VIROTOUR

Project Deployment and Operations Guide

University of Maryland Global Campus
SWEN 670 : Team A
Spring 2023
Version 2.0

April 4, 2023

Document History

Version	Issue Date	Changes
0.1	3/22/2023	Initial Draft
1.0	3/25/2023	Milestone 3 Submission
2.0	4/4/2023	Milestone 4 Re-submission

Contents

1. Introduction

This Project Deployment and Operations Guide (DepOps Guide) is designed to provide a comprehensive overview of the front-end development of ViroTour's, a camera application that captures three-dimensional panoramic images, architecture, configuration, and troubleshooting procedures. Following the instructions in this document is critical to ensure that the system operates smoothly and efficiently.

1.1. Purpose

The purpose of this document is to provide the users with sufficient information to deploy the application on their choice of device. The DepOps Guide outlines the important steps involved in installing the system, resources, key persons, deployment schedule and completion requirements.

1.2. Intended Audience

The development team and stakeholders are the target audience of this DepOps Guide. The users of this DepOps Guide will be able to sufficiently deploy and test the ViroTour application.

1.3. Technical Project Stakeholders

The technical stakeholders of the project are listed below:

Stakeholder Name	Project Role
Dr. Mir Assadullah	Client/Professor
Roy Gordon	Project Mentor
Robert Wilson	DevSecOps Mentor
Khoa Nguyen	Project Manager (PM)
Jacob Lynn	Product Owner (PO)/Test Manager
Viet Nguyen	Lead Developer (Dev Lead)
Tilahun Abreha	Business Analyst (BA)
Fedor Menchukov	Software Engineer (Dev)
Jude Ibe	Software Engineer (Dev)
Nicholas Platt	Software Engineer (Dev)
Samson Alemneh	Software Engineer (Dev)
Jeffrey Welch	Software Engineer (Dev)
Shawn Kagwa	Software Engineer (Dev)
Christian Dovel	Software Engineer (Dev)

Table 1.3 - Technical Project Stakeholders

1.4. Project Documents

There are various documents created for this effort to provide the stakeholders, namely the project team, the client, and external users with sufficient information and understanding for the success of the project. These documents are summarized below.

	Document	Version	Date
1	Project Management Plan (PMP)	4.0	4/4/2023
2	Software Requirements Specification (SRS)	4.0	4/4/2023
3	Technical Design Document (TDD)	3.0	4/4/2023
4	Software Test Plan (STP)	2.0	4/4/2023
5	Programmers Guide (PG)	2.0	4/4/2023
6	Project Deployment and Operations Guide (DepOps)	2.0	4/4/2023
7	User Guide (UG)	1.0	4/4/2023
8	Test Report (TR)	1.0	4/4/2023

Table 1.4 - Project Documents

1.5. References

- Assadullah, M. (2023). Software Engineering Project. Retrieved from: https://learn.umgc.edu/d2l/home/732302
- Android Studio (n.d.). System requirements. Retrieved February 21, 2023, from https://developer.android.com/studio
- Developer.Android (n.d.). Emulator system requirements. Retrieved on Feb 23, 2023, from https://developer.android.com/studio/run/emulator#requirements
- Developer.Android (n.d.). Run apps on a hardware device. Retrieved on Feb 28, 2023, from https://developer.android.com/studio/run/device
- Flutter (n.d.). Install. Retrieved February 21, 2023, https://docs.flutter.dev/get-started/install. Flutter. (n.d.). Flutter SDK. Retrieved on Feb 18, 2023, from https://snapcraft.io/flutter
- GitHub (n.d.). dart-lang/sdk. Retrieved on Feb 18, 2023, from https://github.com/dart-lang/sdk/wiki/Building
- Pub.dev (n.d.). Flutter packages. Retrieved on Feb 23, 2023, from https://pub.dev/packages?q=sdk%3Aflutter.

1.6. Definitions, Acronyms, and Abbreviations

- API - Application Program Interface
- APK - Android Package Kit
- AVD - Azure Virtual Desktop (AVD and formerly Windows Virtual Desktop, WVD) is a desktop and app virtualization service that runs on the cloud.
- CPU – Central Processing Unit
- CRUD - CREATE, READ, UPDATE and DELETE
- DepOps - Deployment and Operations
- HTTP - Hypertext Transfer Protocol - is the foundation of the World Wide Web, and is used to load webpages using hypertext links. HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack.
- HTTPS - A combination of the Hypertext Transfer Protocol (HTTP) with the Secure Socket Layer (SSL)/Transport Layer Security (TLS) protocol. TLS is an authentication and security protocol widely implemented in browsers and Web servers.
- Hotspots - transition points, as in points that indicate a direction of a new image
- Hotspot Information Points - points of interest, information in the form of text and potentially a picture that is docked to a point in a picture
- GB - Gigabyte
- IDE - Integrated Development Environment
- IOS - iPhone Operating System
- IPA - iOS Package App
- OS - Operating System
- PM - Project Manager
- PR – Pull Request
- QR - Quick Response
- RAM – Random Access Memory
- SDK - Software Development Kit
- SRS – Software Requirement Specification
- STP – Software Test Plan
- TDD – Technical Design Documentation
- UI - User Interface
- UMGC – University of Maryland Global Campus
- USB – Universal Serial Bus
- URL - Uniform Resource Locators
- VR – Virtual Reality
- VT – ViroTour
- Wi-Fi – Not an Anacronym - a facility allowing computers, smartphones, or other devices to connect to the internet or communicate with one another wirelessly within a particular area.

2. Development Team Workflow

2.1. Description of Deployment

ViroTour application code is stored using git on GitHub. The GitHub repository name is umgc/spring2023. All team members contribute their code by branching out from the team's development branch, merging their feature branches back to the team's development branch before merging with the other team's code in the joint development branch. Once the code has been confirmed and verified by the stakeholders, the joint development branch will be merged into the class development branch, then the main branch. The workflow is described in the diagram below.

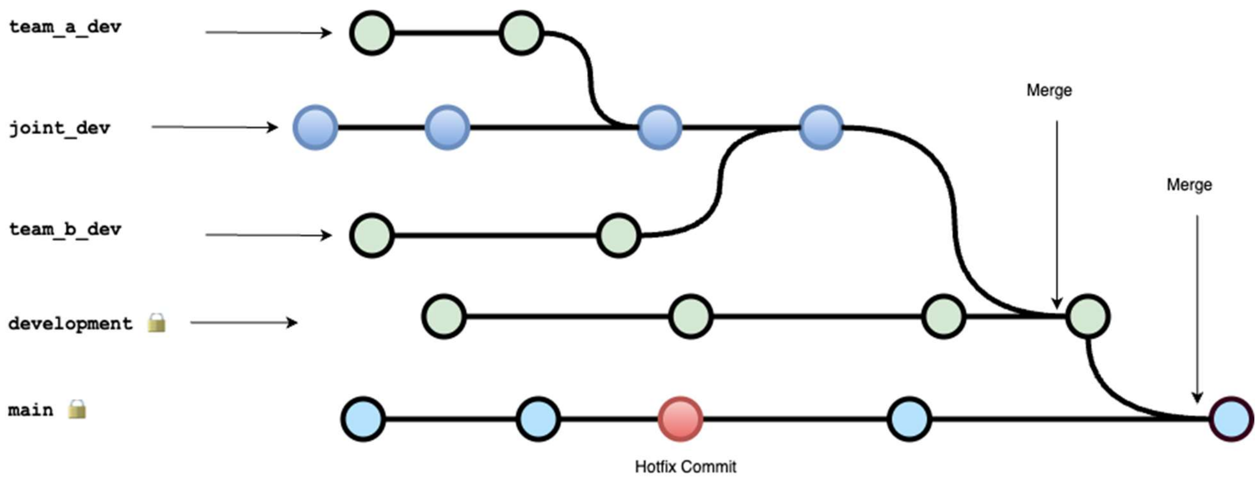


Figure 2.1 - GitHub Workflow

Each developer does his/her own unit testing before merging into the team's branch. A developer creates a Pull Request (PR) to the branch that he/she wants to merge the feature branch code to. Once a PR is created, the maintainers or code owners of the target branch will be requested to review the PR. Once the maintainers approve the PR, then the developer can merge the code.

The application is built using Flutter. It works on mobile devices using iOS and Android. It also works on computers using MacOS and Windows. The application is built and deployed to different platforms using Flutter built-in commands.

2.2. Points of Contact

Name	Project Role
Dr. Mir Assadullah	Professor / Client
Roy Gordon	Project Mentor

Robert Wilson	Project Mentor
Khoa Nguyen	Project Manager
Jacob Lynn	Product Owner

Table 2.2 - Points of Contacts

2.3. Major Tasks

- Project planning and documentation
- Requirement identification and documentation
- Branch out from the class repository
- Setting environment, language, framework, emulators and plugins
- Researching on each requirement and requirement breakdown into functions
- Development of each function into the codebase
- Test throughout the iteration
- Execution and revision
- Deployment of the App and completion.

2.4. Deployment Schedule

The development of the application will be completed before the third milestone of the project. All individual feature testing will be performed and verified before all features are completed. The revision and the final product testing will happen at the end of the fourth milestone by April 4th, 2023.

MS3 Sprints	Work Area	Testing	Duration (days)	Start	Finish
Sprint 1	Develop	Developing Product and first demo	14	2/12	2/26
Sprint 2	Develop & Test	Develop & Execute Unit Test for Component 1	7	2/26	3/5
Sprint 2	Develop & Test	Develop & Execute Unit Test for Component 2	7	2/26	3/5
Sprint 2	Develop & Test	Develop & Execute Unit Test for Component 3	7	2/26	3/5
Sprint 3	Develop & Test	Develop & Execute Integration Test for Component 1	7	3/5	3/12
Sprint 3	Develop & Test	Develop & Execute Integration Test for Component 2	7	3/5	3/12
Sprint 3	Develop & Test	Develop & Execute Integration Test for Component 3	7	3/5	3/12

Sprint 4	Develop & Test	Develop & Execute System Test for Component 1	7	3/12	3/19
Sprint 4	Develop & Test	Develop & Execute System Test for Component 2	7	3/12	3/19
Sprint 4	Develop & Test	Develop & Execute System Test for Component 3	7	3/12	3/19
Sprint 4	Develop & Test	Develop & Execute User Acceptance Testing	7	3/12	3/19
Sprint 5	Test	Test Report	33	2/26	3/31

Table 10 – Develop and Test Schedule

3. Components and Features of the Mobile Application

3.1. Tour Navigation

3.1.1 Tour List

'Tour List' is the first page that the user sees after clicking the ViroTour icon on their phone to access the application. This page shows all the tours that were recorded and edited. There is an option to edit on each tour.

3.1.2 Hamburger Menu

The hamburger menu is displayed on the top left corner of the screen at all times. When clicking it, the dropdown menu shows three options including Create Tour, Edit Tours, and a Search Bar.

3.1.3 Wheel Menu

The Wheel Menu is displayed on the top right of the screen at all times. The Wheel Menu has two options which are to enable the VR View and Glow Effect.

3.1.4 Create Tour

This page allows users to create a new tour. The new tour contains a Tour Name and an Upload Images option or button that opens the users' device Photos app or folder to select tour images. The page must be able to handle errors when creating a new tour. It also redirects the users to their tours page once receiving status 200 from the backend.

3.1.5 Tour Page

This page displays a single tour with the 360-degree image. The mobile application has a zoom and pane while navigating the tour. This functionality is always displayed and active to be utilized by the user at any moment in the tour navigation. Smooth transition between each transitional hotspots and locations is required. The creator of the tour should see the options to perform CRUD operations for informational hotspots.

3.1.6 View Tour

The start tour icon on the mobile application starts the selected tour from the list. At this stage all the features of tour navigation and customization will engage. These are the zoom and pane, back, forward and reload buttons, and the glow effect slider bar.

This functionality is a screen navigation function where the user can move back and forth to the different displays opened following the directory location. It is displayed and activated depending on the availability of directory to move back and forth. The reload button reloads all the contents of the current display to include updates and enable ease of navigation through the virtual tour.

3.1.6 VR View

To enable Vr view the user can navigate to the Vr View selection in the hamburger menu at the top left of their screen upon startup. The user can then use the "toggle vr" button in the top left corner of their screen to change from regular view to google cardboard view to vr stereo view. Each click will take the user to the next option in the order listed. This feature does not yet function on iOS, but is available on Android and Chrome browsers.

3.2. View Customization

3.2.2 Glow Effect

The mobile application glow effect functionality uses a slide bar to gage the amount of lighting in the display to enable the user to choose the level of lighting on a picture or tour. The slider bar is placed on the home screen while starting a tour to customize the tour navigation. This feature can be accessed from the Wheel Menu.

3.3. Hotspot Customization

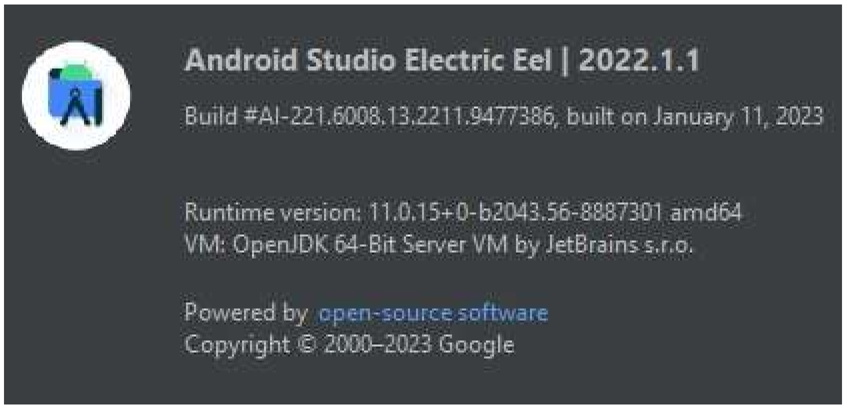
3.3.1 Hotspot Search

The search function will list all available tours that contain a given search term in a grid view. A user can enter a search term in the search box and click the find tours tab to go to the tour of choice and start the virtual tour.

4. Software Installation

4.1. Android Studio

Android Studio is an IDE (integrated development environment) for Android application development. It uses a Gradle-based build system, Android Emulator, code templates and GitHub integration. Android Studio is available for macOS, Windows and Linux desktop platforms. Android Studio latest stable release is Electric Eel | 2022.1.1.



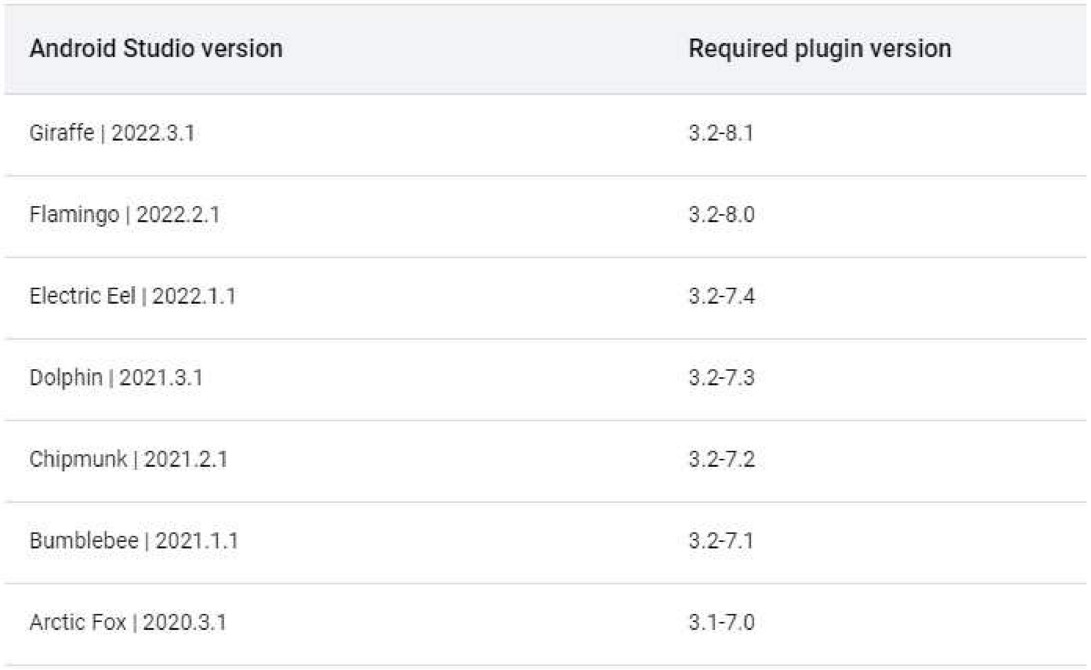
Android Studio Electric Eel | 2022.1.1

Build #AI-221.6008.13.2211.9477386, built on January 11, 2023

Runtime version: 11.0.15+0-b2043.56-8887301 amd64
VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.

