Event Management App

Src:

App :

Dashboard

Dashboard.component.css:

```
.modal-content {
  border-radius: 0;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.modal-title {
  color: #3F51B5;
}

.form-control {
  border-radius: 0;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.form-control:focus {
  border-color: #3F51B5;
  box-shadow: 0 2px 5px rgba(63, 81, 181, 0.5);
}

.form-label {
  color: #3F51B5;
}

.btn-primary {
  background-color: #3F51B5;
  border-color: #3F51B5;
}

.btn-primary:hover {
  background-color: #303F9F;
  border-color: #303F9F;
}
```

```
.btn-primary:focus {
  box-shadow: 0 0 0 0.2rem rgba(63, 81, 181, 0.5);
}

.btn-danger {
  background-color: #DC3545;
  border-color: #DC3545;
}

.btn-danger:hover {
  background-color: #B02A37;
  border-color: #B02A37;
}

.btn-danger:focus {
  box-shadow: 0 0 0 0.2rem rgba(220, 53, 69, 0.5);
}
```

Dashboard.component.html:

```
<nav class="navbar navbar-light bg-light">
    <div class="container-fluid">
        <a class="navbar-brand text-primary" href="#">Employee Master List</a>
        <div class="d-flex justify-content-end">
            <button class="btn btn-outline-success mx-2"
(click)="showModel=true">Add Employee</button>
            <form action="signin">
                <button type="submit" class="btn btn-outline-
warning">Logout</button>
            </form>
            <button class="btn btn-outline-dark btn-sm"
(click)="showPassModel=true">Change Password</button>
        </div>
    </div>
</nav>

<div class="container mt-5 text-center">
    <table class="table table-bordered table-striped table-hover w-75 mx-auto">
        <thead class="thead-dark">
            <tr>
                <th>ID</th>
                <th>First Name</th>
                <th>Last Name</th>
```

```
                <th>Email</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            <tr *ngFor="let row of employeeData">
                <td>{{row.id}}</td>
                <td>{{row.first_name}}</td>
                <td>{{row.last_name}}</td>
                <td>{{row.email}}</td>
                <td>
                    <button type="button" (click)="onEdit(row)" class="btn btn-
outline-warning btn-sm mr-2">Edit</button>
                    <button type="button" (click)="deleteEmployee(row)"
class="btn btn-outline-danger btn-sm">Delete</button>
                </td>
            </tr>
        </tbody>
    </table>
</div>

<!-- Modal for Employee Details -->
<div class="modal fade" id="employeeModal" tabindex="-1" role="dialog"
[ngClass]="{'show':showModel}">
    <!-- ... (Rest of your modal content) -->
</div>

<!-- Modal for Change Password -->
<div class="modal fade" id="passwordModal" tabindex="-1" role="dialog"
[ngClass]="{'show':showPassModel}">
    <!-- ... (Rest of your modal content) -->
</div>
```

Dashboard.component.spec.ts:

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { DashboardComponent } from './dashboard.component';

describe('DashboardComponent', () => {
  let component: DashboardComponent;
  let fixture: ComponentFixture<DashboardComponent>;
```

```
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [DashboardComponent]
    })
      .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(DashboardComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Dashboard.component.ts:

```
import { Component, OnInit } from '@angular/core';
import { CrudHttpService } from '../crud-http.service';
import { FormBuilder, FormGroup } from '@angular/forms';
import { EmployeeModel } from '../employeedashboard-module';

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

  formValue!: FormGroup;
  empModel: EmployeeModel = new EmployeeModel();
  employeeData!: any;

  empList: any = [];
  showModel: any;
  showPassModel: any;

  constructor(private crudHttpService: CrudHttpService, private formBuilder:
FormBuilder) { }
```

```
ngOnInit(): void {
  this.getAllEmployeeDetails();
  this.formValue = this.formBuilder.group({
    id: [''],
    firstname: [''],
    lastname: [''],
    email: ['']
  });
}

postEmployeeDetails(): void {
  this.empModel.id = this.formValue.value.id;
  this.empModel.first_name = this.formValue.value.firstname;
  this.empModel.last_name = this.formValue.value.lastname;
  this.empModel.email = this.formValue.value.email;

  this.crudHttpService.postEmployee(this.empModel).subscribe(
    res => {
      console.log(res);
      alert("Employee Added successfully");
      this.formValue.reset();
      let ref = document.getElementById('cancel');
      ref?.click();
      this.getAllEmployeeDetails();
    },
    err => {
      alert("Something Went Wrong");
    }
  );
}

getAllEmployeeDetails(): void {
  this.crudHttpService.getEmployee().subscribe(
    res => {
      this.formValue.reset();
      this.employeeData = res;
    }
  );
}

deleteEmployee(row: any): void {
  this.crudHttpService.deleteEmployee(row.id).subscribe(
    res => {
      this.getAllEmployeeDetails();
```

```
      }
    );
  }

  onEdit(row: any): void {
    this.empModel.id = row.id;
    this.formValue.controls['id'].setValue(row.id);
    this.formValue.controls['firstname'].setValue(row.first_name);
    this.formValue.controls['lastname'].setValue(row.last_name);
    this.formValue.controls['email'].setValue(row.email);
  }

  updateEmployeeDetails(): void {
    this.empModel.id = this.formValue.value.id;
    this.empModel.first_name = this.formValue.value.firstname;
    this.empModel.last_name = this.formValue.value.lastname;
    this.empModel.email = this.formValue.value.email;

    this.crudHttpService.updateEmployee(this.empModel,
this.empModel.id).subscribe(
      res => {
        this.formValue.reset();
        let ref = document.getElementById('cancel');
        ref?.click();
        this.getAllEmployeeDetails();
      }
    );
  }

  changePass(): void {
    alert("Password Updated Successfully");
  }

}
```

