**Online Test Application**

Code:

index.html

<!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>

app.module.ts

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 {}

app.component.html

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

  <a class="navbar-brand ml-2 fw-bolder" routerLink="/quiz">Online Quiz</a>

  <button

    class="navbar-toggler"

    type="button"

    data-toggle="collapse"

    data-target="#navbarNav"

    aria-controls="navbarNav"

    aria-expanded="false"

    aria-label="Toggle navigation"

  >

    <span class="navbar-toggler-icon"></span>

  </button>

  <div class="collapse navbar-collapse" id="navbarNav">

    <ul class="navbar-nav ms-auto">

      <li class="nav-item">

        <a

          class="nav-link btn-outline-primary btn-sm text-light fw-bolder"

          routerLink="/quiz"

          >Home</a

        >

      </li>

    </ul>

  </div>

</nav>

<div class="container">

  <router-outlet></router-outlet>

</div>

<footer class="bg-dark text-center text-lg-start mt-5">

  <!-- Copyright -->

  <div class="text-center p-3 text-light fw-bold">

    © 2021 Copyright:

    <a class="text-light fw-bold" routerLink="/quiz">Online Quiz Application</a>

  </div>

  <!-- Copyright -->

</footer>

app.component.ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'online-test-application';

}

quiz.component.html

<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>

      </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(pager.index - 1)"

      >

        Prev

      </button>

      <button

        class="btn btn-outline-primary mx-2"

        (click)="goTo(pager.index + 1)"

      >

        Next

      </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-info mx-2" (click)="mode = 'review'">Review</button>

    <button class="btn btn-primary mx-2" (click)="onSubmit()">

      Submit Quiz

    </button>

  </div>

</div>

quiz.component.ts

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,

  };

  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.mode = 'quiz';

  }

  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';

  }

}

quiz-list.component.html

<div class="container mt-5">

  <div class="row">

    <div \*ngFor="let tempQuiz of quizes" class="col-md-4">

      <div class="card">

        <img

          src="{{ tempQuiz.imageUrl }}"

          class="card-img-top"

          alt="Quiz Image"

        />

        <div class="card-body text-center">

          <h5 class="card-title">

            {{ tempQuiz.name }}

          </h5>

          <hr />

          <p class="card-text">

            {{ tempQuiz.description }}

          </p>

          <a

            routerLink="/quiz/{{ tempQuiz.name }}"

            class="btn btn-outline-primary"

            >Give {{ tempQuiz.name }} Quiz</a

          >

        </div>

      </div>

    </div>

  </div>

</div>

quiz-list.component.ts

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

import { Quiz } from 'src/app/models/quiz';

import { QuizService } from 'src/app/services/quiz.service';

@Component({

  selector: 'app-quiz-list',

  templateUrl: './quiz-list.component.html',

  styleUrls: ['./quiz-list.component.css'],

})

export class QuizListComponent implements OnInit {

  quizes: any[] = [];

  constructor(private quizService: QuizService) {}

  ngOnInit(): void {

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

  }

}

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);

            //console.log(`config: ${JSON.stringify(this.config)}`);

            this.questions = [];

            data.questions.forEach(q => {

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

            });

        }

    }

}

quiz.service.spec.ts

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

import { QuizService } from './quiz.service';

describe('QuizService', () => {

  let service: QuizService;

  beforeEach(() => {

    TestBed.configureTestingModule({});

    service = TestBed.inject(QuizService);

  });

  it('should be created', () => {

    expect(service).toBeTruthy();

  });

});

quiz.service.ts

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:

          "This is a basic level quiz and contains 10 Questions.",

        imageUrl: 'assets/images/JS.png',

      },

    ];

  }

}

javascript.json

{

    "id": 1,

    "name": "JavaScript Quiz",

    "description": "JavaScript Quiz (Basic Multiple Choice Questions for JavaScript Developers)",

