LP 02: Architecture with TypeScript
ARCHITECTURE API
CONTROLLERS
COURSE CONTROLLER.CS

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

namespace Architecture_API.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.");
            }
        }
    }
}
```

WEATHERFORCAST.CS

```
using Microsoft.AspNetCore.Mvc;

namespace Architecture_API.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class WeatherForecastController : ControllerBase
```

```
    {
        private static readonly string[] Summaries = new[]
        {
        "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy",
"Hot", "Sweltering", "Scorching"
    };

        private readonly ILogger<WeatherForecastController> _logger;

        public WeatherForecastController(ILogger<WeatherForecastController>
logger)
        {
            _logger = logger;
        }

        [HttpGet(Name = "GetWeatherForecast")]
        public IEnumerable<WeatherForecast> Get()
        {
            return Enumerable.Range(1, 5).Select(index => new WeatherForecast
            {
                Date = DateTime.Now.AddDays(index),
                TemperatureC = Random.Shared.Next(-20, 55),
                Summary = Summaries[Random.Shared.Next(Summaries.Length)]
            })
            .ToArray();
        }
    }
}
```

MIGRATIONS
20240225175319_INITIAL.CS

```
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Architecture_API.Migrations
{
    public partial class initial : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Courses",
                columns: table => new
                {
                    CourseId = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
```

```
                    Name = table.Column<string>(type: "nvarchar(max)",
nullable: false),
                    Description = table.Column<string>(type: "nvarchar(max)",
nullable: false),
                    Duration = table.Column<string>(type: "nvarchar(max)",
nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Courses", x => x.CourseId);
                });

            migrationBuilder.InsertData(
                table: "Courses",
                columns: new[] { "CourseId", "Description", "Duration", "Name"
},
                values: new object[,]
                {
                    { 1, "Year 1, Semester 1. Academic Information
Management", "Semester", "AIM101" },
                    { 2, "Year 1, Semester 2. Academic Literacy for IT",
"Semester", "ALL121" },
                    { 3, "Year 1. Systems Analysis and Design", "Year",
"INF171" },
                    { 4, "Year 2. Systems Analysis and Design", "Year",
"INF271" },
                    { 5, "Year 2. Programming", "Year", "INF272" },
                    { 6, "Year 2, Semester 1. Databases", "Semester", "INF214"
},
                    { 7, "Year 3, Semester 1. Programming Management",
"Semester", "INF315" },
                    { 8, "Year 3, Semester 2. IT Trends", "Semester", "INF324"
},
                    { 9, "Year 3, Semester 1. Programming", "Semester",
"INF354" },
                    { 10, "Year 3. Project", "Year", "INF370" }
                });
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Courses");
        }
    }
}
```

MODELS
APPDBCONTEXT.CS

```
using Microsoft.EntityFrameworkCore;

namespace Architecture_API.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"
                }
            );

            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"
                }
            );
        }
    }
}
```

COURSE.CS

```
using System.ComponentModel.DataAnnotations;

namespace Architecture_API.Models
{
    public class Course
    {
        [Key]
        public int CourseId { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public string Duration { get; set; }

    }
}
```

COURSEREPOSITORY.CS

```
using Microsoft.EntityFrameworkCore;
using System;

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

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

ICOURSEREPOSITORY.CS

```
namespace Architecture_API.Models
{
    public interface ICourseRepository
    {
        // Course
        Task<Course[]> GetAllCourseAsync();

    }
}
```

ADDITIONAL

PROGRAM.CS

```
using Architecture_API.Models;
using Microsoft.EntityFrameworkCore;
using System;

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("DefaultConnect
ion")));
builder.Services.AddScoped<ICourseRepository, CourseRepository>();

var app = builder.Build();

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

app.UseHttpsRedirection();

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

app.MapControllers();

app.Run();
```

ARCHITECTURE ANGULAR
APP
COURSE.COMPONENT.TS

```
import { Component, OnInit } from '@angular/core';
import { Course } from '../../shared/course';
import { DataService } from '../../services/data.service';


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

```
export class CoursesComponent implements OnInit {
  courses:Course[] = []

  constructor(private dataService: DataService) { }

  ngOnInit(): void {
    this.GetCourses()
    console.log(this.courses)
  }

  GetCourses()
  {
    this.dataService.GetCourses().subscribe(result => {
      let courseList:any[] = result
      courseList.forEach((element) => {
        this.courses.push(element)
      });
    })
  }
}
```

SERVICES

DATA.SERVICES.TS

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

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

  apiUrl = 'https://localhost:7049/api/'

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

  constructor(private httpClient: HttpClient) {
  }

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

}
```

APP.COMPONENT.TS

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { CoursesComponent } from './course/courses/courses.component';
import { ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent,
    CoursesComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

LP 03: API 1

API

• An API (Application Programming Interface) connects two or more applications. It allows two applications to communicate with one another.

• An API making use of HTTP is called a Web API.

Server ← API → Client ←GUI→ User

Different types of Web APIs

Remote Procedure Call, RPC.

• Clients can call functions on the server.

Remote Method Invocation, RMI.

• Clients can call methods on objects on the server.

Representational State Transfer, REST.

• Clients can apply CRUD operations on resources on the server.

Representational State Transfer (REST)

• A style of software architecture for distributed hypermedia systems such as the World Wide Web.

• Introduced in the doctoral dissertation of Roy Fielding

• One of the principal authors of the HTTP specification.

• Rest is a set of principles that define how web services should be designed and interact with each other.

• A collection of network architecture principles that outline how resources are defined and addressed

Representational State Transfer (REST)

• Rest, emphasizes a stateless client-server architecture.
• This principle means that the server does not store any state about the client session on the server side.
• Instead, each request from the client to the server must contain all the information necessary to understand and complete the request.
• Adhering to a stateless communication model, RESTful services aim to enhance performance, reliability, and scalability while ensuring a uniform and standardized web services architecture.

Representational State Transfer (REST)

• Implication of having REST as a stateless client-server
• Self-contained Requests
• Session State
• Scalability
• Caching
• Simplicity and Uniform Interface

REST - not a Standard

REST is not a standard

• JSR 311: JAX-RS: The JavaTM API for RESTful Web Services

But it uses several standards:

• HTTP
• URL
• XML/HTML/GIF/JPEG/etc (Resource Representations)
• text/xml, text/html, image/gif, image/jpeg, etc (Resource Types, MIME Types)

ASP.NET Core

ASP.NET Core provides a powerful and flexible platform for building restful APIs with its support for

• HTTP verbs
• Routing
• Model binding,
• Response types.

HTTP Verbs

• GET – (Read) Get information. GET requests must be safe and idempotent, which means that no matter how many times they are repeated with the same parameters, the results must be the same.
• POST – (Create) Instruct the resource at the URI to perform an action on the provided entity.
• PUT – (Update) Add an entity to a URI. PUT can be used to create a new entity or to update an existing one.
• PATCH - Update only the specified fields of a URI entity.
• DELETE - Request the removal of a resource; however, the resource does not have to be removed right away..

