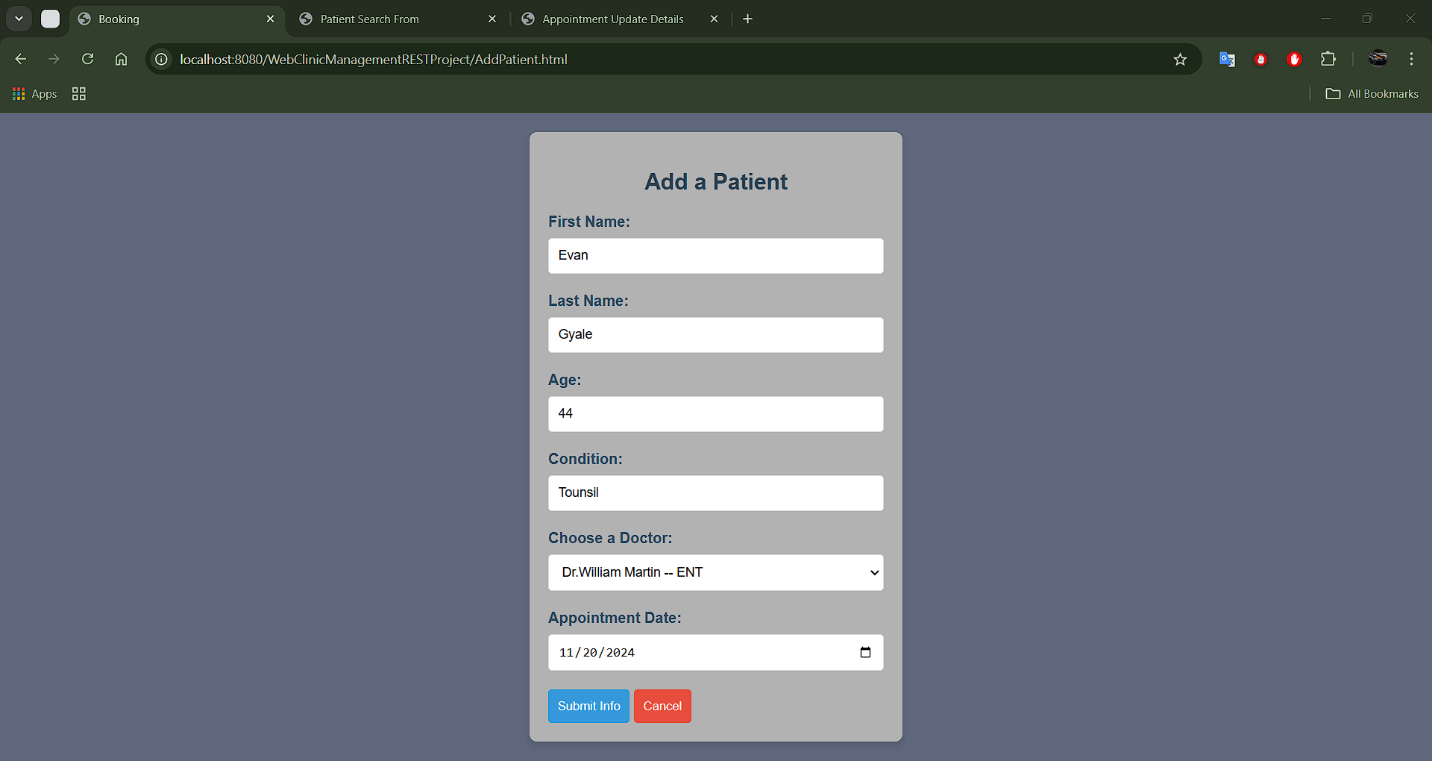
****

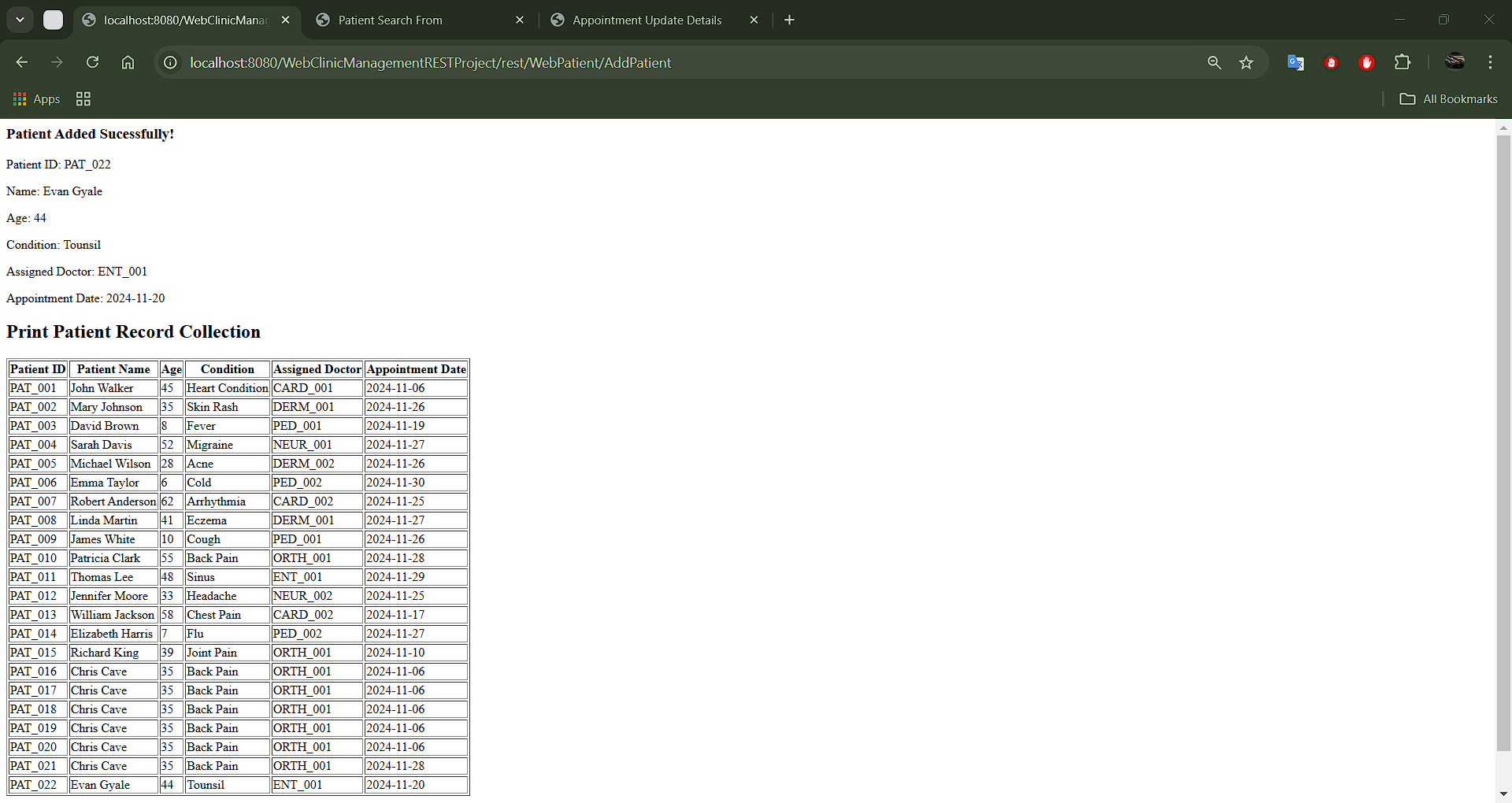
