5.9 Routing Mechanisms

This section will guide you to:

* Add a router to your Angular application
* Change the source code to implement the sign-up page

This guide has four subsections, namely:

5.9.1 Configuring the Angular application

5.9.2 Changing source code to implement the routers

5.9.3 Completing the functionality by implementing validators

5.9.4 Pushing the code to Github repositories

**Step 5.9.1:** Configuring the Angular application

* Create new angular project using command as ng new angular-routing
* Open the IDE and import the application by following the steps mentioned below:
* Click on **File**  **Open Folder…** and provide the path to where you have installed your Angular application.
* Click **View** 🡪 **Terminal** to open the terminal of the IDE.
* Use **ng serve** in the terminal to start the local server to run at port 4200.

**Step 5.9.2:** Changing source code to implement the routers

* Open the **app.component.html** available in the **app** folder of the Angular application. Replace all the code available in the file to the source code mentioned below:
* <div>
* <router-outlet></router-outlet>
* </div>

* Create new component as **ng generate component signin**
* Open Signin.component.ts file

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

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

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

@Component({

  selector: 'app-signin',

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

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

})

export *class* SigninComponent {

*constructor*(private *formBuilder*: FormBuilder,public *router*:Router) { }

   status = 'You haven\'t signed up yet';

  name = '';

  submitted = false;

  registerForm: FormGroup;

  ontyping(*event*:*Event*) {

    this.name = (<*HTMLInputElement*>*event*.target).value;

  }

  signup() {

     this.status = 'Oops! We are working on it!';

  }

  ngOnInit() {

    this.registerForm = this.formBuilder.group({

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

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

    });

}

*get* f() {

  if(this.registerForm?.controls!=null){

    return this.registerForm.controls;

  }

  return null;

}

msg:*string* ="";

onSubmit() {

    this.submitted = true;

    // stop here if form is invalid

    if (this.registerForm?.invalid) {

        return;

    }else {

*let* login = this.registerForm.value;

      if(login.email=="admin@gmail.com" && login.password=="admin@123"){

        alert("successfully login")

          this.router.navigate(["home"]);

      }else {

        alert("failure try once again")

        this.msg = "Invalid emailid or password";

      }

    }

}

}

* Open the **signin.component.hml** file and add the source code mentioned below:

<div class="jumbotron">

    <div class="container">

        <div class="row">

            <div class="col-md-6 offset-md-3">

                <h2>Angular Reactive Form Validation and Verification</h2>

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

                    <div class="form-group">

                        <label>Email</label>

                        <input type="text" formControlName="email" class="form-control" [ngClass]="{ 'is-invalid': submitted && f['email'].errors }" />

                        <div \*ngIf="submitted && f['email'].errors" class="invalid-feedback">

                            <div \*ngIf="f['email'].errors?.['required']">Email is required</div>

                            <div \*ngIf="f['email'].errors?.['email']">Email must be a valid email address</div>

                        </div>

                    </div>

                    <div class="form-group">

                        <label>Password</label>

                        <input type="password" formControlName="password" class="form-control" [ngClass]="{ 'is-invalid': submitted && f['password'].errors }" />

                        <div \*ngIf="submitted && f['password'].errors" class="invalid-feedback">

                            <div \*ngIf="f['password'].errors?.['required']">Password is required</div>

                            <div \*ngIf="f['password'].errors?.['minlength']">Password must be at least 6 characters</div>

                        </div>

                    </div>

                    <div class="form-group">

                        <button class="btn btn-primary">Register</button>

                    </div>

                </form>

            </div>

        </div>

    </div>

  </div>

* Create another component as **ng generate component home**

<div>

    <h2>Welcome to Home page</h2>

</div>

<router-outlet></router-outlet>

* Open the [**home.component.css**](http://home.component.css)file

h2 {

*background-color*: yellow;

*font-size*: 30pt;

*color*: wheat;

*text-align*: center;

}

* Open app-routing.module.ts file

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

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

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

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

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

*const* routes: Routes = [

{

  path:'home',

  component:HomeComponent

},

{

  path:'',

  component:SigninComponent

}

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export *class* AppRoutingModule { }

Check the code in app.module.ts file

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

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

import { ReactiveFormsModule} from '@angular/forms'

