Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

School of Computer Engineering and Technology
Academic Year: 2023-2024 Sem V

Fullstack Development

Lab Assignment : 06

Title: Develop a set of REST API using Express and Node.

Prepared By
Saurabh Jitendra Jadhav
Roll No:-PA12
Batch A1
November 22,2023

Aim: Develop a set of REST API using Express and Node.

Objectives:

1) To define HTTP GET and POST operations.
2) To understand and make use of ‘REST’, ‘a REST endpoint’, ‘API Integration’, and API Invocation’
3) To understand the use of a REST Client to make POST and GET requests to an API.

Theory:-

What is REST API?

REST (Representational State Transfer) API is an architectural style for designing networked applications. It defines a set of constraints that are intended to create scalable, maintainable, and stateless communication between systems. REST APIs use standard HTTP methods for communication and are commonly used in web development to enable communication between a client and a server.

Key Principles of REST:

- **Stateless:** Each request from a client to a server contains all the information needed to understand and fulfill that request. The server doesn't store any information about the client's state between requests.
- **Client-Server Architecture:** Separates the concerns of the client and the server. The client is responsible for the user interface and user experience, while the server is responsible for processing requests and managing resources.

- **Uniform Interface:** The interface between clients and servers should be uniform to promote simplicity and scalability. This includes resource-based addressing, the use of standard HTTP methods (GET, POST, PUT, DELETE), and a self-descriptive message format.
- **Resource-Based:** Resources, such as data or services, are identified by URIs (Uniform Resource Identifiers). Clients interact with these resources through the standard HTTP methods to perform CRUD operations (Create, Read, Update, Delete).
- **Representation:** Resources can have multiple representations (e.g., JSON, XML, HTML). The client can choose the representation that best suits its needs.

Main Purpose of REST API:

The main purpose of a REST API is to facilitate the exchange of data between different systems in a standardized and scalable manner. Some key aspects of its purpose include:

- **Interoperability:** REST APIs enable interoperability between systems, allowing applications developed in different languages or running on different platforms to communicate seamlessly.
- **Scalability:** REST APIs are designed to be scalable. By adhering to stateless communication and using a resource-based approach, they can handle a large number of clients and requests.
- **Simplicity:** REST APIs are simple to understand and use. They leverage existing HTTP standards and are based on a straightforward architecture, making them widely adopted in web development.
- **Flexibility:** Clients can interact with resources using different representations (e.g., JSON or XML), and servers can evolve independently without affecting clients that adhere to the defined interface.

- **Statelessness:** The stateless nature of REST APIs simplifies server-side logic and allows for better scalability. Each request from a client contains all the information needed for the server to fulfill that request.

FAQ:-

FAQ:

What are HTTP Request types?

-->HTTP (Hypertext Transfer Protocol) defines a set of request methods, also known as HTTP request types, that indicate the desired action to be performed for a given resource. Each HTTP request type corresponds to a specific operation that can be performed on the server.

Here are the common HTTP request types:

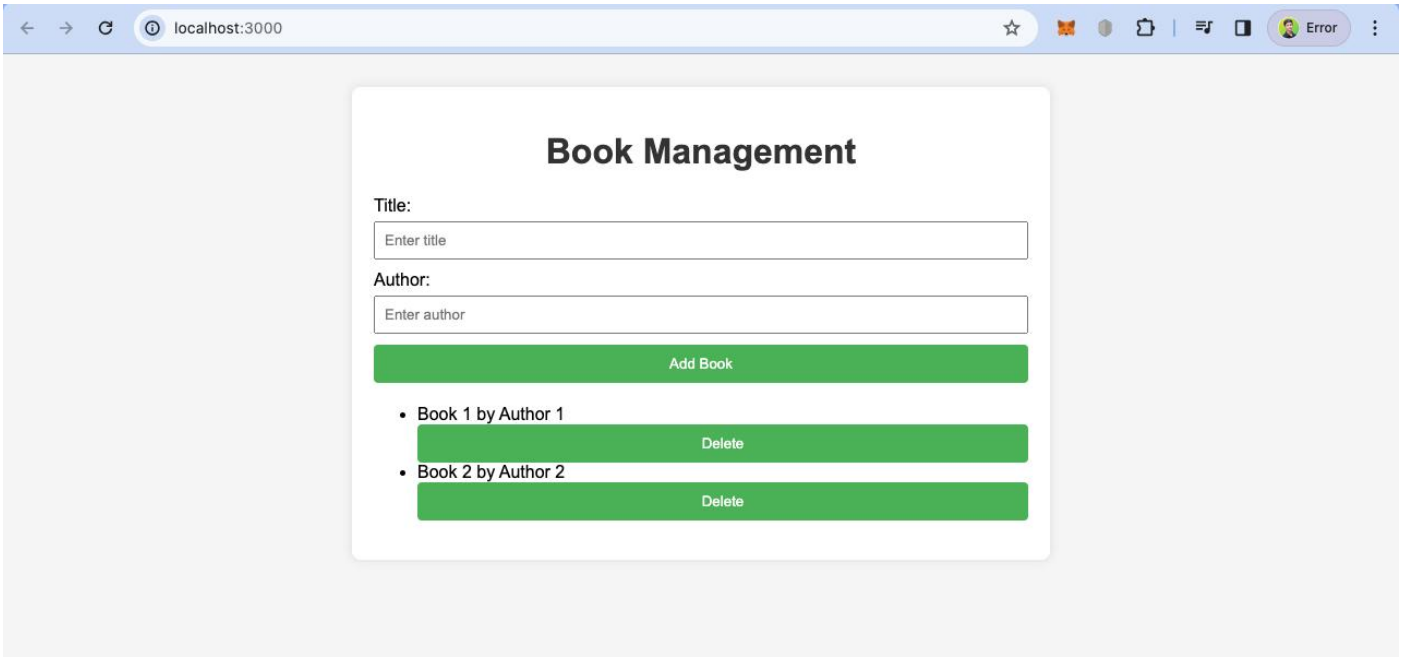
- **GET:**
  - **Purpose:** Retrieve data from the specified resource.
  - **Idempotent:** Yes (Multiple identical requests should have the same effect as a single request.)
  - **Safe:** Yes (Should not have the significance of taking an action other than retrieving data.)
- **POST:**
  - **Purpose:** Submit data to be processed to a specified resource.
  - **Idempotent:** No (Multiple identical requests may have different effects.)
  - **Safe:** No (May have side effects, such as creating a new resource.)
- **PUT:**

