




Deployment and Operation Guide (Runbook

TEAM: CHARLIE

Presented by: Michael Le, Debashis Jena, Austin Johnson, Prince Antwi Aboagye,
Didimus Kimbi, Damion Sevilla

SWEN 670 – SOFTWARE ENGINEERING PROJECT
JULY 23, 2021
REVISION 2.0



Project name: Mnemosyne, Disability Mobile Application

Date: July 23, 2021

Project Leader: Michael Le

Phase: Design & Engineering and Execution

For approval: Michael Le

Michael le Date: 07/23/2021

For approval: Dr. Mir Mohammed Assadullah

_____ Date: 07/23/2021

Revision History

Version Number	Date	Description	Author
1.0	07/23/2021	Initial Deployment and Operations Guide Release	Team Charlie
2.0	08/06/2021	Updates to the document	Debashis Jena, Austin Johnson

Table of Contents

1. Introduction	4
1.1 Purpose	4
1.2 Intended audience and reading suggestions	4
1.3. Technical Project Stakeholders	4
1.4 References	5
1.5 Definitions, Acronyms, and Abbreviations	6
2. Mobile Application.....	8
2.1 Features, Packages, Plugins, and Widgets.....	8
2.1.1 Features.....	8
2.1.2 Packages	8
2.1.3 Plugins	9
3 Software Installation.....	11
3.1 Flutter and Dart.....	11
3.2 Android Studio	14
3.3 Android Emulator	19
4. Prepare the Mobile Application for Use.....	23
4.1 Cloud Speech-to-Text setup	23
5 Testing the Mobile Application.....	32
5.1 Testing Objective.....	32
5.2 Testing Procedures	33
5.3 Possible Software Issues in the Testing Process	34
6 Troubleshooting.....	36
6.1 Issue installing Flutter on MacOS	36
6.2 Emulator does not respond or slow	36
6.3 Unable to launch Emulator from Editor	38
6.4 Slow Running the App (Stuck at Running Gradle task 'assembleDebug').....	38
6.5 Dependency Issues	39

1. Introduction

1.1 Purpose

The purpose of this "Deployment and Operations Guide (Runbook) is to illustrate the details and outline steps that can deploy the Mnemosyne application.

1.2 Intended audience and reading suggestions

The audience for this document is mostly service professionals, software developers, students, and whoever is concerned about the application. It is intent on showing ease for the user to use. There should be no question when operators or users walk through this document since it shows specific step detail from point A to Z.

1.3. Technical Project Stakeholders

The table below shows a list of stakeholders involves in the project for the Mnemosyne Mobile Application:

Table 1 - Project Stakeholders

Name	Role
Dr. Mir Assadullah	Stakeholder (Project Owner)
UMGC Professors	Software Engineer Staff Department
Michael Le	Project Manager
Debashis Jena	Lead Developer
Austin Johnson	Developer
Didimus Kimbi	Developer

Damon Sevilla	Tester / CO-PM
Prince Aboagye	Business Analyst

1.4 References

Table 2 - Referenced Documents

Title	Reference
Install Flutter	http://flutter.dev
Android Studio additionally requires that a Java Development Kit (JDK) install	https://www.oracle.com/java/technologies/javase-downloads.html
Download Android Studio	https://developer.android.com/studio
Configure hardware acceleration for the Android Emulator	https://developer.android.com/studio/run/emulator-acceleration
Cloud Speech-to-Text setup	https://cloud.google.com/speech-to-text https://cloud.google.com/speech-to-text/docs/reference/rest/?apix=true
Clone the GitHub repository github.com/umgc/summer2021.charlie	https://github.com/umgc/summer2021.charlie

1.5 Definitions, Acronyms, and Abbreviations

Table 3 - Definitions, Acronyms, and Abbreviations

Acronyms and Abbreviations	Definitions
AVD	Android Virtual Device
APK	Android Application Package
AMD	Advanced Micro Devices
iOS	iPhone operating system
RAM	Random Access Memory
Flutter CLI	Flutter Command Line Tool
CPU	Central Processing Units
VM	Virtual Machine
UI / API	User Interface / Application Programm Interface
SRS	Software Requirement Specification
UMGC	University Maryland Global Campus
IDE	Integrated Development Environment
tts	Text to speech
SDK	Software Development Kit

ENV	Environment Variables
JDK	Java Development Kit

2. Mobile Application

2.1 Features, Packages, Plugins, and Widgets

2.1.1 Features

Easy Navigation - Simpler the design of the app, it is easier for users to navigate. The interface of the Mnemosyne app is organized and has easy-to-understand navigation, which helps enhance the user experience.

Android and IOS Devices Support - One best way to get customers' attention is to provide mobile apps compatible on multiple or many platforms. This feature will provide users with options on when and how to access the Mnemosyne app.

Usability/Learnability - The amount of effort and time required to learn the Mnemosyne software is less, and this makes the software user-friendly even for IT illiterate people.

Portability - Mnemosyne app can perform the same functions across all environments and platforms.

2.1.2 Packages

- Flutter_local_auth - contains the service to protect personal information using biometric as per the device.
- Flutter_tts(text to speech)- contains the service to convert text to speech
- Path_provider – navigate to different paths in the app's memory
- flutter encrypt – this service encrypts and decrypts the text using a secret key
- avatar_glow – display glow effect on buttons when it is clicked or pressed
- work manager – to run batch jobs in the background. This plugin allows you to schedule background work on Android and iOS.
- rxdart – streaming services efficiently

- `sound_stream` - a flutter plugin for streaming audio data from Mic and data to the Audio engine without using a file.
- `audio players`– use to display an audio player to playback the audio
- `sound_recorder` – a library that provides utility methods to record a user's voice.
- `google_speech` – use for speech-to-text transcription
- `sound_stream` – to stream mic audio to external services
- `highlight_text` – to display the saved vocabulary texts in a highlighted format

2.1.3 Plugins

Flutter and Dart Plugin installation

2.1.3.1 *Windows and Linus Install*

1. Open Android studio
2. Click on Configure.
3. Select Plugins
4. On the marketplace section, search flutter on the "Type/to see options" search bar.
5. Click install beside the Flutter.
6. Read and accept the third-party privacy notice.
7. Click "Yes" to install the required plugins.
8. Once's Flutter is done installing, click on "Restart IDE" to restart android studio.

2.1.3.4 *macOS Install*

1. Open Android studio
2. Click on Configure.

3. Select Preferences

3. Select Plugins

4. On the marketplace section, search flutter on the "Type/to see options" search bar.

5. Click install beside the Flutter.

6. Read and accept the third-party privacy notice.

7. Click "Yes" to install the required plugins.

8. Once's Flutter is done installing, click on "Restart IDE" to restart android studio.

2.2.4 Widgets

- audio_recorder widget – records the voice of the user to save it as a sample wave file.
- Stateful widget – manage states of the component without having to traverse between the server-side to client-side.
- GetHomeWidget– checks if an audio file exists.
- Stateless widget - for text and titles
- audio_recognize widget - displays the Mic, which changes the icon from Mic to stop based on the current state.

3 Software Installation

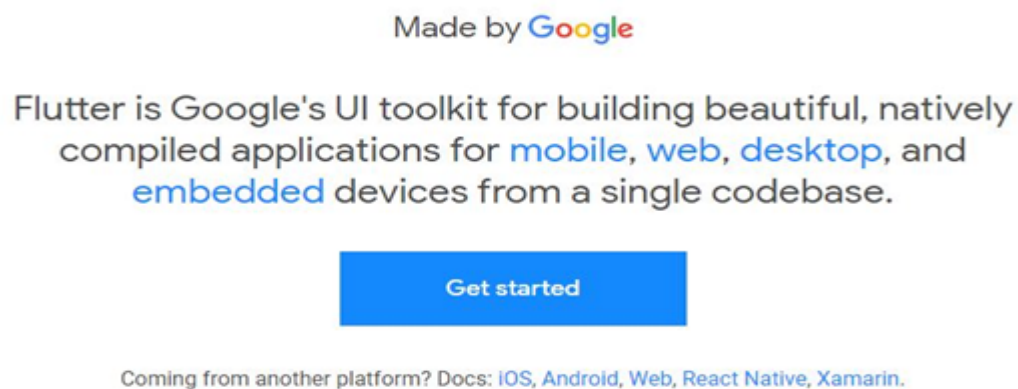
3.1 Flutter and Dart

Flutter System Requirements

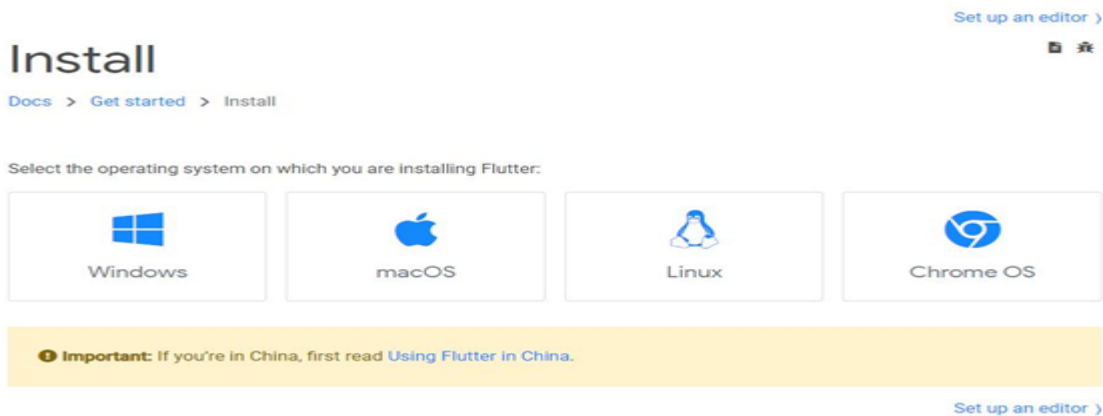
- Operating Systems: Windows 7 SP1 or later (64-bit), x86-64 based.
- Disk Space: 1.64 GB (Does not include disk space for IDE/tools).
- Tools: Flutter depends on these tools being available in your environment.
- Windows PowerShell 5.0 or newer. (Pre-installed with Windows 10)
- Git for Windows 2.x with the Use Git from the Windows Command Prompt option.

The following instructions assume a Windows environment, but the steps are essentially the same for other supported operating systems.

1. Before starting, ensure that Windows PowerShell 5.0 and Git for Windows 2.x are both installed.
2. To download Flutter, first, navigate to <http://flutter.dev> in your web browser.
3. Click the "Get started" button.



4. Select "Windows" as your operating system.



5. Download the Flutter SDK zip file by using the blue button below "Get the Flutter SDK."

Get the Flutter SDK

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

[flutter_windows_2.2.3-stable.zip](#)

For other release channels, and older builds, see the [SDK releases](#) page.

2. Extract the zip file and place the contained `flutter` in the desired installation location for the Flutter SDK (for example, `C:\Users\<your-user-name>\Documents`).

Warning: Do not install Flutter in a directory like `C:\Program Files\` that requires elevated privileges.

