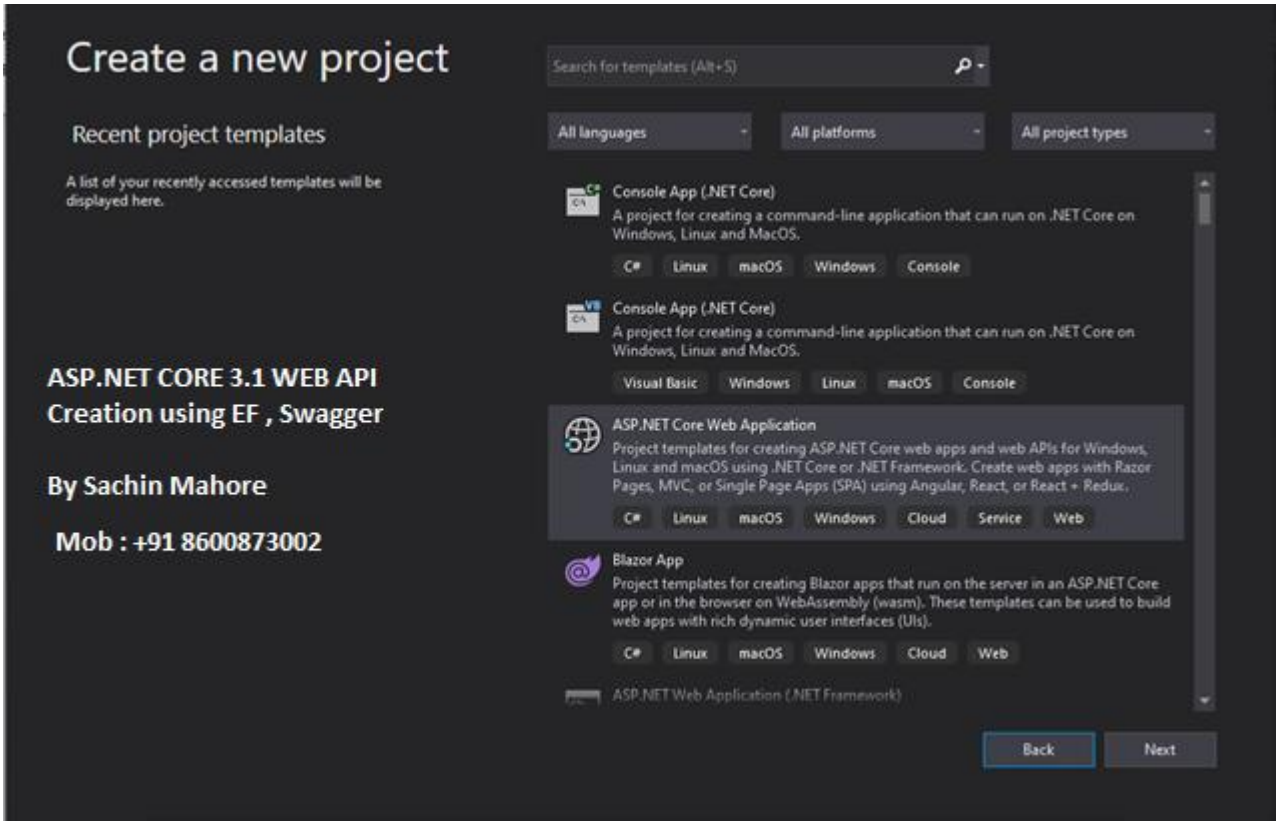
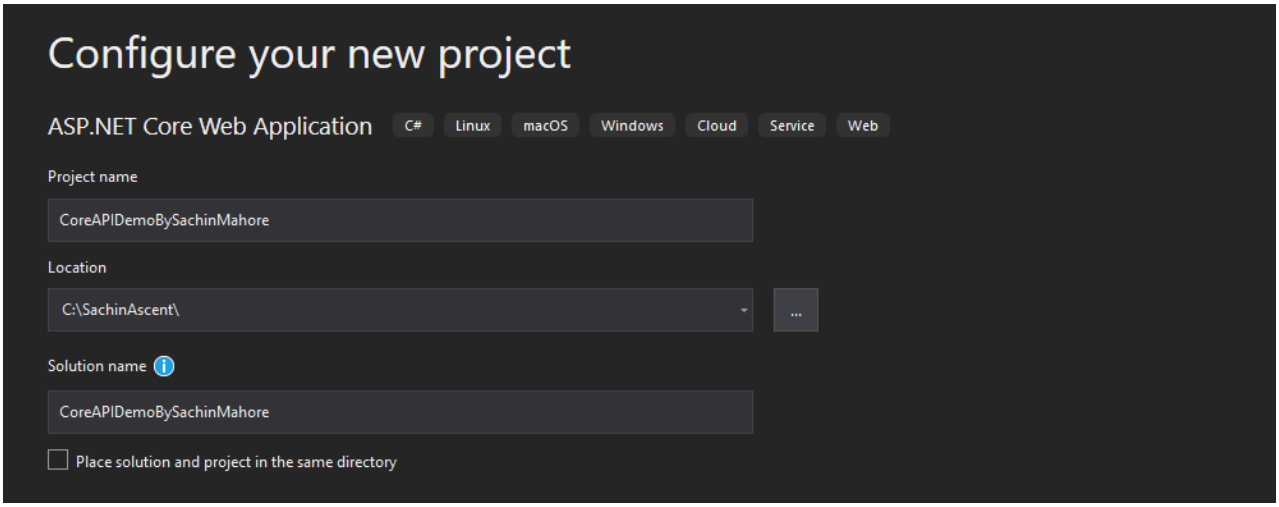
