Experiment No. 1

Aim: To install and configure the Flutter Environment

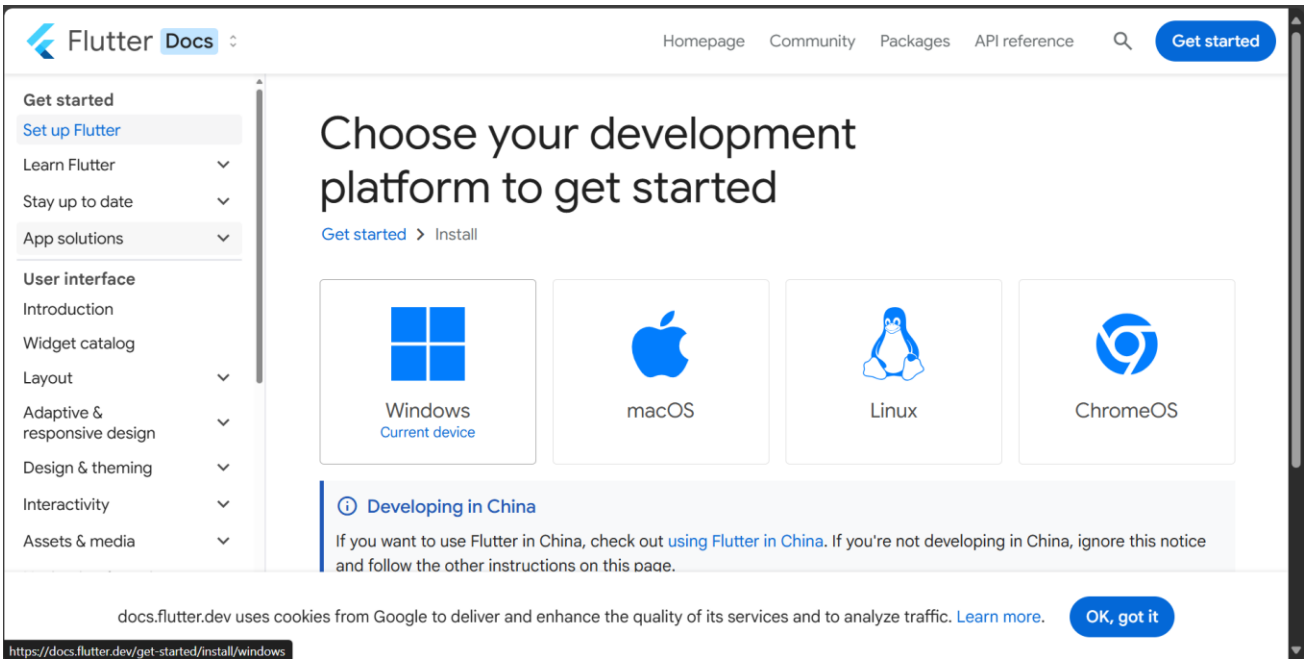
Theory:

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

- Fast: Flutter code compiles to ARM or Intel machine code as well as JavaScript, for fast performance on any device.
- Productive: Build and iterate quickly with Hot Reload. Update code and see changes almost instantly, without losing state.
- Flexible: Control every pixel to create customized, adaptive designs that look and feel great on any screen.

Installation of 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.