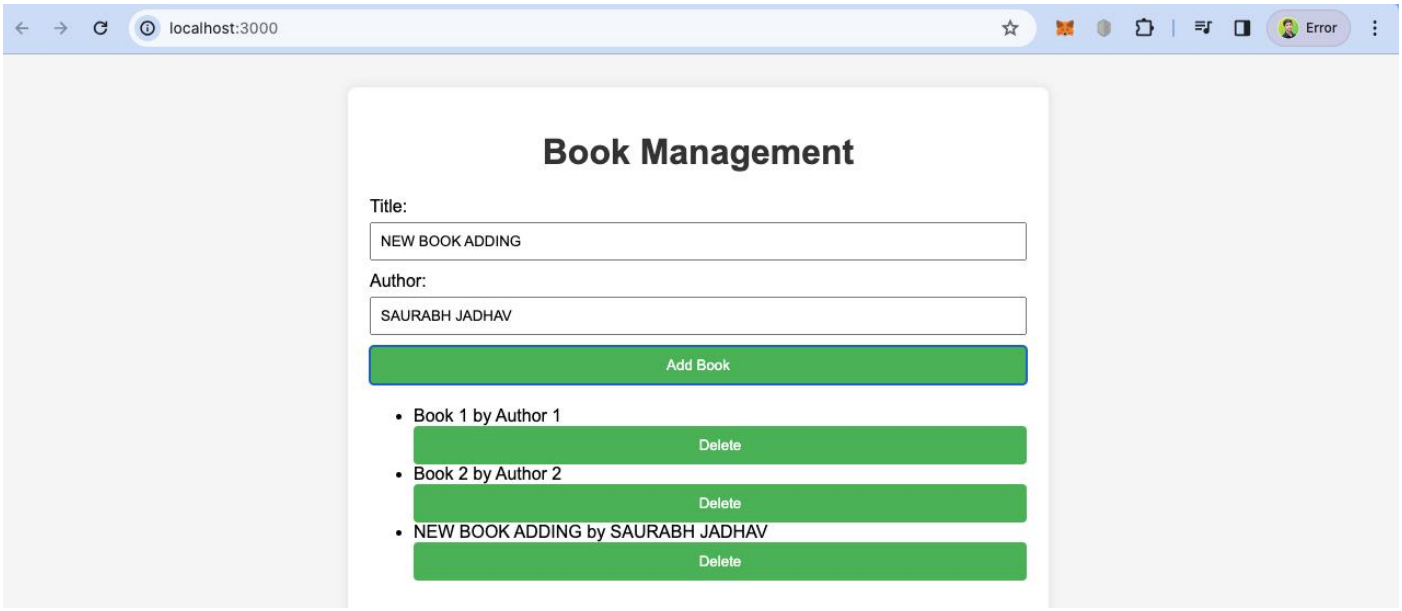
    - **Purpose:** Update a resource or create a new resource if it does not exist.
    - **Idempotent:** Yes (Multiple identical requests should have the same effect as a single request.)
    - **Safe:** No (May have side effects, such as creating a new resource.)
- **DELETE:**
    - **Purpose:** Request that a resource be removed.
    - **Idempotent:** Yes (Multiple identical requests should have the same effect as a single request.)
    - **Safe:** No (May have side effects, such as removing a resource.)
- **PATCH:**
    - **Purpose:** Apply partial modifications to a resource.
    - **Idempotent:** No (Multiple identical requests may have different effects.)
    - **Safe:** No (May have side effects, depending on the specific implementation.)
- **HEAD:**
    - **Purpose:** Same as GET but without the response body. Used to retrieve metadata about a resource without transferring the actual data.
    - **Idempotent:** Yes (Multiple identical requests should have the same effect as a single request.)
    - **Safe:** Yes (Should not have the significance of taking an action other than retrieving metadata.)
- **OPTIONS:**
    - **Purpose:** Describes the communication options for the target resource. It allows the client to determine the options and/or requirements associated with a resource, or the capabilities of a server, without implying a resource action.
    - **Idempotent:** Yes (Multiple identical requests should have the same effect as a single request.)

    - **Safe:** Yes (Should not have the significance of taking an action other than retrieving metadata.)
- **TRACE:**
    - **Purpose:** Echoes the received request to the client, useful for diagnostic purposes.
    - **Idempotent:** Yes (Multiple identical requests should have the same effect as a single request.)
    - **Safe:** Yes (Should not have the significance of taking an action other than retrieving diagnostic information.)

