**Experiment Number : 4 Date: 10-02-2025**

## **Create a RESTful API to Serve JSON Data**

### **PRE LAB EXERCISE**

**QUESTIONS:**

1. **What is a RESTful API?**

A RESTful API (Representational State Transfer API) is a web service that follows REST architectural principles. It enables clients to interact with resources over HTTP using standard methods like GET, POST, PUT, and DELETE for CRUD (Create, Read, Update, Delete) operations.

1. **What are the different HTTP methods used in RESTful APIs?**

Common HTTP methods in RESTful APIs include:

* **GET** – Retrieve data from a server
* **POST** – Create a new resource
* **PUT** – Update an existing resource
* **DELETE** – Remove a resource
* **PATCH** – Partially update a resource
* **OPTIONS** – Retrieve supported HTTP methods for a resource
* **HEAD** – Retrieve headers without the response body

1. **What is the difference between GET, POST, PUT, and DELETE methods?**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Purpose** | **Request Example** | **Use Case** |
| **GET** | Retrieve data from the server | GET /users | Fetching all users or a specific user |
| **POST** | Send data to the server to create a resource | POST /users (with JSON body) | Adding a new user to the database |
| **PUT** | Update an existing resource (or create if it doesn’t exist) | PUT /users/1 (with JSON body) | Updating user details (e.g., name, age) |
| **DELETE** | Remove a resource from the server | DELETE /users/1 | Deleting a user from the database |

1. **How do you create an API in Node.js using Express?’**

#### ****Step 1: Install Express****

npm init -y

npm install express

#### ****Step 2: Create**** server.js

#### ****Step 3: Run the Server****

node server.js

Visit http://localhost:3000/users to test the API.

### **IN LAB EXERCISE**

#### ****OBJECTIVE:****

* To create a RESTful API in Node.js that serves JSON data.
* To use Express.js for handling API requests.

#### ****RESOURCES:****

* Node.js installed
* Express.js
* Postman for API testing

#### ****PROGRAM LOGIC & IMPLEMENTATION:****

* Create a New Project Directory
* Initialize a Node.js Project **npm init –y**
* Install Express.js  **npm install express**
* Define routes for GET, POST, PUT, and DELETE.

### **RESTful API (**server.js**)**

const express = require('express');

const app = express();

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

let users = [

    { id: 1, name: "Krittika", age: 20 },

    { id: 2, name: "Mohan", age: 29 }

];

// GET request - Fetch users

app.get('/users', (req, res) => {

    res.json(users);

});

// POST request - Add a new user

app.post('/users', (req, res) => {

    const newUser = req.body;

    users.push(newUser);

    res.status(201).json(newUser);

});

// PUT request - Update a user

app.put('/users/:id', (req, res) => {

    const userId = parseInt(req.params.id);

    let user = users.find(u => u.id === userId);

    if (!user) return res.status(404).send('User not found');

    user.name = req.body.name;

    user.age = req.body.age;

    res.json(user);

});

// DELETE request - Remove a user

app.delete('/users/:id', (req, res) => {

    users = users.filter(u => u.id !== parseInt(req.params.id));

    res.json({ message: 'User deleted' });

});

app.listen(3000, () => console.log('Server running on port 3000'));

* Run the Server - node server.js

1. **Serve JSON data as API responses.**

#### GET All Users

Returns all users in JSON format.

**curl.exe -X GET** [**http://localhost:3000/users**](http://localhost:3000/users)

**A black screen with white text

AI-generated content may be incorrect.**

* **POST (Add a New User)**

Adds a new user and returns the created user in JSON.

**curl.exe -X POST http://localhost:3000/users -H "Content-Type: application/json" -d "{\"id\":3, \"name\":\"Abi\", \"age\":28}"**



* **PUT (Update a User)**

Updates an existing user and returns the updated JSON object.

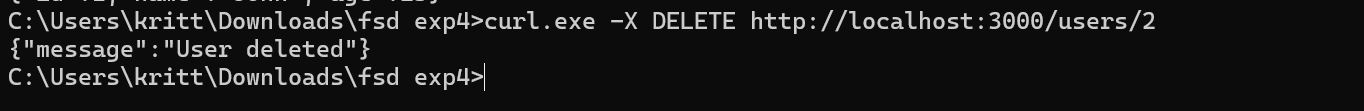
**curl.exe -X PUT http://localhost:3000/users/1 -H "Content-Type: application/json" -d "{\"name\":\"John\", \"age\":23}"**



* **DELETE (Remove a User)**

Deletes a user and returns a confirmation message in JSON.

