Contents

Angular 2

Page adjust

```
<div class="flex-container" fxLayout.xs="column">
    <div class="flex-item">1</div>
    <div class="flex-item">2</div>
    <div class="flex-item">3</div>
    <div class="flex-item">4</div>
    <div class="flex-item">5</div>
    <div class="flex-item">6</div>
</div>
```

Css

```
.flex-container{
    display:flex;
    flex-flow: row wrap;

    justify-content: space-around;
}

@media all and (max-width: 800px){
    .flex-container{
        justify-content: flex-start;
    }
}

.flex-item {
    width: 200px;
    height: 150px;
    margin-top: 5px;
    background: rgb(71, 99, 255);
    color: white;
    font-weight: bold;
    font-size: 3em;
    text-align: center;
```

```
    line-height: 150px;
}
```

Filtering html

```
<div class="mat-elevation-z8">
    <mat-form-field appearance="fill">
        <mat-label>Filter</mat-label>
        <input matInput (keyup)="applyFilter($event)"
placeholder="start typing..." #input>
    </mat-form-field>

    <table mat-table [dataSource]="dataSource"
matSort>

      <ng-container matColumnDef="id">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> No. </th>
        <td mat-cell *matCellDef="let element">
{{element.id}} </td>
      </ng-container>

      <ng-container matColumnDef="name">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> Name </th>
        <td mat-cell *matCellDef="let element">
{{element.name}} </td>
      </ng-container>

      <ng-container matColumnDef="image">
        <th mat-header-cell *matHeaderCellDef></th>
        <td mat-cell *matCellDef="let element">  <img
src={{element.image}}>  </td>
      </ng-container>

      <ng-container matColumnDef="detailsbutton">
```

```
        <th mat-header-cell *matHeaderCellDef mat-
sort-header>  </th>
        <td mat-cell *matCellDef="let element">
<button mat-button [routerLink]="['/hero',
element.id]">
            <mat-icon>face</mat-icon>
            Details
          </button>
        </td>
      </ng-container>

      <ng-container matColumnDef="deletebutton">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header>  </th>
        <td mat-cell *matCellDef="let element">
<button mat-button class="deletebutton"
(click)="deleteHero(element.id)">
            <mat-icon>delete</mat-icon>
          </button>
        </td>
      </ng-container>
      <tr mat-header-row
*matHeaderRowDef="displayedColumns"></tr>
      <tr mat-row *matRowDef="let row; columns:
displayedColumns;"></tr>

      <tr class="mat-row" *matNoDataRow>
        <td class="mat-cell" colspan="4">No data
matching the filter "{{input.value}}"</td>
      </tr>
    </table>
    <mat-paginator [pageSize]="10"
[pageSizeOptions]="[3, 5, 10]" showFirstLastButtons>
</mat-paginator>
```

```
  </div>
```

Css

```
table {
    width: 100%;
}

.mat-form-field {
    font-size: 14px;
    width: 100%;
}

img {
    max-width: 12em;
    max-height: 12em;
    margin: 0.75em;
  }
```

Filtering.ts

```
import { AfterViewInit, Component, OnInit, ViewChild }
from '@angular/core';
import { MatPaginator } from
'@angular/material/paginator';
import { MatSnackBar, MatSnackBarRef } from
'@angular/material/snack-bar';
import { MatSort } from '@angular/material/sort';
import { MatTableDataSource } from
'@angular/material/table';
import { Router } from '@angular/router';
import { HeroService } from
'../services/hero.service';
import { Hero } from '../shared/hero';
```

```
@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.scss']
})
export class HeroesComponent implements AfterViewInit,
OnInit {
  displayedColumns: string[] = ['id', 'name', 'image',
'detailsbutton', 'deletebutton'];
  dataSource = new MatTableDataSource<Hero>();
  constructor(private heroService: HeroService,
private snackBar: MatSnackBar) { }


  @ViewChild(MatPaginator) paginator!: MatPaginator;
  @ViewChild(MatSort) sort!: MatSort;

  ngAfterViewInit() {
    this.dataSource.paginator = this.paginator;
    this.dataSource.sort = this.sort;
  }

  ngOnInit(): void {
    this.heroService.getHeroes().subscribe((heroes:any
) => {this.dataSource.data = heroes})
  }

  applyFilter(event: Event) {
    const filterValue = (event.target as
HTMLInputElement).value;
    this.dataSource.filter =
filterValue.trim().toLowerCase();
  }
```

```
  async deleteHero(id: any){
    await this.heroService.deleteHero(id)
    this.showSnackBar()
  }

  showSnackBar() {
    const snackBarRef: MatSnackBarRef<any> =
this.snackBar.open('Deleted successfully', 'X', {
duration: 500 });
    snackBarRef.afterDismissed().subscribe(() => {
      location.reload();
    });
  }
}
```

Angular Data service

```
import { Injectable } from '@angular/core';
import { Observable, of} from 'rxjs';
import { Hero } from '../shared/hero';

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

  constructor() {
    if(!localStorage.getItem('heroes')) {
      let heroes = [{
        "id": 1,
        "name": "Tony Stark (Iron Man)",
        "age": 53,
        "birthday": "May 29",
        "height": "185cm",
```

```
        "image": "assets/images/tony-stark-iron-
man.webp",
        "alive": false,
      },
      {
        "id": 2,
        "name": "Steve Rogers (Captain America)",
        "age": 34,
        "birthday": "July 4",
        "height": "185cm",
        "image": "assets/images/steve-rogers.webp",
        "alive": false,
      },
      {
        "id": 3,
        "name": "Bruce Banner (The Hulk)",
        "age": 54,
        "birthday": "December 18",
        "height": "250cm",
        "image": "assets/images/The-Incredible-
Hulk.webp",
        "alive": true,
      },
      {
        "id": 4,
        "name": "Thor",
        "age": 1059,
        "birthday": null,
        "height": "192cm",
        "image": "assets/images/thor-lightning.webp",
        "alive": true,
      },
      {
        "id": 5,
```

```
        "name": "Natasha Romanoff (Black Widow)",
        "age": 39,
        "birthday": "December 3",
        "height": "164cm",
        "image": "assets/images/black-widow-1.webp",
        "alive": true,
      },
      {
        "id": 6,
        "name": "Peter Parker (Spider-Man)",
        "age": 19,
        "birthday": "August 10",
        "height": "170cm",
        "image": "assets/images/peter-parker-
Cropped.webp",
        "alive": true,
      },
      {
        "id": 7,
        "name": "Clint Barton (Hawkeye)",
        "age": 53,
        "birthday": "June 18",
        "height": "173cm",
        "image": "assets/images/hawkeye.webp",
        "alive": true,
      },
      {
        "id": 8,
        "name": "Colonel James 'Rhodey' Rhodes (War
Machine)",
        "age": 55,
        "birthday": "October 6",
        "height": "173cm",
```

```
        "image": "assets/images/Don-cheadle-as-rhodey-
Cropped.webp",
        "alive": true,
      },
      {
        "id": 9,
        "name": "Samuel Thomas 'Sam' Wilson
(Falcon/Captain America)",
        "age": 40,
        "birthday": "September 23",
        "height": "178cm",
        "image": "assets/images/Anthony-Mackie-
Captain-America-4.webp",
        "alive": true,
      },
      {
        "id": 10,
        "name": "Wanda Maximoff (Scarlet Witch)",
        "age": 30,
        "birthday": "February 10",
        "height": "168cm",
        "image": "assets/images/Wanda-Scarlet-Witch-
Cropped.webp",
        "alive": true,
      },
      {
        "id": 11,
        "name": "Vision",
        "age": 3,
        "birthday": "May 29",
        "height": "May",
        "image": "assets/images/Vision-Civil-War-
Cropped.webp",
        "alive": true,
```

```
      },
      {
        "id": 12,
        "name": "Scott Lang (Ant-Man)",
        "age": null,
        "birthday": null,
        "height": "178cm",
        "image": "assets/images/antman-and-the-wasp-
marvel-4.webp",
        "alive": true,
      }
      ]
      localStorage.setItem('heroes',
JSON.stringify(heroes))
    }
   }


  getHeroes(): Observable<any[]> {
    let heroes:any[]=[]
    if (localStorage.getItem('heroes'))
    {
      heroes =
JSON.parse(localStorage.getItem('heroes')!)
    }
    return of(heroes)
  }

  getHero(id:number): Observable<any>
  {
    let heroes:Hero[]  = [];

    if (localStorage.getItem('heroes'))
    {
```

```
      heroes =
JSON.parse(localStorage.getItem('heroes')!)
    }

    let hero:any = heroes.find(hero => hero.id === id)

    return of(hero)
  }

  async deleteHero(id: any){
    let heroes:Hero[] = []
    if (localStorage.getItem('heroes'))
    {
      heroes =
JSON.parse(localStorage.getItem('heroes')!)
    }

    let hero = heroes.find(hero => hero.id === id)

    if (hero)
    {
      let index = heroes.indexOf(hero)
      heroes.splice(index, 1)
      await localStorage.setItem('heroes',
JSON.stringify(heroes))
    }
  }
}
```

API 2

Base API functions

Packages needed:

- MS.EFCore.SQLServer
- MS.EFCore.Tools
- Swashbuckle.AspNetCore

Backend

API Controller (CRUD

```
using APIII.Models;
using APIII.ViewModel;
using Microsoft.AspNetCore.Components.Forms;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Reflection.Metadata.Ecma335;

namespace APIII.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CustomerController : ControllerBase
    {
        private readonly IRepository _repository;

        public CustomerController(IRepository repository)
        {
            _repository = repository;
        }

        [HttpGet]
        [Route("GetAllCustomers")]
        public async Task<IActionResult> GetAllCustomers()
        {
            try
            {
                var results = await _repository.GetAllCustomersAsync();
                return Ok(results);
            }
            catch (Exception)
            {
                return StatusCode(500,"Internal Server Error. Please contact
support.");
            }
        }

        [HttpGet]
        [Route("GetCustomer/{custId}")]
        public async Task<IActionResult> GetCustomerAsync(int custId)
        {
            try
            {
                var result = await _repository.GetCustomerAsync(custId);

                if (result == null) return NotFound("Customer does not exist");

                return Ok(result);
            }
```

```
            catch (Exception)
            {
                return StatusCode(500, "Internal Server Error. Please contact
support");
            }
        }

        [HttpPost]
        [Route("AddCustomer")]
        public async Task<IActionResult> AddCustomer(CustomerViewModel cvm)
        {
            var customer = new Customer { LastName = cvm.LastName, FirstName =
cvm.FirstName, Address = cvm.Address, City = cvm.City, State = cvm.State,
PostalCode = cvm.PostalCode, PhoneNumber = cvm.PhoneNumber };

            try
            {
                _repository.Add(customer);
                await _repository.SaveChangesAsync();
            }
            catch (Exception)
            {
                return BadRequest("Invalid transaction");
            }

            return Ok(customer);
        }

        [HttpPut]
        [Route("EditCustomer/{custId}")]
        public async Task<ActionResult<CustomerViewModel>> EditCustomer(int
custId, CustomerViewModel customerModel)
        {
            try
            {
                var existingCustomer = await
_repository.GetCustomerAsync(custId);
                if (existingCustomer == null) return NotFound($"The customer does
not exist");

                existingCustomer.LastName = customerModel.LastName;
                existingCustomer.FirstName = customerModel.FirstName;
                existingCustomer.Address = customerModel.Address;
                existingCustomer.City = customerModel.City;
                existingCustomer.State = customerModel.State;
                existingCustomer.PostalCode = customerModel.PostalCode;
                existingCustomer.PhoneNumber = customerModel.PhoneNumber;

                if (await _repository.SaveChangesAsync())
                {
                    return Ok(existingCustomer);
                }
            }
            catch (Exception)
            {
                return StatusCode(500, "Internal Server Error. Please contact
support.");
            }
            return BadRequest("Your request is invalid.");
        }

        [HttpDelete]
        [Route("DeleteCustomer/{custId}")]
```

```
        public async Task<IActionResult> DeleteCustomer(int custId)
        {
            try
            {
                var existingCustomer = await
_repository.GetCustomerAsync(custId);

                if (existingCustomer == null) return NotFound($"The customer does
not exist");

                _repository.Delete(existingCustomer);

                if (await _repository.SaveChangesAsync()) return
Ok(existingCustomer);

            }
            catch (Exception)
            {
                return StatusCode(500, "Internal Server Error. Please contact
support.");
            }
            return BadRequest("Your request is invalid.");
        }


    }
}
```

