SOURCE CODE FOR PHASE END PROJECT EVENT MANAGEMENT APPLICATION

ng version

npm install -g json-server

json-server --watch db.json

npm run json:server

http://localhost:3000

http://localhost:3000/employees

npm install @ng-bootstrap/ng-bootstrap

npm install bootstrap

npm install --save-dev @types/bootstrap

ng g c employee-create

ng g c employee-details

ng g c employee-edit

ng g c employee-list

ng g c home

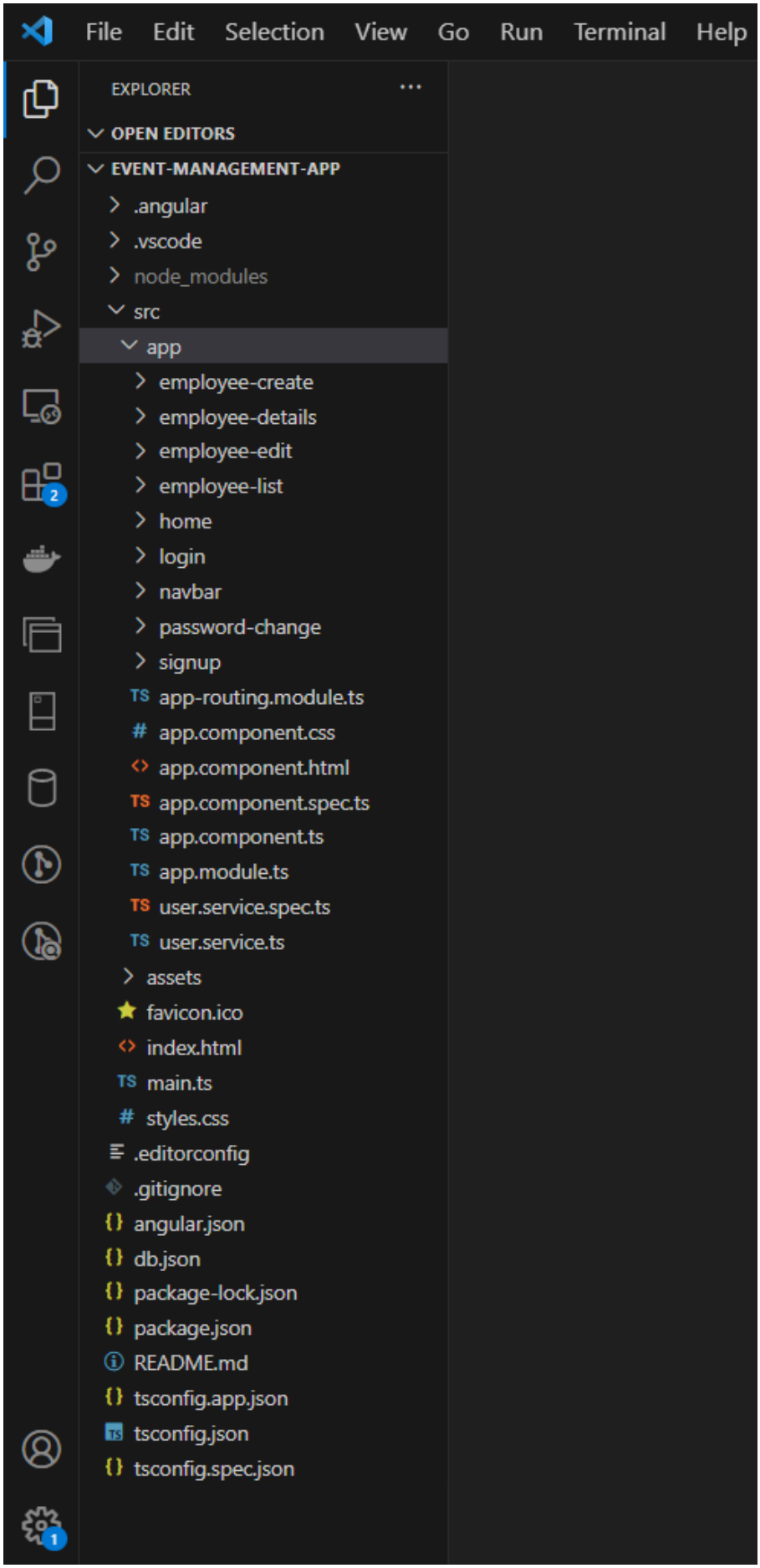
ng g c login

ng g c navbar

ng g c password-change

ng g c signup

ng serve


File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
EVENT-MANAGEMENT-APP
.angular
.vscode
node_modules
src
app
employee-create
employee-details
employee-edit
employee-list
home
login
navbar
password-change
signup
app-routing.module.ts
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.module.ts
user.service.spec.ts
user.service.ts
assets
favicon.ico
index.html
main.ts
styles.css
.editorconfig
.gitignore
angular.json
db.json
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json

employee-create.component.css :

```css
.form-container {
  max-width: 500px;
  margin: auto;
  padding: 20px;
  background-color: #f2f2f2;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.form-group {
  margin-bottom: 20px;
}

label {
  font-weight: bold;
}

input[type="text"],
input[type="email"] {
  width: 100%;
  padding: 8px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.success-message {
  color: green;
  margin-bottom: 10px;
}

.error-message {
  color: red;
  margin-bottom: 10px;
}

button[type="submit"] {
  background-color: #007bff;
  color: #fff;
  border: none;
  margin-left: 160px;
  padding: 10px 20px;
  border-radius: 4px;
  cursor: pointer;
}

button[type="submit"]:hover {
  background-color: #0056b3;
```

```
}
```