****

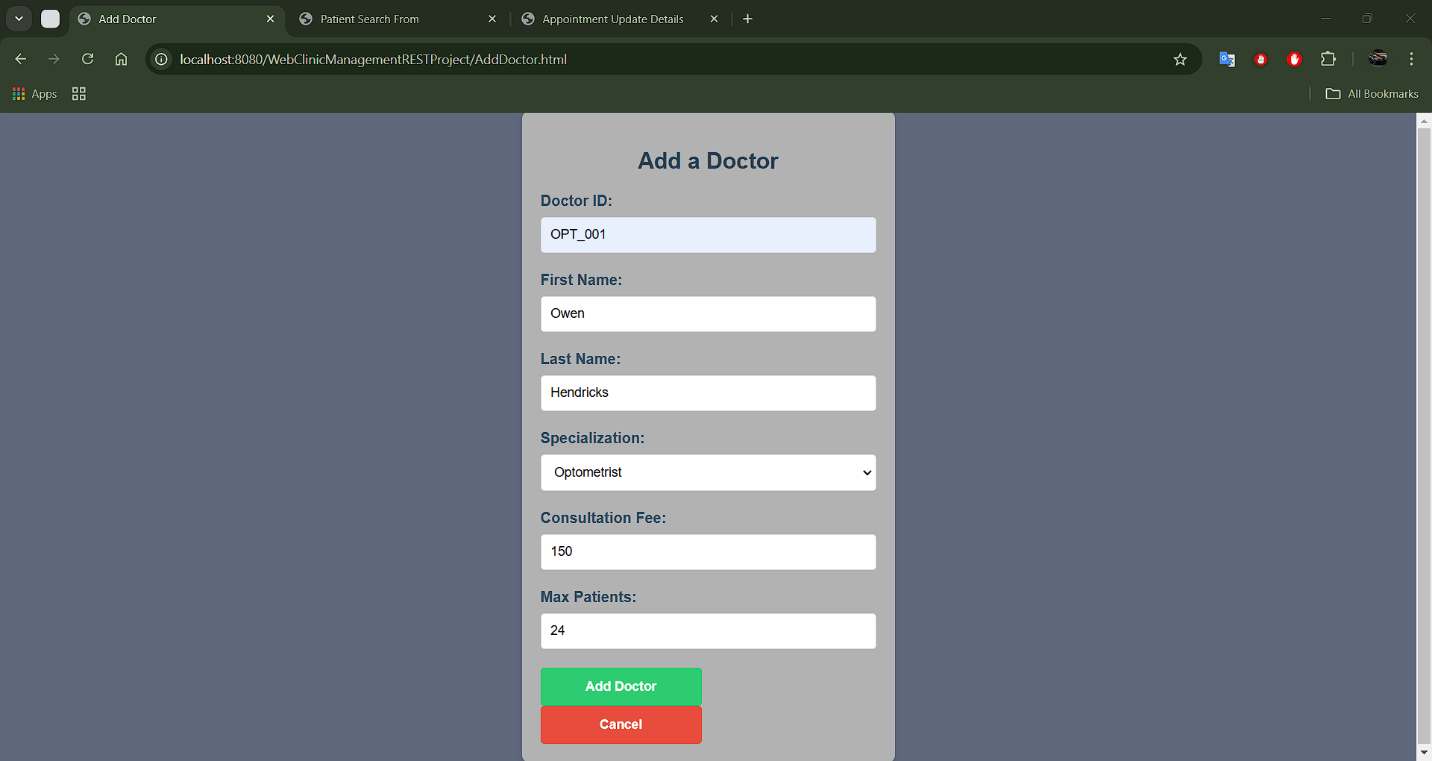
