

## Group Members:

* Smile Smile(200563908)
* Dev Dev(200562142)
* Robin Robin(200571744)

**Assignment Instructions:**

**You will create an express application using 3 JavaScript files:**

1. Install the full Express required and recommended dependencies:

* Install Express
* Install Nodemon

2. Add a folder named “**data**” inside the project folder

3. Inside the “data” folder add a **JSON file** with **at least 6 objects**

4. Add/Create 3 different .js files with any label/title you prefer but each file name for each JavaScript file will end with a number: 1, 2, and 3 accordingly

5. Follow the instructions to write the code for each file as explained below

6. Add full comments to explain your code/lines briefly and clearly

7. You will need to use **nodemon** to run your .js files

8. You will need to install **Postman in** this lab (for JS file#3)

**Assignment Submission Materials:**

**1. The PDF file that contains the screenshots explained below:**

* Use **MS-Word** to collect all the assignment images (screenshots), put and arrange them all in one professional document then convert it to a PDF file to be uploaded/submitted. Please consider the following screenshots (images):
* The folder structure inside IntelliJ IDEA/Visual Studio Code or any other Editor that you are using that shows the 2 listed files above
* The VS Code embedded terminal window that shows the current working directory (the path of your folder)
* The PDF file should prepared in a professional way with a cover page

**2. The link to your GitHub repository where you have your assignment (project) uploaded**

**1. Introduction**

The **Train Management System** is a simple CRUD (Create, Read, Update, Delete) API built using **Node.js and Express.js**. This project demonstrates how to manage train details using API endpoints. Users can fetch train details, add new trains, update existing trains, and delete trains using HTTP requests.

**2. Technologies Used**

* **Node.js** – JavaScript runtime for backend development
* **Express.js** – Web framework for handling routes and requests
* **Postman** – API testing tool
* **JSON** – Data storage format
* **Nodemon** – Auto-restart server during development

**Step1 :** **Install Node.js**

Node.js is required for this project because it provides the runtime environment necessary to execute JavaScript code outside of a web browser.

Runs JavaScript on the server-side  
Required for Express.js and npm package installation  
Handles HTTP requests for APIs  
Supports non-blocking, asynchronous programming  
Essential for local development and testing

**Step2: Create Project Folder**

We will create a folder named as Express-crud-app and after that we will Initialize Node.js Project to create a package.json file with default configurations.

**Step 3: Install Required Packages**

In this step we will Install Express.js and Install Nodemon (for automatic server restart)

**Step 4: Create Project Files**

In step 4 ,we will create all the essential files to get the output and the project structure is as following

