**ANGULAR WITH RXJS AND MATERIAL QUIZ APP**

**Part 1: Setup**

**1.1 Create an Angular Application That Uses SCSS and Install Angular Material**

**Try it**:  
Run this in your project:

ng add @angular/material

Choose:

* Indigo/Pink theme
* Yes to typography and animations

**1.2 Confirm Angular Material styles**

**Try it**:  
Open angular.json and check for:

"styles": [

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

"src/styles.scss"

]

**Part 2: Backend & Data Model**

**2.1 Create db.json**

**Try it**:  
Create db.json in your project root:

{

"questions": [

{

"id": 1,

"question": "What is Angular?",

"options": ["A framework", "A library", "A database", "A design system"],

"answer": "A framework"

},

{

"id": 2,

"question": "What is RxJS?",

"options": ["A styling tool", "A reactive programming library", "A routing module", "A linter"],

"answer": "A reactive programming library"

}

]

}

Then run:

json-server --watch db.json --port 3000

Visit <http://localhost:3000/questions>, you should see your questions.

**2.2 Create the QuizService**

**Try it**:  
Create src/app/services/quiz.service.ts.

**Then compare your solution**:

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

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

import { Observable } from 'rxjs';

export interface Question {

id: number;

question: string;

options: string[];

answer: string;

}

@Injectable({ providedIn: 'root' })

export class QuizService {

private apiUrl = 'http://localhost:3000/questions';

private score = 0;

constructor(private http: HttpClient) {}

getQuestions(): Observable<Question[]> {

return this.http.get<Question[]>(this.apiUrl);

}

getScore(): number {

return this.score;

}

setScore(score: number): void {

this.score = score;

}

}

**Part 3: State Management with BehaviorSubject**

**3.1 Set up the component and state**

**Try it**:  
Create quiz.component.ts with this structure:

* questions$ = new BehaviorSubject<Question[]>([])
* currentQuestionIndex$ = new BehaviorSubject<number>(0)
* selectedAnswer$ = new BehaviorSubject<string>('')
* score$ = new BehaviorSubject<number>(0)

**Full Component So Far:**

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

import { QuizService, Question } from '../../services/quiz.service';

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

import { BehaviorSubject, combineLatest } from 'rxjs';

import { map } from 'rxjs/operators';

import { NgIf, NgFor, AsyncPipe } from '@angular/common';

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

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

import { MatButtonModule } from '@angular/material/button';

@Component({

standalone: true,

selector: 'app-quiz',

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

styleUrls: ['./quiz.component.scss'],

imports: [

NgIf,

NgFor,

AsyncPipe,

RouterModule,

MatRadioModule,

MatButtonModule

]

})

export class QuizComponent implements OnInit {

private questions$ = new BehaviorSubject<Question[]>([]);

private currentQuestionIndex$ = new BehaviorSubject<number>(0);

public selectedAnswer$ = new BehaviorSubject<string>('');

private score$ = new BehaviorSubject<number>(0);

currentQuestion$ = combineLatest([

this.questions$,

this.currentQuestionIndex$

]).pipe(

map(([questions, index]) => questions[index])

);

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

ngOnInit(): void {

this.quizService.getQuestions().subscribe(questions => {

this.questions$.next(questions);

});

}

selectAnswer(option: string) {

this.selectedAnswer$.next(option);

}

nextQuestion() {

const questions = this.questions$.value;

const index = this.currentQuestionIndex$.value;

const selected = this.selectedAnswer$.value;

if (selected === questions[index]?.answer) {

this.score$.next(this.score$.value + 1);

}

this.selectedAnswer$.next('');

const nextIndex = index + 1;

if (nextIndex >= questions.length) {

this.quizService.setScore(this.score$.value);

this.router.navigate(['/result']);

} else {

this.currentQuestionIndex$.next(nextIndex);

}

}

}

**Part 4: Quiz UI with Angular Material**

**4.1 Build the HTML**

**Try it**:

* Use \*ngIf="currentQuestion$ | async as question"
* Loop through question.options with mat-radio-button
* Disable "Next" until something is selected

**quiz.component.html**

<div \*ngIf="currentQuestion$ | async as question">

<h2>{{ question.question }}</h2>

<mat-radio-group

[value]="selectedAnswer$ | async"

(change)="selectAnswer($event.value)">

<mat-radio-button

\*ngFor="let option of question.options"

[value]="option">

{{ option }}

</mat-radio-button>

</mat-radio-group>

<button

mat-raised-button

color="primary"

(click)="nextQuestion()"

[disabled]="!(selectedAnswer$ | async)">

Next

</button>

</div>

**Part 5: Show Result**

**5.1 Create ResultComponent**

**Try it**:

* Use combineLatest to combine score and question count
* Use \*ngIf="result$ | async as result" in the template

**//result.component.ts**

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

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

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

import { Observable, combineLatest, of } from 'rxjs';

import { map } from 'rxjs/operators';

import { AsyncPipe, NgIf } from '@angular/common';

@Component({

standalone: true,

selector: 'app-result',

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

styleUrls: ['./result.component.scss'],

imports: [RouterModule, AsyncPipe, NgIf]

})

export class ResultComponent implements OnInit {

result$!: Observable<{ score: number; total: number }>;

constructor(private quizService: QuizService) {}

ngOnInit(): void {

const score$ = of(this.quizService.getScore());

const totalQuestions$ = this.quizService.getQuestions().pipe(

map(q => q.length)

);

this.result$ = combineLatest([score$, totalQuestions$]).pipe(

map(([score, total]) => ({ score, total }))

);

}

}

**result.component.html**

<div \*ngIf="result$ | async as result">

<h2>Quiz Result</h2>

<p>Your Score: {{ result.score }}</p>

<p>Total Questions: {{ result.total }}</p>

<a routerLink="/quiz">Try Again</a>

</div>

**Part 6: Routing and Bootstrapping**

**6.1 Define routes**

**//app.routes.ts**

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

import { QuizComponent } from './pages/quiz/quiz.component';

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

export const routes: Routes = [

{ path: '', redirectTo: 'quiz', pathMatch: 'full' },

{ path: 'quiz', component: QuizComponent },

{ path: 'result', component: ResultComponent }

];

**6.2 Setup config and bootstrap**

**//app.config.ts**

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

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

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

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

import { routes } from './app.routes';

export const appConfig: ApplicationConfig = {

providers: [provideHttpClient(), provideRouter(routes), provideAnimations()]

};

**//main.ts**

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

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

import { appConfig } from './app/app.config';

bootstrapApplication(AppComponent, appConfig)

.catch(err => console.error(err));

**//app.component.ts**

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

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

@Component({

selector: 'app-root',

standalone: true,

imports: [RouterOutlet],

template: `<router-outlet></router-outlet>`

})

export class AppComponent {}

**Run the App**

ng serve

[http://localhost:4200](http://localhost:4200/)

(Take the quiz!)