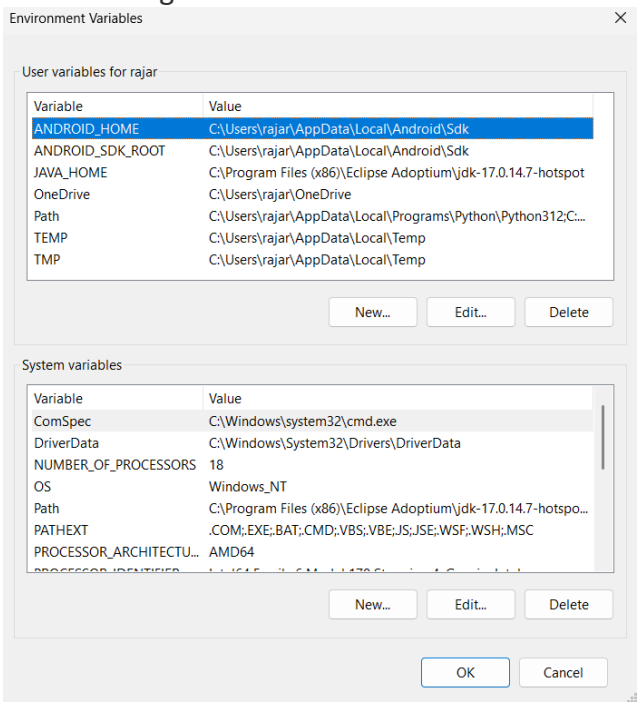


Step 2: Next, to download the latest Flutter SDK, click on the Windows **icon**. Here, you will find the download link for SDK.

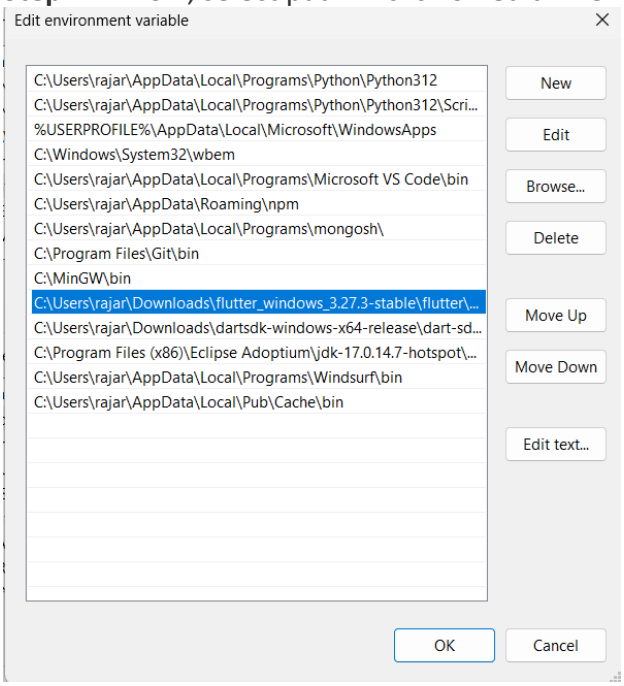
Step 3: When your download is complete, extract the **zip** file and place it in the desired installation folder or location, for example, C: /Flutter.

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 -> advanced tab -> environment variables. You will get the following screen.



Step 4.2: Now, select path -> click on edit. The following screen appears



Step 4.3: In the above window, click on New->write path of Flutter bin folder in variable value -> ok -> ok -> ok.

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

```
C:\Users\rajar>flutter
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).
    --version               Reports the version of this tool.
    --enable-analytics      Enable telemetry reporting each time a flutter or dart command runs.
    --disable-analytics     Disable telemetry reporting each time a flutter or dart command runs, until it is
                            re-enabled.
    --suppress-analytics    Suppress analytics reporting for the current CLI invocation.

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.

C:\Users\rajar>
```

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.

```
C:\Users\rajar>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.27.3, on Microsoft Windows [Version 10.0.26100.3775], locale en-IN)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[!] Android toolchain - develop for Android devices (Android SDK version 35.0.1)
    ! Some Android licenses not accepted. To resolve this, run: flutter doctor --android-licenses
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2024.2)
[√] VS Code (version 1.99.2)
[√] Connected device (3 available)
[√] Network resources

! Doctor found issues in 2 categories.
```

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.

