ICT4153

Mobile Application Development

Practical 03

Create first Flutter Project – 01

Objective

To set up Android Studio for Flutter development and run a sample Flutter application.

Creating an empty template project

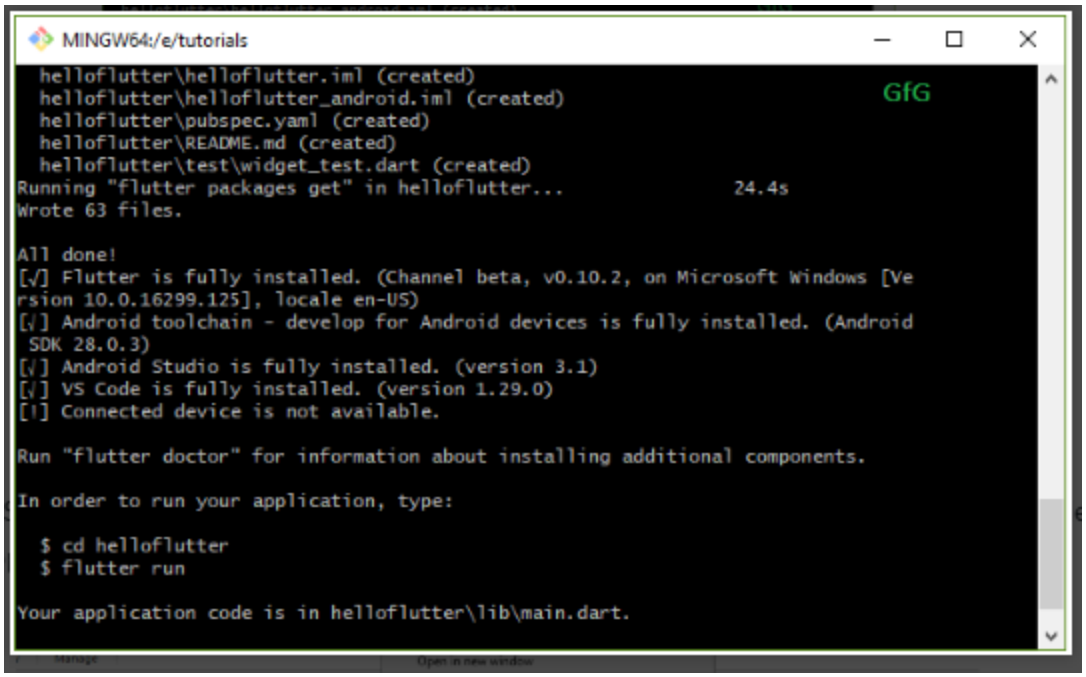
1. Navigate to the place where you want your project to be created. Open a command prompt (you can also use the context **'Git Bash here'** by right clicking, to open Git Bash in this location) and type in the command for creating a new project:

