

**BACKEND LAB**

NAME :- Muskan Mittal

SAP ID :- 500124380

SPECIALIZATION:- Fullstack

BATCH :- 1

EXPERIMENT-2

Create an API for managing a list of items (e.g., a to-do list) using HTTP modules and express framework.

1. Using HTTP module:-

FILE-1: exp-2-http.js

const http = require("http");

let items = [];

let idCounter = 1;

function sendJSON(res, statusCode, data) {

  res.writeHead(statusCode, { "Content-Type": "application/json" });

  res.end(JSON.stringify(data));

}

const server = http.createServer((req, res) => {

  if (req.url === "/items" && req.method === "GET") {

    sendJSON(res, 200, items);

  }

  else if (req.url === "/items" && req.method === "POST") {

    let body = "";

    req.on("data", chunk => {

      body += chunk.toString();

    });

    req.on("end", () => {

      const parsed = JSON.parse(body);

      const newItem = { id: idCounter++, title: parsed.title, completed: false };

      items.push(newItem);

      sendJSON(res, 201, newItem);

    });

  }

  else if (req.url.startsWith("/items/") && req.method === "PUT") {

    const id = parseInt(req.url.split("/")[2]);

    let body = "";

    req.on("data", chunk => { body += chunk.toString(); });

    req.on("end", () => {

      const parsed = JSON.parse(body);

      const item = items.find(i => i.id === id);

      if (!item) return sendJSON(res, 404, { error: "Item not found" });

      if (parsed.title !== undefined) item.title = parsed.title;

      if (parsed.completed !== undefined) item.completed = parsed.completed;

      sendJSON(res, 200, item);

    });

  }

  else if (req.url.startsWith("/items/") && req.method === "DELETE") {

    const id = parseInt(req.url.split("/")[2]);

    const index = items.findIndex(i => i.id === id);

    if (index === -1) return sendJSON(res, 404, { error: "Item not found" });

    const deleted = items.splice(index, 1);

    sendJSON(res, 200, deleted[0]);

  }

  else {

    sendJSON(res, 404, { error: "Not found" });

  }

});

server.listen(3000, () => {

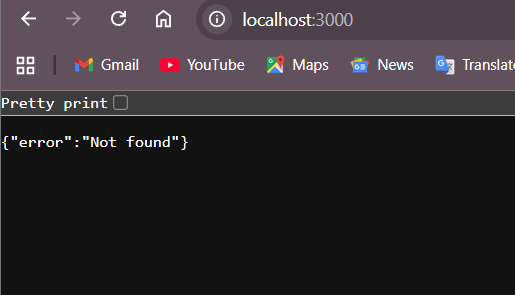
  console.log("HTTP Server running at http://localhost:3000");

});

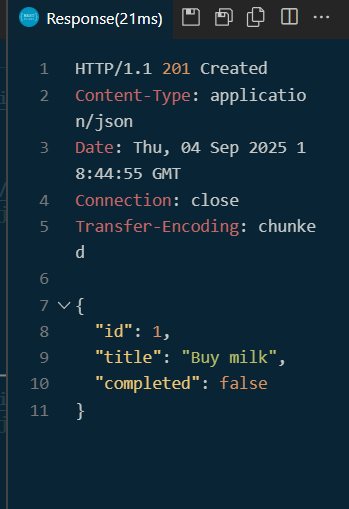
FILE-2: test.http



* Here, I had used vs code extension to run the operations
* So, in first condition if I run this code in terminal then at local host 3000 it shows error of not found



* Now, we move to file-2 in which I get a separate button of send reques for each command once I click that button then I got my desirable result.
* Eg :- when I send request for post



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Test Case** | **HTTP Method** | **Endpoint (URL)** | **Request Body** | **Expected Response** | **Status Code** | | 1. Get empty list | GET | /items | – | [] (empty array) | 200 OK | | 2. Add a new item | POST | /items | { "title": "Buy Milk" } | { "id": 1, "title": "Buy Milk", "completed": false } | 201 Created | | 3. Add another item | POST | /items | { "title": "Do Homework" } | { "id": 2, "title": "Do Homework", "completed": false } | 201 Created | | 4. Get all items | GET | /items | – | [ { "id": 1, "title": "Buy Milk", "completed": false }, { "id": 2, "title": "Do Homework", "completed": false } ] | 200 OK | | 5. Invalid POST (missing title) | POST | /items | { } | { "error": "Title is required" } | 400 Bad Request | | 6. Invalid JSON in POST | POST | /items | { title: Buy Milk } (invalid JSON) | { "error": "Invalid JSON" } | 400 Bad Request | | 7. Unknown route | GET | /random | – | { "error": "Not found" } | 404 Not Found | |