If you don't want to install a fixed version of the installation bundle, you can skip steps 1 and 2. Instead, get the source code from the [Flutter repo](#) on GitHub, and change branches or tags as needed. For example:

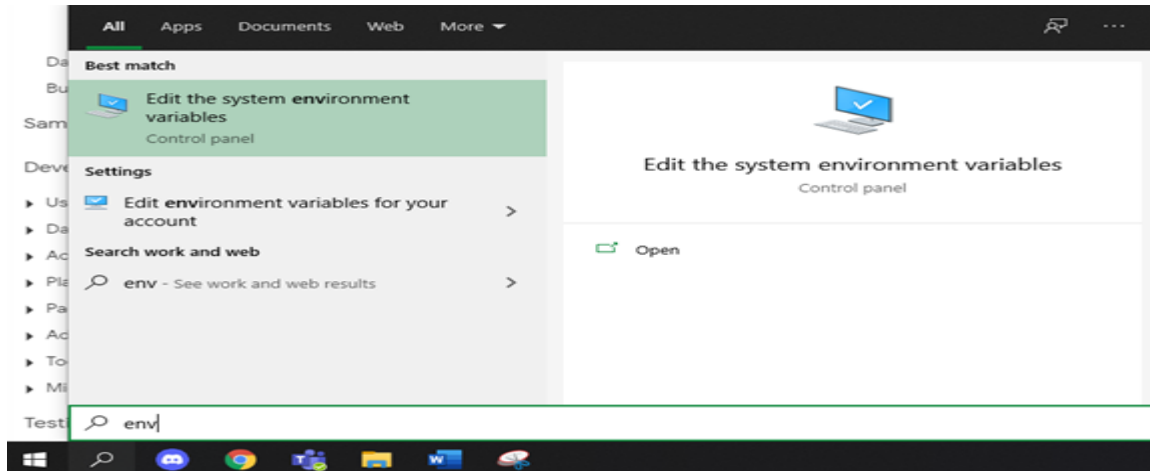
```
C:\src>git clone https://github.com/flutter/flutter.git -b stable
```

You are now ready to run Flutter commands in the Flutter Console.

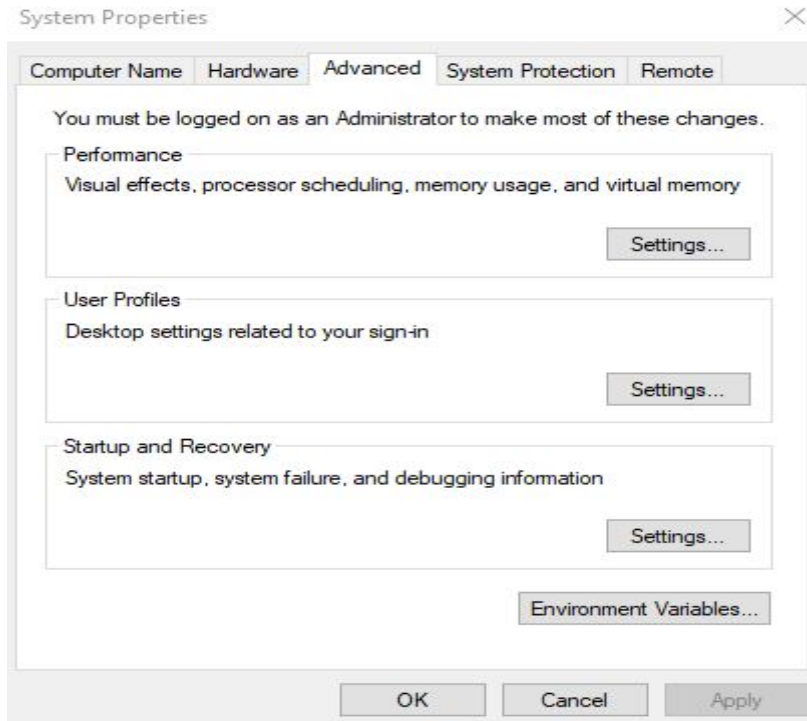
6. After the file download, navigate the file on your machine and unzip it to your desired location.

At this point, you are capable of executing flutter commands by running the `flutter_console.bat` inside the flutter directory you just extracted. If you would like to run flutter commands in the normal windows command prompt, continue.

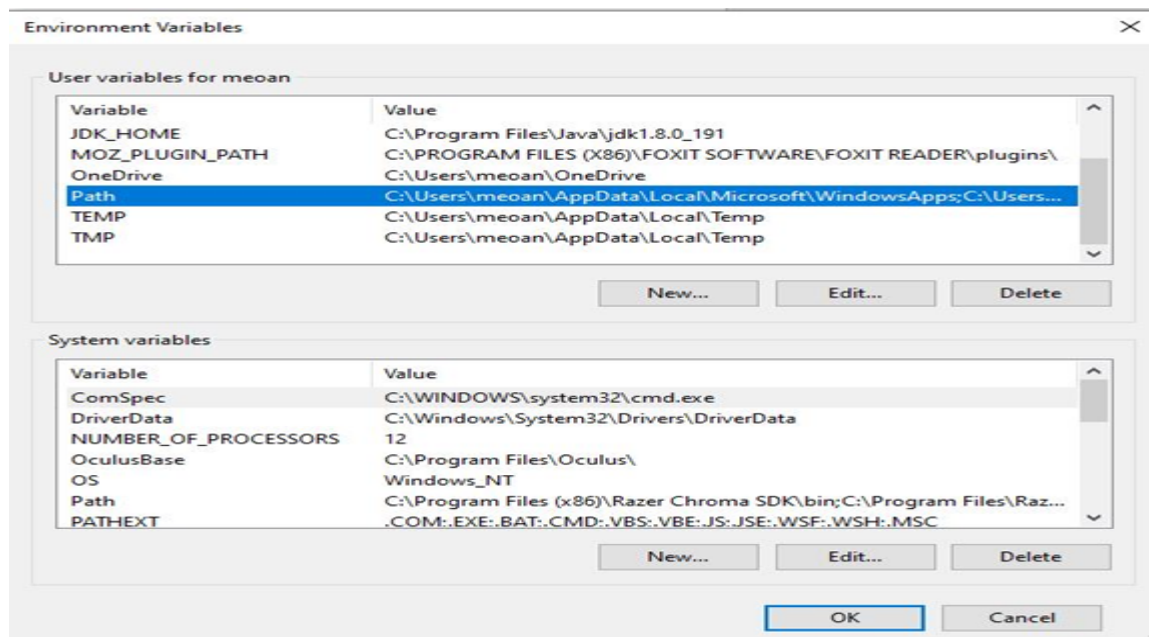
1. Using the Start search bar, search for "env" and select "Edit environment variables for your account."



2. Click the "Environment Variables..." button on the window that comes up.

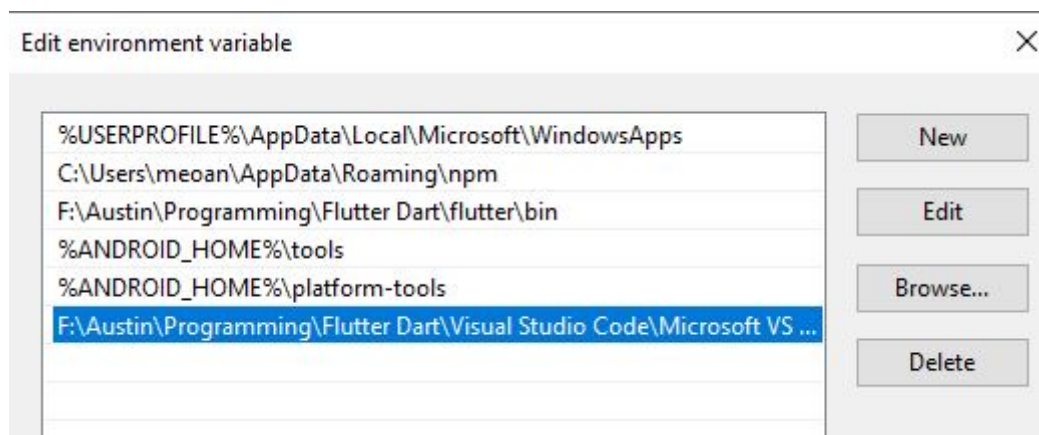


3. Under User variables at the top of the new window, click on the Path variable entry and then click on the "Edit..." button below it.



4. Click the "New" button and enter the directory path to the bin sub-folder in the Flutter SDK directory you previously extracted.

To find this path, navigate to the Flutter SDK folder previously extracted, open the bin folder, then copy the file path at the top.



5. Open the command prompt and enter "flutter doctor" to confirm that it works and see if any dependencies are missing.

3.2 Android Studio

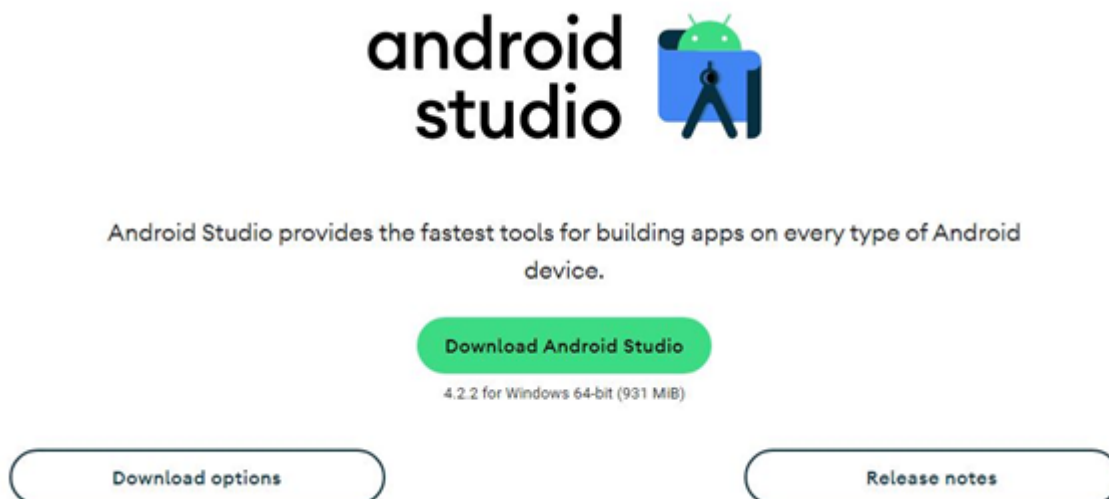
Android Studio System Requirements

- 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
- 8GB RAM or more
- 8GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

Android Studio additionally requires that a Java Development Kit (JDK) install. Visit <https://www.oracle.com/java/technologies/javase-downloads.html> to download and install a JDK if necessary.

The following instructions assume a Windows environment, but the steps are essentially the same for other supported operating systems:

1. In your web browser, navigate to <https://developer.android.com/studio>
2. Click on the "Download Android Studio" button near the top.



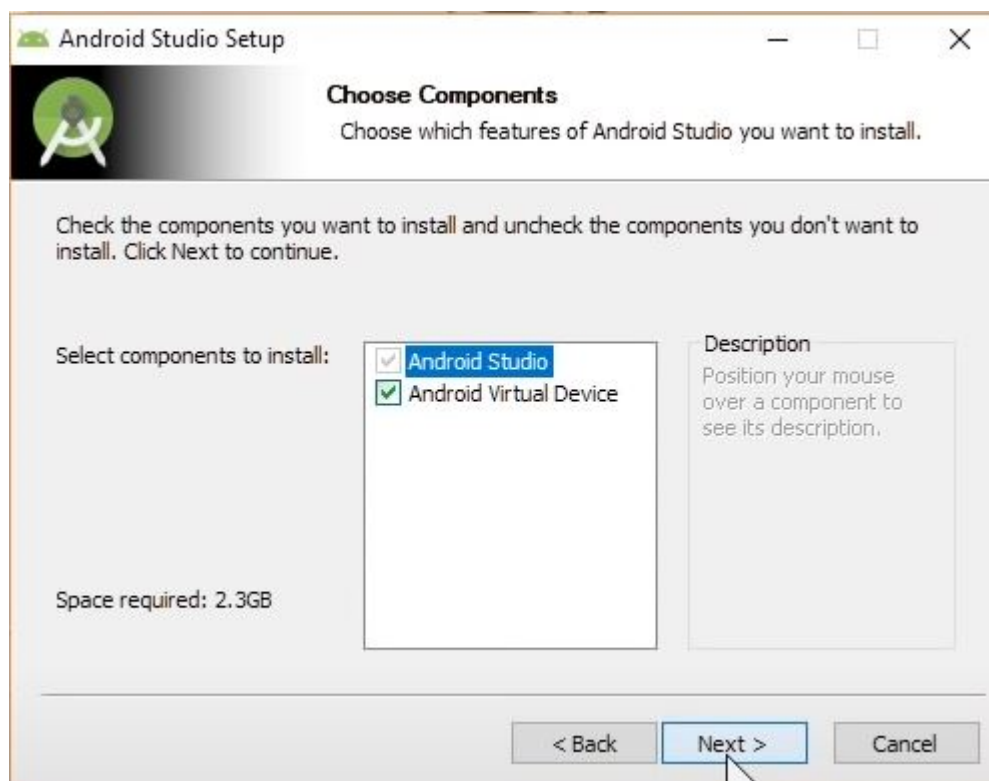
3. Look through the Terms and Conditions, scroll to the bottom, click the check box to confirm that you have read the above, and then click the "Download Android Studio for Windows" button.

