Online Test App
Project Source Code

Version History:

Author:	Balanarenthiran. A
Purpose:	Project Source Code and GITHUB
Date:	26th Mar, 2022.
Version:	1.0

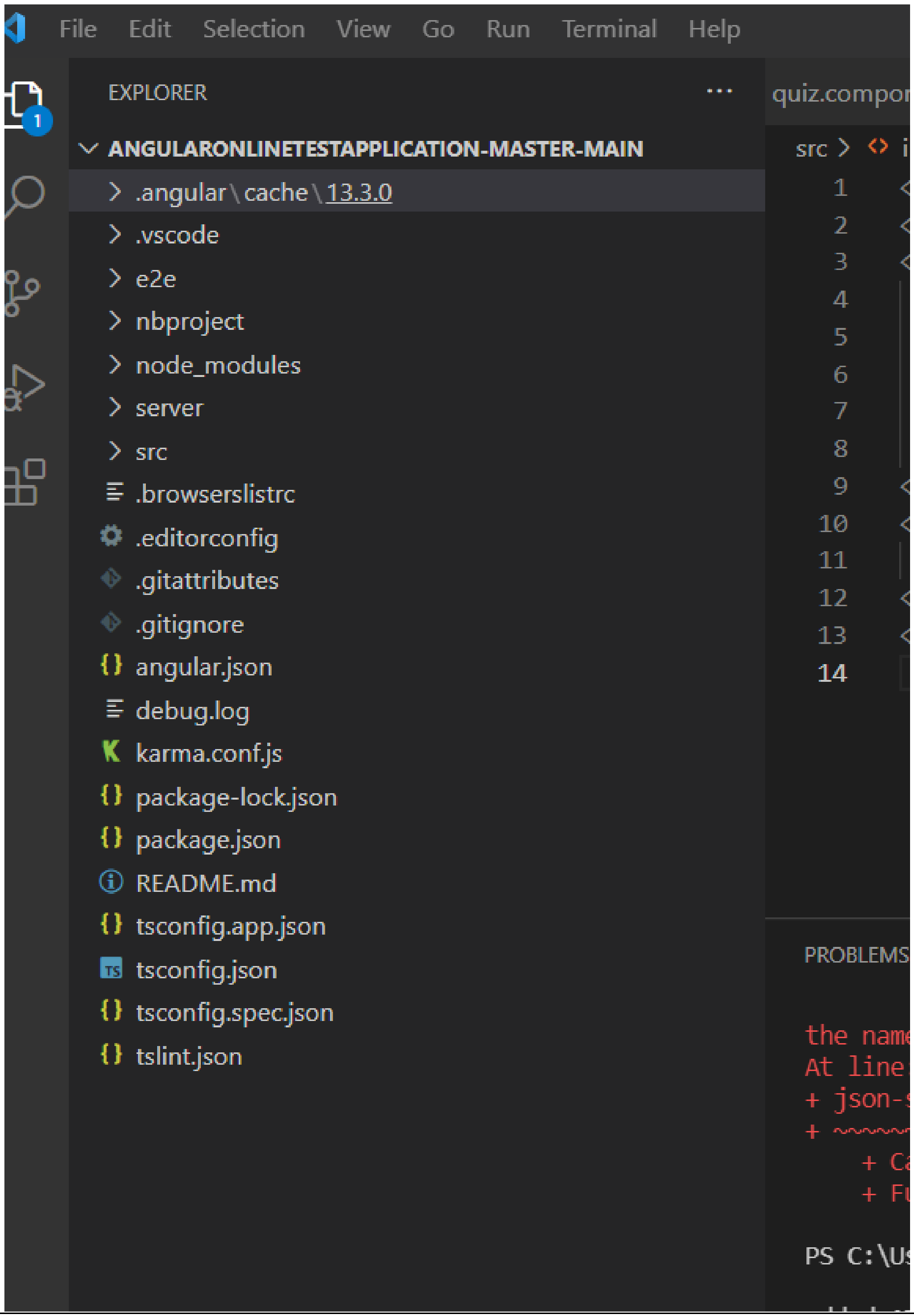
Table of Content:

