Deployment and Operation Guide (Runbook) – Milestone 3

Team: charlie

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

SWEN 670 – sOFTWARE eNGINEERING pROJECT

June 11, 2021

reVision 1.0

Project name: Mnemosyne, Disability Mobile Application

Date: July 20, 2021

Project Leader: Michael Le

Phase: Design & Engineering and Execution

For approval: Michael Le

Michael le Date: 07/20/2021

For approval: Dr. Mir Mohammed Assadullah

Date: 07/20/2021

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version Number | Date | Description | Approved By |
| 1.0 | 07/20/2021 | Initial Deployment and Operations Guide Release | Michael Le |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 4](#_Toc77574148)

[2 Mobile Application 5](#_Toc77574149)

[2.2 Features, Packages, Plugins, and Widgets 5](#_Toc77574150)

[2.2.1 Features 5](#_Toc77574151)

[2.2.2 Packages 5](#_Toc77574152)

[2.2.3 Plugins 5](#_Toc77574153)

[2.2.4 Widgets 6](#_Toc77574154)

[3. Software Installation 7](#_Toc77574155)

[3.1 Flutter and Dart 7](#_Toc77574156)

[3.3 Android Emulator 15](#_Toc77574157)

[4 Prepare the Mobile Application for Use 19](#_Toc77574158)

[4.1 Firebase 19](#_Toc77574159)

[5 Testing the Mobile Application 20](#_Toc77574160)

[6 Troubleshooting 23](#_Toc77574161)

[6.1 Issue installing Flutter on MacOS 23](#_Toc77574162)

[6.2 Emulator does not respond or slow 23](#_Toc77574163)

[6.3 Unable to launch Emulator from Editor 24](#_Toc77574164)

[6.4 Slow Running the App (Stuck at Running Gradle task ‘assembleDebug’) 25](#_Toc77574165)

[6.5 Dependency Issues 25](#_Toc77574166)

# 1 Introduction

1.1 Purpose

1.2 Intended audience and reading suggestions

1.3 Technical project stakeholders

1.4 References

1.5 Definitions, Acronyms, and Abbreviations

# 2 Mobile Application

## 2.2 Features, Packages, Plugins, and Widgets

### 2.2.1 Features

### 2.2.2 Packages

* Flutter\_local\_auth
* Flutter\_tts(text to speech)- contains service to convert text to speech
* Flutter encrypt library - contains service to encrypt and decrypt the text using a secret key.
* Flutter material.dart
* googleapis

### 2.2.3 Plugins

Flutter and Dart Plugin were installed

**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. One’s flutter is done installing, click on “Restart IDE” to restart android studio.

**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. One’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 wav file

Stateless widget –

Audio\_recognize widget - displays the mic which changes the icon from mic to stop based on the current state.

# 3. Software Installation

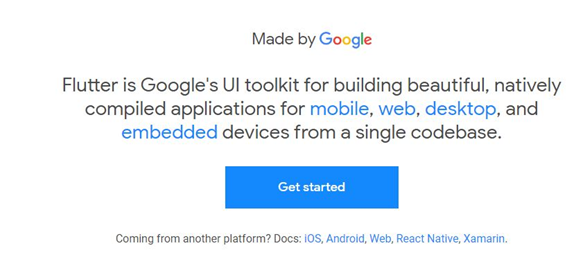
## 3.1 Flutter and Dart

**Flutter System Requirements**

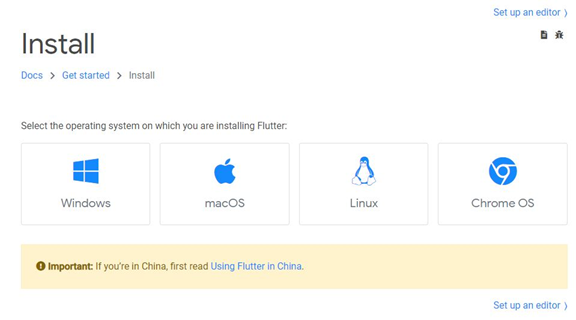
* Operating Systems: Windows 7 SP1 or later (64-bit), x86-64 based.
* Disk Space: 1.64 GB (Does not include disk space for IDE/tools).
* Tools: Flutter depends on these tools being available in your environment.
  + Windows PowerShell 5.0 or newer. (Pre-installed with Windows 10)
  + Git for Windows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

