**Batch: D - 1 Roll No.: 16010122096**

**Experiment / assignment / tutorial No.06**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| Title: Implementation of Express concept |

**AIM:** To demonstrate the working Express

**Problem Definition:**

**\_\_\_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.**

**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) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Expected OUTCOME of Experiment:

. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Express .js Deep API reference, by Azat Marden, Apress, 2nd edition, 2015.
2. <https://codeburst.io/building-a-rest-api-using-mongo-db-75cac3403fab>
3. <https://www.edureka.co/blog/rest-api-with-node-js/>
4. https://bezkoder.com/node-express-mongodb-crud-rest-api/

**Pre Lab/ Prior Concepts:**

Kindly refer the concepts of Lab/Tut to add the prior concept

**Implementation Details:**

server.js

const express = require('express')

const path = require('path');

const app = express()

const port = 6969

// Middleware to serve static files

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

// Route to serve the HTML file

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

  res.sendFile(path.join(\_\_dirname, 'static-files', 'index.html'));

});

// Start the server

app.listen(port, () => {

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

});

Employees.json

[

    { "name": "John Doe", "position": "Software Engineer" },

    { "name": "Jane Smith", "position": "Project Manager" },

    { "name": "Samuel Brown", "position": "UX Designer" }

]

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 Example</title>

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

</head>

<body>

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

    <img src="./logo.png" alt="Logo">

    <div id="employee-info"></div>

    <script>

        fetch("./employees.json")

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

            .then(data => {

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

                employeeInfo.innerHTML = '<h2>Employee Information:</h2>' +

                '<ul>' +

                data.map(employee => `<li>${employee.name} - ${employee.position}</li>`).join('') +

                '</ul>';

            });

    </script>

</body>

</html>

Styles.css

body {

    font-family: Arial, sans-serif;

    text-align: center;

    margin-top: 50px;

}

h1 {

    color: #333;

}

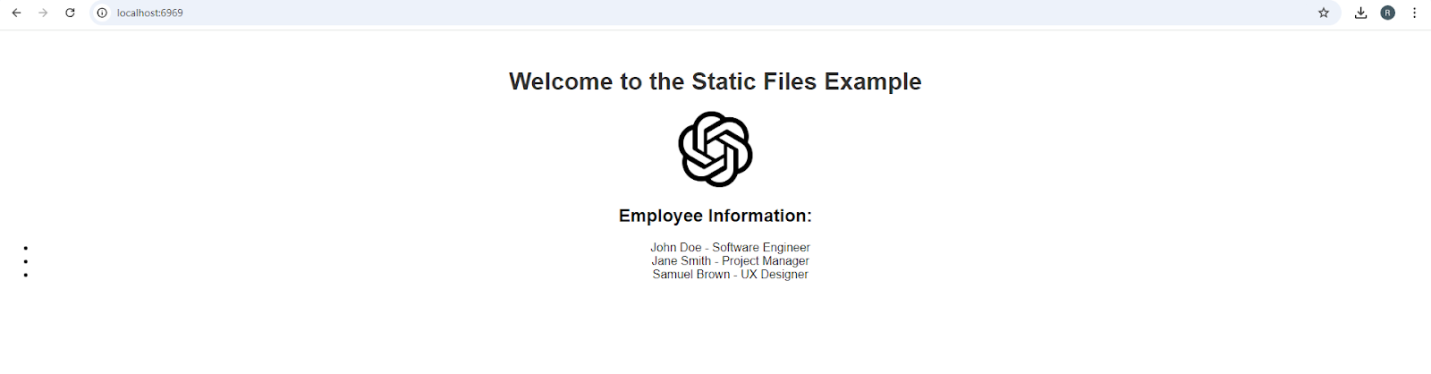
img {

    width: 100px;

    height: auto;

}





server.js

const express = require('express');

const path = require('path');

const app = express();

const port = 3000;

// Set up EJS view engine

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

app.set('views', path.join(\_\_dirname, 'views')); // Ensure this path is correct