Powered by open-source software
Copyright © 2000–2023 Google

Figure 4.1a: Electric Eel | 2022.1.1



Android Studio version	Required plugin version
Giraffe | 2022.3.1	3.2-8.1
Flamingo | 2022.2.1	3.2-8.0
Electric Eel | 2022.1.1	3.2-7.4
Dolphin | 2021.3.1	3.2-7.3
Chipmunk | 2021.2.1	3.2-7.2
Bumblebee | 2021.1.1	3.2-7.1
Arctic Fox | 2020.3.1	3.1-7.0

Figure 4.1b: Versions of Android Studio

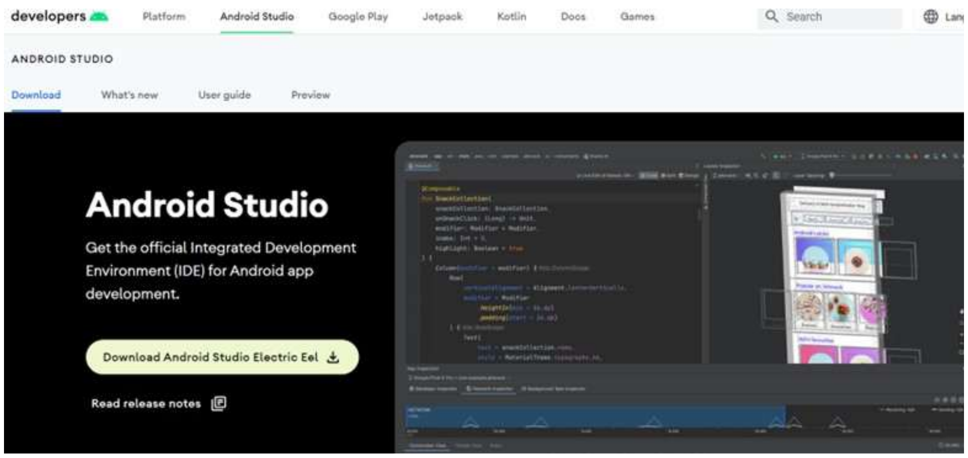
4.1.1. Installation

Windows

System Requirements (https://developer.android.com/studio)

- 64-bit Microsoft® Windows® 8/10
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor Framework
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

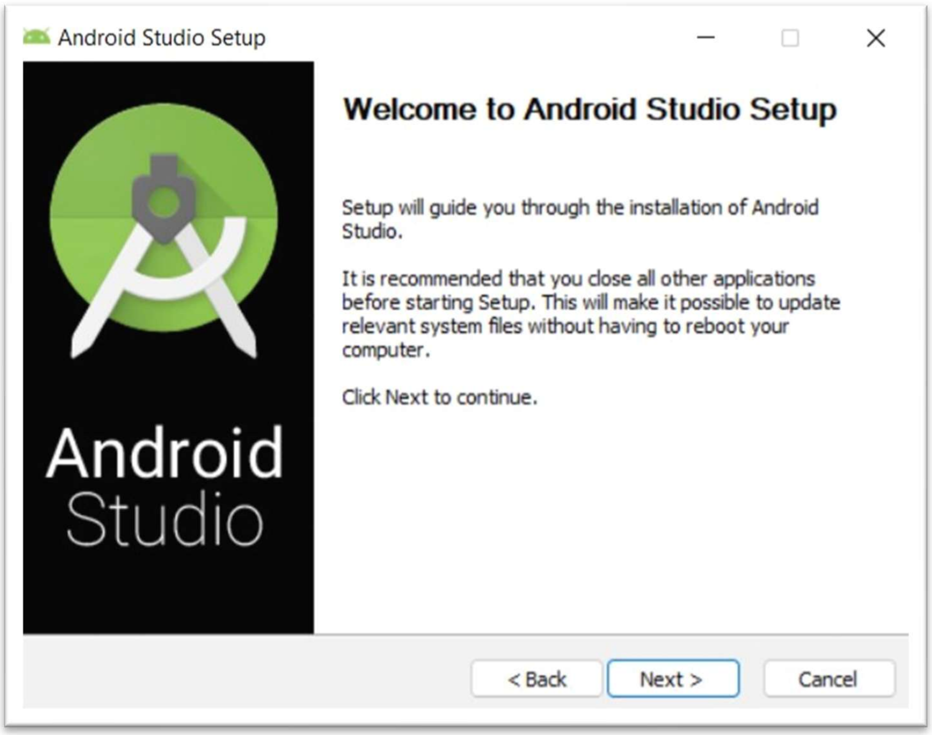
Download the exe file (recommended) from https://developer.android.com/studio or alternatively the ZIP file from the same location.



Get the official Integrated Development Environment (IDE) for Android app development.

Figure 4.1.1a: Download exe

Initiate installation by double-clicking on the exe file and follow the wizard through the installation steps.



Welcome to Android Studio Setup

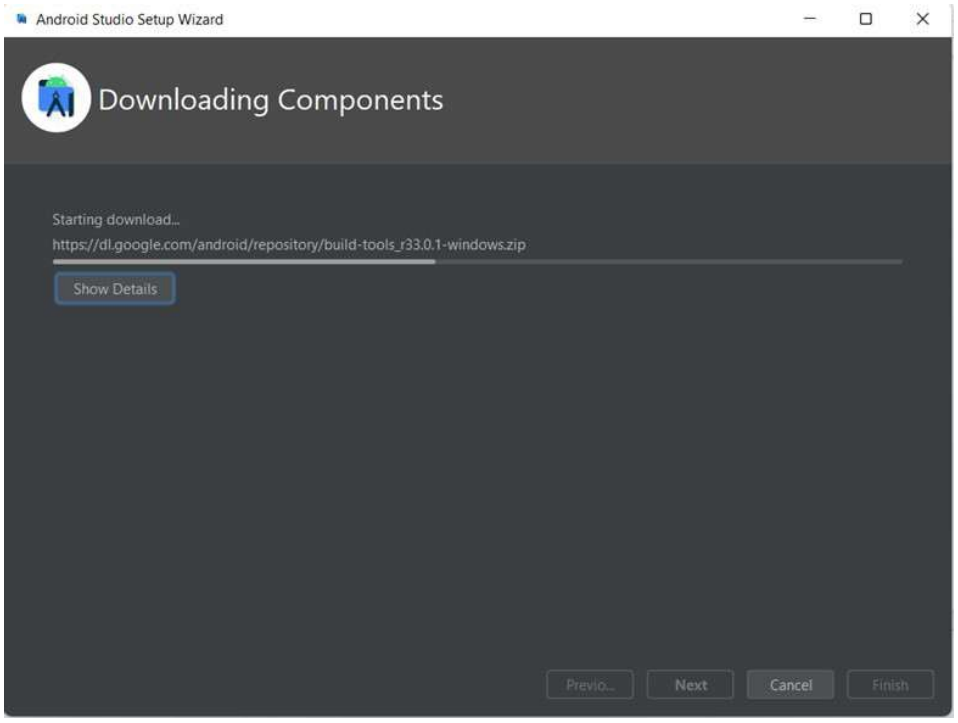
Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 4.1.1b: Android Studio Setup

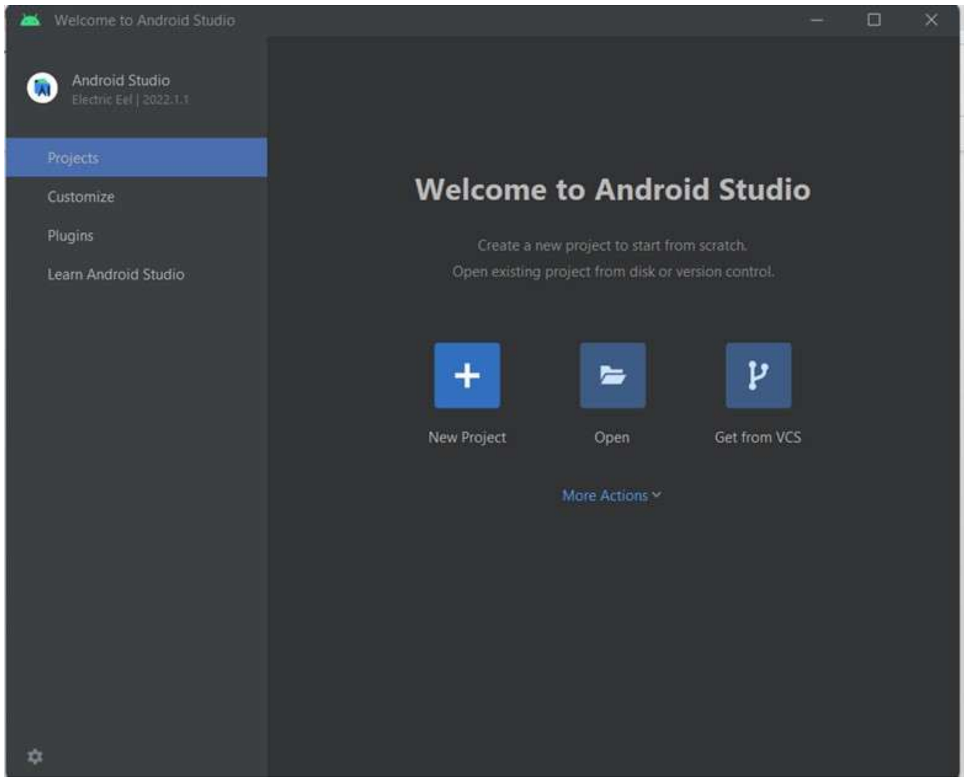
Follow the on-screen instructions, accept agreements and continue, then it will download components.



Starting download...
https://dl.google.com/android/repository/build-tools_r33.0.1-windows.zip

Figure 4.1.1c: Downloading Components

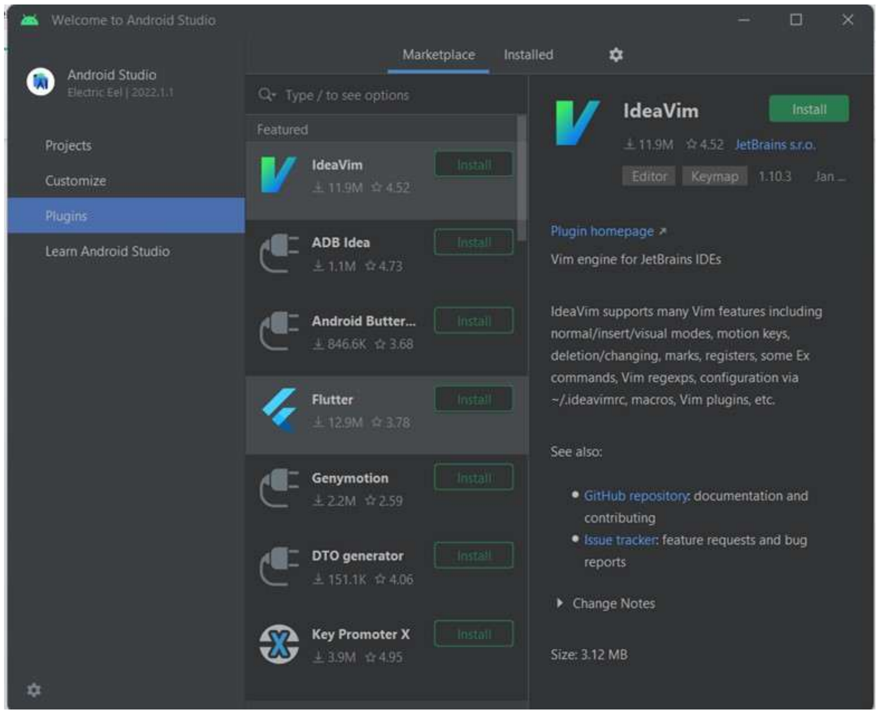
Once it finishes downloading components, you will see the following screen.



Create a new project to start from scratch.
Open existing project from disk or version control.

Figure 4.1.1d: Android Studio Installed

Click on Plugins and install Flutter, Dart and other plugins as applicable. For installation instructions of Flutter and Dart SDK, please refer to next section (4.1.2).

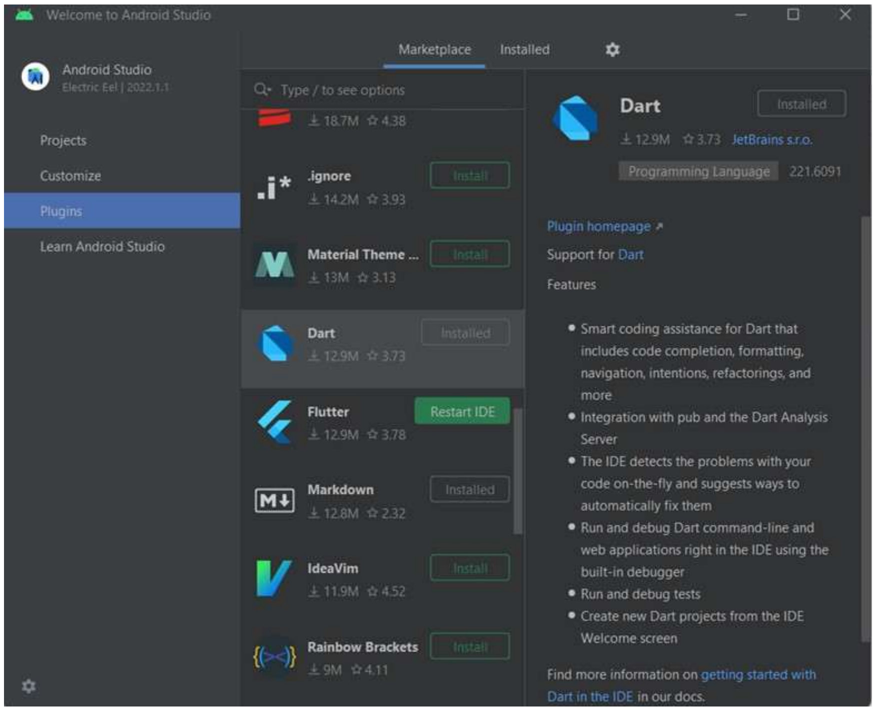

Welcome to Android Studio
Android Studio
Electric Eel | 2022.1.1
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
Type / to see options
Featured
IdeaVim 11.9M 4.52 Install
ADB Idea 1.1M 4.73 Install
Android Butter... 846.6K 3.68 Install
Flutter 12.9M 3.78 Install
Genymotion 2.2M 2.59 Install
DTO generator 151.1K 4.06 Install
Key Promoter X 3.9M 4.95 Install
IdeaVim
Install
11.9M 4.52 JetBrains s.r.o.
Editor Keymap 1.10.3 Jan ...
Plugin homepage
See also:
Change Notes
Size: 3.12 MB

Vim engine for JetBrains IDEs

IdeaVim supports many Vim features including normal/insert/visual modes, motion keys, deletion/changing, marks, registers, some Ex commands, Vim regexps, configuration via ~/.ideavimrc, macros, Vim plugins, etc.