Android Studio:

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

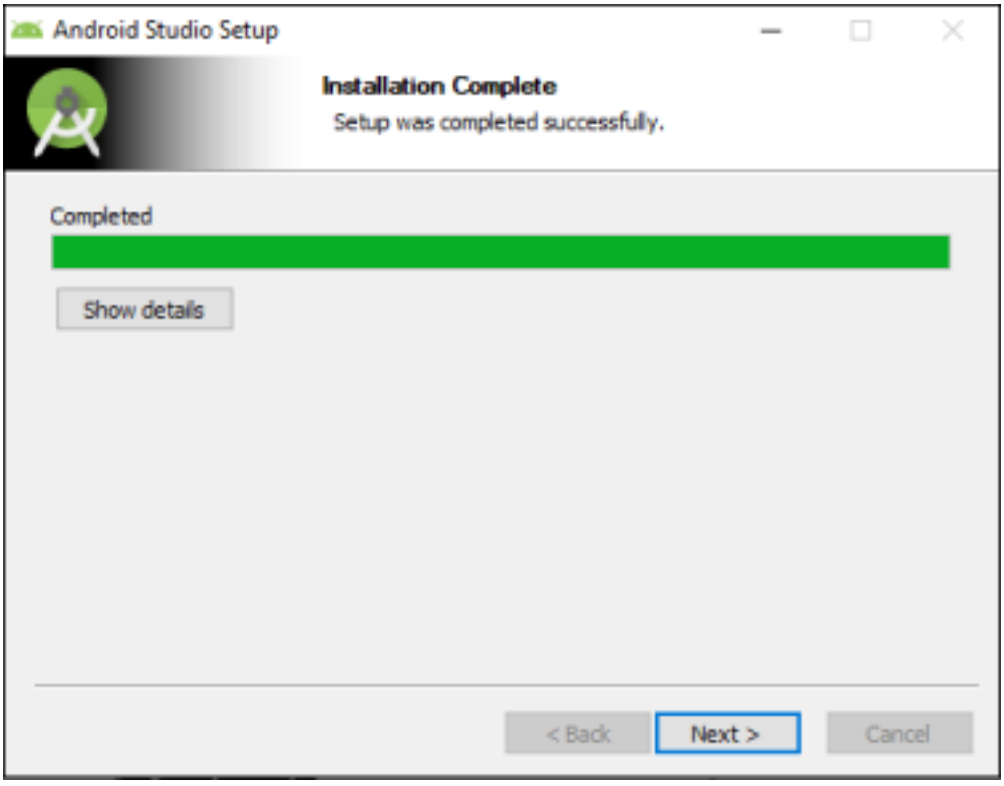
Installation of Android Studio:

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.

