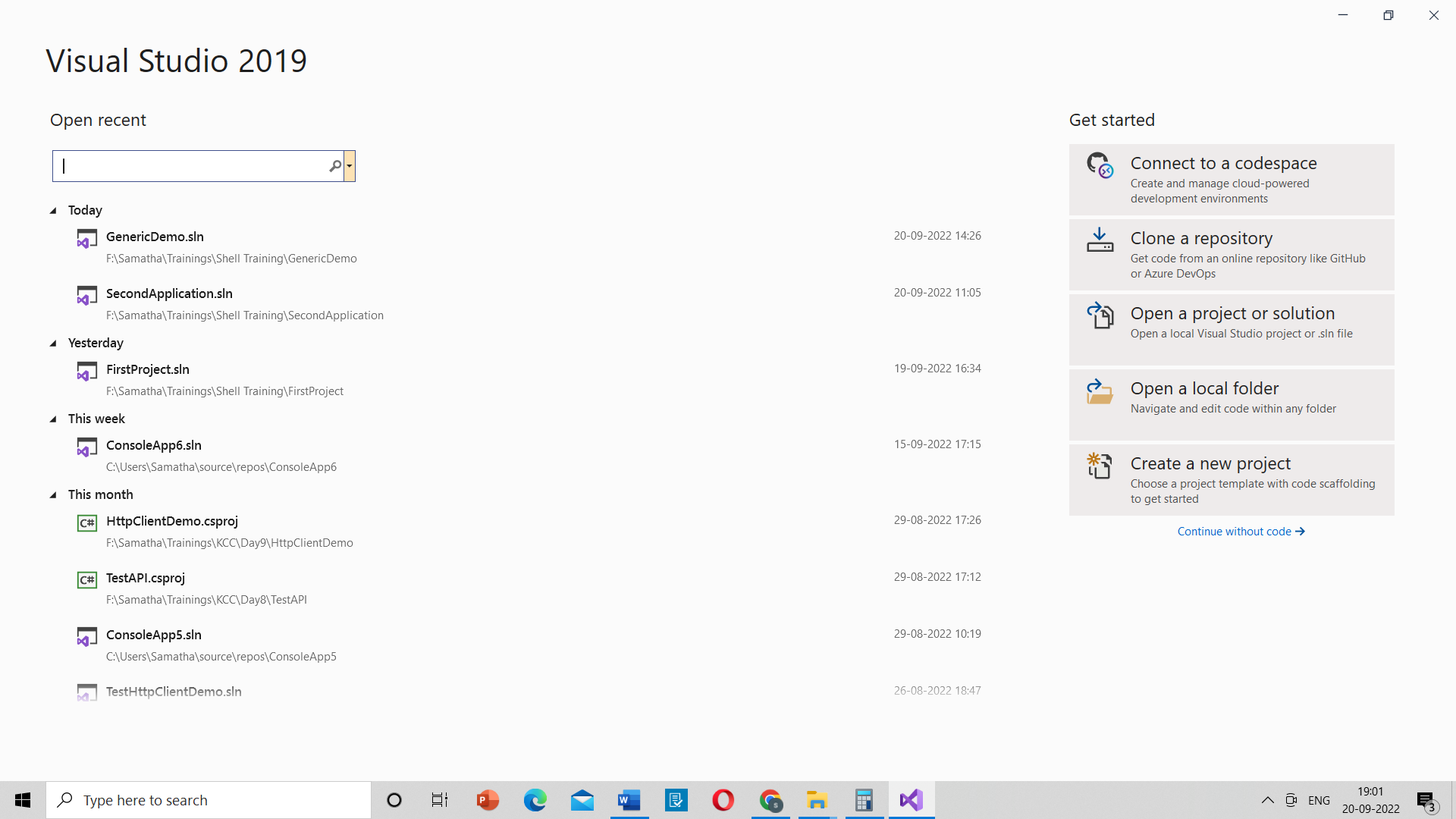
