**HR Management System Backend Part of Project**

**Program.cs**

using Microsoft.EntityFrameworkCore;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.DependencyInjection;

using Databasehr.DAL.DBContext;

using Databasehr.BLL.Services.Implementation;

using Databasehr.BLL.Services;

using Databasehr.DAL.Repositories;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<HRdbContext>(opt =>

{

opt.UseSqlServer(builder.Configuration.GetConnectionString("dbconn"));

});

builder.Services.AddAutoMapper(typeof(Program).Assembly);

builder.Services.AddScoped<IRepositoryWrapper, RepositoryWrapper>();

builder.Services.AddScoped<IAttendanceService, AttendanceService>();

builder.Services.AddScoped<IDepartmentService, DepartmentService>();

builder.Services.AddScoped<IEmployeeService, EmployeeService>();

builder.Services.AddScoped<ILeaveRequestService, LeaveRequestService>();

builder.Services.AddScoped<ISalaryService, SalaryService>();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

// Enable CORS for all origins, headers, and methods

app.UseCors(builder => builder.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod());

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Appsettings.json**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"ConnectionStrings": {

"dbconn": "Server=LTIN284148\\SQLEXPRESS;Database=HRdbClone;Trusted\_Connection=True;"

},

"AllowedHosts": "\*"

}

**Appsetings.Development.json**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

}

}

**Migrations**

**Inbuild file**

using System;

using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Databasehr.Migrations

{

public partial class init : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Departments",

columns: table => new

{

DepartmentId = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

DepartmentName = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Departments", x => x.DepartmentId);

});

migrationBuilder.CreateTable(

name: "HrLogins",

columns: table => new

{

Id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

HrName = table.Column<string>(type: "nvarchar(max)", nullable: false),

Email = table.Column<string>(type: "nvarchar(max)", nullable: false),

Password = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_HrLogins", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Employees",

columns: table => new

{

EmployeeId = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

FirstName = table.Column<string>(type: "nvarchar(max)", nullable: false),

LastName = table.Column<string>(type: "nvarchar(max)", nullable: false),

Email = table.Column<string>(type: "nvarchar(max)", nullable: false),

Phone = table.Column<string>(type: "nvarchar(max)", nullable: false),

Address = table.Column<string>(type: "nvarchar(max)", nullable: false),

JoinDate = table.Column<DateTime>(type: "datetime2", nullable: false),

DepartmentId = table.Column<int>(type: "int", nullable: false),

Password = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Employees", x => x.EmployeeId);

table.ForeignKey(

name: "FK\_Employees\_Departments\_DepartmentId",

column: x => x.DepartmentId,

principalTable: "Departments",

principalColumn: "DepartmentId",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "Attendances",

columns: table => new

{

AttendanceId = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

EmployeeId = table.Column<int>(type: "int", nullable: false),

Date = table.Column<DateTime>(type: "datetime2", nullable: false),

CheckInTime = table.Column<DateTime>(type: "datetime2", nullable: false),

CheckOutTime = table.Column<DateTime>(type: "datetime2", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Attendances", x => x.AttendanceId);

table.ForeignKey(

name: "FK\_Attendances\_Employees\_EmployeeId",

column: x => x.EmployeeId,

principalTable: "Employees",

principalColumn: "EmployeeId",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "LeaveRequests",

columns: table => new

{

LeaveRequestId = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

EmployeeId = table.Column<int>(type: "int", nullable: false),

LeaveStartDate = table.Column<DateTime>(type: "datetime2", nullable: false),

LeaveEndDate = table.Column<DateTime>(type: "datetime2", nullable: false),

LeaveType = table.Column<string>(type: "nvarchar(max)", nullable: false),

Status = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_LeaveRequests", x => x.LeaveRequestId);

table.ForeignKey(

name: "FK\_LeaveRequests\_Employees\_EmployeeId",

column: x => x.EmployeeId,

principalTable: "Employees",

principalColumn: "EmployeeId",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "Salaries",

columns: table => new

{

SalaryId = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

EmployeeId = table.Column<int>(type: "int", nullable: false),

Month = table.Column<int>(type: "int", nullable: false),

Year = table.Column<int>(type: "int", nullable: false),

BasicSalary = table.Column<decimal>(type: "decimal(18,2)", nullable: false),

Allowances = table.Column<decimal>(type: "decimal(18,2)", nullable: false),

Deductions = table.Column<decimal>(type: "decimal(18,2)", nullable: false),

NetSalary = table.Column<decimal>(type: "decimal(18,2)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Salaries", x => x.SalaryId);

table.ForeignKey(

name: "FK\_Salaries\_Employees\_EmployeeId",

column: x => x.EmployeeId,

principalTable: "Employees",

principalColumn: "EmployeeId",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_Attendances\_EmployeeId",

table: "Attendances",

column: "EmployeeId");

migrationBuilder.CreateIndex(

name: "IX\_Employees\_DepartmentId",

table: "Employees",

column: "DepartmentId");

migrationBuilder.CreateIndex(

name: "IX\_LeaveRequests\_EmployeeId",

table: "LeaveRequests",

column: "EmployeeId");

migrationBuilder.CreateIndex(

name: "IX\_Salaries\_EmployeeId",

table: "Salaries",

column: "EmployeeId");

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Attendances");

migrationBuilder.DropTable(

name: "HrLogins");

migrationBuilder.DropTable(

name: "LeaveRequests");

migrationBuilder.DropTable(

name: "Salaries");

migrationBuilder.DropTable(

name: "Employees");

migrationBuilder.DropTable(

name: "Departments");

}

}

}

**Created File**

// <auto-generated />

using System;

using Databasehr.DAL.DBContext;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