- GitHub repository: documentation and contributing
- Issue tracker: feature requests and bug reports

Figure 4.1.1e: Install Flutter


Welcome to Android Studio
Android Studio
Electric Eel | 2022.1.1
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
Type / to see options
18.7M 4.38
.ignore 14.2M 3.93 Install
Material Theme ... 13M 3.13 Install
Dart 12.9M 3.73 Installed
Flutter 12.9M 3.78 Restart IDE
Markdown 12.8M 2.32 Installed
IdeaVim 11.9M 4.52 Install
Rainbow Brackets 9M 4.11 Install
Dart
Installed
12.9M 3.73 JetBrains s.r.o.
Programming Language 221.6091
Plugin homepage

Support for Dart

Features

- Smart coding assistance for Dart that includes code completion, formatting, navigation, intentions, refactorings, and more
- Integration with pub and the Dart Analysis Server
- The IDE detects the problems with your code on-the-fly and suggests ways to automatically fix them
- Run and debug Dart command-line and web applications right in the IDE using the built-in debugger
- Run and debug tests
- Create new Dart projects from the IDE Welcome screen

Find more information on getting started with Dart in the IDE in our docs.

Figure 4.1.1f: Install Dart

Once installation of plugins has been completed, restart Android Studio and you will see the following screen with "New Flutter Project" which means that you are done.

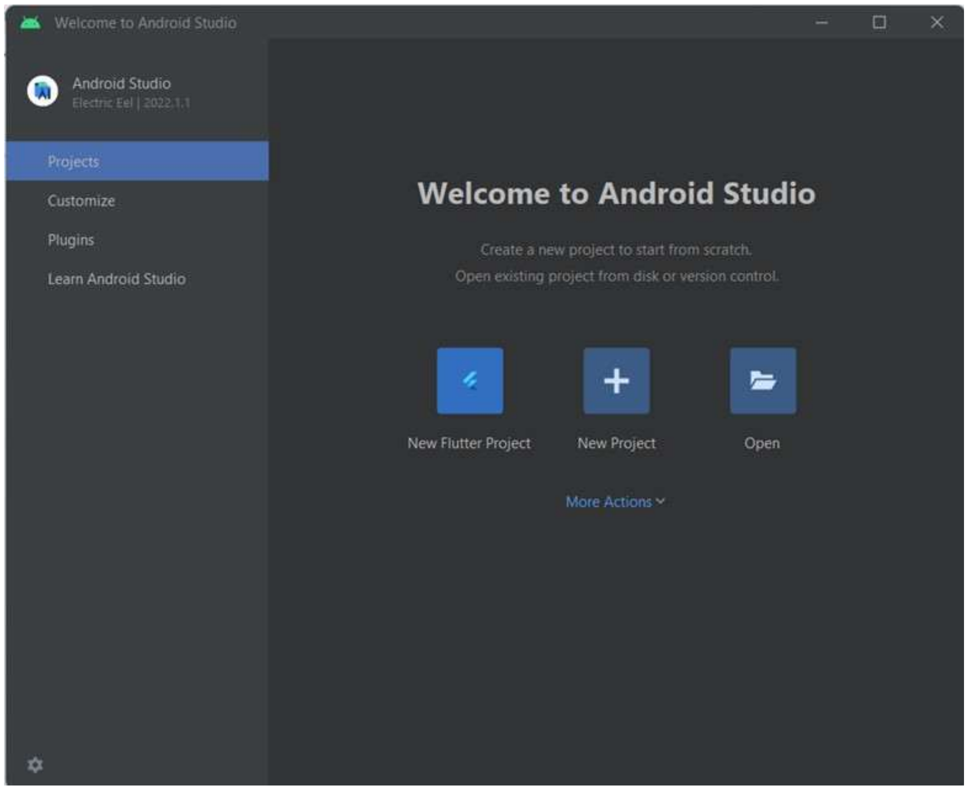

Welcome to Android Studio
Android Studio
Electric Eel | 2022.1.1
Projects
Customize
Plugins
Learn Android Studio
Welcome to Android Studio
Create a new project to start from scratch.
Open existing project from disk or version control.
New Flutter Project
New Project
Open
More Actions

Figure 4.1.1g: Android Studio is Ready

MacOS

System Requirements (https://developer.android.com/studio)

- MacOS® 10.14 (Mojave) or higher
- ARM-based chips, or 2nd generation Intel Core or newer with support for Hypervisor.Framework
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

1. Download the android-studio-2022.1.1.20-mac_arm.dmg from https://developer.android.com/studio.
2. Drag and drop Android Studio into the Applications folder, open the Applications folder, and launch Android Studio.
3. Go to Preferences => Plugins => Search for Flutter and install it with its depencey Dart.
4. Restart Android Studio

Linux

System Requirements (https://developer.android.com/studio)

- Any 64-bit Linux distribution that supports Gnome, KDE, or Unity DE; GNU C Library (glibc) 2.31 or later.
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD processor with support for AMD Virtualization (AMD-V) and SSSE3
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

1. Download the android-studio-2022.1.1.20-linux.tar.gz file from

   https://developer.android.com/studio#downloads:~:text=android%2Dstudio%2D2022.1.1.20%2Dlinux.tar.gz

2. Extract tar.gz file and move contents to where application files are stored
3. Open a terminal, change to the android-studio/bin directory, and execute studio.sh. This will launch the setup wizard
4. Follow the setup wizard and install the SDK dependencies

Chrome OS

System Requirements (https://chromeos.dev/en/android-environment#install-android-studio-on-chrome-os)

- 8 GB RAM or more recommended
- 20 GB of available disk space minimum
- 1280 x 800 minimum screen resolution
- Intel i5 or higher (U series or higher) recommended

1. Download the android-studio-2022.1.1.20-cros.deb from

   https://developer.android.com/studio#downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dcros.deb

2. Open the Files app and locate the Debian (DEB) package that was downloaded in the Downloads folder under My files
3. Right-click the DEB package and select "Install with Linux (Beta)"
4. Follow the setup wizard and install the SDK dependencies

4.2. Flutter and Dart

4.2.1. Flutter

Flutter SDK can be downloaded and run on different operating systems, Windows, mac OS, Linux, Chrome OS from Flutter's official site https://docs.flutter.dev/get-started/install.

Windows

To install and run Flutter on Windows OS, the development environment must meet these minimum system requirements (https://docs.flutter.dev/get-started/install/windows).

Operating Systems: Windows 10 or later (64-bit), x86-64 based.

Disk Space: 1.64 GB (does not include disk space for IDE/tools).

Tools: Flutter depends on these tools being available in your environment.

Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10)

Git for Windows 2.x, with the Use Git from the Windows Command Prompt option.

If Git for Windows is already installed, make sure you can run git commands from the command prompt or PowerShell.

1. Download the following installation bundle to get the latest stable release of the Flutter SDK: https://storage.googleapis.com/flutter_infra_release/releases/stable/windows/flutter_windows_3.7.3-stable.zip

2. Extract the zip file and place the contained flutter in the desired installation location for the Flutter SDK (for example, C:\src\flutter). Do not install Flutter to a path that contains special characters or spaces. Do not also install Flutter in a directory like C:\Program Files\ that requires elevated privileges. After extraction has been completed, open the folder and click on flutter console which will open the Flutter command line.

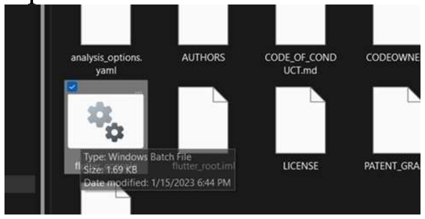
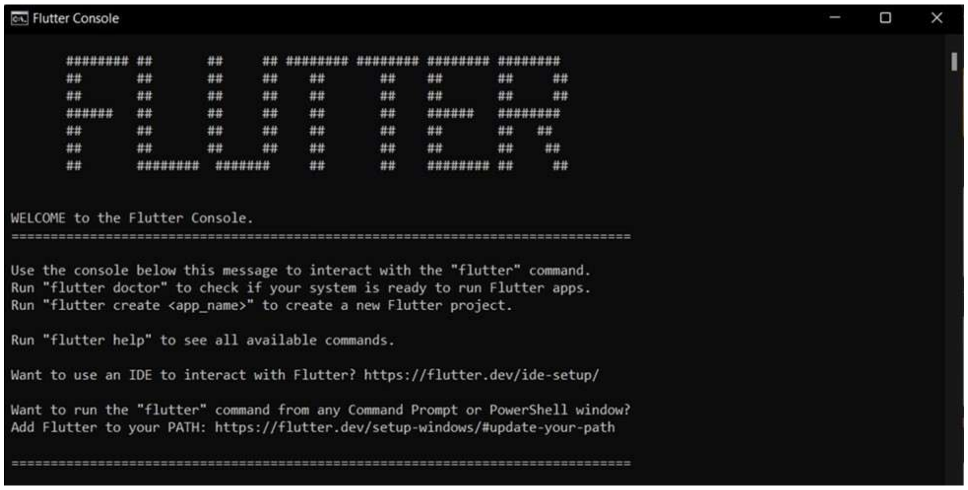


Figure 4.2.1a Flutter Consol Icon



Flutter Console

```
  ######## ##       ##     ## ######## ######## ######## ########
  ##       ##       ##     ##    ##       ##    ##       ##     ##
  ##       ##       ##     ##    ##       ##    ##       ##     ##
  ######   ##       ##     ##    ##       ##    ######   ########
  ##       ##       ##     ##    ##       ##    ##       ##   ##
  ##       ##       ##     ##    ##       ##    ##       ##    ##
  ##       ########  #######     ##       ##    ######## ##     ##


WELCOME to the Flutter Console.
================================================================================

Use the console below this message to interact with the "flutter" command.
Run "flutter doctor" to check if your system is ready to run Flutter apps.
Run "flutter create <app_name>" to create a new Flutter project.

Run "flutter help" to see all available commands.

Want to use an IDE to interact with Flutter? https://flutter.dev/ide-setup/

Want to run the "flutter" command from any Command Prompt or PowerShell window?
Add Flutter to your PATH: https://flutter.dev/setup-windows/#update-your-path

================================================================================
```

Figure 4.2.1b Flutter Consol

3. Update your path

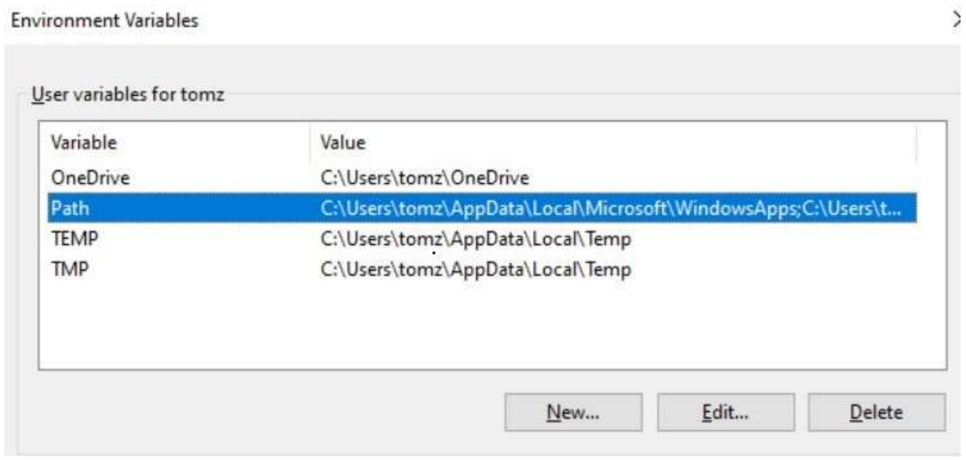
If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the PATH environment variable:

- From the Start search bar, enter 'env' and select Edit environment variables for your account.
- Under User variables check if there is an entry called Path:

If the entry exists, append the full path to flutter\bin using; as a separator from existing values.

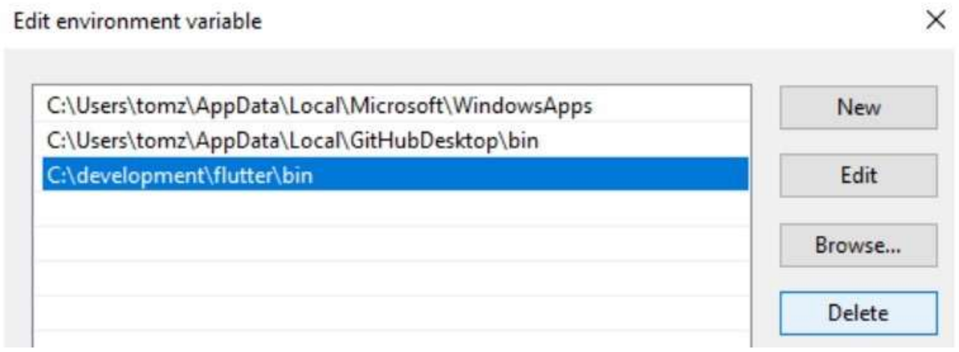
If the entry doesn't exist, create a new user variable named Path with the full path to flutter\bin as its value.

You have to close and reopen any existing console windows for these changes to take effect.



Variable	Value
OneDrive	C:\Users\tomz\OneDrive
Path	C:\Users\tomz\AppData\Local\Microsoft\WindowsApps;C:\Users\t...
TEMP	C:\Users\tomz\AppData\Local\Temp
TMP	C:\Users\tomz\AppData\Local\Temp

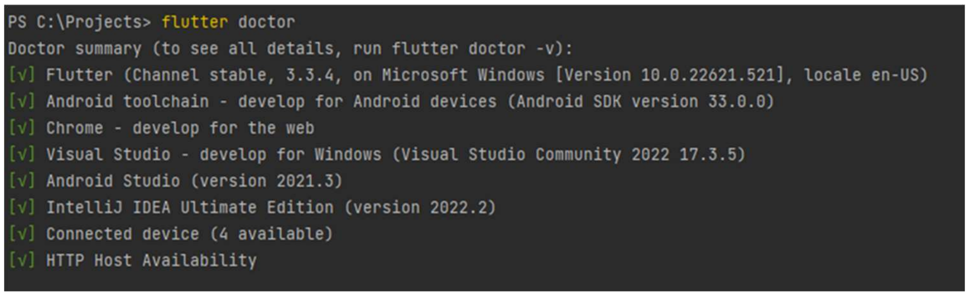
Figure 4.2.1c Update Path



C:\Users\tomz\AppData\Local\Microsoft\WindowsApps
C:\Users\tomz\AppData\Local\GitHubDesktop\bin
C:\development\flutter\bin

Figure 4.2.1d Edit/Add Path

4. To make sure that all needed dependencies are installed and ready, run Flutter doctor.



```
PS C:\Projects> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.3.4, on Microsoft Windows [Version 10.0.22621.521], locale en-US)
[√] Android toolchain - develop for Android devices (Android SDK version 33.0.0)
[√] Chrome - develop for the web
[√] Visual Studio - develop for Windows (Visual Studio Community 2022 17.3.5)
[√] Android Studio (version 2021.3)
[√] IntelliJ IDEA Ultimate Edition (version 2022.2)
[√] Connected device (4 available)
[√] HTTP Host Availability
```

Figure 4.2.1e Flutter Doctor Summary

MacOS

To install and run Flutter on macOS, the development environment must meet these minimum system requirements (https://docs.flutter.dev/get-started/install/macos).

Operating Systems: macOS

Disk Space: 2.8 GB (does not include disk space for IDE/tools).

Tools: Flutter uses git for installation and upgrade. Installing Xcode which includes git is recommended, but git can be installed separately.