****

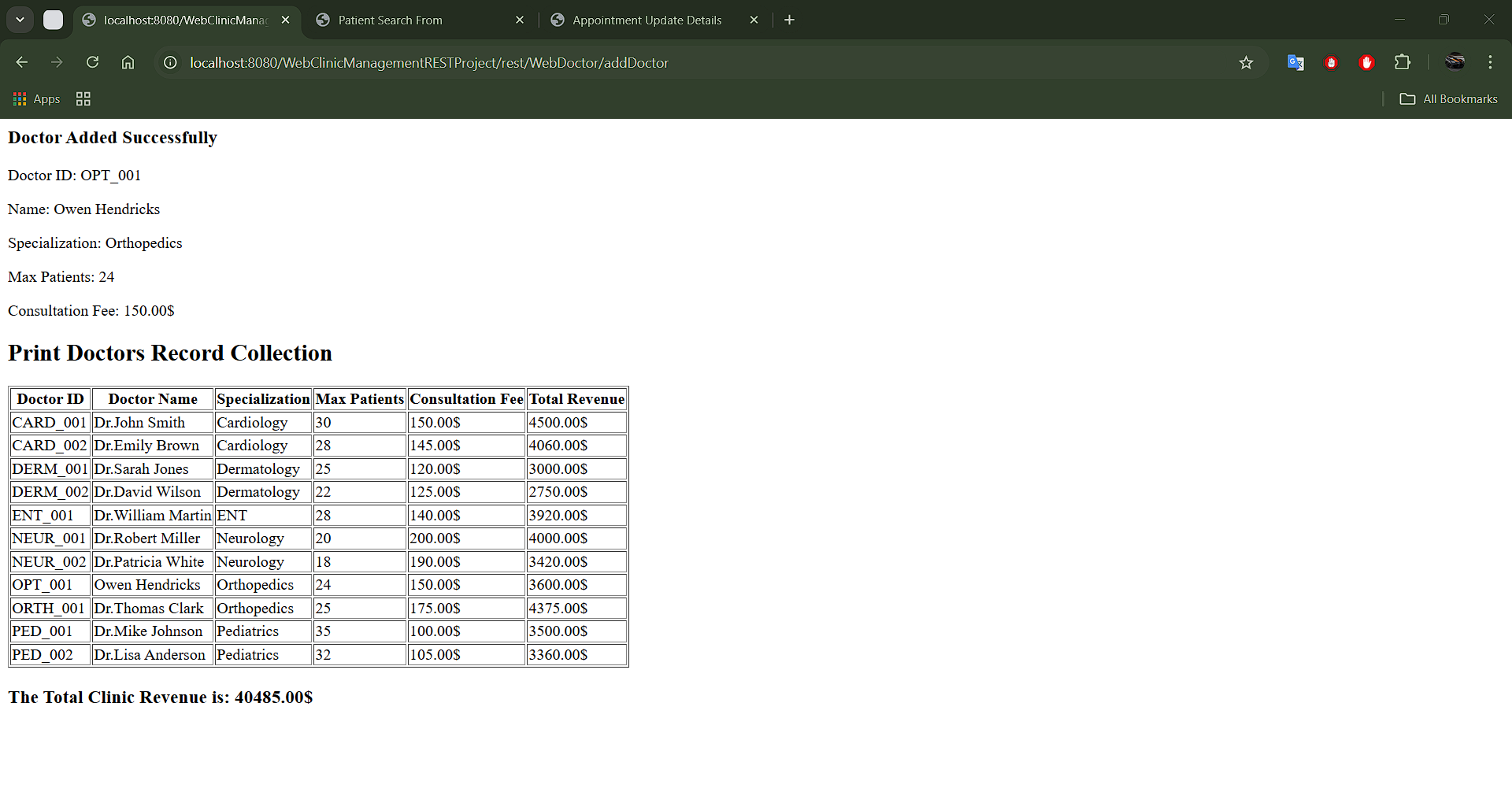
****

