Assignment: Build a RESTful API using Node.js and Express (100 marks)

1. Project Overview

This project is a simple REST API built using Express.js that allows us to **manage user information** stored in an in-memory JavaScript array.

Using this API, we can perform the following operations:

- View all users
- View a specific user by ID
- Add a new user
- Update an existing user
- Delete a user

All these operations are performed through HTTP methods like **GET, POST, PUT, and DELETE**, and each request returns meaningful responses through proper status codes.

2. Middleware Overview

Middlewares help us interact with every incoming request before it reaches the actual route.

a) Logging Middleware

Whenever we hit an API endpoint, this middleware records useful information such as:

- Which HTTP method was used (GET/POST/PUT/DELETE)
- Which URL was accessed
- What response status code was sent back

Why is it useful?

It helps us debug easily and monitor every request for performance and errors.

b) Validation Middleware

This middleware checks whether the request body contains **all required fields** before adding or updating a user.

- For adding a user, fields like **id, first name, last name, and hobby must be present**.
- For updating a user, fields like **first name, last name, and hobby must not be empty**.

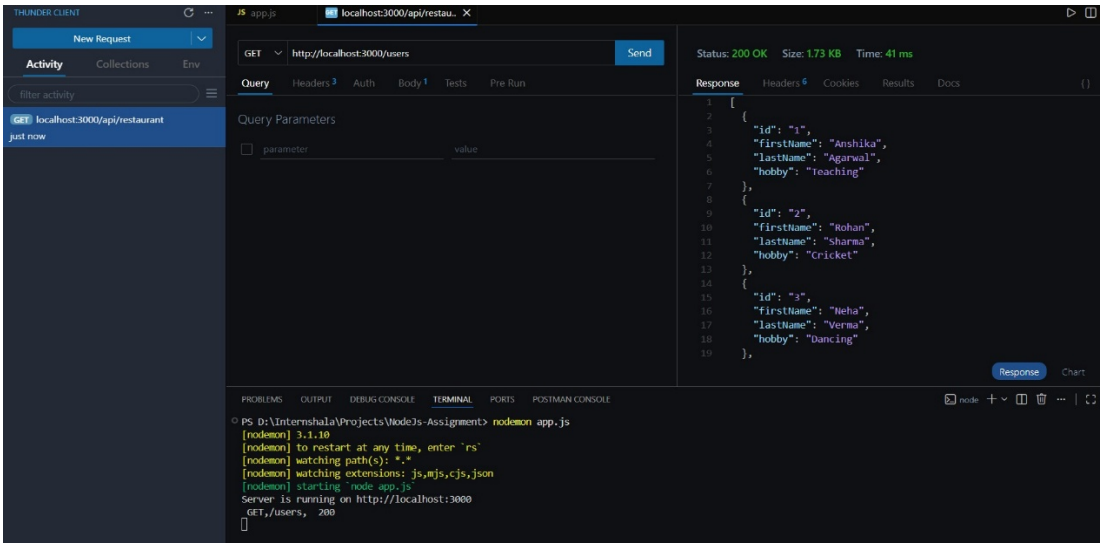
Why is this important?

It prevents storing incomplete or invalid user information. If required data is missing, a **400 (Bad Request)** response is sent with a meaningful message.

3. API Routes Explanation

GET /users — Fetch All Users

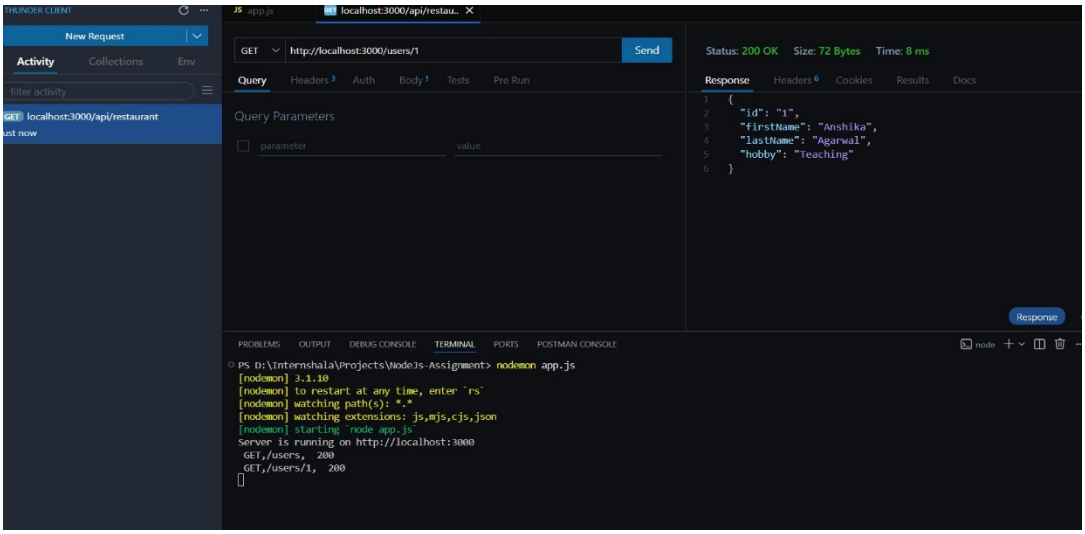
- This request retrieves the entire list of users.



