Hospital Management System

Project Backend Report

Database and Postman Results

Mandepudi Gopi Chakradhar

Abstract

This report provides a detailed overview of the backend development and testing of the hospital management system. Designed for streamlined hospital operations, the system incorporates features such as secure authentication, robust role management, efficient appointment scheduling, and comprehensive medical record handling. The APIs were rigorously tested using Postman, with results validated through screenshots and explanations. This document underscores the backend's functional capabilities, ensuring it meets operational and security requirements.

Overview

An RBAC-powered system for managing a hospital's operations.
The is a centralized system designed to streamline hospital operations, enhance patient care, and ensure secure access to medical and administrative information.

1 Introduction

The hospital management system's backend is engineered to handle complex hospital workflows, ensuring efficiency, security, and accessibility. Core functionalities include secure user authentication, dynamic role-based access control, and efficient medical record management. This report details the testing and validation of these features using Postman. Screenshots and thorough descriptions are provided for each tested feature, offering a comprehensive understanding of the system's reliability and robustness.

2 System Features

2.1 User Role Management

The system employs a well-structured role-based model to delegate responsibilities and secure data access effectively. The primary user roles and their functions are as follows:

- **Admin:** Maintains full control over the system. Responsible for managing user roles, permissions, and overseeing overall system operations.
- **Doctor:** Manages patient treatments and medical records, generates prescriptions, and utilizes AI-powered diagnostic tools for enhanced care delivery.

- **Nurse:** Monitors and updates patient vitals, assists doctors during procedures, and ensures timely updates to patient records.
- **Receptionist:** Oversees appointment scheduling, patient check-ins, and the management of daily operational logistics.
- **Patient:** Provides patients with secure access to their personal medical records, appointment scheduling, and communication with healthcare providers.

2.2 Modules and Functionality

The hospital backend provides several modules and functionality to streamline operations.

2.2.1 User Authentication and Authorization

- Secure login with role-specific dashboards.
- Multi-factor authentication for sensitive roles (e.g., Admin, Doctor).

2.2.2 Patient Management

- **Patient Registration:** Add or update patient profiles, including medical history.
- **Medical Records:** Store and access patient diagnoses, prescriptions, and treatment history.
- **Appointment Scheduling:** Book, update, or cancel appointments.
- **Billing System:** Generate and manage invoices for consultations, tests, and treatments.

2.3 Endpoints and Functionality

This section outlines the key endpoints and their respective functionalities within the hospital management system. These endpoints cover user and role management, appointment scheduling, emergency overrides, and medical record handling, ensuring a robust and secure system for hospital operations.

- **User Management:**
  - Create, read, update, and delete users.
  - Assign roles to users.
- **Role Management:**
  - Add new roles.
  - Map permissions to roles.
- **Permission Management:**
  - View existing permissions.
  - Assign specific permissions to roles.
- **Appointments:**

  - Schedule new appointments.
  - Update or cancel existing appointments.
  - View appointments by patient or doctor.

- **Emergency Overrides:**
  - Allow admins to override access for critical conditions.
- **Medical Records:**
  - Securely store and retrieve patient records.

Additionally, each endpoint was rigorously tested using **Postman**. Screenshots documenting successful responses for all operations are included, ensuring thorough validation and reliability.

2.4 Postman Testing

Each endpoint was rigorously tested using Postman to validate functionality. The following steps were performed for every endpoint:

- **Create Requests:** All endpoints such as user management, role management, appointment scheduling, and emergency overrides were tested by sending requests and verifying the responses.
- **Verify Responses:** The responses were captured and analyzed to ensure that the system operated as expected across all functional areas.
- **Screenshots of Successful Responses:** Screenshots document successful responses for each tested operation, ensuring that each feature works as intended.

2.4.1 Doctor and Nurse Management

- View and update assigned patient records.
- Generate prescriptions and treatment plans.
- View patient vitals updated by nurses in real-time.

3 Database Design

The RBAC system is built on a relational database schema designed to manage user roles, permissions, and access levels in a hospital management system.

Figure 1: Database created using SQL in PostgreSQL via pgAdmin and tested using psql.

3.1 Database Tables

3.1.1 users

Purpose: Stores details of all users, including their names, emails, hashed passwords, and assigned roles. **Key Fields:**

- **user_id:** Primary key.
- **name, email, password:** User details.
- **role_id:** Foreign key linking to the *roles* table.

3.1.2 roles

Purpose: Stores different roles in the system (e.g., Admin, Doctor, Nurse, Patient, Receptionist). **Key Fields:**

- **role_id:** Primary key.
- **role_name:** Name of the role.

3.1.3 permissions

Purpose: Defines specific permissions associated with different actions (e.g., viewing patients, editing appointments). **Key Fields:**

- **permission_id:** Primary key.
- **permission_name:** Name of the permission.

3.1.4 role_permission

Purpose: Establishes a many-to-many relationship between roles and permissions. **Key Fields:**

- **role_id:** Foreign key linking to the *roles* table.
- **permission_id:** Foreign key linking to the *permissions* table.

3.1.5 admins, doctors, nurses, receptionists, patients

Purpose: Separate tables to store additional details specific to each type of user (Admin, Doctor, Nurse, Receptionist, Patient).

3.1.6 appointments

Purpose: Manages appointment data between doctors and patients. **Key Fields:**

- **appointment_id:** Primary key.
- **patient_id, doctor_id:** Foreign keys linking to the respective tables.
- **date, status:** Appointment details.

3.1.7 medical_records

Purpose: Stores patient diagnosis and treatment history. **Key Fields:**

- **record_id:** Primary key.
- **patient_id, doctor_id:** Foreign keys linking to the respective tables.

3.1.8 emergency_overrides

Purpose: Tracks cases where emergency access is granted to a user for critical situations. **Key Fields:**

- **override_id:** Primary key.
- **user_id, patient_id:** Foreign keys.
- **reason, timestamp:** Emergency details.

4 Authentication and Login

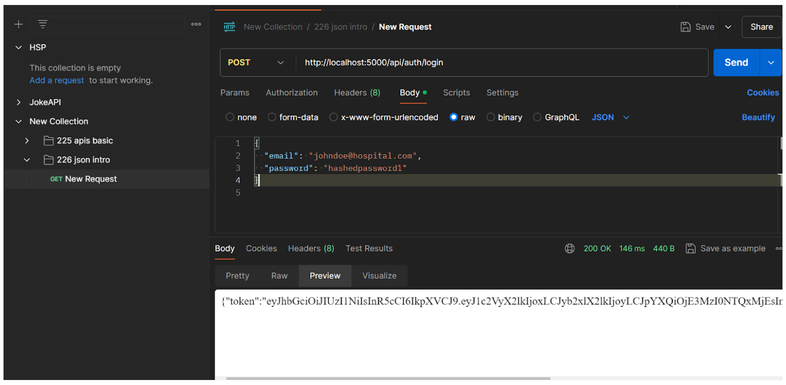
4.1 Login to Get JWT Token

To authenticate users, a POST request is made to the login endpoint with the credentials (email and password). This returns a JWT token that is used for further requests.

