Problem statement:

Consider the basic concepts of Express.js, which are useful in the creation of an application.

Considering the following points, demonstrate the functionality of each with a simple script

1) Serving static files using Express.js: With the help of Built in middleware, express. Static () to demonstrate the usage of serving static files in express.

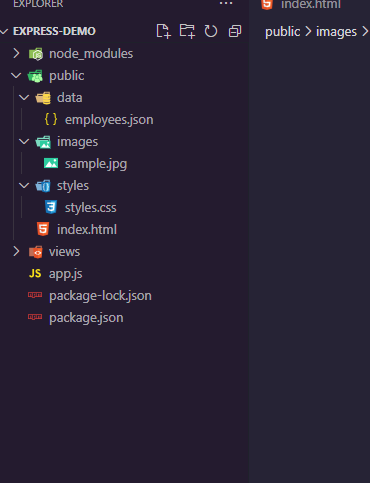
To demonstrate the above make a use of

· Use of images where it should accept any type of image

· Use of CSS and HTML files.

· Make a Use json file of employee information, add file to the static folder, and show the response on the browser.

Note:Assume your own data whenever required to perform the operation.



<!DOCTYPE *html*>

<html *lang*="en">

<head>

<meta *charset*="UTF-8">

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

<title>Static Files </title>

<link *rel*="stylesheet" *href*="./styles/styles.css">

</head>

<body>

<h1>Welcome to the Static Files </h1>

<img *src*="/images/sample.jpg" *alt*="Sample Image">

<script>

fetch('/employees')

.then(*response* => *response*.json())

.then(*data* => {

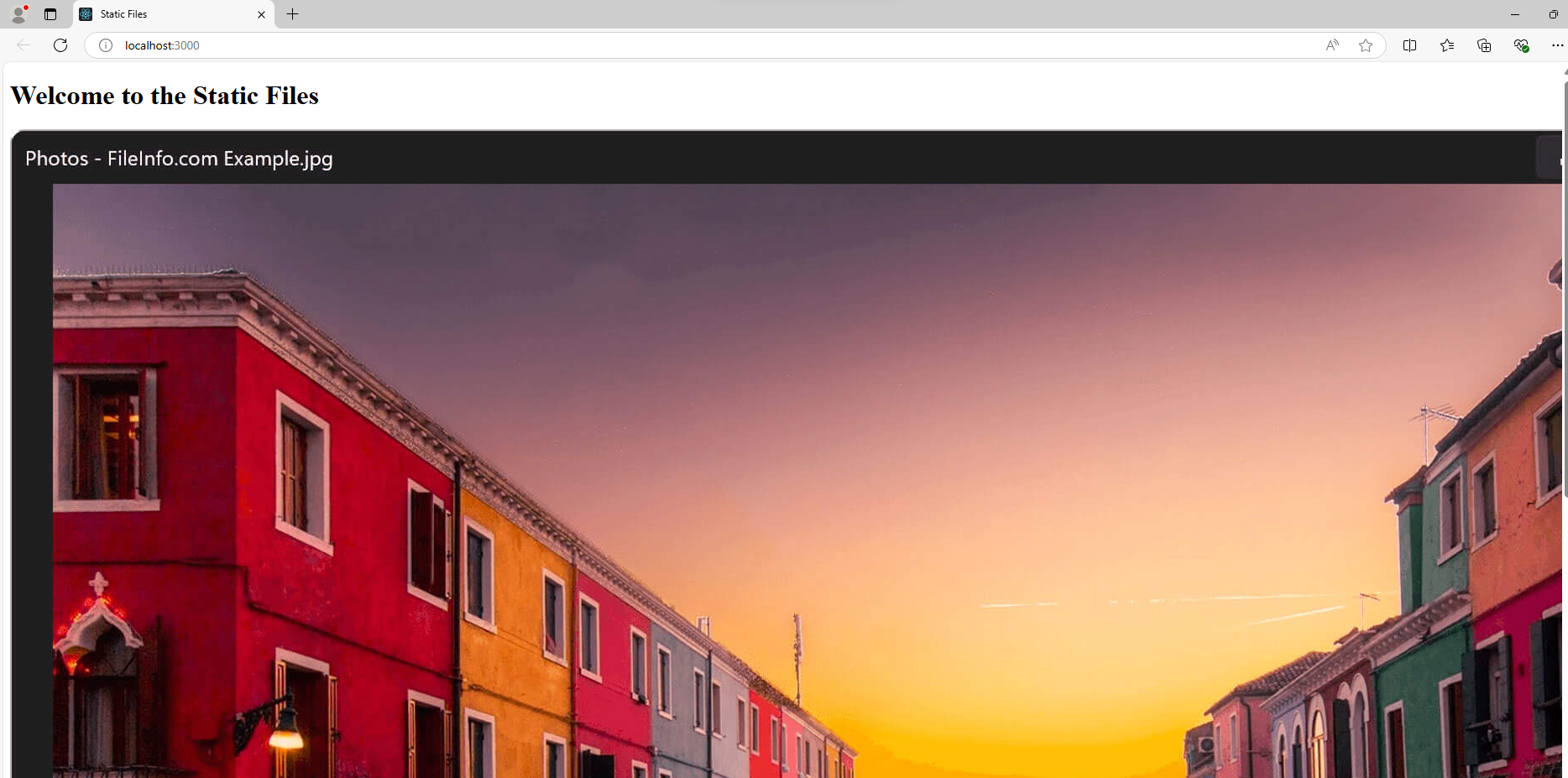
console.log(*data*); *// Log employee data to the console*