#nullable disable

namespace Databasehr.Migrations

{

[DbContext(typeof(HRdbContext))]

partial class HRdbContextModelSnapshot : ModelSnapshot

{

protected override void BuildModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "6.0.26")

.HasAnnotation("Relational:MaxIdentifierLength", 128);

SqlServerModelBuilderExtensions.UseIdentityColumns(modelBuilder, 1L, 1);

modelBuilder.Entity("Databasehr.DAL.Entities.Attendance", b =>

{

b.Property<int>("AttendanceId")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("AttendanceId"), 1L, 1);

b.Property<DateTime>("CheckInTime")

.HasColumnType("datetime2");

b.Property<DateTime>("CheckOutTime")

.HasColumnType("datetime2");

b.Property<DateTime>("Date")

.HasColumnType("datetime2");

b.Property<int>("EmployeeId")

.HasColumnType("int");

b.HasKey("AttendanceId");

b.HasIndex("EmployeeId");

b.ToTable("Attendances");

});

modelBuilder.Entity("Databasehr.DAL.Entities.Department", b =>

{

b.Property<int>("DepartmentId")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("DepartmentId"), 1L, 1);

b.Property<string>("DepartmentName")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("DepartmentId");

b.ToTable("Departments");

});

modelBuilder.Entity("Databasehr.DAL.Entities.Employee", b =>

{

b.Property<int>("EmployeeId")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("EmployeeId"), 1L, 1);

b.Property<string>("Address")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<int>("DepartmentId")

.HasColumnType("int");

b.Property<string>("Email")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("FirstName")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<DateTime>("JoinDate")

.HasColumnType("datetime2");

b.Property<string>("LastName")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("Password")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("Phone")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("EmployeeId");

b.HasIndex("DepartmentId");

b.ToTable("Employees");

});

modelBuilder.Entity("Databasehr.DAL.Entities.HrLogin", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("Id"), 1L, 1);

b.Property<string>("Email")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("HrName")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("Password")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.ToTable("HrLogins");

});

modelBuilder.Entity("Databasehr.DAL.Entities.LeaveRequest", b =>

{

b.Property<int>("LeaveRequestId")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("LeaveRequestId"), 1L, 1);

b.Property<int>("EmployeeId")

.HasColumnType("int");

b.Property<DateTime>("LeaveEndDate")

.HasColumnType("datetime2");

b.Property<DateTime>("LeaveStartDate")

.HasColumnType("datetime2");

b.Property<string>("LeaveType")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("Status")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("LeaveRequestId");

b.HasIndex("EmployeeId");

b.ToTable("LeaveRequests");

});

modelBuilder.Entity("Databasehr.DAL.Entities.Salary", b =>

{

b.Property<int>("SalaryId")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("SalaryId"), 1L, 1);

b.Property<decimal>("Allowances")

.HasColumnType("decimal(18,2)");

b.Property<decimal>("BasicSalary")

.HasColumnType("decimal(18,2)");

b.Property<decimal>("Deductions")

.HasColumnType("decimal(18,2)");

b.Property<int>("EmployeeId")

.HasColumnType("int");

b.Property<int>("Month")

.HasColumnType("int");

b.Property<decimal>("NetSalary")

.HasColumnType("decimal(18,2)");

b.Property<int>("Year")

.HasColumnType("int");

b.HasKey("SalaryId");

b.HasIndex("EmployeeId");

b.ToTable("Salaries");

});

modelBuilder.Entity("Databasehr.DAL.Entities.Attendance", b =>

{

b.HasOne("Databasehr.DAL.Entities.Employee", "Employee")

.WithMany("Attendances")

.HasForeignKey("EmployeeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Employee");

});

modelBuilder.Entity("Databasehr.DAL.Entities.Employee", b =>

{

b.HasOne("Databasehr.DAL.Entities.Department", "Department")

.WithMany("Employees")

.HasForeignKey("DepartmentId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Department");

});

modelBuilder.Entity("Databasehr.DAL.Entities.LeaveRequest", b =>

{

b.HasOne("Databasehr.DAL.Entities.Employee", "Employee")

.WithMany("LeaveRequests")

.HasForeignKey("EmployeeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Employee");

});

modelBuilder.Entity("Databasehr.DAL.Entities.Salary", b =>

{

b.HasOne("Databasehr.DAL.Entities.Employee", "Employee")

.WithMany("Salaries")

.HasForeignKey("EmployeeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Employee");

});

modelBuilder.Entity("Databasehr.DAL.Entities.Department", b =>

{

b.Navigation("Employees");

});

modelBuilder.Entity("Databasehr.DAL.Entities.Employee", b =>

{

b.Navigation("Attendances");

b.Navigation("LeaveRequests");

b.Navigation("Salaries");

});

#pragma warning restore 612, 618

}

}

}

**Data Access Layer**

**DB Context**using System;

using System.Collections.Generic;

using Databasehr.DAL.Entities;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata;

//using Databasehr.DAL.Models;

namespace Databasehr.DAL.DBContext

{

public partial class HRdbContext : DbContext

{

public HRdbContext()

{

}

public HRdbContext(DbContextOptions<HRdbContext> options)

: base(options)

{

}

public virtual DbSet<Attendance> Attendances { get; set; } = null!;

public virtual DbSet<Department> Departments { get; set; } = null!;

public virtual DbSet<Employee> Employees { get; set; } = null!;

public virtual DbSet<HrLogin> HrLogins { get; set; } = null!;

public virtual DbSet<LeaveRequest> LeaveRequests { get; set; } = null!;

public virtual DbSet<Salary> Salaries { get; set; } = null!;

public IEnumerable<object> Employee { get; internal set; }

}

}

**Entities**

using System;

using System.Collections.Generic;

namespace Databasehr.DAL.Entities

{

public partial class Attendance

{

public int AttendanceId { get; set; }

public int EmployeeId { get; set; }

public DateTime Date { get; set; }

public DateTime CheckInTime { get; set; }

public DateTime CheckOutTime { get; set; }

public virtual Employee Employee { get; set; } = null!;

