Presented by Team Mesmerize

Presley Muwan, Karen Crumb, Kevin Bell, Teresa Balbi, Sami Salim, Daniel Avery & Christian Cruz Jimenez  
  
UMGC SWEN 670

Deployment and Operations Guide (Runbook)

MEMORY MAGIC APP

Project name: Memory Magic Application.

Date: October 19, 2021

Project Manager: Presley Muwan

Phase: Design & Implementation

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Date | Description | Approved By |
| 1.0 | 10/19/2021 | Initial Release | Presley Muwan |

Table of Contents

[Introduction 5](#_Toc85470946)

[1.1 Purpose 5](#_Toc85470947)

[1.2 Intended Audience and Reading Suggestions 5](#_Toc85470948)

[1.3 Technical Project Stakeholders 5](#_Toc85470949)

[1.4 References 6](#_Toc85470950)

[1.5 Definitions, Acronyms, and Abbreviations 7](#_Toc85470951)

[Mobile Application 8](#_Toc85470952)

[2.1 Features, Packages, Plugins, and Widgets 8](#_Toc85470953)

[2.1.1      Features 8](#_Toc85470954)

[2.1.2      Packages 9](#_Toc85470955)

[2.1.3      Plugins 10](#_Toc85470956)

[2.1.4      Widgets 11](#_Toc85470957)

[Software Installation 13](#_Toc85470958)

[3.1 Android Studio 13](#_Toc85470959)

[3.2 Dart and Flutter 14](#_Toc85470960)

[3.3 Android Emulator 15](#_Toc85470961)

[3.4 Flutter and Dart plugins 16](#_Toc85470962)

[3.5 Test your development environment 17](#_Toc85470963)

[3.6 GitHub Desktop 17](#_Toc85470964)

[Prepare the Mobile Application for Use 18](#_Toc85470965)

[4.1 Cloning GitHub repository 18](#_Toc85470966)

[4.2 Run the Flutter application 19](#_Toc85470967)

[Testing the Mobile Application 20](#_Toc85470968)

[5.1 Testing Objectives 21](#_Toc85470969)

[5.1.1 Unit Tests 21](#_Toc85470970)

[5.1.2 Integration Tests 26](#_Toc85470971)

[5.2 Testing Procedures 27](#_Toc85470972)

[Troubleshooting 28](#_Toc85470973)

[6.1 Emulator not responding 28](#_Toc85470974)

[6.2 Out of Memory Error 30](#_Toc85470975)

[6.3 Stuck at Running Gradle task 'assembleDebug'... 31](#_Toc85470976)

[6.4 Dependency Errors 32](#_Toc85470977)

[Appendices 32](#_Toc85470978)

[7.1 Credits 32](#_Toc85470979)

# Introduction

## 1.1 Purpose

The purpose of this Deployment and Operations Guide (Runbook) is to outline the steps and illustrate the details necessary to install and deploy the Memory Magic App.

## 1.2 Intended Audience and Reading Suggestions

The intended audience for this guide is software developers, students, and technical stakeholders interested in installing/deploying the Memory Magic App. The purpose of this document is to show each step, in detail, which any user can effortlessly follow.

Reading suggestions to assist the intended audience understand the systems used: *Flutter for Beginners*, by Alessandro Biessek and *Programming Flutter* by Carmine Zaccagnino.

## 1.3 Technical Project Stakeholders

Table 1 shows the project stakeholders for the Memory Magic App:

**Table 1**

*Project Stakeholders*

|  |  |
| --- | --- |
| **Name** | **Role** |
| Dr. Mir Assadullah | Stakeholder (Project Owner) |
| Roy Gordon | Stakeholder (Project Advisor) |
| Presley Muwan | Project Manager |
| Teresa Balbi | Business Analyst |
| Karen Crumb | Lead Developer |
| Daniel Avery | Developer |
| Sami Salim | Developer |
| Christian Cruz Jimenez | Tester |
| Kevin Bell | UI/UX Designer |

## 1.4 References

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

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 |

# Mobile Application

## 2.1 Features, Packages, Plugins, and Widgets

### 2.1.1      Features

* Simplicity – Easy to navigate and keep user engaged. Clear, uncluttered screens and no ambiguity, which leads to an easier and improved user experience.
* Speed – Fast loading screens that keep users engaged.
* Flexibility – Compatible with Android and iOS operating systems. The application is able to perform the necessary functions across different platforms.
* Security – Provide users the necessary security to deal with storing personal and sensitive information.
* Search Options –Provide users the ability to search for user specified information throughout the application.
* Notifications – Provide users relevant and personalized notifications.

### 2.1.2      Packages

Packages can be defined as “shared packages contributed by other developers to the 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 into pub.dev; this webpage contains packages and dependencies that are compatible with Flutter.

2.1.2.1 Installing a package dependency into an app

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.

Table 3 shows the core packages used in the Memory Magic App.