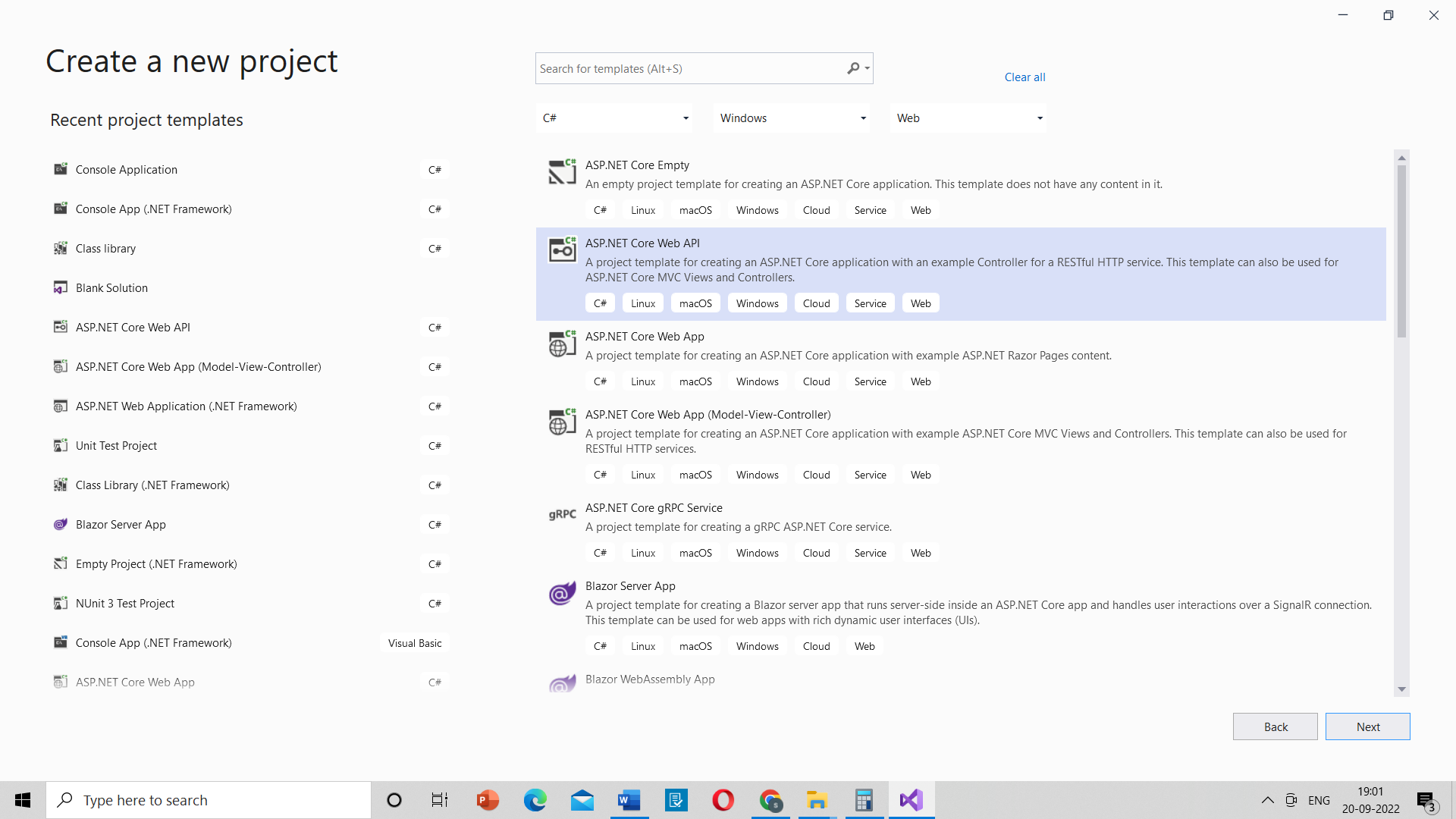
