Employee Management Application Report

1. Introduction

This report describes the Employee Management Application, a web-based system designed to manage employee data efficiently. The application provides functionalities to add, view, update, and delete employee records. It also incorporates user authentication to secure access to the data.

2. Application Flow

The following steps outline the application's workflow:

1. **User Access:** The user accesses the application through a web browser.
2. **Token Check:** The application checks if a valid token exists in the browser's localStorage.
    - If a token exists, the application proceeds to fetch and display employee data.
    - If a token does not exist, the user is redirected to the login page (/login.html).
3. **Login:**
    - The user enters their credentials (username and password) on the login page.
    - The application sends a POST request to the server to verify the credentials.
    - Upon successful verification, the server generates a unique token and sends it back to the application.
    - The application stores the token in localStorage and redirects the user to the main page.
4. **Employee Data Management:**
    - The application sends a GET request to the server (including the token for authentication) to retrieve the list of employees.
    - The server retrieves the employee data from the database and sends it back to the application.
    - The application displays the employee data in a tabular format.
5. **Add Employee:**
    - The user fills out the "Add Employee" form with the employee's details (name, position, salary).
    - The application sends a POST request to the server with the new employee data.
    - The server adds the employee to the database and sends a success/failure response.
    - The application updates the employee table with the new data.
6. **Edit Employee:**
    - The user selects an employee from the table to edit.

    - The application displays the employee's details in editable input fields.
    - The user modifies the data and submits the changes.
    - The application sends a PUT request to the server with the updated employee data (including the employee's ID).
    - The server updates the employee's record in the database and sends a success/failure response.
    - The application updates the employee table with the modified data.
7. **Delete Employee:**
    - The user selects an employee from the table to delete.
    - The application prompts the user for confirmation.
    - Upon confirmation, the application sends a DELETE request to the server (including the employee's ID).
    - The server deletes the employee's record from the database and sends a success/failure response.
    - The application removes the employee's row from the table.

3. Technologies Used

The application is built using the following technologies:

- **Frontend:**
    - HTML: Provides the structure of the web pages.
    - CSS (Tailwind CSS): Handles the styling and visual presentation of the application, ensuring a responsive design.
    - JavaScript: Implements the application's logic, including handling user interactions, making API requests, and updating the UI.
- **Backend:**
    - Node.js: The runtime environment.
    - Express.js: A web framework for Node.js, used to build the server-side API.
    - JSON Web Tokens (JWT): Used for user authentication and authorization.
    - MongoDB: The database used to store employee data.

4. Packages Used and Why

- **Tailwind CSS:**
    - Why: Tailwind CSS is a utility-first CSS framework that allows for rapid UI development. It provides a set of pre-defined CSS classes that can be combined to create custom designs without writing CSS from scratch. This promotes consistency and speeds up the development process.
- **Express.js:**
    - Why: Express.js is a lightweight and flexible Node.js web application framework. It provides a robust set of features for building web applications

    and APIs, including routing, middleware support, and request/response handling.
- **JSON Web Tokens (JWT):**
    - Why: JWT is a standard method for representing claims securely between two parties. In this application, JWT is used for authentication. When a user logs in, the server generates a JWT that contains information about the user and sends it to the client. The client then includes this token in the headers of subsequent requests to the server. The server verifies the token to ensure that the user is authorized to access the requested resources.
- **MongoDB:**
    - Why: MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. It's well-suited for applications like this because it allows for a dynamic schema, which can be useful for storing employee data where the fields might vary.

5. Authentication Flow

The application uses JSON Web Tokens (JWT) for authentication. The authentication flow works as follows:

1. **Login Request:** The user sends a login request with their username and password.
2. **Credential Verification:** The server verifies the username and password against the data stored in the database.
3. **Token Generation:** If the credentials are valid, the server generates a JWT. This token contains a payload with user-specific information (e.g., user ID) and is signed using a secret key.
4. **Token Response:** The server sends the JWT back to the client.
5. **Token Storage:** The client stores the JWT in localStorage.
6. **Subsequent Requests:** For every subsequent request to the server, the client includes the JWT in the Authorization header of the request.
7. **Token Verification:** The server verifies the JWT's signature and extracts the user information from the payload. If the token is valid, the server processes the request. If the token is invalid or expired, the server returns an error, and the client is typically redirected to the login page.

6. Conclusion

The Employee Management Application provides a comprehensive solution for managing employee data, with features for adding, viewing, updating, and deleting records. It uses a combination of frontend and backend technologies to deliver a user-friendly and efficient experience. The application also implements a secure authentication mechanism using JWT to protect sensitive data.