- URL: `http://localhost:5000/api/auth/login`
- Method: POST
- Body Content:

```
{
    "email": "doctor@example.com",
    "password": "yourPassword"
}
```

Postman Results: Check 2 for how the data is entered and 3 for the results in pretty mode.



New Collection / 226 json intro / New Request

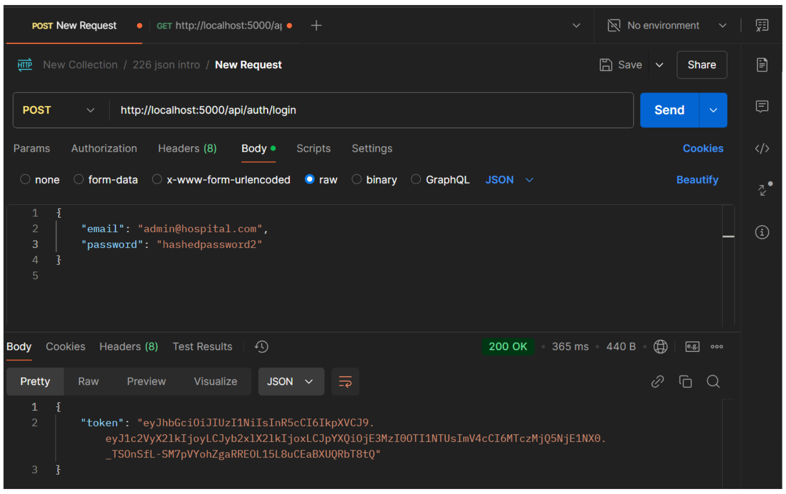
POST http://localhost:5000/api/auth/login

```
{
  "email": "johndoe@hospital.com",
  "password": "hashedpassword1"
}
```

200 OK 146 ms 440 B

```
{"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoxLCJyb2xlX2lkIjoyLCJpYXQiOjE3MzI0NTQxMjEsIr
```

Figure 2: Postman Login Request (Data Entry)



POST http://localhost:5000/api/auth/login

```
{
    "email": "admin@hospital.com",
    "password": "hashedpassword2"
}
```

200 OK • 365 ms • 440 B

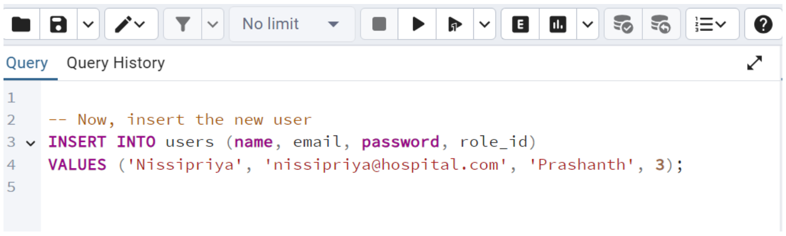
```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VyX2lkIjoyLCJyb2xlX2lkIjoxLCJpYXQiOjE3MzI0OTI1NTUsImV4cCI6MTczMjQ5NjE1NX0.
        _TSOnSfL-SM7pVYohZgaRREOL15L8uCEaBXUQRbT8tQ"
}
```

Figure 3: Postman Login Request (Pretty Mode Results)

5 Trigger Function and User Roles

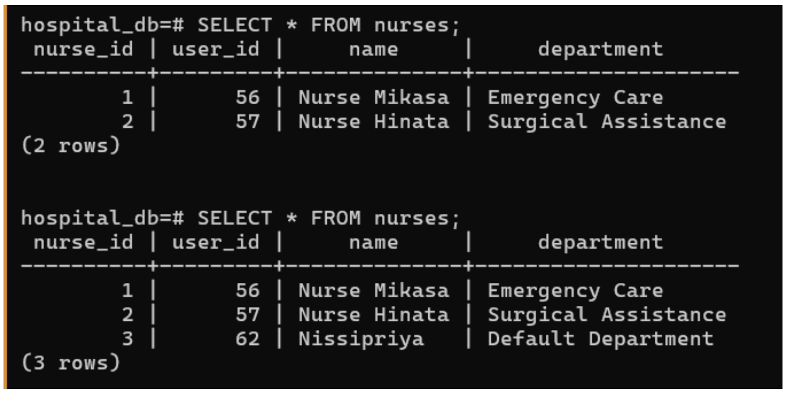
Once a new user is created, a trigger function ensures that the user is added to the appropriate role table.

- Code for Trigger Function: Check 4
- Results: Check 5



```sql
-- Now, insert the new user
INSERT INTO users (name, email, password, role_id)
VALUES ('Nissipriya', 'nissipriya@hospital.com', 'Prashanth', 3);
```

Figure 4: Trigger Function Code



```
hospital_db=# SELECT * FROM nurses;
 nurse_id | user_id |     name      |      department
----------+---------+---------------+----------------------
        1 |      56 | Nurse Mikasa | Emergency Care
        2 |      57 | Nurse Hinata | Surgical Assistance
(2 rows)


hospital_db=# SELECT * FROM nurses;
 nurse_id | user_id |     name      |      department
----------+---------+---------------+----------------------
        1 |      56 | Nurse Mikasa | Emergency Care
        2 |      57 | Nurse Hinata | Surgical Assistance
        3 |      62 | Nissipriya   | Default Department
(3 rows)
```

Figure 5: Trigger Function Results (Before and After Database Table Modifications)

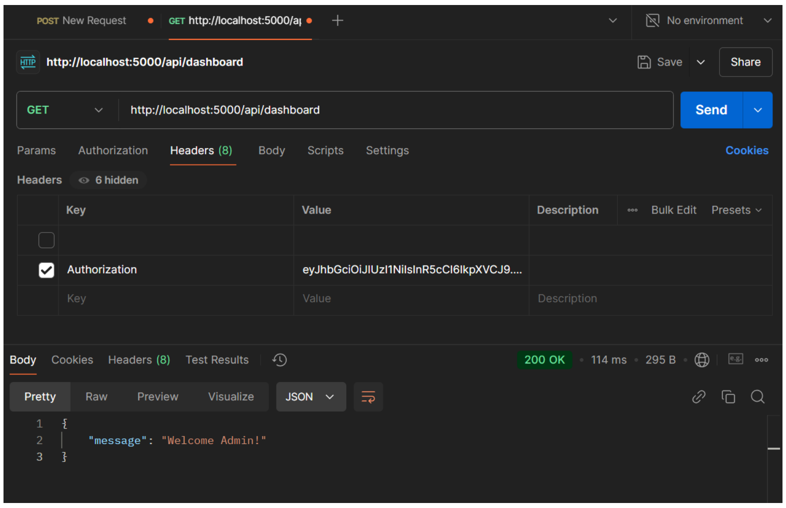
6 Database Table Modifications

After adding a new user, check the modifications in the database table, which is updated with the user's assigned role.

