

MemorEZ Deployment and Operations Guides (Runbooks)

Combined Version 1

Combined by James Eble

UMGC SWEN670 – Spring 2022 (From submissions by FlutteringMind and RememberAll)

Professor Dr. Mir Assadullah

Table of Contents (Section 1: Patient Mode)

1 Introduction	6
1.1 Purpose	6
1.2 Intended Audience and Reading Suggestions.....	6
1.3 Technical Project Stakeholders	7
1.4 References	8
1.5 Definitions, Acronyms, and Abbreviations	9
2 Mobile Application.....	10
2.1 Features, Packages, Plugins, and Widgets	10
2.1.1 Features	10
2.1.2 Packages.....	10
2.1.2.1 Installing a Package Dependency Into an App.....	10
2.1.3 Plugins.....	13
2.1.3.1 macOS Installation	13
2.1.3.2 Windows Installation	13
2.1.4 Widgets	13
3 Software Installation	15
3.1 Android Studio	15
3.2 Flutter and Dart.....	16
3.3 Flutter and Dart Plugins	17
3.4 Android Emulator.....	18
3.5 Test Your Development Environment.....	20
3.6 GitHub Desktop.....	21
3.7 Task Management Tool.....	21
4 Prepare the Mobile Application for Use	23
4.1 Cloning GitHub Repository.....	23
4.2 Run the Flutter Application.....	24
5 Testing the Mobile Application	25
5.1 Testing Objectives.....	25
5.1.1 Unit Tests	25

MemorEz Runbooks - Combined

5.1.2 Integration Tests	29
5.2 Testing Procedures	31
6 Troubleshooting	32
6.1 Emulator Freezing	32
6.2 Emulator Not Responding	33
6.3 Out of Memory Error	34
6.4 Stuck at Running Gradle Task 'assembleDebug'	35
6.5 Dependency Errors	36
7 Appendices	37
7.1 Credits	37
7.2 Credits to Previous Cohorts	38

Table of Contents (Section 2: Caregiver Mode)

1 Introduction	39
1.1 Purpose	39
1.2 Intended Audience and Reading Suggestions.....	39
1.3 Technical Project Stakeholders	39
1.4 References	39
1.5 Definitions, Acronyms, and Abbreviations	39
2 Mobile Applications	42
2.1 Features, Packages, Plugins, and Widgets	42
2.1.1 Features	42
2.1.2 Packages.....	42
2.1.2.1 Installing a Package Dependency Into an App.....	43
2.1.3 Plugins.....	44
2.1.3.1 macOS Installation	44
2.1.3.2 Windows Installation	44
2.2 Features, Packages & Plugins.....	45
2.2.1 Widgets	50
3 Software Installation	53
3.1 Android Studio	53
3.2 Dart and Flutter.....	55
3.3 Flutter and Dart Plugins / Android Studio System Settings	58
3.4 New Project from Source Control.....	64
3.5 GitHub Desktop.....	67
3.6 Configuring Physical Android Devices for Running MemorEZ	69
3.7 Configuring Emulators for Running MemorEZ.....	69
3.8 Setting Up an Android Emulator	69
3.9 Setting up a Simulator (macOS)	69
4 Prepare the Mobile Application for Use	70
4.1 Fetching Updated Code/Dependencies.....	70
4.2 Attaching the LEX Credentials (enables language processing)	71

MemorEz Runbooks - Combined

4.3 Run the MemorEZ application	72
5 Testing the Mobile Application	73
5.1 Testing Objectives	73
5.1.1 Unit Tests	73
5.1.2 Integration Tests	79
5.2 Testing Procedures	79
6 Troubleshooting	81
6.1 Emulator Not Responding	81
6.2 Out of Memory Error	82
6.3 Stuck at Running Gradle Task 'assembleDebug'	83
6.4 Dependency Errors	84

Revision History for Combined Deployment and Operations Guides (Runbook):

Version	Date	Reason	Approved By
1.0	3/24/2022	Milestone 3	James Eble

Section 1: Patient Mode

Information in section one is copied from team FlutteringMind's Deployment and Operations Guide (Runbook)

1. Introduction

Team FlutteringMind's Runbook documentation describes all the steps and instructions in detail that all the team members who were involved as developers, testers, and DevSecOps performed, how they worked on, what they completed, and how they implemented the entire app together as a whole as a class of Spring 2022 at UMGC which shall help the future class of UMGC. Some of the sections in this documentation have been inspired by the last semester's cohorts and their documentation as our application MemorEZ is being built on the existing cohort's Memory Magic application.

1.1 Purpose

The purpose of this Deployment and Operations Guide (Runbook) is to outline the instructions with necessary steps and illustrate the details necessary to install and deploy the MemorEZ Application to user's devices. This document helps explain all the steps required to take to perform a successful installation.

1.2 Intended Audience and Reading Suggestions

The intended audience for this Runbook guide is the software developers, current peers, future students, and technical stakeholders who are interested in installing the MemorEZ Application. The purpose of this document is to show each step of the installation process in detail to provide instructions which any user can follow easily. A good reading suggestion to assist the intended audience understand the systems used would be, *Flutter for Beginners*, by Alessandro Biessek and *Programming Flutter* by Carmine Zaccagnino.

There are some very helpful guided free of charge YouTube beginner video tutorials to help users start with flutter. There are also some courses online that users could take to learn flutter in depth.

1.3 Technical Project Stakeholders

Table 1 shows the project stakeholders for the MemorEZ Application:

Table 1

Project Stakeholders

Name	Role
Dr. Mir Assadullah	Stakeholder (Project Owner)
Andrea	Previous stakeholder (nursing homeowner)
Dr. Evangelista	Current stakeholder
Roy Gordon	Stakeholder (Project Advisor)
Selina Zaman	Project Manager
Vanessa Stringer	Business Analyst
Joshua Fischer	UI/UX Designer
Daryle Urea	Lead Developer
Joseph Jewell	Developer
Anusha Ramanan	Tester
Sean LaMonica	Tester

1.4 References

Android studio. (n.d.). In *Developers*. Retrieved from <https://developer.android.com/studio#downloads>

Avery, D., Balbi, T., Bell, K., Crumb, K., Cruz Jimenez, C., Muwan, P., & Salim, S. (2021, October 10). Deployment and Operations Runbook - Memory Magic App. University of Maryland Global Campus

Biessek, A. (2019). *Flutter for beginners: An introductory guide to building cross-platform mobile applications with Fuller and Dart 2*. Birmingham, UK: Packt Publishing Ltd.

Cloning a repository. (n.d.). In *GitHub Docs*. Retrieved from <https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository>

Dart. (n.d.). In *Dart.dev*. Retrieved from <http://dart.dev>

Flutter. (n.d.). In *Flutter.dev*. Retrieved from <http://flutter.dev>

Flutter App stuck at "Running Gradle task 'assembleDebug'... ". (2019, December 29). In *Stackoverflow*. Retrieved from <https://stackoverflow.com/questions/59516408/flutter-app-stuck-at-running-gradle-task-assembledebug>

Flutter favorites. (n.d.). In *Pub.dev*. Retrieved from <https://pub.dev/>

Flutter widget index. (n.d.). In *Flutter.dev*. Retrieved from <https://flutter.dev/docs/reference/widgets>

Introduction to widgets. (n.d.). In *Flutter.dev*. Retrieved from <https://flutter.dev/docs/development/ui/widgets-intro>

Ladd, S. (2015, December 1). mkustermann / flutter.github.io. In GitHub. Retrieved from <https://github.com/mkustermann/flutter.github.io/blob/master/getting-started.md>

Peter, C. (n.d.). Understanding and using Flutter packages. In *Educative*. Retrieved from <https://www.educative.io/edpresso/understanding-and-using-flutter-packages>

Set up an editor. (n.d.). In *Flutter.dev*. Retrieved from <https://flutter.dev/docs/get-started/editor>

Speech-to-Text. (n.d.). In *Google Cloud*. Retrieved from <https://cloud.google.com/speech-to-text>

Testing Flutter apps. (n.d.). In *Flutter.dev*. Retrieved from <https://flutter.dev/docs/testing>

Using packages. (n.d.). In *Flutter.dev*. Retrieved from <https://flutter.dev/docs/development/packages-and-plugins/using-packages>

Zaccagnino, C. (2020). *Programming Flutter: Native, cross-platform apps the easy way*. Raleigh, NC: The Pragmatic Bookshelf.

1.5 Definitions, Acronyms, and Abbreviations

Table 2 shows the most used acronyms/abbreviations and their definitions:

Table 2

Acronyms, Abbreviations, and Definitions

Acronyms and Abbreviations	Definitions
PM	Project Manager
BA	Business Analyst
DSO	DevSecOps
STML	Short Term Memory Loss
UI	User Interface
NLP	Natural Language Processing
NLU	Natural Language Understanding
OS	Operating System
LEO	UMGC online platform class
iOS	iPhone operating system
Flutter CLI	Flutter Command-line tool
API	Application Program Interface
TTS	Text to Speech
UMGC	University of Maryland Global Campus
VM	Virtual Machine

2 Mobile Application

2.1 Features, Packages, Plugins, and Widgets

2.1.1 Features

Simplistic- The application is easy to navigate for the user. The layout has enlarged buttons and text, helping in the navigation process of the application. Clear and uncluttered screens makes the application easy to navigate, and improves the users experience.

Speed- The application has fast loading screens and loads desired content on demand, ensuring the engagement of the user.

Flexibility- The application is compatible with both Android and IOS users, ensuring that it is able to perform all functions across platforms quickly and effectively.

Security- Users are provided with the necessary security that will keep personal and sensitive information secure.

Searchability- Users have the ability to search for desired information throughout the application.

Notifications- Users will receive personalized notifications that are relevant to them.

Language- Five different languages are offered through this application, ensuring a variety of users can use the application

Logins- Logins will be available for users that are taking care of STML patients. This allows them to set certain features that the user can have access to.

Up To Date Information- The caregiver/provider will be kept up to date with all pertinent information via text message that the user has entered.

2.1.2 Packages

In Flutter, packages can be a useful tool that allows developers to quickly build an application using tools that other developers have created. These packages can be found in pub.dev, where they have been published. This webpage contains a variety of packages and dependencies that are compatible and usable in Flutter.

2.1.2.1 Installing a Package Dependency Into an App

1. Depend

- When in the application, open the pubsec.yaml folder and add a specific package under dependencies.

2. Install

- Open terminal and run flutter pub get.
or
- From Android Studio: Click on the “Packages get” located in the action ribbon at the top of pubspec.yaml.
- From VS Code: Click on the “Get Packages” which can be found on right side of the action ribbon at the top of pubspec.yaml.

3. Import

- Add an import statement that goes along with the package in the Dart code.

Table 3 shows the core packages used in the MemorEZ application.

Table 3

Packages and Descriptions

Packages	Description
flutter_localizations	Package dependency is used to set up a localization.
speech_to_text	Package that exposes the device specific speech to text recognition ability.
translator	Package that allows the device specific translation abilities.
highlight_text	Package that allows the ability to highlight words in a text.
avatar_glow	Package that allows the ability to animate the background.
font_awesome_flutter	Package that allows a set of Flutter icons. Includes free icons like regular, solid, and brands.
path_provider	Package that allows finding commonly used files in the filesystem.
xml	Package that is a lightweight library for parsing, traversing, querying, transforming and building XML documents.
encrypt	Package that allows a set of high-level APIs over Pointycastle for two-way cryptography.
shared_preferences	Package that Wraps platform specific persistent storage for simple data
timeago	Package that gives a library useful for creating fuzzy timestamps.
flutter_search_bar	Package that allows for an expandable floating search bar.
flutter_tts	Package for Text-to-Speech.

intl	Package that contains code to deal with internationalized/localized messages, bi-directional text, other internationalization issues, and date and number formatting and parsing.
porcupine	Package that allows always listening voice enabled hearing into the application.
mobx	Package that is a state management library that simplifies the ability to connect and reactivate data with the UI of the application
flutter_mobx	Package that provides a set of Observer widgets that will automatically rebuild when the tracked observables change.
provider	Package that is a wrapper around InheritedWidgets, which makes it easier to reuse them.
quiver	Package that is a set of utility libraries that makes using Dart libraries more convenient and easier, in addition to adding functionality.
http	Package that is a composable, multi-platform, Future-based API for HTTP requests.
json_serializable	Package that provides builders for handling a json package
amazon_cognito_identity_dart_2	Package that is the unofficial amazon cognito identity.
google_fonts	Package that allows the use of many fonts in the application
random_color	Package that allows for random colors in the application.
chat_bubbles	Package that gives the ability to use chat bubbles in the application
dcdg	Package that is a small command line that gives the ability to create a class diagram from a dart package.
sigv4	Package that is a dart library used for signing Amazon Web Services.
sqlite	Package that connects SQLite plugin to Flutter.
toast	Package that is a library for Flutter

Note. See <https://github.com/umgc/spring2022/blob/development/pubspec.yaml> for a complete list of packages.

2.1.3 Plugins

The installation of Flutter and Dart plugin instructions are different, depending on the platform being used:

2.1.3.1 macOS Installation

The following instructions should be used for macOS:

1. Start Android Studio.
2. Open plugin preferences (Android Studio > Preferences > Plugins).
3. Select the Flutter plugin and click Install.
4. When prompted click Yes to install the Dart plugin.
5. When prompted click Restart.

2.1.3.2 Windows Installation

The following instructions should be used for Windows:

1. Open plugin preferences.
2. Search for the necessary plugin.
3. Read and “Apply” privacy notice.
4. Click the “Ok” button.
5. Once installation is completed, click Restart for changes to take effect.

2.1.4 Widgets

Widgets are created by Flutter and are built using a modern framework. Widgets describe what the application should look like, and when it is changed, the widget will rebuild to the description given. Table 4 shows the core widgets that are used in the MemorEZ application. More widgets are used, and a complete list of widgets can be found here: <https://flutter.dev/docs/reference/widgets>

Table 4

Widgets and Descriptions

Widgets	Description
Text	This widget will allow the ability to create a run of text within the application.

Container	This widget allows for a rectangular visual element to be added into the application.
ElevatedButton	This widget allows for a button that will elevate its material when it is pressed.
Row, Column	This widget allows for the ability to create a layout horizontally (row) or vertically (column) within the application.
Padding	This widget gives padding, or space to its child.
Image	This widget displays an image.
Scaffold	This widget implements a Material Design visual layout structure.
Table	This widget allows for a table layout for its children.
Align	This widget aligns its children inside of itself and will adjust its size based on the size of its child.

Note. See <https://flutter.dev/docs/reference/widgets> for a complete list of widgets.

3 Software Installation

These instructions will guide the user to set up the development environment in Windows and MacOS operating systems. While some steps could be different from one system to the other, this section will provide the fundamental steps to help the user identify the correct tools. For full step-by-step instructions, please refer to the official web pages provided below for each one of the systems. Steps to get started:

- Install the Flutter SDK
- Install Android Studio
- Install the Android Emulator

3.1 Android Studio

Download and Installation

1. Navigate to <https://developer.android.com/studio> and download the the version of Arctic Fox of Android Studio for your Operating System (OS).
2. In order to install Android Studio on your computer, follow the instructions for your intended OS at <https://developer.android.com/studio/install>
3. After installing Android Studio successfully, you can continue installing the other tools that are required within android studio to get started, such as; flutter, dart, and android emulator.

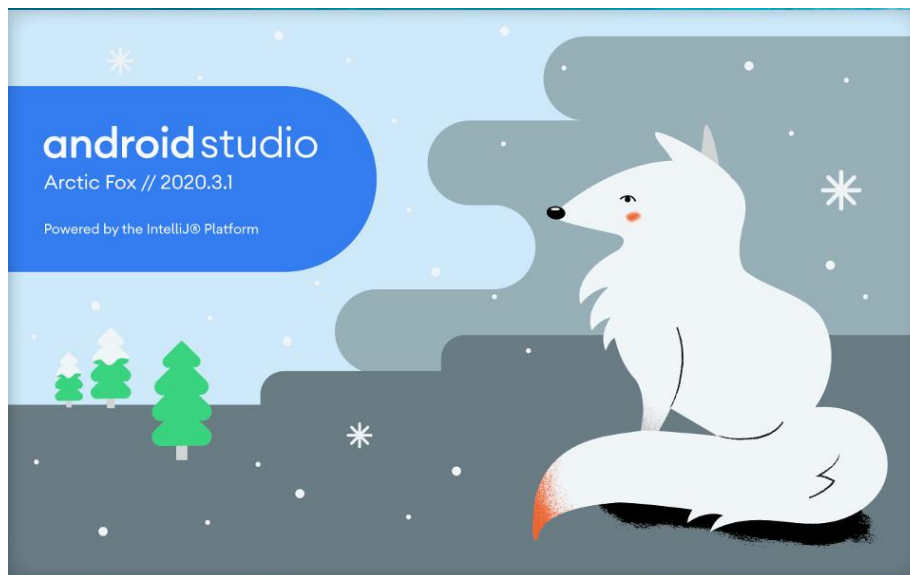
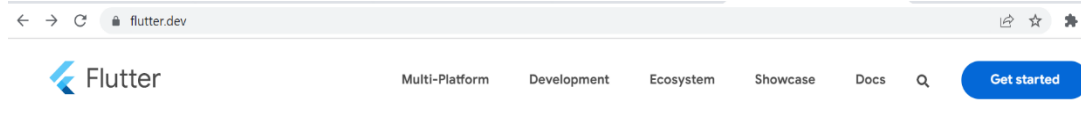


Figure 1: Android Studio

3.2 Flutter and Dart

Download and Installation

- To download Flutter, visit the official website <https://flutter.dev/>, and click on “Get Started”



- Select the correct OS and follow the installation instructions for your OS.

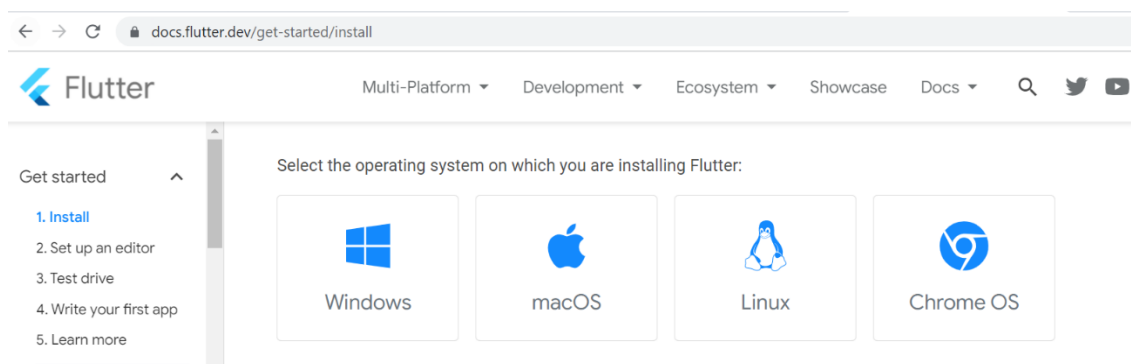


Figure 2: Installing Flutter

- After following the installation instructions, open command prompt for Windows or Terminal for MacOS
- Run “flutter doctor” to verify that your installation has been completed correctly and that Flutter is located the Android Studio installation.
- If all the installation is correctly and successfully completed, everything should be checked in green, and no issues should be found.

```
C:\Users\Selina Z>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 2.8.1, on Microsoft Windows [Version 10.0.19043.1526], locale en-US)
[✓] Android toolchain - develop for Android devices (Android SDK version 32.0.0)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2020.3)
[✓] VS Code (version 1.63.2)
[✓] Connected device (2 available)

• No issues found!
```

Figure 3 Flutter Doctor Summary from command prompt (cmd)

- If Flutter cannot locate Android Studio, run `Flutter config --android-studio-dir <android studio directory here>` to indicate the location of Android Studio.

