Project Brief: Book Manager REST API

Project Overview

Your task is to build a RESTful API for managing a collection of books. This project simulates a simplified backend service like those used in internal tools at tech companies such as Amazon or Google. No frontend is required. You will expose API endpoints to Create, Read, Update, and Delete (CRUD) book records.

API Requirements

- GET /books - Return a list of all books.

- GET /books/<book_id> - Return details of a specific book by ID.

- POST /books - Add a new book. Accepts JSON payload: { "title": "...", "author": "..." }

- PUT /books/<book_id> - Update a book's title and/or author.

- DELETE /books/<book_id> - Remove a book from the collection.

Data Structure

Use an in-memory list or dictionary to store book records. Each book should be a JSON object like:

```
{
  "id": 1,
  "title": "1984",
  "author": "George Orwell"
}
```

Key Expectations

- Each book must have a unique ID.

- Return proper HTTP status codes (e.g., 200, 201, 404).

Project Brief: Book Manager REST API

- Return responses in JSON format only.

- Handle edge cases (e.g., invalid IDs, missing data).

Bonus Challenges (Optional)

- Implement search by author: GET /books?author=George

- Prevent empty or invalid input using validation checks.

- Store and read data from a JSON file (instead of in-memory).

Notes

This task is intentionally kept simple to focus on backend logic and API design. You can use Flask with request/jsonify only. No external DB or ORM is required at this point.