**EX:2**

**DATE:**

**Development of a NodeJS server using Express that stores data from a form as a JSON file and displays it in another page. The redirect page should be prepared using Handlebars.**

**AIM:**

To create a web application using the Express framework that accepts user data from an HTML form, stores this data persistently in a JSON file, and then displays all stored data on a separate page using the Handlebars templating engine.

**PROCEDURE:**

**1.Project Initialization:** Create a new project folder (e.g., express-form-app) and navigate into it.

**2.Initialize npm:** Run npm init -y to create a package.json file.

**3.Install Dependencies:** Install Express and Handlebars by running npm install express express-handlebars.

**4.Create Directory Structure:** Create a views folder, and inside it, create a layouts folder.

**5.Create Server and Data Files:** Create app.js in the root and data.json (with [] inside) in the root.

**6.Create Handlebars Layout:** Create views/layouts/main.hbs and add basic HTML structure with a {{{body}}} placeholder.

**7.Create Form View:** Create views/home.hbs to contain the HTML form for data submission.

**8.Create Display View:** Create views/display.hbs to display the list of submitted data using Handlebars' {{#each}} helper.

**9.Setup Express App:** In app.js, import and initialize Express.

**10.Configure View Engine:** Configure express-handlebars as the view engine for the application.

**11.Add Middleware:** Use express.urlencoded() middleware to parse incoming form data.

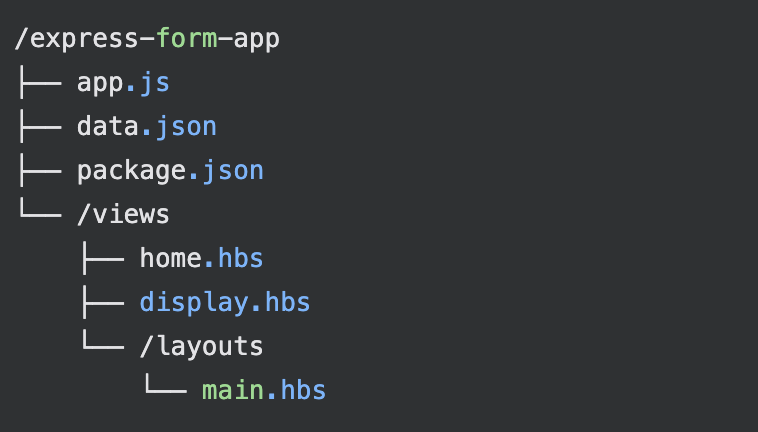
**12.Implement Routes:** Create three routes: GET / to show the form, POST /submit to process and save data, and GET /data to display the saved data.

**13.Implement File I/O:** In the /submit and /data routes, use the fs module to read from and write to data.json.

**14.Start the Server:** Make the Express app listen on port 3000 and log a confirmation message.

**15.Run and Test:** Execute node app.js, navigate to http://localhost:3000, submit the form, and verify that the data is displayed correctly on the /data page.

**DESIGN:**

****

1.You start the application by running the command node server.js in your terminal from the EX1 directory.

2.The server.js script executes, creating a web server that starts listening for requests on port 3000.

3.A user opens their web browser and navigates to the URL http://localhost:3000/.

4.The Node.js server receives a request for the root path (/).

5.The server's logic is programmed to map this root request to the index.html file.

6.The server reads the index.html file from its directory and sends its content back to the browser as a response.

7.The browser receives the HTML and begins to render it. While parsing the HTML, it discovers tags linking to /style.css and /script.js.

8.The browser then sends two new, separate requests to the server: one for /style.css and one for /script.js.

9.The server handles these requests by finding the corresponding files in its directory and sending their content back.

10.The browser receives the CSS and applies it to the HTML, then receives and executes the JavaScript. The final, styled, and interactive webpage is then displayed to the user.

