

**Programmer Guide**

MailSpeak Application (MSA)

University of Maryland Global Campus (UMGC)

Software Engineering (SWEN) 670

Fall Cohort 2022

Team B

November 5, 2022

Approval Signatures

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ☐ APPROVE  ☐ DISAPPROVE | |  | | --- | |  | | Tatiana Kozhevnikova, Product Owner  Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ | |  | |
| ☐ APPROVE  ☐ DISAPPROVE | |  | | --- | |  | | Michael Conatser, Project Manager  Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ | |

**Document Control**

**Document Information**

|  | Information |
| --- | --- |
| Document Id | USPS-MSA-Programmer-Guide-20221029-Fall2022 |
| Document Owner | UMGC SWEN 670 |
| Issue Date | October 29, 2022 |
| Last Saved Date | November 5, 2022 |
| File Name | Programmers Guide - Team B - MSA.docx |

**Document History**

| Version | Issue Date | Changes |
| --- | --- | --- |
| 0.1 | 9/18//2022 | Initial Draft |
| 0.2 | 10/29/2022 | Review Updates |
| 1.0 | 10/29/2022 | Final Deliverable |
| 1.1 | 10/30/2022 | References style and document suite updates |
| 1.2. | 11/4/2022 | Feedback updates |
| 2.0 | 11/5/2022 | Final deliverable |

**Table of Contents**

[1 Introduction 6](#_Toc117921143)

[1.1 Purpose 6](#_Toc117921144)

[1.2 Intended Audience 6](#_Toc117921145)

[1.3 Technical Project Stakeholders 6](#_Toc117921146)

[1.4 Project Documentation 6](#_Toc117921147)

[1.4.1 Project Suite of Documents 6](#_Toc117921148)

[1.4.2 Document References 7](#_Toc117921149)

[2 Development Environment 7](#_Toc117921150)

[2.1 Flutter and Dart](#_Toc117921151)

[2.2 Android Studio](#_Toc117921152)

[2.3 Setting up Android Emulator](#_Toc117921153)

[2.4 Setting up Android Device](#_Toc117921154)

[2.5 Setting up iOS Simulator](#_Toc117921155)

[2.6 Setting up iOS Device](#_Toc117921156)

[2.7 Testing Development Environment](#_Toc117921157)

[2.8 Why GitHub Desktop](#_Toc117921158)

[3 Development Code](#_Toc117921159)

[3.1 Code Repository](#_Toc117921160)

[3.2 Cloning the Repository](#_Toc117921161)

[3.3 Code Branches](#_Toc117921162)

[4 Application Architecture](#_Toc117921163)

[4.1 Architecture](#_Toc117921164)

[4.2 Design Pattern](#_Toc117921165)

[5 Code Structure](#_Toc117921166)

[6 Development](#_Toc117921167)

[6.1 Programming Languages](#_Toc117921168)

[6.2 Flutter Plugins and Libraries](#_Toc117921169)

[6.3 Implementing Google Cloud Vision API](#_Toc117921170)

[6.4 Implementing Digest Email Parsing/IMAP](#_Toc117921171)

[6.5 Implementing Chatbot](#_Toc117921172)

[6.6 Implementing Mail Search](#_Toc117921173)

[6.7 Implementation of Accessibility Features](#_Toc117921174)

[6.8 Implementation of Gmail Sign In/Reading Gmail’s Emails](#_Toc117921175)

[7 Testing](#_Toc117921176)

[7.1 Create a Unit Test](#_Toc117921177)

[7.2 Running Unit Tests](#_Toc117921178)

[Appendices](#_Toc117921179)

[Appendix A - Definitions, Acronyms, and Abbreviations](#_Toc117921180)

# Introduction

## Purpose

The programmer’s guide (PG) for the MailSpeak Application (MSA) will provide detailed information on the technical setup to create a full development environment as well as provide technical reasoning for the key design decisions adopted. The technical setup will provide an environment which will support development, testing, configuration management, change management, and implementation of the application under development. This document should be used in conjunction with the other documents provided for this development effort detailed in section 1.4.

## Intended Audience

The intended audience for the programmer’s guide is the technical project team and stakeholders. This document will be a necessary guideline for future development efforts regarding MSA and will provide crucial technical and programmatic information allowing for the continued innovation of the application.

## Technical Project Stakeholders

The technical project stakeholders for this project are provided in the following table:

Figure 1.3.1 - Project Stakeholders

|  |  |
| --- | --- |
| Stakeholder Name | Project Role |
| Mir Assadullah | Professor, Program Manager |
| Roy Gordon | Project Mentor |
| Robert Wilson | DevSecOps Mentor |
| Alexander Chan | Software Engineer (SE) II |
| Andrew Asavarungsrikul | Software Engineer (SE) I |
| Erin Sauter | Software Engineer (SE) I |
| Jonay Simmons | Software Engineer (SE) I |
| Lawrence Van Tassel | Software Engineer (SE) III |
| Michael Conatser | Project Manager (PM), Scrum Master |
| Minyahil Kebebegn | Software Engineer (SE) II |
| Sarah Johnson | Software Engineer (SE) I |
| Shane Knowles | DevSecOps Engineer (Principal) |
| Tatiana Kozhevnikova | Product Owner |
| TraMel Perry | Principal Software Engineer (SE) |

## Project Documentation

### Project Suite of Documents

This Programmer Guide is part of a set of essential documents created to adequately manage, control and deliver the Mail Speak Application. Artifacts that are provided within the document package contain vital information for the application’s ongoing support and operation throughout its life cycle. Each document is created within the specific Milestone of the project. Therefore, the version and date of some documents could be marked as “to be determined” (TBD) in Table 1.1.

Table 1.4.1 - Project Documentation Package

| Document | Version | Date |
| --- | --- | --- |
| Project Management Plan (PMP) | 4.0 | 11/05/2022 |
| Software Requirements Specification | 4.0 | 11/05/2022 |
| Technical Design Document | 3.0 | 11/05/2022 |
| Software Test Plan | 3.0 | 11/05/2022 |
| Programmer Guide | 2.0 | 11/05/2022 |
| Development and Operations Guide | 2.0 | 11/05/2022 |
| User Guide | 1.0 | 11/05/2022 |
| Test Report | 1.0 | 11/05/2022 |

### Document References

During the process of writing current Programmer Guide the following documents were referenced:

* Team B. (2022). *United Global Master Coders Team B PMP*. <https://umgcdev361.sharepoint.com/:w:/r/sites/SWEN670Fall2022/Shared%20Documents/Team%20B%20Channel/Milestone%201%20(SAT%20SEP%203)/ProjectManager-Project-Plan-Template.docx?d=w671384dfe89d46d7a2583b60416fb909&csf=1&web=1&e=xeRN2o>
* Team B. (2022). *United Global Master Coders Team B SRS*. <https://umgcdev361.sharepoint.com/:w:/r/sites/SWEN670Fall2022/Shared%20Documents/Team%20B%20Channel/Milestone%201%20(SAT%20SEP%203)/Informed%20Delivery%20Enhancement%20Team%20B%20SRS.docx?d=w9b27ad03c5c145a09edc6b22427bc8ba&csf=1&web=1&e=85qCsH>
* Team B. (2022). *United Global Master Coders Team B TDD*. <https://umgcdev361.sharepoint.com/:w:/r/sites/SWEN670Fall2022/Shared%20Documents/Team%20B%20Channel/Milestone%202%20(SAT%20SEP%2017)/TDD%20-%20Team%20B%20-%20MSA.docx?d=wf0a49ae72fc0497b9dfb0322352b8025&csf=1&web=1&e=fgAyQh>

# Development Environment

## Flutter and Dart

Flutter is an open-source framework by Google for building multi-platform applications from a single codebase.

Dart is a programming language designed for client development, such as for web and mobile apps. It is developed by Google and can also be used to build servers and desktop applications.

Dart is an object-oriented, class-based, garbage-collected language with C-style syntax. It can compile to either native code or JavaScript, and supports interfaces, mixins, abstract classes, reified generics and type inference.

The Flutter framework is utilized in conjunction with Dart applications to create fast, natively compiled applications. For developing multiplatform applications, Flutter prevents the hassle of writing separate codebases for each platform, such as for Android and iOS. This saves development effort by basing the application off of a single codebase.

While the Dart Software Development Kit (SDK) can be installed separately, the Dart SDK with Command Line Interface (CLI) is also part of the Flutter SDK. The Flutter SDK installation varies depending on the OS, and there are multiple methods of installing Flutter, such as using Git or Homebrew.

Flutter enables developers to create cross-platform applications using one of the following platforms as listed in the official documentation <https://docs.flutter.dev/get-started/install>

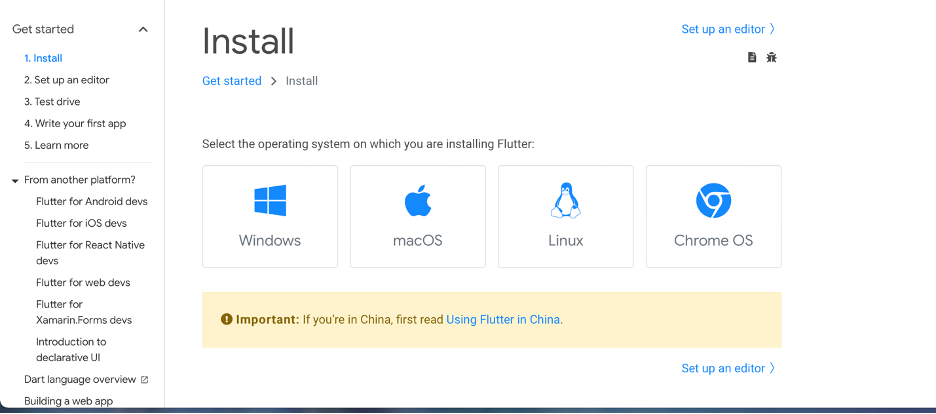


Figure 2.1.1 - Flutter Installation Options

**Windows**

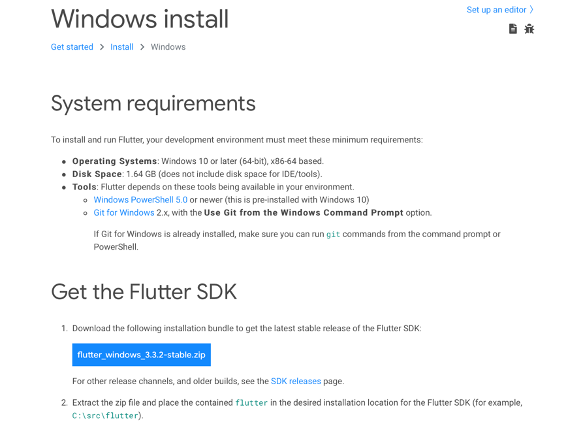


Figure 2.1.2 - Windows System Requirements and Flutter SDK Installation

1. Download Flutter SDK package by clicking link below (as of Sep 2022)

[flutter\_windows\_3.3.2-stable.zip](https://storage.googleapis.com/flutter_infra_release/releases/stable/windows/flutter_windows_3.3.2-stable.zip)

1. Extract the zip file and place the contained flutter in the desired installation location for the Flutter SDK (for example, C:\src\flutter).
2. update Path Variable for Windows PowerShell

* From the Start search bar, enter ‘env’ and select **Edit environment variables for your account**.
* Under **User variables** check if there is an entry called **Path**:
  + If the entry exists, append the full path to flutter\bin using ; as a separator from existing values.
  + If the entry doesn’t exist, create a new user variable named Path with the full path to flutter\bin as its value.