    "questions": [

        {

            "id": 1010,

            "name": "Which HTML tag do we use to put the JavaScript?",

            "questionTypeId": 1,

            "options": [

                {

                    "id": 1055,

                    "questionId": 1010,

                    "name": "<javascript>",

                    "isAnswer": false

                },

                {

                    "id": 1056,

                    "questionId": 1010,

                    "name": "<script>",

                    "isAnswer": true

                },

                {

                    "id": 1057,

                    "questionId": 1010,

                    "name": "<js>",

                    "isAnswer": false

                },

                {

                    "id": 1058,

                    "questionId": 1010,

                    "name": "None of the above",

                    "isAnswer": false

                }

            ],

            "questionType": {

                "id": 1,

                "name": "Multiple Choice",

                "isActive": true

            }

        },

        {

            "id": 1011,

            "name": "Which built-in method calls a function for each element in the array?",

            "questionTypeId": 1,

            "options": [

                {

                    "id": 1055,

                    "questionId": 1010,

                    "name": "while()",

                    "isAnswer": false

                },

                {

                    "id": 1057,

                    "questionId": 1010,

                    "name": "loop",

                    "isAnswer": false

                },

                {

                    "id": 1056,

                    "questionId": 1010,

                    "name": "forEach",

                    "isAnswer": true

                },

                {

                    "id": 1058,

                    "questionId": 1010,

                    "name": "takeUntil",

                    "isAnswer": false

                }

            ],

            "questionType": {

                "id": 1,

                "name": "Multiple Choice",

                "isActive": true

            }

        },

        {

            "id": 1012,

            "name": "What is the difference between let and var?",

            "questionTypeId": 1,

            "options": [

                {

                    "id": 1055,

                    "questionId": 1010,

                    "name": "let has local scope",

                    "isAnswer": true

                },

                {

                    "id": 1057,

                    "questionId": 1010,

                    "name": "Both are same",

                    "isAnswer": false

                },

                {

                    "id": 1056,

                    "questionId": 1010,

                    "name": "var is new data type",

                    "isAnswer": false

                },

                {

                    "id": 1058,

                    "questionId": 1010,

                    "name": "let consumes more cpu and ram",

                    "isAnswer": false

                }

            ],

            "questionType": {

                "id": 1,

                "name": "Multiple Choice",

                "isActive": true

            }

        },

        {

            "id": 1013,

            "name": "What is TypeScript?",

            "questionTypeId": 1,

            "options": [

                {

                    "id": 1055,

                    "questionId": 1010,

                    "name": "A Language based on Javascript",

                    "isAnswer": true

                },

                {

                    "id": 1057,

                    "questionId": 1010,

                    "name": "script that runs on browser",

                    "isAnswer": false

                },

                {

                    "id": 1056,

                    "questionId": 1010,

                    "name": "A DataType Collection of Javascript",

                    "isAnswer": false

                },

                {

                    "id": 1058,

                    "questionId": 1010,

                    "name": "None of the above",

                    "isAnswer": false

                }

            ],

            "questionType": {

                "id": 1,

                "name": "Multiple Choice",

                "isActive": true

            }

        },

        {

            "id": 1014,

            "name": "Which of the following is right syntex for arrow function?",

            "questionTypeId": 1,

            "options": [

                {

                    "id": 1055,

                    "questionId": 1010,

                    "name": "a -> { return b; }",

                    "isAnswer": false

                },

                {

                    "id": 1057,

                    "questionId": 1010,

                    "name": "x <= x + y;",

                    "isAnswer": false

                },

                {

                    "id": 1056,

                    "questionId": 1010,

                    "name": "x <- x + 5;",

                    "isAnswer": false

                },

                {

                    "id": 1058,

                    "questionId": 1010,

                    "name": "x => x + 5;",

                    "isAnswer": true

                }

            ],

            "questionType": {

                "id": 1,

                "name": "Multiple Choice",

                "isActive": true

            }

        },

        {

            "id": 1015,

            "name": "Which new ES6 syntax helps with formatting output text - mixing variables with string literals, for example.",