**express-crud-app/**

**├─ data/**

**│ └── train.json # JSON file containing train data**

**├── task1.js # Displays group names**

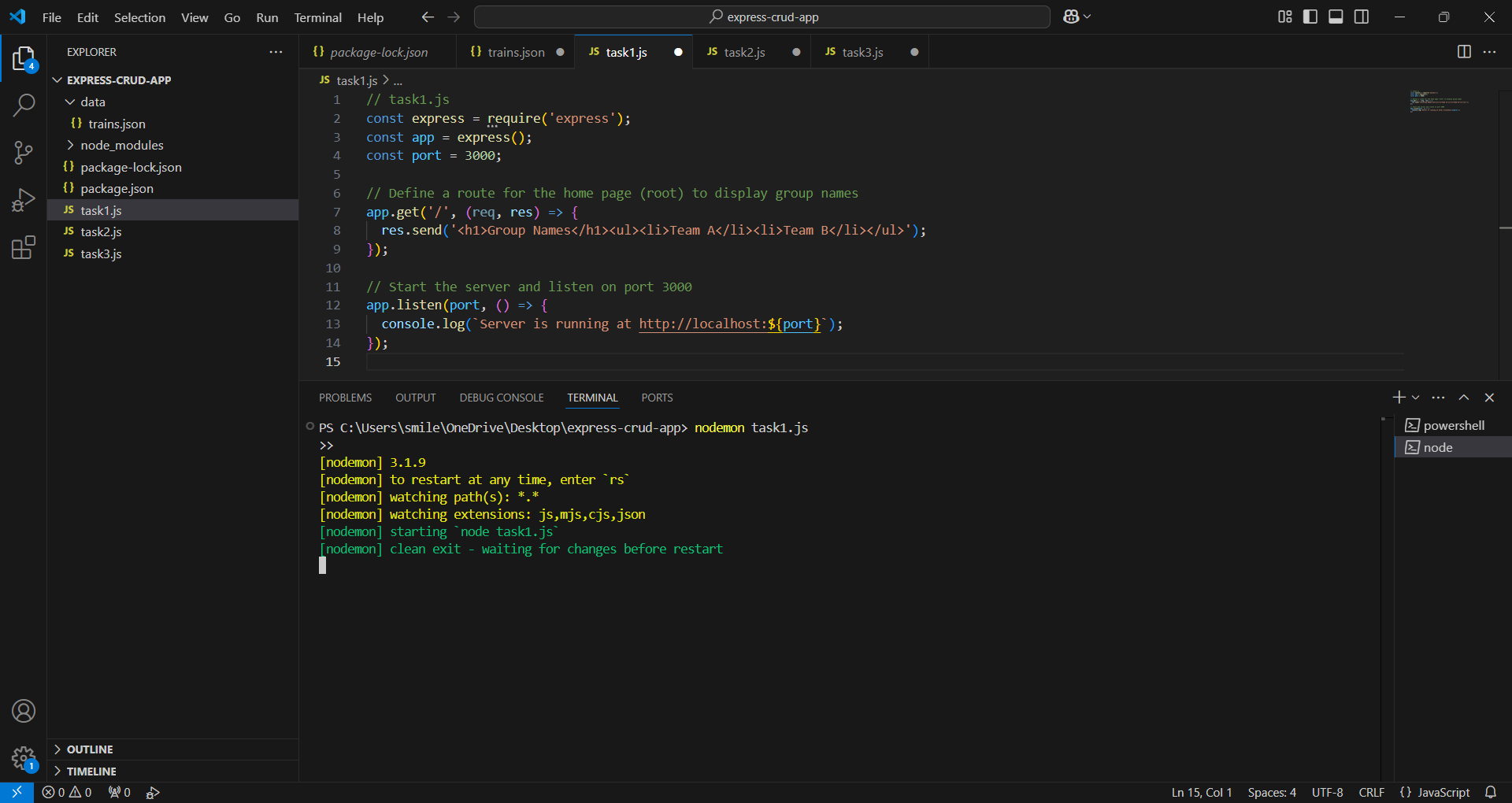
**├── task2.js # Fetches train data from JSON file**

**├── task3.js # CRUD operations for trains**

**├── package.json # Project dependencies**

**├── package-lock.json # Dependency lock file**

