Name Musadique Hussain Roll no SE-21031

LAB SESSION 1

SETTING UP ANDROID STUDIO

Welcome to the first lab of the Mobile Application Development course! In this lab, you'll get started with Flutter, an open-source UI toolkit, and set up your development environment using Android Studio. Flutter allows you to build natively compiled applications for mobile, web, and desktop from a single codebase.

**Theoretical Concepts:**

1. **What is Flutter?**

Flutter is an open-source UI toolkit developed by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It uses the Dart programming language and provides a rich set of pre-designed widgets for building user interfaces.

1. **Flutter Architecture:** 
   * + **Dart Programming Language:**
     + Dart is the programming language used for Flutter development. It's designed for building web, server, and mobile applications.
     + **Flutter Framework:**
     + The Flutter framework provides a reactive-style framework, a set of pre-designed widgets, and tools for building user interfaces.
2. **Setting up the Development Environment:**

**Installing Flutter and Dart:**

Follow these steps to install Flutter and Dart on your machine:

* 1. Download the Flutter SDK from flutter.dev.
  2. Extract the downloaded ZIP file to your preferred location.
  3. Add the Flutter  **bin**  directory to your system's PATH.

**Configuring Android Studio:**

* 1. Install Android Studio from developer.android.com.
  2. Install the Flutter and Dart plugins in Android Studio.
  3. Configure the Flutter SDK path in Android Studio.

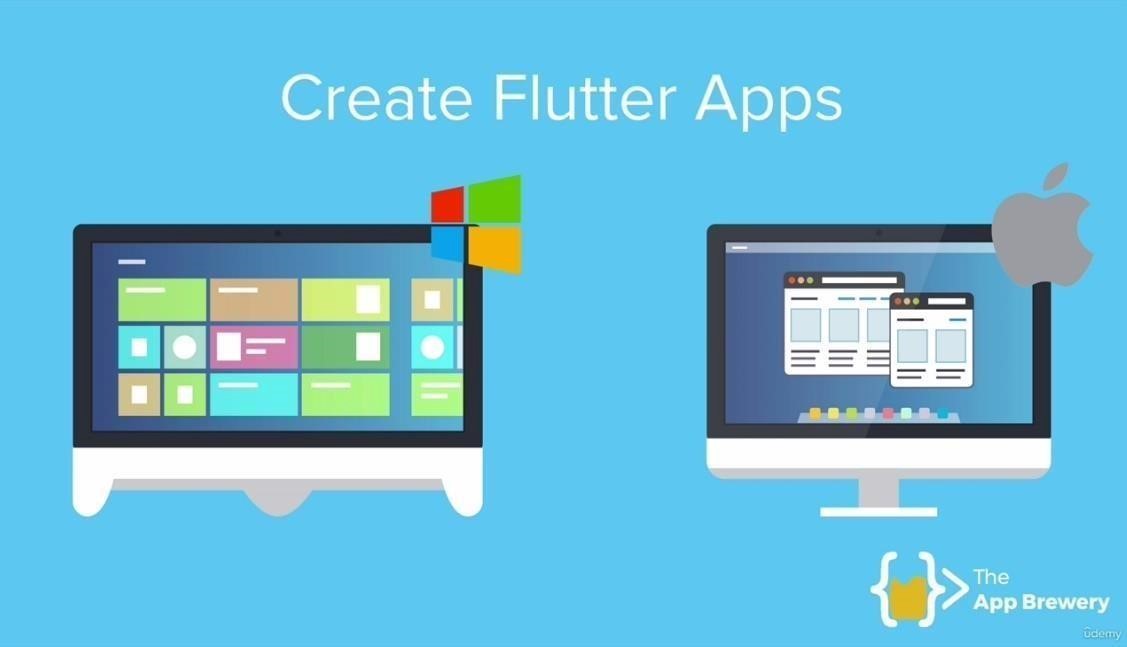
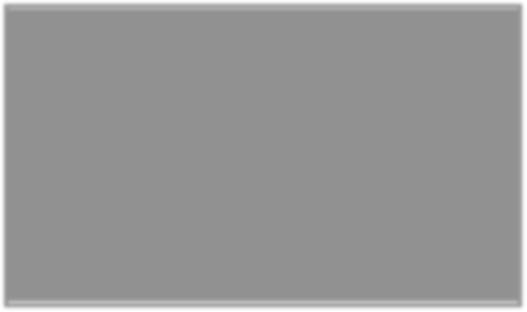
1. **Introduction to Flutter Project Structure:**

A typical Flutter project consists of the following key directories and files:

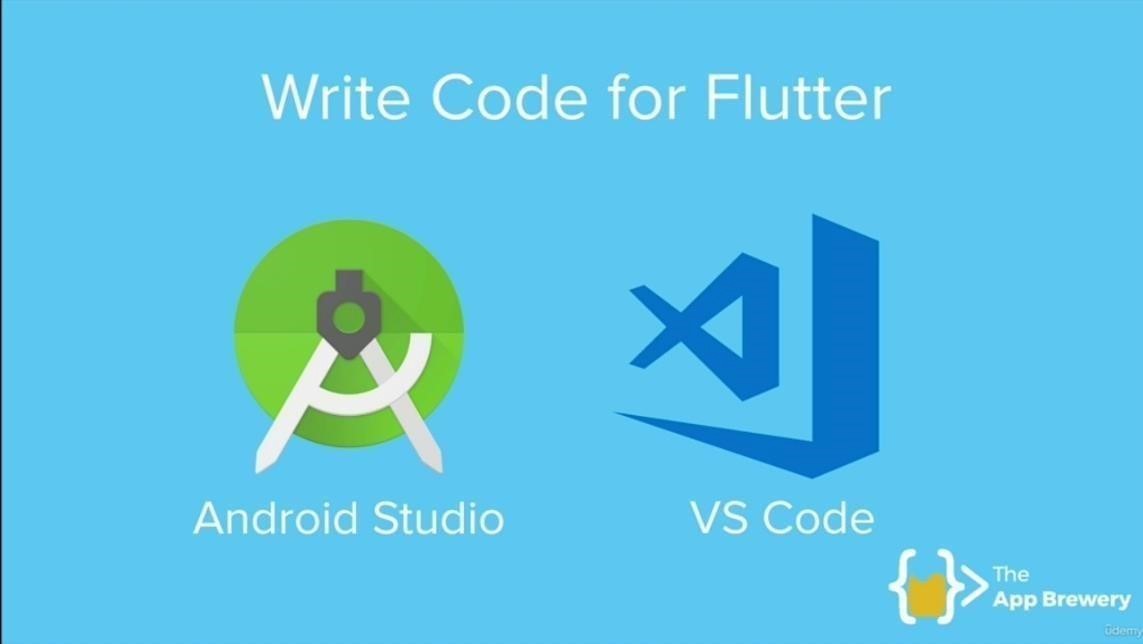
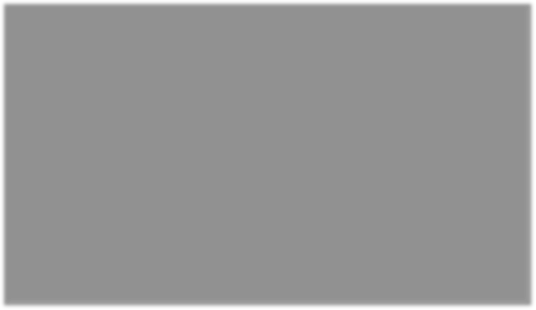
* **lib Directory:**
* Contains the main Dart code for the application.
* **android and ios Directories:**
* Contain platform-specific code for Android and iOS.
* **pubspec.yaml File:**
* Defines the project's dependencies and metadata.

**Explanation:**

Before installing flutter for application development, we should know what we need to be able to work with flutter and also run your applications on ios and android. Firstly, we are going to need a computer which can be mac or windows.

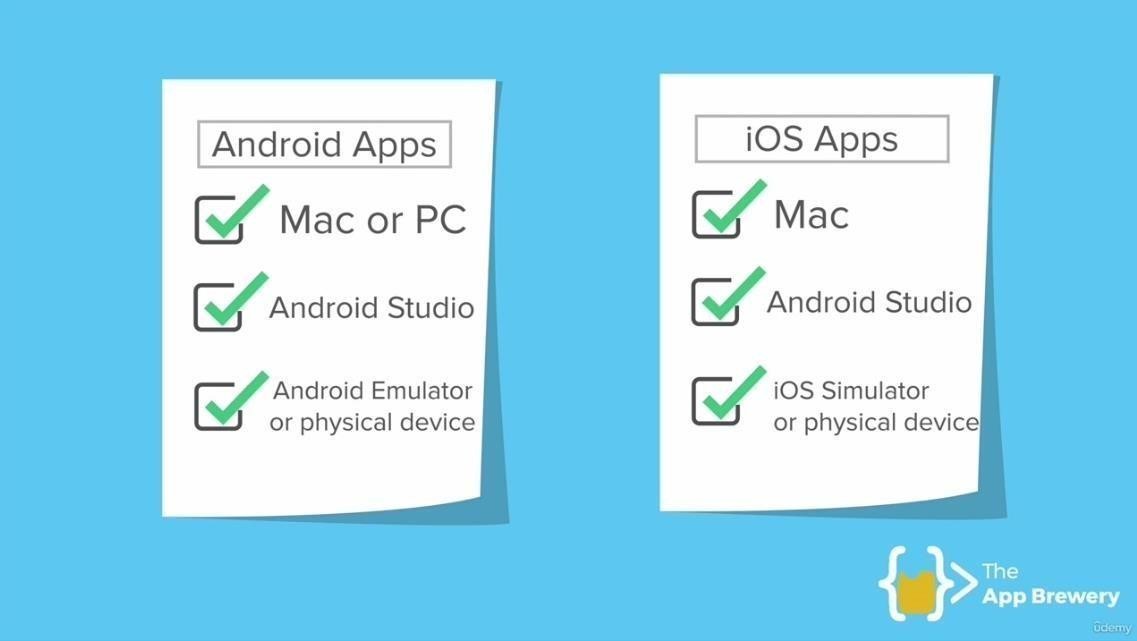
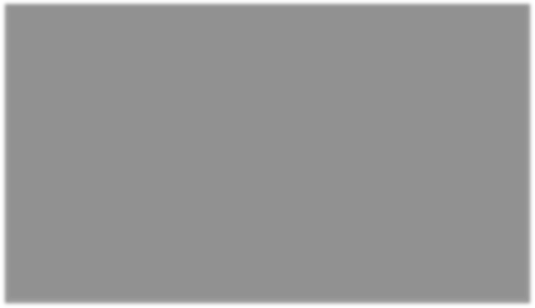


The next step is to have a code editor in order to be able to build flutter applicationsand the two contenders that you should consider are android studio and VS code and they both are pretty capable in terms of developing flutter applications.