- If users are available, the API returns them with a **200 (OK)** status.
- If the list doesn't contain any data, we send a message saying no users are available.

GET /users/:id — Fetch One User by ID

- This route allows us to find a single user based on their unique ID.

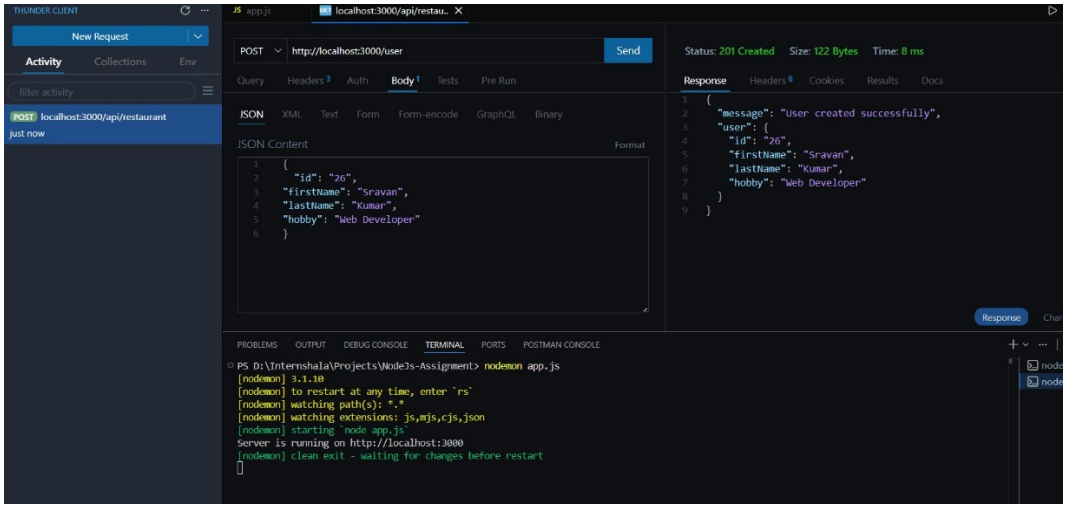


- If a matching user is found, their information is sent back with a **200 (OK)** status.
- If the user does not exist, the API returns **404 (Not Found)** with a clear message.

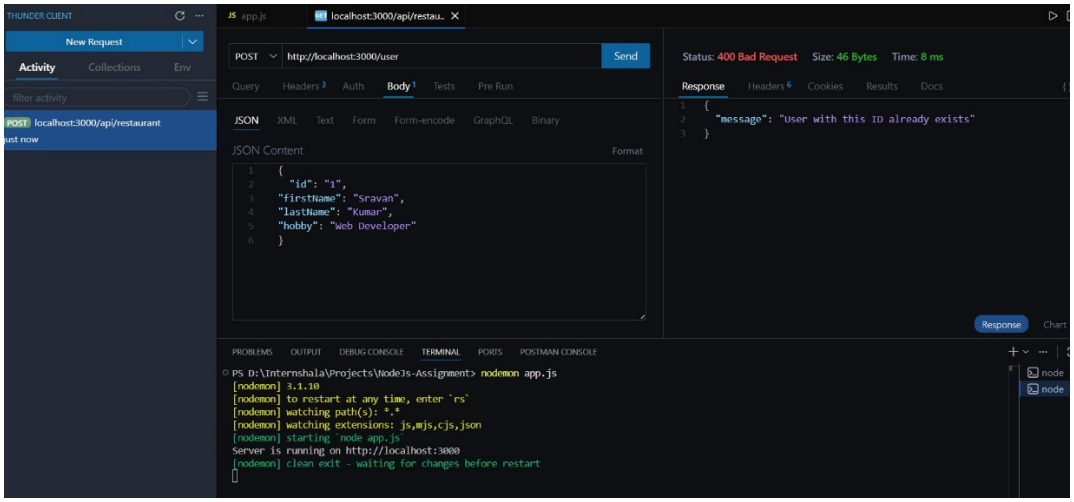
POST /user — Add a New User

- This route is used to create or register a new user.

- Validation middleware ensures all required fields are provided.



