|  |
| --- |
| <div id="quiz" class="container mt-5"> |
|  | <h2 class="text-center fw-bold text-uppercase">{{ quiz.name }}</h2> |
|  | <hr /> |
|  |  |
|  | <div \*ngIf="mode == 'quiz' && quiz"> |
|  | <div \*ngFor="let question of filteredQuestions"> |
|  | <div class="mb-4"> |
|  | <span class="badge bg-info"> |
|  | Question {{ pager.index + 1 }} of {{ pager.count }}. |
|  | </span> |
|  |  |
|  | <span \*ngIf="config.duration" class="badge bg-info float-end"> |
|  | Time: {{ ellapsedTime }} / {{ duration }} |
|  | </span> |
|  | </div> |
|  |  |
|  | <h3 class="fw-normal mb-4"> |
|  | {{ pager.index + 1 }}. |
|  | <span [innerHTML]="question.name"></span> |
|  | </h3> |
|  | <div class="row text-left options"> |
|  | <div class="col-6" \*ngFor="let option of question.options"> |
|  | <div class="option"> |
|  | <label class="fw-normal" [attr.for]="option.id"> |
|  | <input |
|  | id="{{ option.id }}" |
|  | type="checkbox" |
|  | [(ngModel)]="option.selected" |
|  | (change)="onSelect(question, option)" |
|  | /> |
|  | {{ option.name }} |
|  | </label> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | <hr /> |
|  | <br /> |
|  | <div class="text-center"> |
|  | <button |
|  | class="btn btn-outline-primary mx-2" |
|  | \*ngIf="config.allowBack" |
|  | (click)="goTo(0)" |
|  | > |
|  | First |
|  | </button> |
|  | <button |
|  | class="btn btn-outline-primary mx-2" |
|  | \*ngIf="config.allowBack" |
|  | (click)="goTo(pager.index - 1)" |
|  | > |
|  | Prev |
|  | </button> |
|  | <button |
|  | class="btn btn-outline-primary mx-2" |
|  | (click)="goTo(pager.index + 1)" |
|  | > |
|  | Next |
|  | </button> |
|  | <button |
|  | class="btn btn-outline-primary mx-2" |
|  | \*ngIf="config.allowBack" |
|  | (click)="goTo(pager.count - 1)" |
|  | > |
|  | Last |
|  | </button> |
|  | </div> |
|  | <br /> |
|  | </div> |
|  |  |
|  | <div class="row text-center" \*ngIf="mode == 'review'"> |
|  | <div |
|  | class="col-4 cursor-pointer" |
|  | \*ngFor="let question of quiz.questions; let index = index" |
|  | > |
|  | <div |
|  | (click)="goTo(index)" |
|  | class="p-3 mb-2 {{ |
|  | isAnswered(question) == 'Answered' ? 'bg-info' : 'bg-warning' |
|  | }}" |
|  | > |
|  | {{ index + 1 }}. {{ isAnswered(question) }} |
|  | </div> |
|  | </div> |
|  | </div> |
|  | <div class="result" \*ngIf="mode == 'result'"> |
|  | <h2> |
|  | Quiz Result: |
|  | <span class="badge bg-success" |
|  | >Your Score is: {{ quizScore }} Out of {{ quizTotalScore }}</span |
|  | > |
|  | </h2> |
|  | <div \*ngFor="let question of quiz.questions; let index = index"> |
|  | <div class="result-question"> |
|  | <h5>{{ index + 1 }}. {{ question.name }}</h5> |
|  | <div class="row"> |
|  | <div class="col-6" \*ngFor="let Option of question.options"> |
|  | <input |
|  | id="{{ Option.id }}" |
|  | type="checkbox" |
|  | disabled="disabled" |
|  | [(ngModel)]="Option.selected" |
|  | /> |
|  | {{ Option.name }} |
|  | </div> |
|  | </div> |
|  | <div |
|  | class="p-1 m-2 alert {{ |
|  | isCorrect(question) == 'correct' ? 'alert-success' : 'alert-danger' |
|  | }}" |
|  | > |
|  | Your answer is {{ isCorrect(question) }}. |
|  | </div> |
|  | </div> |
|  | </div> |
|  | <h4 class="alert alert-info text-center">You may close this window now.</h4> |
|  | </div> |
|  | <hr /> |
|  | <div \*ngIf="mode != 'result'"> |
|  | <button class="btn btn-warning mx-2" (click)="mode = 'quiz'">Quiz</button> |
|  | <button class="btn btn-info mx-2" (click)="mode = 'review'">Review</button> |
|  | <button class="btn btn-primary mx-2" (click)="onSubmit()"> |
|  | Submit Quiz |
|  | </button> |
|  | </div> |
|  | </div> |