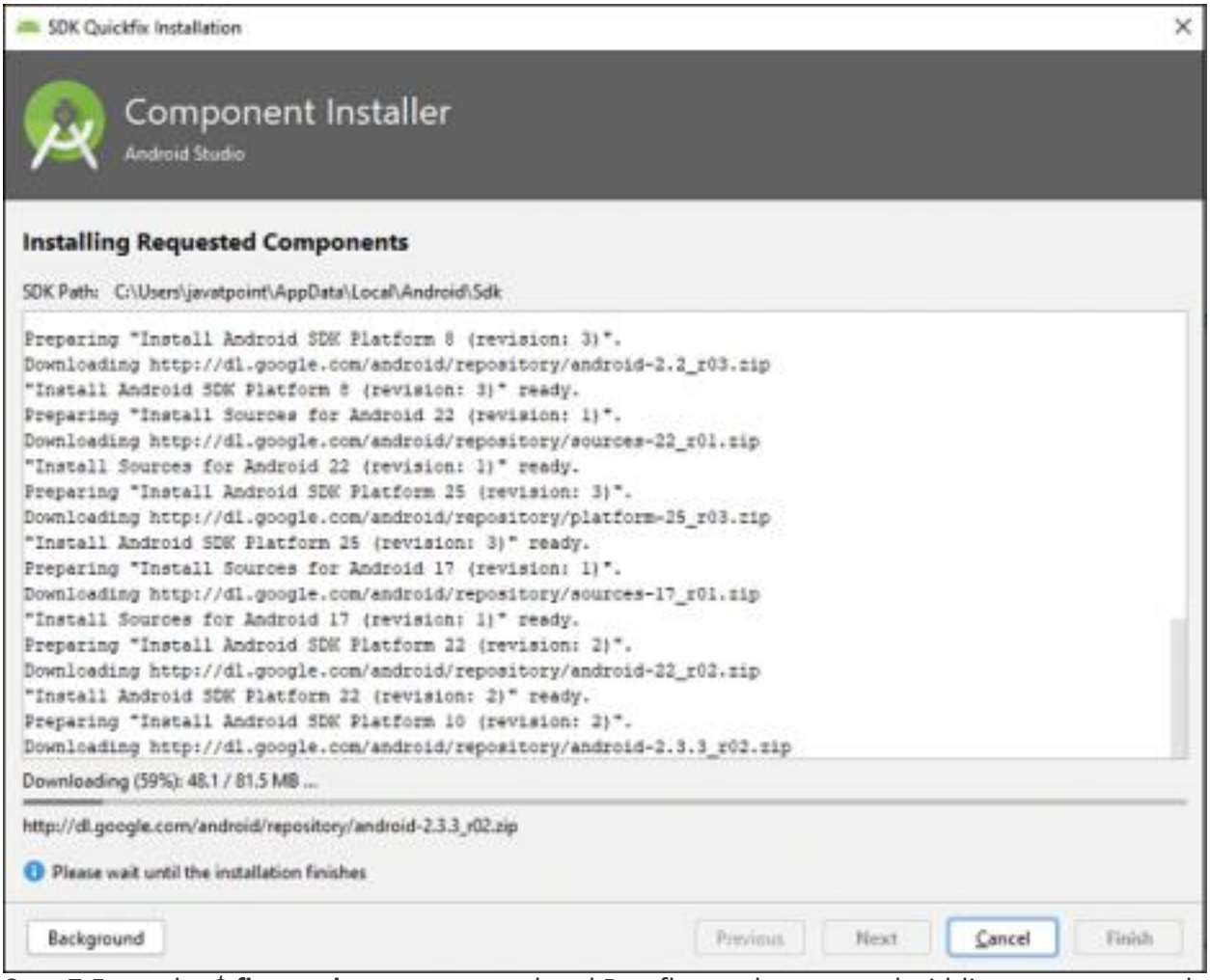
Step 7.2: When the download is complete, open the **.exe** file and run it. You will get the following dialog box.

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 run the $ **flutter doctor** command and Run flutter doctor --android-licenses command.

Step 8: Next, you need to set up an Android emulator. It is responsible for running and testing the Flutter application.

Step 8.1: To set an Android emulator, go to Android Studio > Tools > Android > AVD Manager and select Create Virtual Device. Or, go to Help->Find Action->Type Emulator in the search box. You will get the following screen.

Step 8.2: Choose your device definition and click on Next.

Step 8.3: Select the system image for the latest Android version and click on Next.

Step 8.4: Now, verify the all AVD configuration. If it is correct, click on Finish. The following screen appears.

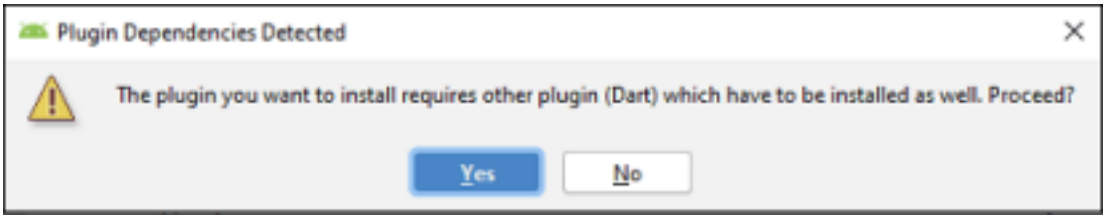
Step 8.5: Last, click on the icon pointed into the red color rectangle. The Android emulator displayed as below screen.

Step 9: Now, install Flutter and Dart plugin for building Flutter application in Android Studio.

These plugins provide a template to create a Flutter application, give an option to run and debug Flutter application in the Android Studio itself. Do the following steps to install these plugins.

Step 9.1: Open the Android Studio and then go to File->Settings->Plugins.