**└── README.md # Project documentation**



Step 5: Our **Express.js CRUD application** consists of three JavaScript files:

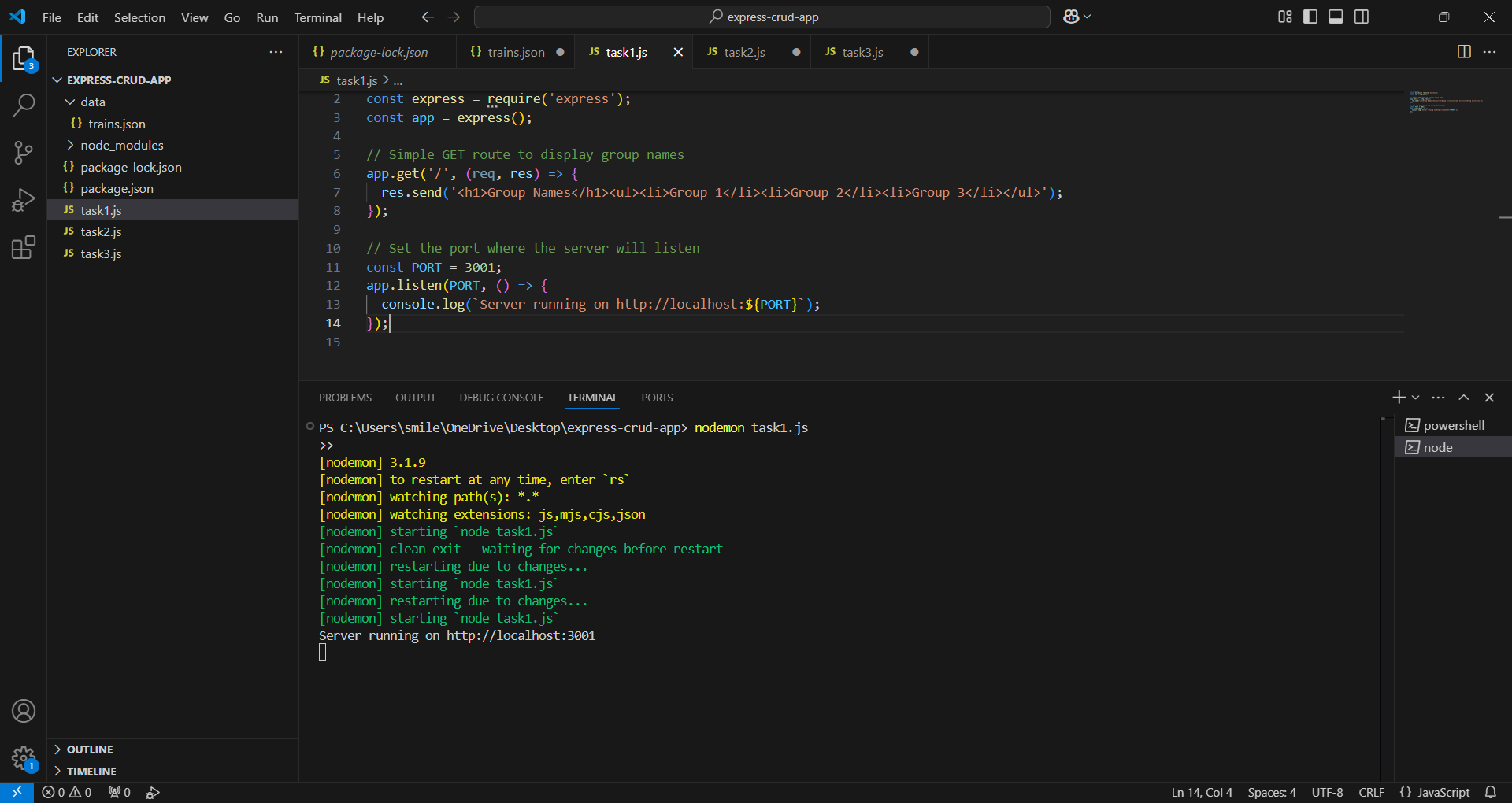
* task1.js → Displays a simple webpage with group names.
* task2.js → Reads and displays JSON data (train details).
* task3.js → Implements **CRUD operations** (Create, Read, Update, Delete) for train data.

