RESTful API Security

Name : Mohd Gulam Ansari
Roll No : 242CS023
Class : M. Tech. CSE

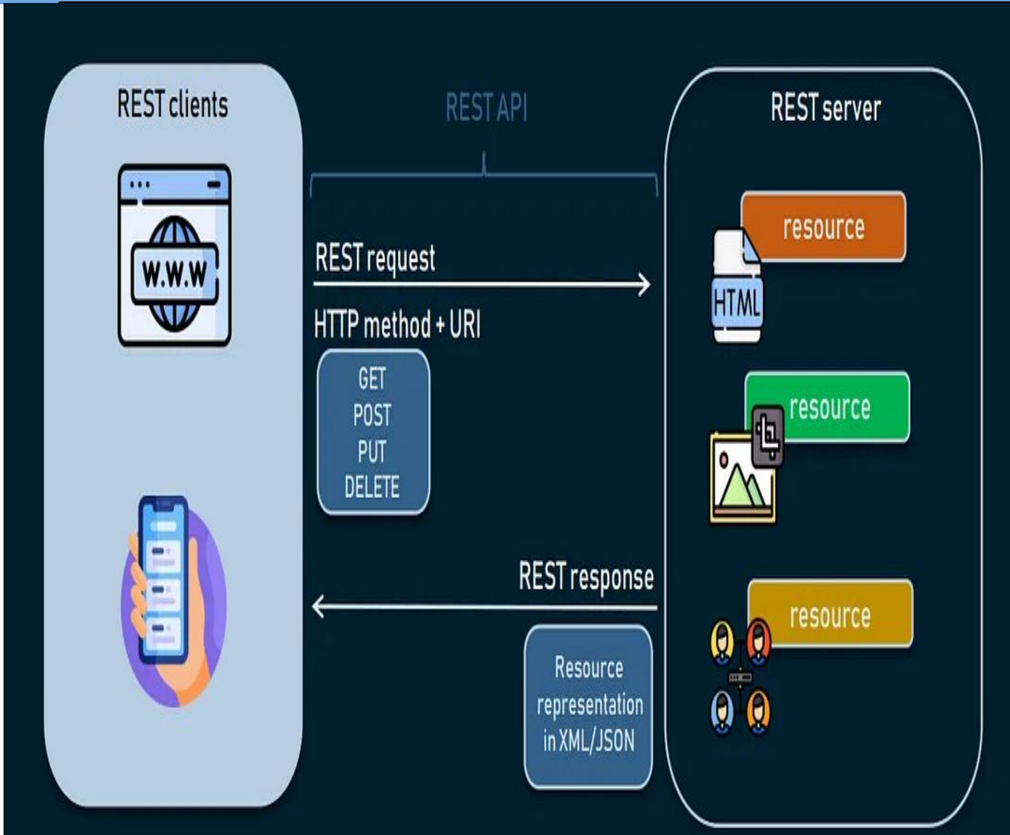
Under The Guidance :
Professor P. Santhi Thilagam

Overview

- Introduction
- REST API Architecture
- HTTP Methods and Status Code
- Implementation of REST API Demonstration
- Authentication & Authorization
- Session Based Authentication
- JWT Based Authentication
- DDoS Attack and Mitigation
- Conclusion
- References

Introduction

- **REST API** stands for **Representational State Transfer API**. It is a type of API (Application Programming Interface) that allows communication between different systems over the internet.
- APIs are the backbone of modern applications, enabling seamless communication between services, platforms, and devices.
- If APIs aren't properly secured, hackers can use them to **steal data or break into systems**.