If user is installing on an Apple Silicon Mac, the Rosetta translation environment should be made available for some ancillary tools. This can be installed manually by running the following command.

```
$ sudo softwareupdate --install-rosetta --agree-to-license
```

1. Download the installation bundle to get the latest stable release of the Flutter SDK from the link below.
   - Intel

     https://storage.googleapis.com/flutter_infra_release/releases/stable/macos/flutter_macos_3.7.4-stable.zip

   - Apple Silicon

     https://storage.googleapis.com/flutter_infra_release/releases/stable/macos/flutter_macos_arm64_3.7.4-stable.zip

2. Extract the file in the desired location, for example:

```
$ cd ~/development
$ unzip ~/Downloads/flutter_macos_3.7.4-stable.zip
```

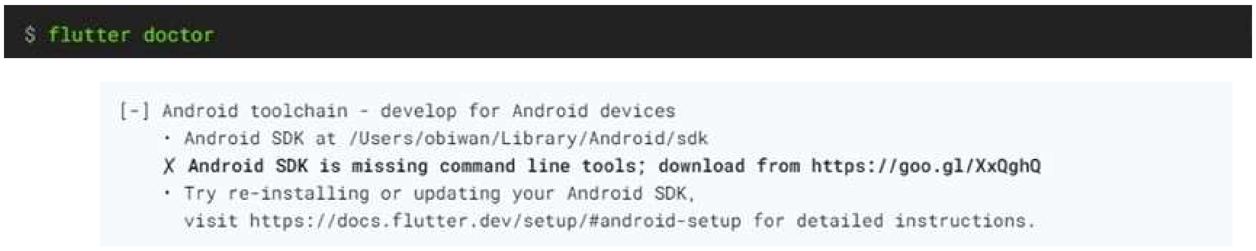
3. Add the flutter tool to your path

```
$ export PATH="$PATH:`pwd`/flutter/bin"
```

This command sets the PATH variable for the current terminal window only. To permanently add Flutter to your path please follow the instructions from https://docs.flutter.dev/get-started/install/macos#update-your-path.

4. Run flutter doctor

Run the following command to see if there are any dependencies you need to install to complete the setup (for verbose output, add the -v flag)



```
$ flutter doctor
```

```
[-] Android toolchain - develop for Android devices
    • Android SDK at /Users/obiwan/Library/Android/sdk
    ✗ Android SDK is missing command line tools; download from https://goo.gl/XxQghQ
    • Try re-installing or updating your Android SDK,
      visit https://docs.flutter.dev/setup/#android-setup for detailed instructions.
```

Figure 4.2.1f Flutter Doctor Summary

Once all missing dependencies have been installed, run the flutter doctor command again to verify that everything set up correctly.

Linux

To install and run Flutter on Linux, the development environment must meet these minimum system requirements (https://docs.flutter.dev/get-started/install/linux)

Operating Systems: Linux (64-bit)

Disk Space: 600 MB (does not include disk space for IDE/tools).

Tools: Flutter depends on these command-line tools being available in your environment.

- bash
- curl
- file
- git 2.x
- mkdir
- rm
- unzip
- which
- xz-utils
- Zip

Shared libraries: Flutter test command depends on this library being available in your environment.

- libGLU.so.1 - provided by mesa packages such as libglu1-mesa on Ubuntu/Debian and mesa-libGLU on Fedora.

There are two ways to install Flutter on Linux

a. Install Flutter using snapd

The easiest way to install Flutter on Linux is to use snapd. See https://snapcraft.io/docs/installing-snapd on how to install snapd. After snapd has been installed, you can install Flutter from the Snap store (https://snapcraft.io/flutter), or at the command line.

```
$ sudo snap install flutter --classic
```

Once the snap is installed, you can use the following command to display your Flutter SDK path.

```
$ flutter sdk-path
```

b. Install Flutter manually

To install Flutter manually, the following steps should be followed.

1. Download the following installation bundle to get the latest stable release of the Flutter SDK

   https://storage.googleapis.com/flutter_infra_release/releases/stable/linux/flutter_linux_3.7.4-stable.tar.xz

2. Extract the file in the desired location, for example:

```
$ cd ~/development
$ tar xf ~/Downloads/flutter_linux_3.7.4-stable.tar.xz
```

3. Add the flutter tool to your path

```
$ export PATH="$PATH:`pwd`/flutter/bin"
```

4. Run flutter doctor

Run the following command to see if there are any dependencies you need to install to complete the setup.

```
$ flutter doctor
```

Once all missing dependencies have been installed, run the flutter doctor command again to verify that everything was set up correctly.

Chrome OS

To install and run Flutter, your development environment must meet these minimum requirements.

Operating Systems: Chrome OS (64-bit) with Linux (Beta) turned on

Disk Space: 600 MB (does not include disk space for IDE/tools).

Tools: Flutter depends on these command-line tools being available in your environment.

- bash
- curl
- git 2.x
- mkdir
- rm

- unzip
- which
- xz-utils

Shared libraries: Flutter test command depends on this library being available in your environment.

- libGLU.so.1 - provided by mesa packages such as libglu1-mesa on Ubuntu/Debian

1. Download the following installation bundle to get the latest stable release of the Flutter SDK.

   https://storage.googleapis.com/flutter_infra_release/releases/stable/linux/flutter_linux_3.7.4-stable.tar.xz

2. In the Files app, drag-and-drop the downloaded file from "Downloads" to "Linux Files" to access Flutter from your Linux container.
3. Extract the file in the desired location, for example

```
$ cd ~/development
$ tar xf ~/Downloads/flutter_linux_3.7.4-stable.tar.xz
```

4. Add the flutter tool to your path

```
$ export PATH="$PATH:`pwd`/flutter/bin"
```

5. Run flutter doctor

Run the following command to see if there are any dependencies you need to install to complete the setup.

```
$ flutter doctor
```

Once all missing dependencies have been installed, run the flutter doctor command again to verify that everything was set up correctly.

4.2.2. Dart

The Dart SDK is supported on Windows, Linux, and macOS. The System requirements for the Operating Systems are as follows.

Windows

- **Supported versions**: Windows 10 and 11.
- **Supported architectures**: x64, IA32, ARM64.

  Support for ARM64 is experimental and is available only on the dev channel.

Linux

- **Supported versions**: Debian stable and Ubuntu LTS under standard support.
- **Supported architectures**: x64, IA32, ARM64, ARM, RISC-V (RV64GC).

Support for RISC-V is experimental, and is available only on the dev channel.

macOS

- **Supported versions:** Latest three major versions. Dart supports the following macOS versions as of November 2022:
  - macOS 11 (Big Sur)
  - macOS 12 (Monterey)
  - macOS 13 (Ventura)
- **Supported architectures:** x64, ARM64.

The Dart SDK contains the libraries and command-line tools that are needed to develop Dart command-line, server, and non-Flutter web apps.

When the Flutter SDK is installed, the full Dart SDK also comes with it, and it places Dart's dart command-line interface in its bin folder. However, to easily install and update a stable channel Dart SDK a package manager can be used. The SDK can be built from source (https://github.com/dart-lang/sdk/wiki/Building), a Dart Docker image (https://hub.docker.com/_/dart), or install from any release channel by downloading the SDK as a zip file https://dart.dev/get-dart/archive.

4.2.3. Flutter and Dart Plugins

The following packages are useful general-purpose packages for a wide range of projects.

Package	**Description**	**Commonly used APIs**
archive	Encodes and decodes various archive and compression formats.	Archive, ArchiveFile, TarEncoder, TarDecoder, ZipEncoder, ZipDecoder
characters	String manipulation for user-perceived characters (Unicode grapheme clusters).	String.characters, Characters, CharacterRange
http	A set of high-level functions and classes that make it easy to consume HTTP resources.	delete(), get(), post(), read()
intl	Internationalization and localization facilities, with support for plurals and genders, date and number formatting and parsing, and bidirectional text.	Bidi, DateFormat, MicroMoney, TextDirection
json_serializable	An easy-to-use code generation package. For more information, see JSON Support.	@JsonSerializable

logging	A configurable mechanism for adding message logging to your application.	LoggerHandler, Level, LogRecord
mockito	A popular framework for mocking objects in tests. Especially useful if you are writing tests for dependency injection. Used with the test package.	Answering, Expectation, Verification
path	Common operations for manipulating different types of paths.	absolute(), basename(), extension(), join(), normalize(), relative(), split()
quiver	Utilities that make using core Dart libraries more convenient. Some of the libraries where Quiver provides additional support include async, cache, collection, core, iterables, patterns, and testing.	CountdownTimer (quiver.async); MapCache (quiver.cache); MultiMap, TreeSet (quiver.collection); EnumerateIterable (quiver.iterables); center(), compareIgnoreCase(), isWhiteSpace() (quiver.strings)
shelf	Web server middleware for Dart. Shelf makes it easy to create and compose web servers, and parts of web servers.	Cascade, Pipeline, Request, Response, Server
stack_trace	Methods for parsing, inspecting, and manipulating stack traces produced by the underlying Dart implementation. Also provides functions to produce string representations of stack traces in a more readable format than the native StackTrace implementation.	Trace.current(), Trace.format(), Trace.from()
test	A standard way of writing and running tests in Dart.	expect(), group(), test()
yaml	A parser for YAML.	loadYaml(), loadYamlStream()

Table 4.2.3a - Dart/Flutter Useful Plugins

The following package builds upon a core library, adding functionality and filling in missing features.

Package	Description	Commonly used APIs
async	Expands on dart:async, adding utility classes to work with asynchronous computations.	AsyncMemoizer, CancelableOperation, FutureGroup, LazyStream, Result, StreamCompleter, StreamGroup, StreamSplitter
collection	Expands on dart:collection, adding utility functions and classes to make working with collections easier.	Equality, CanonicalizedMap, MapKeySet, MapValueSet, PriorityQueue, QueueList
convert	Expands on dart:convert, adding encoders and decoders for converting between different data representations. One of the data representations is *percent encoding*, also known as *URL encoding*.	HexDecoder, PercentDecoder
io	Contains two libraries, ansi and io, to simplify working with files, standard streams, and processes. Use the ansi library to customize terminal output. The io library has APIs for dealing with processes, stdin, and file duplication.	copyPath(), isExecutable(), ExitCode, ProcessManager, sharedStdIn

Table 4.2.3b - Dart/Flutter Useful Plugins

The following are some Flutter packages (https://pub.dev/flutter).

Package Name	Description
http: ^0.13.5	A composable, Future-based library for making HTTP requests. This package contains a set of high-level functions and classes that make it easy to consume HTTP resources. It's multi-platform, and supports mobile, desktop, and the browser.
provider: ^6.0.5	A wrapper around InheritedWidget to make them easier to use and more reusable.
cached_network_image: ^3.2.3	A flutter library to show images from the internet and keep them in the cache directory.
google_fonts: ^4.0.3	A Flutter package to use fonts from fonts.google.com.
permission_handler: ^10.2.0	On most operating systems, permissions aren't just granted to apps at install time. Rather, developers have to ask the user for permissions while the app is running.
flutter_bloc: ^8.1.2	Widgets that make it easy to integrate blocs and cubits into Flutter. Built to work with package:bloc.
firebase_core: ^2.7.0	A Flutter plugin to use the Firebase Core API, which enables connecting to multiple Firebase apps.
firebase_auth: ^4.2.9	A Flutter plugin to use the Firebase Authentication API.
firebase_messaging: ^14.2.5	A Flutter plugin to use the Firebase Cloud Messaging API.

geolocator: ^9.0.2	A Flutter geolocation plugin which provides easy access to platform specific location services (FusedLocationProviderClient or if not available the LocationManager on Android and CLLocationManager on iOS).
font_awesome_flutter: ^10.4.0	The free Font Awesome Icon pack available as set of Flutter Icons - based on font awesome version 6.2.1.
equatable: ^2.0.5	Being able to compare objects in Dart often involves having to override the == operator as well as hashCode.
get_it: ^7.2.0	This is a simple Service Locator for Dart and Flutter projects with some additional goodies highly inspired by Splat. It can be used instead of InheritedWidget or Provider to access objects e.g., from your UI.

Table 4.2.3c - Flutter packages

4.2.4. Manual Installing a Package Dependency

If your code editor does not automatically install dependencies, do these steps:

- Add Dependencies: Open the pubspec.yaml file located inside the app folder and add the specific package name under dependencies.
- Install it
  - From the terminal: Run flutter pub get.
    OR
  - From Android Studio: Click Pub get in the action ribbon at the top of pubspec.yaml.
  - From VS Code: Click Get Packages located in right side of the action ribbon at the top of pubspec.yaml.
- Import it: Add a corresponding import statement in the Dart code.
- Stop and restart the app, if necessary

If the package brings platform-specific code (Kotlin/Java for Android, Swift/Objective-C for iOS), that code must be built into your app. Hot reload and hot restart only update the Dart code, so a full restart of the app might be required to avoid errors like MissingPluginException when using the package.

4.2.5. Android Emulator

Android Emulator is used to simulate Android devices on a computer to allow testing application on a variety of devices eliminating the need to acquire multiple devices for testing purpose. Testing on emulator is also faster than testing on a device connected over a USB.

Since the emulator comes with Android Studio, it does not need to be installed separately. There are four basic steps to set up and start running emulator on Android Studio.

Verify that you have the system requirements

To get the best experience from the emulator the computer needs to meet at least the following requirements (https://developer.android.com/studio/run/emulator#requirements).

- 16 GB RAM
- 64-bit Windows, macOS, Linux, or Chrome OS operating system
- 16 GB disk space

Create an Android Virtual Device (AVD)

Creating an AVD that models each device that the app is going to run on is essential to carry out the test effectively.