|  |
| --- |
| import { ComponentFixture, TestBed } from '@angular/core/testing'; |
|  |  |
|  | import { QuizComponent } from './quiz.component'; |
|  |  |
|  | describe('QuizComponent', () => { |
|  | let component: QuizComponent; |
|  | let fixture: ComponentFixture<QuizComponent>; |
|  |  |
|  | beforeEach(async () => { |
|  | await TestBed.configureTestingModule({ |
|  | declarations: [ QuizComponent ] |
|  | }) |
|  | .compileComponents(); |
|  | }); |
|  |  |
|  | beforeEach(() => { |
|  | fixture = TestBed.createComponent(QuizComponent); |
|  | component = fixture.componentInstance; |
|  | fixture.detectChanges(); |
|  | }); |
|  |  |
|  | it('should create', () => { |
|  | expect(component).toBeTruthy(); |
|  | }); |
|  | }); |

|  |
| --- |
| import { Component, OnInit } from '@angular/core'; |
|  | import { ActivatedRoute } from '@angular/router'; |
|  | import { Option } from 'src/app/models/option'; |
|  | import { Question } from 'src/app/models/question'; |
|  | import { Quiz } from 'src/app/models/quiz'; |
|  | import { QuizConfig } from 'src/app/models/quiz-config'; |
|  | import { QuizService } from 'src/app/services/quiz.service'; |
|  |  |
|  | @Component({ |
|  | selector: 'app-quiz', |
|  | templateUrl: './quiz.component.html', |
|  | styleUrls: ['./quiz.component.css'], |
|  | }) |
|  | export class QuizComponent implements OnInit { |
|  | quizes: any[]; |
|  | quiz: Quiz = new Quiz(null); |
|  | mode = 'quiz'; |
|  | quizName: string; |
|  | quizScore: number; |
|  | quizTotalScore: number; |
|  |  |
|  | config: QuizConfig = { |
|  | allowBack: true, |
|  | allowReview: true, |
|  | autoMove: false, // if true, it will move to next question automatically when answered. |
|  | duration: 300, // indicates the time (in secs) in which quiz needs to be completed. 0 means unlimited. |
|  | pageSize: 1, |
|  | requiredAll: false, // indicates if you must answer all the questions before submitting. |
|  | richText: false, |
|  | shuffleQuestions: false, |
|  | shuffleOptions: false, |
|  | showClock: false, |
|  | showPager: true, |
|  | theme: 'none', |
|  | }; |
|  |  |
|  | pager = { |
|  | index: 0, |
|  | size: 1, |
|  | count: 1, |
|  | }; |
|  | timer: any = null; |
|  | startTime: Date; |
|  | endTime: Date; |
|  | ellapsedTime = '00:00'; |
|  | duration = ''; |
|  |  |
|  | constructor( |
|  | private quizService: QuizService, |
|  | private route: ActivatedRoute |
|  | ) {} |
|  |  |
|  | ngOnInit() { |
|  | this.quizes = this.quizService.getAll(); |
|  | this.quizScore = 0; |
|  | // this.quizName = this.quizes[0].id; |
|  | // this.loadQuiz(this.quizName); |
|  | this.route.paramMap.subscribe(() => { |
|  | this.handleQuiz(); |
|  | }); |
|  | } |
|  |  |
|  | handleQuiz() { |
|  | const quizName: string = this.route.snapshot.paramMap.get('quizName'); |
|  | const quiz: any = this.quizes.find((o) => o.name === quizName); |
|  | const quizUrl: string = quiz.id; |
|  |  |
|  | this.loadQuiz(quizUrl); |
|  | } |
|  |  |
|  | loadQuiz(quizName: string) { |
|  | this.quizService.get(quizName).subscribe((res) => { |
|  | //console.log(`Quiz Name: ${quizName} Response: ${JSON.stringify(res)} ` ) |
|  | this.quiz = new Quiz(res); |
|  | this.pager.count = this.quiz.questions.length; |
|  | this.quizTotalScore = this.quiz.questions.length; |
|  | this.startTime = new Date(); |
|  | this.ellapsedTime = '00:00'; |
|  | this.timer = setInterval(() => { |
|  | this.tick(); |
|  | }, 1000); |
|  | this.duration = this.parseTime(this.config.duration); |
|  | }); |
|  | this.mode = 'quiz'; |
|  | } |
|  |  |
|  | tick() { |
|  | const now = new Date(); |
|  | const diff = (now.getTime() - this.startTime.getTime()) / 1000; |
|  | if (diff >= this.config.duration) { |
|  | this.onSubmit(); |
|  | } |
|  | this.ellapsedTime = this.parseTime(diff); |
|  | } |
|  |  |
|  | parseTime(totalSeconds: number) { |
|  | let mins: string | number = Math.floor(totalSeconds / 60); |
|  | let secs: string | number = Math.round(totalSeconds % 60); |
|  | mins = (mins < 10 ? '0' : '') + mins; |
|  | secs = (secs < 10 ? '0' : '') + secs; |
|  | return `${mins}:${secs}`; |
|  | } |
|  |  |
|  | get filteredQuestions() { |
|  | return this.quiz.questions |
|  | ? this.quiz.questions.slice( |
|  | this.pager.index, |
|  | this.pager.index + this.pager.size |
|  | ) |
|  | : []; |
|  | } |
|  |  |
|  | onSelect(question: Question, option: Option) { |
|  | if (question.questionTypeId === 1) { |
|  | question.options.forEach((x) => { |
|  | if (x.id !== option.id) x.selected = false; |
|  | }); |
|  |  |
|  | if (this.isCorrect(question) == 'correct') { |
|  | this.quizScore = this.quizScore + 1; |
|  | } |
|  | } |
|  |  |
|  | if (this.config.autoMove) { |
|  | this.goTo(this.pager.index + 1); |
|  | } |
|  | } |
|  |  |
|  | goTo(index: number) { |
|  | if (index >= 0 && index < this.pager.count) { |
|  | this.pager.index = index; |
|  | this.mode = 'quiz'; |
|  | } |
|  | } |
|  |  |
|  | isAnswered(question: Question) { |
|  | return question.options.find((x) => x.selected) |
|  | ? 'Answered' |
|  | : 'Not Answered'; |
|  | } |
|  |  |
|  | isCorrect(question: Question) { |
|  | return question.options.every((x) => x.selected === x.isAnswer) |
|  | ? 'correct' |
|  | : 'wrong'; |
|  | } |
|  |  |
|  | onSubmit() { |
|  | let answers = []; |
|  | this.quiz.questions.forEach((x) => |
|  | answers.push({ |
|  | quizId: this.quiz.id, |
|  | questionId: x.id, |
|  | answered: x.answered, |
|  | }) |
|  | ); |
|  |  |
|  | // Post your data to the server here. answers contains the questionId and the users' answer. |
|  | //console.log(this.quiz.questions); |
|  | this.mode = 'result'; |
|  | } |
|  | } |