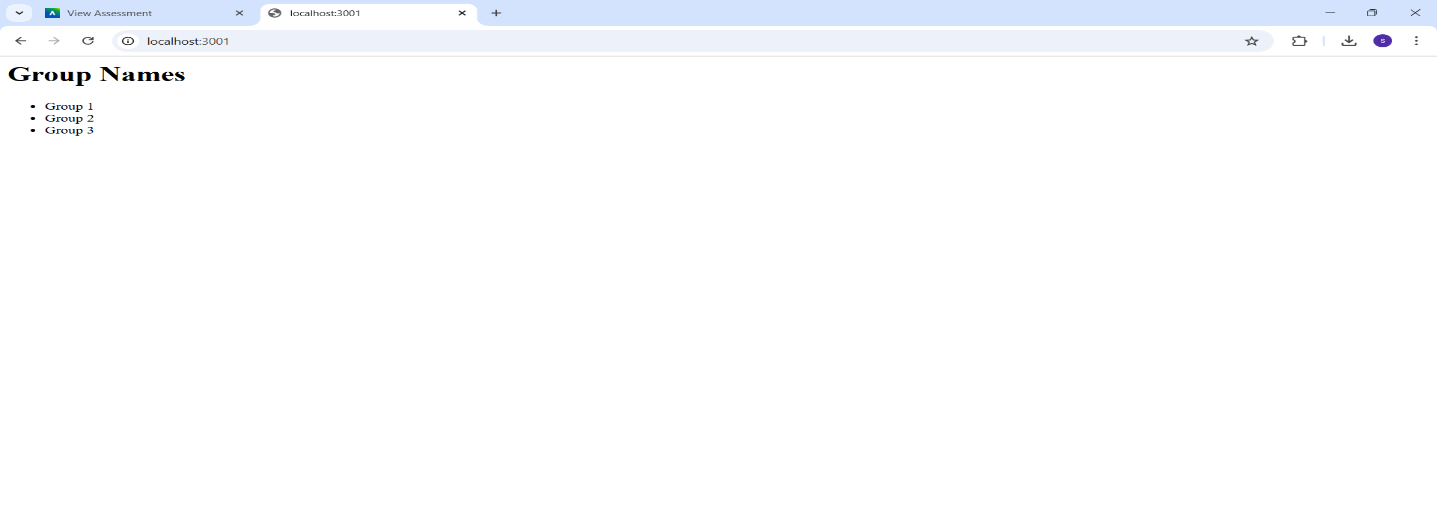
**task1.js – Basic Express Server with Homepage**

**Role:**  
It Sets up an Express server on **port 3001**  
It Handles a simple **GET request** (/)  
It Displays **group names** in an HTML list

**Functionality:**

* When a user visits **http://localhost:3001/**, we will see:

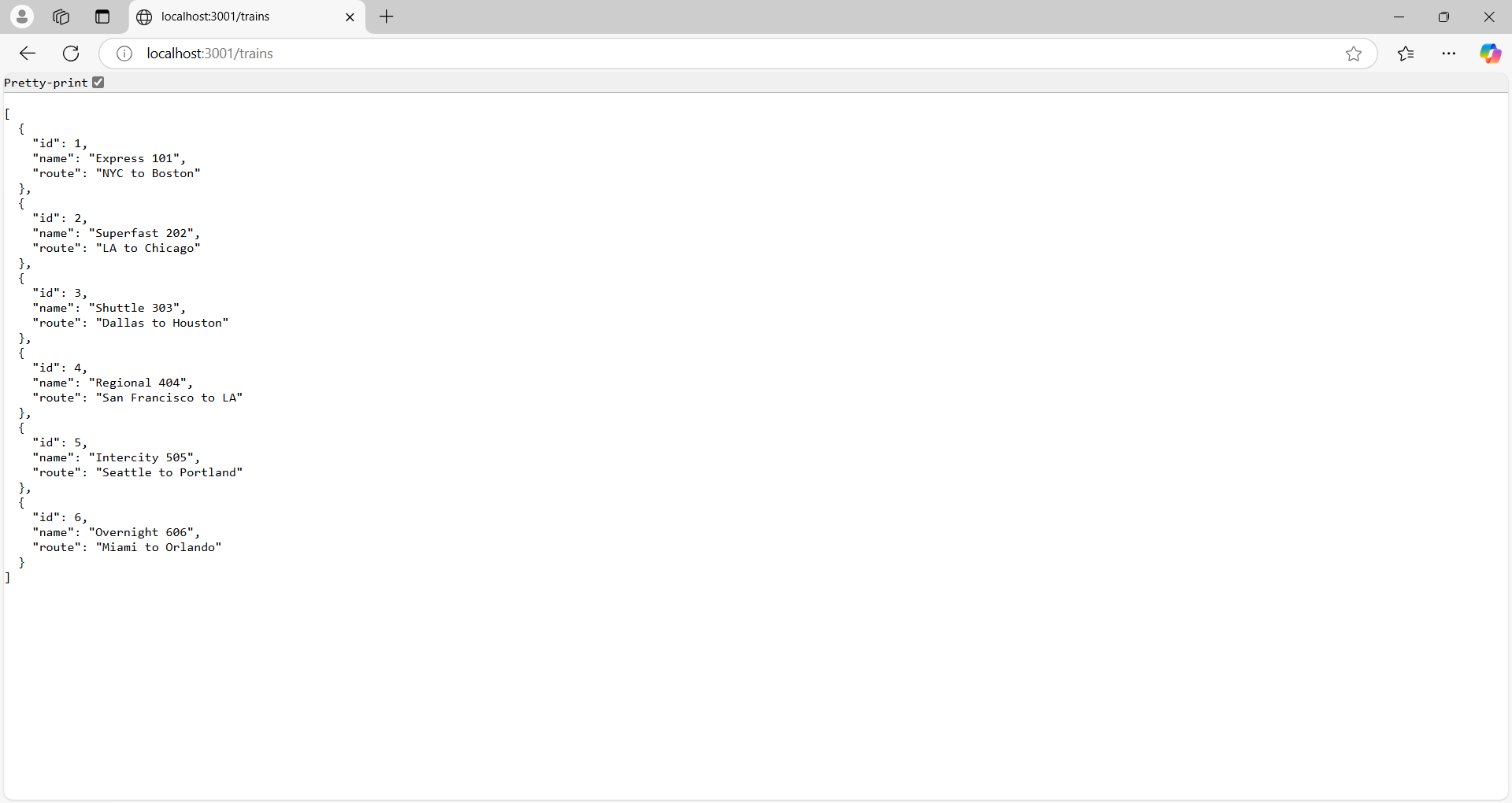


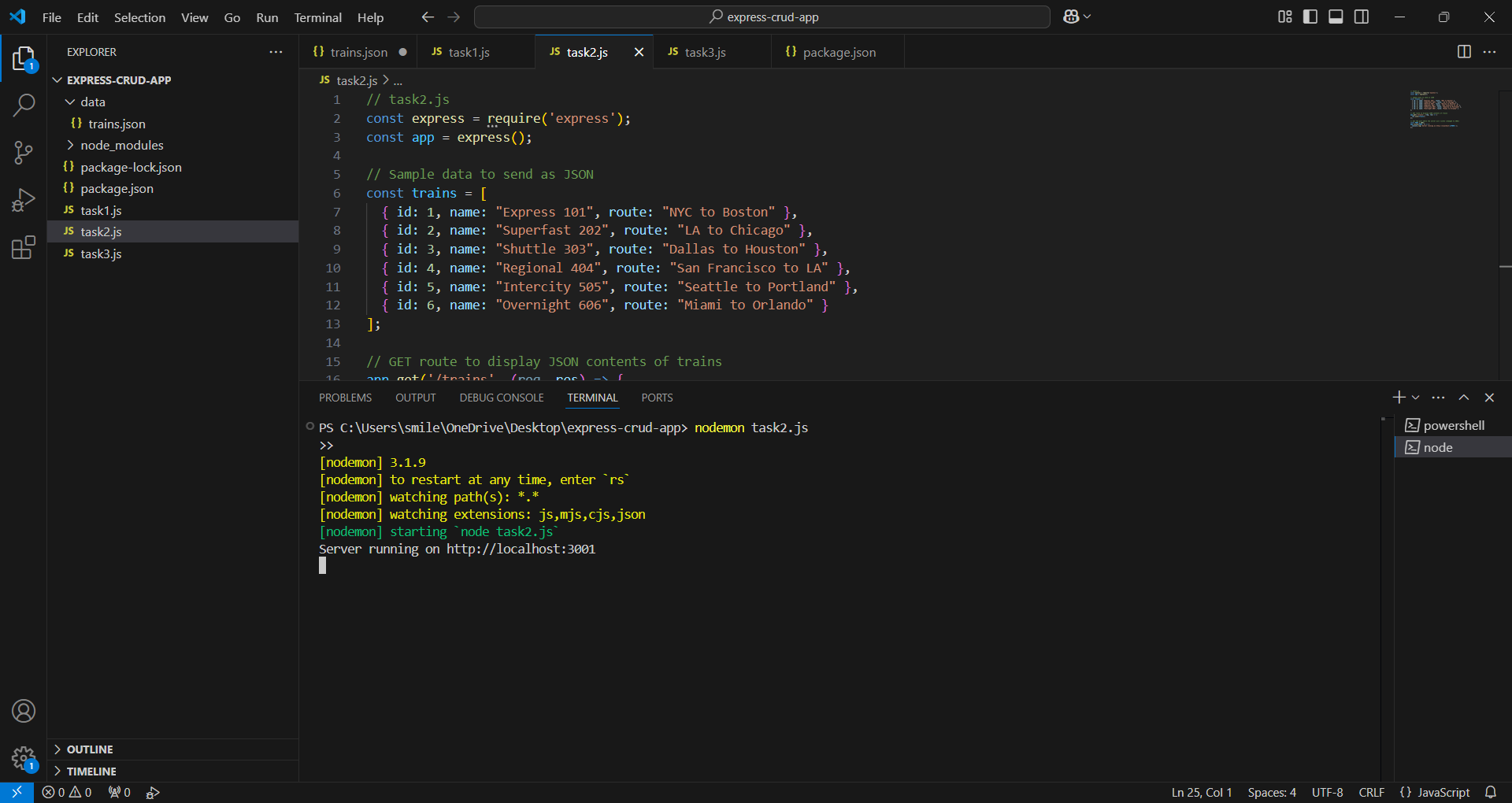


**task2.js – Display JSON Data in Browser**

**Role:**  
It Reads data from train.json (which contains train details)  
It Sends JSON data as a **response** to a GET request