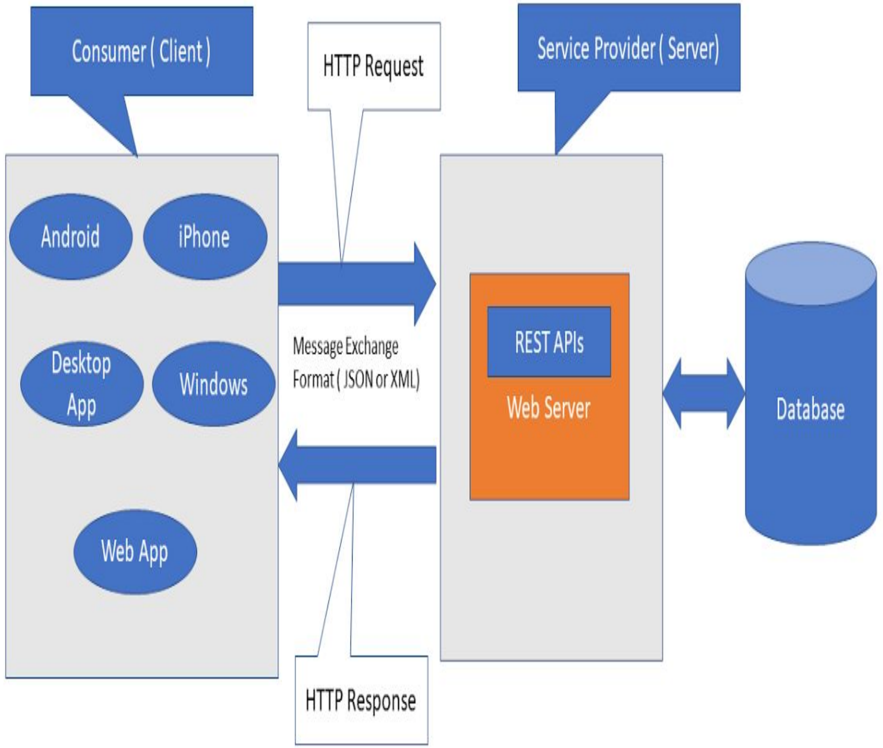


RESTful API Architecture

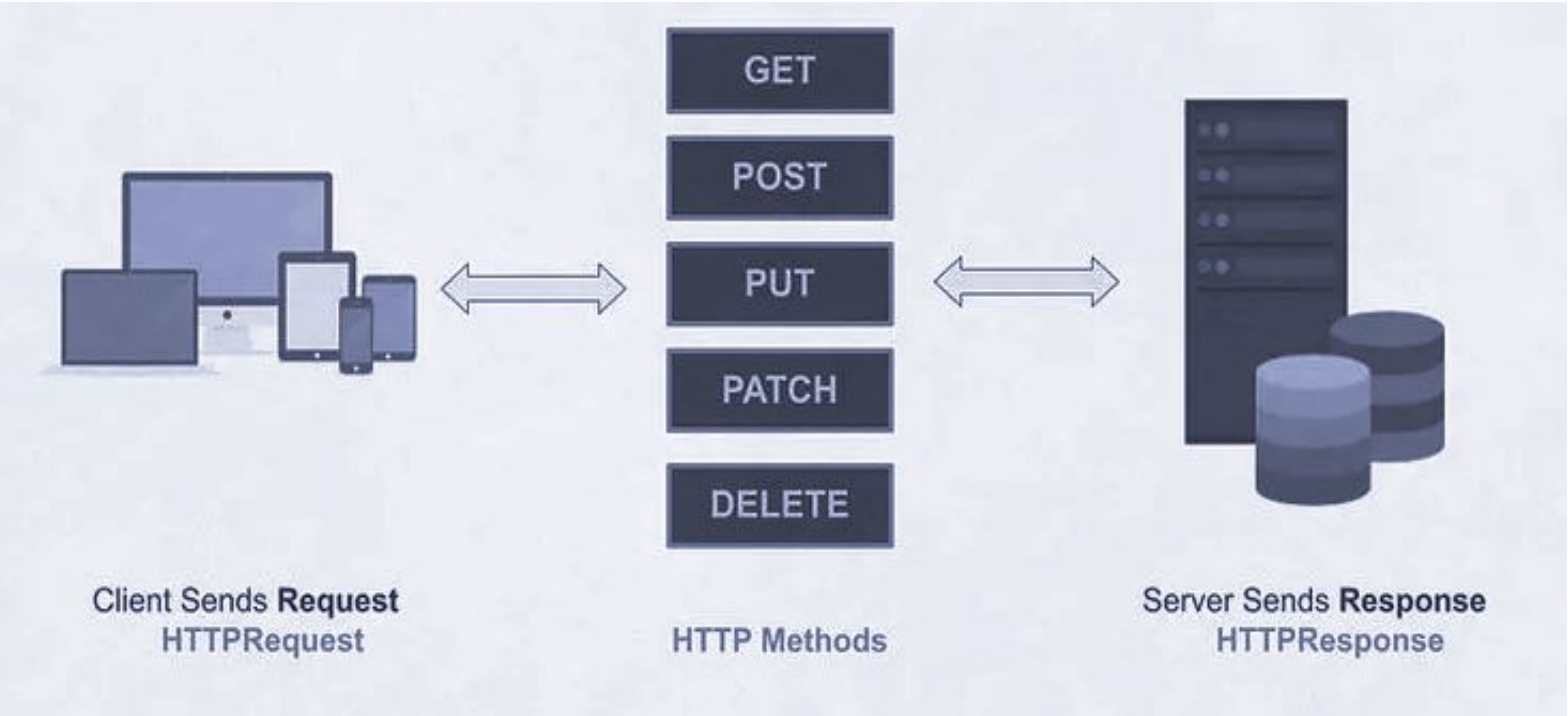
1. **Client and Server Model :**
   The client (like a phone or browser) sends a request, and the server replies with the data.

2. **Stateless Communication :**
   Each request from the client must contain all the info the server needs — it doesn't remember past requests.

3. **Use of HTTP Methods :**
   REST APIs use common web actions like GET (to read), POST (to send), PUT (to update), and DELETE (to remove data).



HTTP Methods and Status Codes



HTTP Methods and Status Codes

Each method tells the API what action to perform, and the status code shows if it worked or failed.