HTTP Verbs

Get: https://myapi.com/api/employees (Get all Employees)
Get: https://myapi.com/api/employees/{id} (Get single Employees by id)
POST: https://myapi.com/api/employees (Create an Employee)
PUT: https://myapi.com/api/employees /{id} (Update an Employees by id)
Delete: https://myapi.com/api/employees /{id} (Delete an Employee by id)

Web API Routing

• It routes incoming HTTP requests to a specific action method on a Web API controller.
• Attribute routing uses [Route()] attribute to define routes. The Route attribute can be applied on any controller or action method.

```
public class StudentController : ApiController
{
[Route("api/student/names")]
public IEnumerable<string> Get()
{
return new string[] { "student1", "student2" };
}
}
```

API ZAHIKES
CONTROLLER
PROVINCECONTROLLER.CS

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

namespace ZAHikes.API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ProvinceController : ControllerBase
    {

        private readonly ZAHikesDbContext dbContext;
        public ProvinceController(ZAHikesDbContext dbContext)
        {
            this.dbContext = dbContext;
        }

        [HttpGet]
        public IActionResult GetAll()
        {
            var province = dbContext.Provinces.ToList();


            return Ok(province);
        }
    }
}
```

STUDENTCONTROLLER.CS

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

namespace ZAHikes.API.Controllers
```

```
{

   // https://localhost:portnumber/api/students
    [Route("api/[controller]")]
    [ApiController]
    public class StudentController : ControllerBase
    {
        //GET:https://localhost:portnumber/api/students
        [HttpGet]
        public IActionResult GetALLStudents()
        {
            string[] studentNames = new string[] { "Joe", "Mbali", "Khanyi",
"Faith", "Sindi" };
            return Ok(studentNames);
        }
    }
}
```

DATA

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

namespace ZAHikes.API.Data
{
    public class ZAHikesDbContext:DbContext
    {
        public ZAHikesDbContext(DbContextOptions dbContextOptions):
base(dbContextOptions)
        //public ZAHikesDbContext(DbContextOptions<ZAHikesDbContext> options)
: base(options)
        {

        }
        public DbSet<Difficulty> Difficulties { get; set; }
        public DbSet<Province> Provinces { get; set; }
        public DbSet<Hike> Hikes { get; set; }
    }
}
```

MIGRATIONS
20240303213138_INITIAL MIGRATION.CS

```
using System;
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable
```

```
namespace ZAHikes.API.Migrations
{
    /// <inheritdoc />
    public partial class InitialMigration : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Difficulties",
                columns: table => new
                {
                    Id = table.Column<Guid>(type: "uniqueidentifier",
nullable: false),
                    Name = table.Column<string>(type: "nvarchar(max)",
nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Difficulties", x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "Provinces",
                columns: table => new
                {
                    Id = table.Column<Guid>(type: "uniqueidentifier",
nullable: false),
                    Code = table.Column<string>(type: "nvarchar(max)",
nullable: false),
                    Name = table.Column<string>(type: "nvarchar(max)",
nullable: false),
                    ProvinceImageUrl = table.Column<string>(type:
"nvarchar(max)", nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Provinces", x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "Hikes",
                columns: table => new
                {
                    Id = table.Column<Guid>(type: "uniqueidentifier",
nullable: false),
                    Name = table.Column<string>(type: "nvarchar(max)",
nullable: false),
```

```
                    Description = table.Column<string>(type: "nvarchar(max)",
nullable: false),
                    LengthInKm = table.Column<double>(type: "float", nullable:
false),
                    HikeImageUrl = table.Column<string>(type: "nvarchar(max)",
nullable: true),
                    DifficultyId = table.Column<Guid>(type:
"uniqueidentifier", nullable: false),
                    ProvinceId = table.Column<Guid>(type: "uniqueidentifier",
nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Hikes", x => x.Id);
                    table.ForeignKey(
                        name: "FK_Hikes_Difficulties_DifficultyId",
                        column: x => x.DifficultyId,
                        principalTable: "Difficulties",
                        principalColumn: "Id",
                        onDelete: ReferentialAction.Cascade);
                    table.ForeignKey(
                        name: "FK_Hikes_Provinces_ProvinceId",
                        column: x => x.ProvinceId,
                        principalTable: "Provinces",
                        principalColumn: "Id",
                        onDelete: ReferentialAction.Cascade);
                });

            migrationBuilder.CreateIndex(
                name: "IX_Hikes_DifficultyId",
                table: "Hikes",
                column: "DifficultyId");

            migrationBuilder.CreateIndex(
                name: "IX_Hikes_ProvinceId",
                table: "Hikes",
                column: "ProvinceId");
        }

        /// <inheritdoc />
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Hikes");

            migrationBuilder.DropTable(
                name: "Difficulties");
```

```
            migrationBuilder.DropTable(
                name: "Provinces");
        }
    }
}
```

MODELS

DIFFICULTY.CS

```
namespace ZAHikes.API.Models.Domain
{
    public class Difficulty
    {
        public Guid Id { get; set; }
        public string Name { get; set; }
    }
}
```

HIKES.CS

```
namespace ZAHikes.API.Models.Domain
{
    public class Hike
    {
        public Guid Id { get; set; }
        public string Name { get; set;}
        public string Description { get; set;}
        public double LengthInKm { get; set;}
        public string? HikeImageUrl { get; set;}
        public Guid DifficultyId { get; set;}
        public Guid ProvinceId { get; set;}

        //Navigation property
        public Difficulty Difficulty { get; set;}
        public Province Province { get; set;}
    }
}
```

PROVINCE.CS

```
namespace ZAHikes.API.Models.Domain
{
    public class Province
    {
        public Guid Id { get; set; }
        public string Code { get; set; }
        public string Name { get; set; }
        public string? ProvinceImageUrl { get; set; }
    }
```

LP 04: API II

API 2
CONTROLLER
CUSTOMERCONTROLLER.CS

```
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.");
        }


    }
}
```

TRIPCONTROLLER.CS

```
using APIII.Models;
using APIII.ViewModel;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

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

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

        [HttpPost]
        [Route("AddTripGuide/{guideNum}")]
        public async Task<IActionResult> AddTripGuide(string guideNum,
TripViewModel tvm)
        {
            var trip = new Trip { TripName = tvm.TripName, Startlocation =
tvm.Startlocation, State = tvm.State, Distance = tvm.Distance, MaxGroupSize =
tvm.MaxGroupSize, Type = tvm.Type, Season = tvm.Season };

            var guide = await _repository.GetGuideAsync(guideNum);
            if (guide == null) return NotFound($"The guide does not exist");

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

            return Ok();
        }

        [HttpPut]
        [Route("EditTripGuide/{guideNum}/{tripId}")]
        public async Task<IActionResult> EditTripGuide(string guideNum, int
tripId)
```