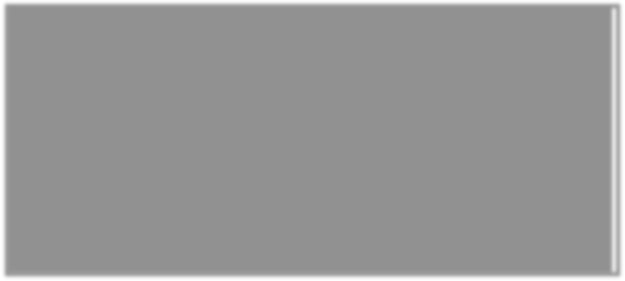
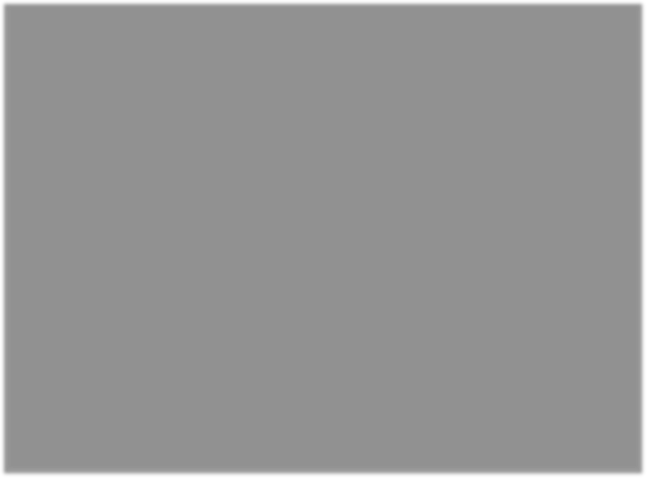
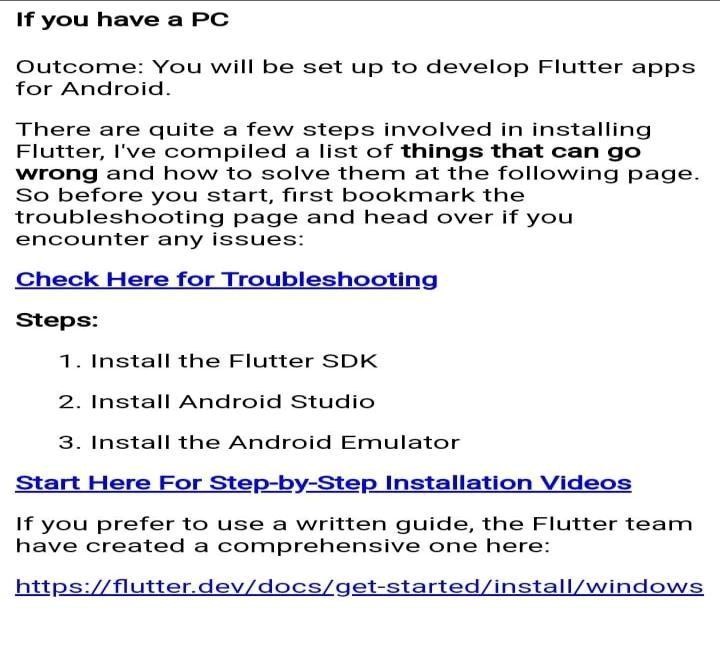
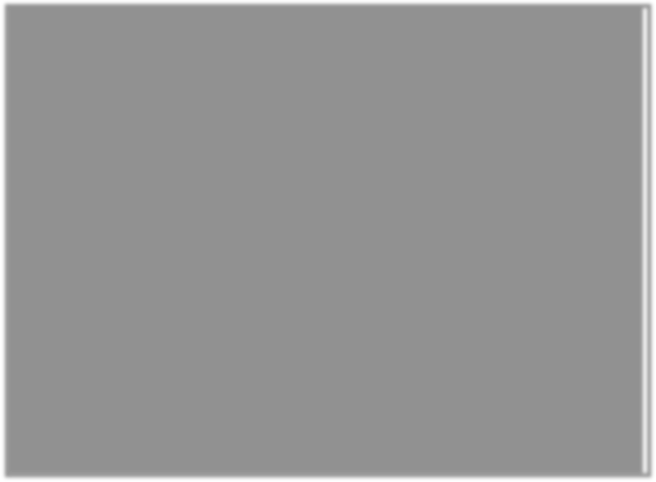


But our preference for the labs will be the android studio because:

* It makes it easier to work with the emulator.
* It allows us to upgrade android x or jet pack easily.
* Even if you are using vs code but down the line you will still be needing android studio.

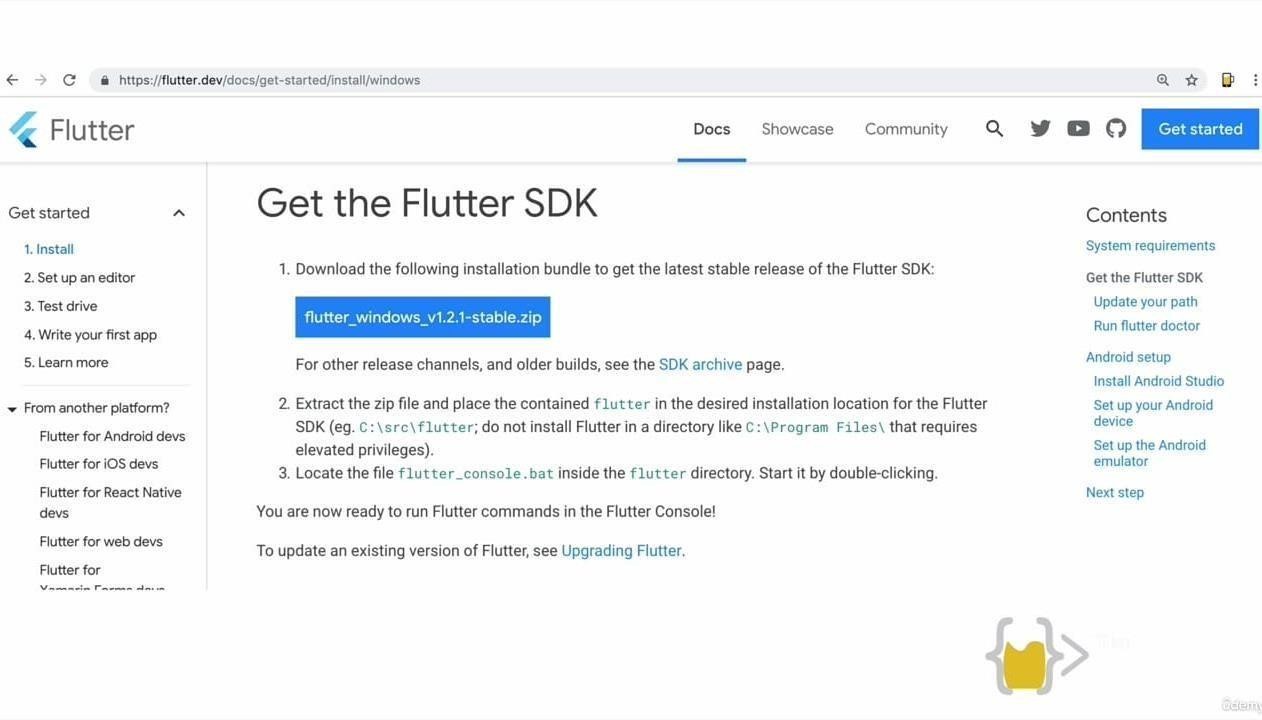
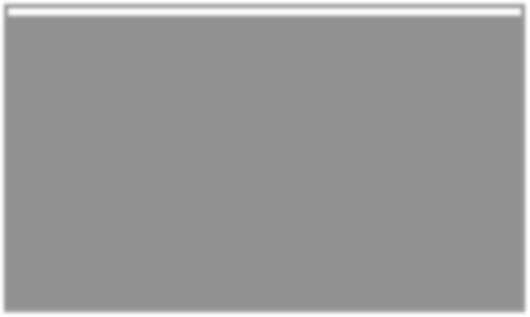
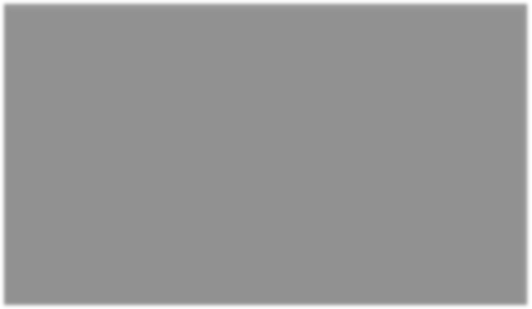


Android studio is a fully powered IDE since it suggests for completion and works seamlessly with our emulators and simulators and running our applications on devices as well.

