

Experiment 1

Experiment No 1

1: Installation and Configuration of Flutter Environment.

ROLL NO	32
NAME	Priyank Kulkarni
CLASS	D15-B
SUBJECT	MAD & PWA Lab
LO-MAPPED	

Introduction:

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications from a single codebase for any web browser, Fuchsia, Android, iOS, Linux, macOS, and Windows. First described in 2015, Flutter was released in May 2017.

The major components of Flutter include:

- Dart platform
- Flutter engine
- Foundation library
- Design-specific widgets
- Flutter Development Tools (DevTools)

Dart language

Flutter apps are written in the Dart language and make use of many of the language's more advanced features. For better performance, release versions of Flutter apps on all platforms use ahead-of-time (AOT) compilation, except for on the Web, where code is transpiled to JavaScript.

Flutter inherits Dart's Pub package manager and software repository, which allows users to publish and use custom packages as well as Flutter-specific plugins.

Flutter engine

Flutter's engine, written primarily in C++, provides low-level rendering support using either Google's Skia graphics library or the custom "Impeller" graphics layer. Additionally, it interfaces with platform-specific SDKs such as those provided by Android and iOS to implement accessibility, file and network I/O, native plugin support, and more.

Foundation library

The Foundation library, written in Dart, provides basic classes and functions that are used to construct applications using Flutter, such as APIs to communicate with the engine.

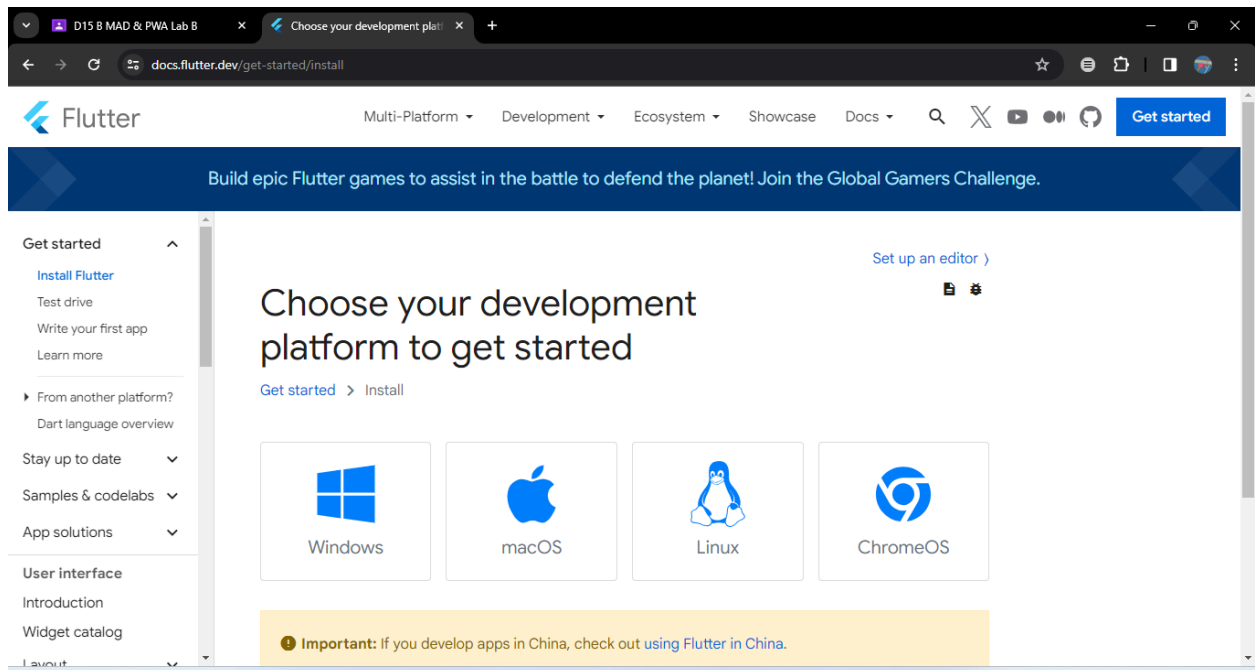
Design-specific widgets

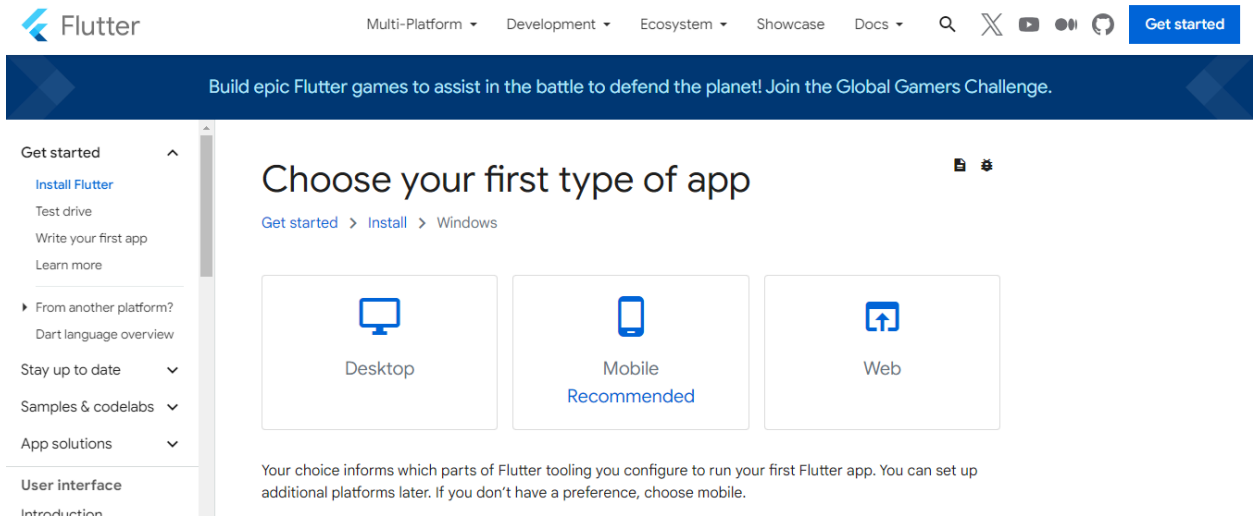
The Flutter framework contains two sets of widgets that conform to specific design languages: Material Design widgets implement Google's design language of the same