7 Dashboard Access Based on Role

The user is granted access to the dashboard based on their role. A GET request to the `/api/dashboard` endpoint returns a role-specific message.

- URL: `http://localhost:5000/api/dashboard`
- Method: GET
- Authorization Header: `Bearer your-jwt-token-here`



GET http://localhost:5000/api/dashboard

Headers

Key	Value	Description
✓ Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	

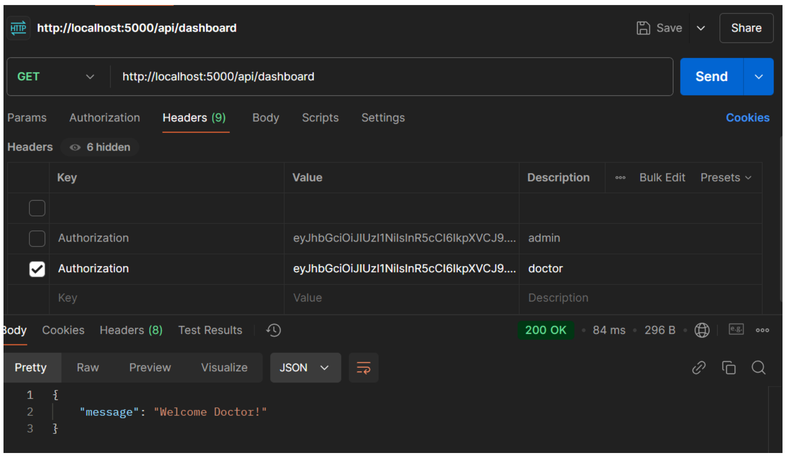
200 OK • 114 ms • 295 B

```
{
    "message": "Welcome Admin!"
}
```

Figure 6: Dashboard of role after using jwt token came from log in)

Postman Results: The response depends on the user's role. Refer to the following for role-based access:

- Admin: `"message": "Welcome Admin!"` (See 7)
- Doctor: `"message": "Welcome Doctor!"` (See 8)
- Nurse: `"message": "Welcome Nurse!"`
- Receptionist: `"message": "Welcome Receptionist!"`
- Patient: `"message": "Welcome Patient!"`



http://localhost:5000/api/dashboard

GET http://localhost:5000/api/dashboard

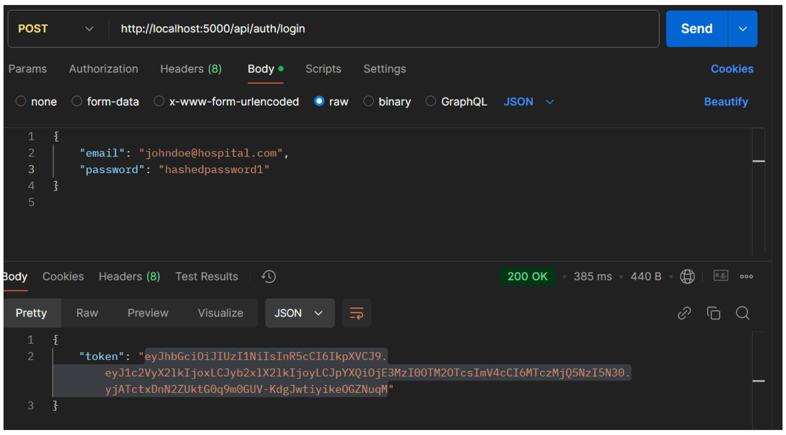
Headers (9) — 6 hidden

	Key	Value	Description
☐			
☐	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	admin
✓	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor

200 OK • 84 ms • 296 B

```
{
    "message": "Welcome Doctor!"
}
```

Figure 7: Postman Response for Admin Role



POST http://localhost:5000/api/auth/login

```
{
    "email": "johndoe@hospital.com",
    "password": "hashedpassword1"
}
```

200 OK • 385 ms • 440 B

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VyX2lkIjoxLCJyb2xlX2lkIjoyLCJpYXQiOjE3MzI0OTM2OTcsImV4cCI6MTczMjQ5NzI5N30.
        yjATctxDnN2ZUktG0q9m0GUV-KdgJwtiyikeOGZNuqM"
}
```

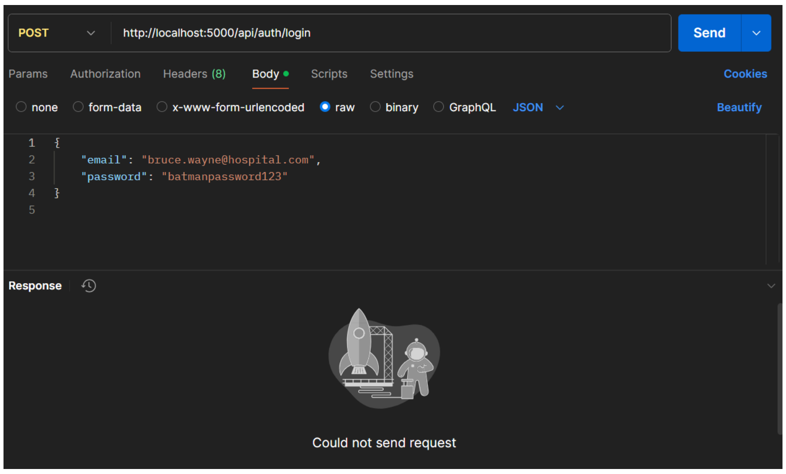
Figure 8: Postman Response for Doctor Role

8 Error Handling

In case of issues such as invalid tokens or missing roles, appropriate error messages are displayed. Example error messages:

- `error: "Invalid role"`
- `error: "Access denied"`
- `error: "Server Error"`

Postman Results: See 9 for error handling.



POST http://localhost:5000/api/auth/login

