**1. Folder Structure**

/workspace

├── dotnetapp/

│ ├── Controllers/

│ │ └── PostController.cs

│ ├── Models/

│ │ └── Post.cs

│ ├── Repository/

│ │ ├── IPostRepository.cs

│ │ └── PostRepository.cs

│ ├── Program.cs

│ └── dotnetapp.csproj

└── MyTestNUnitProject/

├── PostControllerTests.cs

└── MyTestNUnitProject.csproj

**2. Post Model**

Models/Post.cs

using System;

namespace dotnetapp.Models

{

public class Post

{

public int PostId { get; set; }

public string Title { get; set; }

public string Content { get; set; }

public DateTime DateCreated { get; set; }

}

}

**3. Repository Interface**

Repository/IPostRepository.cs

using System.Collections.Generic;

using dotnetapp.Models;

namespace dotnetapp.Repository

{

public interface IPostRepository

{

IEnumerable<Post> GetPosts();

Post GetPost(int postId);

Post CreatePost(Post post);

Post UpdatePost(Post post);

bool DeletePost(int postId);

}

}

**4. Repository Implementation (In-memory example)**

Repository/PostRepository.cs

using System;

using System.Collections.Generic;

using System.Linq;

using dotnetapp.Models;

namespace dotnetapp.Repository

{

public class PostRepository : IPostRepository

{

private readonly List<Post> \_posts = new List<Post>();

public IEnumerable<Post> GetPosts() => \_posts;

public Post GetPost(int postId)

{

return \_posts.FirstOrDefault(p => p.PostId == postId);

}

public Post CreatePost(Post post)

{

post.PostId = \_posts.Count + 1;

post.DateCreated = DateTime.Now;

\_posts.Add(post);

return post;

}

public Post UpdatePost(Post post)

{

var existing = \_posts.FirstOrDefault(p => p.PostId == post.PostId);

if (existing == null)

return null;

existing.Title = post.Title;

existing.Content = post.Content;

return existing;

}

public bool DeletePost(int postId)

{

var post = \_posts.FirstOrDefault(p => p.PostId == postId);

if (post == null)

return false;

\_posts.Remove(post);

return true;

}

}

}

**5. PostController**

Controllers/PostController.cs

using Microsoft.AspNetCore.Mvc;

using dotnetapp.Models;

using dotnetapp.Repository;

namespace dotnetapp.Controllers

{

[ApiController]

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

public class PostController : ControllerBase

{

private readonly IPostRepository \_repository;

public PostController(IPostRepository repository)

{

\_repository = repository;

}

[HttpGet]

public IActionResult GetAllPosts()

{

return Ok(\_repository.GetPosts());

}

[HttpGet("{id}")]

public IActionResult GetPost(int id)

{

var post = \_repository.GetPost(id);

if (post == null)

return NotFound();

return Ok(post);

}

[HttpPost]

public IActionResult CreatePost([FromBody] Post post)

{

if (post == null || string.IsNullOrEmpty(post.Title))

return BadRequest();

var created = \_repository.CreatePost(post);

return CreatedAtAction(nameof(GetPost), new { id = created.PostId }, created);

}

[HttpPut("{id}")]

public IActionResult UpdatePost(int id, [FromBody] Post post)

{

if (id != post.PostId)

return BadRequest();

var updated = \_repository.UpdatePost(post);

if (updated == null)

return NotFound();

return Ok(updated);

}

[HttpDelete("{id}")]

public IActionResult DeletePost(int id)

{

var result = \_repository.DeletePost(id);

if (!result)

return NotFound();

return NoContent();

}

}

}

**6. Program.cs**

Use dependency injection to register repository:

using dotnetapp.Repository;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddSingleton<IPostRepository, PostRepository>();

var app = builder.Build();

app.MapControllers();

app.Run();

**7. NUnit Test Project**

**Create the Test Project**

dotnet new nunit -n MyTestNUnitProject

dotnet sln add dotnetapp/dotnetapp.csproj MyTestNUnitProject/MyTestNUnitProject.csproj

dotnet add MyTestNUnitProject/MyTestNUnitProject.csproj reference dotnetapp/dotnetapp.csproj

**8. NUnit Test Class**

MyTestNUnitProject/PostControllerTests.cs

using NUnit.Framework;

using dotnetapp.Controllers;

using dotnetapp.Models;

using dotnetapp.Repository;

using Microsoft.AspNetCore.Mvc;

using System.Linq;

namespace MyTestNUnitProject

{

public class PostControllerTests

{

private PostController \_controller;

private IPostRepository \_repository;

[SetUp]

public void Setup()

{

\_repository = new PostRepository();

\_controller = new PostController(\_repository);

}

[Test]

public void CreatePost\_ShouldReturnCreatedAtAction()

{

var post = new Post { Title = "Test Post", Content = "This is test content." };

var result = \_controller.CreatePost(post) as CreatedAtActionResult;

Assert.IsNotNull(result);

Assert.AreEqual(201, result.StatusCode);

Assert.AreEqual("Test Post", ((Post)result.Value).Title);

}

[Test]

public void GetAllPosts\_ShouldReturnOkResult()

{

\_repository.CreatePost(new Post { Title = "P1", Content = "C1" });

var result = \_controller.GetAllPosts() as OkObjectResult;

Assert.IsNotNull(result);

var posts = result.Value as System.Collections.Generic.IEnumerable<Post>;

Assert.IsTrue(posts.Any());

}

[Test]

public void UpdatePost\_ShouldReturnOk()

{

var post = \_repository.CreatePost(new Post { Title = "Old", Content = "Old content" });

post.Title = "Updated";

var result = \_controller.UpdatePost(post.PostId, post) as OkObjectResult;

Assert.IsNotNull(result);

Assert.AreEqual("Updated", ((Post)result.Value).Title);

}

[Test]

public void DeletePost\_ShouldReturnNoContent()

{

var post = \_repository.CreatePost(new Post { Title = "Del", Content = "Del content" });

var result = \_controller.DeletePost(post.PostId) as NoContentResult;

Assert.IsNotNull(result);

Assert.AreEqual(204, result.StatusCode);

}

[Test]

public void GetPost\_InvalidId\_ShouldReturnNotFound()

{

var result = \_controller.GetPost(999);

Assert.IsInstanceOf<NotFoundResult>(result);

}

}

}

**9. Run Tests**

cd MyTestNUnitProject

dotnet test

You should see output like:

Passed! - Failed: 0, Passed: 5, Skipped: 0

**10. Run API**

cd dotnetapp

dotnet restore

dotnet build

dotnet run

Then open:  
[**http://localhost:8080/api/posts**](http://localhost:8080/api/posts)

**11. Key Learning Outcomes**

* Implement CRUD APIs using ASP.NET Core.
* Follow Repository pattern.
* Unit testing with NUnit (Assert.AreEqual, Assert.IsTrue, etc.).
* Mock repository using in-memory data.
* Ensure each controller action has positive & negative test cases.