name, and *Cupertino* widgets implement Apple's iOS Human interface guidelines. Flutter allows the developer to use either set of widgets on either platform, i.e. even Cupertino widgets on Android. Third party packages can be used to automatically adjust the app's design to the current operating system.

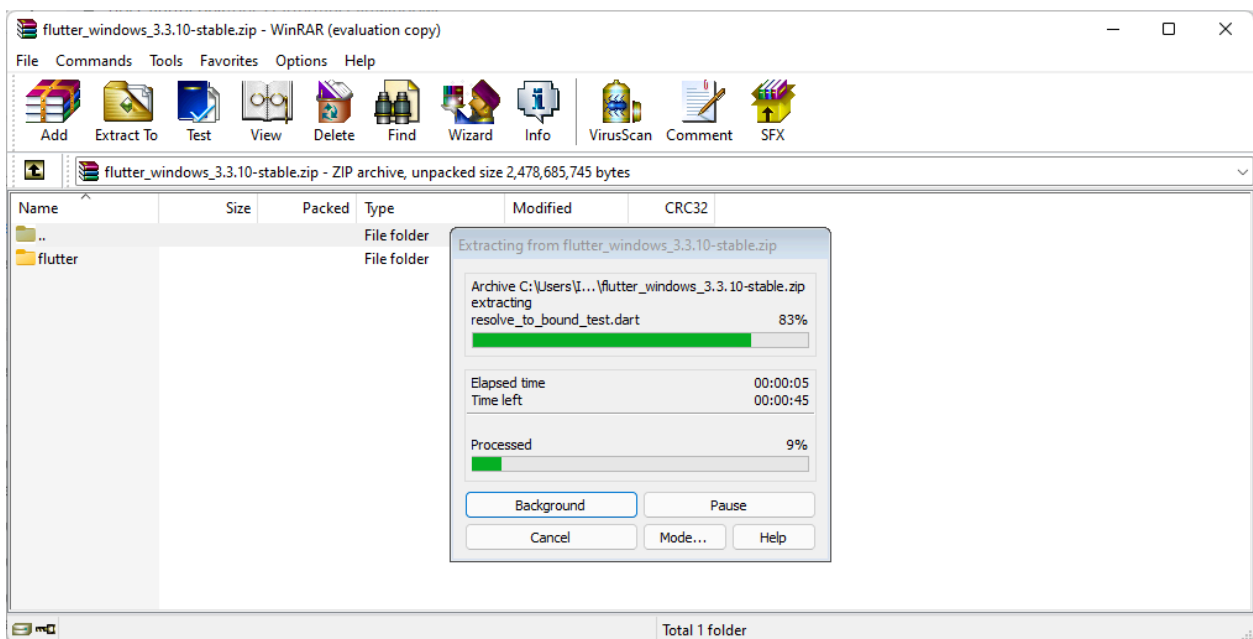
Flutter Installation:

Step 1: Install the Flutter SDK Download the Flutter Software Development Kit from the official website for Windows.





Step 2: Once the download is complete extract the zip file and place it in the desired folder.



Step 3: To run the Flutter command in the regular windows console, you need to update the system path to include the flutter bin directory. The following Steps are required to do this: Go to THIS PC -> Properties -> Advanced system settings -> Environment variables.

Step 4: Select the Path -> click on Edit and add the path to the Flutter bin folder.