☒ I have read and agree with the above terms and conditions

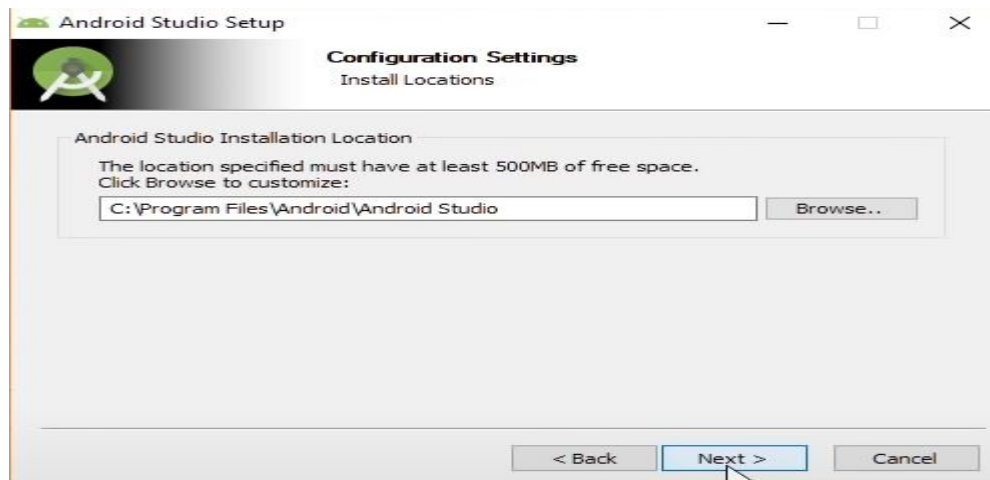
Download Android Studio for Windows

android-studio-ide-202.7486908-windows.exe

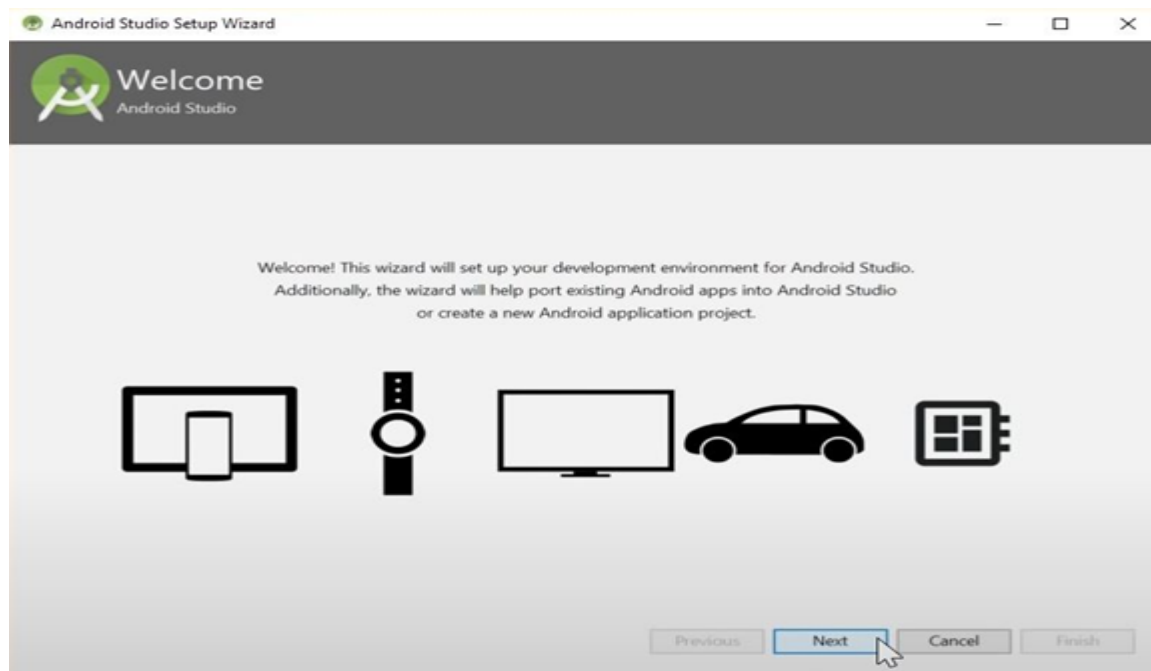
4. Once the file download, execute the executable file to begin the installation.
5. Click "Next" until the Choose Components screen, where you will have to check the box next to "Android Virtual Device."



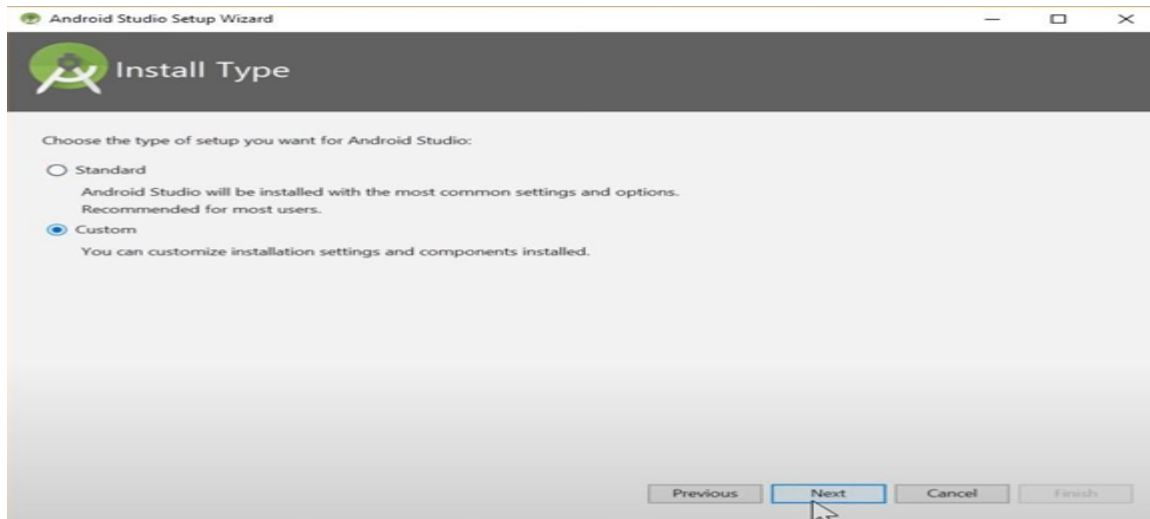
6. Again, click "Next" until you get to the Installation Location screen, where you will enter the location that you want Android Studio to install.



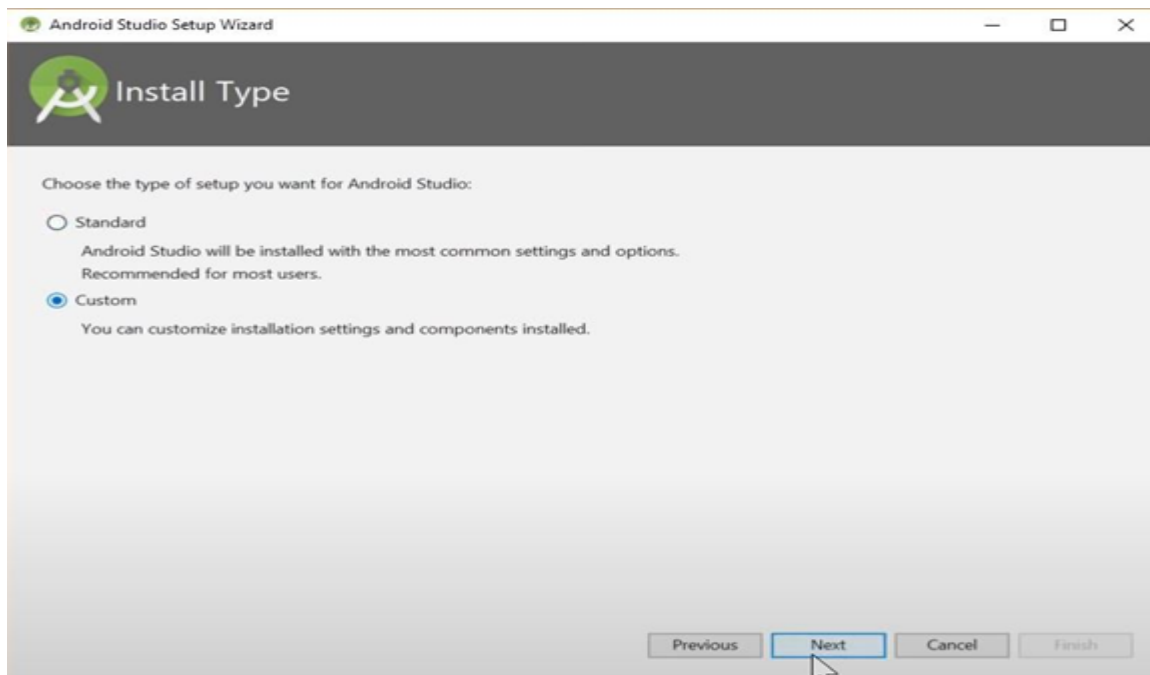
7. Click "Next" until the application install. This process may take several minutes.
8. Once installed, start Android Studio by clicking on the shortcut or searching for it using the Start search bar.
9. Click "Next" on the Welcome screen.



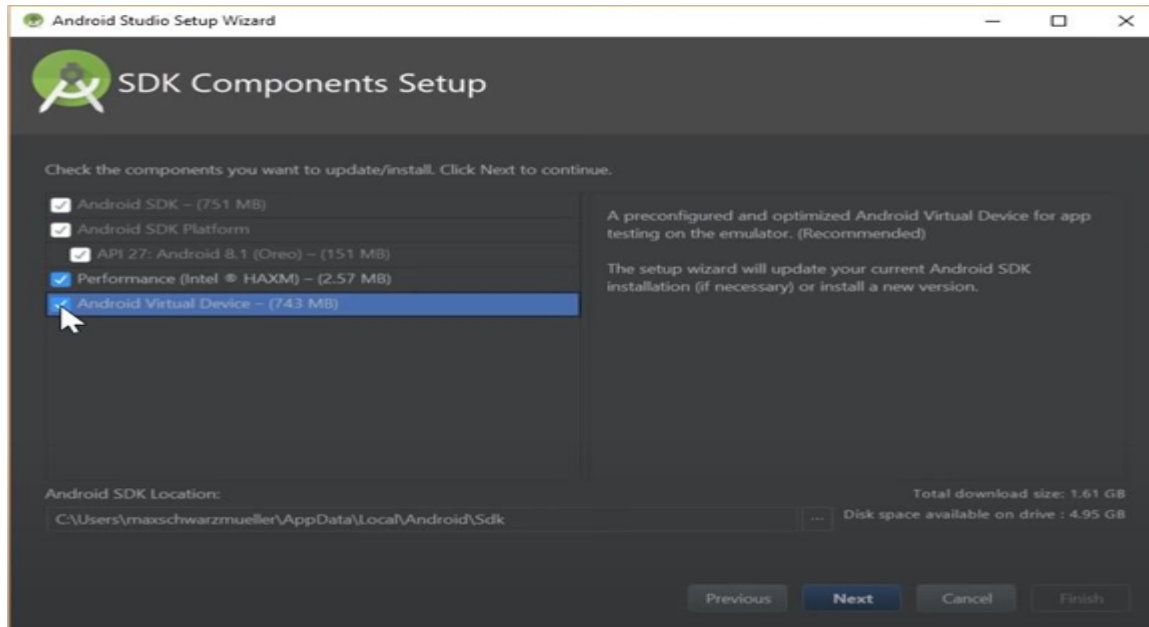
10. On the Install Type screen, click the radio button next to Custom and then click "Next."



