Perfect! Let’s go **one by one**, keeping everything **super simple**, using **Node + Express + HTML/CSS/JS**, no database. I’ll start with **Experiment 1 – Portfolio Website**.

Experiment 1: Portfolio Website

Objective

Create a simple portfolio website to display **your details** for a potential recruiter.

---

1 Folder Structure

```
portfolio/
│
├── server.js
├── package.json
│
├── public/
│   ├── index.html
│   ├── style.css
│   └── script.js
```

---

2 Initialize Project

```
npm init -y
npm install express
```

---

3 server.js

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

app.use(express.static("public"));

app.listen(PORT, () => console.log(`Server running at
http://localhost:${PORT}`));
```

---

4 public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>My Portfolio</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>🧑‍💻 Eunice S</h1>
  <h2>About Me</h2>
  <p>Hi! I am a BTech IT student. I am interested in web development,
freelancing, and UI/UX design.</p>

  <h2>Skills</h2>
  <ul>
    <li>HTML, CSS, JavaScript</li>
    <li>Node.js, Express</li>
    <li>UI/UX Design</li>
  </ul>

  <h2>Projects</h2>
  <ul>
    <li>Personal Finance Tracker</li>
    <li>Hotel Booking System</li>
    <li>Job Portal</li>
  </ul>

  <h2>Contact</h2>
  <p>Email: eunice@example.com</p>
</body>
</html>
```

5️⃣ public/style.css

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background: #f5f5f5;
  padding: 20px;
}

h1 {
  color: #007bff;
}

ul {
  list-style: none;
  padding: 0;
}
```

🖥️ How to Run

1. Terminal → project folder
2. Run:

```
node server.js
```

3. Open browser → http://localhost:3000

✅ Expected Output

- Shows **your name**, **about me**, **skills**, **projects**, and **contact info**.
- Simple, clean, one-page portfolio.

Perfect! Let's do **Experiment 2 – TO-DO List App** in the **simplest possible way**.

Experiment 2: TO-DO List App

Objective

Create a simple web application where users can **manage their to-do list**. No database — data stored in memory.

---

1 Folder Structure

```
todo-app/
│
├── server.js
├── package.json
│
├── public/
│   ├── index.html
│   ├── style.css
│   └── script.js
```

---

2 Initialize Project

```
npm init -y
npm install express
```

---

3 server.js

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

app.use(express.static("public"));
app.use(express.json());

let todos = [];

// Get all todos
app.get("/todos", (req, res) => {
  res.json(todos);
});

// Add a todo
```

```
app.post("/todos", (req, res) => {
  const { task } = req.body;
  todos.push({ id: todos.length + 1, task });
  res.json({ message: "Task added", todos });
});

// Delete a todo
app.delete("/todos/:id", (req, res) => {
  todos = todos.filter(t => t.id != req.params.id);
  res.json({ message: "Task deleted", todos });
});

app.listen(PORT, () => console.log(`Server running at
http://localhost:${PORT}`));
```

---

4 public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>TO-DO List</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>📝 TO-DO List</h1>

  <input type="text" id="taskInput" placeholder="Enter your task">
  <button onclick="addTask()">Add Task</button>

  <h2>Tasks</h2>
  <div id="todoList"></div>

<script src="script.js"></script>
</body>
</html>
```

---

5 public/style.css

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background: #f5f5f5;
  padding: 20px;
}

input {
  padding: 5px;
  width: 200px;
}

button {
  padding: 5px 10px;
  margin-left: 5px;
  border-radius: 5px;
  background: #007bff;
  color: white;
  border: none;
  cursor: pointer;
}

button:hover {
  background: #0056b3;
}

.task-item {
  background: white;
  margin: 10px auto;
  padding: 10px;
  border-radius: 10px;
  width: 60%;
  max-width: 400px;
  text-align: left;
}
```

---

6 public/script.js

```
let todos = [];

async function loadTodos() {
  const res = await fetch("/todos");
  todos = await res.json();
  displayTodos();
}

function displayTodos() {
  const container = document.getElementById("todoList");
  container.innerHTML = "";

  todos.forEach(todo => {
    const div = document.createElement("div");
    div.className = "task-item";
    div.innerHTML = `
      ${todo.task}
      <button onclick="deleteTask(${todo.id})">Delete</button>
    `;
    container.appendChild(div);
  });
}

async function addTask() {
  const task = document.getElementById("taskInput").value;
  if(!task) { alert("Enter a task"); return; }

  await fetch("/todos", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ task })
  });

  document.getElementById("taskInput").value = "";
  loadTodos();
}

async function deleteTask(id) {
  await fetch(`/todos/${id}`, { method: "DELETE" });
  loadTodos();
}

window.onload = loadTodos;
```