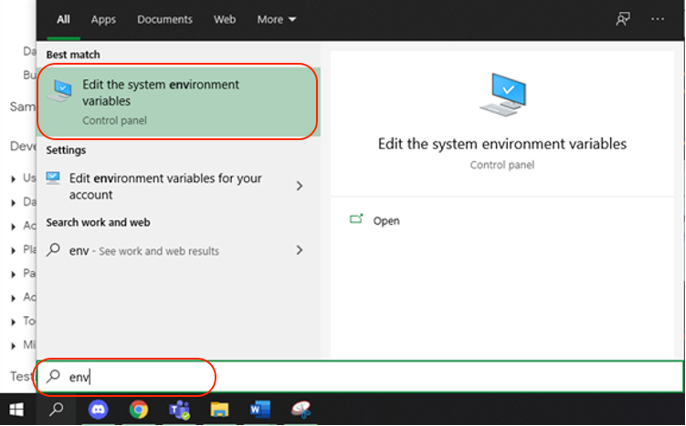


Figure 2.1.3 - Windows Environment Variable Setup

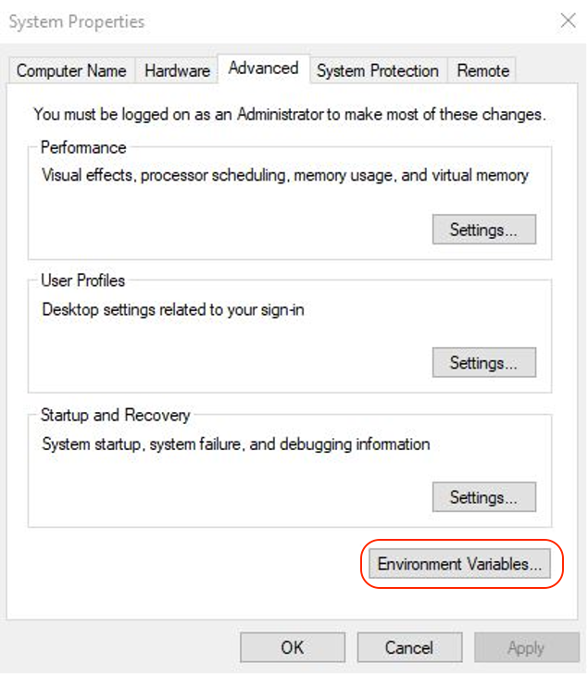


Figure 2.1.4 - Windows Environment Variable Setup

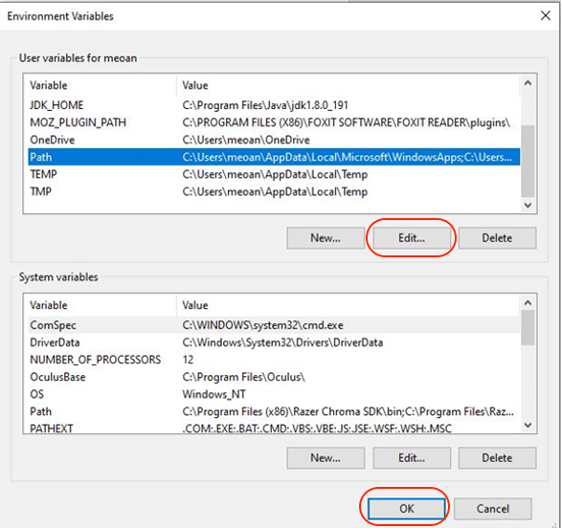


Figure 2.1.5 - Windows Environment Variable Setup

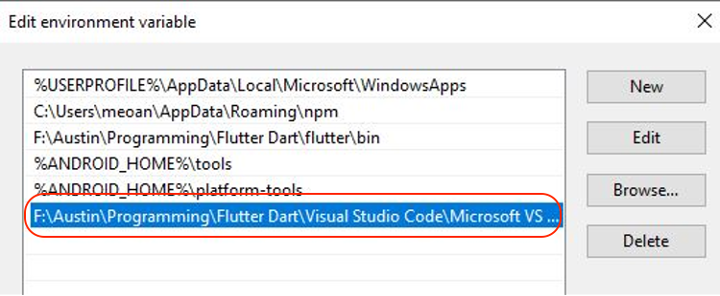


Figure 2.1.6 - Windows Environment Variable Setup

1. Open the command prompt and enter “flutter doctor” to confirm.

$ flutter doctor

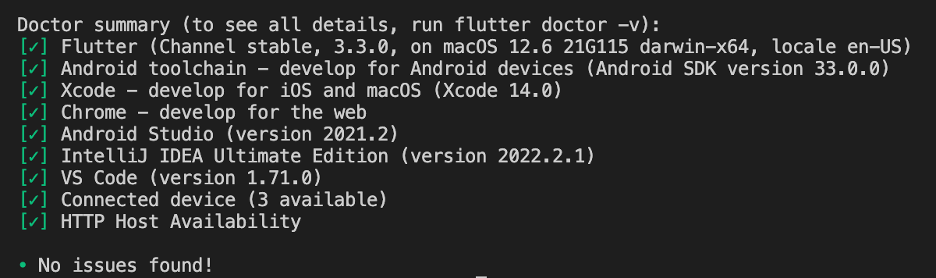


Figure 2.1.7 – “flutter doctor” Output

**macOS**

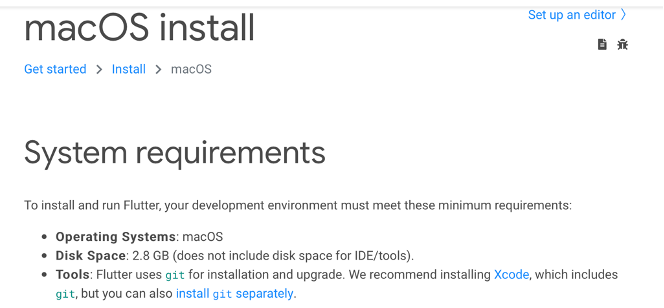


Figure 2.1.8 - MacOS System Requirements

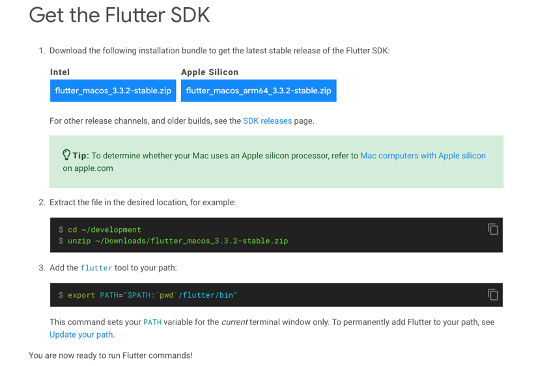


Figure 2.1.9 - macOS Flutter SDK Installation

* 1. Download the flutter SDK from the link below (as of Sep 2022)

**Apple Silicon:** [flutter\_macos\_arm64\_3.3.2-stable.zip](https://storage.googleapis.com/flutter_infra_release/releases/stable/macos/flutter_macos_arm64_3.3.2-stable.zip)

**Intel:** [flutter\_macos\_3.3.2-stable.zip](https://storage.googleapis.com/flutter_infra_release/releases/stable/macos/flutter_macos_3.3.2-stable.zip)

* 1. Extract the file in the desired location
  2. Add the flutter tool to your path from CLI

$ *export PATH="$PATH:/[flutter\_path]/flutter/bin"*

Note: replace ‘flutter\_path’ with the location of flutter package

* 1. Run the following command to confirm

*$ flutter doctor*

* 1. Flutter SDK can also be installed directly from GitHub using the following git command

*$ git clone* [*https://github.com/flutter/flutter.git*](https://github.com/flutter/flutter.git) *-b stable*

**Linux**

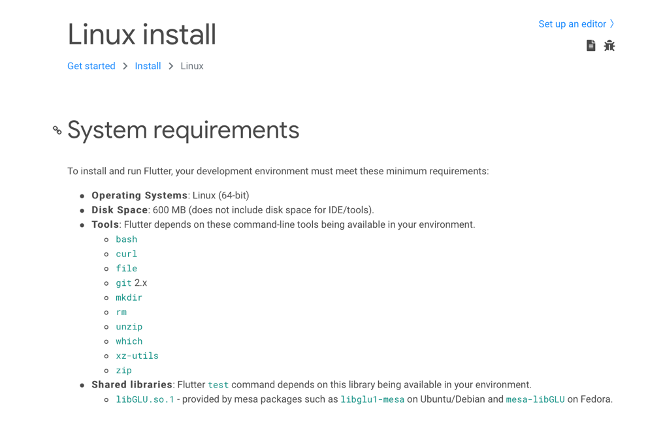


Figure 2.1.10 - Linux System Requirements

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

[flutter\_linux\_3.3.2-stable.tar.xz](https://storage.googleapis.com/flutter_infra_release/releases/stable/linux/flutter_linux_3.3.2-stable.tar.xz)

1. Extract the file in the desired location, for example:

$ cd ~/[flutter\_path]

$ tar xf ~/[desired\_location]/flutter\_linux\_3.3.2-stable.tar.xz

1. Add flutter path to your tool

$ *export PATH="$PATH:/[flutter\_path]/flutter/bin"*

1. Run the following command to confirm

*$ flutter doctor*

**Chrome OS**

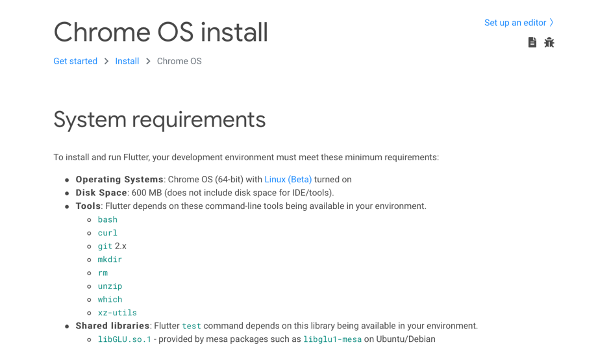


Figure 2.1.11 – Chrome OS System Requirements

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

[flutter\_linux\_3.3.2-stable.tar.xz](https://storage.googleapis.com/flutter_infra_release/releases/stable/linux/flutter_linux_3.3.2-stable.tar.xz)

1. Extract the file in the desired location, for example:

$ cd ~/[flutter\_path]

$ tar xf ~/[desired\_location]/flutter\_linux\_3.3.2-stable.tar.xz

1. Add flutter path to your tool

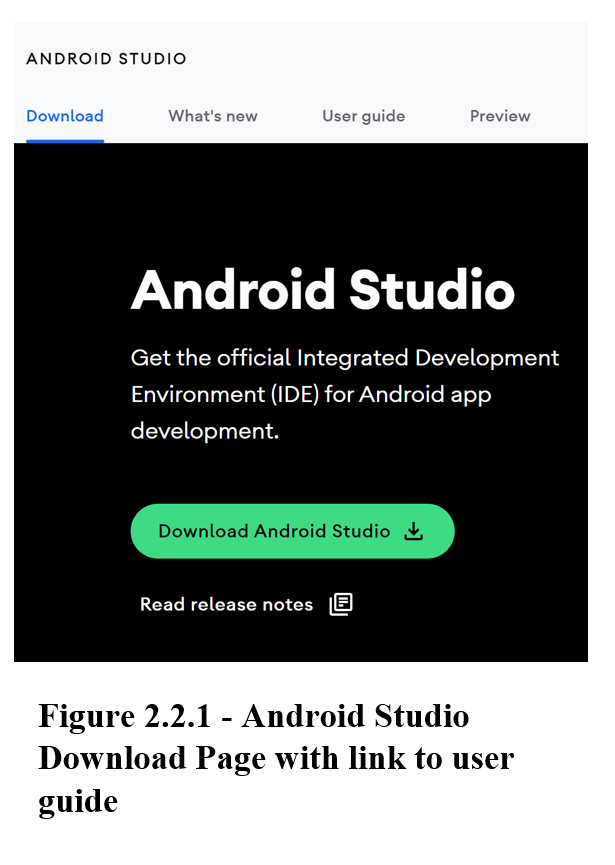
$ *export PATH="$PATH:/[flutter\_path]/flutter/bin"*

1. Run the following command to confirm

*$ flutter doctor*

## Android Studio

Android Studio is the official IDE for Google’s Android OS and thus is designed specifically for Android development. It is available for download and use on Windows, macOS, and Linux devices. The most current release, as of September 2022, is Dolphin.

