**WEEK-2**

Task-5

* Understand how Mocking can enhance Test-Driven Development (TDD)
  + Mocking, Isolation, Test doubles, Mock Vs Fake Vs Stub, Key advantages of TDD
* Explain the meaning of Mocking in Unit Testing and why use mocks in Unit Testing
  + Mocking and Isolation in Unit Testing, Isolating dependencies in Tests using Mocks and Stubs
* Understand the basics of DI (Dependency Injection) and how dependency injection helps unit testing in applications
  + Dependency Injection, Constructor Injection, Method Injection
* Demonstrate on how to create a testable code with Moq.
  + Testable code
* Demonstrate on how to create a mock object that access database for unit tests
  + Mock database for Unit Tests
* Demonstrate on mock object that access the file system for unit tests
  + Mock files for Unit Tests

Code:

**Mocking** enhances **Test-Driven Development (TDD)** by isolating components using **test doubles** like **mocks, fakes**, and **stubs**. **Mocks** simulate behavior to test code in isolation, while **stubs** provide fixed data, and **fakes** are simple implementations. **Dependency Injection (DI)**, including **constructor** and **method injection**, enables swapping real dependencies with mocks. With **Moq**, create testable code like:

namespace CustomerCommLib

{

public class MailSender:IMailSender

{

public bool SendMail(string toAddress, string message)

{

MailMessage mail = new MailMessage();

SmtpClient SmtpServer = new SmtpClient("smtp.gmail.com");

mail.From = new MailAddress("your\_email\_address@gmail.com");

mail.To.Add(toAddress);

mail.Subject = "Test Mail";

mail.Body = message;

SmtpServer.Port = 587;

SmtpServer.Credentials = new NetworkCredential("username", "password");

SmtpServer.EnableSsl = true;

SmtpServer.Send(mail);

}

}

}

Name:Ayushman Upadhyay

Email: [2230164@kiit.ac.in](mailto:2230164@kiit.ac.in)

Superset: 6358125