🖥 How to Run

1. Terminal → project folder
2. Run:

```
node server.js
```

3. Open browser → http://localhost:3000

✅ Expected Output

- Users can **add tasks** using the input field.
- Tasks are displayed below with a **Delete** button.
- Users can **delete tasks** immediately.

Perfect! Let’s do **Experiment 3 – Microblogging App** in the **simplest possible way**.

Experiment 3: Microblogging App (like Twitter)

Objective

Create a simple microblogging app where users can **post content** and view all posts.
No database — all data stored in memory.

---

1 Folder Structure

```
microblog-app/
│
├── server.js
├── package.json
│
├── public/
│   ├── index.html
│   ├── style.css
│   └── script.js
│
```

---

2 Initialize Project

```
npm init -y
npm install express
```

---

3 server.js

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

app.use(express.static("public"));
app.use(express.json());

let posts = [];

// Get all posts
app.get("/posts", (req, res) => {
  res.json(posts);
});

// Add a post
```

```
app.post("/posts", (req, res) => {
  const { username, content } = req.body;
  posts.unshift({ id: posts.length + 1, username, content }); // newest first
  res.json({ message: "Post added", posts });
});

app.listen(PORT, () => console.log(`Server running at
http://localhost:${PORT}`));
```

---

4 public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Microblogging App</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>🐦 Microblogging App</h1>

  <h2>New Post</h2>
  <input type="text" id="username" placeholder="Your Name"><br>
  <textarea id="content" placeholder="What's on your mind?"></textarea><br>
  <button onclick="addPost()">Post</button>

  <h2>All Posts</h2>
  <div id="postsList"></div>

<script src="script.js"></script>
</body>
</html>
```

---

5 public/style.css

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background: #f5f5f5;
  padding: 20px;
}

input, textarea {
  width: 60%;
  padding: 5px;
  margin: 5px 0;
}

textarea {
  height: 60px;
}
```

```
button {
  padding: 5px 10px;
  margin: 5px 0;
  border-radius: 5px;
  background: #007bff;
  color: white;
  border: none;
  cursor: pointer;
}

button:hover {
  background: #0056b3;
}

.post-item {
  background: white;
  margin: 10px auto;
  padding: 10px;
  border-radius: 10px;
  width: 70%;
  max-width: 500px;
  text-align: left;
}
```

---

6 public/script.js

```
let posts = [];

async function loadPosts() {
  const res = await fetch("/posts");
  posts = await res.json();
  displayPosts();
}

function displayPosts() {
  const container = document.getElementById("postsList");
  container.innerHTML = "";

  posts.forEach(post => {
    const div = document.createElement("div");
    div.className = "post-item";
    div.innerHTML = `<strong>${post.username}</strong><br>${post.content}`;
    container.appendChild(div);
  });
}

async function addPost() {
  const username = document.getElementById("username").value;
  const content = document.getElementById("content").value;

  if(!username || !content){ alert("Enter both name and content"); return; }

  await fetch("/posts", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ username, content })
  });

  document.getElementById("username").value = "";
  document.getElementById("content").value = "";
  loadPosts();
}

window.onload = loadPosts;
```

🖥 How to Run

1. Terminal → project folder
2. Run:

```
node server.js
```

3. Open browser → http://localhost:3000

✅ Expected Output

- Users can **enter name and content** to post a message.
- All posts appear below, **newest posts first**.
- Simple, functional microblogging platform without database.

Great! Let’s do **Experiment 4 – Food Delivery Website** in the **simplest possible way**.

Experiment 4: Food Delivery Website

Objective

Create a simple food delivery site where users can **view menu items** and **place orders**.
No database — all data stored in memory.

---

1 Folder Structure

```
food-delivery/
│
├── server.js
├── package.json
│
├── public/
│   ├── index.html
│   ├── style.css
│   └── script.js
```

---

2 Initialize Project

