

## EXPERIMENT NO.1

**Experiment No 1**

**1: Installation and Configuration of Flutter Environment.**

<b>ROLL NO</b>	<b>28</b>
<b>NAME</b>	<b>Anushka Karhadkar</b>
<b>CLASS</b>	<b>D15-B</b>
<b>SUBJECT</b>	<b>MAD &amp; PWA Lab</b>
<b>LO-MAPPE D</b>	

## **Experiment No 1: Installation and Configuration of Flutter Environment.**

**Aim:** To install and configure Flutter

### **Theory:**

In the last few years of this decade, we have seen a lot of app startups emerging from all across the globe. With the rise in technology and the availability of smartphones, many startups find it easy to connect with users and clients via apps. The app market has also grown in the last few years and is expected to grow exponentially in the coming decade. The app development market has also been on a rise and has allowed countless app developers to exhibit their skills and find a suitable job. With this shift into apps, much development, and research have been done to deliver the best and to make the app development process faster and much simpler.

Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.

### Features of flutter

Flutter structure offers the accompanying elements to designers:

- Present day and receptive structure.
- Utilizes Dart programming language, and it is extremely simple to learn.
- Delightful and liquid UIs.
- Colossal gadget list.
- Runs the same UI for numerous stages.
- Superior execution application
- Fast and responsive layout.
- Easy connection of back-end and asynchronization.

### Advantages

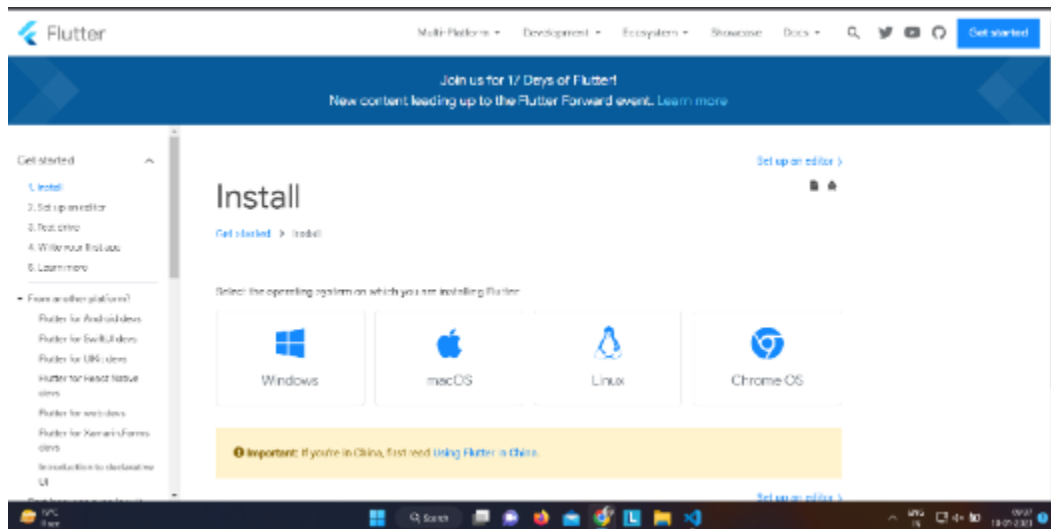
- 1. Cross-platform Operations: Apps made with flutter can be operated on both the platform (iOS and Android). There is no need for reconfiguration and redesigning.
- 2. Less Need of Developers: This can be advantageous for the companies, as they require a smaller number of developers and the app can also work on both the platforms.
- 3. Less Development Cost: Since there are a smaller number of developers needed, the cost incurred for the development of the app also reduces.

## Installing the Flutter SDK

Step 1: Download the installation bundle of the Flutter Software Development Kit for windows.

To download Flutter SDK, Go to its official website

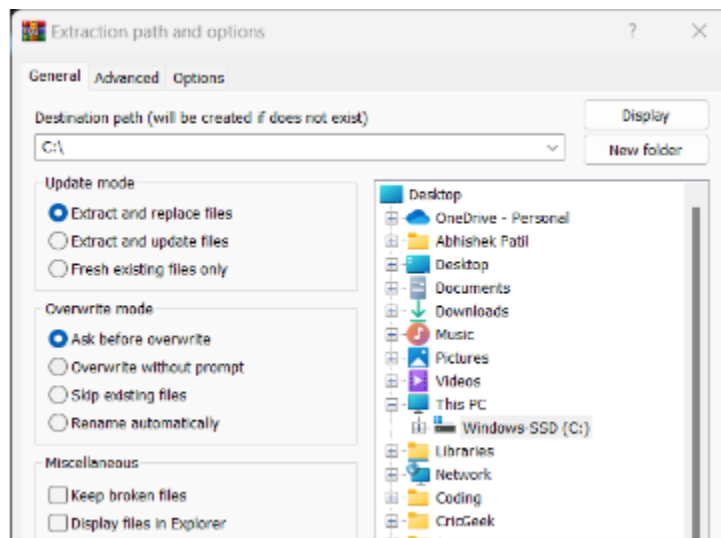
<https://docs.flutter.dev/get-started/install> ,  
you will get the following screen.