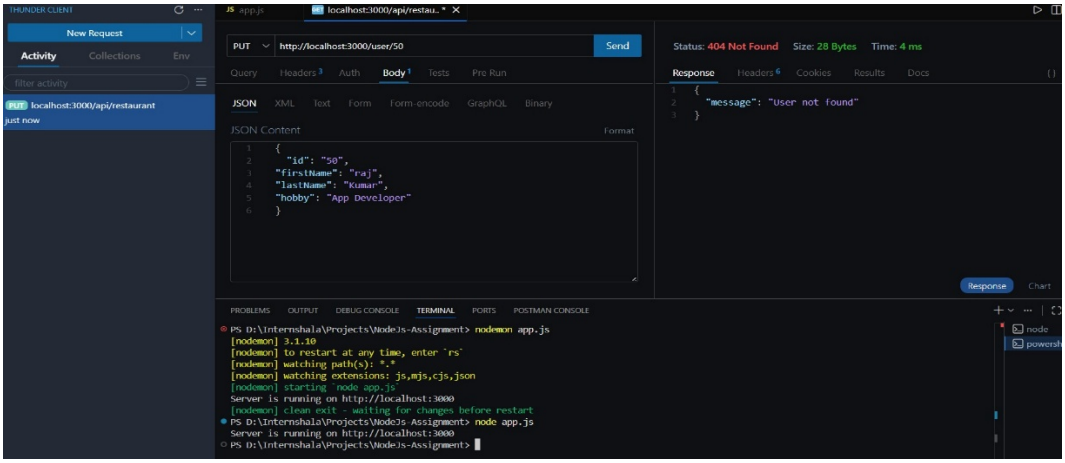
- If a user with the same ID already exists, we return a **400 (Bad Request)** message.



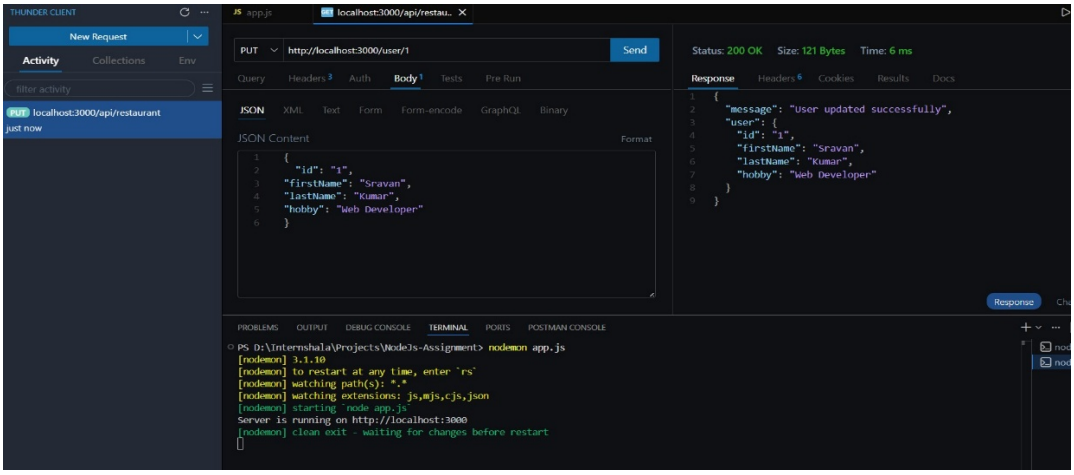
- If added successfully, a confirmation message with user details is returned with **201 (Created)**.

PUT /user/:id — Update Existing User

- This route updates the information of an existing user using their ID.



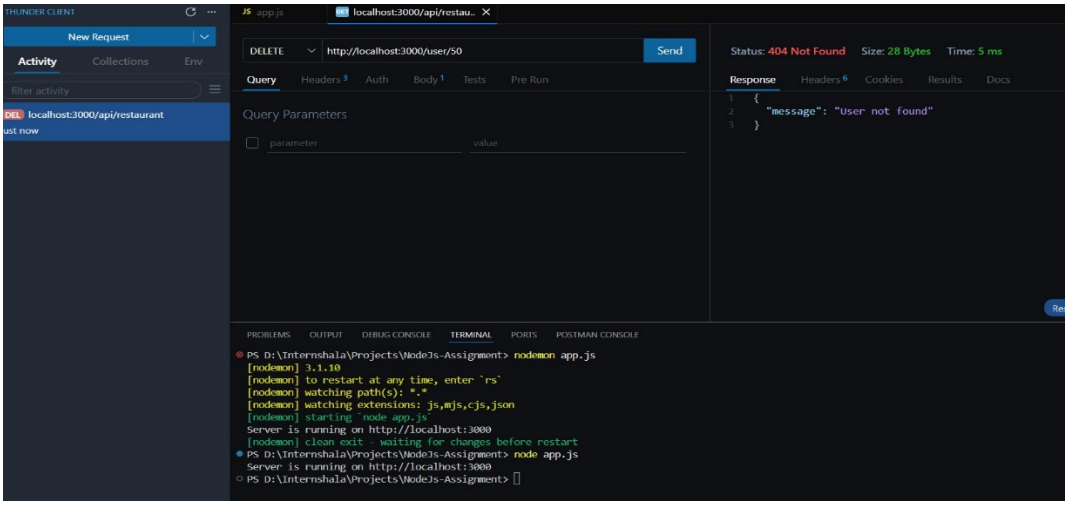
- If the user does not exist, the API sends a **404 (Not Found)** message.



