**Event Management**

**Source code :**

**Employee Component:**

**css :**

body {

font-family: sans-serif;

background-color: #f5f5f5; /\* Set a light background color \*/

}

.container {

max-width: 500px; /\* Set a maximum width for responsiveness \*/

margin: 0 auto; /\* Center the container horizontally \*/

padding: 20px; /\* Add some padding for breathing room \*/

border-radius: 5px; /\* Add rounded corners for a softer look \*/

background-color: #fff; /\* Set a white background for contrast \*/

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1); /\* Add a subtle shadow depth \*/

}

h2 {

text-align: center; /\* Center the heading \*/

margin-bottom: 20px; /\* Add space below the heading \*/

}

form {

margin-bottom: 0; /\* Remove any default margin below the form \*/

}

.form-group {

margin-bottom: 15px; /\* Add space between form groups \*/

}

label {

display: block; /\* Display labels as block elements for better spacing \*/

margin-bottom: 5px; /\* Add space between labels and inputs \*/

font-weight: bold; /\* Make labels bold for emphasis \*/

}

.form-control {

width: 100%; /\* Ensure inputs take up full width \*/

padding: 10px; /\* Add padding to inputs for visual appeal \*/

border: 1px solid #ccc; /\* Set a border for inputs \*/

border-radius: 3px; /\* Add rounded corners to inputs \*/

box-sizing: border-box; /\* Ensure consistent box sizing \*/

}

button[type="submit"] {

background-color: #007bff; /\* Set primary blue background for the button \*/

color: #fff; /\* Set white text color for the button \*/

border: none; /\* Remove the default button border \*/

padding: 10px 20px; /\* Add padding to the button \*/

border-radius: 5px; /\* Add rounded corners to the button \*/

cursor: pointer; /\* Set a pointer cursor on hover \*/

}

button[type="submit"]:hover {

background-color: #0069d9; /\* Darken the button background on hover \*/

}

**html :**

<div class="container">

<h2>Add Employee</h2>

<form (ngSubmit)="saveEmployee()">

<div class="form-group">

<label for="firstName">First Name:</label>

<input type="text" id="firstName" [(ngModel)]="employee.first\_name"

name="firstName" required>

</div>

<div class="form-group">

<label for="lastName">Last Name:</label>

<input type="text" id="lastName" [(ngModel)]="employee.last\_name"

name="lastName" required>

</div>

<div class="form-group">

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

<input type="email" id="email" [(ngModel)]="employee.email"

name="email" required>

</div>

<button type="submit">Save</button>

</form>

</div>

**Ts:**

import { Component } from '@angular/core';

import { EmployeeService } from '../employee.service';

import { Employee } from '../employee';

import { Router } from '@angular/router';

@Component({

selector: 'app-employee',

templateUrl: './employee.component.html',

styleUrl: './employee.component.css'

})

export class EmployeeComponent {

employee: Employee = new Employee();

constructor(private employeeService: EmployeeService,private router:Router)

{

}

saveEmployee() {

this.employeeService.addEmployee(this.employee).subscribe();

this.router.navigate(['/employee-list']);

console.log(this.employee.id)

}

}

**Employee-details**

**Css**

body {

font-family: sans-serif;

}

.container {

max-width: 500px;

margin: 0 auto;

padding: 20px;

border-radius: 5px;

background-color: #fff;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

h2 {

text-align: center;

margin-bottom: 20px;

}

form {

margin-bottom: 0;

}

.form-group {

margin-bottom: 15px;

}

label {

display: block;

margin-bottom: 5px;

}

.form-control {

width: 100%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 3px;

box-sizing: border-box;

}

.btn-primary {

background-color: #007bff;

border-color: #007bff;

color: #fff;

padding: 10px 20px;

border-radius: 3px;

cursor: pointer;

}

.btn-primary:hover {

background-color: #0069d9;

border-color: #0062cc;