**Download:**

<https://developer.android.com/studio#downloads>

**Installation Videos:**

https://developer.android.com/studio/install

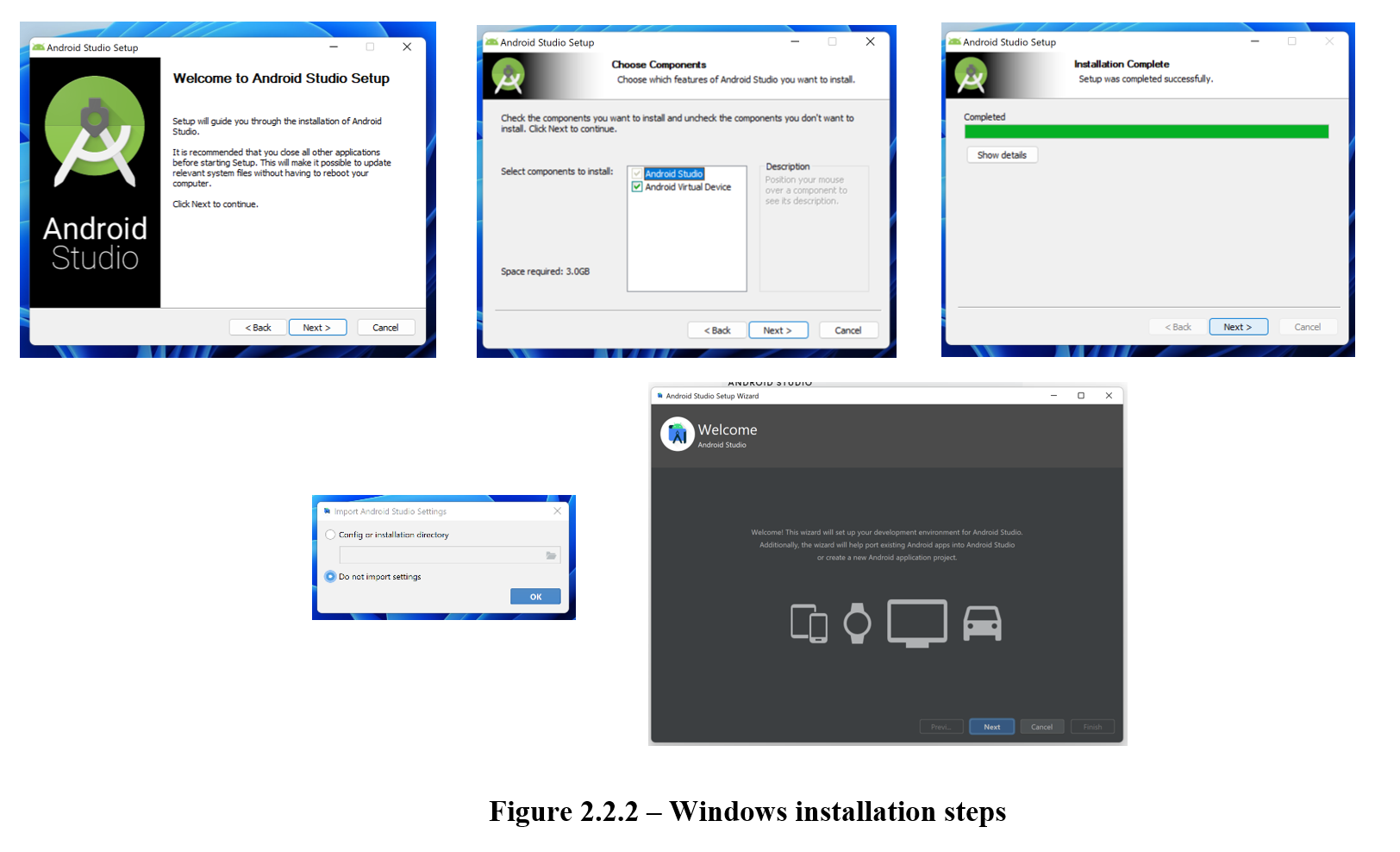
**Windows**:

System Requirements: 64-bit of Windows 8/10/11, x86\_64 CPU architecture, 2nd generation Intel Core or newer, minimum 8 GB RAM, minimum 8 GM available disk space, minimum 1280 x 800 screen resolution.

Packages:

1. 64-bit .exe (recommended)
2. 64-bit .zip (no .exe installer)

Installation:



1. If .exe file downloaded, double-click to start application.  
   If .zip file downloaded, extract the folder, copy the “android-studio” folder into your “Program Files” folder. Then open the “android-studio > bin” folder and launch “studio64.exe”.
2. Follow the Android Studio Setup Wizard and install the recommended SDK packages.

**MacOS:**

System Requirements: MacOS 10.14 (Mojave) or higher, ARM-based chips or 2nd generation Intel Core or newer, minimum 8 GM RAM, minimum 8 GB available disk space, minimum 1280 x 800 screen resolution

Packages:

1. 64-bit
2. 64-bit, ARM

Installation:

Graphical user interface, application

Description automatically generated

1. Launch the Android Studio DMG file.
2. Drag and drop the Android Studio icon into the Applications folder.
3. Launch Android Studio.
4. Select whether you want to import previous Android Studio settings, then click the “OK” button.
5. Follow the Android Studio Setup Wizard and install the recommended SDK packages.

**Linux:**

System Requirements: Any 64-bit Linux distribution that supports Gnome, KDE, or Unity DE, x86\_64 CPU architecture, 2nd generation Intel Core or newer, minimum 8 GM RAM, minimum 8 GB available disk space, minimum 1280 x 800 screen resolution.

Packages:

1. 64-bit

Installation:

Graphical user interface, application

Description automatically generated

1. Unzip the zip file to an appropriate location for your applications.
2. To launch, open a terminal, navigate to the “android-studio/bin/” directory, and execute “studio.sh”.
3. Select whether you want to import previous Android Studio settings, then click the “OK” button.
4. Follow the Android Studio Setup Wizard and install the recommended SDK packages.

**Chrome OS:**

System Requirements: Linux for Chrome OS, Intel i5 or higher recommended, minimum 8 GM RAM, minimum 4 GB available disk space, minimum 1280 x 800 screen resolution.

Graphical user interface, application

Description automatically generatedPackages:

1. Chrome

Installation:

1. Right click the downloaded DEB file and select “Install with Linux (Beta)”.
2. Select whether you want to import previous Android Studio settings, then click the “OK” button.
3. Follow the Android Studio Setup Wizard and install the recommended SDK packages.
4. To launch, either start from the Launcher, or from the Chrome OS Linux terminal by running “studio.sh” in the default installation directory: “/opt/android-studio/bin/studio.sh”.

## Setting up Android Emulator

Android Emulator allows users to simulate Android devices so that the application can be tested on multiple Android devices without having to have the actual devices. This emulator provides almost all the capabilities and functionality of the actual device, so the application can be tested in multiple situations.

Installation: Android Emulator comes with Android Studio by default.

System Requirements: 64-bit Windows, macOS, or Linux operating system, 16 GB RAM, and 16 GB available disk space.

Setup:

1. Create an Android Virtual Device (AVD) with the desired Android version and hardware specifications. This can be done by:
   1. Opening the “Device Manager”.
   2. Click the “Create Device” button.
   3. Follow the setup to select the desired hardware and system image
2. Run the application on the emulator by selecting the desired AVD from the target device drop-down menu and clicking the “Run” button.

Graphical user interface

Description automatically generated

## Setting up Android Device

It is crucial that the application is also tested on Android devices. Using Android Studio, an Android device can be set up to run the application, using either a USB or Wi-Fi.

Setup:

1. On the device, open the “Settings” app, select “Developer Options”, then enable “USB debugging” (if connecting with a USB).
   1. If “Developer Options” is not enabled, go to “Settings > About Phone > Software Information” and tap the “Build Number” option 7 times.

Graphical user interface

Description automatically generated

1. Setup your system to detect the device.
   1. Chrome OS and macOS: no additional configuration required.
   2. Windows: install a USB driver for ADB (if using a USB to connect device). Instructions found [here](https://developer.android.com/studio/run/oem-usb).
   3. Ubuntu Linux: each user needs to be in the plugdev group, and the system needs to have udev rule installed.
2. Connect to device using USB:
   1. Once the device has been plugged in using a USB, click the “Run” button in Android Studio to build and run the application on your device.
3. Connect using Wi-Fi:
   1. Ensure the device is running Android 11 or higher.
   2. Ensure you have the “Canary” version of Android Studio and update to the latest version of “SDK Platform-Tools".
   3. Open Android Studio and select “Pair Devices Using Wi-Fi" from the run configurations drop-down menu.
   4. Enable “Pair device with QR code” by going to “Settings > Developer Options > Wireless Debugging > Pair device with QR code” and scan the QR code on the screen.

Qr code

Description automatically generated

## Setting up iOS Simulator

Simulator is a development tool to simulate iOS, iPadOS, tvOS, and watchOS on the Mac. It helps rapid prototyping and testing builds by simulating different devices during the development process. IOS simulator is Installed as part of the XCode tools. IOS Simulator runs on the Mac and behaves like a standard Mac app while simulating an iPhone, iPad, Apple Watch, or Apple TV environment.

An iOS Simulator needs to be set up to test and debug the application on an iOS platform. It mimics the application that runs on an iOS device. For the development of the USPS Informed Delivery App, we have to simulate to an iOS device to test the “Scan mail “feature that utilizes the device camera. iOS simulators do not allow access to cameras.

**Setting Up an iOS Simulator (Mac User)**

To prepare to run and test a Flutter app on the iOS simulator, follow these steps:

Find an IOS simulator from

1. Spotlight Search or



Figure 2.5.1 - Spotlight Search

1. By running a command ‘$ open -a simulator’ in a terminal window or command line interface.



Figure 2.5.2 - Simulator Device

To select a different iOS device as a simulator, right-click on file and select open simulator and choose preferred device from the drop-down menu.

One of the main features of the MailSpeak application is the ability to scan an image of a letter and QR/Barcode using the user's phone camera. Since the iOS Simulator does not support camera functionality, the application needs to be deployed to a physical iOS device, such as an iPhone or iPad.