}

}

using System;

using System.Collections.Generic;

namespace Databasehr.DAL.Entities

{

public partial class Department

{

public Department()

{

Employees = new HashSet<Employee>();

}

public int DepartmentId { get; set; }

public string DepartmentName { get; set; } = null!;

public virtual ICollection<Employee> Employees { get; set; }

}

}

using System;

using System.Collections.Generic;

namespace Databasehr.DAL.Entities

{

public partial class Employee

{

public Employee()

{

Attendances = new HashSet<Attendance>();

LeaveRequests = new HashSet<LeaveRequest>();

Salaries = new HashSet<Salary>();

}

public int EmployeeId { get; set; }

public string FirstName { get; set; } = null!;

public string LastName { get; set; } = null!;

public string Email { get; set; } = null!;

public string Phone { get; set; } = null!;

public string Address { get; set; } = null!;

public DateTime JoinDate { get; set; }

public int DepartmentId { get; set; }

public string Password { get; set; } = null!;

public virtual Department Department { get; set; } = null!;

public virtual ICollection<Attendance> Attendances { get; set; }

public virtual ICollection<LeaveRequest> LeaveRequests { get; set; }

public virtual ICollection<Salary> Salaries { get; set; }

}

}

using System;

using System.Collections.Generic;

namespace Databasehr.DAL.Entities

{

public partial class HrLogin

{

public int Id { get; set; }

public string HrName { get; set; } = null!;

public string Email { get; set; } = null!;

public string Password { get; set; } = null!;

}

}

using System;

using System.Collections.Generic;

namespace Databasehr.DAL.Entities

{

public partial class LeaveRequest

{

public int LeaveRequestId { get; set; }

public int EmployeeId { get; set; }

public DateTime LeaveStartDate { get; set; }

public DateTime LeaveEndDate { get; set; }

public string LeaveType { get; set; } = null!;

public string Status { get; set; } = null!;

public virtual Employee Employee { get; set; } = null!;

}

}

using System;

using System.Collections.Generic;

namespace Databasehr.DAL.Entities

{

public partial class Salary

{

public int SalaryId { get; set; }

public int EmployeeId { get; set; }

public int Month { get; set; }

public int Year { get; set; }

public decimal BasicSalary { get; set; }

public decimal Allowances { get; set; }

public decimal Deductions { get; set; }

public decimal NetSalary { get; set; }

public virtual Employee Employee { get; set; } = null!;

}

}

**Repositories**

using Microsoft.EntityFrameworkCore;

using System.Linq.Expressions;

using Databasehr.DAL.Entities;

using System.Data;

using Databasehr.DAL.DBContext;

using Databasehr.BLL.DTO.Request;

using System.Threading.Tasks;

namespace Databasehr.DAL.Repositories

{

public interface IBaseRepository<T> where T : class

{

Task<T> CreateAsync(T entity);

Task<T> UpdateAsync(int id, T entity);

Task<bool> DeleteAsync(T entity);

Task<IEnumerable<T>> GetAllAsync(params string[] navsToInclude);

Task<T?> GetById(int id);

Task<IEnumerable<T>> GetByCondition(Expression<Func<T, bool>> condition);

}

public abstract class RepositoryBase<T> : IBaseRepository<T> where T : class

{

private readonly HRdbContext context;

private readonly DbSet<T> dbset;

public RepositoryBase(HRdbContext context)

{

this.context = context;

this.dbset = this.context.Set<T>();

}

public async Task<T> CreateAsync(T entity)

{

var result = await this.dbset.AddAsync(entity);

return result.Entity;

}

public async Task<bool> DeleteAsync(T entity)

{

var result = this.dbset.Remove(entity);

return await Task.FromResult(result.Entity is not null);

}

public async Task<IEnumerable<T>> GetAllAsync(params string[] navsToInclude)

{

IQueryable<T> query = this.dbset;

if (navsToInclude.Length > 0)

{

foreach (var item in navsToInclude)

{

query = query.Include(item);

}

}

var result = query.AsEnumerable();

return await Task.FromResult(result);

}

public async Task<IEnumerable<T>> GetByCondition(Expression<Func<T, bool>> condition)

{

var result = this.dbset.Where(condition);

return await Task.FromResult(result);

}

public async Task<T?> GetById(int id)

{

var result = this.dbset.FindAsync(id);

return await result;

}

public async Task<T> UpdateAsync(int id, T entity)

{

var dbEntity = await this.dbset.FindAsync(id);

if (dbEntity == null)

throw new KeyNotFoundException($"Resource with Id:{id} not Found");

context.Entry(dbEntity).CurrentValues.SetValues(entity);

return entity;

}

}

public interface IAttendanceRepository : IBaseRepository<Attendance> { }

public class AttendanceRepository : RepositoryBase<Attendance>, IAttendanceRepository

{

public AttendanceRepository(HRdbContext context) : base(context)

{

}

}

public interface IDepartmentRepository : IBaseRepository<Department> { }

public class DepartmentRepository : RepositoryBase<Department>, IDepartmentRepository

{

public DepartmentRepository(HRdbContext context) : base(context)

{

}

}

public interface IEmployeeRepository : IBaseRepository<Employee> {

Task<int> Login(LoginRequestDTO d);