Project GITHUB Link:

Repository Name:	course5-onlinetestappproj
Github Link:	https://github.com/bala-tw/course5-onlinetestappproj

Project Code:

Folder Structure



```
ANGULARONLINETESTAPPLICATION-MASTER-MAIN
  > .angular\cache\13.3.0
  > .vscode
  > e2e
  > nbproject
  > node_modules
  > server
  > src
    .browserslistrc
    .editorconfig
    .gitattributes
    .gitignore
    angular.json
    debug.log
    karma.conf.js
    package-lock.json
    package.json
    README.md
    tsconfig.app.json
    tsconfig.json
    tsconfig.spec.json
    tslint.json
```

```
PROBLEMS
the name
At line
+ json-
+ ~~~~~~
    + Ca
    + Fu

PS C:\U
```

app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
```

```
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'onlinetestapplication';
}
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule,NO_ERRORS_SCHEMA, CUSTOM_ELEMENTS_SCHEMA } from
'@angular/core';
import { AppComponent } from './app.component';
import { QuizComponent } from './quiz/quiz.component';
import { ResultComponent } from './result/result.component';
import { ReviewComponent } from './review/review.component';
import { ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { QuizService } from './services/quiz.service';
import { WelcomepageComponent } from './welcomepage/welcomepage.component';
import { MatCardModule } from '@angular/material/card';
import { MatRadioModule } from '@angular/material/radio';
import { MatIconModule } from '@angular/material/icon';
import { MatButtonModule } from '@angular/material/button';
import{ RoutingModule } from './routing-module';
@NgModule({
  declarations: [
    AppComponent,
    QuizComponent,
    ResultComponent,
    ReviewComponent,
    WelcomepageComponent
  ],
  imports: [
    BrowserModule,
    ReactiveFormsModule,
    HttpClientModule,
    BrowserModule,
    ReactiveFormsModule,
    MatCardModule,
    MatRadioModule,
    MatIconModule,
    MatButtonModule,
    RoutingModule
  ],
  providers: [ QuizService],
  bootstrap: [AppComponent],
  schemas: [
    CUSTOM_ELEMENTS_SCHEMA,
    NO_ERRORS_SCHEMA
  ]
})
export class AppModule { }
```

routing module.ts

```
 import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { QuizComponent } from './quiz/quiz.component';
import { ResultComponent } from './result/result.component';
import { WelcomepageComponent } from './welcomepage/welcomepage.component';

const appRoutes: Routes = [
{path:'welcome',component:WelcomepageComponent},
{ path:'question', component: QuizComponent },
{ path:'question/:questionId', component: QuizComponent },
{ path:'results', component: ResultComponent },
{path:'**', redirectTo:'welcome'}
];

@NgModule({
    imports:[RouterModule.forRoot(appRoutes)],
exports:[RouterModule]
})
export class RoutingModule { }
```

welcomepage.component.ts

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

@Component({
  selector: 'app-welcomepage',
  templateUrl: './welcomepage.component.html',
  styleUrls: ['./welcomepage.component.scss']
})
export class WelcomepageComponent {