**curl.exe -X DELETE** [**http://localhost:3000/users/1**](http://localhost:3000/users/1)

****

### **POST LAB EXERCISE**

* 1. **How does express.json() help in handling API requests?**

express.json() is a built-in middleware in Express.js that parses incoming JSON payloads in request bodies. It is necessary for handling **POST** and **PUT** requests, where the client sends data in JSON format.

**Without express.json()**

* The req.body would be undefined, meaning Express.js wouldn’t be able to process JSON data.

#### ****With**** express.json()

* The middleware automatically parses the JSON request body and makes it accessible via req. body.

**Example:**

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

* 1. **What is the difference between req.params and req.body?**

|  |  |  |
| --- | --- | --- |
| **Feature** | **req.params** | **req.body** |
| Purpose | Used for URL parameters (e.g., /users/:id) | Used for data sent in the request body |
| Example URL | /users/1 | /users (with JSON data) |
| Example Access | req.params.id → 1 | req.body.name → "Charlie" |
| Common Methods | **GET, PUT, DELETE** | **POST, PUT** |

**Example:**

// Using req.params to get a user ID from the URL

app.get('/users/:id', (req, res) => {

    console.log(req.params.id); // Extracts user ID from the URL

});

// Using req.body to get JSON data from the request body

app.post('/users', (req, res) => {

    console.log(req.body); // Extracts data sent in the request body

});

* 1. **Test the API using Postman or cURL and observe the response.**

### **Using Postman**

**Step 1: Open Postman & Create a Workspace**

* **Launch Postman**.
* On the **left sidebar**, click on **Workspaces**.
* Click **Create Workspace** (or use the default "My Workspace").
* **Enter a Name** (e.g., My API Testing).
* **Choose Visibility**: Select **Personal**.
* Click **Create Workspace**.

**Testing the API Endpoints**

* 1. **GET All Users**
* Click on **New Tab (+)** in Postman.
* Select **GET** from the dropdown (left of the URL bar).
* URL - <http://localhost:3000/users>
* Click **Send**.

A screenshot of a computer

AI-generated content may be incorrect.

1. **POST (Add a New User)**
   * + Method: POST
     + URL: http://localhost:3000/users
     + **Headers:** Content-Type: application/json
       - Click Body → raw → Select JSON (from the dropdown).

**{**

**"id": 3,**

**"name": "Prakash",**

**"age": 18**

**}**

* + - Click **Send** and verify if the user was added.

A screenshot of a computer

AI-generated content may be incorrect.

1. **PUT (Update a User)**
   * + Method: PUT
     + URL: http://localhost:3000/users/1
     + **Body (raw, JSON format):**

**{**

**"name": "Krittika Vellingiri","age": 21**

**}**

Click **Send** and check if the update was successful.

A screenshot of a computer

AI-generated content may be incorrect.

1. **DELETE (Remove a User)**
   * + Method: DELETE
     + URL: http://localhost:3000/users/1
     + Click **Send** and verify if the user was deleted.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. **Modify the API to return users based on age.**

#### ****Get Users by Age (Example: Age 29)****

**curl.exe -X GET** [**http://localhost:3000/users?age=29**](http://localhost:3000/users?age=29)

or in Postman:

* Method: GET
* URL: <http://localhost:3000/users?age=28>

A screenshot of a computer

AI-generated content may be incorrect.A black screen with white text

AI-generated content may be incorrect.**A screen shot of a computer program

AI-generated content may be incorrect.**

**ASSESSMENT PATTERN:**

|  |  |  |
| --- | --- | --- |
| **Description** | **Max Marks** | **Marks Awarded** |
| Pre Lab Exercise | **5** |  |
| In Lab Exercise | **10** |  |
| Post Lab Exercise | **5** |  |
| Viva | **10** |  |
| **Total** | **30** |  |
| **Faculty Signature** | |  |