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

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

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

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

@NgModule({

  declarations: [

    AppComponent,

    SigninComponent,

         HomeComponent,

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,ReactiveFormsModule

  ],

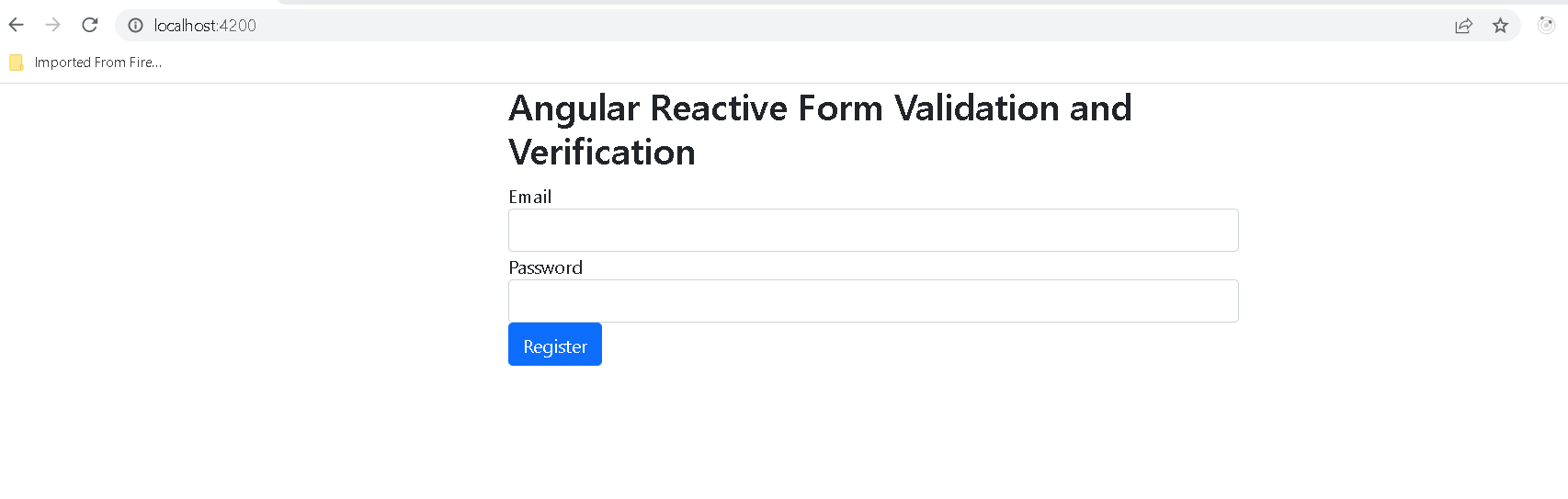
  providers: [],

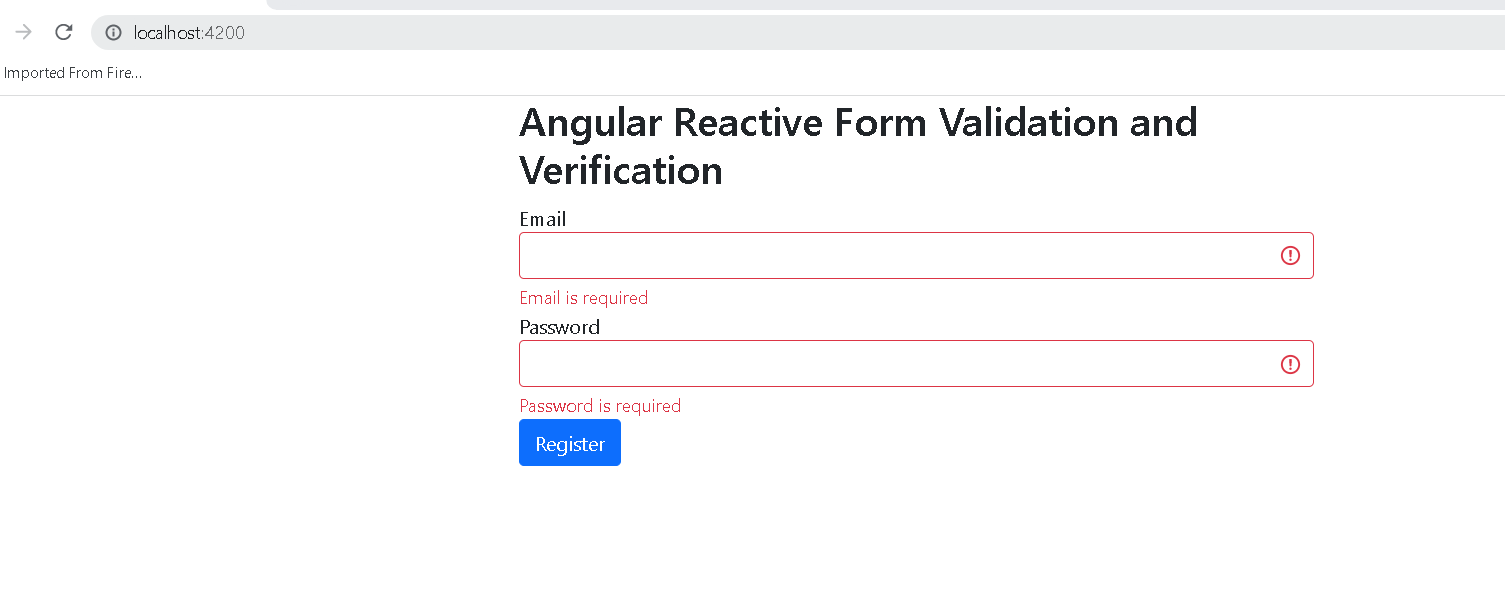
  bootstrap: [AppComponent]