Step 5: Run the flutter command in the Command Prompt.

```
Command Prompt - flutter
C:\Users\ASUS>flutter

A new version of Flutter is available!
To update to the latest version, run "flutter upgrade".

Manage your Flutter app development.

Common commands:

  flutter create <output directory>
    Create a new Flutter project in the specified directory.

  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.

Usage: flutter <command> [arguments]

Global options:
-h, --help                Print this usage information.
-v, --verbose             Verbose logging, including all shell commands executed.
-d, --device-id           Target device id or name (prefixes allowed).
--version                Reports the version of this tool.
--suppress-analytics      Suppress analytics reporting for the current CLI invocation.
--disable-telemetry       Disable telemetry reporting when this command runs.

Available commands:

Flutter SDK
  bash-completion         Output command line shell completion setup scripts.
  channel                 List or switch Flutter channels.
  config                  Configure Flutter settings.
  doctor                  Show information about the installed tooling.
  downgrade               Downgrade Flutter to the last active version for the current channel.
  precache                Repopulate the Flutter tool's cache of binary artifacts.
  upgrade                Upgrade your copy of Flutter.

Project
  analyze                 Analyze the project's Dart code.
  assemble                Assemble and build Flutter resources.
  build                   Build an executable app or install bundle.
  clean                   Delete the build/ and dart.tool/ directories.
  create                  Create a new Flutter project.
  drive                   Run integration tests for the project on an attached device or emulator.
  gen-l10n                Generate localizations for the current project.
  pub                     Commands for managing Flutter packages.
  run                     Run your Flutter app on an attached device.
  test                    Run Flutter unit tests for the current project.

Tools & Devices
  attach                 Attach to a running app.
  custom-devices          List, reset, add and delete custom devices.
  devices                List all connected devices.
  emulators              List, launch and create emulators.
  install                 Install a Flutter app on an attached device.
  logs                   Show log output for running Flutter apps.
  screenshot             Take a screenshot from a connected device.
  symbolize               Symbolize a stack trace from an AOT-compiled Flutter app.

Run "flutter help <command>" for more information about a command.
Run "flutter help -v" for verbose help output, including less commonly used options.
```

Also, run the flutter doctor command. This command checks for all the requirements of Flutter app development and displays a report of the status of your Flutter installation.

```
C:\Users\ASUS>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.10.5, on Microsoft Windows [Version 10.0.22621.3007], locale en-US)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 33.0.2)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop for Windows (Visual Studio Community 2022 17.5.0)
[✓] Android Studio (version 2022.1)
[✓] VS Code (version 1.85.1)
[✓] Connected device (3 available)
[✓] Network resources

• No issues found!

C:\Users\ASUS>
```

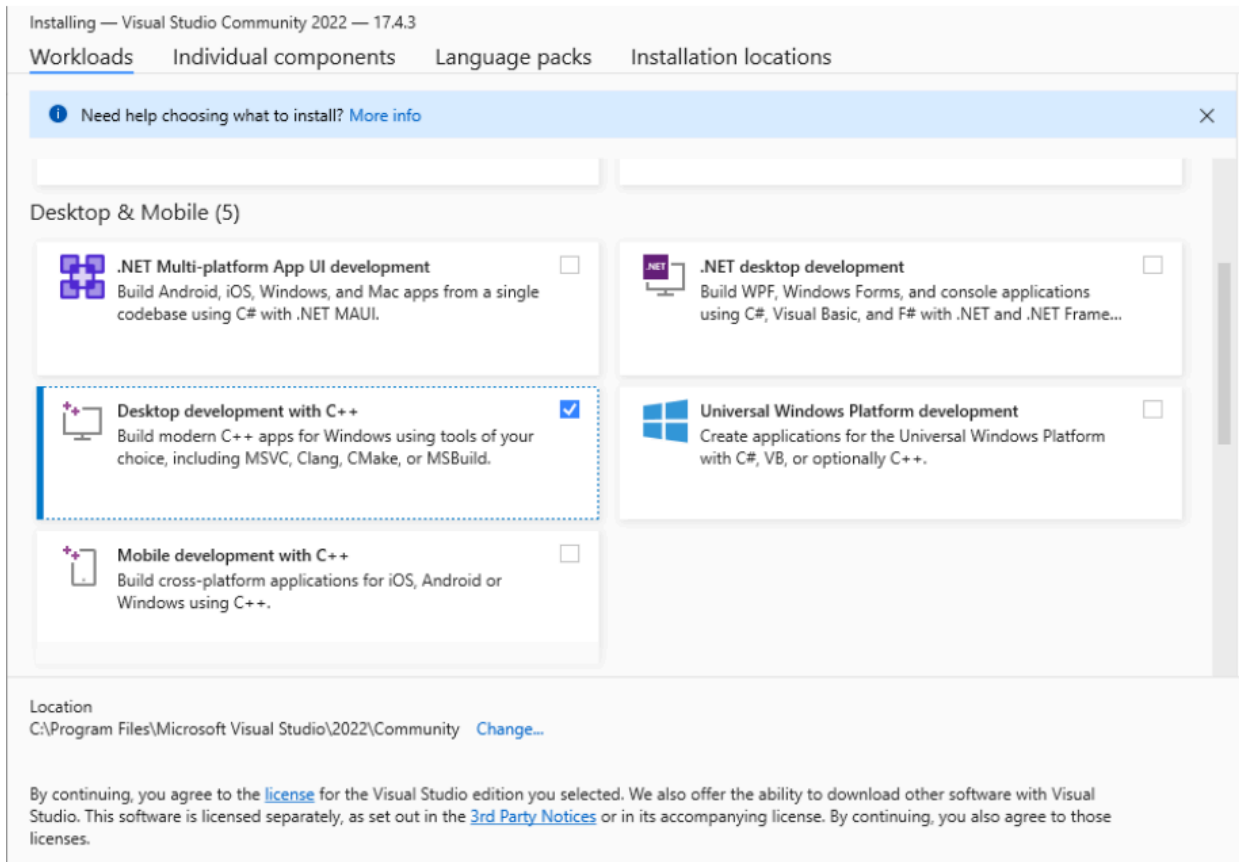
Step 6: Now we will resolve all the issues and install and add the missing tools required for Flutter app development.

