A picture containing logo

Description automatically generated

**Project Deployment/Ops Guide**

University of Maryland Global Campus

SWEN 670 – Team A

Fall Semester

Version 1.1

October 29, 2022

Document Control

Document Information

|  |  |
| --- | --- |
| © | Information |
| Document Id | Mail Speak Deployment/Ops Guide |
| Document Owner | UMGC SWEN 670 TEAM A |
| Issue Date | October 29, 2022 |
| Last Saved Date | November 5, 2022 |
| File Name | DepOps – Team A (v1.1).docx |

Document History

|  |  |  |
| --- | --- | --- |
| Version | Issue Date | Changes |
| 1.0 | 10/29/2022 | Initial Version |
| 1.1 | 11/5/2022 | Added images and captions |
|  |  |  |
|  |  |  |

# **Table of Contents**

[Table of Contents](#_Toc117956009)

[1. Introduction 1](#_Toc117956010)

[1.1. Purpose 1](#_Toc117956011)

[1.2. Intended Audience and Reading Suggestions 1](#_Toc117956012)

[1.3. Technical Project Stakeholders 1](#_Toc117956013)

[1.4. Project Documents 2](#_Toc117956014)

[1.5. References 2](#_Toc117956015)

[1.6. Definitions, Acronyms, and Abbreviations 3](#_Toc117956016)

[2. Development Team Workflow 4](#_Toc117956017)

[2.1. Description of Deployment 4](#_Toc117956018)

[2.2. Points of Contact 5](#_Toc117956019)

[2.3. Major Tasks 5](#_Toc117956020)

[2.4. Deployment Schedule 6](#_Toc117956021)

[3. Mobile Application 6](#_Toc117956022)

[3.1. Feature 6](#_Toc117956023)

[3.1.3. Internal feedback 7](#_Toc117956024)

[3.2. Packages 7](#_Toc117956025)

[3.3. Installing a package Dependency into an App 7](#_Toc117956026)

[3.4. 3.4 Plugins 8](#_Toc117956027)

[4. Software Installation 9](#_Toc117956028)

[4.1. Android Studio 9](#_Toc117956029)

[4.2. Flutter and Dart 11](#_Toc117956030)

[4.3. Flutter and Dart Plugins 14](#_Toc117956031)

[4.4. Android Emulator 16](#_Toc117956032)

[4.5. Test Your Development Environment 18](#_Toc117956033)

[5. Prepare the Mobile Application for Use 19](#_Toc117956034)

[5.1. Cloning GitHub Repository 19](#_Toc117956035)

[5.2. Run the Flutter Application 20](#_Toc117956036)

[6. Testing the Mobile Application 21](#_Toc117956037)

[6.1. Testing Objectives 21](#_Toc117956038)

[6.2. Unit Tests 22](#_Toc117956039)

[6.3. Testing Procedures 22](#_Toc117956040)

[7. Troubleshooting 23](#_Toc117956041)

[7.1. Emulator Freezing 23](#_Toc117956042)

[7.2. Emulator Not Responding 23](#_Toc117956043)

[7.3. Out of Memory Error 24](#_Toc117956044)

[8. Deploying the Mobile Application 25](#_Toc117956045)

[8.1. Google Play Store: 25](#_Toc117956046)

[8.2. Apple App Store (Note – Must have Xcode): 26](#_Toc117956047)

[8.3. General Production Deployment Guidelines: 27](#_Toc117956048)

[9. Appendices. 27](#_Toc117956049)

[9.1. Credits 27](#_Toc117956050)

[9.2. Credits to Previous Cohorts. 28](#_Toc117956051)

# Introduction

The deployment and operations guide for the MailSpeak App is the guiding document detailing the steps the team members followed while participating in development, testing and DevSecOps activities. This guide includes information on how team members worked together, leveraged capabilities and successfully satisfy project requirements as a group. Throughout this document, there may be some similarities to the work of the Summer 2022 cohort since the Fall 2022 cohort project was built on top of their framework.

## Purpose

The objective of this document is to enable the reader to understand the workflow taken to develop the app, to install and set the development environment, and finally to deploy the app on the device of their choice with little to no help from other person. The Deployment and Operations Guide also informs the audience about the Repository and how to use it, testing and troubleshooting steps in detail.

## Intended Audience and Reading Suggestions

The audience of the MailSpeak App Deployment and Operation Guide are:

* + The Stakeholders of fall2022 project
  + Students in the coming years as a reference
  + Developers who would continue to work on the app
  + Anyone who has access to the document who intends to know further about the app’s development background and wants to deploy and see the features in-person

## Technical Project Stakeholders

The project stakeholders for the MailSpeak (United States Postal Service (USPS) Informed Delivery Enhancement) App development are shown in the table below

|  |  |
| --- | --- |
| Name | Role |
| Dr. Mir Assadullah | Professor |
| Roy Gordon | Project Mentor |
| Robert Wilson | Project Mentor |
| Michael Conatser | (SE III)Overall project Manager + |
| Christopher Thorn | (SE III)Product Owner + |
| Jack Shira | (SE III) Senior Developer + |
| Imoh Noah Jr | (SE III) Senior Developer + |
| Ali Fahimi | (SE II) Senior Developer + |
| Harsh Gadani | (SE II) Developer + |
| Jahan Brahamabhatt | (SE II) Developer + |
| Natan Tafese | (SE I) Developer + |
| Fahed Masood | (SE I) Developer + |
| Scott Huber | (SE I) Developer + |
| Kuleni Digga | (SE I) Developer + |

Table 1 - Stakeholders

## Project Documents

The following documents are included in the project's software documentation package:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Document** | **Version** | **Date** |
| 1 | Project Management Plan (PMP) | 1.2 | 9-17-2022 |
| 2 | Software Requirements Specification (SRS) | 1.2 | 10-29-2022 |
| 3 | Technical Design Document (TDD) | 1.2 | 10-29-2022 |
| 4 | Software Test Plan (STP) | 1.1 | 10-29-2022 |
| 5 | Programmers Guide (PG) | 1.1 | 11-05-2022 |
| 6 | Deployment and Operations (DevOps) | 1.1 | 11-05-2022 |
| 7 | User Guide (UG) | 1.0 | 11-05-2022 |
| 8 | Test Report (TR) | 1.0 | 11-05-2022 |