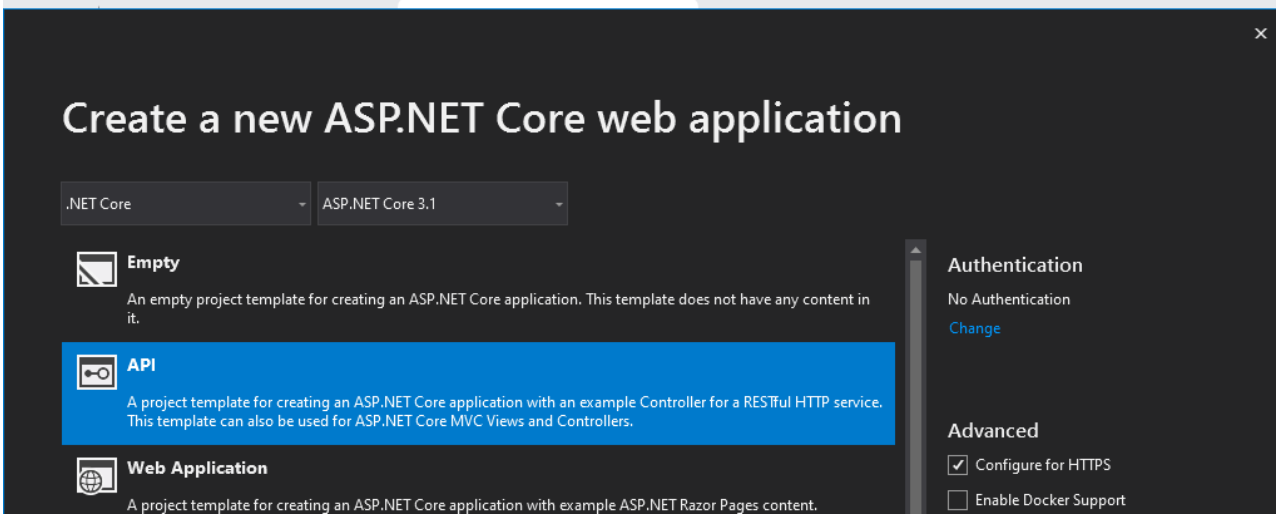
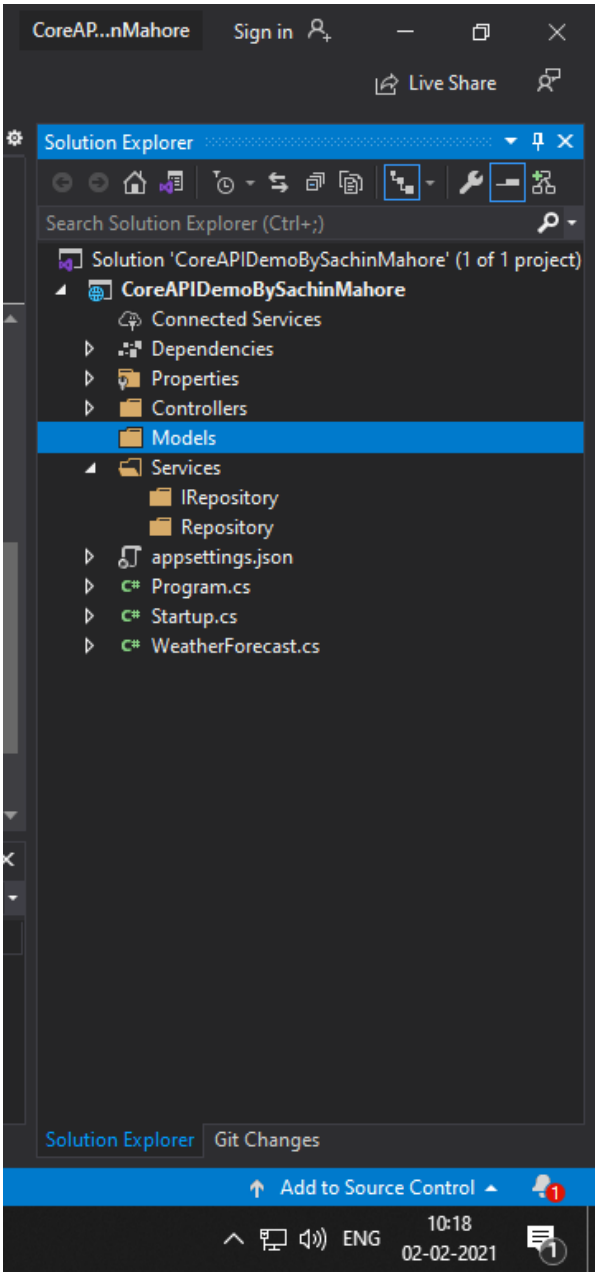