HTTP Method	Action	Example Endpoint	What it Does
GET	Fetch all patients	`/patients/`	Returns a list of all patients
GET	Fetch one patient	`/patients/5`	Returns details of patient with ID = 5
POST	Create a new patient	`/patients/`	Adds a new patient to the system
PUT	Update patient info	`/patients/5`	Updates info of patient with ID = 5
DELETE	Delete a patient	`/patients/5`	Deletes patient with ID = 5

HTTP Methods and Status Codes

Status Code	Meaning	In Our Patient API - Action
200 OK	Success	Data fetched, updated, or deleted successfully
201 Created	New resource added	New patient was created and added to the database
400 Bad Request	Input is invalid	Missing or incorrect data (e.g. empty name or age field)
401 Unauthorized	Not logged in or invalid	Accessing a protected route without token (if auth used)
404 Not Found	Resource doesn't exist	Patient with the given ID doesn't exist
500 Server Error	Internal crash	Server error like DB crash or unhandled exception

RESTful API Implementation

- **Tech Stack:**
  Programming Language : **Python**
  Libraries and Tools : FastAPI, Pydantic for data validation, Uvicorn, Render for deployment

- **Key Features:**
  1. Create, Read, Update, Delete (CRUD) for Patient data

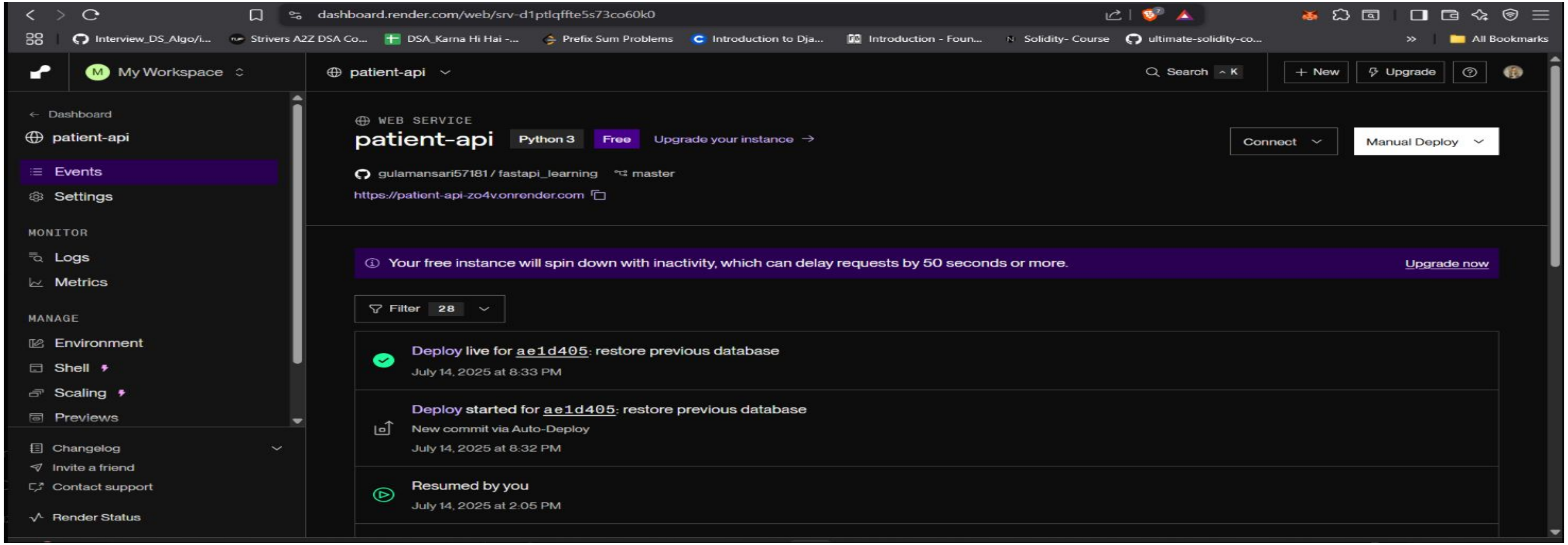
  2.Input validation using Pydantic as python is dynamically type language

  3.API Documentation with Swagger UI

- **Live Demo Link:**
  https://patient-api-zo4v.onrender.com

Implementation Snapshots

Patient API is live and hosted on Render at: https://patient-api-zo4v.onrender.com