```
        {
            var trip = await _repository.GetTripAsync(tripId);
            if (trip == null) return NotFound($"The trip does not exist");

            var guide = await _repository.GetGuideAsync(guideNum);
            if (guide == null) return NotFound($"The guide does not exist");

            try
            {
                var guidesToRemove = trip.Guides.ToList();
                guidesToRemove.ForEach(g => trip.Guides.Remove(g));

                trip.Guides.Add(guide);
                await _repository.SaveChangesAsync();
            }
            catch (Exception)
            {
                return BadRequest("Invalid transaction");
            }

            return Ok();
        }
    }
}
```

MODELS

APPDBCONTEXT.CS

```
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;
    }
```

GUIDE.CS

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

namespace APIII.Models
{
    public class Guide
    {

        [StringLength(4)]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        [Key]
        public string GuideNum { get; set; }
        public string LastName { get; set; } = string.Empty;
        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)]
        [RegularExpression(@"^\d+$")]
        public string? PostalCode { get; set; }

        [StringLength(10)]
        public string PhoneNumber { get; set; } = string.Empty;
        [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}")]
        public DateTime HireDate { get; set; }

        public List<Trip> Trips { get; set; } = new List<Trip>();
    }
}
```

IREPOSITORY.CS

```
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);

    }
}
```

REPOSITORY.CS

```
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;
        }
    }
}
```

TABLEONE1TO1EX.CS

```
using System.ComponentModel.DataAnnotations;

namespace APIII.Models
{
    public class TableOne1to1Ex
    {
        [Key]
        public int TableOneId { get; set; }
        public string TableOneName { get; set; } = string.Empty;
        public string TableOneDescription { get; set; } = string.Empty;
    }
}
```

TABLEONE1TOMANYEX.CS

```
using System.ComponentModel.DataAnnotations;

namespace APIII.Models
{
    public class TableOne1toManyEx
    {
        [Key]
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }

        public List<TableTwo1toManyEx> TableTwo1toManyEx { get; set; } = new
List<TableTwo1toManyEx>();
    }
}
```

TABLETWO1TO1EX.CS

```
using System.ComponentModel.DataAnnotations;

namespace APIII.Models
{
    public class TableTwo1to1Ex
    {
        [Key]
        public int TableTwoId { get; set; }
        public string TableTwoName { get; set; } = string.Empty;
        public string TableTwoDescription { get; set; } = string.Empty;

        public TableOne1to1Ex TableOne1to1Ex { get; set; }
    }
}
```

TABLETWO1TOMANYEX.CS

```
using System.ComponentModel.DataAnnotations;

namespace APIII.Models
{
    public class TableTwo1toManyEx
    {
        [Key]
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
    }
}
```

TRIP.CS

```
using System.ComponentModel.DataAnnotations;

namespace APIII.Models
{
    public class Trip
    {
        [Key]
        public int TripId { get; set; }

        [MaxLength(75)]
        public string TripName { get; set;} = string.Empty;

        [MaxLength(50)]
        public string? Startlocation { get; set; }

        [StringLength(2)]
```

```
        public string? State { get; set; }

        [Range(0, 500)]
        public int Distance { get; set; }

        [Range(1,100)]
        public int MaxGroupSize { get; set; }

        [MaxLength(50)]
        public string? Type { get; set; }

        [MaxLength(50)]
        public string? Season { get; set; }

        public List<Guide> Guides { get; set; } = new List<Guide>();


    }
}
```

TRIPGUIDE2.CS

```
namespace APIII.Models
{
    public class TripGuide2
    {
        public DateTime DateConfirmed { get; set; }
    }
}
```

VIEWMODEL
CUSTOMERVIEWMODEL.CS

```
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;
    }
}
```

TRIPVIEWMODEL.CS

```
using System.ComponentModel.DataAnnotations;

namespace APIII.ViewModel
{
    public class TripViewModel
    {
        public string TripName { get; set; }

        public string? Startlocation { get; set; }

        public string? State { get; set; }
        public int Distance { get; set; }

        public int MaxGroupSize { get; set; }

        public string? Type { get; set; }

        public string? Season { get; set; }
    }
}
```

LP 05: Angular 1

Angular

- Angular is a framework for frontend web development
- With the aid of the Angular framework, you may develop sophisticated HTML web apps.
- It can be used on other frameworks such as Ionic to make native mobile application
- Responsive, user-friendly, scalable
- Developed by Google
- Angular CLI enables you to create things faster
- Angular enhances the UI experience for users
- Angular uses the concept of single page application (SPA)
- Angular code is written in TypeScript language

Features

- An XML or HTML page is treated by the document object model (DOM) as a tree structure, with each node denoting a distinct section of the content.
- Typescript - JavaScript users can write more understandable code by using typescript, which specifies a set of types. Any platform can easily support the smooth operation of the TypeScript code thanks to JavaScript compilation.
- The Angular framework provides a robust set of tools and components that allow developers to create scalable, maintainable, and efficient web applications.
- It includes features like
- two-way data binding,
- templating,
- RESTful API handling,
- modularization,
- AJAX handling,
- dependency injection,

• the use of directives to create reusable components.

Architecture
The architecture diagram identifies the eight main building blocks of an Angular application:
1. Module
2. Components
3. Templates
4. Metadata
5. Data binding
6. Directives
7. Services
8. Dependency injection

1. Module - A root module called AppModule serves as the application's bootstrap mechanism in an Angular application.
2. Components: A view is a portion of the screen that is controlled by a component.
The application logic of a component—what it does to assist the view—is specified in a class. Using a set of attributes and methods on the API, the class communicates with the view.
3. Templates: is a form of HTML that tells Angular how to render the component. The Angular template combines the Angular markup with HTML to modify HTML elements before they are displayed.
4. Metadata: Metadata tells Angular how to process a class. It is used to decorate the class to configure the expected behavior of a class. In Typescript, you attach metadata by using a decorator.
5. Data binding: supported by Angular is a method for coordinating parts of a template with parts of a component. It plays an important role in communication between a template and its component.
6. Directives are easily distinguished since they begin with the prefix "- ng." Consider their markers on the DOM element that tell Angular to alter the element entirely or to attach a specific behavior to it.
7. Services: Any value, feature, or function that your application requires falls under the wide category of service. A service can be almost anything. A service might be a function or value as well as a TypeScript class having a specific, well-defined purpose.
8. Dependency injection: A method for providing a new instance of a class with the dependencies it needs is called dependency injection. Dependencies are typically services. Dependency injection is a technique used by Angular to supply new components with the services they require. By examining the kinds of a component's constructor parameters, Angular may determine which services the component requires.

Data Binding
• Interpolation Binding: Interpolation is a procedure that allows the user to bind a value to the user interface element. Interpolation binds the data one way, which means that data moves in one direction from the components to HTML elements. It uses the “{{ }}" syntax for data binding.
• Property Binding: is a one-way data binding mechanism that allows you to set the properties for HTML elements. It uses the "[ ]" syntax for data binding.
• Event binding: type is when information flows from the view to the component when an event is triggered. The event could be a mouse click or keypress. It uses the “( )" syntax for data binding.
• Two-way Data Binding
• is a mechanism where data flows from the component to the view and back. This binding ensures that the component and view are always in sync. It uses the “[( )]" syntax for data binding.