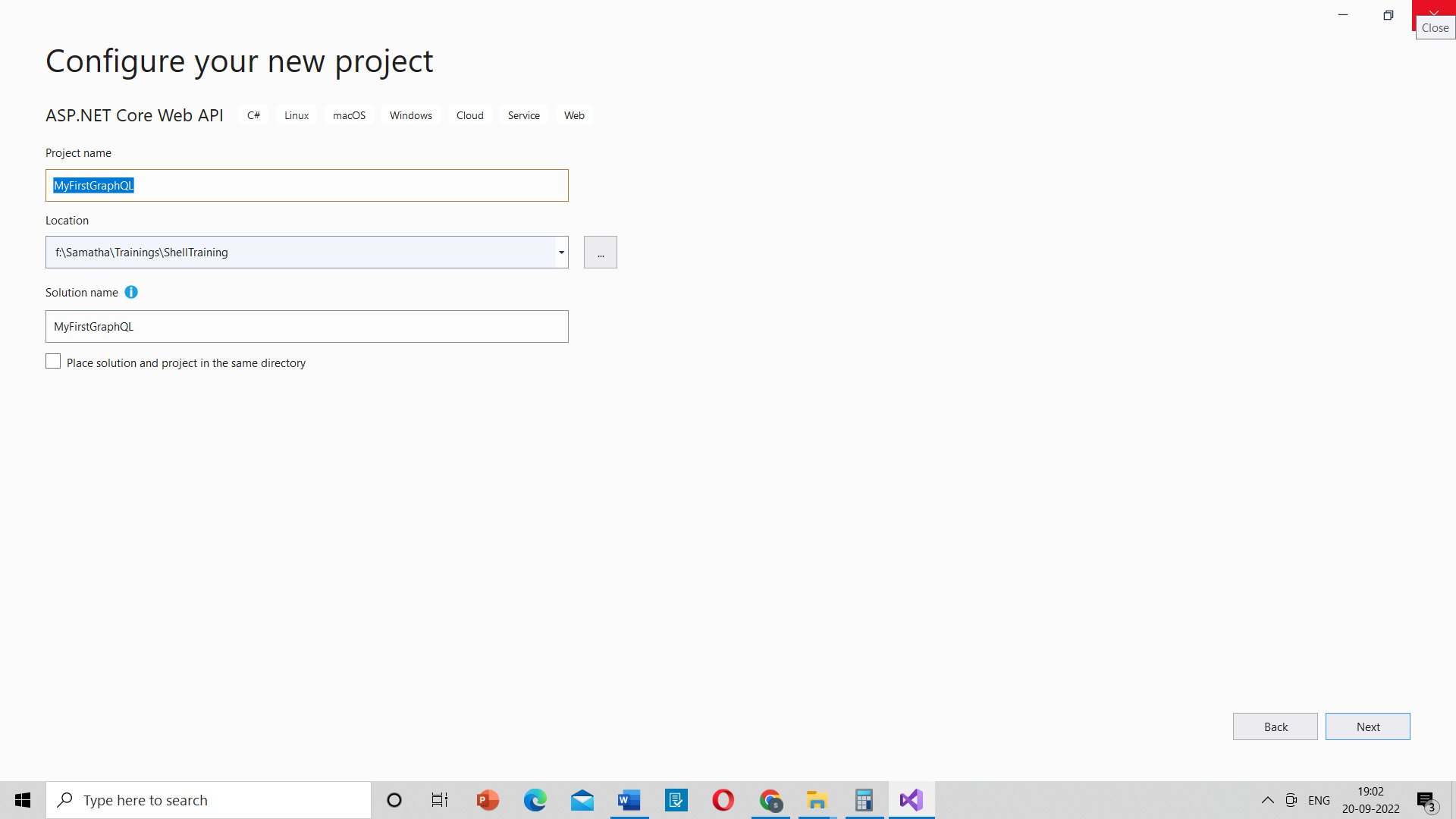
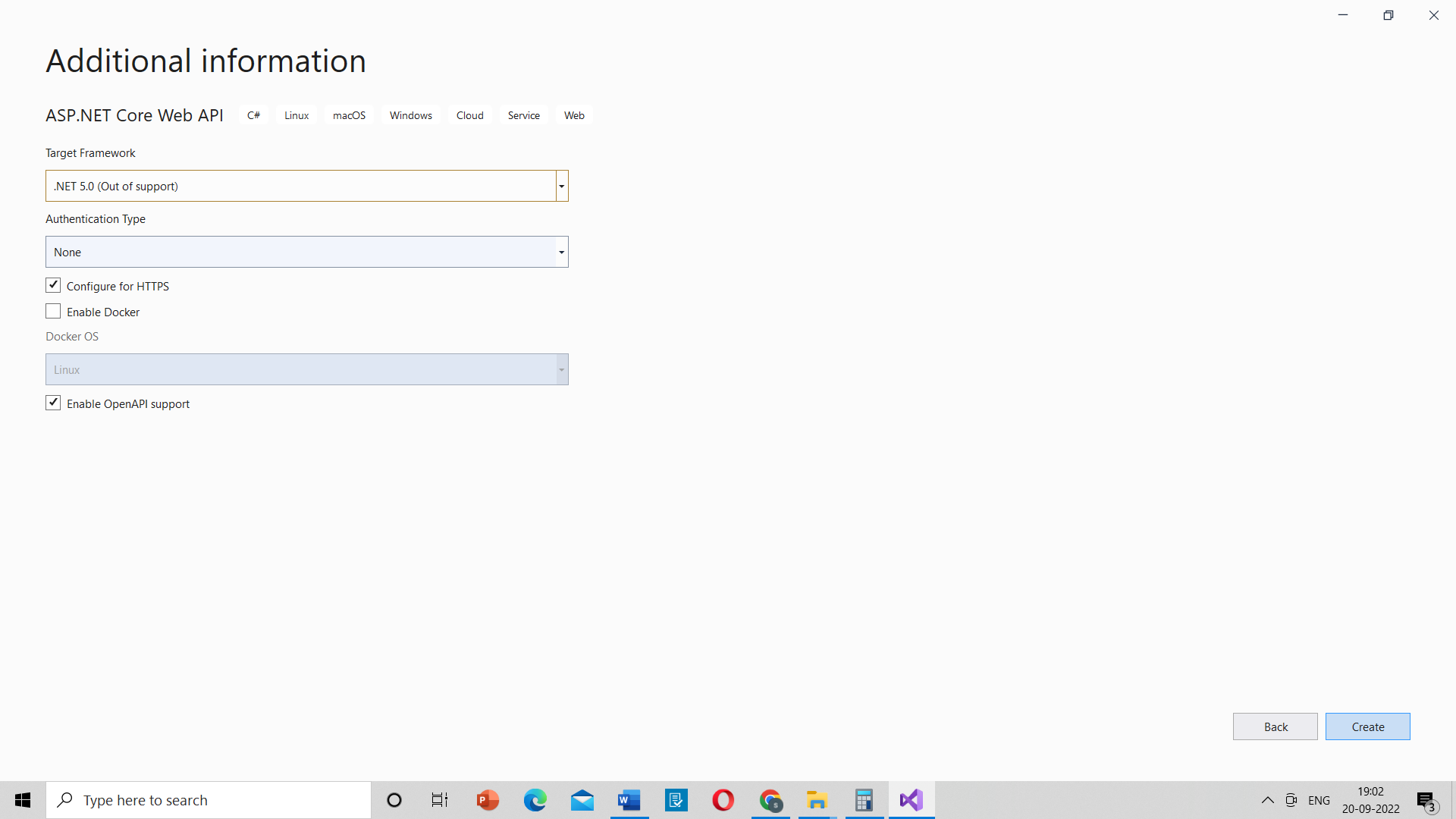