Implementation Screenshot:-

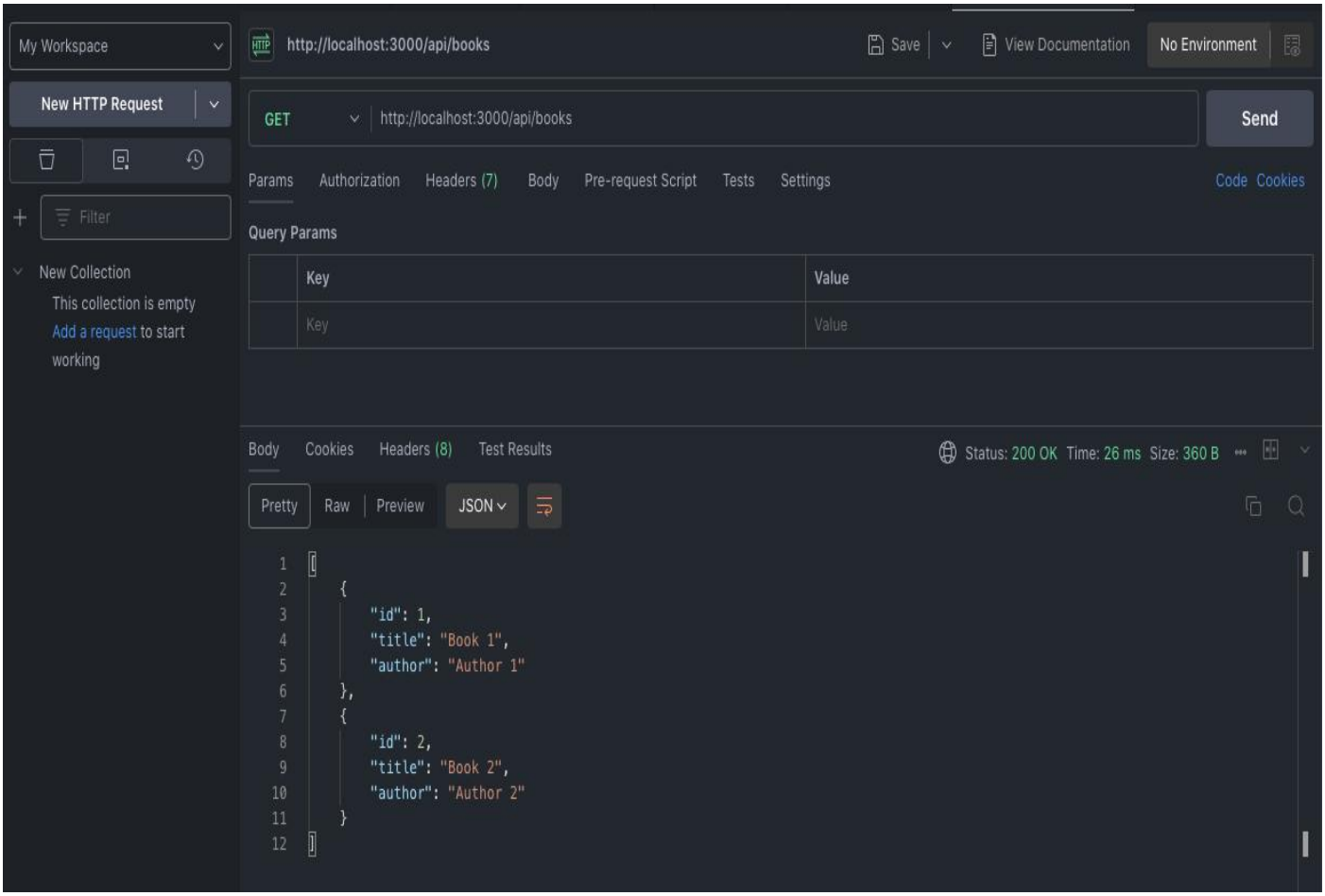
GET req





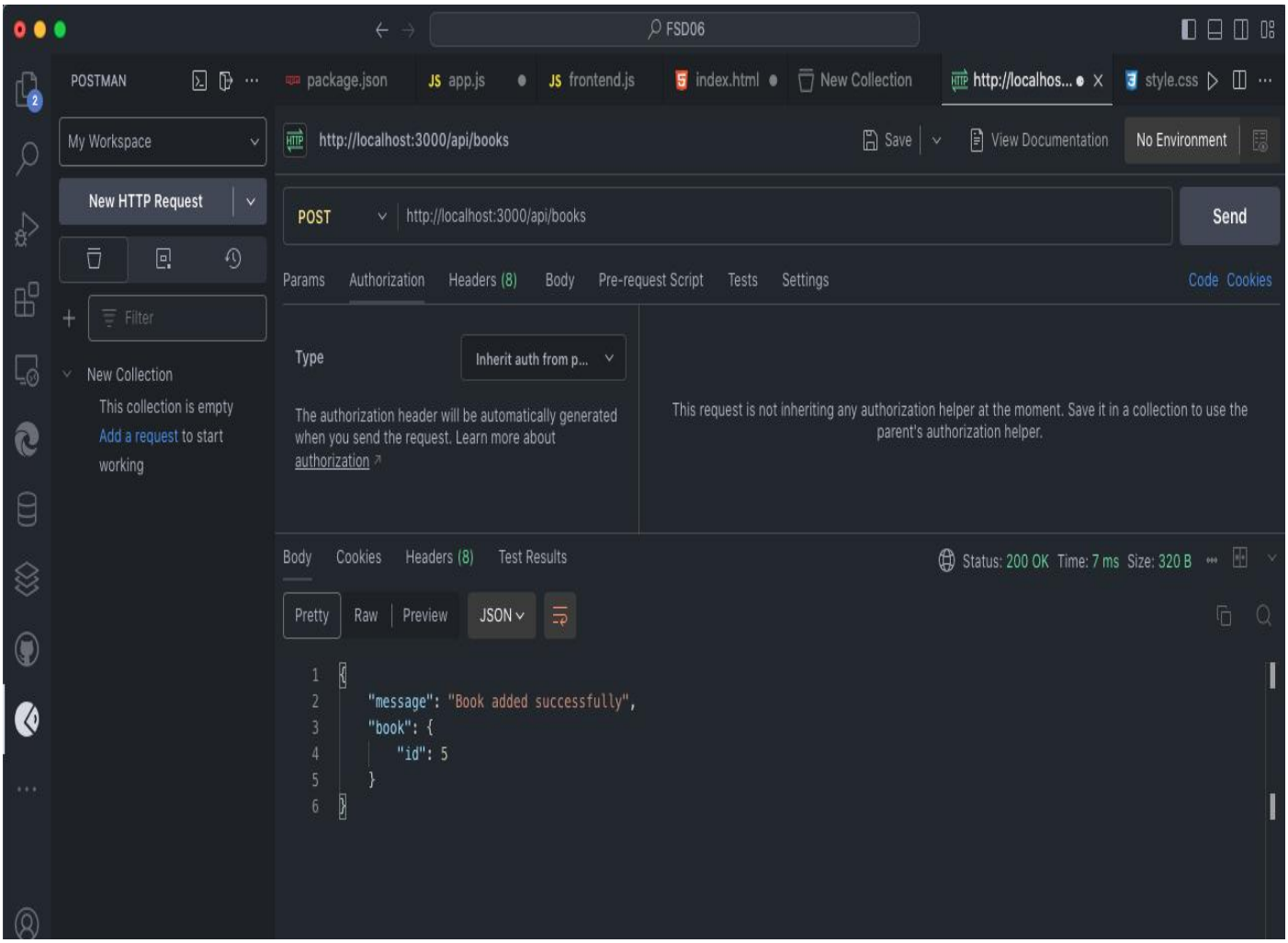
GET Request (Retrieve Books):

- **Description:** Fetches the list of books from the server.
- **Postman Request:** GET - http://localhost:3000/api/books



2)POST Request (Add a Book):

- **Description:** Adds a new book to the list on the server.
- **Postman Request:** POST - http://localhost:3000/api/books
  - **Body:** JSON - {"title": "New Book", "author": "New Author"}



DELETE Request (Delete a Book):

- **Description:** Deletes a specific book from the server.
- **Postman Request:** DELETE - http://localhost:3000/api/books/{bookId}
    - **Replace {bookId} with the actual ID of the book to delete.