To start creating a new device from the Android Studio welcome screen, select More Actions > Virtual Device Manager or alternatively from Tools menu select Device Manager

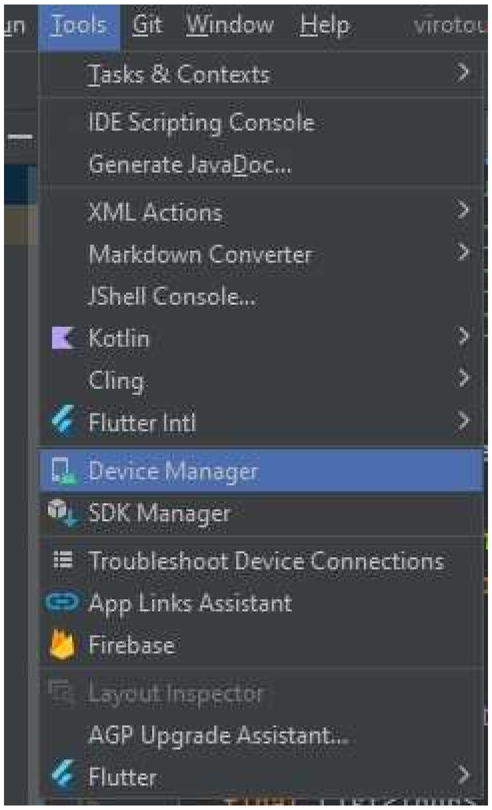
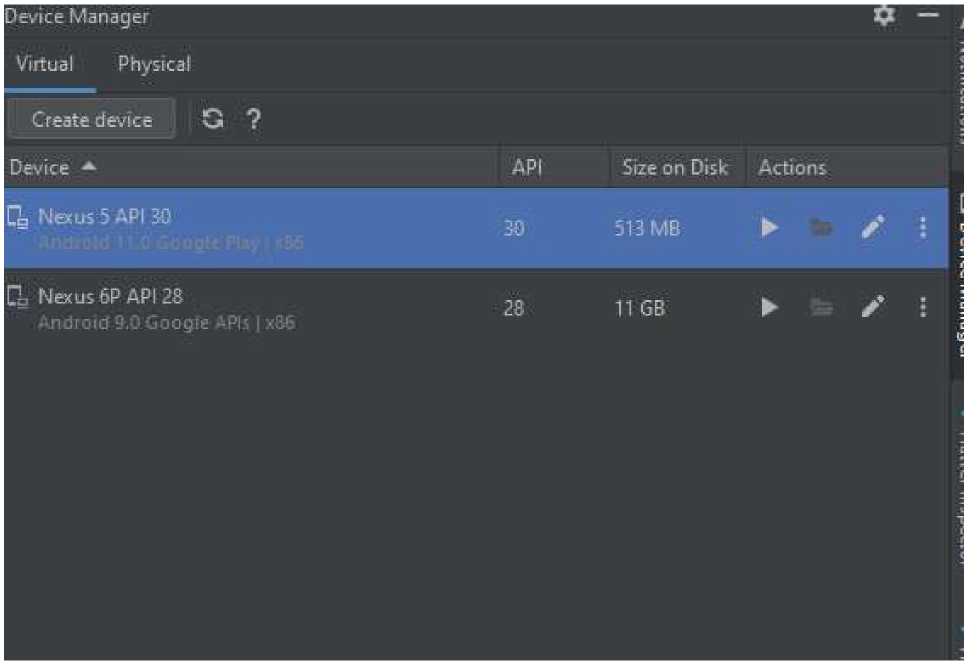
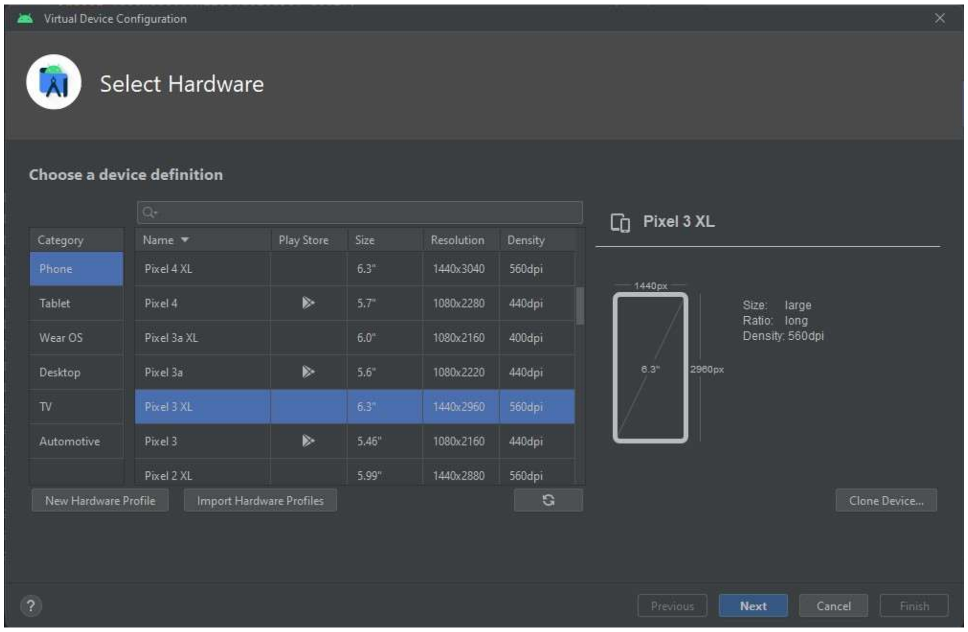


Figure 4.2.5a - Select Device Manager



Device	API	Size on Disk	Actions
Nexus 5 API 30 Android 11.0 Google Play | x86	30	513 MB	
Nexus 6P API 28 Android 9.0 Google APIs | x86	28	11 GB	

Figure 4.2.5b Click on Create Device



Category
Phone
Tablet
Wear OS
Desktop
TV
Automotive

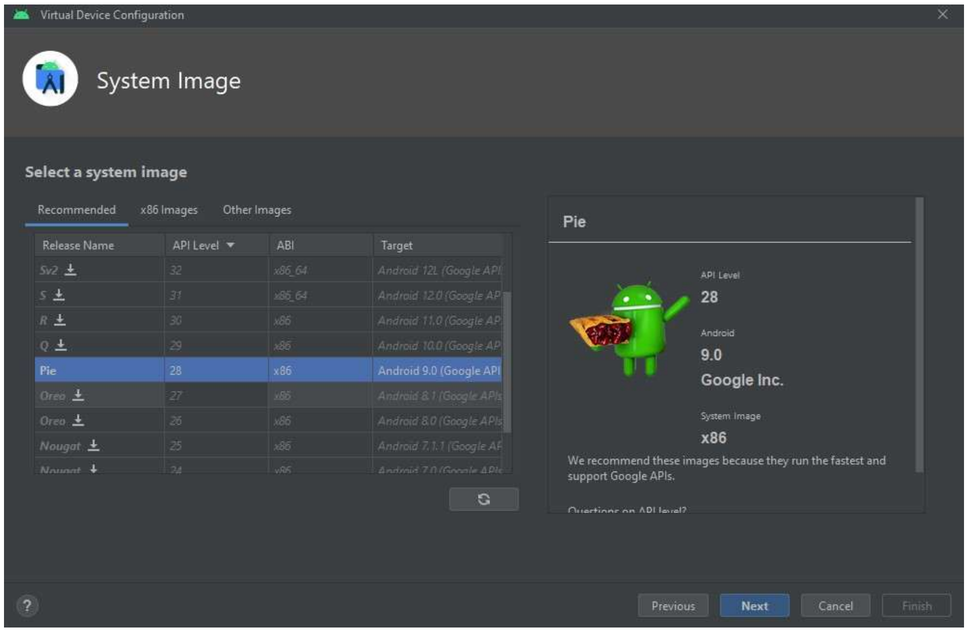
Name	Play Store	Size	Resolution	Density
Pixel 4 XL		6.3"	1440x3040	560dpi
Pixel 4	▶	5.7"	1080x2280	440dpi
Pixel 3a XL		6.0"	1080x2160	400dpi
Pixel 3a	▶	5.6"	1080x2220	440dpi
Pixel 3 XL		6.3"	1440x2960	560dpi
Pixel 3	▶	5.46"	1080x2160	440dpi
Pixel 2 XL		5.99"	1440x2880	560dpi

Pixel 3 XL

1440px
6.3"
2960px

Size: large
Ratio: long
Density: 560dpi

Figure 4.2.5c Select Hardware



Select a system image

Release Name	API Level	ABI	Target
Sv2	32	x86_64	Android 12L (Google API
S	31	x86_64	Android 12.0 (Google AP
R	30	x86	Android 11.0 (Google AP
Q	29	x86	Android 10.0 (Google AP
Pie	28	x86	Android 9.0 (Google API
Oreo	27	x86	Android 8.1 (Google APIs
Oreo	26	x86	Android 8.0 (Google APIs
Nougat	25	x86	Android 7.1.1 (Google AP
Nougat	24	x86	Android 7.0 (Google APIs

Pie

API Level
28

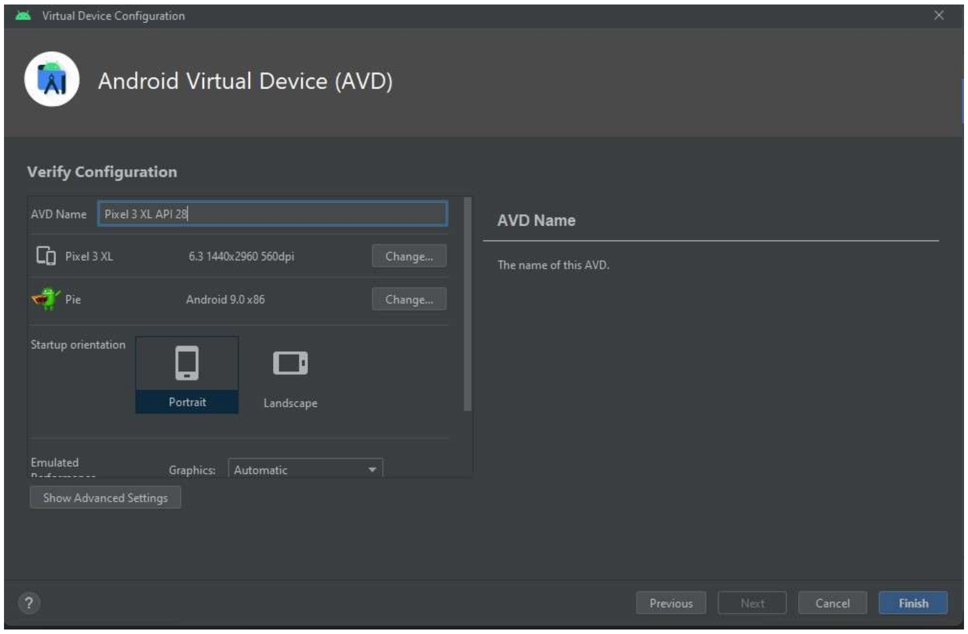
Android
9.0
Google Inc.

System Image
x86

We recommend these images because they run the fastest and support Google APIs.

Questions on API level?

Figure 4.2.5d Select a System Image



Verify Configuration

AVD Name: Pixel 3 XL API 28

Pixel 3 XL — 6.3 1440x2960 560dpi

Pie — Android 9.0 x86

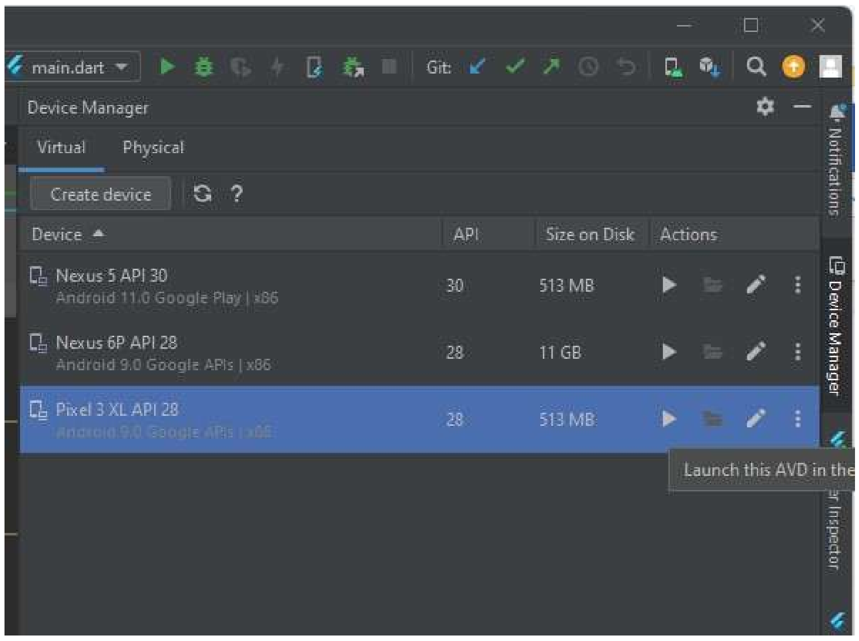
Startup orientation: Portrait / Landscape

Emulated Performance — Graphics: Automatic

AVD Name

The name of this AVD.

Figure 4.2.5e Verify Configuration



Device	API	Size on Disk	Actions
Nexus 5 API 30 Android 11.0 Google Play | x86	30	513 MB	
Nexus 6P API 28 Android 9.0 Google APIs | x86	28	11 GB	
Pixel 3 XL API 28 Android 9.0 Google APIs | x86	28	513 MB	

Launch this AVD in the

Figure 4.2.5f Launch Emulator

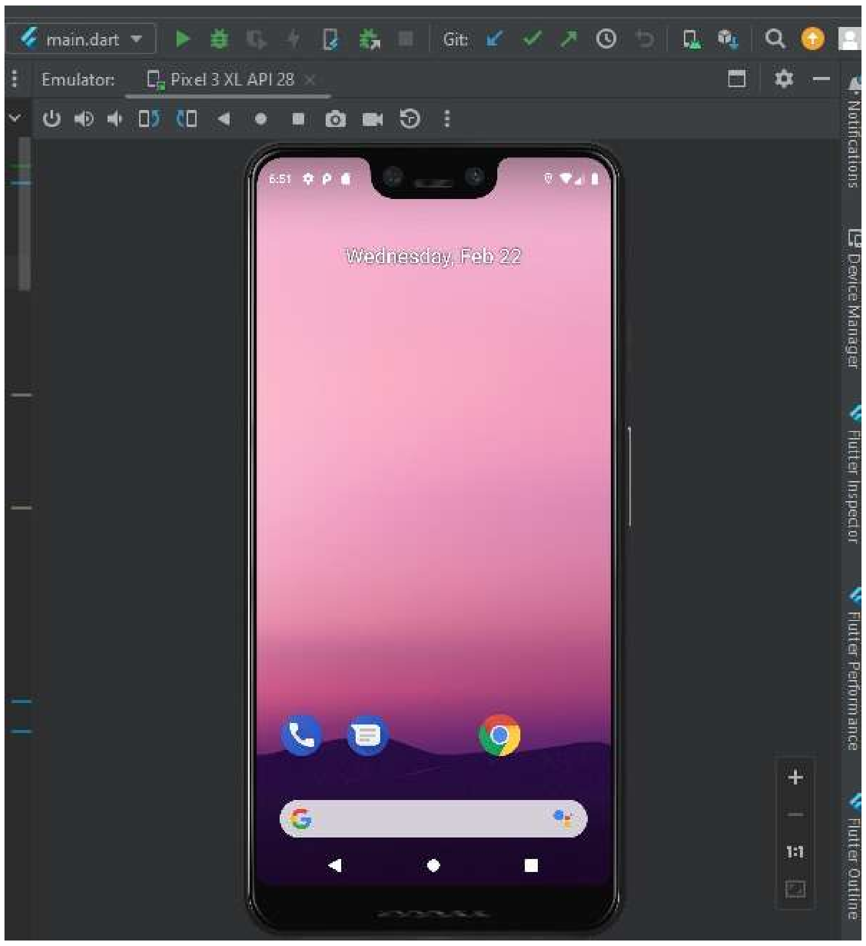


Figure 4.2.5g Emulator is ready

Run your app on the emulator

Once the AVD is created successfully, you can start the emulator and run the app you are testing.

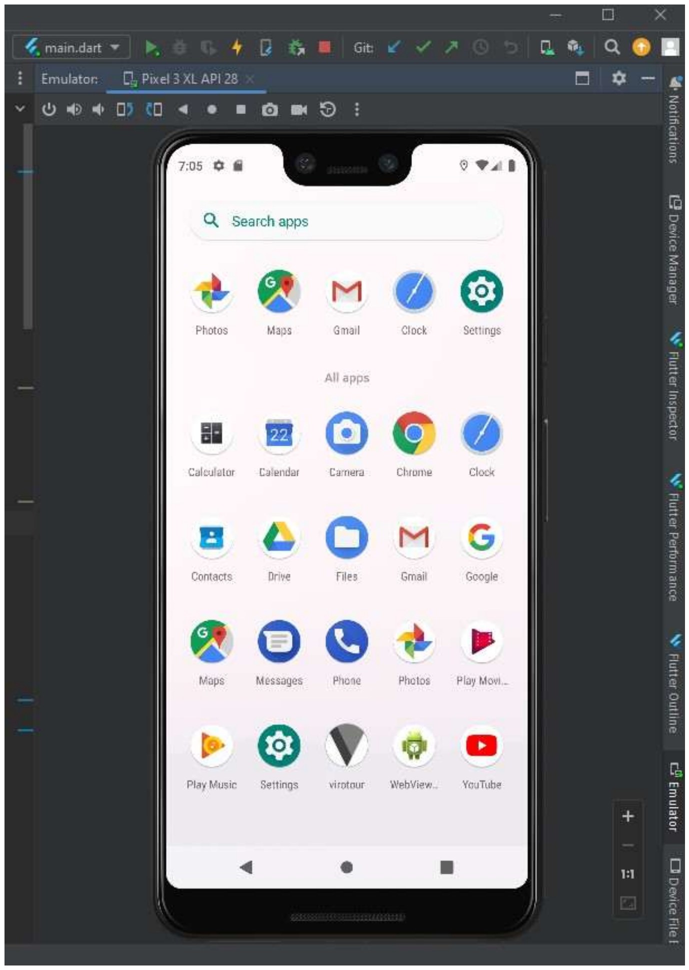


Figure 4.2.5h Run your app on emulator

Navigate the emulator

To navigate the emulator, you can use the mouse pointer to mimic finger on the touchscreen and make selections by clicking on buttons and controls. Computer keyboard can be used to enter characters to the emulator.