```
npm init -y
npm install express
```

---

3 server.js

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

app.use(express.static("public"));
app.use(express.json());

let menu = [
  { id: 1, name: "Pizza", price: 250 },
  { id: 2, name: "Burger", price: 120 },
  { id: 3, name: "Pasta", price: 200 },
  { id: 4, name: "Fries", price: 80 }
];
```

```
let orders = [];

app.get("/menu", (req, res) => {
  res.json(menu);
});

app.post("/order", (req, res) => {
  const { itemId, quantity } = req.body;
  const item = menu.find(m => m.id == itemId);
  if(item){
    orders.push({ item: item.name, quantity, total: item.price * quantity });
    res.json({ message: `Ordered ${quantity} ${item.name}(s)` });
  } else {
    res.json({ message: "Item not found" });
  }
});

app.listen(PORT, () => console.log(`Server running at
http://localhost:${PORT}`));
```

4 public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Food Delivery</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>🍔 Food Delivery</h1>

  <h2>Menu</h2>
  <div id="menuList"></div>

<script src="script.js"></script>
</body>
</html>
```

5 public/style.css

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background: #f5f5f5;
  padding: 20px;
}

.menu-item {
  background: white;
  margin: 10px auto;
  padding: 15px;
  border-radius: 10px;
  width: 70%;
  max-width: 400px;
  text-align: left;
}

input {
  width: 50px;
  padding: 5px;
  margin-left: 5px;
}

button {
  padding: 5px 10px;
  margin-left: 5px;
  border-radius: 5px;
  background: #007bff;
  color: white;
  border: none;
  cursor: pointer;
}

button:hover {
  background: #0056b3;
}
```

---

6 public/script.js

```
let menu = [];

async function loadMenu() {
  const res = await fetch("/menu");
  menu = await res.json();
  displayMenu();
}

function displayMenu() {
  const container = document.getElementById("menuList");
  container.innerHTML = "";

  menu.forEach(item => {
    const div = document.createElement("div");
    div.className = "menu-item";
    div.innerHTML = `
      <strong>${item.name}</strong> - ₹${item.price}<br>
      Quantity: <input type="number" min="1" value="1" id="qty${item.id}">
      <button onclick="placeOrder(${item.id})">Order</button>
    `;
    container.appendChild(div);
  });
}

async function placeOrder(id) {
  const qty = Number(document.getElementById(`qty${id}`).value);
  if(qty <= 0) { alert("Enter valid quantity"); return; }

  const res = await fetch("/order", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ itemId: id, quantity: qty })
  });

  const data = await res.json();
  alert(data.message);
}

window.onload = loadMenu;
```

🖥 How to Run

1. Terminal → project folder
2. Run:

```
node server.js
```

3. Open browser → http://localhost:3000

✅ Expected Output

- Shows **menu items** with name and price.
- Users can **enter quantity** and **place an order**.
- Alert confirms the order immediately.

Perfect! Let's do **Experiment 5 – Classifieds Web Application** in the **simplest possible way**.

🧩 Experiment 5: Classifieds Web Application

🎯 Objective

Create a simple web app where users can **buy and sell used products**.
No database — all data stored in memory.

1️⃣ Folder Structure

```
classifieds-app/
│
├── server.js
├── package.json
│
├── public/
│   ├── index.html
│   ├── style.css
│   └── script.js
│
```

2️⃣ Initialize Project

```
npm init -y
npm install express
```

3️⃣ server.js

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

app.use(express.static("public"));
app.use(express.json());

let products = [];

// Get all products
app.get("/products", (req, res) => {
  res.json(products);
});
```

```
// Add a product
app.post("/products", (req, res) => {
  const { name, price, seller } = req.body;
  products.unshift({ id: products.length + 1, name, price, seller });
  res.json({ message: "Product added", products });
});

app.listen(PORT, () => console.log(`Server running at
http://localhost:${PORT}`));
```

---

4 public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Classifieds App</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>🛒 Classifieds Web Application</h1>

  <h2>Sell a Product</h2>
  <input type="text" id="productName" placeholder="Product Name"><br>
  <input type="text" id="productPrice" placeholder="Price"><br>
  <input type="text" id="productSeller" placeholder="Seller Name"><br>
  <button onclick="addProduct()">Add Product</button>

  <h2>Available Products</h2>
  <div id="productList"></div>

<script src="script.js"></script>
</body>
</html>
```