API DBContext

```
using Microsoft.EntityFrameworkCore;

namespace APIII.Models
{
    public class AppDbContext:DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options): base
(options)
        {
        }
        public DbSet<Guide> Guides { get; set; }
        public DbSet<Trip> Trips { get; set; }
        public DbSet<Customer> Customers { get; set; }

        // 1To1 Example (Uncomment code below and run migration to generate
tables)
        //public DbSet<TableTwo1to1Ex> TableTwo1to1Ex { get; set; }
        //public DbSet<TableOne1to1Ex> TableOne1to1Ex { get; set; }

        // 1ToM Example (Uncomment code below and run migration to generate
tables)
        //public DbSet<TableOne1toManyEx> TableOne1toManyEx { get; set; }
        //public DbSet<TableTwo1toManyEx> TableTwo1toManyEx { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            // For the M2M payload (Uncomment code below and run migration to
generate tables)
            //modelBuilder.Entity<Trip2>()
```

```
            //    .HasMany(t => t.Guides2)
            //    .WithMany(g => g.Trips2)
            //    .UsingEntity<TripGuide2>
            //     (tg => tg.HasOne<Guide2>().WithMany(),
            //      tg => tg.HasOne<Trip2>().WithMany())
            //     .Property(tg => tg.DateConfirmed)
            //     .HasDefaultValueSql("getdate()");
        }
    }
}
```

Customer.cs

```
using System.ComponentModel.DataAnnotations;

namespace APIII.Models
{
    public class Customer
    {
        [Key]
        public int CustId { get; set; }

        [MaxLength(50)]
        public string LastName { get; set; } = string.Empty;

        [MaxLength(50)]
        public string FirstName { get; set; } = string.Empty;

        public string? Address { get; set; }
        public string? City { get; set; }

        [StringLength(2)]
        public string? State { get; set; }

        [StringLength(5)]
        public string? PostalCode { get; set;}

        [StringLength(10)]
        public string PhoneNumber { get; set; } = string.Empty;
    }
}
```

API IRepository

```
namespace APIII.Models
{
    public interface IRepository
    {
        void Add<T>(T entity) where T : class;
        void Delete<T>(T entity) where T : class;
        Task<bool> SaveChangesAsync();
        // Customer
        Task<Customer[]> GetAllCustomersAsync();
        Task<Customer> GetCustomerAsync(int custId);

        // Trip
        Task<Trip> GetTripAsync(int tripId);

        // Guide
        Task<Guide> GetGuideAsync(string guideNum);

    }
}
```

API Repository

```
using Microsoft.EntityFrameworkCore;

namespace APIII.Models
{
    public class Repository : IRepository
    {
        private readonly AppDbContext _appDbContext;

        public Repository(AppDbContext appDbContext)
        {
                _appDbContext = appDbContext;
        }

        public void Add<T>(T entity) where T : class
        {
            _appDbContext.Add(entity);
        }

        public void Delete<T>(T entity) where T : class
        {
            _appDbContext.Remove(entity);
        }
        public async Task<Customer[]> GetAllCustomersAsync()
        {
            IQueryable<Customer> query = _appDbContext.Customers;
            return await query.ToArrayAsync();
        }
        public async Task<Customer> GetCustomerAsync(int custId)
        {
            IQueryable<Customer> query = _appDbContext.Customers.Where(c =>
c.CustId == custId);
            return await query.FirstOrDefaultAsync();
        }

        public async Task<Guide> GetGuideAsync(string guideNum)
        {
            IQueryable<Guide> query = _appDbContext.Guides.Where(c => c.GuideNum
== guideNum);
            return await query.FirstOrDefaultAsync();
        }

        public async Task<Trip> GetTripAsync(int tripId)
        {
            IQueryable<Trip> query = _appDbContext.Trips.Include(g =>
g.Guides).Where(c => c.TripId == tripId);
            return await query.FirstOrDefaultAsync();
        }

        public async Task<bool> SaveChangesAsync()
        {
            return await _appDbContext.SaveChangesAsync() > 0;
        }
    }
}
```

CustomerViewModel

```
using System.ComponentModel.DataAnnotations;

namespace APIII.ViewModel
{
```

```
    public class CustomerViewModel
    {
        public string LastName { get; set; } = string.Empty;

        public string FirstName { get; set; } = string.Empty;
        public string? Address { get; set; }
        public string? City { get; set; }
        public string? State { get; set; }
        public string? PostalCode { get; set; }
        public string PhoneNumber { get; set; } = string.Empty;
    }
}
```

Api Program.cs

```
using APIII.Models;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddCors(options => options.AddDefaultPolicy(
                include =>
                {
                    include.AllowAnyHeader();
                    include.AllowAnyMethod();
                    include.AllowAnyOrigin();
                }));

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection
")));
builder.Services.AddScoped<IRepository, Repository>();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseCors();
app.UseAuthorization();
app.UseAuthentication();

app.MapControllers();

app.Run();
```

API appsettings.json

```
{
  "Logging": {
```

```
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection":
"Server=.;Database=APIII;Trusted_Connection=True;MultipleActiveResultSets=True"
  }
}
```

Ionic 2

Heroes program …

Packages needed:

- MS.EFCore.SQLServer
- MS.EFCore.Tools
- Swashbuckle.AspNetCore

Backend

Ionic Controller

```
using Ionic_II.Models;
using Microsoft.AspNetCore.Components.Forms;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Reflection.Metadata.Ecma335;

namespace Ionic_II.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class HeroController : ControllerBase
    {
        private readonly IHeroRepository _heroRepository;

        public HeroController(IHeroRepository HeroRepository)
        {
            _heroRepository = HeroRepository;
        }

        [HttpGet]
        [Route("GetAllHeroes")]
        public async Task<IActionResult> GetAllHeroes()
        {
            try
            {
                var results = await _heroRepository.GetAllHeroesAsync();
                return Ok(results);
            }
            catch (Exception)
            {
                return StatusCode(500,"Internal Server Error. Please contact
support.");
            }
        }

        [HttpGet]
        [Route("GetHero/{heroId}")]
        public async Task<IActionResult> GetHero(int heroId)
        {
            try
            {
                var result = await _heroRepository.GetHeroAsync(heroId);

                if (result == null) return NotFound("Hero does not exist. You
need to create it first");

                return Ok(result);
            }
            catch (Exception)
            {
                return StatusCode(500, "Internal Server Error. Please contact
support");
            }
        }
    }
}
```

Ionic DBContext

```
using Microsoft.EntityFrameworkCore;

namespace Ionic_II.Models
{
    public class AppDbContext:DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options): base
(options)
        {
        }

        public DbSet<Hero> Heroes { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            // Hero

            modelBuilder.Entity<Hero>()
                .HasData(
                new
                {
                    HeroId = 1,
                    Name = "Samuel Thomas 'Sam' Wilson (Falcon/Captain America)",
                    Age = 40,
                    Birthday = "September 23",
                    Height = "178cm",
                    isAlive = true,
                    FileName = "Anthony-Mackie-Captain-America-4.webp",
                    ImageBase64 = "Image”

}
            );

            modelBuilder.Entity<Hero>()
                .HasData(
                new
                {
                    HeroId = 2,
                    Name = "Scott Lang (Ant-Man)",
                    Age = 34,
                    Birthday = "",
                    Height = "178cm",
                    isAlive = true,
                    FileName = "antman-and-the-wasp-marvel-4.webp",
                    ImageBase64 = "Image”

                }
      }
}
```

Ionic Repository

```
using Microsoft.EntityFrameworkCore;

namespace Ionic_II.Models
{
    public class HeroRepository : IHeroRepository
    {
        private readonly AppDbContext _appDbContext;
```

```
        public HeroRepository(AppDbContext appDbContext)
        {
                _appDbContext = appDbContext;
        }
        public async Task<Hero[]> GetAllHeroesAsync()
        {
            IQueryable<Hero> query = _appDbContext.Heroes;
            return await query.ToArrayAsync();
        }

        public async Task<Hero> GetHeroAsync(int heroId)
        {
            IQueryable<Hero> query = _appDbContext.Heroes.Where(c => c.HeroId ==
heroId);
            return await query.FirstOrDefaultAsync();
        }
    }
}
```

Ionic IRepository

```
namespace Ionic_II.Models
{
    public interface IHeroRepository
    {
        // Hero
        Task<Hero[]> GetAllHeroesAsync();

        Task<Hero> GetHeroAsync(int heroId);

    }
}
```

Ionic appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection":
"Server=LAPTOPJOE\\SQLEXPRESS01;Database=Ionic_II;Trusted_Connection=True;Multipl
eActiveResultSets=True"
  }
}
```

Ionic program.cs

```
using Ionic_II.Models;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddCors(options => options.AddDefaultPolicy(
```

```
                include =>
                {
                    include.AllowAnyHeader();
                    include.AllowAnyMethod();
                    include.AllowAnyOrigin();
                }));

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection
")));
builder.Services.AddScoped<IHeroRepository, HeroRepository>();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseCors();
app.UseAuthorization();
app.UseAuthentication();

app.MapControllers();

app.Run();
```

Hero.cs

```
using System.ComponentModel.DataAnnotations;

namespace Ionic_II.Models
{
    public class Hero
    {
        [Key]
        public int HeroId { get; set; }
        public string Name { get; set; }
        public int Age { get; set; }
        public string Birthday { get; set; }
        public string Height { get; set; }
        public bool isAlive { get; set; }
        public string FileName { get; set; }
        public string ImageBase64 { get; set; }

    }
}
```

Frontend

Hero-detail (Shows hero details)

```
<ion-header [translucent]="true">
  <ion-toolbar>
```

```
    <ion-buttons slot="start">
      <ion-back-button></ion-back-button>
    </ion-buttons>
    <ion-title></ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-img src={{heroDetail?.imageBase64}}></ion-img>
  <ion-card no-margin>
    <ion-card-header>
    <ion-card-title>
      {{heroDetail?.name}}
    </ion-card-title>
  </ion-card-header>
  </ion-card>

  <ion-card>
    <ion-list lines="none">

      <ion-item>
        <ion-label>Age</ion-label>
        <ion-chip
color="primary">{{heroDetail?.age}}</ion-chip>
      </ion-item>

      <ion-item>
        <ion-label>Height</ion-label>
        <ion-chip
color="secondary">{{heroDetail?.height}}</ion-chip>
      </ion-item>

      <ion-item>
        <ion-label>Birthday</ion-label>
```

```
        <ion-chip>{{heroDetail?.birthday}}</ion-chip>
      </ion-item>

    </ion-list>
  </ion-card>

  <ion-fab vertical="top" horizontal="end"
slot="fixed">
    <ion-fab-button
(click)="openModal(heroDetail?.isAlive)">
      <ion-icon name="eye"></ion-icon>
    </ion-fab-button>
  </ion-fab>
</ion-content>
```

Hero-details.ts

```
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { IonicModule, ModalController, ToastController
} from '@ionic/angular';
import { HeroService } from
'../services/hero.service';
import { ActivatedRoute } from '@angular/router';
import { HerostatusPage } from
'../herostatus/herostatus.page';

@Component({
  selector: 'app-hero-detail',
  templateUrl: './hero-detail.page.html',
  styleUrls: ['./hero-detail.page.scss'],
  standalone: true,
  imports: [IonicModule, CommonModule, FormsModule]
})
```

```
export class HeroDetailPage implements OnInit {
  heroDetail:any
  constructor(private _toastController:
ToastController, private _heroService: HeroService,
private _modal:ModalController, private
route:ActivatedRoute) {

    this._heroService.getHero(+this.route.snapshot.par
ams['heroId']).subscribe(result => {

      this.heroDetail = result
      const toast = this._toastController.create({
        message: "Hero " + this.heroDetail.name + " is
viewable",
        duration: 3000,
        position:"bottom"
      })

      toast.then((toastMessage) => {
        toastMessage.present();
      })
    })
  }

  ngOnInit():void {  }

  async openModal(status:boolean)
  {
    const statusModal = await this._modal.create({
      component: HerostatusPage,
      componentProps:{
        value:status
      }
    })
```

```
    return await statusModal.present()
  }
}
```

Hero.ts (model folder for heroes)

```
export interface Hero {
    heroId: Number;
    name:String;
    age:number;
    birthday:String;
    height:string;
    isAlive:boolean;
    filename:String;
    imageBase64:String;
}
```

Heroestab1.html

```
<ion-header [translucent]="true">
  <ion-toolbar>
    <ion-title>
      Home
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div *ngIf="!Heroes">
  <ion-card>
    <ion-skeleton-text style="height:200px;" animated></ion-
skeleton-text>
    <ion-card-header></ion-card-header>
  </ion-card>
</div>
```

```
<ion-refresher slot="fixed" (ionRefresh)="refreshHeroes($event)">
<ion-refresher-content refreshingText="Loading Heroes..."></ion-
refresher-content>
</ion-refresher>

