REST API Handling (POST & GET)

Objective

The objective of Day 4 task is to understand, design, and implement REST APIs using Flask with proper request handling, validation, error handling, and MongoDB integration.

Technology Stack

- Python 3.x
- Flask
- MongoDB
- PyMongo
- Postman

API Overview

This document covers POST and GET APIs for a Task Management System. The APIs follow REST standards and use JSON for communication.

Base URL

http://localhost:5000/api

POST API – Create Task

Endpoint: POST /tasks

Purpose:
Used to create a new task and store it in MongoDB.

Request Headers:
Content-Type: application/json

Request Body Example:
```
{
  "title": "API Documentation",
  "description": "Prepare Day 4 REST API document",
  "status": "Pending",
  "priority": "High",
  "due_date": "2025-01-10"
}
```

Validation Rules:
- Title is mandatory
- Status must be valid
- JSON format must be correct

Success Response:
201 Created

Error Responses:
400 – Bad Request
415 – Unsupported Media Type
500 – Internal Server Error

GET API – Fetch Tasks

Endpoint: GET /tasks

Purpose:
Fetch all tasks stored in MongoDB.

Request Parameters:
None

Success Response:
200 OK

Sample Response:
```
[
 {
  "_id": "65a1234",
  "title": "API Documentation",
  "status": "Pending"
 }
]
```

Error Responses:
404 – No records found
500 – Database error

HTTP Status Codes Used

200 – OK
201 – Created
400 – Bad Request
404 – Not Found
415 – Unsupported Media Type
500 – Internal Server Error

Testing Approach

- Tested APIs using Postman
- Verified request validation
- Verified database insertion and retrieval
- Checked error handling scenarios