Enrollment No: 202203103510097

PRACTICAL 12

Aim: Develop ASP.NET Core Web API which retrieves data from MySQL database. Also create APIs to create, delete, and update objects.

Code:

Add Entity Framework Core and Tools

```
>> dotnet add package Pomelo.EntityFrameworkCore.MySql
>> dotnet add package Microsoft.EntityFrameworkCore.Design
>> dotnet add package Swashbuckle.AspNetCore
```

Configure Database in 'appsettings.json' :

```
"ConnectionStrings": {
   "DefaultConnection": "Server=localhost;Port=3306;Database=PRACTICAL12db;User
ID=root;Password=yourPassword!;"
  },
```

Program.cs :

```
using Microsoft.EntityFrameworkCore;
using PRACTICAL12.Data;
using Microsoft.Extensions.DependencyInjection;
builder.Services.AddDbContext<ApplicationDbContext>(options =>
   options.UseMySql(builder.Configuration.GetConnectionString("DefaultConnection"),
   new MySqlServerVersion(new Version(8, 0, 21)))); // Adjust version as needed
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
if (app.Environment.IsDevelopment())
{
   app.UseDeveloperExceptionPage();
   app.UseSwagger();
   app.UseSwaggerUI(c =>
   {
      c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");
      c.RoutePrefix = "swagger";
   });
}
```

Product.cs : (Models / Product.cs)

```
using System.ComponentModel.DataAnnotations;
namespace PRACTICAL12.Models {
   public class Product {
      [Key]
```

```
        public int Id { get; set; }
        public string Name { get; set; } = string.Empty;
        public decimal Price { get; set; }
        public string Description { get; set; } = string.Empty;
    }
}
```

ApplicationDbContext.cs : (Data / ApplicationDbContext.cs)

```
using Microsoft.EntityFrameworkCore;
using PRACTICAL12.Models;
namespace PRACTICAL12.Data {
    public class ApplicationDbContext : DbContext {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) :
base(options) { }
        public DbSet<Product> Products { get; set; }
    }
}
```

ProductsController.cs : (Controllers / ProductsController.cs)

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using PRACTICAL12.Data;
using PRACTICAL12.Models;
using System.Collections.Generic;
using System.Threading.Tasks;
namespace PRACTICAL12.Controllers {
    [Route("api/[controller]")]
    [ApiController]
    public class ProductsController : ControllerBase {
        private readonly ApplicationDbContext _context;
        public ProductsController(ApplicationDbContext context) {
            _context = context;
        }
        // GET: api/products
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Product>>> GetProducts() {
            return await _context.Products.ToListAsync();
        }
        // GET: api/products/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Product>> GetProduct(int id) {
            var product = await _context.Products.FindAsync(id);
            if (product == null) {
                return NotFound();
```