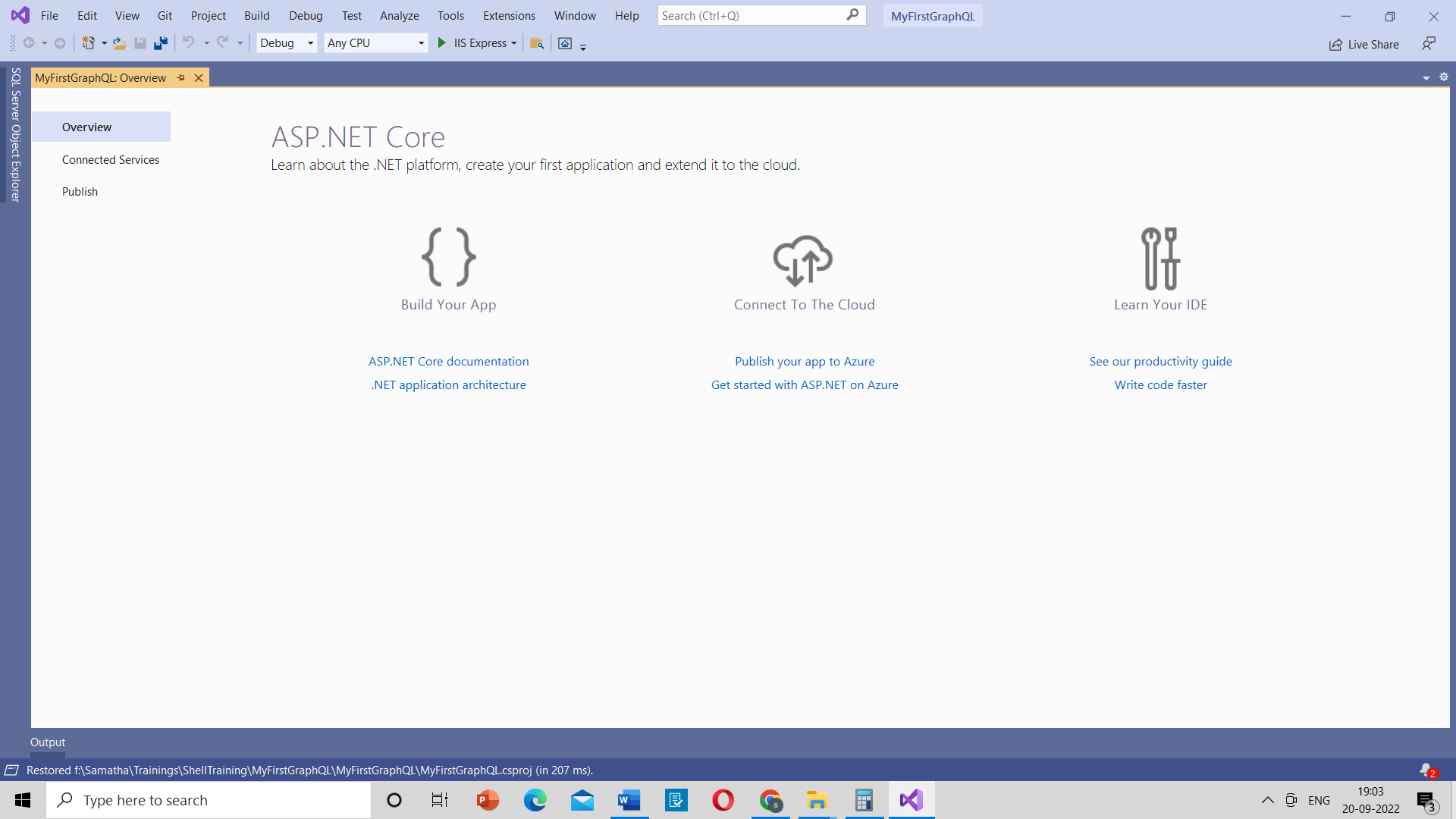
Step1: Create .Net Core API project