Step 2: Next, to download the latest Flutter SDK, click on the Windows icon. Here, you will

find the download link for SDK.

Step 3: When your download is complete, extract the zip file and place it in the desired installation folder or location, for example, C: /Flutter.



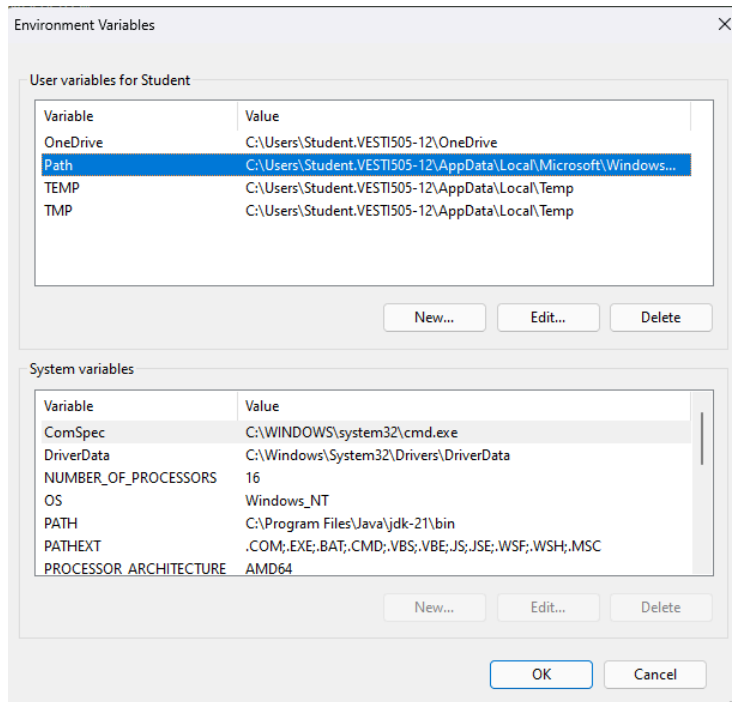
Step 4: To run the Flutter command in regular windows console, you need to update the system

path to include the flutter bin directory. The following steps are required to do this:

Step 4.1: Go to MyComputer properties -&gt; advanced tab -&gt; environment variables.

You will get

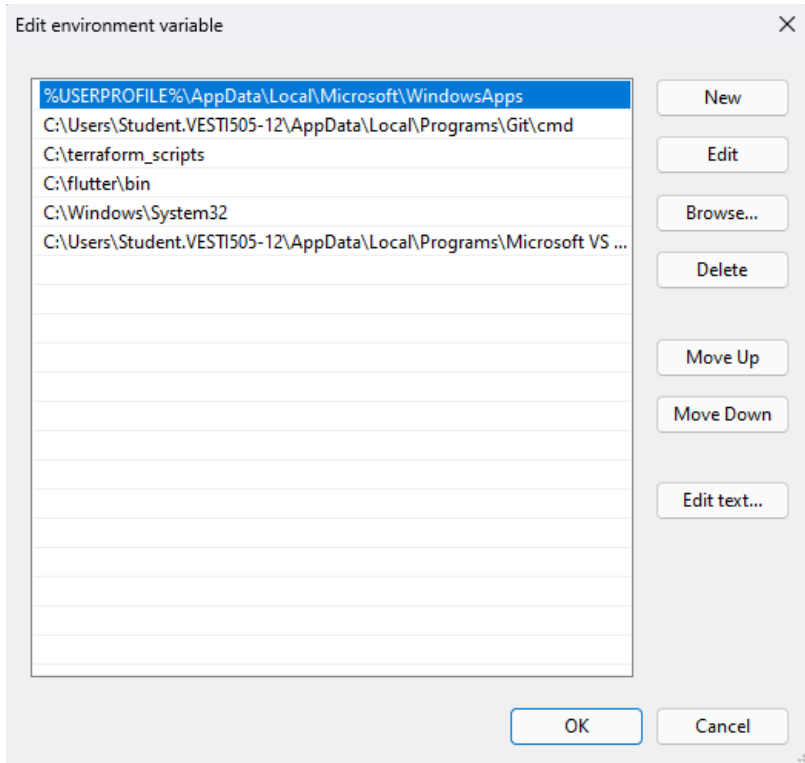
the following screen.