- After successfully installing Android Studio and Flutter SDK, continue to the next step to set up the Android Studio emulator.

3.3 Flutter and Dart Plugins

Install Flutter and Dart plugins in Android Studio

1. In order to install the correct plugins in Android Studio, please follow the official instructions from <https://flutter.dev/docs/get-started/editor?tab=androidstudio>.
2. In Android Studio, go under File > Settings > Plugins > Search for “Flutter”> Apply
3. In Android Studio, go under File > Settings > Plugins > Search for “Dart”> Apply

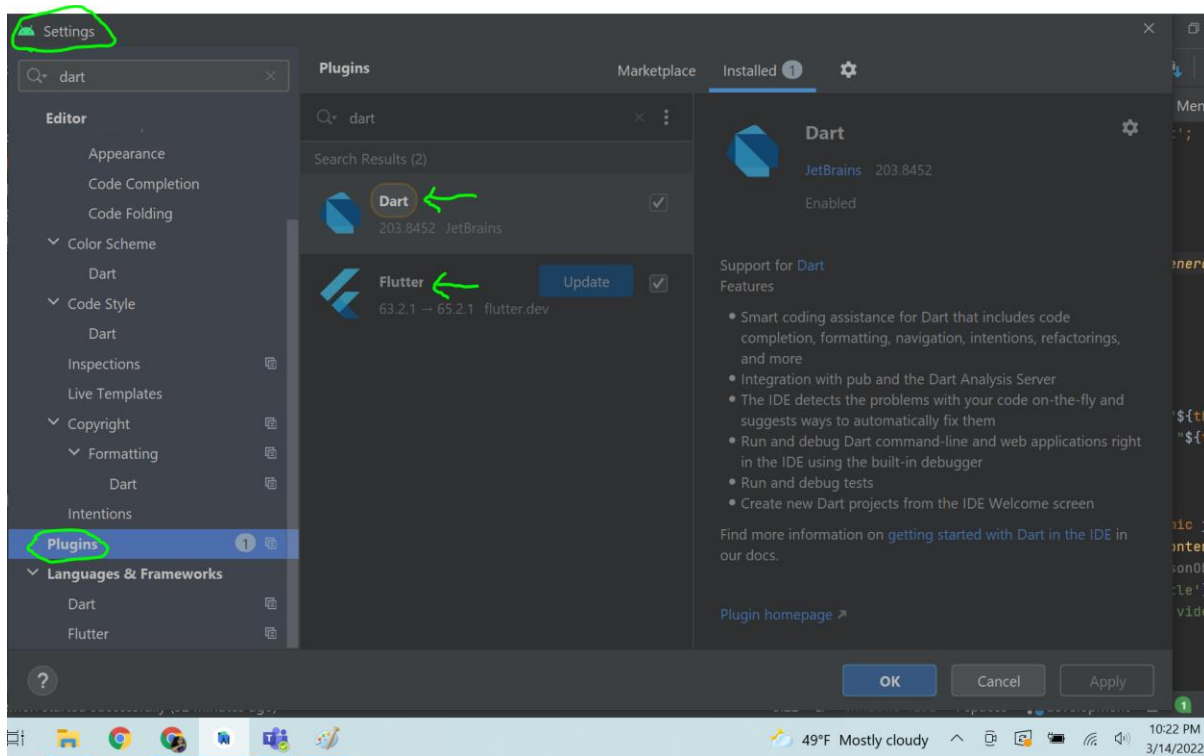


Figure 4 Flutter and Dart Plugins

4. After installing the correct plugins, the flutter environment should be ready to use.

3.4 Android Emulator

Set up your Android Emulator

1. In order to set up the emulator correctly, please follow the official instructions for your OS on the Flutter web page.
 - Windows instructions can be found at <https://flutter.dev/docs/get-started/install/windows>
 - MacOS instructions can be found at <https://flutter.dev/docs/get-started/install/macos>
2. After successfully installing the emulator, you should be able to see it integrated into the Android Studio and you should be able to run it by opening the AVD Manager from the Android Studio IDE and clicking on the green “run” button for your specifically selected Android emulator.

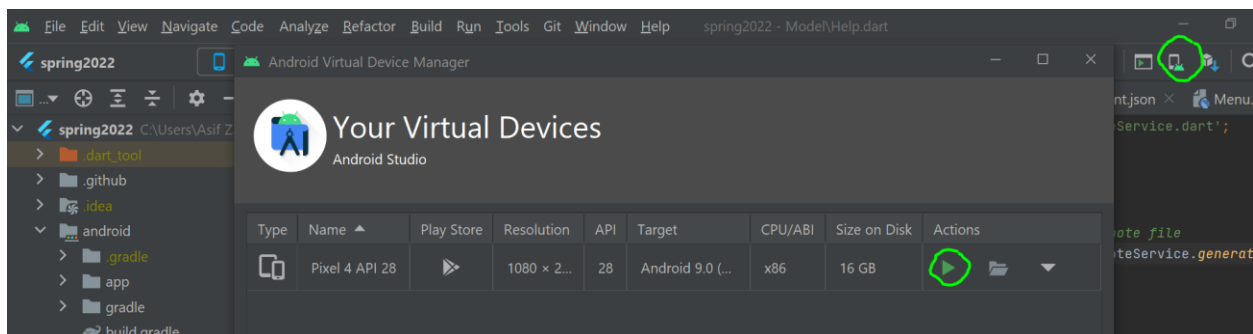


Figure 5 Virtual Devices

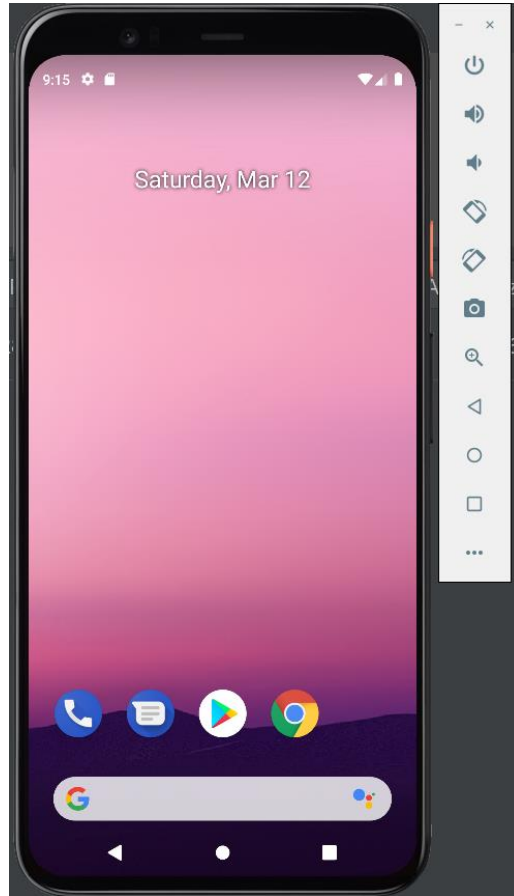


Figure 6 Android Emulator

3. After installing Flutter, Android Studio, and setting up the emulator, you must agree to the licenses of the Android SDK platform to use the development environment correctly. Open your console terminal and run the command, **flutter doctor --android-licenses**. Make sure that you have installed all other additional requirements for your specific OS.

3.5 Test Your Development Environment

Test your development tools

1. To test the development environment, create a simple Flutter application in Android Studio:

Steps:

1. Open the Android Studio IDE
 2. Select Create New Flutter Project.
 3. Select Flutter Application as the project type and click Next.
 4. Verify the Flutter SDK path specifies the SDK's location (select Install SDK, if the text field is blank).
 5. Enter a project name (for example, myapp). Then click Next.
 6. Click Finish.
 7. Wait for Android Studio to install the SDK and create the project.
2. After Android Studio has created the Flutter project you can run the application in your emulator.
 3. For more instructions to run a test drive, please follow the official instructions from <https://flutter.dev/docs/get-started/test-drive?tab=androidstudio>.

3.6 GitHub Desktop

GitHub will be used as the application code repository. This tool makes the process of development easier for the application. It is beneficial to download the desktop version called GitHub Desktop for this purpose.

Download and Install GitHub Desktop

1. Visit the official website to download the application at <https://desktop.github.com/>
2. Once downloaded, install the application and sync GitHub account with GitHub Desktop, or create a new GitHub account and sync it.
3. After setting up the GitHub account, users can access the application repository and collaborate on the code.

In order to understand how to perform the process using GitHub and Desktop GitUI for cloning, practicing creating feature branches and switching between feature and development branches will be helpful. Users can fetch origin to get the latest version of the development branch.

3.7 Task Management Tool

The team utilized Trello board as the application requirement and development task management tool. The Trello board has multiple sections for the purpose. The current Trello board consists of the following:

1. Backlog - holds all the requirements for the application from the stakeholders.
2. Sprint backlog - taken from the backlog and added to the sprint backlog for each sprint which is divided amongst all the team members who are developing the application based on the requirements.
3. In progress - requirement task cards that are currently in progress in the feature branch.
4. Problem - issues that the team members are currently facing during development.
5. Ready for branch testing - requirements that are currently complete and ready to be tested.
6. Ready to merge into the development branch - requirements that have been tested in the feature branch and currently awaiting approval from the lead developers from both teams who confirm the pull request after viewing the code before they merge it to the development branch.
7. Ready for integration testing - testing that is required to be completed successfully in the development branch as an extra precaution before merging to the main branch.
8. Ready to merge into Main branch - holds all the trello cards that are ready to be merged to the main branch by DevSecOps team.
9. Merged to Main branch - holds all the trello cards that are merged to the main branch.

MemorEz Runbooks - Combined

10. Resources - any available resources that the team is utilizing.
11. Meeting minutes - minutes from the class meeting are also added to the trello board so everyone in the class remains updated.
12. Mega Backlog (Priority level 2) - any trello cards that hold the requirements which would be a “nice to have” feature or subfeatures but fall in priority level 2 as those might not provide enough time for the development teams to implement.

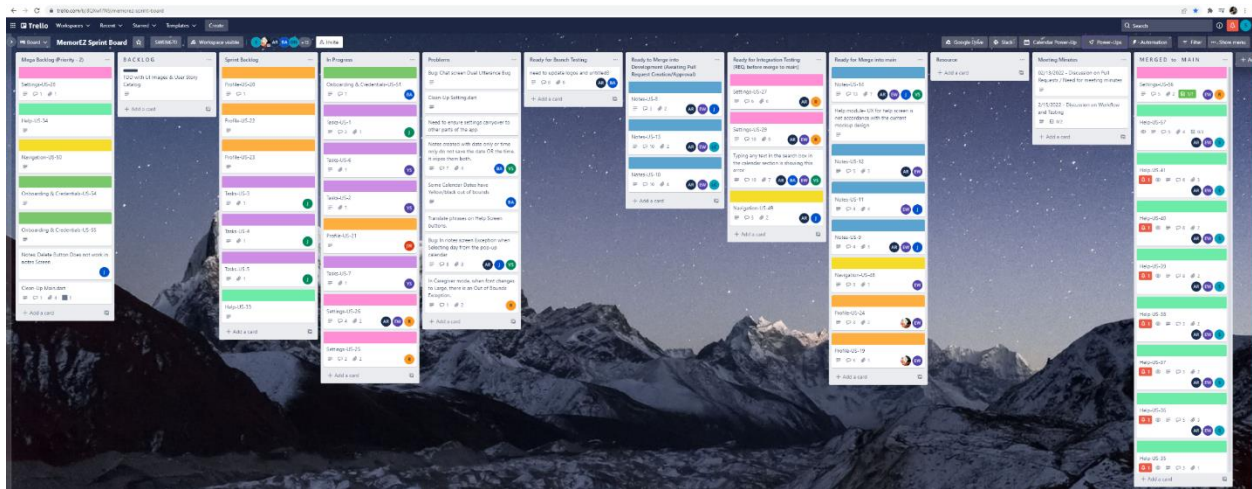


Figure 7 Trello Board Class of Spring 2022

4 Prepare the Mobile Application for Use

4.1 Cloning GitHub Repository

Cloning the repository from Github to a local computer helps minimize and fix conflict, add and remove files, as well as push commits to the overall product.

In order to complete this, the following steps must be completed:

1. Navigate to <https://github.com/umgc/spring2022>
2. Click on the green button that reads “Code”
3. Ensure the HTTPS is clicked, then click “Download to ZIP” to download a ZIP file of the cloned code

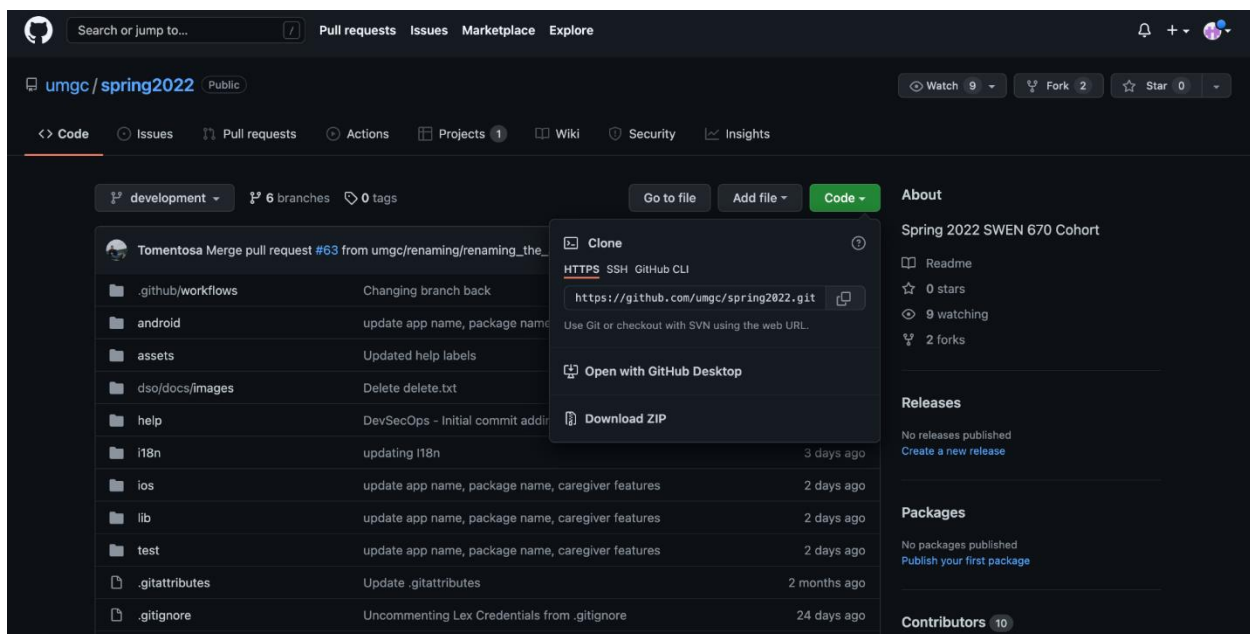


Figure 8 GitHub Download

4.2 Run the Flutter Application

To run the Flutter application on a device, use the following steps:

1. Unzip the cloned project
2. Open Android Studio
3. Click File, New, Import Project
4. Locate the unzipped file and open the project.
5. The application will then be open on your device.

5 Testing the Mobile Application

5.1 Testing Objectives

5.1.1 Unit Tests

A unit test assesses a single function/method/class. The goal is to verify the accuracy of a logical unit. Table 5 shows the Test Matrix for the General Tests.

Table 5A *Unit Tests*

General Tests:	iPhone	Android
1. Application Opens Up Success case: The application opens up to the Menu screen in Patient mode		
2. The system successfully completes onboarding credentials Success case: The system asks for Caregiver Credentials, App Settings, and gives Patient Intro Video when first opened		
3. From any page, click Menu Success case: The application will bring the user to the home screen.		
4. From any page, click on Chat Success case: The application will bring you to the chat screen.		
5. From any page, click on Help Success case: The application will navigate to the Help screen.		
Menu Screen:	iPhone	Android
6. Click on "Tasks" Success case: The screen will navigate to the Tasks section.		
7. Click on "Notes"		

Success case: The screen will navigate to the Notes section.		
8. Click on “Calendar” Success case: The screen will navigate to the Calendar section.		
9. Click on “Profile” Success case: The screen will navigate to the Profile section.		
10. Click on “Settings” Success case: The screen will navigate to the Settings section.		
Tasks Screen:	iPhone	Android
11. Double tap on Active Task Success case: Task moves to Completed Task.		
12. Double tap on Completed Task Success case: Task moves to Active Task.		
13. The user can complete a mood task Success case: The user completes the mood task and results are sent to the caregiver.		
Note Screen:	iPhone	Android
14. Search for note Success case: Searched note will appear..		
15. Add Note Success case: Note will be added		
16. Delete Note Success case: Note is deleted.		
17. Update Note Success case: Note will be updated.		
Calendar Screen	iPhone	Android

18. View the Calendar in week, month, or day Success case: Ability to view the calendar in week, month, and day mode.		
19. Click a date to see note/appointment Success case: Note/appointment appears when clicking on a day.		
20. Search for a note/appointment Success case: Note appears when searching for the note/appointment.		
Profile Screen	iPhone	Android
21. User can view self profile information Success case: User can view the self profile in read only mode.		
22. User can view contacts information Success case: User can view the contacts in read only mode.		
23. User can view care team information Success case: User can view the care team information in read only mode.		
24. User can view preferred transportation information Success case: User can view the preferred transportation in read only mode.		
Settings Screen	iPhone	Android
25. User can change the font size Success case: Users can change the font size to small, medium, or large.		
26. User can change the language Success case: User can change the language to English, Spanish, Portuguese, Chinese, or Arabic.		
27. User can change the color of the app		

Success case: User can change the color to pink or blue.		
28. User can login to caregiver mode Success case: User clicks the caregiver mode button to login to and enable caregiver mode.		
29. User can select cancel, save, or reset Success case: The user can cancel changes, save changes, or reset changes that were made..		
Chat Screen	iPhone	Android
30. User can enable/disable the Speech-to-text feature Success case: The user can click the microphone to enable the speech-to-text feature.		
31. Manage a note Success case: User can make changes to notes, or add notes using the speech-to-text feature.		
32. Trigger words Success case: The user can use trigger words to enable the creation of a note.		
Help Screen	iPhone	Android
33. Assistance on Tasks Success case: The user can watch videos of the overview of the tasks option, how to create a task, and how to edit a task.		
34. Assistance on Notes Success case: The user can watch a video on the overview of notes, and other topics on notes.		
35. Assistance on Calendar Success case: The user can watch a video on the overview of the calendar, and other topics on the calendar.		
36. Assistance on Profile		

Success case: The user can watch a video on the overview of the profile, and other topics on the profile.		
37. Assistance on Settings Success case: The user can watch a video on the overview of the settings, and other topics on the settings.		
38. Assistance on Chat Success case: The user can watch a video on the overview of the chat, and other topics on the chat.		