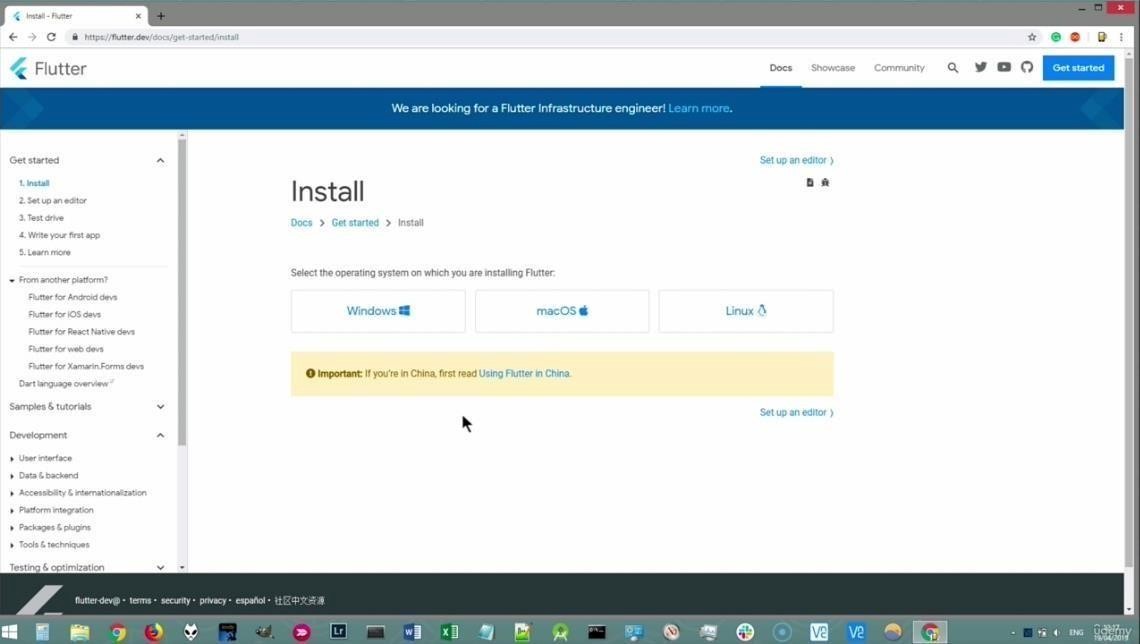
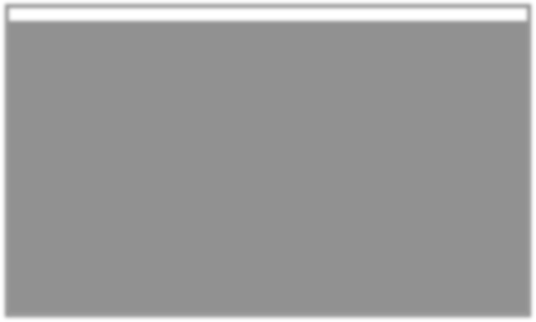


Minimum system requirements for flutter development

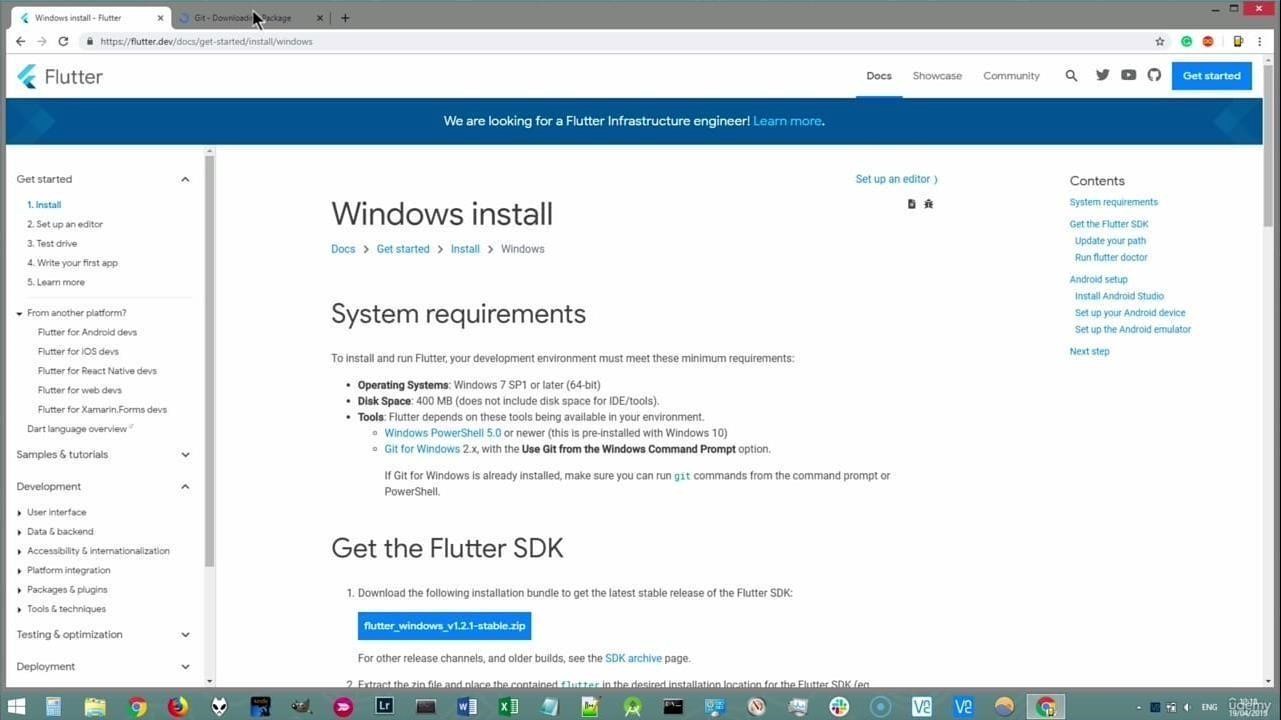
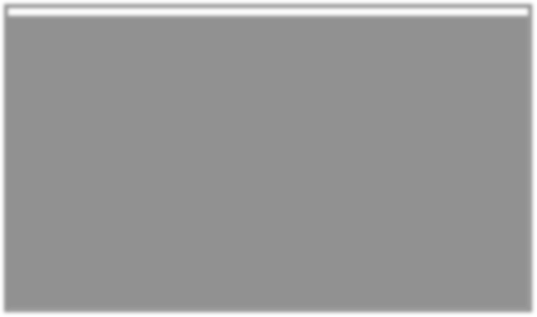
First we are going to need the flutter SDK



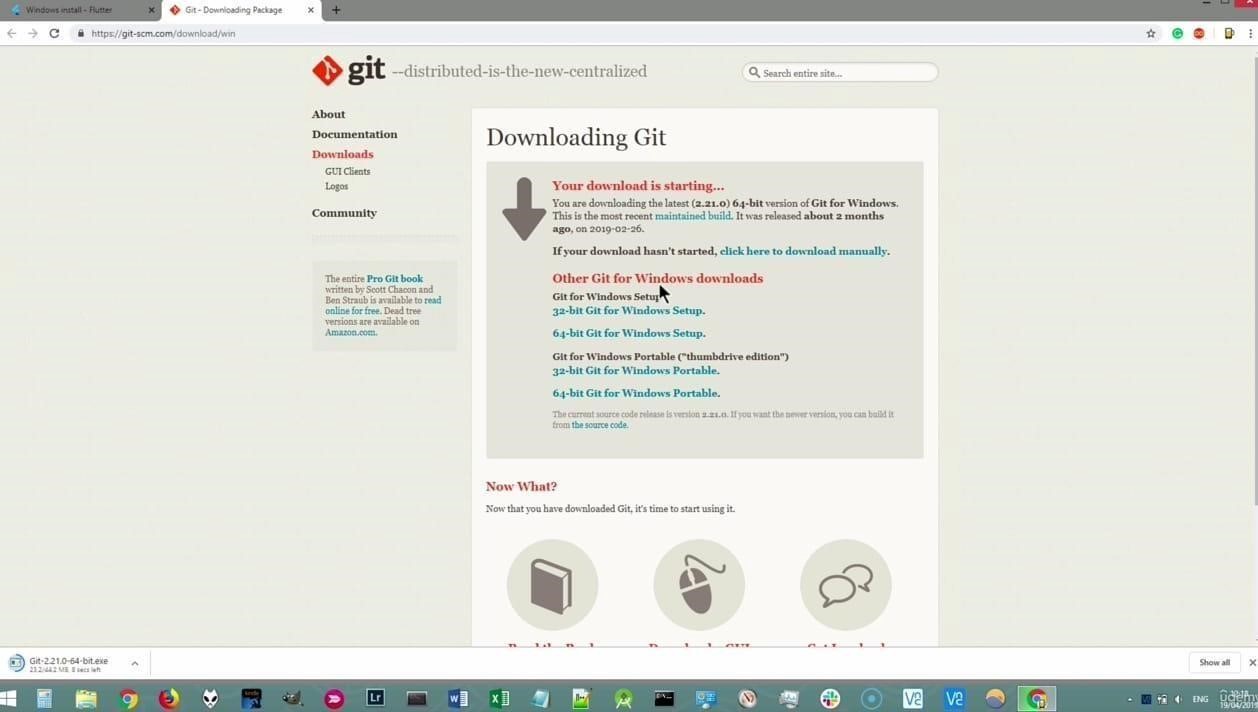
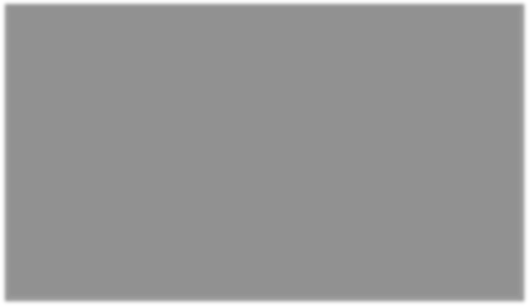
Once you have clicked on get started you are going to select the OS with whichyou will be installing flutter.



