Moq Handson Question and Answers

1. How Mocking Enhances TDD

Mocking allows you to isolate the code under test, replacing real dependencies with test doubles (mocks, fakes, stubs). This makes tests faster, repeatable, and focused. TDD benefits from mocking because it helps write tests before real implementations.

2. Meaning of Mocking in Unit Testing

Mocking replaces real dependencies with lightweight stand-ins that simulate behavior. This isolates the unit being tested from external systems like databases or mail servers.

3. Mock vs Fake vs Stub

Mock: verifies interaction; Stub: returns fixed data; Fake: working but simple implementation (e.g., in-memory DB).

4. Key Advantages of TDD

Improved design, early bug detection, better code coverage, and safer refactoring**.**

5. Basics of Dependency Injection

DI means passing dependencies (like services or repositories) into a class instead of hardcoding them. Helps in isolating dependencies for testing.

6. Types of Dependency Injection

Constructor Injection (preferred), Method Injection, Property Injection.

7. Testable Code with Moq

```
public interface IEmailService {
   bool Send(string to, string msg);
}
public class Notifier {
   IEmailService _email;
   public Notifier(IEmailService email) { _email = email; }
   public bool Notify(string msg) => _email.Send("x@x.com", msg);
}
```

8. Mocking a Database Access

```
var mockDb = new Mock<IUserRepository>();
mockDb.Setup(db => db.GetUserById(1)).Returns(new User { Id = 1, Name = "Test" });
```

9. Mocking File System Access

```
var mockFile = new Mock<IFileSystem>();
mockFile.Setup(f => f.ReadFile("data.txt")).Returns("mock content");
```