}

**Html :**

<div class="container mt-5">

<h2>{{ isEditMode ? 'Edit Employee' : 'Add New Employee' }}</h2>

<form [formGroup]="employeeForm" (ngSubmit)="saveEmployee()">

<div class="form-group">

<label for="first\_name">First Name:</label>

<input type="text" id="first\_name" formControlName="first\_name"

class="form-control" required>

</div>

<div class="form-group">

<label for="last\_name">Last Name:</label>

<input type="text" id="last\_name" formControlName="last\_name"

class="form-control" required>

</div>

<div class="form-group">

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

<input type="email" id="email" formControlName="email" class="form

control" required>

</div>

<button type="submit" class="btn btn-primary">{{ isEditMode ? 'Update' :

'Add' }} Employee</button>

</form>

</div>

**Ts :**

import { Component, OnInit } from '@angular/core';

import { EmployeeService } from '../employee.service';

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { ActivatedRoute, Router } from '@angular/router';

@Component({

selector: 'app-employee-details',

templateUrl: './employee-details.component.html',

styleUrl: './employee-details.component.css'

})export class EmployeeDetailsComponent implements OnInit {

employeeForm!: FormGroup;

isEditMode: boolean = false;

employeeId!: number;

constructor(

private formBuilder: FormBuilder,

private employeeService: EmployeeService,

private route: ActivatedRoute,

private router: Router

) {}

ngOnInit(): void {

this.employeeId = +this.route.snapshot.paramMap.get('id')!;

this.isEditMode = !!this.employeeId;

this.initForm();

if (this.isEditMode) {

this.employeeService.getEmployeeById(this.employeeId).subscribe(

(employee) => {

this.employeeForm.patchValue(employee);

},

(error) => {

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

}

);

}

}

initForm(): void {

this.employeeForm = this.formBuilder.group({

id: [null],

first\_name: ['', [Validators.required]],

last\_name: ['', [Validators.required]],

email: ['', [Validators.required, Validators.email]],

});

}

saveEmployee(): void {

const employeeData = this.employeeForm.value;

if (this.isEditMode) {

this.employeeService.updateEmployee(employeeData).subscribe(

() => {

console.log('Employee updated successfully.');

this.router.navigate(['/employee-list']);

},

(error) => {

console.error('Error updating employee:', error);

}

);

}

else {

this.employeeService.addEmployee(employeeData).subscribe(

() => {

console.log('Employee added successfully.');

this.router.navigate(['/employee-list']);

},

(error) => {

console.error('Error adding employee:', error);

}

);

}

}

}

**Employee-List**

**Css :**

body {

font-family: sans-serif;

}

.container {

max-width: 800px;

margin: 0 auto;

}

h2 {

text-align: center;

margin-bottom: 40px;

}

table {

width: 100%;

border-collapse: collapse;

margin-bottom: 40px;

}

th, td {

padding: 10px;

text-align: left;

border: 1px solid #ddd;

}

th {

background-color: #f2f2f2;

font-weight: bold;

}

tr:nth-child(even) {

background-color: #f5f5f5;

}

.btn {

margin-right: 5px;

}

.btn-info {

background-color: #007bff;

border-color: #007bff;

color: #fff;

cursor: pointer;

}

.btn-info:hover {

background-color: #0069d9;

border-color: #0062cc;

}

.btn-danger {

background-color: #dc3545;

border-color: #dc3545;

color: #fff;

cursor: pointer;

}

.btn-danger:hover {

background-color: #c82333;

border-color: #bd2130;

}

**Html :**

<div class="container mt-5">

<h2>Employee List</h2>

<table class="table">

<thead>

<tr>

<th>ID</th>

th>First Name</th>

<th>Last Name</th>

<th>Email</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<tr \*ngFor="let employee of employees">

<td>{{ employee.id }}</td>

<td>{{ employee.first\_name }}</td>

<td>{{ employee.last\_name }}</td>

<td>{{ employee.email }}</td>

<td>

<button class="btn btn-info"