  <ion-card button *ngFor="let hero of (Heroes | async)"
[routerLink]="['hero-detail', hero.heroId]">
    <ion-img [src]="hero.imageBase64"></ion-img>
  </ion-card>
</ion-content>
```

Tabs.html

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="tab1">
      <ion-icon aria-hidden="true" name="people-sharp"></ion-icon>
      <ion-label>Heroes</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab2">
      <ion-icon aria-hidden="true" name="information-circle-
sharp"></ion-icon>
      <ion-label>Popover</ion-label>
    </ion-tab-button>

    <ion-tab-button tab="tab3">
      <ion-icon aria-hidden="true" name="person-sharp"></ion-icon>
      <ion-label>Profile</ion-label>
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```

Tabs.routes.ts

```
import { Routes } from '@angular/router';
import { TabsPage } from './tabs.page';

export const routes: Routes = [
  {
    path: 'tabs',
    component: TabsPage,
    children: [
      {
        path: 'tab1',
        loadChildren: () => [
          {
          path: '',
          loadComponent: () =>
          import('../tab1/tab1.page').then((m) => m.Tab1Page),
          },
          {
            path: 'hero-detail/:heroId',
            loadComponent: () => import('../hero-detail/hero-
detail.page').then( m => m.HeroDetailPage)
          },
        ]
      },
      {
        path: 'tab2',
        loadComponent: () =>
          import('../tab2/tab2.page').then((m) => m.Tab2Page),
      },
      {
        path: 'tab3',
        loadComponent: () =>
          import('../tab3/tab3.page').then((m) => m.Tab3Page),
      },
      {
```

```
        path: '',
        redirectTo: '/tabs/tab1',
        pathMatch: 'full',
      },
    ],
  },
  {
    path: '',
    redirectTo: '/tabs/tab1',
    pathMatch: 'full',
  },
];
```

Ionic app.html

```
<ion-app>
  <ion-router-outlet></ion-router-outlet>
</ion-app>
```

Application Security

Application security is basically normal database connection but just with added JWT token and encryption headers

Packages needed:

- MS.EFCore.SQLServer
- MS.EFCore.Tools
- Swashbuckle.AspNetCore
- MS.Extentions.Identity.Stores
- MS.Extentions.Identity.Core
- MS.AspNetCore.Identity.EFCore

Backend

Security Controller

```
using ApplicationSecurity_Backend.Models;
using ApplicationSecurity_Backend.ViewModels;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
```

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace ApplicationSecurity_Backend.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CourseController : ControllerBase
    {
        private readonly UserManager<AppUser> _userManager;
        private readonly IRepository _repository;
        private readonly IUserClaimsPrincipalFactory<AppUser>
_claimsPrincipalFactory;
        private readonly IConfiguration _configuration;

        public CourseController(UserManager<AppUser> userManager,
IUserClaimsPrincipalFactory<AppUser> claimsPrincipalFactory, IConfiguration
configuration, IRepository repository)
        {
            _userManager = userManager;
            _claimsPrincipalFactory = claimsPrincipalFactory;
            _configuration = configuration;
            _repository = repository;
        }

        [HttpGet]
        [Route("GetAllCourses")]
        [Authorize(AuthenticationSchemes =
JwtBearerDefaults.AuthenticationScheme)]
        //[Authorize]
        public async Task<IActionResult> GetAllCoursesAsync()
        {
            try
            {
                var results = await _repository.GetAllCoursesAsync();
                return Ok(results);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError,
"Internal Server Error, please contact support");
            }
        }

        [HttpPost]
        [Route("Register")]
        public async Task<IActionResult> Register(UserViewModel uvm)
        {
            var user = await _userManager.FindByIdAsync(uvm.emailaddress);

            if (user == null)
            {
                user = new AppUser
                {
                    Id = Guid.NewGuid().ToString(),
                    UserName = uvm.emailaddress,
                    Email = uvm.emailaddress
                };
```

```
                var result = await _userManager.CreateAsync(user, uvm.password);

                if (result.Errors.Count() > 0) return
StatusCode(StatusCodes.Status500InternalServerError, "Internal Server Error.
Please contact support.");
            }
            else
            {
                return Forbid("Account already exists.");
            }

            return Ok();
        }

        [HttpPost]
        [Route("Login")]
        public async Task<ActionResult> Login(UserViewModel uvm)
        {
            var user = await _userManager.FindByNameAsync(uvm.emailaddress);

            if (user != null && await _userManager.CheckPasswordAsync(user,
uvm.password))
            {
                try
                {
                    //var principal = await
_claimsPrincipalFactory.CreateAsync(user);
                    //await
HttpContext.SignInAsync(IdentityConstants.ApplicationScheme, principal);
                    return GenerateJWTToken(user);
                }
                catch (Exception)
                {
                    return StatusCode(StatusCodes.Status500InternalServerError,
"Internal Server Error. Please contact support.");
                }
            }
            else
            {
                return NotFound("Does not exist");
            }

            //var loggedInUser = new UserViewModel { EmailAddress =
uvm.EmailAddress, Password = uvm.Password };

            //return Ok(loggedInUser);
        }

        [HttpGet]
        private ActionResult GenerateJWTToken(AppUser user)
        {
            // Create JWT Token
            var claims = new[]
            {
                new Claim(JwtRegisteredClaimNames.Sub, user.Email),
                new Claim(JwtRegisteredClaimNames.Jti,
Guid.NewGuid().ToString()),
                new Claim(JwtRegisteredClaimNames.UniqueName, user.UserName)
            };

            var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Tokens:Key"]));
```

```
            var credentials = new SigningCredentials(key,
SecurityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(
                _configuration["Tokens:Issuer"],
                _configuration["Tokens:Audience"],
                claims,
                signingCredentials: credentials,
                expires: DateTime.UtcNow.AddHours(3)
            );

            return Created("", new
            {
                token = new JwtSecurityTokenHandler().WriteToken(token),
                user = user.UserName
            });
        }

    }
}
```

Factory

```
using ApplicationSecurity_Backend.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.Extensions.Options;

namespace ApplicationSecurity_Backend.Factory
{
    public class AppUserClaimsPrincipalFactory:
UserClaimsPrincipalFactory<AppUser, IdentityRole>
    {
        public AppUserClaimsPrincipalFactory(UserManager<AppUser> userManager,
        RoleManager<IdentityRole> roleManager,
        IOptions<IdentityOptions> optionsAccessor)
        : base(userManager, roleManager, optionsAccessor)
        {
        }

    }
}
```

Security DbContext

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace ApplicationSecurity_Backend.Models
{
    public class AppDbContext:IdentityDbContext<AppUser>
    {

        public AppDbContext(DbContextOptions<AppDbContext> options) :
base(options)
        {
        }

        public DbSet<Course> Courses { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
```

```
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 1,
                    Name = "AIM101",
                    Duration = "Semester",
                    Description = "Year 1, Semester 1. Academic Information
Management"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 2,
                    Name = "ALL121",
                    Duration = "Semester",
                    Description = "Year 1, Semester 2. Academic Literacy for IT"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 3,
                    Name = "INF171",
                    Duration = "Year",
                    Description = "Year 1. Systems Analysis and Design"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 4,
                    Name = "INF271",
                    Duration = "Year",
                    Description = "Year 2. Systems Analysis and Design"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 5,
                    Name = "INF272",
                    Duration = "Year",
                    Description = "Year 2. Programming"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 6,
                    Name = "INF214",
                    Duration = "Semester",
                    Description = "Year 2, Semester 1. Databases"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
```

```
                new
                {
                    CourseId = 7,
                    Name = "INF315",
                    Duration = "Semester",
                    Description = "Year 3, Semester 1. Programming Management"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 8,
                    Name = "INF324",
                    Duration = "Semester",
                    Description = "Year 3, Semester 2. IT Trends"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 9,
                    Name = "INF354",
                    Duration = "Semester",
                    Description = "Year 3, Semester 1. Programming"
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 10,
                    Name = "INF370",
                    Duration = "Year",
                    Description = "Year 3. Project"
                }
            );
        }
    }
}
```

Courses.cs

```
namespace ApplicationSecurity_Backend.Models
{
    public class Course
    {
        public int CourseId { get; set; }
        public string Name { get; set; }
        public string Duration { get; set; }
        public string Description { get; set; }
    }
}
```

Appuser.cs

```
using Microsoft.AspNetCore.Identity;

namespace ApplicationSecurity_Backend.Models
{
    public class AppUser: IdentityUser
    {
    }
```

```
}
```

Viewmodel

```
namespace ApplicationSecurity_Backend.ViewModels
{
    public class UserViewModel
    {
        public string emailaddress { get; set; }
        public string password { get; set; }

    }
}
```

Security appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection":
"Server=.;Database=ApplicationSecurity;Trusted_Connection=True;MultipleActiveResu
ltSets=True"
  },
  "Tokens": {
    "Key": "y+VRv[&)0XhxJ<sk=yUpW{yE5CH@xh",
    "Issuer": "localhost",
    "Audience": "localhost"
  }
}
```

Security program.cs

```
using ApplicationSecurity_Backend.Factory;
using ApplicationSecurity_Backend.Models;
using Microsoft.AspNetCore.Http.Features;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddCors(options => options.AddDefaultPolicy(
    include =>
    {
        include.AllowAnyHeader();
        include.AllowAnyMethod();
        include.AllowAnyOrigin();
    }));

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
```

```
builder.Services.AddSwaggerGen(c =>
{
    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
    {
        In = ParameterLocation.Header,
        Description = "Add Bearer Token",
        Name = "Authorization",
        Type = SecuritySchemeType.Http,
        BearerFormat = "JWT",
        Scheme = "bearer"
    });
    c.AddSecurityRequirement(new OpenApiSecurityRequirement
                    {
                        {
                            new OpenApiSecurityScheme
                            {
                                Reference=new OpenApiReference
                                {
                                    Type=ReferenceType.SecurityScheme,
                                    Id = "Bearer"
                                }
                            },
                            new string[]{ }
                        }
                    });
});

builder.Services.AddIdentity<AppUser, IdentityRole>(options =>
{
    options.Password.RequireUppercase = false;
    options.Password.RequireLowercase = false;
    options.Password.RequireNonAlphanumeric = false;
    options.Password.RequireDigit = true;
    options.User.RequireUniqueEmail = true;
})
.AddEntityFrameworkStores<AppDbContext>()
.AddDefaultTokenProviders();

builder.Services.AddAuthentication()
                .AddCookie()
                .AddJwtBearer(options =>
                {
                    options.TokenValidationParameters = new
TokenValidationParameters()
                    {
                        ValidIssuer = builder.Configuration["Tokens:Issuer"],
                        ValidAudience = builder.Configuration["Tokens:Audience"],
                        IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Tokens:Key"]))
                    };
                });

builder.Services.AddScoped<IUserClaimsPrincipalFactory<AppUser>,
AppUserClaimsPrincipalFactory>();

builder.Services.Configure<DataProtectionTokenProviderOptions>(options =>
options.TokenLifespan = TimeSpan.FromHours(3));

builder.Services.AddDbContext<AppDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection
")));
```

```
builder.Services.AddScoped<IRepository, Repository>();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseCors();
app.UseAuthorization();
app.UseAuthentication();
app.MapControllers();

app.Run();
```

Frontend

Login.html

```
<div class="login-wrapper" fxLayout="row"
fxLayoutAlign="center center">
  <button mat-stroked-button color="primary"
class="btn-block" (click)="Login()">Log in</button>


  <button mat-stroked-button color="primary"
class="btn-block" (click)="GetCourses()">Get
Courses</button>

  <div>{{courses | json}}</div>
  </div>
```

Login.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { DataService } from
'../services/data.service';

@Component({
  selector: 'app-login',
```

```
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})
export class LoginComponent {

  courses:any[] = []

  constructor(private router: Router, private
dataService: DataService) { }


  Login(){
    this.dataService.Login().subscribe((result: any)
=>
      localStorage.setItem('Token',
JSON.stringify(result))
    )
  }

  GetCourses(){
    this.dataService.Courses().subscribe((result:
any[]) =>
      {this.courses = result}
    )
  }

}
```

Security data.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { map, Observable } from 'rxjs';

@Injectable({
```

```
  providedIn: 'root'
})
export class DataService {

  apiUrl = 'http://localhost:5240/api/'

  // httpOptions ={
  //   headers: new HttpHeaders({
  //     ContentType: 'application/json'
  //   })
  // }

  constructor(private httpClient: HttpClient) {
  }

  Login(){
    let user = new UserCredentials
    return
this.httpClient.post(`${this.apiUrl}Course/Login`,
user)
  }

  Courses(){
    return
this.httpClient.get<any>(`${this.apiUrl}Course/GetAllC
ourses`)
  }

}

class UserCredentials  {
  EmailAddress:string = 'Addyouremailaddresshere';
  Password:string = 'Addyourpasswordhere'
}
```