11. Select the UI Theme that you prefer, and click "Next."



12. On the SDK Components Setup screen, click the checkbox next to "Android Virtual Device" and then click "Next."



13. Select the location that you want the Android SDK to install, and then click "Next."

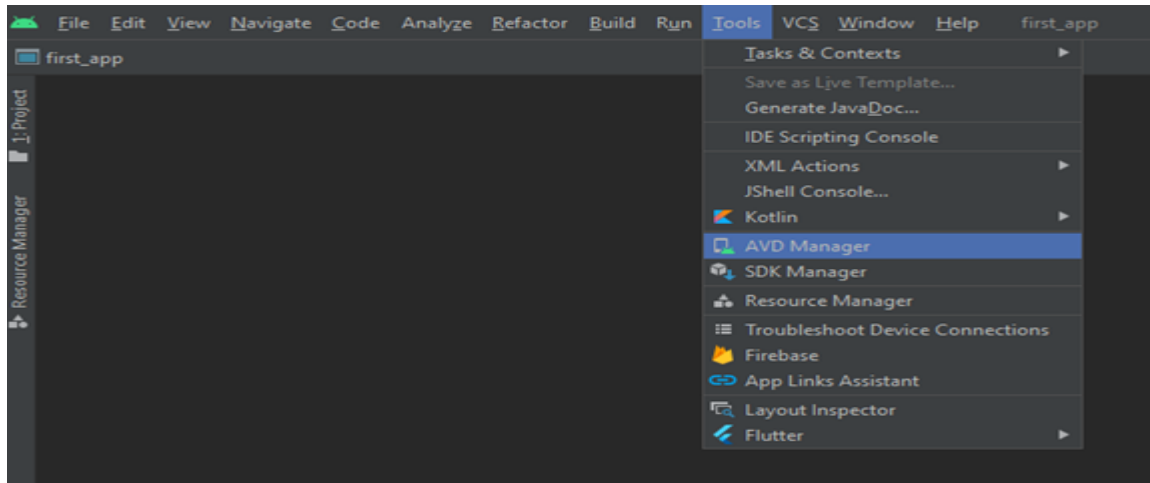
14. Click "Finish" to begin installing Android Studio. This process may take several minutes.

3.3 Android Emulator

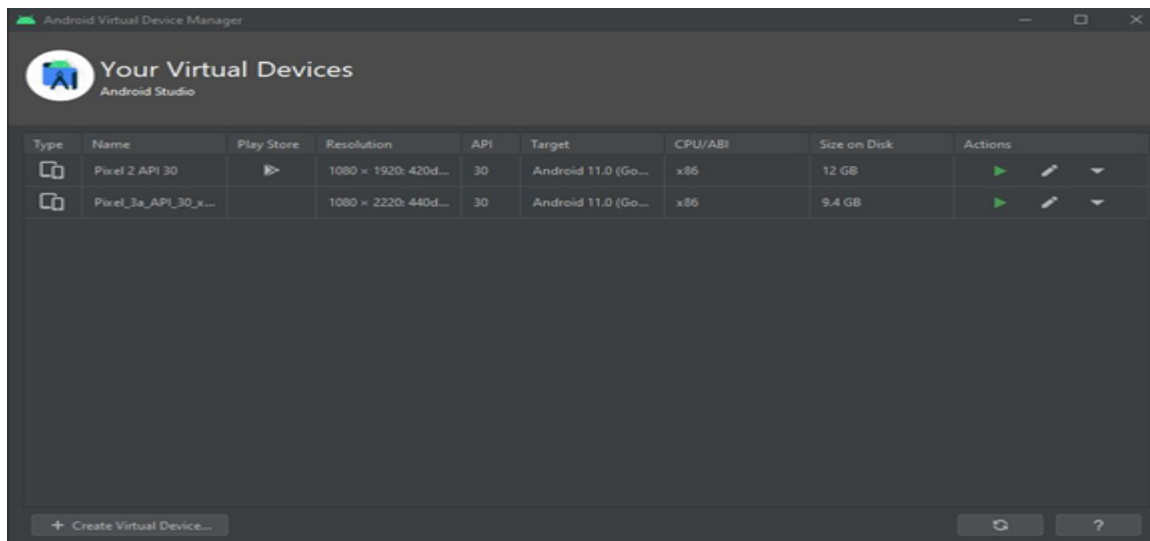
Before the Android Emulator could set up, you need to make sure that VM acceleration enables on your computer. Use the link below if you need help doing so.

<https://developer.android.com/studio/run/emulator-acceleration>

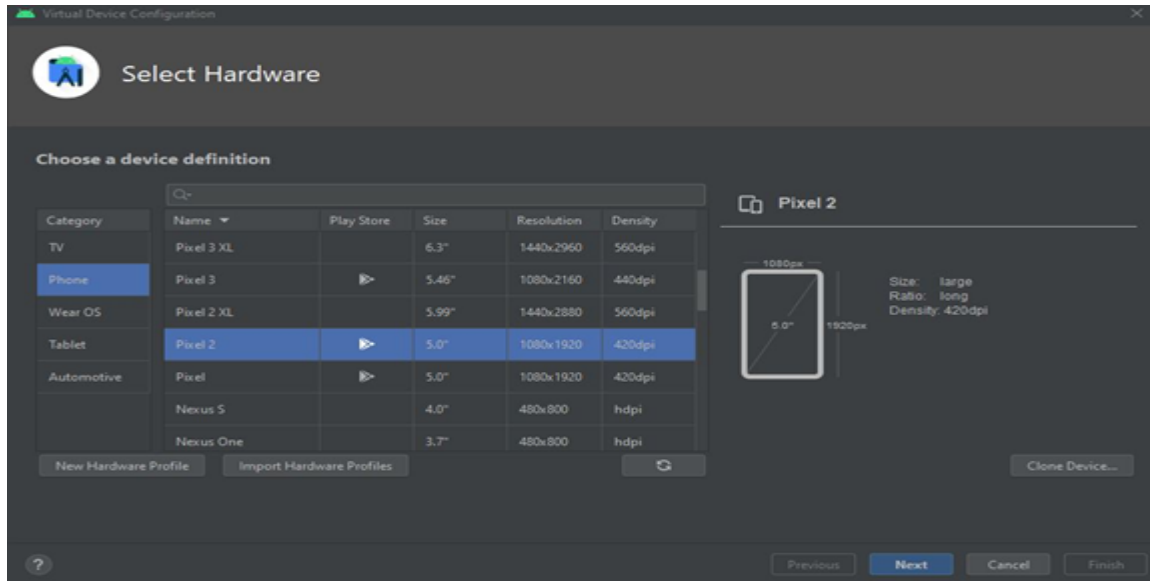
1. Open Android Studio
2. On the taskbar at the top, click Tools and then AVD Manager.



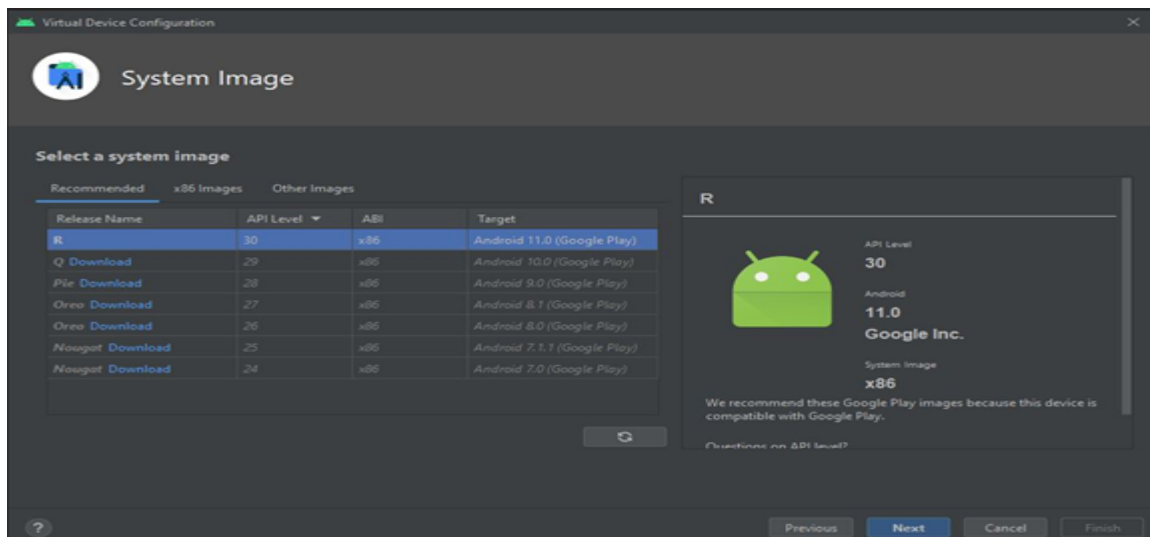
3. In the new window that comes up, click "Create Virtual Device..." at the bottom.



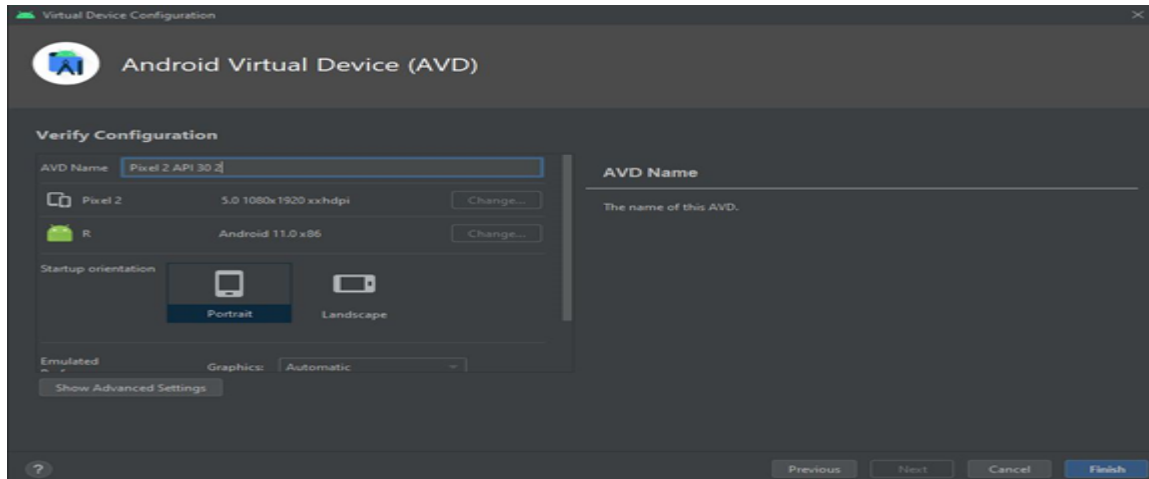
4. On the Select Hardware screen, select Phone on the left and then Pixel 2 in the center, then click "Next."



- On the System Image screen, select the topmost entry in the Recommended tab named "R" and then click "Next."



- On the Android Virtual Device (AVD) screen, type a name for your emulator that you can remember and then click "Finish."



