**SRC:**

**Main.ts**

import { bootstrapApplication } from '@angular/platform-browser';

import { provideHttpClient, withFetch } from '@angular/common/http';

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

bootstrapApplication(AppComponent, {

  providers: [provideHttpClient(withFetch())],

}).catch((err) => console.error(err));

**Main.server.ts**

import { bootstrapApplication } from '@angular/platform-browser';

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

import { config } from './app/app.config.server';

const bootstrap = () => bootstrapApplication(AppComponent, config);

export default bootstrap;

**index.html**

<!doctype html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>FinalPreparation</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>

**App.routes.ts**

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

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

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

export const routes: Routes = [

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

  { path: 'login', component: LoginComponent },  // Route to login component

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

**app.config.ts**

import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';

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

import { routes } from './app.routes';

import { provideClientHydration } from '@angular/platform-browser';

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

export const appConfig: ApplicationConfig = {

  providers: [provideZoneChangeDetection({ eventCoalescing: true }), provideRouter(routes), provideClientHydration(), provideHttpClient()]};

**app.config.server.ts**

import { mergeApplicationConfig, ApplicationConfig } from '@angular/core';

import { provideServerRendering } from '@angular/platform-server';

import { appConfig } from './app.config';

const serverConfig: ApplicationConfig = {

  providers: [

    provideServerRendering()

  ]

};

export const config = mergeApplicationConfig(appConfig, serverConfig);

**app.component.ts**

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

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

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

@Component({

  selector: 'app-root',

  standalone: true,

  imports: [LoginComponent, SignupComponent],

  templateUrl: 'app.component.html',

  styleUrls: ['./app.component.css'],

})

export class AppComponent {

  title = 'final-preparation'; // Define the title property here

}

**App.component.spec.ts**

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

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

describe('AppComponent', () => {

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      imports: [AppComponent],

    }).compileComponents();

  });

  it('should create the app', () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app).toBeTruthy();

  });

  it(`should have the 'final-preparation' title`, () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app.title).toEqual('final-preparation');

  });

  it('should render title', () => {

    const fixture = TestBed.createComponent(AppComponent);

    fixture.detectChanges();

    const compiled = fixture.nativeElement as HTMLElement;

    expect(compiled.querySelector('h1')?.textContent).toContain('Hello, final-preparation');

  });

});

**HTML**

<h1>Welcome to Our App!</h1>

    <div>

      <app-login></app-login>

      <hr />

      <app-signup></app-signup>

    </div>

**Signup**

**HTML**

<h2>Signup</h2>

<form [formGroup]="signupForm" (ngSubmit)="onSubmit()">

  <div>

    <label for="name">Name:</label>

    <input id="name" formControlName="name" />

    <span \*ngIf="signupForm.controls['name'].invalid && signupForm.controls['name'].touched" style="color: red;">

      Name is required!

    </span>

  </div>

  <div>

    <label for="email">Email:</label>

    <input id="email" formControlName="email" />

    <span \*ngIf="signupForm.controls['email'].invalid && signupForm.controls['email'].touched" style="color: red;">

      Valid email is required!

      <p>{{ signupForm.controls['email'].value | uppercase }}</p>

    </span>

  </div>

  <div>

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

    <input type="password" id="password" formControlName="password" />

    <span

      \*ngIf="signupForm.controls['password'].invalid && signupForm.controls['password'].touched"

      style="color: red;"

    >

      Password must be at least 6 characters!

    </span>

  </div>

  <div>

    <label for="confirmPassword">Confirm Password:</label>

    <input type="password" id="confirmPassword" formControlName="confirmPassword" />

    <span

      \*ngIf="signupForm.controls['confirmPassword'].invalid && signupForm.controls['confirmPassword'].touched"

      style="color: red;"

    >

      Confirm password is required!

    </span>

  </div>

  <button type="submit" [disabled]="signupForm.invalid">Signup</button>

</form>

**Specs.ts**

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

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

describe('SignupComponent', () => {

  let component: SignupComponent;

  let fixture: ComponentFixture<SignupComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      imports: [SignupComponent]

    })

    .compileComponents();

    fixture = TestBed.createComponent(SignupComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

**Signup.ts**

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

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

import { ApiService } from '../api.service';

import { CommonModule } from '@angular/common';

@Component({

  selector: 'app-signup',

  standalone: true,

  imports: [ReactiveFormsModule, CommonModule, FormsModule], // Removed FormsModule entirely

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

  styleUrls: ['./signup.component.css'],

})

export class SignupComponent {

  signupForm = new FormGroup({

    name: new FormControl('', [Validators.required]),

    email: new FormControl('', [Validators.required, Validators.email]),

    password: new FormControl('', [Validators.required, Validators.minLength(6)]),

    confirmPassword: new FormControl('', [Validators.required]),

  });

  constructor(private apiService: ApiService) {}

  onSubmit() {

    if (this.signupForm.valid) {

      const formData = this.signupForm.value;

      if (formData.password !== formData.confirmPassword) {

        alert('Passwords do not match!');

        return;

      }

      this.apiService.signup(formData).subscribe(

        (response) => {

          console.log('Signup successful:', response);

          alert('Signup successful!');

        },

        (error) => {

          console.error('Signup failed:', error);

          alert('Signup failed! Please try again.');

        }

      );

    } else {

      alert('Please fill out the form correctly.');

    }

  }

}

**LOGIN**

**HTML**

<h2>Login</h2>

<form [formGroup]="loginForm" (ngSubmit)="onSubmit()">

  <div>

    <label for="email">Email:</label>

    <input id="email" formControlName="email" />

    <span \*ngIf="loginForm.controls['email'].invalid && loginForm.controls['email'].touched" style="color: red;">

      Valid email is required!