  constructor(private router: Router) {}

  startQuiz() {
    this.router.navigateByUrl('/question/1');
  }


}
```

Quiz.service.ts

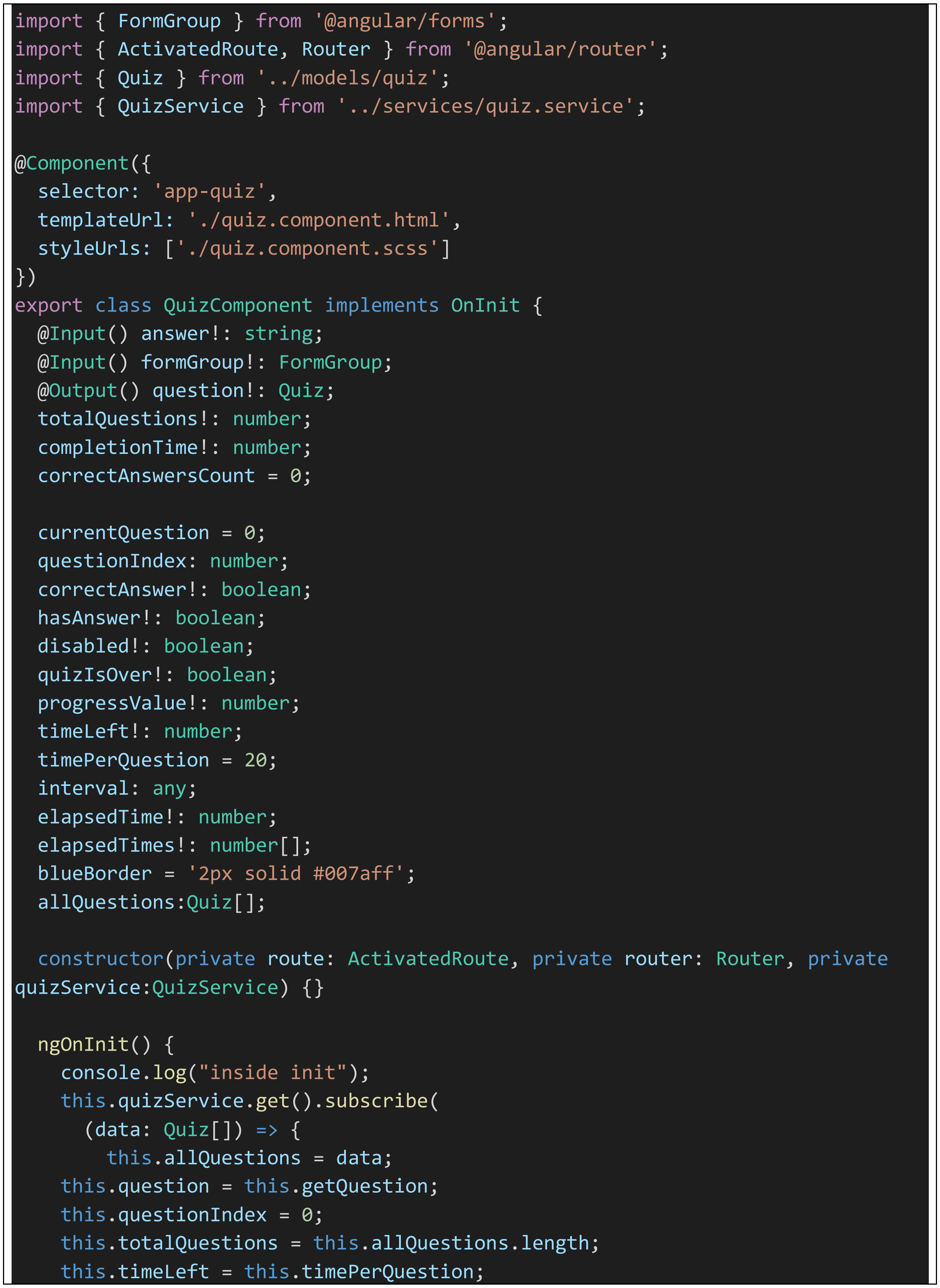
```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class QuizService {

  readonly rootUrl = 'http://localhost:3000/data';
  constructor(private http: HttpClient) { }

