Online Quiz Application

e2e

app.e2e-spec.ts

```
import { AppPage } from './app.po';

describe('workspace-project App', () => {
  let page: AppPage;

  beforeEach(() => {
    page = new AppPage();
  });

  it('should display welcome message', () => {
    page.navigateTo();
    expect(page.getParagraphText()).toEqual('Welcome to chat-application!');
  });
});
```

app.po.ts

```
import { browser, by, element } from 'protractor';

export class AppPage {
  navigateTo() {
    return browser.get('/');
  }
  getParagraphText() {
    return element(by.css('app-root h1')).getText();
  }
}
```

Protractor.config.js

```
// Protractor configuration file, see link for more information
// https://github.com/angular/protractor/blob/master/lib/config.ts

const { SpecReporter } = require('jasmine-spec-reporter');

exports.config = {
  allScriptsTimeout: 11000,
  specs: [
    './src/**/*.e2e-spec.ts'
  ],
  capabilities: {
    'browserName': 'chrome'
  },
  directConnect: true,
  baseUrl: 'http://localhost:4200/',
  framework: 'jasmine',
  jasmineNodeOpts: {
    showColors: true,
    defaultTimeoutInterval: 30000,
    print: function() {}
  },
  onPrepare() {
    require('ts-node').register({
      project: require('path').join(__dirname, './tsconfig.e2e.json')
    });
    jasmine.getEnv().addReporter(new SpecReporter({ spec: { displayStacktrace:
true } }));
  }
};
```

Tsconfig.e2e.json

```
{
  "extends": "../tsconfig.json",
  "compilerOptions": {
    "outDir": "../out-tsc/app",
    "module": "commonjs",
    "target": "es5",
    "types": [
      "jasmine",
      "jasminewd2",
      "node"
    ]
  }
}
```

.editorconfig

```
# Editor configuration, see http://editorconfig.org
root = true
```

```
[*]
charset = utf-8
indent_style = space
indent_size = 2
insert_final_newline = true
trim_trailing_whitespace = true

[*.md]
max_line_length = off
trim_trailing_whitespace = false
```

App

app.component.css

```
.label-font{
    color: #116a65;
    font-weight: 700;
}
```