});

</script>

</body>

</html>



<!DOCTYPE *html*>

<html *lang*="en">

<head>

<meta *charset*="UTF-8">

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

<title>Static Files</title>

<link *rel*="stylesheet" *href*="/styles/style.css">

</head>

<body>

<h1>Welcome to the Static Files</h1>

<div *id*="employee-list">

*<!-- Employee details will be injected here -->*

</div>

<script>

*// Fetch employee data from the server*

fetch('/employees')

.then(*response* => *response*.json())

.then(*data* => {

console.log(*data*); *// Log employee data to the console*

*// Get the container where employee data will be displayed*

const employeeListDiv = document.getElementById('employee-list');

*// Check if data is available*

if (*data*.length > 0) {

*// Create a list to display employee details*

let html = '<ul>';

*// Iterate over the data and create list items*

*data*.forEach(*employee* => {

html += `<li>Name: ${*employee*.name}, Position: ${*employee*.position}</li>`;

});

html += '</ul>';

*// Inject the list into the div*

employeeListDiv.innerHTML = html;

} else {

*// Display a message if no employees are found*

employeeListDiv.innerHTML = '<p>No employees found.</p>';

}

})

.catch(*error* => {

console.error('Error fetching employee data:', *error*);

});

</script>

</body>

</html>

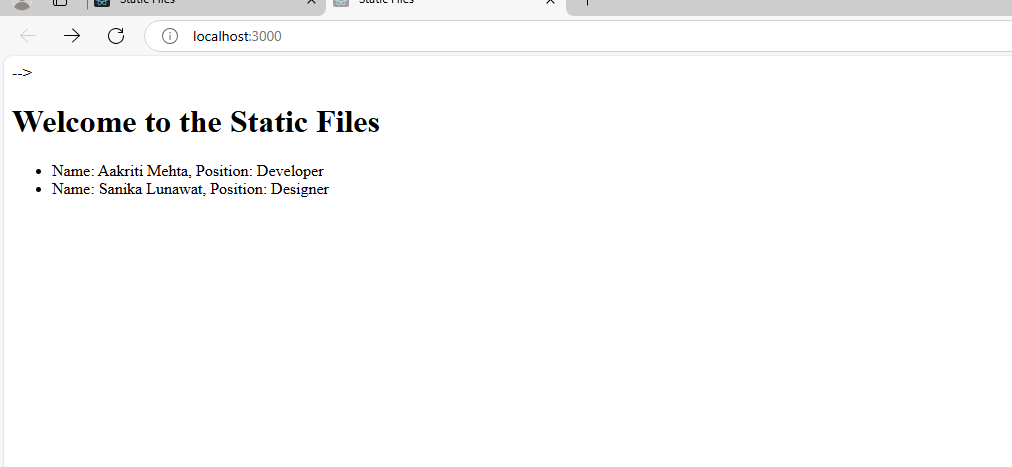
Json file

[

{ "id": 1, "name": "Aakriti Mehta", "position": "Developer" },

{ "id": 2, "name": "Sanika Lunawat", "position": "Designer" }

]

  
2) Implement any one Template engine (ejs/hbs/pug) to cater the dynamic content using Express Js.

