Angular.json

{

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

  "version": 1,

  "newProjectRoot": "projects",

  "projects": {

    "angular-app": {

      "projectType": "application",

      "schematics": {},

      "root": "",

      "sourceRoot": "src",

      "prefix": "app",

      "architect": {

        "build": {

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

          "options": {

            "outputPath": "dist/angular-app",

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

            "main": "src/main.ts",

            "polyfills": [

              "zone.js"

            ],

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

            "assets": [

              "src/favicon.ico",

              "src/assets"

            ],

            "styles": [

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

              "@angular/material/prebuilt-themes/indigo-pink.css",

              "src/styles.css"

            ],

            "scripts": []

          },

          "configurations": {

            "production": {

              "budgets": [

                {

                  "type": "initial",

                  "maximumWarning": "500kb",

                  "maximumError": "1mb"

                },

                {

                  "type": "anyComponentStyle",

                  "maximumWarning": "2kb",

                  "maximumError": "4kb"

                }

              ],

              "outputHashing": "all"

            },

            "development": {

              "buildOptimizer": false,

              "optimization": false,

              "vendorChunk": true,

              "extractLicenses": false,

              "sourceMap": true,

              "namedChunks": true

            }

          },

          "defaultConfiguration": "production"

        },

        "serve": {

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

          "configurations": {

            "production": {

              "browserTarget": "angular-app:build:production"

            },

            "development": {

              "browserTarget": "angular-app:build:development"

            }

          },

          "defaultConfiguration": "development"

        },

        "extract-i18n": {

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

          "options": {

            "browserTarget": "angular-app:build"

          }

        },

        "test": {

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

          "options": {

            "polyfills": [

              "zone.js",

              "zone.js/testing"

            ],

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

            "assets": [

              "src/favicon.ico",

              "src/assets"

            ],

            "styles": [

              "@angular/material/prebuilt-themes/indigo-pink.css",

              "src/styles.css"

            ],

            "scripts": []

          }

        }

      }

    }

  },

  "cli": {

    "analytics": "0fce5559-9dcf-4d10-94bf-fcbdc7534491"

  }

}

App-routing.module.ts:

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

import { RouterModule, Routes } from '@angular/router';

import { NotfoundComponent } from './components/notfound/notfound.component';

import { QuestionComponent } from './components/question/question.component';

import { ResultComponent } from './components/result/result.component';

import { TestsComponent } from './components/tests/tests.component';

const routes: Routes = [

  {path:'',component:TestsComponent},

  {path:'question',component:QuestionComponent},

  {path:'result',component:ResultComponent},

  {path:'\*\*',component:NotfoundComponent}

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

**Models:**

**Question.model.ts**

import { AnswerOption } from "./answer-option.model";

import {Deserializable} from "./deserializable.model";

export class Question implements Deserializable {

    textOfQuestion: string | undefined;

    answerOptions: AnswerOption[] | undefined;

    deserialize(input: any) {

        Object.assign(this, input);

        this.answerOptions = input.answerOptions.map((answerOption: any) => new AnswerOption().deserialize(answerOption));

        return this;

      }

    constructor (textOfQuestion?: string, answerOptions?: AnswerOption[]) {

        this.textOfQuestion = textOfQuestion;

        this.answerOptions = answerOptions;

    }

}

**Deserializable.model.ts**

export interface Deserializable {

    deserialize(input: any): this;

}

**Answer.model.ts**

import { Deserializable } from "./deserializable.model";

export class Answer implements Deserializable{

    questionText:string|undefined;

    answerText:string|undefined;

    isCorrect:boolean|undefined;

    deserialize(input: any) {

        Object.assign(this, input);

        return this;

      }

    constructor(questionText?:string, answerText?:string, isCorrect?:boolean){

        this.questionText=questionText;

        this.answerText=answerText;

        this.isCorrect=isCorrect;

    }

}

**Answer-option.model.ts**

import { Deserializable } from "./deserializable.model";

export class AnswerOption implements Deserializable{

    textOfAnswer:string|undefined;

    isCorrect:boolean|undefined;

    deserialize(input: any) {

        Object.assign(this, input);

        return this;

      }

    constructor(textOfAnswer?:string, isCorrect?:boolean){

        this.textOfAnswer=textOfAnswer;

        this.isCorrect=isCorrect;

    }

}

**Components:**

**Notfound**

**Notfound.component.ts**

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

@Component({

  selector: 'app-notfound',

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

  styleUrls: ['./notfound.component.css']

})

export class NotfoundComponent {

}

**Notfound.component.html**

<p>notfound works!</p>

**Question**

**Question.component.ts**

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

import { Router } from '@angular/router';

import { Answer } from 'src/app/models/answer.model';

import { Question } from 'src/app/models/question.model';

import { ComponentExchangeService } from 'src/app/services/component-exchange.service';

import { RecieveTestService } from 'src/app/services/recieve-test.service';

import { ResultsService } from 'src/app/services/results.service';