```
{
    "email": "bruce.wayne@hospital.com",
    "password": "batmanpassword123"
}
```

Response

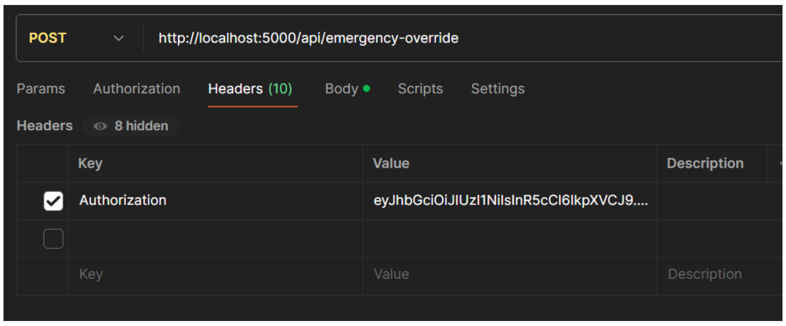
Could not send request

Figure 9: Postman Error Response

9 Emergency Override

A POST request to the `/api/emergency-override` endpoint triggers an emergency override action. The request body includes necessary data.

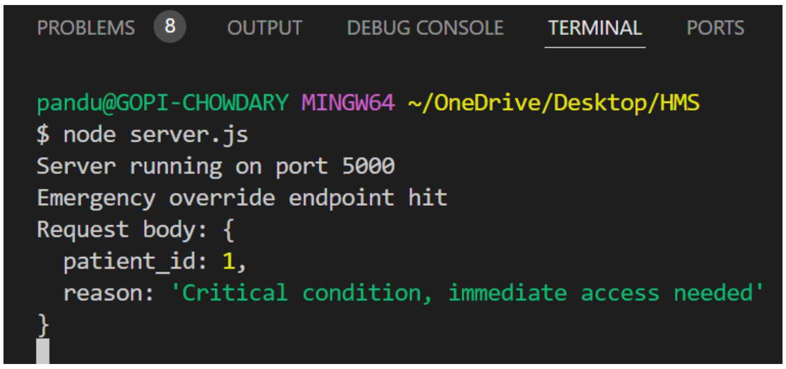
- URL: `http://localhost:5000/api/emergency-override`
- Method: POST
- Body Content: Refer to 10



POST http://localhost:5000/api/emergency-override

	Key	Value	Description
✓	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	
☐			
	Key	Value	Description

Figure 10: Emergency Override Request Body



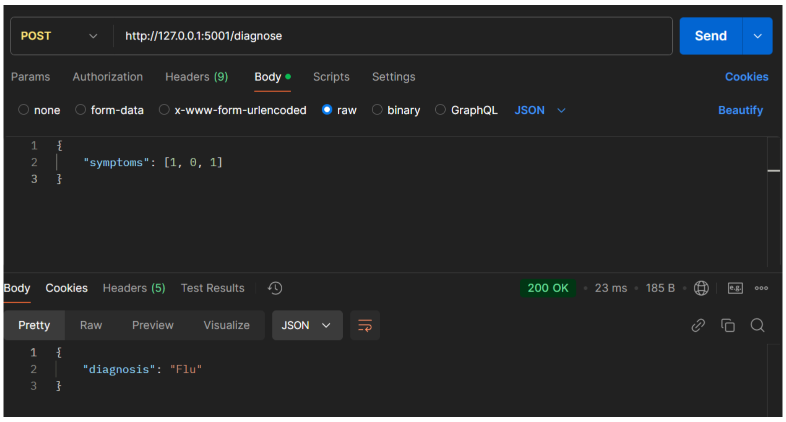
```
pandu@GOPI-CHOWDARY MINGW64 ~/OneDrive/Desktop/HMS
$ node server.js
Server running on port 5000
Emergency override endpoint hit
Request body: {
  patient_id: 1,
  reason: 'Critical condition, immediate access needed'
}
```

Figure 11: Emergency Override Request server side

10 Diagnosis

A POST request is made to the `/diagnose` endpoint to submit symptoms and receive diagnosis results.

- URL: `http://localhost:5000/diagnose`
- Method: POST
- Body Content: Refer to 12



POST http://127.0.0.1:5001/diagnose

```
{
    "symptoms": [1, 0, 1]
}
```

200 OK · 23 ms · 185 B

```
{
    "diagnosis": "Flu"
}
```

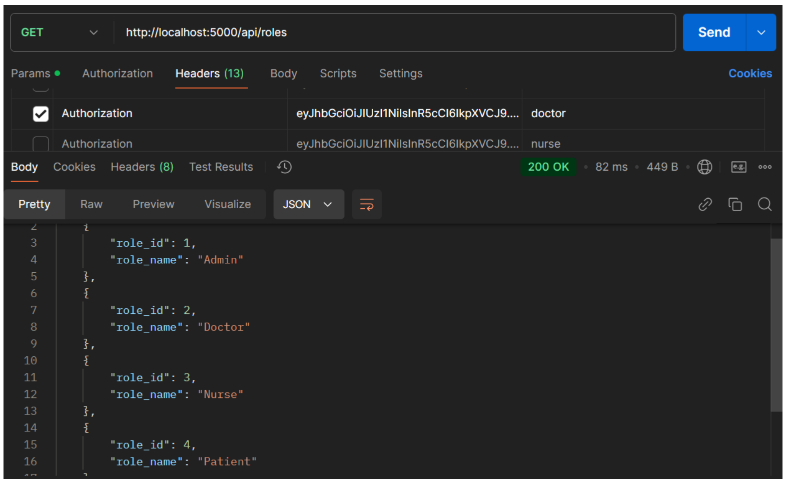
Figure 12: Diagnosis Request Body

11 Role Management

The Role Management API allows for fetching, creating, assigning permissions, and deleting roles.

- GET `/roles`: Fetch all roles (Refer 13)
- GET `/roles/:role`$_id/permissions : Fetchpermissions(Refer14)POST/$`roles` $: Createanewrole(Refer17)$
- POST `/roles/:role`$_id/permissions : Assignpermissionstoarole(Refer18)DELETE/$`roles/:role_id` $: Deletearole(Refer19)$

- DELETE `/roles/:role`$_id/permissions/ : permission_id : Deletearolepermission(Refer20)$



GET http://localhost:5000/api/roles

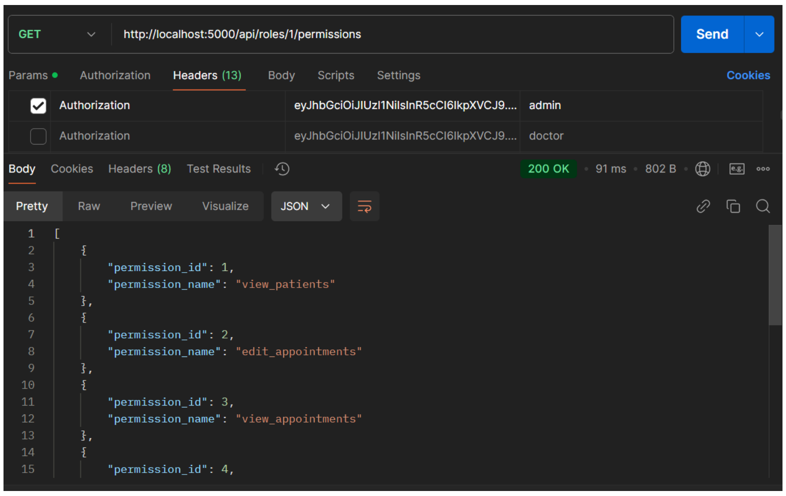
Headers (13)

	Key	Value	Description
✓	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor
	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	nurse

200 OK · 82 ms · 449 B

