**MILESTONE ASSESSMENT- 5**

**Ecommerce-Stationary**

**Angular**

**App-module**

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

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

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

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

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

import { BooksComponent } from './books/books.component';

import { PensComponent } from './pens/pens.component';

import { PaperComponent } from './paper/paper.component';

import { IndexComponent } from './books/index/index.component';

import { ViewComponent } from './books/view/view.component';

import { BaseComponent } from './base/base.component';

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

import {MatGridListModule} from '@angular/material/grid-list';

import { BookserviceService } from './services/bookservice.service';

@NgModule({

declarations: [

AppComponent,

BooksComponent,

PensComponent,

PaperComponent,

IndexComponent,

ViewComponent,

BaseComponent

],

imports: [

BrowserModule,

AppRoutingModule,

BrowserAnimationsModule,

MatGridListModule,

HttpClientModule

],

providers: [BookserviceService],

bootstrap: [AppComponent]

})

export class AppModule { }

**App-component.html**

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<a class="navbar-brand" href="#">Stationary</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a routerLink = 'base' class="nav-link" href="#">Check Store</a>

</li>

</ul>

</div>

</nav>

<router-outlet></router-outlet>

<!-- <li class="nav-item active">

<a href="#">Home <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item active">

<a href="#">Check Store</a>

</li> -->

**Index.html**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Ms5</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

<link rel="preconnect" href="https://fonts.gstatic.com">

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap" rel="stylesheet">

<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

</head>

<body class="mat-typography">

<div class="container">

<app-root></app-root>

</div>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

</body>

</html>

**App-routing-module**

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

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

import{BooksComponent} from '../app/books/books.component';

import { IndexComponent } from './books/index/index.component';

import { ViewComponent } from './books/view/view.component';

import { PensComponent } from './pens/pens.component';

import { PaperComponent } from './paper/paper.component';

import { BaseComponent } from './base/base.component';

const routes: Routes = [

{path:'books',component:IndexComponent},

{path:'book/:id/view',component:ViewComponent},

{path:'paper',component:PaperComponent},

{path:'pen',component:PensComponent},

{path:'base',component:BaseComponent}

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

**Book-service**

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

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

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class BookserviceService {

url:string = "<https://localhost:44341/api>"

constructor(

private client:HttpClient,

) { }

getAll():Observable<any>{

return this.client.get(this.url+"/books");

}

find(id:number): Observable<any> {

return this.client.get(this.url + '/books/' + id)

}

}

**Book-Model**

export interface books{

id:number;

name:string;

discription:string;

imgLink:string;

price:number;

}

**Index-component.ts**

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

import { BookserviceService } from 'src/app/services/bookservice.service';

import { books } from 'src/app/model/books';

@Component({

selector: 'app-index',

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

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

})

export class IndexComponent implements OnInit {

books:books[] = []

constructor(

private serviece:BookserviceService

) { }

ngOnInit(): void {

this.serviece.getAll().subscribe(data=>{

this.books = data;

console.log(this.books);

})

}

}

**Index-component.html**

<mat-grid-list cols="3" rowHeight="2:1">

<mat-grid-tile \*ngFor="let item of books">

<div class="card" style="width: 18rem;">

<img class="card-img-top" src={{item.imgLink}} alt="Card image cap">

<div class="card-body">

<h5 class="card-titleard">{{item.name}}</h5>

<p class="card-text">{{item.discription}}</p>

<a href="#" [routerLink] = "['/book/',item.id,'view']" class="btn btn-primary">View More</a>

</div>

</div>

</mat-grid-tile>

</mat-grid-list>

**View-component.ts**

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

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

import { BookserviceService } from 'src/app/services/bookservice.service';

import { books } from 'src/app/model/books';

@Component({

selector: 'app-view',

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

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

})

export class ViewComponent implements OnInit {

id!:number;

book!:books;

constructor(

public service: BookserviceService,

private route: ActivatedRoute,

private router: Router

) { }

ngOnInit(): void {

this.id = this.route.snapshot.params['id'];

this.service.find(this.id).subscribe((data: books)=>{

this.book = data;

console.log(this.book)

})