```
        }
        return product;
    }
    // POST: api/products
    [HttpPost]
    public async Task<ActionResult<Product>> PostProduct(Product product) {
        _context.Products.Add(product);
        await _context.SaveChangesAsync();
        return CreatedAtAction(nameof(GetProduct), new { id = product.Id }, product);
    }
    // PUT: api/products/5
    [HttpPut("{id}")]
    public async Task<IActionResult> PutProduct(int id, Product product) {
        if (id != product.Id) {
            return BadRequest();
        }
        try {
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException) {
            if (!ItemExists(id)) return NotFound();
            throw;
        }
        return NoContent();
    }
    // DELETE: api/products/5
    [HttpDelete("{id}")]
    public async Task<IActionResult> DeleteProduct(int id) {
        var product = await _context.Products.FindAsync(id);
        if (product == null) {
            return NotFound();
        }
        _context.Products.Remove(product);
        await _context.SaveChangesAsync();
        return NoContent();
    }
    private bool ItemExists(int id) {
        return _context.Products.Any(e => e.Id == id);
    }
  }
}
```

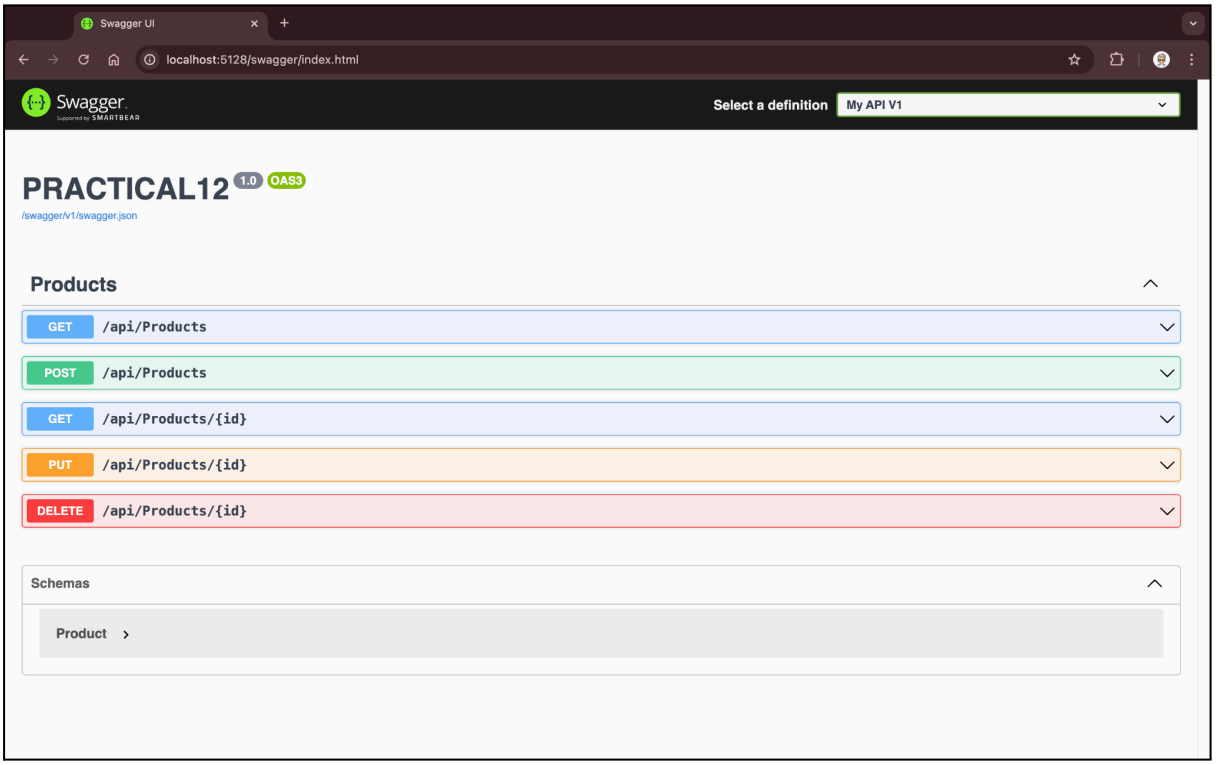
Add Migrations and Update Database :

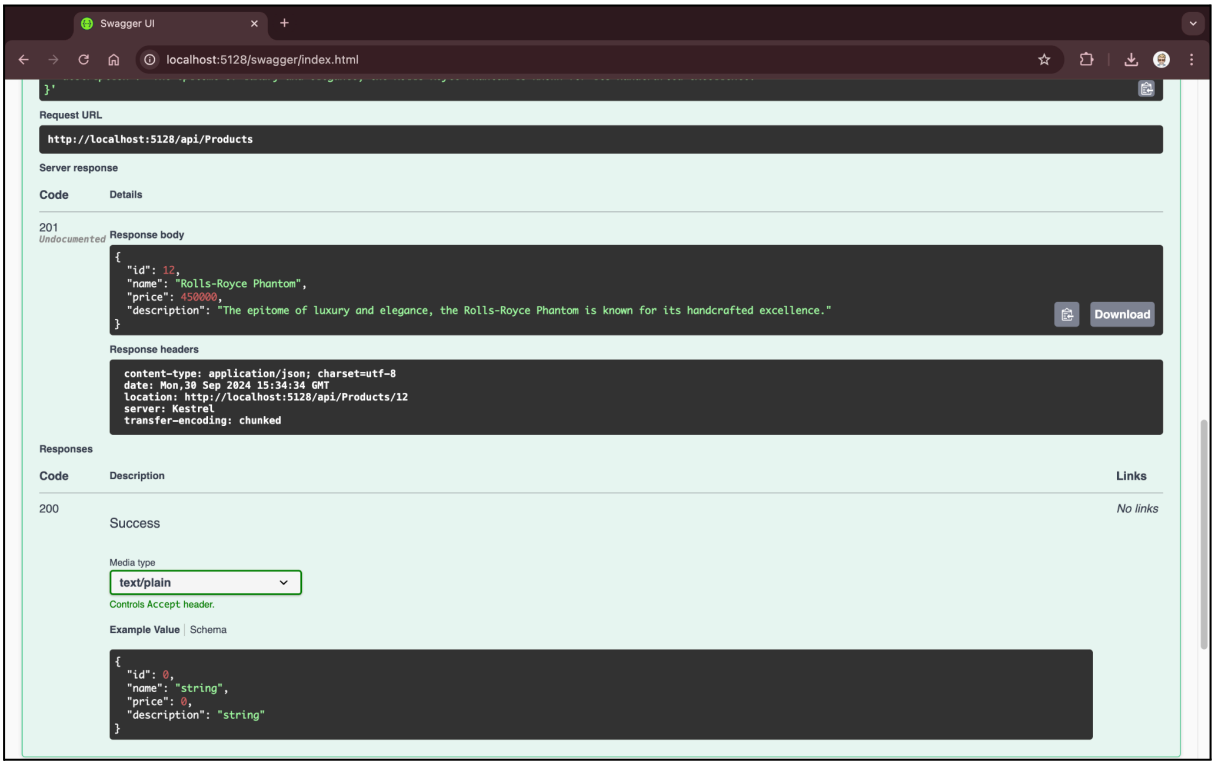
>> dotnet ef migrations add InitialCreate

>> dotnet ef database update

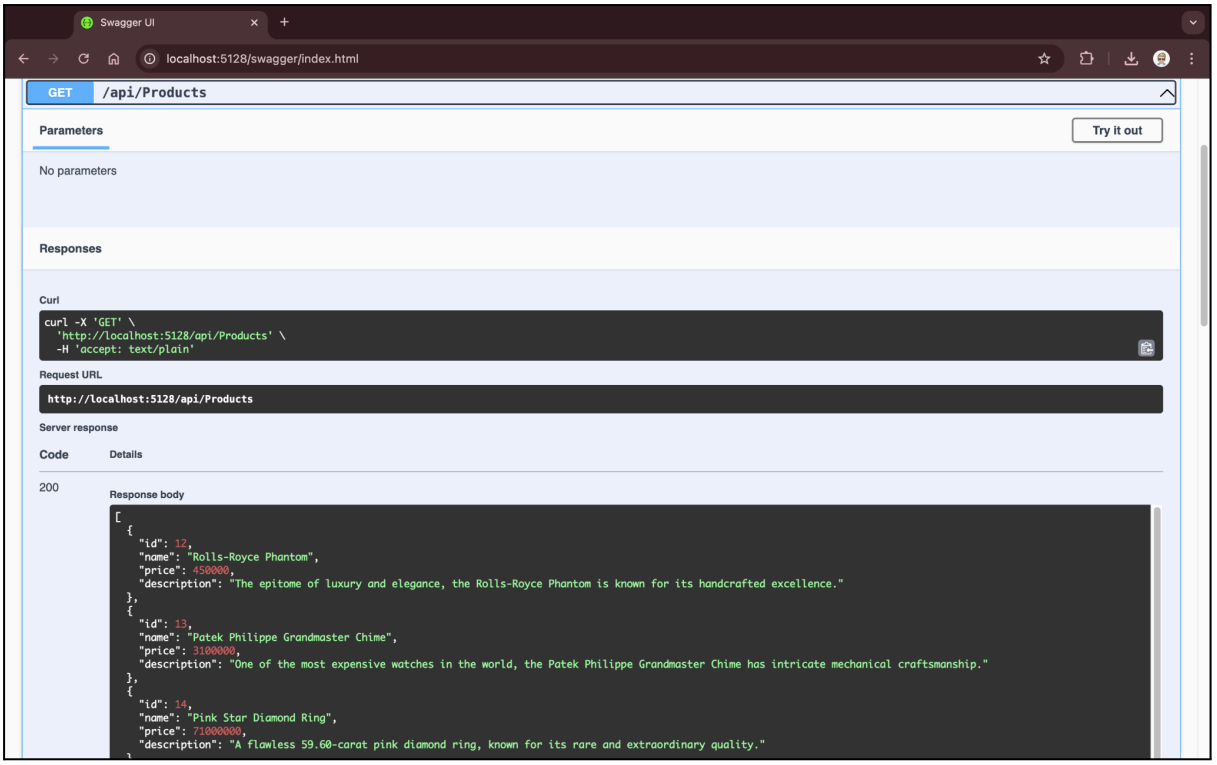
Enrollment No: 202203103510097

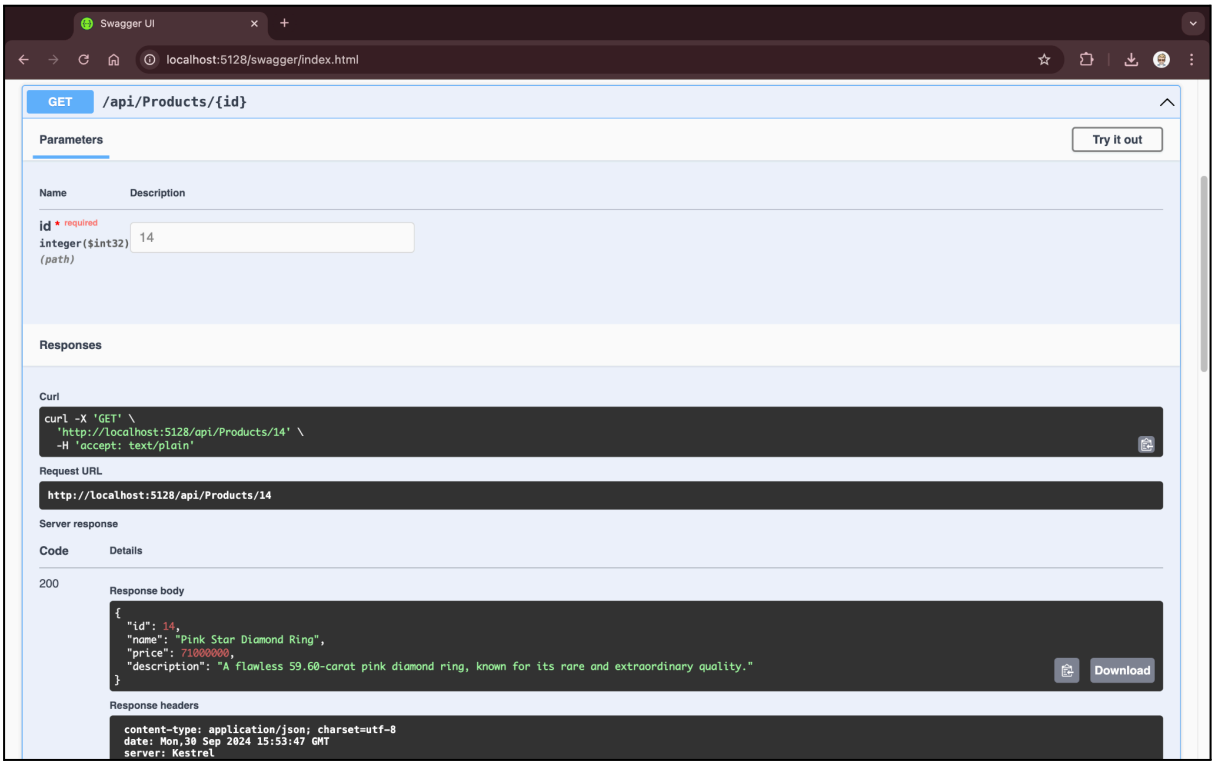
Output:





Enrollment No: 202203103510097





Enrollment No: 202203103510097