}

public class EmployeeRepository : RepositoryBase<Employee>, IEmployeeRepository

{

private readonly HRdbContext \_db;

public EmployeeRepository(HRdbContext context) : base(context)

{

\_db = context;

}

public async Task<int> Login(LoginRequestDTO d)

{

var data = await \_db.Employees.Where(i => i.Email == d.email && i.Password == d.password).FirstOrDefaultAsync();

return data.EmployeeId;

}

}

public interface ILeaveRequestRepository : IBaseRepository<LeaveRequest> { }

public class LeaveRequestRepository : RepositoryBase<LeaveRequest>, ILeaveRequestRepository

{

public LeaveRequestRepository(HRdbContext context) : base(context)

{

}

}

public interface ISalaryRepository : IBaseRepository<Salary> { }

public class SalaryRepository : RepositoryBase<Salary>, ISalaryRepository

{

public SalaryRepository(HRdbContext context) : base(context)

{

}

}

public interface IRepositoryWrapper

{

IAttendanceRepository AttendanceRepository { get; set; }

IDepartmentRepository DepartmentRepository { get; set; }

IEmployeeRepository EmployeeRepository { get; set; }

ILeaveRequestRepository LeaveRequestRepository { get; set; }

ISalaryRepository SalaryRepository { get; set; }

Task<int> SaveAsync();

}

public class RepositoryWrapper : IRepositoryWrapper

{

public readonly HRdbContext context;

public IAttendanceRepository AttendanceRepository { get; set; }

public IDepartmentRepository DepartmentRepository { get; set; }

public IEmployeeRepository EmployeeRepository { get; set; }

public ILeaveRequestRepository LeaveRequestRepository { get; set; }

public ISalaryRepository SalaryRepository { get; set; }

public RepositoryWrapper(HRdbContext context)

{

this.context = context;

AttendanceRepository = new AttendanceRepository(context);

DepartmentRepository = new DepartmentRepository(context);

EmployeeRepository = new EmployeeRepository(context);

LeaveRequestRepository = new LeaveRequestRepository(context);

SalaryRepository = new SalaryRepository(context);

}

public async Task<int> SaveAsync()

{

return await this.context.SaveChangesAsync();

}

}

}

**Controllers**

using AutoMapper;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

using Databasehr.BLL.Services;

using Microsoft.AspNetCore.Mvc;

namespace Databasehr.Controllers

{

[Route("api/[controller]")] // This line specifies the base route for all the endpoints in this controller.

// In this case, the base route is /api/Attendance

[ApiController] //This attribute indicates that this class is an API controller,

//which means it handles HTTP requests and returns data in response

public class AttendanceController : ControllerBase

{

private readonly IAttendanceService AttendanceService; //The constructor initializes the controller and receives an instance of the IBookingService interface.

//Dependency injection ensures that the controller has access to the necessary services.

private readonly IMapper Mapper;

public AttendanceController(IAttendanceService AttendanceService,IMapper Mapper)

{

this.Mapper = Mapper;

this.AttendanceService = AttendanceService;

}

[HttpGet]

public async Task<IActionResult> Get()

{

var result = await this.AttendanceService.GetAll();

return Ok(result);

//It calls the bookingService.GetAll() method to retrieve all booking items.

//The await keyword indicates that this operation is asynchronous(non - blocking).

}

[HttpPost]

public async Task<IActionResult> Post(AttendanceRequestDTO customerRequest)

{

var result = await this.AttendanceService.Add(customerRequest);

var res = Mapper.Map<AttendanceResponseDTO>(result);

return Ok(res);

}

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

try

{

var result = await this.AttendanceService.Delete(id);

if (result)

{

return Ok("Deleted");

}

else

{

return NotFound("Not found");

}

}

catch (Exception ex)

{

return StatusCode(500, ex);

}

}

[HttpPut("{id}")]

public async Task<IActionResult> Edit(int id, AttendanceRequestDTO v)

{

await this.AttendanceService.Update(id, v);

return Ok(v);

}

[HttpGet("id")]

public async Task<IActionResult> GetbyId(int id)

{

var result = await this.AttendanceService.GetById(id);

return Ok(result);

}

}

}

using AutoMapper;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.Services;

using Microsoft.AspNetCore.Mvc;

namespace Databasehr.Controllers

{

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

[ApiController]

public class DepartmentController : ControllerBase

{

private readonly IDepartmentService DepartmentService;

public DepartmentController(IDepartmentService DepartmentService)

{

this.DepartmentService = DepartmentService;

}

[HttpGet]

public async Task<IActionResult> Get()

{

var result = await this.DepartmentService.GetAll();

return Ok(result);

}

[HttpPost]

public async Task<IActionResult> Post(DepartmentRequestDTO customerRequest)

{

var result = await this.DepartmentService.Add(customerRequest);

return Ok(result);

}

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

try

{

var result = await this.DepartmentService.Delete(id);

if (result) return Ok("Deleted");

return NotFound("Not found");

}

catch (Exception ex)

{

return StatusCode(500, ex);

}

}

[HttpPut("{id}")]

public async Task<IActionResult> Edit(int id, DepartmentRequestDTO v)

{

await this.DepartmentService.Update(id, v);

return Ok(v);

}

[HttpGet("id")]

public async Task<IActionResult> GetbyId(int id)

{

var result = await this.DepartmentService.GetById(id);

return Ok(result);

}

}

}

using AutoMapper;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.Services;

using Microsoft.AspNetCore.Mvc;

namespace Databasehr.Controllers

{

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

[ApiController]

public class EmployeeController : ControllerBase

{

private readonly IEmployeeService EmployeeService;

public EmployeeController(IEmployeeService EmployeeService)

{

this.EmployeeService = EmployeeService;