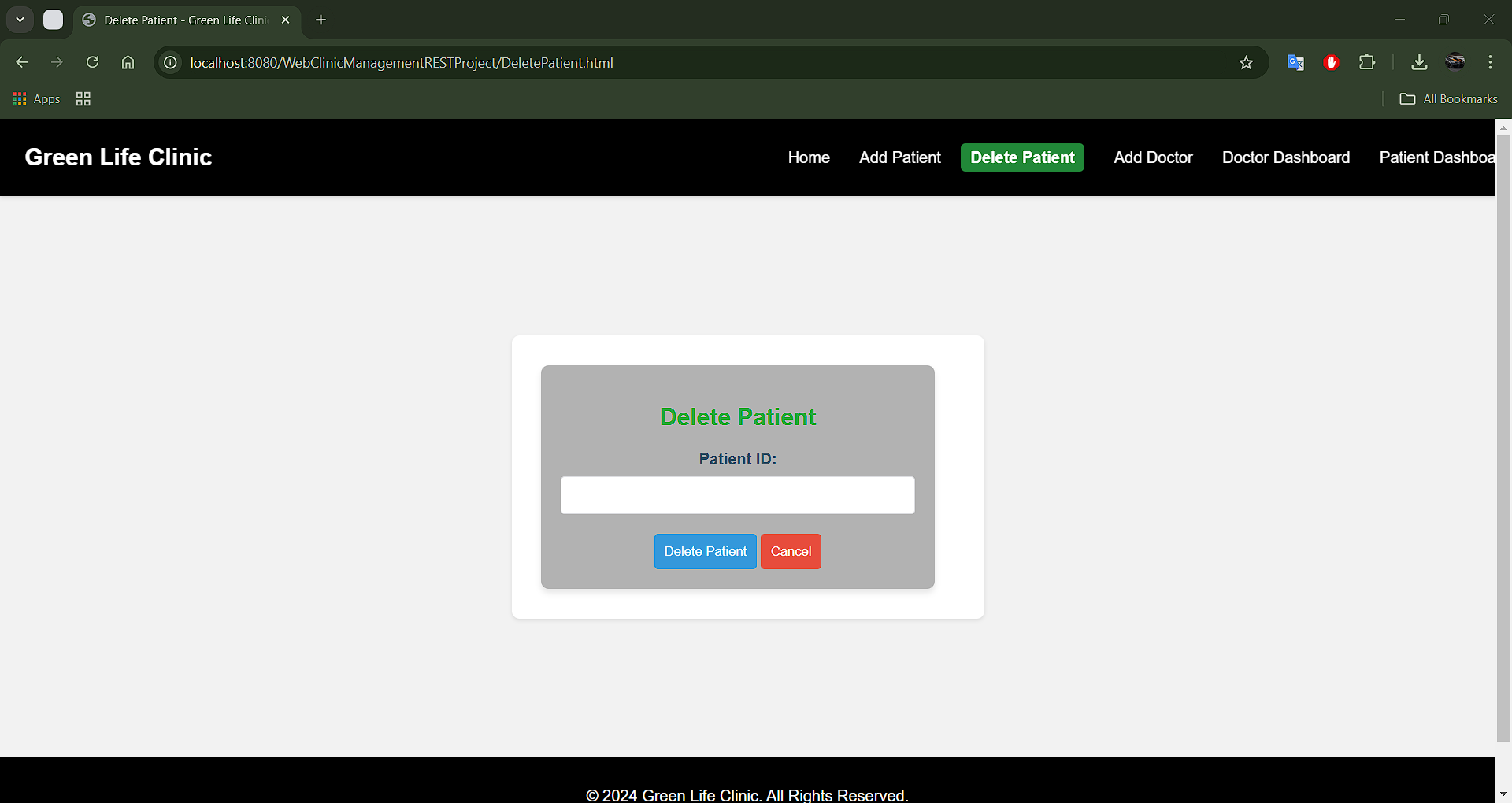
****

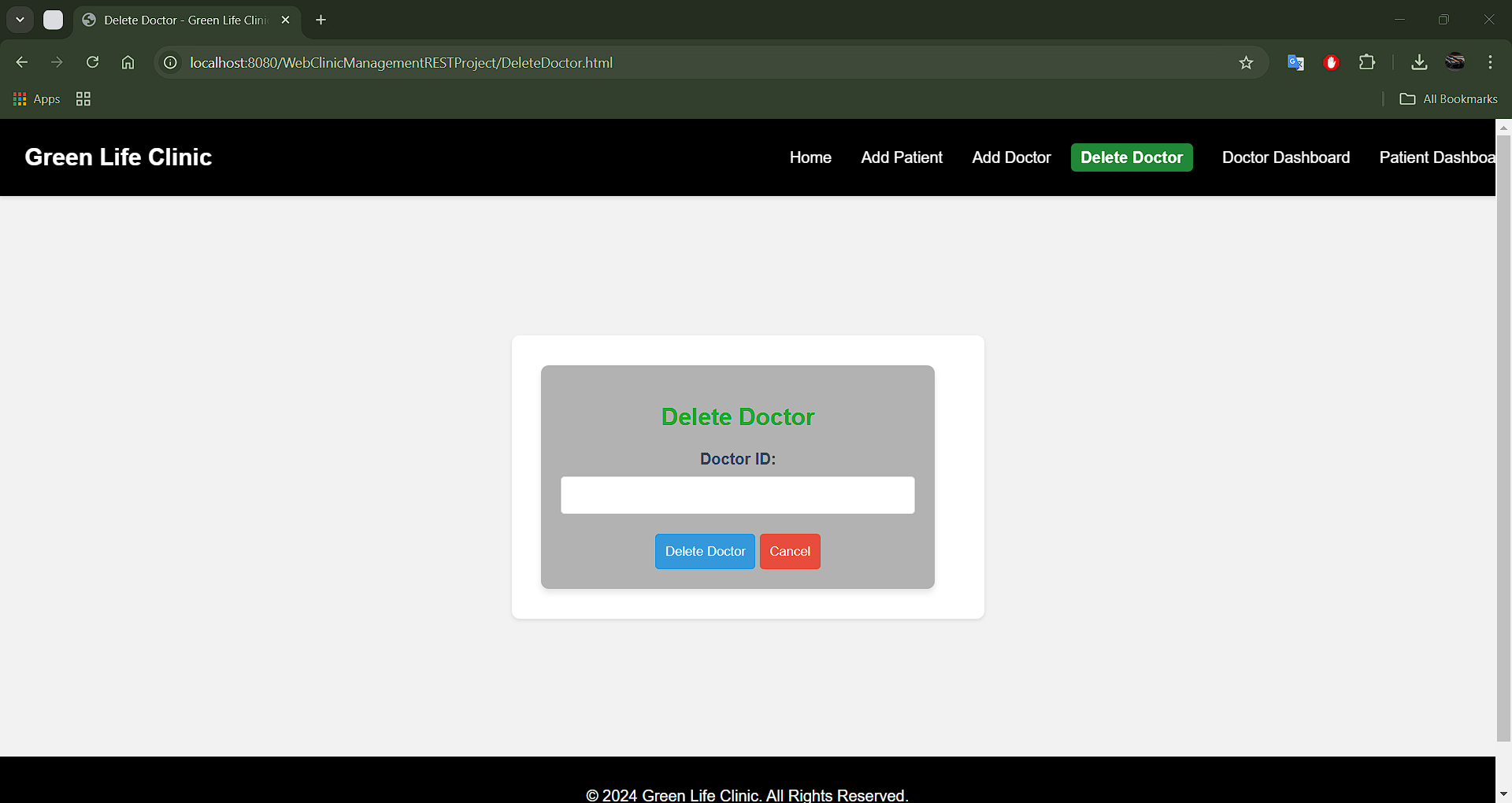
