1. Unit-тест:

Определение: Unit-тест — это тест, который проверяет работу отдельных модулей или функций приложения в изоляции от других частей кода.

Назначение:

Проверка логики небольших, изолированных блоков кода, таких как функции, методы или классы.

Обеспечение быстрого обнаружения ошибок на ранних стадиях разработки.

Поддержание стабильности при внесении изменений в код.

Пример: Проверка функции, которая складывает два числа:

int add(int a, int b) => a + b;

void main() {

test('add function test', () {

expect(add(2, 3), 5);

});

}

2. Widget-тест:

Определение: Widget-тест проверяет отдельные виджеты (компоненты пользовательского интерфейса) в изоляции от остального приложения.

Назначение:

Проверка правильного рендеринга виджетов и их поведения при взаимодействии.

Эмуляция взаимодействий, таких как нажатия кнопок, ввод текста и прокрутка.

Обеспечение корректного отображения и функциональности интерфейса.

Пример: Тест нажатия кнопки:

testWidgets('Button click increments counter', (WidgetTester tester) async {

await tester.pumpWidget(MyApp());

// Ищем текст '0'

expect(find.text('0'), findsOneWidget);

// Нажимаем кнопку

await tester.tap(find.byIcon(Icons.add));

await tester.pump();

// Проверяем, что текст изменился на '1'

expect(find.text('1'), findsOneWidget);

});

3. Integration-тест:

Определение: Integration-тест проверяет, как различные части приложения работают вместе, включая взаимодействие с базами данных, API и пользовательским интерфейсом.

Назначение:

Проверка взаимодействия между разными модулями и компонентами.

Эмуляция поведения пользователя в реальных сценариях, включая навигацию и обработку данных.

Обеспечение работоспособности приложения как единого целого.

Пример интеграционного теста:

void main() {

IntegrationTestWidgetsFlutterBinding.ensureInitialized();

testWidgets('Full app test', (WidgetTester tester) async {

await tester.pumpWidget(MyApp());

// Проверка нажатия кнопки и перехода на новый экран

await tester.tap(find.byType(ElevatedButton));

await tester.pumpAndSettle();

expect(find.text('Details Screen'), findsOneWidget);

});

}

4. Библиотека Mockito:

Определение: Mockito — это библиотека для создания имитаций (mock-объектов), которая используется для тестирования взаимодействий между объектами.

Назначение:

Подмена зависимостей для тестирования отдельных модулей.

Проверка вызова методов, их аргументов и последовательности вызовов.

Уменьшение зависимости от реальных сервисов (например, баз данных или API).

Пример использования Mockito:

import 'package:mockito/mockito.dart';

import 'package:flutter\_test/flutter\_test.dart';

// Создание Mock-класса

class MockService extends Mock implements ApiService {}

void main() {

test('fetchData returns expected value', () async {

final mockService = MockService();

// Задание поведения mock-объекта

when(mockService.fetchData()).thenAnswer((\_) async => 'Mock Data');

// Вызов метода и проверка результата

expect(await mockService.fetchData(), 'Mock Data');

// Проверка, что метод был вызван один раз

verify(mockService.fetchData()).called(1);

});

}