Pipes

• Angular Pipes transform the output. You can think of them as makeup rooms where they beautify the data into a more desirable format. They do not alter the data but change how they appear to the user.
• Pipes are defined using the pipe "|" symbol.
• Some commonly used predefined Angular pipes are:
• DatePipe: Formats a date value.
• UpperCasePipe: Transforms text to uppercase.
• LowerCasePipe: Transforms text to lowercase.
• CurrencyPipe: Transforms a number to the currency string.
• PercentPipe: Transforms a number to the percentage string.
• DecimalPipe: Transforms a number into a decimal point string

TASKMANAGER CODE

DASHBOARD
DASHBOARD.COMPONENT.HTML

```
<nav>
    <ol class="breadcrumb">
      <li class="breadcrumb-item"><a routerLink="/">Home</a></li>
      <li class="breadcrumb-item active">Dashboard</li>
    </ol>
  </nav>
  <h5>Dashboard</h5>
  <h6>{{currentDate}}</h6>
  <div class="row">
    <div class="col-lg-3 pb-3" style="background-color: #e9e6e6">
      <div class="row">
        <div
          class="col-11 text-white text-center mx-auto rounded pt-2 pb-2 font-
weight-bold"
          style="
            background-color: #a39e9e;
            font-family: 'Arial Narrow Bold', sans-serif;
          "
        >
          {{ Designation}}
        </div>
        <div class="col-12 text-center mt-2">
          <img src="assets/person.png" width="120px" />
        </div>
        <div
          class="col-12 text-center pt-2 pb-2 font-weight-bold"
          style="font-family: Tahoma"
        >
          {{ Username}}
        </div>
        <div class="col-12 text-center pt-2 pb-2" style="font-family: Arial">
          TEAM SUMMARY
        </div>
```

```
        <div class="col-12 mb-3">
          <ul class="list-group">
            <li
              class="list-group-item d-flex justify-content-between align-
items-center"
            >
              NO. OF TEAM MEMBERS
            <span>{{ NoOfTeamMembers }}
            </span>
            </li>
            <li
              class="list-group-item d-flex justify-content-between align-
items-center"
            >
              TOTAL COST OF ALL PROJECTS
              <span>R {{ TotalCostOfAllProjects }} k</span>
            </li>
            <li
              class="list-group-item d-flex justify-content-between align-
items-center"
            >
              PENDING TASKS
              <span>{{ PendingTasks }}</span>
            </li>
            <li
              class="list-group-item d-flex justify-content-between align-
items-center"
            >
              UPCOMING PROJECTS
              <span>{{ UpComingProjects}}</span>
            </li>
          </ul>
        </div>
        <div
          class="col-12 text-center pt-2 pb-2"
          style="font-family: 'Arial Narrow'"
        >
          CLIENTS
        </div>
        <div class="col-12">
          <ul class="list-group list-group-flush">
            <li class="list-group-item">BMW IT HUB</li>
            <li class="list-group-item">MOYO</li>
            <li class="list-group-item">DERIVCO</li>
          </ul>
        </div>
      </div>
    </div>
```

```
    <div class="col-lg-6">
      <div class="row">
        <div class="col-12 pt-0 pb-2" style="background-color: #e9e6e6">
          <div class="row mt-2">
            <div class="col-6 text-left">
              <div class="dropdown">
                <button
                  class="btn btn-secondary dropdown-toggle"
                  type="button"
                  id="dropdownMenuButton1"
                  data-toggle="dropdown"
                >
                  Project A
                </button>
                <div class="dropdown-menu">
                  <a class="dropdown-item" href="#">Project A</a>
                  <a class="dropdown-item" href="#">Project B</a>
                  <a class="dropdown-item" href="#">Project C</a>
                  <a class="dropdown-item" href="#">Project D</a>
                </div>
              </div>
            </div>
            <div class="col-6 text-right">
              <div class="dropdown">
                <button
                  class="btn btn-secondary dropdown-toggle"
                  type="button"
                  id="dropdownMenuButton2"
                  data-toggle="dropdown"
                >
                  2024
                </button>
                <div class="dropdown-menu">
                  <a class="dropdown-item" href="#">2023</a>
                  <a class="dropdown-item" href="#">2022</a>
                  <a class="dropdown-item" href="#">2021</a>
                  <a class="dropdown-item" href="#">2020</a>
                </div>
              </div>
            </div>
          </div>
        </div>
        <div
          class="col-11 mx-auto mt-1 text-white text-center pt-2 pb-2 mx-auto
rounded font-weight-bold"
          style="background-color: #718d97; font-family: 'Arial Narrow'"
        >
          PROJECT BRIEFING
```

```
        </div>
        <div class="col-12">
          <table class="table table-borderless">
            <tr>
              <td>Project Cost</td>
              <td>R{{ ProjectCost }}</td>
            </tr>
            <tr>
              <td>Current Expenditure</td>
              <td>R{{ CurrentExpenditure }}</td>
            </tr>
            <tr>
              <td>Available Funds</td>
              <td>R{{ AvailableFunds }}</td>
            </tr>
          </table>
        </div>
        <div
          class="col-11 mx-auto mt-1 text-white text-center pt-2 pb-2 mx-2
rounded font-weight-bold"
          style="background-color: #718d97; font-family: 'Arial Narrow'"
        >
          TEAM MEMBERS SUMMARY
        </div>
        <div class="col-12">
          <table class="table">
            <tr>
              <th>Region</th>
              <th>Team Members Count</th>
              <th>Temporarily Unavailable Members</th>
            </tr>
            <tr>
              <td>
                <b>Gauteng</b>
              </td>
              <td>20</td>
              <td>4</td>
            </tr>
            <tr>
              <td>
                <b>Western Cape</b>
              </td>
              <td>15</td>
              <td>8</td>
            </tr>
            <tr>
              <td>
                <b>KwaZuluNatal</b>
```

```
              </td>
              <td>17</td>
              <td>1</td>
            </tr>
            <tr>
              <td>
                <b>Limpopo</b>
              </td>
              <td>15</td>
              <td>6</td>
            </tr>
          </table>
        </div>
      </div>
    </div>
    <div class="col-lg-3" style="background-color: #e9e6e6">
      <div class="row">
        <div
          class="col-11 mx-auto mt-1 text-white text-center pt-1 pb-2 mx-2
rounded font-weight-bold"
          style="background-color: #718d97; font-family: 'Arial Narrow'"
        >
          TEAM MEMBERS
        </div>
        <div class="col-lg-12 pt-2">
          <div class="accordion" id="accordion1">
            <!--  gauteng starts -->
            <div class="card">
              <div class="card-header bg-secondary" id="card1">
                <h2 class="mb-0">
                  <button
                    class="btn btn-link text-white"
                    type="button"
                    data-toggle="collapse"
                    data-target="#cardbody1"
                  >
                    Gauteng
                  </button>
                </h2>
              </div>
              <div id="cardbody1" class="collapse show" data-
parent="#accordion1">
                <div class="card-body">
                  <table class="table">
                    <thead>
                      <tr>
                        <th>ID</th>
                        <th>Name</th>
```

```
                    <th>Status</th>
                  </tr>
                </thead>
                <tbody>
                  <tr>
                    <td>1</td>
                    <td>Ridewaan</td>
                    <td><i class="fa fa-phone"></i> Available</td>
                  </tr>
                  <tr>
                    <td>2</td>
                    <td>Dean</td>
                    <td><i class="fa fa-phone"></i> Available</td>
                  </tr>
                  <tr>
                    <td>3</td>
                    <td>Victoria</td>
                    <td><i class="fa fa-user-times"></i> Busy</td>
                  </tr>
                  <tr>
                    <td>4</td>
                    <td>Neo</td>
                    <td><i class="fa fa-user-times"></i> Busy</td>
                  </tr>
                </tbody>
              </table>
            </div>
          </div>
        </div>
        <!-- gauteng ends -->

        <!-- western cape starts -->
        <div class="card">
          <div class="card-header bg-secondary" id="card2">
            <h2 class="mb-0">
              <button
                class="btn btn-link text-white"
                type="button"
                data-toggle="collapse"
                data-target="#cardbody2"
              >
                Western Cape
              </button>
            </h2>
          </div>
          <div id="cardbody2" class="collapse" data-parent="#accordion1">
            <div class="card-body">
              <table class="table">
```

```
                <thead>
                  <tr>
                    <th>ID</th>
                    <th>Name</th>
                    <th>Status</th>
                  </tr>
                </thead>
                <tbody>
                  <tr>
                    <td>1</td>
                    <td>David</td>
                    <td><i class="fa fa-user-times"></i> Busy</td>
                  </tr>
                  <tr>
                    <td>2</td>
                    <td>Charles</td>
                    <td><i class="fa fa-phone"></i> Available</td>
                  </tr>
                  <tr>
                    <td>3</td>
                    <td>Geordin</td>
                    <td><i class="fa fa-user-times"></i> Busy</td>
                  </tr>
                  <tr>
                    <td>4</td>
                    <td>Michelle</td>
                    <td><i class="fa fa-phone"></i> Available</td>
                  </tr>
                </tbody>
              </table>
            </div>
          </div>
        </div>
        <!-- western cape ends -->

        <!-- kwazulunatal starts -->
        <div class="card">
          <div class="card-header bg-secondary" id="card3">
            <h2 class="mb-0">
              <button
                class="btn btn-link text-white"
                type="button"
                data-toggle="collapse"
                data-target="#cardbody3"
              >
                KwaZuluNatal
              </button>
            </h2>
```

```
            </div>
            <div id="cardbody3" class="collapse" data-parent="#accordion1">
              <div class="card-body">
                <table class="table">
                  <thead>
                    <tr>
                      <th>ID</th>
                      <th>Name</th>
                      <th>Status</th>
                    </tr>
                  </thead>
                  <tbody>
                    <tr>
                      <td>1</td>
                      <td>Thabiso</td>
                      <td><i class="fa fa-user-times"></i> Busy</td>
                    </tr>
                    <tr>
                      <td>2</td>
                      <td>Turner</td>
                      <td><i class="fa fa-phone"></i> Available</td>
                    </tr>
                    <tr>
                      <td>3</td>
                      <td>Debra</td>
                      <td><i class="fa fa-user-times"></i> Busy</td>
                    </tr>
                    <tr>
                      <td>4</td>
                      <td>Xolile</td>
                      <td><i class="fa fa-phone"></i> Available</td>
                    </tr>
                  </tbody>
                </table>
              </div>
            </div>
          </div>
          <!-- kwazulunatal ends -->

          <!-- limpopo starts -->
          <div class="card">
            <div class="card-header bg-secondary" id="card4">
              <h2 class="mb-0">
                <button
                  class="btn btn-link text-white"
                  type="button"
                  data-toggle="collapse"
                  data-target="#cardbody4"
```

```
                >
                  Limpopo
                </button>
              </h2>
            </div>
            <div id="cardbody4" class="collapse" data-parent="#accordion1">
              <div class="card-body">
                <table class="table">
                  <thead>
                    <tr>
                      <th>ID</th>
                      <th>Name</th>
                      <th>Status</th>
                    </tr>
                  </thead>
                  <tbody>
                    <tr>
                      <td>1</td>
                      <td>Mandisa</td>
                      <td><i class="fa fa-user-times"></i> Busy</td>
                    </tr>
                    <tr>
                      <td>2</td>
                      <td>James</td>
                      <td><i class="fa fa-phone"></i> Available</td>
                    </tr>
                    <tr>
                      <td>3</td>
                      <td>Taurayi</td>
                      <td><i class="fa fa-user-times"></i> Busy</td>
                    </tr>
                    <tr>
                      <td>4</td>
                      <td>Siya</td>
                      <td><i class="fa fa-phone"></i> Available</td>
                    </tr>
                  </tbody>
                </table>
              </div>
            </div>
          </div>
          <!-- limpopo ends -->
        </div>
      </div>
    </div>
  </div>
</div>
```