|  |
| --- |
| Export class Option { |
|  | id: number; |
|  | questionId: number; |
|  | name: string; |
|  | isAnswer: boolean; |
|  | selected: boolean; |
|  |  |
|  | constructor(data: any) { |
|  | data = data || {}; |
|  | this.id = data.id; |
|  | this.questionId = data.questionId; |
|  | this.name = data.name; |
|  | this.isAnswer = data.isAnswer; |
|  | } |
|  | } |

|  |
| --- |
| import { Option } from './option'; |
|  |  |
|  | export class Question { |
|  | id: number; |
|  | name: string; |
|  | questionTypeId: number; |
|  | options: Option[]; |
|  | answered: boolean; |
|  |  |
|  | constructor(data: any) { |
|  | data = data || {}; |
|  | this.id = data.id; |
|  | this.name = data.name; |
|  | this.questionTypeId = data.questionTypeId; |
|  | this.options = []; |
|  | data.options.forEach(o => { |
|  | this.options.push(new Option(o)); |
|  | }); |
|  | } |
|  | } |

|  |
| --- |
| export class QuizConfig { |
|  | allowBack: boolean; |
|  | allowReview: boolean; |
|  | autoMove: boolean; // if boolean; it will move to next question automatically when answered. |
|  | duration: number; // indicates the time in which quiz needs to be completed. 0 means unlimited. |
|  | pageSize: number; |
|  | requiredAll: boolean; // indicates if you must answer all the questions before submitting. |
|  | richText: boolean; |
|  | shuffleQuestions: boolean; |
|  | shuffleOptions: boolean; |
|  | showClock: boolean; |
|  | showPager: boolean; |
|  | theme: string; |
|  |  |
|  | constructor(data: any) { |
|  | data = data || {}; |
|  | this.allowBack = data.allowBack; |
|  | this.allowReview = data.allowReview; |
|  | this.autoMove = data.autoMove; |
|  | this.duration = data.duration; |
|  | this.pageSize = data.pageSize; |
|  | this.requiredAll = data.requiredAll; |
|  | this.richText = data.richText; |
|  | this.shuffleQuestions = data.shuffleQuestions; |
|  | this.shuffleOptions = data.shuffleOptions; |
|  | this.showClock = data.showClock; |
|  | this.showPager = data.showPager; |
|  | } |
|  | } |