employee-create.component.html :

```
<app-navbar></app-navbar>

<div class="form-container">
  <form [formGroup]="employeeForm" (ngSubmit)="onSubmit()">
    <div class="form-group">
      <label for="id">ID</label>
      <input type="text" class="form-control" id="id" formControlName="id"
required>
    </div>
    <div class="form-group">
      <label for="first_name">First Name</label>
      <input type="text" class="form-control" id="first_name"
formControlName="first_name" required>
    </div>
    <div class="form-group">
      <label for="last_name">Last Name</label>
      <input type="text" class="form-control" id="last_name"
formControlName="last_name" required>
    </div>
    <div class="form-group">
      <label for="email">Email</label>
      <input type="email" class="form-control" id="email"
formControlName="email" required>
    </div>
    <button type="submit" class="btn btn-primary">Add Employee</button>
  </form>

  <!-- Success message -->
  <div *ngIf="successMessage" class="success-message">{{ successMessage
}}</div>

  <!-- Error message -->
  <div *ngIf="errorMessage" class="error-message" style="text-align:
center;">{{ errorMessage }}</div>
</div>
```

employee-create.component.spec.ts :

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

import { EmployeeCreateComponent } from './employee-create.component';

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

```
  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [EmployeeCreateComponent]
    });
    fixture = TestBed.createComponent(EmployeeCreateComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

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

employee-create.component.ts :

```
import { Component } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';

@Component({
  selector: 'app-employee-create',
  templateUrl: './employee-create.component.html',
  styleUrls: ['./employee-create.component.css']
})
export class EmployeeCreateComponent {
  employeeForm: FormGroup;
  successMessage: string | null = null;
  errorMessage: string | null = null;

  constructor(
    private http: HttpClient,
    private formBuilder: FormBuilder,
    private router: Router
  ) {
    this.employeeForm = this.formBuilder.group({
      id: ['', Validators.required],
      first_name: ['', Validators.required],
      last_name: ['', Validators.required],
      email: ['', [Validators.required, Validators.email]]
    });
  }

  onSubmit() {
    if (this.employeeForm.invalid) {
      return;
    }
```

```
    const newEmployee = this.employeeForm.value;

    this.http.post('http://localhost:3000/employees', newEmployee)
      .subscribe(
        () => {
          this.errorMessage = null;
          this.router.navigate(['/employees']);
        },
        (error) => {
          if (error.status === 409) {
            this.errorMessage = 'Employee with the same ID already exists';
            this.successMessage = null;
          } else {
            this.errorMessage = 'Error occurred, Please check the id
confliction !';
            this.successMessage = null;
          }
        }
      );
  }
}
```

employee-details.component.css :

```
.container {
  margin-top: 20px;
}

.employee-details {
  width: 100%;
}

.employee-details h2 {
  margin-bottom: 10px;
}

.actions {
  margin-top: 20px;
  margin-left: 16px;
  margin-bottom: 10px;
  width: 100%;
  display: flex;
}

.actions button {
  margin-right: 10px;
}
```

```
.btn-primary {
  background-color: #0dcaf0;
  color: black;
}
```

employee-details.component.html :

```
<app-navbar></app-navbar>

<div class="container">
  <div class="row">
    <div class="col-md-6">
      <div class="card employee-details">
        <div class="card-header">
          <!-- <h2>Employee Details</h2> -->
          <p>
            Details of
            <b>{{ employee?.first_name }} {{ employee?.last_name }}</b>
          </p>
        </div>
        <div class="card-body">
          <ul class="list-group">
            <li class="list-group-item">
              <b>Employee ID:</b
              >          &nb
sp;     
              {{ employee?.id }}
            </li>
            <li class="list-group-item">
              <b>Employee First Name:</b>   {{ employee?.first_name
}}
            </li>
            <li class="list-group-item">
              <b>Employee Last Name:</b>   {{ employee?.last_name }}
            </li>
            <li class="list-group-item">
              <b>Employee Email:</b
              >          &nb
sp;
              {{ employee?.email }}
            </li>
          </ul>
        </div>
        <div class="row">
          <div class="col-md-6">
            <div class="actions">
              <button class="btn btn-primary" (click)="goToEmployeeList()">
                Employee List
```

```
              </button>
              <button class="btn btn-danger" (click)="deleteEmployee()">
                Delete
              </button>
              <button class="btn btn-secondary"
(click)="goToUpdateEmployee()">
                Update
              </button>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

<!-- Delete Confirmation Modal -->
<ng-template #deleteConfirmationModal let-modal>
  <div class="modal-header">
    <h4 class="modal-title">Confirm Delete</h4>
    <button
      type="button"
      class="close"
      aria-label="Close"
      (click)="modal.dismiss()"
    >
      <span aria-hidden="true">&times;</span>
    </button>
  </div>
  <div class="modal-body">
    <p>Are you sure you want to delete this employee?</p>
  </div>
  <div class="modal-footer">
    <button type="button" class="btn btn-secondary" (click)="modal.dismiss()">
      Cancel
    </button>
    <button type="button" class="btn btn-danger" (click)="confirmDelete()">
      Delete
    </button>
  </div>
</ng-template>
```

employee-details.component.spec.ts :

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

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

describe('EmployeeDetailsComponent', () => {
```

```
  let component: EmployeeDetailsComponent;
  let fixture: ComponentFixture<EmployeeDetailsComponent>;

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [EmployeeDetailsComponent]
    });
    fixture = TestBed.createComponent(EmployeeDetailsComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

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

employee-details.component.ts :

```
import { Component, OnInit, ViewChild } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { HttpClient } from '@angular/common/http';
import { NgbModal } from '@ng-bootstrap/ng-bootstrap';