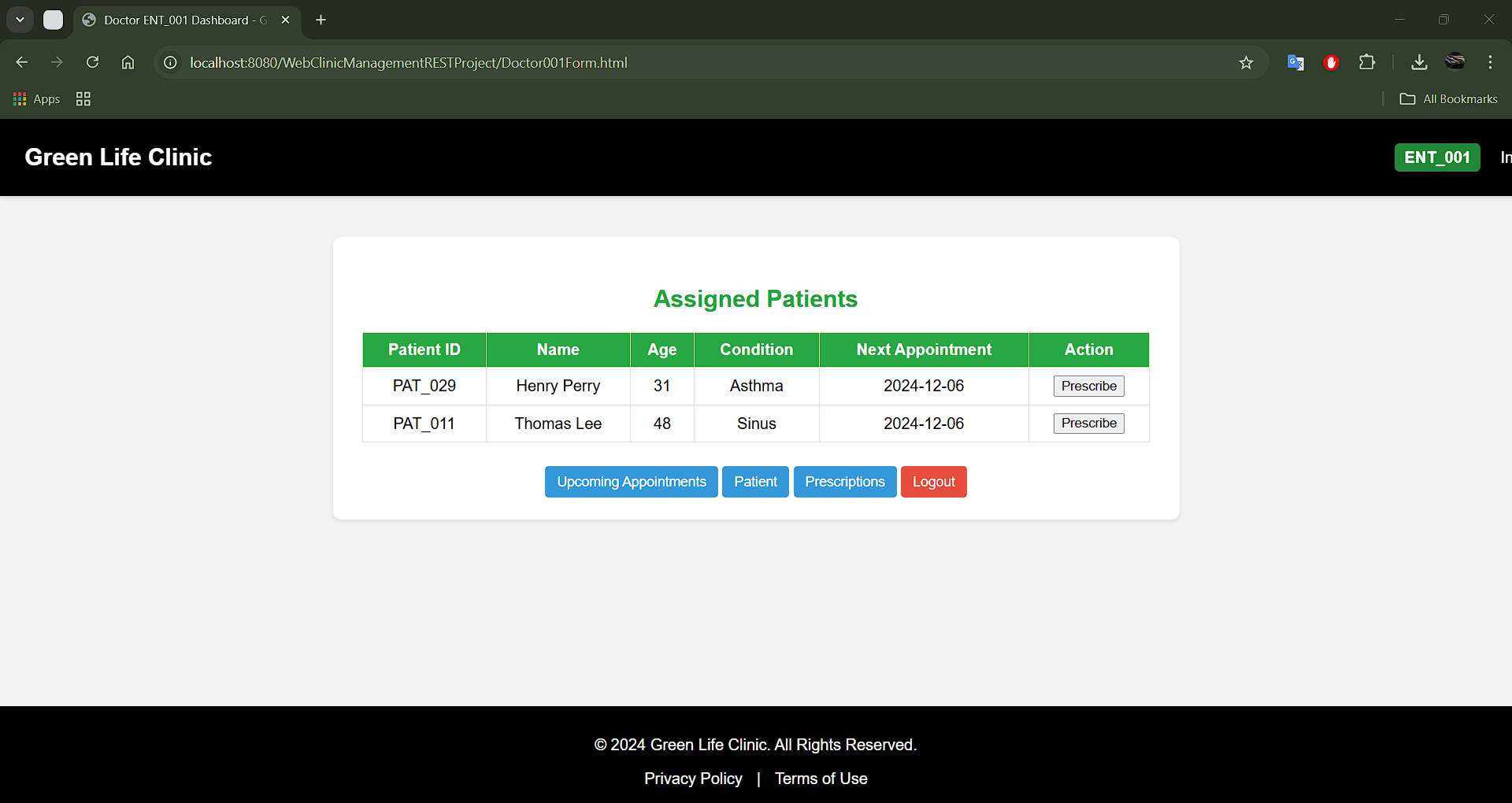
****