WEB SERVICE

patient-api Python 3 Free Upgrade your instance →

gulamansari57181 / fastapi_learning master

https://patient-api-zo4v.onrender.com

Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more. Upgrade now

Filter 28

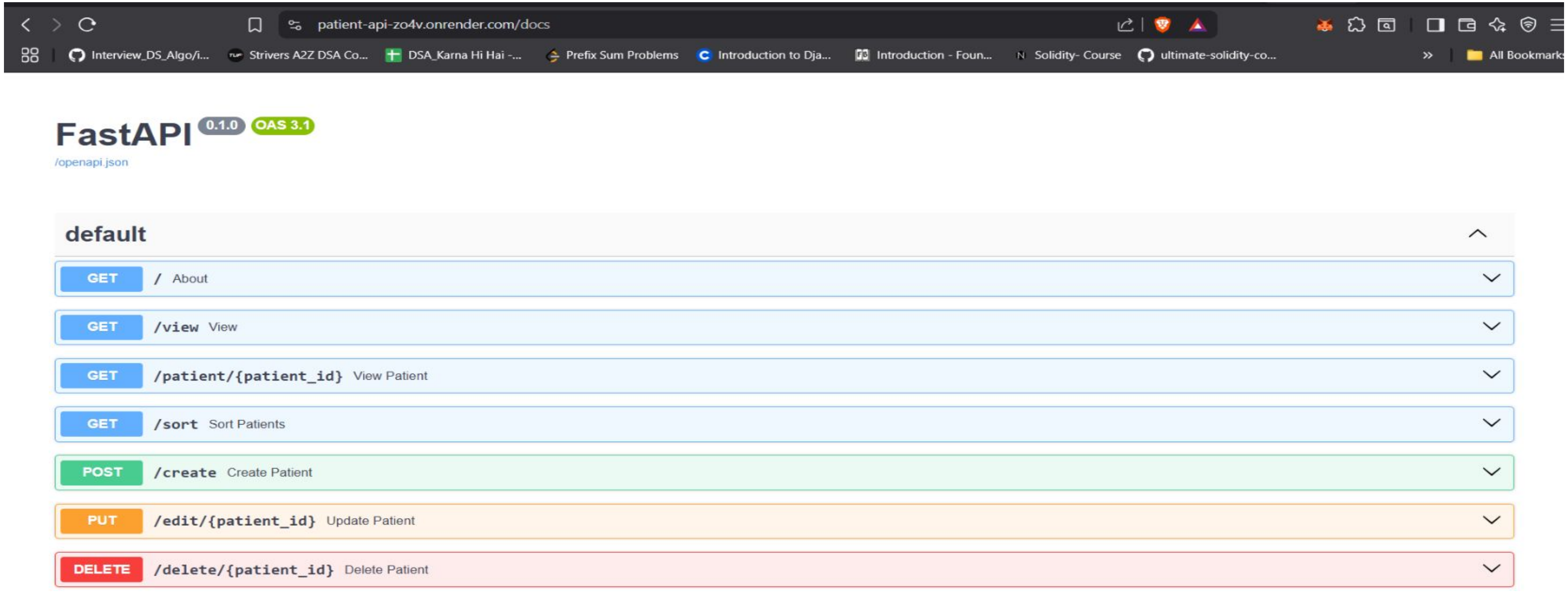
Deploy live for ae1d405: restore previous database
July 14, 2025 at 8:33 PM

Deploy started for ae1d405: restore previous database
New commit via Auto-Deploy
July 14, 2025 at 8:32 PM

Resumed by you
July 14, 2025 at 2:05 PM

Implementation Snapshots

The interactive Swagger documentation displays all available Patient API endpoints with HTTP methods.



patient-api-zo4v.onrender.com/docs

Interview_DS_Algo/i... Strivers A2Z DSA Co... DSA_Karna Hi Hai -... Prefix Sum Problems Introduction to Dja... Introduction - Foun... Solidity- Course ultimate-solidity-co... All Bookmarks

FastAPI 0.1.0 OAS 3.1

/openapi.json

default

GET / About

GET /view View

GET /patient/{patient_id} View Patient

GET /sort Sort Patients

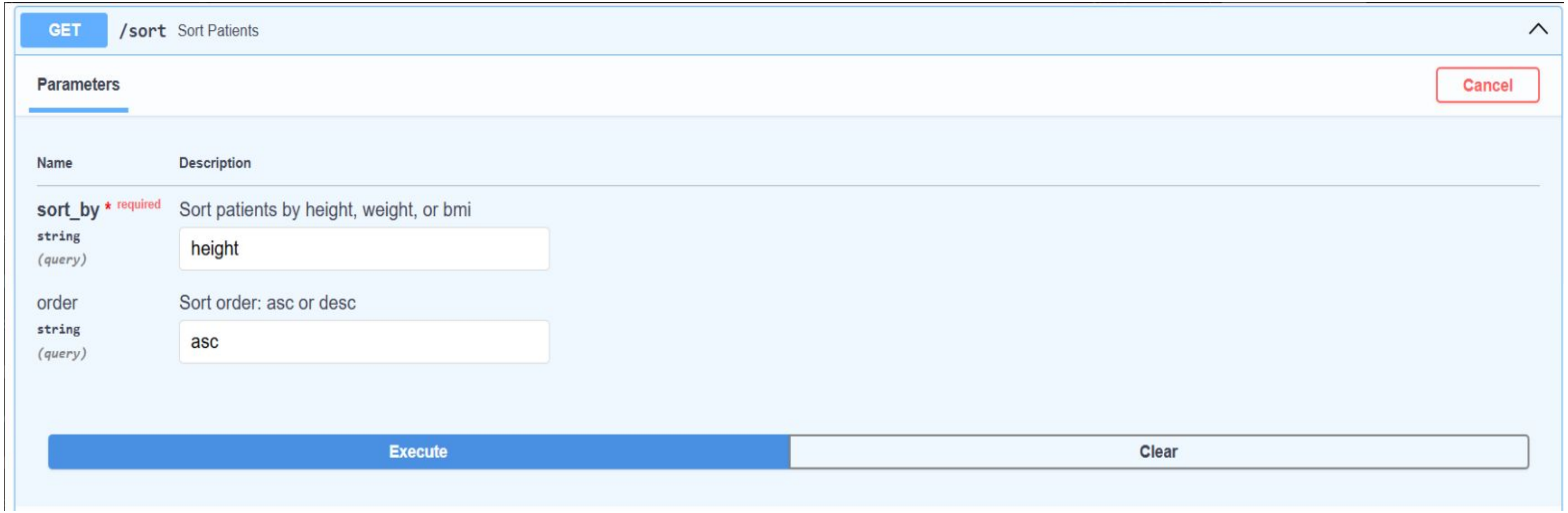
POST /create Create Patient

PUT /edit/{patient_id} Update Patient

DELETE /delete/{patient_id} Delete Patient

Implementation Snapshots

GET /sort Endpoint : This endpoint allows sorting patient data by height, weight, or BMI using query parameters.