      <p>{{ loginForm.controls['email'].value | uppercase }}</p>

    </span>

  </div>

  <div>

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

    <input type="password" id="password" formControlName="password" />

    <span \*ngIf="loginForm.controls['password'].invalid && loginForm.controls['password'].touched" style="color: red;">

      Password must be at least 6 characters!

    </span>

  </div>

  <button type="submit" [disabled]="loginForm.invalid">Login</button>

</form>

**Specs.ts**

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

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

describe('LoginComponent', () => {

  let component: LoginComponent;

  let fixture: ComponentFixture<LoginComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      imports: [LoginComponent]

    })

    .compileComponents();

    fixture = TestBed.createComponent(LoginComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

**Login.ts**

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

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

import { ApiService } from '../api.service';

import { CommonModule } from '@angular/common';

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

@Component({

  selector: 'app-login',

  standalone: true,

  imports: [ReactiveFormsModule, CommonModule, FormsModule], // Importing necessary modules

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

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

})

export class LoginComponent {

  loginForm = new FormGroup({

    email: new FormControl('', [Validators.required, Validators.email]),

    password: new FormControl('', [Validators.required, Validators.minLength(6)]),

  });

  constructor(private apiService: ApiService) {}

  onSubmit() {

    if (this.loginForm.valid) {

      this.apiService.login(this.loginForm.value).subscribe(

        (response) => {

          console.log('Login successful:', response);

          alert('Login successful!');

        },

        (error) => {

          console.error('Login failed:', error);

          alert('Login failed! Please check your credentials.');

        }

      );

    } else {

      alert('Please fill out the form correctly.');

    }

  }

}

**API**

**Spec.ts**

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

import { ApiService } from './api.service';

describe('ApiService', () => {

  let service: ApiService;

  beforeEach(() => {

    TestBed.configureTestingModule({});

    service = TestBed.inject(ApiService);

  });

  it('should be created', () => {

    expect(service).toBeTruthy();

  });

});

**Api.ts**

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

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

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root', // Ensures this service is available application-wide

})

export class ApiService {

  private baseUrl = 'http://localhost:3600/api';

  constructor(private http: HttpClient) {}

  // Signup API call

  signup(data: any): Observable<any> {

    return this.http.post(`${this.baseUrl}/signup`, data);

  }

  // Login API call

  login(data: any): Observable<any> {

    return this.http.post(`${this.baseUrl}/login`, data);

  }

}

**SERVER**

**Index.js**

import express from 'express';

import cors from 'cors';

import bodyParser from 'body-parser';

const app = express();

app.use(cors()); // Enable Cross-Origin Resource Sharing

app.use(bodyParser.json()); // Parse incoming JSON requests

// In-memory storage for users (for testing purposes)

const users = [];

// Signup Route

app.post('/api/signup', (req, res) => {

  const { name, email, password } = req.body;

  // Validation

  if (!name || !email || !password) {

    return res.status(400).json({ message: 'All fields are required' });

  }

  // Check if user already exists

  const existingUser = users.find(user => user.email === email);

  if (existingUser) {

    return res.status(400).json({ message: 'User already exists' });

  }

  // Save new user

  users.push({ name, email, password });

  console.log(users,"u");

  res.status(201).json({ message: 'Signup successful' });

});

// Login Route

app.post('/api/login', (req, res) => {

  const { email, password } = req.body;

  // Validation

  if (!email || !password) {

    return res.status(400).json({ message: 'Email and password are required' });

  }

  // Find user

  const user = users.find(u => u.email === email && u.password === password);

  if (!user) {

    return res.status(401).json({ message: 'Invalid email or password' });

  }

  res.status(200).json({ message: 'Login successful', user: { name: user.name, email: user.email } });

});

// Default Route

app.get('/', (req, res) => {

  res.send('Welcome to the server!');

});

// Start the server

app.listen(3600, () => {

  console.log('Server is running on http://localhost:3600');

});

**Package.json**

{

  "name": "server",

  "version": "1.0.0",

  "description": "",

  "type": "module",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "node index.js",

    "dev": "nodemon index.js"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "dependencies": {

    "body-parser": "^1.20.3",

    "cors": "^2.8.5",

    "express": "^4.21.2",

    "nodemon": "^3.1.9",

    "server": "file:"

  }

}