Table 2 - Project Documents

## References

Eble, J. (2022, August 17). MemorEZ Deployment and Operations Guides (Runbooks). Adelphi; University of Maryland Global Campus. (Capstone project document from Spring 2022)

*Test drive*. Flutter. (n.d.). Retrieved October 22, 2022, from <https://docs.flutter.dev/get-started/test-drive?tab=androidstudio>

*Using packages*. Flutter. (n.d.). Retrieved October 22, 2022, from <https://docs.flutter.dev/development/packages-and-plugins/using-packages>

Google. (n.d.). *Create and set up your app - play console help*. Google. Retrieved October 23, 2022, from <https://support.google.com/googleplay/android-developer/answer/9859152?hl=en>

*Create and manage virtual devices:   android developers*. Android Developers. (n.d.). Retrieved October 23, 2022, from <https://developer.android.com/studio/run/managing-avds#:~:text=Run%20and%20stop%20an%20emulator%2C%20and%20clear%20data,-From%20the%20Virtual&text=To%20run%20an%20emulator%20that,Menu%20and%20select%20Wipe%20Data>.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| API - | Application Program Interface |
| BA - | Business Analyst |
| CE - | Code Editor |
| GCP - | Google Cloud Service Platform |
| HI - | Hardware Interface |
| IDE - | Integrated Development Environment |
| iOS - | iPhone Operating System |
| OS | Operating System |
| PM - | Project Manager |
| SDK - | Software Development Kit |
| SE - | Software Engineer |
| SI - | Software Interface |
| UI - | User Interface |
| UMGC- | University of Maryland Global Campus |
| UML - | Unified Modeling Language |
| VM - | Virtual Machine |

# Development Team Workflow

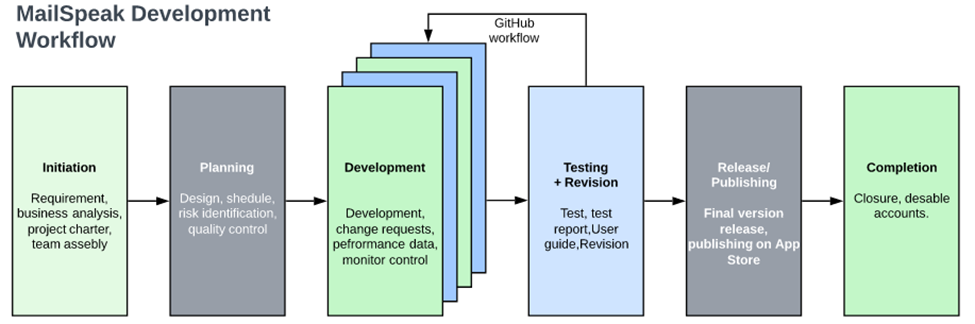


Figure 1 - Team Workflow

## Description of Deployment

The **MailSpeak** App deployment started using the GitHub repository by moving the summer project codebase to this semester and creating the fall2022 repository. All team members joined GitHub to be able to contribute. Since Mailspeak App is an enhancement to the previously developed software, USPS-informed delivery for the visually impaired, studying the working and not working features of the previous app was the beginning task.

Depending on the Project plan, SRS, and results from research on the previous work, the deployment process continued with opening task breakdowns on new features to be added to the App.

Each developer started their branch from the Master to work on different issues from GitHub. Branching allowed everyone to be able to work simultaneously without affecting each other. As a team member is done developing and testing the feature, to let the changes known and get a review, the next step is to submit a pull request (PR). Pull requests were reviewed by DevSecOps, who are responsible for version control, before being merged. As the work was reviewed and approved the next step was merging it into the Master.

****

Figure 2 - ci -cd options

**Text

Description automatically generated with low confidence**

Figure 3 - ci - cd options expanded android

**A picture containing text

Description automatically generated**

Figure 4 - ci - cd options expanded iOS

MailSpeak App is developed for use on the Android and iOS devices. For this reason, work was done on pipeline workflows for each operating system. The pipeline used is referred to as a Continuous Integration / Continuous Delivery (CI/CD), which let the deployment have continuous integration and continuous deployment. After the integration testing and user testing were performed, the MailSpeak App is deployed. The publishing of the app is done on Play Store and App Store to make it available to users of both operating systems.The details of deploying the App is discussed on section 8 of this document.

## Points of Contact

|  |  |
| --- | --- |
| **Point of Contact** | **Role** |
| Dr. Mir Assadullah | Professor |
| Roy Gordon | Project Mentor |
| Robert Wilson | Project Mentor |
| Michael Conatser | Overall Project Manager |
| Christopher Thorn | Product Owner |

Table 3 - Points of Contact

## Major Tasks

Major tasks in the MailSpeak App development :

* Project Initialization
* Get familiar with previous codebase
* Start new repository for the new project
* Project planning and documentation
* Requirement identification and documentation
* Setting environment, language, framework, emulators and plugins
* Researching on each requirement and requirement breakdown in to functions
* Development of each function into the codebase
* Test throughout the iteration
* Execution and revision
* Deployment of the App and completion.

## Deployment Schedule

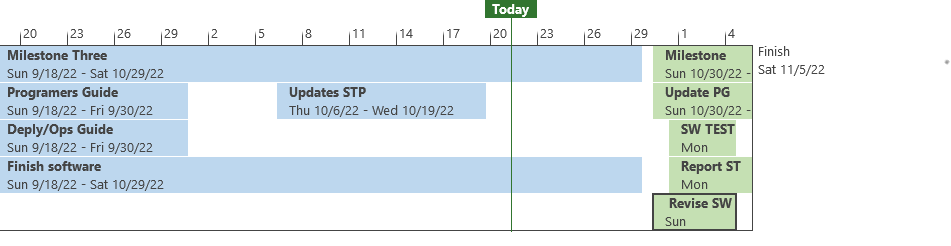
As per the Project Plan, the development of the application gets completed before the third milestone of the project, and revision followed by deployment of the final product takes place at the end of the fourth milestone on the 5th of November, 2022. 