Auth-interceptor.ts (Used to store JWT token)

```
import { HttpEvent, HttpHandler, HttpInterceptor,
HttpRequest } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { Observable } from "rxjs";

@Injectable()
export class AuthInterceptor implements
HttpInterceptor {

    intercept(req: HttpRequest<any>,
              next: HttpHandler):
Observable<HttpEvent<any>> {

        if (localStorage.getItem('Token')) {
            const jwt =
JSON.parse(localStorage.getItem('Token')!)
            const token = jwt.token

            const cloned = req.clone({
                headers:
req.headers.set("Authorization",
                    "Bearer " + token)
            });

             return next.handle(cloned);
        }
        else {
            return next.handle(req);
        }
    }
}
```

Reporting

Reporting uses charts and data from the database to display charts

Packages needed:

- MS.EFCore.SQLServer
- MS.EFCore.Tools
- Newtonsoft.Json
- Swashbuckle.AspNetCore

Backend

Reporting Controller

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ZAHike.API.Data;
using ZAHike.API.Models.Domain;

namespace ZAHike.API.Controllers
{
    [Route("[controller]")]
    [ApiController]
    public class RegionsController : ControllerBase
    {
        private readonly ZAHikeDbContext zAHikeDbContext;
        public RegionsController(ZAHikeDbContext zAHikeDbContext)
        {
            this.zAHikeDbContext = zAHikeDbContext;
        }

        [HttpGet]
       public async Task<IActionResult> GetAllRegions()
       {
            var listOfRegions = await zAHikeDbContext.Regions
                                      .Include(t => t.Trails)
                                      .ToListAsync();

            return Ok(listOfRegions);
        }
    }
}
```

Reporting DBContext

```
using Microsoft.EntityFrameworkCore;
using ZAHike.API.Models.Domain;
```

```
namespace ZAHike.API.Data
{
    public class ZAHikeDbContext:DbContext
    {
        public ZAHikeDbContext(DbContextOptions<ZAHikeDbContext> options):
base(options)
        {

        }
        public DbSet<Region> Regions { get; set; }
        public DbSet<HikeTrail> Trails { get; set; }
        public DbSet<TrailDifficulty> TrailDifficulty { get; set; }
    }
}
```

HikeTrail.cs

```
using System.ComponentModel.DataAnnotations;

namespace ZAHike.API.Models.Domain
{
    public class HikeTrail
    {
        [Key]
        public Guid Id { get; set; }
        public string Name { get; set; }
        public double Length { get; set; }
        public Guid RegionId { get; set; }
        public Guid TrailDifficultyId { get; set; }

        //Navigation Property
        public Region Region { get; set; }
        public TrailDifficulty TrailDifficulty { get; set; }
    }
}
```

Reporting appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "ZAHike":
"server=.;database=ZAHikeDb;Trusted_Connection=true;TrustServerCertificate=true"
  }
}
```

Reporting program.cs

```
using Microsoft.AspNetCore.Mvc.Formatters;
using Microsoft.EntityFrameworkCore;
using System.Text.Json.Serialization;
using System.Text.Json;
using ZAHike.API.Data;

var builder = WebApplication.CreateBuilder(args);
```

```
// Add services to the container.

builder.Services.AddControllers(options => {
options.OutputFormatters.RemoveType<SystemTextJsonOutputFormatter>();
options.OutputFormatters.Add(new SystemTextJsonOutputFormatter(new
JsonSerializerOptions(JsonSerializerDefaults.Web) { ReferenceHandler =
ReferenceHandler.Preserve, })); }
);
// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<ZAHikeDbContext>(options =>
{
    options.UseSqlServer(builder.Configuration.GetConnectionString("ZAHike"));
});
//builder.Services.AddCors(p => p.AddPolicy("corsapp", builder => {
builder.WithOrigins("*").AllowAnyMethod().AllowAnyHeader(); }));
builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowAngularOrigins",
    builder =>
    {
        builder.WithOrigins("*",
                            "http://localhost:4200"
                            )
                            .AllowAnyHeader()
                            .AllowAnyMethod();
    });
});

var app = builder.Build();

app.UseCors("AllowAngularOrigins");

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

Reporting Controller

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace ZAHike.API.Controllers
{
    [Route("[controller]")]
    [ApiController]
```

```
    public class MusicController : ControllerBase
    {
        [HttpGet("gospel")]
        public string GospelMusic()
        {
            return "Gospel Music playing";
        }

        [HttpGet("rnb")]
        public string RnBMusic()
        {
            return "RnB Music playing";
        }

        [HttpGet("house")]
        public string HouseMusic()
        {
            return "House Music playing";
        }
    }
}
```

Frontend

Chart.component.html

```
<div class="row">
    <div class="col-lg-6">
        <h2>Line Chart</h2>
        <canvas id="linechart"></canvas>
    </div>
    <div class="col-lg-6">
        <h2>Bar Chart</h2>
        <canvas id="barchart"></canvas>

    </div>
    <div class="col-lg-6">
        <h2>Pie Chart</h2>
        <canvas id="piechart"></canvas>

    </div>
    <div class="col-lg-6">
        <h2>Doughnut Chart</h2>
        <canvas id="dochart"></canvas>
    </div>
```

```
    <div class="col-lg-6">
        <h2>polarArea Chart</h2>
        <canvas id="pochart"></canvas>
    </div>

    <div class="col-lg-6">
        <h2>Radar Chart</h2>
        <canvas id="rochart"></canvas>
    </div>
</div>
```

Chart.component.ts

```
import { Component, ElementRef, OnInit, ViewChild }
from '@angular/core';
import { Chart,registerables } from
'node_modules/chart.js';
import { RegionService } from
'../service/region.service';
import { RegionModel } from
'../service/Models/regionsModel';


Chart.register(...registerables);

@Component({
  selector: 'app-charts',
  templateUrl: './charts.component.html',
  styleUrls: ['./charts.component.css']
})

export class ChartsComponent implements OnInit{

  data: any;
  @ViewChild('myTemp')
```

```
  myTempRef!: ElementRef;


  constructor(private regionService : RegionService)
{}

  ngOnInit(): void {
    this.regionService.getRegions().subscribe(response
=> {
      let regionList = response;

      this.data = response.$values;

      this.populateChartData(this.data);
      console.log('data',regionList)
      return regionList
    });
  }

  populateChartData(data: RegionModel[]) {

    let labelsData: string [] = [];
    let labelsPopulation: number [] = [];

    data.forEach((element: any) => {
      labelsData.push(element.code);
      labelsPopulation.push(element.population)
    });


    new Chart("barchart", {
      type: 'bar',
      data: {
        labels: labelsData,
```

```
    datasets: [{
      label: '# of Population',
      data: labelsPopulation,
      borderWidth: 1
    }]
  },

  options: {
    scales: {
      y: {
        beginAtZero: true
      },
    }

  }
});


new Chart("piechart", {
  type: 'pie',
  data: {
    labels: labelsData,
    datasets: [{
      label: '# of Population',
      data: labelsPopulation,
      borderWidth: 1
    }]
  },
  options: {
    scales: {
      y: {
        beginAtZero: true
      }
    }
```

```
  }
});

new Chart("dochart", {
  type: 'doughnut',
  data: {
    labels: labelsData,
    datasets: [{
      label: '# of Population',
      data: labelsPopulation,
      borderWidth: 1
    }]
  },
  options: {
    scales: {
      y: {
        beginAtZero: true
      }
    }
  }
});

new Chart("pochart", {
  type: 'polarArea',
  data: {
    labels: labelsData,
    datasets: [{
      label: '# of Population',
      data: labelsPopulation,
      borderWidth: 1
    }]
  },
  options: {
    scales: {
```

```
      y: {
        beginAtZero: true
      }
    }
  }
});

new Chart("rochart", {
  type: 'radar',
  data: {
    labels: labelsData,
    datasets: [{
      label: '# of Population',
      data: labelsPopulation,
      borderWidth: 1
    }]
  },
  options: {
    scales: {
      y: {
        beginAtZero: true
      }
    }
  }
});

new Chart("linechart", {
  type: 'line',
  data: {
    labels: labelsData,
    datasets: [{
      label: '# of Population',
      data: labelsPopulation,
      borderWidth: 1
```

```
    }]

  },
  options: {
    scales: {
      y: {
        beginAtZero: true
      }
    }
  }
});

new Chart("bubchart", {
  type: 'bubble',
  data: {
    labels: labelsData,
    datasets: [{
      label: '# of Population',
      data: labelsPopulation,
      borderWidth: 1

    }]

  },
  options: {
    scales: {
      y: {
        beginAtZero: true
      }
    }
  }
});
```

```
  }

}
```

---

regionsModel.ts (Model needed to call info from api)

```
export class RegionModel
{
    id: string = '';
    code: string = '';
    name: string = '';
    area: number = 0;
    lat: number = 0;
    long: number = 0;
    population: number = 0;
}
```

Region.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpClientModule, HttpHeaders }
from '@angular/common/http';
import { Observable } from 'rxjs/internal/Observable';

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

  constructor(private httpclient : HttpClient) { }


  public getRegions(): Observable<any> {
    let appheaders = this.getHeaderConfigurations();
```

```
    return
this.httpclient.get<any[]>('https://localhost:7250/Reg
ions', { headers: appheaders});
  }

  private getHeaderConfigurations()
    {
        return new HttpHeaders({
            'Content-Type': 'application/json;
charset=utf-8',
                'Access-Control-Allow-Origin': '*'
        });
    }
}
```

Chatbot

Chatbot has 3 important bits
Intent: What user says

Actions: How system replies

Stories: Basic flow of events

Frontend

Chat-support.component.html

```
<div id="assistant">
  <button id="assistant-popup-button" (click)="openSupportPopup()">
    Chat Support?
  </button>
  <div id="assistant-popup" [style.display]="isOpen ? 'block' : 'none'">
    <div id="assistant-popup-header">
      Your friendly Assistant
      <button id="assistant-popup-close-button"
(click)="openSupportPopup()">
       X
```

```
    </button>
  </div>
  <div id="assistant-popup-body">
    <div class="messages" #scrollMe>
      <div *ngFor="let message of messages" class="message">
        <div [class]="message.type">
          {{ message.message }}
        </div>
      </div>
      <div
        *ngIf="loading"
        class="message"
        style="width: 100%; display: block"
      >
        <div [class]="'client'">...</div>
      </div>
    </div>
  </div>
  <form id="assistant-popup-footer" [formGroup]="chatForm">
    <input
      formControlName="message"
      type="text"
      id="assistant-popup-input"
      placeholder="Type your message here..."
    />
    <button
      id="assistant-popup-submit-button"
      [disabled]="!chatForm.valid"
      (click)="sendMessage()"
    >
      Submit
    </button>
  </form>
</div>
```

```
</div>
```

Chat-support.component.ts

```
import { Component, ViewChild } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { MessageService } from '../service/api.service';

export interface Message {
  type: string;
  message: string;
}

@Component({
  selector: 'app-chat-support',
  templateUrl: './chat-support.component.html',
  styleUrls: ['./chat-support.component.scss'],
})
export class ChatSupportComponent {
  isOpen = false;
  loading = false;
  messages: Message[] = [];
  chatForm = new FormGroup({
    message: new FormControl('', [Validators.required]),
  });
  @ViewChild('scrollMe') private myScrollContainer: any;

  constructor(private messageService: MessageService) {
  }

  openSupportPopup() {
    this.isOpen = !this.isOpen;
  }
```

```
  sendMessage() {
    const sentMessage = this.chatForm.value.message!;
    this.loading = true;
    this.messages.push({
      type: 'user',
      message: sentMessage,
    });
    this.chatForm.reset();
    this.scrollToBottom();
    this.messageService.sendMessage(sentMessage).subscribe((response: any)
=> {
      for (const obj of response) {
        let value
        if (obj.hasOwnProperty('text') ) {
          value = obj['text']
          this.pushMessage(value)

        }
        if (obj.hasOwnProperty('image') ) {
          value = obj['image']
          this.pushMessage(value)
        }
      }
    });
  }

  pushMessage(message:string){
     this.messages.push({
        type: 'client',
        message: message,
      });
      this.scrollToBottom();
  }
```

```
  scrollToBottom() {
    setTimeout(() => {
      try {
        this.myScrollContainer.nativeElement.scrollTop =
          this.myScrollContainer.nativeElement.scrollHeight + 500;
      } catch (err) {}
    }, 150);
  }
}
```

---

api.service.ts

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

@Injectable({
  providedIn: 'root',
})
export class MessageService {
  constructor(private http: HttpClient) {}

  sendMessage(message: string) {
    return this.http.post('http://localhost:5005/webhooks/rest/webhook', {
message: message });
  }
}
```