```
    {
        "role_id": 1,
        "role_name": "Admin"
    },
    {
        "role_id": 2,
        "role_name": "Doctor"
    },
    {
        "role_id": 3,
        "role_name": "Nurse"
    },
    {
        "role_id": 4,
        "role_name": "Patient"
```

Figure 13: Fetching All Roles

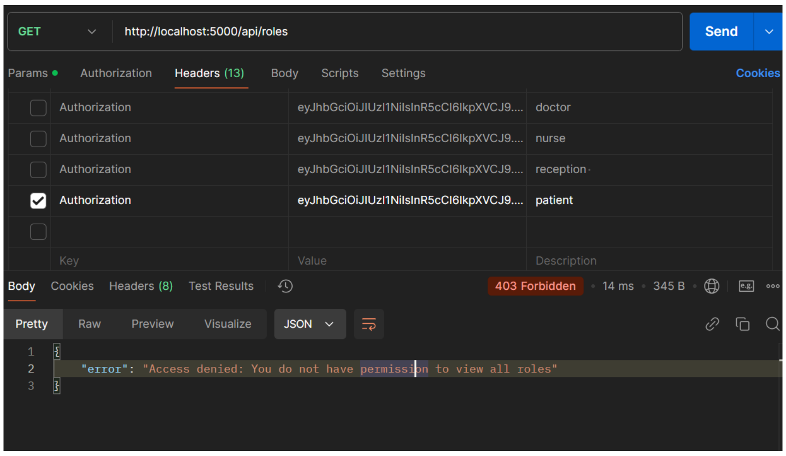


GET http://localhost:5000/api/roles/1/permissions

✓ Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	admin
Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor

```
[
    {
        "permission_id": 1,
        "permission_name": "view_patients"
    },
    {
        "permission_id": 2,
        "permission_name": "edit_appointments"
    },
    {
        "permission_id": 3,
        "permission_name": "view_appointments"
    },
    {
        "permission_id": 4,
```

Figure 14: Fetching All Roles permissions

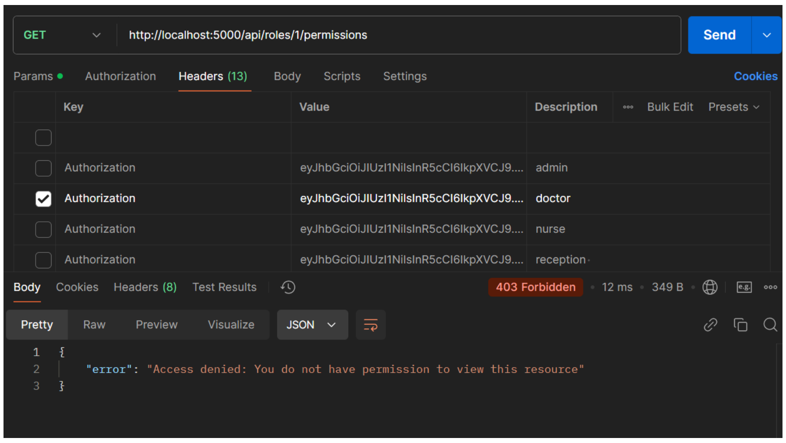


GET http://localhost:5000/api/roles

Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor
Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	nurse
Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	reception
✓ Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	patient

```
{
    "error": "Access denied: You do not have permission to view all roles"
}
```

Figure 15: ERROR other roles like patient can’t see the roles



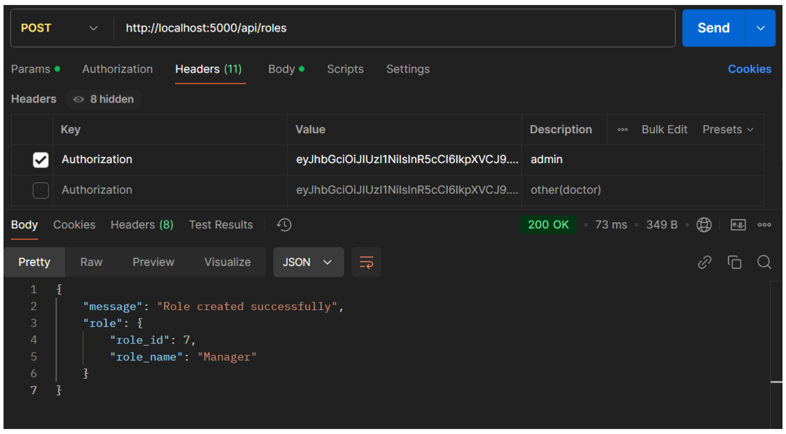
http://localhost:5000/api/roles/1/permissions

	Key	Value	Description
☐			
☐	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	admin
✓	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor
☐	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	nurse
☐	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	reception

403 Forbidden · 12 ms · 349 B

```
{
    "error": "Access denied: You do not have permission to view this resource"
}
```

Figure 16: ERROR other roles like patient can't see the roles permissions



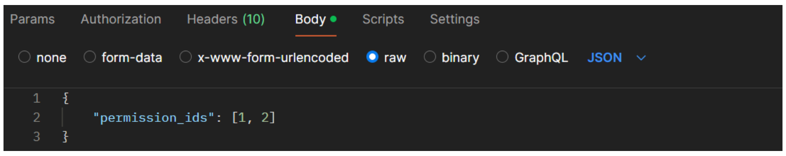
http://localhost:5000/api/roles

	Key	Value	Description
✓	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	admin
☐	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	other(doctor)

200 OK · 73 ms · 349 B