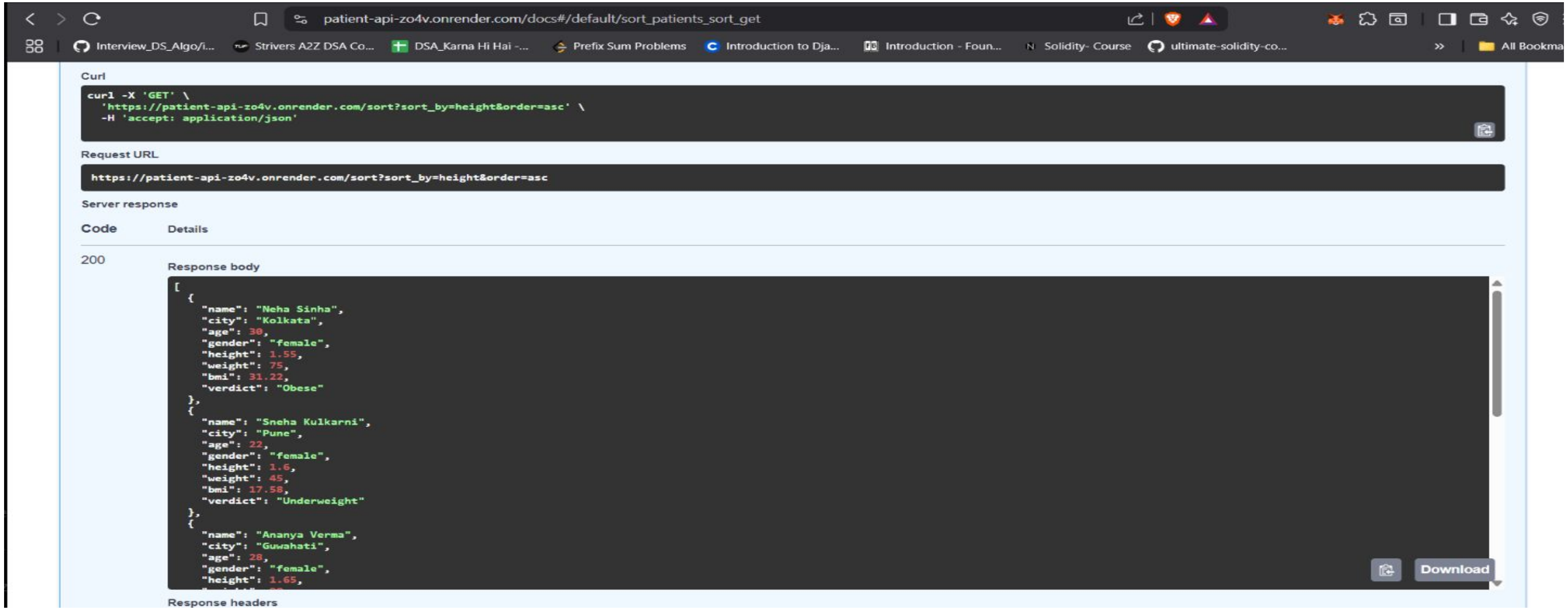
GET /sort Sort Patients

Parameters

Name	Description
sort_by * required string (query)	Sort patients by height, weight, or bmi height
order string (query)	Sort order: asc or desc asc

Implementation Snapshots

Sort API Response Example : /sort endpoint showing patients sorted by height in ascending order.



patient-api-zo4v.onrender.com/docs#/default/sort_patients_sort_get

Curl

```
curl -X 'GET' \
  'https://patient-api-zo4v.onrender.com/sort?sort_by=height&order=asc' \
  -H 'accept: application/json'
```

Request URL

```
https://patient-api-zo4v.onrender.com/sort?sort_by=height&order=asc
```

Server response

Code | Details

200

Response body

```
[
  {
    "name": "Neha Sinha",
    "city": "Kolkata",
    "age": 30,
    "gender": "female",
    "height": 1.55,
    "weight": 75,
    "bmi": 31.22,
    "verdict": "Obese"
  },
  {
    "name": "Sneha Kulkarni",
    "city": "Pune",
    "age": 22,
    "gender": "female",
    "height": 1.6,
    "weight": 45,
    "bmi": 17.58,
    "verdict": "Underweight"
  },
  {
    "name": "Ananya Verma",
    "city": "Guwahati",
    "age": 28,
    "gender": "female",
    "height": 1.65,
```

Response headers

Authentication & Authorization

- **Authentication :**

  Authentication is an important part of identity and access management (IAM), which dictates who can view data and what they can do with it.

- **Authorization :**
  Authorization is concerned with permissions, or what someone is allowed to do once they gain access to a protected system or resource.

  They protect our RESTful API from unauthorized access and misuse.