5.1.2 Integration Tests

Integration testing will occur in the scenarios that will combine the unit tests together to form one application. The main focus of the integration tests will be the integration scenarios between patient mode, and caregiver mode. In addition to this, ensuring the NLU has been integrated successfully has been run, as well as translation services. Below are a list of integration tests run, ensuring caregiver mode and patient mode are successfully integrated, as well as ensuring a functioning NLU and translation services.

Table 5B *Integration Tests*

Integration Tests	iPhone	Android
**Integration with NLU:		
1. Application handles successful response from NLU API. Success case: Application responds accordingly.		
2. Application handles all non-success responses from NLU API. Success case: Error is given, but handled in a graceful way.		
**Integration with Translation Services:		
3. Application handles successful response from Translation Services. Success case: Application responds accordingly.		
4. Application handles all non-success responses from Translation Services.		

Success case: Error is given, but handled in a graceful way.		
**Integration with Caregiver:		
5. Application handles successful link with caregiver mode. Success case: Successfully alters what was changed in Caregiver mode (view, settings, etc.).		
6. Application handles all non-success responses Caregiver mode. Success case: Error is handled gracefully		

5.2 Testing Procedures

- Choose Windows (version 10 or later) operating system (64-bit with a x64 based processor laptop) or iOS operating system.
- Install Microsoft Teams – to be used in collaboration with RememberAll team, DevOps and team FlutteringMind members.
- Install Flutter version 2.8.1.
- Install Android Studio version 4.3 with Flutter and Dart plugins.
- On Mac, Install xCode.
- Install GitHub - to be used in collaboration with RememberAll team, DevOps and team FlutteringMind members.
- Clone the GitHub repository: <https://github.com/umgc/spring2022>
- Execute the application
- Join the Trelloboard MemorEZ Sprint Board
- Manually branch test each feature of the application, when moved to the appropriate column
- Document the test results
- If any of the required tests fail, fix the issues in collaboration with RememberAll team, DevOps and FlutteringMind team members.
- Rerun the updated failed tests
- Document test results
- Run Integration testing prior to commit to main
- Document the test results
- If any of the required tests fail, fix the issues in collaboration with RememberAll team, DevOps and FlutteringMind team members.
- Rerun the updated failed tests
- Document test results
- Documentation - test report

6 Troubleshooting

After installing and starting using the emulator in the development environment, users might find some issues when using Flutter and Android Studio. This section will cover and go over some of the most common issues and troubleshooting solutions to problems of flutter. Some of the troubleshooting steps provided in this section may be different depending on the OS used. For more specific solutions to problems, visit the official Flutter website at <https://flutter.dev/docs> and/or Android Studios at <https://developer.android.com/studio/troubleshoot>.

6.1 Emulator Freezing

While running the emulator, if it gets frozen, the user can launch “Cold Boot Now”. This helps clearing memory (RAM) of the internal data. Also, after any type of troubleshooting is performed, it is recommended to perform a cold boot to clear caches and RAM contents, so when the user runs the app next time, the app runs without any issues. Steps are below:

1. If the user already has a project opened in Android Studio, click on the ADV manager from the top menu as shown in the screenshot below or click on Tools > ADV Manager.
2. After opening the ADV Manager, the user has to find the emulator they want to use and click on the dropdown arrow > Cold Boot Now.

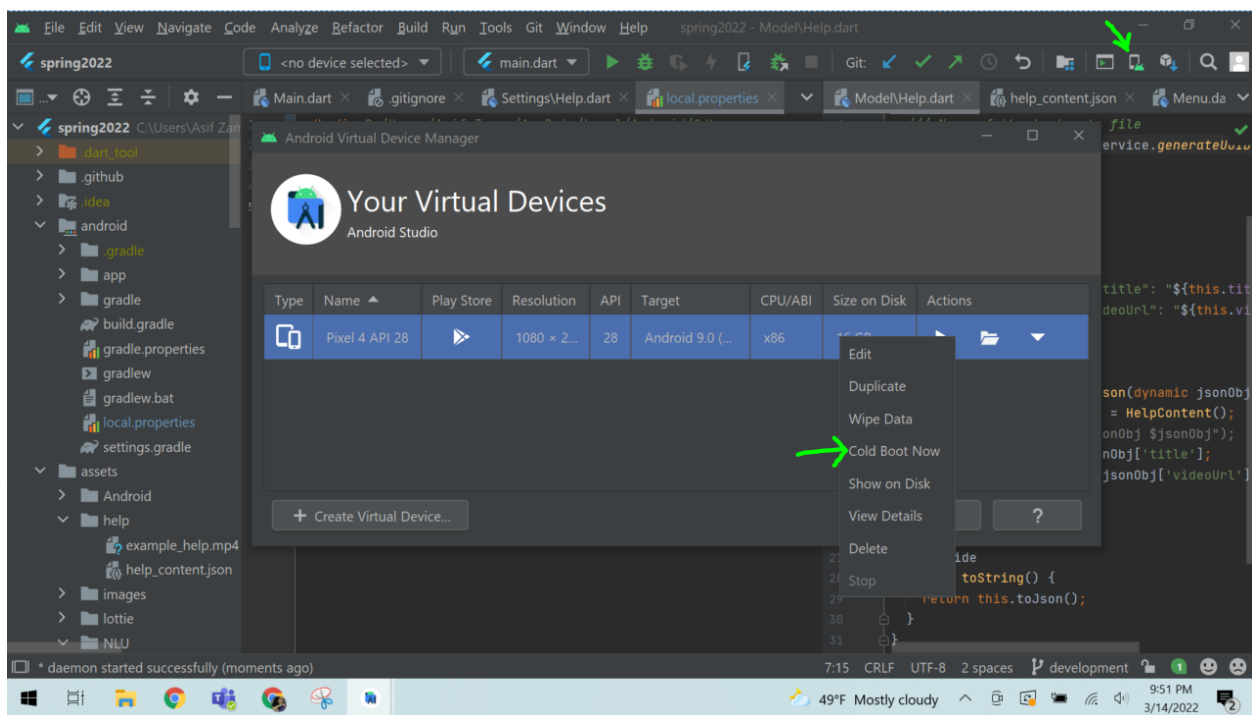


Figure 9 ADV Manager, Virtual Devices

6.2 Emulator Not Responding

After installing the mobile emulator in android studio, user may encounter issues when trying to start the emulator and/or after using it a couple of times. Generally, the emulator is unable to execute/run properly due to configuration/resources used from the local/host machine. One of the troubleshooting solutions is to wipe the emulator data and return it to the same state as when it was first defined.

1. In order to perform wiping out data, the emulator should not be running. If it is running, the user has to exit from it.
2. If the user already has a project opened in Android Studio, click on the ADV manager from the top menu as shown in the screenshot below or click on Tools > ADV Manager.
3. After opening the ADV Manager, the user has to find the emulator they want to use and click on the dropdown arrow > Wipe Data.
4. After wiping your device data, the user can perform a “Cold Boot Now”. Click on the dropdown arrow > Cold Boot Now.

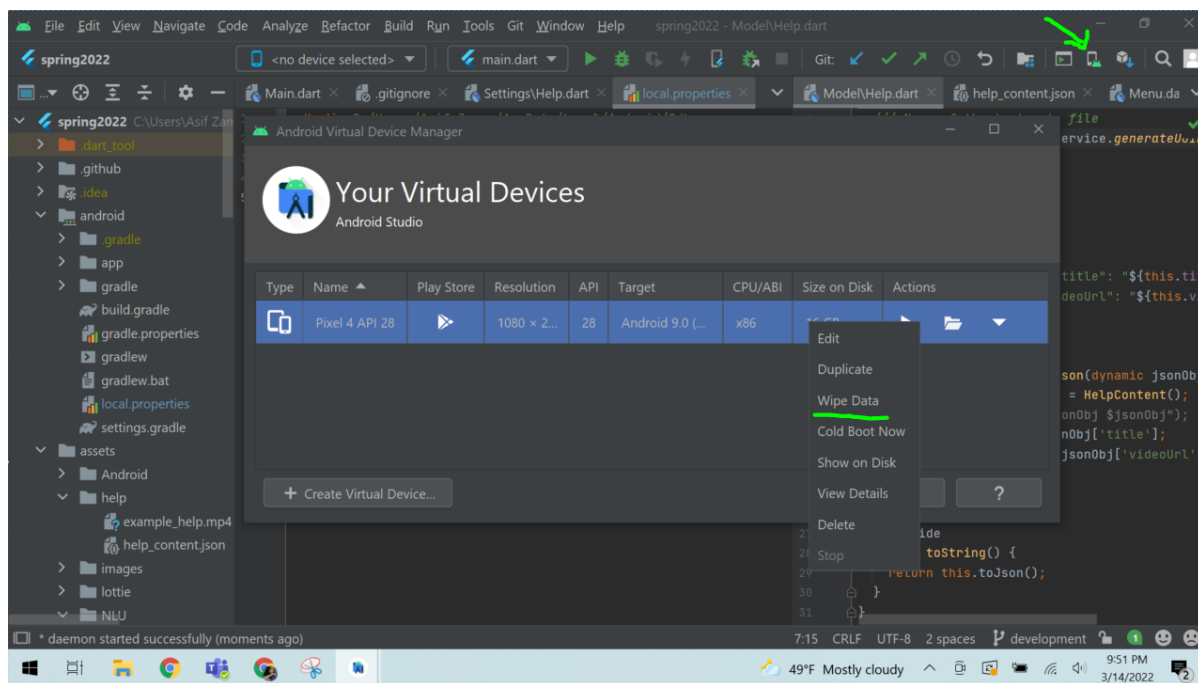


Figure 10 ADV Manager, Virtual Devices

6.3 Out of Memory Error

If the local machine used to run the emulator does not have sufficient resources to handle the virtual device, this would be a common error. In this case, the user has to close all unnecessary applications running in the background to free up virtual memory. It is also helpful to increase the assigned memory to the virtual emulator by following these steps:

1. On Android Studio click on File > Settings to open the general settings.
2. On the general settings screen click on Appearance & Behavior > System Setting > Memory Settings.
3. Customize the amount of memory to the recommended 2,048 MB.
4. After assigning the memory, click Apply > Ok to save the changes.
5. Restart your Android Studio. For more information visit <https://developer.android.com/topic/performance/memory>

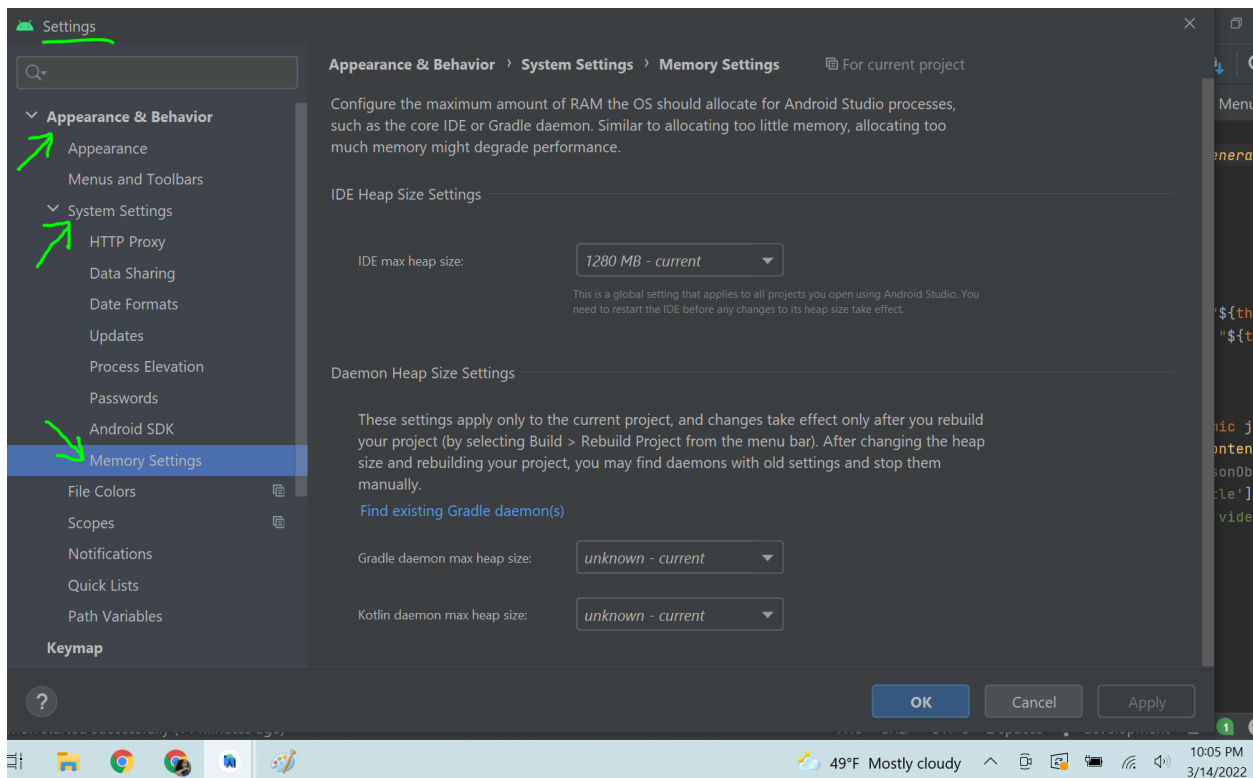


Figure 11 Settings

6.4 Stuck at Running Gradle Task 'AssembleDebug'...

This is a common issue when trying to start the app in the emulator. If the user receives the message "Running Gradle task 'assembleDebug'... (this is taking an unexpectedly long time.)," consider the following steps for troubleshooting:

1. Open the Flutter terminal.
2. Navigate to the root directory of the Flutter project.
3. On your Flutter project directory run this command `./gradlew clean`.
4. The user can either build the Gradle from the directory or let Android Studio build it when you run the app. In order to build Gradle from the directory run `./gradlew build`. User can also combine both commands to clean and build the Gradle by running `./gradlew clean build`. (koderstory, 2020).

6.5 Dependency Errors

Another common issue is to get unexpected dependency errors in the code. This generally occurs when multiple versions of a library get imported to the project. A common solution to this problem is to clear the Flutter build files and re-import the libraries. Steps to follow are below:

1. Open the Flutter terminal.
2. Navigate to the root directory of the Flutter project.
3. Run the command 'flutter clean' to remove all the dependencies.
4. Now re-import the newest libraries to the correct dependencies by running flutter "Get dependencies" and "Upgrade dependencies".

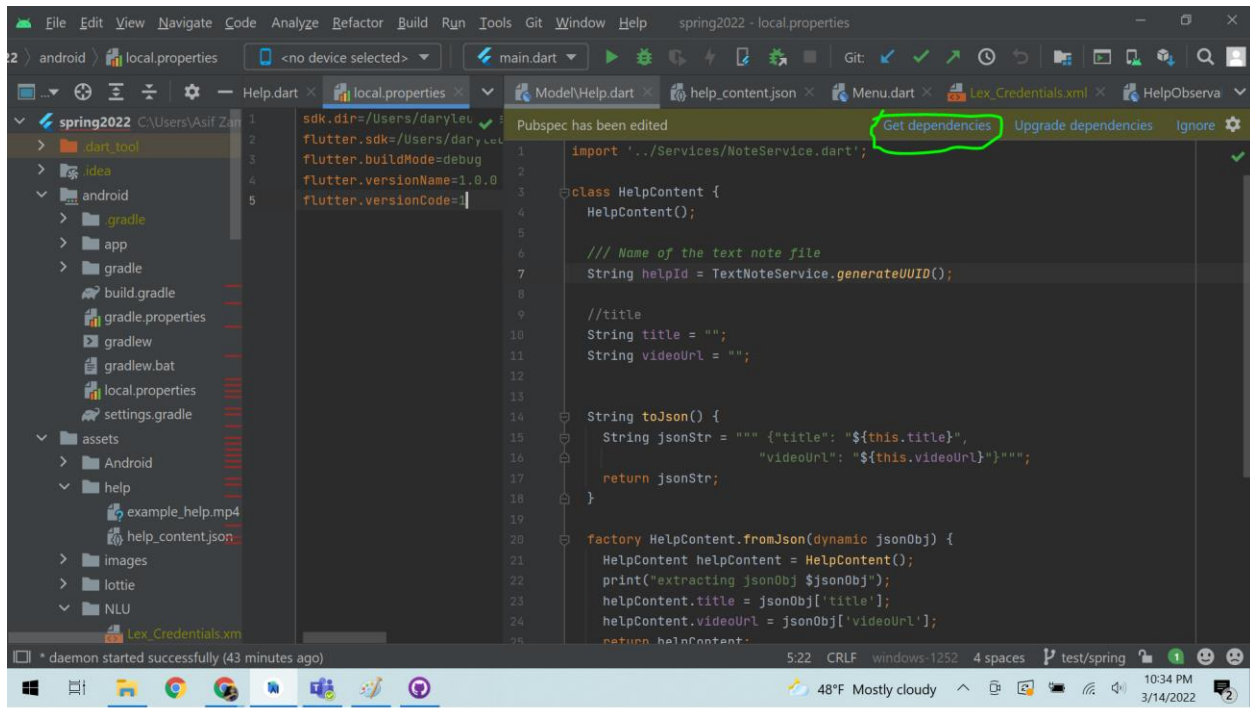


Figure 12 Flutter Project

7 Appendices

7.1 Credits

The following members contributed to the development of this software:

- Dr. Mir Mohammed Assadullah (Product Owner, Stakeholder)
- Dr. Andrea Evangelista (Subject Matter Expert)
- James Eble
- Brian Avadikian
- Selina Zaman
- Lizset Chavez Chacaltana
- Robert Edwards
- Joshua Fischer
- Joseph Jewell
- Sean LaMonica
- Andrew Nicolette
- Anusha Ramanan
- Yusufu Sanu
- Vivek Singh
- Vanessa Stringer
- Daryle Urrea
- Eyob Woldehana
- Robert Wren

7.2 Credit to Previous Cohorts

Johnny Lockhart, Jeroen Soeurt, Michelle Monfort, Robert Wilson, Ayodeji Famudehin, Chauntika Broggin, Christian Ahmed, Mitchell Olshansky, Mod Drammeh, Nicholas Ballo, Shawn Kelly, Raul Benavides, Maddison Dunning, Alec Baileys, Benjamin Cushing, Elshaday Mesfin, Tyler Puschinsky, Michael Le, Debashis Jena, Austin Johnson, Prince Antwi Aboagye, Didimus Kimbi, Damion Sevilla, Rebecca Johnson, Addisu Worku, Matthew Setiawan, Obinna Okonkwo, Andrew Rohn, Joseph Kalfus, Firehiwot Chari, Eskedar Endashw, Malik Webster, Leela Subramanian, Presley Muwan, Christian Cruz Jimenez, Daniel Avery, Karen Crumb, Kevin Bell, Sami Salim, Teresa Balbi, Endalkachew Girma

Section 2: Caregiver Mode

Information in section one is copied from team RememberAll's Deployment and Operations Guide (Runbook)

1 Introduction

1.1 Purpose

The purpose of this Deployment and Operations Guide (Runbook) is to describe the installation and configuration process as part of the deployment for the MemorEzApp. It will outline the plan and steps needed to successfully deploy the MemorEz App.

