**Online Test Application Source Code:**

Index.ts

export \* from './option';

export \* from './question';

export \* from './quiz';

export \* from './quiz-config';

Option.ts

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;

}

}

Question.ts

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));

});

}

}

Quiz-config.ts

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;

}

}

Quiz.ts

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);

this.questions = [];

data.questions.forEach(q => {

this.questions.push(new Question(q));

});

}

}

}

Quiz.html

<nav class="navbar navbar-default navbar-static-top bg-info">

<div class="container-fluid">

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

<img src="https://image.pngaaa.com/664/2956664-middle.png" alt="" width="50" height="40" class="d-inline-block align-text-top">

Online Test/Quizz Application

</a>

<div class="col-6 text-right mt-3" style="color: white;">

Select Quiz Type:

<select class="form-select form-select-lg mb-3" [(ngModel)]="quizName" (change)="loadQuiz(quizName)">

<option \*ngFor="let quiz of quizes" [value]="quiz.id">{{quiz.name}}</option>

</select>

</div>

</div>

</nav>

<div class="container">

<div id="quiz">

<h2 class="text-center font-weight-normal">{{quiz.name}}</h2>

<hr />

<br/>

<div \*ngIf="mode=='quiz' && quiz">

<div \*ngFor="let question of filteredQuestions;">

<div class="badge badge-info">Question {{pager.index + 1}} / {{pager.count}}.</div>

<div \*ngIf="config.duration" class="badge badge-warning float-right">Time: {{ellapsedTime}} / {{duration}}</div>

<h3 class="font-weight-normal">{{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="font-weight-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 />

<div class="text-center">

<button class="btn btn-default" \*ngIf="config.allowBack" (click)="goTo(0);">First</button>

<button class="btn btn-default" \*ngIf="config.allowBack" (click)="goTo(pager.index - 1);">Prev</button>

<button class="btn btn-primary" (click)="goTo(pager.index + 1);">Next</button>

<button class="btn btn-default" \*ngIf="config.allowBack" (click)="goTo(pager.count - 1);">Last</button>

<!-- <pagination \*ngIf="config.showPager" direction-links="false" total-items="totalItems" items-per-page="itemsPerPage" ng-model="currentPage" ng-change="pageChanged()"></pagination> -->

</div>

</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</h2>

<div class="alert-info" style="padding: 2rem;">

<h4>Correct Answers: {{correctAns}}</h4>

<h4>Your Score:{{percentScore}}%</h4>

</div>

<hr/>

<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 class = "text-center row mb-5" \*ngIf="mode!='result'">

<div class="col-2"></div>

<div class="col-3"><button class="btn btn-secondary" style="width:90%" (click)="mode = 'quiz'">Quiz</button></div>

<div class="col-3"><button class="btn btn-primary" style="width:100%" (click)="onSubmit();">Submit Quiz</button></div>

<div class="col-3"><button class="btn btn-info" style="width:90%" (click)="mode = 'review'">Review</button></div>

</div>

</div>

</div>

Quiz.component.ts

import { Component, OnInit } from "@angular/core";

import { QuizService } from "../services/quiz.service";

import { HelperService } from "../services/helper.service";

import { Option, Question, Quiz, QuizConfig } from "../models/index";

@Component({

selector: "app-quiz",

templateUrl: "./quiz.component.html",

styleUrls: ["./quiz.component.css"],

providers: [QuizService],

})

export class QuizComponent implements OnInit {

quizes: any[];

quiz: Quiz = new Quiz(null);

mode = "quiz";

quizName: string;

config: QuizConfig = {

allowBack: true,

allowReview: true,

autoMove: false, // if true, it will move to next question automatically when answered.

duration: 600, // 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 = "";

correctAns: number = 0;

percentScore: number = 0;

constructor(private quizService: QuizService) {}

ngOnInit() {

this.quizes = this.quizService.getAll();

this.quizName = this.quizes[0].id;

this.loadQuiz(this.quizName);

}

loadQuiz(quizName: string) {

this.quizService.get(quizName).subscribe((res) => {

this.quiz = new Quiz(res);

this.pager.count = 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.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,

isCorrect: x.options,

})

);

answers.forEach(function (elem) {

elem.isCorrect.forEach(function (opt) {

if (opt.isAnswer === true) {

console.log(opt.name);