Step 6.1: Download and install Visual Studio.

The screenshot shows the Visual Studio Downloads page. The browser's address bar displays 'visualstudio.microsoft.com/downloads/'. The page features a dark navigation bar with the Microsoft logo, 'Visual Studio', and links for 'Developer Tools', 'Downloads', 'Buy', 'Subscriptions', and a 'Free Visual Studio' button. The main content area is titled 'Downloads' and includes a purple banner with the Visual Studio logo and the text 'Visual Studio 2022 |'. Below this, there are five columns: 'Visual Studio 2022 |' (describing it as the most comprehensive IDE for .NET and C++ developers), 'Community' (Powerful IDE, free for students, open-source contributors, and individuals), 'Professional' (Professional IDE best suited to small teams), 'Enterprise' (Scalable, end-to-end solution for teams of any size), and 'Preview' (Get early access to latest features not yet in the main release). Each column has a 'Free download' or 'Free trial' button. At the bottom of the banner, there are links for 'Release notes', 'Compare Editions', and 'How to install offline'. A 'Feedback' button is located on the right side of the banner. The taskbar at the bottom shows 'VisualStudioSetup.exe' running.

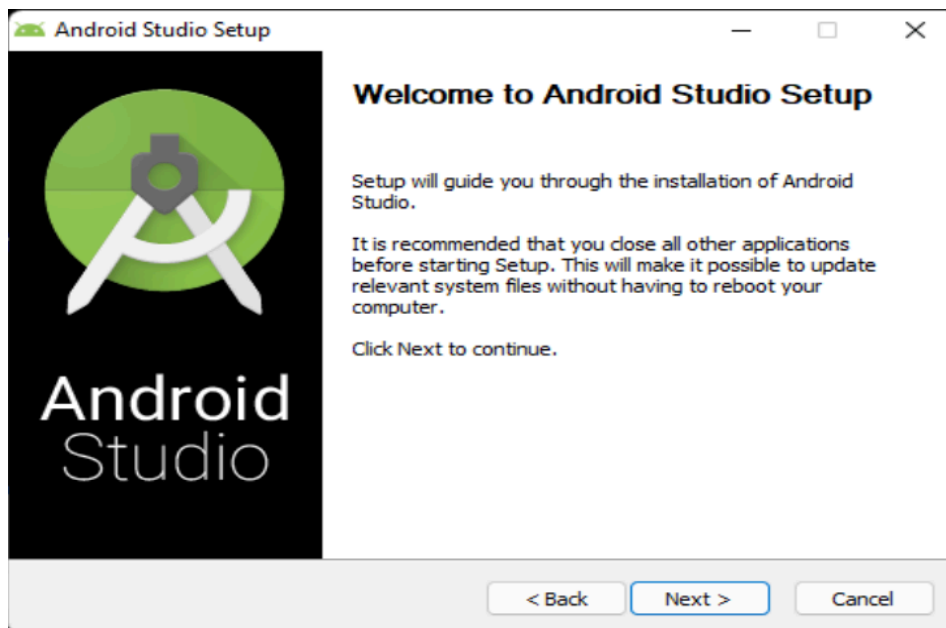
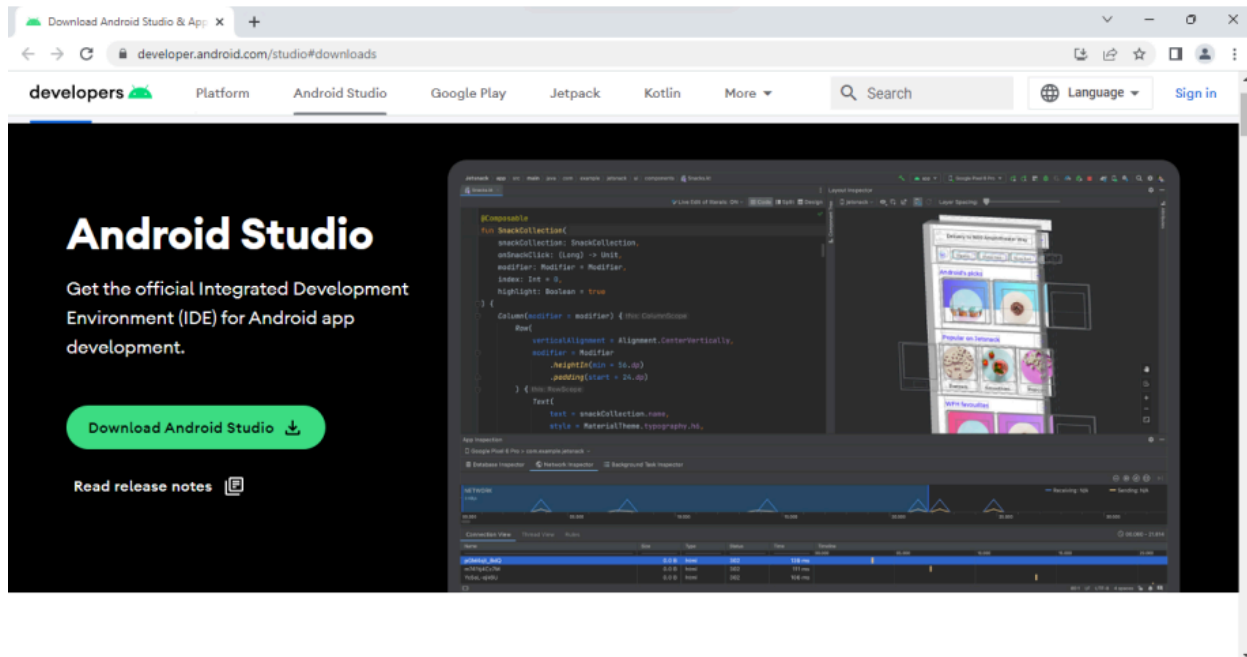
The screenshot shows the Visual Studio Installer window. The title bar reads 'Visual Studio Installer'. The main text says 'Getting the Visual Studio Installer ready.' Below this, there are two progress bars. The first bar is labeled 'Downloading: 15.86 MB of 16.7 MB' and '4.04 MB/sec'. The second bar is labeled 'Installing'. A 'Cancel' button is located at the bottom right of the window.

