Presented by Team Mesmerize

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

Programmer’s guide

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 4](#_Toc85471009)

[1.1 Purpose 4](#_Toc85471010)

[1.2 Intended Audience 4](#_Toc85471011)

[1.3 Technical Project Stakeholders 4](#_Toc85471012)

[1.4 References 5](#_Toc85471013)

[1.5 Definitions, Acronyms, and Abbreviations 5](#_Toc85471014)

[System Architecture 6](#_Toc85471015)

[Setting Up the Application 9](#_Toc85471016)

[3.1 Installing Flutter on Windows 9](#_Toc85471017)

[3.1.1 System Requirements 9](#_Toc85471018)

[3.1.2 Get the Flutter SDK 10](#_Toc85471019)

[3.1.3 Android Set Up 10](#_Toc85471020)

[3.2 Installing Flutter on macOS 13](#_Toc85471021)

[3.2.1 System Requirements 13](#_Toc85471022)

[3.2.2 Get the Flutter SDK 13](#_Toc85471023)

[3.1.3 iOS Set Up 14](#_Toc85471024)

[3.3 Installing Dart 17](#_Toc85471025)

[3.3.1 System Requirements 17](#_Toc85471026)

[3.3.2 Installing Dart on Windows OS 18](#_Toc85471027)

[3.3.3 Installing Dart on macOS 18](#_Toc85471028)

[Code Structure 18](#_Toc85471029)

[4.1 Front-end Components 20](#_Toc85471030)

[4.2 Back-End Components 21](#_Toc85471031)

[Appendices 22](#_Toc85471032)

[5.1 Credits 22](#_Toc85471033)

# Introduction

## 1.1 Purpose

The purpose of this Programmer's Guide document is to explain the technical assessment of the Memory Magic App. Future developers might use the detailed descriptions as guidance to integrate new features into the Memory Magic App. For instructions on the mobile application and user environment, refer to the Deployment and Operations Guide document.

## 1.2 Intended Audience

This Programmer’s Guide document is intended for current and future developers of the Memory Magic App. This document will provide details on technical programming guidance and

assist future developers to incorporate new features without needing previous knowledge of the Memory Magic App.

## 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

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

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

Get the dart SDK. (n.d.). In *Dart.dev*. Retrieved from https://dart.dev/get-dart

Install homebrew. (n.d.). In *Homebrew*. Retrieved from https://brew.sh/

MacOS install. (n.d.). In *Flutter.dev*. Retrieved from https://flutter.dev/docs/get-started/install/macos

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

The package manager for Windows modern software automation. (n.d.). In *Chocolatey*. Retrieved from https://chocolatey.org/

Windows install. (n.d.). In *Flutter.dev*. Retrieved from https://flutter.dev/docs/get-started/install/windows#android-setup

## 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 |

# System Architecture

This section will provide a high-level overview of the different objects used in the development of the Memory Magic App. The application is developed using the Flutter framework and Dart as the programming language.  Flutter is well recognized for being a cross-platform framework to develop applications for Android, iOS, Linux, Windows, and the web as it allows users to use the same User Interface (UI) and Business Logic on all platforms.

Flutter uses widgets as components where the class of the code resides. The main idea with Flutter is to build the UI using widgets and take advantage of the stateless and stateful characteristics of each widget to manage the different types of events on the application. Memory Magic will use these two types of widgets to develop the application; the section below will provide a list of some of the most important dart files in the application.

* **main.dart** -> contains the main class for the application to launch.
* **Search.dart** -> contains the search widget to search for information in the application.
* **i18n.dart** -> used to support local languages and cultural settings. Will be used to support the different languages provided by the app.
* **Setting.dart** -> used to handle the general settings of the app based on the user's assigned settings.
* **recordnote.dart** -> used to convert speech to text and add it into a note.
* **Note.dart** -> contains the note information and specific details of each note to present to the user.
* ***\*name*Observable.dart** -> the observables are used to capture asynchronous object changes or events and allow the application to handle them for different functionalities.
* **Menu.dart** -> contains the UI configuration and functionalities for the Menu screen.
* **Mic.dart** -> contains the UI configuration and functionalities for the Microphone screen.
* **Note.dart** -> contains the UI configurations and functionalities for the Note screen.
* **Permission.dart** -> used during the onboarding portion to request and assign the necessary permissions to use the local device resources.
* **SelectLanguage.dart** -> used during the onboarding portion to assign the supported language of the application.
* **Trigger.dart** -> contains the UI configurations and functionalities for the Trigger screen.
* **Checklist**.dart -> contains the UI configurations and functionalities for the Checklist screen.
* **Main.dart** -> this is the stateful widget that the main application instantiates.
* **TranslationService.dart** -> service used to translate data in the app based on the locale, eg., (“en”, “US”).
* **CalendarUtility.dart** -> contains the configuration and different utilities of the calendar for integration.
* **EncryptionUtil.dart** -> used to encrypt and decrypt the text notes.
* **SaveNote.dart**->contains the UI configurations and functionalities for the Save Note screen.
* **Boarding.dart** -> contains the UI configurations and functionalities for the Onboarding portion of the app.
* **NotificationScreen**-> contains the UI configurations and functionalities for the Notification screen.

