**ANGULAR QUIZ INSTRUCTIONS**

1. Create a new application with routing, that uses SCSS.
2. Create the questions.json file. Where would you put it?
3. Let’s create the service called question, inside a service folder. first.
   1. It will use HttpClient and get your questions.json file in a method called getQuestionJson()
4. Create four components: header, question, service and welcome.
5. We will use Bootstrap. (So, install it and configure it or use the CDN). It includes font-awesome.
6. Add the code to the header component template, to implement Bootstrap classes and look nice:

<nav class="navbar navbar-dark bg-primary">

<div class="container-fluid">

**INSIDE**

</div>

</nav>

1. Where it says **INSIDE**
   1. Add what will be a link back to the top of the page, which is the beginning of the quiz:

<a class="navbar-brand text-bold" href="#">

<span class="text-uppercase">Angular Quiz</span>

</a>

1. Work in the question component next. In the TypeScript file:
   1. Import interval (from RxJS) and your QuestionService
   2. Create the next ten variables, unless specified they are initialized to empty string or zero. They are all public and will need to be referred after initialization with this, as in this.name. Do you need to specify public?
   3. name is a string
   4. questionList is an array that accepts any data type
   5. currentQuestion is a number
   6. points is a number
   7. counter is 30
   8. correctAnswer is a number
   9. inCorrectAnswer is a number
   10. interval is an observable of type any – note that the dollar sign at the end is a convention. We will put a dollar sign on the end of interval throughout this code.
       1. interval$: any;
   11. progress is a string initialized at the string 0 (zero).
   12. isQuizCompleted is a boolean initialized at false
2. In the constructor of this file, use the QuestionService
3. In ngOnInit()return void. Inside:
   1. Set name (remember to refer to it as this.name) to get the localStorage item called name.
   2. Run getAllQuestions() and also refer to it with this.getAllQuestions()
   3. Run startCounter()
4. Next create getAllQuestions()
   1. Use the question service to run getQuestionJson and subscribe Hint:

.subscribe(res => {

this.questionList = res.questions;

})

1. Add a nextQuestion() method which is this.currentQuestion variable incremented by one, and a previousQuestion method which is the current question decremented by one.
2. Create an answer method that takes currentQno which is a number and option which is any
3. Use currentQno to see if the end of the question list has been reached. If so, set this.isQuizCompleted to true and stop the counter with this.stopCounter();
4. When correct, add five points, increment the correct answer count, and reset the counter with the following:

if (option.correct) {

this.points += 5;

this.correctAnswer++;

setTimeout(() => {

this.currentQuestion++;

this.resetCounter();

this.getProgressPercent();

}, 1000);

1. Otherwise increment incorrect answers and add zero points:

} else {

setTimeout(() => {

this.currentQuestion++;

this.inCorrectAnswer++;

this.resetCounter();

this.getProgressPercent();

}, 1000);

this.points -=0;

}

1. Write the startCounter() function. It will set the interval to 1000, use subscribe, decrement counter, and have a test: if counter is zero, currentQuestion is incremented, counter is set to 30, and zero points are given, as above – points are decremented by zero.
2. Create a stopCounter() function that unsubscribes from interval and sets counter to zero.
3. Create resetCounter() that stops the counter, sets counter to 30, and starts the counter again (both with methods you create)
4. Create a resetQuiz() method that calls resetCounter(), gets all questions with getAllQuestions(), sets points to 0, counter to 30, the currentQuestion to 0, and progress to the string "0"
5. Create a getProgressPercent() method that assigns progress to the current question divided by number of questions, multiplies it by 100 and returns progress.
6. Go to the question component template, and add the following, filling in bolded all caps areas in this with variables from the TypeScript file

<div class="container mt-5">

<div class="card">

<div class="d-flex justify-content-between p-3">

<div class="quiz-header">

Welcome {{name}}

</div>

</div>

<ng-container **IF TEST IS NOT COMPLETED**>

<div class="d-flex justify-content-around py-3">

<div class="score">

<h5>**POINTS** Points</h5>

</div>

<div class="question-remain">

Question **CURRENT QUESTION PLUS ONE** of **NUMBER OF QUESTIONS**

</div>

<div class="timer">

<h5>**HOW MANY REMAINING (USE COUNTER)** seconds</h5>

</div>

</div>

<div class="progress mb-3">

<div class="progress-bar progress-bar-striped bg-success" role="progressbar" **NGSTYLE DIRECTIVE**="{'width':progress+'%'} aria-valuenow="25" aria-valuemin="0" aria-valuemax="100"></div>