}

[HttpGet]

public async Task<IActionResult> Get()

{

var result = await this.EmployeeService.GetAll();

return Ok(result);

}

[HttpPost]

public async Task<IActionResult> Post(EmployeeRequestDTO customerRequest)

{

var result = await this.EmployeeService.Add(customerRequest);

return Ok(result);

}

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

try

{

var result = await this.EmployeeService.Delete(id);

if (result) return Ok("Deleted");

return NotFound("Not found");

}

catch (Exception ex)

{

return StatusCode(500, ex);

}

}

[HttpPut("{id}")]

public async Task<IActionResult> Edit(int id, EmployeeRequestDTO v)

{

await this.EmployeeService.Update(id, v);

return Ok(v);

}

[HttpGet("id")]

public async Task<IActionResult> GetbyId(int id)

{

var result = await this.EmployeeService.GetById(id);

return Ok(result);

}

[HttpPost("api/[controller]/Login")]

public async Task<IActionResult> Login(LoginRequestDTO d)

{

var res = await this.EmployeeService.Login(d);

return Ok(res);

}

}

}using AutoMapper;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.Services;

using Databasehr.DAL.DBContext;

using Microsoft.AspNetCore.Mvc;

namespace Databasehr.Controllers

{

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

[ApiController]

public class LeaveRequestController : ControllerBase

{

private readonly ILeaveRequestService LeaveRequestService;

private readonly HRdbContext \_db;

public LeaveRequestController(ILeaveRequestService LeaveRequestService,HRdbContext db)

{

\_db = db;

this.LeaveRequestService = LeaveRequestService;

}

[HttpGet]

public async Task<IActionResult> Get()

{

var result = await this.LeaveRequestService.GetAll();

return Ok(result);

}

[HttpPost]

public async Task<IActionResult> Post(LeaveRequestRequestDTO customerRequest)

{

var result = await this.LeaveRequestService.Add(customerRequest);

return Ok(result);

}

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

try

{

var result = await this.LeaveRequestService.Delete(id);

if (result) return Ok("Deleted");

return NotFound("Not found");

}

catch (Exception ex)

{

return StatusCode(500, ex);

}

}

[HttpPut("{id}")]

public async Task<IActionResult> Edit(int id, LeaveRequestRequestDTO v)

{

await this.LeaveRequestService.Update(id, v);

return Ok(v);

}

[HttpGet("id")]

public async Task<IActionResult> GetbyId(int id)

{

var result = await this.LeaveRequestService.GetById(id);

return Ok(result);

}

[HttpGet("api/[controller]/GetPastLeave")]

public async Task<IActionResult> GetPastLeave(int id)

{

var d = \_db.LeaveRequests.Where(i => i.EmployeeId == id).ToList();

return Ok(d);

}

[HttpGet("api/[controller]/EditStatusApprove")]

public async Task<IActionResult> EditApprove(int id)

{

var d = \_db.LeaveRequests.Find(id);

d.Status ="Approved";

\_db.LeaveRequests.Update(d);

\_db.SaveChanges();

return Ok();

}

[HttpGet("api/[controller]/EditStatusDeny")]

public async Task<IActionResult> EditDeny(int id)

{

var d = \_db.LeaveRequests.Find(id);

d.Status = "Denied";

\_db.LeaveRequests.Update(d);

\_db.SaveChanges();

return Ok();

}

}

}

using AutoMapper;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.Services;

using Databasehr.DAL.DBContext;

using Microsoft.AspNetCore.Mvc;

namespace Databasehr.Controllers

{

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

[ApiController]

public class SalaryController : ControllerBase

{

private readonly ISalaryService SalaryService;

private readonly HRdbContext \_db;

public SalaryController(ISalaryService SalaryService,HRdbContext db)

{

this.SalaryService = SalaryService;

\_db = db;

}

[HttpGet]

public async Task<IActionResult> Get()

{

var result = await this.SalaryService.GetAll();

return Ok(result);

}

[HttpPost]

public async Task<IActionResult> Post(SalaryRequestDTO customerRequest)

{

var result = await this.SalaryService.Add(customerRequest);

return Ok(result);

}

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

try

{

var result = await this.SalaryService.Delete(id);

if (result) return Ok("Deleted");

return NotFound("Not found");

}

catch (Exception ex)

{

return StatusCode(500, ex);

}

}

[HttpPut("{id}")]

public async Task<IActionResult> Edit(int id, SalaryRequestDTO v)

{

await this.SalaryService.Update(id, v);

return Ok(v);

}

[HttpGet("id")]

public async Task<IActionResult> GetbyId(int id)

{

var result = await this.SalaryService.GetById(id);

return Ok(result);

}

[HttpGet("api/[controller]/{id}")]

public async Task<IActionResult> GetSalaryByUserId(int id)

{

return Ok(\_db.Salaries.Where(i=>i.EmployeeId==id));

}

}

}

Properties

{

"$schema": "https://json.schemastore.org/launchsettings.json",

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:45834",

"sslPort": 44348

}

},

"profiles": {

"Databasehr": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"launchUrl": "swagger",

"applicationUrl": "https://localhost:7211;http://localhost:5280",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"launchUrl": "swagger",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

}

Business Logic Layer

Automapper

using Databasehr.DAL.Entities;

using AutoMapper;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

namespace Databasehr.BLL.AutoMapper

{

public class DefaultProfile : Profile

{

public DefaultProfile()

{

CreateMap<AttendanceRequestDTO, Attendance>();

CreateMap<Attendance, AttendanceResponseDTO>();

CreateMap<DepartmentRequestDTO, Department>();

CreateMap<Department, DepartmentResponseDTO>();

CreateMap<EmployeeRequestDTO, Employee>();