Signin

Signin.component.css:

```
.title {
  font-size: xx-large;
  font-weight: bold;
  color: #3F51B5;
```

```
  text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
}

.form-control {
  border-radius: 0;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.form-control:focus {
  border-color: #3F51B5;
  box-shadow: 0 2px 5px rgba(63, 81, 181, 0.5);
}

.form-label {
  color: #3F51B5;
}

.btn-primary {
  background-color: #3F51B5;
  border-color: #3F51B5;
}

.btn-primary:hover {
  background-color: #303F9F;
  border-color: #303F9F;
}

.btn-primary:focus {
  box-shadow: 0 0 0 0.2rem rgba(63, 81, 181, 0.5);
}
```

Signin.component.html:

```
<section class="vh-100 mt-5">
  <div class="container-fluid h-custom">
      <p class="title text-center">EventManagement Admin Portal</p>
      <div class="row d-flex justify-content-center align-items-center h-100">
          <div class="col-md-9 col-lg-6 col-xl-5">
              <img src="https://wallpapercave.com/wp/28su9mp.jpg" class="img-
fluid rounded" alt="Sample image">
          </div>
          <div class="col-md-8 col-lg-6 col-xl-4 offset-xl-1">
              <form action="dashboard">
```

```
                <div class="divider d-flex align-items-center my-4">
                    <p class="text-center fw-bold mx-3 mb-0">Sign In</p>
                </div>

                <!-- Email input -->
                <div class="form-outline mb-4">
                    <input type="email" id="email" class="form-control form-
control-lg"
                        placeholder="Enter a valid email address" />
                    <label class="form-label" for="email">Email address</label>
                </div>

                <!-- Password input -->
                <div class="form-outline mb-3">
                    <input type="password" id="password" class="form-control
form-control-lg"
                        placeholder="Enter password" />
                    <label class="form-label" for="password">Password</label>
                </div>

                <div class="d-flex justify-content-between align-items-center">
                    <!-- Checkbox -->
                    <div class="form-check mb-0">
                        <input class="form-check-input me-2" type="checkbox"
value="" id="form2Example3" />
                        <label class="form-check-label" for="form2Example3">
                            Remember me
                        </label>
                    </div>
                    <a href="#!" class="text-body">Forgot password?</a>
                </div>

                <div class="text-center text-lg-start mt-4 pt-2">
                    <button onclick="submit()" type="submit" class="btn btn-
primary btn-lg"
                        style="padding-left: 2.5rem; padding-right:
2.5rem;">Login</button>
                </div>

            </form>
        </div>
    </div>
  </div>
</section>
```

Signin.component.spec.ts:

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { SigninComponent } from './signin.component';

describe('SigninComponent', () => {
  let component: SigninComponent;
  let fixture: ComponentFixture<SigninComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [SigninComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(SigninComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Signin.component.ts:

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

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

  ngOnInit() {}

}
```

App-routing.module.ts:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { DashboardComponent } from './dashboard/dashboard.component';
import { SigninComponent } from './signin/signin.component';

const routes: Routes = [
  { path: 'signin', component: SigninComponent },
  { path: 'dashboard', component: DashboardComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
export const routingComponents = [SigninComponent, DashboardComponent];
```

app.component.html:

```
<!-- Correct structure in app.component.html -->
<app-signin></app-signin>
<router-outlet></router-outlet>
```

App.component.spec.ts:

```
import { TestBed } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
```

```
        AppComponent
      ],
    }).compileComponents();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'Event-management-app'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('Event-management-app');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement as HTMLElement;
    expect(compiled.querySelector('.content
span')?.textContent).toContain('Event-management-app app is running!');
  });
});
```

App.component.ts:

```
import { Component } from '@angular/core';
import { CrudHttpService } from './crud-http.service';

@ Component ({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: [ './app.component.css' ]
})
export class AppComponent {
  title = 'Event-management-app';
}
```

App.module.ts:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { SigninComponent } from './signin/signin.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { ReactiveFormsModule } from '@angular/forms';
import { FormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent,
    SigninComponent,
    DashboardComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    ReactiveFormsModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

crud-http.service.spec.ts:

```
import { TestBed } from '@angular/core/testing';

import { CrudHttpService } from './crud-http.service';

describe('CrudHttpService', () => {
  let service: CrudHttpService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(CrudHttpService);
  });

  it('should be created', () => {
```

```
    expect(service).toBeTruthy();
  });
});
```

