Titles

Project Name – Build a RESTful API using Node.js and Express

Build by – Shashikant Sahu

Batch – 1st March 2025

Table of Contents

Introduction

This assignment demonstrates the creation of a RESTful API using Node.js and Express.js. The objective is to develop a server-side application that allows basic CRUD (Create, Read, Update, Delete) operations for managing user data. The project focuses on the core concepts of routing, middleware, HTTP methods, status codes, and error handling, using in-memory storage for simplicity.

The API includes endpoints for:

- Retrieving all users
- Retrieving a user by ID
- Adding a new user
- Updating an existing user
- Deleting a user

Creativity and Presentation

This assignment ensures that each responsibility is clearly separated, making the codebase easy to understand and maintain.

Technologies Used

To build and structure this RESTful API project efficiently, the following technologies were used:

- Node.js
  A JavaScript runtime environment used to build the server-side application. It allows writing backend logic using JavaScript.
- Express.js
  A fast and minimal web framework for Node.js that simplifies routing, handling HTTP requests, and middleware integration.
- JavaScript (ES6+)
  The core programming language used throughout the project, leveraging modern syntax for cleaner and more efficient code.
- ThunderClient *(API Testing Tool)*
  Used to test all API endpoints (GET, POST, PUT, DELETE) to ensure proper functionality, validation, and error handling.
- VS Code *(Code Editor)*
  A powerful and customizable editor used to write and manage project files effectively.

Code

Index.js

```
const express = require("express");
const app = express();
const port = 3000;

// Middleware to parse JSON
app.use(express.json());

// Sample User Object Structure:
let users = [
  {
    id: "1",
    firstName: "Anshika",
    lastName: "Agarwal",
    hobby: "Teaching",
  },
  {
    id: "2",
    firstName: "shashikant",
    lastName: "sahu",
    hobby: "cricket",
  },
  {
    id: "3",
    firstName: "vivek",
    lastName: "kumar",
    hobby: "nothing",
  },
];

// Middleware for logging requests
app.use((req, res, next) => {
  res.on("finish", () => {
    console.log(`${req.method} ${req.originalUrl} - ${res.statusCode}`);
```

```
    console.log("next middleware");
  });
  next();
});

// validation middleware
function validateUser(req, res, next) {
  const { firstName, lastName, hobby } = req.body;
  if (!firstName || !lastName || !hobby) {
    return res.status(400).json({ error: "All fields are required" });
  }
  next();
}

// GET /users – Fetch the list of all users.
app.get("/users", (req, res) => {
  res.status(200).json(users);
});

// GET /users/:id – Fetch details of a specific user by ID.
app.get("/users/:id", (req, res) => {
  const userId = req.params.id; //for dynamic routing.
  const user = users.find((user) => user.id === userId);
  if (!user) {
    return res.status(404).json({ error: "User not found with this id" });
  }
  res.status(200).json(user);
});

// POST /user – Add a new user.
app.post("/user", validateUser, (req, res) => {
  const lastId = users.length > 0 ? parseInt(users[users.length - 1].id) : 0;
  const newUser = {
    id: (lastId + 1).toString(),
    firstName: req.body.firstName,
    lastName: req.body.lastName,
    hobby: req.body.hobby,
  };
```