CreateMap<Employee, EmployeeResponseDTO>();

CreateMap<LeaveRequestRequestDTO, LeaveRequest>();

CreateMap<LeaveRequest, LeaveRequestResponseDTO>();

CreateMap<SalaryRequestDTO, Salary>();

CreateMap<Salary, SalaryResponseDTO>();

}

}

}

**DTO**

**Request DTO**

namespace Databasehr.BLL.DTO.Request

{

public class AttendanceRequestDTO

{

public int EmployeeId { get; set; }

public DateTime Date { get; set; }

public DateTime CheckInTime { get; set; }

public DateTime CheckOutTime { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

namespace Databasehr.BLL.DTO.Request

{

public class DepartmentRequestDTO

{

[StringLength(50)]

public string DepartmentName { get; set; } = null!;

}

}

using System.ComponentModel.DataAnnotations;

namespace Databasehr.BLL.DTO.Request

{

public class EmployeeRequestDTO

{

[StringLength(50)]

public string FirstName { get; set; } = null!;

[StringLength(50)]

public string LastName { get; set; } = null!;

public string Email { get; set; } = null!;

public string Phone { get; set; } = null!;

public string Address { get; set; } = null!;

public DateTime JoinDate { get; set; }

public int DepartmentId { get; set; }

public string Password { get; set; } = null!;

}

}

namespace Databasehr.BLL.DTO.Request

{

public class LeaveRequestRequestDTO

{

public int EmployeeId { get; set; }

public DateTime LeaveStartDate { get; set; }

public DateTime LeaveEndDate { get; set; }

public string LeaveType { get; set; } = null!;

public string Status { get; set; }

}

}

namespace Databasehr.BLL.DTO.Request

{

public class LoginRequestDTO

{

public string email { get; set; }

public string password { get; set; }

}

}

namespace Databasehr.BLL.DTO.Request

{

public class SalaryRequestDTO

{

public int EmployeeId { get; set; }

public int Month { get; set; }

public int Year { get; set; }

public decimal BasicSalary { get; set; }

public decimal Allowances { get; set; }

public decimal Deductions { get; set; }

public decimal NetSalary { get; set; }

}

}

**Response**

namespace Databasehr.BLL.DTO.Response

{

public class AttendanceResponseDTO

{

public int AttendanceId { get; set; }

public int EmployeeId { get; set; }

public DateTime Date { get; set; }

public DateTime CheckInTime { get; set; }

public DateTime CheckOutTime { get; set; }

}

}

namespace Databasehr.BLL.DTO.Response

{

public class DepartmentResponseDTO

{

public int DepartmentId { get; set; }

public string DepartmentName { get; set; } = null!;

}

}

namespace Databasehr.BLL.DTO.Response

{

public class EmployeeResponseDTO

{

public int EmployeeId { get; set; }

public string FirstName { get; set; } = null!;

public string LastName { get; set; } = null!;

public string Email { get; set; } = null!;

public string Phone { get; set; } = null!;

public string Address { get; set; } = null!;

public DateTime JoinDate { get; set; }

public int DepartmentId { get; set; }

public string Password { get; set; } = null!;

}

}

namespace Databasehr.BLL.DTO.Response

{

public class LeaveRequestResponseDTO

{

public int LeaveRequestId { get; set; }

public int EmployeeId { get; set; }

public DateTime LeaveStartDate { get; set; }

public DateTime LeaveEndDate { get; set; }

public string LeaveType { get; set; } = null!;

public string Status { get; set; } = null!;

}

}

namespace Databasehr.BLL.DTO.Response

{

public class SalaryResponseDTO

{

public int SalaryId { get; set; }

public int EmployeeId { get; set; }

public int Month { get; set; }

public int Year { get; set; }

public decimal BasicSalary { get; set; }

public decimal Allowances { get; set; }

public decimal Deductions { get; set; }

public decimal NetSalary { get; set; }

}

}

**Services**

**Implementation**

using AutoMapper;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

using Databasehr.DAL.Entities;

using Databasehr.DAL.Repositories;

using Databasehr.BLL.Services;

namespace Databasehr.BLL.Services.Implementation

{

public class AttendanceService : IAttendanceService

{

private readonly IRepositoryWrapper repositoryWrapper;

private readonly IMapper mapper;

public AttendanceService(IRepositoryWrapper repositoryWrapper, IMapper mapper)

{

this.repositoryWrapper = repositoryWrapper;

this.mapper = mapper;

}

public async Task<AttendanceResponseDTO> Add(AttendanceRequestDTO requestDTO)

{

var cart = mapper.Map<Attendance>(requestDTO);

var cartResponse = await this.repositoryWrapper.AttendanceRepository.CreateAsync(cart);

await this.repositoryWrapper.SaveAsync();

var result = mapper.Map<AttendanceResponseDTO>(cartResponse);

return result;

}

public async Task<bool> Delete(int id)

{

var orderdetailtodelete = await this.repositoryWrapper.AttendanceRepository.GetById(id);

if (orderdetailtodelete == null)

{

return false;

}

await this.repositoryWrapper.AttendanceRepository.DeleteAsync(orderdetailtodelete);

await this.repositoryWrapper.SaveAsync();

return true;

}

public async Task<IEnumerable<AttendanceResponseDTO>> GetAll()

{

var carts = await this.repositoryWrapper.AttendanceRepository.GetAllAsync();

var result = mapper.Map<IEnumerable<AttendanceResponseDTO>>(carts);

return result;

}

public async Task<AttendanceResponseDTO> GetById(int id)

{

var data = await repositoryWrapper.AttendanceRepository.GetById(id);

var res = mapper.Map<AttendanceResponseDTO>(data);

return res;