```
{
    "message": "Role created successfully",
    "role": {
        "role_id": 7,
        "role_name": "Manager"
    }
}
```

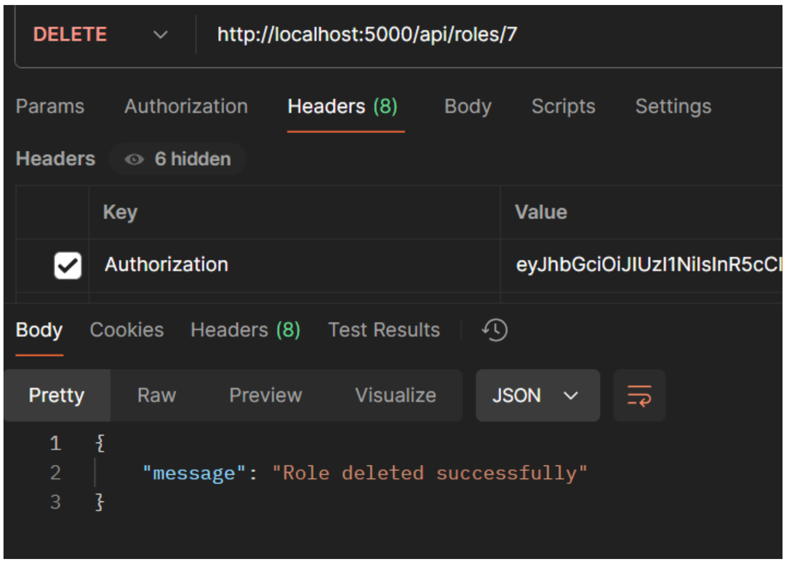
Figure 17: Roles creating



○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "permission_ids": [1, 2]
}
```

Figure 18: Permissions Creating

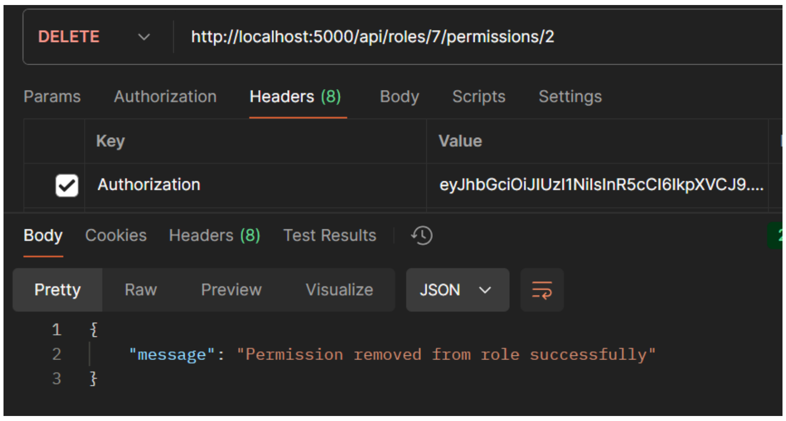


```
DELETE http://localhost:5000/api/roles/7
```

Key	Value
Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI

```
{
    "message": "Role deleted successfully"
}
```

Figure 19: Deleting Roles



```
DELETE http://localhost:5000/api/roles/7/permissions/2
```

Key	Value
Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....

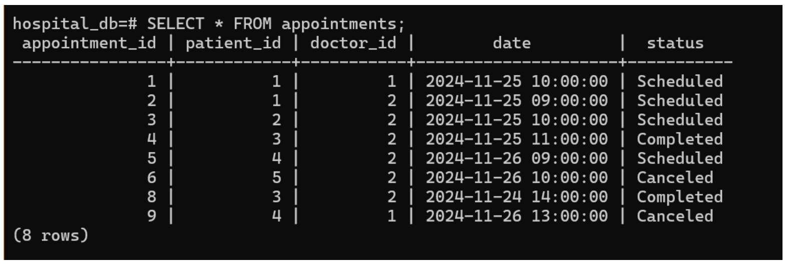
```
{
    "message": "Permission removed from role successfully"
}
```

Figure 20: Deleting permissions

12 Appointments Management

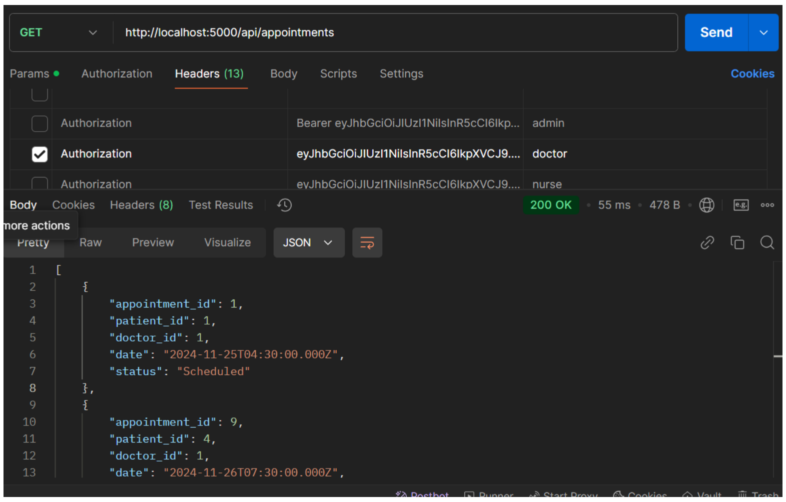
The Appointment APIs allow users to create, fetch, and update appointment details.

- POST `/appointments`: Create an appointment (Refer 24)
- GET `/appointments`: Fetch all appointments (Refer 25)
- PATCH `/appointments/:appointment_id`: Update appointment status



```
hospital_db=# SELECT * FROM appointments;
 appointment_id | patient_id | doctor_id |        date         |  status
----------------+------------+-----------+---------------------+-----------
              1 |          1 |         1 | 2024-11-25 10:00:00 | Scheduled
              2 |          1 |         2 | 2024-11-25 09:00:00 | Scheduled
              3 |          2 |         2 | 2024-11-25 10:00:00 | Scheduled
              4 |          3 |         2 | 2024-11-25 11:00:00 | Completed
              5 |          4 |         2 | 2024-11-26 09:00:00 | Scheduled
              6 |          5 |         2 | 2024-11-26 10:00:00 | Canceled
              8 |          3 |         2 | 2024-11-24 14:00:00 | Completed
              9 |          4 |         1 | 2024-11-26 13:00:00 | Canceled
(8 rows)
```