The image\_picker Flutter plugin is used for taking new pictures with the camera and selecting images from an iOS photo library. The image\_picker is added as a dependency in the pubspec.yaml file.

dependencies:

image\_picker: ^0.8.5+3

The user will need to explicitly grant access to the MailSpeak application for it to use the iOS Camera app. In order for the user to grant access, the following <key> and <string> variables need to be added in the “Info.plist” file under the application's “ios/Runner” folder. These will make the "permission pop-up screen" appear the first time the user opens the application.

* NSCameraUsageDescription: A message that tells the user why the app is requesting access to the device’s camera.
* NSPhotoLibraryUsageDescription: A message that tells the user why the app is requesting access to the user’s photo library.
* NSPhotoLibraryAddUsageDescription: A message that tells the user why the app is requesting add-only access to the user’s photo library.

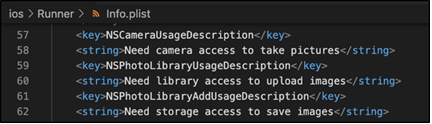


Figure 2.5.3 - Info.plist

The user can also modify the permission settings for an application in the iOS device's Settings app, then the "Privacy" section.

The following are the steps to deploy the MailSpeak application to an iOS device.

## Setting up iOS Device

To deploy an application to a physical iOS device, a physical device deployment in XCode needs to be set up. It is assumed that the user has an existing Apple ID which will be used as an Apple Developer account to access a limited set of developer-oriented privileges for free. Follow the XCode signing process to provide an application.

1. Open the application's default XCode workspace by doing one of the following steps.
   * Option 1: Run the "open ios/Runner.xcworkspace" in a Terminal window or command line interface from the application project directory to open the project.

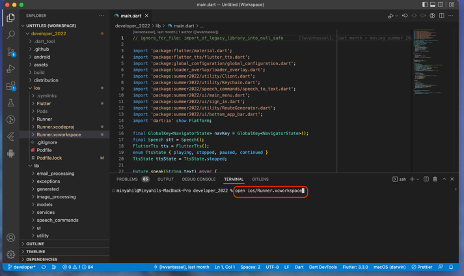


Figure 2.6.1 - Terminal Window

* + Option 2: Double click the "Runner.xcworkspace" file from the application project "ios" directory to open the project.

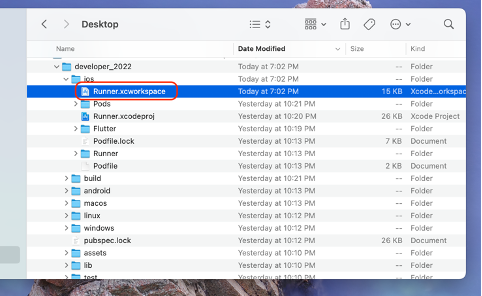


Figure 2.6.2 - iOS Directory

1. Once the application project is opened in XCode, connect an iOS device using a USB cable and select an iOS device.

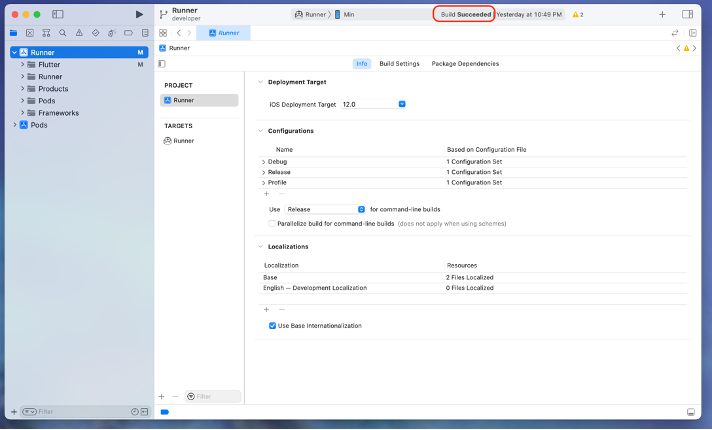


Figure 2.6.3 - XCode Application

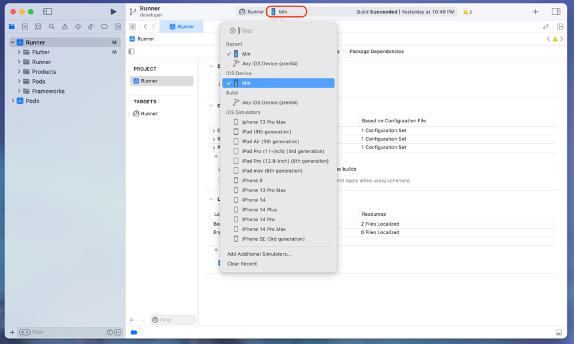


Figure 2.6.4 - XCode Application

1. In the "Runner" Target settings page, select the Development Team in the "Team" section of the "Signing & Capabilities" tab and check the "Automatic manage signing" checkbox. This tells XCode to create and download a Development Certificate, register a device with the account, and create and download a provisioning profile (if needed).

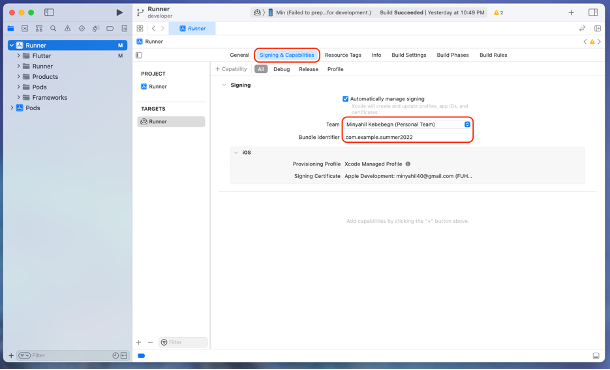


Figure 2.6.5 - Runner Settings Page

If a user is already set up, select the account as the Development Team. Otherwise, select "Add an Account" to set up the user's Development Team account. An Apple ID will suffice for debugging and testing purposes.

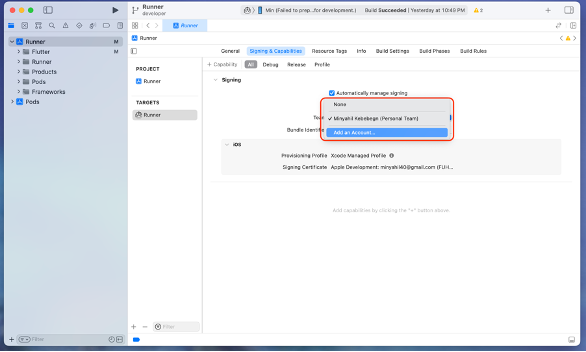


Figure 2.6.6 - Runner settings

If there is a Bundle Identifier registration error, provide a unique bundle identifier, e.g. com.example.fall2022.

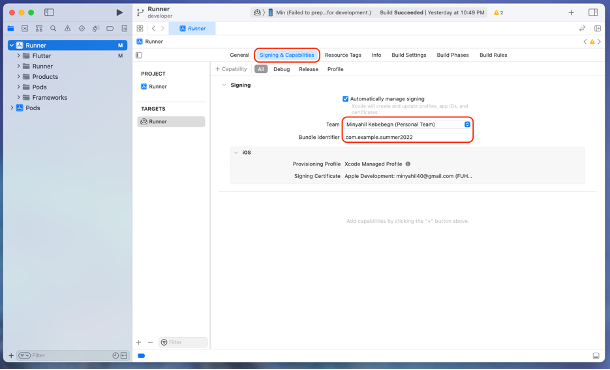


Figure 2.6.7 - Adding bundle identifier

In the "Runner" Target settings page, if necessary, update the application's display name in the "Identity" section of the "General" tab.

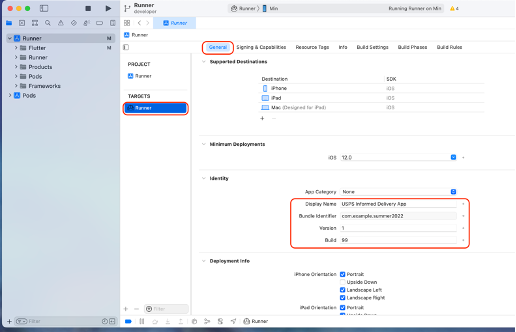


Figure 2.6.8 - Runner Settings General Tab

1. Click the "Run" button in XCode to build and deploy the application to the connected iOS device.

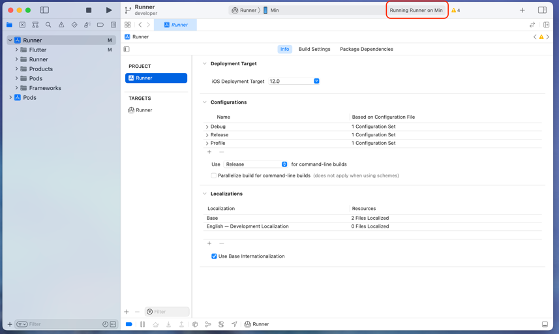


Figure 2.6.9 - Click Run

1. If this is the first time the application is being deployed to this particular iOS device, the application needs to be trusted on that device. On the iOS device, go to the Settings app and navigate to the "General" section. From the "General" section, navigate to the "(VPN &) Device Management" section, then click on "Apple Development." under the “Developer App” and click “Trust Apple Development.” to trust the application.

***Note:*** For iOS 16, When connecting an iOS device to Mac for the first time, the XCode scheme selector’s destination list shows it as an “Unavailable Device”, because Developer Mode is disabled. For locally installed apps to run on iOS, iPadOS, and watchOS devices, the developer app should be enabled.

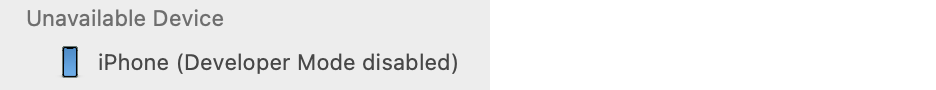


Figure 2.6.10 - Developer Mode

To enable Developer Mode, go to Settings > Privacy & Security > Developer Mode to show the Developer Mode toggle switch.

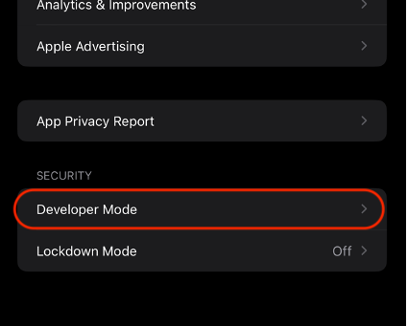




Figure 2.6.11 - Switch Developer Mode

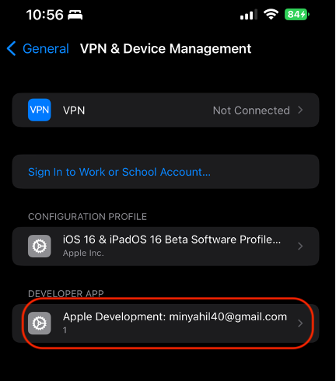
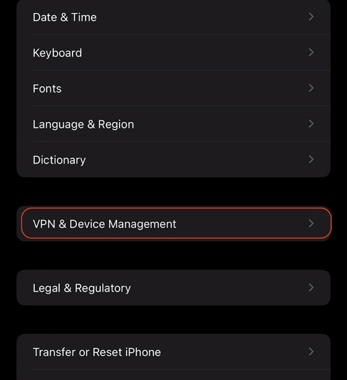


Figure 2.6.12 - Settings General VPN & Device Management

## Testing Development Environment

Testing will be performed using Apple iOS and Android OS. The majority of developmental testing will be performed using an Android Emulator and iOS Emulator as described in Sections 2.3 and 2.5. Additional testing can be done using native Apple and Android accessories by following the steps provided in Sections 2.4 and 2.6.