Step 4.2: Now, select path -&gt; click on edit.

Step 4.3: In the above window, click on New-&gt;write path of Flutter bin folder in variable value -

&gt; ok -&gt; ok -&gt; ok.



Step 5: Now, run the \$ flutter command in command prompt.

Now, 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.

```

Command Prompt - flutter  X  +  v
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       Noisy logging, including all shell commands executed.
                    If used with "--help", shows hidden options. If used with "flutter doctor", shows additional
                    diagnostic information. (Use "-vv" to force verbose logging in those cases.)
-d, --device-id     Target device id or name (prefixes allowed).
                    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         Populate 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.

```

Step 6: When you run the above command, it will analyze the system and show its report, as shown in the below image. Here, you will find the details of all missing tools, which required to run Flutter as well as the development tools that are available but not connected with the device.

```
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>
```

```
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              Noisy logging, including all shell commands executed.
                           If used with "--help", shows hidden options. If used with "flutter doctor", shows additional
                           target device id or name (prefixes allowed).
-d, --device-id            Reports the version of this tool.
                           Suppresses analytics reporting for the current CLI invocation.
--suppress-analytics       Suppress analytics reporting when this command runs.
--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                  Prepopulate 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.

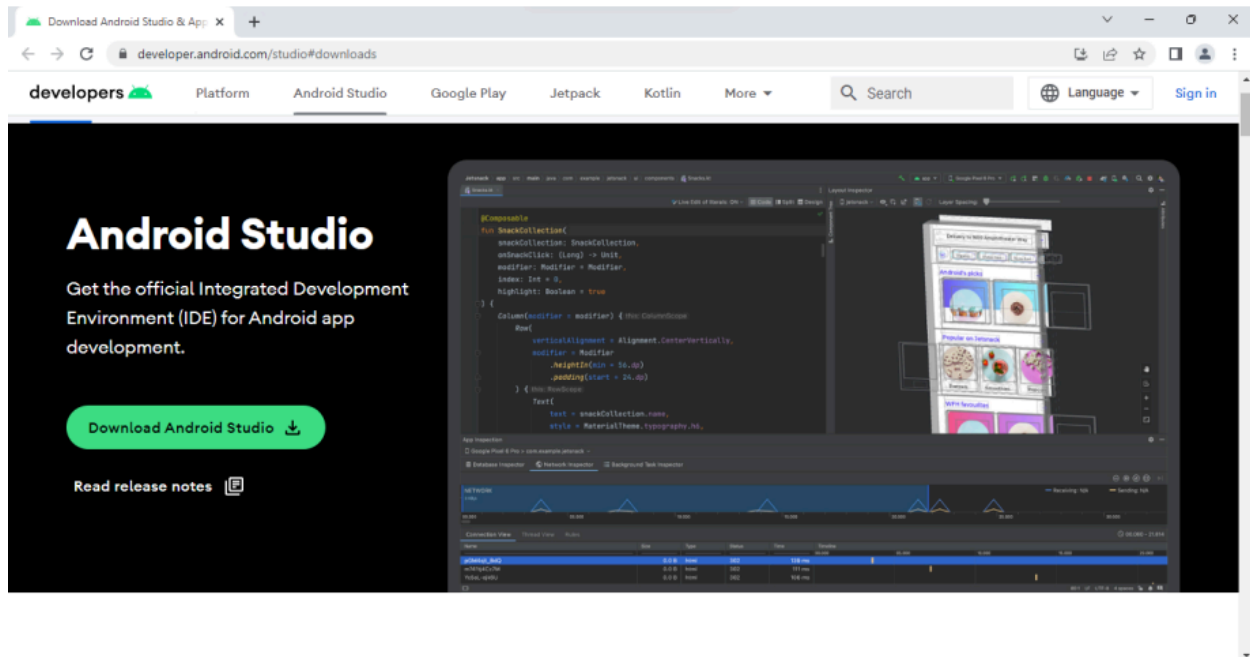
Tools & Devices
attach                    Attach to a running app.
custom-devices             List, reset, add and delete custom devices.
devices                   List all connected devices.
emulators                 List, launch and create emulators.
install                   Install a Flutter app 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.
```

