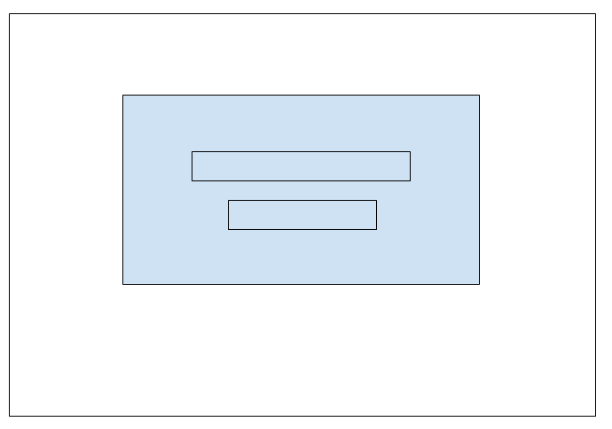
# Instructions – Exercise 7.2 – Reactive Forms

**Layout**

sign-in



gpa-calculator-app, part 3

**Instructions**

* Make a copy of the gpa-calcualtor-app2 from Assignment 6.4 and add it to your week-7 directory
* Rename the application to gpa-calculator-app3
* Delete the node\_modules directory
* Delete the package-lock.json file
* Open the angular.json file and find and replace all “gpa-calculator-app2” entries with “gpa-calculator-app3”
* Open the package.json file and change the name to “gpa-calculator-app3”
* Run npm install and ng serve
  + You are doing this to test the application and confirm there are no errors
* base-layout.component.ts
  + Update the title value to “Exercise 7.2 - Reactive Forms’;
* Install ngx-cookie-service
  + npm install ngx-cookie-service
  + Note: we are going to use this library to add “Cookies” to the users browser, once they sign-in
* Generate a new guard and name it sign-in
  + ng g g sign-in
  + Select \*CanActivate\*
* Generate a new service and name it sign-in
  + ng s sign-in
* Generate a new component and name it sign-in
  + ng g c sign-in
* app.module.ts
  + Add an import statement for the ReactiveFormsModule
    - import { FormsModule, ReactiveFormsModule } from ‘@angular/forms’;
* app.routing.ts
  + Import the LoginComponent
    - import { SignInComponent } from ‘./sign-in/sign-in.component.ts’;
  + Add a entry in the AuthLayoutComponents children array for the LoginComponent
    - path: sign-in,
    - component: SignInComponent
* sign-in.service.ts
  + Add a variable named studentIds of type Array<number>
  + In the services constructor populate the array with the following values: 1007, 1008, 1009, 1010, 1011, 1012
    - studentIds: Array<number> = [...]
  + Add a function named validate(studentId: number)
  + In the body of the validate function iterate over the studentIds array and return true if you find a match.  If there are no matches return false.
    - return this.studentIds.some(id => id === studentId);
* Sign-in.component.ts
  + Import the SignInService, Router, FormBuilder, FormGroup, and CookieService
    - import SignInService from ‘../sign-in.service.ts’;
    - import { Router } from ‘@angular/router’;
    - import { FormBuilder, FormGroup } from ‘@angular/forms’;
    - import { CookieService } from ‘ngx-cookie-service’;
  + In the components constructor add references to the Router, CookieService, FormBuilder, and SignInService
    - constructor(private router; Router, private cookieService: CookieService, private fb: FormBuilder, private signinService: SignInService)
  + Add a variable named form of type FormGroup
    - signinForm: FormGroup;
  + Add a variable named errorMessage of type string
    - errorMessage: string
  + In the components ngOnInit function use the FormBuilder to create a new FormGroup with one FormControl named studentId
    - this.signinForm = this.fb.group({ studentId: ‘’ })
  + Add a new function named onSubmit()
  + In the body of the onSubmit function, get the studentId value from the studentId FormControl.  Next, create an if statement that compares the returned value from the signinService.validate() function.  If true, add a cookie to the users browser and use the Router to navigate them to the root path.  Otherwise, assign an error message to the errorMessage variable
    - const studentId = parseInt(formValues.studentId);
    - if (this.signinService.validate(studentId)) {}
    - Note: you will need to assign the form value to a local variable to access the FormControl
      * const formValues = this.signinForm.value;
    - To set the cookie use:
      * this.cookieService.set(‘session\_user’, studentId.toString(), 1)
* sign-in.component.html
  + Build an Angular material form with one form-field and input of type text and one button of type submit.  The form should include a call to our FormGroup and an ngSubmit call to our submit function
    - <form [FormGroup]=”signinForm” (ngSubmit)=”onSubmit(); signinForm.reset()”></form>
    - Inside the <form> add a <div> element and an \*ngIf to check if the errorMessage variable has a value.  On the true condition output the results
      * <div class=”msg-error” \*ngIf=”errorMessage”> {{ errorMessage }}</div>
    - Set the mat-card-title to “Student Sign In”
    - Set the mat-card-subtitle to “You must enter a valid student ID to proceed.”
  + Styling requirements:
    - Use a mat-card, centered with 35% width and 15% top margin.
    - Center text in the mat-card-title and mat-card-subtitle
    - The error message should have a red font color (D8000C) and a background color of #FFD2D2; give it a padding of 12px and a margin of 10px 0px;
* base-layout.component.ts
  + Add an import statement for the CookieService and Router and add them it to the components constructor
    - import { CookieService } from ‘ngx-cookie-service’;
    - import { Router } from ‘@angular/router’;
    - constructor(private cookieService: CookieService, private router: Router)
  + Add a new function named signOut()
  + In the body of the signOut function call the cookieService.deleteAll function and then use the Router to navigate users to the sign-in page
    - this.cookieService.deleteAll()
    - this.router.navigate([‘/session/sign-in’]);
* base-layout.component.html
  + Add a new link to the top nav menu bar called Sign Out.  Also, add a click event that calls the signOut function
* Run and test the application
  + You should be checking the functionality of the sign-in form and Sign Out link
* sign-in.guard.ts
  + Add import statements for Router and CookieService and add them to the classes constructor (Note: you may need to create a constructor)
    - import { Router } from ‘@angular/router’;
    - import { CookieService } from ‘ngx-cookie-service’;
    - constructor(private router: Router, private cookieService: CookieService)
  + In the body of the canActivate function add a variable named sessionUser and assign it the response from the cookieService.get(‘session\_user’) call
  + Next, add an if statement to check the value, if true, return true.  If false, use the router to redirect users back to the sign-in page
    - If (sessionUser) { return true; } else { // redirect to the sign-in page and return false }
* App-routing.module.ts
  + add an import statement for the SignInGuard and add it to the parent BaseLayoutComponent
    - Import { SignInGuard } from ‘./sign-in.guard’;
    - children: [], canActivate: [ SignInGuard ]
* Run and test the application
  + You are checking to see if the guard is working and if you can sign in/out of the application