## Why GitHub Desktop

Team B consists of varied experience levels of development, with some not being familiar with the command line nature of GitHub. To allow experienced and novice users to have the ability to participate in the development efforts of the project it was decided to implement the use of GitHub Desktop. GitHub desktop is an open-source tool that provides a more user-friendly graphical user interface for the more novice user and developers.

# Development Code

## Code Repository

The code repository for this project is hosted on GitHub (<https://github.com/umgc/fall2022>) and is publicly accessible via internet browser and GitHub Desktop. GitHub also provides version control and collaboration tools. For a local development environment for this project, the use of GitHub Desktop is recommended for interfacing with the code repository.

## Cloning the Repository

To clone the repository via GitHub Desktop, select *Current Repository* *> Add > Clone repository*.

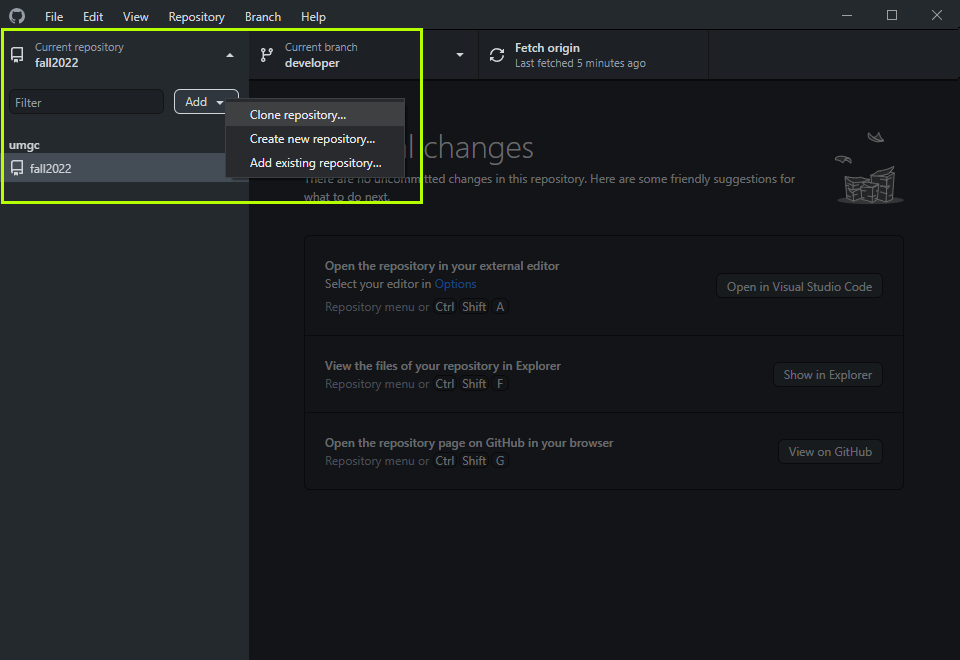


Figure 3.2.1 - Clone Repository Option

In the resulting *Clone a repository* modal, a list of repositories available to your GitHub account should display. Select this project’s repository and set the local path of the repository to the desired directory. Select *Clone* to clone the repository to your local machine.

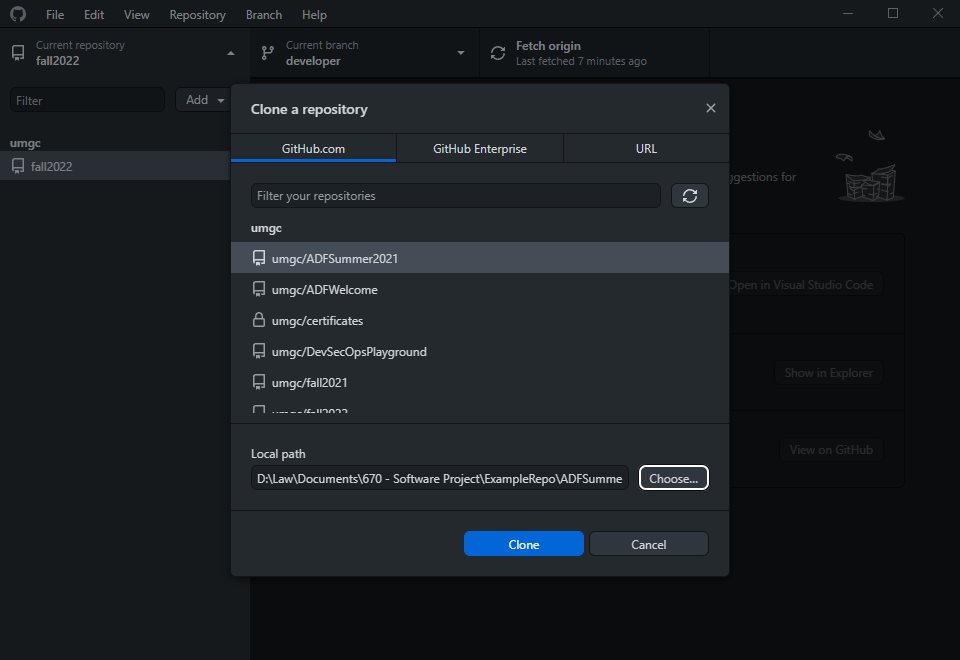


Figure 3.2.2 - Clone a Repository Modal

## Code Branches

The project’s code repository is organized into several branches.

* **release**: The release branch contains the current, production ready code that reflects the features implemented and defects addressed in the most recently completed iteration.
* **developer**: The developer branch contains the baseline code for the current development iteration. All features and bugs addressed in the current iteration are to be based on and merged into this branch. At the end of an iteration, the developer branch is to be merged into the release branch.
* **feature/[branch name]**: All features are to be implemented in a feature branch that is located within the feature directory. Upon successful implementation of the feature, a pull request into the developer branch is to be created. Upon approval of the pull request, the feature branch will be merged into the development branch.
* **bug/[branch name]**: All defects are to be addressed in a bug branch that is located within the bug directory. Upon successful correction of the defect, a pull request into the developer branch is to be created. Upon approval of the pull request, the bug branch will be merged into the development branch

# Application Architecture

## Architecture

The application is organized into several layers. The frontend consists of a UI layer that is composed of flutter widgets. The backend contains as a both a business layer and data layer, performing business logic and reading from and writing to the database. The business layer also interfaces with 3rd party services.

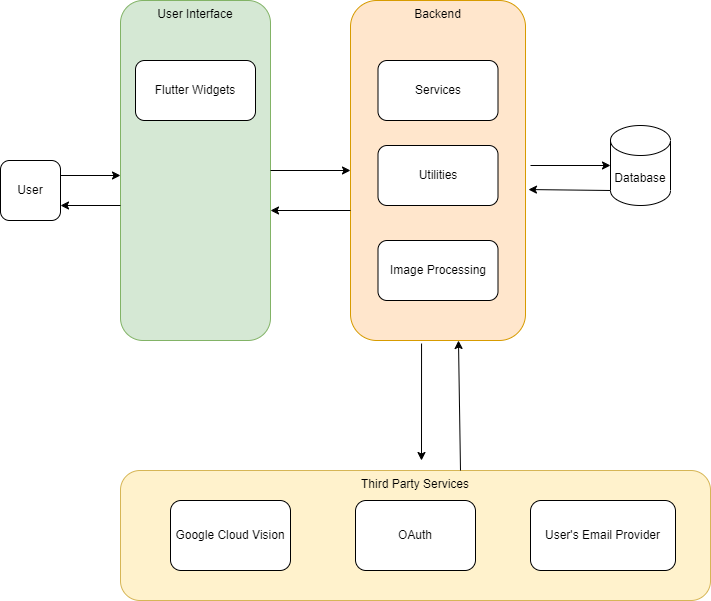


Figure 4.1.1 - Application Architecture

## Design Pattern

The application follows a model-view-controller (MVC) design pattern. Data models representing the data objects of the system comprise the model component. Flutter UI widgets represent the view component, displaying data to and taking input from the user. The backend services that perform business logic represent the controller component. Included in the services that comprise the controller are the utility and image processing logic.

# Code Structure

The code of MailSpeak is built upon the pre-existing architecture of the 2022 Summer Cohort. The application is written in the Dart programming language, using Flutter to enable cross platform development. The primary components being developed are located under the **lib** folder with the unit tests in the **test** folder. The newly implemented functionality will follow the architecture previously implemented unless change is required. The basic folder structure can be broken out as follows

* **Email Processing**: Handles connection to IMAP and processing of returned mail items.
* **Exceptions**: Custom exceptions to be handled by the system.
* **Generated**: Auto-generated folder to register plugins such as the camera, flutter storage, and image picker.
* **Image Processing**: Responsible for converting images into mail components.
* **Models**: Models used across service and UI layer.
* **Services**: This is a folder added by the Fall 2022 Cohort to better encapsulate the new functionality such as mail fetching, mail storage, cache service, analytics, chat bot, sqlite database, etc.
* **UI**: User interface components. Each element the user will interact with is defined here. Gesture and Voice driven functionality is enhanced by the tags provided to the User Interface components.
* **Utility**: Folder containing helpers and extension methods to reduce complexity across other areas of the application.
* **Test**: Folder containing all unit tests for each individual component of the system.

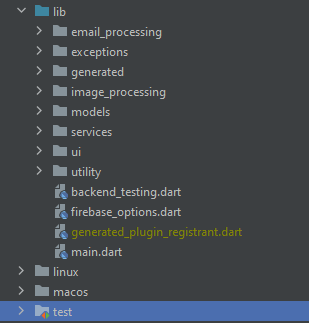


Figure 4.2.1 - Project Folder Structure

# Development

## Programming Languages

Flutter is an open-source software development kit which enables smooth and easy cross-platform mobile app development. Flutter enables to build high quality natively compiled apps for iOS and Android quickly, without having to write the code for the two apps separately.

The Flutter SDK is based on the Dart programming language. Dart makes it possible to create cross-platform mobile applications efficiently.