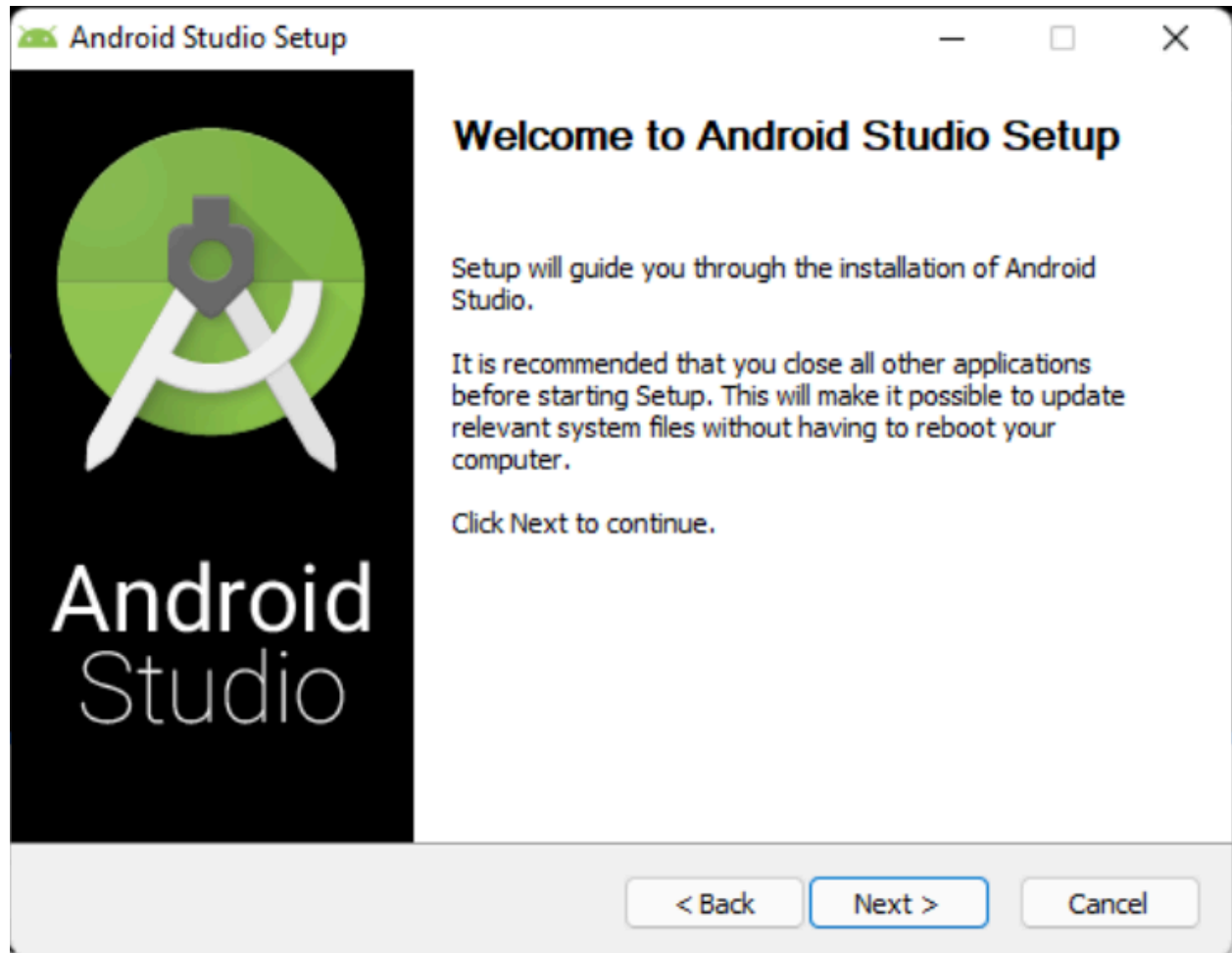
Step 7: Install the Android SDK. If the flutter doctor command does not find the Android SDK

tool in your system, then you need first to install the Android Studio IDE. To install Android Studio IDE, do the following steps.

Step 7.1: Download the latest Android Studio executable or zip file from the official site.

