**Build a REST API that returns dummy data**

Here’s a simple **REST API** using **Node.js, Express, and JSON data** that returns dummy data.

**Steps to Run:**

1. Install Node.js (if not installed).
2. Run: npm init -y
3. Install Express: npm install express
4. Save the following code as server.js
5. Run the API: node server.js
6. Open http://localhost:3000/api/users in a browser or Postman.

**Code for REST API (server.js)**

javascript

const express = require("express");

const app = express();

const PORT = 3000;

// Dummy Data

const users = [

{ id: 1, name: "John Doe", email: "john@example.com" },

{ id: 2, name: "Jane Smith", email: "jane@example.com" },

{ id: 3, name: "Alice Johnson", email: "alice@example.com" }

];

// Get all users

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

res.json(users);

});

// Get user by ID

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

const user = users.find(u => u.id === parseInt(req.params.id));

if (user) {

res.json(user);

} else {

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

}

});

// Start the server

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

**How It Works**

✅ GET /api/users → Returns all users  
✅ GET /api/users/1 → Returns user with ID 1  
✅ GET /api/users/99 → Returns { "error": "User not found" }

**Implement middleware for request logging**

Here's how you can **implement middleware for request logging** in your **REST API** using **Express.js**.

**Steps to Implement Request Logging Middleware**

1. **Create a middleware function** to log requests.
2. **Use app.use()** to apply middleware globally.
3. **Log request details** like method, URL, timestamp, etc.

**Updated Code (server.js)**

javascript

const express = require("express");

const app = express();

const PORT = 3000;

// Middleware for request logging

const requestLogger = (req, res, next) => {

console.log(`[${new Date().toISOString()}] ${req.method} ${req.url}`);

next(); // Pass control to the next middleware

};

// Apply middleware

app.use(requestLogger);

// Dummy Data

const users = [

{ id: 1, name: "John Doe", email: "john@example.com" },

{ id: 2, name: "Jane Smith", email: "jane@example.com" },

{ id: 3, name: "Alice Johnson", email: "alice@example.com" }

];

// Routes

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

res.json(users);

});

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

const user = users.find(u => u.id === parseInt(req.params.id));

if (user) {

res.json(user);

} else {

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

}

});

// Start the server

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

**How the Middleware Works**

✅ Logs every request with:

* **Timestamp** (ISO format)
* **HTTP Method** (GET, POST, etc.)
* **Requested URL**

**Example Log Output**

bash

[2025-03-17T10:30:00.123Z] GET /api/users

[2025-03-17T10:30:05.456Z] GET /api/users/2

**Create a CRUD API with Express.js**

how to **create a CRUD API with Express.js** in **Node.js**. This example will perform basic **Create, Read, Update, and Delete** operations using a simple in-memory data store (you can later replace it with a real database like MongoDB or MySQL).

**Steps:**

1. **Install dependencies**:

csharp

npm init -y

npm install express body-parser

1. **Create server.js** and implement CRUD operations.

**Code (server.js)**

javascript

const express = require("express");

const bodyParser = require("body-parser");

const app = express();

const PORT = 3000;

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

// Dummy data to simulate a database

let items = [

{ id: 1, name: "Item 1", description: "Description of item 1" },

{ id: 2, name: "Item 2", description: "Description of item 2" },

{ id: 3, name: "Item 3", description: "Description of item 3" }

];

// CREATE - Add new item

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

const { name, description } = req.body;

const newItem = {

id: items.length + 1, // Simple way to generate an ID

name,

description

};

items.push(newItem);

res.status(201).json(newItem); // Send the newly created item back

});

// READ - Get all items

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

res.json(items); // Return all items

});

// READ - Get item by ID

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

const item = items.find((item) => item.id === parseInt(req.params.id));

if (!item) {

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

}

res.json(item); // Return the item

});

// UPDATE - Update item by ID

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

const item = items.find((item) => item.id === parseInt(req.params.id));

if (!item) {

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

}

const { name, description } = req.body;

item.name = name || item.name;

item.description = description || item.description;

res.json(item); // Return the updated item

});

// DELETE - Delete item by ID

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

const itemIndex = items.findIndex((item) => item.id === parseInt(req.params.id));

if (itemIndex === -1) {

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

}

items.splice(itemIndex, 1); // Remove the item from the array

res.status(204).end(); // No content to return, just respond with status code 204

});

// Start the server

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

**Explanation of Endpoints**

1. **Create Item (POST /api/items)**
   * Accepts name and description in the request body.
   * Adds a new item to the data store.
2. **Get All Items (GET /api/items)**
   * Returns a list of all items.
3. **Get Item by ID (GET /api/items/:id)**
   * Fetches an item by its ID.
4. **Update Item by ID (PUT /api/items/:id)**
   * Updates the name and description of an item based on its ID.
5. **Delete Item by ID (DELETE /api/items/:id)**
   * Deletes an item by its ID.

**Testing the API**

You can test the API with **Postman** or **cURL**.

**1. Create a new item:**

json

POST /api/items

{

"name": "Item 4",

"description": "Description of item 4"

}

**2. Get all items:**

json

GET /api/items

**3. Get item by ID:**

json

GET /api/items/2

**4. Update item by ID:**

json

PUT /api/items/2

{

"name": "Updated Item 2",

"description": "Updated description of item 2"

}

**5. Delete item by ID:**

json

DELETE /api/items/3

**Running the Server**

* Run the server by executing node server.js in your terminal.
* The server will run at http://localhost:3000.