Crud-http.serveic.ts:

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpErrorResponse, HttpHeaders } from
'@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { catchError, map } from 'rxjs/operators';

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

  apiUrl: string = 'http://localhost:3000/employees';
  headers = new HttpHeaders().set('Content-Type', 'application/json');

  constructor(private http: HttpClient) { }

  postEmployee(data: any): Observable<any> {
    return this.http.post<any>(this.apiUrl, data).pipe(
      map((res: any) => {
        return res;
      }),
      catchError(this.handleError)
    );
  }

  getEmployee(): Observable<any> {
    return this.http.get<any>(this.apiUrl).pipe(
      map((res: any) => {
        return res;
      }),
      catchError(this.handleError)
    );
  }

  updateEmployee(data: any, id: number): Observable<any> {
    return this.http.put<any>(`${this.apiUrl}/${id}`, data).pipe(
      map((res: any) => {
```

```
        return res;
      }),
      catchError(this.handleError)
    );
  }

  deleteEmployee(id: number): Observable<any> {
    return this.http.delete<any>(`${this.apiUrl}/${id}`).pipe(
      map((res: any) => {
        return res;
      }),
      catchError(this.handleError)
    );
  }

  // Handle API errors
  private handleError(error: HttpErrorResponse): Observable<never> {
    if (error.error instanceof ErrorEvent) {
      console.error('An error occurred:', error.error.message);
    } else {
      console.error(`Backend returned code ${error.status}, ` +
        `body was: ${error.error}`);
    }
    return throwError('Something bad happened; please try again later.');
  }
}
```

Employeedashboard-module.spec.ts:

```
import { EmployeeModel } from "./employeedashboard-module";

describe('EmployeedashboardModule', () => {
  it('should create an instance', () => {
    expect(new EmployeeModel()).toBeTruthy();
  });
});
```

Employeedashboard-module.ts:

```
export class EmployeeModel {
    id: number = 0;
```

```
    first_name: string = "";
    last_name: string = "";
    email: string = "";
  }
```

Db.json:

```
{
  "employees" : [
   {
      "id" :  1 ,
      "first_name" :  "Sebastian" ,
      "last_name" :  "Eschweiler" ,
      "email" :  "sebastian@codingthesmartway.com"
   },
   {
      "id" :  2 ,
      "first_name" :  "Steve" ,
      "last_name" :  "Palmer" ,
      "email" :  "steve@codingthesmartway.com"
   },
   {
      "id" :  3 ,
      "first_name" :  "Ann" ,
      "last_name" :  "Smith" ,
      "email" :  "ann@codingthesmartway.com"
   },
   {
      "id" :  4 ,
      "first_name" :  "James" ,
      "last_name" :  "Watt" ,
      "email" :  "James@codingthesmartway.com"
   },
   {
      "id" :  5 ,
      "first_name" :  "Davi" ,
      "last_name" :  "Jones" ,
      "email" :  "davi@codingthesmartway.com"
   },
   {
      "id" :  6 ,
      "first_name" :  "Luther" ,
      "last_name" :  "Chadwick" ,
```

```
      "email" :  "luther@codingthesmartway.com"
    }
 ]
}
```

Package.json:

```
{
  "name" :  "event-management-app" ,
  "version" :  "0.0.0" ,
  "scripts" : {
    "ng" :  "ng" ,
    "start" :  "ng serve" ,
    "build" :  "ng build" ,
    "watch" :  "ng build --watch --configuration development" ,
    "test" :  "ng test"
 },
  "private" :  true ,
  "dependencies" : {
    "@angular/animations" :  "^14.2.0" ,
    "@angular/common" :  "^14.2.0" ,
    "@angular/compiler" :  "^14.2.0" ,
    "@angular/core" :  "^14.2.0" ,
    "@angular/forms" :  "^14.2.0" ,
    "@angular/platform-browser" :  "^14.2.0" ,
    "@angular/platform-browser-dynamic" :  "^14.2.0" ,
    "@angular/router" :  "^14.2.0" ,
    "@fortawesome/fontawesome-free" :  "^6.2.0" ,
    "bootstrap" :  "^5.2.1" ,
    "jquery" :  "^3.6.1" ,
    "popper.js" :  "^1.16.1" ,
    "rxjs" :  "~7.5.0" ,
    "tslib" :  "^2.3.0" ,
    "zone.js" :  "~0.11.4"
 },
  "devDependencies" : {
    "@angular-devkit/build-angular" :  "^14.2.3" ,
    "@angular/cli" :  "~14.2.3" ,
    "@angular/compiler-cli" :  "^14.2.0" ,
    "@types/jasmine" :  "~4.0.0" ,
    "jasmine-core" :  "~4.3.0" ,
    "karma" :  "~6.4.0" ,
    "karma-chrome-launcher" :  "~3.1.0" ,
```

```
    "karma-coverage" :  "~2.2.0" ,
    "karma-jasmine" :  "~5.1.0" ,
    "karma-jasmine-html-reporter" :  "~2.0.0" ,
    "typescript" :  "~4.7.2"
 }
}
```