DASHBOARD.COMPONENT.TS

```
import { Component, OnInit } from '@angular/core';
import { TaskPipe } from '../task.pipe';

@Component({
  selector: 'app-dashboard',
  standalone: true,
  imports: [TaskPipe],
  templateUrl: './dashboard.component.html',
  styleUrl: './dashboard.component.scss'
})
export class DashboardComponent implements OnInit
{
  Designation: string = "";
  Username: string = "";
  NoOfTeamMembers: number = 0;
  TotalCostOfAllProjects: number = 0;
  PendingTasks: number = 0;
  UpComingProjects: number = 0;
  ProjectCost: number = 0;
  CurrentExpenditure: number = 0;
  AvailableFunds: number = 0;
  currentDate = new Date();


  Clients: string[] = [];
  Projects: string[] = [];
  Years: number[] = [];
  TeamMembersSummary: any = [];
  TeamMembers: any = [];

  ngOnInit()
  {
    this.Designation = 'Team Leader';
    this.Username = 'Tim Ade';
    this.NoOfTeamMembers = 50;
    this.TotalCostOfAllProjects = 240;
    this.PendingTasks = 15;
    this.UpComingProjects = 2;
    this.ProjectCost = 2113507;
    this.CurrentExpenditure = 96788;
    this.AvailableFunds = 52536;
  }

}
```

APP
APP.COMPONENT.HTML

```
<!-- Navbar -->
<nav class="navbar navbar-expand-sm bg-success navbar-dark">
  <a class="navbar-brand" href="#"> FGNR Task Manager </a>
  <!-- Faerie Glen Nature Reserve -->
  <button
    class="navbar-toggler"
    type="button"
    data-toggle="collapse"
    data-target="#mynav"
  >
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="mynav">
    <ul class="navbar-nav me-auto mb-2 mb-lg-0">
      <li class="nav-item">
        <a class="nav-link" routerLink="dashboard">Dashboard</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" routerLink="about">About</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" routerLink="contact us">Contact Us</a>
      </li>
    </ul>
    <form class="d-flex input-group w-auto">
      <input
          type="search"
          class="form-control"
          placeholder="Search"
          aria-label="Search"
        />

      <button class="btn btn-secondary my-sm-0" type="button">Search</button>
    </form>

  </div>

</nav>
<!-- Navbar -->

<div class="container-fluid">
```