1. Using express :-

FILE-1: exp-2-express.js

const express = require("express");

const app = express();

app.use(express.json()); *// JSON body parse karne ke liye*

let items = []; *// memory storage*

let idCounter = 1;

app.get("/items", (req, res) => {

  res.json(items);

});

app.post("/items", (req, res) => {

  const { title } = req.body;

  if (!title) {

    return res.status(400).json({ error: "Title is required" });

  }

  const newItem = { id: idCounter++, title, completed: false };

  items.push(newItem);

  res.status(201).json(newItem);

});

app.put("/items/:id", (req, res) => {

  const id = parseInt(req.params.id);

  const { title, completed } = req.body;

  const item = items.find(it => it.id === id);

  if (!item) {

    return res.status(404).json({ error: "Item not found" });

  }

  if (title !== undefined) item.title = title;

  if (completed !== undefined) {

    if (typeof completed !== "boolean") {

      return res.status(400).json({ error: "Completed must be boolean" });

    }

    item.completed = completed;

  }

  res.json(item);

});

app.delete("/items/:id", (req, res) => {

  const id = parseInt(req.params.id);

  const index = items.findIndex(it => it.id === id);

  if (index === -1) {

    return res.status(404).json({ error: "Item not found" });

  }

  items.splice(index, 1);

  res.json({ message: "Item deleted successfully" });

});

app.use((req, res) => {

  res.status(404).json({ error: "Not found" });

});

app.listen(3000, () => {

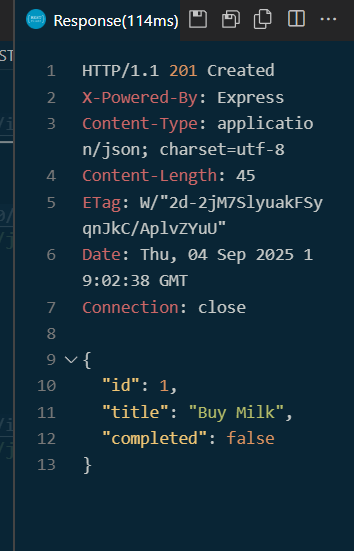
  console.log("Express server running on http://localhost:3000");

});

FILE-2: tests.express



* Also to run this, I performed the same procedure as mentioned above in http module



TEST CASES:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Test Case** | **HTTP Method** | **Endpoint (URL)** | **Request Body** | **Expected Response** | **Status Code** | | 1. Get empty list | GET | /items | – | [] (empty array) | 200 OK | | 2. Add a new item | POST | /items | { "title": "Buy Milk" } | { "id": 1, "title": "Buy Milk", "completed": false } | 201 Created | | 3. Add another item | POST | /items | { "title": "Do Homework" } | { "id": 2, "title": "Do Homework", "completed": false } | 201 Created | | 4. Get all items | GET | /items | – | [ { "id": 1, "title": "Buy Milk", "completed": false }, { "id": 2, "title": "Do Homework", "completed": false } ] | 200 OK | | 5. Update item title | PUT | /items/1 | { "title": "Buy Bread" } | { "id": 1, "title": "Buy Bread", "completed": false } | 200 OK | | 6. Update item completion | PUT | /items/1 | { "completed": true } | { "id": 1, "title": "Buy Bread", "completed": true } | 200 OK | | 7. Invalid update (completed not boolean) | PUT | /items/1 | { "completed": "yes" } | { "error": "Completed must be boolean" } | 400 Bad Request | | 8. Delete item | DELETE | /items/1 | – | { "message": "Item deleted successfully" } | 200 OK | | 9. Delete non-existing item | DELETE | /items/99 | – | { "error": "Item not found" } | 404 Not Found | | 10. Unknown route | GET | /random | – | { "error": "Not found" } | 404 Not Found | |