## Flutter Plugins and Libraries

The Flutter plugins and libraries are published and available at [pub.dev](https://pub.dev). The following Flutter plugins and libraries and corresponding version are used to support the implementation of the MailSpeak mobile application.

json\_annotation: ^4.5.

is a flutter plugin that Defines the annotations used by json\_serializable to create code for JSON serialization and deserialization.

intl: ^0.17.0

Contains code to deal with internationalized/localized messages, date and number formatting and parsing, bi-directional text, and other internationalization issues.

path\_provider: ^2.0.11

Flutter plugin for getting commonly used locations on host platform file systems, such as the temp and app data directories.

path\_provider\_android: ^2.0.11

Android implementation of the path\_provider plugin.

global\_configuration: ^2.0.0-nullsafety.0

A flutter package for managing different configurations and making them available everythere inside the app.

speech\_to\_text: ^5.6.0

A Flutter plugin that exposes device specific speech to text recognition capability.

flutter\_tts: ^3.5.0

A flutter plugin for Text to Speech. This plugin is supported on iOS, Android, Web, & macOS

xml: ^6.1.0

A lightweight library for parsing, traversing, querying, transforming and building XML documents.

http: ^0.13.4

A composable, multi-platform, Future-based Application Programming Interface (API) for HTTP requests.

sqflite: ^2.1.0

A plugin to support SQLite including CRUD operations and connections

sqlite3\_flutter\_libs: ^0.5.10

A plugin to allow for native sqlite3 libraries

sqlite3\_flutter\_libs: ^0.5.10

A plugin to allow for native sqlite3 libraries

google\_sign\_in: ^5.2.1

Flutter plugin for Google Sign-In, a secure authentication system for signing in with a Google account on Android and iOS.

flutter\_appauth: ^4.2.0

This plugin provides an abstraction around the Android and iOS AppAuth SDKs so it can be used to communicate with OAuth 2.0 and OpenID Connect providers

flutter\_secure\_storage: ^6.0.0

Flutter Secure Storage provides API to store data in secure storage. Keychain is used in iOS, KeyStore based solution is used in Android.

cupertino\_icons: ^1.0.2

Default icons asset for Cupertino widgets based on Apple styled icons

google\_mlkit\_barcode\_scanning: ^0.3.0

A Flutter plugin to use Google's ML Kit Barcode Scanning to read data encoded using most standard barcode formats.

image\_picker: ^0.8.5+3

Flutter plugin for selecting images from the Android and iOS image library, and taking new pictures with the camera.

googleapis\_auth: ^1.3.1

Obtain Access credentials for Google services using OAuth 2.0

googleapis: ^9.1.0

Auto-generated client libraries for accessing Google APIs described through the API discovery service

json\_serializable: ^6.2.0

Automatically generate code for converting to and from JSON by annotating Dart classes.

flutter\_svg: ^1.1.0

An SVG rendering and widget library for Flutter, which allows painting and displaying Scalable Vector Graphics 1.1 files.

toggle\_switch: ^2.0.1

Toggle Switch - A simple toggle switch widget. It can be fully customized with desired icons, width, colors, text, corner radius etc. It also maintains selection state.

flutter\_datetime\_picker: ^1.5.1

A date time picker for flutter, you can choose date / time / date&time in English Dutch and Chinese, and you can also custom your own picker content

camera: ^0.9.8+1

A Flutter plugin for controlling the camera. Supports previewing the camera feed, capturing images and video, and streaming image buffers to Dart.

url\_launcher: ^6.1.4

Flutter plugin for launching a URL. Supports web, phone, SMS, and email schemes.

enough\_mail: ^2.1.

IMAP, POP3 and SMTP for email developers. Choose between a low level and a high level API for mailing. Parse and generate MIME messages. Discover email settings.

loader\_overlay: ^2.0.7

A simple package to simplify screen management. When loading any async task, this package prevent the user from interacting with the screen until the async task finishes.

permission\_handler: ^10.0.0

Permission plugin for Flutter. This plugin provides a cross-platform (iOS, Android) API to request and check permissions.

image\_gallery\_saver: ^1.7.1

A new flutter plugin project for save image to gallery, iOS need to add the following keys to your Info.plist file.

flutter\_chat\_ui: ^1.6.4

Actively maintained, community-driven chat UI implementation with an optional Firebase Backend-as-a-Service (BaaS).

uuid: ^3.0.6

Request for Comments (RFC)4122 (v1, v4, v5) universally unique identifier (UUID) Generator and Parser for all Dart platforms (Web, Virtual Machine (VM), Flutter)

firebase\_analytics: ^9.3.5

Flutter plugin for Google Analytics for Firebase, an app measurement solution that provides insight on app usage and user engagement on Android and iOS.

firebase\_core: ^1.22.0

Flutter plugin for Firebase Core, enabling connecting to multiple Firebase apps.

get\_it: ^7.2.0

Simple direct Service Locator that allows to decouple the interface from a concrete implementation and to access the concrete implementation from everywhere in your App"

syncfusion\_flutter\_datepicker: ^20.2.50

The Flutter Date Range Picker widget allows users to easily select dates or a range of dates. It has four built-in views that allow quick navigation to the desired date.

sqflite: ^2.1.0

Flutter plugin for SQLite, a self-contained, high-reliability, embedded, SQL database engine.

sqlite3\_flutter\_libs: ^0.5.10

Flutter plugin to include native sqlite3 libraries with your app.

flutter\_local\_notifications: ^11.0.1

A cross platform plugin for displaying and scheduling local notifications for Flutter applications with the ability to customise for each platform.

workmanager: ^0.5.0

Flutter Workmanager. This plugin allows to schedule background work on Android and iOS.

google\_sign\_in: ^5.2.1

Flutter plugin for Google Sign-In, a secure authentication system for signing in with a Google account on Android and iOS.

receive\_intent: ^0.2.2

Flutter plugin for passing Android Intents to the Flutter environment.

flutter\_typeahead: ^4.1.1

A highly customizable typeahead (autocomplete) text input field for Flutter

layout: ^1.0.3

Layout encourage consistency across platforms, environments, and screen sizes by using uniform elements and spacing.

firebase\_auth: ^3.11.2

Flutter plugin for Firebase Auth, enabling Android and iOS authentication using passwords, phone numbers and identity providers like Google, Facebook and Twitter.

flutter\_signin\_button: ^2.0.0

A Flutter plugin for iOS and Android for generating sign in buttons for different social media account.

extension\_google\_sign\_in\_as\_googleapis\_auth: ^2.0.7

A bridge package between google\_sign\_in and googleapis\_auth, to create Authenticated Clients from google\_sign\_in user credentials.

flutter\_html: ^3.0.0-alpha.6

A Flutter widget rendering static HTML and CSS as Flutter widgets.

## Implementing Google Cloud Vision API

**Why use Google Cloud Vision API?**

The Google Cloud Vision API is used for image processing from USPS Informed Delivery emails so we can treat individual mail pieces as searchable items in our system. Using Google Cloud Vision, the system can convert all text in an image to a digestible format where the system can determine sender, recipient, and content. The API also contains libraries to interpret logos into text representation.

**How does it work?**

In combination with the newly implemented database and cache service, the system can reduce the number of times this API is called by storing the results in text format in the database. This will allow much faster searching, fetching and displaying of a user’s data.

**How to setup Google Cloud Vision API?**

To enable the Google Cloud Vision API, a valid credentials.json must be presented in the project at path *assets/credentials.json*. Currently the value is hardcoded, but when a proper license is purchased, this value should be populated via a GitHub secret using the CI/CD pipeline.

To get the credentials needed for cloud vision, enable the Cloud Vision API in the Google Cloud console (console.cloud.google.com), then navigate to Credentials -> Service Accounts and generate a key.

Table 6.3.1 - Google Cloud Vision Classes and Methods

| Class | Method (private methods denoted with \_ preface) | Description |
| --- | --- | --- |
| CloudVisionAPI | convertImageToText | Returns list of TextAnnotations after converting image using Cloud Vision API with type “TEXT\_DETECTION” |
| CloudVisionAPI | searchImageForText | Attempts to parse List of TextAnnotations for specific Address blocks and converts to them to a list of AddressObjects |
| CloudVisionAPI | searchImageForLogo | Searches for logos using Cloud Vision API “LOGO\_DETECTION” |
| CloudVisionAPI | \_parseBlocksForAddress2 | Goes through blocks using provided index and puts them into proper list |
| CloudVisionAPI | \_blockHasPostage |  |
| CloudVisionAPI | \_validateNameHasNoSpecialSymbols | Validates the name has no special symbols |
| CloudVisionAPI | \_validateCityStateZip | Validates city follows city, state, zip guidelines |
| CloudVisionAPI | \_findLineWithCityStateZip | Find line that contains city state zip within a block and returns the line index |
| CloudVisionAPI | \_validateAddress1 | Validates the address follows mail guidelines |
| CloudVisionAPI | \_findBlocksWithAddresses | Checks and determines which blocks contain zip and returns a list of potential blocks with addresses |
| CloudVisionAPI | \_checkForUnits | Checks for suite, apartment, etc unit specification |

## Implementing Digest Email Parsing/IMAP

**Why use IMAP?**

Using the IMAP provider from the flutter plugin *enough\_mail* will allow the application to retrieve emails from the user provided account. Once retrieved the system can parse these emails and perform the necessary functions to make them viewable and usable as individual mail items.

**How does it work?**

Upon successful login for the application, all USPS Informed Delivery emails received after the stored latest fetched date are retrieved using the IMAP provider. If any new emails are retrieved, the system will use the Google Cloud Vision API to parse any detected images and store the resulting text conversion to the database with the mail item ID. This storage allows fast searching of image text and for the system to treat individual mail items appropriately instead of being attached to an email. By displaying actual mail pieces instead of just the email, this allows the application to offer a greater benefit to the user and more closely resemble searching for your mail instead of just using capabilities already available in your email client.

The MailFetcher class is responsible for fetching the emails from the IMAP provider taking the specific parameters to search appropriate items using the supplied lastTimestamp and hardcoded subject and email address to ensure only relevant emails are returned via the public fetchMail function and performing the conversion of images using the Google Cloud Vision API. The CacheService class is responsible for initiating the MailFetcher retrieval and sending the results to the MailStorage class for database saving. The MailService class is responsible for parsing search parameters and finding relevant items in the database. The private \_processEmail function is responsible for splitting out image content and processing that to be stored as an individual mail item.

**How to setup IMAP?**

Configuration of the ImapClient is handled by the enough\_mail library and MailFetcher class. The client gets created by the \_login function in the MailFetcher class and will use the provided username and password to access the user’s email items.

Table 6.4.1 - Classes and Methods

| Class | Method (private methods denoted with \_ preface) | Description |
| --- | --- | --- |
| MailFetcher | fetchMail | Fetch all pieces of mail since the provided timestamp from usps informed delivery with Daily Digest subject |
| MailFetcher | \_processEmail | Process an individual email, converting it to a list of MailPieces |
| MailFetcher | \_getAttachments | Retrieve a list of the mail image “attachments” with accompanying metadata |
| MailFetcher | \_getHeader | Get particular header value from a MimeData part |
| MailFetcher | \_isContentType | Check if MimeData part is of a specified content type |
| MailFetcher | \_getEmails | Retrieve emails based on a start date, sender filter, and subject filter |
| MailFetcher | \_login | Log into the IMAP client for email retrieval |
| MailFetcher | \_logout | Logout of the IMAP client |
| MailFetcher | \_formatTargetDateForSearch | Format the DateTime object as the format expected by IMAP |
| MailFetcher | \_processMailImage | Process an individual mail image, converting it into a MailPiece |
| MailFetcher | \_getOcrScan | Perform OCR scan once on the mail image to get the results for further processing |
| MailService | fetchMail | Retrieves all mail from local cache that matches the search parameters |
| MailService | \_formatDateTimeForSearch | Formats date range for search |
| MailService | searchMailPieces | Returns all mail pieces that match the provided query |
| MailStorage | saveMailPiece | Persist a piece of mail to the database. Returns true if new mail item was saved |
| MailStorage | getMailPiece | Retrieves a mail piece by its ID |
| MailStorage | updateMailPiece | Updates a single mail piece that matches the provided id |
| MailStorage | deleteMailPiece | Deletes a single mail piece that matches provided id |
| MailStorage | deleteAllMailPieces | Deletes all mail pieces from database |
| MailStorage | searchMailPieces | Returns all mail pieces that match the provided query |
| CacheService | updateMail | Builds a default CacheService and uses it to fetch the latest mail |
| CacheService | fetchAndProcessLatestMail | Fetches mail since the last time received and stores/processes that mail. Updates notifications as necessary |
| CacheService | clearEverything | Calls clearAllCacheData on resources |
| CacheService | clearAllCacheData | Clears all cache data from mailstorage and mailnotifier |

## Implementing Chatbot

**Why create Chatbot?**

The Chatbot feature was designed as a simple communication interface that allows a user to view and execute available commands based on the current area of the application. The feature is loosely integrated into the application to allow for an easier replacement if something more robust like DialogFlow was to be added later. The decision not to include DialogFlow in this release is due to the short time-frame making that hard to keep in scope, and not wanting to add a dependency on another paid service.