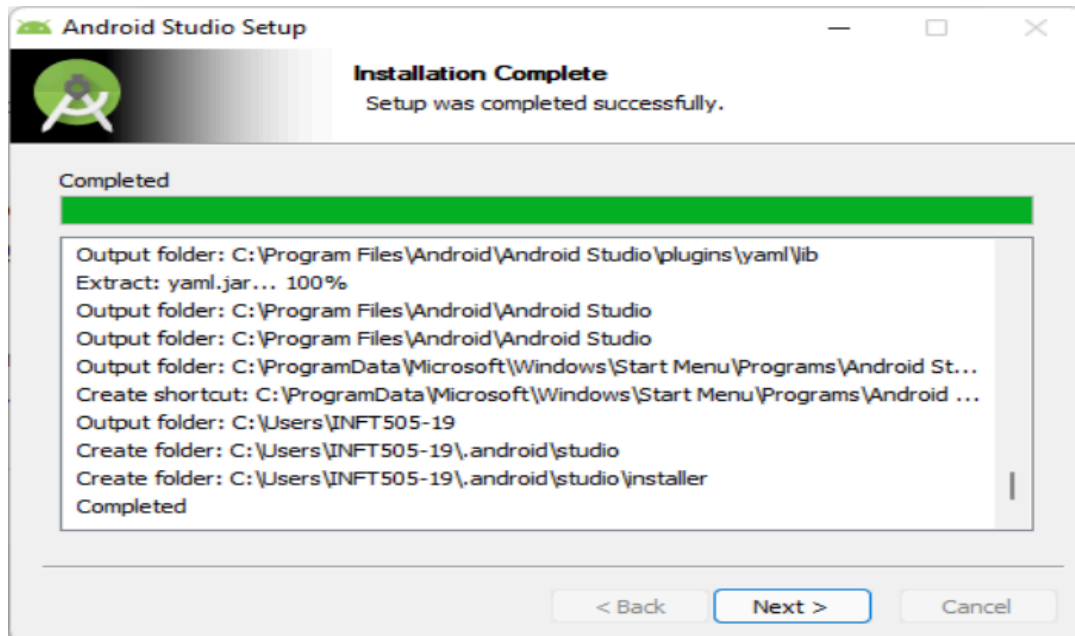
While Installing Visual Studio, make sure to select Desktop development with C++.



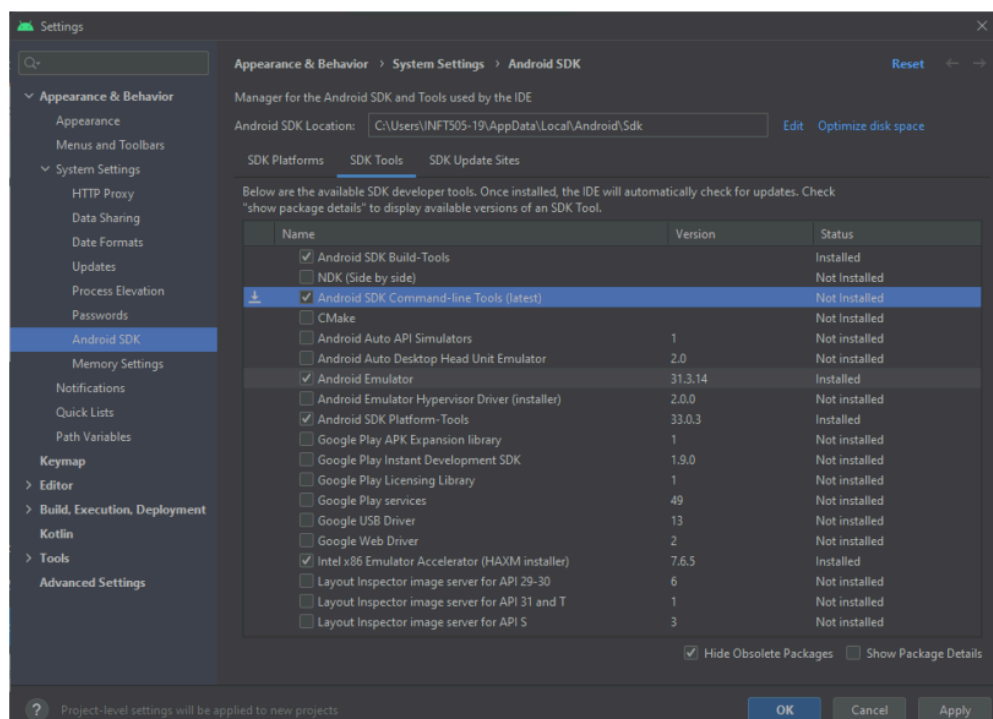
We can see that the Visual Studio issue has been solved.