---

5 public/style.css

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background: #f5f5f5;
  padding: 20px;
}

input {
  padding: 5px;
  margin: 5px 0;
  width: 200px;
}

button {
  padding: 5px 10px;
  margin: 5px 0;
  border-radius: 5px;
  background: #007bff;
  color: white;
  border: none;
  cursor: pointer;
}

button:hover {
  background: #0056b3;
}

.product-item {
  background: white;
  margin: 10px auto;
  padding: 10px;
  border-radius: 10px;
  width: 70%;
  max-width: 400px;
  text-align: left;
}
```

---

6 public/script.js

```
let products = [];

async function loadProducts() {
  const res = await fetch("/products");
  products = await res.json();
  displayProducts();
}

function displayProducts() {
  const container = document.getElementById("productList");
  container.innerHTML = "";

  products.forEach(p => {
    const div = document.createElement("div");
    div.className = "product-item";
    div.innerHTML = `
      <strong>${p.name}</strong> - ₹${p.price} <br>
      Seller: ${p.seller}
    `;
    container.appendChild(div);
  });
}

async function addProduct() {
  const name = document.getElementById("productName").value;
  const price = document.getElementById("productPrice").value;
  const seller = document.getElementById("productSeller").value;

  if(!name || !price || !seller){ alert("Enter all fields"); return; }
  await fetch("/products", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ name, price, seller })
  });

  document.getElementById("productName").value = "";
  document.getElementById("productPrice").value = "";
  document.getElementById("productSeller").value = "";

  loadProducts();
}

window.onload = loadProducts;
```

🖥 How to Run

1. Terminal → project folder
2. Run:

```
node server.js
```

3. Open browser → http://localhost:3000

✅ Expected Output

- Users can **add a product** with name, price, and seller.
- All products are **listed below** with seller info.
- New products appear **immediately** after adding.

Perfect! Let’s do **Experiment 6 – Leave Management System** in the **simplest possible way**.

Experiment 6: Leave Management System

Objective

Create a simple leave management system where users can **apply for leaves** (casual or medical) and **view available leave days**.
No database — data stored in memory.

---

1 Folder Structure

```
leave-management/
│
├── server.js
├── package.json
│
├── public/
│   ├── index.html
│   ├── style.css
│   └── script.js
```

---

2 Initialize Project

```
npm init -y
npm install express
```

---

3 server.js

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

app.use(express.static("public"));
app.use(express.json());

let leaveBalance = { casual: 10, medical: 5 };
let leaveApplications = [];

// Get leave balance and applications
app.get("/leaves", (req, res) => {
  res.json({ leaveBalance, leaveApplications });
});

// Apply leave
app.post("/apply", (req, res) => {
  const { type, days } = req.body;
  if(leaveBalance[type] >= days){
    leaveBalance[type] -= days;
    leaveApplications.push({ type, days });
    res.json({ message: `Applied ${days} ${type} leave`, leaveBalance,
leaveApplications });
  } else {
    res.json({ message: "Not enough leave balance" });
  }
});

app.listen(PORT, () => console.log(`Server running at
http://localhost:${PORT}`));
```

4️⃣ public/index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Leave Management</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>🏢 Leave Management System</h1>

  <h2>Leave Balance</h2>
  <div id="leaveBalance"></div>

  <h2>Apply Leave</h2>
  <select id="leaveType">
    <option value="casual">Casual</option>
    <option value="medical">Medical</option>
  </select>
  <input type="number" id="leaveDays" min="1" placeholder="Days">
  <button onclick="applyLeave()">Apply</button>

  <h2>Leave Applications</h2>
  <div id="applications"></div>

<script src="script.js"></script>
</body>
</html>
```

5️⃣ public/style.css

```css
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background: #f5f5f5;
  padding: 20px;
}

input, select {
  padding: 5px;
  margin: 5px;
}

button {
  padding: 5px 10px;
  margin: 5px;
  border-radius: 5px;
  background: #007bff;
  color: white;
  border: none;
  cursor: pointer;
}

button:hover {
  background: #0056b3;
}

.item {
  background: white;
  margin: 10px auto;
  padding: 10px;
  border-radius: 10px;
  width: 60%;
  max-width: 400px;
  text-align: left;
}
```

---

6 public/script.js

