**Day3\_Lab: Mocking Repository Layer & Testing Service Layer in xUnit with Moq**

**Objective**

* Learn how to mock a repository layer using **Moq**.
* Write unit tests for a **Service layer** that depends on the repository.
* Verify method calls, return values, and exception handling.

**Project Setup**

1. Create a new **.NET Core Console or Class Library project** named StudentManagement.
2. Add a **Test Project** named StudentManagement.Tests (xUnit).
3. Install dependencies in the test project:
4. dotnet add package xunit
5. dotnet add package Moq
6. dotnet add package Microsoft.NET.Test.Sdk
7. dotnet add package xunit.runner.visualstudio

**Step 1 – Define the Domain Model**

In StudentManagement project:

namespace StudentManagement.Models

{

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

public double Grade { get; set; }

}

}

**Step 2 – Define the Repository Contract**

using System.Collections.Generic;

using StudentManagement.Models;

namespace StudentManagement.Repositories

{

public interface IStudentRepository

{

Student GetStudentById(int id);

IEnumerable<Student> GetAllStudents();

void AddStudent(Student student);

}

}

**Step 3 – Create the Service Layer**

using System;

using System.Collections.Generic;

using System.Linq;

using StudentManagement.Models;

using StudentManagement.Repositories;

namespace StudentManagement.Services

{

public class StudentService

{

private readonly IStudentRepository \_repository;

public StudentService(IStudentRepository repository)

{

\_repository = repository;

}

public Student GetStudent(int id)

{

var student = \_repository.GetStudentById(id);

if (student == null)

throw new Exception("Student not found");

return student;

}

public double GetAverageGrade()

{

var students = \_repository.GetAllStudents();

if (!students.Any())

return 0;

return students.Average(s => s.Grade);

}

public void RegisterStudent(Student student)

{

if (string.IsNullOrWhiteSpace(student.Name))

throw new ArgumentException("Student name is required");

\_repository.AddStudent(student);

}

}

}

**Step 4 – Write Unit Tests with xUnit + Moq**

In StudentManagement.Tests:

using Moq;

using System;

using System.Collections.Generic;

using StudentManagement.Models;

using StudentManagement.Repositories;

using StudentManagement.Services;

using Xunit;

namespace StudentManagement.Tests

{

public class StudentServiceTests

{

public StudentServiceTests()

{

}

[Fact]

public void GetStudent\_ExistingId\_ReturnsStudent()

{

// Arrange

// Act

// Assert

}

[Fact]

public void GetStudent\_NonExistingId\_ThrowsException()

{

// Arrange

// Act & Assert

}

[Fact]

public void GetAverageGrade\_WithStudents\_ReturnsCorrectAverage()

{

// Arrange

// Act

// Assert

}

[Fact]

public void RegisterStudent\_EmptyName\_ThrowsException()

{

// Arrange

// Act & Assert

}

}

}

**Lab Tasks for additional practice**

1. **Run the tests** and ensure all pass
2. **Modify the service** to add a new method GetTopStudent() that returns the highest grade student.
   * Write **unit tests** for this method.
3. Add negative test cases (e.g., when repository returns null or empty list).

At the end of this lab, students will have hands-on practice mocking a repository and testing the service layer with **xUnit + Moq**.