@Component({
  selector: 'app-employee-details',
  templateUrl: './employee-details.component.html',
  styleUrls: ['./employee-details.component.css']
})
export class EmployeeDetailsComponent implements OnInit {
  employee: any;
  @ViewChild('deleteConfirmationModal') deleteConfirmationModal: any;

  constructor(
    private route: ActivatedRoute,
    private router: Router,
    private http: HttpClient,
    private modalService: NgbModal
  ) { }

  ngOnInit() {
    this.route.params.subscribe(params => {
      const employeeId = +params['id'];
      this.getEmployeeDetails(employeeId);
    });
  }

  getEmployeeDetails(employeeId: number) {
    this.http.get<any>(`http://localhost:3000/employees/${employeeId}`)
```

```
      .subscribe(employee => {
        this.employee = employee;
      });
  }

  goToEmployeeList() {
    this.router.navigate(['/employees']);
  }

  deleteEmployee() {
    this.modalService.open(this.deleteConfirmationModal).result.then((result)
=> {
      // Delete confirmed, perform delete operation
      if (result === 'delete') {
        const employeeId = this.employee.id;
        this.http.delete(`http://localhost:3000/employees/${employeeId}`)
          .subscribe(() => {
            this.goToEmployeeList();
          });
      }
    }, (dismissReason) => {
      // Dismissed without confirmation
    });
  }

  confirmDelete() {
    // Perform the delete operation
    const employeeId = this.employee.id;
    this.http.delete(`http://localhost:3000/employees/${employeeId}`)
      .subscribe(() => {
        this.goToEmployeeList();
      });
  }

  goToUpdateEmployee() {
    const employeeId = this.employee.id;
    this.router.navigate(['/employees', employeeId, 'edit']);
  }
}
```

employee-edit.component.css :

```
/* CSS for update employee form */
.container {
  margin-top: 20px;
}

h2 {
  margin-bottom: 20px;
```

```
}

.form-group {
  margin-bottom: 15px;
}

label {
  font-weight: bold;
}

.btn-primary {
  margin-right: 10px;
  background-color: #0dcaf0;
  color: black;
}

.btn-secondary {
  margin-right: 10px;
}
```

employee-edit.component.html :

```
<app-navbar></app-navbar>
<div class="container">
  <h2>Update Employee</h2>
  <form (ngSubmit)="updateEmployee()" #employeeForm="ngForm">
    <div class="form-group">
      <label for="firstName">First Name</label>
      <input type="text" class="form-control" id="firstName" name="firstName"
[(ngModel)]="employee.first_name" required>
    </div>
    <div class="form-group">
      <label for="lastName">Last Name</label>
      <input type="text" class="form-control" id="lastName" name="lastName"
[(ngModel)]="employee.last_name" required>
    </div>
    <div class="form-group">
      <label for="email">Email</label>
      <input type="email" class="form-control" id="email" name="email"
[(ngModel)]="employee.email" required>
    </div>
    <button type="submit" class="btn btn-primary">Update</button>
    <button type="button" class="btn btn-secondary"
(click)="goToEmployeeDetails()">Cancel</button>
  </form>
</div>
```

employee-edit.component.spec.ts :

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

import { EmployeeEditComponent } from './employee-edit.component';

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

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [EmployeeEditComponent]
    });
    fixture = TestBed.createComponent(EmployeeEditComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

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

employee-edit.component.ts :

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-employee-edit',
  templateUrl: './employee-edit.component.html',
  styleUrls: ['./employee-edit.component.css']
})
export class EmployeeEditComponent implements OnInit {
  employee: any = {}; // Initialize employee object

  constructor(
    private route: ActivatedRoute,
    private router: Router,
    private http: HttpClient
  ) { }

  ngOnInit() {
    this.route.params.subscribe(params => {
      const employeeId = +params['id'];
      this.getEmployeeDetails(employeeId);
    });
```

```
  }

  getEmployeeDetails(employeeId: number) {
    this.http.get<any>(`http://localhost:3000/employees/${employeeId}`)
      .subscribe(
        (employee) => {
          this.employee = employee;
        },
        (error) => {
          console.error('Error:', error);
        }
      );
  }

  updateEmployee() {
    this.http.put(`http://localhost:3000/employees/${this.employee.id}`,
this.employee)
      .subscribe(() => {
        this.goToEmployeeDetails();
      });
  }

  goToEmployeeDetails() {
    const employeeId = this.employee.id;
    this.router.navigate(['/employees', employeeId]);
  }

  goToEmployeeList() {
    this.router.navigate(['/employees']);
  }
}
```

employee-list.component.css :

employee-list.component.html :

```
<app-navbar></app-navbar>

<div class="container" style="margin-top: 10px;">
  <button class="btn btn-primary mb-3"
(click)="navigateToCreateEmployee()">Add New Employee</button>

  <div class="table-responsive">
    <table class="table table-bordered">
      <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)="viewDetails(employee.id)">View Details</button>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

employee-list.component.spec.ts :

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

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

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

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [EmployeeListComponent]
    });
    fixture = TestBed.createComponent(EmployeeListComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

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

employee-list.component.ts :

```
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
```

```
@Component({
  selector: 'app-employee-list',
  templateUrl: './employee-list.component.html',
  styleUrls: ['./employee-list.component.css']
})
export class EmployeeListComponent implements OnInit {
  employees: any[] = [];

  constructor(private http: HttpClient, private router: Router) {}

  ngOnInit() {
    this.getEmployees();
  }

  getEmployees() {
    this.http.get<any[]>('http://localhost:3000/employees')
      .subscribe(employees => {
        this.employees = employees;
      });
  }

  navigateToCreateEmployee() {
    this.router.navigate(['/employees/create']);
  }

  viewDetails(employeeId: number) {
    this.router.navigate(['/employees/', employeeId]);
  }
}
```

home.component.css :