})

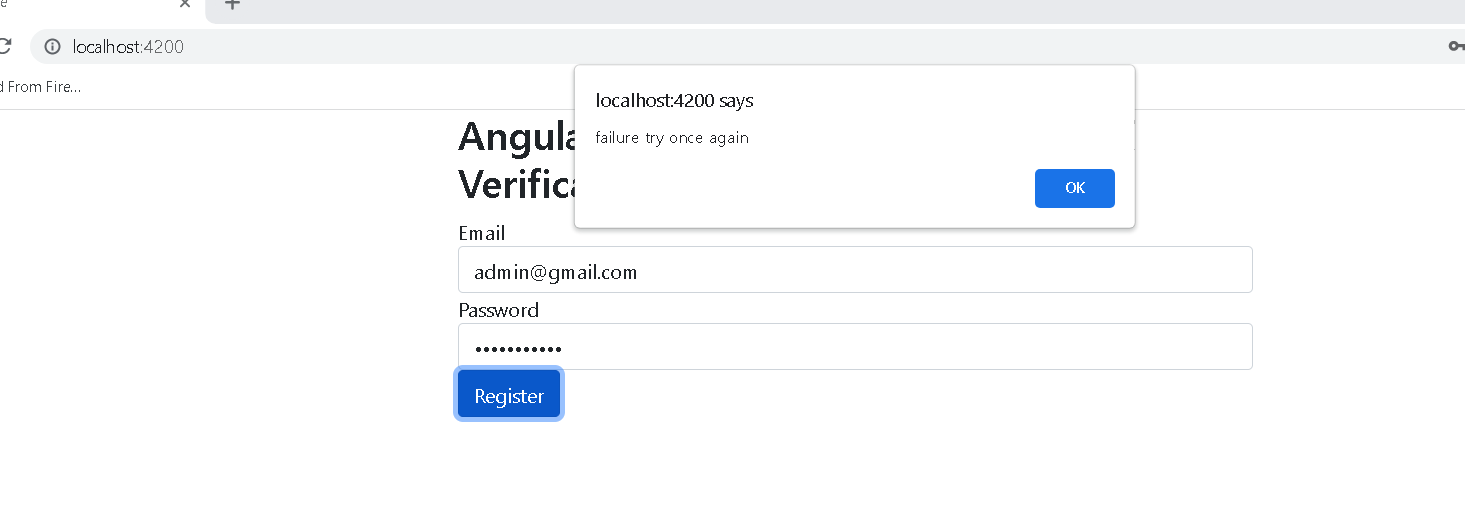
export *class* AppModule { }

**Now run the application**





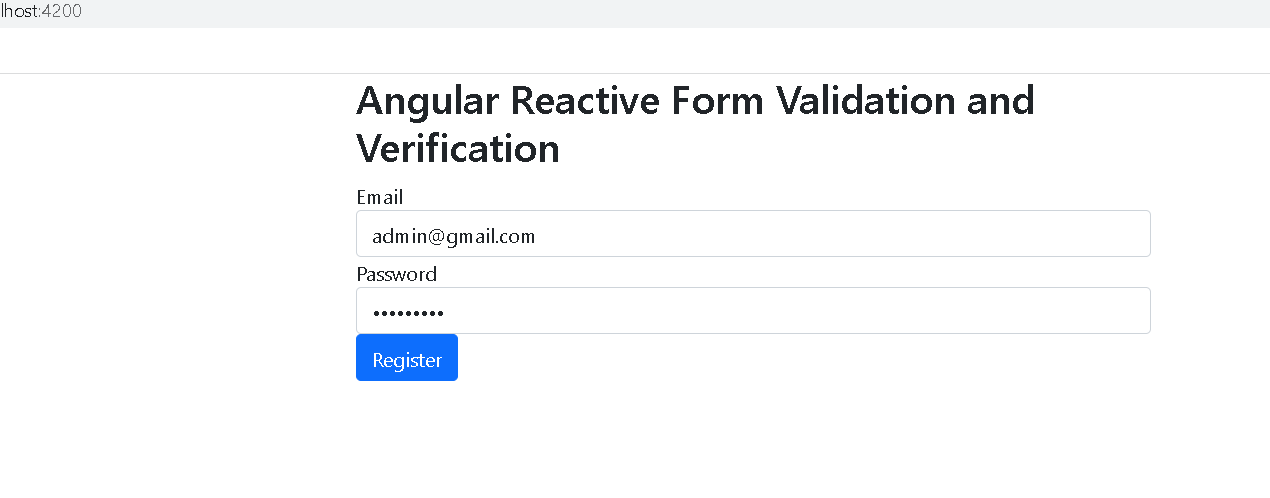
If we give wrong email id or password then