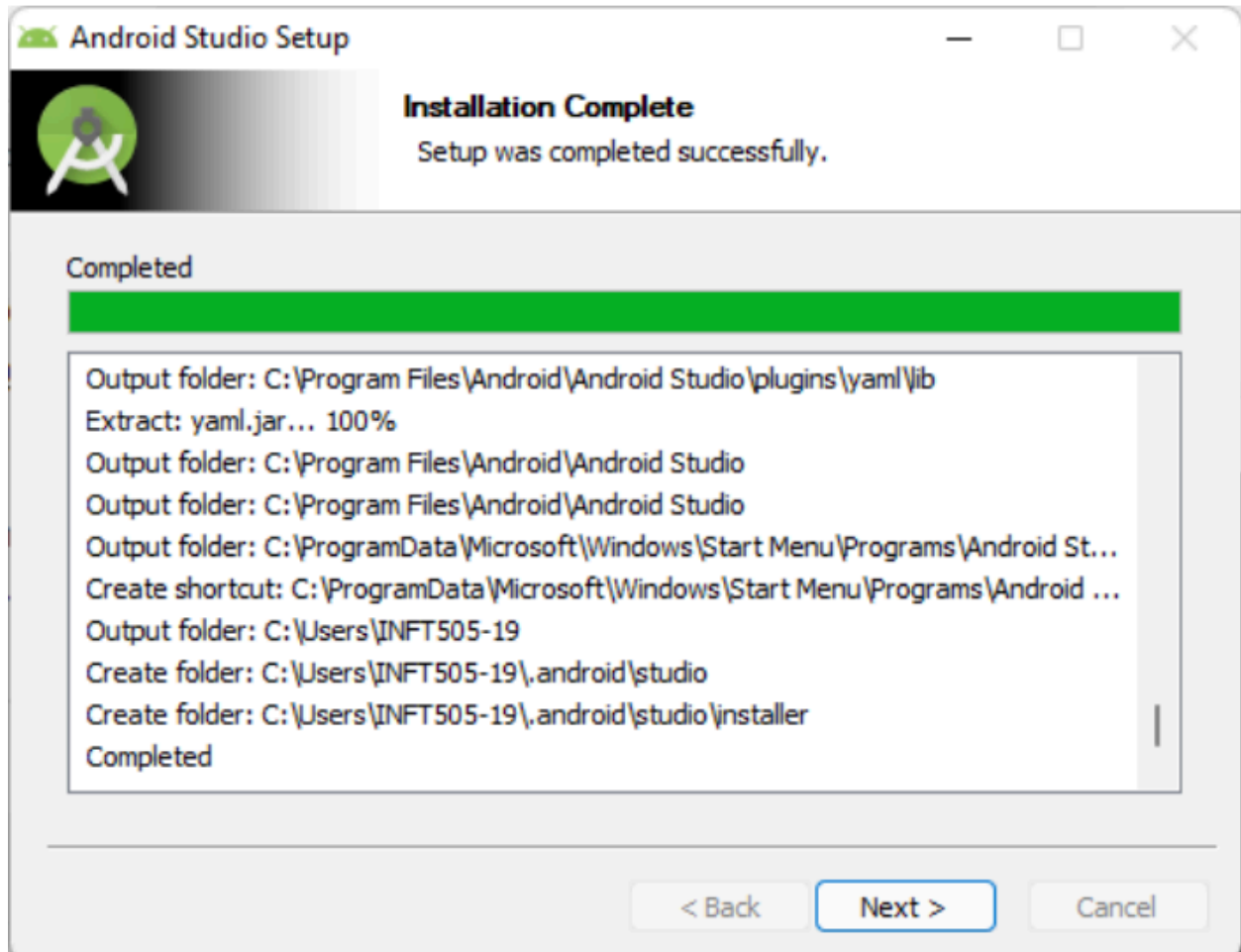


Step 7.2: When the download is complete, open the .exe file and run it. You will get the following dialog box.