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { RouterModule, RouterOutlet } from '@angular/router';
import { DashboardComponent } from "./dashboard/dashboard.component";
import { AboutComponent } from "./about/about.component";
import { TaskPipe } from './task.pipe';

@Component({
    selector: 'app-root',
    standalone: true,
    templateUrl: './app.component.html',
    styleUrl: './app.component.scss',
    imports: [RouterOutlet, RouterModule, DashboardComponent, AboutComponent,
TaskPipe]
})
export class AppComponent {
  title = 'TaskManager';
}
```

APP.CONFIG.SERVER.TS

```
import { mergeApplicationConfig, ApplicationConfig } from '@angular/core';
import { provideServerRendering } from '@angular/platform-server';
import { appConfig } from './app.config';

const serverConfig: ApplicationConfig = {
  providers: [
    provideServerRendering()
  ]
};

export const config = mergeApplicationConfig(appConfig, serverConfig);
```

APP.CONFIG.TS

```
import { ApplicationConfig } from '@angular/core';
import { provideRouter } from '@angular/router';

import { routes } from './app.routes';
import { provideClientHydration } from '@angular/platform-browser';

export const appConfig: ApplicationConfig = {
  providers: [provideRouter(routes), provideClientHydration()]
};
```

APP.ROUTE.TS

```
import { Routes } from '@angular/router';
import { DashboardComponent } from './dashboard/dashboard.component';
import { AboutComponent } from './about/about.component';
import { ContactUsComponent } from './contact-us/contact-us.component';

export const routes: Routes = [
    {path:'dashboard', component: DashboardComponent},
    {path:'about', component:AboutComponent},
    {path:'contact us', component:ContactUsComponent}
    // { path: '', redirectTo: 'dashboard', pathMatch: 'full' },
];
```

TASK.PIPE.TS

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'task',
  standalone: true
})
export class TaskPipe implements PipeTransform {

  transform(value: any, ...args: any[]): any {
    return null;
  }

}
```

LP 07: Angular 2

Angular Architecture

Install & Configuration

- Node.js (website)
- npm install -g typescript
- npm install -g @angular/cli
- ng add @angular/material
- npm install
- ng build
- npm start
- ng serve –o
- ng new myapp

Angular CLI

- https://angular.io/cli
- ng new testapp
- ng generate something [options]
- ng g component favourite-component
- ng g c favourites/favourite-component --flat
- ng g c favourite-component --flat -it -is --skip-Tests
- ng g [interface or module or service] etc…

ANGULAR 2 CODE

DASHBOARD
DASHBOARD.COMPONENT.HTML

```
<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>
```

DASHBOARD.COMPONENT.SCSS

```
.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;
}
```

DASHBOARD.COMPONENT.TS

```
import { Component } from '@angular/core';
import { MaterialModule } from '../shared/material.module';

@Component({
  selector: 'app-dashboard',
  standalone: true,
  imports: [MaterialModule],
  templateUrl: './dashboard.component.html',
  styleUrl: './dashboard.component.scss'
})
export class DashboardComponent {

}
```

HERO-DETAILS

HERO-DETAILS.COMPONENT.HTML

```
<div class="flex-container" fxLayout.xs="column">
    <div class="flex-item">Age: {{hero?.age}}</div>
    <div class="flex-item">Birthday: {{hero?.birthday}}</div>
    <div class="flex-item"></div>
    <div class="flex-item">Height: {{hero?.height}}</div>
    <div class="flex-item">Alive: {{hero?.alive}}</div>
    <div class="flex-item"></div>

</div>
<button mat-button [routerLink]="['/heroes']">
    <mat-icon>home</mat-icon>
    return
  </button>
```

HERO-DETAILS.COMPONENTS.TS

```
import { Component, OnInit } from '@angular/core';
import { HeroService } from '../../services/hero.service';
import { ActivatedRoute, RouterModule } from '@angular/router';
import { MaterialModule } from '../../shared/material.module';

@Component({
  selector: 'app-hero-details',
  standalone: true,
  imports: [MaterialModule, RouterModule],
  templateUrl: './hero-details.component.html',
  styleUrl: './hero-details.component.scss'
})
export class HeroDetailsComponent implements OnInit {
  hero:any
```

```
  constructor(private heroService: HeroService, private route: ActivatedRoute)
{ }

  ngOnInit(): void {
    this.heroService.getHero(+this.route.snapshot.params['id']).subscribe((her
o: any) => {this.hero = hero})
  }
}
```

HEROES
HEROES.COMPONENTS.HTML

```
<div class="mat-elevation-z8">
    <mat-form-field appearance="outline">
        <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>
```

HEROES.COMPONENT.SCSS

```
table {
    width: 100%;
}

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

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

HERO.COMPONENT.TS

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

@Component({
  selector: 'app-heroes',
  standalone: true,
  imports: [MaterialModule, RouterModule],
```

```
  templateUrl: './heroes.component.html',
  styleUrl: './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();
    });
  }
}
```

INPUT-OUTPUT EMMITER

CHILD.COMPONENT.HTML

```
<button mat-raised-button (click)="incrementCount()">
    Increment Counter (Current: {{ calculateTotal() }})
```

```
</button>
```

CHILD.COMPONENT.TS

```
import { Component, EventEmitter, Input, Output } from '@angular/core';

@Component({
  selector: 'app-child',
  standalone: true,
  imports: [],
  templateUrl: './child.component.html',
  styleUrl: './child.component.scss'
})
export class ChildComponent {
  @Input() initialCount: any;
  @Output() countChange = new EventEmitter<number>();

  currentCount: number = 0;

  incrementCount() {
    this.currentCount++;
    this.countChange.emit(this.calculateTotal());
  }

  calculateTotal(){
    if (this.currentCount <=1)
      return this.currentCount = this.initialCount + this.currentCount;
    else
      return this.currentCount;
  }
}
```

PARENT.COMPONENT.HTML

```
<div style="text-align: center">
    <h1>Parent Component Count: {{ parentCount }}</h1>
    <app-child [initialCount]="parentCount"
(countChange)="handleCountChange($event)"></app-child>
</div>
```

PARENT.COMPONENT.TS

```
import { Component } from '@angular/core';
import { ChildComponent } from './child.component';

@Component({
```

```
  selector: 'app-parent',
  standalone: true,
  imports: [ChildComponent],
  templateUrl: './parent.component.html',
  styleUrl: './parent.component.scss'
})
export class ParentComponent {
  parentCount: number = 5;

  handleCountChange(newCount: number) {
    this.parentCount = newCount;
  }
}
```

SERVICES

HERO.SERVICES.TS

```
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() {
  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))
    }
  }
}
```

SHARED

HERO.TS

```
import { Ihero } from "./ihero";

type allowNull = number | string | null;

export class Hero implements Ihero {
    id!: number;
    name!: string;
    age!: allowNull;
```

```
    birthday!: allowNull;
    height!: string;
    image!: string;
    alive!: boolean;
}
```

IHERO.TS

```
export interface Ihero {
    id: number;
    name: string;
    image: string;
}
```