App.component.html

```
<div class="container">
  <h3 style="text-align:center">
    Welcome to Quiz Application &#x263A;
  </h3>
  <div class="card" [hidden]="isQuestionCardShow">
    <div class="card-body">
      <h5 class="card-title" style="text-align:center">Choose Option &#x263A;
</h5>
```

```
      <!-- <p class="card-text">Some quick example text to build on the card title
and make up the bulk of the card's content.</p> -->
      <div class="row justify-content-md-center">
        <div class="col-sm-12 col-lg-3">
          <button class="btn btn-block btn-outline-primary cursor-restriction"
(click)="addQuestion()" style="margin-top: 24px;">
            Add Questions </button>


        </div>
        <div class="col-sm-12 col-lg-3">
          <button class="btn btn-block btn-outline-primary cursor-restriction"
style="margin-top: 24px;" (click)="startQuiz()">
            Start Quiz </button>
        </div>
      </div>
    </div>
  </div>
  <div class="card" [hidden]="!isQuestionCardShow">
    <div class="card-body">
      <form #questionTest="ngForm" novalidate>
        <div *ngFor="let ques of allQuestions">
          <div>{{ques.question}}</div>
          <div class="form-group row">
            <label class="col-md-3 col-form-label">Choose Correct Option</label>
            <div class="col-md-9 col-form-label">
              <div class="form-check form-check-inline mr-1">
                <input class="form-check-input" type="radio" name="{{ques.id}}"
value="a" [(ngModel)]="ques.selected">
                <label class="form-check-label">a) {{ques.a}} </label>
              </div>
              <div class="form-check form-check-inline mr-1">
                <input class="form-check-input" type="radio" name="{{ques.id}}"
value="b" [(ngModel)]="ques.selected">
                <label class="form-check-label">b){{ques.b}}</label>
              </div>
              <div class="form-check form-check-inline mr-1">
                <input class="form-check-input" type="radio" name="{{ques.id}}"
value="c" [(ngModel)]="ques.selected">
                <label class="form-check-label">c){{ques.c}}</label>
              </div>
              <div class="form-check form-check-inline mr-1">
                <input class="form-check-input" type="radio" name="{{ques.id}}"
value="d" [(ngModel)]="ques.selected">
                <label class="form-check-label">d){{ques.d}}</label>
              </div>
            </div>
          </div>
          <hr>
        </div>
      </form>
      <div class="row">
        <div class="col-12" style="justify-content: center;display: flex;">
          <button type="button" class="btn btn-default" (click)="HomePage()"
style="margin-right: 30px;">Cancel</button>
          <button type="button" class="btn btn-primary"
(click)="submitTest()">Submit</button>
        </div>
      </div>
```

```
    </div>
  </div>
</div>
<div bsModal #submitModal="bs-modal" class="modal fade"
[config]="{backdrop: 'static'}" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel"
  aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-body text-center" style="color: #196f77;">
        <h4>Total Questions: {{allQuestions.length}} </h4>
        <h4>
          Total Answered: {{totalAnswered}}
        </h4>
        <h4>
          Right Answer : {{rightAnswer}}
        </h4>
        <span>
          <button type="button" class="btn btn-primary" (click)="checkAnswers()"
style="margin-right: 17px;"> Answers</button>
        </span>
        <button type="button" class="btn btn-default"
(click)="submitModal.hide()" style="margin-left: 17px;">Close</button>
      </div>
    </div>
    <!-- /.modal-content -->
  </div>
  <!-- /.modal-dialog -->
</div>
```

```
<!-- /.modal -->

<div bsModal #addQuestionModal="bs-modal" class="modal fade"
[config]="{backdrop: 'static'}" tabindex="-1" role="dialog"

  aria-labelledby="myModalLabel" aria-hidden="true">

  <div class="modal-dialog modal-lg" role="document">

    <div class="modal-content">

      <div class="modal-body text-center" style="color: #196f77;">

        <form #questionForm="ngForm"  novalidate>

          <div class="row">

            <div class="col-sm-12 col-12 col-lg-12">

              <div class="form-group">

                <label for="question" class="label-font"> Add Question</label>

                <span style="color:red;" [hidden]="!(question?.invalid)">*</span>

                <input type="text" class="form-control" name="question"
placeholder="Enter Question" [(ngModel)]="questionObj.question"
[ngClass]="{

                      'is-invalid': question?.invalid && (question?.dirty
||question?.touched),

                      'is-valid': question?.valid && (question?.dirty ||
question?.touched)

                 }" #question="ngModel" required>

                <div class="invalid-feedback" *ngIf="question.errors &&
(question.dirty || question.touched)">

                  <p *ngIf="question.errors.required">Question Description is
required</p>

                </div>

              </div>

            </div>

          </div>

          <div class="row">
            <div class="col-sm-12 col-12 col-lg-6">
              <div class="form-group">
                <label for="a">Option a.</label>
                <span style="color:red;" [hidden]="!(a?.invalid)">*</span>
                <input type="text" class="form-control" name="a"
placeholder="Enter First Option" [(ngModel)]="questionObj.a" [ngClass]="{
                'is-invalid': a?.invalid && (a?.dirty ||a?.touched),