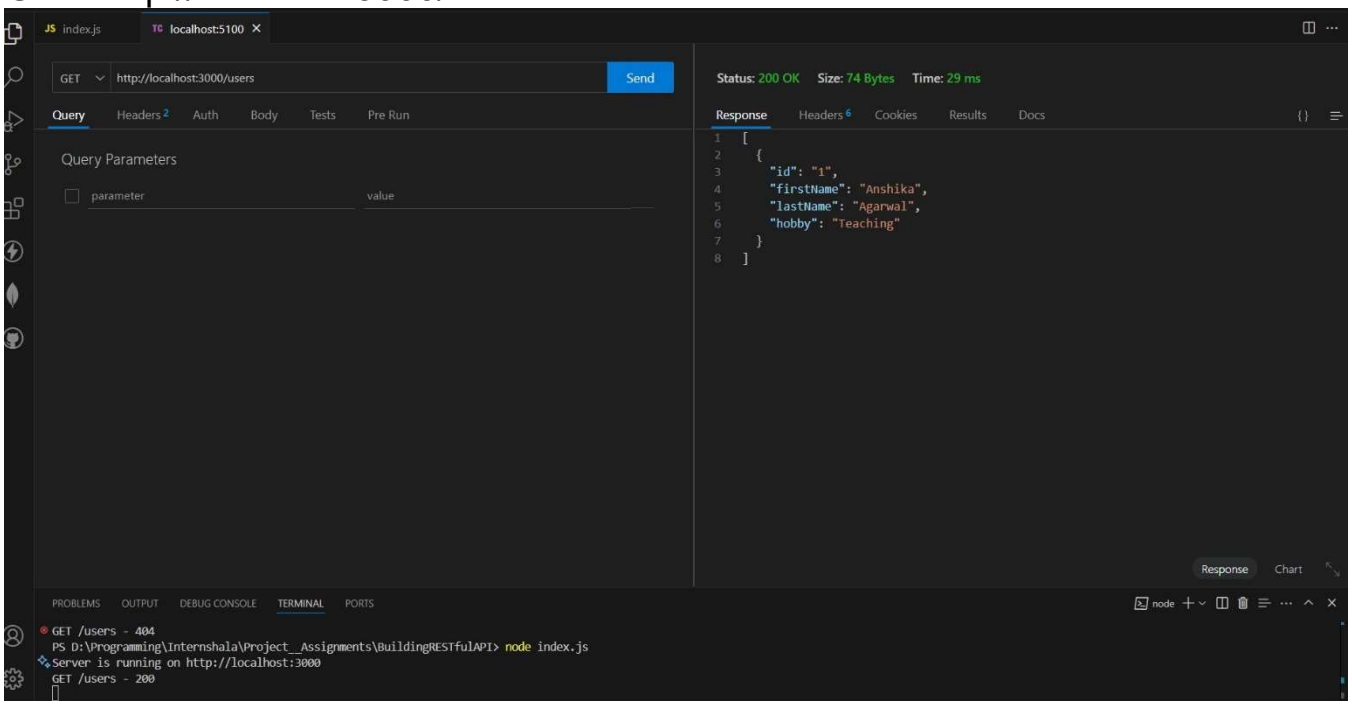
```
  users.push(newUser);
  res.status(201).json(newUser);
});

// PUT /user/:id – Update details of an existing user.
app.put("/user/:id", validateUser, (req, res) => {
  const userId = req.params.id;
  const userIndex = users.findIndex((user) => user.id === userId);