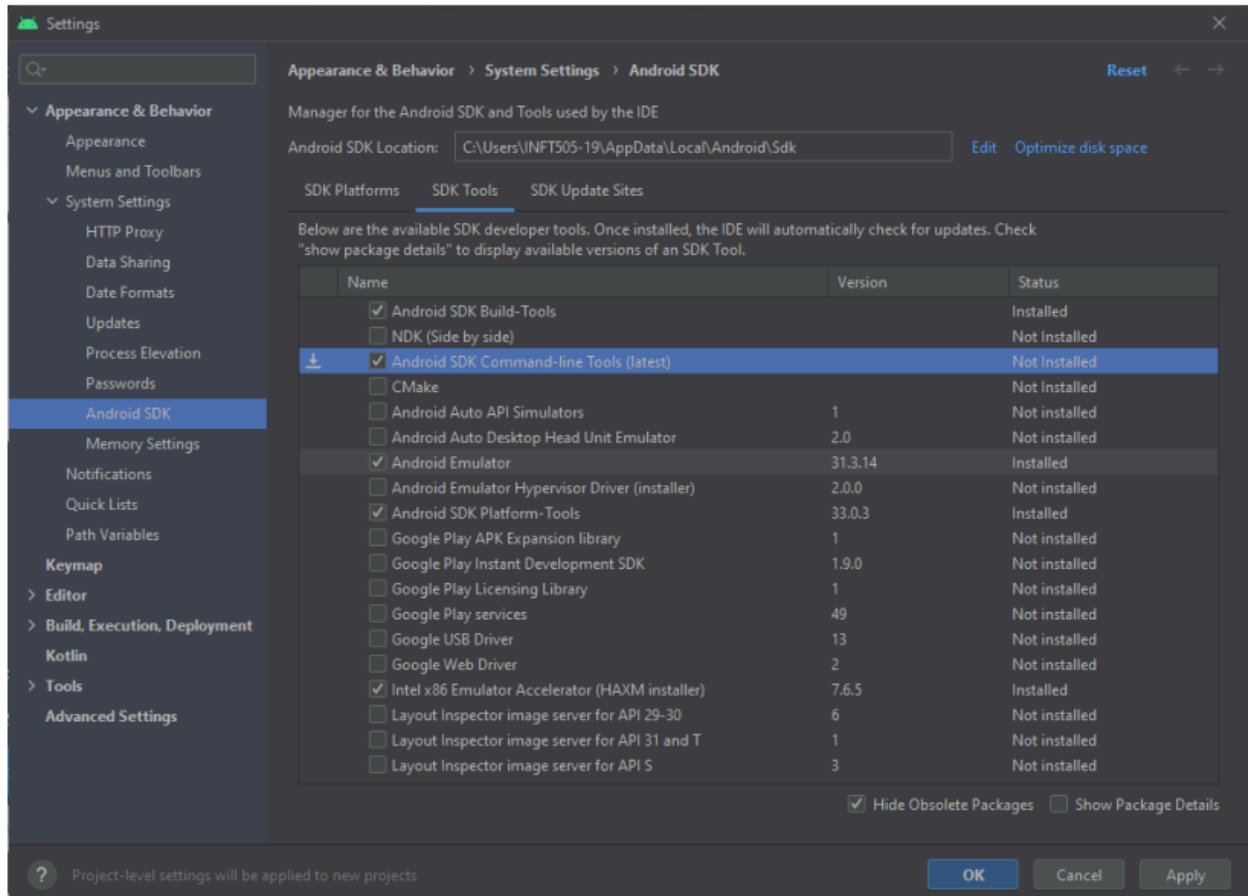


Step 7.3: Follow the steps of the installation wizard. Once the installation wizard completes, you will get the following screen.