                'is-valid': a?.valid && (a?.dirty || a?.touched)
              }" #a="ngModel" required>
                <div class="invalid-feedback" *ngIf="a.errors && (a.dirty ||
a.touched)">
                  <p *ngIf="a.errors.required">Option is required</p>
                </div>
              </div>
            </div>
            <div class="col-sm-12 col-12 col-lg-6">
              <div class="form-group">
                <label for="b">Option b.</label>
                <span style="color:red;" [hidden]="!(b?.invalid)">*</span>
                <input type="text" class="form-control" name="b"
placeholder="Enter Second Option" [(ngModel)]="questionObj.b" [ngClass]="{
              'is-invalid': b?.invalid && (b?.dirty ||b?.touched),
              'is-valid': b?.valid && (b?.dirty || b?.touched)
                  }" #b="ngModel" required>
                <div class="invalid-feedback" *ngIf="b.errors && (b.dirty ||
b.touched)">
                  <p *ngIf="b.errors.required">Option is required</p>
                </div>
              </div>
```

```
          </div>
        </div>
        <div class="row">
          <div class="col-sm-12 col-12 col-lg-6">
            <div class="form-group">
              <label for="c">Option c.</label>
              <span style="color:red;" [hidden]="!(c?.invalid)">*</span>
              <input type="text" class="form-control" name="c"
placeholder="Enter Third Option" [(ngModel)]="questionObj.c" [ngClass]="{
                              'is-invalid': c?.invalid && (c?.dirty ||c?.touched),
                              'is-valid': c?.valid && (c?.dirty || c?.touched)
                          }" #c="ngModel" required>
              <div class="invalid-feedback" *ngIf="c.errors && (c.dirty ||
c.touched)">
                <p *ngIf="c.errors.required">Option is required</p>
              </div>
            </div>
          </div>
          <div class="col-sm-12 col-12 col-lg-6">
            <div class="form-group">
              <label for="b">Option d.</label>
              <span style="color:red;" [hidden]="!(d?.invalid)">*</span>
              <input type="text" class="form-control" name="d"
placeholder="Enter fourth Option" [(ngModel)]="questionObj.d" [ngClass]="{
                            'is-invalid': d?.invalid && (d?.dirty ||d?.touched),
                            'is-valid': d?.valid && (d?.dirty || d?.touched)
                        }" #d="ngModel" required>
              <div class="invalid-feedback" *ngIf="d.errors && (d.dirty ||
d.touched)">
```

```
              <p *ngIf="d.errors.required">Option is required</p>
             </div>
           </div>
         </div>
       </div>

       <div class="form-group row">
         <label class="col-md-12 label-font">Choose Correct Answer Option
           <span style="color:red;" [hidden]="!(answer?.invalid)">*</span>
         </label>
         <div class="col-md-12">
           <div class="form-check form-check-inline mr-2">
             <input class="form-check-input" type="radio" name="answer"
value="a" [(ngModel)]="questionObj.answer" #answer="ngModel" required>
             <label class="form-check-label">a.</label>
           </div>
           <div class="form-check form-check-inline mr-2">
             <input class="form-check-input" type="radio" name="answer"
value="b" [(ngModel)]="questionObj.answer" #answer="ngModel" required>
             <label class="form-check-label">b.</label>
           </div>
           <div class="form-check form-check-inline mr-2">
             <input class="form-check-input" type="radio" name="answer"
value="c" [(ngModel)]="questionObj.answer" #answer="ngModel" required>
             <label class="form-check-label">c.</label>
           </div>
           <div class="form-check form-check-inline mr-2">
             <input class="form-check-input" type="radio" name="answer"
value="d" [(ngModel)]="questionObj.answer" #answer="ngModel" required>
```

```
                <label class="form-check-label">d.</label>

              </div>

            </div>

          </div>

        </form>

        <span>

          <button type="button" class="btn btn-primary"
(click)="submitAddQuestion()" [disabled]="questionForm.invalid"
[class.cursor-restriction]="questionForm.invalid"

            style="margin-right: 17px;">ADD Question</button>

        </span>

        <button type="button" class="btn btn-default"
(click)="addQuestionModal.hide()">Close</button>

      </div>

    </div>

    <!-- /.modal-content -->

  </div>

  <!-- /.modal-dialog -->

</div>

<!-- /.modal -->

<div bsModal #answerModal="bs-modal" class="modal fade"
[config]="{backdrop: 'static'}" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel"

  aria-hidden="true">

  <div class="modal-dialog" role="document">

    <div class="modal-content">

      <div class="modal-body text-center" style="color: #196f77;">

        <div *ngFor="let ques of allQuestions" [style.color]="!ques.selected?
'#ffc107' : ques.selected == ques.answer ? 'green': 'red'">
```

```
        <div>{{ques.question}}</div>

        <div>Your Answer: {{ques[ques.selected]}}</div>

        <div>Right Answer: {{ques.answer}}) {{ques[ques.answer]}}</div>

        <hr>

       </div>

       <button type="button" class="btn btn-default"
(click)="answerModal.hide()" style="margin-left: 17px;">Close</button>

      </div>

    </div>

    <!-- /.modal-content -->

  </div>

  <!-- /.modal-dialog -->

</div>

<!-- /.modal -->
```