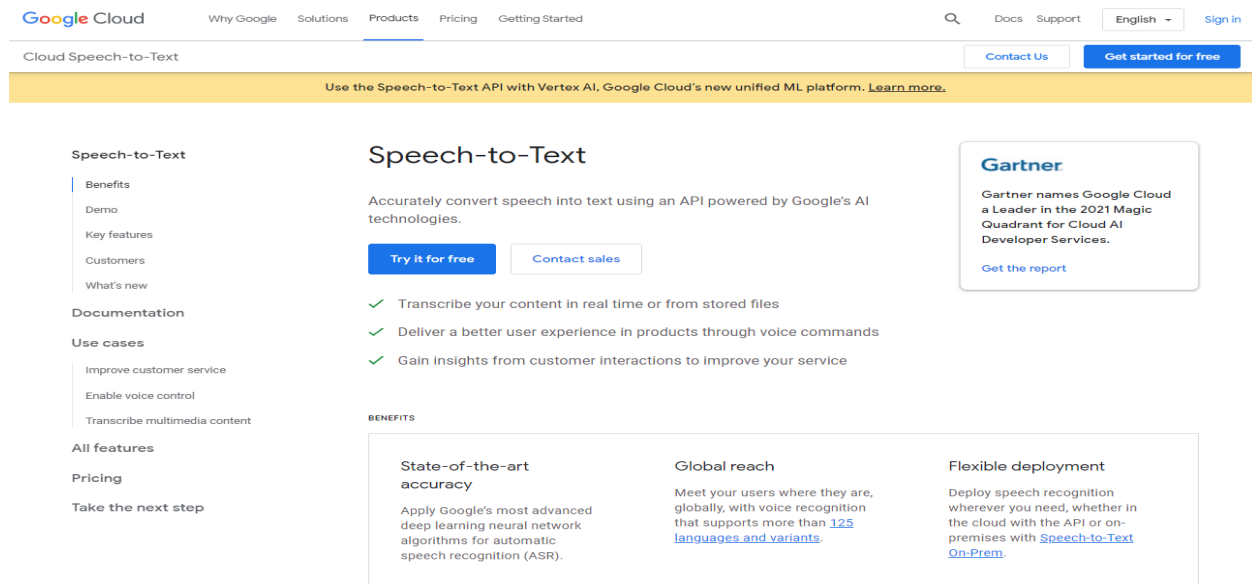
To open this emulator from the command prompt, follow these instructions:

1. Open command prompt
2. Using the `cd` command, navigate to the Android SDK folder that you chose earlier, and then to the "emulator" folder inside of it.
3. Use the command "emulator" followed by your emulator's name.
4. The formatting of your emulator's name is, to begin with, the @ character, followed by the emulator name you entered previously. Be sure to use underscores (_ characters) instead of spaces.
5. For example, if you named your emulator Pixel 2 API 30, use the command "emulator @Pixel_2_API_30."

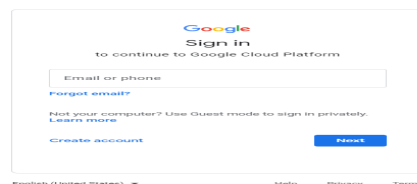
4. Prepare the Mobile Application for Use

4.1 Cloud Speech-to-Text setup

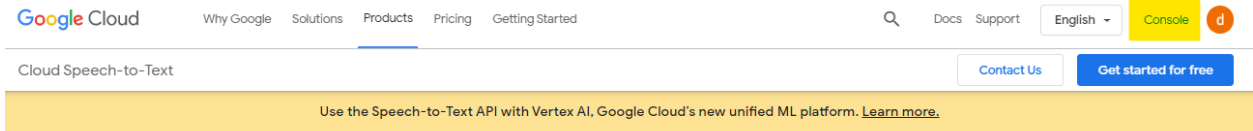
1. Open a browser and go to <https://cloud.google.com/speech-to-text>



2. Click the circle in the "Get started" at the top right-hand corner of the screen to open the login screen. From here, enter or create your Google account credentials and log in.



3. Once logged in, click the "Go to console" button in the top right corner of the window



Speech-to-Text

- Benefits
- Demo
- Key features
- Customers
- What's new

Speech-to-Text

Accurately convert speech into text using an API powered by Google's AI technologies.

[Try it for free](#)

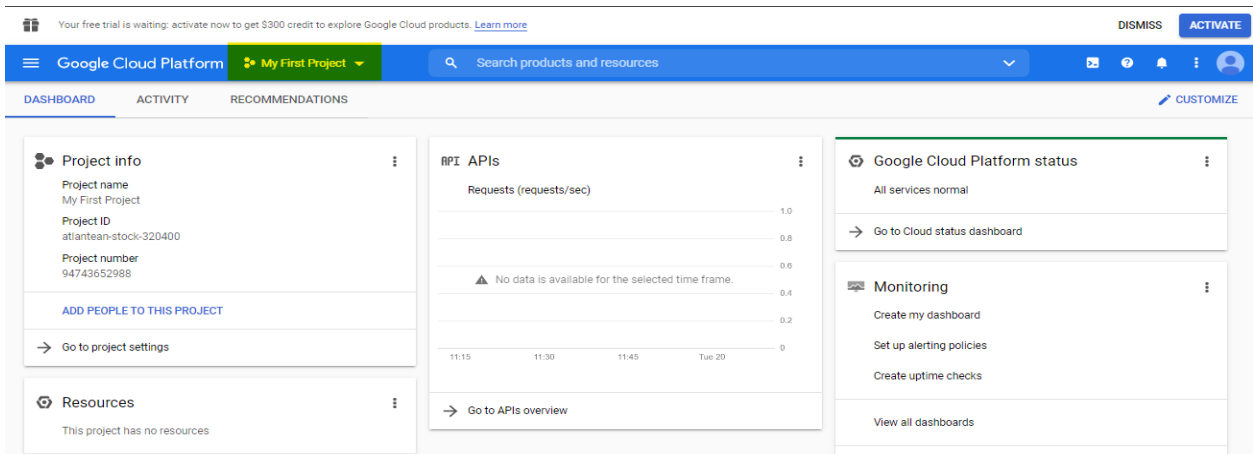
[Contact sales](#)

Gartner

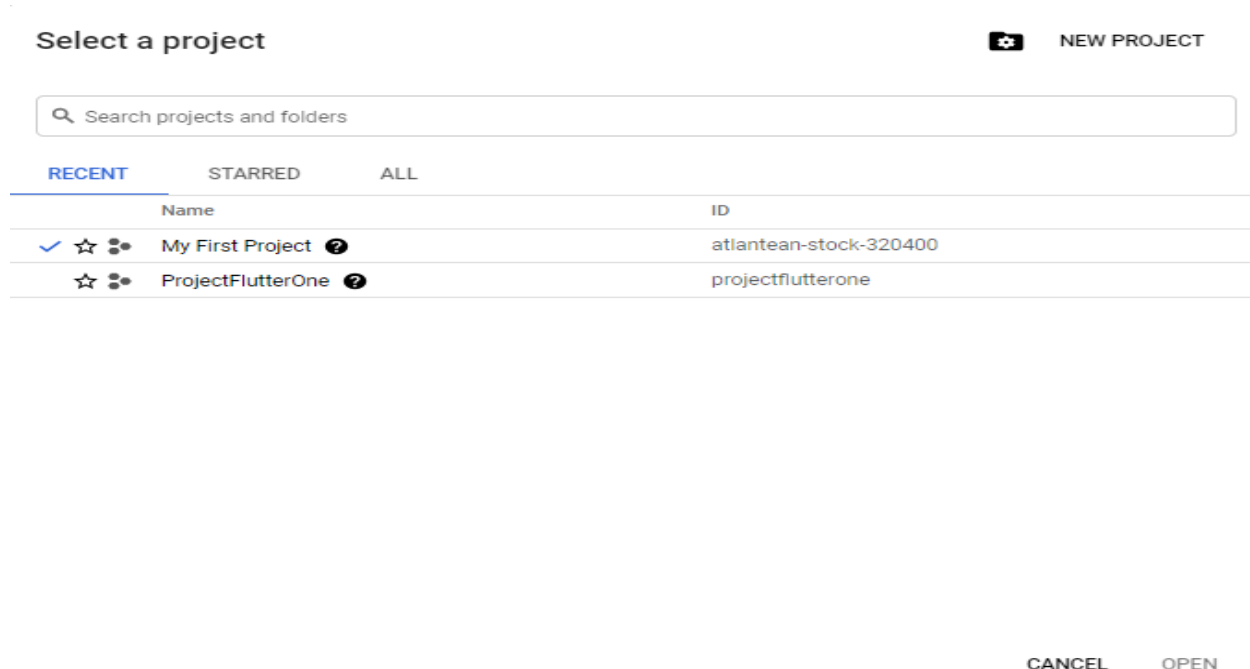
Gartner names Google Cloud a Leader in the 2021 Magic Quadrant for Cloud AI Developer Services.

[Get the report](#)