   flutter create project_name

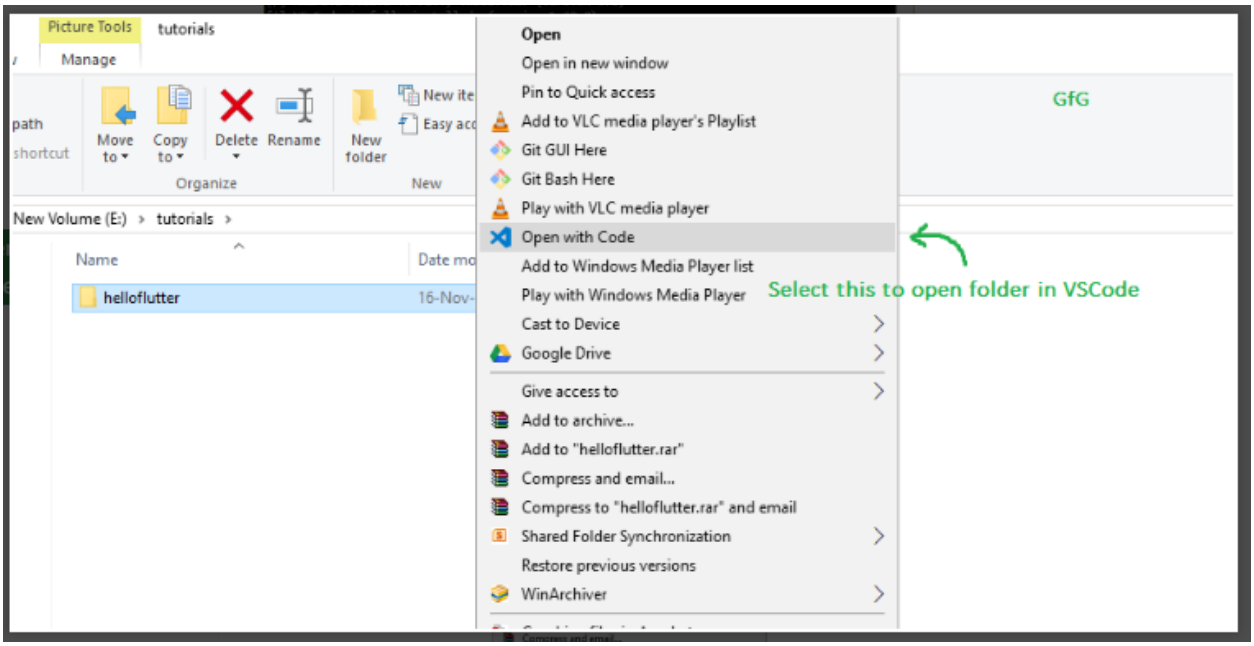
2. For example: for a project named **'helloflutter'** executing

   flutter create helloflutter

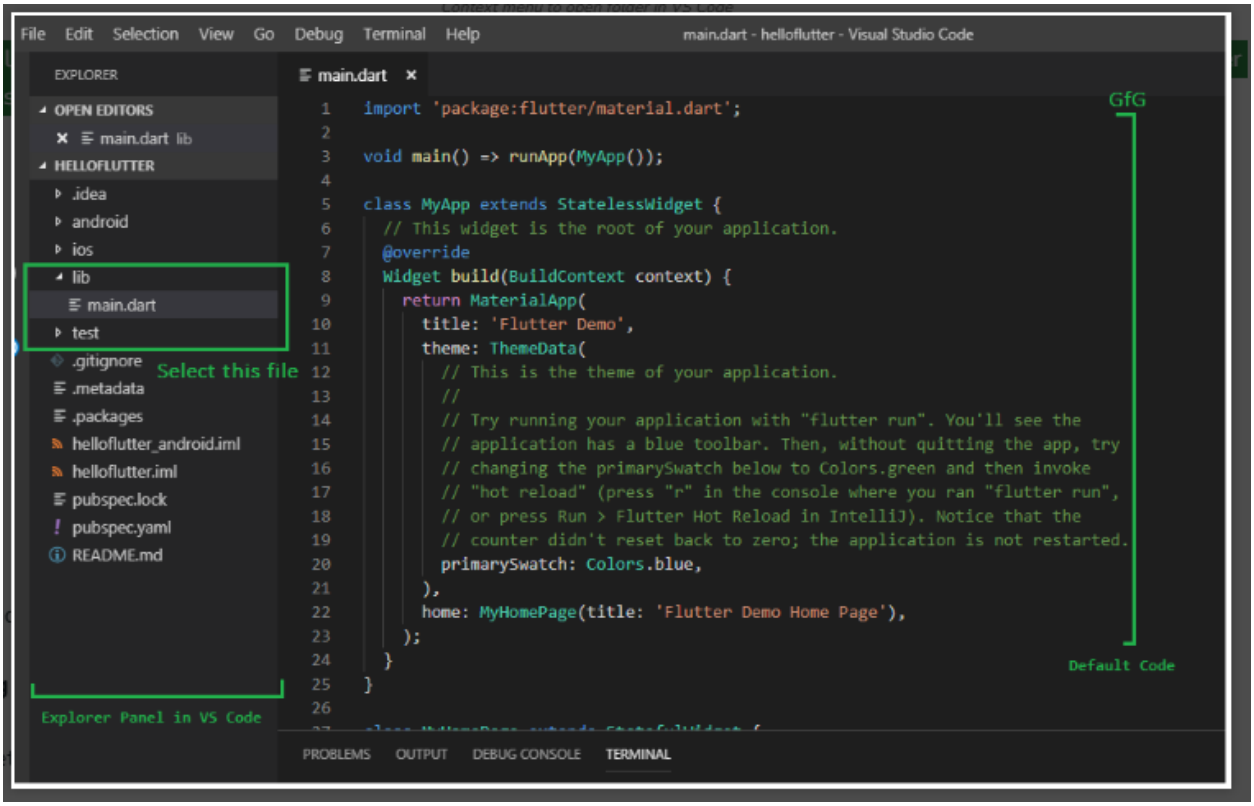
3. will create a new Flutter project with name *helloflutter*



