Online Test Application

(Source Code)

Author	Aman Kumar
Purpose	Project Source Code
Date	24/02/2022
Version	1.0

Version History:

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>

        <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>
```

Quiz.component.spec.ts

```
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();
  });
});
```

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,
  };
  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';
  }
}
```

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.spec.ts

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

import { QuizListComponent } from './quiz-list.component';

describe('QuizListComponent', () => {
  let component: QuizListComponent;
  let fixture: ComponentFixture<QuizListComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ QuizListComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(QuizListComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

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();
  }
}
```

Options.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:
          "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',
      },
    ];
  }
}
```

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>
```

```html
<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.spec.ts

```typescript
import { TestBed } from '@angular/core/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'online-test-application'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('online-test-application');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement;
    expect(compiled.querySelector('.content span').textContent).toContain('online-
test-application app is running!');
  });
});
```

App.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

```
export class AppComponent {
  title = 'online-test-application';
}
```

Aspnet.json

```
{
    "id": 1,
    "name": "Asp.Net Quiz",
    "description": "Asp.Net Quiz (contains webform, mvc, web API, etc.)",
    "questions": [
        {
            "id": 1010,
            "name": "ASP.NET webform separates the HTML output from program logic
using a feature named as",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "Exception",
                    "isAnswer": false
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "Code-behind",
                    "isAnswer": true
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "Code-front",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "None of the above",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1011,
            "name": "The feature in ASP.NET 2.0 that is used to fire a normal
postback to a different page in the application is called",
            "questionTypeId": 1,
            "options": [
                {
```

```
                "id": 1055,
                "questionId": 1010,
                "name": "Theme",
                "isAnswer": false
            },
            {
                "id": 1057,
                "questionId": 1010,
                "name": "Code-front",
                "isAnswer": false
            },
            {
                "id": 1056,
                "questionId": 1010,
                "name": "Cross Page Posting",
                "isAnswer": true
            },
            {
                "id": 1058,
                "questionId": 1010,
                "name": "None of the above",
                "isAnswer": false
            }
        ],
        "questionType": {
            "id": 1,
            "name": "Multiple Choice",
            "isActive": true
        }
    },
    {
        "id": 1012,
        "name": "What class does the ASP.NET Web Form class inherit from by
default?",
        "questionTypeId": 1,
        "options": [
            {
                "id": 1055,
                "questionId": 1010,
                "name": "System.Web.UI.Page",
                "isAnswer": true
            },
            {
                "id": 1057,
                "questionId": 1010,
                "name": "System.Web.UI.Form",
                "isAnswer": false
            },
            {
                "id": 1056,
                "questionId": 1010,
                "name": "System.Web.GUI.Page",
                "isAnswer": false
            },
            {
                "id": 1058,
                "questionId": 1010,
                "name": "System.Web.Form",
                "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1013,
            "name": "What does MVC stand for?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "Model View Controller",
                    "isAnswer": true
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "Maximum Virtual Control",
                    "isAnswer": false
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "Microsoft Visual Core",
                    "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 does NOT require type casting?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "Session",
                    "isAnswer": false
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "TempData",
                "isAnswer": false
            },
            {
                "id": 1056,
                "questionId": 1010,
                "name": "ViewData",
                "isAnswer": false
            },
            {
                "id": 1058,
                "questionId": 1010,
                "name": "ViewBag",
                "isAnswer": true
            }
        ],
        "questionType": {
            "id": 1,
            "name": "Multiple Choice",
            "isActive": true
        }
    },
    {
        "id": 1015,
        "name": "Which is the correct order of Page life-cycle in asp.net
webform?",
        "questionTypeId": 1,
        "options": [
            {
                "id": 1055,
                "questionId": 1010,
                "name": "Init, PreRender, Load",
                "isAnswer": false
            },
            {
                "id": 1057,
                "questionId": 1010,
                "name": "Load, PreRender, Init",
                "isAnswer": false
            },
            {
                "id": 1056,
                "questionId": 1010,
                "name": "Init, Load, PreRender",
                "isAnswer": true
            },
            {
                "id": 1058,
                "questionId": 1010,
                "name": "None of the above",
                "isAnswer": false
            }
        ],
        "questionType": {
            "id": 1,
            "name": "Multiple Choice",
            "isActive": true
        }
    },
    {
            "id": 1016,
            "name": "Which of these data source controls do not implement
caching?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "ObjectDataSource",
                    "isAnswer": false
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "LinqDataSource",
                    "isAnswer": true
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "SqlDataSource",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "XmlDataSource",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1017,
            "name": "Which tag asp:Label control by default renders to?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "div",
                    "isAnswer": false
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "span",
                    "isAnswer": true
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "body",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "label",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1018,
            "name": "Which method do you use to explicitly kill a user's
session?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "Session.Terminate()",
                    "isAnswer": false
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "Session.TimeOut()",
                    "isAnswer": false
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "Session.Abondon()",
                    "isAnswer": true
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "Session.Kill()",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1019,
            "name": "Which of the following object is ideal for keeping data alive
for a single request?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "HttpContext",
                    "isAnswer": true
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "Session",
                    "isAnswer": false
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "Cookies",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "SqlServer",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        }
    ]
}
```

