Backend

Controller

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

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

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

        [HttpGet]
        [Route("GetAllCourses")] //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");
        }
    }
}
```

Models

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);
    }
}
```

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; }

    }
}
```

DbContext

```
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"
    }
            );
        }
    }
}
```

ViewModel

```
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; }
    }
}
```

AppSettings

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection":
"Server=DESKTOP-HB3QVAL\SQLEXPRESS;Database=Architecture;Trusted_Connection=True;MultipleActiveResultSet
s=True"
  }
}
```

FrontEnd

Shared

```
export class Course {

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

Services

```
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)
  }
}
```

Components

EditCourse

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>
```

TypeScript

```
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"]);
  };
}
```

Course

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>
```

TypeScript

```
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"]);
  };
}
```

AddCourse

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>
```

TypeScript

```
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"]);
  };
}
</div>
```

Routing

```
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 { }
```