Figure 5 - Project Schedule

# Mobile Application

## Feature

### **Mail View**

The feature lets the user open the resulting email from search, enabling the user to either email, call or text back to the sender or contact on command, enabling to visit websites of links or barcodes attached.

### **Notification**

The feature enables the user to set a notification alert for a specific sender and the user gets notification as mail from the specific sender arrives.

## Internal feedback

The feature provides links to terms and conditions on the USPS website and ensures user agreeance. It also provides real-time analytics concerning mail piece IDs and reports consumer behavior to USPS.

### Voice Assistant

The feature works with Google Assistant and Siri to achieve in-app voice control for using Mailspeak app.

## Packages

The package of the MailSpeak App is an assemblage of files and information about those files. This files, applications made by other developers for everyone’s use, save a lot of effort from developing every feature from scratch. The latest version of packages and dependencies used are in the pubspec.yaml section of document.

Text

Description automatically generated

Figure 6 - pubspec.yaml

## Installing a package Dependency into an App

To add a package, to the app:

1. Add Dependancies
   * Open the pubspec.yaml file located inside the app folder and add the specific package name under dependencies.
2. Install it
   * From the terminal: Run flutter pub get.  
     **OR**
   * From Android Studio: Click **Pub 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.

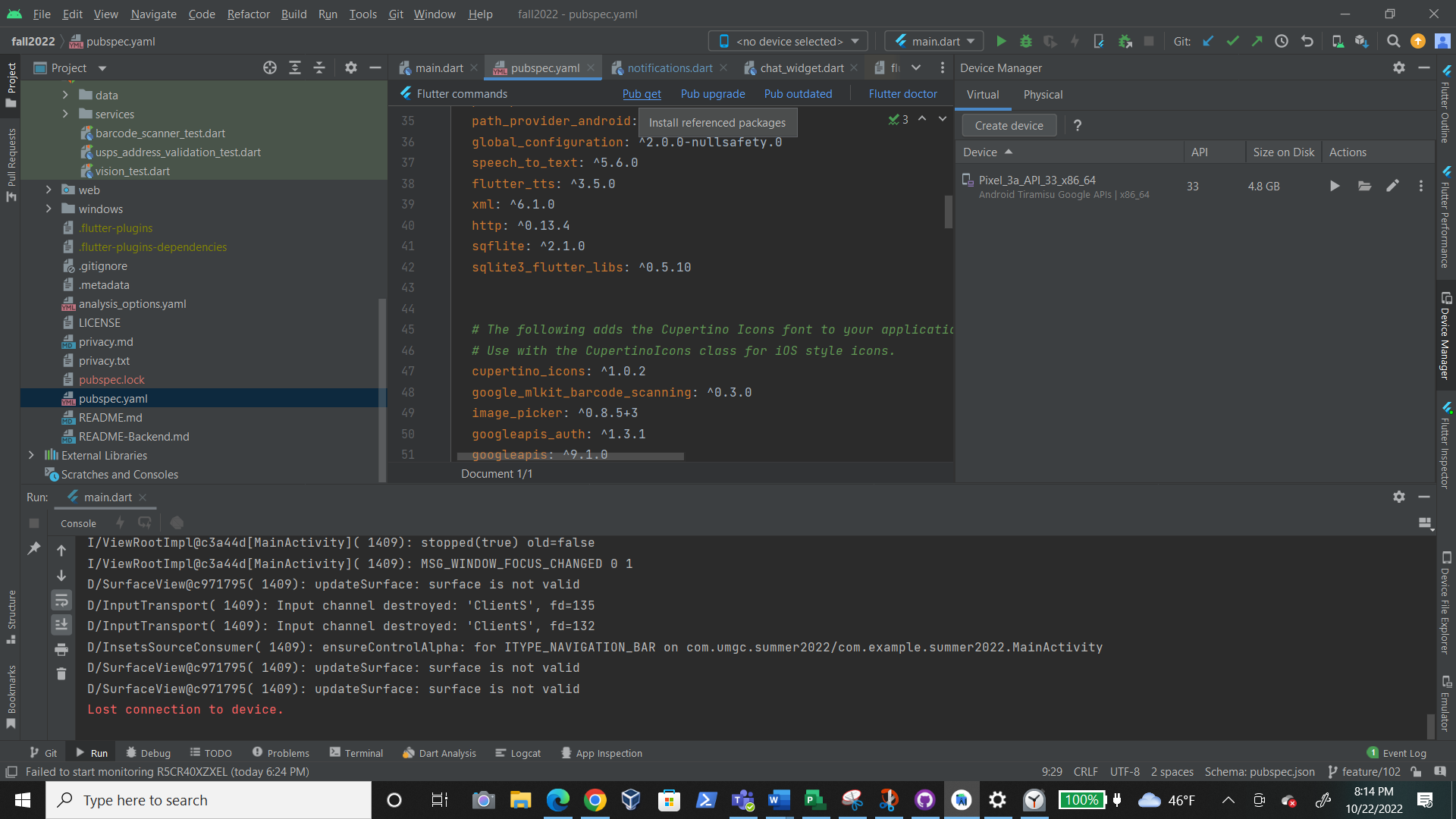


Figure 7 - Install Referenced Packages

1. Import it
   * Add a corresponding import statement in the Dart code.
2. Stop and restart the app, if necessary

If the package brings platform-specific code (Kotlin/Java for Android, Swift/Objective-C for iOS), that code must be built into your app. Hot reload and hot restart only update the Dart code, so a full restart of the app might be required to avoid errors like MissingPluginException when using the package.

## 3.4 Plugins

Installing Flutter and Dart plugins -Android Studio

1. Open Android Studio

2. Open plugin preferences (File > Settings > Plugins)

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

Figure 8 - Install Dart Plugin

3. Select “Marketplace”, search Flutter

plugin and click install

4. Click yes, when prompted to install Dart plugin.

5. Click restart when prompted.

# Software Installation

## Android Studio

Android Studio is an integrated development environment (IDE) for Android development and includes everything one needs to build android apps. It is an IDE specifically designed for developing Android applications. As Flutter is a framework designed for multiplatform application development, the Android Studio IDE is utilized for developing native Flutter applications for Android.