App.component.spec.ts

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

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

describe('AppComponent', () => {

  beforeEach(async(() => {

    TestBed.configureTestingModule({

      declarations: [

        AppComponent

      ],

    }).compileComponents();

  }));

  it('should create the app', async(() => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app).toBeTruthy();
  }));
  it(`should have as title 'app'`, async(() => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app.title).toEqual('app');
  }));
  it('should render title in a h1 tag', async(() => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.debugElement.nativeElement;
    expect(compiled.querySelector('h1').textContent).toContain('Welcome to
chat-application!');
  }));
});
```

App.module.ts

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


import { AppComponent } from './app.component';
import { AccordionModule } from 'ngx-bootstrap';
import { FormsModule } from '@angular/forms';
import { ModalModule } from 'ngx-bootstrap/modal';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';
```

```
import { ToastrModule } from 'ngx-toastr';




@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    ModalModule.forRoot(),
    BrowserAnimationsModule,
    ToastrModule.forRoot(),
  AccordionModule.forRoot(),
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Angular Quiz</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Test.ts

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

import 'zone.js/dist/zone-testing';
import { getTestBed } from '@angular/core/testing';
import {
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting
} from '@angular/platform-browser-dynamic/testing';

declare const require: any;

// First, initialize the Angular testing environment.
getTestBed().initTestEnvironment(
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting()
);
```

```
// Then we find all the tests.
const context = require.context('./', true, /\.spec\.ts$/);
// And load the modules.
context.keys().map(context);
```

Angular.json

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "chat-application": {
      "root": "",
      "sourceRoot": "src",
      "projectType": "application",
      "prefix": "app",
      "schematics": {},
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "src/tsconfig.app.json",
            "assets": [
             "src/favicon.ico",
             "src/assets"
            ],
            "styles": [
             "src/styles.css",
             "./node_modules/bootstrap/dist/css/bootstrap.css",
             "node_modules/ngx-toastr/toastr.css"
            ],
            "scripts": []
          },
          "configurations": {
            "production": {
             "fileReplacements": [
              {
               "replace": "src/environments/environment.ts",
               "with": "src/environments/environment.prod.ts"
              }
             ],
             "optimization": true,
             "outputHashing": "all",
             "sourceMap": false,
             "extractCss": true,
             "namedChunks": false,
             "aot": true,
             "extractLicenses": true,
                 "vendorChunk": false,
                 "buildOptimizer": true
               }
             }
          },
          "serve": {
             "builder": "@angular-devkit/build-angular:dev-server",
             "options": {
               "browserTarget": "chat-application:build"
             },
             "configurations": {
               "production": {
                 "browserTarget": "chat-application:build:production"
               }
             }
          },
          "extract-i18n": {
             "builder": "@angular-devkit/build-angular:extract-i18n",
             "options": {
               "browserTarget": "chat-application:build"
             }
          },
          "test": {
             "builder": "@angular-devkit/build-angular:karma",
             "options": {
               "main": "src/test.ts",
                 "polyfills": "src/polyfills.ts",
                 "tsConfig": "src/tsconfig.spec.json",
                 "karmaConfig": "src/karma.conf.js",
                 "styles": [
                   "src/styles.css"
                 ],
                 "scripts": [],
                 "assets": [
                   "src/favicon.ico",
                   "src/assets"
                 ]
               }
             },
             "lint": {
               "builder": "@angular-devkit/build-angular:tslint",
               "options": {
                 "tsConfig": [
                   "src/tsconfig.app.json",
                   "src/tsconfig.spec.json"
                 ],
                 "exclude": [
                   "**/node_modules/**"
                 ]
               }
             }
          }
```

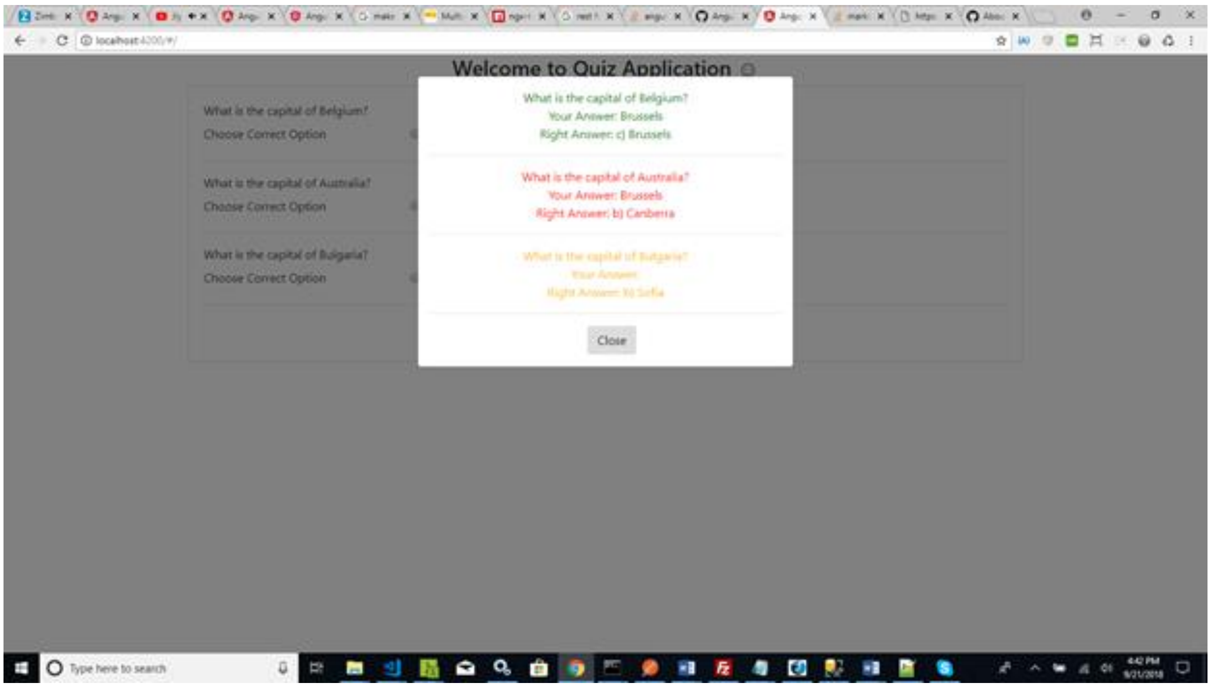
```
},
"chat-application-e2e": {
 "root": "e2e/",
 "projectType": "application",
 "architect": {
  "e2e": {
   "builder": "@angular-devkit/build-angular:protractor",
   "options": {
    "protractorConfig": "e2e/protractor.conf.js",
    "devServerTarget": "chat-application:serve"
   },
   "configurations": {
    "production": {
     "devServerTarget": "chat-application:serve:production"
    }
   }
  },
  "lint": {
   "builder": "@angular-devkit/build-angular:tslint",
   "options": {
    "tsConfig": "e2e/tsconfig.e2e.json",
    "exclude": [
     "**/node_modules/**"
    ]
   }
  }
```