**IMPLEMENTATION:**

**1.**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Form App</title>

<style>

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: #f4f6f8;

margin: 0;

padding: 0;

}

.form-container, .data-container {

max-width: 400px;

margin: 60px auto;

background: #fff;

padding: 2rem 2.5rem 2rem 2.5rem;

border-radius: 12px;

box-shadow: 0 4px 24px rgba(0,0,0,0.08);

}

.form-container h1, .data-container h1 {

text-align: center;

margin-bottom: 1.5rem;

font-size: 2rem;

color: #222;

}

.styled-form .form-group {

margin-bottom: 1.2rem;

display: flex;

flex-direction: column;

}

.styled-form label {

margin-bottom: 0.4rem;

font-weight: 500;

color: #333;

}

.styled-form input[type="text"],

.styled-form input[type="email"] {

padding: 0.6rem;

border: 1px solid #ccc;

border-radius: 6px;

font-size: 1rem;

transition: border 0.2s;

}

.styled-form input:focus {

border-color: #007bff;

outline: none;

}

.styled-form button {

width: 100%;

padding: 0.7rem;

background: #007bff;

color: #fff;

border: none;

border-radius: 6px;

font-size: 1.1rem;

font-weight: 600;

cursor: pointer;

transition: background 0.2s;

}

.styled-form button:hover {

background: #0056b3;

}

*/\* Data page styles \*/*

.submissions-list {

display: flex;

flex-direction: column;

gap: 1rem;

margin-bottom: 1.5rem;

}

.submission-card {

background: #f8fafc;

border: 1px solid #e0e0e0;

border-radius: 8px;

padding: 1rem 1.2rem;

box-shadow: 0 2px 8px rgba(0,0,0,0.03);

display: flex;

flex-direction: column;

gap: 0.3rem;

}

.back-btn {

display: inline-block;

text-align: center;

width: 100%;

padding: 0.7rem;

background: #6c757d;

color: #fff;

border: none;

border-radius: 6px;

font-size: 1.1rem;

font-weight: 600;

text-decoration: none;

margin-top: 0.5rem;

transition: background 0.2s;

}

.back-btn:hover {

background: #495057;

}

</style>

</head>

<body>

{{{*body*}}}

</body>

</html>

**2.**

<div class="data-container">

<h1>Submitted Data</h1>

<div class="submissions-list">

{{#if *submissions.length*}}

{{#each *submissions*}}

<div class="submission-card">

<div><strong>Name:</strong> {{*this.name*}}</div>

<div><strong>Email:</strong> {{*this.email*}}</div>

<div><strong>Phone:</strong> {{*this.phone*}}</div>

<div><strong>Address:</strong> {{*this.address*}}</div>

<div><strong>Age:</strong> {{*this.age*}}</div>

<div><strong>Gender:</strong> {{*this.gender*}}</div>

{{#if *this.comments*}}

<div><strong>Comments:</strong> {{*this.comments*}}</div>

{{/if}}

</div>

{{/each}}

{{else}}

<p>No submissions yet.</p>

{{/if}}

</div>

<a href="/" class="back-btn">Back to form</a>

</div>

**3.**

<div class="form-container">

<h1>Submit your details</h1>

<form action="/submit" method="POST" class="styled-form">

<div class="form-group">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required>