**Installation*:***

**Windows**

*Environment Specifications:* Windows 10 or later (64-bit), x86-64 based, 8 GB disk space, 8 GB Random Access Memory (RAM)

*Package:*

* EXE (preferred): [https://developer.android.com/studio#downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dwindows.exe](https://developer.android.com/studio" \l "downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dwindows.exe)
* ZIP: <https://developer.android.com/studio#downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dwindows.zip>

*Installation:*

1. If the exe file has been installed, double-click to launch

For a zip file, extract the zip file and copy the android-studio folder to the Program Files folder. Open the android-studio/bin folder and launch studio.exe or studio64.exe, based on machine

1. Follow the setup wizard and install the SDK dependencies

**macOS**

*Environment Specifications:* macOS Mojave or higher, 8 GB disk space, 8 GB RAM

*Package:*

* Mac 64-bit: [https://developer.android.com/studio#downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dmac.dmg](https://developer.android.com/studio" \l "downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dmac.dmg)
* Mac 64-bit ARM: [https://developer.android.com/studio#downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dmac\_arm.dmg](https://developer.android.com/studio" \l "downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dmac_arm.dmg)

*Installation:*

1. Launch the Apple specific *Disk iMaGe* (DMG) file
2. Drag and drop Android Studio into the Applications folder, open the Applications folder, and launch Android Studio
3. Follow the setup wizard and install the SDK dependencies

**Linux**

*Environment Specifications:* Linux (64-bit), x86-64 based, 8 GB disk space, 8 GB RAM

*Package:*

* Linux 64-bit: [https://developer.android.com/studio#downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dlinux.tar.gz](https://developer.android.com/studio" \l "downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dlinux.tar.gz)

*Installation:*

1. Extract tar.gz file and move contents to where application files are stored
2. Open a terminal, change to the android-studio/bin directory, and execute studio.sh. This will launch the setup wizard
3. Follow the setup wizard and install the SDK dependencies

**Chrome OS**

*Environment Specifications:* Chrome OS (64-bit), 4 GB disk space, 8 GB RAM, Linux for Chrome enabled

*Package:*

* Chrome: [https://developer.android.com/studio#downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dcros.deb](https://developer.android.com/studio" \l "downloads:~:text=android%2Dstudio%2D2021.2.1.15%2Dcros.deb)

*Installation:*

1. Open the Files app and locate the Debian (DEB) package that was downloaded in the Downloads folder under My files
2. Right-click the DEB package and select “Install with Linux (Beta)”
3. Follow the setup wizard and install the SDK dependencies

## Flutter and Dart

Ref: The official Flutter documentation: <https://docs.flutter.dev/get-started/install>

**Installation:**

**Windows**

*Environment Specifications:* Windows 10 or later (64-bit), x86-64 based, 1.64 GB disk space, Git for Windows 2.x and Windows PowerShell 5.0 or newer installed

1. Download installation bundle (as of July 2022): <https://storage.googleapis.com/flutter_infra_release/releases/stable/windows/flutter_windows_3.0.5-stable.zip>
2. Extract the zip file and move the contained *flutter* folder to the desired installation location

(Do not install Flutter to a path that contains special characters or spaces. Also do not install Flutter in a directory like C:\Program Files\ that requires elevated privileges)

1. To use Flutter commands in the Windows console, search for env in the Start search bar, and select “Edit environment variables for your account.” In “User variables,” check if a “Path” entry exists. If the path exists, append the path to *flutter\bin* using a semicolon separator. If the path doesn’t exist, create a new user variable called Path and append the path of *flutter\bin*. Close and reopen the console to check if the changes are in effect.
2. Run the *flutter doctor* command in the command line interface (CLI) to determine if any other dependencies need to be installed

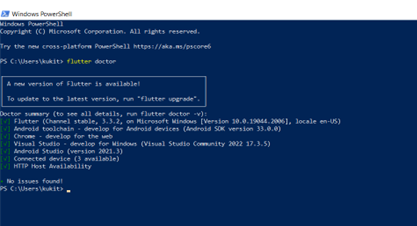


Figure 9 - Flutter Doctor

**macOS**

*Environment Specifications:* macOS, 2.8 Gigabyte (GB) disk space, Git installed

1. Download the bundle that is optimized for the system processor (Intel or Silicon, as of July 2022):

* Intel: <https://storage.googleapis.com/flutter_infra_release/releases/stable/macos/flutter_macos_3.0.5-stable.zip>
* Silicon: <https://storage.googleapis.com/flutter_infra_release/releases/stable/macos/flutter_macos_arm64_3.0.5-stable.zip>

1. Extract the file to the desired location
2. To add Flutter commands to the console, open the *rc* file for the system shell (Bash or Z). The file to edit for Bash is *.bash\_profile* or *.bash\_rc*, and Z shell is *.zshrc*. Then, export the path to the Flutter directory.

Export PATH=”$PATH:[PATH\_OF\_FLUTTER\_GIT\_DIRECTORY]/bin”

And replace [PATH\_OF\_FLUTTER\_GIT\_DIRECTORY] with the directory path. Run *source $HOME/.<rc file>* to refresh the console (replace *rc file* with the rc file). To determine if the installation works, run the *which flutter* command.

1. Run the *flutter doctor* command in the CLI to determine if any other dependencies need to be installed

**Linux**

*Environment Specifications:* Linux (64-bit), 600 MB disk space

1. Download the installation bundle (as of July 2022): <https://storage.googleapis.com/flutter_infra_release/releases/stable/linux/flutter_linux_3.0.5-stable.tar.xz>
2. Extract the file to the desired location
3. To add Flutter commands to the console, open the *.bash\_rc* file. Then, export the path to the Flutter directory.

Export PATH=”$PATH:[PATH\_OF\_FLUTTER\_GIT\_DIRECTORY]/bin”

And replace [PATH\_OF\_FLUTTER\_GIT\_DIRECTORY] with the directory path. Run *source $HOME/.<rc file>* to refresh the console (replace *rc file* with the rc file). To determine if the installation works, run the *which flutter* command.