}

});

});

// Post your data to the server here. answers contains the questionId and the users' answer.

console.log(this.quiz.questions);

this.calculateResult(this.quiz.questions);

this.mode = "result";

}

calculateResult(questions: Question[]) {

this.correctAns=0;

for (var question of questions) {

if (this.isCorrect(question) == "correct") {

this.correctAns++;

}

}

this.percentScore=(this.correctAns/questions.length)\*100;

}

}

Helper.service.ts;

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

@Injectable()

export class HelperService {

static toBool(val) {

if (val === undefined || val === null || val === '' || val === 'false' || val === 'False') {

return false;

} else {

return true;

}

}

static shuffle(array) {

let currentIndex = array.length, temp, randomIndex;

while (0 !== currentIndex) {

randomIndex = Math.floor(Math.random() \* currentIndex);

currentIndex -= 1;

temp = array[currentIndex];

array[currentIndex] = array[randomIndex];

array[randomIndex] = temp;

}

return array;

}

static extend(out) {

out = out || {};

for (let i = 1; i < arguments.length; i++) {

if (!arguments[i]) {

continue;

}

for (const key in arguments[i]) {

if (arguments[i].hasOwnProperty(key)) {

out[key] = arguments[i][key];

}

}

}

return out;

}

}

Quiz.service.ts

import { HttpClient } from '@angular/common/http';

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

@Injectable()

export class QuizService {

constructor(private http: HttpClient) { }

get(url: string) {

return this.http.get(url);

}

getAll() {

return [

{ id: 'data/angular.json', name: 'Angular Quizz' },

{ id: 'data/JSON.json', name: 'JSON Quizz' },

{ id: 'data/JavaScript.json', name: 'JavaScript Quizz' },

];

}

}

App.component.html:

<app-quiz></app-quiz>

<!-- Footer -->

<footer class="footer fixed-bottom navbar-expand-sm navbar-dark bg-dark mt-5">

<!-- Copyright -->

<div class="footer footer-copyright text-center py-3 " style="color: white;">© 2022 Copyright:

Developed by Deepthi Reddy.

<a href="mailto:Deepthireddy@gmail.com?subject=Query%20regarding%20the Quizz%20application%20project">Contact Us</a>

</div>

<!-- Copyright -->

</footer>

<!-- Footer -->

App.module.ts

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

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

import { FormsModule } from '@angular/forms';

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

import { QuizComponent } from './quiz/quiz.component';

import { HttpClientModule } from '@angular/common/http';

@NgModule({

declarations: [

AppComponent,

QuizComponent

],

imports: [

BrowserModule,

FormsModule,

HttpClientModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

Index.html

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Online Quiz</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>

Main.ts

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.log(err));

Test.ts

// This file is required by karma.conf.js and loads recursively all the .spec and framework files

import 'zone.js/dist/zone-testing';

import { getTestBed } from '@angular/core/testing';

import {

BrowserDynamicTestingModule,

platformBrowserDynamicTesting

} from '@angular/platform-browser-dynamic/testing';

declare const require: any;

// First, initialize the Angular testing environment.

getTestBed().initTestEnvironment(

BrowserDynamicTestingModule,

platformBrowserDynamicTesting()

);

// Then we find all the tests.

const context = require.context('./', true, /\.spec\.ts$/);

// And load the modules.

context.keys().map(context);

Style.css

.btn{

border-radius: 4;

}

.h1{

text-decoration-color: aqua;

}

.option{

background-color:#faf8c07d;

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;

text-align: center;

}

h2,.h2{

font-size:20px;

}

h3,.h3{

font-size:18px;

}

.option {

font-size: 14px;

padding: 6px;

}

footer{

opacity: 0.5;

height:3rem;

}

.navbar{

opacity: 0.8;

}

}

Angular.json

{

"$schema": "./node\_modules/@angular/cli/lib/config/schema.json",

"version": 1,

"newProjectRoot": "projects",

"projects": {

"ng6-quiz": {

"root": "",

"sourceRoot": "src",

"projectType": "application",

"prefix": "app",

"schematics": {},

"architect": {

"build": {

"builder": "@angular-devkit/build-angular:browser",

"options": {

"outputPath": "dist/ng6-quiz",

"index": "src/index.html",

"main": "src/main.ts",

"polyfills": "src/polyfills.ts",

"tsConfig": "src/tsconfig.app.json",

"assets": [

"src/favicon.ico",

"src/data",

"src/assets"