Csharp.json

```
{
    "id": 2,
    "name": "C# and .Net Framework",
    "description": "C# and .Net Quiz (contains C#, .Net Framework, Linq, etc.)",
    "config": {
        "shuffleQuestions": true,
        "showPager": false,
        "allowBack": true,
        "autoMove": true
    },
    "questions": [
        {
            "id": 1010,
            "name": "Which of the following assemblies can be stored in Global 
Assembly Cache?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "Private Assemblies",
                    "isAnswer": false
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "Friend Assemblies",
                    "isAnswer": false
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "Public Assemblies",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "Shared Assemblies",
                    "isAnswer": true
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1011,
            "name": "Which of the following .NET components can be used to remove
unused references from the managed heap?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "Language Infrastructure",
                    "isAnswer": false
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "CLR",
                    "isAnswer": false
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "Garbage Collector",
                    "isAnswer": true
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "Class Loader",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "CTS",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1012,
            "name": "Which of the following utilities can be used to compile
managed assemblies into processor-specific native code?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "gacutil",
                    "isAnswer": false
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "ngen",
                    "isAnswer": true
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "dumpbin",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "ildasm",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1013,
            "name": "Which of the following is NOT an Arithmetic operator in
C#.NET?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "** (Double Star)",
                    "isAnswer": true
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "+ (Plus)",
                    "isAnswer": false
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "/ (Divide)",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "% (Modulo)",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1014,
            "name": "Which of the following statements is correct about an
interface used in C#.NET?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "If a class implements an interface partially, then it
should be an abstract class.",
                    "isAnswer": true
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "A class cannot implement an interface partially.",
                    "isAnswer": false
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "An interface can contain static methods.",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "An interface can contain static data.",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1015,
            "name": "What does the term <strong>immutable</strong> means in term
of string objects?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "We can modify characters included in the string",
                    "isAnswer": false
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "We cannot modify characters contained in the string",
                    "isAnswer": true
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "We cannot perform various operation of comparison,
inserting, appending etc",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "None of the above",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1016,
            "name": "Which of the following is NOT a .NET Exception class?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "Exception",
                    "isAnswer": false
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "StackMemoryException",
                    "isAnswer": true
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "DivideByZeroException",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "InvalidOperationException",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1017,
            "name": "In C#.NET if we do not catch the exception thrown at runtime
then which of the following will catch it?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "Compiler",
                    "isAnswer": false
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "CLR",
                    "isAnswer": true
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "Linker",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "Operating system",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1018,
            "name": "Which of the following statements are correct about
delegates?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "Delegates cannot be used to call a static method of a
class.",
                    "isAnswer": false
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "Delegates cannot be used to call procedures that
receive variable number of arguments.",
                    "isAnswer": true
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "If signatures of two methods are same they can be
called through the same delegate object.",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "Delegates cannot be used to call an instance
function. Delegates cannot be used to call an instance subroutine.",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        },
        {
            "id": 1019,
            "name": "Which of the following does NOT represent Integer?",
            "questionTypeId": 1,
            "options": [
                {
                    "id": 1055,
                    "questionId": 1010,
                    "name": "Char",
                    "isAnswer": true
                },
                {
                    "id": 1056,
                    "questionId": 1010,
                    "name": "Byte",
                    "isAnswer": false
                },
                {
                    "id": 1057,
                    "questionId": 1010,
                    "name": "Short",
                    "isAnswer": false
                },
                {
                    "id": 1058,
                    "questionId": 1010,
                    "name": "Long",
                    "isAnswer": false
                }
            ],
            "questionType": {
                "id": 1,
                "name": "Multiple Choice",
                "isActive": true
            }
        }
    ]
}
```

<u>Javascript.json</u>

```
{
    "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,
```

```json
                    "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
            }
        }
    ]
}
```

Angular.json

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "cli": {
    "analytics": false
  },
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "online-test-application": {
      "projectType": "application",
      "schematics": {},
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
```

```json
          "options": {
            "outputPath": "dist/online-test-application",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "aot": true,
            "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,
              "namedChunks": false,
              "extractLicenses": true,
              "vendorChunk": false,
              "buildOptimizer": true,
              "budgets": [
                {
                  "type": "initial",
                  "maximumWarning": "2mb",
                  "maximumError": "5mb"
                },
                {
                  "type": "anyComponentStyle",
                  "maximumWarning": "6kb",
                  "maximumError": "10kb"
                }
              ]
            }
          }
        },
        "serve": {
          "builder": "@angular-devkit/build-angular:dev-server",
          "options": {
            "browserTarget": "online-test-application:build"
          },
          "configurations": {
            "production": {
              "browserTarget": "online-test-application:build:production"
            }
          }
        },
        "extract-i18n": {
          "builder": "@angular-devkit/build-angular:extract-i18n",
          "options": {
            "browserTarget": "online-test-application:build"
          }
        },
        "test": {
          "builder": "@angular-devkit/build-angular:karma",
          "options": {
            "main": "src/test.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.spec.json",
            "karmaConfig": "karma.conf.js",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "src/styles.css"
            ],
            "scripts": []
          }
        },
        "lint": {
          "builder": "@angular-devkit/build-angular:tslint",
          "options": {
            "tsConfig": [
              "tsconfig.app.json",
              "tsconfig.spec.json",
              "e2e/tsconfig.json"
            ],
            "exclude": [
              "**/node_modules/**"
            ]
          }
        },
        "e2e": {
          "builder": "@angular-devkit/build-angular:protractor",
          "options": {
            "protractorConfig": "e2e/protractor.conf.js",
            "devServerTarget": "online-test-application:serve"
          },
          "configurations": {
            "production": {
              "devServerTarget": "online-test-application:serve:production"
            }
          }
        }
      }
    }
  },
  "defaultProject": "online-test-application"
}
```