1. Run the *flutter doctor* command in the CLI to determine if any other dependencies need to be installed

**Chrome OS**

*Environment Specifications:* Chrome OS (64-bit) with Linux Beta turned on, 600 MB disk space

1. Download the installation bundle (as of July 2022): <https://storage.googleapis.com/flutter_infra_release/releases/stable/linux/flutter_linux_3.0.5-stable.tar.xz>
2. Extract the file to the desired location
3. To add Flutter commands to the console, open the *.bash\_rc* file. Then, export the path to the Flutter directory.

Export PATH=”$PATH:[PATH\_OF\_FLUTTER\_GIT\_DIRECTORY]/bin”

And replace [PATH\_OF\_FLUTTER\_GIT\_DIRECTORY] with the directory path. Run *source $HOME/.<rc file>* to refresh the console (replace *rc file* with the rc file).To determine if the installation works, run the *which flutter* command.

1. Run the *flutter doctor* command in the CLI to determine if any other dependencies need to be installed

## Flutter and Dart Plugins

|  |  |
| --- | --- |
|  |  |
| camera: ^0.9.8+1 | Flutter plugin for controlling the camera. Supports previewing the camera feed, capturing images and video, and streaming image buffers to Dart. |
| cupertino\_icons: ^1.0.2 | Default icons asset for Cupertino widgets based on Apple styled icons |
| 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. |
| flutter\_svg: ^1.1.0 | An SVG rendering and widget library for Flutter, which allows painting and displaying Scalable Vector Graphics 1.1 files. |
| 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 |
| flutter\_chat\_ui: ^1.6.4 | Actively maintained, community-driven chat UI implementation with an optional Firebase BaaS. |
| flutter\_tts: ^3.5.0 | A flutter plugin for Text to Speech. This plugin is supported on iOS, Android, Web, & macOS |
| 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. |
| firebase\_core: ^1.22.0 | Flutter plugin for Firebase Core, enabling connecting to multiple Firebase apps. |
| 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. |
| 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" |
| global\_configuration: ^2.0.0-nullsafety.0 | A flutter package for managing different configurations and making them available everythere inside the app. |
| 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. |
| 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 |
| http: ^0.13.4 | A composable, multi-platform, Future-based API for HTTP requests. |
| 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. |
| 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. |
| 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 |
| json\_annotation: ^4.5 | Is a flutter plugin that Defines the annotations used by json\_serializable to create code for JSON serialization and deserialization |
| json\_serializable: ^6.2.0 | Automatically generate code for converting to and from JSON by annotating Dart classes. |
| loader\_overlay: ^2.0.7 | A simple package to simplify screen management. When loading any async task, this package prevents the user from interacting with the screen until the async task finishes. |
| 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. |
| permission\_handler: ^10.0.0 | Permission plugin for Flutter. This plugin provides a cross-platform (iOS, Android) API to request and check permissions. |
| recieve\_intent^0.2.2 | Used to pass intents in the native android application to the dart application. This is used by Google Assistant. |
| speech\_to\_text: ^5.6.0 | A Flutter plugin that exposes device specific speech to text recognition capability. |
| 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. |
| 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. |
| url\_launcher: ^6.1.4 | Flutter plugin for launching a uniform resource locator (URL). Supports web, phone, SMS, and email schemes. |
| uuid: ^3.0.6 | RFC4122 (v1, v4, v5) UUID Generator and Parser for all Dart platforms (Web, VM, Flutter) |
| xml: ^6.1.0 | A lightweight library for parsing, traversing, querying, transforming and building extensible markup language (XML) documents. |

Table 4 - Packages

## Android Emulator

An Android Emulator, included with Android Studio, enables testing the app on different android devices without the need of owning a physical android device. The emulator installs and starts the app faster. The computer specs to use the emulator in Android Studio smoothly are:16GB RAM, 64-bit Windows, macOS, or Linux operating system, and 16GB disk space.

The workflow for using the emulator:

1. Check your Specs.
2. Create a Virtual Device:

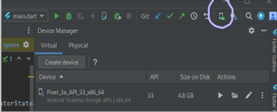


Figure 10 - Virtual Device

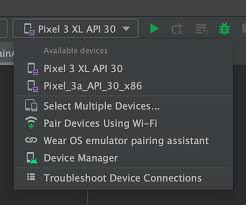


Figure 11 - Device Selection

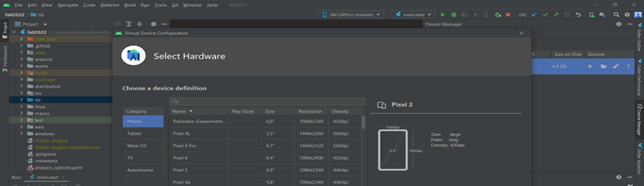


Figure 12 - Select Hardware

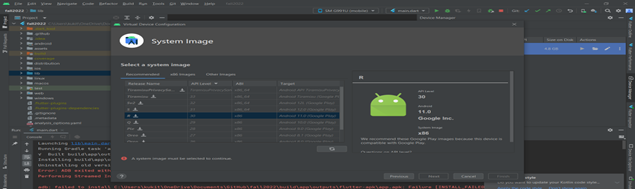


Figure 13 - System Images

1. Run the app on the emulator

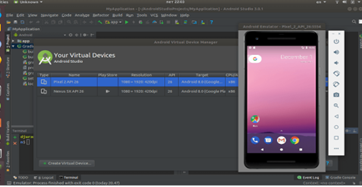


Figure 14 - Run Emulator

1. Navigate the emulator

Ref:Run App on Android Emulator: <https://developer.android.com/studio/run/emulator>

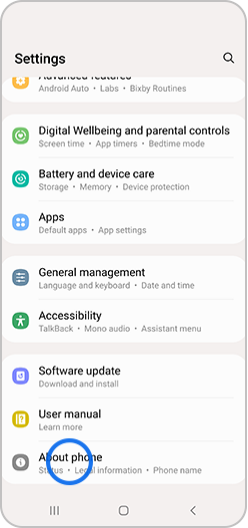
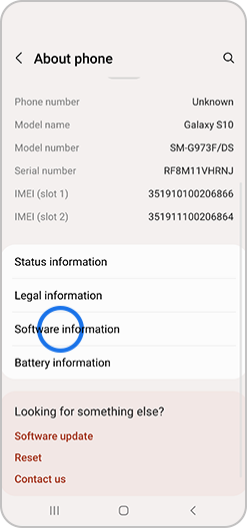
**Android Device (Physical)**

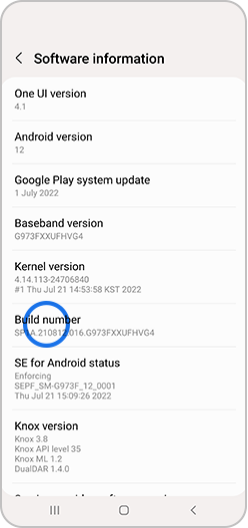
It's important to always test the app on a real device before releasing it to users. Using the physical Android device should not hinder the use of the Android emulator in the same project. The former ensures the product for release, while the latter tests the app's functionality on different versions of Android platform.