Figure 21: Databse Appointment list



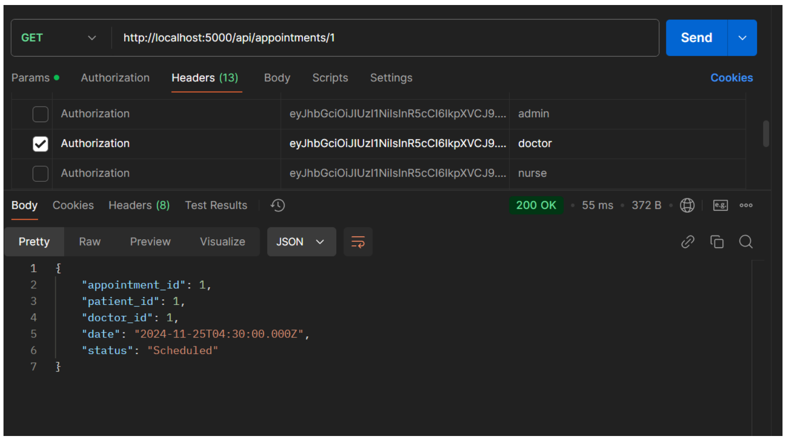
GET http://localhost:5000/api/appointments

	Key	Value	Description
☐	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp...	admin
✓	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor
☐	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	nurse

200 OK · 55 ms · 478 B

```
[
    {
        "appointment_id": 1,
        "patient_id": 1,
        "doctor_id": 1,
        "date": "2024-11-25T04:30:00.000Z",
        "status": "Scheduled"
    },
    {
        "appointment_id": 9,
        "patient_id": 4,
        "doctor_id": 1,
        "date": "2024-11-26T07:30:00.000Z",
```

Figure 22: Checking Fetching all his (doctor)Appointment



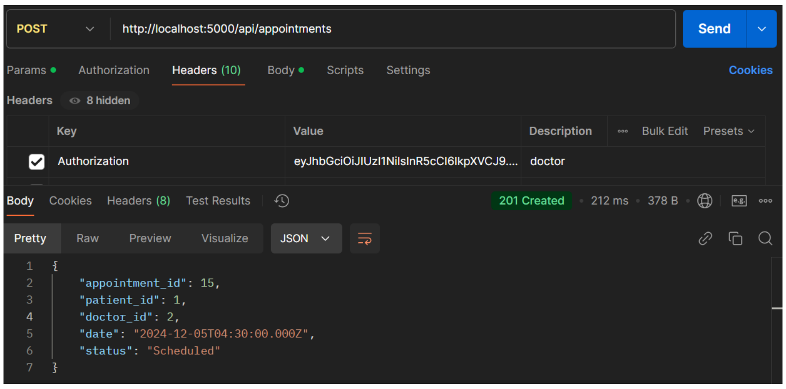
GET http://localhost:5000/api/appointments/1

	Key	Value	Description
☐	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	admin
✓	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor
☐	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	nurse

200 OK · 55 ms · 372 B

```
{
    "appointment_id": 1,
    "patient_id": 1,
    "doctor_id": 1,
    "date": "2024-11-25T04:30:00.000Z",
    "status": "Scheduled"
}
```

Figure 23: Appointment check by patient only his appointments



POST http://localhost:5000/api/appointments

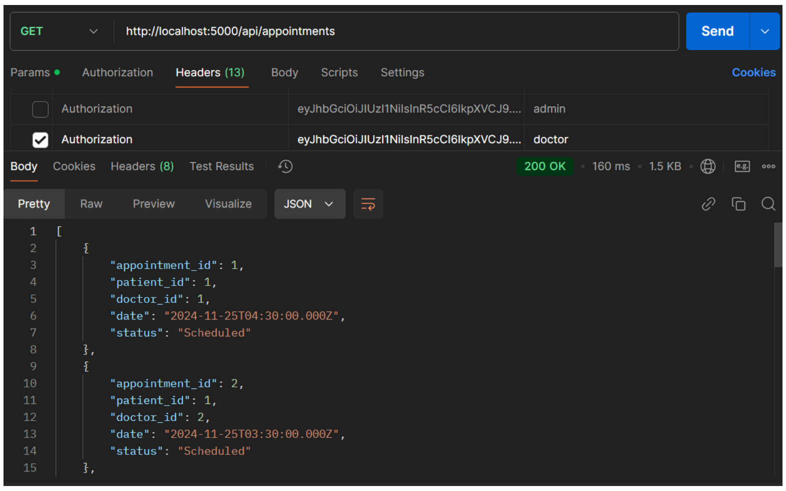
Headers (10)

Key	Value	Description
✓ Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor

201 Created · 212 ms · 378 B

```
{
    "appointment_id": 15,
    "patient_id": 1,
    "doctor_id": 2,
    "date": "2024-12-05T04:30:00.000Z",
    "status": "Scheduled"
}
```

Figure 24: doctor creating a Appointment



GET http://localhost:5000/api/appointments

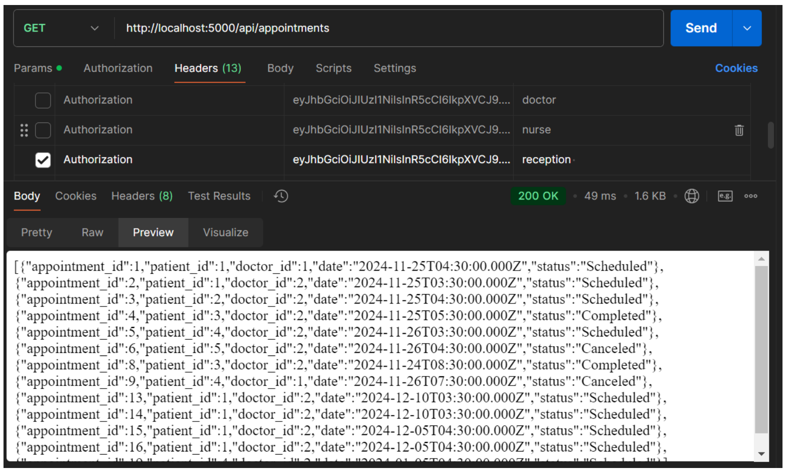
Headers (13)

Key	Value	Description
Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	admin
✓ Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor

200 OK · 160 ms · 1.5 KB

```
[
    {
        "appointment_id": 1,
        "patient_id": 1,
        "doctor_id": 1,
        "date": "2024-11-25T04:30:00.000Z",
        "status": "Scheduled"
    },
    {
        "appointment_id": 2,
        "patient_id": 1,
        "doctor_id": 2,
        "date": "2024-11-25T03:30:00.000Z",
        "status": "Scheduled"
    },
```

Figure 25: Appointment fetch by main doctor can see other doctors appointments also



GET http://localhost:5000/api/appointments

Headers (13)