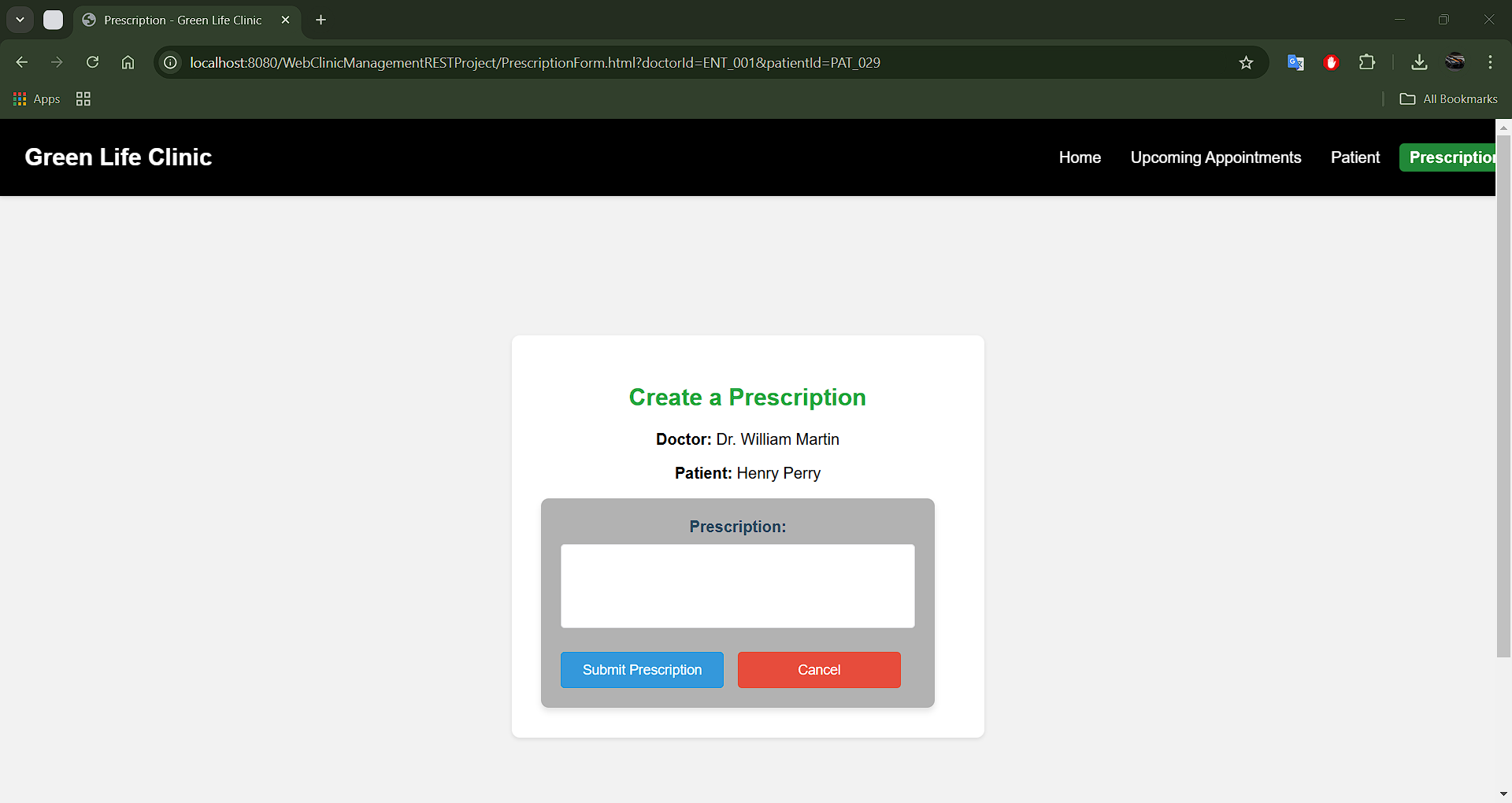
****

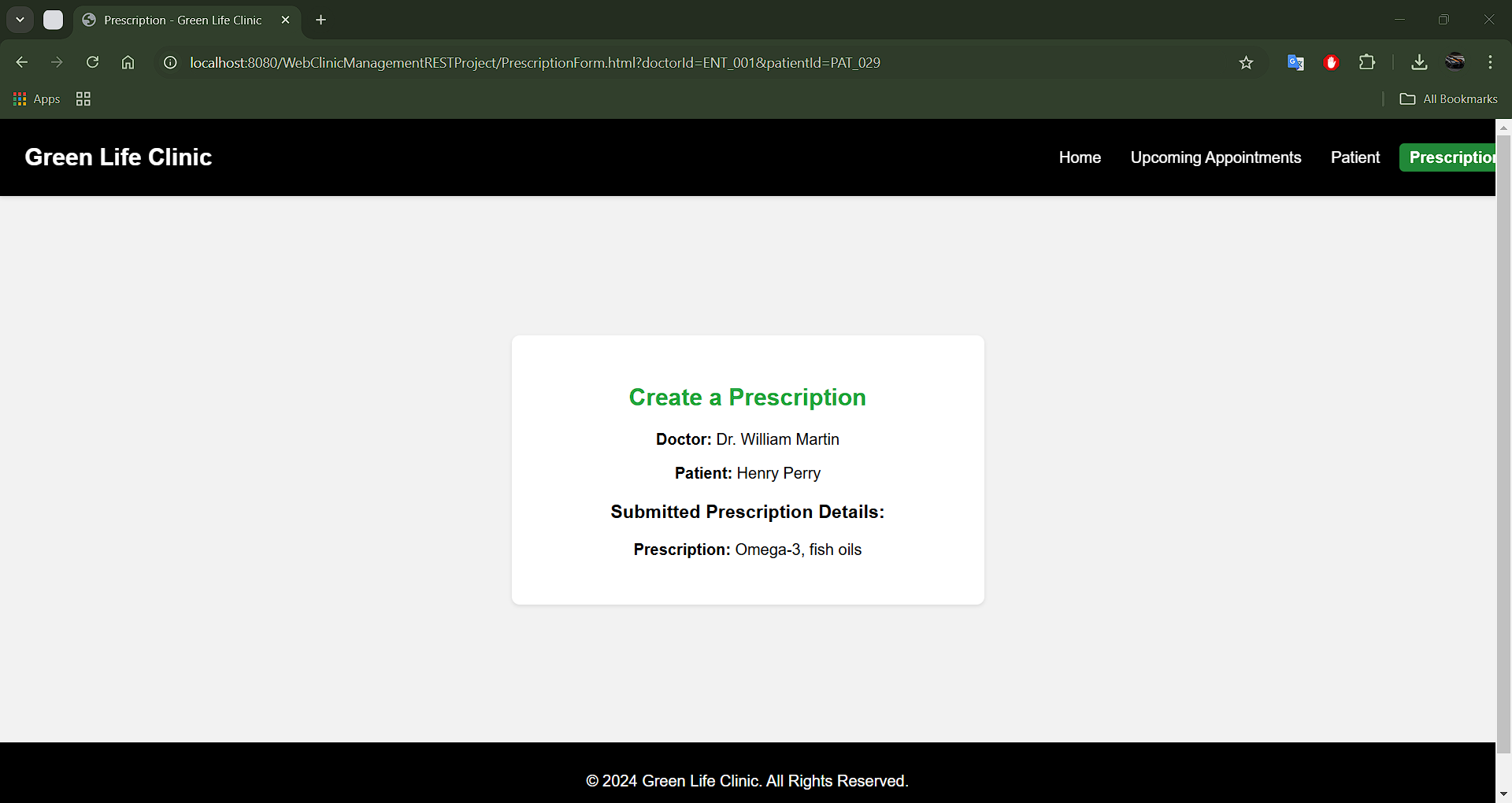