Create a new project
Recent project templates
A list of your recently accessed templates will be displayed here.
ASP.NET CORE 3.1 WEB API
Creation using EF , Swagger
By Sachin Mahore
Mob : +91 8600873002
Search for templates (Alt+S)
All languages
All platforms
All project types
Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Console
Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
Visual Basic Windows Linux macOS Console
ASP.NET Core Web Application
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.
C# Linux macOS Windows Cloud Service Web
Blazor App
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web
ASP.NET Web Application (.NET Framework)
Back
Next


Configure your new project
ASP.NET Core Web Application
C# Linux macOS Windows Cloud Service Web
Project name
CoreAPIDemoBySachinMahore
Location
C:\SachinAscent\
...
Solution name
CoreAPIDemoBySachinMahore
Place solution and project in the same directory


Create a new ASP.NET Core web application
.NET Core
ASP.NET Core 3.1
Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
Web Application
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
Authentication
No Authentication
Change
Advanced
Configure for HTTPS
Enable Docker Support



Create Folders

Model

Services

-IRepository

-Repository

Go to Tools menu => NuGet Package Manager =>Package Manager Console

Run the below commands:

PM> Install-Package Microsoft.EntityFrameworkCore.SqlServer

PM> Install-Package Microsoft.EntityFrameworkCore.Tools

OR

Add Nuget Package

Microsoft.EntityFrameworkCore.SqlServer

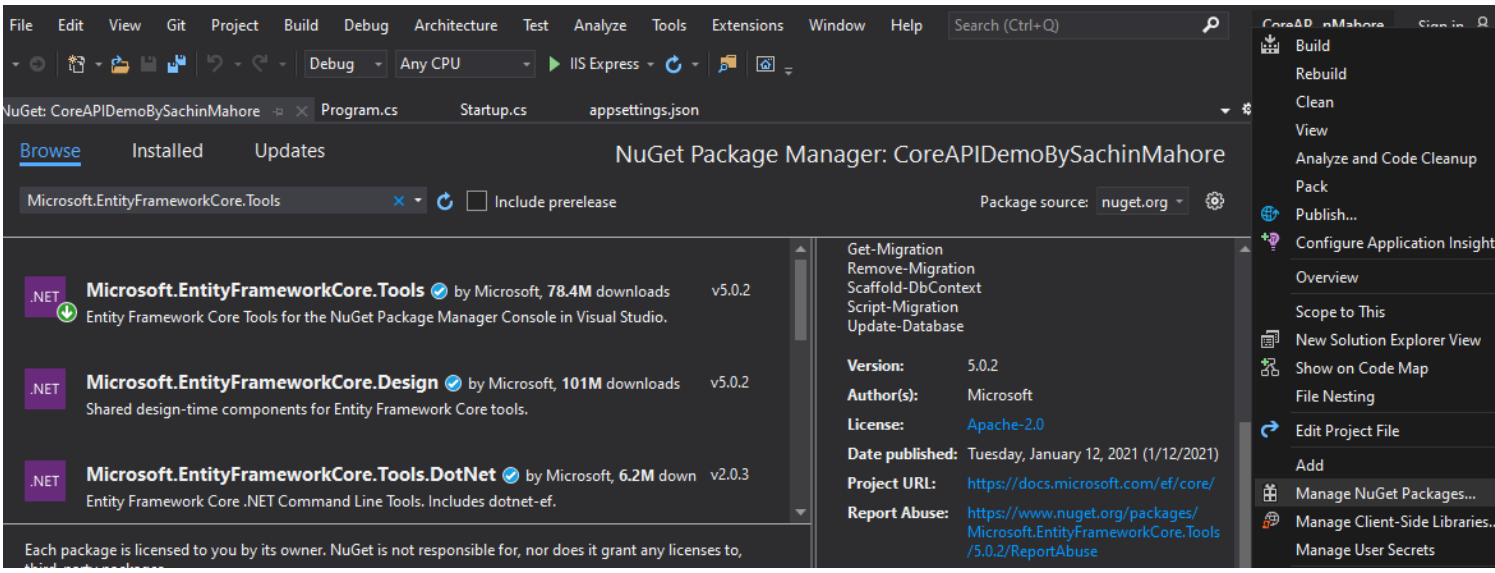
Microsoft.EntityFrameworkCore.Tools