```
      }
    }
  },
  "defaultProject": "chat-application"
}
```

Package.json

```
{
  "name": "angular-quiz-application",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^6.1.8",
    "@angular/common": "^6.0.3",
    "@angular/compiler": "^6.0.3",
    "@angular/core": "^6.0.3",
    "@angular/forms": "^6.0.3",
    "@angular/http": "^6.0.3",
  "@angular/platform-browser": "^6.0.3",
  "@angular/platform-browser-dynamic": "^6.0.3",
  "@angular/router": "^6.0.3",
  "bootstrap": "^4.1.3",
  "core-js": "^2.5.4",
  "ngx-bootstrap": "^3.0.1",
  "ngx-toastr": "^9.1.0",
  "rxjs": "^6.0.0",
  "zone.js": "^0.8.26"
 },
 "devDependencies": {
  "@angular/compiler-cli": "^6.0.3",
  "@angular-devkit/build-angular": "~0.6.8",
  "typescript": "~2.7.2",
  "@angular/cli": "~6.0.8",
  "@angular/language-service": "^6.0.3",
  "@types/jasmine": "~2.8.6",
  "@types/jasminewd2": "~2.0.3",
  "@types/node": "~8.9.4",
  "codelyzer": "~4.2.1",
  "jasmine-core": "~2.99.1",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~1.7.1",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "~2.0.0",
  "karma-jasmine": "~1.1.1",
    "karma-jasmine-html-reporter": "^0.2.2",
    "protractor": "~5.3.0",
    "ts-node": "~5.0.1",
    "tslint": "~5.9.1"
  }
}
```

Screenshot Output



Welcome to Quiz Application

What is the capital of Belgium?
Choose Correct Option

What is the capital of Australia?
Choose Correct Option

What is the capital of Bulgaria?
Choose Correct Option

What is the capital of Belgium?
Your Answer: Brussels
Right Answer: c) Brussels

What is the capital of Australia?
Your Answer: Brussels
Right Answer: b) Canberra

What is the capital of Bulgaria?
Your Answer:
Right Answer: b) Sofia