```javascript
let leaveBalance = {};
let leaveApplications = [];

async function loadLeaves() {
  const res = await fetch("/leaves");
  const data = await res.json();
  leaveBalance = data.leaveBalance;
  leaveApplications = data.leaveApplications;
  displayLeaves();
}

function displayLeaves() {
  const balanceDiv = document.getElementById("leaveBalance");
  balanceDiv.innerHTML = `Casual: ${leaveBalance.casual} days, Medical:
${leaveBalance.medical} days`;

  const appDiv = document.getElementById("applications");
  appDiv.innerHTML = "";
  leaveApplications.forEach((l, i) => {
    const div = document.createElement("div");
    div.className = "item";
    div.innerHTML = `${i+1}. ${l.type} leave - ${l.days} days`;
    appDiv.appendChild(div);
  });
}

async function applyLeave() {
  const type = document.getElementById("leaveType").value;
  const days = Number(document.getElementById("leaveDays").value);
  if(days <= 0){ alert("Enter valid number of days"); return; }

  const res = await fetch("/apply", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ type, days })
  });

  const data = await res.json();
  alert(data.message);
  loadLeaves();
}

window.onload = loadLeaves;
```

🖥 How to Run

1. Terminal → project folder
2. Run:

```
node server.js
```

3. Open browser → http://localhost:3000

✅ Expected Output

- Shows **current leave balance**.
- Users can **apply casual or medical leave** by entering number of days.
- Shows **list of applied leaves**.
- Leaves reduce from balance immediately after applying.

Perfect! Let’s do **Experiment 7 – Simple Dashboard for Project Management** in the **simplest possible way**.

Experiment 7: Project Management Dashboard

Objective

Create a simple dashboard where users can **add tasks** and **update their status** among **Pending, InProgress, or Completed**.
No database — all data stored in memory.

---

1 Folder Structure

```
project-dashboard/
│
├── server.js
├── package.json
│
├── public/
│   ├── index.html
│   ├── style.css
│   └── script.js
│
```

---

2 Initialize Project

```
npm init -y
npm install express
```

---

3 server.js

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

app.use(express.static("public"));
app.use(express.json());

let tasks = [];

// Get all tasks
```

```
app.get("/tasks", (req, res) => {
  res.json(tasks);
});

// Add a task
app.post("/tasks", (req, res) => {
  const { name } = req.body;
  tasks.push({ id: tasks.length + 1, name, status: "Pending" });
  res.json({ message: "Task added", tasks });
});

// Update task status
app.put("/tasks/:id", (req, res) => {
  const task = tasks.find(t => t.id == req.params.id);
  if(task){
    task.status = req.body.status;
    res.json({ message: "Status updated", tasks });
  } else {
    res.status(404).json({ message: "Task not found" });
  }
});

app.listen(PORT, () => console.log(`Server running at
http://localhost:${PORT}`));
```

4️⃣ public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Project Dashboard</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>📊 Project Management Dashboard</h1>

  <h2>Add Task</h2>
  <input type="text" id="taskName" placeholder="Task Name">
  <button onclick="addTask()">Add Task</button>

  <h2>Tasks</h2>
  <div id="taskList"></div>

<script src="script.js"></script>
</body>
</html>
```

5️⃣ public/style.css

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background: #f5f5f5;
  padding: 20px;
}

input, select {
  padding: 5px;
  margin: 5px;
}

button {
  padding: 5px 10px;
  margin: 5px;
  border-radius: 5px;
  background: #007bff;
  color: white;
  border: none;
  cursor: pointer;
}

button:hover {
  background: #0056b3;
}

.task-item {
  background: white;
  margin: 10px auto;
  padding: 10px;
  border-radius: 10px;
  width: 70%;
  max-width: 400px;
  text-align: left;
}
```

---

6 public/script.js

```
let tasks = [];

async function loadTasks() {
  const res = await fetch("/tasks");
  tasks = await res.json();
  displayTasks();
}

function displayTasks() {
  const container = document.getElementById("taskList");
  container.innerHTML = "";

  tasks.forEach(task => {
    const div = document.createElement("div");
    div.className = "task-item";
    div.innerHTML = `
      <strong>${task.name}</strong> - Status:
      <select onchange="updateStatus(${task.id}, this.value)">
```

```
        <option value="Pending"
${task.status=="Pending"?'selected':''}>Pending</option>
        <option value="InProgress"
${task.status=="InProgress"?'selected':''}>InProgress</option>
        <option value="Completed"
${task.status=="Completed"?'selected':''}>Completed</option>
      </select>
    `;
    container.appendChild(div);
  });
}

async function addTask() {
  const name = document.getElementById("taskName").value;
  if(!name){ alert("Enter task name"); return; }

  await fetch("/tasks", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ name })
  });

  document.getElementById("taskName").value = "";
  loadTasks();
}

async function updateStatus(id, status) {
  await fetch(`/tasks/${id}`, {
    method: "PUT",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ status })
  });
  loadTasks();
}

window.onload = loadTasks;
```