@Component({

  selector: 'app-question',

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

  styleUrls: ['./question.component.css']

})

export class QuestionComponent implements OnInit{

  counter:number=0;

  numberOfQuestion:number=0;

  questions:Question[]=[];

  selectedAnswerOption:Answer|any;

  currentQuestion:Question | any;

  constructor(private recieveTestService:RecieveTestService

    ,private router: Router

    ,private resultsService:ResultsService

    ,private componentExchangeService:ComponentExchangeService

    ){

  }

  ngOnInit(): void {

    this.questions=this.componentExchangeService.getCurrentTest();

    this.currentQuestion=this.questions[0];

    this.numberOfQuestion=this.questions.length;

  }

  onSubmit():void{

    this.resultsService.addAnswer(new Answer(

      this.currentQuestion.textOfQuestion,

      this.selectedAnswerOption.textOfAnswer,

      this.selectedAnswerOption.isCorrect)).subscribe(

        data => {

          console.log('message::::', data);

          if(this.counter<this.numberOfQuestion-1){

            this.counter++;

            this.currentQuestion=this.questions[this.counter];

            this.selectedAnswerOption=null;

          }else

          {

            this.router.navigateByUrl('result');

          }

        }

      );

  }

 }

**Question.component.html**

<br>

<section >

    <form #f="ngForm" (ngSubmit)="onSubmit()" class="container">

      <label id="example-radio-group-label">

        <h2>{{currentQuestion.textOfQuestion}}</h2>

      </label>

      <mat-radio-group aria-labelledby="example-radio-group-label"

        class="example-radio-group" [(ngModel)]="selectedAnswerOption" name="selectedAnswerOption">

        <mat-radio-button class="example-radio-button"

          \*ngFor="let answerOption of currentQuestion.answerOptions" [value]="answerOption">

                {{ answerOption.textOfAnswer }}

        </mat-radio-button>

      </mat-radio-group>

      <button [disabled]="!selectedAnswerOption" type="submit" class="btn btn-primary">Next</button>

    </form>

</section>

**Question.component.css**

.example-radio-group {

    display: flex; flex-direction: column;

    /\* display: inline-grid; \*/

    /\* display: grid; \*/

    margin: 15px 0;

  }

**Result**

**Result.component.ts**

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

import { Answer } from 'src/app/models/answer.model';

import { ResultsService } from 'src/app/services/results.service';

@Component({

  selector: 'app-result',

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

  styleUrls: ['./result.component.css']

})

export class ResultComponent implements OnInit{

  result:Answer[]=[];

  resultScore:string="";

  ngOnInit(): void {

    this.resultsService.getAnswers().subscribe((data)=>{

     this.result=data;

     this.resultScore=this.resultsService.getResulScore(data);

    });

  }

  constructor(private resultsService:ResultsService){

  }

}

**Result.component.html**

<br>

<div class="container">

    <h2 class="container">Table of results:</h2>

    <div [ngClass]="{bar:true,

        success:answer?.isCorrect,

        error:!answer?.isCorrect}"

        \*ngFor="let answer of result;  let i = index">

        {{i+1}}. {{answer?.questionText}}  {{answer?.answerText}}

    </div>

    <br>

    <h2 class="container">Your result score is {{resultScore}}</h2>

</div>

**Result.component.css**

/\* (A) THE BASE \*/

.bar {

    padding: 10px;

    margin: 10px;

    color: #333;

    background: #fafafa;

    border: 1px solid #ccc;

  }

  /\* (B) THE VARIATIONS \*/

  .info {

    color: #204a8e;

    background: #c9ddff;

    border: 1px solid #4c699b;

  }

  .success {

    color: #2b7515;

    background: #ecffd6;

    border: 1px solid #617c42;

  }

  .warn {

    color: #756e15;

    background: #fffbd1;

    border: 1px solid #87803e;

  }

  .error {

    color: #ba3939;

    background: #ffe0e0;

    border: 1px solid #a33a3a;

  }

**Tests**

**Tests.component.ts**

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

import { Router } from '@angular/router';

import { ComponentExchangeService } from 'src/app/services/component-exchange.service';

import { RecieveTestService } from 'src/app/services/recieve-test.service';

@Component({

  selector: 'tests',

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

  styleUrls: ['./tests.component.css']

})

export class TestsComponent{

constructor(private recieveTestService:RecieveTestService

  ,private router: Router

  ,private componentExchangeService:ComponentExchangeService

  ){

}

onSubmit():void{

  {

    this.recieveTestService.getQuestions().subscribe((data)=>{

      this.componentExchangeService.setCurrentTest(data);

      this.router.navigateByUrl('question');