(click)="viewEmployee(employee.id)">View</button>

<button class="btn btn-danger"

(click)="deleteEmployee(employee.id)">Delete</button>

</td>

</tr>

<tr>

</tr>

</tbody>

</table>

</div>

**Ts :**

import { Component } from '@angular/core';

import { EmployeeService } from '../employee.service';

import { Router } from '@angular/router';

@Component({

selector: 'app-employee-list',

templateUrl: './employee-list.component.html',

styleUrl: './employee-list.component.css'

})

export class EmployeeListComponent {

employees: any[] = [];

constructor(private employeeService: EmployeeService,private router: Router)

{}

ngOnInit(): void {

this.fetchEmployees();

}

fetchEmployees(): void {

this.employeeService.getEmployees().subscribe(

(data) => {

this.employees = data;

},

(error) => {

console.error('Error fetching employees:', error);

}

);

}

viewEmployee(id: number): void {

// Implement logic to navigate to the employee details page

// You may use router.navigate(['/employee-details', id]) for this

this.router.navigate(['/employee-details',id])

console.log('View Employee with ID:', id);

}

deleteEmployee(id: number): void {

this.employeeService.deleteEmployee(id).subscribe(

() => {

console.log('Employee deleted successfully.');

this.fetchEmployees();

},

(error) => {

console.error('Error deleting employee:', error);

}

);

}

}

**Home**

**Html :**

<h1>Welcome to the Event Management App!</h1>

<ul>

<li>Employee==>Add Employee</li>

<li>Employee Details===>Get particular Employee and update</li>

<li>Employee list===>Get all Employee , have view and delete button</li>

<li>Login===> Validate admin users for Event Management</li>

</ul>

**Login**

**Css**

body {

font-family: sans-serif;

background-color: #f5f5f5;

}

.container {

max-width: 500px;

margin: 0 auto;

padding: 15px;

border-radius: 5px;

background-color: #fff;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

.card {

border: none;

}

.alert.alert-danger {

color: #a94442;

background-color: #f2dede;

border-color: #ebccd1;

padding: 15px;

margin-bottom: 15px;

text-align: center;

font-weight: bold;

}

.card-title {

text-align: center;

margin-bottom: 20px;

}

.form-group {

margin-bottom: 15px;

}

label {

display: block;

margin-bottom: 5px;

}

.form-control {

width: 100%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 3px;

box-sizing: border-box;

}

.btn-primary {

background-color: #007bff;

border-color: #007bff;

color: #fff;

padding: 10px 20px;

border-radius: 3px;

cursor: pointer;

}

.btn-primary:hover {

background-color: #0069d9;

border-color: #0062cc;

}

**Html :**

<div \*ngIf="errorMessage" class="alert alert-danger">{{ errorMessage }}</div>

<div class="container mt-5">

<div class="card">

<div class="card-body">

<h2 class="card-title">Admin Login</h2>

<form (ngSubmit)="login()">

<div class="form-group">

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

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

[(ngModel)]="username" class="form-control" required>

</div>

<div class="form-group">

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

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

[(ngModel)]="password" class="form-control" required>

</div>

<button type="submit" class="btn btn-primary">Login</button>

</form>

</div>

</div>

</div>

**Ts**

import { Component } from '@angular/core';

import { Router } from '@angular/router';

@Component({

selector: 'app-login',

templateUrl: './login.component.html',

styleUrls: ['./login.component.css']

})

export class LoginComponent {

username: string = '';

password: string = '';

errorMessage: string = '';

constructor(private router: Router) {}

login(): void {

// Simulate a backend authentication check

if (this.isValidUser(this.username, this.password)) {

// Navigate to the employee list page on successful login

this.router.navigate(['/employee-list']);

} else {

// Handle login failure

this.errorMessage = 'Invalid username or password. Please try again.';

}

}

private isValidUser(username: string, password: string): boolean {

// In a real application, you would make an API call to a server for

authentication

// For simplicity, let's hardcode valid credentials here

const validUsername = 'admin';

const validPassword = 'password';

return username === validUsername && password === validPassword;

