How to run unit test cases
Option1 From Command Line **dotnet test**
Test menu has option Run All Tests

For develop APi Clearn arcitecure used

Clean Architecture is a software architectural pattern that promotes the separation of concerns and maintains a high level of modularity and testability in your code. It is not explicitly tied to the SOLID principles, but it naturally aligns with these principles. Let's go through how Clean Architecture follows the SOLID principles:

1. **Single Responsibility Principle (SRP):** Clean Architecture adheres to SRP by organizing the code into distinct layers, each with a single responsibility. These layers include the innermost core, which contains the application's business logic, and outer layers responsible for the interface, data access, and frameworks. Each layer has a specific role, ensuring that each class and component has a single reason to change.

2. **Open/Closed Principle (OCP):** Clean Architecture encourages extension over modification. When you want to introduce new features or make changes, you should be able to do so without altering the existing codebase. The core of the application remains stable, and new functionality can be added in outer layers without affecting the inner layers.

3. **Liskov Substitution Principle (LSP):** Clean Architecture supports LSP by emphasizing the use of interfaces and abstractions. Dependencies are inverted, and high-level modules depend on abstractions, not concrete implementations. This allows you to substitute different implementations without affecting the behavior of the higher-level components.

4. **Interface Segregation Principle (ISP):** Clean Architecture often employs the use of small, focused interfaces that cater to the needs of specific clients. This avoids the problem of clients depending on large, monolithic interfaces that contain methods they don't need. This promotes a more modular and maintainable design.

5. **Dependency Inversion Principle (DIP):** Clean Architecture explicitly enforces DIP by inverting the traditional dependency direction. High-level policy modules depend on abstractions (interfaces), while low-level detail modules implement those abstractions. This allows for the easy substitution of implementations and reduces the coupling between different parts of the system.

In summary, Clean Architecture provides a structure that aligns with SOLID principles by separating concerns, promoting modularity, and reducing code dependencies. It encourages a flexible and maintainable design where changes can be made with minimal impact on existing code, making your software more robust and adaptable.

Note:
For develop this api for Generate connection string Factory pattern used
And for APi CQRS and for db call repository pattern used.