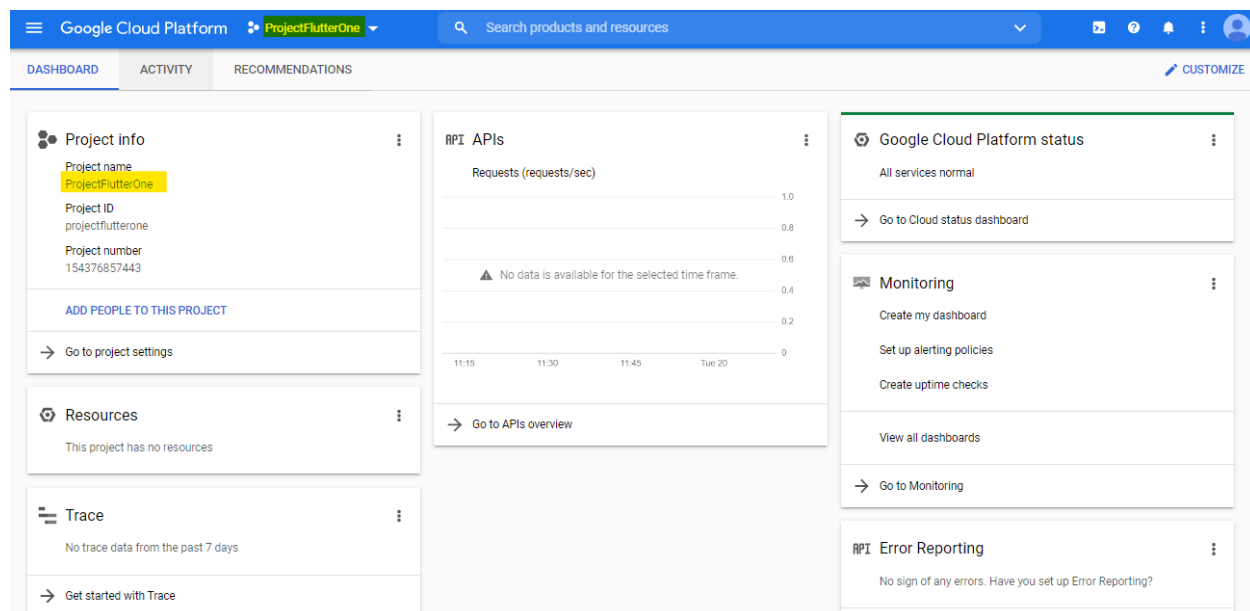
4. Click on the "Add project" button



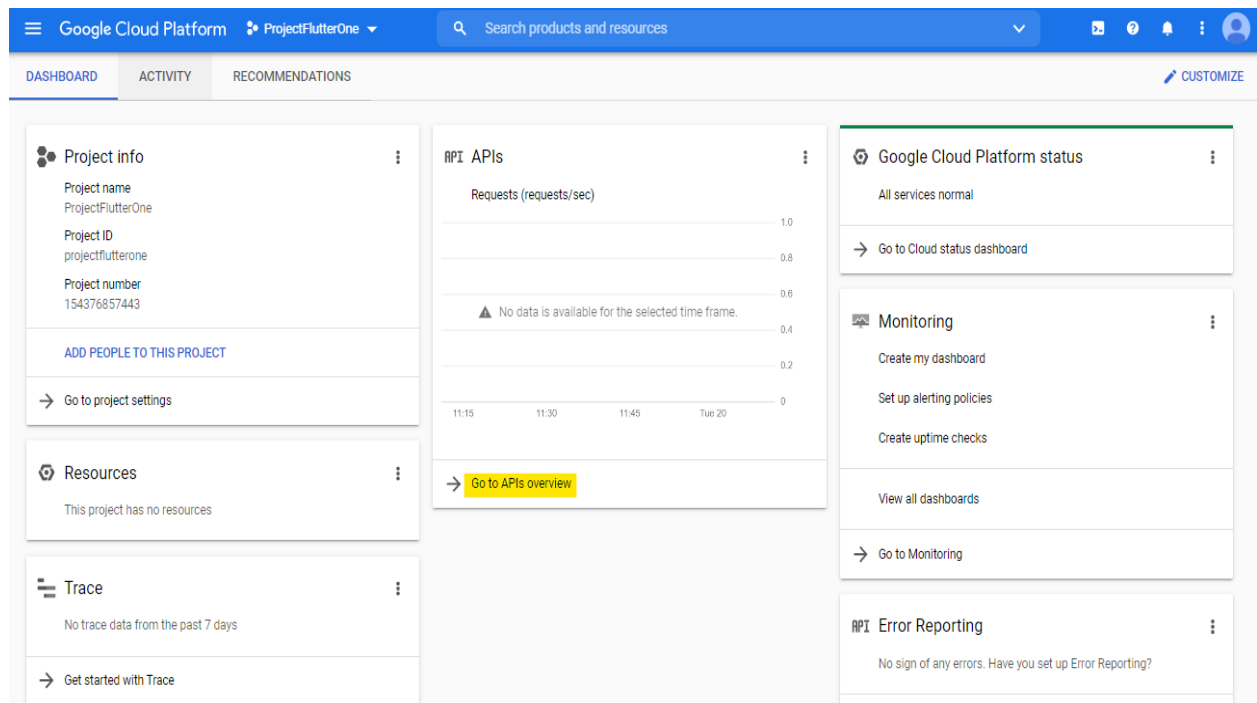
5. Enter the name of your existing Google Cloud Project



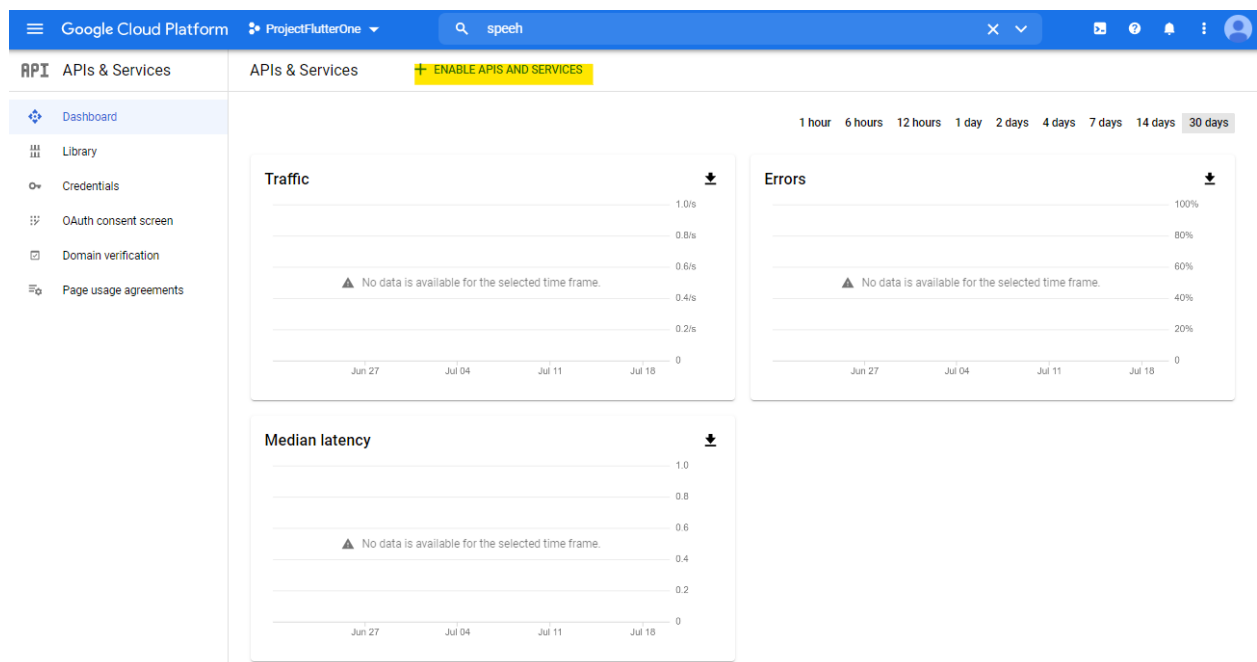
- Choose your created project, and the dashboard should now display project info for this specific project.



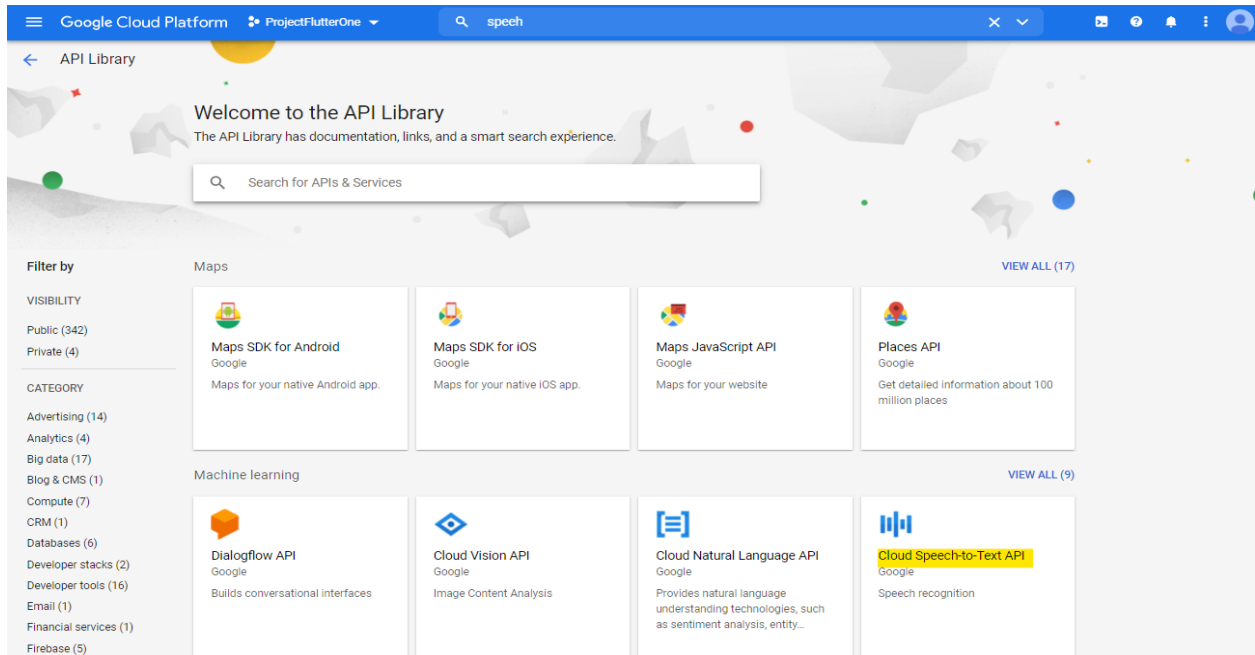
- When the project is ready, click on "Go to APIs Overview."



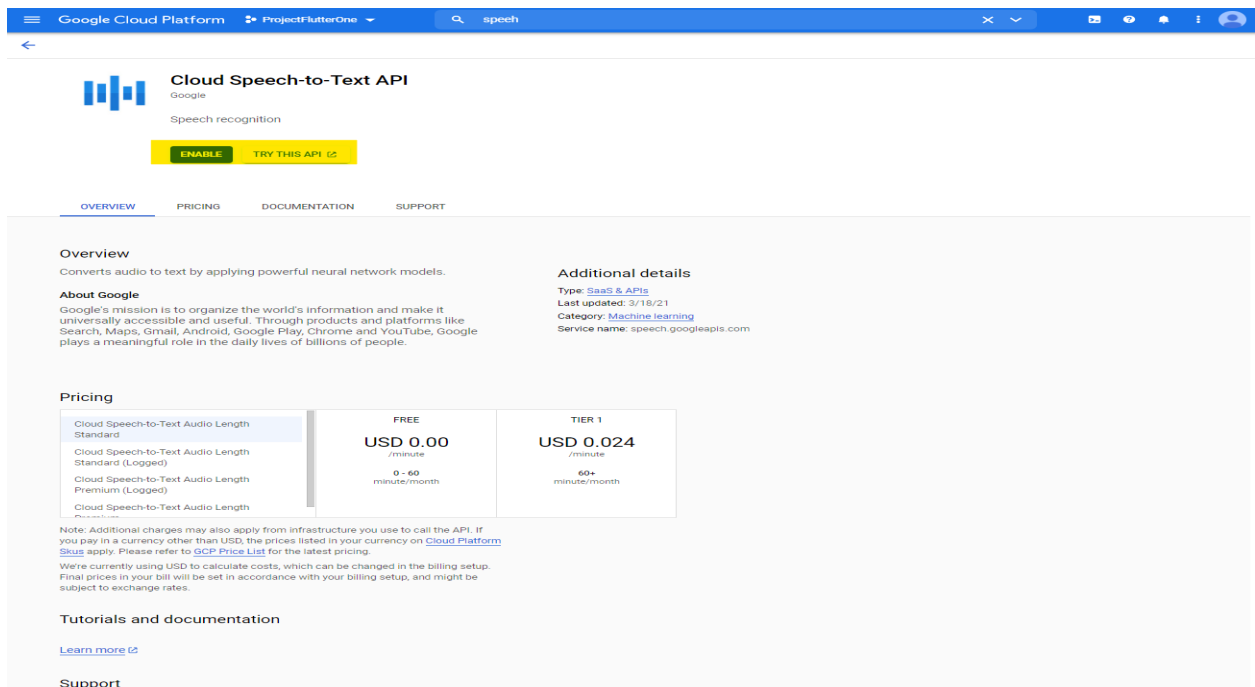
8. On the APIs & Services screen, click the "+ ENABLE APS AND SERVICES."