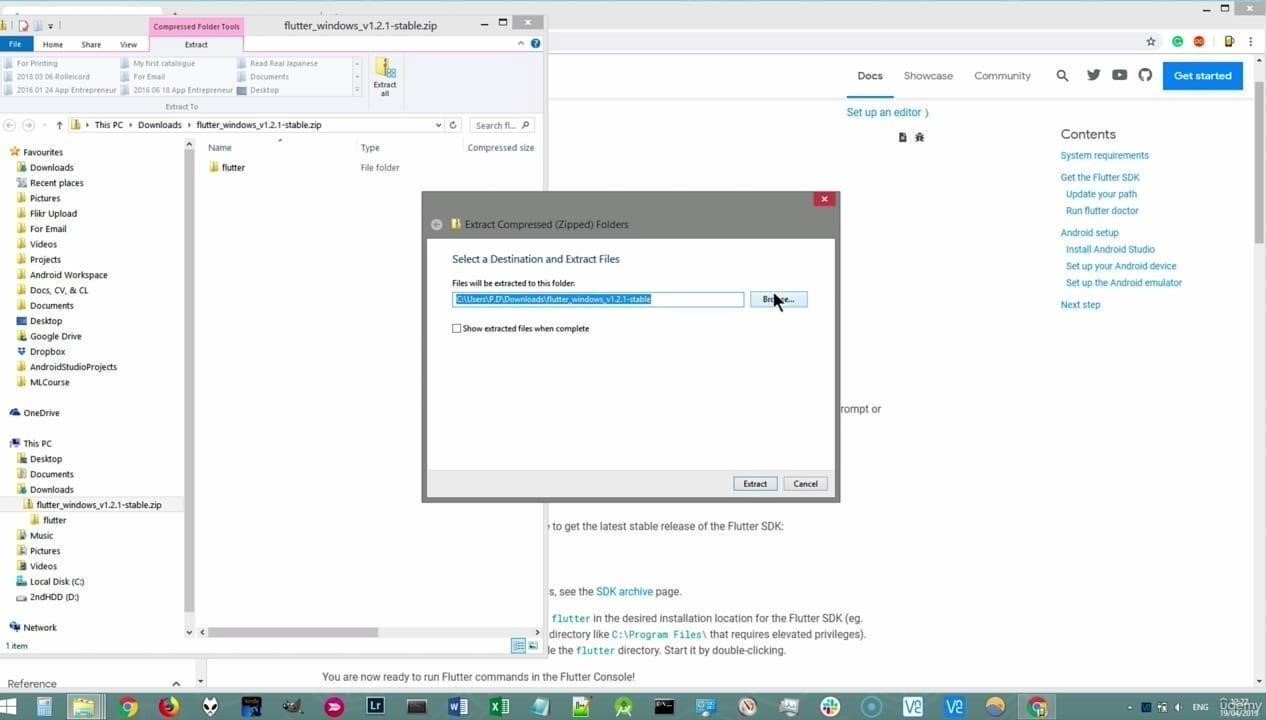
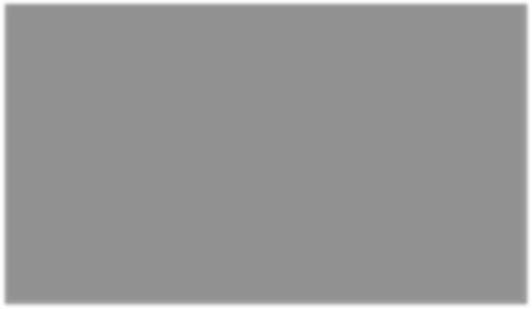
At this point if you don’t have git installed .be sure to install git by clickingon the link git for windows



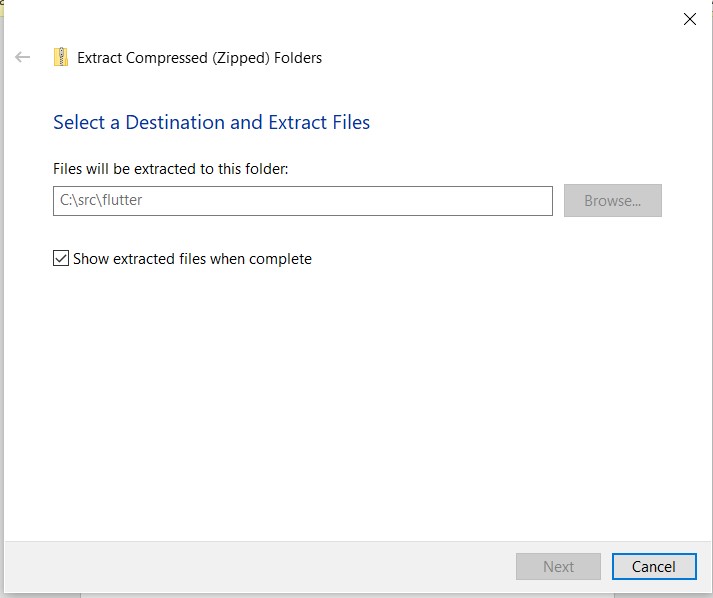
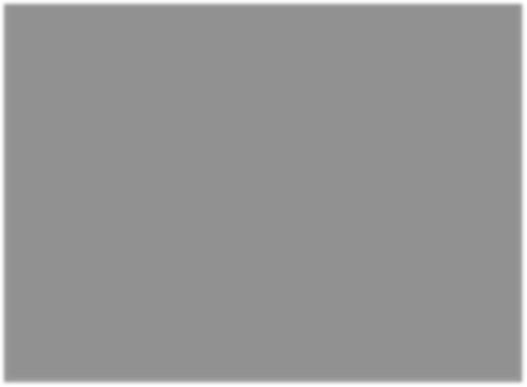
By selecting git it will take you to this window. Download it by a simple step bystep widget.



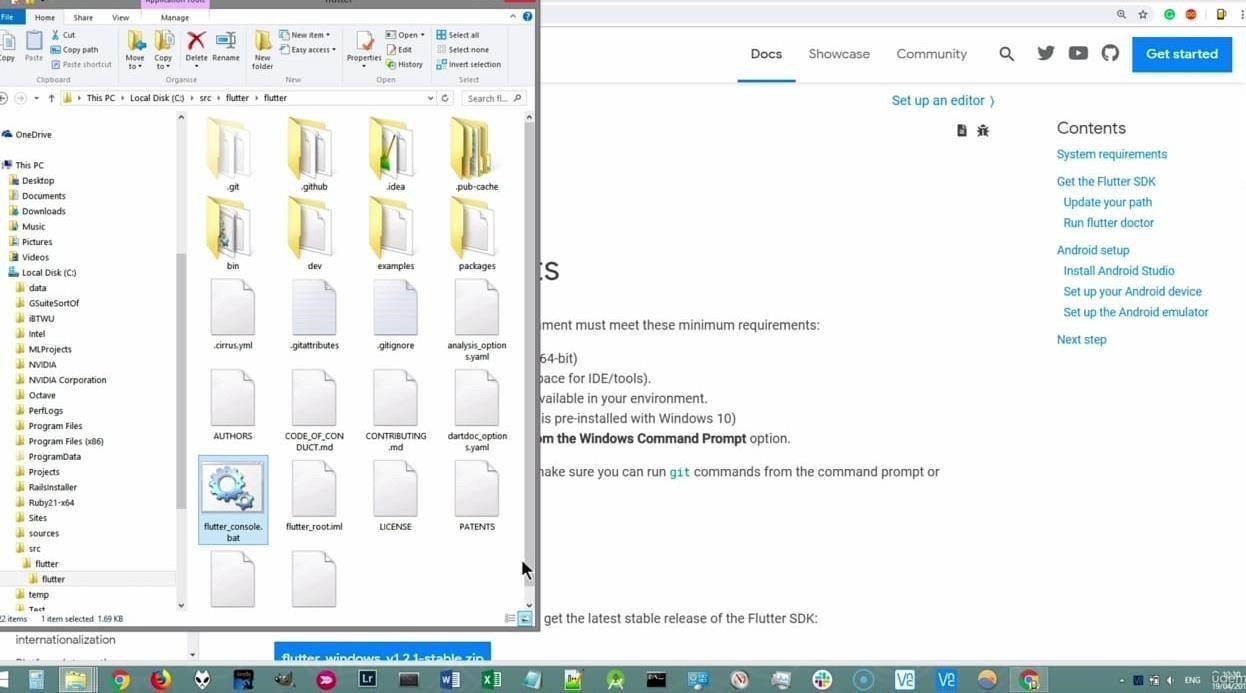
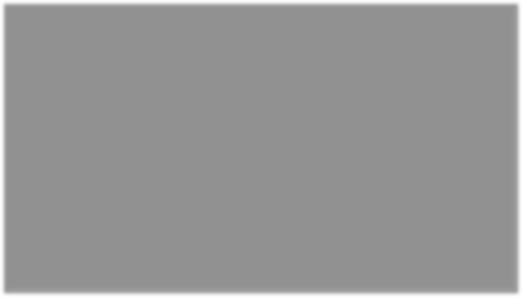
After installing git for windows, the next step is to go for flutter SDK. All you needto do is to click the flutter\_windows button and install the zip file. Once you havedownloaded the zip file click on the file and then click extract all which is very important.