  get() {
    return this.http.get(this.rootUrl );
  }

}
```

Result.component.ts

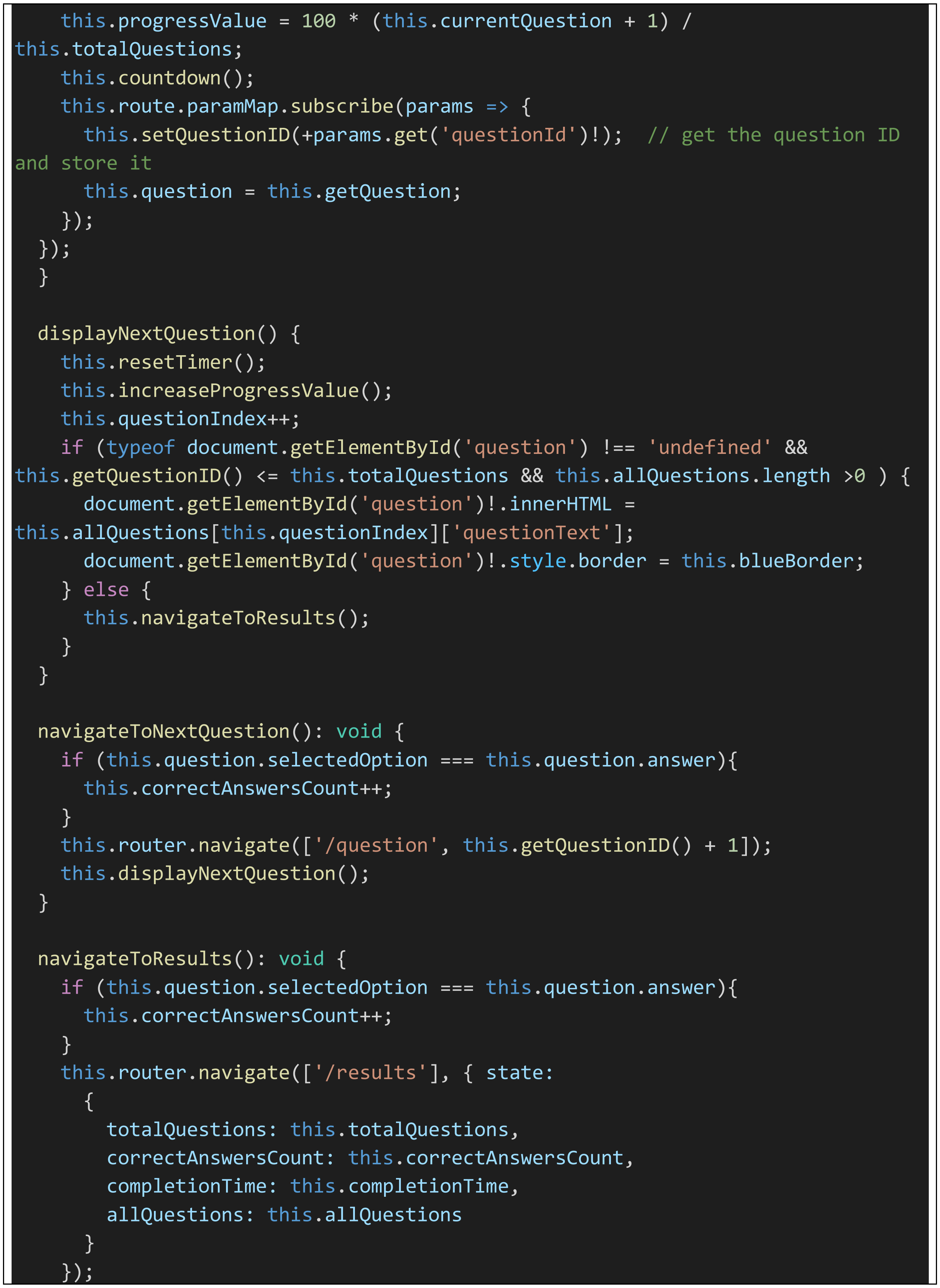
```
import { Component, OnInit, OnChanges, SimpleChanges, Input, Output,
EventEmitter } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { Quiz } from '../models/quiz';