}

}

**App.routing.ts :**

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { LoginComponent } from './login/login.component';

import { EmployeeListComponent } from './employee-list/employee

list.component';

import { EmployeeDetailsComponent } from './employee-details/employee

details.component';

import { EmployeeComponent } from './employee/employee.component';

import { HomeComponent } from './home/home.component';

const routes: Routes = [

{ path: 'home', component: HomeComponent },

{ path: 'login', component: LoginComponent },

{ path: 'employee-list', component: EmployeeListComponent },

{ path: 'employee-details/:id', component: EmployeeDetailsComponent },

{path:'add',component:EmployeeComponent},

{ path: '', redirectTo: '/login', pathMatch: 'full' },

{ path: '\*\*', redirectTo: '/login', pathMatch: 'full' }

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

**App.component.html :**

<a href="/home">Home</a>

<a href="/add">Add Employee</a>

<a href="/employee-list">list of employees</a>

<router-outlet></router-outlet>

**App.module.ts :**

import { NgModule } from '@angular/core';

import { BrowserModule, provideClientHydration } from '@angular/platform

browser';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { LoginComponent } from './login/login.component';

import { EmployeeListComponent } from './employee-list/employee

list.component';

import { EmployeeDetailsComponent } from './employee-details/employee

details.component';

import { RouterModule, Routes } from '@angular/router';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { HttpClientModule } from '@angular/common/http';

import { EmployeeComponent } from './employee/employee.component';

import { HomeComponent } from './home/home.component';

const routes:Routes=[

{ path: 'login', component: LoginComponent },

{ path: 'employee-list', component: EmployeeListComponent },

{ path: 'employee-details/:id', component: EmployeeDetailsComponent },

{path:'add',component:EmployeeComponent},

{ path: 'home', component: HomeComponent }

]

@NgModule({

declarations: [

AppComponent,

LoginComponent,

EmployeeListComponent,

EmployeeDetailsComponent,

EmployeeComponent,

HomeComponent

],

imports: [

BrowserModule,

AppRoutingModule,

FormsModule,

HttpClientModule,

ReactiveFormsModule,

RouterModule.forRoot(routes)

],

providers: [

provideClientHydration()

],

bootstrap: [AppComponent]

})

export class AppModule { }

**Employee.ts :**

export class Employee {

id: number;

first\_name: string;

last\_name: string;

email: string;

}

**Employee.service.ts :**

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { Observable } from 'rxjs';

import { catchError, map } from 'rxjs/operators';

@Injectable({

providedIn: 'root'

})

export class EmployeeService {

private apiUrl = 'http://localhost:3000/employees'; // Adjust the URL

according to your JSON server configuration

constructor(private http: HttpClient) {}

// Get all employees

getEmployees(): Observable<any[]> {

return this.http.get<any[]>(this.apiUrl)

.pipe(

catchError(this.handleError)

);

}

// Get employee by ID

getEmployeeById(id: number): Observable<any> {

const url = `${this.apiUrl}/${id}`;

return this.http.get<any>(url)

.pipe(

catchError(this.handleError)

);

}

// Add a new employee

addEmployee(employee: any): Observable<any> {

return this.http.post<any>(this.apiUrl, employee)

.pipe(

catchError(this.handleError)

);

}

// Update an existing employee

updateEmployee(employee: any): Observable<any> {

const url = `${this.apiUrl}/${employee.id}`;

return this.http.put<any>(url, employee)

.pipe(

catchError(this.handleError)

);

}

// Delete an employee by ID

deleteEmployee(id: number): Observable<any> {

const url = `${this.apiUrl}/${id}`;

return this.http.delete<any>(url)

.pipe(

catchError(this.handleError)

);

}

// Handle errors

private handleError(error: any): Observable<never> {

console.error('An error occurred:', error);

throw error;

}

}