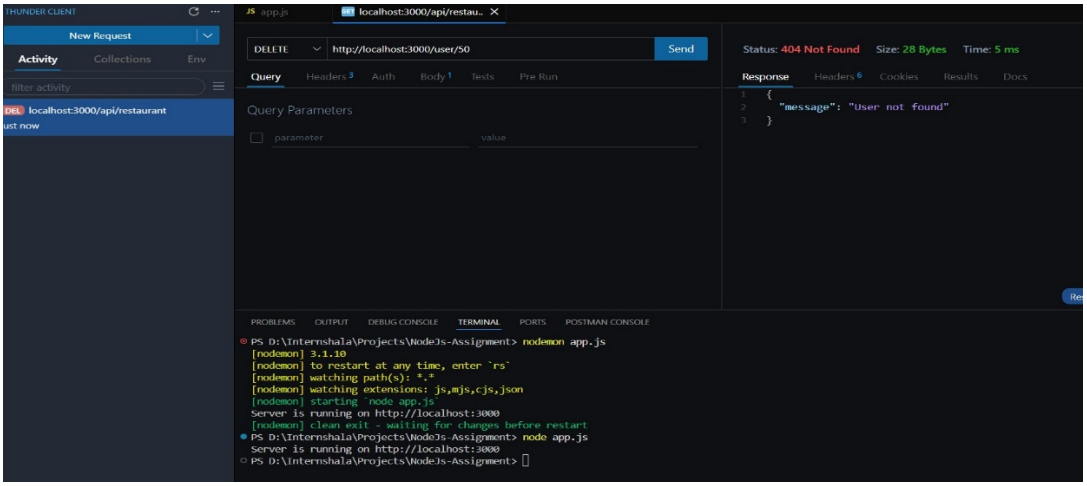
- If the information is valid and user exists, it updates the data and returns **200 (OK)** with the updated result.
- Middleware ensures no empty data is passed while updating.

DELETE /user/:id — Remove a User

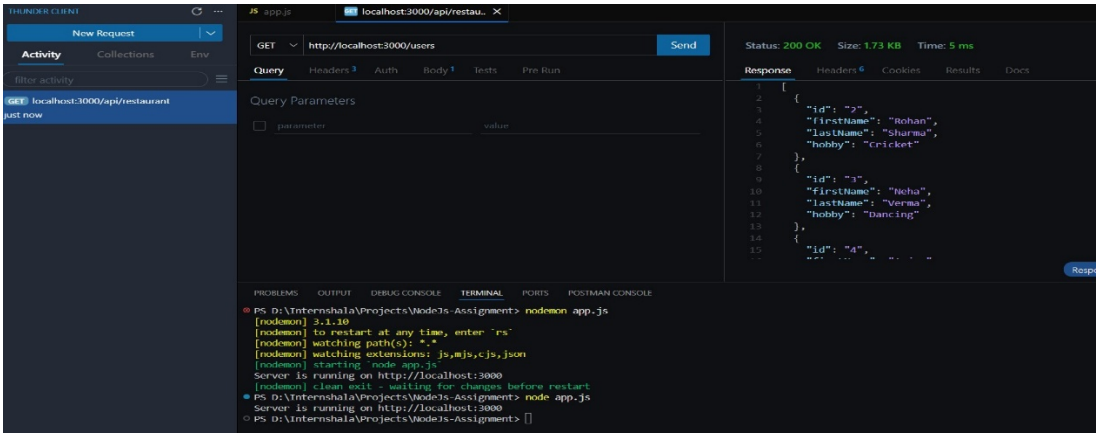
- This request removes a user.
- If the user ID doesn't exist, the API returns **404 (Not Found)**.



- If the user exists, they are deleted and a **200 (OK)** confirmation message is returned.



We can find user after deletion.



4. Error Handling Summary

A global error handler makes sure:

- The server doesn't crash if anything goes wrong internally.
- All errors return understandable messages instead of raw system errors.

- Proper status codes such as **500 (Server Error)** or specific ones like **404, 400, 403**, etc., are sent.