  if (userIndex === -1) {
    return res.status(404).json({ error: "User not found with this id" });
  }
  users[userIndex] = { id: userId, ...req.body };
  res.status(200).json(users[userIndex]);
});

// DELETE /user/:id – Delete a user by ID.
app.delete("/user/:id", (req, res) => {
  const userId = req.params.id;
  const userIndex = users.findIndex((user) => user.id === userId);
  if (userIndex === -1) {
    return res.status(404).json({ error: "User not found with this id" });
  }
  const deletedUser = users.splice(userIndex, 1);
  res.status(200).json(deletedUser[0]);
});

//start server
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

Package.json

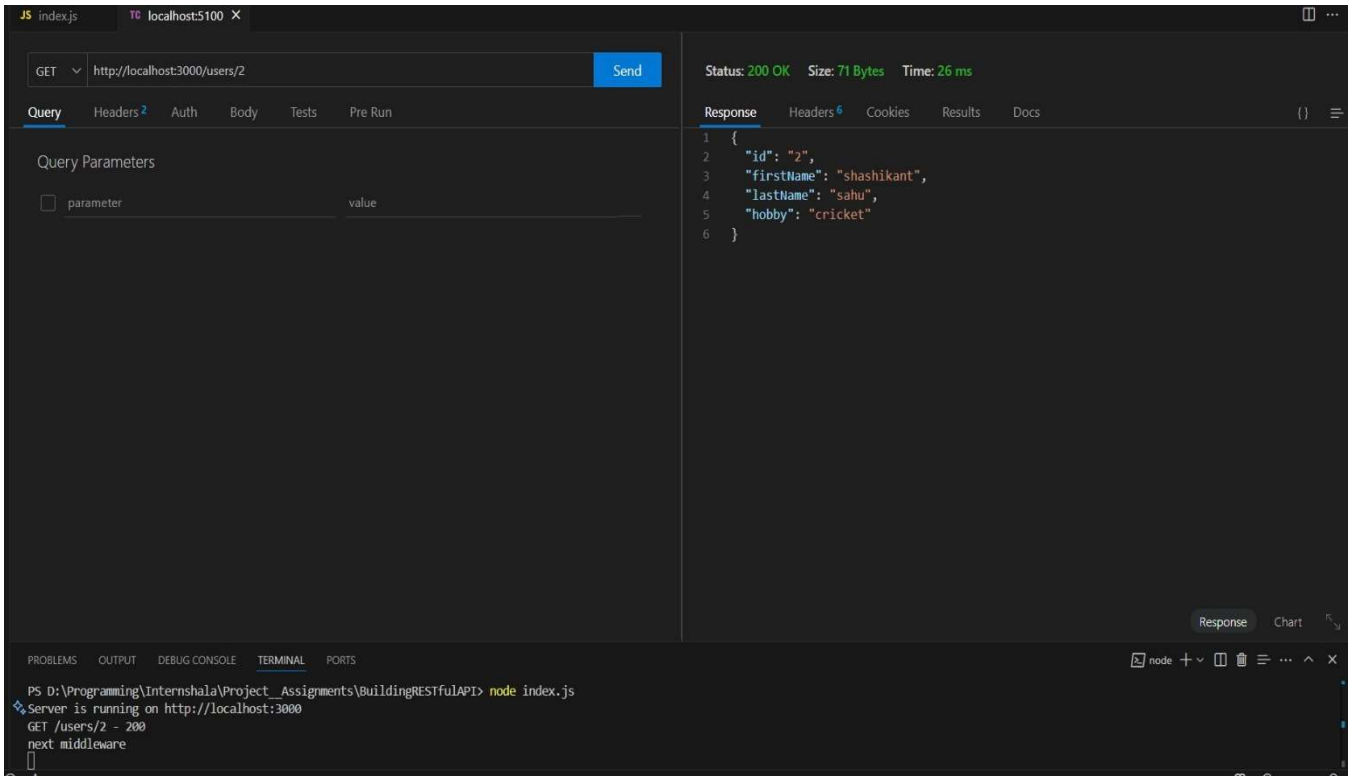
```
{
  "name": "buildingrestfulapi",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^5.1.0"
  }
}
```

Screenshot :-

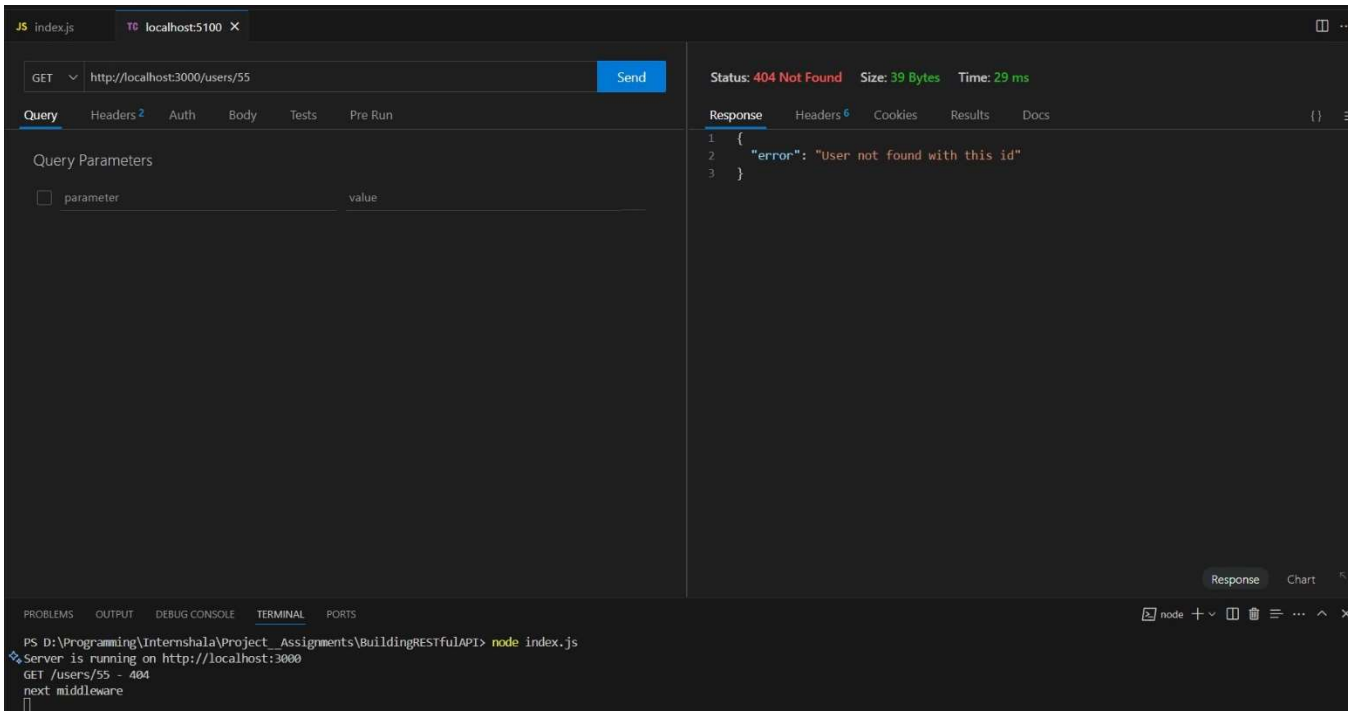