App.component.html (Include the chat support)

```
<router-outlet></router-outlet>
<app-chat-support></app-chat-support>
```

Chat-support.component.css

```
@import
url("https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=
swap");
#assistant {
  font-family: "Roboto", sans-serif;
  #assistant-popup-button {
    position: fixed;
    bottom: 20px;
    right: 20px;
    padding: 10px 20px;
    background-color: #333;
    color: #ffffff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 14px;
    z-index: 1000;
  }

  #assistant-popup {
    position: fixed;
    bottom: 40px;
    right: 20px;
    width: 450px;
    height: 50vh;
    min-height: 450px;
    background-color: white;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
    border-radius: 5px;
    z-index: 1000;
    display: none;
    #assistant-popup-header {
```

```
  background-color: #333;
  color: white;
  font-size: 18px;
  padding: 10px;
  border-top-left-radius: 5px;
  border-top-right-radius: 5px;
  #assistant-popup-close-button {
    float: right;
    border: none;
    background-color: transparent;
    color: #fff;
    font-size: 14px;
    cursor: pointer;
  }
}

#assistant-popup-body {
  height: calc(100% - 133px);
  padding: 10px;
}

#assistant-popup-footer {
  background-color: #333;
  color: white;
  font-size: 14px;
  padding: 10px;
  border-bottom-left-radius: 5px;
  border-bottom-right-radius: 5px;
  #assistant-popup-input {
    width: 100%;
    padding: 10px;
    border: 1px solid #fff;
    border-radius: 5px 5px 0 0;
    box-sizing: border-box;
```

```
    font-size: 14px;
  }

  #assistant-popup-submit-button {
    width: 100%;
    padding: 10px;
    background-color: #2ca1da;
    color: #fff;
    border: none;
    border-radius: 0 0 5px 5px;
    cursor: pointer;
    font-size: 14px;
  }
}

.messages {
  height: 100%;
  overflow: auto;
  .message {
    display: flow-root;
    width: 100%;
    .client {
      background-color: #d7d7d7;
      color: #333;
      padding: 10px;
      border-radius: 5px;
      margin-bottom: 10px;
      display: inline-block;
      max-width: 80%;
    }

    .user {
      border: 0.5px solid #333;
      background-color: #85ff7a;
```

```
            color: #333;
            padding: 10px;
            border-radius: 5px;
            margin-bottom: 10px;
            display: inline-block;
            max-width: 80%;
            text-align: right;
            float: right;
          }
        }
      }
    }
}
```

Backend

nlu.yml

```
version: "3.1"
nlu:
- intent: greet
  examples: |
    - hey
    - hello
    - hi
    - hello there
    - good morning
    - good evening
    - moin.
    - hey there
    - let's go
    - hey dude
    - goodmorning
    - goodevening
    - good afternoon
```

```
- intent: goodbye
  examples: |
    - cu
    - good by
    - cee you later
    - good night
    - bye
    - goodbye
    - have a nice day
    - see you around
    - bye bye
    - see you later
- intent: thankyou
  examples: |
    - thank you
    - great thanks
    - thank you so much
    - thank you bye
    - thank you, keep well
    - thnak you
    - thanks
- intent: affirm
  examples: |
    - yes
    - y
    - indeed
    - of course
    - that sounds good
    - correct
    - sure
    - ok
    - that's right
    - cool
    - you bet
```

```
    - I sure did
    - more or less
    - fine
- intent: deny
  examples: |
    - no
    - n
    - never
    - I don't think so
    - don't like that
    - no way
    - not really
    - none
    - absolutely not
    - no thanks
    - didn't do any
    - no I didn't
    - could have been better
    - not great
    - nope
- intent: inform
  examples: |
    - my email address is [example@gmail.com](email)
    - [user@gmail.com](email) is my email
    - a primary email address of mine is [myname@gmail.com](email)
    - my name is [ridewaan hanslo](name)
    - [Ridewaan hanslo](name)
    - my fullname is [Ridewaan Hanslo](name)
    - firstname: [john](name) lastname: [doe](name)
    - my address is [123 main street Brummeria 0184 Pretoria South
Africa](address)
    - the street address at which i reside is [123 main street Brummeria
Pretoria South Africa](address)
    - [123 main street Brummeria 0184 Pretoria South Africa](address)
```

```
    - my contact number is [+27 78 234 9870](number)
    - [+27 60 123 6879](number)
    - my telephone number is [+27601236879](number)
    - my age is [18](age)
    - i am [25](age) years old
    - [65](age)
    - [ridewaan@gmail.com](email)
    - [Ridewaan Hanslo]{"entity": "name", "value": "ridewaan hanslo"}
    - [123 crazy street brummeria pretoria](address)
    - [+27781839943](number)
    - [39](age)
    - [ridewaan hanslo](name)
    - [123 main street brummeria pretoria](address)
    - [+7818312341](number)
- intent: out_of_scope
  examples: |
    - wait stop
    - you are no help
    - that is not what I said
    - what is the time
    - what is the weather
    - how do you compare to chatGPT
    - can you help me with shopping
    - why are you broken
    - this is not working for me
    - stop asking
    - none of your business
    - I prefer not to answer
    - that does not make sense
    - nevermind
    - I changed my mind
- intent: mood_great
  examples: |
    - perfect
```

```
    - great
    - amazing
    - feeling like a king
    - wonderful
    - I am feeling very good
    - I am great
    - I am amazing
    - I am going to save the world
    - super stoked
    - extremely good
    - so so perfect
    - so good
    - so perfect
- intent: mood_unhappy
  examples: |
    - my day was horrible
    - I am sad
    - I don't feel very well
    - I am disappointed
    - super sad
    - I'm so sad
    - sad
    - very sad
    - unhappy
    - not good
    - not very good
    - extremly sad
    - so saad
    - so sad
    - im sad
- intent: bot_challenge
  examples: |
    - are you a bot?
    - are you a human?
```

```
    - am I talking to a bot?
    - am I talking to a human?
- intent: ask_location
  examples: |
    - where are you located?
    - what is your address?
    - what is your business address?
    - where is the business located?
    - where is your business?
    - business location?
    - location?
    - wher is your business located
    - what is your location
- intent: ask_business_type
  examples: |
    - what type of business are you?
    - what kind of business is this?
    - what do your business specialise in?
    - what do your business specialize with?
    - business type?
    - what is the focus of the business?
    - business focus?
    - what type of business are you
- intent: ask_product_services
  examples: |
    - what services or products do you offer?
    - what services or products does your business offer?
    - products offered?
    - services offered?
    - offered products and services?
    - what products and services do you offer?
- intent: ask_contact_us
  examples: |
    - how can i contact or reach you?
```

```
    - what are your contact details?
    - what are your contact information?
    - your contact information?
    - what is your email address?
    - what is your telephone number?
    - what is your cellphone number?
    - how can we reach you
    - how can we contact you?
    - how can we reach you?
- intent: mood_happy
  examples: |
    - I am feeling great
    - I love my life
    - I am feeling awesome
    - good to be alive
    - I am happy
    - Im happy
    - im happy
- synonym: ridewaan hanslo
  examples: |
    - Ridewaan Hanslo
```

---

rules.yml

```
version: "3.1"

rules:

- rule: Say goodbye anytime the user says goodbye
  steps:
  - intent: goodbye
  - action: utter_goodbye

- rule: Say 'I am a bot' anytime the user challenges
```

```
  steps:
  - intent: bot_challenge
  - action: utter_iamabot
```

Stories.yml

```
version: "3.1"

stories:

- story: happy path
  steps:
  - intent: greet
  - action: utter_greet
  - intent: mood_great
  - action: utter_happy

- story: happy path 2
  steps:
  - intent: greet
  - action: utter_greet
  - intent: mood_happy
  - action: utter_happy

- story: sad path 1
  steps:
  - intent: greet
  - action: utter_greet
  - intent: mood_unhappy
  - action: utter_cheer_up
  # - action: utter_did_that_help
  # - intent: affirm
  # - action: utter_happy
```

```yaml
- story: sad path 2
  steps:
  - intent: greet
  - action: utter_greet
  - intent: mood_unhappy
  - action: utter_cheer_up
  # - action: utter_did_that_help
  # - intent: deny
  # - action: utter_goodbye

- story: business information path 1
  steps:
  - intent: ask_product_services
  - action: utter_product_service_info
  - intent: ask_contact_us
  - action: utter_contact_us_info

- story: onboarding happy path
  steps:
  - intent: greet
  - action: utter_greet
  - intent: affirm
  - action: create_account_form
  - active_loop: create_account_form
  - active_loop: null
  - action: utter_slots_values
  - intent: thankyou
  - action: utter_no_problem
  - action: utter_goodbye

- story: onboarding stop
  steps:
  - intent: greet
  - action: utter_greet
```

```
  - intent: affirm
  - action: create_account_form
  - active_loop: create_account_form
  - intent: out_of_scope
  - action: utter_ask_continue
  - intent: deny
  - action: action_deactivate_loop
  - active_loop: null
  - action: utter_goodbye

- story: onboarding continue
  steps:
  - intent: greet
  - action: utter_greet
  - intent: affirm
  - action: create_account_form
  - active_loop: create_account_form
  - intent: out_of_scope
  - action: utter_ask_continue
  - intent: affirm
  - active_loop: create_account_form
  - active_loop: null
  - action: utter_slots_values

- story: no onboarding
  steps:
  - intent: greet
  - action: utter_greet
  - intent: deny
  - action: utter_goodbye

- story: business information path 2
  steps:
  - intent: ask_location
```

```
  - action: utter_location_info
  - intent: ask_business_type
  - action: utter_business_type_info

- story: interactive_story_1
  steps:
  - intent: greet
  - action: utter_greet
  - intent: ask_business_type
  - action: utter_business_type_info


- story: interactive_story_1
  steps:
  - intent: greet
  - action: utter_greet
  - intent: affirm
  - action: create_account_form
  - active_loop: create_account_form
  - slot_was_set:
    - requested_slot: age
  - intent: inform
    entities:
    - age: '39'
  - slot_was_set:
    - age: '39'
  - action: create_account_form
  - slot_was_set:
    - requested_slot: name
  - intent: inform
    entities:
    - name: ridewaan hanslo
  - slot_was_set:
    - name: ridewaan hanslo
```

```
- action: create_account_form
- slot_was_set:
  - requested_slot: email
- intent: inform
  entities:
  - email: ridewaan@gmail.com
- slot_was_set:
  - email: ridewaan@gmail.com
- action: create_account_form
- slot_was_set:
  - requested_slot: address
- intent: inform
  entities:
  - address: 123 main street brummeria pretoria
- slot_was_set:
  - address: 123 main street brummeria pretoria
- action: create_account_form
- slot_was_set:
  - requested_slot: number
- intent: inform
  entities:
  - number: '+7818312341'
- slot_was_set:
  - number: '+7818312341'
- action: create_account_form
- slot_was_set:
  - requested_slot: null
- active_loop: null
- action: utter_slots_values
- intent: thankyou
- action: utter_no_problem
- action: utter_goodbye
- intent: ask_business_type
- action: utter_business_type_info
```

```
  - intent: ask_contact_us
  - action: utter_contact_us_info
  - intent: ask_location
  - action: utter_location_info
  - intent: ask_product_services
  - action: utter_product_service_info
  - intent: goodbye
  - action: utter_goodbye

- story: interactive_story_1
  steps:
  - intent: mood_unhappy
  - action: utter_cheer_up
  - intent: ask_location
  - action: utter_location_info
  - intent: ask_location
  - action: utter_location_info
  - intent: ask_business_type
  - action: utter_business_type_info
  - intent: ask_contact_us
  - action: action_default_fallback
  - intent: ask_contact_us
  - action: utter_contact_us_info
  - intent: ask_contact_us
  - action: utter_contact_us_info
  - intent: ask_contact_us
  - action: utter_contact_us_info
  - intent: ask_product_services
  - action: utter_product_service_info
  - intent: ask_product_services
  - action: utter_product_service_info
  - intent: ask_location
  - action: utter_location_info
```

```
- story: interactive_story_1
  steps:
  - intent: greet
  - action: utter_greet
  - intent: mood_happy
  - action: utter_happy
```