Feature	Description
Swipe the screen	Point to the screen, press and hold the primary mouse button, swipe across the screen, and then release.
Drag an item	Point to an item on the screen, press and hold the primary mouse button, move the item, and then release.
Tap	Point to the screen, press the primary mouse button, and then release.

Double tap	Point to the screen, double-click the primary mouse button quickly, and then release.
Touch & hold	Point to an item on the screen, press the primary mouse button, hold, and then release.
Type	You can type in the emulator by using your computer keyboard or using a keyboard that pops up on the emulator screen.
Pinch and spread	Pressing Control (Command on macOS) brings up a pinch gesture multi-touch interface. The mouse acts as the first finger, and across the anchor point is the second finger. Drag the cursor to move the first point. Clicking the left mouse button mimics touching down both points and releasing mimics picking both up.
Vertical swipe	Open a vertical menu on the screen and use the scroll wheel (mouse wheel) to scroll through the menu items. Click a menu item to select it.

Table 4.2.5: Gestures for navigating the emulator

4.2.6. iPhone Emulator – iOS Setup

4.2.6.1. Install Xcode

- Install the latest stable version of Xcode
- Configure the Xcode command-line tools to use the newly installed version of Xcode by running the following from the command line:

```
$ sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
$ sudo xcodebuild -runFirstLaunch
```

- Make sure the Xcode license agreement is signed by either opening Xcode once and confirming or running sudo xcodebuild -license from the command line.

4.2.6.2. Set up the iOS Simulator

- On your Mac, find the Xcode Simulator.
- Make sure your simulator is using a 64-bit device (iPhone 5s or later). You can check the device by viewing the settings in the simulator’s **Hardware > Device** or **File > Open Simulator** menus.
- Select Open iOS Simulator under Flutter Device Selection

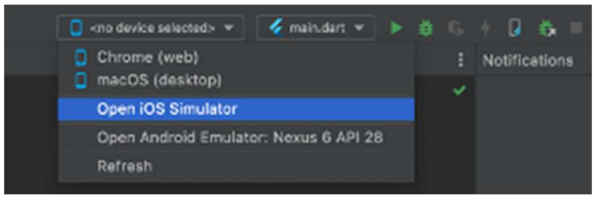


Figure 4.5.2a – Flutter Device Selection – Open iOS Simulator

- Select iOS Device

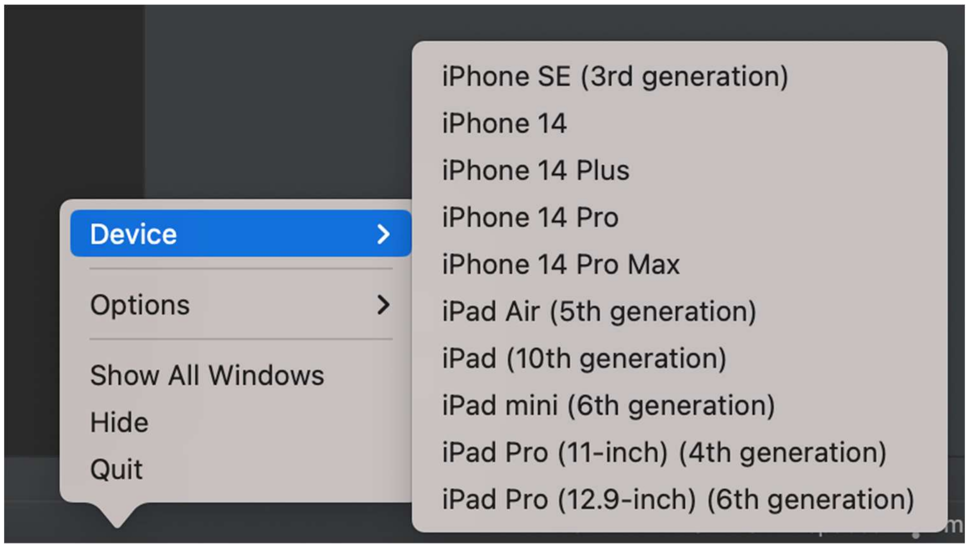

Device
Options
Show All Windows
Hide
Quit
iPhone SE (3rd generation)
iPhone 14
iPhone 14 Plus
iPhone 14 Pro
iPhone 14 Pro Max
iPad Air (5th generation)
iPad (10th generation)
iPad mini (6th generation)
iPad Pro (11-inch) (4th generation)
iPad Pro (12.9-inch) (6th generation)

Figure 4.5.2b – Flutter Device Selection

- Run your application on the emulator

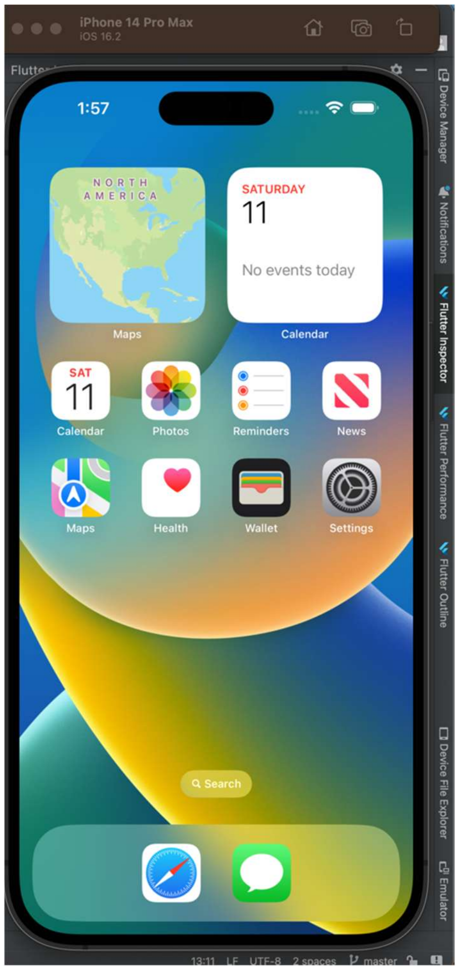


Figure 4.5.2c – iPhone Emulator

4.3. Test App on Hardware Device

It is always advisable to test the application on a physical device before releasing it. This opens a chance to point out issues (if any) that might not have been encountered during virtual testing. There are two ways to setup a physical device to the workstation to perform the testing, one is by connecting the device using USB and the second is connecting the device remotely over Wi-Fi. Connecting over Wi-Fi overcomes issues of connecting device via USB such as driver installation.

To connect a device over Wi-Fi, the following requirements need to be fulfilled.

- The workstation and the device should be on the same wireless network.
- The device needs to be running Android 11 or higher.
- Android Studio Electric Eel needs to be installed.
- The workstation needs to have the latest version of the SDK Platform Tools.

After confirming that these requirements have been fulfilled, the device pairing can be carried out using the following steps (https://developer.android.com/studio/run/device).

1. From the "Run Configurations" menu in Android Studio, select "Pair Devices Using Wi-Fi".

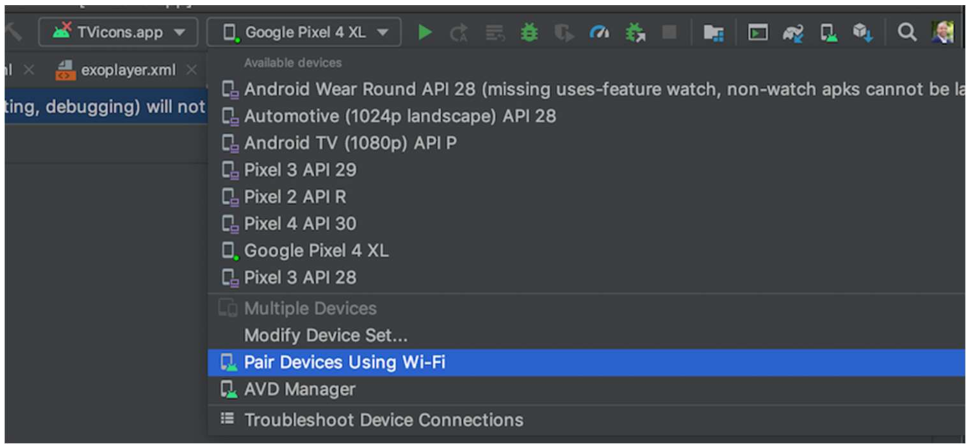
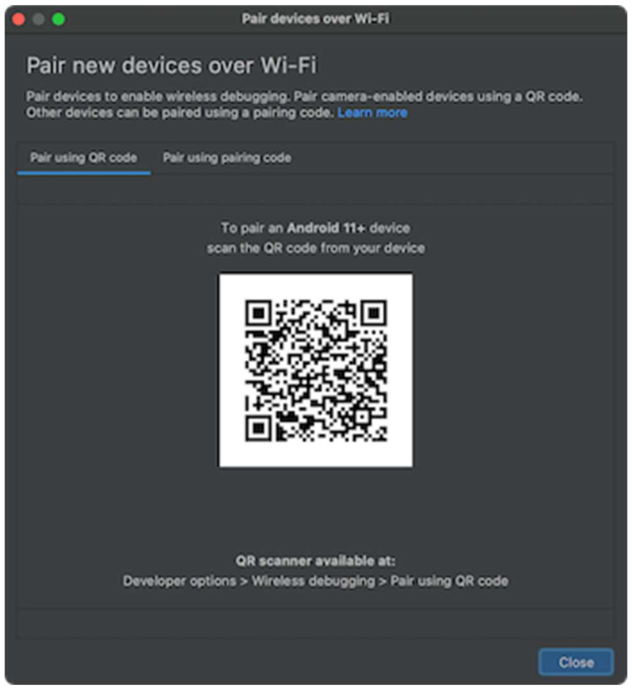


Figure 4.3a: Configuration Menu

The Pair devices over Wi-Fi dialog appears,



Pair new devices over Wi-Fi

Pair devices to enable wireless debugging. Pair camera-enabled devices using a QR code. Other devices can be paired using a pairing code. Learn more

To pair an **Android 11+** device
scan the QR code from your device

QR scanner available at:
Developer options > Wireless debugging > Pair using QR code

Figure 4.3b: Dialog to pair devices using QR code

2. Enable developer options on your device.

The steps may vary depending on the type of device.

Go to Settings -> About phone -> (Software information) -> Build number. Then tap the Build Number option seven times until you see “You are now a developer!” This enables developer options on your device.

3. Enable debugging over Wi-Fi on your device.

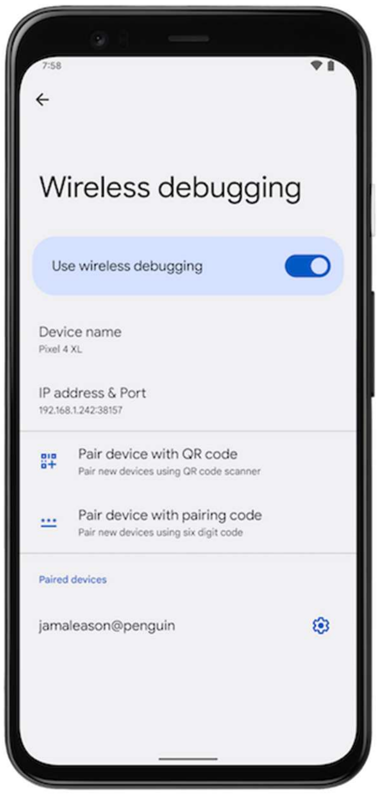
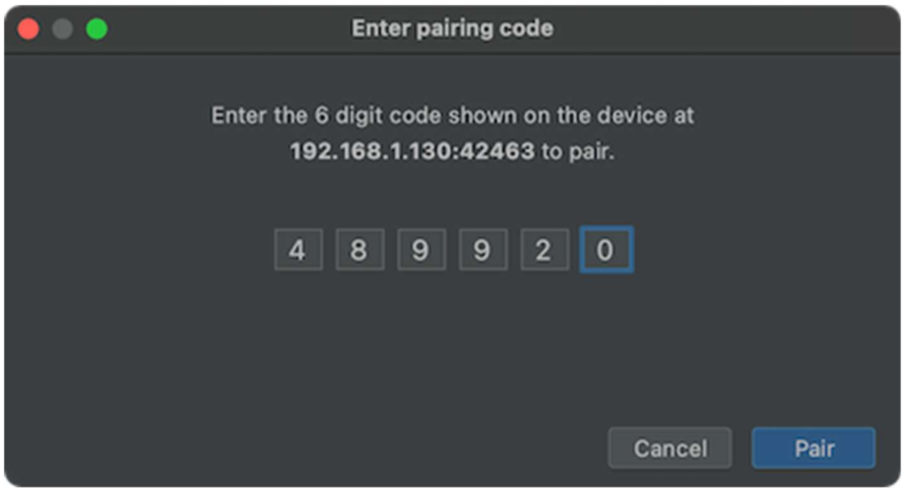


Figure 4.3c: Wireless debugging setting

4. Tap Wireless debugging and pair your device.
   - To pair device with a QR code, select Pair device with QR code and scan the QR code, shown in figure 4.1.7b.
   - To pair device with a pairing code, select Pair device with pairing code from the Pair new devices over Wi-Fi dialog. On your device, select Pair using pairing code. A six-digit code appears. Once your device appears on the Pair devices over Wi-Fi window, enter the six-digit code shown on your device and select Pair.


Enter pairing code

Enter the 6 digit code shown on the device at
192.168.1.130:42463 to pair.

Figure 4.3d: Pairing code entry

5. After pairing, you can attempt to deploy your app to your device.

To pair a new device or forget the paired device on the workstation, go to the Wireless debugging on the device, tap the workstation name under paired devices and select forget.

4.4. Test Your Environment

To test the proper functioning of Android Studio development environment, follow the following steps.

