Model

Supplier

using System.ComponentModel.DataAnnotations;

namespace SupplierAndProductManagement.Model

{

public class Supplier

{

[Key]

public int SupplierId { get; set; }

[Required(ErrorMessage = "Please enter Supplier Name ")]

public string Name { get; set; }

[DataType(DataType.EmailAddress)]

[RegularExpression(@"^([0-9a-zA-Z]([-\.\w]\*[0-9a-zA-Z])\*@([0-9a-zA-Z][-\w]\*[0-9a-zA-Z]\.)+[a-zA-Z]{2,9})$", ErrorMessage = "Invalid email address")]

public string Email { get; set; }

[Required(ErrorMessage = "Doctor's phone number is manditory")]

[RegularExpression(@"^(\+91[\-\s]?)?[0]?(91)?[789]\d{9}$", ErrorMessage = "Invalid phone number")]

public string Phone { get; set; }

public ICollection<Product>? Products { get; set; }

}

}

Product

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace SupplierAndProductManagement.Model

{

public class Product

{

[Key]

public int ProductId { get; set; }

[Required(ErrorMessage = "Please enter Product Name ")]

public string Name { get; set; }

[Required(ErrorMessage = "Please Enter Product Description ")]

public string Description { get; set; }

[Required(ErrorMessage = "Please Product Price ")]

public float Price { get; set; }

public int SupplierId { get; set; }

[ForeignKey("SupplierId")]

public Supplier? Supplier { get; set; }

}

}

Dtos

namespace SupplierAndProductManagement.Model.DTOs

{

public class UpdateProductDescriptionDTO

{

public int Id { get; set; }

public string Description { get; set; }

}

}

namespace SupplierAndProductManagement.Model.DTOs

{

public class UpdateProductNameDTO

{

public int Id { get; set; }

public string Name { get; set; }

}

}

namespace SupplierAndProductManagement.Model.DTOs

{

public class UpdateProductPriceDTO

{

public int Id { get; set; }

public int Price { get; set; }

}

}

namespace SupplierAndProductManagement.Model.DTOs

{

public class UpdateSupplierEmailDTO

{

public int Id { get; set; }

public string Email { get; set; }

}

}

namespace SupplierAndProductManagement.Model.DTOs

{

public class UpdateSupplierNameDTO

{

public int Id { get; set; }

public string Name { get; set; }

}

}

namespace SupplierAndProductManagement.Model.DTOs

{

public class UpdatesupplierPhoneDTO

{

public int Id { get; set; }

public string Phone { get; set; }

}

}

MContext

using Microsoft.EntityFrameworkCore;

using SupplierAndProductManagement.Model;

namespace SupplierAndProductManagement.ManageContext

{

public class MContext : DbContext

{

public MContext(DbContextOptions opts) : base(opts) { }

public DbSet<Supplier> suppliers { get; set; }

public DbSet<Product> products { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Supplier>().HasData(

new Supplier { SupplierId = 1 , Email="abc@gmail.com", Name= "ABC" , Phone="8856904770"},

new Supplier { SupplierId = 2, Email = "def@gmail.com", Name = "DEF", Phone = "8856904771" }

);

modelBuilder.Entity<Product>().HasData(

new Product { ProductId = 1 , Name = "LG", Description="fidge" , Price=20000 , SupplierId = 1},

new Product { ProductId = 2, Name = "LG", Description = "fidge", Price = 20000, SupplierId = 1 }

);

}

}

}

Interfaces

IRepository

namespace SupplierAndProductManagement.Interfaces

{

public interface IRepository<K,T>

{

public List<T> GetAll();

public T Get(K key);

public T Add(T item);

public T Delete(K key);

public T Update(T item);

}

}

IProductServices

using SupplierAndProductManagement.Model;

using SupplierAndProductManagement.Model.DTOs;

namespace SupplierAndProductManagement.Interfaces

{

public interface IProductService

{

Product AddNewProduct(Product product);

List<Product> GetAllProducts();

Product UpdateProductDescription(UpdateProductDescriptionDTO product);

Product UpdateProductName(UpdateProductNameDTO product);

Product UpdateProductPrice(UpdateProductPriceDTO product);

}

}

ISepplierService

using SupplierAndProductManagement.Model;

using SupplierAndProductManagement.Model.DTOs;

namespace SupplierAndProductManagement.Interfaces

{

public interface ISupplierServices

{

List<Supplier> GetAllSupplier();

Supplier AddNewSupplier(Supplier supplier);

Supplier UpdateSupplierName(UpdateSupplierNameDTO updateSupplierNameDTO);

Supplier UpdateSupplierEmail(UpdateSupplierEmailDTO updateSupplierEmailDTO);

Supplier UpdateSupplierPhone(UpdatesupplierPhoneDTO updatedSupplierPhoneDTO);

}

}

Repositories

using Microsoft.EntityFrameworkCore.Migrations;

using SupplierAndProductManagement.Interfaces;

using SupplierAndProductManagement.ManageContext;

using SupplierAndProductManagement.Model;

namespace SupplierAndProductManagement.Repositories

{

public class SupplierRepository : IRepository<int, Supplier>

{

#region injection

private readonly MContext \_context;

public SupplierRepository(MContext context)