4. Open this folder in VS Code. You can right-click and use the context menu to open directly into VS Code, or start VS Code first and then open this folder as a project.



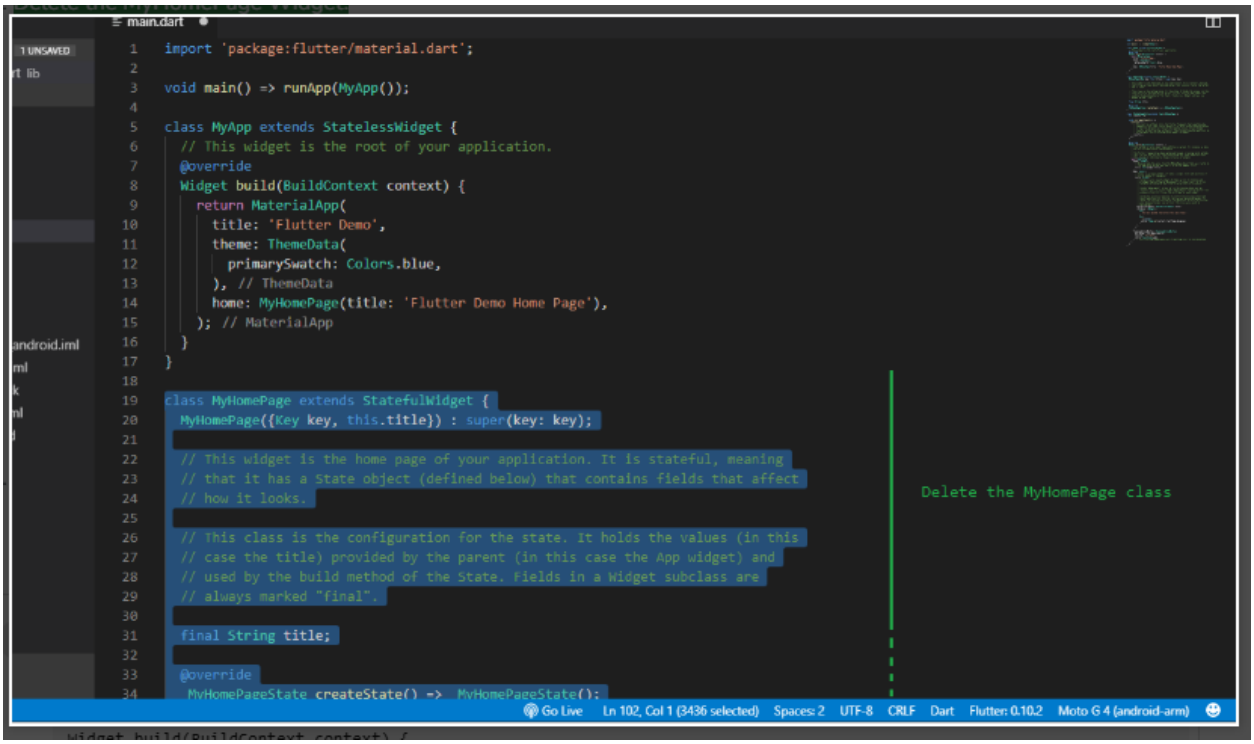
5. The large panel on the left that displays all the files and folders is known as the **Explorer Panel**. Navigate to **'lib'** folder and select the **'main.dart'** file. This file is the entry point from where the app starts its execution.