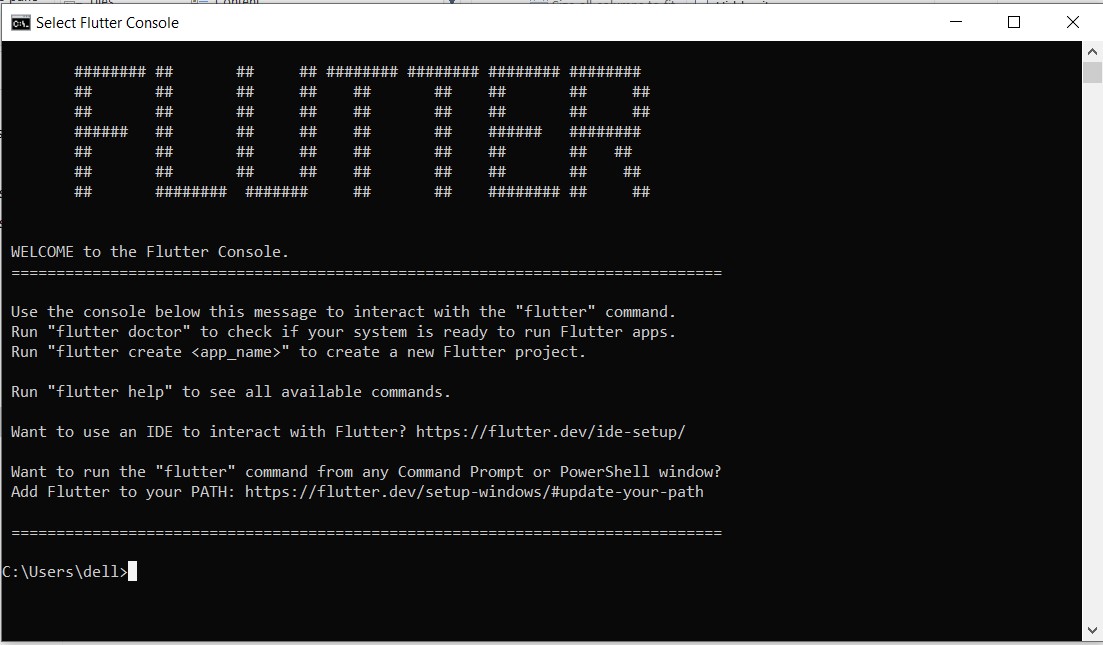
The path should be set as follows:



Once you are done with extraction you will be able to access the folders inside the flutter folder.

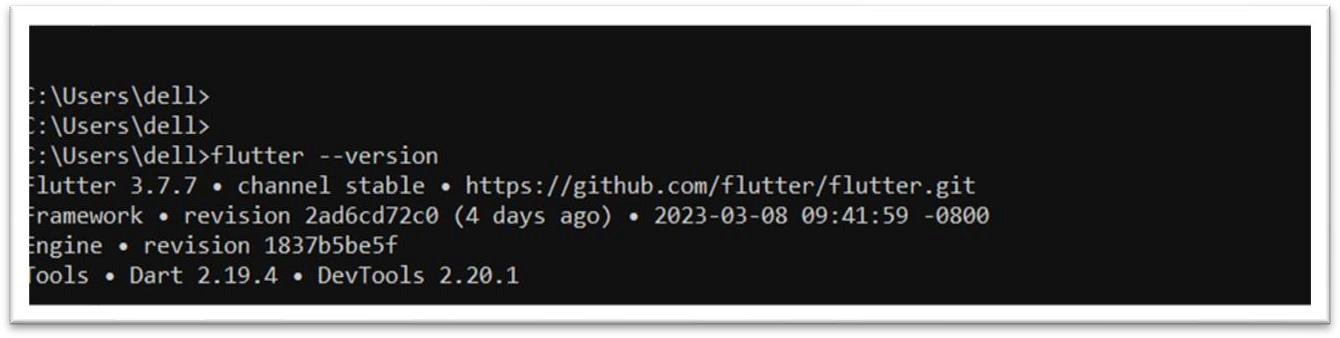


In order to run flutter commands you will be needing flutter\_bat file .

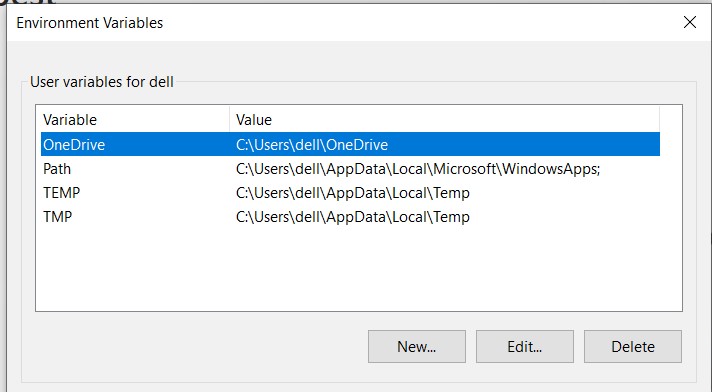
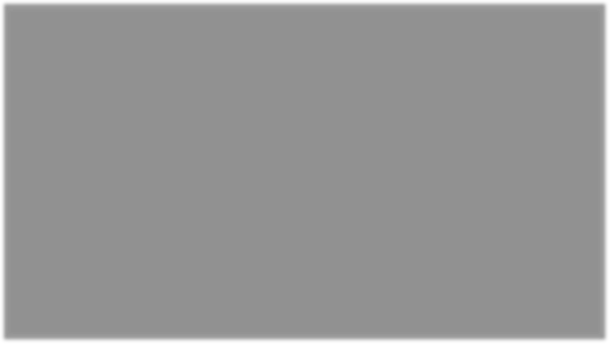


Now you can use the console to run flutter commands like Flutter –

version //which lets you know the version installed.



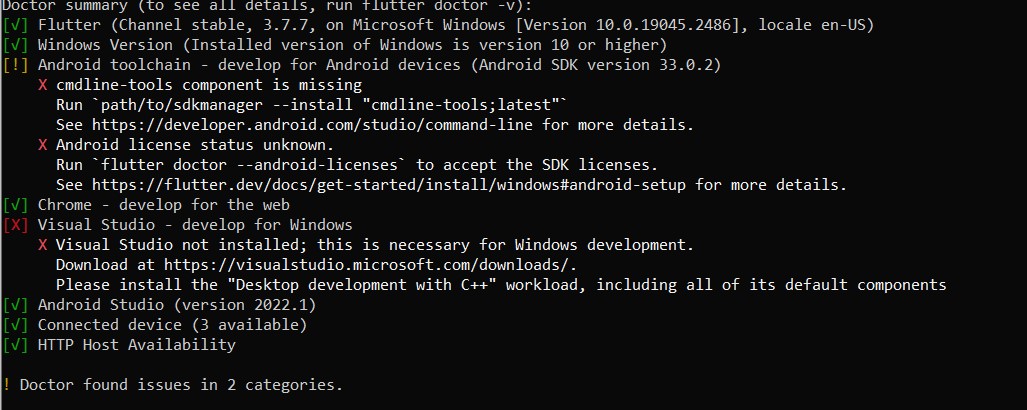
But when we run flutter from command prompt it knows how to locate the flutterpackage. But if we want to use it from somewhere else like android studio or even from a normal command prompt so we would need to locate the flutter package.to prevent having to do it every single time, we are going to set it as our path variable. Search for edit environment variables for your account.



In these environment variables. Check for the entry of path. If the entry is not present you can make a new variable in the name of path. After this you can add the new path as follows:

C:\src\flutter\flutter\bin

By doing this you are able to run flutter from anywhere. Another useful flutter command is flutter doctor which will diagnose how our flutter is setup or if there is any other thing that we still need to sort out before we can continue.