```
.container {
  margin-top: 10px;
  background-color: antiquewhite;
}
```

home.component.html :

```
<app-navbar></app-navbar>
<div class="container">
  <div class="jumbotron">
    <h1 class="display-4" style="text-align: center;">Home Works !!</h1>
    <h2 class="display-5" style="text-align: center;"><b>EVENT MANAGEMENT
SYSTEM</b></h2>
  </div>
</div>
```

home.component.spec.ts :

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

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

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

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [HomeComponent]
    });
    fixture = TestBed.createComponent(HomeComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

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

home.component.ts :

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {

}
```

login.component.css :

```
.container {
  display: inline-flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.card {
  width: 400px;
  padding: 20px;
```

```
  border-radius: 10px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

.card-title {
  text-align: center;
  font-size: 24px;
  margin-bottom: 20px;
}

.form-group label {
  font-weight: bold;
}

.form-control {
  border-radius: 5px;
}

.btn-primary {
  background-color: #007bff;
  border-color: #007bff;
  transition: all 0.3s ease;
}

.btn-primary:hover {
  background-color: #0069d9;
  border-color: #0062cc;
}

.btn-link {
  color: #007bff;
}

.btn-link:hover {
  color: #0056b3;
  text-decoration: none;
}
```

login.component.html :

```
<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-6">
      <div class="card mt-5">
        <div class="card-body">
          <h5 class="card-title text-center">Login</h5>
          <form (ngSubmit)="login(loginForm)" #loginForm="ngForm" novalidate>
            <!-- Form fields -->
            <div class="form-group">
```

```
      <label for="username">Username</label>
      <input
        type="text"
        class="form-control"
        id="username"
        name="username"
        placeholder="Enter your username"
        [(ngModel)]="username"
        required
      />
      <div
        *ngIf="
          loginForm &&
          loginForm.controls['username']?.invalid &&
          (loginForm.controls['username']?.dirty ||
            loginForm.controls['username']?.touched)
        "
        class="text-danger"
      >
        Username is required.
      </div>
    </div>
    <div class="form-group">
      <label for="password">Password</label>
      <input
        type="password"
        class="form-control"
        id="password"
        name="password"
        placeholder="Enter your password"
        [(ngModel)]="password"
        required
      />
      <div
        *ngIf="
          loginForm &&
          loginForm.controls['password']?.invalid &&
          (loginForm.controls['password']?.dirty ||
            loginForm.controls['password']?.touched)
        "
        class="text-danger"
      >
        Password is required.
      </div>
    </div>
    <div class="form-check">
      <input
        type="checkbox"
```

```
                class="form-check-input"
                id="rememberMe"
                name="rememberMe"
                [(ngModel)]="rememberMe"
              />
              <label class="form-check-label" for="rememberMe"
                >Remember me</label
              >
            </div>
            <div class="text-center mt-3">
              <button type="submit" class="btn btn-primary">Login</button>
            </div>
            <div
              *ngIf="showError"
              class="text-danger mt-3"
              style="text-align: center"
            >
              Invalid username or password.
            </div>
          </form>

          <div class="text-center mt-3">
            <a [routerLink]="['/password-change']">Forgot password?</a>
          </div>
          <div class="text-center mt-3">
            <p>
              Don't have an account? <a [routerLink]="['/signup']">Sign up</a>
            </p>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

login.component.spec.ts :

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

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

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

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [LoginComponent]
    });
```

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

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

login.component.ts :

```
import { Component, ViewChild, ElementRef, AfterViewInit } from
'@angular/core';
import { Router } from '@angular/router';
import { NgForm } from '@angular/forms';
import { UserService } from '../user.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'],
})
export class LoginComponent {
  @ViewChild('loginForm', { static: false }) loginForm!: NgForm;
  username: string = '';
  password: string = '';
  rememberMe: boolean = false;
  showError: boolean = false;

  constructor(private router: Router, private userService: UserService) {}

  login(loginForm: NgForm): void {
    if (this.loginForm && this.loginForm.invalid) {
      return;
    }

    if (this.username === 'admin') {
      if (this.userService.isPasswordChangeRequired(this.username)) {
        this.router.navigate(['/password-change']);
      } else {
        this.router.navigate(['/home']);
      }
    } else {
      const isValidUser = this.userService.validateUser(this.username,
this.password);

      if (isValidUser) {
        this.router.navigate(['/home']);
```

```
      } else {
this.showError = true;
      }
    }
  }
}
```

navbar.component.css :

```
.navbar-nav {
  justify-content: flex-end;
}
.form-inline {
  display: flex;
  align-items: center;
}
.navbar-collapse form.form-inline {
  margin-left: auto;
}
```

navbar.component.html :

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container">
    <a class="navbar-brand" href="#">Portal</a>

    <button
      class="navbar-toggler"
      type="button"
      data-bs-toggle="collapse"
      data-bs-target="#navbarNav"
      aria-controls="navbarNav"
      aria-expanded="false"
      aria-label="Toggle navigation"
    >
      <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav ml-auto">
        <li class="nav-item">
          <a class="nav-link" routerLink="/home">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" routerLink="/employees">Employees</a>
        </li>
        <li class="nav-item">
```

```
          <a class="nav-link" routerLink="/login">Logout</a>
        </li>
      </ul>