</div>

<div class="form-group">

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

</div>

<div class="form-group">

<label for="phone">Phone:</label>

<input type="tel" id="phone" name="phone" pattern="[0-9]{10}" required>

</div>

<div class="form-group">

<label for="address">Address:</label>

<input type="text" id="address" name="address" required>

</div>

<div class="form-group">

<label for="age">Age:</label>

<input type="number" id="age" name="age" min="1" max="120" required>

</div>

<div class="form-group">

<label for="gender">Gender:</label>

<select id="gender" name="gender" required>

<option value="">Select</option>

<option value="Male">Male</option>

<option value="Female">Female</option>

<option value="Other">Other</option>

</select>

</div>

<div class="form-group">

<label for="comments">Comments:</label>

<textarea id="comments" name="comments" rows="3"></textarea>

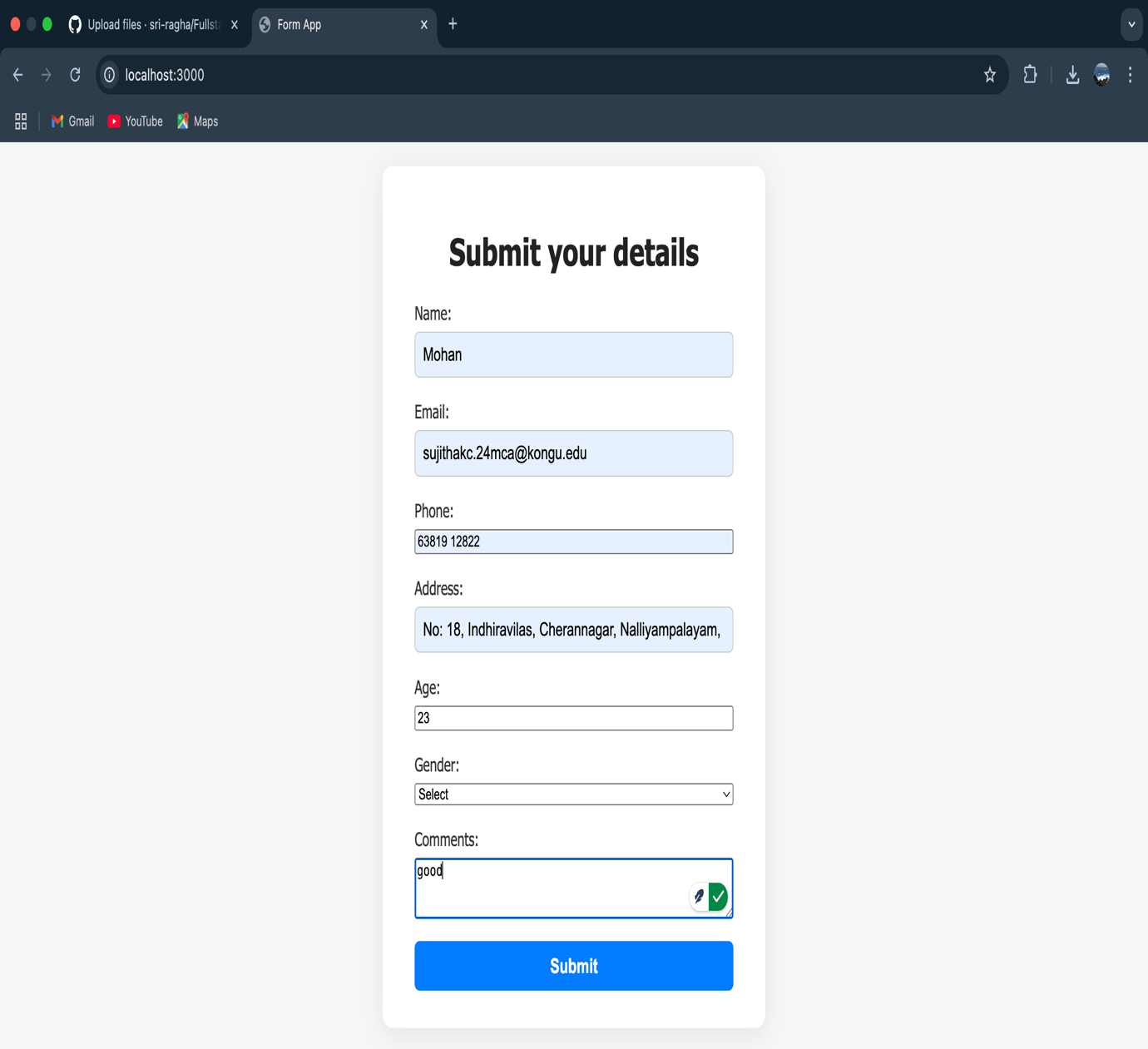
</div>

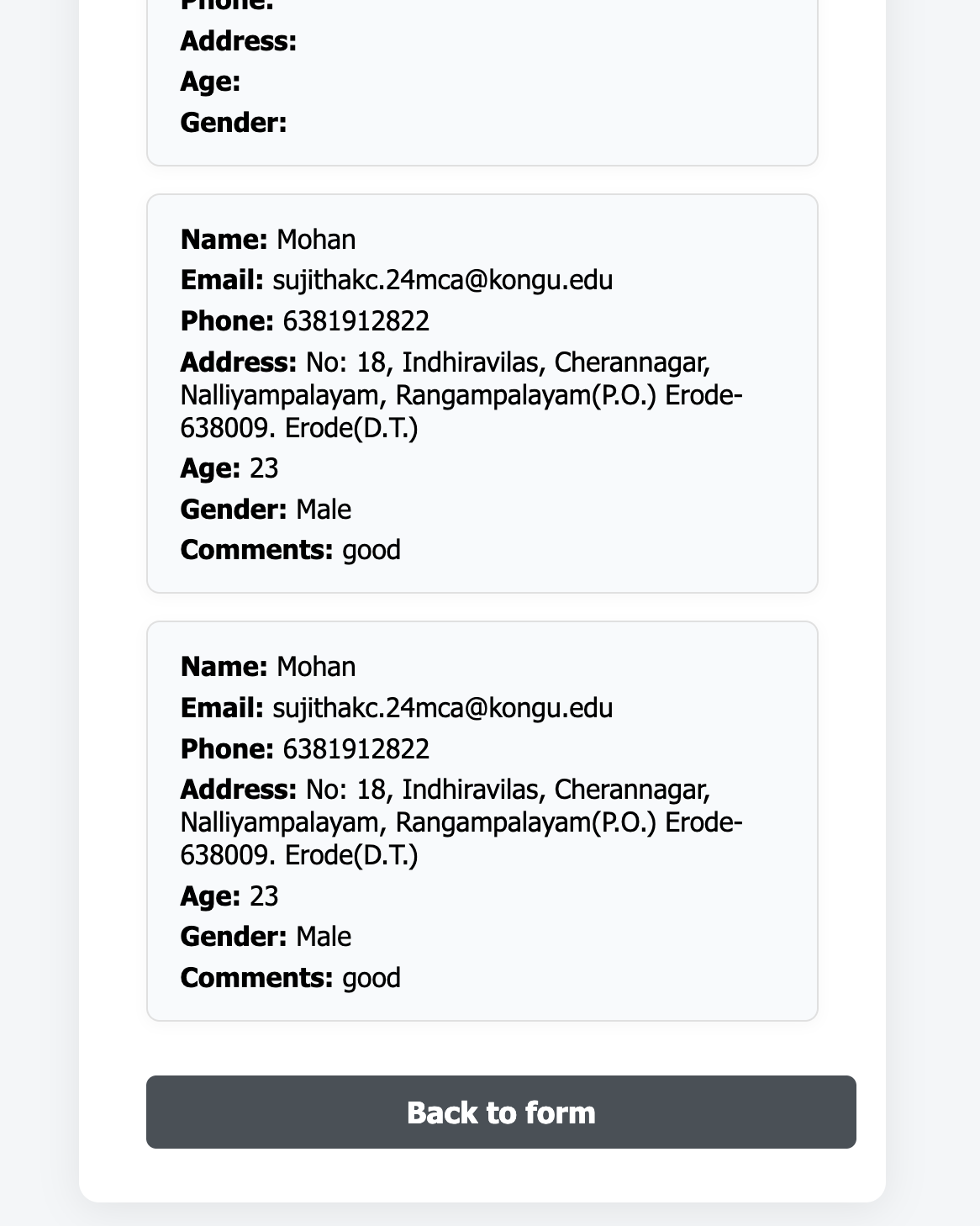
<button type="submit">Submit</button>

</form>

</div>

**OUTPUTSCREEN SHOT:**

****

****

|  |  |
| --- | --- |
| **COE(30)** |  |
| **RECORD(20)** |  |
| **VIVA(10)** |  |
| **TOTAL(60)** |  |

**RESULT:**

The Express application was successfully built. It correctly displays a form, captures user input via a POST request, saves the data to a data.json file, and then redirects to a new page where all submitted data is dynamically rendered using Handlebars templates. The application successfully fulfills all requirements and hosted in <http://localhost:3000/>.