**How does it work?**

The ChatBotService implements an abstract class called ChatBot. The public method available is to performChatFunction which will return an ApplicationFunction even if there was no matching result. The ApplicationFunction will then call the appropriate UI function on the area of the application the user is currently located. These functions are currently limited to navigation and page level functions such as populating inputs.

The dedicated service for the Chatbot makes it easy to unit test submitting an input and receiving the desired application function. The custom exception InvalidCommandException is thrown when the user’s command cannot be interpreted and the Chatbot UI will display the appropriate text response to the user. All classes related to the Chatbot feature are written in the Dart programming language.

The Chatbot feature allows the user to open the chat window from any page of the application and enter commands or enter ‘Help’ to view a list of available commands for the current page. Available commands are stored in the ChatBotService in the ChatFunctions variable as seen in Figure 6.5.1.

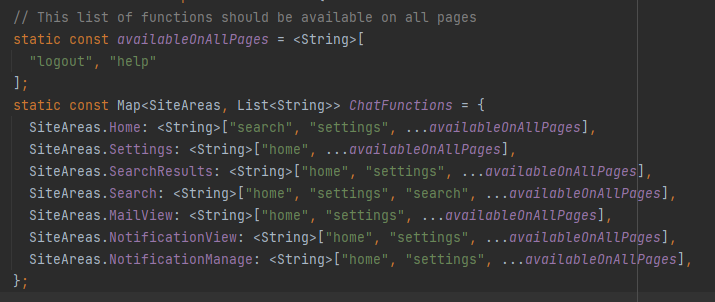


Figure 6.5.1 - ChatFunction

Table 6.5.1 - Chatbot Classes and Methods

| Class | Method (private methods denoted with \_ preface) | Description |
| --- | --- | --- |
| ChatBotService | performChatFunction | Perform user entered chat function. Returns an application to be consumed by the UI |
| ChatBotService | \_implementCommand | Converts String command to ApplicationFunction for implementation |
| ChatWidget | \_handleSendPressed | Takes user input and passes it through the ChatBotService. This function is aware of all possible actions performable by the ChatBot and will make UI/Navigation changes to reflect the entered command |

## Implementing Mail Search

**Why implement Mail Search?**

The Mail Search feature allows users to search for mail pieces by sender, text on the mail piece, and the date range within which the mail piece appeared in the user’s informed delivery emails. Through the Mail Search feature, users may view a list of received mail that matches the provided search criteria. This provides users with an efficient means of locating mail pieces which have appeared in their Informed Delivery emails.

**How does it work?**

The MailService class contains functionality to retrieve mail pieces stored in the local database based on a set of filter criteria. The public method, fetchMail, takes a MailSearchParameters object, which contains fields for keyword (or sender and mail body keyword if performing an advanced search), start date, and end date, and returns a List of all mail pieces that match the criteria provided. From the SearchWidget, the user may input a keyword (or sender and mail body keyword) and a date range by which received mail pieces may be filtered. Upon selecting to search mail pieces by the provided criteria, a MailSearchParameters instance is populated with the provided keyword (or sender keyword and mail body keyword), start date, and end date and passed to the fetchMail method. The mail pieces that are returned are then displayed in the MailViewWidget and may be selected by the user to view as individual mail pieces.

Table 6.6.1 - Mail Search Classes and Methods

| Class | Method (private methods denoted with \_ preface) | Description |
| --- | --- | --- |
| **MailService** | fetchMail | Takes a MailSearchParameters instance and returns a list of MailPiece objects that match the given criteria from the local cache. |
| **MailService** | \_formatDateTimeForSearch | Sets the start and end date to values which ensure that all mail pieces received on days within the given date range are returned from the searchMailPieces method. |
| **MailService** | searchMailPieces | Queries the local database for mail pieces that match the given criteria and returns them as a list of MailPiece objects. |
| **SearchWidgetState** | applyFilters | Sets the values of the \_keyword, \_senderKeyword, \_mailBodyKeyword, \_startDate, and \_endDate variables to the values provided in the search filter by the user. |

## Implementation of Accessibility Features

**Why remove old screen reader/voice commands?**

Users that require voice/screen reading capabilities will be using the device integrated features, so to better accommodate those users we decided to integrate with that ability instead of duplicating the effort. The previous implementation would cause duplicate readings of the screen and processing of voice commands to occur, so removing the previously implemented capability is not only less work on the development team, but more in line with how accessibility-impaired users would actually interact with the application.

**How does it work?**

By enabling ‘TalkBack’ on Android or ‘VoiceOver’ on iOS, the device will be ready to read and interpret voice commands for the application. Installing the “Voice Access” app, recommended by Android, or enabling “Voice Control” on iOS will allow the user to input voice commands to interact with the application. For development, we have provided the necessary attributes to integrate with these built-in features so the device integrated tools are aware of each element in the application.

Most widgets provided by Flutter already have default semantics and are well-recognized by the OS accessibility tools, those widgets have semanticsLabel and/or excludeSemantics properties that can be modified. However, some elements don’t have semantics-related properties and require adjustments for better integration with accessibility flow that uses a semantics tree. To include or exclude a widget from the semantics tree the Semantics widget is used.

The Semantics wrapper allows annotating widgets and passing callback functions. Properties used within the Semantics widget include:

* label – provides a textual description of the widget, which is used by screen readers and voice commands
* excludeSemantics – removes semantics from the semantic tree, which is used for elements that are not important for accessibility purposes or replaces default semantics annotation with custom.
* button, link, textField, etc. – a type of node that is introduced to the semantics tree, which is used by screen readers
* onTap() – a function that will be called when the “Tap” voice command is used
* sortKey – determines the accessibility traversal order, which is used by screen readers

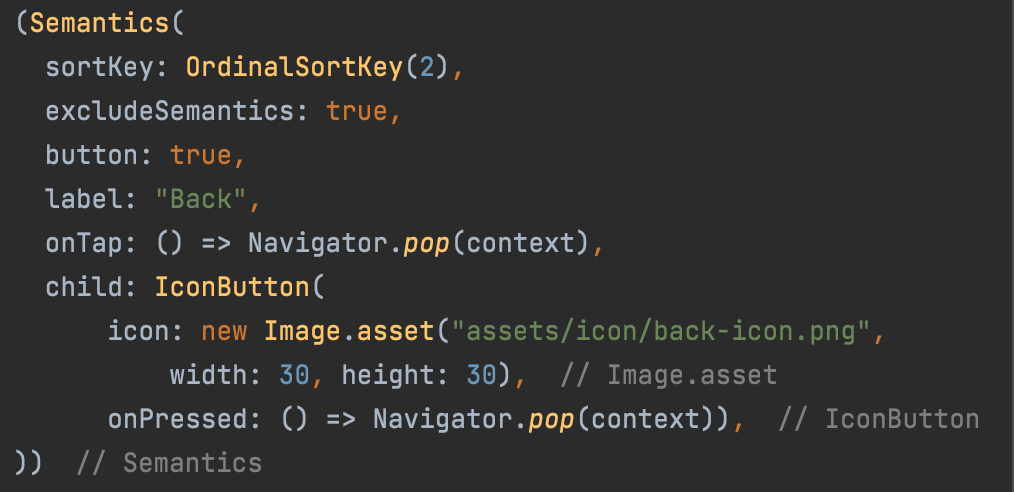


Figure 6.7.1 - Use of label, excludeSemantics, button, sortKey and onTap() Properties

* hint – additional information about the widget, which is used by screen readers

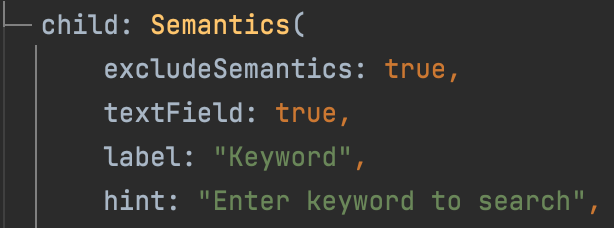


Figure 6.7.2 - Use of hint Property

* explicitChildNodes – adds semantics information from child to parent, which now can be accessed by screen readers and voice commands

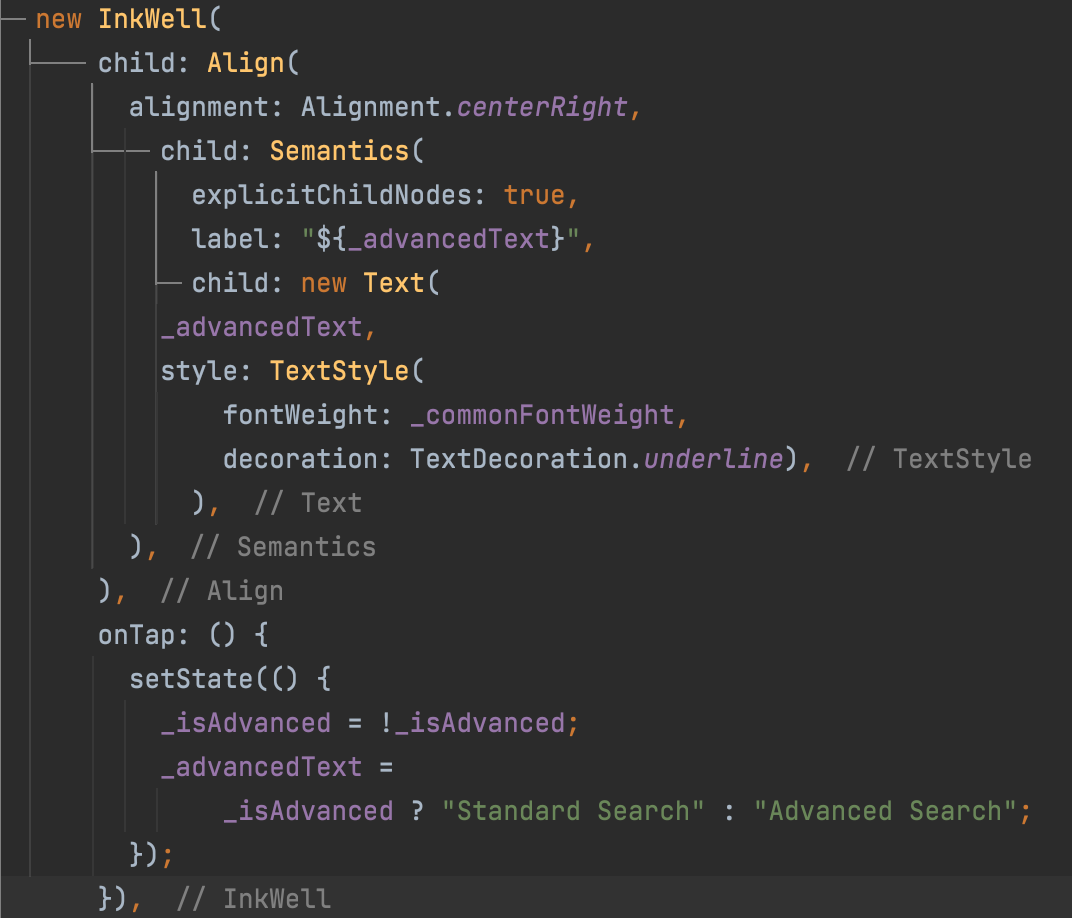


Figure 6.7.3 - Use of explicitChildNodes Property

MergeSemantics wrapper combines semantics within a subtree to be used as one node and all static data is read one by one in one tap.