Step 6.2: Install the Android SDK. If the flutter doctor command does not find the Android SDK tool in your system, then you need first to install the Android Studio IDE.



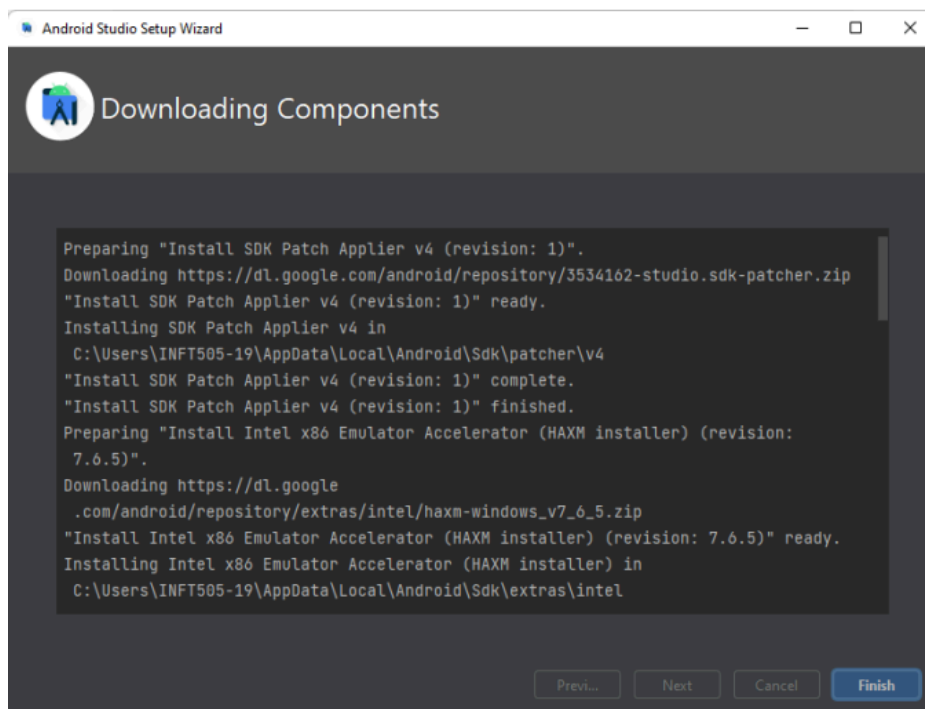
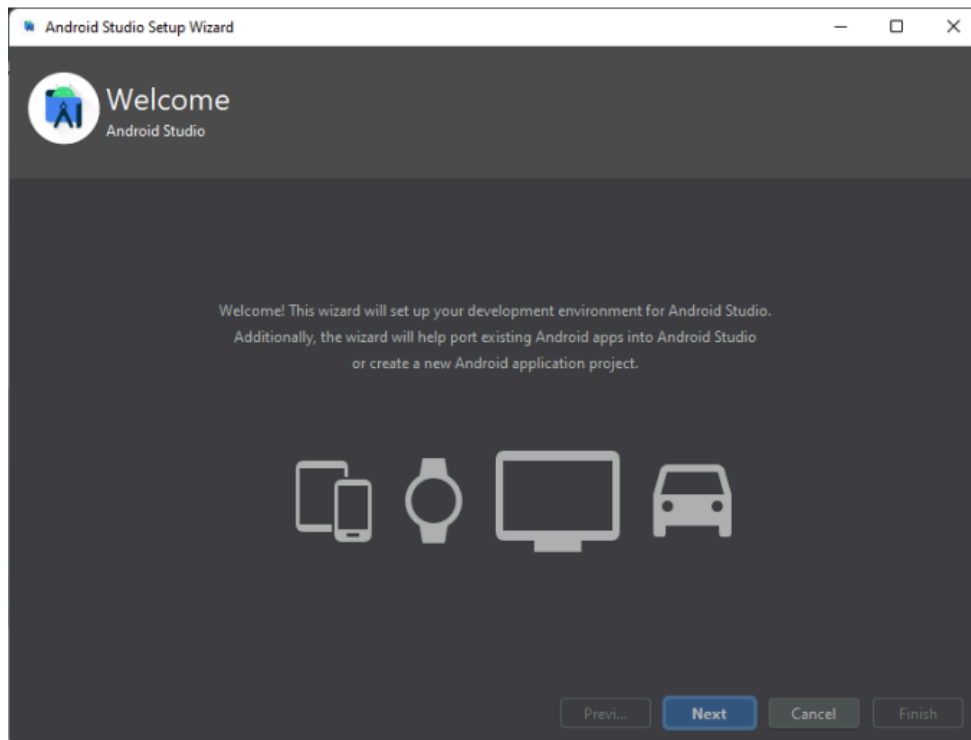


Once the android Studio is installed, Download the Android SDK Command-Line Tools present in the Android SDK section in Android Studio.