</div>

<div class="question">

<div class="card">

<h3>{{questionList[currentQuestion]?.questionText}}</h3>

</div>

</div>

<div class="options">

<ol \*ngFor="let option of questionList[currentQuestion]?.options">

<li (click)="answer(currentQuestion+1,option)">

<div appChangeBg **isCorrect DIRECTIVE**="option.correct" class="card">

{{option.text}}

</div>

</li>

</ol>

</div>

<div class="d-flex justify-content-between">

<button **DISABLED DIRECTIVE IF CURRENT QUESTION IS 0** class="btn" **PREVIOUS QUESTION ON EVENT**><i class="fa text-primary fa-chevron-left fa-3x" aria-hidden="true"></i></button>

<button class="btn" **RESET QUIZ ON EVENT**><i class="fa fa-refresh text-primary fa-3x" aria-hidden="true"></i></button>

<button class="btn" **NEXT QUESTION ON EVENT**><i class="fa text-primary fa-chevron-right fa-3x" aria-hidden="true"></i></button>

</div>

</ng-container>

<ng-container **STRUCTURAL DIRECTIVE IF QUIZ IS COMPLETED**>

<div class="row d-flex justify-content-between">

<img style="width: 50%;" class="img-fluid col-sm-12 mx-auto" src="https://icon-library.com/images/celebration-icon-png/celebration-icon-png-7.jpg" alt="">

<div class="result text-center col-md-6 col-sm-12">

<h3>Congratulations!! <br>You have completed the quiz. <br>Below is your result:</h3>

<p>Total Question Attempted : **NUMBER OF QUESTIONS** </p>

<p>Total Correct Answered : **ANSWERED CORRECTLY VARIABLE**</p>

<p>Total Wrong Answered : **ANSWERED INCORRECTLY VARIABLE**</p>

<p>Your Score : **POINTS VARIABLE** Points </p>

</div>

</div>

</ng-container>

</div>

</div>

1. Add code to the welcome component template

<div class="container bg-light py-5">

<p class="col-md-8 fs-4">This quiz has 20 questions. A correct answer is 5 points, and an incorrect answer is 0 points.

You have 30 seconds to answer each question. Refereshing the page will reset the quiz.

<div class="name col-md-4 my-3">

**FORM ELEMENT**

</div>

<button class="btn btn-primary btn-lg" **LINK TO QUESTION START THE QUIZ ON AN EVENT**>Start the Quiz!</button>

</div>

1. Where it says **FORM ELEMENT**
   1. Add a form element with class form-control, and id name. Also create a label element for the field.

<label for="name">Enter your name:</label>

<input #name type="text" class="form-control" id="name">

1. Add an Angular router link to path /question where it says **LINK TO QUESTION**
2. Call the function to start the quiz (startQuiz() that we will create later) on an event, and also make a part clickable, where it says **START THE QUIZ ON AN EVENT**
3. In the welcome component ts, we will set localStorage:

import { Component, OnInit,ViewChild,ElementRef } from '@angular/core';

@Component({

selector: 'app-welcome',

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

styleUrls: ['./welcome.component.scss']

})

export class WelcomeComponent implements OnInit {

@ViewChild('name') nameKey!: ElementRef;

constructor() { }

ngOnInit(): void {

}

startQuiz(){

localStorage.setItem("name",this.nameKey.nativeElement.value);

}

}

1. App component will have the header components and router outlet.
2. Use the CLI and make a custom changeBg directive with the following:

import { Directive, ElementRef, HostListener, Input, Renderer2 } from '@angular/core';

@Directive({

selector: '[appChangeBg]'

})

export class ChangeBgDirective {

@Input() isCorrect : Boolean = false;

constructor(private el : ElementRef, private render : Renderer2) { }

@HostListener('click') answer(){

if(this.isCorrect){

this.render.setStyle(this.el.nativeElement,'background','green');

this.render.setStyle(this.el.nativeElement,'color','#fff');

this.render.setStyle(this.el.nativeElement,'border','2px solid grey');

}else{

this.render.setStyle(this.el.nativeElement,'background','red');

this.render.setStyle(this.el.nativeElement,'color','#fff');

this.render.setStyle(this.el.nativeElement,'border','2px solid grey');

}

}

}

1. In the routing module, map paths for default, question and welcome pages. If you didn’t create the project with routing, just create a file now.