Domain.yml

```
version: '3.1'
intents:
- affirm
- ask_business_type
- ask_contact_us
- ask_location
- ask_product_services
- bot_challenge
- deny
- goodbye
- greet
- inform
- mood_great
- mood_happy
- mood_unhappy
- out_of_scope
- thankyou
slots:
  email:
    type: any
    mappings:
    - type: from_entity
      entity: email
  name:
    type: any
    mappings:
```

```
    - type: from_entity
      entity: name
  address:
    type: any
    mappings:
    - type: from_entity
      entity: address
  number:
    type: any
    mappings:
    - type: from_entity
      entity: number
  age:
    type: any
    mappings:
    - type: from_entity
      entity: age
entities:
- email
- age
- name
- address
- number
responses:
  utter_greet:
  - text: Hey! I am your friendly Takealot assistant!
I can help you with some information of our company
and what we offer. Further, you can create an account
in a few easy steps with my help. Would you like to
create an account now?
  utter_cheer_up:
  - text: 'Here is something to cheer you up:'
    image: https://i.imgur.com/nGF1K8f.jpg
  utter_did_that_help:
```

```
  - text: Did that help you?
  utter_happy:
  - text: Great, carry on!
  utter_goodbye:
  - text: Bye
  utter_iamabot:
  - text: I am a bot, powered by Rasa.
  utter_location_info:
  - text: Our Central Office is 12th Floor, 10 Rua
Vasco Da Gama Plain, Foreshore, Cape Town, 8001. [View
in Google
Maps](https://www.google.com/maps/place/Takealot+Centr
al+Office/@-
33.9184311,18.4284028,15z/data=!4m6!3m5!1s0x1dcc5d88dd
db4bfd:0xf34157712fde5e18!8m2!3d-
33.9184311!4d18.4284028!16s%2Fg%2F11b5ys2cy3)
  utter_business_type_info:
  - text: 'Today, takealot.com is the leading
ecommerce retailer in South Africa and one of the
largest, most innovative ecommerce retailers on the
African continent. The business was initiated with a
simple vision in mind: To be the largest, simplest,
most customer-centric online shopping destination in
Africa.'
  utter_product_service_info:
  - text: 'We a popular online retailer in South
Africa that offers a wide range of products across
various categories. Here are some of the product
categories you can find on our website:  Electronics:
This category includes items such as smartphones,
laptops, TVs, cameras, audio equipment, gaming
consoles, and accessories.  Home & Kitchen: Here, you
can find appliances, furniture, home decor,
kitchenware, bedding, and other household items.  Baby
```

```
& Toddler: This category offers products for infants
and young children, including baby gear, toys,
clothing, nursery furniture, and feeding
essentials.  Beauty, Health & Wellness: We provide a
selection of skincare, haircare, makeup, personal care
products, vitamins, supplements, fitness equipment,
and health-related items.  Fashion: You can explore a
variety of clothing, shoes, bags, accessories, and
fashion brands for men, women, and children.  Books,
Movies & TV Series: We offer a wide range of books,
eBooks, movies, TV series, music, and related media in
various formats.  Sports, Outdoor & Travel: This
category covers sports equipment, outdoor gear,
camping supplies, luggage, travel accessories, and
fitness products.  Toys & Games: Here, you can find
toys, games, puzzles, board games, action figures, and
educational items for children of all
ages.  Automotive: We provide automotive accessories,
car care products, tools, and equipment.  DIY, Tools &
Home Improvement: This category includes tools,
hardware, power equipment, DIY supplies, and home
improvement products.  Please note that this list is
not exhaustive, and We offer many more product
categories and subcategories. You can visit our
website or app to explore our complete range of
products.'
  utter_contact_us_info:
  - text: Please visit our [Contact
Us](https://www.takealot.com/about/contact-us) page
for detailed information on how to contact us.
  utter_ask_age:
  - text: What is your age? Note you must be over 18
to create an account.
  utter_ask_name:
```

```
  - text: Can you please provide your Fullname,
including your Firstname and Lastname?
  utter_ask_address:
  - text: Can you please provide your delivery address
for us to dispatch your awesome purchases to?
  utter_ask_number:
  - text: Please provide your contact number so that
we can contact you for deliveries (format e.g. +27 82
123 4456).
  utter_ask_email:
  - text: Can you please provide your preferred email
address to create this account. This also allow us to
provide updates and communications on your purchases
and delivery thereof?
  utter_slots_values:
  - text: |2-
        Here's the captured information you provided:
        - Age: {age}
        - Fullname: {name}
        - Address: {address}
        - Contact number: {number}
        - Email address: {email}
        NB: We will use this information to create your
account and provide you an update via email when
completed.
  utter_no_problem:
  - text: No problem :) - Your email of account
creation status will be finalized shortly.
  utter_ask_continue:
  - text: Sorry, I don't quite understand. Do you want
to continue?
forms:
  create_account_form:
    required_slots:
```

```
    - age
    - name
    - email
    - address
    - number
session_config:
  session_expiration_time: 60
  carry_over_slots_to_new_session: true
actions:
- utter_happy
- utter_greet
- utter_product_service_info
- utter_location_info
- utter_cheer_up
- utter_contact_us_info
- utter_business_type_info
- utter_no_problem
- utter_slots_values
- utter_goodbye
```

Endpoints.yml (Check where the port is)

```
action_endpoint:
 url: "http://localhost:5055/webhook"
```

Assignment 1 (Basic database connect and display in table)

Assign 1 Controller

```
using Architecture.Models;
using Architecture.ViewModel;
using Microsoft.AspNetCore.Components.Forms;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Reflection.Metadata.Ecma335;

namespace Architecture.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CourseController : ControllerBase
    {
        private readonly ICourseRepository _courseRepository;

        public CourseController(ICourseRepository courseRepository)
        {
            _courseRepository = courseRepository;
        }

        [HttpGet]
        [Route("GetAllCourses")]
        public async Task<IActionResult> GetAllCourses()
        {
            try
            {
                var results = await _courseRepository.GetAllCourseAsync();
                return Ok(results);
            }
            catch (Exception)
            {
                return StatusCode(500,"Internal Server Error. Please contact
support.");
            }
        }

        [HttpGet]
        [Route("GetCourse/{courseId}")]
        public async Task<IActionResult> GetCourseAsync(int courseId)
        {
            try
            {
                var result = await _courseRepository.GetCourseAsync(courseId);

                if (result == null) return NotFound("Course does not exist. You
need to create it first");

                return Ok(result);
            }
            catch (Exception)
            {
                return StatusCode(500, "Internal Server Error. Please contact
support");
            }
        }

        [HttpPost]
        [Route("AddCourse")]
        public async Task<IActionResult> AddCourse(CourseViewModel cvm)
```

```
        {
            var course = new Course { Name = cvm.Name, Duration = cvm.Duration,
Description = cvm.Description };

            try
            {
                _courseRepository.Add(course);
                await _courseRepository.SaveChangesAsync();
            }
            catch (Exception)
            {
                return BadRequest("Invalid transaction");
            }

            return Ok(course);
        }

        [HttpPut]
        [Route("EditCourse/{courseId}")]
        public async Task<ActionResult<CourseViewModel>> EditCourse(int courseId,
CourseViewModel courseModel)
        {
            try
            {
                var existingCourse = await
_courseRepository.GetCourseAsync(courseId);
                if (existingCourse == null) return NotFound($"The course does not
exist");

                existingCourse.Name = courseModel.Name;
                existingCourse.Duration = courseModel.Duration;
                existingCourse.Description = courseModel.Description;

                if (await _courseRepository.SaveChangesAsync())
                {
                    return Ok(existingCourse);
                }
            }
            catch (Exception)
            {
                return StatusCode(500, "Internal Server Error. Please contact
support.");
            }
            return BadRequest("Your request is invalid.");
        }

        [HttpDelete]
        [Route("DeleteCourse/{courseId}")]
        public async Task<IActionResult> DeleteCourse(int courseId)
        {
            try
            {
                var existingCourse = await
_courseRepository.GetCourseAsync(courseId);

                if (existingCourse == null) return NotFound($"The course does not
exist");

                _courseRepository.Delete(existingCourse);

                if (await _courseRepository.SaveChangesAsync()) return
Ok(existingCourse);
```

```
            }
            catch (Exception)
            {
                return StatusCode(500, "Internal Server Error. Please contact
support.");
            }
            return BadRequest("Your request is invalid.");
        }
    }
}
```

Assign 1 DB Context

```
using Microsoft.EntityFrameworkCore;

namespace Architecture.Models
{
    public class AppDbContext:DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options): base
(options)
        {
        }

        public DbSet<Course> Courses { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            // Course

            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 1,
                    Name = "AIM101",
                    Duration = "Semester",
                    Description = "Year 1, Semester 1. Academic Information
Management",
                    LocationId = 5
                }
            );

            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 2,
                    Name = "ALL121",
                    Duration = "Semester",
                    Description = "Year 1, Semester 2. Academic Literacy for IT",
                    LocationId = 4
                }
             );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 3,
```

```
            Name = "INF171",
            Duration = "Year",
            Description = "Year 1. Systems Analysis and Design",
            LocationId = 3
        }
    );
    modelBuilder.Entity<Course>()
        .HasData(
        new
        {
            CourseId = 4,
            Name = "INF271",
            Duration = "Year",
            Description = "Year 2. Systems Analysis and Design",
            LocationId = 2
        }
    );
    modelBuilder.Entity<Course>()
        .HasData(
        new
        {
            CourseId = 5,
            Name = "INF272",
            Duration = "Year",
            Description = "Year 2. Programming",
            LocationId = 1
        }
    );
    modelBuilder.Entity<Course>()
        .HasData(
        new
        {
            CourseId = 6,
            Name = "INF214",
            Duration = "Semester",
            Description = "Year 2, Semester 1. Databases",
            LocationId = 2
        }
    );
    modelBuilder.Entity<Course>()
        .HasData(
        new
        {
            CourseId = 7,
            Name = "INF315",
            Duration = "Semester",
            Description = "Year 3, Semester 1. Programming Management",
            LocationId = 3
        }
    );
    modelBuilder.Entity<Course>()
        .HasData(
        new
        {
            CourseId = 8,
            Name = "INF324",
            Duration = "Semester",
            Description = "Year 3, Semester 2. IT Trends",
            LocationId = 4
        }
    );
    modelBuilder.Entity<Course>()
        .HasData(
```

```
                new
                {
                    CourseId = 9,
                    Name = "INF354",
                    Duration = "Semester",
                    Description = "Year 3, Semester 1. Programming",
                    LocationId = 1
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 10,
                    Name = "INF370",
                    Duration = "Year",
                    Description = "Year 3. Project",
                    LocationId = 5
                }
            );
        }
    }
}
```

Course.cs

```
using System.ComponentModel.DataAnnotations;
```

```
namespace Architecture.Models
{
    public class Course
    {
        [Key]
        public int CourseId { get; set; }
        public string Name { get; set; } = String.Empty;
        public string Description { get; set; } = String.Empty;
        public string Duration { get; set; } = String.Empty;
    }
}
```

Assign 1 Repository

```
using Microsoft.EntityFrameworkCore;

namespace Architecture.Models
{
    public class CourseRepository : ICourseRepository
    {
        private readonly AppDbContext _appDbContext;

        public CourseRepository(AppDbContext appDbContext)
        {
                _appDbContext = appDbContext;
        }

        public void Add<T>(T entity) where T : class
        {
            _appDbContext.Add(entity);
        }
```

```
        public void Delete<T>(T entity) where T : class
        {
            _appDbContext.Remove(entity);
        }

        public async Task<Course[]> GetAllCourseAsync()
        {
            IQueryable<Course> query = _appDbContext.Courses;
            return await query.ToArrayAsync();
        }

        public async Task<Course> GetCourseAsync(int courseId)
        {
           IQueryable<Course> query = _appDbContext.Courses.Where(c => c.CourseId
== courseId);
           return await query.FirstOrDefaultAsync();
        }

        public async Task<bool> SaveChangesAsync()
        {
            return await _appDbContext.SaveChangesAsync() > 0;
        }
    }
}
```

Assign 1 Irepository

```
namespace Architecture.Models
{
    public interface ICourseRepository
    {
        void Add<T>(T entity) where T : class;
        void Delete<T>(T entity) where T : class;

        Task<bool> SaveChangesAsync();

        // Course
        Task<Course[]> GetAllCourseAsync();
        Task<Course> GetCourseAsync(int courseId);

    }
}
```

Course ViewModel

```
namespace Architecture.ViewModel
{
    public class CourseViewModel
    {
        public string Name { get; set; }
        public string Duration { get; set; }
        public string Description { get; set; }
    }
}
```