Microsoft.EntityFrameworkCore.Design

Automapper

FluentValidation

Swashbuckle.AspNetCore



Add connection string in the appsettings.json file.

"DbConnection": " Server=eSankalp\SQLEXPRESS; Database=CoreAPIDemoBySachinMahore; User ID=sa;Password=123456;"

Now configure the database and add model classes. To do this, follow these steps:

DB First Approach:

Run the below commands:

PM> Scaffold-DbContext "Server=eSankalp\SQLEXPRESS; Database=CoreAPIDemoBySachinMahore; User ID=sa;Password=123456;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

OR

Code First Approach:

Add New Folder – Models- AppDbContext – AppDbContext.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using CoreAPIDemoBySachinMahore.Models.Tables;

namespace CoreAPIDemoBySachinMahore.Models.AppDbContext
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions dbContextOptions): base(dbContextOptions)
        {

        }
        public DbSet<Category> Category { get; set; }
    }
}
```

Add New Folder – Models- Tables – Category.cs

```
    [Table("tbl_Category")]
    public class Category
    {
        [Key]
        public int Id { get; set; }
        [Required]
        [StringLength(100)]
        public string CategoryName { get; set; }
    }
```

Now it's time to manage dependency injection.

To do this open the **Startup.cs** file.

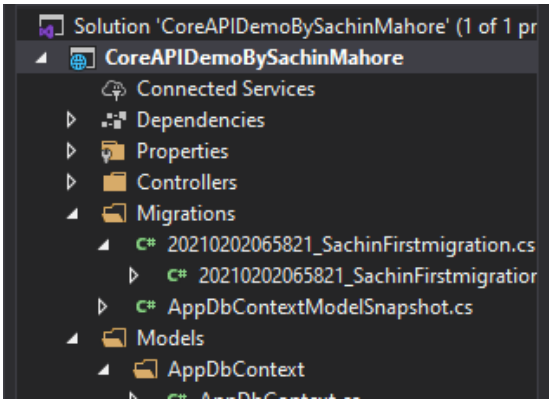
Add the below lines inside the function ConfigureServices.

```
            services.AddHttpClient();
            services.AddDbContext<sa>(options =>
            options.UseSqlServer(Configuration["DefaultConnection"]));
```

Go to Tools menu => NuGet Package Manager =>Package Manager Console

