# Lab 26

Instructor: Sidra Khatoon

E-mail: skhatoon[@uit.edu](mailto:ad@uit.edu)

# Objective

# The objective of lab is to testing in flutter

**Student Information**

|  |  |
| --- | --- |
| **Student Name** |  |
| **Student ID** |  |
| **Date** |  |

**Assessment**

|  |  |
| --- | --- |
| **Marks Obtained** |  |
| **Remarks** |  |
| **Signature** |  |

# Objective

# The objective of lab is to testing in flutter

# Instructions

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

# How to Submit?

Submit lab work using Teams.

**Unit Testing**

# Unit tests are handy for verifying the behavior of a single function, method, or class. The test package provides the core framework for writing unit tests, and the flutter\_test package provides additional utilities for testing widgets.

# This recipe demonstrates the core features provided by the test package using the following steps:

# Add the test or flutter\_test dependency.

# Create a test file.

# Create a class to test.

# Write a test for our class.

# Combine multiple tests in a group.

# Run the tests.

**Add the test dependency**

# The test package provides the core functionality for writing tests in Dart. This is the best approach when writing packages consumed by web, server, and Flutter apps. To add the test package as a dev dependency

# flutter pub add dev:test

**Create a test file**

In this project, create two files: counter.dart and counter\_test.dart.

The counter.dart file contains a class that you want to test and resides in the lib folder. The counter\_test.dart file contains the tests themselves and lives inside the test folder.

In general, test files should reside inside a test folder located at the root of your Flutter application or package. Test files should always end with \_test.dart, this is the convention used by the test runner when searching for tests.

lib/

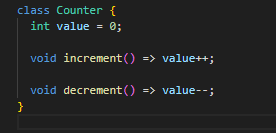
counter.dart

test/

counter\_test.dart

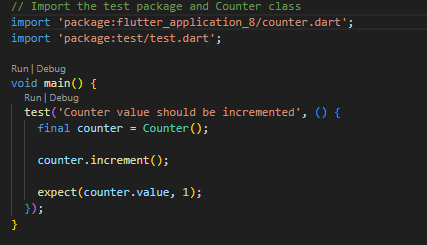
**Create a class to test**

Next, you need a "unit" to test. Remember: "unit" is another name for a function, method, or class. For this example, create a Counter class inside the lib/counter.dart file. It is responsible for incrementing and decrementing a value starting at 0.



**Write a test for our class**

Inside the counter\_test.dart file, write the first unit test. Tests are defined using the top-level test function, and you can check if the results are correct by using the top-level expect function. Both of these functions come from the test package.



**Combine multiple tests in a group**

If you want to run a series of related tests, use the flutter\_test package group function to categorize the tests. Once put into a group, you can call flutter test on all tests in that group with one command.

**Assessment:**

1: What is unit testing, and why is it important in Flutter development?

2: How would you write a unit test for a function that calculates the sum of two integers in Flutter?

(Provide code as an example.)

3: What is the difference between unit tests and widget tests in Flutter?