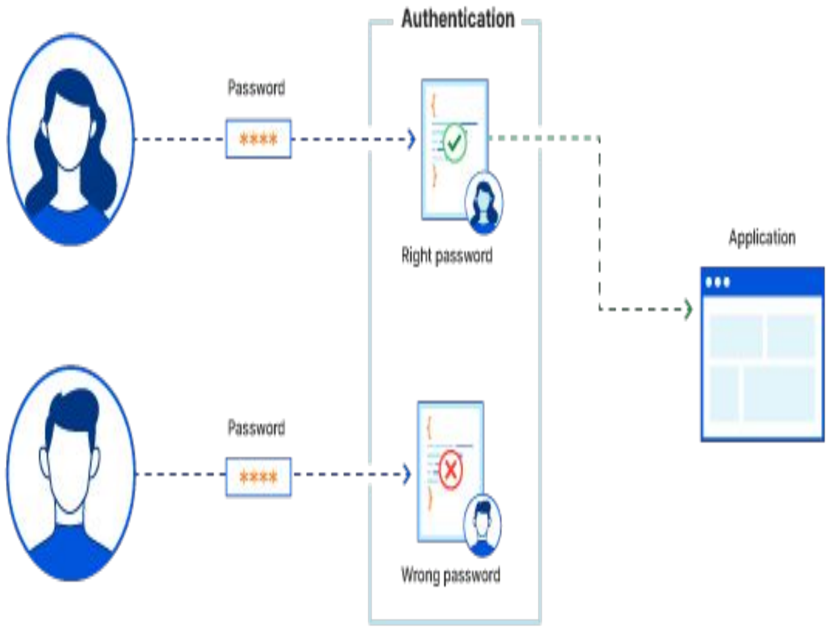


Image source : https://www.cloudflare.com/learning/access-management/what-is-authentication/

Session-Based Authentication

- **Working:**

1. When a user logs in, the server creates a **session** and stores it (usually in memory or a database).
2. The server sends a session ID to the user's browser, which is stored as a cookie.
3. For every request, the browser automatically sends the session ID, and the server uses it to identify the user.

It is for websites because it's simple and browser-friendly, but it struggles with scalability and doesn't work well for stateless APIs or mobile apps

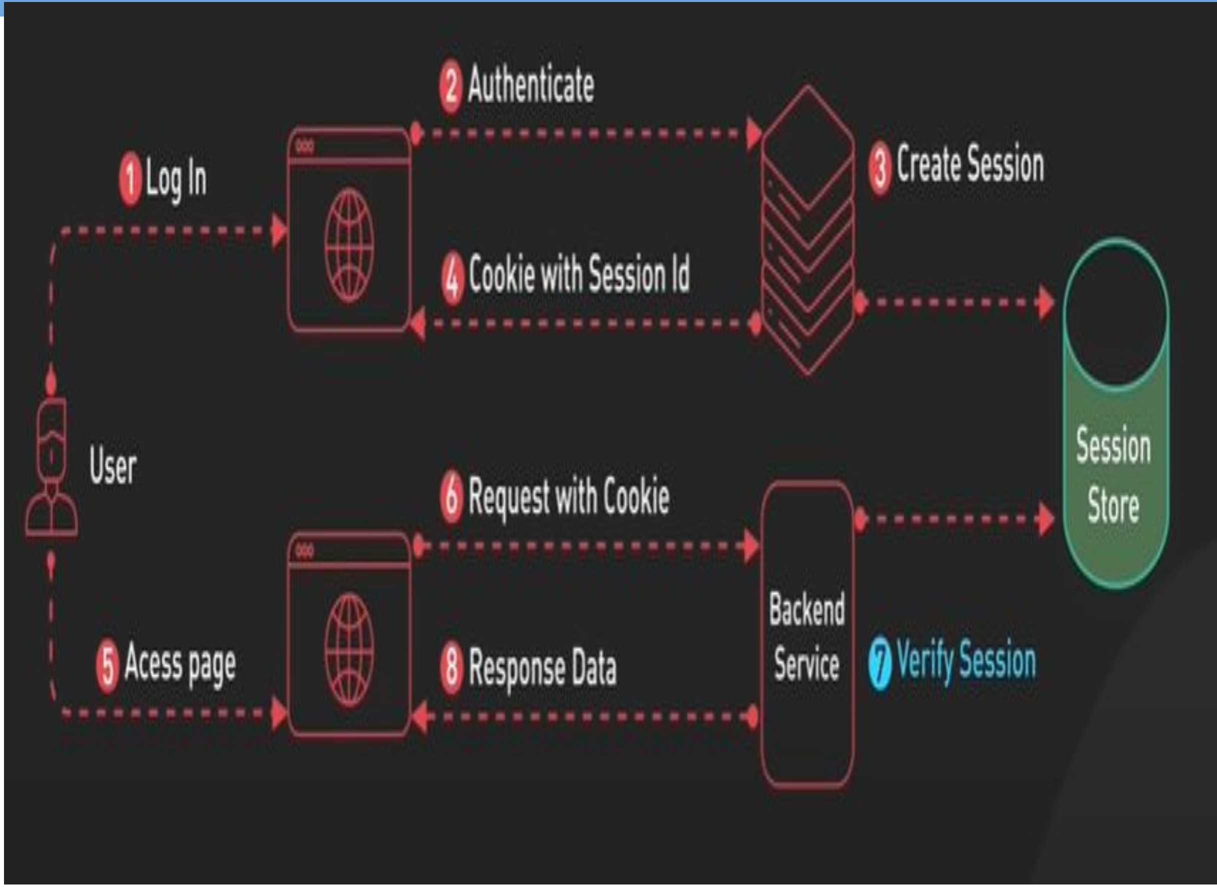


Image Source :Bytesystem.com/session-based-authentication

JWT-Based Authentication

- **Working:**

1. When a user logs in, the server generates a **JWT (JSON Web Token)** and sends it to the client.

2. The client stores this token (usually in localStorage or memory).

3. The token is sent with every request (in the header), and the server verifies it.

   JWT is perfect for APIs and mobile apps because it's stateless and scalable, but harder to revoke access of the user (token expiry mechanism).