      <form class="form-inline my-2 my-lg-0" (submit)="searchEmployees()">
        <div class="input-group">
          <input
            class="form-control"
            type="search"
            placeholder="Search"
            aria-label="Search"
            [(ngModel)]="searchText"
            name="searchText"
          />
          <button class="btn btn-outline-light" type="submit">Search</button>
        </div>
      </form>

    </div>
  </div>
</nav>

<div
  class="modal fade"
  #employeeDetailsModal
  tabindex="-1"
  role="dialog"
  aria-labelledby="employeeDetailsModalLabel"
  aria-hidden="true"
>
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="employeeDetailsModalLabel">
          Employee Details
        </h5>
        <button
          type="button"
          class="close"
          data-dismiss="modal"
          aria-label="Close"
          (click)="hideEmployeeDetailsModal()"
        >
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <p><strong>ID:</strong> {{ employeeId }}</p>
```

```
        <p><strong>Name:</strong> {{ employeeName }}</p>
        <p><strong>Email:</strong> {{ employeeEmail }}</p>
      </div>
      <div class="modal-footer">
        <button
          type="button"
          class="btn btn-secondary"
          data-dismiss="modal"
          (click)="hideEmployeeDetailsModal()"
        >
          Close
        </button>
      </div>
    </div>
  </div>
</div>
<div
  class="modal fade"
  #employeeNotFoundModal
  tabindex="-1"
  role="dialog"
  aria-labelledby="employeeNotFoundModalLabel"
  aria-hidden="true"
>
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="employeeNotFoundModalLabel">
          Employee Not Found
        </h5>
        <button
          type="button"
          class="close"
          data-dismiss="modal"
          aria-label="Close"
          (click)="hideEmployeeNotFoundModal()"
        >
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <p style="color: red;">Employee information doesn't exist.</p>
      </div>
      <div class="modal-footer">
        <button
          type="button"
          class="btn btn-secondary"
          data-dismiss="modal"
```

```
          (click)="hideEmployeeNotFoundModal()"
        >
          Close
        </button>
      </div>
    </div>
  </div>
</div>
```

navbar.component.spec.ts :

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

import { NavbarComponent } from './navbar.component';

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

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [NavbarComponent],
    });
    fixture = TestBed.createComponent(NavbarComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

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

navbar.component.ts :

```
import { Component, OnInit, ViewChild, ElementRef } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css'],
})
export class NavbarComponent implements OnInit {
  @ViewChild('employeeDetailsModal') employeeDetailsModal!: ElementRef;
  @ViewChild('employeeNotFoundModal') employeeNotFoundModal!: ElementRef;
  employeeId: string = '';
  employeeName: string = '';
  employeeEmail: string = '';
```

```
  searchText: string = '';

  constructor(private http: HttpClient) {}

  ngOnInit() {}

  searchEmployees() {
    this.http
      .get<any[]>(
        `http://localhost:3000/employees?q=${this.searchText}&_sort=last_name`
      )
      .subscribe(
        (response) => {
          console.log('Server Response:', response);
          if (response.length > 0) {
            const employee = response[0];
            this.employeeId = employee.id;
            this.employeeName = `${employee.first_name}
${employee.last_name}`;
            this.employeeEmail = employee.email;
            this.showEmployeeDetailsModal();
          } else {
            this.employeeId = '';
            this.employeeName = '';
            this.employeeEmail = '';
            this.showEmployeeNotFoundModal();
          }
        },
        (error) => {
          console.error('Error:', error);
        }
      );
  }

  showEmployeeDetailsModal() {
    const modalElement: HTMLElement = this.employeeDetailsModal.nativeElement;
    modalElement.classList.add('show');
    modalElement.style.display = 'block';
  }

  hideEmployeeDetailsModal() {
    const modalElement: HTMLElement = this.employeeDetailsModal.nativeElement;
    modalElement.classList.remove('show');
    modalElement.style.display = 'none';
  }

  // Add the show and hide methods for the employee not found modal
  showEmployeeNotFoundModal() {
```

```
    const modalElement: HTMLElement =
this.employeeNotFoundModal.nativeElement;
    modalElement.classList.add('show');
    modalElement.style.display = 'block';
  }

  hideEmployeeNotFoundModal() {
    const modalElement: HTMLElement =
this.employeeNotFoundModal.nativeElement;
    modalElement.classList.remove('show');
    modalElement.style.display = 'none';
  }
}
```

password-change.component.css :

```
.container {
  margin-top: 20px;
}

.card {
  border: 1px solid #ccc;
  border-radius: 5px;
}

.card-body {
  padding: 20px;
}

.card-title {
  font-size: 20px;
  font-weight: bold;
}

.form-group {
  margin-bottom: 20px;
}

label {
  font-weight: bold;
}

input[type="email"],
input[type="password"],
input[type="text"] {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 4px;
```

```
}

.btn-primary {
  width: 100%;
  padding: 10px;
  background-color: #007bff;
  color: #fff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.btn-primary:hover {
  background-color: #0069d9;
}

.text-success {
  color: #28a745;
}

.text-danger {
  color: #dc3545;
}

.text-center {
  text-align: center;
}

.mt-3 {
  margin-top: 15px;
}

.mb-3 {
  margin-bottom: 15px;
}
```

password-change.component.html :

```
<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-6">
      <div class="card mt-5">
        <div class="card-body">
          <h5 class="card-title text-center">Change Password</h5>
          <form
            (ngSubmit)="changePassword()"
            #passwordChangeForm="ngForm"
            novalidate
          >
```