1.2 Intended Audience and Reading Suggestions

The intended audience for this guide is software developers, project managers, and technical stakeholders as DevOps team and any other stakeholders participating in the installation/deployment of the MemorEz App. This document is layout in a way that is easy to follow outlining each step-in deployment in detail to facilitate the deployment of the MemorEz App. It is recommended that each section is read thoroughly in order before moving the next section. Some previous knowledge of Flutter and GitHub is needed to facilitate the deployment of the application.

1.3 Technical Project Stakeholders

Table 1 shows the project stakeholders for the MemorEz App:

Table 1: Project Stakeholders

Name	Role
Dr. Mir Assadullah	Stakeholder (Project Owner)
Roy Gordon	Stakeholder (Project Advisor)
Robert Wilson	Stakeholder (Project Advisor)
Daniel Avery	Stakeholder (Project Advisor)
James Eble	General Project Manager
Brian Avadikian	RememberAll Project Manager
Genet Asmelash	Business Analyst

MemorEz Runbooks - Combined

Vivek Singh	Business Analyst
Lizset Chavez	Developer
Johnnie Webb	Developer
Robert Edwards	Developer
Yusufu Sanu	Developer
Eyob Woldehana	Business Analyst/ Tester
Fluttering Mind	Co-Development Team

1.4 References

Presley Muwan, Karen Crumb, Kevin Bell, Teresa Balbi, Sami Salim, Daniel Avery & Christian Cruz Jimenez. (2021, October 18). DEPLOYMENT AND OPERATIONS GUIDE (RUNBOOK) MEMORY MAGIC APP. <https://umgc-cappms.azurewebsites.net/Previousprojects>. Retrieved March 15, 2022, from <https://umgc-cappms.azurewebsites.net/previousprojects>

1.5 Definitions, Abbreviations, and Acronyms

Table 2: Acronyms, Abbreviations, and Definitions

Acronyms and Abbreviations	Definitions
PM	Project Manager
BA	Business Analyst
DSO	DevSecOps
STML	Short Term Memory Loss
UI	User Interface
NLP	Natural Language Processing
NLU	Natural Language Understanding
OS	Operating System
iOS	iPhone operating system
Flutter CLI	Flutter Command Line Tool

API	Application Program Interface
TTS	Text to Speech
UMGC	University of Maryland Global Campus

2 Mobile Application

2.1 Features, Packages, Plugins, and Widgets

2.1.1 Features

- **Simplicity:** Easy navigation menu and icons provides an easy user-friendly experience to quickly engage in the required tasks and perform necessary actions.
- **Performance:** Loading screen time is quick and fast that provides a smooth uninterrupted user experience.
- **Cross-platform:** Application is cross-platform compatible such as Android and iOS because of the same code-base. Since the application is able to perform the necessary functions across different platforms maintenance because easier across both platforms because of the same codebase.
- **Hot reload:** One of the neat and instant features is the hot reload, which makes it easier for the code to load instantly on the emulator to reflect the changes. This makes it faster on debugging and fixing layout and other related issues.
- **Security:** Most of the sensitive information such as personal and health information is securely managed by the app to comply with the Health Insurance Portability and Accountability Act (HIPPA). All information is being saved in the secured served and nothing is being shared or released to the third party
- **Search Options:** Enables the user to search for pertinent information throughout the module that is being used.
- **Notifications:** The app enables the users to be notified of crucial health-related tasks and reminders, which could be customized, and disabled as desired.
- **Open-Source:** Since, flutter is open source, there are many widgets and designs that was imported can be easily updated if required. Flutter has a big community and user base, which make it easier to discuss or share ideas, questions, and comments with a quick response from professional all over the world.

2.1.2 Packages

As an open-source platform, most packages are mostly created by professional developers from all over the world and shared through Flutter and Dart ecosystems. This allows developers to quickly build an app without having to develop everything from scratch” (Peter, n.d., para. 2). Packages are published in pub.dev; this webpage contains packages and dependencies that are compatible with Flutter (Muwan et al., 2021). They are contained in the directory with a pubspec.yaml and can be divided into two types, application package, and library package. They also have dependencies with other packages such as FocusDetector, which used 0.1.2 version package. The two types of packages are as follows:

- **Application packages:** All application packages are contained in main.dart file
- **Library packages:** All the other packages that are created in the lib folder are library packages.

2.1.2.1 Installing a Package Dependency into an App

The following are the steps to install package dependency (Muwan et al., 2021):

1. Depend
 - Open the pubspec.yaml file located inside the app folder and add a specific package under dependencies.
2. Install
 - From the terminal: Run flutter pub get.
OR
 - From Android Studio/IntelliJ: Click “**Packages 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.
3. Import
 - Add a corresponding import statement in the Dart code.

2.1.3 Plugins

The installation of Flutter and Dart plugins are listed below according to the platform (Muwan et al., 2021):

2.1.3.1 macOS Installation

Use the following instructions for macOS:

1. Start Android Studio.
2. Open plugin preferences (**Preferences > Plugins**).
3. Select the Flutter plugin and click **Install**.
4. When prompted click **Yes** to install the Dart plugin.
5. When prompted click **Restart**.

2.1.3.2 Windows Installation

Use the following instructions for Windows:

1. Open plugin preferences (**File > Settings > Plugins**).
2. Select **Marketplace**, select the Flutter plugin and click **Install**.
3. Read/accept privacy notice.
4. When prompted, click Yes to install plugins
5. Once installation is completed, click Restart for changes to take effect.

2.2 Features, Packages & Plugins

Table 3: List of all the features, packages and Plugins used on the application

Package	Reference
AdminPage	import 'package:untitled3/Screens/AdminPage.dart';
async	import 'dart:async';
avatar_glow	import 'package:avatar_glow/avatar_glow.dart';
Boarding	import 'package:untitled3/Screens/Onboarding/Boarding.dart';
bubble_special_one	import 'package:chat_bubbles/bubbles/bubble_special_one.dart';
Calendar	import 'package:untitled3/Screens/Calendar/Calendar.dart';
CalendarEvent	import 'package:untitled3/Model/CalendarEvent.dart';
CalendarFormatBar	import 'package:untitled3/Screens/Calendar/CalendarFormatBar.dart';
CalendarUtility	import 'package:untitled3/Utility/CalendarUtility.dart';
CalenderObservable	import 'package:untitled3/Observables/CalenderObservable.dart';
CancelButton	import 'package:untitled3/Components/CancelButton.dart';
care_team_card	import 'package:untitled3/Screens/Profile/care_team_card.dart';
ChatBubble	import 'package:untitled3/Screens/Mic/ChatBubble.dart';
Checklist	import 'Checklist.dart';
CheckListObservable	import 'package:untitled3/Observables/CheckListObservable.dart';
chewie	import 'package:chewie/chewie.dart';
CloudSetup	import 'package:untitled3/Screens/Onboarding/CloudSetup.dart';
collection	import 'dart:collection';
comHelper	import 'package:untitled3/Comm/comHelper.dart';
Constant	import '../Utility/Constant.dart';

	import 'package:untitled3/Utility/Constant.dart';
contact_card	import 'package:untitled3/Screens/Profile/contact_card.dart';
convert	import 'dart:convert';
CreateAdmin	import 'package:untitled3/Screens/CreateAdmin.dart';
cupertino	import 'package:flutter/cupertino.dart';
date_symbols	import 'package:intl/date_symbols.dart';
DbHelper	import '../DatabaseHandler/DbHelper.dart';
DbHelper	import 'package:untitled3/DatabaseHandler/DbHelper.dart';
dcdg	import 'package:dcdg/dcdg.dart';
EncryptionUtil	import 'package:untitled3/Utility/EncryptionUtil.dart';
ffi	import 'dart:ffi';
FileUtil	import '../Utility/FileUtil.dart';
flutter_local_notifications	import 'package:flutter_local_notifications/flutter_local_notifications.dart';
flutter_localizations	import 'package:flutter_localizations/flutter_localizations.dart';
flutter_mobx	import 'package:flutter_mobx/flutter_mobx.dart';
flutter_search_bar	import 'package:flutter_search_bar/flutter_search_bar.dart';
flutter_tts	import 'package:flutter_tts/flutter_tts.dart';
flutternotifications	import 'package:flutternotifications/flutternotifications.dart';
font_awesome_flutter	import 'package:font_awesome_flutter/font_awesome_flutter.dart';
FontUtil	import 'Utility/FontUtil.dart';
foundation	import 'package:flutter/foundation.dart';
genLoginSignupHeader	import 'package:untitled3/Comm/genLoginSignupHeader.dart';
genTextFormField	import 'package:untitled3/Comm/genTextFormField.dart';
Help	import 'package:untitled3/Model/Help.dart'; import 'package:untitled3/Screens/Settings/Help.dart';

HelpObservable	import 'package:untitled3/Observables/HelpObservable.dart';
HelpService	import 'package:untitled3/Services/HelpService.dart';
Home	import 'package:untitled3/Screens/Home.dart';
HomePage	import 'package:untitled3/Screens/HomePage.dart';
i18n	import 'generated/i18n.dart'; import 'package:untitled3/generated/i18n.dart';
image_picker	import 'package:image_picker/image_picker.dart';
imezone	import 'package:timezone/timezone.dart' as tz;
intl	import 'package:intl/intl.dart';
io	import 'dart:io';
json_annotation	import 'package:json_annotation/json_annotation.dart';
latest	import 'package:timezone/data/latest.dart' as tz;
LocaleService	import 'package:untitled3/Services/LocaleService.dart';
LoginPage	import 'package:untitled3/Screens/LoginPage.dart';
Main	import 'package:untitled3/Screens/Main.dart';
material	import 'package:flutter/material.dart';
math	import 'dart:math' as math;
Menu	import 'package:untitled3/Screens/Menu/Menu.dart';
MenuObservable	import 'package:untitled3/Observables/MenuObservable.dart';
Mic	import 'package:untitled3/Screens/Mic/Mic.dart';
MicObservable	import 'package:untitled3/Observables/MicObservable.dart';
mobx	import 'package:mobx/mobx.dart';
NLUAction	import 'package:untitled3/Model/NLUAction.dart';
NLULibService	import 'package:untitled3/Services/NLU/Bot/NLULibService.dart';
NLUResponse	import 'package:untitled3/Model/NLUResponse.dart';
NLUState	import 'package:untitled3/Model/NLUState.dart';
Note	import '../Model/Note.dart';

	import 'Note/Note.dart'; import 'package:untitled3/Screens/Note/Note.dart';
NoteObservable	import '.././Observables/NoteObservable.dart'; import 'package:untitled3/Observables/NoteObservable.dart';
NoteSearchDelegate	import 'package:untitled3/Screens/Note/NoteSearchDelegate.dart';
NoteService	import 'package:untitled3/Services/NoteService.dart';
NoteTable	import 'NoteTable.dart';
NotificationObservable	import 'package:untitled3/Observables/NotificationObservable.dart';
NotificationScreen	import 'package:untitled3/Screens/NotificationScreen.dart';
OnboardObservable	import 'package:untitled3/Observables/OnboardObservable.dart';
painting	import 'package:flutter/painting.dart';
path	import 'package:path/path.dart';
path_provider	import 'package:path_provider/path_provider.dart';
peech_to_text	import 'package:speech_to_text/speech_to_text.dart';
Permission	import 'package:untitled3/Screens/Onboarding/Permission.dart';
permission_handler	import 'package:permission_handler/permission_handler.dart';
profile_card	import 'package:untitled3/Screens/Profile/profile_card.dart';
profile_constants	import 'package:untitled3/Screens/Profile/profile_constants.dart';
provider	import 'package:provider/provider.dart';
qflite	import 'package:sqflite/sqflite.dart';
rendering	import 'package:flutter/rendering.dart';
SaveNote	import 'SaveNote.dart';
ScreenNavigator	import '.././Observables/ScreenNavigator.dart'; import 'package:untitled3/Observables/ScreenNavigator.dart';

SelectLanguage	import 'package:untitled3/Screens/Onboarding/SelectLanguage.dart';
services	import 'package:flutter/services.dart';
Setting	import '../Model/Setting.dart'; import 'package:untitled3/Model/Setting.dart'; import 'package:untitled3/Screens/Settings/Setting.dart'; import 'Settings/Setting.dart';
SettingObservable	import 'package:untitled3/Observables/SettingObservable.dart';
SettingService	import 'package:untitled3/Services/SettingService.dart';
share_plus	import 'package:share_plus/share_plus.dart';
shared_preferences	import 'package:shared_preferences/shared_preferences.dart';
SplashScreen	import 'Screens/Splash/SplashScreen.dart';
SyncToCloud	import 'package:untitled3/Screens/Settings/SyncToCloud.dart';
table_table_calendar	import 'package:table_calendar/table_calendar.dart';
tasks	import 'package:untitled3/Screens/Tasks/tasks.dart';
TasksObservable	import 'package:untitled3/Observables/TasksObservable.dart';
ThemeUtil	import 'package:untitled3/Utility/ThemeUtil.dart'; import 'Utility/ThemeUtil.dart';
timeago	import 'package:timeago/timeago.dart' as timeago;
TranslationService	import 'package:untitled3/Services/TranslationService.dart';
translator	import 'package:translator/translator.dart';
Trigger	import 'package:untitled3/Screens/Settings/Trigger.dart';
ui	import 'dart:ui';
UpdateAdmin	import 'package:untitled3/Screens/UpdateAdmin.dart';
UserModel	import 'package:untitled3/Model/UserModel.dart';
UserProfile	import 'Profile/UserProfile.dart';
uuid	import 'package:uuid/uuid.dart';
Video_Player	import 'package:untitled3/Utility/Video_Player.dart';

video_player	import 'package:video_player/video_player.dart';
VideoPlayer	import 'package:untitled3/Screens/Components/VideoPlayer.dart';
ViewNote	import 'package:untitled3/Screens/Note/ViewNote.dart';
VoiceOverTextService	import 'package:untitled3/Services/VoiceOverTextService.dart';
Walkthrough	import 'package:untitled3/Screens/Onboarding/Walkthrough.dart';
widgets	import 'package:flutter/widgets.dart';

2.2.1 Widgets

Flutter widgets are basic structural elements such as buttons, icons, texts, images, or tables that enhance the user interface in an app. In Flutter widgets are structured and layered that are arranged in widget tree in forms of parent and child relationship with the app being the topmost layer. There are two types of widgets: a stateless widget that doesn't store any value to be changed and the second is stateful which creates a state object to track the value that may or may not change. When a widget holds several other widgets then it is known as a container widget.

The basic widgets along with its description are tabulated below in table 4 (Muwan et al., 2021):

Table 4: Widgets and Descriptions

Widgets	Description
Text	This widget lets you create a run of styled text within your application.
Container	This widget lets you create a rectangular visual element.
ElevatedButton	A filled button whose material elevates when pressed.
Row, Column	These flex widgets let you create flexible layouts in both the horizontal (Row) and vertical (Column) directions.
Padding	This widget that insets its child by the given padding.
Image	This widget that displays an image.

Scaffold	Implements the basic Material Design visual layout structure. This class provides APIs for showing drawers, snack bars, and bottom sheets.
Table	A widget that uses the table layout algorithm for its children.
DropDownButton	Shows the currently selected item and an arrow that opens a menu for selecting another item
Align	This widget aligns its child within itself and optionally sizes itself based on the child's size.

Table 5: List of all widgets used in the application along with reference

Widget	Reference
_buildHourColumn	Widget _buildHourColumn(EdgeInsetsDirectional additionalPadding, Widget selectionOverlay) {...}
_buildPickerNumberLabel	Widget _buildPickerNumberLabel(String text, EdgeInsetsDirectional padding) {...}
_buildTopBar	Widget _buildTopBar(
_buildActionBar	Widget _buildActionBar() {...}
_buildAmPmPicker	Widget _buildAmPmPicker(double offAxisFraction, TransitionBuilder itemPositioningBuilder, Widget selectionOverlay) {...}
_buildBottomBar	Widget _buildBottomBar {...}
_buildControls	Widget _buildControls{...}
_buildDayPicker	Widget _buildDayPicker(double offAxisFraction, TransitionBuilder itemPositioningBuilder, {...}
_buildHitArea	Widget _buildHitArea() {...}
_buildHourPicker	Widget _buildHourPicker(EdgeInsetsDirectional additionalPadding, Widget selectionOverlay) {...}
_buildLive	Widget _buildLive(Color iconColor) {...}

_buildMinuteColumn	Widget _buildMinuteColumn(EdgeInsetsDirectional additionalPadding, Widget selectionOverlay) {...}
_buildMinutePicker	Widget _buildMinutePicker(EdgeInsetsDirectional additionalPadding, Widget selectionOverlay) {...}
_buildMonthPicker	Widget _buildMonthPicker(double offAxisFraction, TransitionBuilder itemPositioningBuilder, Widget selectionOverlay) {...}
_buildOptionsButton	Widget _buildOptionsButton() {...}
_buildPlayerWithControls	Widget _buildPlayerWithControls(
_buildPosition	Widget _buildPosition(Color iconColor) {...}
_buildProgressBa	Widget _buildProgressBa() {...}
_buildRemaining	Widget _buildRemaining(Color iconColor) {...}
_buildSubtitles	Widget _buildSubtitles(Subtitles subtitles) {...}
_buildSubtitles	Widget _buildSubtitles(BuildContext context, Subtitles subtitles) {...}
_buildSubtitleToggle	Widget _buildSubtitleToggle() {...}
_buildYearPicker	Widget _buildYearPicker(double offAxisFraction, TransitionBuilder itemPositioningBuilder, Widget selectionOverlay) {...}
_changeScreen	Widget _changeScreen(screen, index) {...}
build	Widget build(BuildContext context) {...}

3 Software Installation

This section of the runbook will describe the steps required to initialize a development environment for MemorEz. Android studio will be used as the IDE for this example, and GitHub will be used as the revision control device.

3.1 Android Studio

Download and Installation

1. Navigate to <https://developer.android.com/studio/archive> and download the latest version of Android Studio **Arctic Fox** for your Operating System (OS).

-NOTE-
Installing the **BumbleBee** version of android studio was reported by the development teams as disrupting the use of emulators. This runbook recommends using the latest available version of **ArcticFox**
-NOTE-

Android Studio download archives

This page provides an archive of Android Studio releases.

However, we recommend that you download the [latest stable version](#) or the [latest preview version](#).

Before downloading, you must agree to the following terms and conditions.

Figure 1: Android studio archives

Page Break

2. After reading and understanding the Android Studio License Agreement, scroll to the bottom and click the “Agree to Terms” button.