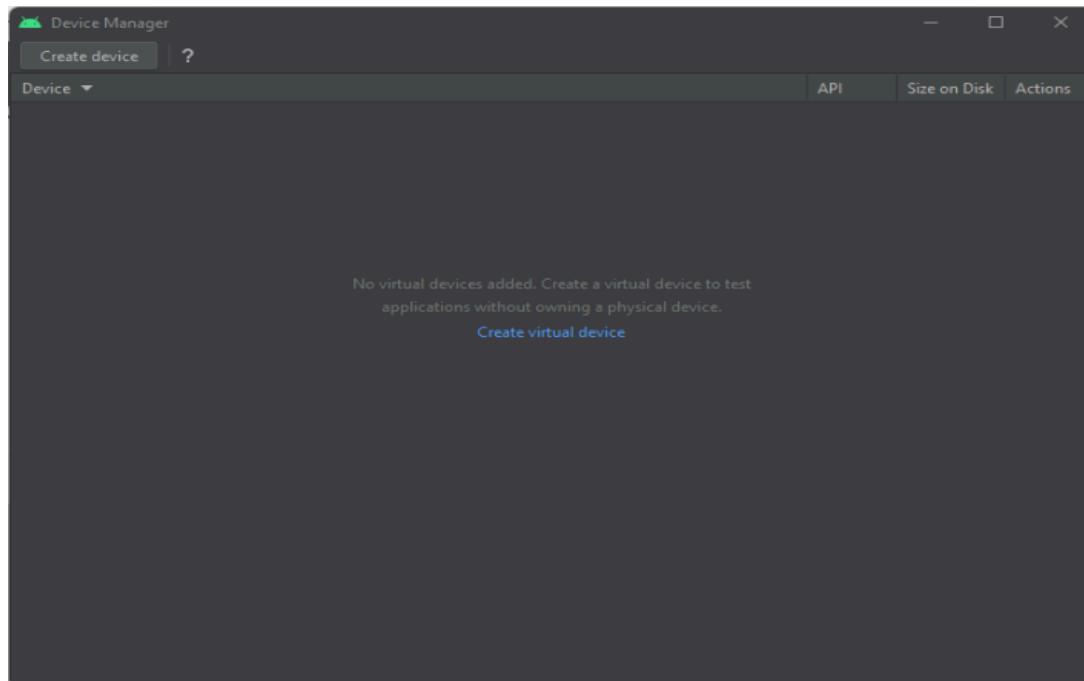


Run the flutter doctor command and run flutter doctor –android licenses to accept all the required licenses.

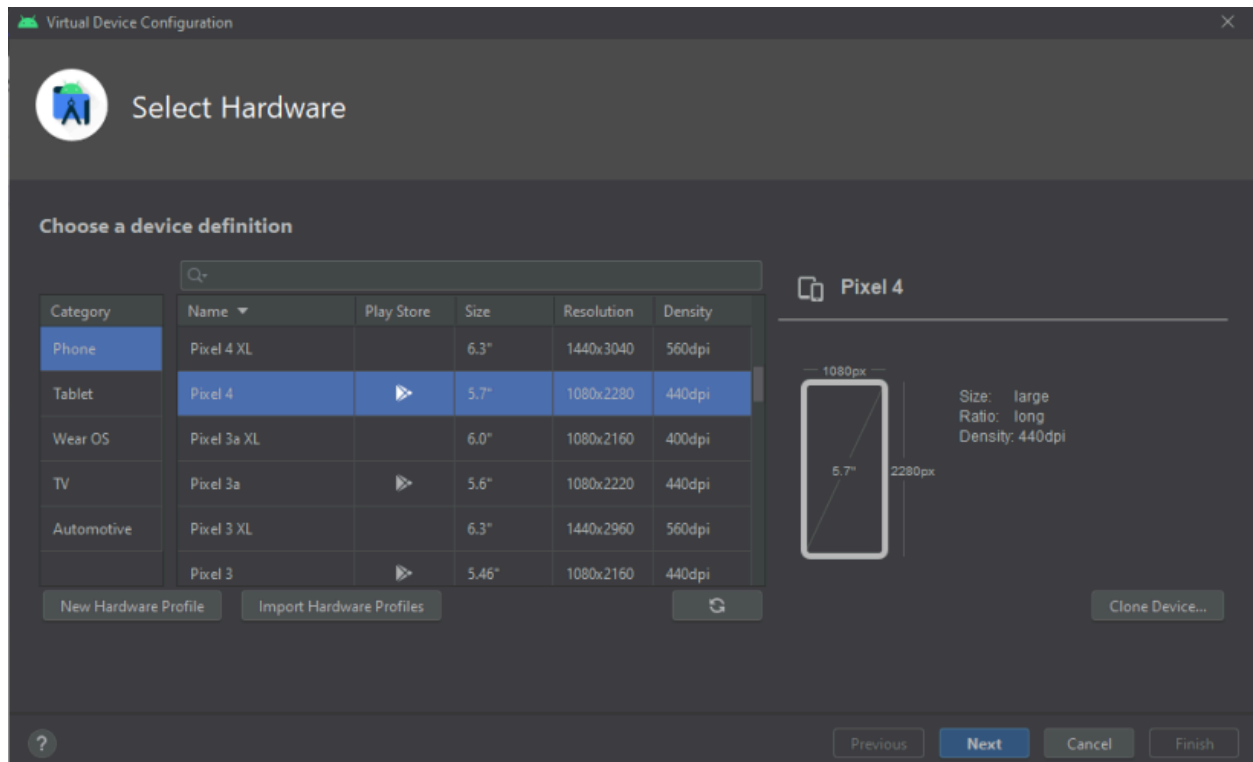
Step 7: Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.