3) Scaffolding:

1. Demonstrate express scaffolding to fulfill the following requirements.

Example: Consider Grocery Delivery Application and demonstrate the Scaffolding

Scaffold the application to create different routes such as.

Sign up Page: (Root/ Homepage)



Title: Implementation of Express. js

Problem statement:

Consider the basic concepts of Express.js, which are useful in the creation of an application.

Considering the following points, demonstrate the functionality of each with a simple script

1) Serving static files using Express.js: With the help of Built in middleware, express. Static () to demonstrate the usage of serving static files in express.

To demonstrate the above make a use of

· Use of images where it should accept any type of image

· Use of CSS and HTML files.

· Make a Use json file of employee information, add file to the static folder, and show the response on the browser.

Note:Assume your own data whenever required to perform the operation.

index.html   
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Static Files </title>

<link rel="stylesheet" href="/style.css">

</head>

<body>

<h1>Welcome to the Static Files - Image + Info </h1>

<img src="/image.jpg" alt="Sample Image">

<div id="employee-info"></div>

<script>

fetch('/employees.json')

.then(response => response.json())

.then(data => {

const employeeDiv = document.getElementById('employee-info');

employeeDiv.innerHTML = `<h2>Employee List:</h2><ul>${data.employees.map(emp => `<li>${emp.name} - ${emp.position}</li>`).join('')}</ul>`;

});

</script>

</body>

</html>

Style.css  
body {

font-family: Arial, sans-serif;

text-align: center;

}

img {

width: 300px;

height: auto;

}

Employees.json  
{

"employees": [

{ "name": "Aakriti ", "position": "Software Engineer" },

{ "name": "Esha", "position": "Data Science Engineer " },

{ "name": "Kavya", "position": "Manager" }

]

}

app.js  
  
const express = require('express');

const path = require('path');

const app = express();

const port = 3000;

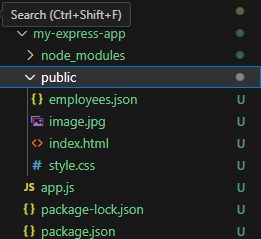
app.use(express.static(path.join(\_\_dirname, 'public')));

// Start the server

app.listen(port, () => {

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

});


2) Implement any one Template engine (ejs/hbs/pug) to cater the dynamic content using Express Js.  
  
Index.ejs  
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Dynamic EJS </title>

<link rel="stylesheet" href="/style.css">

</head>

<body>

<h1> Dynamic EJS </h1>

<img src="/image.jpg" alt="Sample Image">

<h2>Employee List:</h2>

<ul>

<% employees.forEach(function(employee) { %>

<li><%= employee.name %> - <%= employee.position %></li>

<% }); %>

</ul>

</body>

</html>

app.js  
const express = require('express');

const path = require('path');

const app = express();

const port = 3000;

// Set the view engine to EJS

app.set('view engine', 'ejs');

app.set('views', path.join(\_\_dirname, 'views'));

// Serve static files from the 'public' directory

app.use(express.static(path.join(\_\_dirname, 'public')));

// Route to render the EJS template

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

// Sample data to be passed to the EJS template

const employees = [

{ name: "Aakriti", position: "Software Engineer" },

{ name: "Esha", position: "Data Science Engineer" },

{ name: "Kavya", position: "Manager" }

];

// Render the EJS template with data

res.render('index', { employees: employees });