{

\_context = context;

}

#endregion

#region Add

public Supplier Add(Supplier item)

{

\_context.suppliers.Add(item);

\_context.SaveChanges();

return item;

}

#endregion

#region Delete

public Supplier Delete(int key)

{

var suuplier = Get(key);

if(suuplier != null)

{

\_context.suppliers.Remove(suuplier);

\_context.SaveChanges();

return suuplier;

}

return null;

}

#endregion

#region Get

public Supplier Get(int key)

{

var supplier = \_context.suppliers.FirstOrDefault(s => s.SupplierId == key);

return supplier;

}

#endregion

#region GetAll

public List<Supplier> GetAll()

{

return \_context.suppliers.ToList();

}

#endregion

#region Update

public Supplier Update(Supplier item)

{

\_context.Entry<Supplier>(item).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

\_context.SaveChanges();

return item;

}

#endregion

}

}

ProductRepository

using SupplierAndProductManagement.Interfaces;

using SupplierAndProductManagement.ManageContext;

using SupplierAndProductManagement.Model;

namespace SupplierAndProductManagement.Repositories

{

public class ProductRepository : IRepository<int , Product>

{

#region injection

private readonly MContext \_context;

public ProductRepository(MContext context)

{

\_context = context;

}

#endregion

#region Add

public Product Add(Product item)

{

\_context.products.Add(item);

\_context.SaveChanges();

return item;

}

#endregion

#region Delete

public Product Delete(int key)

{

var product = Get(key);

if (product != null)

{

\_context.products.Remove(product);

\_context.SaveChanges();

return product;

}

return null;

}

#endregion

#region Get

public Product Get(int key)

{

var product = \_context.products.FirstOrDefault(p=>p.ProductId == key);

return product;

}

#endregion

#region GetAll

public List<Product> GetAll()

{

return \_context.products.ToList();

}

#endregion

#region Update

public Product Update(Product item)

{

\_context.Entry<Product>(item).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

\_context.SaveChanges();

return item;

}

#endregion

}

}

Service

productService

using SupplierAndProductManagement.Interfaces;

using SupplierAndProductManagement.Model;

using SupplierAndProductManagement.Model.DTOs;

namespace SupplierAndProductManagement.Services

{

public class ProductService :IProductService

{

private readonly IRepository<int, Product> \_repo;

public ProductService(IRepository<int , Product> repository)

{

\_repo=repository;

}

public Product AddNewProduct(Product product)

{

return \_repo.Add(product);

}

public List<Product> GetAllProducts()

{

return \_repo.GetAll();

}

public Product UpdateProductDescription(UpdateProductDescriptionDTO product)

{

var pro = \_repo.Get(product.Id);

if(pro != null)

{

pro.Description = product.Description;

return \_repo.Update(pro);

}

return null;

}

public Product UpdateProductName(UpdateProductNameDTO product)

{

var pro = \_repo.Get(product.Id);

if (pro != null)

{

pro.Name = product.Name;

return \_repo.Update(pro);

}

return null;

}

public Product UpdateProductPrice(UpdateProductPriceDTO product)

{

var pro = \_repo.Get(product.Id);

if (pro != null)

{

pro.Price = product.Price;

return \_repo.Update(pro);

}

return null;

}

}

}

SupplierService

using Microsoft.AspNetCore.Mvc;

using SupplierAndProductManagement.Interfaces;

using SupplierAndProductManagement.Model;

using SupplierAndProductManagement.Model.DTOs;

namespace SupplierAndProductManagement.Services

{

public class SupplierServices : ISupplierServices

{

private readonly IRepository<int, Supplier> \_repo;

public SupplierServices( IRepository<int , Supplier> repository )

{

\_repo= repository;

}

#region Add

public Supplier AddNewSupplier(Supplier supplier)

{

return \_repo.Add(supplier);

}

#endregion

#region GetAll Supplier

public List<Supplier> GetAllSupplier()

{

return \_repo.GetAll();

}

#endregion

#region Update Email

public Supplier UpdateSupplierEmail(UpdateSupplierEmailDTO updateSupplierEmailDTO)

{

var emp = \_repo.Get(updateSupplierEmailDTO.Id);

if (emp != null)

{

emp.Email = updateSupplierEmailDTO.Email;

return \_repo.Update(emp);

}

return null;

}

#endregion

#region Update Name

public Supplier UpdateSupplierName(UpdateSupplierNameDTO updateSupplierNameDTO)

{

var emp = \_repo.Get(updateSupplierNameDTO.Id);

if (emp != null)

{

emp.Name = updateSupplierNameDTO.Name;

return \_repo.Update(emp);

}

return null;

}

#endregion

#region Update Phone

public Supplier UpdateSupplierPhone(UpdatesupplierPhoneDTO updatedSupplierPhoneDTO)

{

var emp = \_repo.Get(updatedSupplierPhoneDTO.Id);

if (emp != null)