Before starting Debuging:

1 Decide the connection type, Wi-Fi or universal serial bus (USB), to connect with the device

2 On the device, open the **Settings app**, select **Developer options**, and then enable **USB**  **debugging** (if applicable)

Shape Shape Shape

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

Description automatically generated

Figure 15 - Developer Options

3 Set up your system to detect your device

## Test Your Development Environment

Step 1: Creat new flutter project

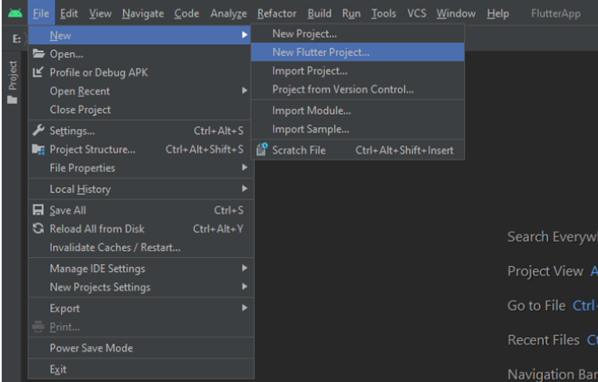


Figure 16 - Creating a new Flutter Project

Step 2: Select flutter application as project type

Step 3. Verify SDK path specifies the SDK location

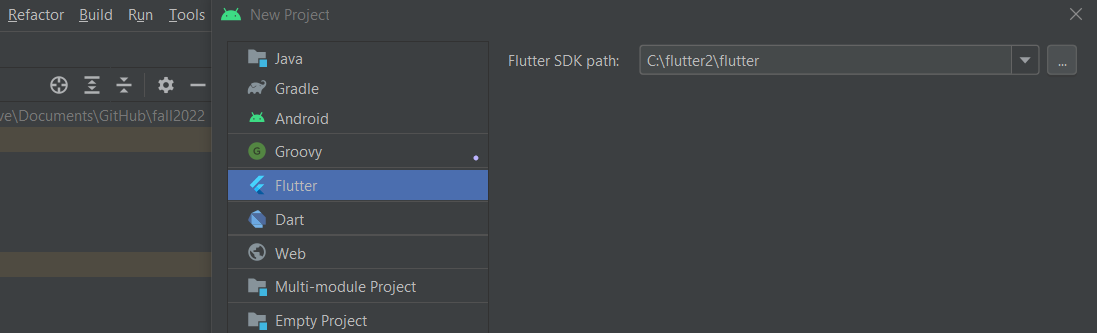


Figure 17 - Verify SDK Path

Step 4. Enter project name and click finish

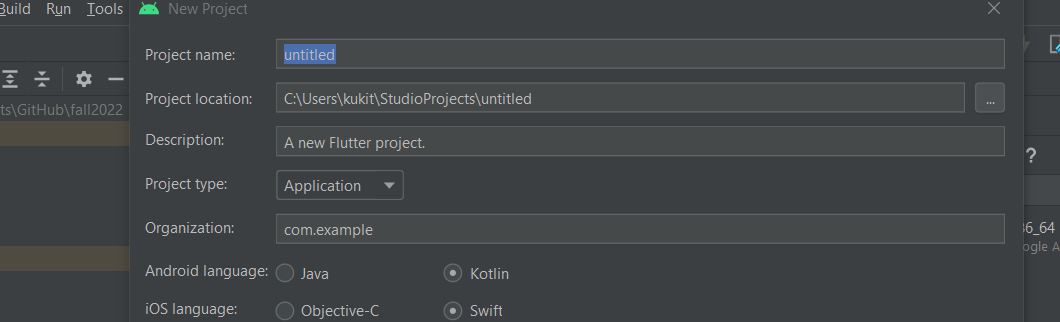


Figure 18 - Project Name



Figure 19 - Finish

# Prepare the Mobile Application for Use

## Cloning GitHub Repository

Cloning the repository from GitHub to a local computer helps minimize and fix conflict, add and remove files, as well as push commits to the overall product.In order to complete this, the following steps must be completed:

1. Navigate to <https://github.com/umgc/fall2022>
2. Click on the green “Code” button
3. Ensure the Hypertext Transfer Protocol Secure (HTTPS) tab is selected, then click copy icon to copy the GitHub url