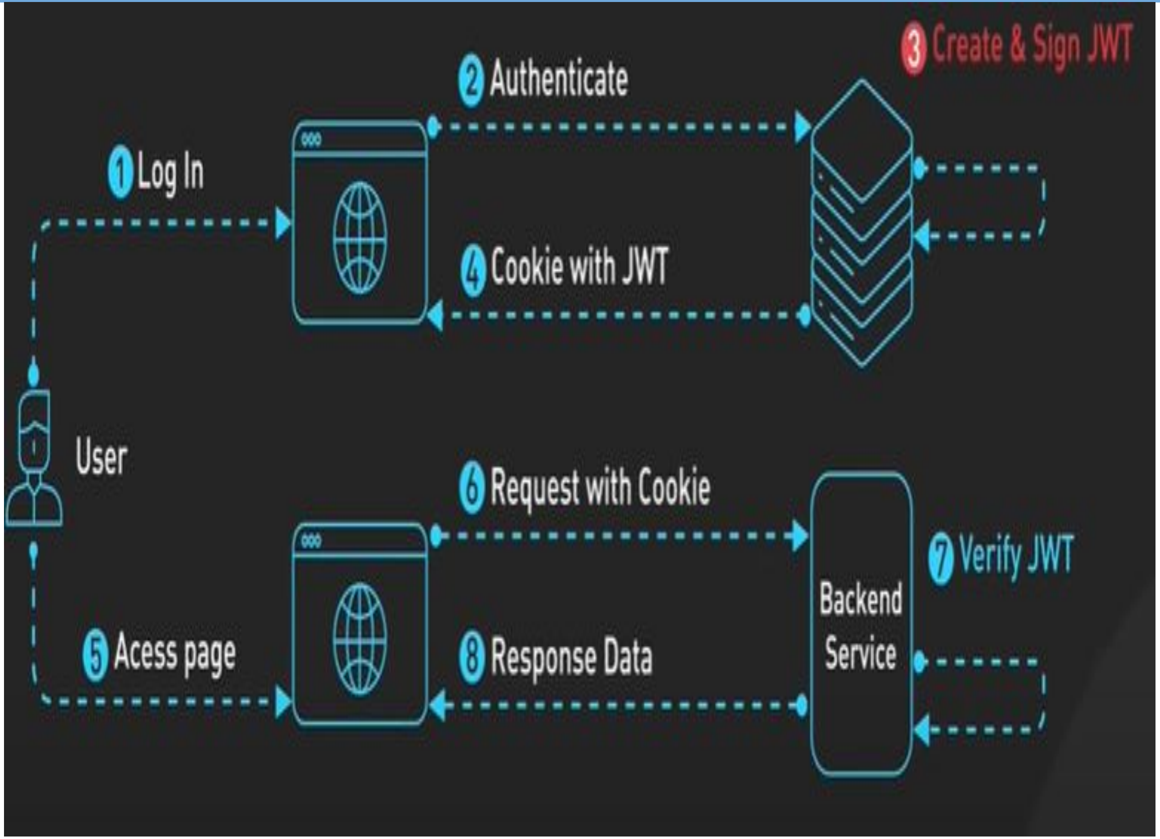


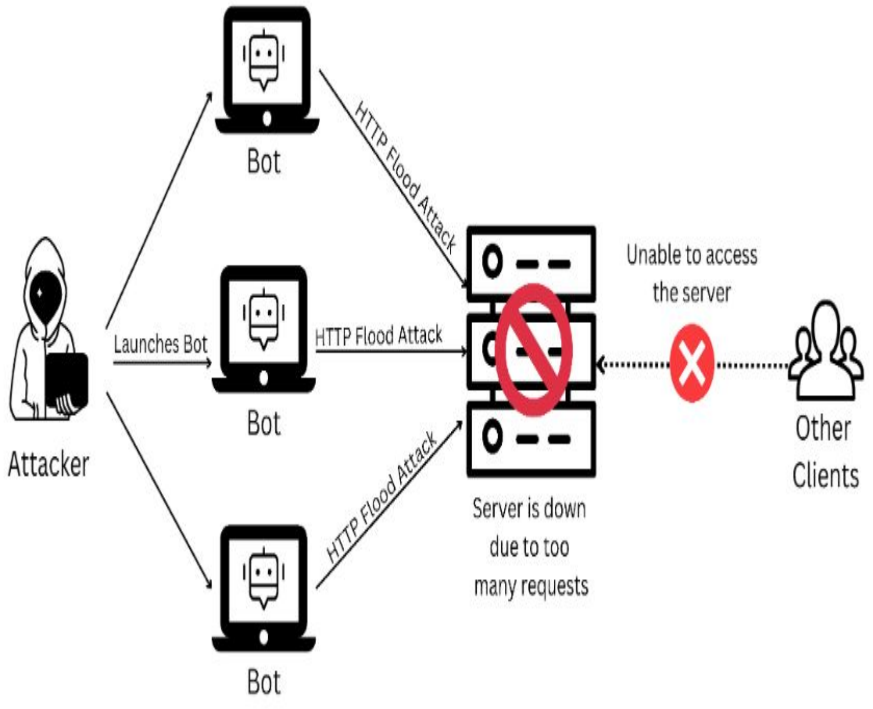
Image Source :ByteSystem.com/jwt-authentication