```
<!-- Form fields -->
<!-- ... -->
<div class="form-group">
  <label for="email">Email</label>
  <input
    type="email"
    class="form-control"
    id="email"
    name="email"
    placeholder="Enter your email"
    [(ngModel)]="email"
    required
  />
  <div
    *ngIf="
      passwordChangeForm.controls['email']?.invalid &&
      (passwordChangeForm.controls['email']?.dirty ||
        passwordChangeForm.controls['email']?.touched)
    "
    class="text-danger"
    style="text-align: center"
  >
    Email is required.
  </div>
</div>
<!-- ... -->

<button
  type="button"
  class="btn btn-primary"
  (click)="validateEmail()"
>
  Validate Email
</button>
<div class="form-group">
  <label for="newPassword">New Password</label>
  <input
    type="password"
    class="form-control"
    id="newPassword"
    name="newPassword"
    placeholder="Enter your new password"
    [(ngModel)]="newPassword"
    required
  />
  <div
    *ngIf="
      passwordChangeForm.controls['newPassword']?.invalid &&
```

```
          (passwordChangeForm.controls['newPassword']?.dirty ||
            passwordChangeForm.controls['newPassword']?.touched)
        "
        class="text-danger"
        style="text-align: center"
      >
        New Password is required.
      </div>
    </div>
    <div class="form-group">
      <label for="confirmPassword">Confirm Password</label>
      <input
        type="password"
        class="form-control"
        id="confirmPassword"
        name="confirmPassword"
        placeholder="Confirm your new password"
        [(ngModel)]="confirmPassword"
        required
      />
      <div
        *ngIf="
        passwordChangeForm.controls['confirmPassword']?.invalid &&
        (passwordChangeForm.controls['confirmPassword']?.dirty ||
          passwordChangeForm.controls['confirmPassword']?.touched)

        "
        class="text-danger"
        style="text-align: center"
      >
        Confirm Password is required.
      </div>
      <div *ngIf="passwordsMismatch" class="text-danger">
        Passwords do not match.
      </div>
    </div>
    <div class="text-center mt-3">
      <button type="submit" class="btn btn-primary">
        Change Password
      </button>
    </div>
    <div
      *ngIf="successMessage"
      class="text-success mt-3"
      style="text-align: center"
    >
      {{ successMessage }}
    </div>
```

```
            <div
              *ngIf="errorMessage"
              class="text-danger mt-3"
              style="text-align: center"
            >
              {{ errorMessage }}
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>
```

password-change.component.spec.ts :

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

import { PasswordChangeComponent } from './password-change.component';

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

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [PasswordChangeComponent]
    });
    fixture = TestBed.createComponent(PasswordChangeComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

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

password-change.component.ts :

```
import { Component, ViewChild } from '@angular/core';
import { Router } from '@angular/router';
import { NgForm } from '@angular/forms';
import { UserService } from '../user.service';

@Component({
  selector: 'app-password-change',
  templateUrl: './password-change.component.html',
  styleUrls: ['./password-change.component.css'],
```

```
})
export class PasswordChangeComponent {
  @ViewChild('passwordChangeForm', { static: false })
  passwordChangeForm!: NgForm;
  email: string = '';
  newPassword: string = '';
  confirmPassword: string = '';
  passwordsMismatch: boolean = false;
  emailValidated: boolean = false;
  successMessage: string = '';
  errorMessage: string = '';

  constructor(private router: Router, private userService: UserService) {}

  validateEmail(): void {
    // Check if the email exists in the signup page data
    const isEmailValid = this.userService.isEmailExists(this.email);

    if (isEmailValid) {
      // Display success message
      this.successMessage = 'Email validated successfully.';
      this.emailValidated = true;
    } else {
      // Display error message
      this.errorMessage = 'Email does not exist.';
      this.emailValidated = false;
    }
  }

  changePassword(): void {
    // Check if the email is validated
    if (!this.emailValidated) {
      this.errorMessage = 'Please validate your email first.';
      return;
    }

    // Check if the new passwords match
    if (this.newPassword !== this.confirmPassword) {
      this.passwordChangeForm.controls['confirmPassword'].markAsTouched();
      this.passwordsMismatch = true;
      return;
    }

    // Change the password
    this.userService.changePassword(this.email, this.newPassword);

    // Display success message
    this.successMessage = 'Password changed successfully.';
```

```
    // Redirect to the login page
    this.router.navigate(['/login']);
  }
}
```

signup.component.css :

```
.container {
  display: inline-flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.card {
  width: 400px;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

.card-title {
  text-align: center;
  font-size: 24px;
  margin-bottom: 20px;
}

.form-group label {
  font-weight: bold;
}

.form-control {
  border-radius: 5px;
}

.error-message {
 color: red;
}

.btn-primary {
  background-color: #007bff;
  border-color: #007bff;
  transition: all 0.3s ease;
}

.btn-primary:hover {
  background-color: #0069d9;
  border-color: #0062cc;
```

```
}

.btn-link {
  color: #007bff;
}

.btn-link:hover {
  color: #0056b3;
  text-decoration: none;
}
```

signup.component.html :

```
<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-6">
      <div class="card mt-5">
        <div class="card-body">
          <h5 class="card-title text-center">Sign Up</h5>
          <form (ngSubmit)="signup()" [formGroup]="signupForm">
            <!-- Form fields -->
            <div class="form-group">
              <label for="username">Username</label>
              <input
                type="text"
                class="form-control"
                id="username"
                formControlName="username"
                placeholder="Enter your username"
              />
            </div>
            <div class="form-group">
              <label for="email">Email</label>
              <input
                type="email"
                class="form-control"
                id="email"
                formControlName="email"
                placeholder="Enter your email"
              />
            </div>
            <div class="form-group">
              <label for="password">Password</label>
              <input
                type="password"
                class="form-control"
                id="password"
                formControlName="password"
                placeholder="Enter your password"
```