6. The code that opens up is that of the template application. Try running this simple app right away!

<u>Saying Hello Flutter!</u>

7. Delete the MyHomePage Widget.



8. Create a new Stateless Widget and name it HelloFlutter. Stateless Widgets are used to define Widgets which don't have to deal with changes to its internal state. They are mostly used to build components which once drawn, are not required to update.

```
class HelloFlutter extends StatelessWidget {
  const HelloFlutter({Key? key}) : super(key: key);

@override
Widget build(BuildContext context) {
    return Container(

    );
}
}
```

```
13        ), // ThemeData
14        home: MyHomePage(title: 'Flutter Demo Home Page'),        GfG
15      ); // MaterialApp
16    }
17  }
18
19  class HelloFlutter extends StatelessWidget {
20    @override
21    Widget build(BuildContext context) {
22      return Container(
23
24      );
25    }
26  }
```

new Widget named HelloFlutter

9. Replace the Container widget with a Scaffold widget: A Scaffold implements basic material design visual layout structure. This Widget provides APIs for showing drawers, appbars and the body of the app. The body property of the Scaffold will be used here to display the contents of the app.

```
class HelloFlutter extends StatelessWidget {
  const HelloFlutter({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return const Scaffold(

    );
  }
}
```

10. Declare a Container Widget in the body of the Scaffold. A Container Widget is a useful widget which combines common painting, positioning, and sizing widgets. You can wrap any widget with a Container and control the above mentioned properties.

```
class HelloFlutter extends StatelessWidget {
  const HelloFlutter({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(

      ),
    );
  }
}
```

11. The Container widget has an alignment property which will help to position the Widget to the center of the screen. Set the alignment with the Alignment class:

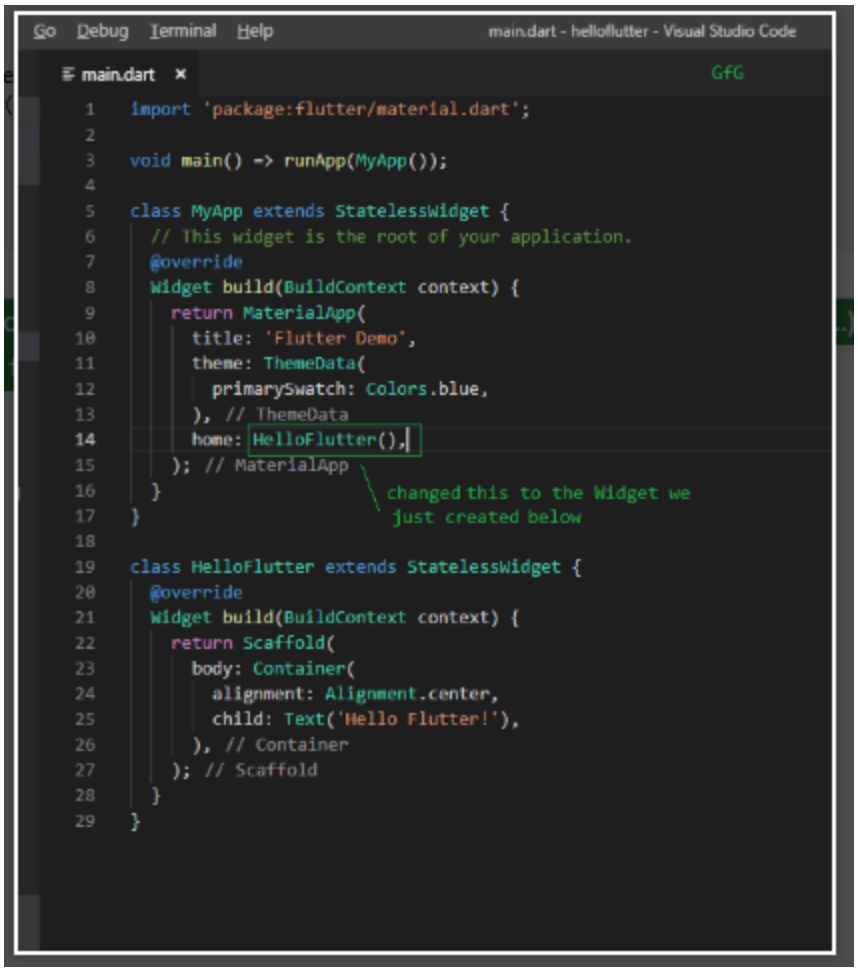
```
alignment: Alignment.center
```

12. In the child property of the Container Widget, declare a Text Widget: The Text Widget deals with displaying and handling text. After creating the Text Widget, put in 'Hello Flutter' between the parentheses in single quotes. Whatever is put in between the single quotes is displayed by the Text Widget.

```
class HelloFlutter extends StatelessWidget {
  const HelloFlutter({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        alignment: Alignment.center,
        child: const Text('Hello Flutter!'),
      ),
    );
  }
}
```

13. Finally, in the home property of the main My App class above, change it from MyHomePage(…) to HelloFlutter(). This allows the main MyApp class to refer to the Hello Flutter just created.