{

emp.Phone = updatedSupplierPhoneDTO.Phone;

return \_repo.Update(emp);

}

return null;

}

#endregion

}

}

Controller

SupplierController

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using SupplierAndProductManagement.Interfaces;

using SupplierAndProductManagement.Model;

using SupplierAndProductManagement.Model.DTOs;

namespace SupplierAndProductManagement.Controllers

{

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

[ApiController]

public class SupplierController : ControllerBase

{

private readonly ISupplierServices \_service;

private readonly IRepository<int, Supplier> \_repo;

public SupplierController(ISupplierServices supplierServices , IRepository<int , Supplier> repository)

{

\_service = supplierServices;

\_repo= repository;

}

#region Get All Supplier

[HttpGet("GetAllSuppliers")]

public IActionResult Get()

{

var result = \_service.GetAllSupplier();

if(result == null)

{

return NotFound("No Supplier Found");

}

return Ok(result);

}

#region Delete

[HttpDelete]

public ActionResult Delete(int id)

{

try

{

var result = \_repo.Delete(id);

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

#endregion

#endregion

#region AddSupplier

[HttpPost("addSupplier")]

public ActionResult AddSupplier(Supplier supplier)

{

if(ModelState.IsValid)

{

try

{

var result = \_service.AddNewSupplier(supplier);

return Created("",result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

return BadRequest(ModelState.Keys);

}

#endregion

#region UpdateName

[HttpPut("UpdateEmail")]

public ActionResult UpdateEmail(UpdateSupplierEmailDTO email) {

try

{

var result = \_service.UpdateSupplierEmail(email);

if (result == null)

return NotFound();

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

#endregion

#region Update Name

[HttpPut("UpadeName")]

public ActionResult UpdateName(UpdateSupplierNameDTO name)

{

try

{

var result = \_service.UpdateSupplierName(name);

if(result == null)

return NotFound();

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

#endregion

#region update phone

[HttpPut("updatePhone")]

public ActionResult updatePhone(UpdatesupplierPhoneDTO phone)

{

try

{

var result = \_service.UpdateSupplierPhone(phone);

if(result == null)

return NotFound();

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

#endregion

}

}

ProductController

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using SupplierAndProductManagement.Interfaces;

using SupplierAndProductManagement.Model;

using SupplierAndProductManagement.Model.DTOs;

using SupplierAndProductManagement.Services;

namespace SupplierAndProductManagement.Controllers

{

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

[ApiController]

public class ProductController : ControllerBase

{

private readonly IProductService \_service;

private readonly IRepository<int, Product> \_repo;

public ProductController(IProductService service, IRepository<int, Product> repository)

{

\_service = service;

\_repo = repository;

}

#region Get

[HttpGet]

public ActionResult Get()

{

var result = \_service.GetAllProducts();

if (result == null)

{

return NotFound("No products are there at this moment");

}

return Ok(result);

}

#endregion

#region Add Product

[HttpPost]

public ActionResult AddProduct(Product product)

{

if (ModelState.IsValid)

{

try

{

var result = \_service.AddNewProduct(product);

return Created("", result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

return BadRequest(ModelState.Keys);

}

#endregion

#region Update Price

[HttpPut("Price")]

public ActionResult UpdatePrice(UpdateProductPriceDTO price)

{

try

{

var result = \_service.UpdateProductPrice(price);

if (result == null)

return NotFound();

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

#endregion

#region Update Price

[HttpPut("Description")]

public ActionResult UpdateDec(UpdateProductDescriptionDTO d)

{

try

{

var result = \_service.UpdateProductDescription(d);

if (result == null)

return NotFound();

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

#endregion

#region Update Price

[HttpPut("Name")]

public ActionResult UpdateName(UpdateProductNameDTO name)

{

try

{

var result = \_service.UpdateProductName(name);

if (result == null)

return NotFound();

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

#endregion

#region Delete

[HttpDelete]

public ActionResult DeleteProduct(int id)

{

try

{

var result = \_repo.Delete(id);

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

#endregion

}

}

Application.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"ConnectionStrings": {

"DefaultConnection": "Host=localhost;Port=5432;Database=dbSPManagement;UserId=postgres;Password=root;"

},

"AllowedHosts": "\*"

}

Program.cs

using Microsoft.EntityFrameworkCore;

using SupplierAndProductManagement.Interfaces;

using SupplierAndProductManagement.ManageContext;

using SupplierAndProductManagement.Model;

using SupplierAndProductManagement.Repositories;

using SupplierAndProductManagement.Services;

namespace SupplierAndProductManagement

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<MContext>(opts =>

{

opts.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection"));

});

builder.Services.AddScoped<IRepository<int , Supplier> , SupplierRepository > ();

builder.Services.AddScoped<IRepository<int, Product>, ProductRepository>();

builder.Services.AddScoped<ISupplierServices, SupplierServices>();

builder.Services.AddScoped<IProductService , ProductService>();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseAuthorization();

app.MapControllers();

app.Run();

}

}

}