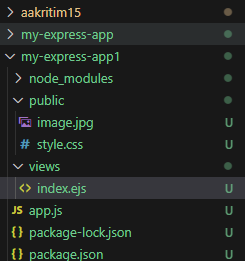
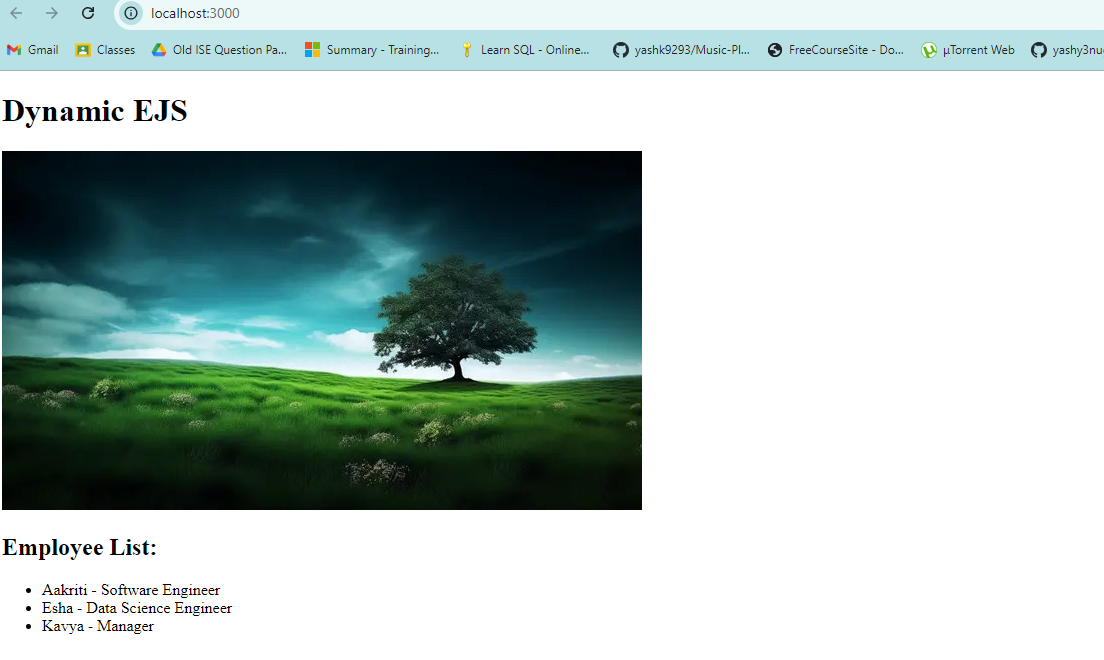
});

// Start the server

app.listen(port, () => {

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

});

3) Scaffolding:

1. Demonstrate express scaffolding to fulfill the following requirements.

Example: Consider Grocery Delivery Application and demonstrate the Scaffolding

Scaffold the application to create different routes such as.

Sign up Page: (Root/ Homepage)

index.ejs   
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Grocery Delivery Home</title>

<link rel="stylesheet" href="/style.css">

</head>

<body>

<h1>Welcome to Grocery Delivery</h1>

<p><a href="/signup">Sign Up</a></p>

</body>

</html>

Signup.ejs  
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Sign Up</title>

<link rel="stylesheet" href="/style.css">

</head>

<body>

<h1>Sign Up for Grocery Delivery</h1>

<form action="/signup" method="post">

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

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

<br>

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

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

<br>

<label for="password">Password:</label>

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

<br>

<button type="submit">Sign Up</button>

</form>

<p><a href="/">Back to Home</a></p>

</body>

</html>

style.css   
body {

font-family: Arial, sans-serif;

text-align: center;

padding: 20px;

}

form {

margin: 0 auto;

max-width: 300px;

}

label {

display: block;

margin-top: 10px;

}

input {

display: block;

width: 100%;

margin-top: 5px;

padding: 8px;

}

button {

margin-top: 10px;

padding: 10px 20px;

font-size: 16px;

}

index.js

const express = require('express');

const router = express.Router();

// Route for homepage

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

res.render('index');

});

module.exports = router;

users.js   
const express = require('express');

const router = express.Router();

// Route for sign-up page

router.get('/signup', (req, res) => {

res.render('signup');

});

// Handle sign-up form submission

router.post('/signup', (req, res) => {

const { name, email, password } = req.body;

// Here you would handle form data, e.g., save to a database

console.log(`Name: ${name}, Email: ${email}, Password: ${password}`);

res.send('Sign-up successful!');

});

module.exports = router;

app.js

const express = require('express');

const path = require('path');

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

const app = express();

const port = 3000;

// Set EJS as the view engine

app.set('view engine', 'ejs');

app.set('views', path.join(\_\_dirname, 'views'));

// Serve static files

app.use(express.static(path.join(\_\_dirname, 'public')));

// Parse form data

app.use(bodyParser.urlencoded({ extended: false }));

// Import and use routes

const indexRoutes = require('./routes/index');

const userRoutes = require('./routes/users');

app.use('/', indexRoutes);

app.use('/', userRoutes);

// Start the server

app.listen(port, () => {

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

});