@Component({
  selector: 'app-review',
  templateUrl: './review.component.html',
  styleUrls: ['./review.component.scss']
})
export class ReviewComponent implements OnInit, OnChanges {
  @Output() answer = new EventEmitter<string>();
  @Output() formGroup!: FormGroup;
  @Input() question!: Quiz;
  option = '';
  grayBorder = '2px solid #979797';

  constructor() {}
  ngOnInit() {
    this.buildForm();
  }

  ngOnChanges(changes: SimpleChanges) {
    if (changes.question && changes.question.currentValue &&
!changes.question.firstChange) {
      this.formGroup.patchValue({answer: ''});
    }
  }

  buildForm() {
    this.formGroup = new FormGroup({
      answer: new FormControl(['', Validators.required])
    });
  }

  radioChange(answer: string) {
    this.question.selectedOption = answer;
    this.answer.emit(answer);
  }

  // mark the correct answer regardless of which option is selected once
answered
  isCorrect(option: string): boolean {
    if( this.question.selectedOption && option === this.question.answer)
    return true;
    else
    return false;
  }

  // mark incorrect answer if selected
  isIncorrect(option: string): boolean {
    return option !== this.question.answer && option ===
this.question.selectedOption;
  }

  onSubmit() {
    this.formGroup.reset({answer: null});
  }
}
```

Quiz.component.ts

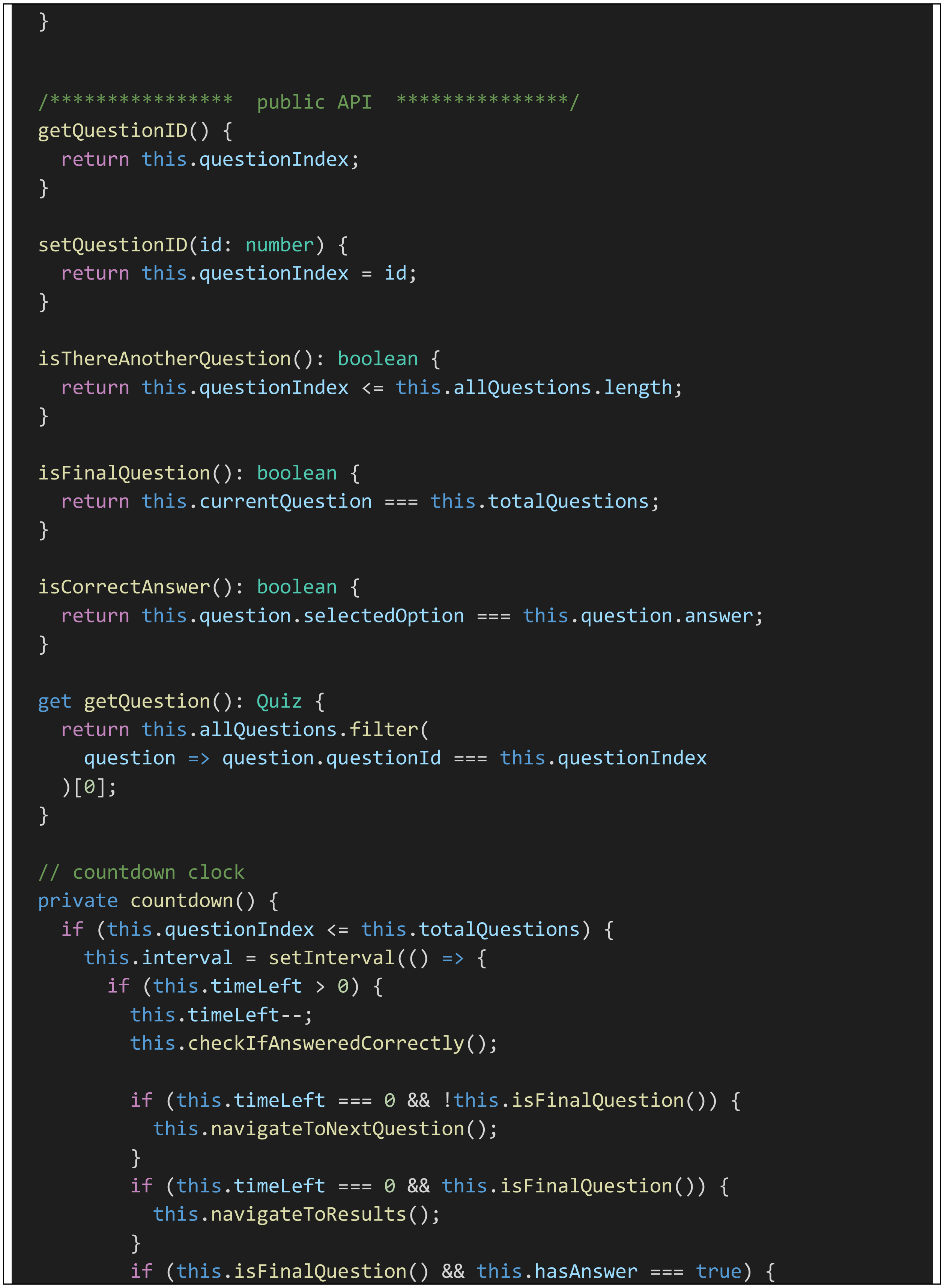
```
import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';
```