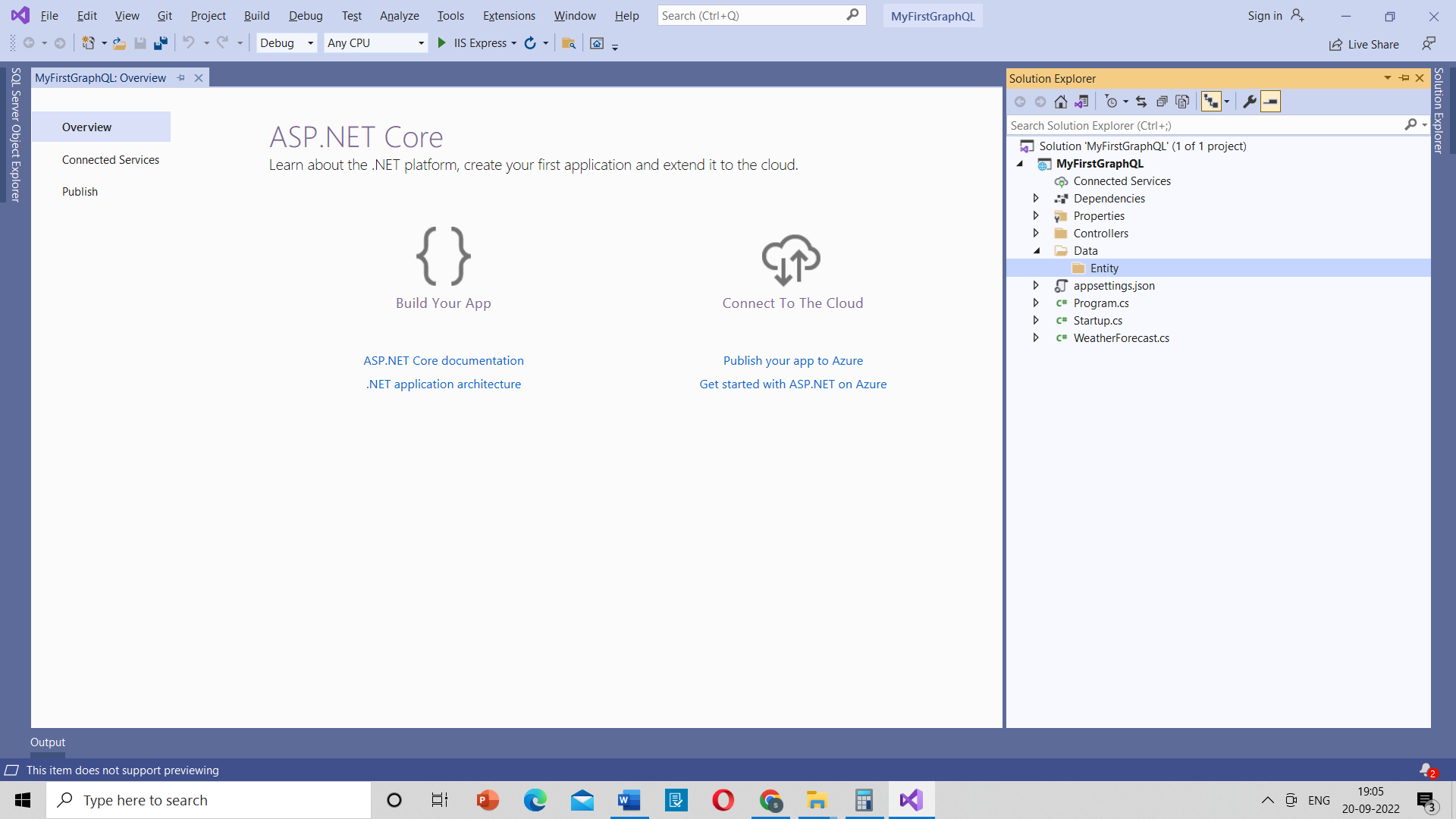








Step2: Data Folder 🡪 Entity SubFolder 🡪 Product Class



Step3: Create a DBContext class under Data Folder

public class Product

{

public int Id { get; set; }

[StringLength(100)]

public string Name { get; set; }

public ProductType Type { get; set; }

public string Description { get; set; }

[Column(TypeName = "decimal(18,2)")]

public decimal Price { get; set; }

public int Stock { get; set; }

public int Rating { get; set; }

public DateTimeOffset IntroducedAt { get; set; }

[StringLength(100)]

public string PhotoFileName { get; set; }

}

Step4: Create a class Initial Data and Seed Method

public static class InitialData

{

public static void Seed(this CarvedRockDbContext dbContext)

{

if (!dbContext.Products.Any())

{

dbContext.Products.Add(new Product

{

Name = "Mountain Walkers",

Description = "Use these sturdy shoes to pass any mountain range with ease.",

Price = 219.5m,

Rating = 4,

Type = ProductType.Boots,

Stock = 12,

PhotoFileName = "shutterstock\_66842440.jpg",

IntroducedAt = DateTimeOffset.Now.AddMonths(-1)

});

dbContext.Products.Add(new Product

{

Name = "Army Slippers",

Description = "For your everyday marches in the army.",

Price = 125.9m,

Rating = 3,

Type = ProductType.Boots,

Stock = 56,

PhotoFileName = "shutterstock\_222721876.jpg",

IntroducedAt = DateTimeOffset.Now.AddMonths(-1)