# Setting Up the Application

These instructions will guide the user to set up the development environment in Windows and macOS operating systems. Some steps might be different from one system to the other, but this section will provide the fundamental steps to help users identify the correct tools. For the full step-by-step instructions, refer to the official web pages provided below.

## 3.1 Installing Flutter on Windows

* Navigate to https://flutter.dev/docs/get-started/install
* Choose the Operating System on which to install Flutter – in this case Windows OS.

**Figure 1**

*Install in Windows OS*

Graphical user interface, text, application

Description automatically generated

### 3.1.1 System Requirements

To install and run Flutter, the development environment must meet the below minimum requirements:

* Operating Systems: Windows 7 SP1 or later (64-bit), x86-64 based
* Disk Space: 1.64 GB

### 3.1.2 Get the Flutter SDK

1.     Download the following installation bundle to get the latest stable release of the Flutter SDK: flutter\_windows\_2.5.2-stable.zip (find latest .zip version at https://flutter.dev/docs/get-started/install/windows#android-setup).

2.     Extract the zip file and place the contained flutter in the desired installation location for the Flutter SDK.

3.     The system is ready to run Flutter commands in the Flutter Console.

### 3.1.3 Android Set Up

**\*\*\***Flutter relies on a full installation of Android Studio to supply its Android platform dependencies\*\*\*

* **Install Android Studio**

1. Download and install Android Studio - https://developer.android.com/studio
2. Start Android Studio, and follow the ‘Android Studio Setup Wizard’.
3. The Android Studio Setup Wizard installs the latest version of Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools.

* **Set up the Android emulator**

To run and test the Flutter app on the Android emulator, follow the below steps:

1. Enable VM acceleration on your machine.
2. Launch Android Studio, click the AVD Manager icon, and select Create Virtual Device

**Figure 2**

*AVD Manager*

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

Description automatically generated

1. Choose a device definition and select Next.

**Figure 3**

*Select Hardware*

*Graphical user interface

Description automatically generated*

1. Select one or more system images for the Android versions you want to emulate and select Next. Recommendation: *x86* or *x86\_64* image.

**Figure 4**

*System Image*

*Graphical user interface, application, Teams

Description automatically generated*

1. Under Emulated Performance, select Hardware - GLES 2.0 to enable hardware acceleration.
2. Verify the AVD configuration is correct and click Finish.

**Figure 5**

*AVD Configuration*

*Graphical user interface, application, Teams

Description automatically generated*

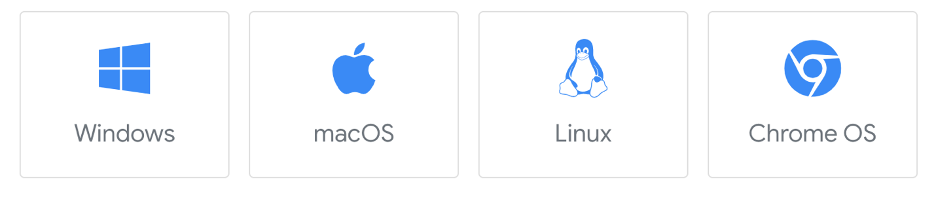
1. In Android Virtual Device Manager, click Run in the toolbar. The emulator starts up and displays the default canvas for your selected OS version and device.

## 3.2 Installing Flutter on macOS

* Navigate to https://flutter.dev/docs/get-started/install
* Choose the Operating System on which to install Flutter – In this case macOS.

**Figure 6**

*Install in macOS*



### 3.2.1 System Requirements

To install and run Flutter, the development environment must meet the below minimum requirements:

* Operating Systems**:**macOS
* Disk Space**:**2.8 GB

### 3.2.2 Get the Flutter SDK

1. Download the following installation bundle to get the latest stable release of the Flutter SDK: flutter\_macos\_2.5.2-stable.zip ((find latest .zip version at https://flutter.dev/docs/get-started/install/macos).
2. Extract the file in the desired location.
3. Add the flutter tool to your path: $ export PATH= “$PATH: `pwd`/flutter/bin”
4. This command sets your PATH variable for the *current* terminal window only.
5. The system is ready to run Flutter commands.

### 3.1.3 iOS Set Up

* **Install Xcode**

To develop applications for iOS using Flutter, Xcode must be installed.

1. Install the latest stable version of Xcode.
2. Configure the Xcode command-line tools by running the following from the command line:

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

$ sudo xcodebuild –runFirstLaunch

1. Make sure the Xcode license agreement is signed by opening Xcode once and confirming.
2. With Xcode, the system is able to run Flutter applications on an iOS device or on the simulator.