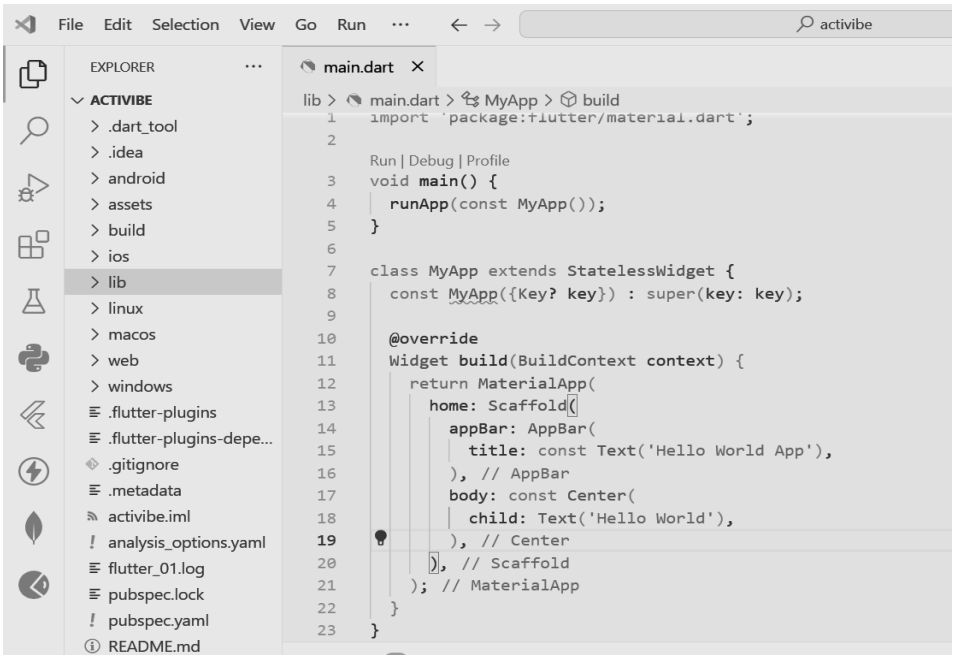
Step 9.2: Now, search the Flutter plugin. If found, select Flutter plugin and click install. When you click on install, it will ask you to install Dart plugin as below screen. Click yes to proceed.



Step 9.3: Restart the Android Studio.

Creation of flutter project:

1. Create a folder with the name of your project.
2. Then open it with vscode.
3. Then click on the open integrated terminal and type the command: "flutter create ."
4. This will create the project with the folder name.
5. Now enter the code in main.dart.



6. Then in command type "flutter run".

```
PS C:\Users\Sadneya\Documents\flutter_projects\activibe> flutter run
Connected devices:
Windows (desktop) • windows • windows-x64    • Microsoft Windows [Version 10.0.22631.4751]
Chrome (web)      • chrome  • web-javascript • Google Chrome 132.0.6834.160
Edge (web)        • edge    • web-javascript • Microsoft Edge 132.0.2957.140
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (or "q" to quit): 2
Launching lib\main.dart on Chrome in debug mode...
Waiting for connection from debug service on Chrome...             20.6s
This app is linked to the debug service: ws://127.0.0.1:64181/PDw-GlZ-sbE=/ws
Debug service listening on ws://127.0.0.1:64181/PDw-GlZ-sbE=/ws

  To hot restart changes while running, press "r" or "R".
For a more detailed help message, press "h". To quit, press "q".

A Dart VM Service on Chrome is available at: http://127.0.0.1:64181/PDw-GlZ-sbE=
The Flutter DevTools debugger and profiler on Chrome is available at: http://127.0.0.1:9101?uri=http://127.0.0.1:64181/PDw-GlZ-sbE=
```

Conclusion: This experiment demonstrates the complete process of setting up the Flutter development environment. The installation involves multiple components including the Flutter SDK, Android Studio IDE, and plugins that work together to create a functional development environment. The Flutter doctor tool helps identify and fix any missing dependencies. Once properly configured, developers can create Flutter projects and run them on emulators or physical devices, providing a foundation for mobile application development using Flutter's cross-platform capabilities.