# Instructions – Exercise 7.3 Form Validation

gpa-calculator-app, final version

**Instructions**

* ~~Make a copy of the gpa-calcualtor-app3 from Assignment 7.2 and add it to your week-7 directory~~
* ~~Rename the application to gpa-calculator-app~~
* ~~Delete the node\_modules directory~~
* ~~Delete the package-lock.json file~~
* ~~Open the angular.json file and find and replace all “gpa-calculator-app3” entries with “gpa-calculator-app”~~
* ~~Open the package.json file and change the name to “gpa-calculator-app”~~
* ~~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 variable to “GPA Calculator”~~
* sign-in.component.ts
  + ~~Add an import statement for Validators (this can be added to the @angular/forms import you already have~~
    - ~~import { FormBuilder, FormGroup, Validators } from ‘@angular/forms~~
  + ~~Add Angulars built-in required Validator to the form.  Only allow numeric values and make the field required~~
    - ~~studentId: [‘’, Validators.compose([Validators.required, Validators.pattern(‘^[0-9]\*$)’])]~~
  + ~~Add a get() function  named form that returns the signinForm controls~~
    - ~~get form() { return this.signinForm.controls; }~~
    - ~~Note: we are creating this as a helper function to return the forms controls.  This way we can apply client-side validation in a predictable way~~
* sign-in.component.html
  + ~~Disable the submit button when the form is invalid~~
    - ~~<button type=”submit” [disabled]=”!signinForm.valid” …~~
  + ~~Add a new div with an \*ngIf that checks for errors and whether the control has been touched.  Inside the div add another div and apply our error styling.  Next, add the error message “Student ID must be a numeric value.~~
    - ~~<div \*ngIf=”form.studentId.errors && form.studentId.touched”><div> </div></div>~~
  + ~~Add an HTML 5 required directive to the input field~~
    - ~~<input type=”text” required~~
* ~~Run and test the application~~
  + ~~You are verifying whether validation has been applied to the sign in page~~
* home.component.ts
  + ~~Add an import statement for FormGroup, FormBuilder, and Validators~~
    - ~~import { FormGroup, FormBuilder, Validators } from ‘@angular/forms’;~~
  + ~~Add the FormBuilder to the components constructor~~
    - ~~constructor(private fb: FormBuilder)~~
  + ~~Add a new variable named transcriptForm of type FormGroup~~
  + ~~In the body of the ngOnInit function use the FormBuilder to build a new FormGroup with two FormControls: course and grade~~
    - ~~this.transcriptForm = this.fb.group({ course: [‘’, Validators.required ], grade: [‘’, Validators.required]});~~
  + Note: we are replacing the template driven form we originally used with a more flexible reactive form
  + ~~Add a get() function named form that returns the transcriptForm controls~~
    - ~~get form() {return this.transcriptForm.controls; }~~
  + ~~Rename the saveEntry() function to onSubmit() and add a parameter called event~~
    - ~~saveEntry() => onSubmit(event)~~
  + ~~Remove transactionEntry variable~~
  + ~~Remove the transactionEntry assignment from the components constructor~~
  + ~~Remove the transactionEntry from the onSubmit() function~~
  + ~~In the body of the onSubmit(event) function push a new object literal to the transcriptEntires array using the values captured from the from~~
    - ~~this.transcriptEntries.push({ course: this.form.course.value, grade: this.form.grade.value });~~
    - ~~Next, use the event object to reset Validation~~
      * ~~event.currentTarget.reset();~~
* home.component.html
  + ~~Remove the click event from the form, change the buttons type to submit, and disable it when the form is invalid~~
    - ~~<button type=”submit” [disabled]=”!transcriptForm.valid”~~
  + ~~Add the FormGroup to the HTML form and the onSubmit() function to the ngSubmit directive~~
    - ~~<form [fromGroup]=”transcriptForm” (ngSubmit)=”onSubmit($event); transcriptForm.reset()”>~~
  + ~~Replace the ngModel “banana in a boat” calls to formControlName entries~~
    - ~~formControlName=”course”~~
* Run and test the application
  + You are verifying the home component’s form now uses data validation and reactive forms.  You are also confirming there are no errors in the application and you did not introduce new bugs.
* Deploy the application to GitHub pages (use the deployment guide I provided under the weekly resources section)
* Update your personal portfolio with a link to the deployed website on GitHub pages
  + Clicking the link in your personal portfolio should take users to the gpa-calculator-app running as a static website on GitHub pages