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

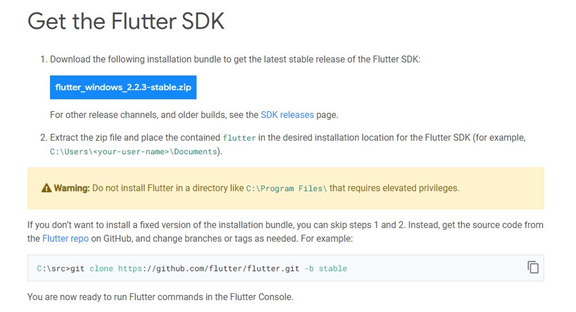
1. Before starting, ensure that Windows PowerShell 5.0 and Git for Windows 2.x are both installed.
2. To download Flutter, first navigate to [http://flutter.dev](http://flutter.dev/) in your web browser.
3. Click the “Get started” button.



1. Select “Windows” as your operating system.



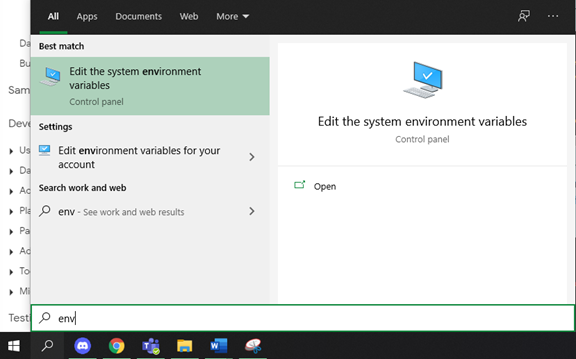
1. Download the Flutter SDK zip file by using the blue button below “Get the Flutter SDK”



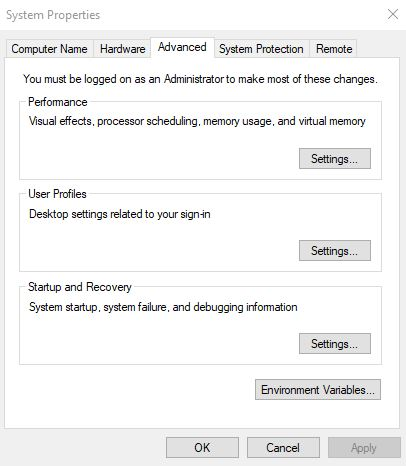
1. After the file is downloaded, navigate to the file on your machine and unzip it to your desired location.

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 be able to run flutter commands in the normal windows command prompt, continue on.

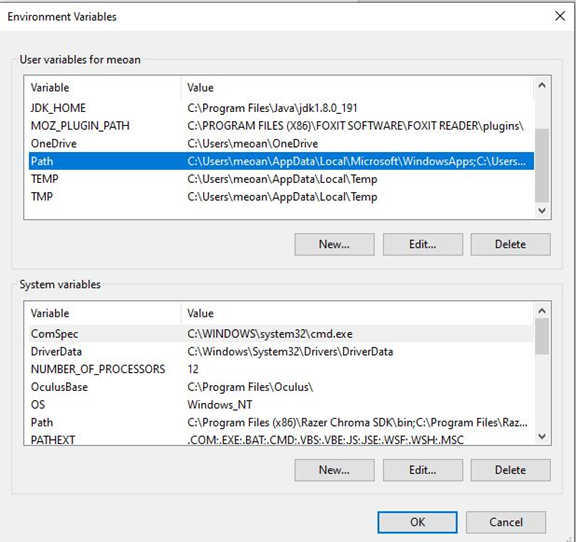
1. Using the Start search bar, search for “env” and select “Edit environment variables for your account”



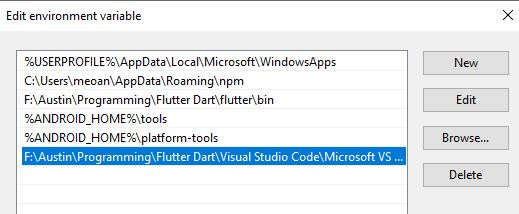
1. Click the “Environment Variables…” button on the window that comes up.



1. 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.



1. Click the “New” button and enter the directory path to the bin sub-folder in the Flutter SDK directory you previously extracted.
   1. To find this path, navigate to the Flutter SDK folder previously extracted, open the bin folder, then copy the file path at the top.



1. Open command prompt and enter “flutter doctor” to confirm that it is working and to see if any dependencies are missing.