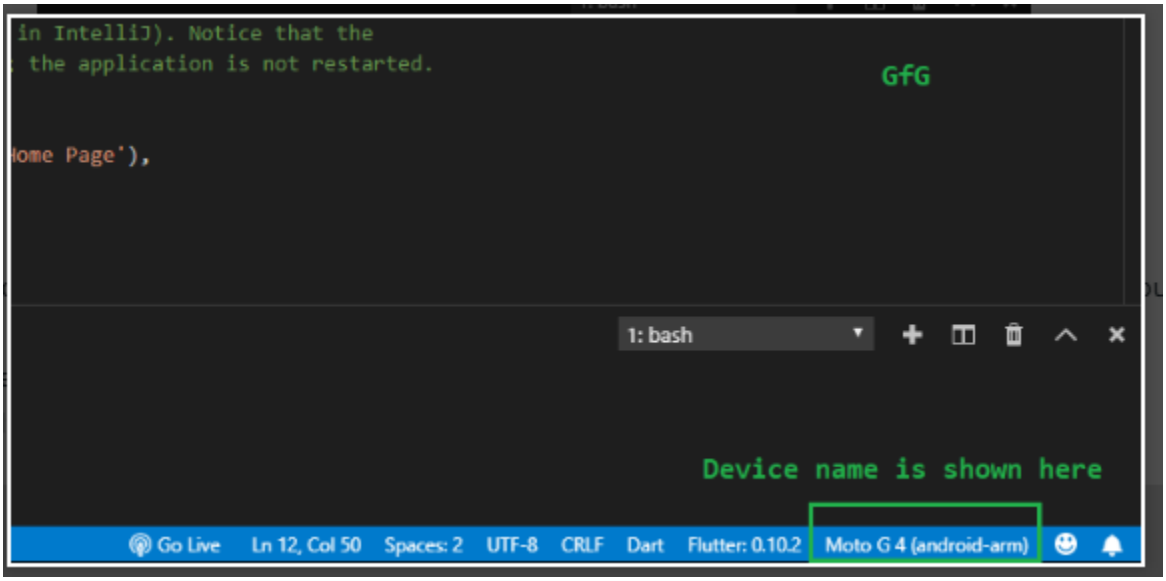


14. Now run the app using the '*flutter run*' command.

15. A text 'Hello Flutter!' will appear written in the middle of the screen.

Running the HelloFlutter App

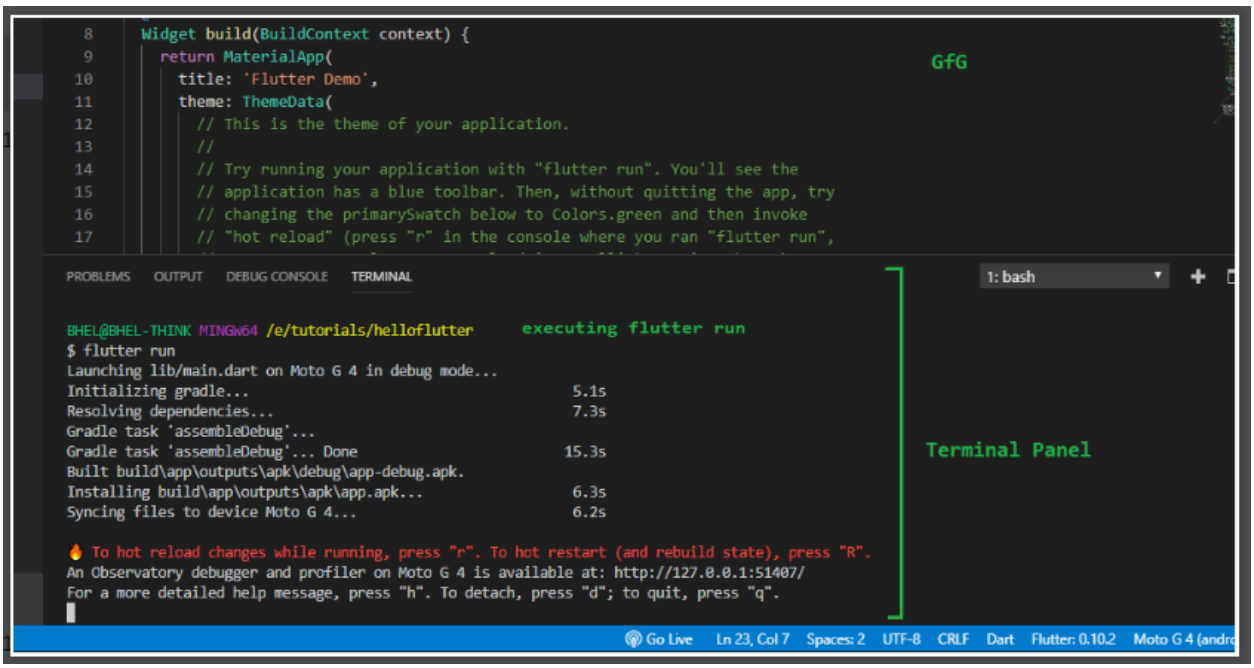
16. Connect a physical device to the PC and enable Developer Mode. If the device is successfully recognised by the PC, the device name would appear in the lower-right corner of VS Code.



17. If you would like to setup an emulator instead, see: [Set up the Android emulator](). The emulated device would also show up here similarly.
18. Open the integrated terminal by pressing the key combination [CTRL + `] (Control key + backtick).
19. Run the command:

```
flutter run
```

20. Wait for a few minutes. As this is first run, some downloads and installation take place in the background related to gradle. Subsequent compilations would be a lot faster.



21. After compilation, the app will get installed and run on the connected device or emulator automatically.

22. Stop the app by pressing ‘d’ in the terminal. This is what the compilation and running of any app will be like.