Employee Management System - Angular 15 with SQL Server Connectivity

Project Overview

This project will create an Employee Management System where users can add, view, edit, and delete employee records. The front end will be developed using Angular 15, and the back end will use Node.js with Express to connect to a SQL Server database.

Technologies Used

- Angular 15
- Node.js
- Express
- SQL Server
- TypeScript

Step 1: Setting Up the Angular Project

1. Install Angular CLI:

```bash
npm install -g @angular/cli
```

2. Create a New Angular Project:

```bash
ng new employee-management
cd employee-management
```

3. Generate Components and Services:

```bash
ng generate component employee
ng generate service employee
```

Step 2: Setting Up the Back-End with Node.js and Express

1. Initialize Node.js Project:

```bash
mkdir backend
cd backend
npm init -y
```

2. Install Required Packages:

```bash
npm install express body-parser mssql cors
```

3. Create Server File (server.js):

```javascript
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const sql = require('mssql');

const app = express();
app.use(bodyParser.json());
app.use(cors());

// Database configuration
const dbConfig = {
  user: 'yourUsername',
  password: 'yourPassword',
  server: 'yourServer',
  database: 'yourDatabase',
};

// Connect to the database
sql.connect(dbConfig, (err) => {
  if (err) console.log(err);
  console.log('Connected to the database');
});

// CRUD operations
app.get('/employees', (req, res) => {
  const request = new sql.Request();
  request.query('SELECT * FROM Employees', (err, result) => {
```

```
        if (err) console.log(err);
        res.send(result.recordset);
    });
});

app.post('/employees', (req, res) => {
    const { name, position, department } = req.body;
    const request = new sql.Request();
    request.query(`INSERT INTO Employees (Name, Position, Department) VALUES
('${name}', '${position}', '${department}')`, (err, result) => {
        if (err) console.log(err);
        res.send('Employee added successfully');
    });
});

app.put('/employees/:id', (req, res) => {
    const { name, position, department } = req.body;
    const { id } = req.params;
    const request = new sql.Request();
    request.query(`UPDATE Employees SET Name='${name}', Position='${position}',
Department='${department}' WHERE Id=${id}`, (err, result) => {
        if (err) console.log(err);
        res.send('Employee updated successfully');
    });
});

app.delete('/employees/:id', (req, res) => {
    const { id } = req.params;
    const request = new sql.Request();
    request.query(`DELETE FROM Employees WHERE Id=${id}`, (err, result) => {
        if (err) console.log(err);
        res.send('Employee deleted successfully');
    });
});

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```

Step 3: Connecting Angular with the Back-End

1. Install HttpClient Module:

```bash
npm install @angular/common@latest
```

2. Update App Module (app.module.ts):

```typescript
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';
import { EmployeeComponent } from './employee/employee.component';
import { EmployeeService } from './employee.service';

@NgModule({
 declarations: [
  AppComponent,
  EmployeeComponent
 ],
 imports: [
  BrowserModule,
  HttpClientModule
 ],
 providers: [EmployeeService],
 bootstrap: [AppComponent]
})
export class AppModule { }
```

3. Employee Service (employee.service.ts):

```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
 providedIn: 'root'
})
export class EmployeeService {
```

```
  private apiUrl = 'http://localhost:5000/employees';

  constructor(private http: HttpClient) { }

  getEmployees(): Observable<any> {
    return this.http.get(this.apiUrl);
  }

  addEmployee(employee: any): Observable<any> {
    return this.http.post(this.apiUrl, employee);
  }

  updateEmployee(id: number, employee: any): Observable<any> {
    return this.http.put(`${this.apiUrl}/${id}`, employee);
  }

  deleteEmployee(id: number): Observable<any> {
    return this.http.delete(`${this.apiUrl}/${id}`);
  }
}
```

4. Employee Component (employee.component.ts):

```typescript
import { Component, OnInit } from '@angular/core';
import { EmployeeService } from '../employee.service';

@Component({
  selector: 'app-employee',
  templateUrl: './employee.component.html',
  styleUrls: ['./employee.component.css']
})
export class EmployeeComponent implements OnInit {

  employees: any[] = [];
  employee = { name: '', position: '', department: '' };

  constructor(private employeeService: EmployeeService) { }

  ngOnInit(): void {
    this.getEmployees();
  }
```

```
  getEmployees(): void {
    this.employeeService.getEmployees().subscribe((data: any) => {
      this.employees = data;
    });
  }

  addEmployee(): void {
    this.employeeService.addEmployee(this.employee).subscribe(() => {
      this.getEmployees();
      this.employee = { name: '', position: '', department: '' };
    });
  }

  updateEmployee(id: number): void {
    this.employeeService.updateEmployee(id, this.employee).subscribe(() => {
      this.getEmployees();
      this.employee = { name: '', position: '', department: '' };
    });
  }

  deleteEmployee(id: number): void {
    this.employeeService.deleteEmployee(id).subscribe(() => {
      this.getEmployees();
    });
  }
}
```

5. Employee Component HTML (employee.component.html):

```html
<div>
  <h2>Employee Management</h2>

  <form (submit)="addEmployee()">
    <label>Name:</label>
    <input [(ngModel)]="employee.name" name="name">
    <label>Position:</label>
    <input [(ngModel)]="employee.position" name="position">
    <label>Department:</label>
    <input [(ngModel)]="employee.department" name="department">
    <button type="submit">Add Employee</button>
  </form>
```

```
  <ul>
    <li *ngFor="let employee of employees">
      {{employee.name}} - {{employee.position}} - {{employee.department}}
      <button (click)="deleteEmployee(employee.Id)">Delete</button>
      <button (click)="updateEmployee(employee.Id)">Edit</button>
    </li>
  </ul>
</div>
```

Step 4: Running the Application

1. Start the Back-End Server:

```bash
node server.js
```

2. Start the Angular Application:

```bash
ng serve
```

Summary

This mini-project demonstrates creating a simple Employee Management System with Angular 15 for the front end and Node.js with SQL Server for the back end. The project covers setting up an Angular project, creating a back-end with Node.js and Express, connecting Angular to the back-end, and performing CRUD operations.