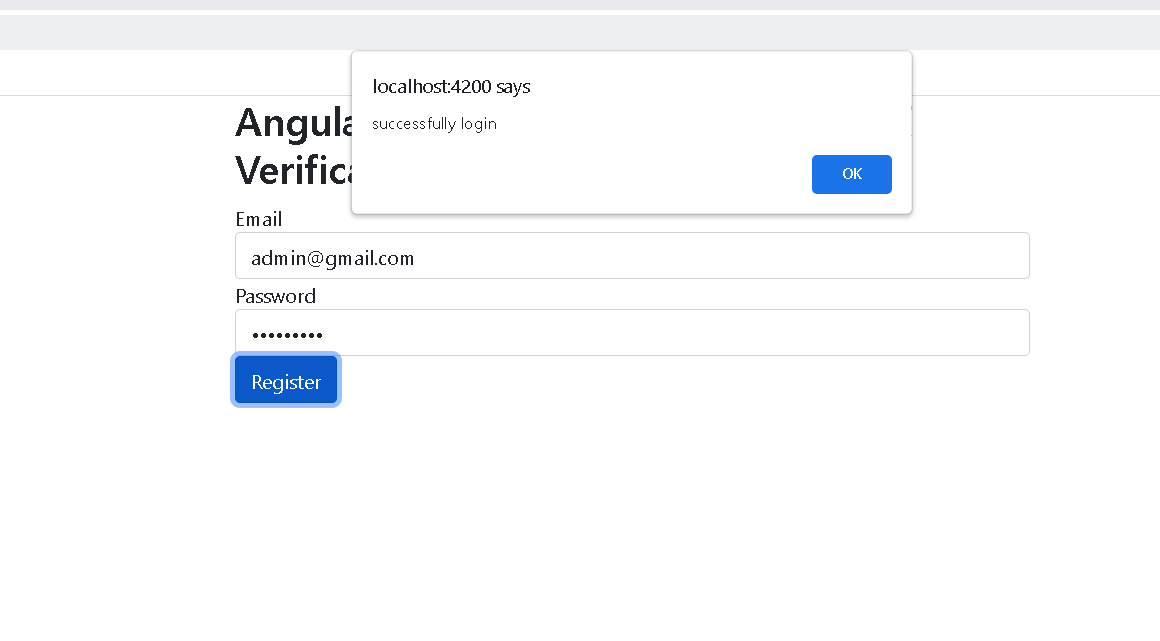


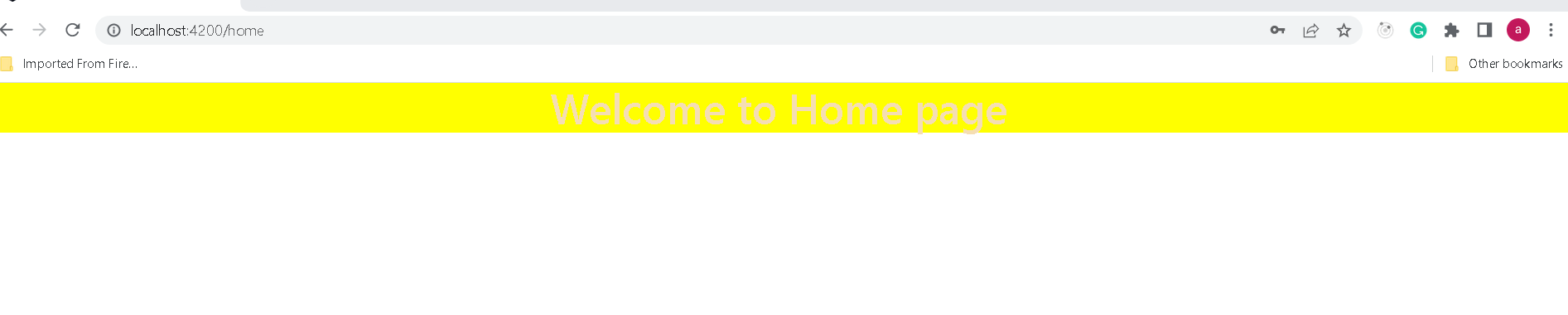
If we give correct email id and password as

emailId : [admin@gmail.com](mailto:admin@gmail.com)

password : **admin@123**







**Step 5.9.4:** Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you initially created using the following command:

git push -u origin master