MemorEz Runbooks - Combined

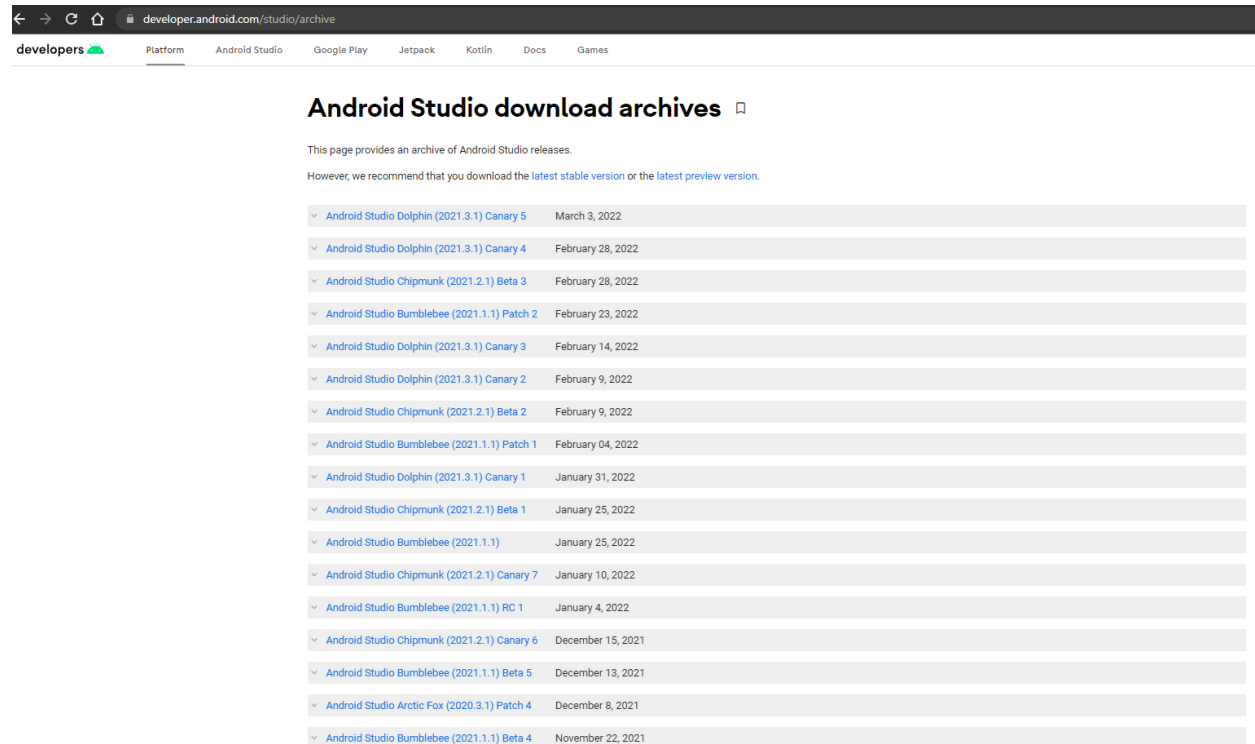


Figure 2: Selecting Android Studio Arctic Fox (2020.3.1) Patch 4

- Expand the link for Android Studio Arctic Fox (2020.3.1) Patch 4 and select the appropriate Installer for your development operating system. (This example will show a Windows (64-bit) Installation.

Installers
 Windows (64-bit): [android-studio-2020.3.1.26-windows.exe](#) (959108544 bytes)
 Chrome OS: [android-studio-2020.3.1.26-cros.deb](#) (852022056 bytes)
 Mac: [android-studio-2020.3.1.26-mac.dmg](#) (996589573 bytes)
 Mac (Apple silicon): [android-studio-2020.3.1.26-mac_arm.dmg](#) (993738651 bytes)

Figure 3: Available Installers for Android Studio Arctic Fox (2020.3.1) Patch 4

- Once downloaded, run the executable as an administrator to Install the IDE.

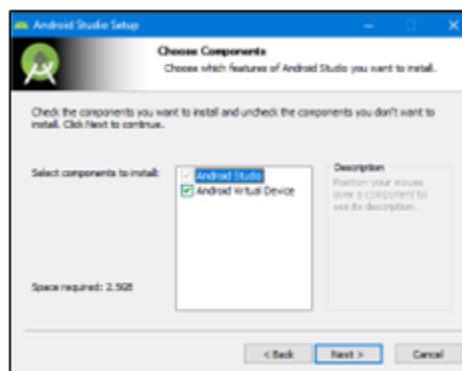


Figure 4: Component selection for Android Studio Installation

5. If you intend to use Virtual Devices like emulator to test MemorEz be sure that the “Android Virtual Devices” box is checked. Click next to continue through choosing the installation directory for the IDE.

3.2 Dart and Flutter

Download and Installation

To download Flutter, visit the official website <https://docs.flutter.dev/get-started/install>, and select your operating system.

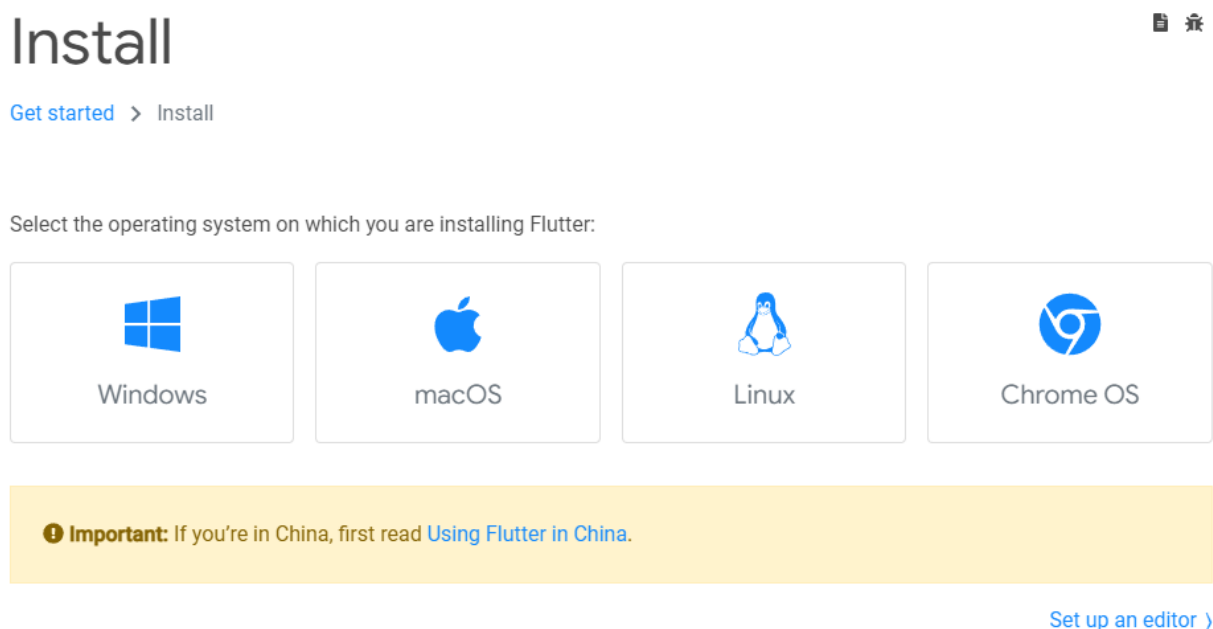


Figure 5: Flutter download operating system selection

Get the Flutter SDK

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

`flutter_windows_2.10.3-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`).

Figure 6: Download the flutter.zip package

6. Select your operating system and then download the .zip file by selecting the "flutter_windows_2.10.3-stable.zip" button. Extract this zip to a read/write enabled directory (this example will extract to C:\tools).
7. Follow flutter.dev's instructions for updating the system path variables to include flutter\bin.

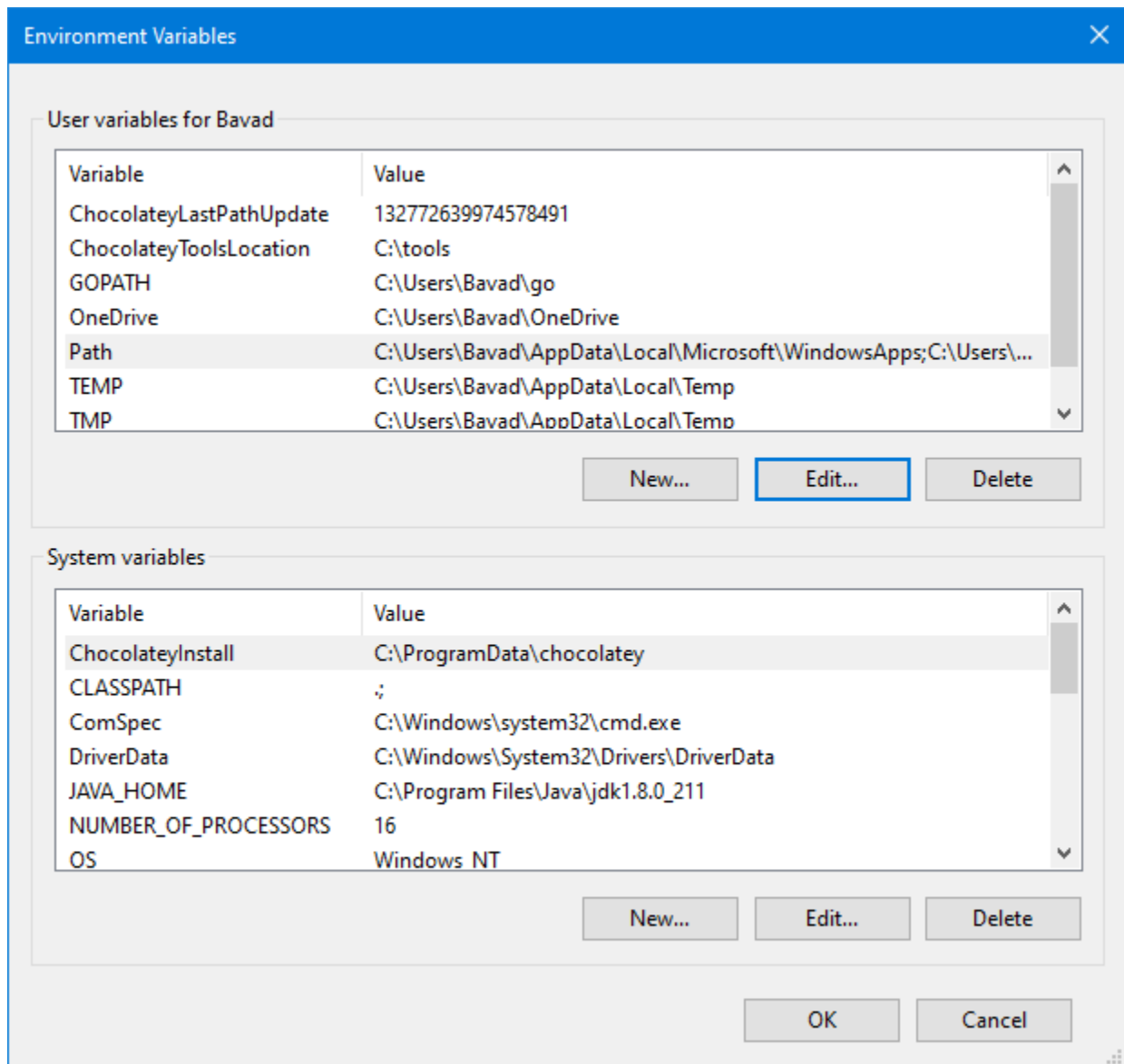
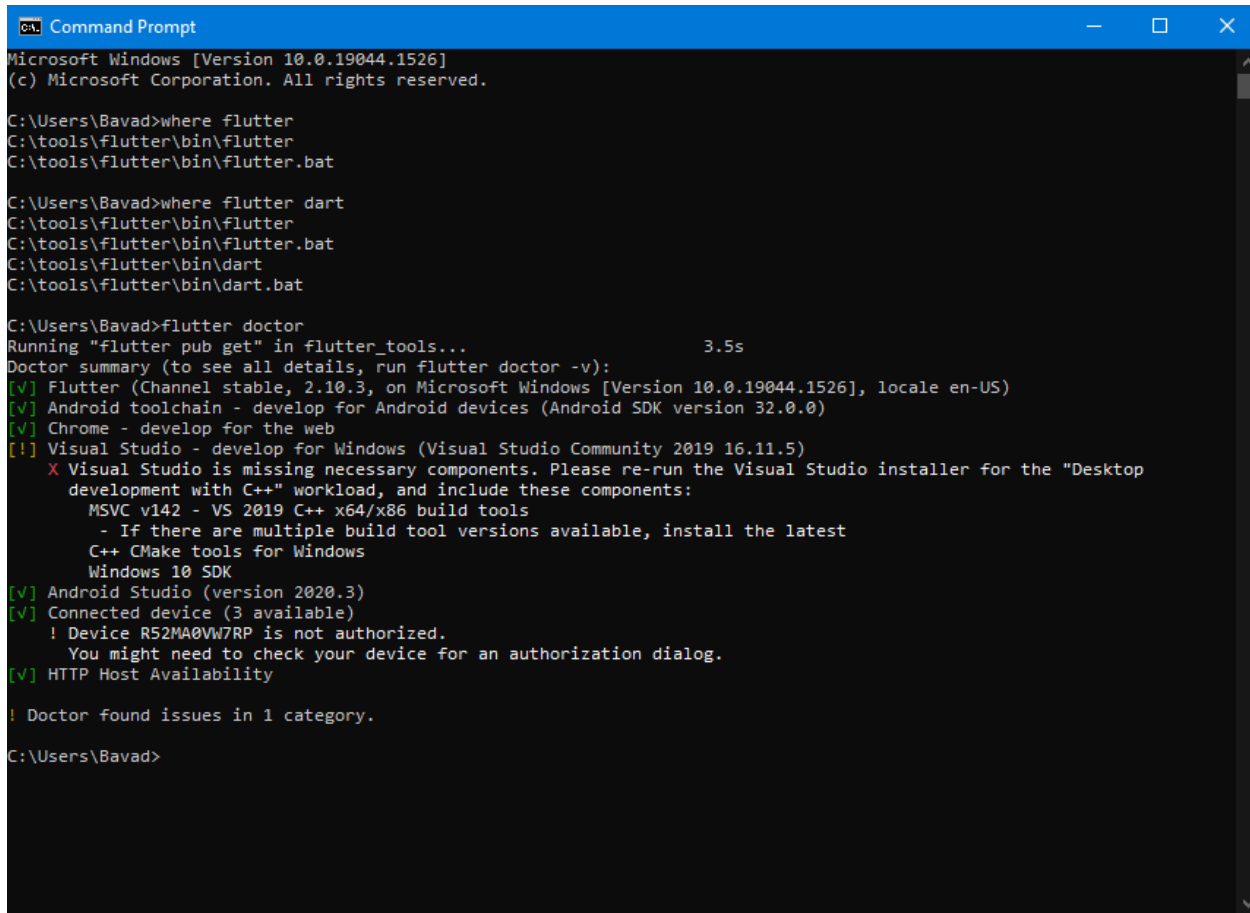


Figure 7: Adding flutter to the environment variable "Path"

-NOTE-
 After adding the path to the flutter\bin you need to **restart**
your computer for the path to be added completely
-NOTE-

8. Check whether your flutter installation was successful by opening a command prompt terminal and typing 'where flutter', 'where flutter dart' and 'flutter doctor'



```

Command Prompt
Microsoft Windows [Version 10.0.19044.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Bavad>where flutter
C:\tools\flutter\bin\flutter
C:\tools\flutter\bin\flutter.bat

C:\Users\Bavad>where flutter dart
C:\tools\flutter\bin\flutter
C:\tools\flutter\bin\flutter.bat
C:\tools\flutter\bin\dart
C:\tools\flutter\bin\dart.bat

C:\Users\Bavad>flutter doctor
Running "flutter pub get" in flutter_tools... 3.5s
Doctor summary (to see all details, run flutter doctor -v):
[V] Flutter (Channel stable, 2.10.3, on Microsoft Windows [Version 10.0.19044.1526], locale en-US)
[V] Android toolchain - develop for Android devices (Android SDK version 32.0.0)
[V] Chrome - develop for the web
[!] Visual Studio - develop for Windows (Visual Studio Community 2019 16.11.5)
    X Visual Studio is missing necessary components. Please re-run the Visual Studio installer for the "Desktop
      development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
        - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[V] Android Studio (version 2020.3)
[V] Connected device (3 available)
    ! Device R52MA0VW7RP is not authorized.
      You might need to check your device for an authorization dialog.
[V] HTTP Host Availability

! Doctor found issues in 1 category.

C:\Users\Bavad>

```

Figure 8: Command line Flutter installation check

After following the official installation instructions, run Flutter Doctor to verify that your installation has been completed correctly and that Flutter has located the Android Studio installation.

- If Windows does not recognize the 'where flutter', or 'where flutter dart' commands
 - Ensure that your path variable is correct and you have restarted the computer after updating it.
- If Flutter did not locate Android Studio
 - run 'Flutter config --android-studio-dir <android studio directory here>' to indicate the location of Android Studio.

After completing the installation of Android Studio / Flutter, and checking the installations in Step 8 we can move on to setting up the Android Studio environment.

3.3 Flutter and Dart Plugins / Android Studio System Settings

Now that we have android studio installed, we need to configure its plugins/System Settings to ensure it can process/markup flutter/dart code.