GET- http://localhost:3000/users



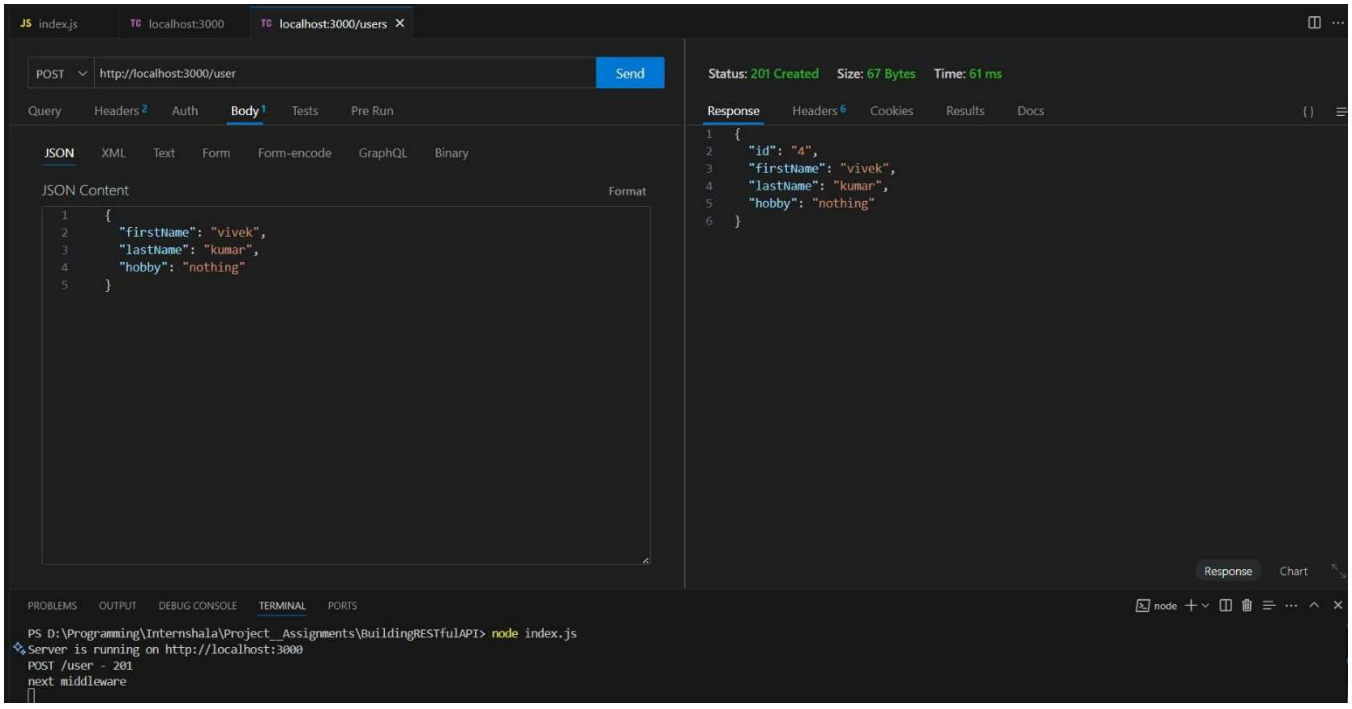
GET- http://localhost:3000/users/2



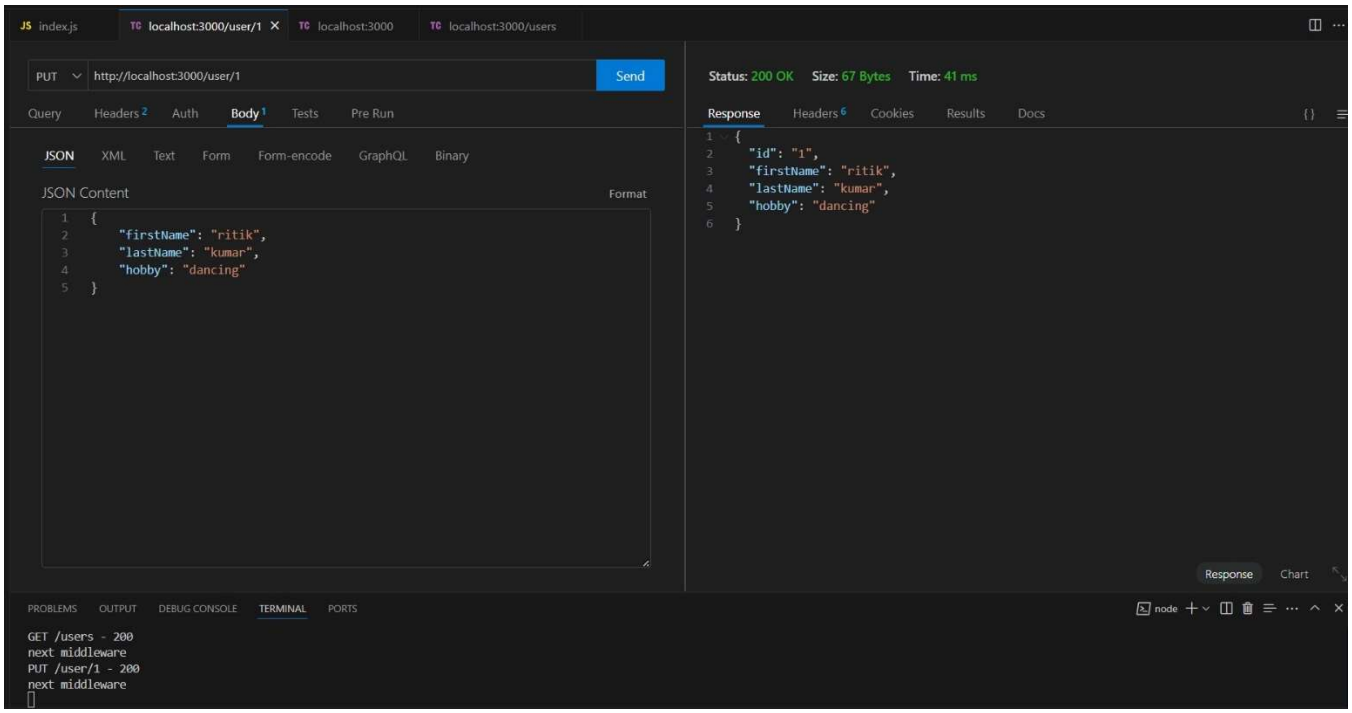
GET - http://localhost:3000/users/55



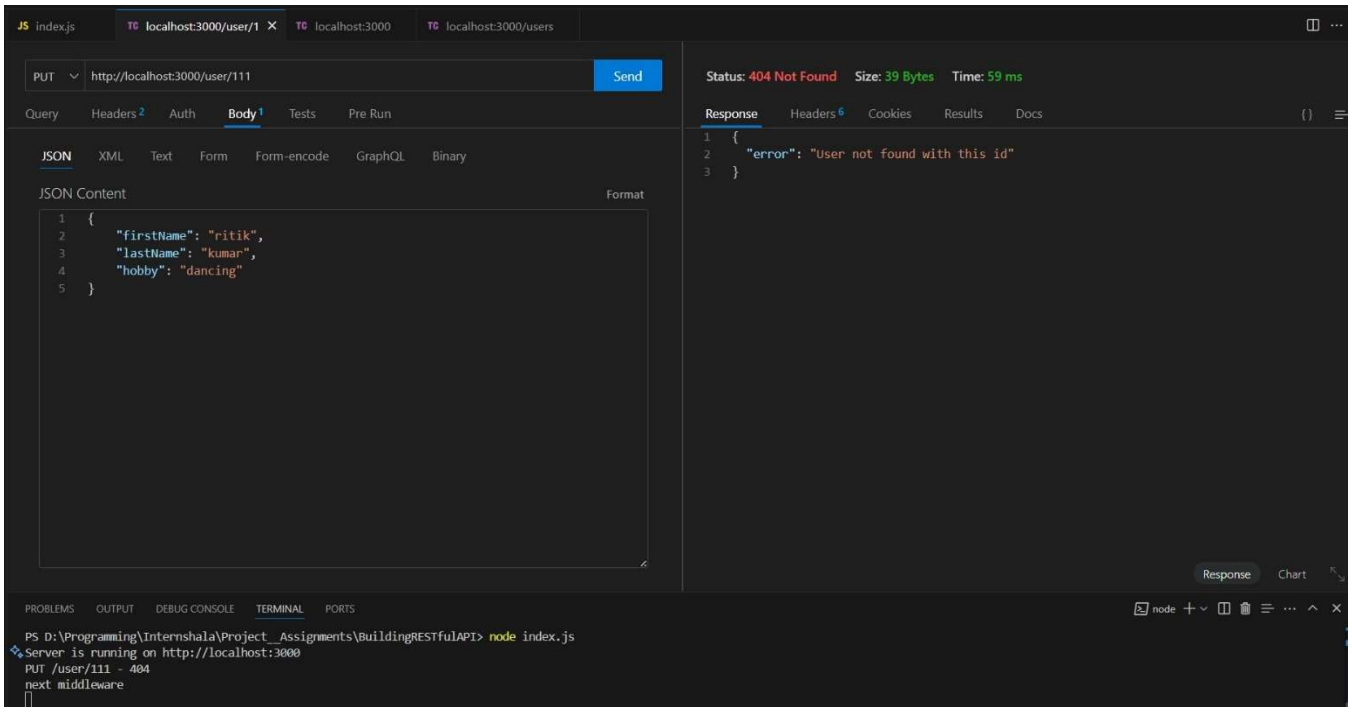
POST - http://localhost:3000/user