// Middleware to serve static files

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

// Route to render the EJS template

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

  const employees = [

    { name: 'John Doe', position: 'Software Engineer' },

    { name: 'Jane Smith', position: 'Project Manager' },

    { name: 'Samuel Brown', position: 'UX Designer' }

  ];

  res.render('index', { title: 'Dynamic Content Example', employees });

});

// Start the server

app.listen(port, () => {

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

});

Index.ejs

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title><%= title %></title>

    <link rel="stylesheet" href="../static-files/styles.css">

</head>

<body>

    <h1><%= title %></h1>

    <div id="employee-info">

        <h2>Employee Information:</h2>

        <ul>

            <% employees.forEach(employee => { %>

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

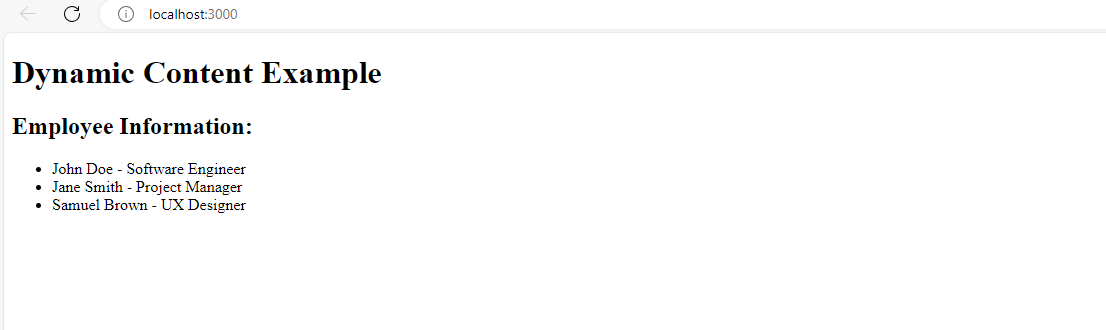
            <% }); %>

        </ul>

    </div>

</body>

</html>



server.js

const express = require('express');

const path = require('path');

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

const app = express();

const port = 3000;

// Set up EJS view engine

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

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

// Middleware to serve static files

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

// Middleware to parse form data

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

// Route for homepage

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

  res.render('home', { title: 'Grocery Delivery Home' });

});

// Route for sign-up page

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

  res.render('signup', { title: 'Sign Up' });

});

// Handle signup form submission

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

  const { username, password } = req.body;

  // Here you would normally handle user creation and validation

  // Redirect to the dashboard after signup

  res.redirect('/dashboard');

});

// Route for user dashboard (after signing up)

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

  res.render('dashboard', { title: 'User Dashboard' });

});

// Start the server

app.listen(port, () => {

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

});