1. Open Android Studio Arctic Fox and click “Plugins” on the left navigation menu

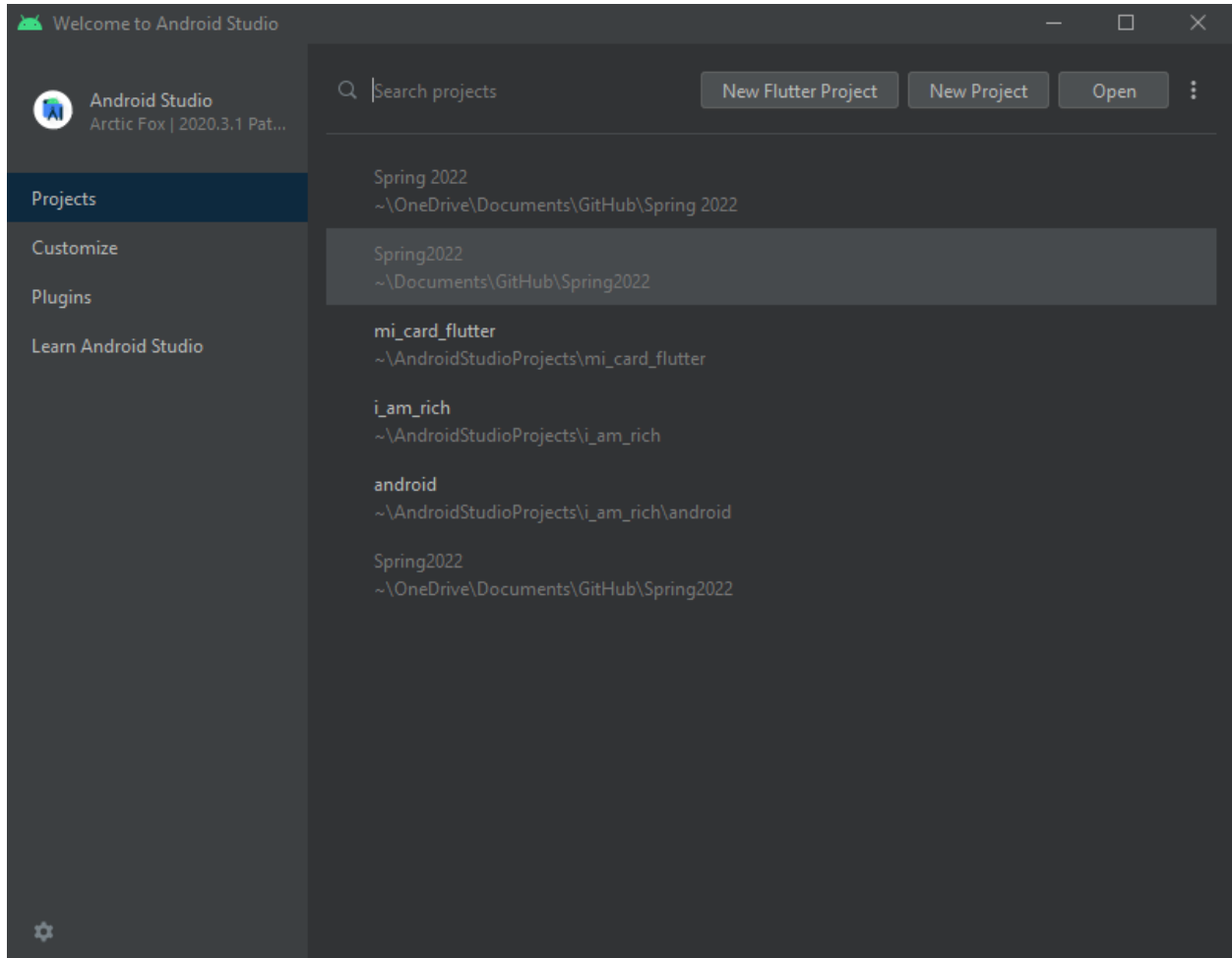


Figure 9: Welcome screen for android studio

2. Ensure that Flutter and Dart plugins are installed as shown below. If not, search for them in the marketplace tab and install them.

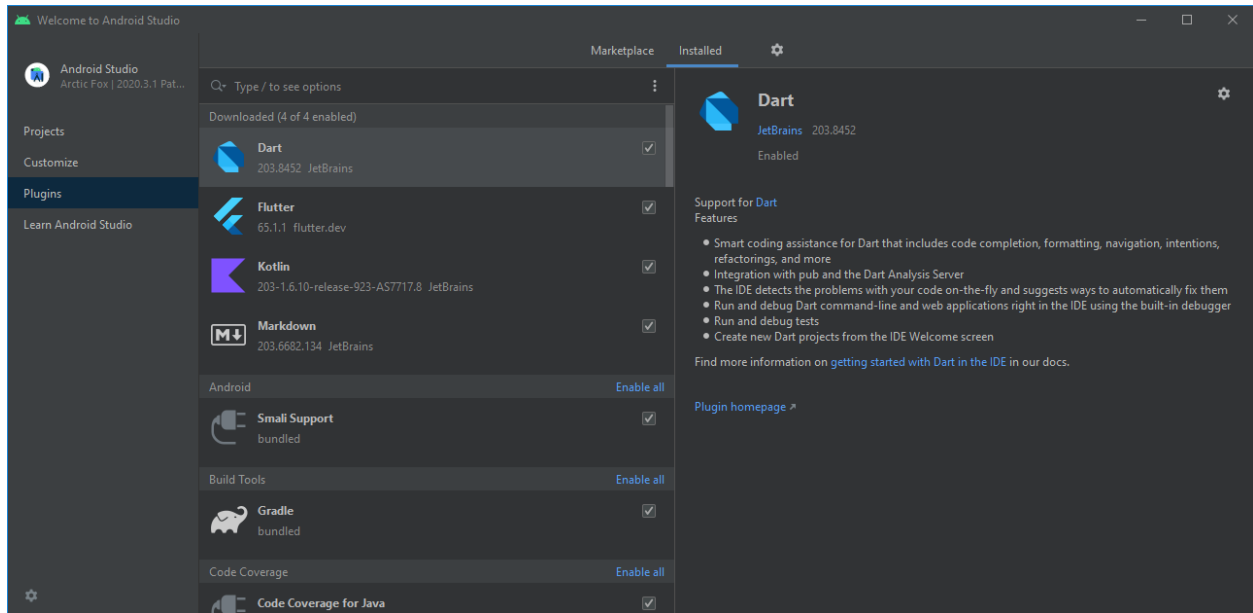


Figure 10: Installed Plugins for Android Studio

- After the plugins are successfully installed click 'Customize' from the navigation menu on the left hand side, and select 'All settings' from the window (circled below)

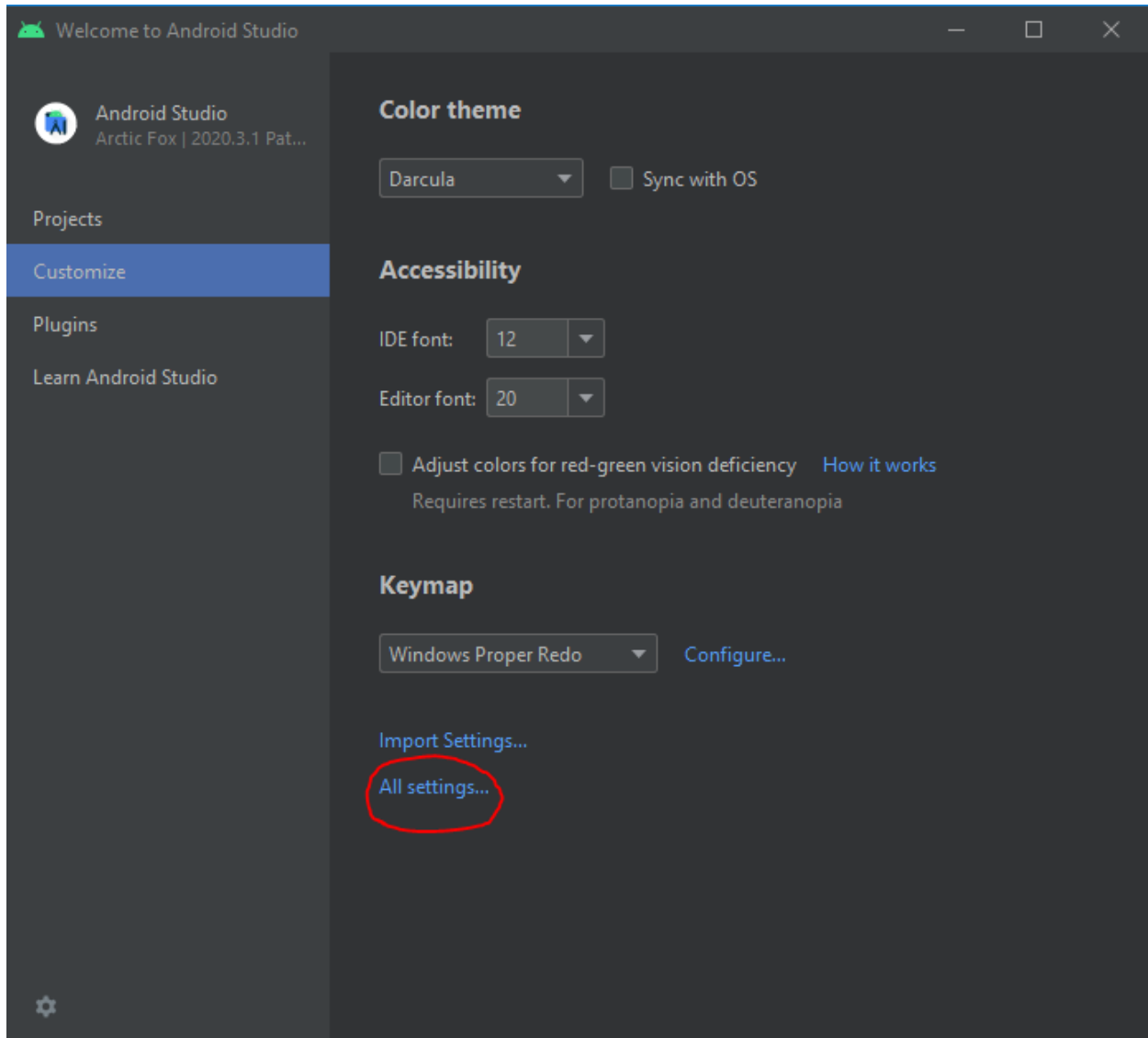


Figure 11: Android Studio All Settings Link

4. Navigate to 'Appearance and Behavior>System Settings>Android SDK' and click the SDK Tools tab. Ensure it looks like below.

-NOTE-
All the SDK tools shown below are required for the IDE to manage
MemorEz correctly. Ensure the path to the Android SDK is populated
and correct
-NOTE-

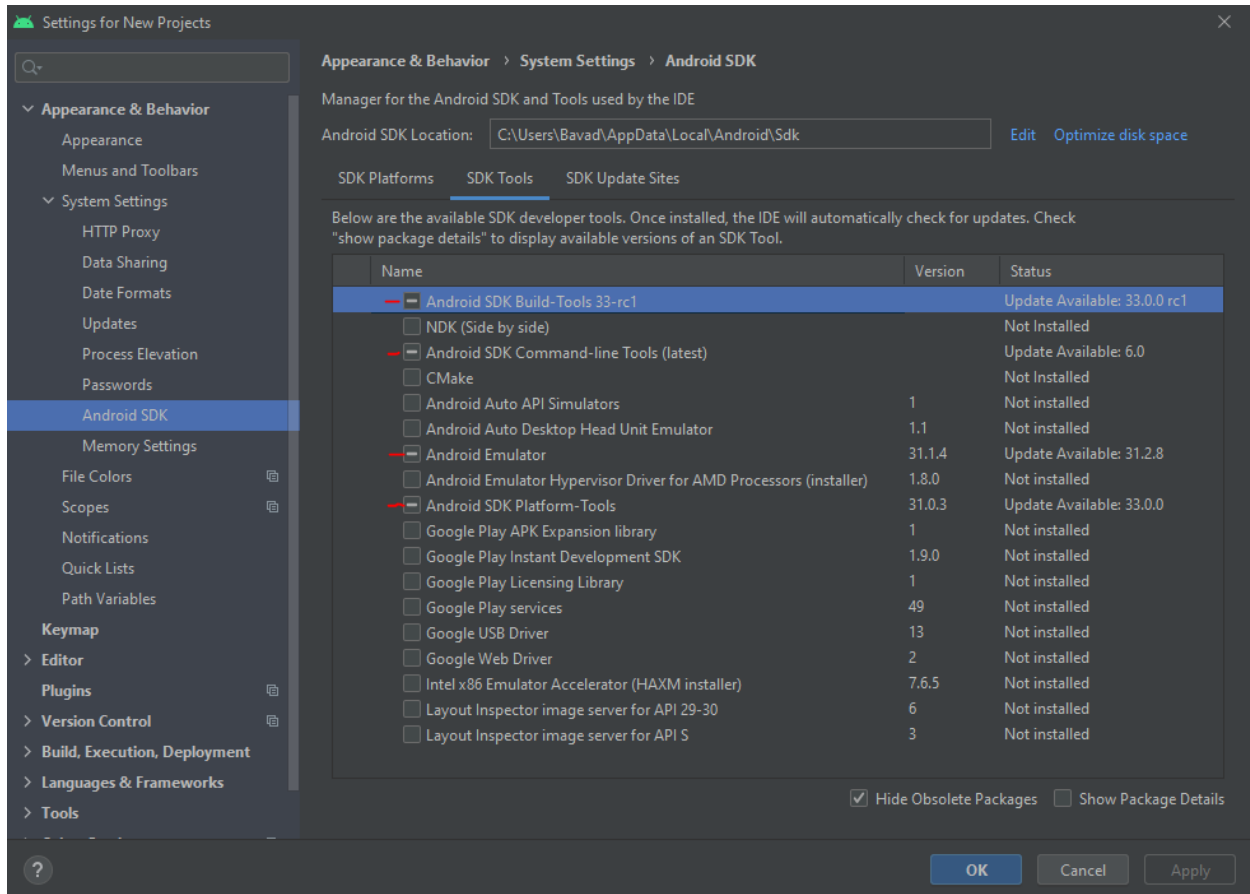
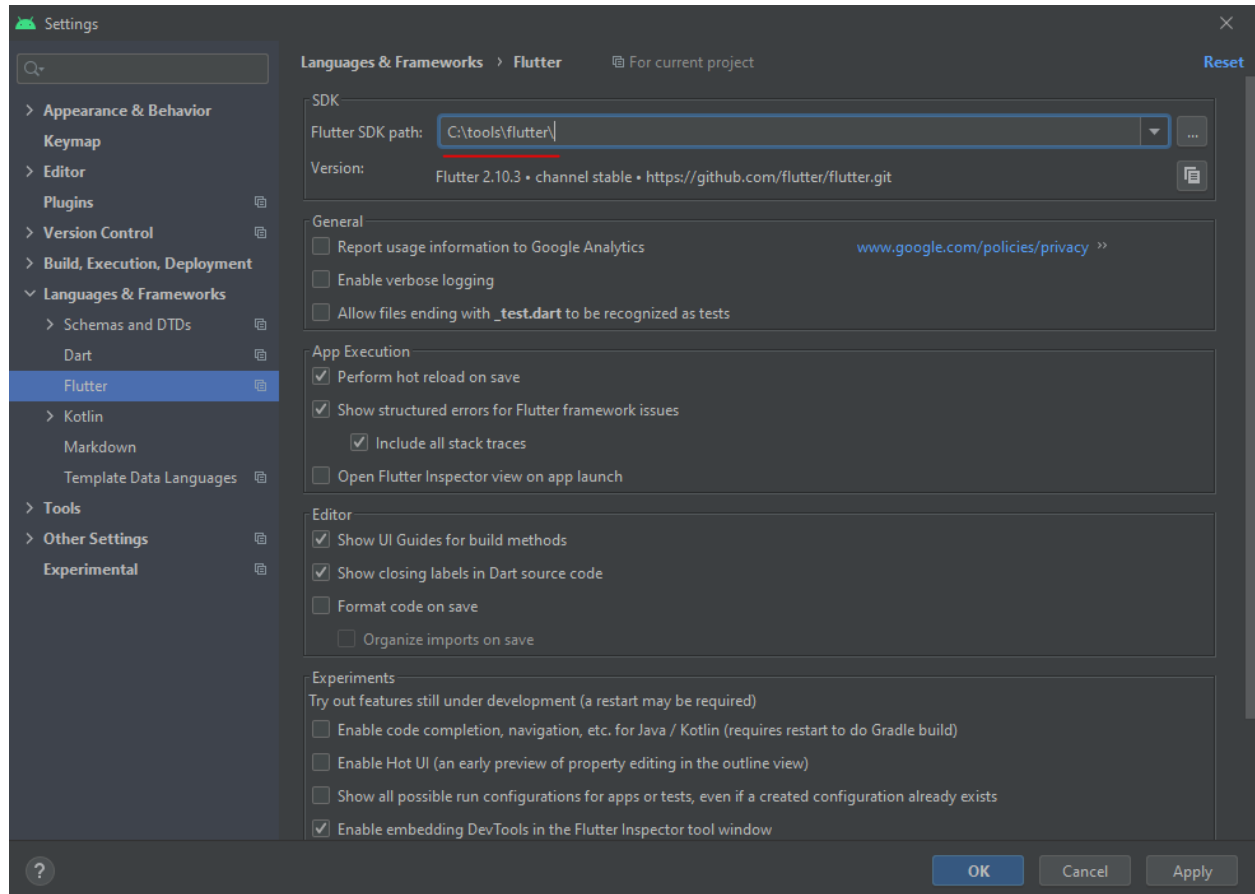


Figure 12: Android SDK System Settings

5. Navigate to 'Language & Frameworks>Flutter and provide the path to the flutter SDK extracted in the previous steps.

MemorEz Runbooks - Combined



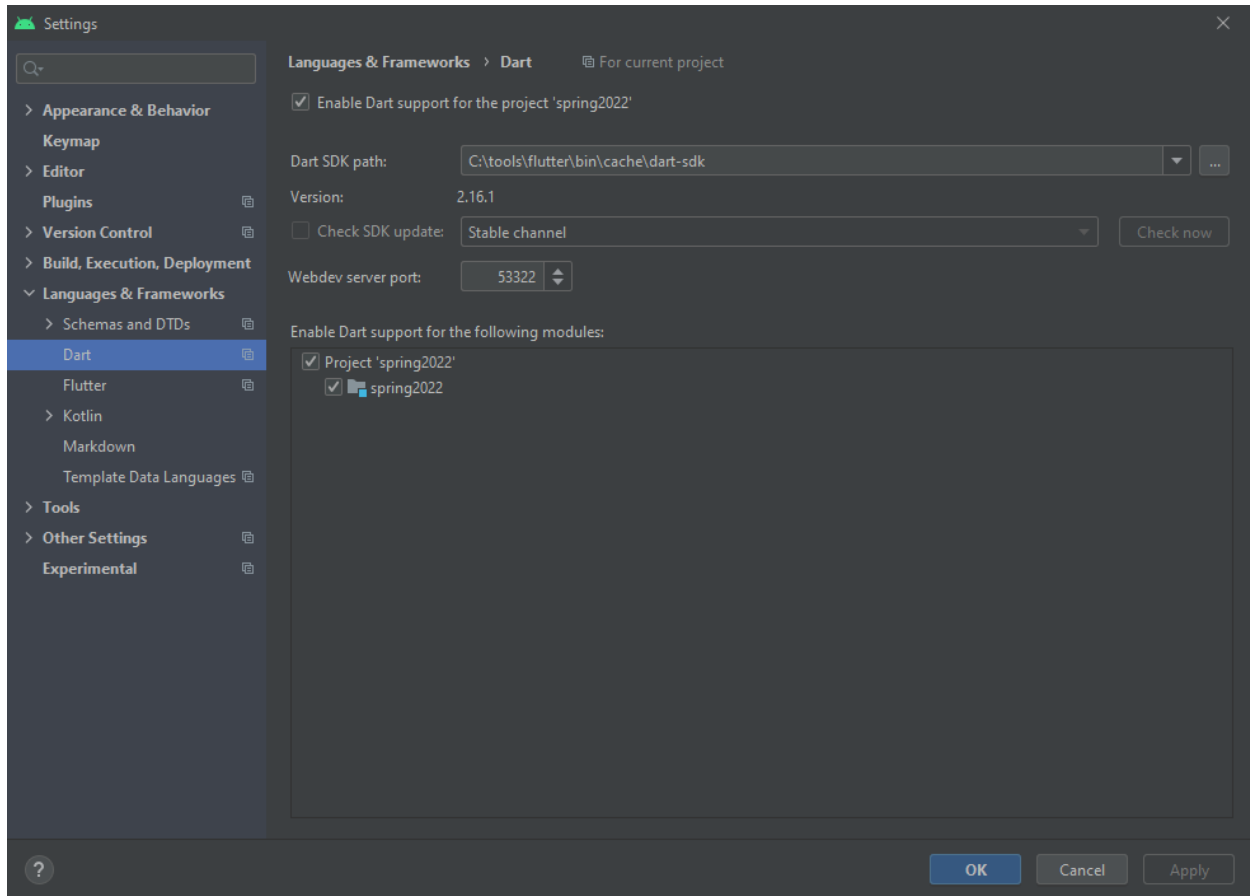


Figure 13: Attaching flutter and Dart Support to Android Studio

Now that the Installation of software/plugins is complete and we've configured the IDE to manage flutter projects, it's time to pull the MemorEz project into the IDE from GitHub.