**Functionality:**When a user visits **http://localhost:3001/trains**, we will see **raw JSON data**





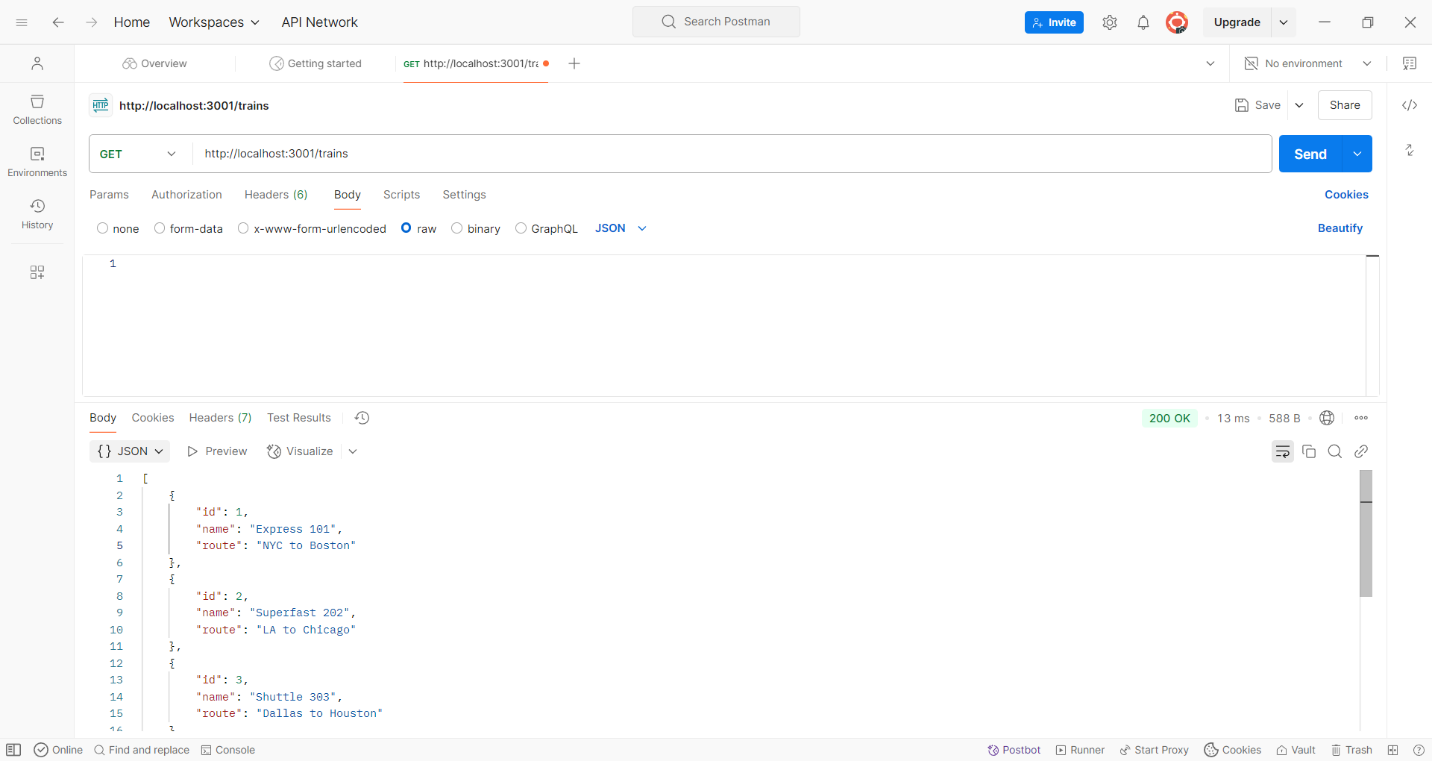
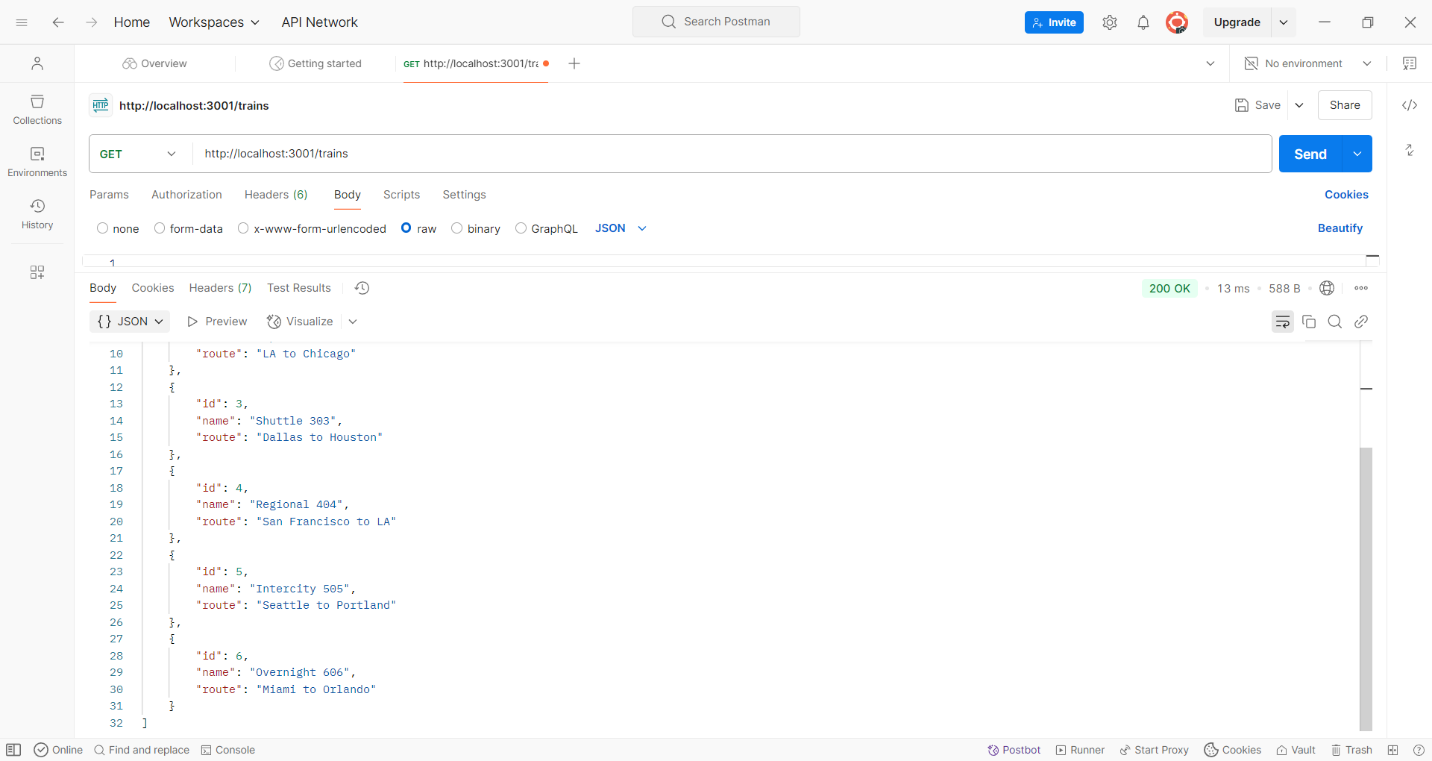
**task3.js – CRUD Operations with Express API**