home.js

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title><%= title %></title>

    <link rel="stylesheet" href="../public/css/dashboard.css">

</head>

<body>

    <header>

        <h1><%= title %></h1>

    </header>

    <nav>

        <a href="/">Home</a>

        <a href="/signup">Sign Up</a>

        <a href="/dashboard">Dashboard</a>

    </nav>

    <div class="container">

        <p>This is the homepage for the Grocery Delivery Application.</p>

    </div>

</body>

</html>

signup.js

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title><%= title %></title>

    <link rel="stylesheet" href="../public/css/signup.css">

</head>

<body>

    <header>

        <h1><%= title %></h1>

    </header>

    <nav>

        <a href="/">Home</a>

        <a href="/signup">Sign Up</a>

        <a href="/dashboard">Dashboard</a>

    </nav>

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

        <label for="username">Username:</label>

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

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

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

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

    </form>

</body>

</html>

dashboard.js

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title><%= title %></title>

    <link rel="stylesheet" href="../public/css/dashboard.css">

</head>

<body>

    <header>

        <h1><%= title %></h1>

    </header>

    <nav>

        <a href="/">Home</a>

        <a href="/signup">Sign Up</a>

        <a href="/dashboard">Dashboard</a>

    </nav>

    <div class="container">

        <div class="dashboard-info">

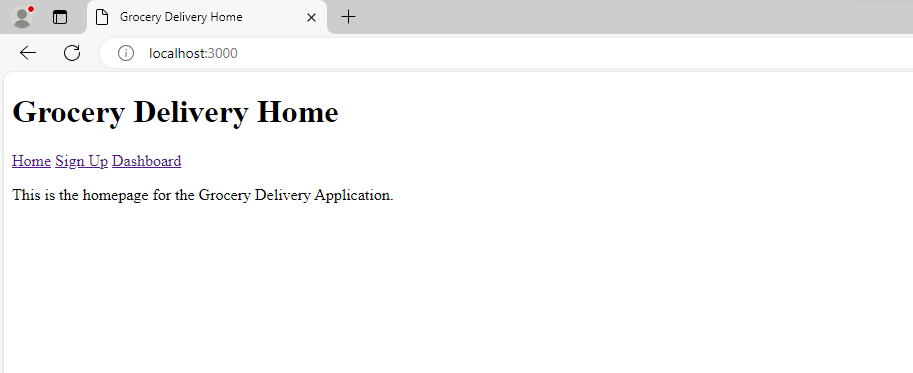
            <p>Welcome to your dashboard. Here you can manage your grocery orders.</p>

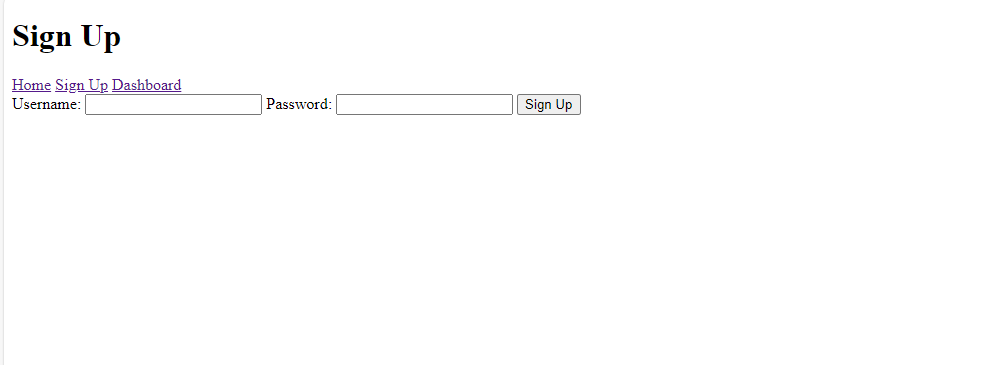
        </div>

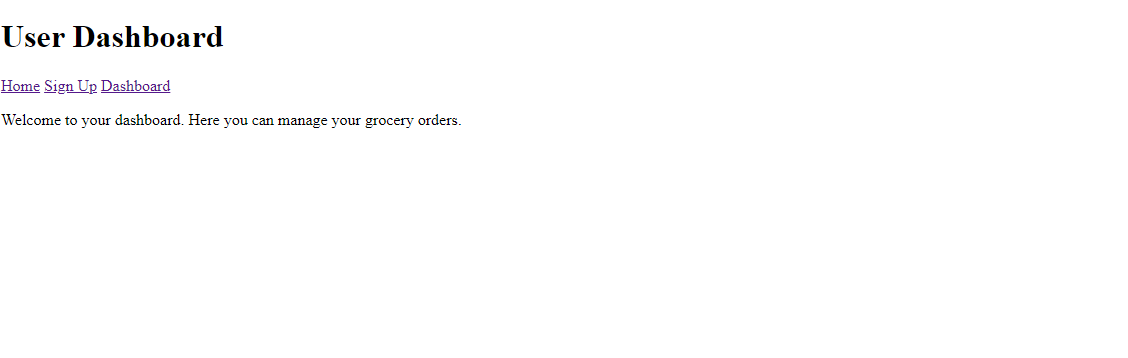
    </div>

</body>

</html>







**Conclusion:**

Express.js simplifies web app development by providing static file serving, template engines, and scaffolding for rapid route creation.