```typescript
import { FormGroup } from '@angular/forms';
import { ActivatedRoute, Router } from '@angular/router';
import { Quiz } from '../models/quiz';
import { QuizService } from '../services/quiz.service';

@Component({
  selector: 'app-quiz',
  templateUrl: './quiz.component.html',
  styleUrls: ['./quiz.component.scss']
})
export class QuizComponent implements OnInit {
  @Input() answer!: string;
  @Input() formGroup!: FormGroup;
  @Output() question!: Quiz;
  totalQuestions!: number;
  completionTime!: number;
  correctAnswersCount = 0;

  currentQuestion = 0;
  questionIndex: number;
  correctAnswer!: boolean;
  hasAnswer!: boolean;
  disabled!: boolean;
  quizIsOver!: boolean;
  progressValue!: number;
  timeLeft!: number;
  timePerQuestion = 20;
  interval: any;
  elapsedTime!: number;
  elapsedTimes!: number[];
  blueBorder = '2px solid #007aff';
  allQuestions:Quiz[];

  constructor(private route: ActivatedRoute, private router: Router, private
quizService:QuizService) {}

  ngOnInit() {
    console.log("inside init");
    this.quizService.get().subscribe(
      (data: Quiz[]) => {
        this.allQuestions = data;
    this.question = this.getQuestion;
    this.questionIndex = 0;
    this.totalQuestions = this.allQuestions.length;
    this.timeLeft = this.timePerQuestion;
```



```
    this.progressValue = 100 * (this.currentQuestion + 1) /
this.totalQuestions;
    this.countdown();
    this.route.paramMap.subscribe(params => {
      this.setQuestionID(+params.get('questionId')!);  // get the question ID
and store it
      this.question = this.getQuestion;
    });
  });
  }

  displayNextQuestion() {
    this.resetTimer();
    this.increaseProgressValue();
    this.questionIndex++;
    if (typeof document.getElementById('question') !== 'undefined' &&
this.getQuestionID() <= this.totalQuestions && this.allQuestions.length >0 ) {
      document.getElementById('question')!.innerHTML =
this.allQuestions[this.questionIndex]['questionText'];
      document.getElementById('question')!.style.border = this.blueBorder;
    } else {
      this.navigateToResults();
    }
  }

  navigateToNextQuestion(): void {
    if (this.question.selectedOption === this.question.answer){
      this.correctAnswersCount++;
    }
    this.router.navigate(['/question', this.getQuestionID() + 1]);
    this.displayNextQuestion();
  }

  navigateToResults(): void {
    if (this.question.selectedOption === this.question.answer){
      this.correctAnswersCount++;
    }
    this.router.navigate(['/results'], { state:
      {
        totalQuestions: this.totalQuestions,
        correctAnswersCount: this.correctAnswersCount,
        completionTime: this.completionTime,
        allQuestions: this.allQuestions
      }
    });
  }