            "questionTypeId": 1,

            "options": [

                {

                    "id": 1055,

                    "questionId": 1010,

                    "name": "Generator Functions",

                    "isAnswer": false

                },

                {

                    "id": 1057,

                    "questionId": 1010,

                    "name": "Arrow Functions",

                    "isAnswer": false

                },

                {

                    "id": 1056,

                    "questionId": 1010,

                    "name": "Template Strings",

                    "isAnswer": true

                },

                {

                    "id": 1058,

                    "questionId": 1010,

                    "name": "Set Data Structure",

                    "isAnswer": false

                }

            ],

            "questionType": {

                "id": 1,

                "name": "Multiple Choice",

                "isActive": true

            }

        },

        {

            "id": 1016,

            "name": "Which ES6 feature helps in merging of a number of changed properties into an existing object?",

            "questionTypeId": 1,

            "options": [

                {

                    "id": 1055,

                    "questionId": 1010,

                    "name": "Class syntex",

                    "isAnswer": false

                },

                {

                    "id": 1056,

                    "questionId": 1010,

                    "name": "Object.assign()",

                    "isAnswer": true

                },

                {

                    "id": 1057,

                    "questionId": 1010,

                    "name": "map data structure",

                    "isAnswer": false

                },

                {

                    "id": 1058,

                    "questionId": 1010,

                    "name": "Array.includes(obj);",

                    "isAnswer": false

                }

            ],

            "questionType": {

                "id": 1,

                "name": "Multiple Choice",

                "isActive": true

            }

        },

        {

            "id": 1017,

            "name": "What is the difference between == and === ?",

            "questionTypeId": 1,

            "options": [

                {

                    "id": 1055,

                    "questionId": 1010,

                    "name": "=== throws syntex error",

                    "isAnswer": false

                },

                {

                    "id": 1056,

                    "questionId": 1010,

                    "name": "== checks values only, === checks types as well",

                    "isAnswer": true

                },

                {

                    "id": 1057,

                    "questionId": 1010,

                    "name": "=== is reference type check only",

                    "isAnswer": false

                },

                {

                    "id": 1058,

                    "questionId": 1010,

                    "name": "Both are same",

                    "isAnswer": false

                }

            ],

            "questionType": {

                "id": 1,

                "name": "Multiple Choice",

                "isActive": true

            }

        },

        {

            "id": 1018,

            "name": "Which of the following is NOT the method of an Array?",

            "questionTypeId": 1,

            "options": [

                {

                    "id": 1055,

                    "questionId": 1010,

                    "name": ".map()",

                    "isAnswer": false

                },

                {

                    "id": 1057,

                    "questionId": 1010,

                    "name": ".includes()",

                    "isAnswer": false

                },

                {

                    "id": 1056,

                    "questionId": 1010,

                    "name": ".subscribe()",

                    "isAnswer": true

                },

                {

                    "id": 1058,

                    "questionId": 1010,

                    "name": ".flatMap()",

                    "isAnswer": false

                }

            ],

            "questionType": {

                "id": 1,

                "name": "Multiple Choice",

                "isActive": true

            }

        },

        {

            "id": 1019,

            "name": "What will be the output of the following code: ['a', 'b', 'c'].fill(7, 1, 2);?",

            "questionTypeId": 1,

            "options": [

                {

                    "id": 1055,

                    "questionId": 1010,

                    "name": "['a', 7, 'c']",

                    "isAnswer": true

                },

                {

                    "id": 1056,

                    "questionId": 1010,

                    "name": "['a', 7, 7, 'b', 'c']",

                    "isAnswer": false

                },

                {

                    "id": 1057,

                    "questionId": 1010,

                    "name": "['a', 'b', 'c']",

                    "isAnswer": false

                },

                {

                    "id": 1058,

                    "questionId": 1010,

                    "name": "['7', 7, 'c']",

                    "isAnswer": false

                }

            ],

            "questionType": {

                "id": 1,

                "name": "Multiple Choice",

                "isActive": true

            }

        }

    ]

}