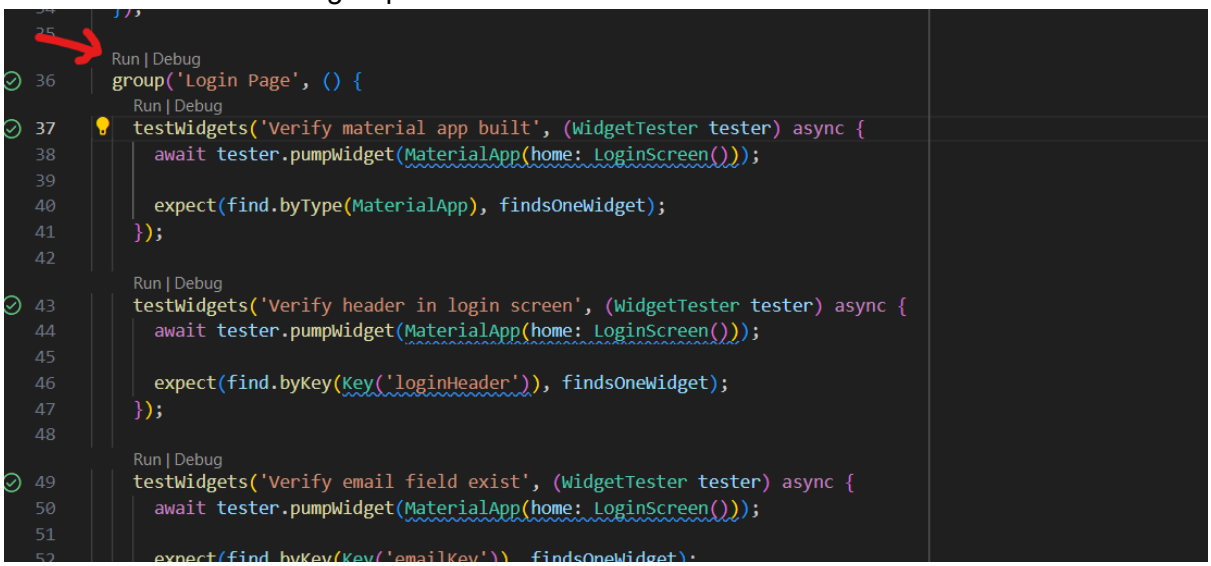
## Ventalis Unit & Integration Testing

- Unit & Integration Test Case for Login, Dashboard and Profile Screen.
- Run test case as group



```
Run | Debug
group('Login Page', () {
  Run | Debug
  testWidgets('Verify material app built', (WidgetTester tester) async {
    await tester.pumpWidget(MaterialApp(home: LoginScreen()));

    expect(find.byType(MaterialApp), findsOneWidget);
  });

  Run | Debug
  testWidgets('Verify header in login screen', (WidgetTester tester) async {
    await tester.pumpWidget(MaterialApp(home: LoginScreen()));

    expect(find.byKey(Key('loginHeader')), findsOneWidget);
  });

  Run | Debug
  testWidgets('Verify email field exist', (WidgetTester tester) async {
    await tester.pumpWidget(MaterialApp(home: LoginScreen()));
```

- To run a single test case simply click on Run above the testWidgets('*')
- To run whole file

```
flutter test test\login\login_test.dart
```

**Note: Change file path to run other files.**

### Unit Testing

Unit tests are used to verify the correctness of individual functions, methods, or classes. The goal is to test each piece of code in isolation from the rest of the codebase.

### Structure of a Unit Test

A unit test typically consists of three parts:

1. **Arrange**: Set up the conditions for the test.
2. **Act**: Perform the action that you want to test.
3. **Assert**: Check that the results are what you expect.

Example Unit Test

```dart
void main() {
```

```
  testWidgets('Email validation', (WidgetTester tester) async {
     await tester.pumpWidget(MaterialApp(home: LoginScreen()));

     final emailWidget = find.byKey(Key('emailKey'));

     await tester.enterText(emailWidget, 'test@gmail.com');

     expect(isEmailValid('luc@gmail.com'), true);

     expect(isEmailValid('lucgmail.com'), false);
 });
```

In this example, we're testing the `validate` method of the `Email`. We expect it to return `true` when given the correct email.

**Integration Testing**

Integration tests are used to test how multiple units work together. In the context of Flutter, this often means testing how different widgets interact with each other.

**Structure of an Integration Test**

An integration test in Flutter typically involves the following steps:

1. **Build the app**: Use the `pumpWidget` method to build the widget tree.
2. **Find widgets**: Use the `find` object to locate widgets in the widget tree.
3. **Interact with widgets**: Use the `tap`, `drag`, and `enterText` methods to simulate user interactions.
4. **Check results**: Use the `expect` function to verify the results.

### Example Integration Test for `login_test.dart`

```dart
void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();

  testWidgets('Login Integration Test', (WidgetTester tester) async {
    // Build the app
    await tester.pumpWidget(MyApp());

    // Find the TextFields and RaisedButton
    final usernameField = find.byKey(Key('usernameField'));
    final passwordField = find.byKey(Key('passwordField'));
    final loginButton = find.byKey(Key('loginButton'));

    // Enter text into the TextFields
    await tester.enterText(usernameField, 'username');
```

```
    await tester.enterText(passwordField, 'password');

    // Tap the RaisedButton
    await tester.tap(loginButton);
    await tester.pumpAndSettle();

    // Check the results
    expect(find.text('Welcome, username'), findsOneWidget);
  });testWidgets('Try to login user with correct credentials',
        (WidgetTester tester) async {
      runApp(MainApp());
      await tester.pumpWidget(MaterialApp(home: LoginScreen()));

      final emailWidget = find.byKey(Key('emailKey'));
      final passwordWidget = find.byKey(Key('passwordKey'));
      final loginButton = find.byKey(Key('loginButton'));

      await tester.enterText(emailWidget, 'luc@gmail.com');
      await tester.enterText(passwordWidget, 'luc');

      await tester.tap(loginButton);
      await tester.pumpAndSettle(const Duration(seconds: 2));
    });
}
```

In this example, we're testing the login process of the app. We simulate a user entering their username and password, tapping the login button. Output will be shown in terminal.

---