3.2 Android Studio

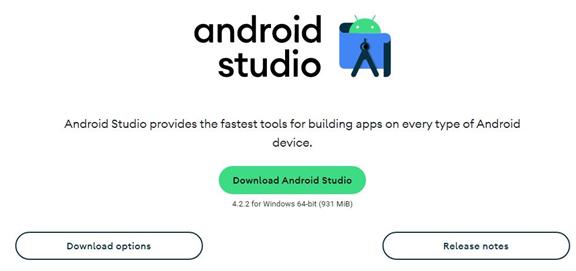
**Android Studio System Requirements**

* 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) to be installed. 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 largely the same for other supported operating systems.

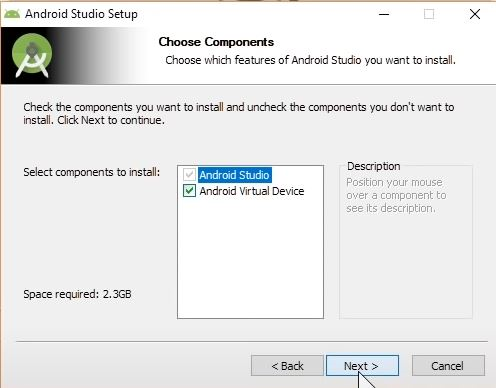
1. In your web browser, navigate to <https://developer.android.com/studio>
2. Click on the “Download Android Studio” button near the top.



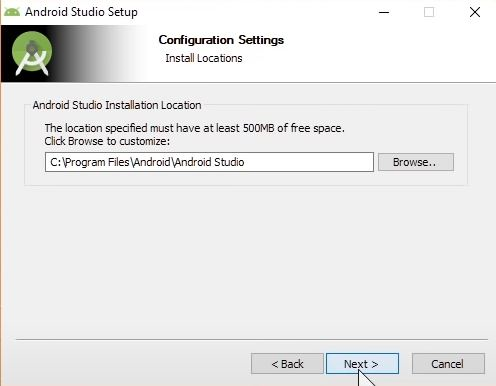
1. Look through the Terms and Conditions, scroll to the bottom, click the check box to confirm that you have read the above, and then click the “Download Android Studio for Windows” button.



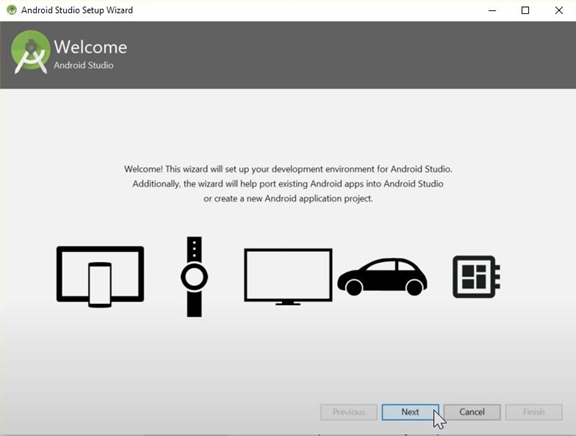
1. Once the file has been downloaded, execute the executable file to begin installation.
2. Click “Next” until the Choose Components screen, where you will have to check the box next to “Android Virtual Device”



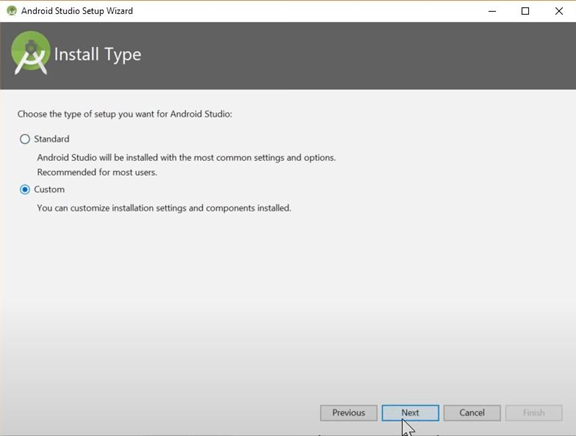
1. Again, click “Next” until you get to the Installation Location screen where you will enter the location that you want Android Studio to be installed to.



