skillzam

ToDo List App using EJS, JSON, NodeJS, ExpressJS

To-Do List Application using EJS , HTML , CSS , Express.js , Node.js , and a JSON file for data storage. This application will allow users to add, modify, and delete items from to-do list.

Step 1: Set Up the Project

1. Initialize the project :

```
mkdir todo-app

cd todo-app

npm init -y
```

2. Install required dependencies :

```
npm install express ejs body-parser
```

3. Create the project structure :

```
todo-app/
├── public/
│   ├── css/
│   │   └── styles.css
├── views/
│   ├── index.ejs
│   └── edit.ejs
├── data.json
├── app.js
└── package.json
```

Step 2: Backend Code

1. app.js

This is the main entry point of the application.

```
const express = require('express');
const fs = require('fs');
const path = require('path');
const bodyParser = require('body-parser');

// Initialize Express app
const app = express();

// Middleware
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'public')));

// Set EJS as the view engine
app.set('view engine', 'ejs');

// Path to JSON file
const DATA_FILE = path.join(__dirname, 'data.json');

// Helper function to read data from JSON file
function readData() {
 const data = fs.readFileSync(DATA_FILE, 'utf8');
 return JSON.parse(data || '[]');
}

// Helper function to write data to JSON file
function writeData(data) {
 fs.writeFileSync(DATA_FILE, JSON.stringify(data, null, 2),
'utf8');
}

// Routes
```

```
// Home page (display to-do list)
app.get('/', (req, res) => {
 const todos = readData();
 res.render('index', { todos });
});

// Add a new to-do item
app.post('/add', (req, res) => {
 const { task } = req.body;
 if (!task) return res.redirect('/');

 const todos = readData();
 todos.push({ id: Date.now(), task });
 writeData(todos);
 res.redirect('/');
});

// Edit a to-do item (render edit form)
app.get('/edit/:id', (req, res) => {
 const { id } = req.params;
 const todos = readData();
 const todo = todos.find((t) => t.id == id);

 if (!todo) return res.redirect('/');

 res.render('edit', { todo });
});

// Update a to-do item
app.post('/update/:id', (req, res) => {
 const { id } = req.params;
 const { task } = req.body;

 const todos = readData();
 const todo = todos.find((t) => t.id == id);
```

```
 if (!todo) return res.redirect('/');

 todo.task = task;
 writeData(todos);
 res.redirect('/');
});

// Delete a to-do item
app.post('/delete/:id', (req, res) => {
 const { id } = req.params;

 const todos = readData();
 const updatedTodos = todos.filter((t) => t.id != id);
 writeData(updatedTodos);
 res.redirect('/');
});

// Start the server
const PORT = 3000;
app.listen(PORT, () => {
 console.log(`Server running on http://localhost:${PORT}`);
});
```

Step 3: Frontend Code

1. `views/index.ejs`

This is the homepage displaying the to-do list.

```
<!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="/css/styles.css">
</head>
<body>
 <div class="container">
   <h1>To-Do List</h1>

   <!-- Add new task form -->
   <form action="/add" method="POST" class="add-form">
     <input type="text" name="task" placeholder="Add a new
task" required>
     <button type="submit">Add</button>
   </form>

   <!-- Display tasks -->
   <ul class="todo-list">
     <% if (todos.length === 0) { %>
       <li>No tasks available.</li>
     <% } else { %>
       <% todos.forEach(todo => { %>
         <li>
           <span><%= todo.task %></span>
           <div class="actions">
             <a href="/edit/<%= todo.id %>"
class="edit-btn">Edit</a>
             <form action="/delete/<%= todo.id %>"
method="POST" style="display:inline;">
               <button type="submit"
class="delete-btn">Delete</button>
             </form>
           </div>
         </li>
       <% }) %>
     <% } %>
   </ul>
 </div>
</body>
```

```
</html>
```

---

2. `views/edit.ejs`

This page allows users to edit an existing task.

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width,
initial-scale=1.0">
 <title>Edit Task</title>
 <link rel="stylesheet" href="/css/styles.css">
</head>
<body>
 <div class="container">
   <h1>Edit Task</h1>

   <!-- Edit task form -->
   <form action="/update/<%= todo.id %>" method="POST"
class="edit-form">
     <input type="text" name="task" value="<%= todo.task %>"
required>
     <button type="submit">Update</button>
   </form>
   <a href="/" class="back-btn">Back to List</a>
 </div>
</body>
</html>
```

---

3. `public/css/styles.css`

Basic styling for the application.

```
body {
 text-align: center;
 color: #333;
```

```
}

.add-form, .edit-form {
 display: flex;
 gap: 10px;
 margin-bottom: 20px;
}

input[type="text"] {
 flex: 1;
 padding: 10px;
 font-size: 16px;
 border: 1px solid #ccc;
 border-radius: 4px;
}

button {
 padding: 10px 15px;
 font-size: 16px;
 background-color: #007bff;
 color: white;
 border: none;
 border-radius: 4px;
 cursor: pointer;
}

button:hover {
 background-color: #0056b3;
}

.todo-list {
 list-style: none;
 padding: 0;
}

.todo-list li {
```

```
 display: flex;
 justify-content: space-between;
 align-items: center;
 padding: 10px;
 border-bottom: 1px solid #ddd;
}

.actions {
 display: flex;
 gap: 10px;
}

.edit-btn, .delete-btn {
 padding: 5px 10px;
 font-size: 14px;
 border: none;
 border-radius: 4px;
 cursor: pointer;
}

.edit-btn {
 background-color: #28a745;
 color: white;
}

.delete-btn {
 background-color: #dc3545;
 color: white;
}

.back-btn {
 display: block;
 margin-top: 20px;
 text-align: center;
 color: #007bff;
 text-decoration: none;
```

```
}

.back-btn:hover {
 text-decoration: underline;
}
```

Step 4: Data Storage (`data.json`)

Create an empty JSON file to store the to-do items:

```
[]
```

Step 5: Run the Application

1. Start the server:

   ```
   node app.js
   ```

2. Open your browser and navigate to `http://localhost:3000`.

Caution:

- The application uses a JSON file (`data.json`) to persist data. In a production environment, you should replace this with a database like MongoDB.