APP.COMPONENT.HTML

```
<mat-toolbar color="primary">
  <mat-toolbar-row>
    <button mat-icon-button (click)="isSidenavOpen = !isSidenavOpen">
      <mat-icon>menu</mat-icon>
    </button>
    <h1>Angular II</h1>
  </mat-toolbar-row>

</mat-toolbar>
<mat-sidenav-container>
  <mat-sidenav #sidenav mode="side" [(opened)]="isSidenavOpen" class="side-
container">
    <mat-nav-list>

      <a mat-list-item [routerLink]="'/dashboard'"> Dashboard </a>
      <a mat-list-item [routerLink]="'/heroes'"> Heroes </a>
      <a mat-list-item [routerLink]="'/parent'"> Input/Output Emitter Parent
</a>

    </mat-nav-list>
  </mat-sidenav>
  <mat-sidenav-content class="content-container">
    <div style="height: 88vh;">
      <router-outlet></router-outlet>
    </div>
  </mat-sidenav-content>
</mat-sidenav-container>
```

APP.COMPONENT.TS

```
import { Component, ViewChild } from '@angular/core';
import { RouterModule, RouterOutlet } from '@angular/router';
import { MaterialModule } from './shared/material.module';
```

```
import { MatSidenav } from '@angular/material/sidenav';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet, RouterModule, MaterialModule],
  templateUrl: './app.component.html',
  styleUrl: './app.component.scss'
})
export class AppComponent {
  isSidenavOpen = true;
  title = 'AngularII';
}
```

APP.ROUTE.TS

```
import { Routes } from '@angular/router';
import { DashboardComponent } from './dashboard/dashboard.component';
import { HeroesComponent } from './heroes/heroes.component';
import { HeroDetailsComponent } from './heroes/hero-details/hero-
details.component';
import { ChildComponent } from './input-output-emitter/child.component';
import { ParentComponent } from './input-output-emitter/parent.component';

export const routes: Routes = [
  {path: 'parent', component:ParentComponent},
  {path: 'dashboard', component:DashboardComponent},
  {path: 'heroes', component:HeroesComponent},
  {path: 'hero/:id', component:HeroDetailsComponent },
  {path: '', redirectTo: '/dashboard', pathMatch:'full'}
];
```

ASSIGNMENT 1

BACKEND API

COURSECONTROLLER.CS

```
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")] //returns a list of courses
        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}")] //returns a specific course
        public async Task<IActionResult> GetCourseAsync(int courseId)
        {
            try
            {
                var results = await
_courseRepository.GetCourseAsync(courseId);

                if (results == null) return NotFound("Course does not exist");

                return Ok(results);
            }
            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");
        }
    }
}
```

MIGRATION
20230328121132_INITIAL.CS

```
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Architecture.Migrations
{
    public partial class initial : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Courses",
                columns: table => new
                {
                    CourseId = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
```

```
                    Name = table.Column<string>(type: "nvarchar(max)",
nullable: false),
                    Description = table.Column<string>(type: "nvarchar(max)",
nullable: false),
                    Duration = table.Column<string>(type: "nvarchar(max)",
nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Courses", x => x.CourseId);
                });

            migrationBuilder.InsertData(
                table: "Courses",
                columns: new[] { "CourseId", "Description", "Duration", "Name"
},
                values: new object[,]
                {
                    { 1, "Year 1, Semester 1. Academic Information
Management", "Semester", "AIM101" },
                    { 2, "Year 1, Semester 2. Academic Literacy for IT",
"Semester", "ALL121" },
                    { 3, "Year 1. Systems Analysis and Design", "Year",
"INF171" },
                    { 4, "Year 2. Systems Analysis and Design", "Year",
"INF271" },
                    { 5, "Year 2. Programming", "Year", "INF272" },
                    { 6, "Year 2, Semester 1. Databases", "Semester", "INF214"
},
                    { 7, "Year 3, Semester 1. Programming Management",
"Semester", "INF315" },
                    { 8, "Year 3, Semester 2. IT Trends", "Semester", "INF324"
},
                    { 9, "Year 3, Semester 1. Programming", "Semester",
"INF354" },
                    { 10, "Year 3. Project", "Year", "INF370" }
                });
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Courses");
        }
    }
}
```

MODEL
DBCONTEXT.CS

```
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"
                }
            );

            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"
                }
            );
        }
    }
}
```

COURSE.CS

```
using System.ComponentModel.DataAnnotations;

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

    }
}
```

COURSEREPOSITORY.CS

```
using Microsoft.EntityFrameworkCore;

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

        public CourseRepository(AppDbContext appDbContext)
        {
                _appDbContext = appDbContext;
        }
        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 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<bool> SaveChangesAsync()
        {
            return await _appDbContext.SaveChangesAsync() > 0;
        }

    }
}
```

ICOURSEREPOSITORY.CS

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

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


    }
}
```

VIEWMODEL

COURSEVIEWMODEL.CS

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

FRONTEND ANGULAR

ADD COURSE
ADD-COURSE.COMPONENT.HTML

```
<div class="container">
    <h1>Add New Course</h1>

    <div class="row">
      <div class="col-6">
        <form #form="ngForm" (ngSubmit)="addCourse()" novalidate>
          <div class="row mb-3">
            <label for="name" class="col-sm-2 col-form-label">Name</label>
            <input style="border-color: darkgray;" type="text" class="form-
control" id="name" name="name" [(ngModel)]="addCourseAtt.name" required
#nameField="ngModel">
          </div>
```

```
          <div class="row mb-3">
            <label for="duration" class="col-sm-2 col-form-
label">Duration</label>
            <input style="border-color: darkgray;" type="text" class="form-
control" id="duration" name="duration" [(ngModel)]="addCourseAtt.duration"
required #durationField="ngModel">
          </div>

          <div class="row mb-3">
            <label for="description" class="col-sm-2 col-form-
label">Description</label>
            <input style="border-color: darkgray;" type="text" class="form-
control" id="description" name="description"
[(ngModel)]="addCourseAtt.description" required #descriptionField="ngModel">
          </div>

          <button type="submit" class="btn-add"
[disabled]="form.invalid">Add</button>
          <button type="button" class="btn-cancel"
(click)="cancel()">Cancel</button>
        </form>
      </div>
    </div>




</div>
```

ADD-COURSE.COMPONENT.SCSS

```
.btn-add {
    background-color: #007bff;
    color: white;
  }

  .btn-cancel {
    background-color: #dc3545;
    color: white;
  }

  .btn-add, .btn-cancel {
    padding: 5px 10px;
    border: none;
    border-radius: 3px;
    cursor: pointer;
  }
```

ADD-COURSE.COMPONENT.TS

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { DataService } from '../services/data.service';
import { Course } from '../shared/course';

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

  addCourseAtt: Course = {
    courseId: 0,
    name: '',
    duration: '',
    description: '',
  };

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

  ngOnInit(): void {
  }

  addCourse(){
    this.dataService.addCourse(this.addCourseAtt).subscribe({
      next: (course) => {
        this.router.navigate(['courses'])
        console.log(course)

      }
    });
  }


  cancel(){
    this.router.navigate(["courses"]);
  };
}
```