3.4 New Project from Source Control

In order to open the MemorEz application as it exists on GitHub.com, we will need to use the feature of Android Studio that allows us to open a new project directly from source control. This is a convenient feature that clones the repository from GitHub while simultaneously placing the local files and opening the project in the IDE.

1. Open Android Studio and click on the three vertical dots next to “New Project”. Select “Get from Version Control” from the pop-up menu.

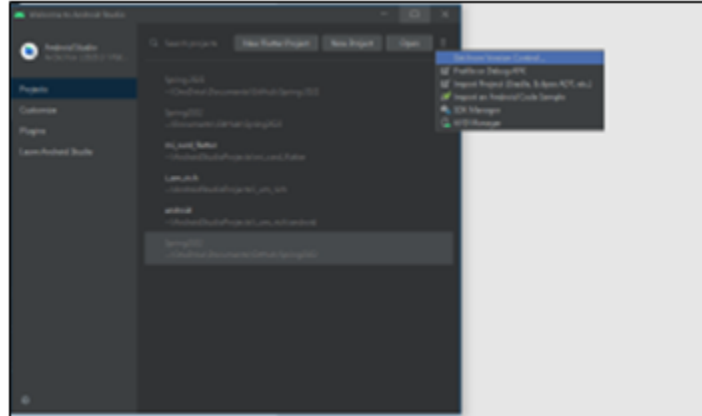


Figure 14: Android Studio New Project from Version Control

2. Enter the URL for the MemorEz repository on GitHub: <https://github.com/umgc/spring2022> and select the directory where the local files of the project will be cloned to and click the 'Clone' button.

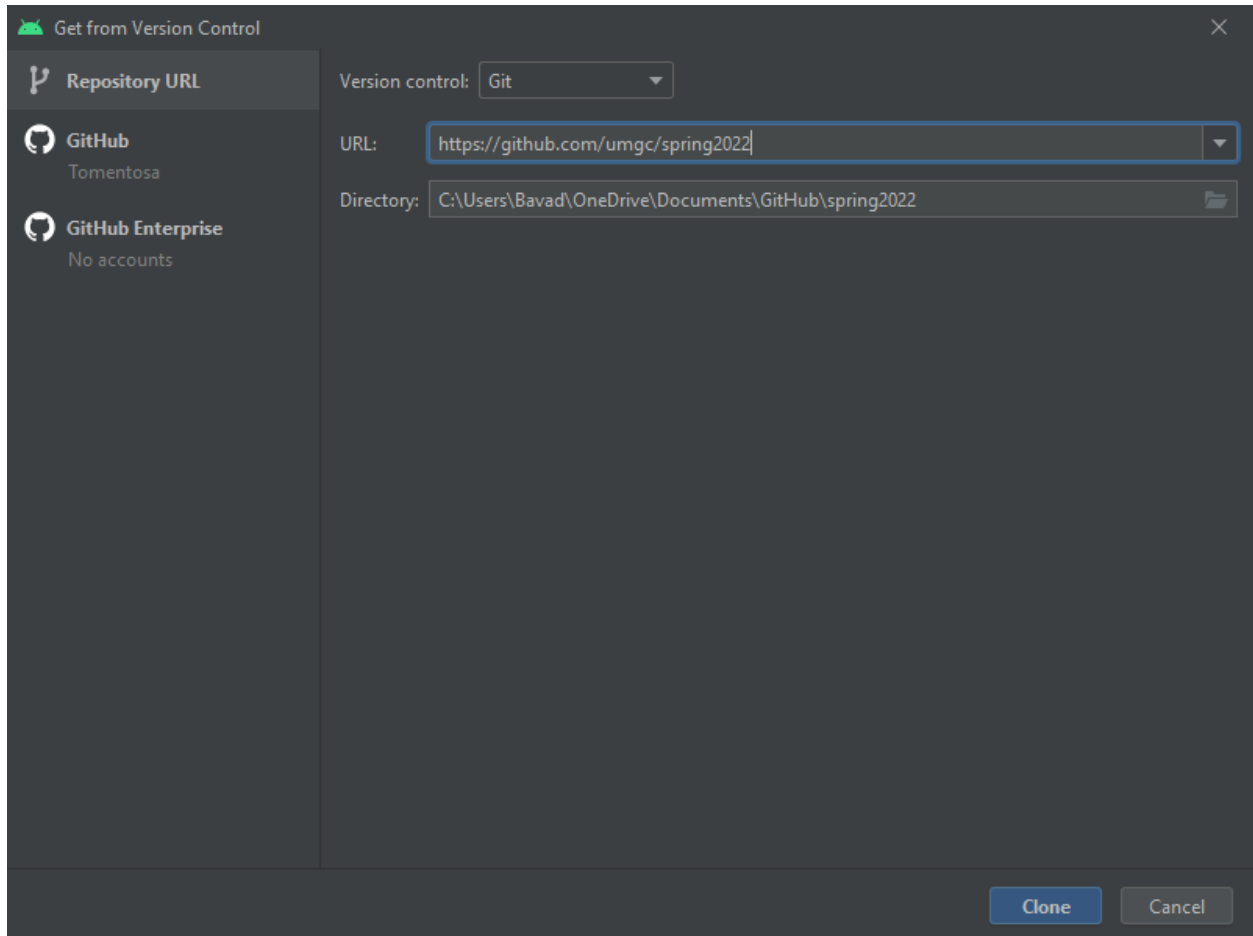


Figure 15: Cloning repository from GitHub URL

At this point, the IDE will clone the repository to whatever local directory you specified, and open the project for when complete. You should see the screen below after the clone is complete.

MemorEz Runbooks - Combined

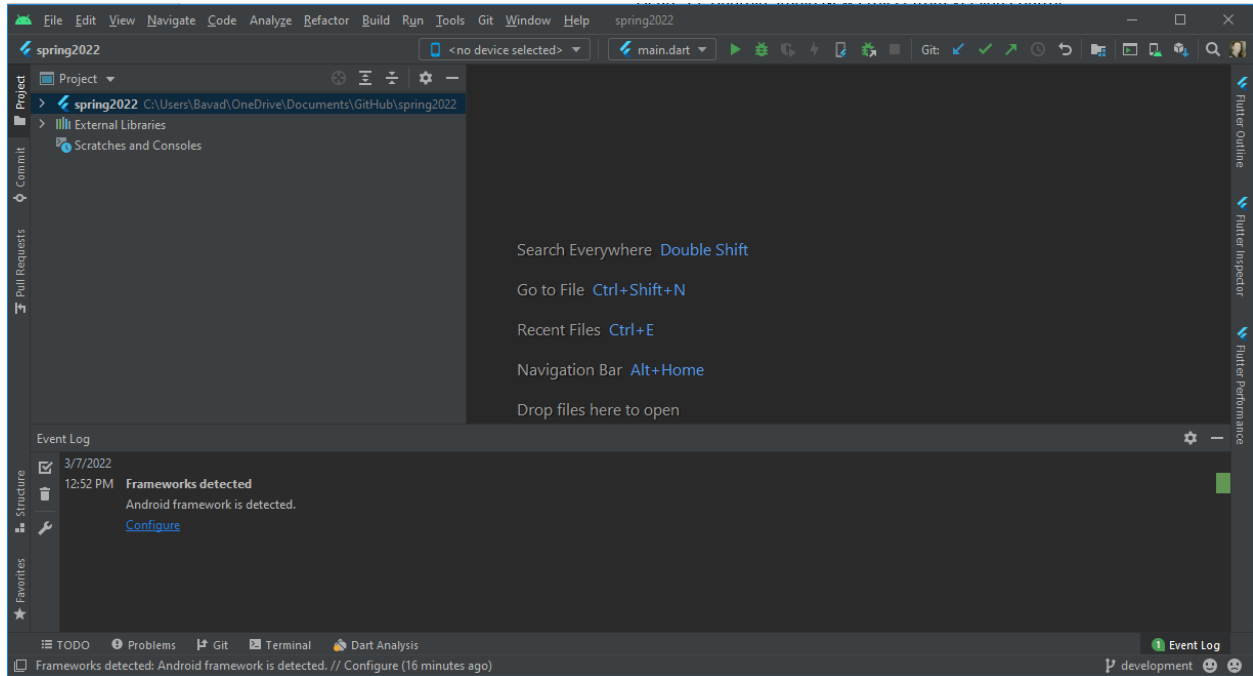


Figure 16: Imported MemorEz from GitHub.com repository

3.5 GitHub Desktop

Since this project will have many students continuously developing and integrating their work into the project, a git version control system will be utilized. All of the functions of Git may be accessed through IDE, but some may find the GitHub Desktop easier to use. If you would like to install this GUI for the git operations, the following steps explain how to download the tool and attach it to your newly established (Step 2 of the previous section) local git repository.

1. Navigate to <https://desktop.github.com/> and download the appropriate installer for your operating system.



Figure 17:GitHub DeskTop Installer

2. Run the downloaded installer and open the GitHub Desktop software. Once the application is open, click on "Add and Existing Repository from your hard drive" and provide the path to the MemorEz repository established in Step 2 of the previous section.

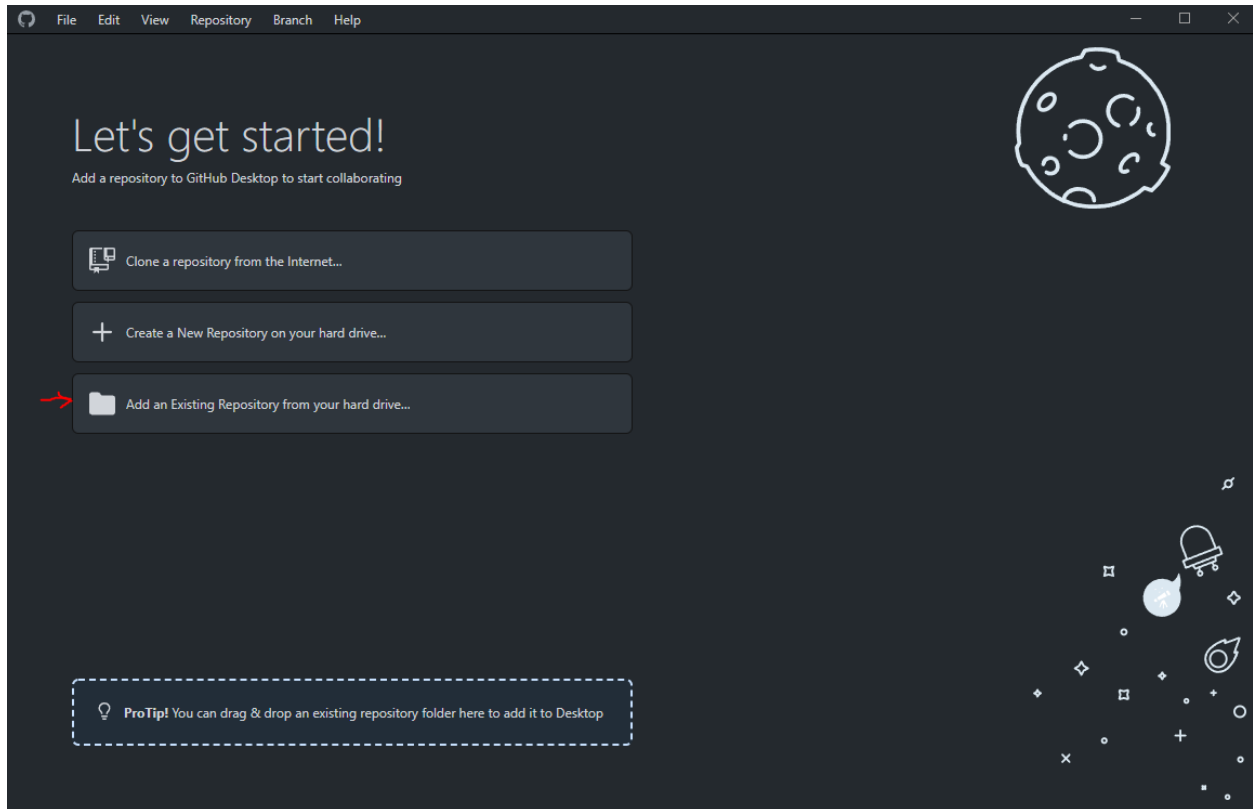


Figure 18: Adding a local repository to the GitHub Desktop Application

At this point we have a copy of the MemorEz application pulled from its source control, and our environment is configured to process and manage flutter/dart code. Next, we will configure our emulators and physical devices we will use to run MemorEz

3.6 Configure Physical Android Devices for Running MemorEz

Android devices make it very easy to interface with Android Studio. By following the steps outlined here: <https://developer.android.com/studio/debug/dev-options>

A MemorEz developer can attach their physical devices and utilize them to run/test the application. Once these steps are followed, after the android is plugged in via USB it becomes available next to the launch button in the IDE

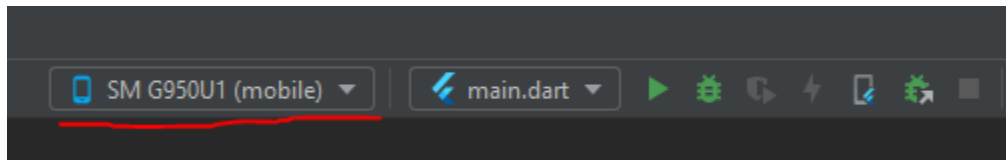


Figure 19: Physical Device Available for application launch

3.7 Configure Emulators for Running MemorEz

Information in this section is taken from the Android Developer website ("Run apps on the Android Emulator," n.d.) and from RememberAll/UMGC 2022 Spring Cohorts Programmers guide.

3.8 Setting up Android Emulator

To install the Android Emulator in Android Studio, select the Android Emulator component in the SDK Tools tab of the SDK Manager. For instructions on updating tools with the SDK Manager see: <https://developer.android.com/studio/intro/update#sdk-manager>

3.9 Setting Up Simulator macOS

Download and Install Xcode: To download and install Xcode visit the Mac App Store: <https://apps.apple.com/us/app/xcode/id497799835?mt=12>

Setting Up and iOS Simulator: For instructions about how to set up an iOS simulator on macOS see this tutorial: <https://www.macinstruct.com/tutorials/setting-up-an-ios-simulator-on-your-mac/>

4 Prepare the Mobile Application for Use

Now that we have all of the development software installed / configured, we have cloned the MemorEz repository, and opened the project in the IDE, there a few things we need to do within the flutter code for it to run. Flutter is built on packages/widgets, and as developers work on the code, these dependencies change. So two major steps to running the application are updating these dependencies. MemorEz also has private credentials (not uploaded to GitHub) that need to be written in to the codebase for it to conduct any Natural Language Processing functions.

4.1 Fetching Updated Code Dependencies

Since this is a project focused around git as the version control system, it is IMPERITIVE that we perform a git-fetch / update to ensure we are working with the latest code.

1. Click Fetch Origin to make the latest GitHub.com files available to your local repository clone.

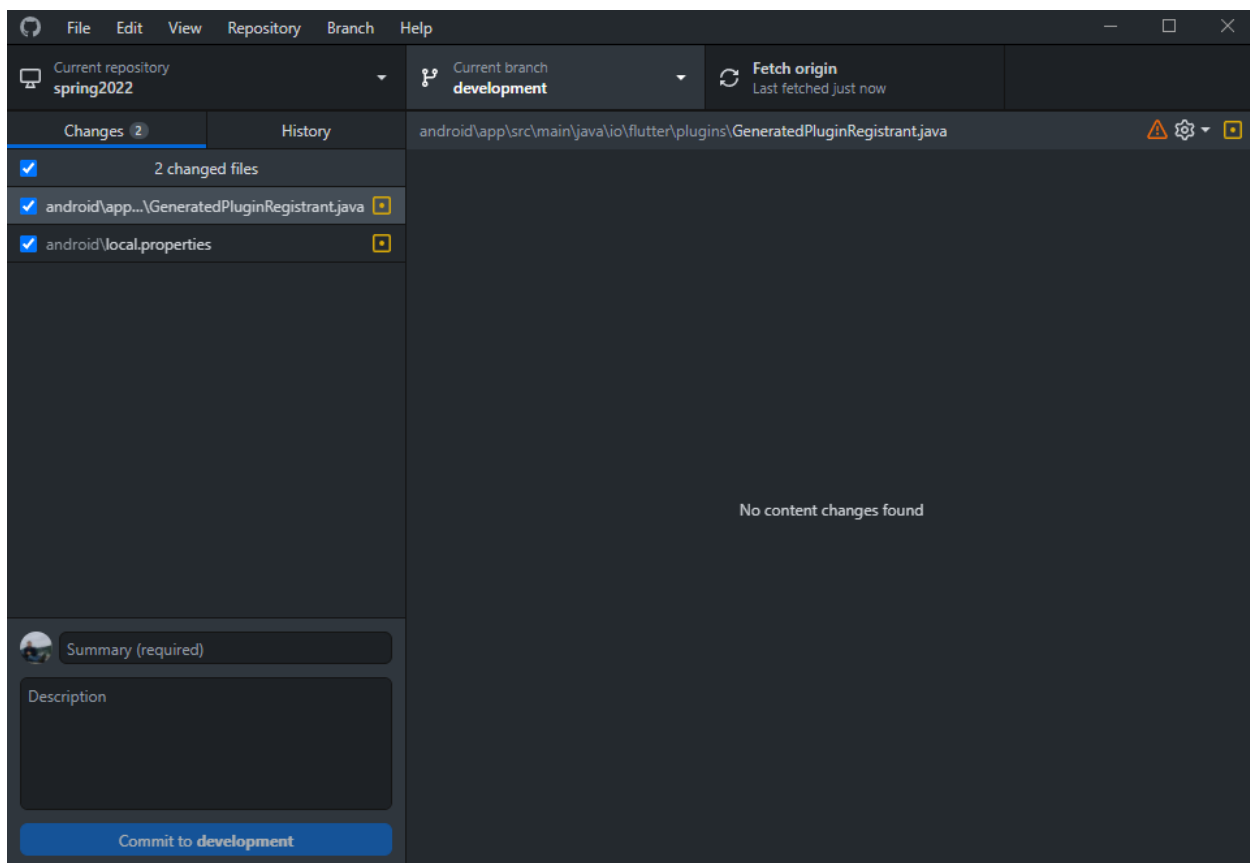


Figure 20: Git Fetch of GitHub.com Repository

Now that we know we have the latest code from the repository, we must update the dependencies from within the MemorEz code base.