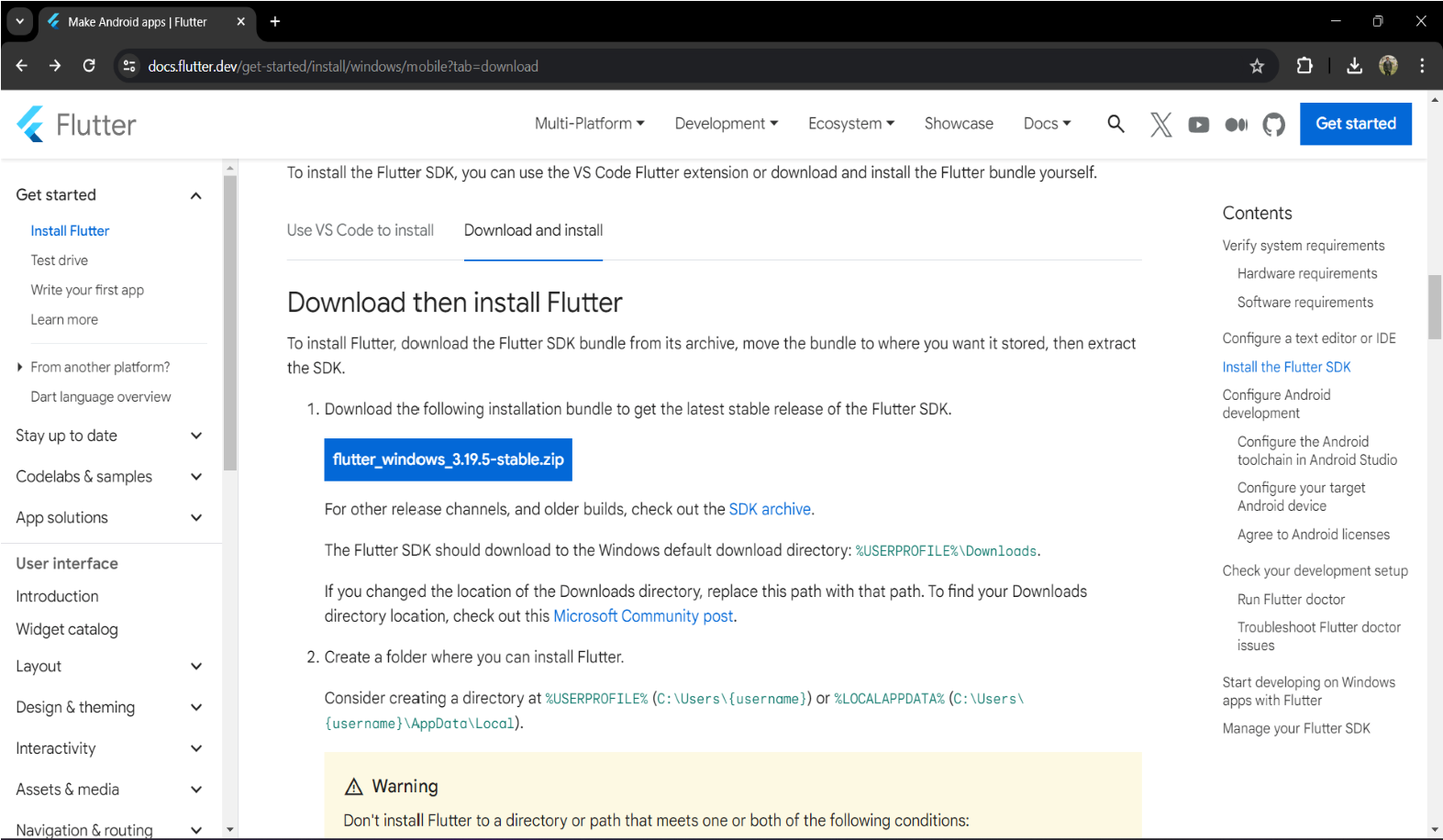


**Lab Exercise:**

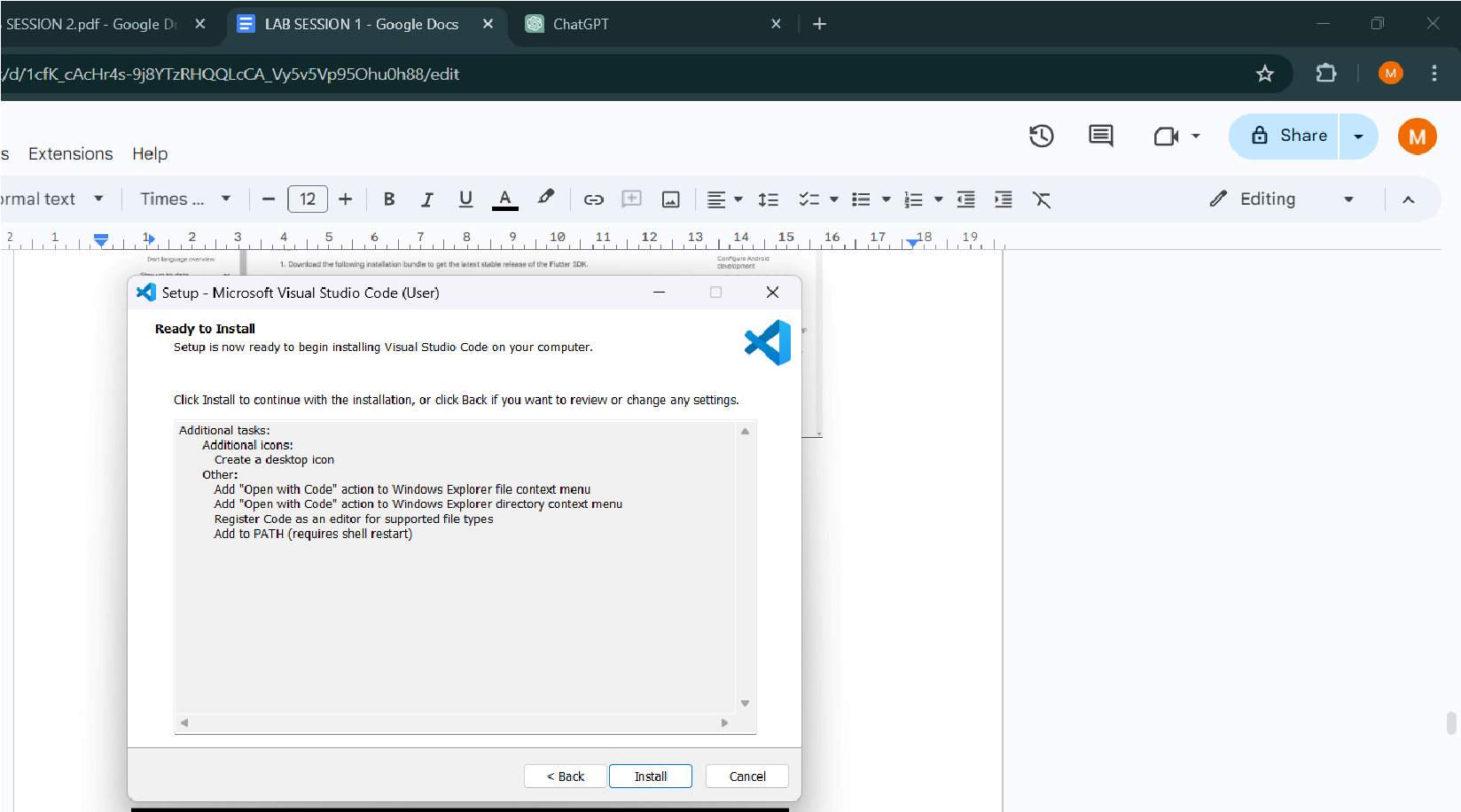
**Perform the following tasks and attach screenshots:**

**1. Installation:**

1. Install Flutter SDK:



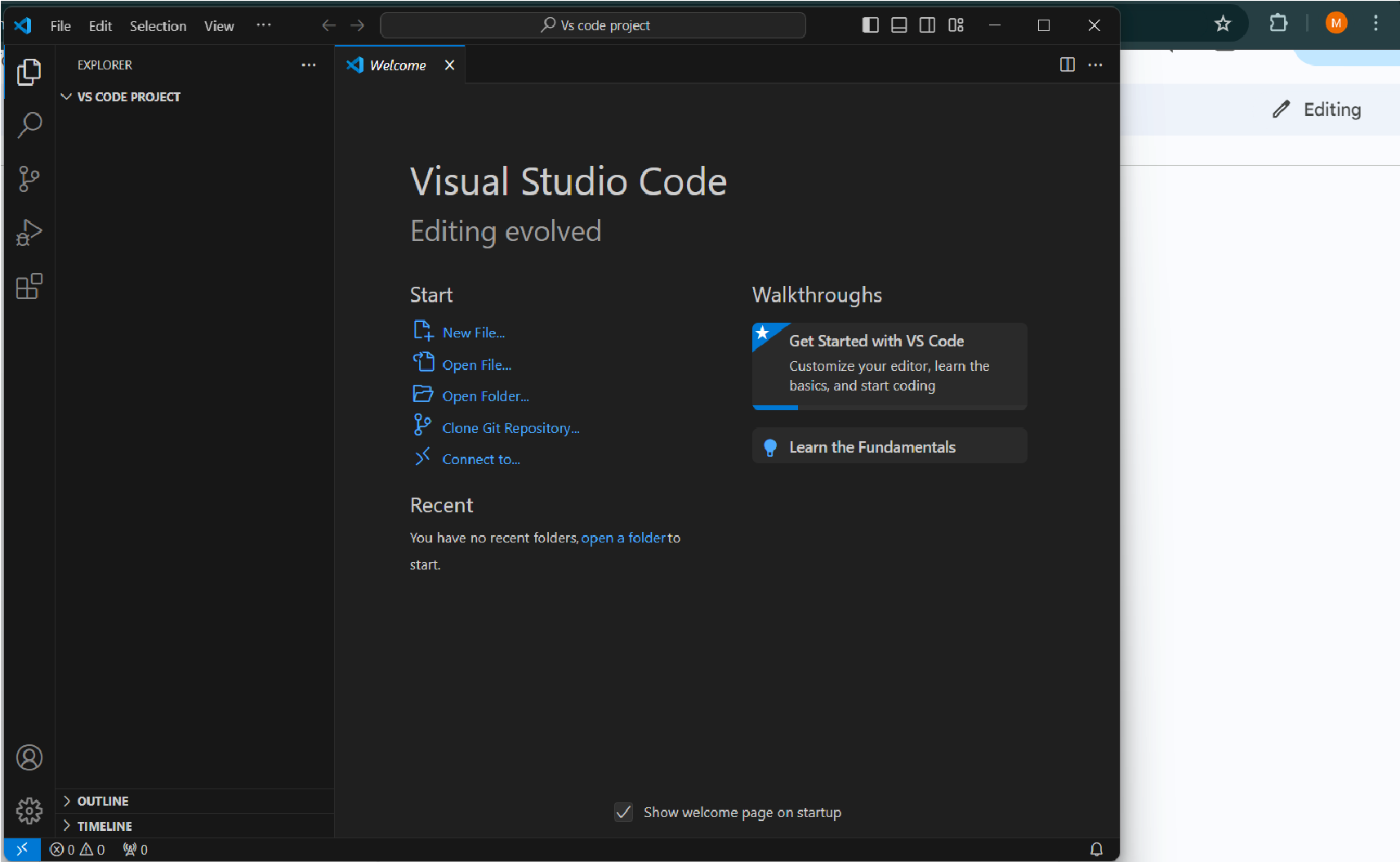
1. Install Android Studio:



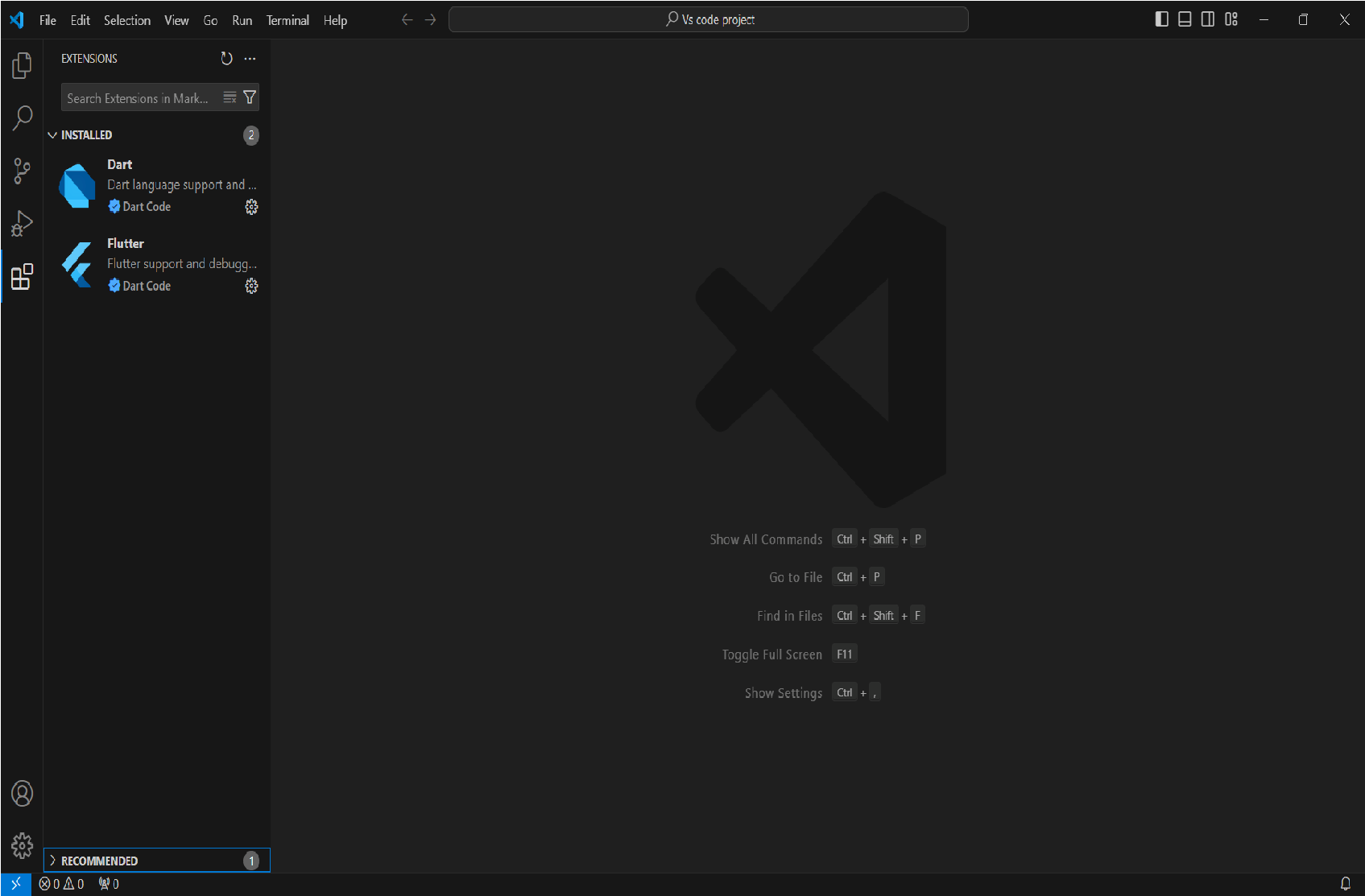
**2. Configuration:**

a. Configure Flutter in Android Studio:

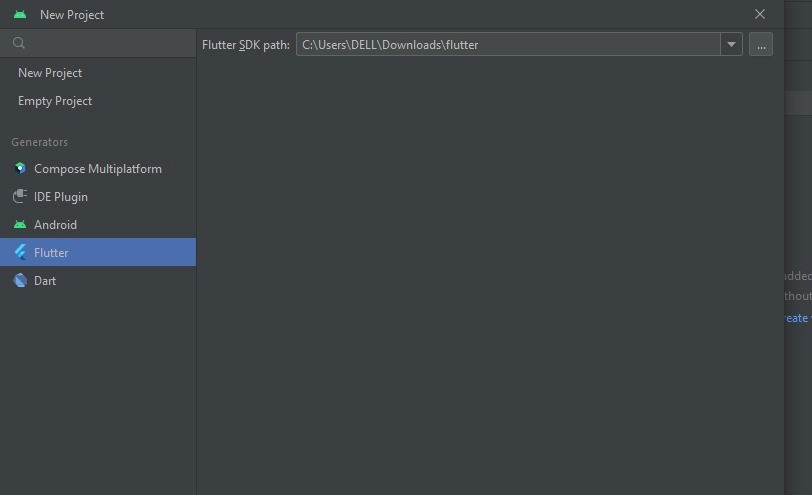
1. Open Android Studio.



1. Install the Flutter and Dart plugins.



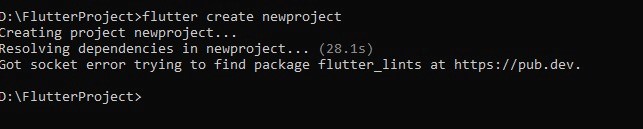
1. Set the Flutter SDK path in Android Studio.



**3. Creating Your First Flutter App:**

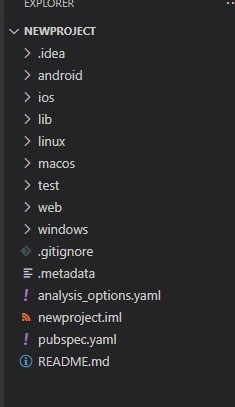
1. Create a New Flutter Project:

Open a terminal and run the following command to create a new Flutter project:



1. Explore the Project Structure:

Navigate into the created project ( **cd my\_first\_app** ) and explore the contents.