****

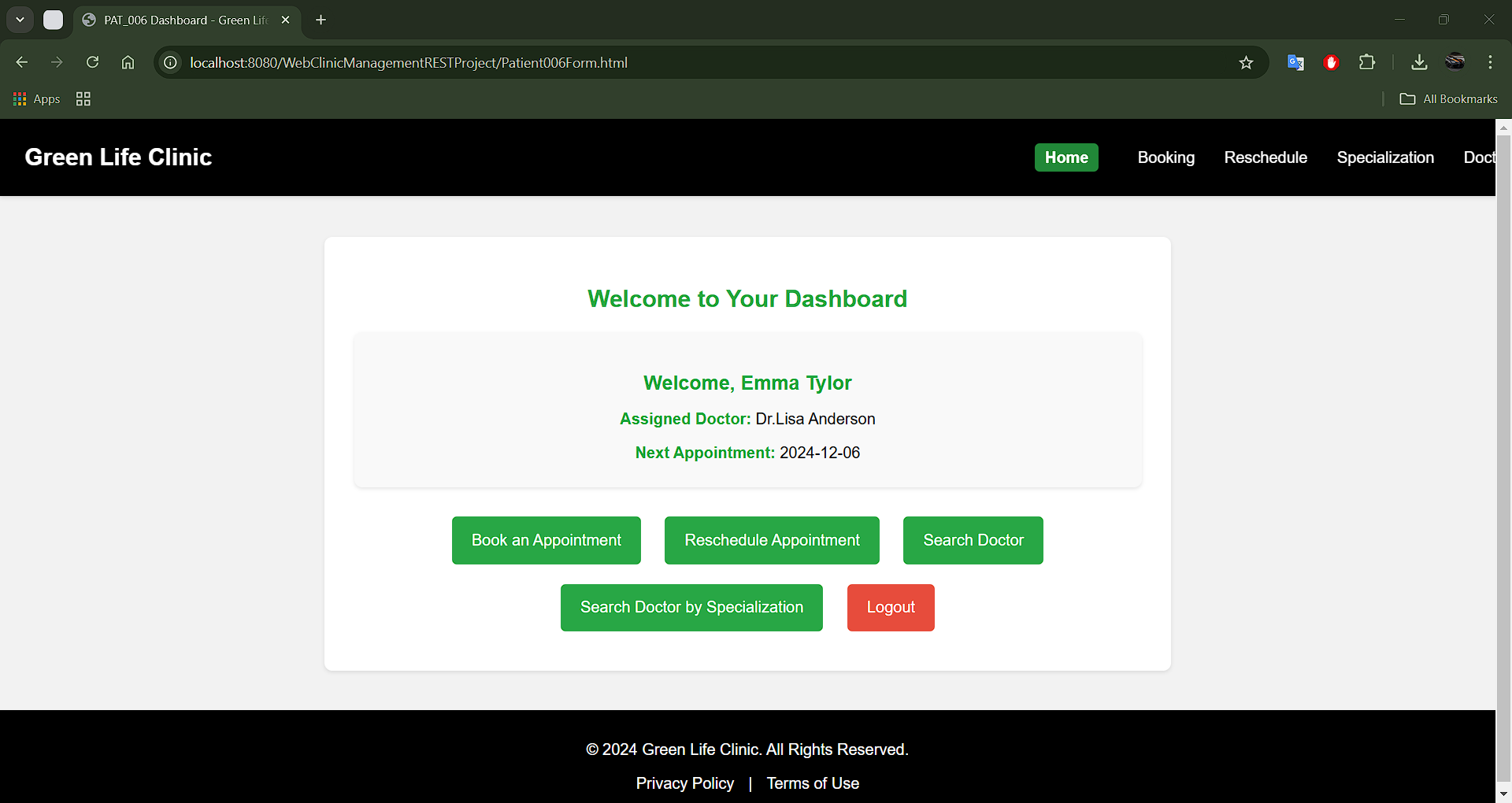
****