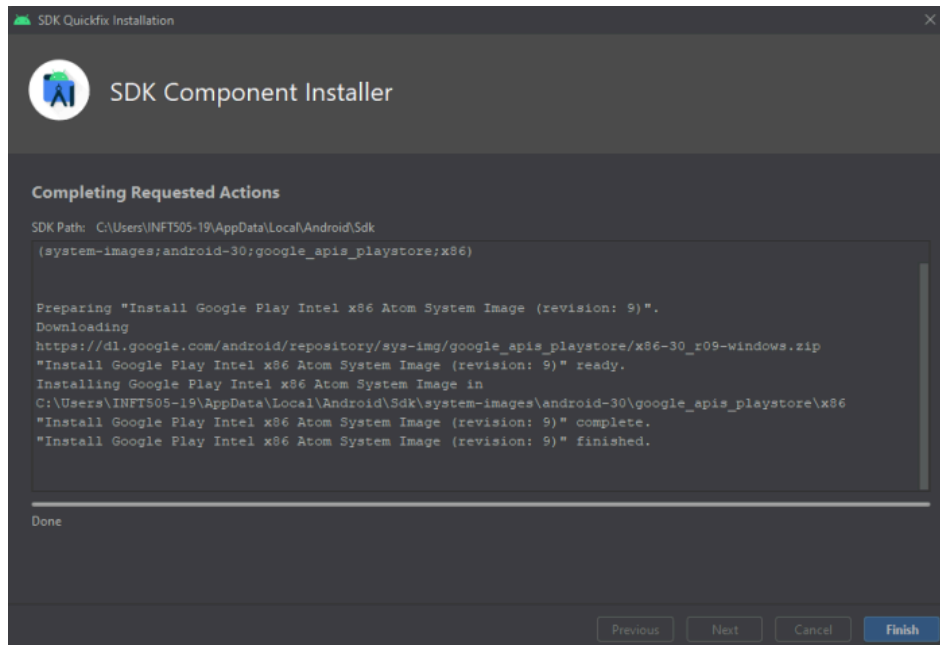
Step 7.1: To set an Android emulator, go to Android Studio > Tools > SDK Manager and select Create Virtual Device. You will get the following screen:



Step 7.2: Choose your device definition and click on Next.



Step 7.3: Select and download the latest operating system for our Emulator and click on Finish.

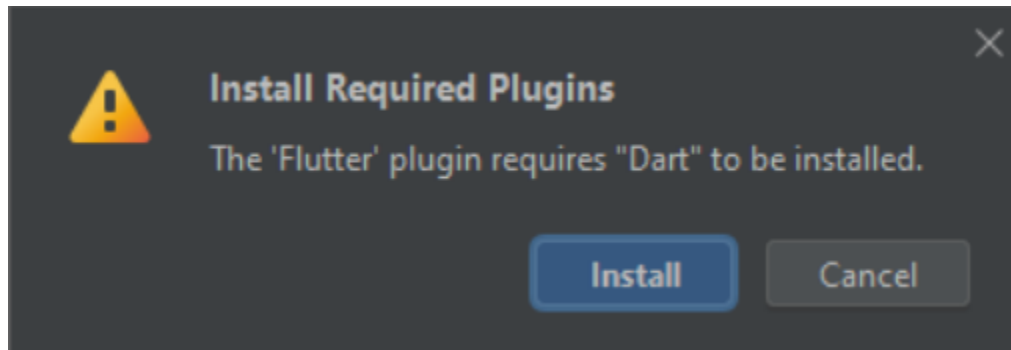


Step 7.4: Click on the Run button and the following screen will be displayed.



Step 8: Now, install Flutter and Dart plugins for building the Flutter application in Android Studio. These plugins provide a template to create a Flutter application and give the option to run and debug the Flutter application in the Android Studio itself.

Open the Android Studio and then go to File->Settings->Plugins. Now, search the Flutter plugin. If found, select the Flutter plugin and click install. When you click on install, it will ask you to install Dart plugin as below screen. Click Install to proceed.



Finally when all these Steps are followed restart the Android Studio once and then your Flutter environment is successfully configured.

Conclusion:

In conclusion, the installation and configuration of Flutter and Android Studio proved to be a pivotal step in creating a robust mobile development environment. By seamlessly integrating these tools, developers gain a versatile platform for building cross-platform applications with efficiency. The synergy between Flutter's UI toolkit and Android Studio's comprehensive development features opens up a world of possibilities, empowering developers to create visually appealing and high-performance apps for both Android and iOS. As technology continues to evolve, mastering the setup of these tools becomes increasingly essential for staying at the forefront of mobile app development.