2. Open Android Studio and open the MemorEz project.

3. In the left hand window, navigate to “Spring2022>pubspec.yaml” and click the Pub get button circled below. This will update the project files with the latest declared dependencies.

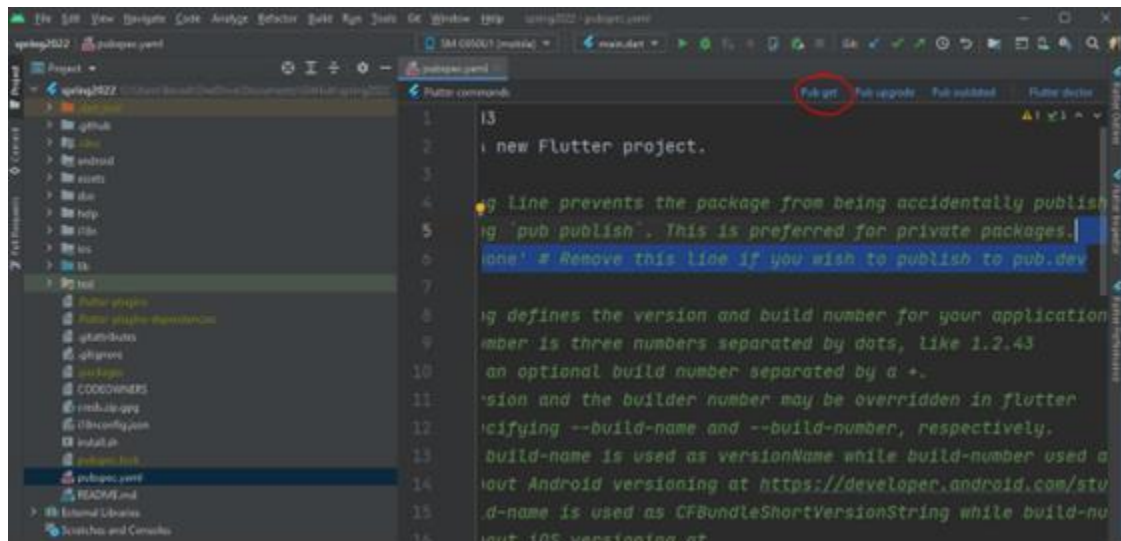


Figure 21: Performing a Pub get operation

4.2 Attaching the LEX Credentials (Enables Language Processing)

Since the LEX credentials are not uploaded to source control, the developer will need to manually add them in the first time they go to run the application.

1. Navigate to “spring2022>assets>NLU” and right click the NLU directory

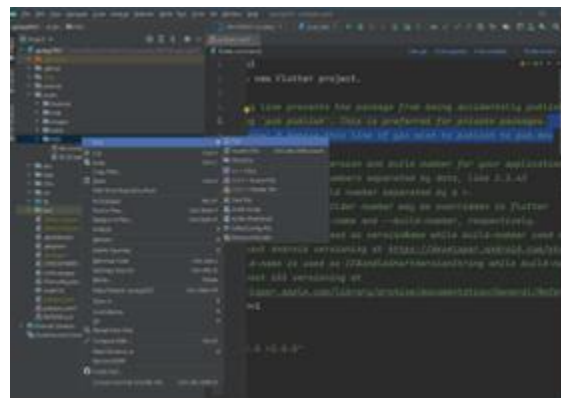


Figure 22: Adding the LEX_Credentials.xml file to the project

1. Create a new file in this directory with the EXACT FILENAME: “LEX_Credentials.xml”
2. Contact the code owners for the Repository to get the contents of the LEX_Credentials.xml document

4.3 Run the MemorEz Application

At this point, the developer has fetched updated code, ensured that the dependencies have been pulled using Pub get, and the LEX_Credentials.xml file has been manually created and populated. From here, the developer needs to select a platform to run the application (in this example we will use the physical device we set up in the last section) and click Run within the IDE which is the green triangle next to the main.dart selection. main.dart is the first file any dart program will run when launching.

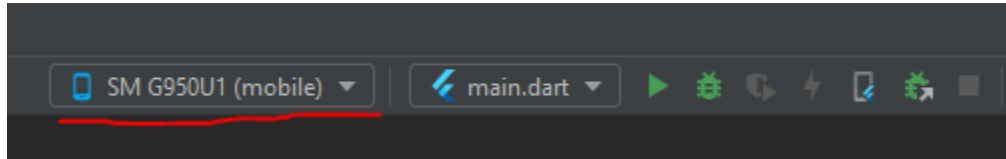


Figure 23:Run bar in the top right hand corner of Android Studio

After 3-5 minutes of loading, the application will open and be available to interact with on the device or emulator.

5 Testing the Mobile Application

This section examines the testing process for the Memory Magic Application. Testing is necessary to ensure that all the required deliverables have been successfully implemented and to provide quality control and assurance.

Several full-system, end-to-end (E2E) smoke tests shall be created for common use cases to ensure intended functionality and to serve as a safeguard in the case of regression of a feature that has already been integrated into the system. Static analysis shall be completed on the application to ensure issues may be rectified before ultimate delivery of the application.

Testing will include features available on Github.com, and/or any necessary third-party tools; this will ensure the code is free of any major errors/bad syntax that may lead to exploits.

5.1 Testing Objectives

5.1.1 Unit Tests

A unit test assesses a single function/method/class. The goal is to verify the accuracy of a logical unit. The Table 6 series shows the Test Matrix for the General Tests.

Table 6:Unit Tests

Test Case #1		Android	iPhone
Feature Being Tested	Change User Mode to Caregiver Mode		
Test Process	1. Open the app and switch to caregiver mode		
Expected Result	The state of the app should switch to caregiver mode		

Test Case #2		Android	iPhone
Feature Being Tested	View MemorEz App Settings		
Test Process	1. Put the app in caregiver mode 2. Move to settings section		
Expected Result	The caregiver should be able to view settings		

Test Case #3		Android	iPhone
Feature Being Tested	Edit MemorEz app setting		
Test Process	<ol style="list-style-type: none"> 1. Place the application in caregiver mode 2. Navigate to the settings section 3. Change a setting 4. Exit the application 5. Open the application 6. Navigate to settings 		
Expected Result	The caregiver should see the changed setting		

Test Case #4		Android	iPhone
Feature Being Tested	View STML User Profile		
Test Process	<ol style="list-style-type: none"> 1. Place the application in caregiver mode 2. Navigate to the profile section 		
Expected Result	The user should be able to view the STML profile		

Test Case #5		Android	iPhone
Feature Being Tested	Edit the STML user profile		
Test Process	<ol style="list-style-type: none"> 1. Place the application in caregiver mode 2. Navigate to the profile section 3. Edit the profile section 4. Close the app 5. Re-open the app in the app and navigate to the profile section 		
Expected Result	The user should be able to see their changes to the profile		

Test Case #6		Android	iPhone
Feature Being Tested	View STML user calendar		
Test Process	<ol style="list-style-type: none"> 1. Place the application in caregiver mode 2. Navigate to the calendar section 		
Expected Result	The user should be able to view the calendar		

Test Case #7		Android	iPhone
Feature Being Tested	Edit STML User Calendar		
Test Process	<ol style="list-style-type: none"> 1. Place the application in caregiver mode 2. Navigate to the calendar section 3. Make an edit in the calendar 4. Close the app 		

MemorEz Runbooks - Combined

	5. Re-open the app and navigate to the calendar section		
Expected Result	The user should be able to see their edit in the calendar		

Test Case #8		Android	iPhone
Feature Being Tested	View STML user reminders		
Test Process	<ol style="list-style-type: none"> 1. Place the application in caregiver mode 2. Navigate to the reminder section 		
Expected Result	The user should be able to view automatic reminders for the STML patient		

Test Case #9		Android	iPhone
Feature Being Tested	Add/Edit STML user reminder		
Test Process	<ol style="list-style-type: none"> 1. Place the application in caregiver mode 2. Navigate to the reminders section 3. Add a reminder and edit a reminder 4. Close the app 5. Re-open the app and navigate to caregiver mode 6. Navigate to the reminders section 		
Expected Result	The user should be able to see the reminder addition/edit		

Test Case #10		Android	iPhone
Feature Being Tested	View STML User Resource		
Test Process	<ol style="list-style-type: none"> 1. Place the application in caregiver mode 2. Navigate to the resources 		
Expected Result	The user should be able to see the STML user resources		

Test Case #11		Android	iPhone
Feature Being Tested	Edit STML user resources		
Test Process	<ol style="list-style-type: none"> 1. Place the application in caregiver mode 2. Navigate to the resources section 3. Edit a resource 4. Close the app 5. Re-open the app and navigate to caregiver mode 6. Navigate to the resources section 		
Expected Result	The user should be able to see the resource edit		

Test Case #12		Android	iPhone
Feature Being Tested	View Transportation Resources		
Test Process	<ol style="list-style-type: none"> 1. Place the application in caregiver mode 		

MemorEz Runbooks - Combined

	2. Navigate to the resources section		
Expected Result	The user should be able to see the transportation resources		

Test Case #13		Android	iPhone
Feature Being Tested	Edit Transportation Resources		
Test Process	<ol style="list-style-type: none"> 1. Place the application in caregiver mode 2. Navigate to the resources section 3. Edit a transportation resource 4. Close the app 5. Re-open the app and navigate to caregiver mode 6. Navigate to the resources section 		
Expected Result	The user should be able to see the transportation resource edit		

5.1.2 Integration Tests

An integration test assesses the application as a whole or a large part of it. The goal is to verify that all features tested can work together as expected. Table 7 shows the Test Matrix for Integration Tests.

Table 7: Integration Tests

Integration Tests	iPhone	Android
**Integration with NLU (Previous Semester):		
1. Application handles successful response from NLU API. Success case: Obtain desired response.		
2. Application handles all non-success responses from NLU API. Success case: Error is handled gracefully		

5.2 Testing Procedures

- Choose Windows (version 10 or later) operating system (64-bit with a x64 based processor laptop) or iOS operating system.
- Install Microsoft Teams – to be used in collaboration with TTT team, DevOps and team Mesmerize members.
- Install Flutter version 2.5.
- Install Android Studio version 4.3 with Flutter and Dart plugins.
- Install GitHub - to be used in collaboration with TTT team, DevOps and team Mesmerize Members.
 - Sign Up for Trello Web based project management tool.
- Clone the GitHub repository <https://github.com/umgc/fall2021>
- Execute the application
- Manually test each feature of the application
- Document the test results

MemorEz Runbooks - Combined

- If any of the required tests fail, fix the issues in collaboration with TTT team, DevOps and Mesmerize team members.
- Rerun the updated failed tests
- Document test results
- Documentation - test report

6 Troubleshooting

After installing and starting using the development environment you can still find some issues when using Flutter and Android Studio. This section will cover some of the most common issues and troubleshooting solutions to problems. Some of the troubleshooting steps provided in this section may be different depending on the OS used. For more specific solutions to problems, visit the official Flutter website at <https://flutter.dev/docs> and/or Android Studios

at <https://developer.android.com/studio/troubleshoot>. (Presley Muwan, Karen Crumb, Kevin Bell, Teresa Balbi, Sami Salim, Daniel Avery & Christian Cruz Jimenez, 2021)

6.1 Emulator Not Responding

** To perform these steps your emulator should not be running.*

1. If you have a Project already open on Android Studio, click on *Tools > ADV Manager*.

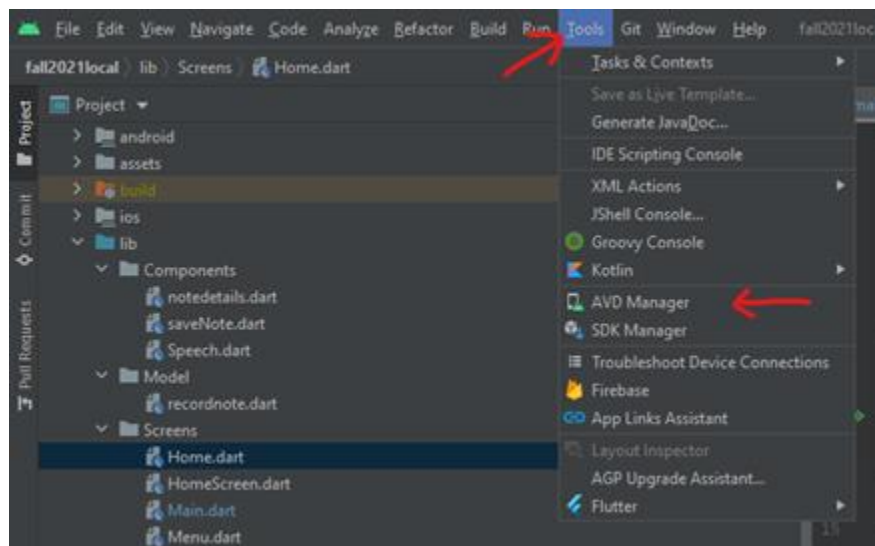


Figure 24: Tools

2. On the Welcome page of Android Studio, click on the three dots at the top right corner > ADV Manager.

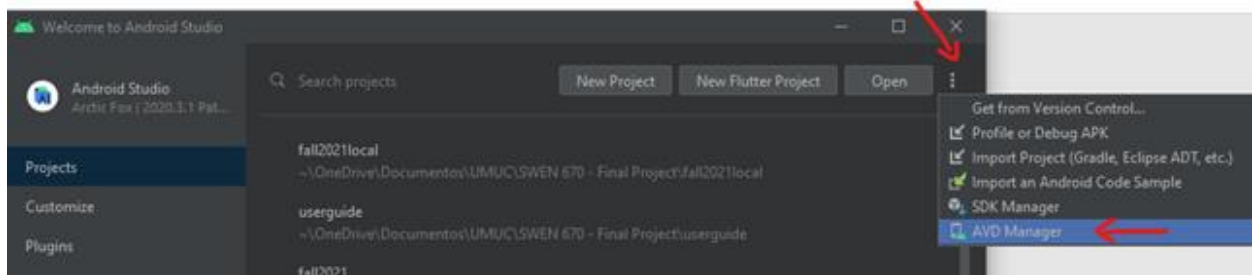


Figure 25:ADV Manager

3. After opening the ADV Manager, find the emulator you want to use and click on the *dropdown arrow* > *Wipe Data*.



Figure 26:Your Virtual Devices

3. After wiping your device data, you can perform a 'Cold Boot' (at the same location you just wiped the data from) to restart the emulator device. Click on the *dropdown arrow* > *Cold Boot Now*.

6.2 Out of Memory Error

This is a common error if the local machine used to run the emulator does not have sufficient resources to handle the virtual device. A common resolution is to close all unnecessary applications running in the background to free up virtual memory. In addition, it may be helpful to increase the assigned memory to your virtual emulator by following these steps: (Presley Muwan, Karen Crumb, Kevin Bell, Teresa Balbi, Sami Salim, Daniel Avery & Christian Cruz Jimenez, 2021)

1. On Android Studio click on *File* > *Settings* to open the general settings.

2. On the general settings screen click on *Appearance & Behavior > System Setting > Memory Settings*.

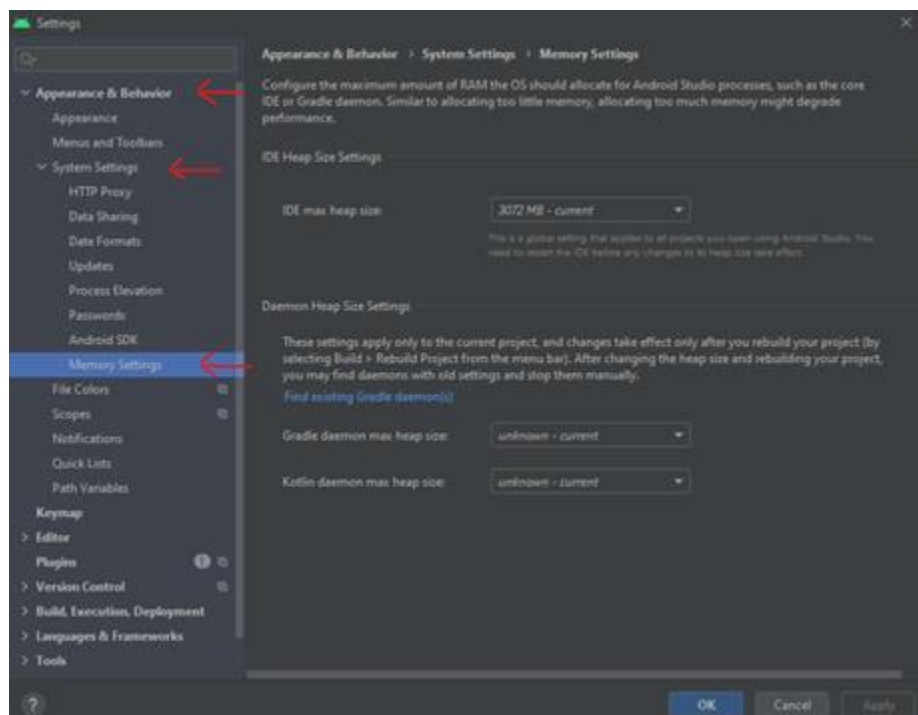


Figure 27: Your Virtual Devices

3. Customize the amount of memory you want to assign to your environment. The amount you assign should be based on the resources of your computer but is recommended to have at least 2,048 MB assigned.
4. After assigning your memory you can click *Apply > Ok* to apply and save the changes.
5. Restart your Android Studio.
6. More information can be found at the official web page of Android Studio at <https://developer.android.com/topic/performance/memory>.

6.3 Stuck at Running Gradle Task 'AssembleDebug'...

This is a common issue when trying to start the app in the emulator. If you receive the message "Running Gradle task 'assembleDebug'... (this is taking an unexpectedly long time.)," you can consider the following troubleshooting steps: (Presley Muwan, Karen Crumb, Kevin Bell, Teresa Balbi, Sami Salim, Daniel Avery & Christian Cruz Jimenez, 2021)

1. Open your terminal app/console.
2. Navigate to the Flutter project directory.

3. On your Flutter project directory run this command `./gradlew clean`.
4. You can build the Gradle from the directory or let Android Studio build it when you run the app.

*To build Gradle from the directory run `./gradlew build`.

*You can also combine both commands to clean and build the Gradle by running `./gradlew clean build`. (koderstory, 2020).

6.4 Dependency Errors

Another common issue is to get unexpected dependency errors on your code. This generally occurs when multiple versions of a library get imported to the project. A common solution to this problem is to clear the Flutter built files and reimport the libraries. To do so follow the below steps: (Presley Muwan, Karen Crumb, Kevin Bell, Teresa Balbi, Sami Salim, Daniel Avery & Christian Cruz Jimenez, 2021)

1. Open your terminal app/console.
2. Navigate to the root directory of your Flutter project.
3. Run the command `flutter clean` to remove all the dependencies.
4. Now re-import the newest libraries to the correct dependencies by running `flutter pub get`.