* **Set up the iOS Simulator**

To run and test the Flutter app on the iOS simulator, follow the below steps:

1.     On Mac, find the Simulator via Spotlight or by using the following command:

$ open -a Simulator

1. Make sure your simulator is using a 64-bit device to verify: check the settings in the simulator’s Hardware > Device menu.

* **Install Android Studio**

1. Download and install Android Studio - https://developer.android.com/studio
2. Start Android Studio and follow the ‘Android Studio Setup Wizard’.
3. The Android Studio Setup Wizard installs the latest version of Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools.

* **Set Up the Android Emulator**

To run and test your Flutter app on the Android emulator, follow the below steps:

1. Enable VM acceleration on your machine.
2. Launch Android Studio, click the AVD Manager icon, and select Create Virtual Device.

**Figure 7**

*AVD Manager*

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

Description automatically generated

1. Choose a device definition and select Next.

**Figure 8**

*Select Hardware*

*Graphical user interface

Description automatically generated*

1. Select one or more system images for the Android versions you want to emulate and select Next. Recommendation: *x86* or *x86\_64* image.

**Figure 9**

*System Image*

*Graphical user interface, application, Teams

Description automatically generated*

1. Under Emulated Performance, select Hardware - GLES 2.0 to enable hardware acceleration.
2. Verify the AVD configuration is correct and click Finish.

**Figure 10**

*AVD Configuration*

*Graphical user interface, application, Teams

Description automatically generated*

1. In Android Virtual Device Manager, click Run in the toolbar. The emulator starts up and displays the default canvas for your selected OS version and device.

## 3.3 Installing Dart

As the below instructions explain, Dart can be easily installed by using a package manager.

### 3.3.1 System Requirements

The Dart SDK is supported on Windows and macOS.

* **Windows**
* Supported versions: Windows 10.
* Supported architectures: x64, ia32.
* **macOS**
* Supported versions:
* macOS 10.14 (Mojave)
* macOS 10.15 (Catalina)
* macOS 11 (Big Sur)
* Supported architectures: x64, arm64.

### 3.3.2 Installing Dart on Windows OS

You can install the Dart SDK using Chocolatey - https://chocolatey.org/

To install the Dart SDK:

C:\> choco install dart-sdk

To upgrade the Dart SDK:

C:\> choco upgrade dart-sdk

### 3.3.3 Installing Dart on macOS

Install Homebrew - https://brew.sh/

Run the following commands:

$ brew tap dart-lang/dart

$ brew install dart

To upgrade when a new release of Dart is available:

$ brew upgrade dart

# Code Structure

The Memory Magic application was developed using Flutter framework and Dart programming language. One of the advantages of using Flutter is that the application will function in either iOS or Android mobile operating systems. Figure 11 shows the components used during the development of this project. These components are further broken down into front-end and back-end components.

**Figure 11**

*Code Structure*

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

Description automatically generated*

Some important folders and files in the code structure include:

* android – files with specific coding for Android development.  This project is using the default setting.
* assets -   can be defined as folders with images/files that can be used throughout the project.  Within the assets folder other folders were created such as Android, and NLU folders. Within the Android folder developers can access Android specific files and images, and within the NLU folder developers can access the files used in the NLU integration.
* i18n –files used in language translation.
* ios – files with specific coding for iOS development. This project is using the default setting.

lib - folder responsible for the front end and back-end code. This will be further discussed in sections 4.1 and 4.2.

* test – folder where unit tests files are stored.
* i18nconfig.json – file where configurations for language translation is stored.
* pubspec.yaml – file containing configuration information for the project.   This includes the project name, the project environment, any dependencies used in the project, and other information such as assets and icons.

## 4.1 Front-end Components

This section contains the code for the screens and the front-end components are organized in the lib > screens section of the application (Figure 12). Within the screens folder, the different screens are further broken down into the following folders:

* Menu – files related to the menu of the application.
* Mic – files related to the microphone screen of the application.
* Note – files related to the note screens of the application, including the details and save screens for the notes.
* Onboarding – files related to the onboarding section of the application that occurs when the user opens the application for the first time.
* Settings - files related to the settings sections of the application.

**Figure 12**

*Code Structure of Front-End Components*

Graphical user interface, text, application

Description automatically generated

## 4.2 Back-End Components

The back-end components are also contained in the lib folder, which has several subfolders containing specific items (Figure 13). The subfolders related to back-end components consist of the following:

* Components – specific component code used in the application.
* Generated – files that are generated for use in the application, specifically the i18.dart file used for language translation
* Model – files with the data model classes used in the application.
* Observables –files used for mobx observables used throughout the application.
* Services – files containing code that is reused throughout the application and it is considered service.
* Utility- files containing code that is considered utility.

**Figure 13**

*Code Structure of Back-End Components*

Graphical user interface, text

Description automatically generated with medium confidence

# Appendices

## 5.1 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