COURSE COMPONENT

COURSE.COMPONENT.HTML

```
<h1>Course Listing</h1>

  <div class="card-container">
    <div class="card" *ngFor="let course of courses">
      <div class="card-header">
        <h2>{{ course.name }}</h2>
        <p>Duration: {{course.duration}}</p>
      </div>
      <div class="card-body">
        <p>Description: {{course.description}}</p>
      </div>
      <div class="card-footer">
        <button class="btn-edit" [routerLink]="['/editCourses',
course.courseId]">Edit</button>
        <button class="btn-delete"
(click)="deleteCourse(course.courseId)">Delete</button>
      </div>
    </div>
  </div>
```

COURSE.COMPONENT.SCSS

```
.card-container {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
  }

  .card {
    border: 1px solid #ccc;
    border-radius: 5px;
    margin: 10px;
    width: 300px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    transition: transform 0.3s;
  }

  .card:hover {
    transform: translateY(-10px);
  }

  .card-header {
    padding: 10px;
    background-color: #f2f2f2;
```

```
    border-bottom: 1px solid #ccc;
  }

  .card-header h2 {
    margin: 0;
  }

  .card-body {
    padding: 10px;
  }

  .card-footer {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px;
  }

  .btn-edit, .btn-delete {
    padding: 5px 10px;
    border: none;
    border-radius: 3px;
    cursor: pointer;
  }

  .btn-edit {
    background-color: #007bff;
    color: white;
  }

  .btn-delete {
    background-color: #dc3545;
    color: white;
  }
```

COURSE.COMPONENT.TS

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { DataService } from '../services/data.service';
import { Course } from '../shared/course';


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

```
export class CoursesComponent implements OnInit {

  courses:Course[] = []

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

  ngOnInit(): void {
    this.GetCourses()

    console.log(this.courses)
  }

  GetCourses()
  {
    this.dataService.GetCourses().subscribe(result => {
      let courseList:any[] = result
      courseList.forEach((element) => {
        this.courses.push(element)
      });
      this.courses.reverse();
    })
  }

  deleteCourse(id:number){
    this.dataService.deleteCourse(id).subscribe({
      next: (response) => {
        alert("Deleted");
        // this.GetCourses();
        window.location.reload();
      }
    });
  }

  cancel(){
    this.router.navigate(["courses"]);
  };
}
```

EDIT COURSE

EDIT-COURSE.COMPONENT.HTML

```
<div class="container">
    <h1>Edit Course</h1>

<div class="row">
    <div class="col-6">
         <form #form="ngForm" (ngSubmit)="updateCourse()">
```

```
            <div class="row mb-3">
                <label for="name" class="col-sm-2 col-form-label">Name</label>

                <input style="border-color: darkgray;" type="text"
class="form-control" id="name" name="name" [(ngModel)] = "courseAtt.name">

            </div>

            <div class="row mb-3">
                <label for="duration" class="col-sm-2 col-form-
label">Duration</label>

                <input style="border-color: darkgray;" type="text"
class="form-control" id="duration" name="duration" [(ngModel)] =
"courseAtt.duration">

            </div>

            <div class="row mb-3">
                <label for="description" class="col-sm-2 col-form-
label">Description</label>

                <input style="border-color: darkgray;" type="text"
class="form-control" id="description" name="description" [(ngModel)] =
"courseAtt.description">

            </div>

            <button type="submit" class="btn-save">Save</button>
            <button type="button" class="btn-cancel" (click)="cancel()"
>Cancel</button>
      </form>
    </div>
</div>
```

EDIT-COURSE.COMPONENT.SCSS

```
.btn-save {
    background-color: #007bff;
    color: white;
  }

  .btn-cancel {
    background-color: #dc3545;
    color: white;
  }

  .btn-save, .btn-cancel {
```

```
    padding: 5px 10px;
    border: none;
    border-radius: 3px;
    cursor: pointer;
  }
```

EDIT-COURSE.COMPONENT.TS

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

  courseAtt: Course = {
    courseId: 0,
    name: '',
    duration: '',
    description: ''
  };


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

  ngOnInit(): void {
    this.route.paramMap.subscribe({
      next: (params) => {
        const courseId = params.get('courseId');

        //Call the API
        if(courseId){
          this.dataService.getCourseId(courseId).subscribe({
            next: (response) => {
              this.courseAtt = response;
            }
          });
        }
      }
    })
  }
```

```
  updateCourse(){
    this.dataService.updateEmployee(this.courseAtt.courseId,
this.courseAtt).subscribe({
      next: (response) =>{
        this.router.navigate(['courses'])
      }
    });

  }

  cancel(){
    this.router.navigate(["courses"]);
  };
}
```

SERVICES

DATA.SERVICE.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) {
  }

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

  addCourse(addCourseAtt: Course){
    return this.httpClient.post<Course>(`${this.apiUrl}Course/AddCourse`,
addCourseAtt)
    .pipe(map(result => result))
```

```
  }

  getCourseId(courseId: string): Observable<Course>{
    return this.httpClient.get<Course>(`${this.apiUrl}Course/GetCourse/` +
courseId)
  }


  updateEmployee(id: number, courseAtt: Course): Observable<Course>{
    return this.httpClient.put<Course>(`${this.apiUrl}Course/EditCourse/` +
id, courseAtt)
  }

  deleteCourse(courseId: number): Observable<Course>{
    return this.httpClient.delete<Course>(`${this.apiUrl}Course/DeleteCourse/`
+ courseId)
  }
}
```

SHARED
COURSE.TS

```
export class Course {

    courseId: number = 0;
    name:String = '';
    duration:String = '';
    description:String = '';
}
```

APP-ROUTING.MODULE.TS

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

const routes: Routes = [

  {path: 'courses', component: CoursesComponent},
  {path: '', redirectTo: '/courses', pathMatch: 'full'},
  {path: 'addCourses', component: AddCourseComponent},
  {path: 'editCourses/:courseId', component: EditCourseComponent}

];
```

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

APP.COMPONENT.HTML

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container-fluid">
      <a class="navbar-brand" href="#">INF 354 Assignment 1</a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item">
            <h5><a class="nav-link active" aria-current="page"
routerLink="courses">Courses</a></h5>
          </li>
          <li class="nav-item">
            <h5><a class="nav-link" routerLink="addCourses">Add
Course</a></h5>
          </li>

        </ul>
      </div>
    </div>
  </nav>

<router-outlet></router-outlet>
```

APP.COMPONENT.TS

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'assignment1';
}
```

APP.MODULE.TS

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { CoursesComponent } from './course/courses.component';
import { HttpClientModule } from '@angular/common/http';
import { AddCourseComponent } from './add-course/add-course.component';
import { EditCourseComponent } from './edit-course/edit-course.component';


@NgModule({
  declarations: [
    AppComponent,
    CoursesComponent,
    AddCourseComponent,
    EditCourseComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ReactiveFormsModule,
    HttpClientModule,
    FormsModule

  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```