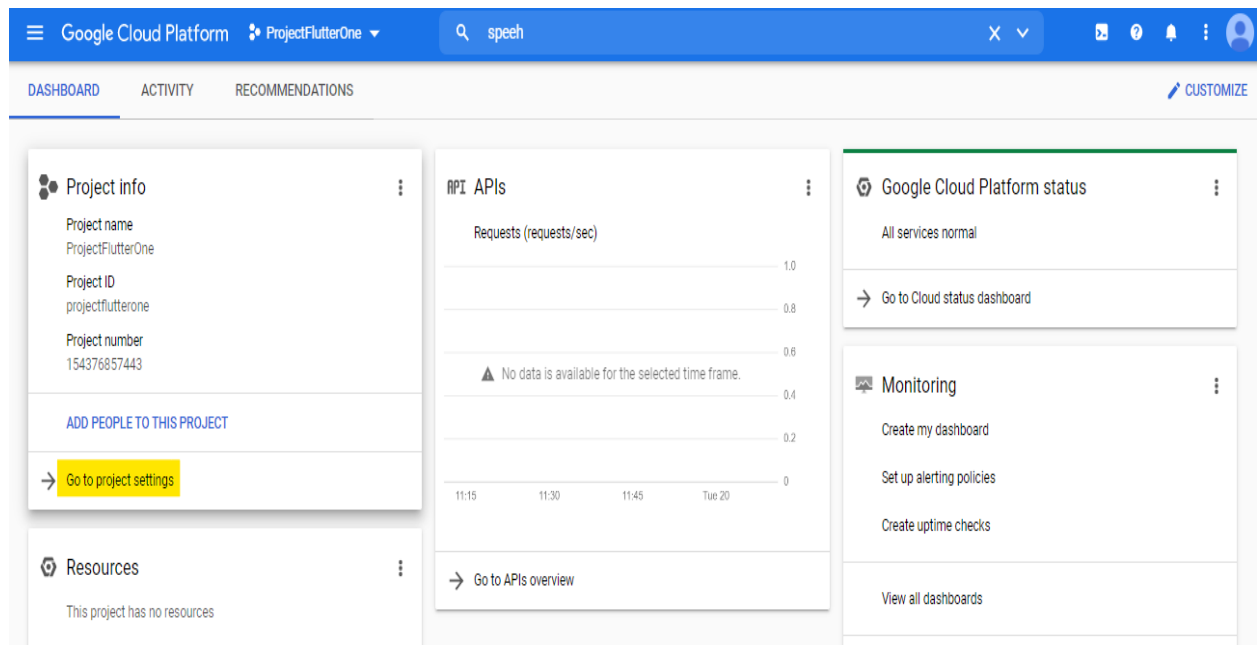
9. Click the "Cloud Speech-to-Text API" under Machine learning.



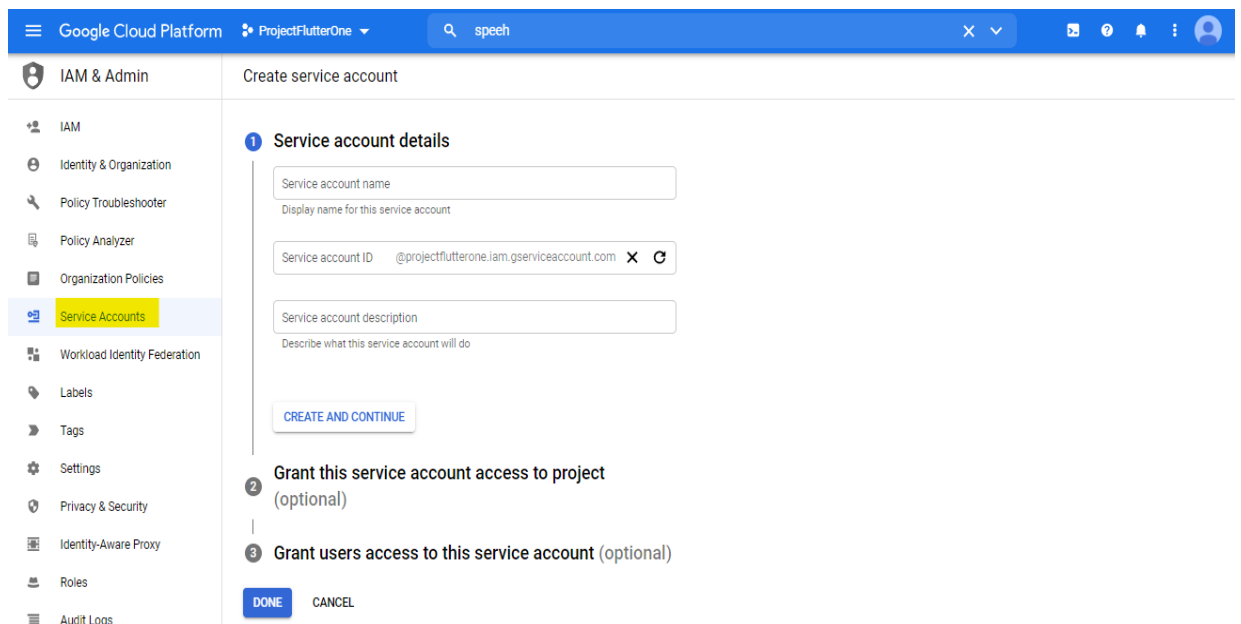
10. Click the enable button.



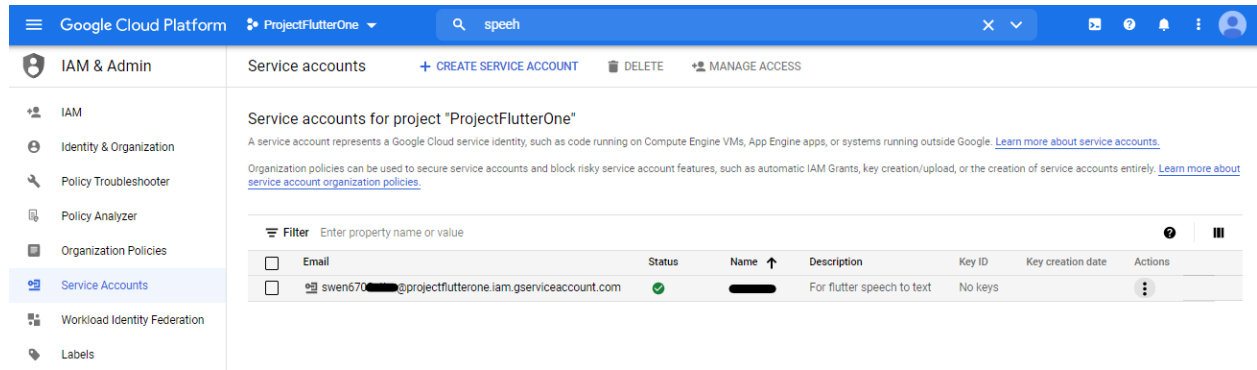
11. Click the "Got to project settings" button from the main Dashboard



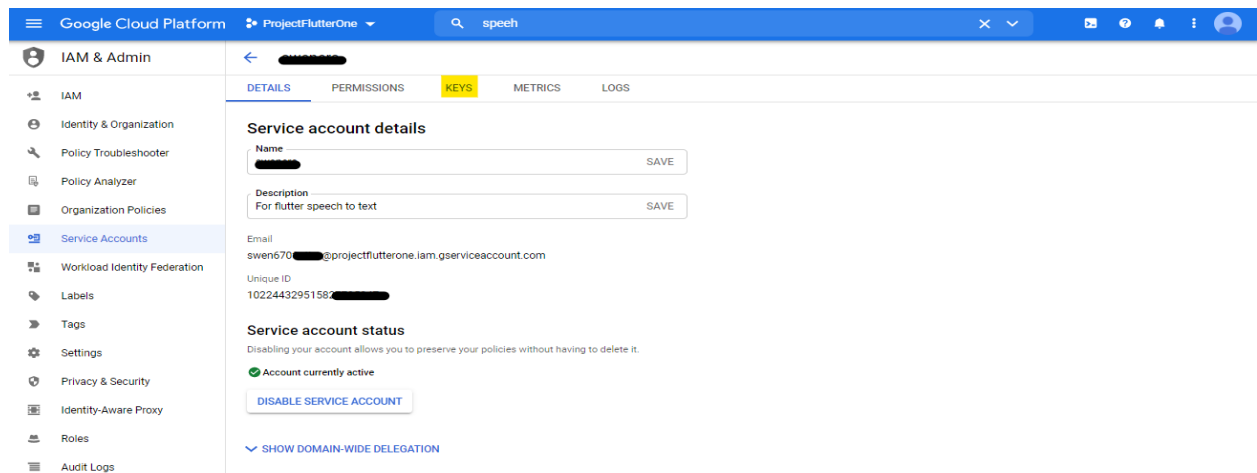
12. Click "Service Account", click "+ CREATE SERVICE ACCOUNT". Fill credentials and click "Done" to create an account.



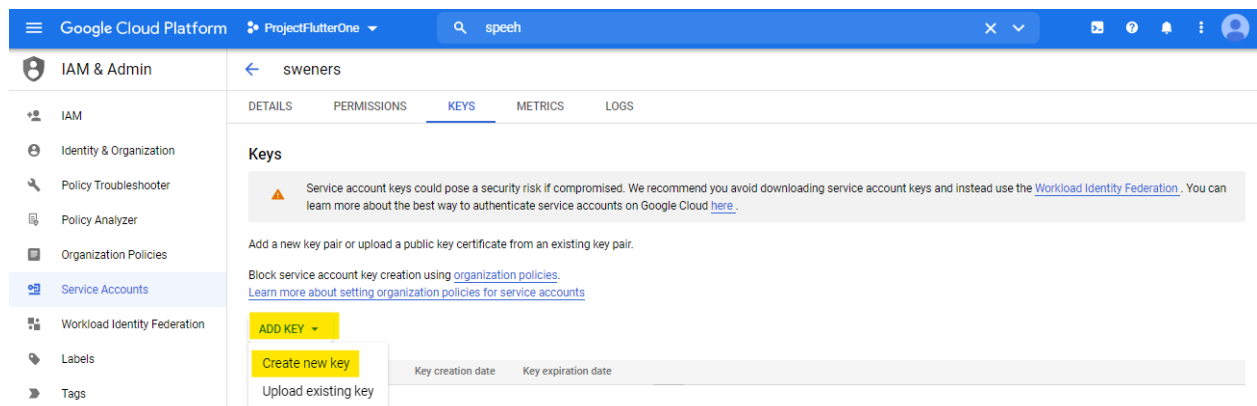
13. Select created service account, with the "Actions" button, click on "Manage Details."



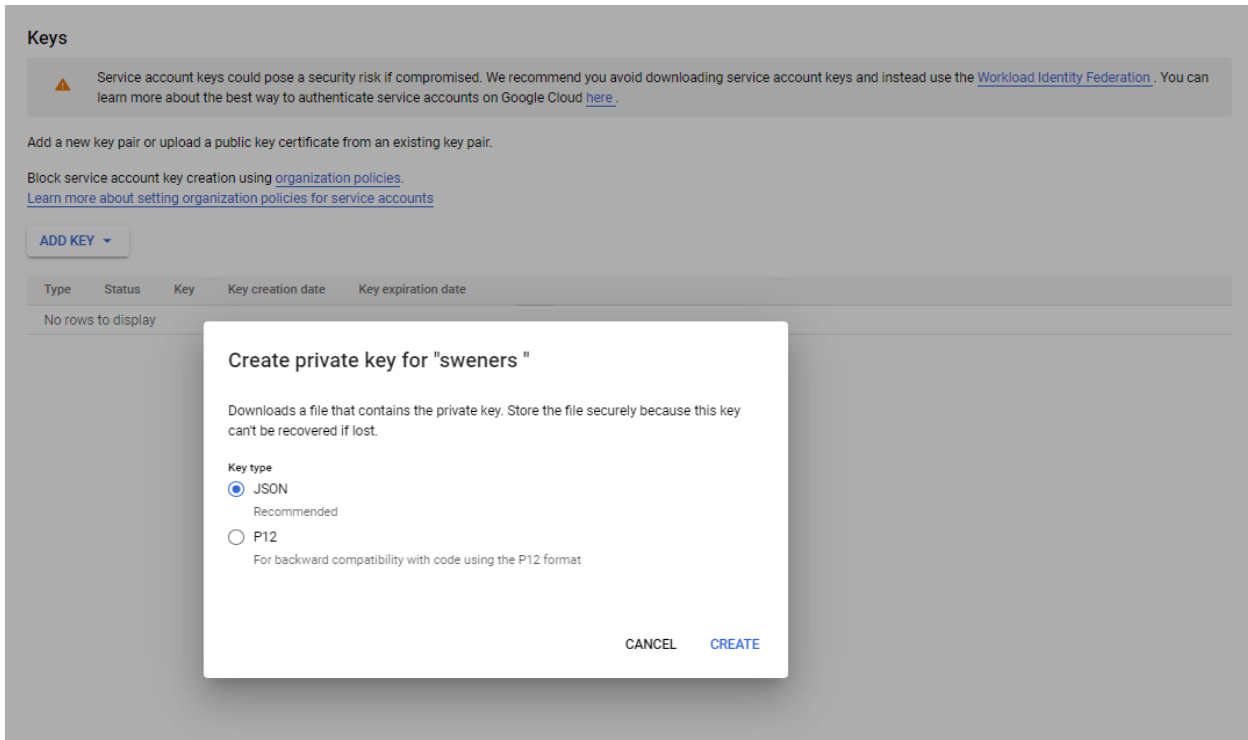
14. Click the "KEYS."