});

dbContext.Products.Add(new Product

{

Name = "Backpack Deluxe",

Description = "This backpack can survive any tornado.",

Price = 199.99m,

Rating = 5,

Type = ProductType.ClimbingGear,

Stock = 66,

PhotoFileName = "shutterstock\_6170527.jpg",

IntroducedAt = DateTimeOffset.Now.AddMonths(-1)

});

dbContext.Products.Add(new Product

{

Name = "Climbing Kit",

Description = "Anything you need to climb the mount Everest.",

Price = 299.5m,

Rating = 5,

Type = ProductType.ClimbingGear,

Stock = 3,

PhotoFileName = "shutterstock\_48040747.jpg",

IntroducedAt = DateTimeOffset.Now.AddMonths(-1)

});

dbContext.Products.Add(new Product

{

Name = "Blue Racer",

Description = "Simply the fastest kayak on earth and beyond for 2 persons.",

Price = 350m,

Rating = 5,

Type = ProductType.Kayaks,

Stock = 8,

PhotoFileName = "shutterstock\_441989509.jpg",

IntroducedAt = DateTimeOffset.Now.AddMonths(-1)

});

dbContext.Products.Add(new Product

{

Name = "Orange Demon",

Description = "One person kayak with hyper boost button.",

Price = 450m,

Rating = 2,

Type = ProductType.Kayaks,

Stock = 1,

PhotoFileName = "shutterstock\_495259978.jpg",

IntroducedAt = DateTimeOffset.Now.AddMonths(-1)

});

dbContext.SaveChanges();

}

}

Step5: In Startup.cs 🡪 In ConfigureServices 🡪 AddDBContext 🡪AddSingleton

private readonly IConfiguration \_config;

//private readonly IHostingEnvironment \_env;

private readonly CarvedRockDbContext dbContext;

public Startup(IConfiguration config, CarvedRockDbContext \_dbContext )

{

\_config = config;

dbContext = \_dbContext;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "MyFirstGraphQL", Version = "v1" });

});

services.AddDbContext<CarvedRockDbContext>(options =>

options.UseSqlServer(Configuration["ConnectionStrings:CarvedRock"]));

services.AddScoped<ProductRepository>();

Step6: In Startup.cs 🡪 In Configure 🡪 dbContext.Seed

dbContext.Seed();

Step7: Create Repositories folder create a Repository Class

public class ProductRepository

{

private readonly CarvedRockDbContext \_dbContext;

public ProductRepository(CarvedRockDbContext dbContext)

{

\_dbContext = dbContext;

}

public Task<List<Product>> GetAll()

{

return \_dbContext.Products.ToListAsync();

}

}

Step8: Install the following nuget packages

GraphQL

GraphQL.Server.Transports.AspNetCore

GraphQL.Server.UI.PlayGround

Step9: Create a folder GraphQL 🡪 Create a class CarvedRockSchema : Schema

public class CarvedRockSchema : Schema

{

public CarvedRockSchema(IServiceProvider resolver) : base(resolver)

{

Query = (IObjectGraphType)resolver.GetService(typeof(CarvedRockQuery));

}

}

Step10: Create a folder Types 🡪 create a class ProductType : ObjectGraphType<Product>

public class ProductType : ObjectGraphType<Product>

{

public ProductType()

{

Field(t => t.Id);

Field(t => t.Name).Description("The name of the product");

Field(t => t.Description);

Field(t => t.IntroducedAt).Description("When the product was first introduced in the catalog");

Field(t => t.PhotoFileName).Description("The file name of the photo so the client can render it");

Field(t => t.Price);

Field(t => t.Rating).Description("The (max 5) star customer rating");

Field(t => t.Stock);

Field<ProductTypeEnumType>("Type", "The type of product");

}

}

Step11: Create a class in GraphQL folder

public class CarvedRockQuery : ObjectGraphType

{

public CarvedRockQuery(ProductRepository productRepository)

{

Field<ListGraphType<MyFirstGraphQL.GraphQL.Types.ProductType>>(

"products",

resolve: context => productRepository.GetAll()

);

}

}