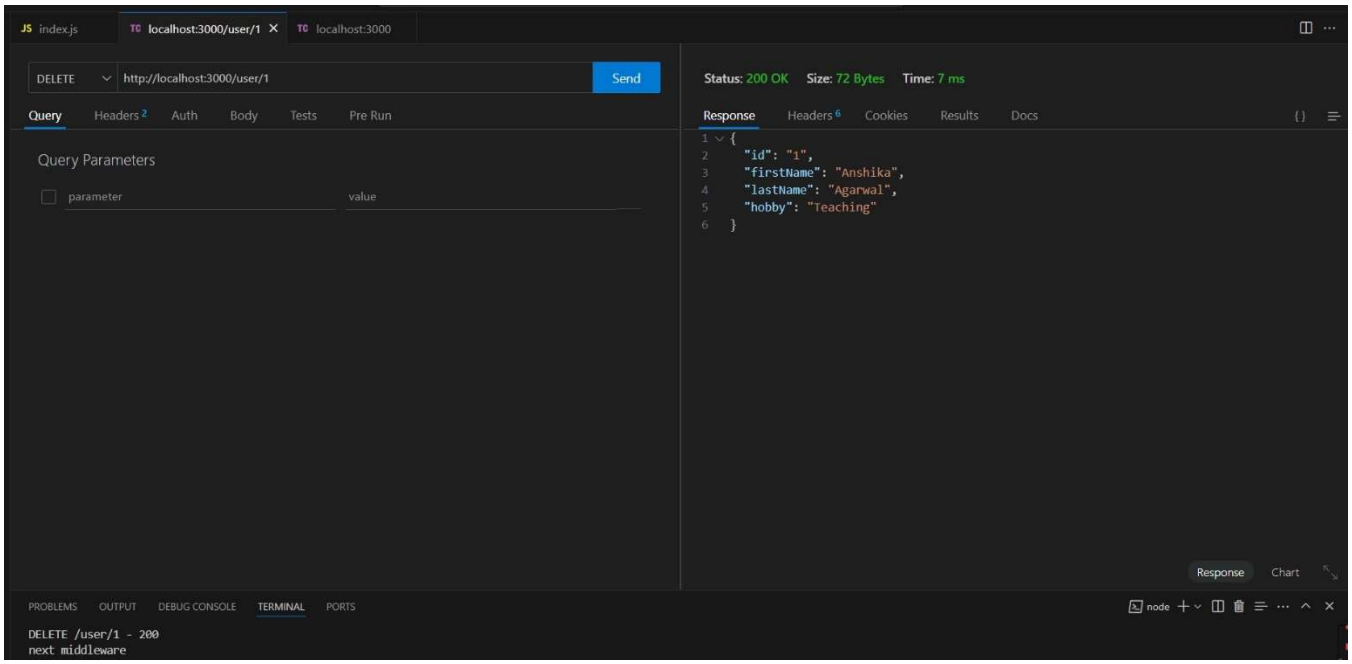
PUT - http://localhost:3000/user/1



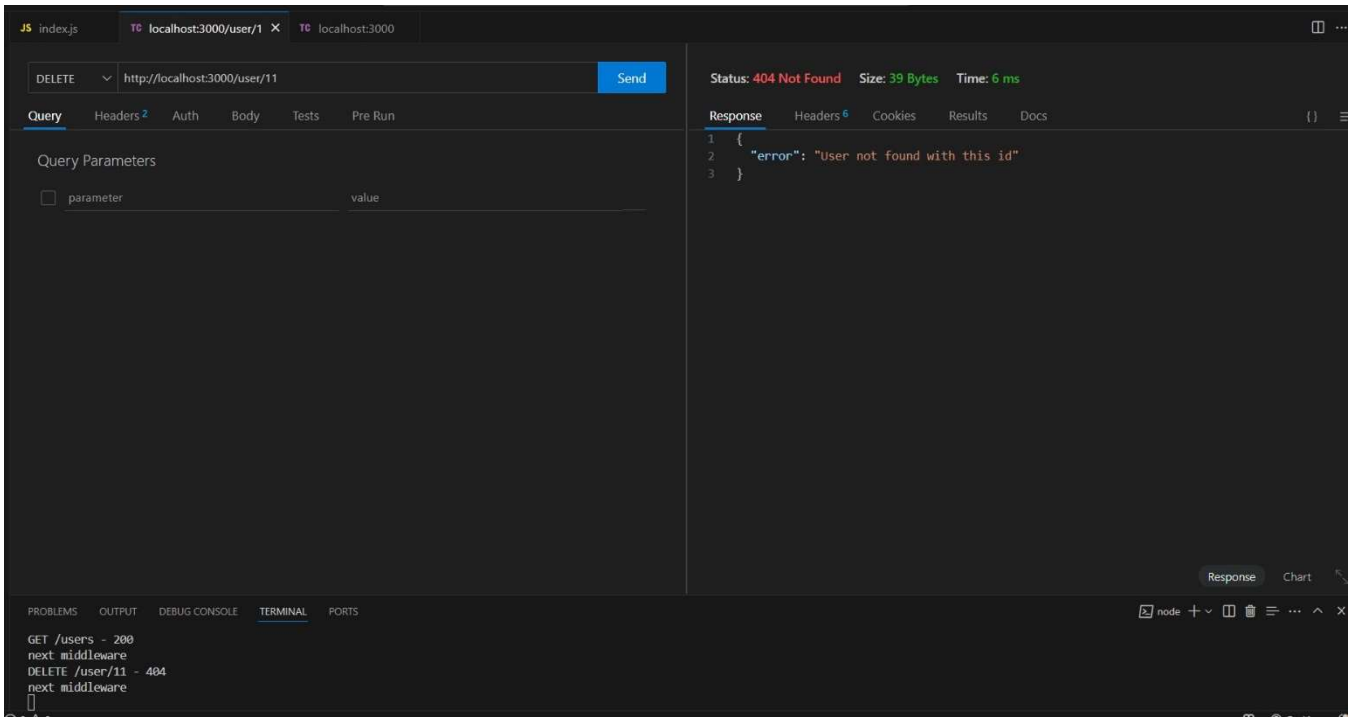
PUT - http://localhost:3000/user/111



DELETE - http://localhost:3000/user/1



DELETE - http://localhost:3000/user/11



GitHub Link:- **https://github.com/shashikant021/BuildingRESTfulAPI**