15. At the bottom of the Keys screen, click on "ADD KEY" and then "Create new key."

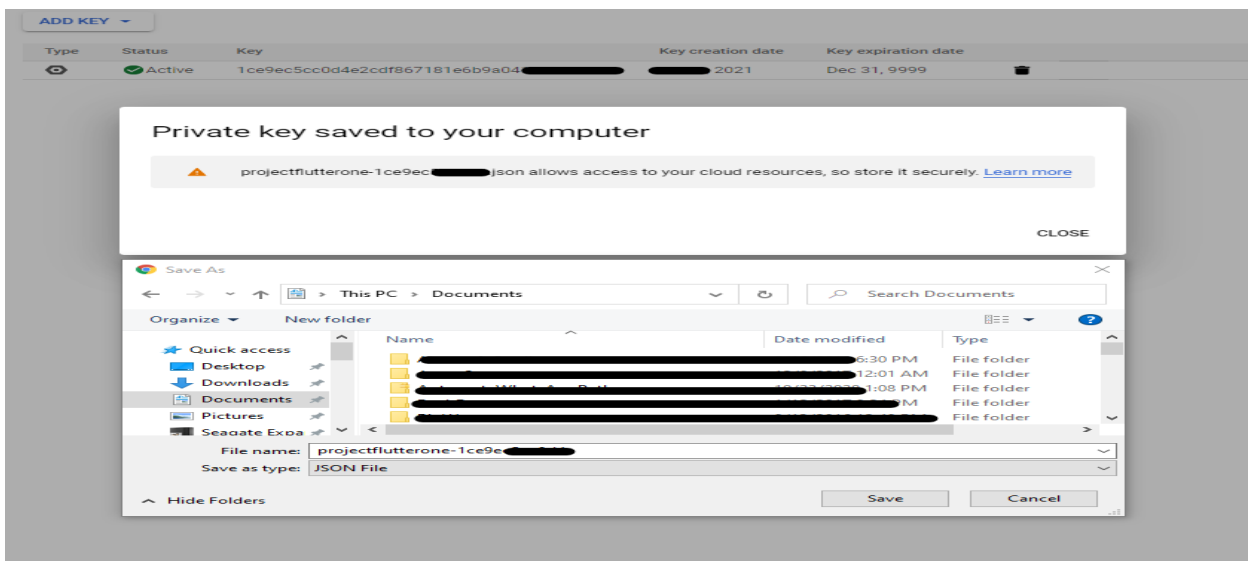


16. Create private key as JSON



17. Download private key. Next, set the environment

variable `GOOGLE_APPLICATION_CREDENTIALS` to the path of the downloaded JSON file.



5 Testing the Mobile Application

This section discusses the testing process complete for the Mnemosyne mobile application. A few testing challenges that may come up in the testing process are the different screen sizes of other mobile devices or issues with the software used in the testing environment. With frequent updates to Android and iOS, it is crucial to continue testing the application and making sure it is compatible with the different versions of Android and iOS that develop in the future.

A testing strategy that implements is to take into account to manage quality and performance.

- Research what mobile devices for Android and iOS are trending in the market for users and test those devices primarily.
- When testing the application, use emulators for iOS and different Android versions, such as Q and R. The bullet points below show the different types of emulators to try.

Device Emulators – Android Studio offers a way for the developer or tester to plug in a mobile device and test the application in real-time.

Operating system Emulator – When testing on the application through Android Studio on a Mac OS, Xcode should be downloaded and updated. Through Xcode, iOS device emulators test on Android Studio.

5.1 Testing Objective

The Mobile App testing for this project focuses on UI testing for end-to-end system integration to ensure that the software product we deliver has met the Software Requirements Specification (SRS) and performs functionalities described in the Project Plan successfully. The testing will cover all Mobile Application functionalities, features, use cases, and code coverage for this software project.

Our tests will simulate the user's (for instance, an elderly person and their nurse) interaction with the system. The system responds to the user's request for each category accurately and successfully to satisfy the stakeholder's and the customer's expectations. Best of all, our test will ensure that the UMGC Software Capstone Project (SWEN670, Summer 2021) will be successful.

5.2 Testing Procedures

- Prepare a Windows (version 8 or later) 64-bit operating system, x64 based processor laptop.
- Install Microsoft Teams for collaboration with team members
- Install Flutter version 6.3
- Install Android Studio version 4.2.2 with Flutter and Dart plugins
- Install GitHub for collaboration with team members
- Install Git, download it from <https://git-scm.com/downloads>.
- Clone the GitHub repository github.com/umgc/summer2021.charlie. Use the Git command as follows: `git clone < https://github.com/umgc/summer2021.charlie >`
- Execute the application
- Manually test the features of the application
- Document the test results
- If any tests fail, collaborate with the Project Manager and the team developers to fix the issues
- Rerun the failed tests when the features fixed
- Document test results
- Write the test report

A detailed description of the tests and their results will document in the Test Report section on milestone 4.

5.3 Possible Software Issues in the Testing Process

There is a strong chance of having issues with Android Studio. Specifically, with loading the devices for the testing environment, even if you have created a device for testing using the AVD manager. It can cause a standstill in the testing process as a critical error.

5.4 Testing Procedures that Circumvent Software Issues

- Prepare a Windows (version 8 or later) 64-bit operating system, x64 based processor laptop.
- Install Microsoft Teams for collaboration with team members
- Install Flutter version 6.3
- Install Android Studio version 4.2.2 with Flutter and Dart plugins
- Install GitHub for collaboration with team members
- Open Android Studio, and got to the AVD Manager, and created your device for testing
- Go to GitHub, and download the apk zip file your team created using the following steps:

1. Click on the Code tab
2. Click the green arrow under the "Go to file" button, and click "details" on option 1
3. On the left side, choose the tab for "Stable Linter, Build, Unit Test."
4. Scroll to the bottom of the page, and click on the APK artifact to download it
5. Once downloaded, use the extract all feature on the zip file.

- Go back to Android Studio, and click the AVD Manager button at the top right
- Click the play button next to your desired/created device
- When the device opens, drag and drop the APK file into the mobile device, and the device will download the app and allow you to start testing.

NOTE: As your team develops new releases/updates, you will need to remove the app from the device and download the updated APK file to test new release versions.

6 Troubleshooting

The Mnemosyne application is a lightweight application. However, while developing new enhancements or fixing any defects, a few issues are expected to be encountered. Here is a list of a few significant and commonly faced problems and the steps to troubleshoot them.

6.1 Issue installing Flutter on MacOS

If the Flutter CLI intended to run on macOS, it must add the bin path to the environment variable. You may experience the issue of CLI not working in a new terminal session. Therefore, the path needs to be saved in the profile permanently. Please follow the below steps to do it.

1. Open terminal.
2. Run Sudo nano /etc/paths.
3. It would prompt to enter the password to the computer user login.
4. After the password is entered and validated, it would display the current paths saved on the profile.
5. Add the bin path of Flutter at the bottom <your local path>/flutter/bin.
6. Hit control-x to quit the terminal session, which will prompt if you want to save the changes.
7. Type Y to save.
8. You can verify the path again in a new terminal window by typing echo \$PATH.

6.2 Emulator does not respond or slow

After using the emulator to open and close the application a few times, it is possible that the emulator becomes slow and starts being unresponsive. After all, being a virtual machine, the emulators have limited capabilities and resources. Therefore, the emulator being unresponsive is

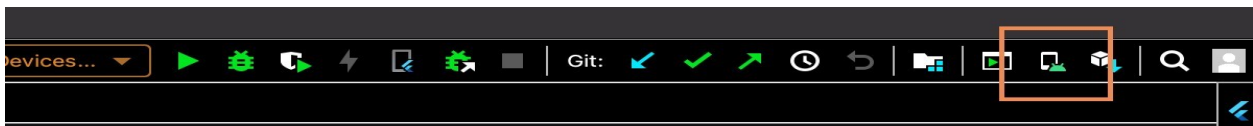
highly possible and may be unavoidable. In such cases, the step below shows how to restart and cold boot the emulator:

1. Open android studio if it is not already open.
2. Open AVD manager by doing one of these.

On the first page, click on configure and then AVD Manager

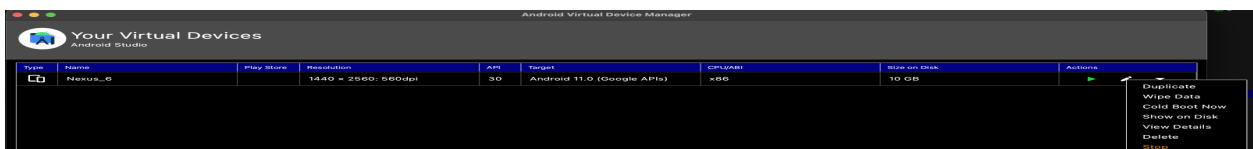


Or, if you have a project open in Android Studio, you can click the AVD Manager button in the toolbar at the top.



These lists show all the available emulators created.

3. Select the dropdown picklist at the end of your favorite emulator, displaying the options to duplicate the emulator, wipe data, etc.



4. Select "Wipe Data," which will completely remove the application from the emulator and the related data. To perform this action, your emulator must not be running. Close it if it is running.
5. Select "Cold Boot Now" to restart the emulator.

6.3 Unable to launch Emulator from Editor

If an editor like Visual Studio Code is being used to write the code, then it is possible that the emulator may fail to open when you start running or debugging the main .dart file. In such cases, the emulator opens from the command line interface. Here are the steps on how to do it.

1. Open command prompt or the terminal app (Terminal is also available in the editor itself)
2. Run flutter emulators to list all the emulators in the AVD manager.
3. Run flutter emulators --launch <your favorite emulator id from the list>. Copy the emulator id and paste after "--launch."

6.4 Slow Running the App (Stuck at Running Gradle task 'assembleDebug')

It is one of the common issues all users may face, regardless of the editor or operating system.

When Flutter tries to start the app in the emulator, the debug console may show "Running Gradle task 'assembleDebug'" for a long time. It happens when Flutter is trying to build the apk file and has many dependencies in the cache. Therefore, cleaning the Gradle will improve this. As the application restarted, it will reimport the required dependencies only and rebuild the apk file.

The following step that cleans the Gradle cache shows below:

1. Open terminal app or the one in the editor.
2. Navigate to the android directory in your project.

3. Run this command: `./gradlew clean`.

6.5 Dependency Issues

As the application built many times, multiple versions of libraries may get imported. That may make Flutter confused and display errors. In such cases, the Flutter built files must clear, and it should reimport again. Follow the below steps:

1. Open terminal app or the one in the editor
2. Navigate to the project root directory.
3. Run this command: "flutter clean." It will remove all the dependencies.
4. To reimport, the libraries run: flutter pub get. It will reimport the dependencies, and you should not get the errors.