**Table 3**

*Packages and Descriptions*

|  |  |
| --- | --- |
| Packages | Description |
| flutter\_localizations | Package dependency used to set up a localization. |
| speech\_to\_text | Package that exposes device specific speech to text recognition capability. |
| encrypt | A set of high-level APIs over PointyCastle for two-way cryptography. |
| timeago | A library useful for creating fuzzy timestamps. |
| flutter\_tts | Package for Text to Speech. |
| intl | Package that contains code to deal with internationalized/localized messages, date and number formatting and parsing, bi-directional text, and other internationalization issues. |
| flutter\_mobx | Package that provides a set of Observer widgets that automatically rebuild when the tracked observables change. |
| http | A composable, multi-platform, Future-based API for HTTP requests. |

*Note.*See https://github.com/umgc/fall2021/blob/developer/pubspec.yaml for a complete list of packages.

### 2.1.3      Plugins

The installation of Flutter and Dart plugins instructions vary by platform:

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.

### 2.1.4      Widgets

Flutter widgets are built using a modern framework. “Widgets describe what their view should look like given their current configuration and state. When a widget’s state changes, the widget rebuilds its description, which the framework diffs against the previous description [to] determine the minimal changes needed” (Introduction to widgets, n.d., para. 1).

Table 4 shows the core widgets used in the Memory Magic App.

**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. |

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

# 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.

## 3.1 Android Studio

**Download and Installation**

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

**Figure 1**

*Android Studio*

Graphical user interface, text, application, chat or text message

Description automatically generated

1. To install Android Studio on your computer, follow the official instructions for your OS at https://developer.android.com/studio/install
2. After installing Android Studio successfully, you can continue installing the other tools.

## 3.2 Dart and Flutter

**Download and Installation**

* To download Flutter, visit the official website https://flutter.dev/, select the correct OS and follow the installation instructions for your OS.

**Figure 2**

*Install*

Graphical user interface, application

Description automatically generated

After following the official installation instructions, run Flutter Doctor to verify that your installation has been completed correctly and that Flutter is located the Android Studio installation. Everything should be “green”, and no issues should be found.

**Figure 3**

*Doctor Summary*

Text

Description automatically generated

* 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 Android Emulator

**Set up your Android Emulator**

1. 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

1. 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.

**Figure 4**

*Android SDK*

Graphical user interface, text

Description automatically generated

A close-up of a cell phone

Description automatically generated with medium confidence

1. 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 OS.

## 3.4 Flutter and Dart plugins

**Install Flutter and Dart plugins in Android Studio**

1. To install the correct plugins in Android Studio, please follow the official instructions from https://flutter.dev/docs/get-started/editor?tab=androidstudio.

**Figure 5**

*Dart Plugin*

Graphical user interface, text, application

Description automatically generated

1. After installing the correct plugins your environment should be ready to use.

## 3.5 Test your development environment

**Test your development tools**

1. To test your development environment, you can create a simple Flutter application in your Android Studio: *Create New Flutter Project*.   
   \* Make sure that your Flutter SDK path reflects the specific SDK location.
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 is not required for the development of the application, but for people with less experience using GitHub with command lines, it would be beneficial to download the desktop version.

**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 your GitHub account with GitHub Desktop, or create a new GitHub account.
3. After setting up your account you should be able to access the application repository and collaborate on the code.

# Prepare the Mobile Application for Use

## 4.1 Cloning GitHub repository

Cloning a GitHub repository to a local computer makes it easier to fix conflicts, add/remove files, and push commits.

1.     Navigate to https://github.com/umgc/fall2021

2.     Click on “Code”

3.     To clone the code using HTTPS, click on HTTPS and download ZIP.

**Figure 6**

*HTTPS GitHub*

A screenshot of a computer

Description automatically generated with medium confidence

## 4.2 Run the Flutter application

To run Flutter applications on a device, use the flutter start command:

$ flutter start

The starter app opens on the device.