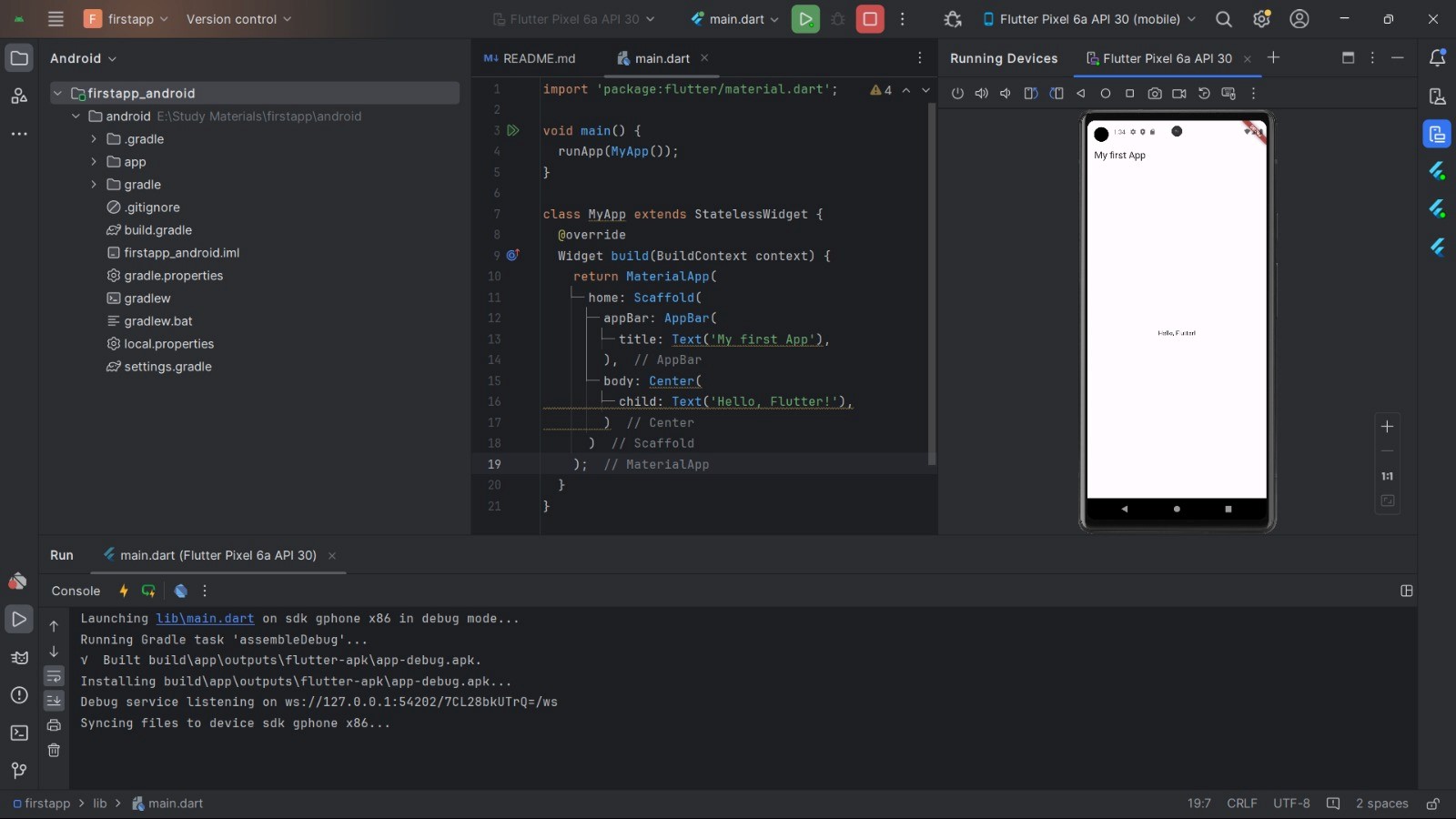


**4. Running the App:**

1. Run on Emulator:

Run the following command to launch the app on an emulator:

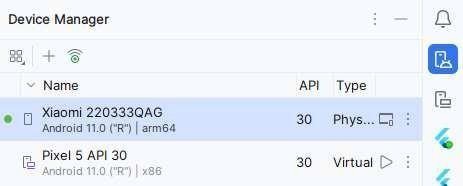




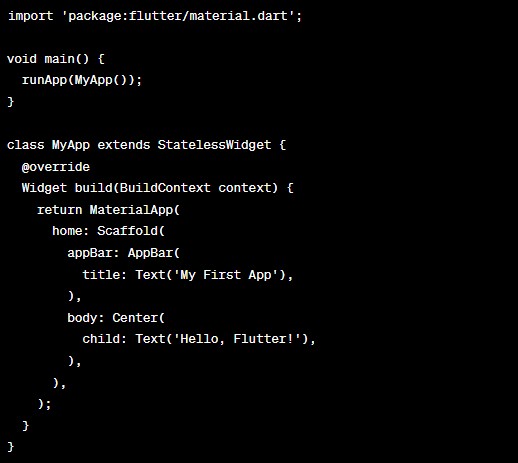
1. Run on a Physical Device:

Connect your physical device and run the app using the same command.

**Code Snippets:**



**Dart Code in lib/main.dart:**



**Answer the following Questions:**

**1. What is Flutter, and what makes it suitable for cross-platform development?**

**FLUTTER:**

Flutter stands out as a pivotal tool in the realm of cross-platform development, offering a versatile solution for crafting applications across mobile, web, and desktop environments. Its significance is underscored by several key attributes:

**Comprehensive SDK:** Flutter arrives equipped with a robust Software Development Kit encompassing libraries, tools, and APIs. This inclusive package streamlines the process of app development, testing, debugging, and deployment across various platforms.

**Reactive Framework and Widget Library:** Leveraging a reactive framework, Flutter boasts an expansive widget library. This arsenal includes reusable UI components, layout widgets, styling provisions, animations, and gesture support, empowering developers with ample resources for crafting dynamic user experiences.

**Dart Language Foundation:** Built upon the Dart programming language, Flutter capitalizes on its object-oriented paradigm. Dart's strengths, such as strong typing, asynchronous programming capabilities, and a reactive programming model, fortify Flutter's development ecosystem.

**Emphasis on Code-based UI Development:** Flutter eschews drag-and-drop interfaces in favor of code-based UI design. This approach fosters flexibility, control, and consistency, enabling developers to sculpt interfaces with precision and clarity.

**Absence of Visual Editors:** In lieu of visual editors, Flutter advocates for code editors supplemented by Flutter plugins. This promotes the cultivation of clean, concise code while harnessing the full potential of Flutter's widget system.

**Single Codebase Efficiency**: With Flutter, developers harness the power of a unified codebase, effortlessly deploying applications across iOS, Android, web, and desktop platforms. This consolidation drastically reduces development overhead by eliminating the need for separate codebases for each platform.

**Rapid Development Cycle:** Flutter accelerates the development process through its hot reload feature, facilitating real-time code modifications and seamless integration of changes without the need for complete application restarts.

**Native Performance Optimization:** By compiling Flutter apps to native machine code, developers unlock near-native performance across diverse platforms. This optimization ensures fluid user experiences and high performance standards.

**Access to Platform-specific Features**: Despite its shared codebase, Flutter grants developers access to platform-specific features and APIs via plugins. This enables the seamless integration of device-specific functionalities, enriching the user experience while maintaining code uniformity.

**2. Explain the role of the Dart programming language in Flutter development.**

**ROLE OF DART PROGRAMMING LANGUAGE IN FLUTTER DEVELOPMENT:**

**Foundational Language:** Dart serves as the primary language for crafting Flutter applications, providing the backbone for implementing logic, defining UI components, managing state, and handling user interactions.

**Object-Oriented Approach:** Dart embraces the object-oriented programming paradigm, empowering developers to structure code into classes and objects. This fosters code reusability, modularity, and facilitates the maintenance of extensive applications.

**Robust Typing System:** Dart adopts static typing, ensuring that variables possess specific types known at compile-time. This robust typing mechanism aids in early error detection, bolstering code reliability and enhancing readability.

**Asynchronous Capabilities**: Dart offers built-in support for asynchronous programming through constructs like async/await and Futures. This feature is instrumental in managing asynchronous tasks such as network operations, file handling, and non-blocking UI updates.

**Reactive Programming Model:** Dart's reactive programming model seamlessly aligns with Flutter's framework, empowering developers to construct reactive user interfaces. This model streamlines data flow, state management, and facilitates UI updates in response to user interactions or data alterations.

**Compilation Efficiency:** Dart employs Just-In-Time (JIT) compilation during development and Ahead-Of-Time (AOT) compilation for production environments, ensuring optimal performance of Flutter applications. JIT compilation facilitates hot reload functionality, enabling developers to witness instant changes during development, while AOT compilation generates optimized native code for superior performance in production settings.

**3. Why is it important to configure Flutterer and Dart in Android Studio?**

**Describe the purpose of the lib directory in a Flutter project.**

**Seamless IDE Integration:** Android Studio seamlessly incorporates Flutter and Dart into the development workflow, offering indispensable tools such as code completion and debugging assistance.

**Enhanced Development Capabilities**: Leveraging the Flutter plugin within Android Studio enhances development by providing essential features like hot reload functionality and widget inspection, fostering a smoother development experience.

**Comprehensive Dart Language Support:** Android Studio offers robust support for Dart, encompassing features such as syntax highlighting and comprehensive code analysis, facilitating efficient coding practices.

**Streamlined Emulator and Device Management:** Android Studio equips developers with tools for efficient management of emulators and physical devices, simplifying the testing process across various Android configurations.

**Access to Community and Resources:** Android Studio's expansive ecosystem grants developers access to a vibrant community and abundant resources, enabling collaborative troubleshooting and knowledge sharing.

**4.** Describe the purpose of the lib directory in a Flutter project.

**PURPOSE OF LIB DIRECTORY IN FLUTTER PROJECT:**

1. **Source Code Storage:** The **lib** directory holds all Dart source code files (.dart) for the project.
2. **Entry Point:** The **main.dart** file within **lib** is the starting point of flutter app, containing the **main()** function.
3. **Organization:** Can organize code into subdirectories within **lib** for better structure and management.
4. **Accessibility:** Code within **lib** can be accessed from any part of the project, making it easy to import and use classes and functions.
5. **Package Structure:** For Flutter packages, the **lib** directory follows conventions and holds the package's source code.

**5,How can you run a Flutter app on an emulator and a physical device?**

Running a Flu er app on an emulator or a physical device is pre y straigh orward:

**On an Emulator:**

* + Open your preferred emulator (like Android Studio's AVD Manager).
  + Make sure the emulator is fully loaded and ready.
  + In your terminal or command prompt, navigate to your Flu er project directory.
  + Type flu er run and hit Enter. This command will build your app and launch it on the emulator.

**On a Physical Device:**

* + Connect your device to your computer using a USB cable.
  + Make sure USB debugging is enabled on your device.
  + On your device, allow USB debugging if prompted.
  + In your terminal or command prompt, navigate to your Flu er project directory.
  + Type flu er devices to see if your device is recognized.
  + If your device is listed, type flu er run -d <device-id> and hit Enter, replacing <device-id> with your device's ID from the previous step.
  + This command will build your app and install it on your device.