}

public async Task<AttendanceResponseDTO> Update(int id, AttendanceRequestDTO res)

{

var f = mapper.Map<Attendance>(res);

f.AttendanceId = id;

var v = await repositoryWrapper.AttendanceRepository.UpdateAsync(id, f);

await repositoryWrapper.SaveAsync();

var k = mapper.Map<AttendanceResponseDTO>(v);

return k;

}

}

}

using AutoMapper;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

using Databasehr.DAL.Entities;

using Databasehr.DAL.Repositories;

using Databasehr.BLL.Services;

namespace Databasehr.BLL.Services.Implementation

{

public class DepartmentService : IDepartmentService

{

private readonly IRepositoryWrapper repositoryWrapper;

private readonly IMapper mapper;

public DepartmentService(IRepositoryWrapper repositoryWrapper, IMapper mapper)

{

this.repositoryWrapper = repositoryWrapper;

this.mapper = mapper;

}

public async Task<DepartmentResponseDTO> Add(DepartmentRequestDTO requestDTO)

{

var cart = mapper.Map<Department>(requestDTO);

var cartResponse = await this.repositoryWrapper.DepartmentRepository.CreateAsync(cart);

await this.repositoryWrapper.SaveAsync();

var result = mapper.Map<DepartmentResponseDTO>(cartResponse);

return result;

}

public async Task<bool> Delete(int id)

{

var orderdetailtodelete = await this.repositoryWrapper.DepartmentRepository.GetById(id);

if (orderdetailtodelete == null)

{

return false;

}

await this.repositoryWrapper.DepartmentRepository.DeleteAsync(orderdetailtodelete);

await this.repositoryWrapper.SaveAsync();

return true;

}

public async Task<IEnumerable<DepartmentResponseDTO>> GetAll()

{

var carts = await this.repositoryWrapper.DepartmentRepository.GetAllAsync();

var result = mapper.Map<IEnumerable<DepartmentResponseDTO>>(carts);

return result;

}

public async Task<DepartmentResponseDTO> GetById(int id)

{

var data = await repositoryWrapper.DepartmentRepository.GetById(id);

var res = mapper.Map<DepartmentResponseDTO>(data);

return res;

}

public async Task<DepartmentResponseDTO> Update(int id, DepartmentRequestDTO res)

{

var f = mapper.Map<Department>(res);

f.DepartmentId = id;

var v = await repositoryWrapper.DepartmentRepository.UpdateAsync(id, f);

await repositoryWrapper.SaveAsync();

var k = mapper.Map<DepartmentResponseDTO>(v);

return k;

}

}

}

using AutoMapper;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

using Databasehr.DAL.Entities;

using Databasehr.DAL.Repositories;

using Databasehr.BLL.Services;

namespace Databasehr.BLL.Services.Implementation

{

public class EmployeeService : IEmployeeService

{

private readonly IRepositoryWrapper repositoryWrapper;

private readonly IMapper mapper;

public EmployeeService(IRepositoryWrapper repositoryWrapper, IMapper mapper)

{

this.repositoryWrapper = repositoryWrapper;

this.mapper = mapper;

}

public async Task<EmployeeResponseDTO> Add(EmployeeRequestDTO requestDTO)

{

var cart = mapper.Map<Employee>(requestDTO);

var cartResponse = await this.repositoryWrapper.EmployeeRepository.CreateAsync(cart);

await this.repositoryWrapper.SaveAsync();

var result = mapper.Map<EmployeeResponseDTO>(cartResponse);

return result;

}

public async Task<bool> Delete(int id)

{

var orderdetailtodelete = await this.repositoryWrapper.EmployeeRepository.GetById(id);

if (orderdetailtodelete == null)

{

return false;

}

await this.repositoryWrapper.EmployeeRepository.DeleteAsync(orderdetailtodelete);

await this.repositoryWrapper.SaveAsync();

return true;

}

public async Task<IEnumerable<EmployeeResponseDTO>> GetAll()

{

var carts = await this.repositoryWrapper.EmployeeRepository.GetAllAsync();

var result = mapper.Map<IEnumerable<EmployeeResponseDTO>>(carts);

return result;

}

public async Task<EmployeeResponseDTO> GetById(int id)

{

var data = await repositoryWrapper.EmployeeRepository.GetById(id);

var res = mapper.Map<EmployeeResponseDTO>(data);

return res;

}

public async Task<EmployeeResponseDTO> Update(int id, EmployeeRequestDTO res)

{

var f = mapper.Map<Employee>(res);

f.EmployeeId = id;

var v = await repositoryWrapper.EmployeeRepository.UpdateAsync(id, f);

await repositoryWrapper.SaveAsync();

var k = mapper.Map<EmployeeResponseDTO>(v);

return k;

}

public async Task<int> Login(LoginRequestDTO d)

{

var data = await repositoryWrapper.EmployeeRepository.Login(d);

return data;

}

}

}

using AutoMapper;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

using Databasehr.DAL.Entities;

using Databasehr.DAL.Repositories;

using Databasehr.BLL.Services;

namespace Databasehr.BLL.Services.Implementation

{

public class LeaveRequestService : ILeaveRequestService

{

private readonly IRepositoryWrapper repositoryWrapper;

private readonly IMapper mapper;

public LeaveRequestService(IRepositoryWrapper repositoryWrapper, IMapper mapper)

{

this.repositoryWrapper = repositoryWrapper;

this.mapper = mapper;

}

public async Task<LeaveRequestResponseDTO> Add(LeaveRequestRequestDTO requestDTO)