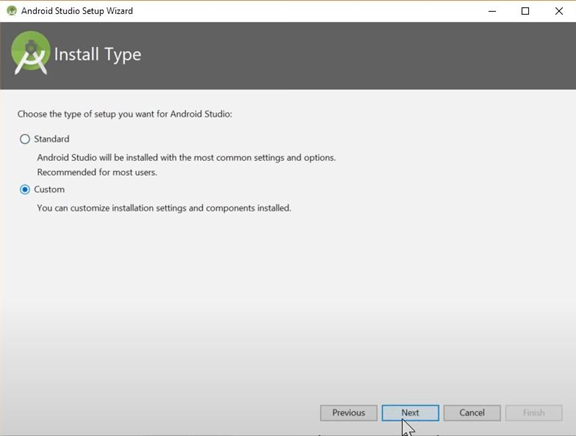
1. Click “Next” until the application is installed. This may take several minutes.
2. Once it is installed, start Android Studio by clicking on the shortcut or searching for it using the Start search bar.
3. Click “Next” on the Welcome screen.



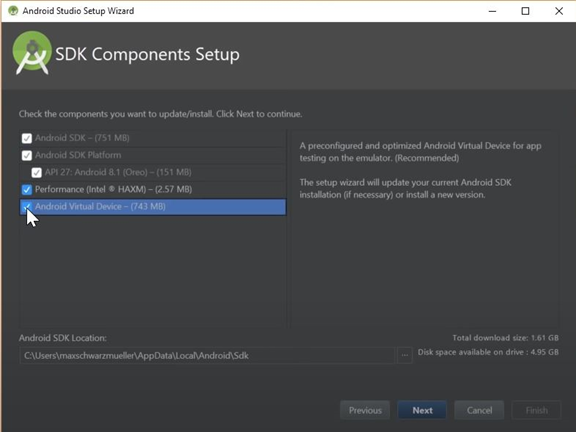
1. On the Install Type screen, click the radio button next to Custom and then click “Next”



1. Select the UI Theme that you prefer, and click “Next”



1. On the SDK Components Setup screen, click the checkbox next to “Android Virtual Device” and then click “Next”



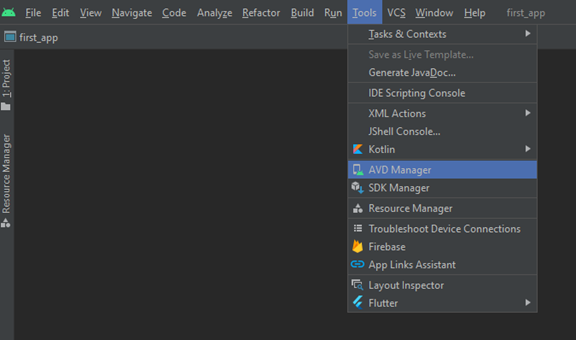
1. Select the location that you want the Android SDK to be installed to, and then click “Next”
2. Click “Finish” to begin installing Android Studio. This may take several minutes.

## 3.3 Android Emulator

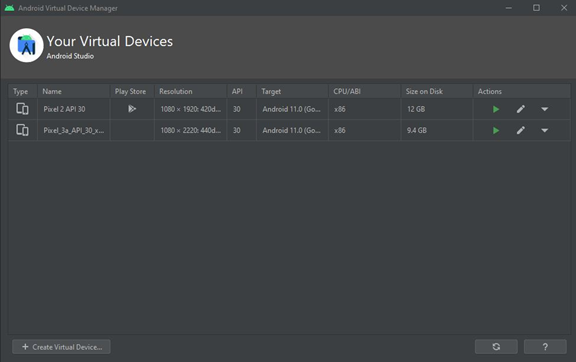
Before the Android Emulator can be set up, you need to make sure that VM acceleration is enabled on your computer. Use the link below if you need help doing so.

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

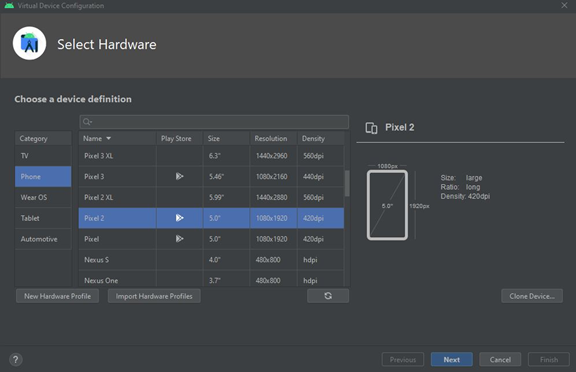
1. Open Android Studio
2. On the taskbar at the top, click Tools, and then AVD Manager.