☐	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor
☐	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	nurse
✓	Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	reception

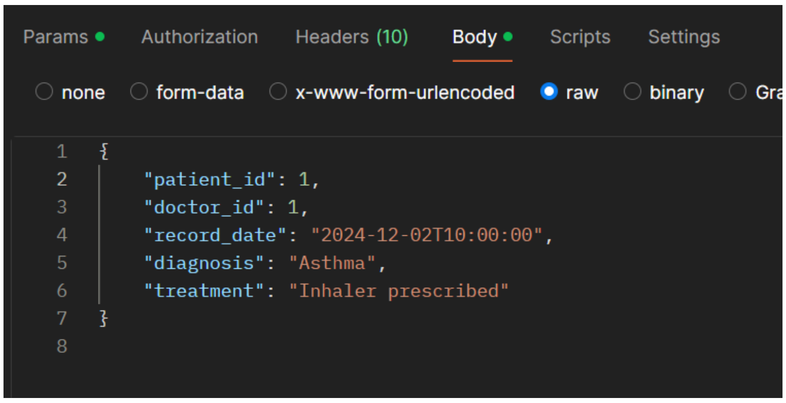
200 OK · 49 ms · 1.6 KB

```
[{"appointment_id":1,"patient_id":1,"doctor_id":1,"date":"2024-11-25T04:30:00.000Z","status":"Scheduled"},
{"appointment_id":2,"patient_id":1,"doctor_id":2,"date":"2024-11-25T03:30:00.000Z","status":"Scheduled"},
{"appointment_id":3,"patient_id":2,"doctor_id":2,"date":"2024-11-25T04:30:00.000Z","status":"Scheduled"},
{"appointment_id":4,"patient_id":3,"doctor_id":2,"date":"2024-11-25T05:30:00.000Z","status":"Completed"},
{"appointment_id":5,"patient_id":4,"doctor_id":2,"date":"2024-11-26T03:30:00.000Z","status":"Scheduled"},
{"appointment_id":6,"patient_id":5,"doctor_id":2,"date":"2024-11-26T04:30:00.000Z","status":"Canceled"},
{"appointment_id":8,"patient_id":3,"doctor_id":2,"date":"2024-11-24T08:30:00.000Z","status":"Completed"},
{"appointment_id":9,"patient_id":4,"doctor_id":1,"date":"2024-11-26T07:30:00.000Z","status":"Canceled"},
{"appointment_id":13,"patient_id":1,"doctor_id":2,"date":"2024-12-10T03:30:00.000Z","status":"Scheduled"},
{"appointment_id":14,"patient_id":1,"doctor_id":2,"date":"2024-12-10T03:30:00.000Z","status":"Scheduled"},
{"appointment_id":15,"patient_id":1,"doctor_id":2,"date":"2024-12-05T04:30:00.000Z","status":"Scheduled"},
{"appointment_id":16,"patient_id":1,"doctor_id":2,"date":"2024-12-05T04:30:00.000Z","status":"Scheduled"},
```

Figure 26: Appointment check by receiptionist (can see all appointments for all doctors similar like admin)

13 Medical Records

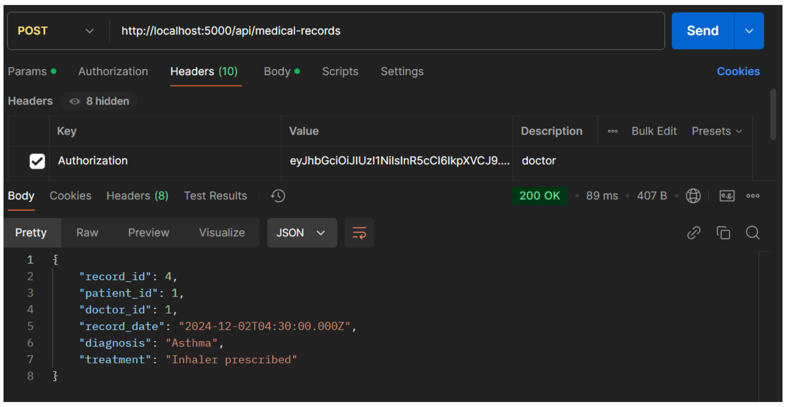
Medical records are managed and retrieved using the `/medical-records` API. See 27 for details of medical information.



Body: raw

```
{
    "patient_id": 1,
    "doctor_id": 1,
    "record_date": "2024-12-02T10:00:00",
    "diagnosis": "Asthma",
    "treatment": "Inhaler prescribed"
}
```

Figure 27: Medical Records Body Text

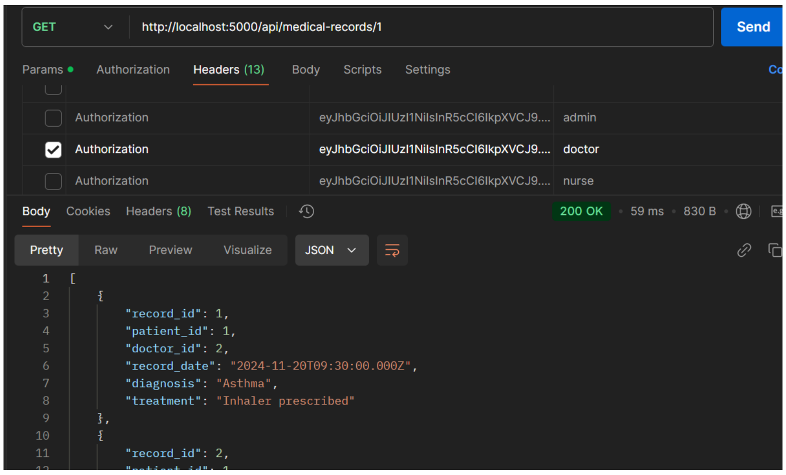


http://localhost:5000/api/medical-records

Key	Value	Description
✓ Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor

```
{
    "record_id": 4,
    "patient_id": 1,
    "doctor_id": 1,
    "record_date": "2024-12-02T04:30:00.000Z",
    "diagnosis": "Asthma",
    "treatment": "Inhaler prescribed"
}
```

Figure 28: Medical Records Result



http://localhost:5000/api/medical-records/1

Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	admin
✓ Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	doctor
Authorization	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....	nurse

```
[
    {
        "record_id": 1,
        "patient_id": 1,
        "doctor_id": 2,
        "record_date": "2024-11-20T09:30:00.000Z",
        "diagnosis": "Asthma",
        "treatment": "Inhaler prescribed"
    },
    {
        "record_id": 2,
```

Figure 29: Medical Records Result checking using specif patient