```
PM> add-migration SachinFirstMigration
PM> update-database
```

After this Migration folder created and Table created in Database



Service creation

Add IRepository – Services - IRepository – Add Class - Interface – ICategoryRepository.cs

Now add the 5 interface methods

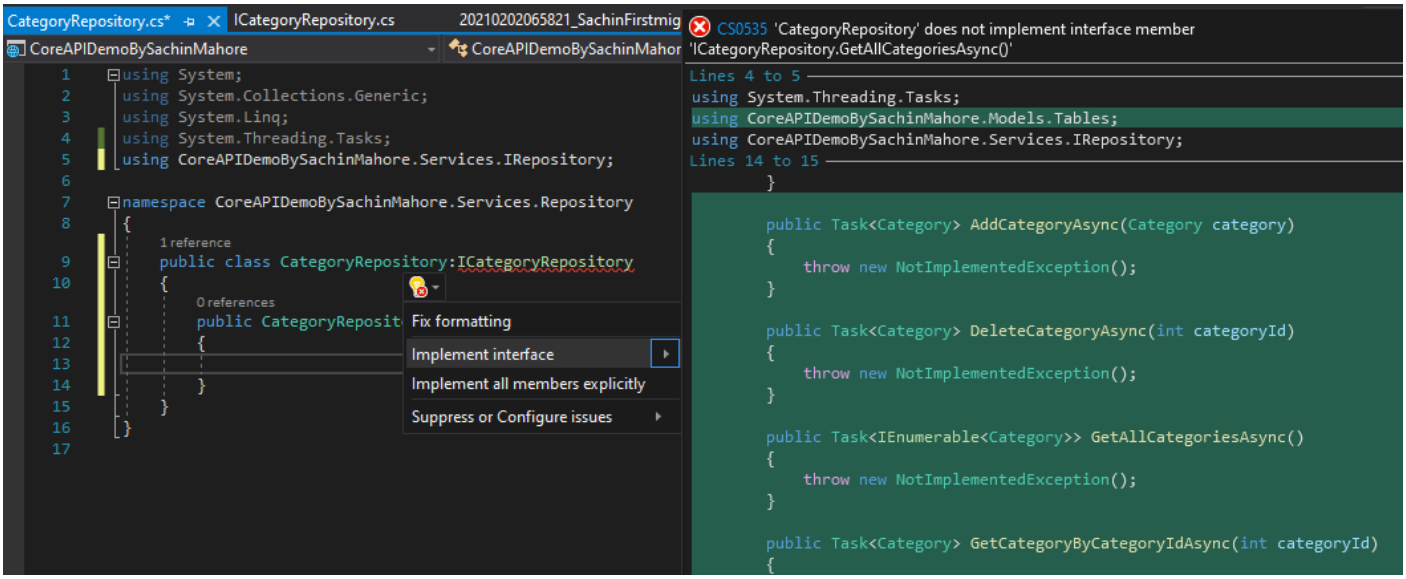
```
public interface ICategoryRepository
    {
        Task<IEnumerable<Category>> GetAllCategoriesAsync();
        Task<Category> GetCategoryByCategoryIdAsync(int categoryId);
        Task<Category> AddCategoryAsync(Category category);
        Task<Category> UpdateCategoryAsync(Category category);
        Task<Category> DeleteCategoryAsync(int categoryId);
    }
```

Add IRepository – Services - Repository – Add Class - Class – CategoryRepository.cs

Now implement the interface method

```
public class CategoryRepository:ICategoryRepository
    {
        Ctor ..   Double press TAB button to generate Constructor
    }
```

Click on Yellow Light – Implement Interface All 5 Methods Generated




```
public class CategoryRepository:ICategoryRepository
    {
        public CategoryRepository()
        {
        }
        public Task<Category> AddCategoryAsync(Category category)
        {
            throw new NotImplementedException();
        }
        public Task<Category> DeleteCategoryAsync(int categoryId)
        {
            throw new NotImplementedException();
        }
        public Task<IEnumerable<Category>> GetAllCategoriesAsync()
        {
            throw new NotImplementedException();
        }
        public Task<Category> GetCategoryByCategoryIdAsync(int categoryId)
        {
            throw new NotImplementedException();
        }
        public Task<Category> UpdateCategoryAsync(Category category)
```

```
        {
            throw new NotImplementedException();
        }
    }
```

Create an instance of the DBContext class using the constructor

```
        private readonly AppDbContext _appDbContext;
        public CategoryRepository(AppDbContext appDbContext)
        {
            _appDbContext = appDbContext;
        }
```

Define GetAllCategoriesAsync, AddCategoryAsync Methods

```
        public async Task<IEnumerable<Category>> GetAllCategoriesAsync()
        {
            var categoryList =await _appDbContext.Category.ToListAsync();
            return categoryList;
        }

        public async Task<Category> AddCategoryAsync(Category category)
        {
            _appDbContext.Category.Add(category);
            await  _appDbContext.SaveChangesAsync();
            return category;
        }
```

Add Controller – API Controller - CategoryAPIController.cs