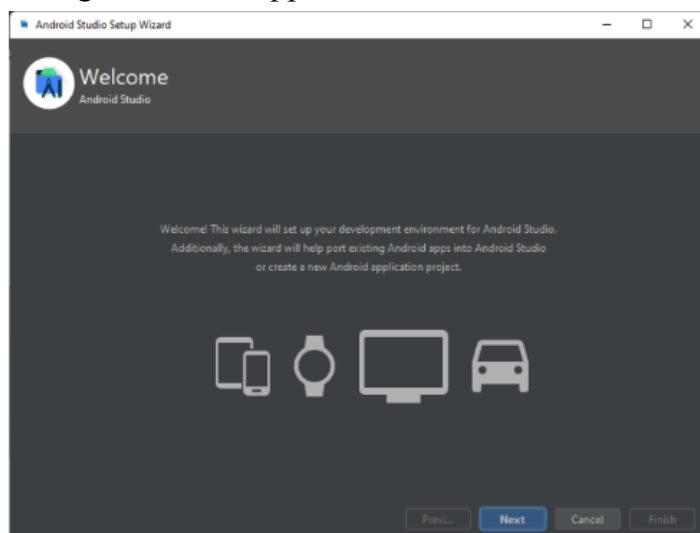


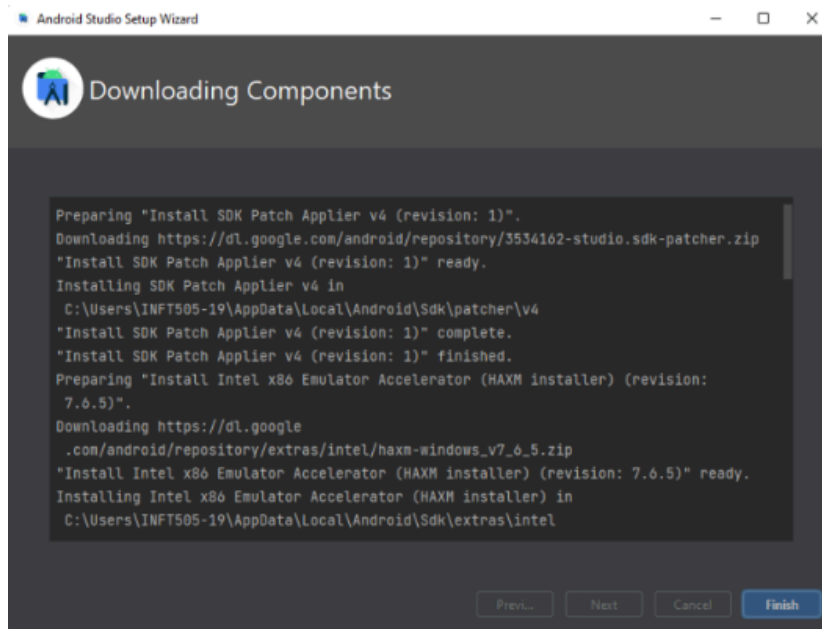


Step 7.4: In the above screen, click Next-> Finish. Once the Finish button is clicked, you need to choose the 'Don't import Settings option' and click OK. It will start the Android Studio.

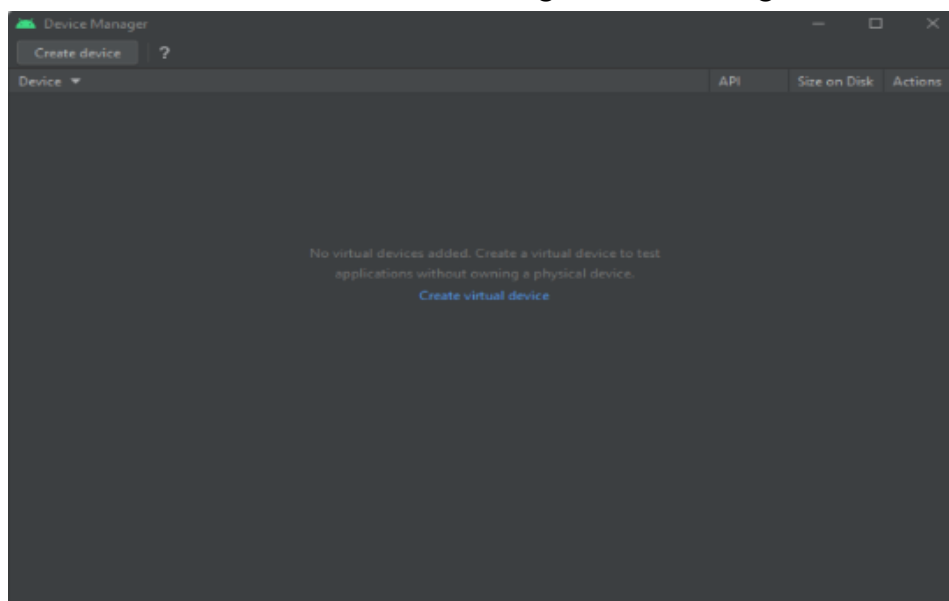


Step 7.5: Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.