**Role:**  
 Implements **Create, Read, Update, and Delete (CRUD)** operations  
 Uses Express.js methods:

* **POST** (/addTrain) → Adds a new train
* **GET** (/trains) → Fetches all trains
* **PUT** (/updateTrain/:id) → Updates train details
* **DELETE** (/deleteTrain/:id) → Removes a train  
  It Allows us to test API endpoints using **Postman**

**Functionality:**  **Get Request:**This route retrieves all trains.

* Method: GET
* URL: http://localhost:3001/trains

Response: 

* **POST** Request: Adds a new train
  + **URL:** http://localhost:3001/addTrain

**Body (JSON):** {

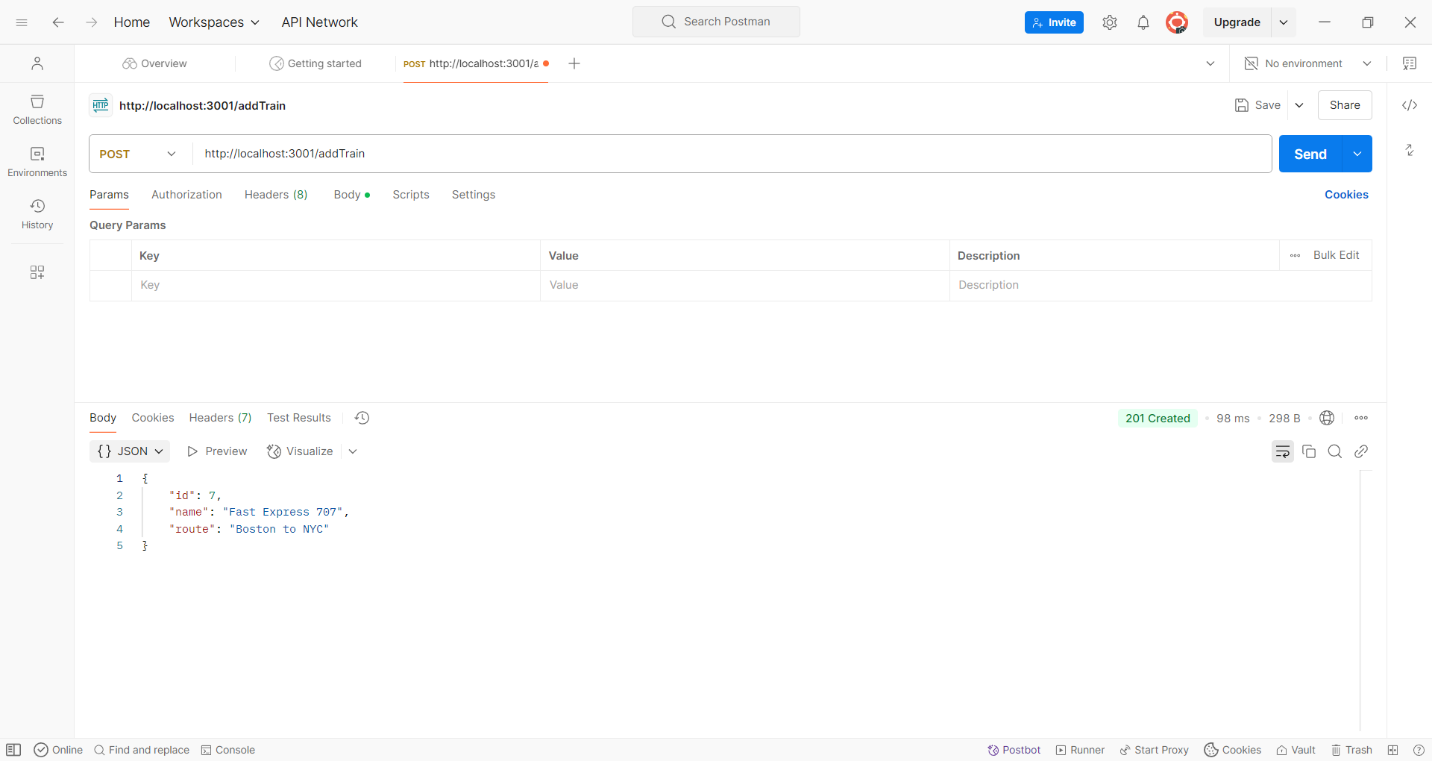
"id": 7,

"name": "Fast Express 707",

"route": "Boston to NYC"

}

Response:



**PUT** Request: Updates a train

* **URL:** http://localhost:3001/updateTrain/7

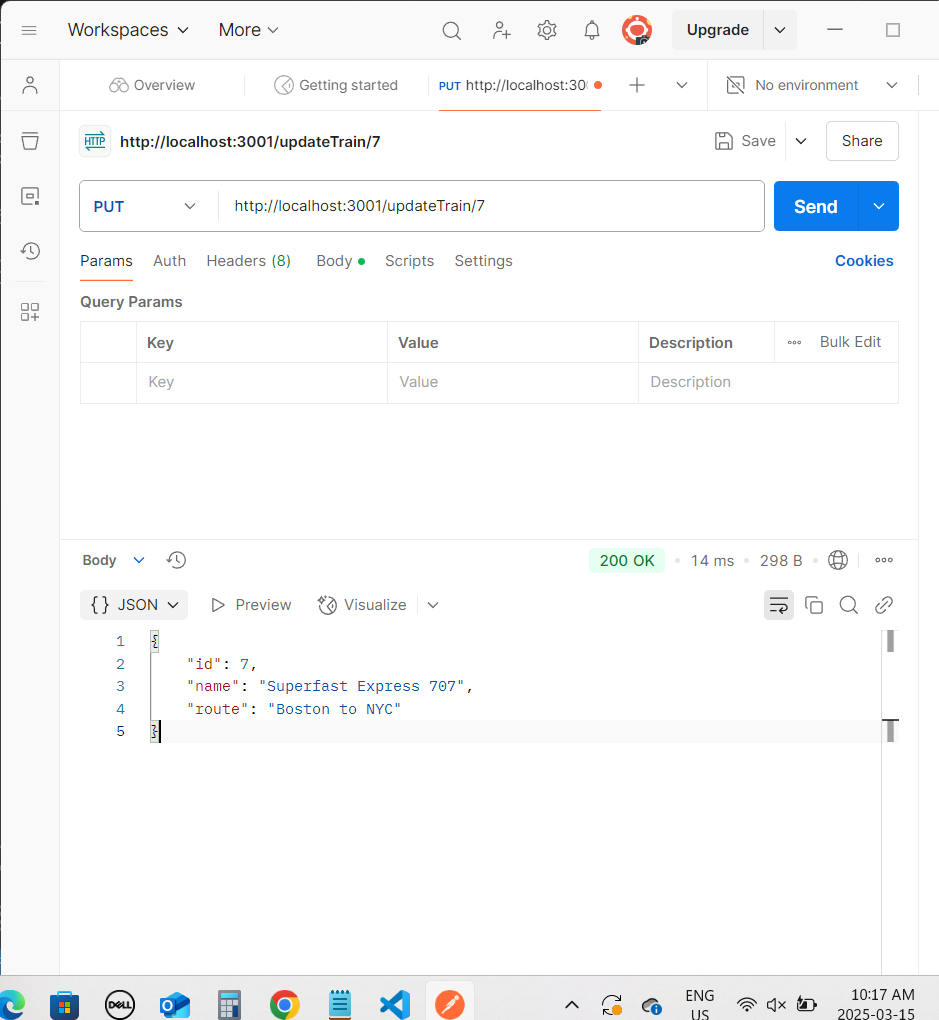
**Body (JSON):** {

"name": "Superfast Express 707",

"route": "Boston to NYC"

}

Response:

* 

**DELETE** Request: Removes a train

* **URL:** http://localhost:3001/deleteTrain/1
* Response : (if the train with ID 7 exists) 