|  |
| --- |
| import { QuizConfig } from './quiz-config'; |
|  | import { Question } from './question'; |
|  |  |
|  | export class Quiz { |
|  | id: number; |
|  | name: string; |
|  | description: string; |
|  | config: QuizConfig; |
|  | questions: Question[]; |
|  |  |
|  | constructor(data: any) { |
|  | if (data) { |
|  | this.id = data.id; |
|  | this.name = data.name; |
|  | this.description = data.description; |
|  | this.config = new QuizConfig(data.config); |
|  | //console.log(`config: ${JSON.stringify(this.config)}`); |
|  | this.questions = []; |
|  | data.questions.forEach(q => { |
|  | this.questions.push(new Question(q)); |
|  | }); |
|  | } |
|  | } |
|  | } |

|  |
| --- |
| import { HttpClient } from '@angular/common/http'; |
|  | import { Injectable } from '@angular/core'; |
|  |  |
|  | @Injectable({ |
|  | providedIn: 'root', |
|  | }) |
|  | export class QuizService { |
|  | constructor(private http: HttpClient) {} |
|  |  |
|  | get(url: string) { |
|  | return this.http.get(url); |
|  | } |
|  |  |
|  | getAll() { |
|  | return [ |
|  | { |
|  | id: 'data/javascript.json', |
|  | name: 'JavaScript', |
|  | description: |
|  | "Let's Play javascript quiz that will help you clear the concepts and will prepare you for interviews. This is a basic level quiz and contains 10 Questions.", |
|  | imageUrl: 'assets/images/JS.png', |
|  | }, |
|  | { |
|  | id: 'data/aspnet.json', |
|  | name: 'Asp.Net', |
|  | description: |
|  | "Let's Play Asp.Net quiz that will help you clear the concepts and will prepare you for interviews. This is a basic level quiz and contains 10 Questions.", |
|  | imageUrl: 'assets/images/ASP.png', |
|  | }, |
|  | { |
|  | id: 'data/csharp.json', |
|  | name: 'C Sharp', |
|  | description: |
|  | "Let's Play C# quiz that will help you clear the concepts and will prepare you for interviews. This is a basic level quiz and contains 10 Questions.", |
|  | imageUrl: 'assets/images/CSHARP.png', |
|  | }, |
|  | ]; |
|  | } |
|  | } |