    })

  }

}

}

**Tests.component.html**

<br>

<div class="container">

  <h2>Press button below to start amazing QUIZ</h2>

  <br>

  <button (click)="onSubmit()" class="btn btn-primary">Start QUIZ</button>

</div>

**Services:**

**Component-exchange.service.ts**

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

import { Question } from '../models/question.model';

@Injectable({

  providedIn: 'root'

})

export class ComponentExchangeService {

  currentTest:Question[]=[];

  public setCurrentTest(test: Question[]) {

    this.currentTest=test;

    //console.log(JSON.stringify(test));

  }

  public getCurrentTest():Question[] {

    return this.currentTest;

  }

}

**Receive-test.service.ts**

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

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

import { Observable } from 'rxjs';

import { map } from 'rxjs/operators';

import { Question } from '../models/question.model';

@Injectable({

  providedIn: 'root'

})

export class RecieveTestService {

  private questions:Question[]=[];

public getQuestions(): Observable<Question[]> {

  return this.http.get<Question[]>(`http://localhost:3002/test`).pipe(

    map(data => data.map(data => new Question().deserialize(data)))

  );

}

  constructor(private http: HttpClient) { }

}

**Result.service.ts**

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

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

import { map, Observable } from 'rxjs';

import { Answer } from '../models/answer.model';

@Injectable({

  providedIn: 'root'

})

export class ResultsService {

  addAnswer(answer:Answer){

    return this.http.post(`http://localhost:3002/addAnswer`, answer)

  }

  getAnswers():Observable<Answer[]>{

    return this.http.get<Answer[]>(`http://localhost:3002/answers`).pipe(

      map(data => data.map(data => new Answer().deserialize(data)))

    );

  }

  getResulScore(answers:Answer[]):string{

    let correctNumber=0;

    answers.forEach(element => {

        if(element?.isCorrect) correctNumber++;

    });

    return correctNumber+"/"+answers.length;

  }

  constructor(private http: HttpClient) { }

}

**App.module.ts**

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

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

import { AppRoutingModule } from './app-routing.module';

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

import { TestsComponent } from './components/tests/tests.component';

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

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

import { MatFormFieldModule } from '@angular/material/form-field';

import { MatInputModule } from '@angular/material/input';

import { MatRadioModule } from '@angular/material/radio';

import { MatIconModule } from '@angular/material/icon';

import { ResultComponent } from './components/result/result.component';

import { NotfoundComponent } from './components/notfound/notfound.component';

import { QuestionComponent } from './components/question/question.component';

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

@NgModule({

  declarations: [

    AppComponent,

    TestsComponent,

    ResultComponent,

    NotfoundComponent,

    QuestionComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    BrowserAnimationsModule,

    HttpClientModule,

    FormsModule,

    ReactiveFormsModule,

    MatFormFieldModule,

    MatInputModule,

    MatRadioModule,

    MatIconModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

**Backend.js**

var express = require('express');

var cors = require('cors');

var bodyParser = require('body-parser');

var app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({

  extended: true

}));

app.use(cors());

var answers=[];

var test = [

    {

        "textOfQuestion":"What is result for 2+2?",

        "answerOptions":[

            {"textOfAnswer":"5","isCorrect":false},

            {"textOfAnswer":"4","isCorrect":true},

            {"textOfAnswer":"3","isCorrect":false},

            {"textOfAnswer":"7","isCorrect":false}

        ]

    },

        {"textOfQuestion":"What is the colour of cat Richard?",

        "answerOptions":[

            {"textOfAnswer":"black","isCorrect":false},

            {"textOfAnswer":"green","isCorrect":false},

            {"textOfAnswer":"gray","isCorrect":true},

            {"textOfAnswer":"red","isCorrect":false},

            {"textOfAnswer":"yellow","isCorrect":false}

        ]

    },

        {"textOfQuestion":"How old is me?",

        "answerOptions":[

            {"textOfAnswer":"42","isCorrect":true},

            {"textOfAnswer":"67","isCorrect":false},

            {"textOfAnswer":"32","isCorrect":false},

            {"textOfAnswer":"12","isCorrect":false},

            {"textOfAnswer":"57","isCorrect":false}

        ]

    }

];

app.get('/test', function (req, res) {

   res.send(test);

});

app.get('/answers', function (req, res) {

  res.send(answers);

});

app.post('/addAnswer', function (req, res) {

  console.log(req.body.questionText)

  answers.push({

      questionText: req.body.questionText,

      answerText: req.body.answerText,

      isCorrect:req.body.isCorrect

  });

  console.log(req.body.answerText);

  res.json("answer addedd");

});

var server = app.listen(3002, function () {

  console.log('backend started');

});