```
              />
            </div>
            <div class="text-center mt-3">
              <button type="submit" class="btn btn-primary">Sign Up</button>
            </div>
          </form>
          <div class="text-center mt-3">
            <p class="error-message" *ngIf="errorMessage">{{ errorMessage
}}</p>
          </div>
          <div class="text-center mt-3">
            <p>
              Already have an account? <a [routerLink]="['/login']">Log in</a>
            </p>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

signup.component.spec.ts :

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

import { SignupComponent } from './signup.component';

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

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [SignupComponent]
    });
    fixture = TestBed.createComponent(SignupComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

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

signup.component.ts :

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

```
import { Router } from '@angular/router';
import { UserService } from '../user.service';

@Component({
  selector: 'app-signup',
  templateUrl: './signup.component.html',
  styleUrls: ['./signup.component.css'],
})
export class SignupComponent {
  signupForm: FormGroup;
  errorMessage: string = '';

  constructor(
    private router: Router,
    private formBuilder: FormBuilder,
    private userService: UserService
  ) {
    this.signupForm = this.formBuilder.group({
      username: ['', Validators.required],
      email: ['', [Validators.required, Validators.email]],
      password: ['', [Validators.required, Validators.minLength(8)]],
    });
  }

  signup(): void {
    // Check if the form is valid
    // if (this.signupForm.invalid) {
    //   // Set error message and return
    //   this.errorMessage = 'Error: Invalid form data.';
    //   return;
    // }

    // Check if any field is empty
    const { username, email, password } = this.signupForm.value;
    if (!username || !email || !password) {
      this.errorMessage = 'Error: All fields are required.';
      return;
    }

    // Check if the email format is valid
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!emailRegex.test(email)) {
      this.errorMessage = 'Error: Invalid email format.';
      return;
    }

    // Check if the password meets the minimum requirements (e.g., at least 8
characters)
```

```
    if (password.length < 8) {
      this.errorMessage = 'Error: Password must be at least 8 characters
long.';
      return;
    }

    // Create a new user object
    const newUser = {
      username,
      email,
      password,
    };

    // Call the userService method to check for duplicate username or email
    if (!this.userService.isUniqueUser(newUser)) {
      this.errorMessage = 'Error: Account already exists.';
      return;
    }

    // Call the userService method to store the user data
    this.userService.addUser(newUser);

    // Redirect to the login page
    this.router.navigate(['/login']);
  }
}
```

app-routing.module.ts :

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } 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 { EmployeeCreateComponent } from './employee-create/employee-
create.component';
import { EmployeeEditComponent } from './employee-edit/employee-
edit.component';
import { SignupComponent } from './signup/signup.component';
import { PasswordChangeComponent } from './password-change/password-
change.component';
import { HomeComponent } from './home/home.component';

const routes: Routes = [
  { path: 'login', component: LoginComponent },
  { path: 'signup', component: SignupComponent},
```

```
  { path: 'password-change', component: PasswordChangeComponent},
  { path: 'home', component: HomeComponent},
  { path: 'employees', component: EmployeeListComponent },
  { path: 'employees/create', component: EmployeeCreateComponent },
  { path: 'employees/:id', component: EmployeeDetailsComponent },
  { path: 'employees/:id/edit', component: EmployeeEditComponent },
  { path: '', redirectTo: '/login', pathMatch: 'full' },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

app.component.css :

app.component.html :

```
<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(() => TestBed.configureTestingModule({
    imports: [RouterTestingModule],
    declarations: [AppComponent]
  }));

  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';

@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 { ReactiveFormsModule } from '@angular/forms';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { LoginComponent } from './login/login.component';
import { SignupComponent } from './signup/signup.component';
import { EmployeeListComponent } from './employee-list/employee-
list.component';
import { EmployeeDetailsComponent } from './employee-details/employee-
details.component';
import { EmployeeCreateComponent } from './employee-create/employee-
create.component';
import { EmployeeEditComponent } from './employee-edit/employee-
edit.component';
import { PasswordChangeComponent } from './password-change/password-
change.component';
import { HomeComponent } from './home/home.component';
import { NavbarComponent } from './navbar/navbar.component';

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
```

```
    SignupComponent,
    EmployeeListComponent,
    EmployeeDetailsComponent,
    EmployeeCreateComponent,
    EmployeeEditComponent,
    PasswordChangeComponent,
    HomeComponent,
    NavbarComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ReactiveFormsModule,
    FormsModule,
    HttpClientModule,
    NgbModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

user.service.spec.ts :

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

import { UserService } from './user.service';

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

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

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});
```

user.service.ts :

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

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

```
  users: any[] = [];
  passwordChangeRequired: boolean = true; // Default value for password change
requirement

  constructor() {}

  isUniqueUser(newUser: any): boolean {
    // Check if the username or email already exists
    const existingUser = this.users.find(
      (user) =>
        user.username === newUser.username || user.email === newUser.email
    );

    return !existingUser; // Return true if the user is unique, false
otherwise
  }

  addUser(user: any): void {
    this.users.push(user); // Add the new user to the list
  }

  isPasswordChangeRequired(username: string): boolean {
    // Implement the logic to check if password change is required for the
given username
    // You can use the username parameter to identify the admin user
    // ...

    return this.passwordChangeRequired; // Return the password change
requirement status
  }

  validateUser(username: string, password: string): boolean {
    // Find the user with the matching username
    const user = this.users.find((user) => user.username === username);

    // Check if the user exists and the password matches
    if (user && user.password === password) {
      return true; // Credentials are valid
    }

    return false; // Credentials are invalid
  }

  changePassword(email: string, newPassword: string): void {
    // Find the user with the matching email
    const user = this.users.find((user) => user.email === email);

    // Update the user's password
```

```
    if (user) {
      user.password = newPassword;
    }

    // Reset the password change requirement
    this.passwordChangeRequired = false;
  }

  isEmailExists(email: string): boolean {
    // Check if the email exists in the signup page data
    const existingUser = this.users.find((user) => user.email === email);
    return !!existingUser; // Return true if the email exists, false otherwise
  }
}
```

index.html :