Addcourse.html

```
<h4>Add Course</h4>
```

```
    <hr>
    <div class="container">
    <div class="row">
    <form [formGroup]="courseForm" class="form-
horizontal"  (ngSubmit)="onSubmit()">
        <div class="form-group">
            <label class=" col-sm-1"
for="name">Name:</label>
            <div class="col-sm-6">
              <input class="form-control" id="name"
placeholder="Enter name" [required]="true"
formControlName="name">
            </div>
        </div>
        <div class="form-group">
            <label class="col-sm-1"
for="duration">Duration:</label>
            <div class="col-sm-6">
              <input class="form-control"
id="duration" placeholder="Enter duration"
[required]="true" formControlName="duration">
            </div>
        </div>
        <div class="form-group">
            <label class="col-sm-1"
for="description">Description:</label>
            <div class="col-sm-6">
              <textarea class="form-control"
type="text" id="description" placeholder="Enter
description" [required]="true"
formControlName="description"></textarea>
            </div>
        </div>
        <div class="form-group">
```

```
          <div class="col-sm-offset-1 col-sm-6">
            <button style="margin-right:1em;"
type="submit" [disabled]="!courseForm.valid"
class="btn btn-sm btn-default">Submit</button>
            <button type="button" class="btn btn-sm
btn-warning" (click)="cancel()">Cancel</button>
          </div>
        </div>
      </form>
      </div>
    </div>
```

Addcourse.ts

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup } from
'@angular/forms';
import { Router } from '@angular/router';
import { DataService } from
'../services/data.service';

@Component({
  selector: 'app-add-course',
  templateUrl: './add-course.component.html',
  styleUrls: ['./add-course.component.scss']
})
export class AddCourseComponent implements OnInit {

  courseForm = new FormGroup(
  {
      name: new FormControl(''),
      duration: new FormControl(''),
      description: new FormControl('')
  })
```

```
  constructor(private dataService: DataService,
private router: Router) { }

  ngOnInit(): void {
  }

  cancel(){
    this.router.navigate(['/courses'])
  }

  onSubmit(){
    this.dataService.addCourse(this.courseForm.value).
subscribe(result => {
          this.router.navigate(['/courses'])
    })
  }
}
```

Editcourse.html

```
<h4>Edit Course</h4>
<hr>
<div class="container">
<div class="row">
<form [formGroup]="courseForm" class="form-
horizontal"  (ngSubmit)="onSubmit()">
    <div class="form-group">
        <label class=" col-sm-1"
for="name">Name:</label>
        <div class="col-sm-6">
          <input class="form-control" id="name"
placeholder="Enter name" [required]="true"
formControlName="name">
        </div>
    </div>
    <div class="form-group">
```

```
        <label class="col-sm-1"
for="duration">Duration:</label>
        <div class="col-sm-6">
          <input class="form-control" id="duration"
placeholder="Enter duration" [required]="true"
formControlName="duration">
        </div>
    </div>
    <div class="form-group">
        <label class="col-sm-1"
for="description">Description:</label>
        <div class="col-sm-6">
          <textarea class="form-control" type="text"
id="description" placeholder="Enter description"
[required]="true"
formControlName="description"></textarea>
        </div>
    </div>
    <div class="form-group">
      <div class="col-sm-offset-1 col-sm-6">
        <button style="margin-right:1em;"
type="submit" [disabled]="!courseForm.valid"
class="btn btn-sm btn-default">Submit</button>
        <button type="button" class="btn btn-sm btn-
warning" (click)="cancel()">Cancel</button>
      </div>
    </div>
  </form>
  </div>
</div>
```

Editcourse.ts

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

```
import { FormControl, FormGroup } from
'@angular/forms';
import { ActivatedRoute, Router } from
'@angular/router';
import { DataService } from
'../services/data.service';
import { Course } from '../shared/course';

@Component({
  selector: 'app-edit-course',
  templateUrl: './edit-course.component.html',
  styleUrls: ['./edit-course.component.scss']
})
export class EditCourseComponent implements OnInit {

  courseForm = new FormGroup(
    {
        name: new FormControl(''),
        duration: new FormControl(''),
        description: new FormControl('')
    })

  course:any
  constructor(private dataService: DataService,
private router: Router, private route:ActivatedRoute)
{ }

  ngOnInit(): void {
    this.dataService.getCourse(+this.route.snapshot.pa
rams['id']).subscribe(result => {
      this.course = result
      this.courseForm.patchValue({
        name: this.course.name,
        duration: this.course.duration,
```

```
        description: this.course.description
      });
  })
  }


  cancel(){
    this.router.navigate(['/courses'])
  }

  onSubmit(){
    this.dataService.editCourse(this.course.courseId,
this.courseForm.value).subscribe(result => {
          this.router.navigate(['/courses'])
    })
  }

}
```

Course.ts (Model)

```
export interface Course {
    courseId: Number;
    name:String;
    description:String;
    duration:String;
}
```

Assign 1 App-routing.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from
'@angular/router';
import { AddCourseComponent } from './course/add-
course.component';
import { CoursesComponent } from
'./course/courses.component';
```

```
import { EditCourseComponent } from './course/edit-
course.component';

const routes: Routes = [
  {path: 'courses/add', component:AddCourseComponent},
  {path: 'courses', component: CoursesComponent},
  {path: 'course/:id', component:
EditCourseComponent},
  {path: '', redirectTo: '/courses', pathMatch:
'full'}
];

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

Assign 1 dataservice.ts

```
import { HttpClient, HttpHeaders } from
'@angular/common/http';
import { Injectable } from '@angular/core';
import { map, Observable, Subject } from 'rxjs';
import { Course } from '../shared/course';

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

  apiUrl = 'http://localhost:5116/api/'

  httpOptions ={
    headers: new HttpHeaders({
```

```
      ContentType: 'application/json'
    })
  }

  constructor(private httpClient: HttpClient) {
  }

  getCourse(courseId: number) {
    return
this.httpClient.get(`${this.apiUrl}Course/GetCourse` +
"/" + courseId)
    .pipe(map(result => result))
  }

  getCourses(): Observable<any>{
    return
this.httpClient.get(`${this.apiUrl}Course/GetAllCourse
s`)
    .pipe(map(result => result))
  }

  addCourse(course: Course)
  {
    return
this.httpClient.post(`${this.apiUrl}Course/AddCourse`,
course, this.httpOptions)
  }

  deleteCourse(courseId: Number)
  {
    return
this.httpClient.delete<string>(`${this.apiUrl}Course/D
eleteCourse` + "/" + courseId, this.httpOptions)
  }
```

```
  editCourse(courseId: number, course: Course)
  {
    return
this.httpClient.put(`${this.apiUrl}Course/EditCourse/$
{courseId}`,course, this.httpOptions)
  }

}
```

Assignment 2

Order.html

```
<ion-item-group class="borderBottom">
  <ion-item lines="none" class="dashedBorderBottom">
    <ion-label>
      {{order?.restaurant?.name}}
      <p>{{order?.restaurant?.address}}</p>
      <ion-text>
        <p>
          R{{order?.grandTotal}}
          <ion-icon name="chevron-forward-
outline"></ion-icon>
        </p>
      </ion-text>
    </ion-label>
    <ion-note slot="end" color="dark">
      {{order?.status}}
      <ion-icon name="checkmark-circle-outline"
color="success"></ion-icon>
    </ion-note>
  </ion-item>
  <ion-item lines="none">
    <ion-label>
      <ion-text color="dark" *ngFor="let item of
order?.order">
        <p>{{item?.name}} x {{item?.quantity}}</p>
      </ion-text>
      <p class="time">{{order?.time}}</p>
      <ion-row>
        <ion-col size="6">
          <ion-button fill="outline" expand="block"
color="primary" (click)="reorderItem()">REORDER</ion-
button>
        </ion-col>
```

```
        <ion-col size="6">
          <ion-button fill="outline" expand="block"
color="dark" (click)="getHelp()">GET HELP</ion-button>
        </ion-col>
      </ion-row>
    </ion-label>
  </ion-item>
</ion-item-group>
```

Order.ts

```
import { Component, OnInit, Input, Output,
EventEmitter } from '@angular/core';
import { Order } from 'src/app/models/order.model';

@Component({
  selector: 'app-orders',
  templateUrl: './orders.component.html',
  styleUrls: ['./orders.component.scss'],
})
export class OrdersComponent implements OnInit {

  @Input() order: Order;
  @Output() reorder: EventEmitter<any> = new
EventEmitter();
  @Output() help: EventEmitter<any> = new
EventEmitter();

  constructor() {}

  ngOnInit() {}

  reorderItem() {
    this.reorder.emit(this.order);
  }
```

```
  getHelp() {
    this.help.emit(this.order);
  }

}
```

Restaurant.html

```
<ion-item lines="none">
  <ion-thumbnail slot="start">
    <img [src]="restaurant?.cover ? restaurant.cover :
'assets/imgs/1.jpg'" />
  </ion-thumbnail>
  <ion-label>
    <h4>{{restaurant?.name}}</h4>
    <ion-text color="medium">
      <p class="pStyle">
        {{getCuisine(restaurant?.cuisines)}}
      </p>
    </ion-text>
    <span>
      <ion-icon name="star"></ion-icon>
      {{restaurant?.rating}} .
    </span>
    {{restaurant?.delivery_time}} mins .
R{{restaurant?.price}} for two
    <ion-text color="tertiary"
*ngIf="restaurant?.distance && restaurant?.distance !=
0">
      <p class="distance">
        {{restaurant?.distance | number: '0.0-2'}} kms
away
      </p>
    </ion-text>
```

```
  </ion-label>
</ion-item>
```

Assignment 3

Login.html

```
<div class="login-wrapper" fxLayout="row"
fxLayoutAlign="center center">
    <mat-card class="box" *ngIf="!isLoading">
      <mat-card-header>
        <mat-card-title>Log in</mat-card-title>
      </mat-card-header>
      <form class="form" [formGroup]="loginFormGroup">
        <mat-card-content>
          <mat-form-field class="full-width">
            <mat-label>Username</mat-label>
            <input matInput placeholder="Enter the
User's Email address" formControlName="emailaddress">
          </mat-form-field>
          <mat-form-field class="full-width">
            <mat-label>Password</mat-label>
            <input matInput type="password"
placeholder="Enter the User's Password"
formControlName="password">
          </mat-form-field>
        </mat-card-content>
        <button mat-stroked-button color="primary"
class="btn-block" (click)="LoginUser()">Log
in</button>
        <div>Don't have an account? Register <a
[routerLink]="['../register']">here</a></div>
      </form>
    </mat-card>
```

```
    <mat-progress-spinner mode="indeterminate"
value="50" *ngIf="isLoading">
    </mat-progress-spinner>
  </div>
```

Register.html

```
<div class="login-wrapper" fxLayout="row"
fxLayoutAlign="center center">
    <mat-card class="box">
      <mat-card-header>
        <mat-card-title>Register</mat-card-title>
      </mat-card-header>
      <form class="form"
[formGroup]="registerFormGroup">
        <mat-card-content>
          <mat-form-field class="full-width">
            <mat-label>Email Address</mat-label>
            <input matInput placeholder="Enter a valid
Email address" formControlName="emailaddress">
          </mat-form-field>
          <mat-form-field class="full-width">
              <mat-label>Password</mat-label>
            <input type="password" matInput
placeholder="Enter between 6 to 16 characters"
formControlName="password">
          </mat-form-field>
        </mat-card-content>
        <button mat-stroked-button color="primary"
class="btn-block"
(click)="RegisterUser()">Register</button>
      </form>
    </mat-card>
  </div>
```

Login.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { APIService } from '../services/api.service';
import { FormBuilder, FormGroup, Validators } from
'@angular/forms';
import { MatSnackBar } from '@angular/material/snack-
bar';
import { HttpErrorResponse } from
'@angular/common/http';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})
export class LoginComponent implements OnInit {

  loginFormGroup: FormGroup = this.fb.group({
    emailaddress: ['', [Validators.required,
Validators.email]],
    password: ['', Validators.required],
  })

  isLoading:boolean = false

  constructor(private router: Router, private
apiService: APIService, private fb: FormBuilder,
private snackBar: MatSnackBar) { }

  ngOnInit(): void {
  }

  async LoginUser(){
```

```
    if(this.loginFormGroup.valid)
    {
      this.isLoading = true

      await
this.apiService.LoginUser(this.loginFormGroup.value).s
ubscribe(result => {
        localStorage.setItem('User',
JSON.stringify(result))
        this.loginFormGroup.reset();
        this.router.navigateByUrl('productListing');
      })
    }
  }

}
```

Register.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from
'@angular/forms';
import { Router } from '@angular/router';
import { APIService } from '../services/api.service';
import { MatSnackBar } from '@angular/material/snack-
bar';
import { HttpErrorResponse } from
'@angular/common/http';

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.scss']
```

```
})
export class RegisterComponent implements OnInit {

  registerFormGroup: FormGroup = this.fb.group({
    emailaddress: ['', [Validators.required,
Validators.email]],
    password: ['', [Validators.required,
Validators.minLength(6), Validators.maxLength(16)]],
  })

  constructor(private router: Router, private
apiService: APIService, private fb: FormBuilder,
private snackBar: MatSnackBar) {


