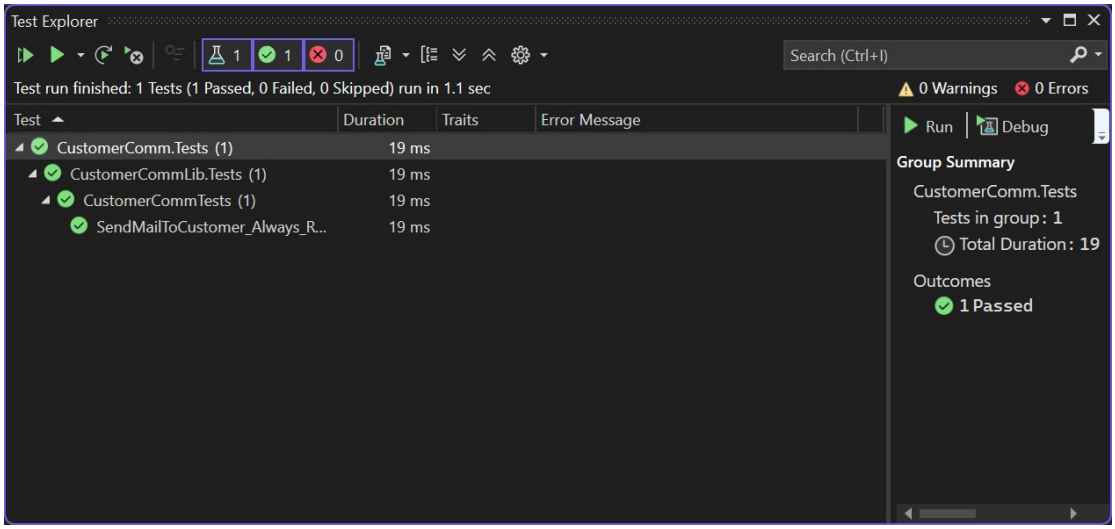
MOQ

1. Write Testable Code with Moq

Scenario

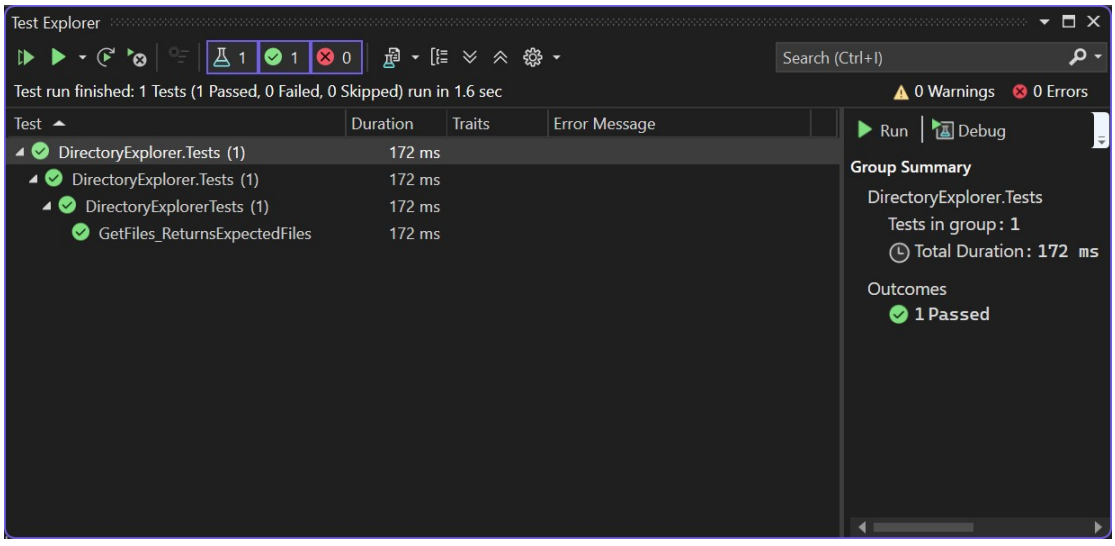
You are tasked to write a unit test code for the below scenario.
The application in which you are teamed up with, deals with a mail server communication in which your application tries to send mail to its users upon every transaction. Your role is to write unit testing the module that contains send mail functionality. You wanted to perform testing the module without sending any email.
After investigating the problem scenario, you found a solution and that is creating **mock** objects of these external dependencies in the unit testing project so that you can achieve speedier test execution and loose coupling of code.



2. Mock file object for Unit Tests

Scenario

You are tasked to write a unit test code for the below scenario.
The application in which you are teamed up with, deals with the file system and it searches for files and retrieves files under the specified path. In the existing system, **Directory.GetFiles()** method has been used. You found that it's not good idea to use Directory.GetFiles from the System.IO being its **static** and **unable to unit tes**t such methods.
After investigating the problem scenario, you found a solution and that is refactoring the code. Instead of using directly the static method Directory.GetFiles, you decided to create your own implementation to the method so that be able to **mock** files in the Unit Tests.



3. Mock database for Unit Tests

Scenario

You are tasked to write a unit test code for the below scenario.
The application in which you are teamed up with, deals with a network database in which your application stores the record or certain players. It involves storing and retrieval of player details. Your role is to write unit testing the player module which involves an external dependency. You can't proceed with unit testing.
After investigating the problem scenario, you found a solution and that is creating **mock** objects of these external dependencies in the unit testing project so that you can achieve speedier test execution and loose coupling of code.