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>EventManagementApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

main.ts :

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';


platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

styles.css :

```
/* You can add global styles to this file, and also import other style files
*/
```

.editorconfig :

```
# Editor configuration, see https://editorconfig.org
root = true
```

```
[*]
charset = utf-8
indent_style = space
indent_size = 2
insert_final_newline = true
trim_trailing_whitespace = true

[*.ts]
quote_type = single

[*.md]
max_line_length = off
trim_trailing_whitespace = false
```

angular.json :

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "event-management-app": {
      "projectType": "application",
      "schematics": {},
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/event-management-app",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": ["zone.js"],
            "tsConfig": "tsconfig.app.json",
            "assets": ["src/favicon.ico", "src/assets"],
            "styles": [
              "node_modules/bootstrap/dist/css/bootstrap.min.css",
              "src/styles.css"
            ],
            "scripts": [
              "node_modules/jquery/dist/jquery.min.js",
              "node_modules/bootstrap/dist/js/bootstrap.min.js"
            ]
          },
          "configurations": {
            "production": {
```

```
            "budgets": [
              {
                "type": "initial",
                "maximumWarning": "500kb",
                "maximumError": "1mb"
              },
              {
                "type": "anyComponentStyle",
                "maximumWarning": "2kb",
                "maximumError": "4kb"
              }
            ],
            "outputHashing": "all"
          },
          "development": {
            "buildOptimizer": false,
            "optimization": false,
            "vendorChunk": true,
            "extractLicenses": false,
            "sourceMap": true,
            "namedChunks": true
          }
        },
        "defaultConfiguration": "production"
      },
      "serve": {
        "builder": "@angular-devkit/build-angular:dev-server",
        "configurations": {
          "production": {
            "browserTarget": "event-management-app:build:production"
          },
          "development": {
            "browserTarget": "event-management-app:build:development"
          }
        },
        "defaultConfiguration": "development"
      },
      "extract-i18n": {
        "builder": "@angular-devkit/build-angular:extract-i18n",
        "options": {
          "browserTarget": "event-management-app:build"
        }
      },
      "test": {
        "builder": "@angular-devkit/build-angular:karma",
        "options": {
          "polyfills": ["zone.js", "zone.js/testing"],
          "tsConfig": "tsconfig.spec.json",
```

```
            "assets": ["src/favicon.ico", "src/assets"],
            "styles": ["src/styles.css"],
            "scripts": []
          }
        }
      }
    }
  },
  "cli": {
    "analytics": false
  }
}
```

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",
    "json:server": "json-server --watch db.json"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^16.1.0",
    "@angular/common": "^16.1.0",
    "@angular/compiler": "^16.1.0",
    "@angular/core": "^16.1.0",
    "@angular/forms": "^16.1.0",
    "@angular/platform-browser": "^16.1.0",
    "@angular/platform-browser-dynamic": "^16.1.0",
    "@angular/router": "^16.1.0",
    "@ng-bootstrap/ng-bootstrap": "^15.1.0",
    "@types/bootstrap": "^5.2.6",
    "bootstrap": "^5.3.0",
    "jquery": "^3.7.0",
    "rxjs": "~7.8.0",
    "tslib": "^2.3.0",
    "zone.js": "~0.13.0"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "^16.1.3",
    "@angular/cli": "~16.1.3",
    "@angular/compiler-cli": "^16.1.0",
    "@types/jasmine": "~4.3.0",
```

```
    "@types/jest": "^29.5.3",
    "@types/jquery": "^3.5.16",
    "jasmine-core": "~4.6.0",
    "karma": "~6.4.0",
    "karma-chrome-launcher": "~3.2.0",
    "karma-coverage": "~2.2.0",
    "karma-jasmine": "~5.1.0",
    "karma-jasmine-html-reporter": "~2.1.0",
    "typescript": "~5.1.3"
  }
}
```

tsconfig.app.json :

```
/* To learn more about this file see: https://angular.io/config/tsconfig. */
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "outDir": "./out-tsc/app",
    "types": []
  },
  "files": [
    "src/main.ts"
  ],
  "include": [
    "src/**/*.d.ts"
  ]
}
```

tsconfig.json :

```
/* To learn more about this file see: https://angular.io/config/tsconfig. */
{
  "compileOnSave": false,
  "compilerOptions": {
    "types": ["jquery", "jest"],
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "noImplicitOverride": true,
    "noPropertyAccessFromIndexSignature": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true,
    "sourceMap": true,
    "declaration": false,
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "moduleResolution": "node",
```

```
    "importHelpers": true,
    "target": "ES2022",
    "module": "ES2022",
    "useDefineForClassFields": false,
    "lib": ["ES2022", "dom"]
  },
  "angularCompilerOptions": {
    "enableI18nLegacyMessageIdFormat": false,
    "strictInjectionParameters": true,
    "strictInputAccessModifiers": true,
    "strictTemplates": true
  }
}
```

tsconfig.spec.json :

```
/* To learn more about this file see: https://angular.io/config/tsconfig. */
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "outDir": "./out-tsc/spec",
    "types": [
      "jasmine"
    ]
  },
  "include": [
    "src/**/*.spec.ts",
    "src/**/*.d.ts"
  ]
}
```