- Create New Flutter Project

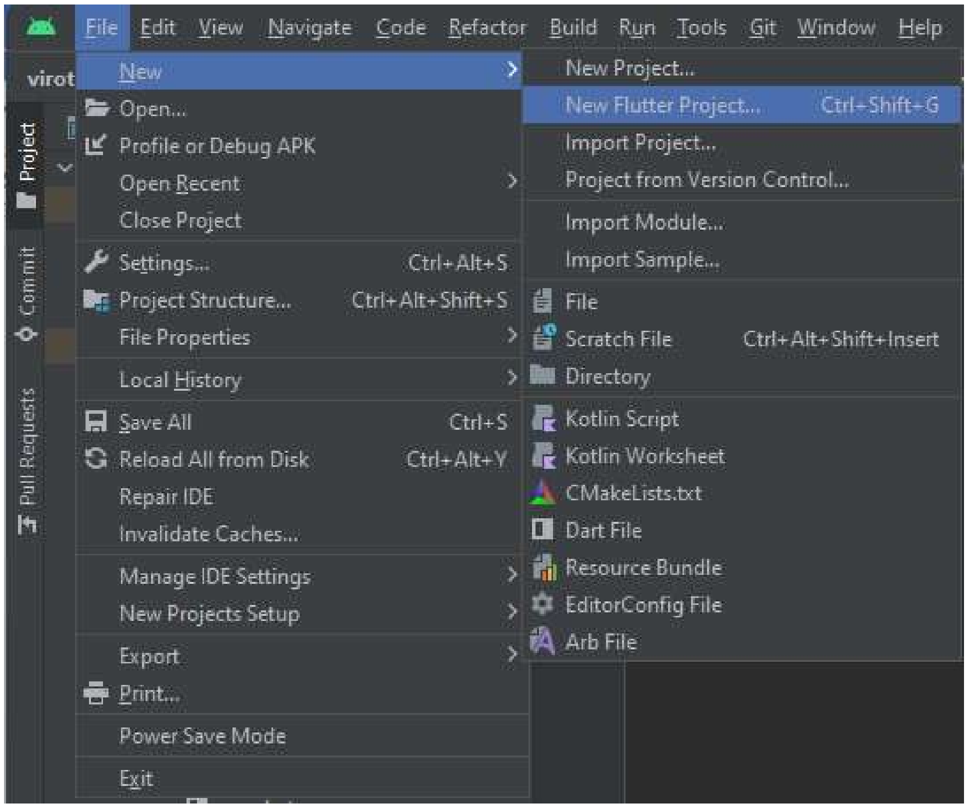
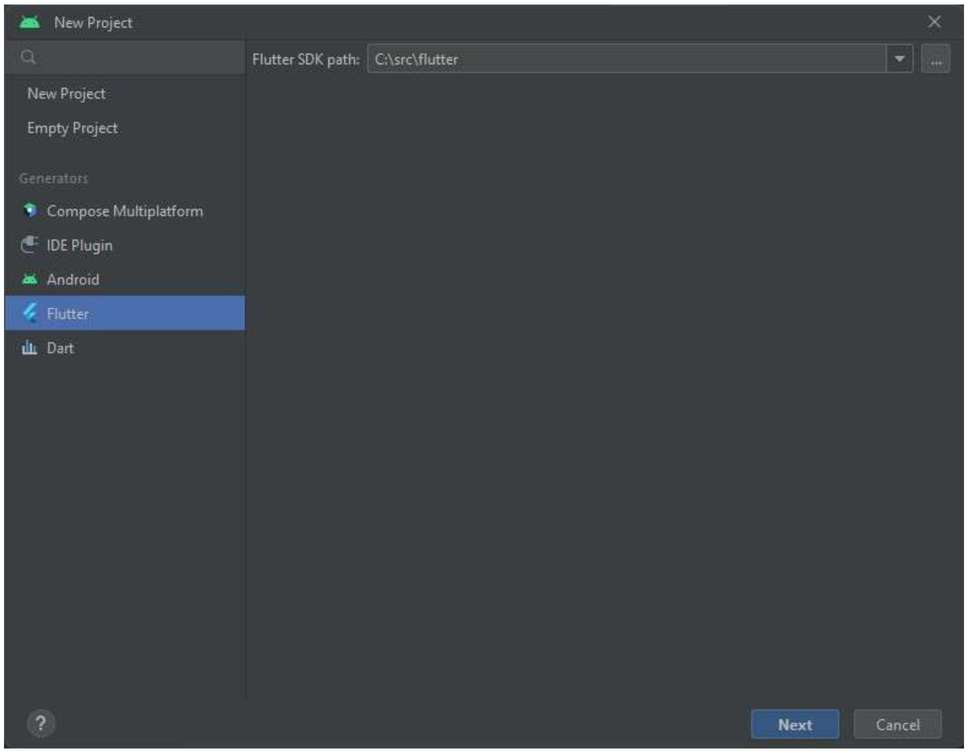


Figure 4.4a: New Flutter Project

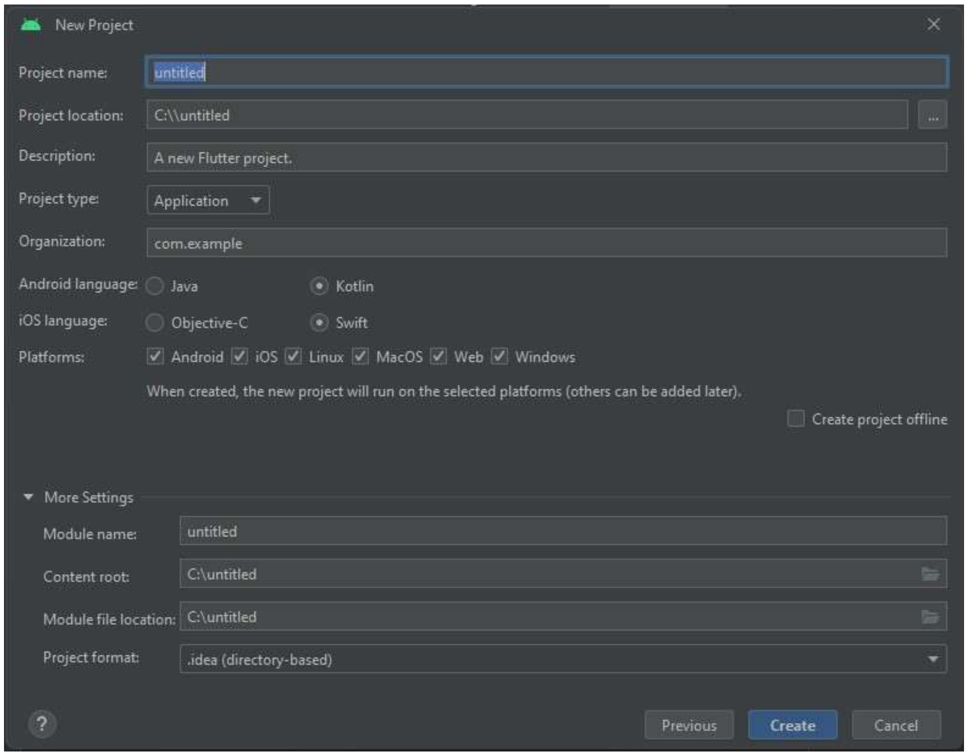
- Verify SDK path



Flutter SDK path: C:\src\flutter

Figure 4.4b: SDK path

- Enter project name and click on Create

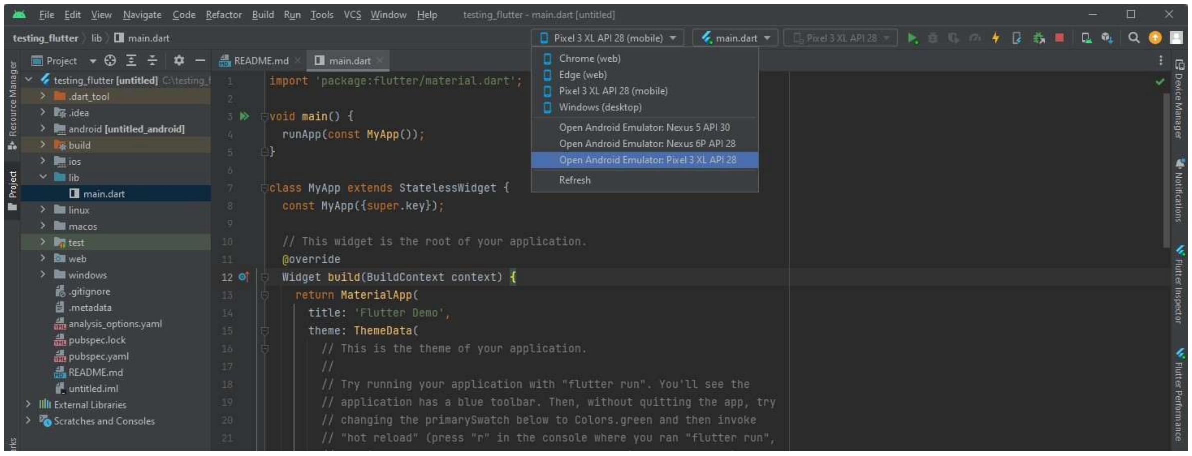


Project name: untitled
Project location: C:\\untitled
Description: A new Flutter project.
Project type: Application
Organization: com.example
Android language: Java [✓ Kotlin]
iOS language: Objective-C [✓ Swift]
Platforms: ✓ Android ✓ iOS ✓ Linux ✓ MacOS ✓ Web ✓ Windows
When created, the new project will run on the selected platforms (others can be added later).
☐ Create project offline

More Settings
Module name: untitled
Content root: C:\untitled
Module file location: C:\untitled
Project format: .idea (directory-based)

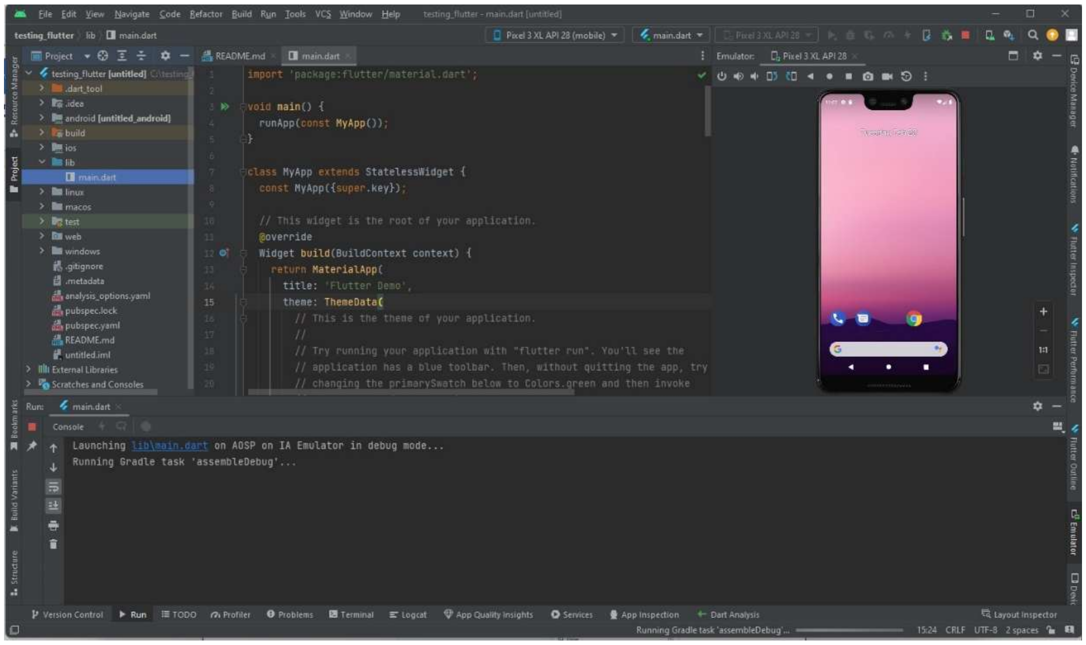
Figure 4.4c: Enter project name and Select Create

- Select virtual device


Pixel 3 XL API 28 (mobile)
Chrome (web)
Edge (web)
Pixel 3 XL API 28 (mobile)
Windows (desktop)
Open Android Emulator: Nexus 5 API 30
Open Android Emulator: Nexus 6P API 28
Open Android Emulator: Pixel 3 XL API 28
Refresh

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
```

Figure 4.4d: Select Virtual Device

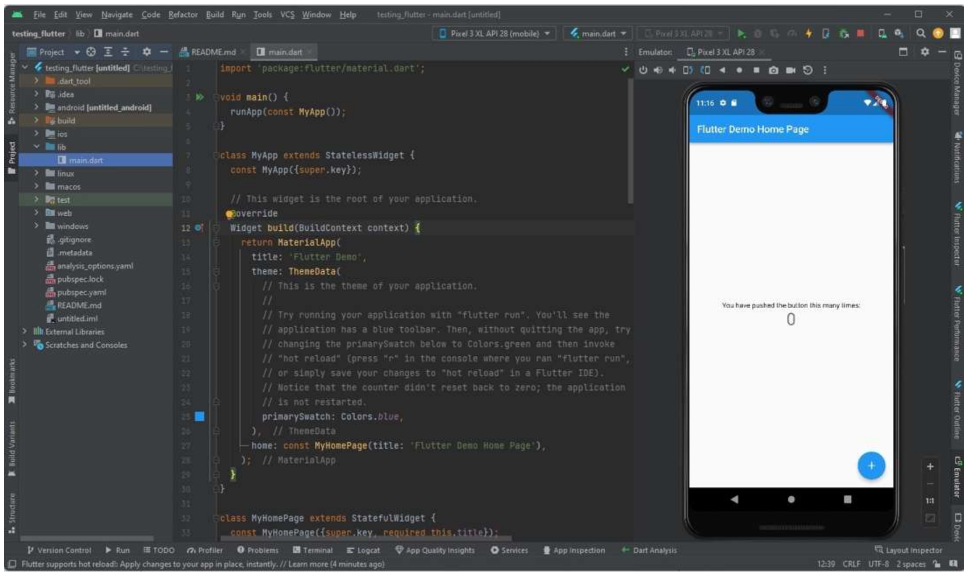


```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
```

```
Launching lib\main.dart on AOSP on IA Emulator in debug mode...
Running Gradle task 'assembleDebug'...
```

Figure 4.4e: Virtual Device appears

- Run the Flutter test project