## Implementation of Gmail Sign In/Reading Gmail’s Emails

**Why Implement Gmail Sign In?**

As there are multiple email providers from Apple to Gmail, we want to support the more popular email service providers. As we are limited with the number of developers with access to an Apple device and development environment, Gmail was selected as the sign in option to include in the application. Google has well documented Gmail API to facilitate retrieval from their email servers to our application.

**How does it work?**

The Gmail API requires an OAuth credential system due to access of a private user’s emails. A few steps are needed for Google to properly register and recognize the application. SHA1 fingerprint generated by the local keystore as well as Gmail API must be enabled Google Developer’s Console. A major consideration when dealing with teams is the keystore must be shared otherwise it will require uninstall/installing SHA1 fingerprints with Google’s backend. The majority of the google sign in process is handled by the google\_sign\_in Flutter package.

The GoogleSignIn class provided by the google\_sign\_in package requires scope to be defined before attempting to sign in. This must be registered during the upload of the SHA1 fingerprint. All Gmail scope codes can be found at <https://developers.google.com/gmail/api/auth/scopes>. For the IDMA and to avoid overreaching privacy concerns, read only access is requested from the user. Upon initially signing in, users are given a Google consent dialog to confirm (or deny) read-only access to their emails.

Once Google Sign in is complete and user successfully authenticates and authorizes IDMA to access emails, GoogleSignIn returns an authenticated client. This is wrapped around the GmailAPI to provide authorized read access to the user’s emails.

The GmailAPIService provides this wrapper and is used primarily to only fetch mail form Gmail’s servers. No mutation functionality is provided as we only requested read-only permission. Emails from Google are wrapped around Google’s own classes; they are named similarly to the MIME standard and enough\_mail’s classes. GmailAPIService’s primary goal is to reshape the mail object from Google into enough\_mail’s mail object. This is done by requesting Google to return the raw email data and utilizing enough\_mail’s parse function from bytes to recreate it; this object will be recognized by enough\_mail. Finally, this gets passed into the fetching/processing email pipeline (section 6.5) and mail is processed as normally.

Table 6.8.1 - Google Sign In Classes and Methods

|  |  |  |
| --- | --- | --- |
| Class | Method/Fields | Description |
| GmailAPIService | fetchMail | Fetch all pieces from given query. Query must match with Google’s search parameters |
| GmailAPIService | get api | Provides an entry point to access the gmail api |
| UserAuthService | get signInGoogleEmail | Attempts to login with Google |
| UserAuthService | get isSignedIntoGoogle | Returns if user is signed into Google |
| UserAuthService | get googleUserId | Gets the user id (email address) user used to sign into Google |
| UserAuthService | get googleClient | After user is signed into Google, this returns the authenticated client |

# Testing

Using a combination of the Flutter Test tools and Mockito, Unit Tests are very simple to create and execute in isolation, as a whole, or as part of the automated CI/CD pipeline.

## Create a Unit Test

Unit testing will be performed with each build in the CI/CD pipeline, whether it be from merging code changes or submitting a pull request. The developers will create these tests during the development process to cover the individual unit they are working on, as well as update any existing tests that may have been affected from their code change. These tests will then be automatically detected and ran by the CI/CD pipeline.

To create a unit test, right click the area in the test folder where you would like the test to be placed (location should be relative to where the item is in the lib folder), then select ‘New->Dart File’. Write out the basic template as shown in Figure 7.1 A default test case is shown where you can create the proper test based on whatever logic needs to be tested.

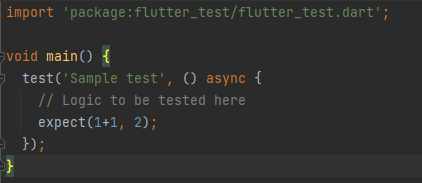


Figure 7.1.1 - Sample Unit Test

Mockito can also be used to stub out any objects used that are not part of the test. This allows the developer to properly test their individual unit without worrying about the other components.

## Running Unit Tests

To run unit tests locally in Android Studio, in navbar select the Run/Debug Configuration drop down at the top and choose one of the test options as shown in Figure 7.2.1.

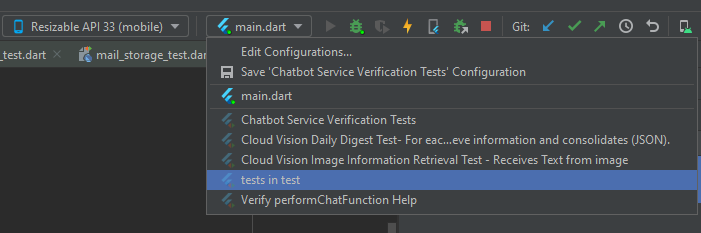


Figure 7.2.1 - Running Unit Tests from Navbar

Running tests from the project tree right click on the test folder and select run ‘tests in test’(Figure 7.2.2) or select a specific test inside the folder.

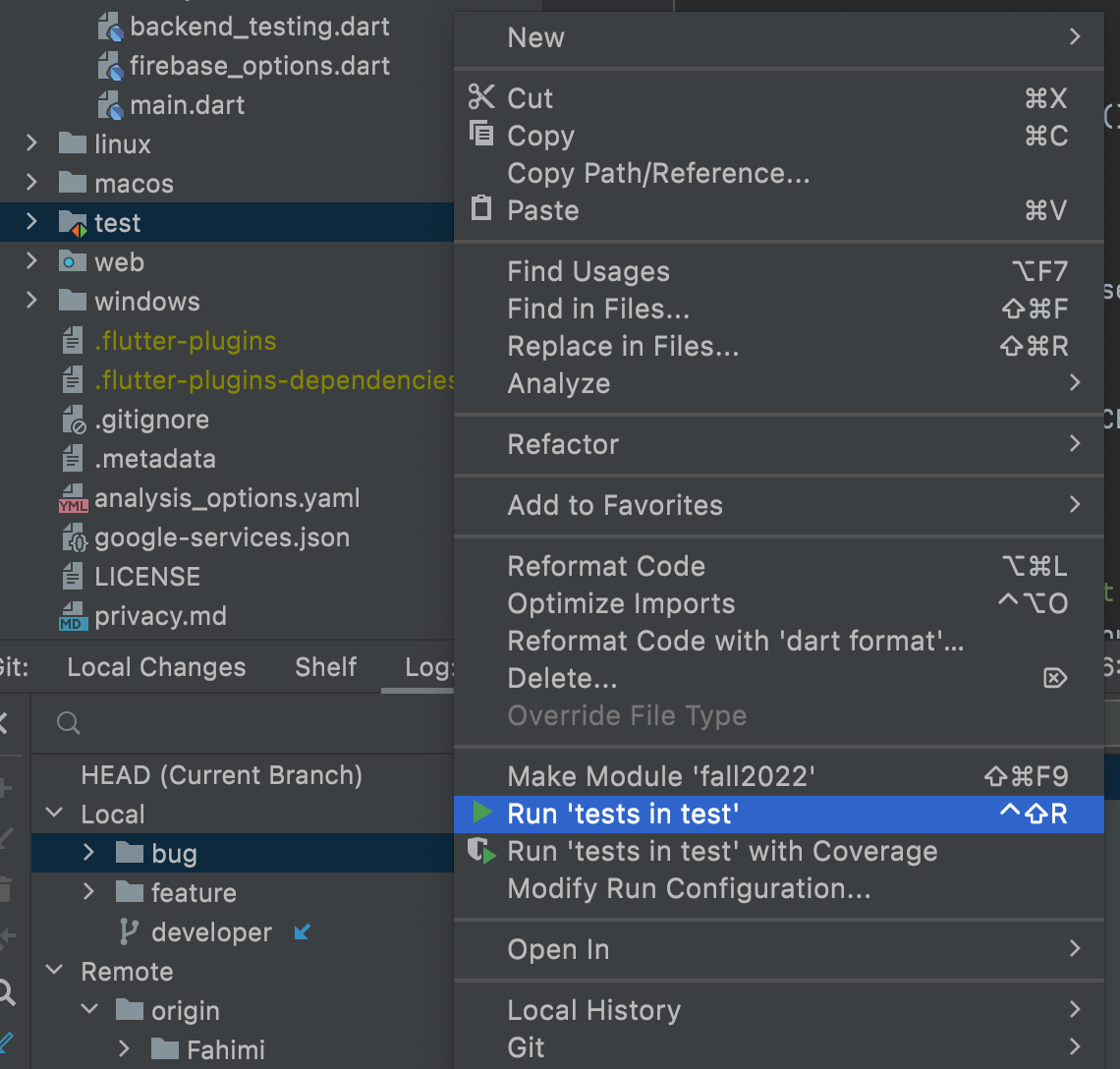


Figure 7.2.2 - Running Unit Tests from Project Tree

Once selected, run the project as you normally would and the test results will be displayed at the bottom along with ignored, failed, and passed descriptors. Tests can also be run individually by right clicking inside the test code and selecting Run ‘<test description>’.

Alternatively, tests will also be run as part of the GitHub actions in the CI/CD pipeline. This task executes the command ‘flutter test’ and fails the build if the tests return any failures. This command can also be run locally if desired.

Appendix A - Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Term | Meaning |
| AppAuth | Client SDK for native apps to authenticate and authorize end-users |
| API | Application Programming Interface |
| BaaS | Backend-as-a-Service |
| CI/CD | Continuous Integration/Continuous Deployment |
| CLI | Command Line Interface |
| CRUD | Create, Read, Update, Delete |
| CSS | Cascading Style Sheets |
| DevSecOps | Development, Security, & Operations |
| HTML | HyperText Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IMAP | Internet Message Access Protocol |
| iOS | iPhone Operating System |
| JSON | JavaScript Object Notation |
| MacOS | Macintosh Operating System |
| MIME | Multipurpose Internet Mail Extensions |
| ML | Machine Learning |
| MS | Microsoft |
| MSA | Mail Speak Application |
| MVC | Model View Controller |
| OAuth | Open Authorization |
| OCR | Object Character Recognition |
| OpenID | Open standard and decentralized authentication protocol |
| OS | Operating System |
| PG | Programmer’s Guide |
| PM | Project Manager |
| PMP | Project Management Plan |
| POP3 | Post Office Protocol |
| RFC | Request for Comments |
| SDK | Software Development Kit |
| SE | Software Engineer |
| SMS | Short Message Service |
| SMTP | Simple Mail Transfer Protocol |
| SQL | Structured Query Language |
| SRS | Software Requirements Specification |
| SWEN | Software Engineering |
| TBD | To Be Determined |
| TDD | Technical Design Document |
| UI | User Interface |
| UMGC | University of Maryland Global Campus |
| URL | Uniform Resource Locators |
| USPS | United States Postal Service |
| UUID | Universally Unique Identifier |
| VM | Virtual Machine |
| VPN | Virtual Private Network |