}

}

**View-Component.html**

<div class="jumbotron">

<h1 class="display-4">{{book.name}}</h1>

<p class="lead">Id: {{book.id}}</p>

<p class="lead">Discription: {{book.discription}}</p>

<p class="lead">Price: {{book.price}}</p>

</div>

**Base-component.ts**

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

@Component({

selector: 'app-base',

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

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

})

export class BaseComponent implements OnInit {

constructor() { }

ngOnInit(): void {

}

}

**Base-component.html**

<div class="d-flex justify-content-center">

<h1 >

Stationary

</h1>

</div>

<div class="row">

<div class="col">

<div class="card" style="width: 18rem;">

<img class="card-img-top" src="data:image/jpeg;base64," alt="book">

<div class="card-body">

<h5 class="card-title">Books</h5>

<p class="card-text">Different types of books.</p>

<a routerLink = '/books' href="#" class="btn btn-primary">Go To Books</a>

</div>

</div>

</div>

<div class="col">

<div class="card" style="width: 18rem;">

<img class="card-img-top" src="data:image/jpeg;base64," alt="Card image cap">

<div class="card-body">

<h5 class="card-title">Pens</h5>

<p class="card-text">Different types of Pens.</p>

<a routerLink = '/pen' href="#" class="btn btn-primary">Go To Pens</a>

</div>

</div>

</div>

<div class="col">

<div class="card" style="width: 18rem;">

<img class="card-img-top" src="data:image/jpeg;base64," alt="Card image cap">

<div class="card-body">

<h5 class="card-title">Paper</h5>

<p class="card-text">Different types of Papers.</p>

<a routerLink = '/paper' href="#" class="btn btn-primary">Go To Papers</a>

</div>

</div>

</div>

</div>

<!-- <router-outlet></router-outlet> -->

**STATIONARY API**

**Startup.cs**

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using Microsoft.OpenApi.Models;

using StationaryApi.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace StationaryApi

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddCors(options =>

{

options.AddDefaultPolicy(

builder =>

{

builder.WithOrigins("http://localhost:4200").AllowAnyHeader()

.AllowAnyMethod();

});

});

services.AddControllers();

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "StationaryApi", Version = "v1" });

});

services.AddDbContext<BooksDbContext>(o => o.UseSqlServer(

Configuration.GetConnectionString("booksConnectionString")));

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseSwagger();

app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "StationaryApi v1"));

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseCors();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

**Appsetting.json**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"booksConnectionString": "server=.;database=books;integrated security=true"

}

}

**BooksController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using StationaryApi.Models;

namespace StationaryApi.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class BooksController : ControllerBase

{

private readonly BooksDbContext \_context;

public BooksController(BooksDbContext context)

{

\_context = context;

}

// GET: api/Books

[HttpGet]

public async Task<ActionResult<IEnumerable<Books>>> GetBooks()

{

return await \_context.Books.ToListAsync();

}

// GET: api/Books/5

[HttpGet("{id}")]

public async Task<ActionResult<Books>> GetBooks(int id)

{

var books = await \_context.Books.FindAsync(id);

if (books == null)

{

return NotFound();

}

return books;

}

// PUT: api/Books/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutBooks(int id, Books books)

{

if (id != books.Id)

{

return BadRequest();

}

\_context.Entry(books).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!BooksExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Books

[HttpPost]

public async Task<ActionResult<Books>> PostBooks(Books books)

{

\_context.Books.Add(books);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetBooks", new { id = books.Id }, books);

}

// DELETE: api/Books/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteBooks(int id)

{

var books = await \_context.Books.FindAsync(id);

if (books == null)

{

return NotFound();

}

\_context.Books.Remove(books);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool BooksExists(int id)

{

return \_context.Books.Any(e => e.Id == id);

}

}

}

**BooksModel.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace StationaryApi.Models

{

public class Books

{

public int Id { get; set; }

public string Name { get; set; }

public Double Price { get; set; }

public string Discription { get; set; }

public string ImgUrl { get; set; }

}

}

**BooksDbContext.cs**

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace StationaryApi.Models

{

public class BooksDbContext:DbContext

{

public DbSet<Books> Books { get; set; }

public BooksDbContext(DbContextOptions<BooksDbContext> o) : base(o) { }

}

}