```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
      ), // ThemeData
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    ); // MaterialApp
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
```

Flutter supports hot reload: Apply changes to your app in place, instantly. // Learn more (4 minutes ago)

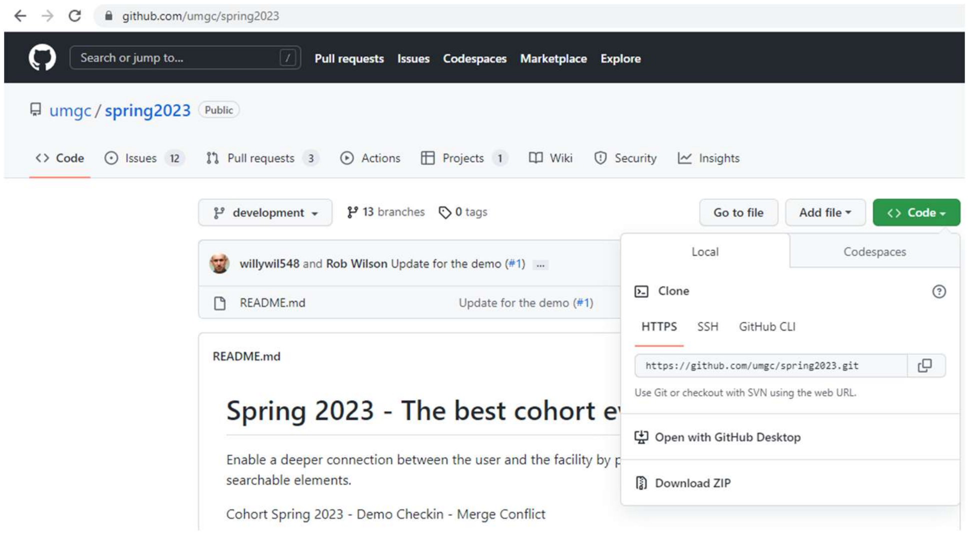
Figure 4.4f: Environment Test Successful

5. Prepare Mobile Application for Use

5.1. Cloning GitHub Repository

To clone the GitHub repository:

1. Navigate to https://github.com/umgc/spring2023
2. Click the green "Code" button
3. Select the "HTTPS" option
4. Ensure the Hypertext Transfer Protocol Secure (HTTPS) tab is selected, then click copy icon to copy the GitHub URL



willywil548 and Rob Wilson Update for the demo (#1)

README.md Update for the demo (#1)

https://github.com/umgc/spring2023.git

Use Git or checkout with SVN using the web URL.

README.md

Spring 2023 - The best cohort e

Enable a deeper connection between the user and the facility by p searchable elements.

Cohort Spring 2023 - Demo Checkin - Merge Conflict

Figure 5.1: Cloning GitHub Repository

5.2. Running the Flutter Application

1. Open Android Studio
2. Click the menu icon and select “Get from Version control”

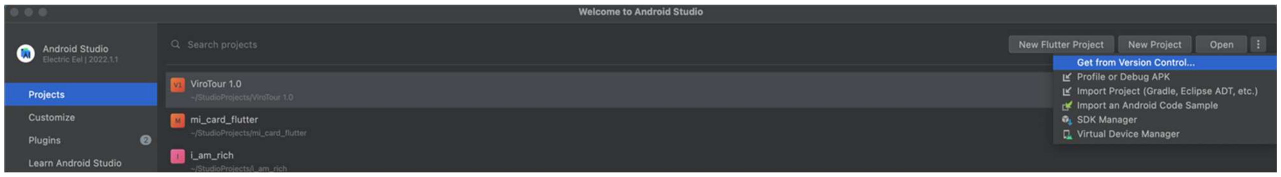
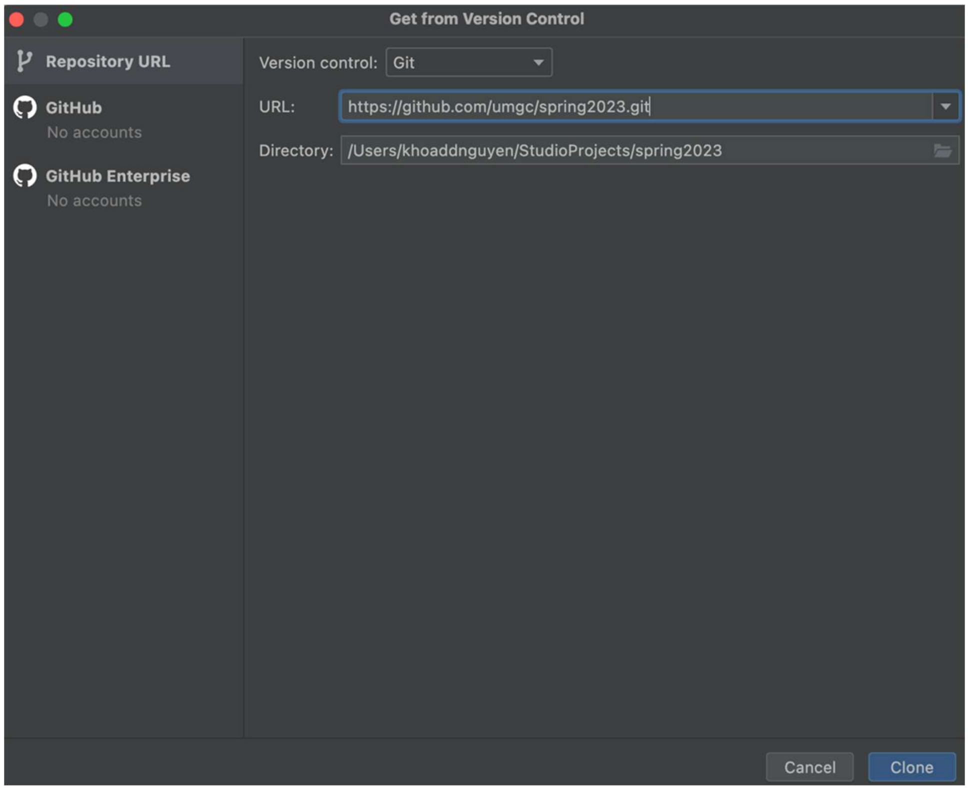


Figure 5.2a - Get from Version Control

3. Paste the copied GitHub url and hit the clone button



Version control: Git
URL: https://github.com/umgc/spring2023.git
Directory: /Users/khoaddnguyen/StudioProjects/spring2023

Figure 5.2b - Cloning the Project

4. The application should open up in Android Studio
5. Open “pubspec.yaml” file and click on the Pub get link to install referenced packages.
6. Select a previously configured emulator and click on the play button to run the application.

6. Testing the Mobile Application

6.1. Testing Objectives

The primary objectives of software testing are to ensure that the software meets its requirements, is of high quality, and is fit for the intended purpose. Some objectives of software testing are:

- Identifying defect or bugs that might affect the software's functionality, usability, and performance
- Verifying functionality working as intended and meeting the requirements specified in the design and development phases
- Improving overall software quality by identifying and removing defects; and ensuring its reliability, maintainability, and performance.
- Enhancing user experience by providing an easy, user friendly, and positive user experience.
- Implementing automated testing to control the execution of tests and comparing of actual outcomes with predicted outcomes

6.2. Testing Procedures

The ViroTour application is tested frequently during the development process. The testing methods being used are unit test, component test, integration test, system test, and user acceptance test. All chosen tests are to make sure that the application operates smoothly from start to finish. The unit tests assure the ability to function of the individual feature of ViroTour application including List Tour (Main Page, Hamburger menu, Wheel Menu), Create Tour, Edit Tour, Glow Effect, and VR View. The component test validates the functionality of a group of related units, such as a module or subsystem. The ViroTour application was designed with three components: Tour Navigation, and View Customization. The tests are designed to ensure that different parts of the software work together as expected. We perform tests for every PR that is merged. The test results are documented in the Test Report. Due to the time constraints of the project all testing was performed manually without the use of a testing suite. Developers tested each other's code systematically following the predetermined testing criteria for each use feature.

7. Troubleshooting

There are many options for developers to troubleshoot issues related to Flutter/Dart applications. Below are ViroTour's recommended troubleshooting options:

- Check the console: Make sure to check your Integrated Development Environment's (IDE) console for error messages, warnings, or other exceptions. These errors can help developers identify the root of the problem. The console can be accessed in both Android Studio, Visual Studio Code (VSCode), or your operating system's terminal.
- Use Flutter's DevTools: Android Studio and VSCode both offer Flutter plugins that provide a suite of performance and debugging tools. Some of these tools include

inspecting UI elements, diagnosing performance/networking/memory issues, and viewing general logs and diagnostic information.

- Check dependencies: Review your local environment's dependencies and ensure that all of the required packages have been installed. All of ViroTour's dependencies can be found in the *pubspec.yaml* file in the root directory. To install Flutter dependencies, run the command 'flutter pub get'.
- Clear the Flutter cache: Clearing the local development cache can sometimes help fix issues. To clear your local cache, run the command 'flutter clean' in the IDE console or your OS' terminal.
- Test on different devices: Flutter applications can be run on different devices with the same source code. For example, if your ViroTour local deployment is not working in the Android emulator environment, try running it in browser and see if the same issues appear.
- Ask for help: Flutter and Dart are both widely used software with an active and helpful community. Some options are Stack Overflow, the Flutter subreddit, or various other forums.

Below are common errors observed during the development of the ViroTour Application with the troubleshooting how-to's:

7.1. Flutter/Dart: Command Not Found

Error Message/Observation: Flutter as a binary is not in a directory where the operating system searches for it.

Solution: Add the location of the flutter binary to the $PATH variable of the operating system in use.

7.2. Git: Not in environmental variable path

Error Message: "'git' is not recognized as an internal or external command,
operable program or batch file."

Solution: Steps to set Windows PATH Environment Variables

- Left-click on the Windows Start Menu and Click on the gear icon to open windows settings.
- In the "Windows Settings" window, search for "System Environment Variable".
- Now select "Edit the system environment variables".
- Next, click the "Environment Variables" button at the bottom-right on the System Properties dialog box.
- Double-click on the "Path" entry under "System variables". If you wish to do it for yourself then double click on the "Path" entry under your User.
- Next, click on "New" button and add the following two paths C:\Program Files\Git\bin\ and C:\Program Files\Git\cmd\ to the end of the list.

- Close all open windows
- Finally, close and re-open your PowerShell or Command Prompt to reload Path variables.

7.3. Git: Not in environmental variable path after adding Environment Variable.

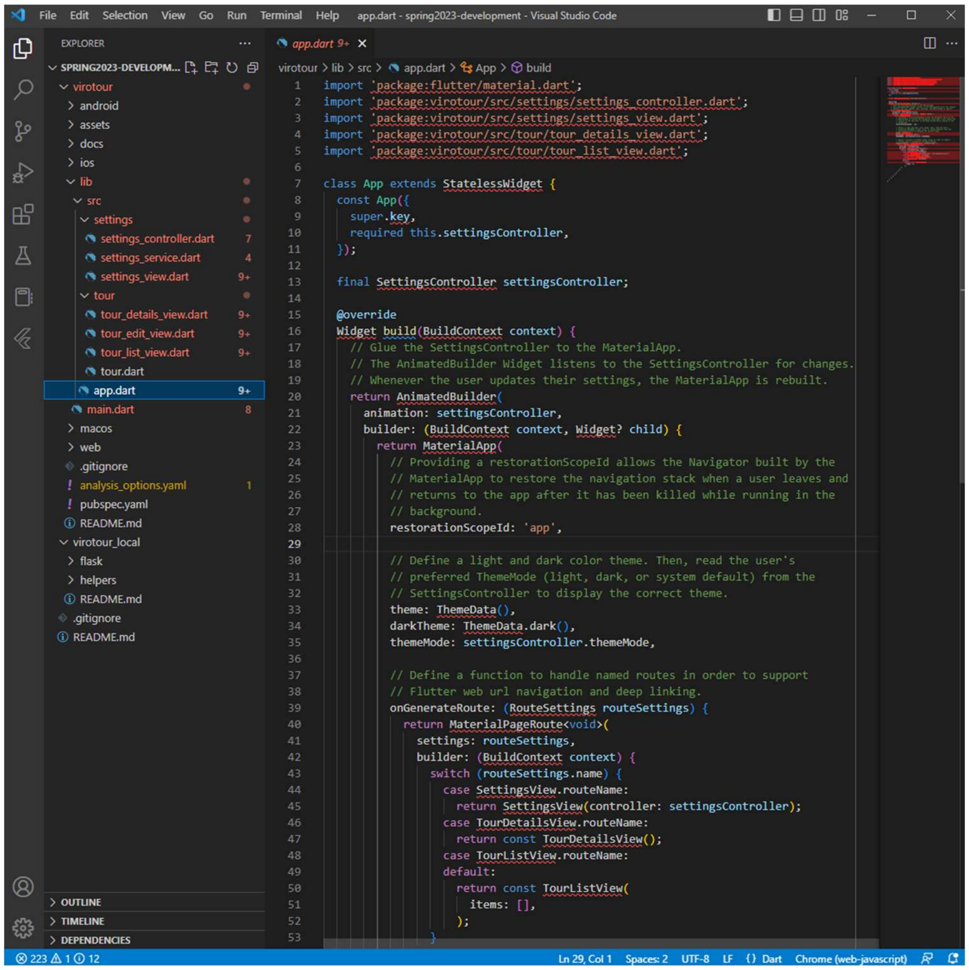
Error Message: “'git' is not recognized as an internal or external command,
operable program or batch file” after adding Git to Environmental Variable PATH. This error may occur in in CMD, PowerShell, Android Studio or VS Code.

Solution:

- Turn off malware detection on your virus protection.
- Run a given application in Administrator mode

7.4. Package Not Found

Review your local environment's dependencies and ensure that all of the required packages have been installed. All of ViroTour's dependencies can be found in the *pubspec.yaml* file in the root directory. To install Flutter dependencies, run the command 'flutter pub get'. See Section 4.4.



```dart
import 'package:flutter/material.dart';
import 'package:virotour/src/settings/settings_controller.dart';
import 'package:virotour/src/settings/settings_view.dart';
import 'package:virotour/src/tour/tour_details_view.dart';
import 'package:virotour/src/tour/tour_list_view.dart';

class App extends StatelessWidget {
  const App({
    super.key,
    required this.settingsController,
  });

  final SettingsController settingsController;

  @override
  Widget build(BuildContext context) {
    // Glue the SettingsController to the MaterialApp.
    // The AnimatedBuilder Widget listens to the SettingsController for changes.
    // Whenever the user updates their settings, the MaterialApp is rebuilt.
    return AnimatedBuilder(
      animation: settingsController,
      builder: (BuildContext context, Widget? child) {
        return MaterialApp(
          // Providing a restorationScopeId allows the Navigator built by the
          // MaterialApp to restore the navigation stack when a user leaves and
          // returns to the app after it has been killed while running in the
          // background.
          restorationScopeId: 'app',

          // Define a light and dark color theme. Then, read the user's
          // preferred ThemeMode (light, dark, or system default) from the
          // SettingsController to display the correct theme.
          theme: ThemeData(),
          darkTheme: ThemeData.dark(),
          themeMode: settingsController.themeMode,

          // Define a function to handle named routes in order to support
          // Flutter web url navigation and deep linking.
          onGenerateRoute: (RouteSettings routeSettings) {
            return MaterialPageRoute<void>(
              settings: routeSettings,
              builder: (BuildContext context) {
                switch (routeSettings.name) {
                  case SettingsView.routeName:
                    return SettingsView(controller: settingsController);
                  case TourDetailsView.routeName:
                    return const TourDetailsView();
                  case TourListView.routeName:
                  default:
                    return const TourListView(
                      items: [],
                    );
                }
```

8. Deploying the Mobile Application

Flutter application can be deployed to both android and iOS platforms. In an android application, the following steps would be carried out:

- The application name would be changed using the android:label entry in the android manifest file known as AndroidManifest.xml. This file is located in the <app>/android/app/src/main. This file contains the details about the android application.
- The launcher icon would be changed using the android icon entry in the manifest file.
- Sign the application using the standard option
- When necessary, enable Proguard and Obfuscation
- Create a release APK fie using the command,

```
# cd /path/the/application
# flutter build apk
```

- The APK will then be installed using the command

```
# flutter install
```

- Publish the application into Google playstore. This would be created using an appbundle and pushed into the playstore using the command,

```
# flutter build appbundle
```

To deploy to iOS applications;

- The iOS application will be registered in the App Store Connect using standard method. The -=Bundle ID used for registering the application should be saved.
- Update the display name in the XCode project setting to set the application name.
- Update the Bundle Identifier in the XCode project setting to set the bundle id.
- Code sign using standard method.
- Add a new app icon using standard method
- Generate IPA file using the command,

```
# flutter build iOS
```

- The application should be tested by pushing the application, IPA (iOS Package App) file into TestFlight using standard methods.
- The application will be pushed into the App Store using standard methods.

ViroTour is available for mobile deployment on both Android and iOS. To build/deploy ViroTour to either environment, the developer must follow the following steps:

- Configure your development environment: Before deploying the ViroTour application, make sure that the development environment is set up correctly. This includes installing Flutter to the local system, configuring the OS to use Flutter's environment variables, and installing other necessary software, depending on the OS.
- Test the application locally: Ensure that the application runs locally on an emulated device. These tests should be looking for performance, usability, and compatibility. Any issues should be fixed before attempting to build the application.
- Create a release build: To deploy the mobile application a release build must be created. A release build is better optimized than the local development build and much smaller in size. To create a release build for iOS, run the command 'flutter build ios'. To create a release build for Android, run the command 'flutter build apk'.
- A developer account is required to install unsigned applications to an Apple iOS device. The application must then be uploaded to Apple TestFlight to utilize the created IPA file on the Apple device. A developer account is not required for Android devices, but the

"Allow Unknown Sources" option in Settings – Security must be enabled to install the ViroTour APK file.

9. Appendix A - Software Installation Checklist

1	Install git on your computer: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
2	Install Flutter on your computer. - Windows: https://docs.flutter.dev/get-started/install/windows - MacOS: https://docs.flutter.dev/get-started/install/macos
3	Install one of the following IDEs. - Android Studio: https://developer.android.com/studio - VSCode: https://code.visualstudio.com/download
4	Install Flutter and Dart Plugins/Extensions in your IDE.
5	Install dependencies manually in the *pubspec.yaml* file located inside the app folder if the code editor does not automatically install them.
6	Setup Android Emulator by creating an Android Virtual Device (AVD)
7	Clone the project GitHub repository: https://github.com/umgc/spring2023
8	Follow the repository instruction: https://github.com/umgc/spring2023/tree/development/virotour Choose one of the following environments: Chrome web browser, Android, iOS. Start the environment. Run the Flutter application.