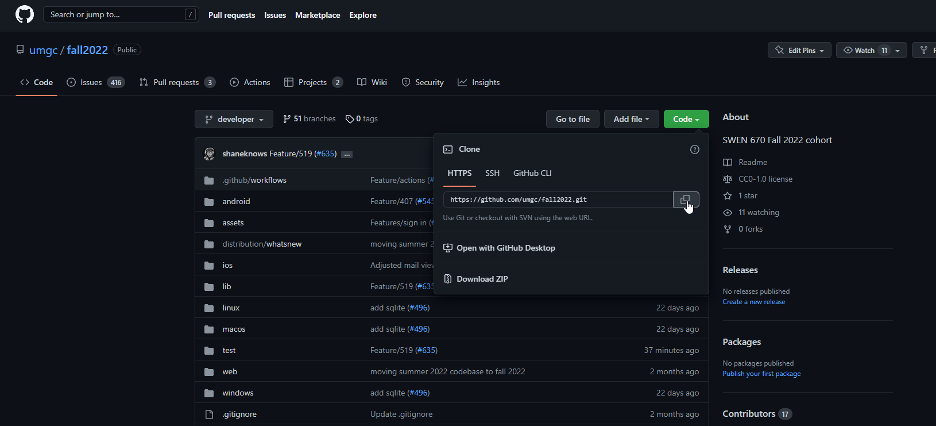


Figure 20 - Getting the Clone URI

## Run the Flutter Application

1. Open Android Studio

2. Click the menu icon and select “Get from Version control”

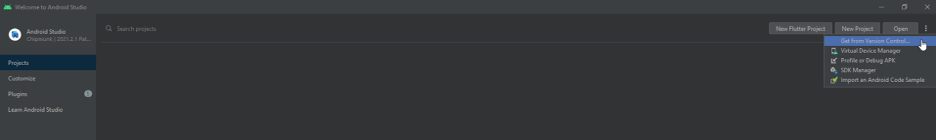


Figure 21 - Get From Version Control

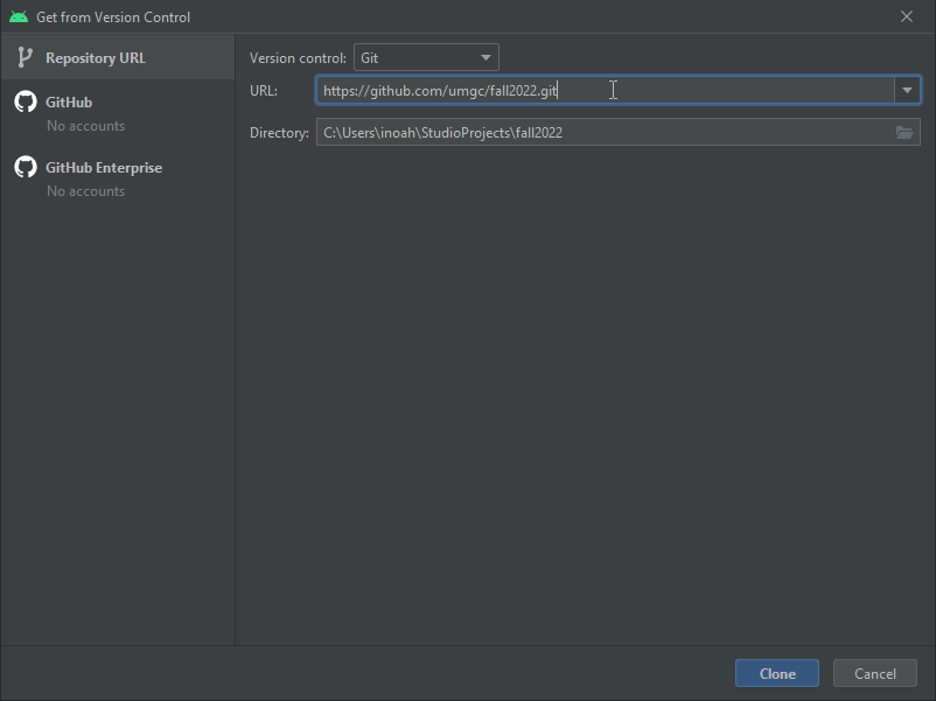
1. Paste the copied GitHub url and hit the clone button 

Figure 22 - Cloning the Project

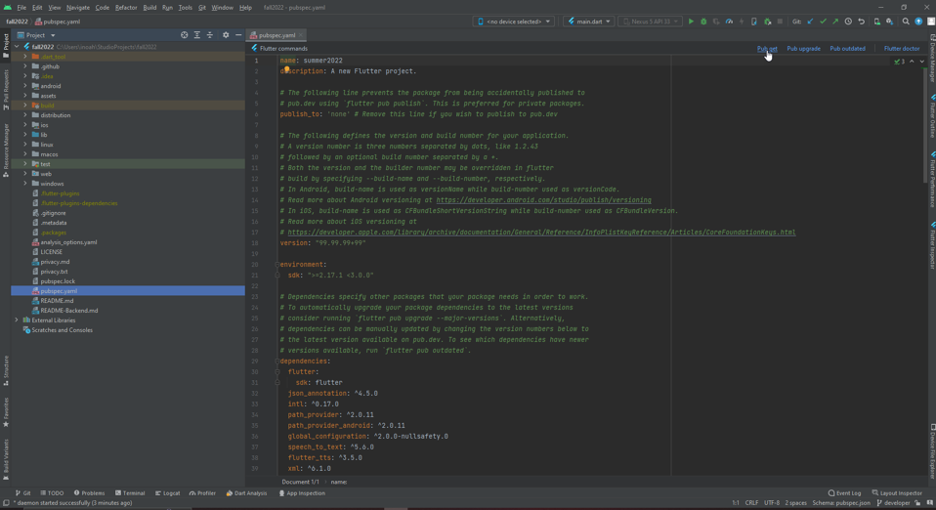
1. The application should open up in Android Studio
2. Open “pubspec.yaml” file and click on the Pub get link to install referenced packages. 

Figure 23 - pubspec.yaml

1. Select a previously configured emulator and click the play button to run the application.

# Testing the Mobile Application

## Testing Objectives

The main testing objectives are to

* Ensure reliable development.
* Create a functioning application.
* Prevent regression in behavior during development.
* Develop a application that passes all user requirements.

## Unit Tests

The application has an extensive suite of unit tests that are designed to alert developers if something they do breaks a functional piece of the application. Developers should add to the unit tests as they add functionality and should run these unit tests frequently. The unit tests are also used when a user creates a pull request. When this pipeline is run, all unit tests are also run, and if any unit tests fail, the user will not be allowed to check in the development branch.

## Testing Procedures

The application is tested frequently during development. The developer is expected to create unit tests as well as run the unit tests as they are developing. As stated in the section above, when a user creates a pull request, their changes are run through a build pipeline. If the unit tests pass the user is allowed to check into the development branch. This pipeline also will upload the resulting apk to the Android Play Store for android builds. This will go into the internal test track, and can be deployed to those users in the test track for testing when the Dev Ops engineer feels it is ready. This build then will be placed on test users phones and allow them to run their manual tests. Manual tests will then be run by testers, and any issues will be written up as bugs to be fixed by the team. This process will be repeated until the application is in such as state that it is able to be released to production. To test Google Assistant on the internal test track, or via closed testing, the tester must download the application from the play store and apply to test unreviewed app actions here: <https://groups.google.com/g/app-actions-development-program>.