{

var cart = mapper.Map<LeaveRequest>(requestDTO);

var cartResponse = await this.repositoryWrapper.LeaveRequestRepository.CreateAsync(cart);

await this.repositoryWrapper.SaveAsync();

var result = mapper.Map<LeaveRequestResponseDTO>(cartResponse);

return result;

}

public async Task<bool> Delete(int id)

{

var orderdetailtodelete = await this.repositoryWrapper.LeaveRequestRepository.GetById(id);

if (orderdetailtodelete == null)

{

return false;

}

await this.repositoryWrapper.LeaveRequestRepository.DeleteAsync(orderdetailtodelete);

await this.repositoryWrapper.SaveAsync();

return true;

}

public async Task<IEnumerable<LeaveRequestResponseDTO>> GetAll()

{

var carts = await this.repositoryWrapper.LeaveRequestRepository.GetAllAsync();

var result = mapper.Map<IEnumerable<LeaveRequestResponseDTO>>(carts);

return result;

}

public async Task<LeaveRequestResponseDTO> GetById(int id)

{

var data = await repositoryWrapper.LeaveRequestRepository.GetById(id);

var res = mapper.Map<LeaveRequestResponseDTO>(data);

return res;

}

public async Task<LeaveRequestResponseDTO> Update(int id, LeaveRequestRequestDTO res)

{

var f = mapper.Map<LeaveRequest>(res);

f.LeaveRequestId = id;

var v = await repositoryWrapper.LeaveRequestRepository.UpdateAsync(id, f);

await repositoryWrapper.SaveAsync();

var k = mapper.Map<LeaveRequestResponseDTO>(v);

return k;

}

}

}

using AutoMapper;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

using Databasehr.DAL.Entities;

using Databasehr.DAL.Repositories;

using Databasehr.BLL.Services;

namespace Databasehr.BLL.Services.Implementation

{

public class SalaryService : ISalaryService

{

private readonly IRepositoryWrapper repositoryWrapper;

private readonly IMapper mapper;

public SalaryService(IRepositoryWrapper repositoryWrapper, IMapper mapper)

{

this.repositoryWrapper = repositoryWrapper;

this.mapper = mapper;

}

public async Task<SalaryResponseDTO> Add(SalaryRequestDTO requestDTO)

{

var cart = mapper.Map<Salary>(requestDTO);

var cartResponse = await this.repositoryWrapper.SalaryRepository.CreateAsync(cart);

await this.repositoryWrapper.SaveAsync();

var result = mapper.Map<SalaryResponseDTO>(cartResponse);

return result;

}

public async Task<bool> Delete(int id)

{

var orderdetailtodelete = await this.repositoryWrapper.SalaryRepository.GetById(id);

if (orderdetailtodelete == null)

{

return false;

}

await this.repositoryWrapper.SalaryRepository.DeleteAsync(orderdetailtodelete);

await this.repositoryWrapper.SaveAsync();

return true;

}

public async Task<IEnumerable<SalaryResponseDTO>> GetAll()

{

var carts = await this.repositoryWrapper.SalaryRepository.GetAllAsync();

var result = mapper.Map<IEnumerable<SalaryResponseDTO>>(carts);

return result;

}

public async Task<SalaryResponseDTO> GetById(int id)

{

var data = await repositoryWrapper.SalaryRepository.GetById(id);

var res = mapper.Map<SalaryResponseDTO>(data);

return res;

}

public async Task<SalaryResponseDTO> Update(int id, SalaryRequestDTO res)

{

var f = mapper.Map<Salary>(res);

f.SalaryId = id;

var v = await repositoryWrapper.SalaryRepository.UpdateAsync(id, f);

await repositoryWrapper.SaveAsync();

var k = mapper.Map<SalaryResponseDTO>(v);

return k;

}

}

}

**Interface of each Table**

using AutoMapper;

using Databasehr.BLL.Services;

using Databasehr.DAL.Repositories;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

using Databasehr.DAL.Entities;

namespace Databasehr.BLL.Services

{

public interface IAttendanceService : IBaseService<AttendanceRequestDTO, AttendanceResponseDTO>

{

}

}

namespace Databasehr.BLL.Services

{

public interface IBaseService<TRequestDTO, TResponseDTO>

{

Task<IEnumerable<TResponseDTO>> GetAll();

Task<TResponseDTO> GetById(int id);

Task<TResponseDTO> Add(TRequestDTO requestDTO);

Task<bool> Delete(int id);

Task<TResponseDTO> Update(int id, TRequestDTO requestDTO);

}

}

using AutoMapper;

using Databasehr.BLL.Services;

using Databasehr.DAL.Repositories;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

using Databasehr.DAL.Entities;

namespace Databasehr.BLL.Services

{

public interface IDepartmentService : IBaseService<DepartmentRequestDTO, DepartmentResponseDTO>

{

}

}  
using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

namespace Databasehr.BLL.Services

{

public interface IEmployeeService : IBaseService<EmployeeRequestDTO, EmployeeResponseDTO>

{

Task<int> Login(LoginRequestDTO d);

}

}  
using AutoMapper;

using Databasehr.BLL.Services;

using Databasehr.DAL.Repositories;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

using Databasehr.DAL.Entities;

namespace Databasehr.BLL.Services

{

public interface ILeaveRequestService : IBaseService<LeaveRequestRequestDTO, LeaveRequestResponseDTO>

{

}

}  
using AutoMapper;

using Databasehr.BLL.Services;

using Databasehr.DAL.Repositories;

using Databasehr.BLL.DTO.Request;

using Databasehr.BLL.DTO.Response;

using Databasehr.DAL.Entities;

namespace Databasehr.BLL.Services

{

public interface ISalaryService : IBaseService<SalaryRequestDTO, SalaryResponseDTO>

{

}

}