  }

  ngOnInit(): void {
  }

  RegisterUser(){

    if(this.registerFormGroup.valid)
    {
        this.apiService.RegisterUser(this.registerForm
Group.value).subscribe(() => {
        this.registerFormGroup.reset();
        this.router.navigate(['']).then((navigated:
boolean) => {
          if(navigated) {
            this.snackBar.open(`Registered
successfully`, 'X', {duration: 5000});
          }
       });
```

```
      })
    }

    // if(this.registerFormGroup.valid)
    // {
    //   this.apiService.RegisterUser(this.registerFor
mGroup.value).subscribe(() => {
    //     this.registerFormGroup.reset();
    //     this.router.navigate(['']).then((navigated:
boolean) => {
    //       if(navigated) {
    //         this.snackBar.open(`Registered
successfully`, 'X', {duration: 5000});
    //       }
    //    });
    //   }, (response: HttpErrorResponse) => {
    //     if (response.status === 403) {
    //       this.snackBar.open(response.error, 'X',
{duration: 5000});
    //     }
    //     if (response.status === 500){
    //       this.snackBar.open(response.error, 'X',
{duration: 5000});
    //     }
    //   })
    // }
  }

}
```

Addproduct.html

```
<div class="login-wrapper" fxLayout="row"
fxLayoutAlign="center center">
    <mat-card class="box" >
```

```
      <mat-card-header>
        <mat-card-title>Add Product</mat-card-title>
      </mat-card-header>
      <form class="form" [formGroup]="productForm"
(submit)="onSubmit()" enctype="multipart/form-data">
        <mat-card-content>
          <mat-form-field class="full-width">
            <mat-label>Name</mat-label>
            <input matInput placeholder="Enter the
Product Name" formControlName="name">
          </mat-form-field>
          <mat-form-field class="full-width">
            <mat-label>Price</mat-label>
            <input type="number" step="0.01" matInput
placeholder="Enter the Product Price"
formControlName="price">
          </mat-form-field>
          <mat-form-field>
            <mat-label>Brand</mat-label>
            <mat-select formControlName="brand">
                <mat-option *ngFor="let item of
brandsData"  [value]="item.brandId">{{item.name}}</mat
-option>
            </mat-select>
        </mat-form-field>
        <mat-form-field>
            <mat-label>Product Type</mat-label>
            <mat-select formControlName="producttype"
>
                <mat-option *ngFor="let item of
productTypesData"  [value]="item.productTypeId">{{item
.name}}</mat-option>
            </mat-select>
        </mat-form-field>
```

```
        <mat-form-field class="full-width">
            <mat-label>Product Description</mat-label>
            <textarea matInput
formControlName="description"> </textarea>
          </mat-form-field>
          <div>
            <input formControlName="file" type="file"
id="file" #file placeholder="Choose file"
(change)="uploadFile(file.files)"
style="display:none;">
            <button mat-stroked-button color="primary"
(click)="file.click()">Upload File</button>
            <span *ngIf="fileNameUploaded.length > 0">
{{fileNameUploaded}}</span>
          </div>
        </mat-card-content>
        <button mat-stroked-button color="primary"
class="btn-block" >Submit</button>

      </form>
    </mat-card>
  </div>
```

Addproduct.ts

```
import { Component, OnInit } from '@angular/core';
import { APIService } from '../services/api.service';
import { FormBuilder, FormGroup, Validators } from
'@angular/forms';
import { Router } from '@angular/router';
import { MatSnackBar } from '@angular/material/snack-
bar';
import { Brands } from '../shared/brands';
import { ProductTypes } from '../shared/product-
types';
```

```
@Component({
  selector: 'app-add-products',
  templateUrl: './add-products.component.html',
  styleUrls: ['./add-products.component.scss']
})
export class AddProductsComponent implements OnInit {
   formData = new FormData();
   brandsData:Brands[]=[]
   productTypesData:ProductTypes[]=[]
   fileNameUploaded = ''

  productForm: FormGroup = this.fb.group({
    name: ['', Validators.required],
    file: ['', Validators.required],
    price: ['', Validators.required],
    brand: [null, Validators.required],
    producttype: [null, Validators.required],
    description: ['', Validators.required]
  })

  constructor(private apiService: APIService, private
fb: FormBuilder, private router: Router, private
snackBar: MatSnackBar) { }

  ngOnInit(): void {
    this.GetBrands()
    this.GetProductTypes()
  }

GetBrands(){
  this.apiService.getBrands().subscribe(result => {
    let brandList:any[] = result
```

```
    brandList.forEach((element) => {
      this.brandsData.push(element)
    });
  });
}

GetProductTypes(){
   this.apiService.getProductTypes().subscribe(result
=> {
    let productTypeList:any[] = result
    productTypeList.forEach((element) => {
      this.productTypesData.push(element)
    });
  });
}

  uploadFile = (files: any) => {
    let fileToUpload = <File>files[0];
    this.formData.append('file', fileToUpload,
fileToUpload.name);
    this.fileNameUploaded = fileToUpload.name
  }

  onSubmit() {
    if(this.productForm.valid)
    {
      this.formData.append('name',
this.productForm.get('name')!.value);
      this.formData.append('price',
this.productForm.get('price')!.value);
      this.formData.append('description',
this.productForm.get('description')!.value);
      this.formData.append('brand',
this.productForm.get('brand')!.value);
```

```
      this.formData.append('producttype',
this.productForm.get('producttype')!.value);

      this.apiService.addProduct(this.formData).subscr
ibe(() => {
        this.clearData()
        this.router.navigateByUrl('productListing').th
en((navigated: boolean) => {
          if(navigated) {
            this.snackBar.open(this.productForm.get('n
ame')!.value + ` created successfully`, 'X',
{duration: 5000});
          }
       });
      });
    }
  }

  clearData(){
    this.formData.delete("file");
    this.formData.delete("name");
    this.formData.delete("price");
    this.formData.delete("description");
    this.formData.delete("brand");
    this.formData.delete("producttype");
  }

}
```

Product listing.html

```
<div class="mat-elevation-z8">
    <mat-form-field appearance="fill">
        <mat-label>Filter</mat-label>
```

```
        <input matInput (keyup)="applyFilter($event)"
placeholder="start typing..." #input>
    </mat-form-field>

    <table mat-table [dataSource]="dataSource"
matSort>

        <ng-container matColumnDef="image">
            <th mat-header-cell
*matHeaderCellDef></th>
            <td mat-cell *matCellDef="let
element">  <img
src='{{element.image}}'>  </td>
          </ng-container>

      <ng-container matColumnDef="name">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> Name </th>
        <td mat-cell *matCellDef="let element">
{{element.name}} </td>
      </ng-container>

      <ng-container matColumnDef="price">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> Price </th>
        <td mat-cell *matCellDef="let element">
{{element.price | currency:'R':'symbol':'1.2-2'}}
</td>
      </ng-container>

      <ng-container matColumnDef="brand">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> Brand </th>
```

```
        <td mat-cell *matCellDef="let element">
{{element.brandName}} </td>
      </ng-container>

      <ng-container matColumnDef="productTypeName">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> Product Type </th>
        <td mat-cell *matCellDef="let element">
{{element.productTypeName}} </td>
      </ng-container>

      <ng-container matColumnDef="description">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> Description </th>
        <td mat-cell *matCellDef="let element">
{{element.description}} </td>
      </ng-container>

      <tr mat-header-row
*matHeaderRowDef="displayedColumns"></tr>
      <tr mat-row class="mat-row" *matRowDef="let row;
columns: displayedColumns;"></tr>

      <tr class="mat-row" *matNoDataRow>
        <td class="mat-cell" colspan="4">No data
matching the filter "{{input.value}}"</td>
      </tr>
    </table>
    <mat-paginator [pageSize]="10"
[pageSizeOptions]="[3, 5, 10]" showFirstLastButtons>
</mat-paginator>
  </div>
```

Product-listing.ts

```
<div class="mat-elevation-z8">
    <mat-form-field appearance="fill">
        <mat-label>Filter</mat-label>
        <input matInput (keyup)="applyFilter($event)"
placeholder="start typing..." #input>
    </mat-form-field>

    <table mat-table [dataSource]="dataSource"
matSort>

        <ng-container matColumnDef="image">
            <th mat-header-cell
*matHeaderCellDef></th>
            <td mat-cell *matCellDef="let
element">  <img
src='{{element.image}}'>  </td>
          </ng-container>

      <ng-container matColumnDef="name">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> Name </th>
        <td mat-cell *matCellDef="let element">
{{element.name}} </td>
      </ng-container>

      <ng-container matColumnDef="price">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> Price </th>
        <td mat-cell *matCellDef="let element">
{{element.price | currency:'R':'symbol':'1.2-2'}}
</td>
      </ng-container>
```

```
      <ng-container matColumnDef="brand">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> Brand </th>
        <td mat-cell *matCellDef="let element">
{{element.brandName}} </td>
      </ng-container>

      <ng-container matColumnDef="productTypeName">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> Product Type </th>
        <td mat-cell *matCellDef="let element">
{{element.productTypeName}} </td>
      </ng-container>

      <ng-container matColumnDef="description">
        <th mat-header-cell *matHeaderCellDef mat-
sort-header> Description </th>
        <td mat-cell *matCellDef="let element">
{{element.description}} </td>
      </ng-container>

      <tr mat-header-row
*matHeaderRowDef="displayedColumns"></tr>
      <tr mat-row class="mat-row" *matRowDef="let row;
columns: displayedColumns;"></tr>

      <tr class="mat-row" *matNoDataRow>
        <td class="mat-cell" colspan="4">No data
matching the filter "{{input.value}}"</td>
      </tr>
    </table>
    <mat-paginator [pageSize]="10"
[pageSizeOptions]="[3, 5, 10]" showFirstLastButtons>
</mat-paginator>
```

```
  </div>
```

Assign 3 data service

```
import { Injectable } from '@angular/core';
import { Observable, map, of} from 'rxjs';
import { HttpClient, HttpHeaders } from
'@angular/common/http';
import { RegisterUser } from '../shared/register-
user';
import { LoginUser } from '../shared/login-user';
import { User } from '../shared/user';
import { Product } from '../shared/product';

@Injectable({
  providedIn: 'root'
})

export class APIService {

apiUrl = 'http://localhost:5240/api/'

httpOptions ={
  headers: new HttpHeaders({
    ContentType: 'application/json'
  })
}
  constructor(private httpClient: HttpClient) {
  }

  RegisterUser(registerUser: RegisterUser){
    return
this.httpClient.post(`${this.apiUrl}Authentication/Reg
ister`, registerUser, this.httpOptions)
  }
```

```
  getProducts() {
    return
this.httpClient.get(`${this.apiUrl}Store/ProductListin
g`)
    .pipe(map(result => result))
  }

  LoginUser(loginUser: LoginUser){
    return
this.httpClient.post<User>(`${this.apiUrl}Authenticati
on/Login`, loginUser, this.httpOptions)
  }

  addProduct(file:FormData){

    return
this.httpClient.post(`${this.apiUrl}Store/AddProduct`,
file)
  }

  getBrands(): Observable<any>
  {
    return
this.httpClient.get(`${this.apiUrl}Store/Brands`)
    .pipe(map(result => result))
  }

  getProductTypes(): Observable<any>
  {
    return
this.httpClient.get(`${this.apiUrl}Store/ProductTypes`
)
    .pipe(map(result => result))
  }
```

```
}
```

Assign 3 app-routing.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from
'@angular/router';
import { LoginComponent } from
'./authentication/login.component';
import { RegisterComponent } from
'./authentication/register.component';
import { ProductListingComponent } from
'./products/product-listing.component';
import { AddProductsComponent } from './products/add-
products.component';

const routes: Routes = [
  {path: 'login', component:LoginComponent},
  {path: 'register', component:RegisterComponent},
  // {path: 'hero/:id', component:HeroDetailsComponent
},
  {path:'productListing',
component:ProductListingComponent},
  {path:'addProduct', component:AddProductsComponent},
  {path: '', redirectTo: 'login', pathMatch:'full'}
];

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

app.component.ts

```
import { Component,AfterContentChecked, ViewChild }
from '@angular/core';
import { MatSidenav } from
'@angular/material/sidenav';
import { Router } from '@angular/router';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent implements
AfterContentChecked {
  @ViewChild('sidenav', {static:true}) sidenav!:
MatSidenav;

  isLoggedIn = false;

  constructor(private router: Router) {}

  toggleSidenav(){
    this.sidenav.toggle();
  }

  ngAfterContentChecked(){
    if(localStorage.getItem('User'))
    {
      this.isLoggedIn = true;
    }
    else{
      this.isLoggedIn = false;
    }
  }
```

```
  logout(){
    if(localStorage.getItem('User'))
    {
      localStorage.removeItem('User')
      this.router.navigateByUrl('login');
    }
  }

}
```