```
        private readonly ICategoryRepository _categoryRepository;
        public CategoryAPIController(ICategoryRepository categoryRepository)
        {
            _categoryRepository=categoryRepository;
        }
        // GET: api/<CategoryAPIController>
        [HttpGet]
        public async Task<IActionResult> GetAllCategoriesAsync()
        {
            try
            {
                var result = await _categoryRepository.GetAllCategoriesAsync();
                return Ok(result);
            }
            catch (Exception ex)
            {
              return StatusCode(StatusCodes.Status500InternalServerError, "Error retriving data from
              the server");
            }
        }
        // POST api/<CategoryAPIController>
        [HttpPost]
        public async Task<IActionResult> AddCategoryAsync(Category category)
        {
            try
            {
                var result = await _categoryRepository.AddCategoryAsync(category);
                return Ok(result);
            }
            catch (Exception ex)
            {
              return StatusCode(StatusCodes.Status500InternalServerError, "Error retriving data from
              the server");
            }
        }
```

StartUp.cs Setting with Swagger

```
public void ConfigureServices(IServiceCollection services)
 {
     services.AddControllers();

     services.AddHttpClient();
     services.AddDbContext<AppDbContext>(options =>
     options.UseSqlServer(Configuration["DefaultConnection"]));

     services.AddTransient<ICategoryRepository, CategoryRepository>();

     services.AddSwaggerGen(options =>
     {
         options.SwaggerDoc("v1.0", new Microsoft.OpenApi.Models.OpenApiInfo
         {
             Title = "CoreAPI Demo By SachinMahore",
             Version = "v1.0"
         });
     });
 }

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
 {
     if (env.IsDevelopment())
     {
         app.UseDeveloperExceptionPage();
     }

     app.UseHttpsRedirection();
     app.UseRouting();
     app.UseAuthorization();
     app.UseEndpoints(endpoints =>
     {
         endpoints.MapControllers();
     });

     app.UseSwagger();
     app.UseSwaggerUI(options =>
     {
         options.SwaggerEndpoint("/swagger/v1.0/swagger.json", "CoreAPI Demo By
SachinMahore");
         options.RoutePrefix = "swagger";
     });
 }
```

For Getting Data, Update, Delete

CategoryRepository.cs

```
public async Task<Category> GetCategoryByCategoryIdAsync(int categoryId)
{
    var result = await _appDbContext.Category.FirstOrDefaultAsync(x => x.Id == categoryId);
    if (result == null)
    {
        return null;
    }

    return result;
}
public async Task<Category> UpdateCategoryAsync(Category category)
{

    _appDbContext.Category.Update(category);
    await _appDbContext.SaveChangesAsync();
    return category;
}
public async Task<Category> DeleteCategoryAsync(int categoryId)
{
    var result = await _appDbContext.Category.FirstOrDefaultAsync(x => x.Id == categoryId);
    if (result == null)
    {
        return null;
    }
    _appDbContext.Category.Remove(result);
    _appDbContext.SaveChangesAsync();
    return result;
}
```

CategoryController

```
// GET api/<CategoryAPIController>/5
[HttpGet("{categoryId:int}")]
public async Task<IActionResult> GetCategoryByCategoryIdAsync(int categoryId)
{
    try
    {
        var result = await _categoryRepository.GetCategoryByCategoryIdAsync(categoryId);
        return Ok(result);
    }
    catch (Exception)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error retriving data
from the server");
    }
}
// PUT api/<CategoryAPIController>/5
[HttpPut]
public async Task<IActionResult> UpdateCategoryAsync(Category category)
{
    try
    {
        var result = await _categoryRepository.UpdateCategoryAsync(category);
        return Ok(result);
    }
    catch (Exception ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, "Error retriving data
from the server");
    }
}
// DELETE api/<CategoryAPIController>/5
[HttpDelete("{categoryId:int}")]
public async Task<IActionResult> DeleteCategoryAsync(int categoryId)
{
    try
    {
```

```
                var result = await _categoryRepository.DeleteCategoryAsync(categoryId);
                return Ok(result);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error retriving data
from the server");
            }
        }
```

Download code : https://github.com/SachinMahore/CoreAPIDemoBySachinMahore