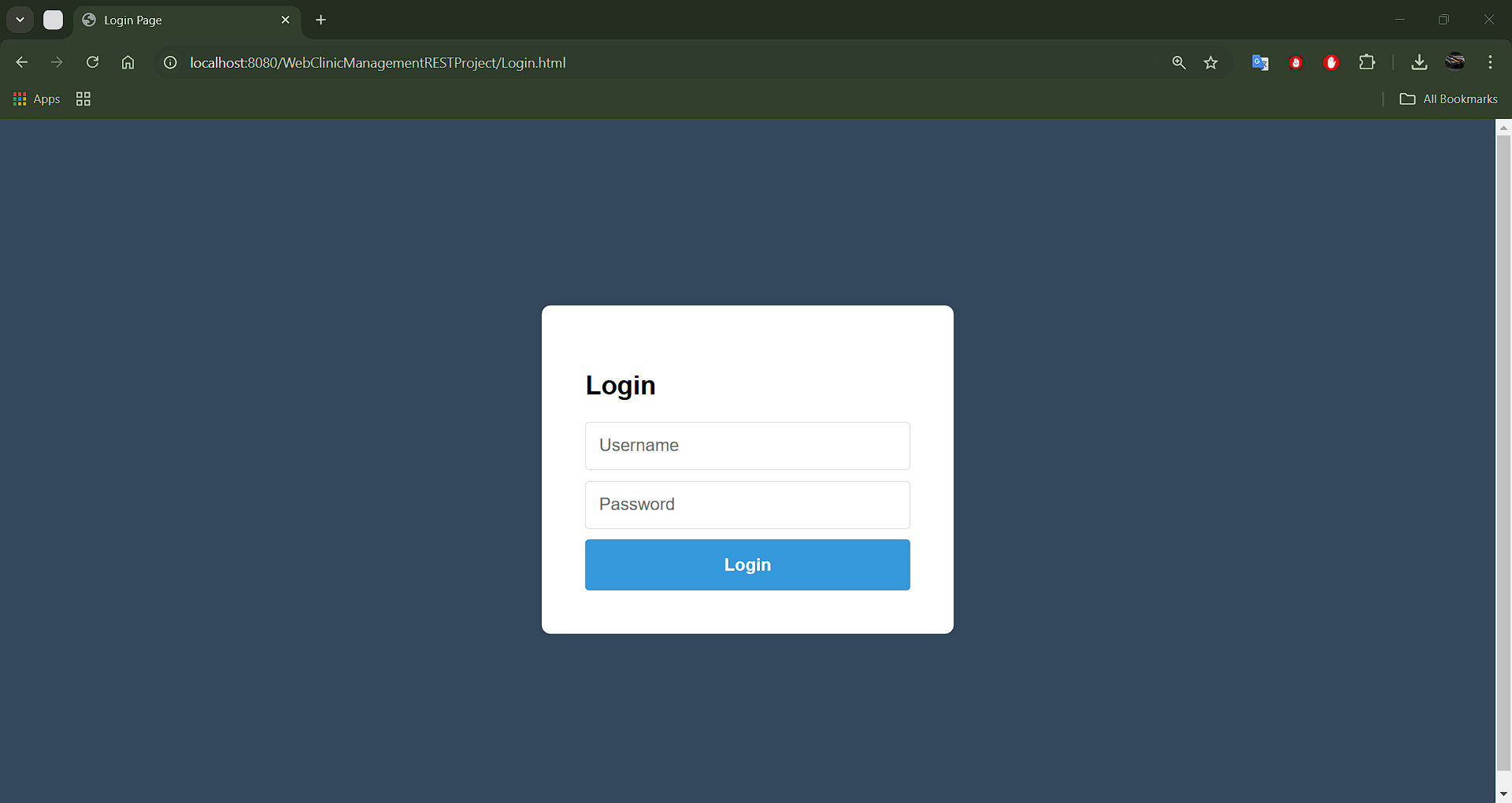
****

**** ****

****

****

****

****