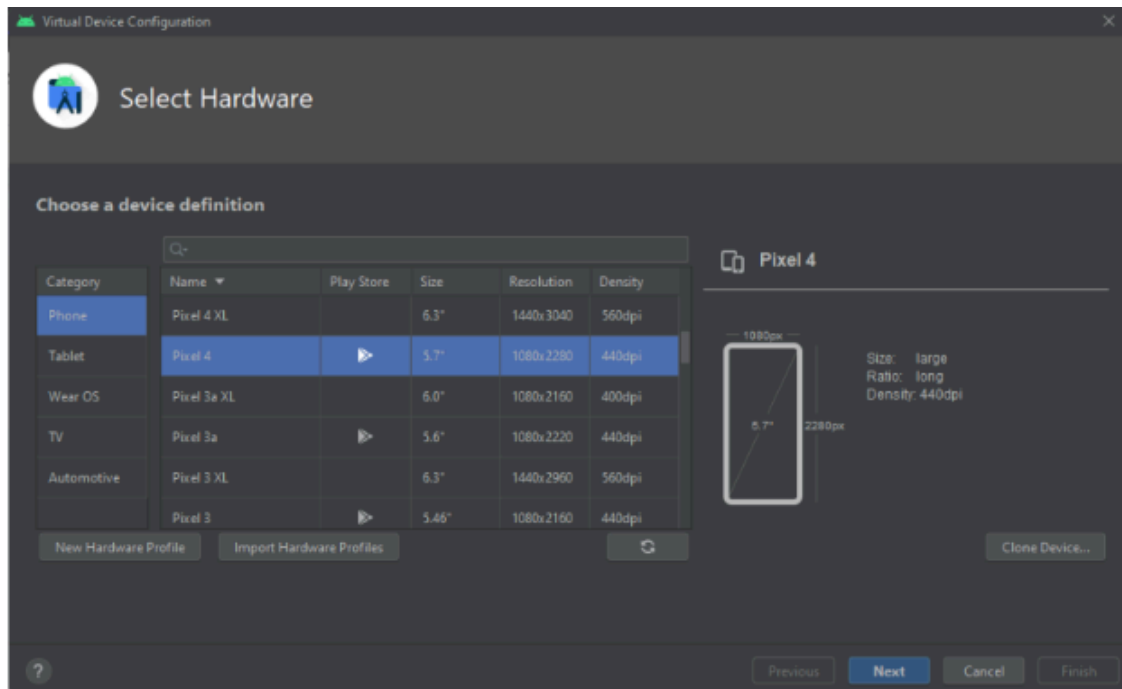




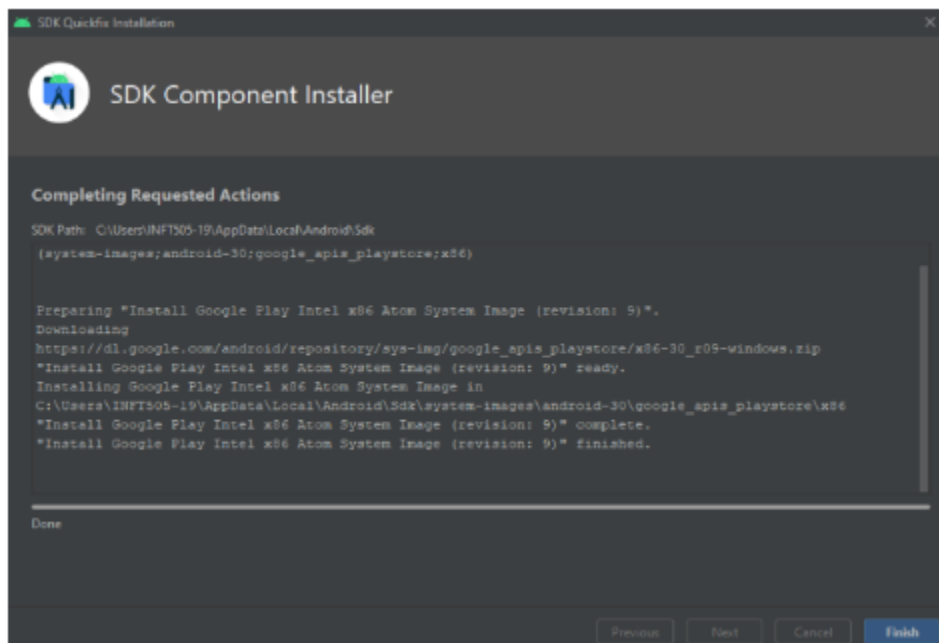
Step 8: To set an Android emulator, go to Android Studio > Tools > SDK Manager and select Create Virtual Device. You will get the following screen:



Step 8.1 : Choose your device definition and click on Next.



Step 8.2: Select and download the latest operating system for our Emulator and click on Finish.

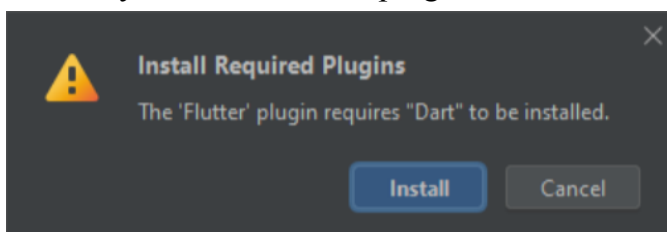


Step 8.3: Click on the Run button and the following screen will be displayed.



Step 8.4: 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:

The installation and configuration of the Flutter environment on a Windows operating system involved the sequential steps of installing the Flutter SDK, using the flutter doctor command for environment validation, and incorporating Android Studio. The latter facilitated development, offering a comprehensive toolkit. Additionally, a virtual device was created within Android Studio for testing and visualizing Flutter applications. This experiment highlights the importance of a systematic approach for a successful Flutter development setup on Windows.