📂 Project Structure

```
/YourProject
 ├── Program.cs
 ├── TemplateFileGenerator.cs
 ├── /Templates
 │    ├── ModelTemplate.txt
 │    ├── RepositoryInterfaceTemplate.txt
 │    ├── RepositoryClassTemplate.txt
 │    └── ControllerTemplate.txt
 ├── /GeneratedFiles
 │    ├── /Models
 │    ├── /Interfaces
 │    ├── /Repositories
 │    └── /Controllers
```

📄 ModelTemplate.txt

```
namespace API1.Models
{
   public class {{Name}}
   {
      public int Id { get; set; }
      // Add more properties here
   }
}
```

📄 RepositoryInterfaceTemplate.txt

```
using System.Collections.Generic;
using System.Threading.Tasks;
using API1.Models;

namespace API1.Interfaces
```

```
{
    public interface I{{Name}}Repository
    {
        Task<{{Name}}> GetByIdAsync(int id);
        Task<IEnumerable<{{Name}}>> GetAllAsync();
        Task AddAsync({{Name}} entity);
        void Update({{Name}} entity);
        void Delete({{Name}} entity);
        Task<int> SaveChangesAsync();
    }
}
```

📄 RepositoryClassTemplate.txt

```
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
using System.Threading.Tasks;
using API1.Models;
using API1.Interfaces;

namespace API1.Repositories
{
    public class {{Name}}Repository : I{{Name}}Repository
    {
        private readonly YourDbContext _context;

        public {{Name}}Repository(YourDbContext context)
        {
            _context = context;
        }

        public async Task AddAsync({{Name}} entity) => await
_context.Set<{{Name}}>().AddAsync(entity);
```

```
        public void Delete({{Name}} entity) => _context.Set<{{Name}}>().Remove(entity);

        public async Task<IEnumerable<{{Name}}>> GetAllAsync() => await
_context.Set<{{Name}}>().ToListAsync();

        public async Task<{{Name}}> GetByIdAsync(int id) => await
_context.Set<{{Name}}>().FindAsync(id);

        public void Update({{Name}} entity) => _context.Set<{{Name}}>().Update(entity);

        public async Task<int> SaveChangesAsync() => await _context.SaveChangesAsync();
    }
}
```

📄 ControllerTemplate.txt

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using API1.Models;
using API1.Interfaces;

namespace API1.Controllers
{
    public class {{Name}}Controller : Controller
    {
        private readonly I{{Name}}Repository _repository;

        public {{Name}}Controller(I{{Name}}Repository repository)
        {
            _repository = repository;
        }

        // GET: {{Name}}Controller
        public async Task<IActionResult> Index()
        {
            var items = await _repository.GetAllAsync();
            return View(items);
        }

        // GET: {{Name}}Controller/Details/5
        public async Task<IActionResult> Details(int id)
        {
            var item = await _repository.GetByIdAsync(id);
            if (item == null) return NotFound();
            return View(item);
        }

        // GET: {{Name}}Controller/Create
        public IActionResult Create()
        {
            return View();
```

```
}

// POST: {{Name}}Controller/Create
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create({{Name}} entity)
{
    if (ModelState.IsValid)
    {
        await _repository.AddAsync(entity);
        await _repository.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(entity);
}

// GET: {{Name}}Controller/Edit/5
public async Task<IActionResult> Edit(int id)
{
    var item = await _repository.GetByIdAsync(id);
    if (item == null) return NotFound();
    return View(item);
}

// POST: {{Name}}Controller/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, {{Name}} entity)
{
    if (id != entity.Id) return BadRequest();

    if (ModelState.IsValid)
    {
        _repository.Update(entity);
        await _repository.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
```

```
            return View(entity);
        }

        // GET: {{Name}}Controller/Delete/5
        public async Task<IActionResult> Delete(int id)
        {
            var item = await _repository.GetByIdAsync(id);
            if (item == null) return NotFound();
            return View(item);
        }

        // POST: {{Name}}Controller/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var item = await _repository.GetByIdAsync(id);
            if (item != null)
            {
                _repository.Delete(item);
                await _repository.SaveChangesAsync();
            }
            return RedirectToAction(nameof(Index));
        }
    }
}
```

📄 TemplateFileGenerator.cs

```
using System;
using System.IO;

public static class TemplateFileGenerator
{
```

```
    private static readonly string TemplateFolder = "Templates";
    private static readonly string OutputFolder = "GeneratedFiles";

    private static string LoadTemplate(string templateFile)
    {
        return File.ReadAllText(Path.Combine(TemplateFolder, templateFile));
    }

    private static void WriteFile(string subFolder, string outputFile, string content)
    {
        string dir = Path.Combine(OutputFolder, subFolder);
        Directory.CreateDirectory(dir);
        string filePath = Path.Combine(dir, outputFile);
        File.WriteAllText(filePath, content);
        Console.WriteLine($"✅ File created: {filePath}");
    }

    private static string ReplacePlaceholders(string template, string name)
    {
        return template.Replace("{{Name}}", name);
    }

    public static void GenerateAll(string name)
    {
        // Model
        var modelTemplate = ReplacePlaceholders(LoadTemplate("ModelTemplate.txt"), name);
        WriteFile("Models", $"{name}.cs", modelTemplate);

        // Repository Interface
        var repoInterface =
ReplacePlaceholders(LoadTemplate("RepositoryInterfaceTemplate.txt"), name);
        WriteFile("Interfaces", $"I{name}Repository.cs", repoInterface);

        // Repository Class
        var repoClass = ReplacePlaceholders(LoadTemplate("RepositoryClassTemplate.txt"),
name);
        WriteFile("Repositories", $"{name}Repository.cs", repoClass);
```

```
        // Controller
        var controller = ReplacePlaceholders(LoadTemplate("ControllerTemplate.txt"), name);
        WriteFile("Controllers", $"{name}Controller.cs", controller);
    }
}
```

📄 Program.cs

```
public class Program
{
    public static void Main(string[] args)
    {
        // Example: Generate Invoice files
        TemplateFileGenerator.GenerateAll("Invoice");

        // Example: Generate Customer files
        TemplateFileGenerator.GenerateAll("Customer");
    }
}
```