],

"styles": [

"node\_modules/bootstrap/dist/css/bootstrap.css",

"src/styles.css"

],

"scripts": [

"node\_modules/jquery/dist/jquery.min.js",

"node\_modules/bootstrap/dist/js/bootstrap.bundle.js"

]

},

"configurations": {

"production": {

"fileReplacements": [

{

"replace": "src/environments/environment.ts",

"with": "src/environments/environment.prod.ts"

}

],

"optimization": true,

"outputHashing": "all",

"sourceMap": false,

"extractCss": true,

"namedChunks": false,

"aot": true,

"extractLicenses": true,

"vendorChunk": false,

"buildOptimizer": true

}

}

},

"serve": {

"builder": "@angular-devkit/build-angular:dev-server",

"options": {

"browserTarget": "ng6-quiz:build"

},

"configurations": {

"production": {

"browserTarget": "ng6-quiz:build:production"

}

}

},

"extract-i18n": {

"builder": "@angular-devkit/build-angular:extract-i18n",

"options": {

"browserTarget": "ng6-quiz:build"

}

},

"test": {

"builder": "@angular-devkit/build-angular:karma",

"options": {

"main": "src/test.ts",

"polyfills": "src/polyfills.ts",

"tsConfig": "src/tsconfig.spec.json",

"karmaConfig": "src/karma.conf.js",

"styles": [

"styles.css"

],

"scripts": [],

"assets": [

"src/favicon.ico",

"src/assets"

]

}

},

"lint": {

"builder": "@angular-devkit/build-angular:tslint",

"options": {

"tsConfig": [

"src/tsconfig.app.json",

"src/tsconfig.spec.json"

],

"exclude": [

"\*\*/node\_modules/\*\*"

]

}

}

}

},

"ng6-quiz-e2e": {

"root": "e2e/",

"projectType": "application",

"architect": {

"e2e": {

"builder": "@angular-devkit/build-angular:protractor",

"options": {

"protractorConfig": "e2e/protractor.conf.js",

"devServerTarget": "ng6-quiz:serve"

}

},

"lint": {

"builder": "@angular-devkit/build-angular:tslint",

"options": {

"tsConfig": "e2e/tsconfig.e2e.json",

"exclude": [

"\*\*/node\_modules/\*\*"

]

}

}

}

}

},

"defaultProject": "ng6-quiz",

"cli": {

"analytics": false

}

}

Package.json

{

"name": "quiz-app",

"version": "0.0.3",

"author": "Manish Kumar Arora",

"scripts": {

"ng": "ng",

"start": "ng serve",

"build": "ng build",

"test": "ng test",

"lint": "ng lint",

"e2e": "ng e2e"

},

"private": true,

"dependencies": {

"@angular/animations": "^8.0.2",

"@angular/common": "^8.0.2",

"@angular/compiler": "^8.0.2",

"@angular/core": "^8.0.2",

"@angular/forms": "^8.0.2",

"@angular/platform-browser": "^8.0.2",

"@angular/platform-browser-dynamic": "^8.0.2",

"@angular/router": "^8.0.2",

"bootstrap": "^4.1.1",

"core-js": "^3.1.4",

"jquery": "^3.3.1",

"rxjs": "^6.0.0",

"zone.js": "^0.9.1"

},

"devDependencies": {

"@angular-devkit/build-angular": "^0.800.6",

"@angular/cli": "~8.0.3",

"@angular/compiler-cli": "^8.0.2",

"@angular/language-service": "^8.0.2",

"@types/jasmine": "~3.3.13",

"@types/jasminewd2": "~2.0.3",

"@types/node": "^12.0.10",

"codelyzer": "~5.1.0",

"jasmine-core": "~3.4.0",

"jasmine-spec-reporter": "~4.2.1",

"karma": "~4.1.0",

"karma-chrome-launcher": "~2.2.0",

"karma-coverage-istanbul-reporter": "~2.0.5",

"karma-jasmine": "~2.0.1",

"karma-jasmine-html-reporter": "^1.4.2",

"protractor": "~5.4.2",

"ts-node": "~8.3.0",

"tslint": "~5.17.0",

"typescript": "3.4.5"

}

}