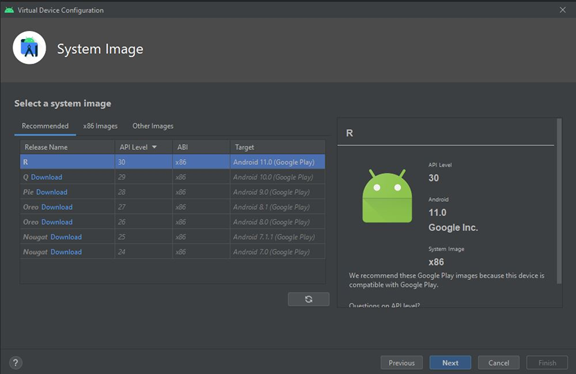
1. In the new window that comes up, click “Create Virtual Device…” at the bottom.



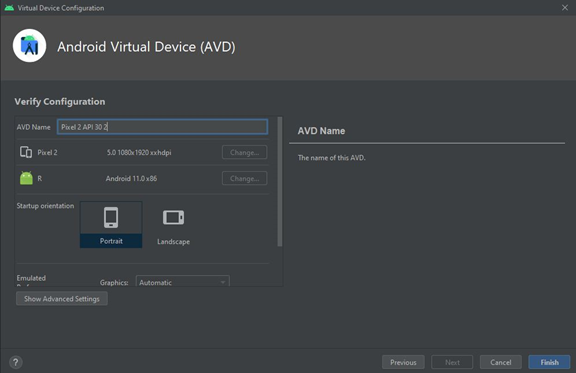
1. On the Select Hardware screen, select Phone on the left and then Pixel 2 in the center, then click “Next”



1. On the System Image screen, select the topmost entry in the Recommended tab named “R” and then click “Next”



1. On the Android Virtual Device (AVD) screen, type a name for you 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 Firebase

# 5 Testing the Mobile Application

This section discusses the testing process that is done for the Mnemozyne mobile application. A few testing challenges that may come up in the testing process are the different screen sizes of different mobile devices, or issues with the software used in the testing environment. With frequent updates to Android and iOS, it is crucial to continue to test the application and make sure it is compatible with the different versions of Android and iOS that will be developed in the future.

A testing strategy that has been implemented 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 test with.
  + 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 can be tested on Android Studio.

**Testing Objective**

The Mobile App testing for this project focuses on UI testing for the end-to-end for the system integration of the product to make sure that the software product that we will deliver has met the Software Requirements Specification (SWS) and performs functionalities as described in the Project Plan successfully. The testing will cover all of the 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, and the system responses to the user’s request for each category accurately and successfully to satisfy the stakeholders and the customer’s expectation. Best of all, our test will assure that the UMGC Software Capstone Project (SWEN670, Summer 2021) will be successful.

**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 are fixed
* Document test results
* Write the test report

The detailed description of the tests and their results will be documented in the Test Report section on milestone 4.

**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. This can cause a standstill in the testing process as a Critical error.

**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 create your device for testing
* Go to GitHub, and download the apk zip file your team has 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 up, drag and drop the APK file into the device, and the device will download the app, and allow you to start testing

NOTE: As your team develops new releases/updates of the application, 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 issues and the steps to troubleshoot them.

## 6.1 Issue installing Flutter on MacOS

If the Flutter CLI is intended to run on MacOS, then it is required to 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 for a few times, it is possible that the emulator becomes slow and starts being unresponsive. Afterall, 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 following steps should be followed 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.
   1. On the first page, click on configure and then AVD Manager

Graphical user interface, application

Description automatically generated

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

A screenshot of a computer

Description automatically generated with medium confidence

This lists all the available emulators that have been previously created.

1. Select the dropdown picklist at the end of your favorite emulator which will display the options to duplicate the emulator, wipe data, etc.

A screenshot of a computer

Description automatically generated

1. 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.
2. 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 can be opened from the command line interface. Here are the steps 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’)

This 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. This happens when flutter is trying to build the apk file and it has a lot of dependencies in the cache. Therefore, cleaning the gradle will improve this. As the application is restarted, it will reimport the required dependencies only and rebuild the apk file.

To clean the gradle cache, the following step can be followed.

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 is built many times, it is possible that multiple versions of libraries get imported. That may make flutter confused and display errors. In such cases, the flutter build files must be cleared, and it should be reimported 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. This will remove all the dependencies.
4. To reimport the libraries run: flutter pub get. This will reimport the dependencies and you should not get the errors.

Appendices:

Appendix A - References