Distributed Denial of Service (DDoS) Attacks:

- **Working:**

1. **Botnet Setup**: Attackers infect multiple devices to create a botnet.
2. **Flood of DNS Queries**: The botnet sends overwhelming requests to the target server.
3. **Resource Overload**: The server becomes overloaded, depleting its resources.
4. **Service Outage**: Legitimate users are denied access as the server fails to respond.

- **Potential Impacts:**
  The attack can cause service disruption, downtime, revenue loss, and network congestion.



RRL - Response Rate Limiting

RRL, or Response Rate Limiting, is an enhancement which serves as a mitigation tool for the problem of **amplification attacks**.

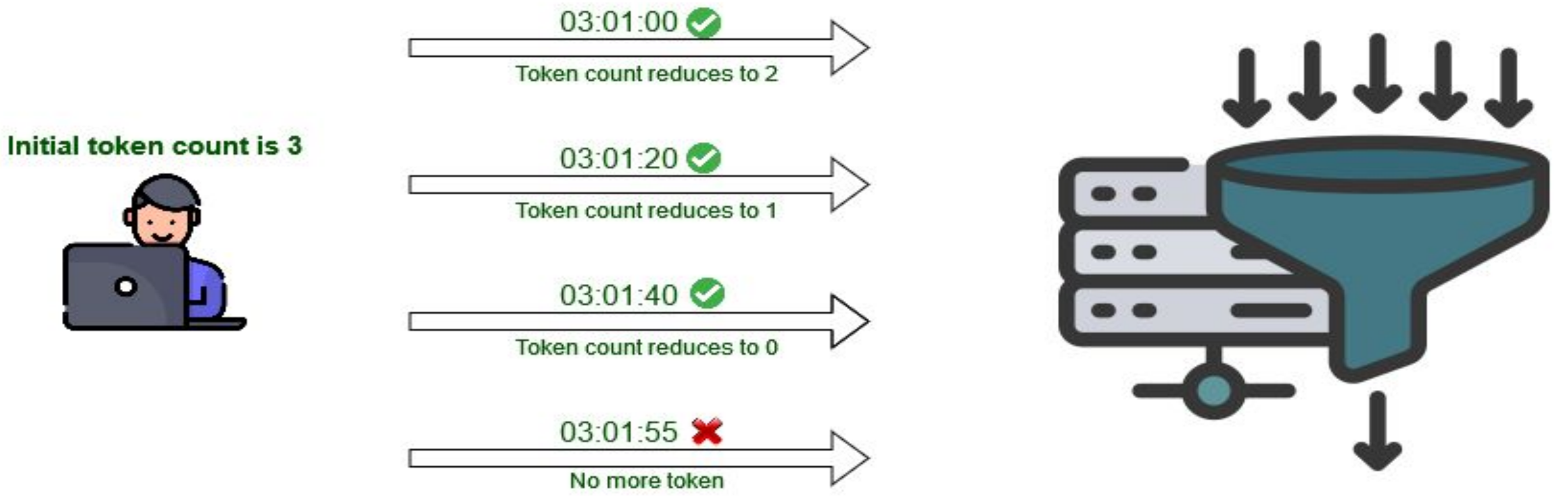
Algorithm :

RRL is an algorithm that limits the number of responses a server sends to a particular client in a specific timeframe.

How it works:

When the server detects that too many responses are being sent to a single source (potentially from spoofed requests), it throttles or blocks further responses. This reduces the effectiveness of amplification by limiting the number of large responses.

RRL - Response Rate Limiting



Token Bucket

Conclusion :

- **Understood how RESTful APIs working** — including HTTP methods, status codes, and how their clients and servers architecture.
- **Learn to built and deployed** my own Patient API — using FastAPI, with Swagger UI for testing and Render for live hosting.
- **Explored and implemented JWT Authentication** — to protect sensitive endpoints and control access using tokens.
- Gained hands-on understanding of common API security threats and how to protect APIs using input validation, authentication and rate limiting.

References :

- “Security measures implemented in RESTful API Development”,Arvind A Kumar
- and Divya TL (2024)
- Lokesh Gupta. (2024, October 23). REST API Best Practices. RESTfulAPI.net — Includes sections on security, versioning, and performance
- Postman. (2024). API Security Best Practices. Postman Learning Center. https://learning.postman.com/docs/sending-requests/authorization/
- FastAPI. (2024). Security - First Steps. FastAPI Official Documentation. https://fastapi.tiangolo.com/tutorial/security/
- https://bytebytego.com/guides/api-web-development/

Thank You