|  |  |  |
| --- | --- | --- |
| import { HttpClient } from '@angular/common/http'; | | |
|  | | | import { Injectable } from '@angular/core'; | |
|  | | |  | |
|  | | | @Injectable({ | |
|  | | | providedIn: 'root', | |
|  | | | }) | |
|  | | | export class QuizService { | |
|  | | | constructor(private http: HttpClient) {} | |
|  | | |  | |
|  | | | get(url: string) { | |
|  | | | return this.http.get(url); | |
|  | | | } | |
|  | | |  | |
|  | | | getAll() { | |
|  | | | return [ | |
|  | | | { | |
|  | | | id: 'data/javascript.json', | |
|  | | | name: 'JavaScript', | |
|  | | | description: | |
|  | | | "Let's Play javascript quiz that will help you clear the concepts and will prepare you for interviews. This is a basic level quiz and contains 10 Questions.", | |
|  | | | imageUrl: 'assets/images/JS.png', | |
|  | | | }, | |
|  | | | { | |
|  | | | id: 'data/aspnet.json', | |
|  | | | name: 'Asp.Net', | |
|  | | | description: | |
|  | | | "Let's Play Asp.Net quiz that will help you clear the concepts and will prepare you for interviews. This is a basic level quiz and contains 10 Questions.", | |
|  | | | imageUrl: 'assets/images/ASP.png', | |
|  | | | }, | |
|  | | | { | |
|  | | | id: 'data/csharp.json', | |
|  | | | name: 'C Sharp', | |
|  | | | description: | |
|  | | | "Let's Play C# quiz that will help you clear the concepts and will prepare you for interviews. This is a basic level quiz and contains 10 Questions.", | |
|  | | | imageUrl: 'assets/images/CSHARP.png', | |
|  | | | }, | |
|  | | | ]; | |
|  | | | } | |
|  | | | } | |
| import { NgModule } from '@angular/core'; | |
|  | | import { FormsModule } from '@angular/forms'; | | |
|  | | import { BrowserModule } from '@angular/platform-browser'; | | |
|  | |  | | |
|  | | import { AppComponent } from './app.component'; | | |
|  | | import { HttpClientModule } from '@angular/common/http'; | | |
|  | | import { QuizService } from './services/quiz.service'; | | |
|  | | import { QuizListComponent } from './components/quiz-list/quiz-list.component'; | | |
|  | | import { QuizComponent } from './components/quiz/quiz.component'; | | |
|  | | import { RouterModule, Routes } from '@angular/router'; | | |
|  | |  | | |
|  | | const routes: Routes = [ | | |
|  | | { path: 'quiz/:quizName', component: QuizComponent }, | | |
|  | | { path: 'quiz', component: QuizListComponent }, | | |
|  | | { path: '', redirectTo: '/quiz', pathMatch: 'full' }, | | |
|  | | { path: '\*\*', redirectTo: '/quiz', pathMatch: 'full' }, | | |
|  | | ]; | | |
|  | |  | | |
|  | | @NgModule({ | | |
|  | | declarations: [AppComponent, QuizListComponent, QuizComponent], | | |
|  | | imports: [ | | |
|  | | BrowserModule, | | |
|  | | FormsModule, | | |
|  | | HttpClientModule, | | |
|  | | RouterModule.forRoot(routes), | | |
|  | | ], | | |
|  | | providers: [QuizService], | | |
|  | | bootstrap: [AppComponent], | | |
|  | | }) | | |
|  | | export class AppModule {} | | |
| import { Component } from '@angular/core'; |
|  |  | | |
|  | @Component({ | | |
|  | selector: 'app-root', | | |
|  | templateUrl: './app.component.html', | | |
|  | styleUrls: ['./app.component.css'] | | |
|  | }) | | |
|  | export class AppComponent { | | |
|  | title = 'online-test-application'; | | |
|  | } | | |

|  |
| --- |
| body{ |
|  | min-height: 100vh; |
|  | display: flex; |
|  | flex-direction: column; |
|  | } |
|  |  |
|  | .btn { |
|  | border-radius: 0; |
|  | } |
|  |  |
|  | .option { |
|  | background-color: #cadaee; |
|  | font-size: 20px; |
|  | padding: 10px; |
|  | margin: 3px; |
|  | } |
|  |  |
|  | #quiz .options input[type=checkbox] { |
|  | height: 22px; |
|  | width: 22px; |
|  | } |
|  |  |
|  | .cursor-pointer { |
|  | cursor: pointer; |
|  | } |
|  |  |
|  | .result-question { |
|  | background-color: #eee; |
|  | margin: 4px; |
|  | padding: 6px; |
|  | } |
|  |  |
|  | @media only screen and (max-width: 480px) { |
|  |  |
|  | h1, |
|  | .h1 { |
|  | font-size: 22px; |
|  | } |
|  |  |
|  | h2, |
|  | .h2 { |
|  | font-size: 20px; |
|  | } |
|  |  |
|  | h3, |
|  | .h3 { |
|  | font-size: 18px; |
|  | } |
|  |  |
|  | .option { |
|  | font-size: 16px; |
|  | padding: 6px; |
|  | } |
|  | } |
|  |  |
|  | .card-img-top { |
|  | width: 100%; |
|  | height: 15vw; |
|  | object-fit: cover; |
|  | } |
|  |  |
|  | footer{ |
|  | margin-top: auto; |
|  | } |

|  |
| --- |
| import { enableProdMode } from '@angular/core'; |
|  | import { platformBrowserDynamic } from '@angular/platform-browser-dynamic'; |
|  |  |
|  | import { AppModule } from './app/app.module'; |
|  | import { environment } from './environments/environment'; |
|  |  |
|  | if (environment.production) { |
|  | enableProdMode(); |
|  | } |
|  |  |
|  | platformBrowserDynamic().bootstrapModule(AppModule) |
|  | .catch(err => console.error(err)); |

|  |
| --- |
| <!doctype html> |
|  | <html lang="en"> |
|  | <head> |
|  | <meta charset="utf-8"> |
|  | <title>OnlineTestApplication</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> |