**Figure 7**

*Flutter Demo*

Chart

Description automatically generated with medium confidence

# 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. Table 5 shows the Test Matrix for the General Tests.

**Table 5**

*Unit Tests*

|  |  |  |
| --- | --- | --- |
| **General Tests:** | **iPhone** | **Android** |
| **1.**     **Application Opens Up**  Success case: Wait for the “Speak” button to appear which means that the application has successfully opened. |  |  |
| **2.**     **From any page, click Calendar**  Success case: Wait for anticipated elements to appear. |  |  |
| **3.**     **From any page, click on Settings**  Success case: Wait for anticipated elements to appear. |  |  |
| **4.**     **From any page, click on Notes**  Success case: Wait for anticipated elements to appear. |  |  |
| **5.**     **From any page, click on Home**  Success case: Wait for anticipated elements to appear. |  |  |
| **6.**     **From any page, click on Bell**  Success case: Wait for anticipated elements to appear. |  |  |
| **7.**     **From any page, click on Mic**  Success case: Wait for anticipated elements to appear. |  |  |
| **Home Screen:** | **iPhone** | **Android** |
| **8.**     **Click on “Speak”**  Success case: Mic screen will appear. |  |  |
| **9.**     **Click on “Text”**  Success case: Enter a note by text screen will appear. |  |  |
| **Mic Screen:** | **iPhone** | **Android** |
| **10.**  **Click on “Press to Record (Mic)”**  Success case: Speak dialog appears. |  |  |
| **Menu Screen:**  **11.**  **Click on sync to cloud**  Success case: Enter state for enabling sync to cloud. |  |  |
| **12.**  **Click on trigger**  Success case: Wait for anticipated elements to appear. |  |  |
| **13.**  **Click on general setting**  Success case: Wait for anticipated elements to appear. |  |  |
| **14.**  **Click on help**  Success case: Wait for anticipated elements to appear. |  |  |
| **Note Screen:** | **iPhone** | **Android** |
| **15.**  **Search for note – no results**  Success case: No results appear. |  |  |
| **16.**  **Search for a note - results**  Success case: Searched for note will appear. |  |  |
| **17.**  **Modify text after search note**  Success case: Note is updated. |  |  |
| **18.**  **Click on “Add A Note”**  Success case: Wait for anticipated elements to appear. |  |  |
| **Delete a Note Screen** | **iPhone** | **Android** |
| **19.**  **Click on Delete Note**  Success case: Note is deleted from datastore. |  |  |
| **20.**  **Click on Cancel** |  |  |
| **Edit a note Screen** | **iPhone** | **Android** |
| **21.**  **Click on Save Changes**  Success case: Note is saved in datastore. |  |  |
| **22.**  **Click on Cancel Changes**  Success case: Note is not saved in datastore. |  |  |
| **Trigger Screen** | **iPhone** | **Android** |
| **23.**  **Change all inputs and save**  Success case: Triggers are saved in datastore. |  |  |
| **24.**  **Change all inputs and cancel**  Success case: Triggers are not saved in datastore. |  |  |
| **Sync to Cloud Screen** | **iPhone** | **Android** |
| **25.**  **Mock login to google cloud**  Success case: After mock response, user is linked to google cloud. |  |  |
| **General Setting Screen** | **iPhone** | **Android** |
| **26.**  **days to keep notes and save**  Success case: Settings are updated for the user. |  |  |
| **27.**  **Change font size notes**  Success case: Settings are updated for the user. |  |  |
| **28.**  **Change font size menu**  Success case: Settings are updated for the user. |  |  |
| **29.**  **Change language from English to Arabic**  Success case: Settings are updated for the user. |  |  |
| **30.**  **Change language from English to Arabic back to English, then save**  Success case: Settings are updated for the user. |  |  |
| **Security Setting Screen** | **iPhone** | **Android** |
| **31.**  **Setup fingerprint**  Success case: Settings are updated for the user. |  |  |
| **32.**  **Reset fingerprint**  Success case: Settings are updated for the user. |  |  |
| **33.**  **Press Cancel Button**  Success case: State is unchanged. |  |  |
| **Calendar Screen** | **iPhone** | **Android** |
| **34.**  **Click a calendar event**  Success case: Wait for anticipated elements to appear. |  |  |

### 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 6 shows the Test Matrix for Integration Tests.

**Table 6**

*Integration Tests*

|  |  |  |
| --- | --- | --- |
| **Integration Tests** | **iPhone** | **Android** |
| **\*\*Integration with NLU (Team B):** |  |  |
| **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 |  |  |
| **Integration with Google Drive:** | **iPhone** | **Android** |
| **3.**     **Application handles successful link with Google API**  Success case: Profile is linked |  |  |
| **4.**     **Application handles unsuccessful link with Google 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.

• Clone the GitHub repository https://github.com/umgc/fall2021

• Execute the application

• Manually test each feature of the application

• Document the test results

• 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

# 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.

## 6.1 Emulator not responding

After installing the mobile emulator, it is possible to encounter issues when trying to start the emulator and/or after using it multiple 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:   
*\* 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 8**

*Tools*

Graphical user interface, text, application

Description automatically generated

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

**Figure 9**

*ADV Manager*

A screenshot of a computer

Description automatically generated with medium confidence

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

**Figure 10**

*Your Virtual Devices*

A screenshot of a computer

Description automatically generated with medium confidence

1. 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:

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 11**

*Your Virtual Devices*

Graphical user interface, text

Description automatically generated

1. 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.
2. After assigning your memory you can click *Apply > Ok*to apply and save the changes.
3. Restart your Android Studio.
4. 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:

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:

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.

# Appendices

## 7.1 Credits

Below are the members that contributed or will contribute to the development of this application:

* Dr. Mir Mohammed Assadullah
* Roy Gordon
* 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