🖥 How to Run

1. Terminal → project folder
2. Run:

```
node server.js
```

3. Open browser → http://localhost:3000

✅ Expected Output

- Users can **add tasks**.

- Tasks are displayed with a **dropdown to change status**.
- Status can be changed among **Pending, InProgress, Completed**.
- Changes appear immediately.

Perfect! Let's do **Experiment 8 – Online Survey Application** in the **simplest possible way**.

Experiment 8: Online Survey Application

Objective

Create a simple survey app where users can **answer 5 random questions** from a predefined collection and **see their answers**.
No database — all data stored in memory.

---

1 Folder Structure

```
survey-app/
│
├── server.js
├── package.json
│
├── public/
│   ├── index.html
│   ├── style.css
│   └── script.js
```

---

2 Initialize Project

```
npm init -y
npm install express
```

---

3 server.js

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

app.use(express.static("public"));
app.use(express.json());

const questions = [
  "What is your favorite color?",
  "What is your hobby?",
  "Which country would you like to visit?",
  "What is your favorite food?",
  "Which subject do you like most?",
  "What is your favorite movie?",
  "What is your dream job?",
  "Do you prefer tea or coffee?",
  "Which is your favorite sport?",
  "What is your favorite book?"
];

let responses = [];

// Get 5 random questions
app.get("/questions", (req, res) => {
  const shuffled = questions.sort(() => 0.5 - Math.random());
  const selected = shuffled.slice(0, 5);
  res.json(selected);
});

// Submit answers
app.post("/submit", (req, res) => {
  const { answers } = req.body;
  responses.push(answers);
  res.json({ message: "Survey submitted successfully", answers });
});

app.listen(PORT, () => console.log(`Server running at
http://localhost:${PORT}`));
```

---

4 public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Online Survey</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>📋 Online Survey</h1>

  <div id="surveyForm"></div>
  <button onclick="submitSurvey()">Submit</button>

<script src="script.js"></script>
</body>
</html>
```

---

5 public/style.css

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background: #f5f5f5;
  padding: 20px;
}

input {
  padding: 5px;
  margin: 5px 0;
  width: 50%;
}

button {
  padding: 5px 10px;
  margin: 10px 0;
  border-radius: 5px;
  background: #007bff;
  color: white;
  border: none;
  cursor: pointer;
}

button:hover {
  background: #0056b3;
}

.question-item {
  background: white;
  margin: 10px auto;
  padding: 10px;
  border-radius: 10px;
  width: 70%;
  max-width: 500px;
  text-align: left;
}
```

---

6 public/script.js

```
let questions = [];

async function loadQuestions() {
  const res = await fetch("/questions");
  questions = await res.json();
  displayQuestions();
}

function displayQuestions() {
  const container = document.getElementById("surveyForm");
  container.innerHTML = "";

  questions.forEach((q, i) => {
    const div = document.createElement("div");
    div.className = "question-item";
    div.innerHTML = `
      <strong>Q${i+1}: ${q}</strong><br>
      <input type="text" id="answer${i}" placeholder="Your answer">
    `;
    container.appendChild(div);
  });
}

async function submitSurvey() {
  const answers = questions.map((q, i) =>
document.getElementById(`answer${i}`).value);
  if(answers.some(a => a.trim() === "")) { alert("Answer all questions");
return; }

  const res = await fetch("/submit", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ answers })
  });

  const data = await res.json();
  alert(data.message);
  loadQuestions(); // load new 5 random questions
}

window.onload = loadQuestions;
```

🖥 How to Run

1. Terminal → project folder
2. Run:

```
node server.js
```

3. Open browser → http://localhost:3000

✅ Expected Output

- Users see **5 random questions**.
- Users can **enter answers** and submit.
- After submission, a **success alert** appears.
- New random questions load automatically.