  // checks whether the question is valid and is answered correctly
  checkIfAnsweredCorrectly() {
    if (this.isThereAnotherQuestion() && this.isCorrectAnswer()) {
    //  this.incrementCorrectAnswersCount();
      this.correctAnswer = true;
      this.hasAnswer = true;
      this.disabled = false;

      this.elapsedTime = Math.ceil(this.timePerQuestion - this.timeLeft);
      if (this.getQuestionID() < this.totalQuestions) {
        this.elapsedTimes = [...this.elapsedTimes, this.elapsedTime];
      } else {
        this.elapsedTimes = [...this.elapsedTimes, 0];
      }
      this.quizDelay(3000);

      if (this.getQuestionID() < this.totalQuestions) {
        this.navigateToNextQuestion();
      } else {
        this.navigateToResults();
      }
    }
  }

  incrementCorrectAnswersCount() {
    if (this.questionIndex <= this.totalQuestions && this.isCorrectAnswer()) {
      if (this.correctAnswersCount === this.totalQuestions) {
        return this.correctAnswersCount;
      } else {
        this.correctAnswer = true;
        this.hasAnswer = true;
        return this.correctAnswersCount++;
      }
    } else {
      this.correctAnswer = false;
      this.hasAnswer = false;
      return 0;
    }
  }

  increaseProgressValue() {
    this.progressValue = parseFloat((100 * (this.getQuestionID() + 1) /
this.totalQuestions).toFixed(1));
```



```
}


/****************  public API  ***************/
getQuestionID() {
  return this.questionIndex;
}

setQuestionID(id: number) {
  return this.questionIndex = id;
}

isThereAnotherQuestion(): boolean {
  return this.questionIndex <= this.allQuestions.length;
}

isFinalQuestion(): boolean {
  return this.currentQuestion === this.totalQuestions;
}

isCorrectAnswer(): boolean {
  return this.question.selectedOption === this.question.answer;
}

get getQuestion(): Quiz {
  return this.allQuestions.filter(
    question => question.questionId === this.questionIndex
  )[0];
}

// countdown clock
private countdown() {
  if (this.questionIndex <= this.totalQuestions) {
    this.interval = setInterval(() => {
      if (this.timeLeft > 0) {
        this.timeLeft--;
        this.checkIfAnsweredCorrectly();

        if (this.timeLeft === 0 && !this.isFinalQuestion()) {
          this.navigateToNextQuestion();
        }
        if (this.timeLeft === 0 && this.isFinalQuestion()) {
          this.navigateToResults();
        }
        if (this.isFinalQuestion() && this.hasAnswer === true) {
            this.navigateToResults();
            this.quizIsOver = true;
          }

          // disable the next button until an option has been selected
          this.question.selectedOption === '' ? this.disabled = true :
this.disabled = false;
        }
      }, 1000);
    }
  }

  private resetTimer() {
    this.timeLeft = this.timePerQuestion;
  }

  quizDelay(milliseconds:any) {
    const start = new Date().getTime();
    let counter = 0;
    let end = 0;

    while (counter < milliseconds) {
      end = new Date().getTime();
      counter = end - start;
    }
  }
}
```

Quiz.ts

```
import {Option} from './option';

export interface Quiz {
  questionId: number;
  questionText: string;
  options: Option[];
  answer: string;
  explanation: string;
  selectedOption: string;
}
```