# Troubleshooting

Determining why users are encountering issues and explaining how to resolve the problem is the goal of the Troubleshooting section. Common encounters during this development process were Emulator Freezing, Emulator not responding at all, and being Out of Memory.

## Emulator Freezing

If the Emulator is Freezing, Cold Boot( hard boot/cold start) will help the Emulator start fresh.

1. In Android Studio, open the icon in the top ribbon or click on “Tools > Device Manager”.
2. Select the emulator that has frozen and click the down arrow in the “Actions” column.
3. Select “Cold Boot Now”.

A screenshot of a computer

Description automatically generated

Figure 24 - Opening Device Manager

A screenshot of a computer

Description automatically generated

Figure 25 - Turning on Cold Boot

## Emulator Not Responding

If the emulator is not responding at all, it probably needs to update the emulator. Before that, Wipe the user Data of the ADV and do a hard boot.

1. In Android Studio, open the icon in the top ribbon or click on “Tools > Device Manager”.

2. Select the emulator that has frozen and click the down arrow in the “Actions” column.

1. Select “Wipe Data”, which would wipe user data of this ADV.

Note: Stop any running emulator before going through the step.

A screenshot of a computer

Description automatically generated

Figure 26 - Wiping Data from a Virtual Device

## Out of Memory Error

If out of memory error occurs, it clearly shows it needs to free up memory space. Start with closing all unnecessary applications running in the background to free up virtual memory. To increase the assigned Memory :

1. On Android Studio click on "File > Settings"

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

3. Set the memory you want to assign to your environment. It is recommended to have at least 2,048 MB assigned.

4. After assigning click "Apply > Ok"

Graphical user interface

Description automatically generated

Figure 27 - Fixing Memory Issues

# Deploying the Mobile Application

## Google Play Store:

Graphical user interface, text, application, email

Description automatically generated

Figure 28 - The Play Console

1. Create developer account

Start at: Google Play Console <https://play.google.com/console/u/0/signup>

1. Create Your App
   1. Open <https://play.google.com/console/u/0/signup>
   2. Select **All apps** > **Create app**.
   3. Select a language and add the name of this App to appear on Google Play. (It can be changed later)
   4. Specify application whether it is an app or a game.(It can be changed later)
   5. Specify whether the application is free or paid.
   6. Add an email address that Play Store users can use to contact the App point of contact about this application.
   7. In the "Declarations" section:
      1. Acknowledge the “Developer Program Policies” and “US export laws” declarations.
      2. Accept the Play App Signing Terms of Service.
   8. Select **Create app**.
2. Set up App content

After creating the App, start setting it up. The App’s dashboard will guide through all the most important steps to get the app available on Google Play. Continue providing details about the App’s content and entering information for the Google Play store listing.

1. Manage your App and App bundle

Google Play uses Android App Bundles <https://support.google.com/googleplay/android-developer/answer/9844279> to generate and deliver Android Package Kits (APKs) that are optimized for each device configuration, providing users with more efficient apps. Package names for app files are unique and permanent, so name them carefully. Package names can't be deleted or re-used in the future.

The release can have one of three statuses:

* **Draft:** APKs that have not been served to users yet
* **Active:** APKs currently being served to users
  + - **Archived:** APKs that were once active but are no longer being served to users

1. Set up store listing and Setting

It is now possible to move onto app release; this guides through pre-release management, testing, and promotion to build pre-release excitement and awareness. Your app's store listing is displayed on Google Play and includes details that help users learn more about the app. Your store listing is shared across tracks, including testing tracks

1. Created App release

For Detail reference: <https://support.google.com/googleplay/android-developer/answer/9859152?hl=en#zippy=%2Cfind-your-apk-files>

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

Description automatically generated

Figure 29 - Promoting a Release

## Apple App Store (Note – Must have XCode):

**Graphical user interface, text, application, email

Description automatically generated**

Figure 30 - Apple App Store Build Screen

1. Register for an Apple developer ID at https://developer.apple.com/programs/enroll/
2. Every iOS application must have a unique Bundle ID. You may set one through the App IDs page of the developer account.
3. Create a new application in App Store Connect

Graphical user interface, text, application

Description automatically generated*Figure 31 – Creating new App in App Store Connect*

1. Create an app bundle using the command “flutter build ipa”
   1. Optionally use –obfuscate to obfuscate Dart code
2. Upload bundle to App Store Connect
   1. Can optionally use the Apple Transport macOS app, run an upload command, manually drag and drop the bundle file, or use the built in Xcode publication features
3. Once uploaded to App Store Connect click “Validate App”
4. After the application has been uploaded to App Store Connect, it can be made available on TestFlight which allows for internal and external testing prior to release.

Graphical user interface, application, Teams

Description automatically generated  
*Figure 32 – Setting up Test Flight Group*

1. Once app store approval is received and testing has been completed you just need to click “Distribute App” in App Store Connect to deploy it to the app store

Documentation for iOS deployment: <https://docs.flutter.dev/deployment/ios>

## General Production Deployment Guidelines:

1. Android and iOS bundles will be deployed to their respective internal test locations first.
2. If the build is determined to be ready for production release, a dev-ops engineer will request reviews from both app stores .
3. If Apple and Google approve the applications, the app will appear in both app stores a few hours after approval.
4. If the reviews are denied, the applications will remain on the internal test tracks and any feedback from Apple/Google will be input as bugs in the github backlog.

# Appendices.

## Credits

Thank you to Professor Assadullah, Robert Wilson, Roy Gordon and Bob Dixon for providing us with this project and providing assistance when needed. This document was developed by

Mike Conatser

Christopher Thorn

Kuleni Digga

Natan Tafese

Fahed Masood

Scott Huber

Harsh Gadani

Jahan Brahmabhatt

Ali Fahimi

Imoh Noah

Jack Shira

## 

## Credits to Previous Cohorts.

Thank you to the Summer 2022 class of SWEN 670 for creating the beginning of this project on which we have been developing.

Thank you to the Spring 2022 Cohort. Some general boilerplate parts of this document for dart applications are from the Spring 2022 Cohort Runbook, the reference to that document for information is listed in the references section above.