

**Deployment and Operations (Dep/Ops) Guide (Runbook)**

MailSpeak Application (MSA)

University of Maryland Global Campus (UMGC)

Software Engineering (SWEN) 670

Fall Cohort 2022

Team B

November 5, 2022

**Document Control**

**Document Information**

| **©** | **Information** |
| --- | --- |
| Document Id | USPS-MSA-DOG-20221029-Fall2022 |
| Document Owner | UMGC SWEN 670 |
| Issue Date | October 29, 2022 |
| Last Saved Date | November 5, 2022 |
| File Name | DOG - 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 5](#_Toc117984488)

[1.1 Purpose 5](#_Toc117984489)

[1.2 Intended Audience 5](#_Toc117984490)

[1.3 Scope 5](#_Toc117984491)

[1.4 Project Documents 5](#_Toc117984492)

[1.5 Document References 6](#_Toc117984493)

[1.6 Identify Stakeholders 6](#_Toc117984494)

[2 Development Team Workflow 6](#_Toc117984495)

[2.1 Description of Deployment 6](#_Toc117984496)

[2.2 Points of Contact 7](#_Toc117984497)

[2.3 Deployment Schedule 8](#_Toc117984498)

[3 Mobile Application 8](#_Toc117984499)

[3.1 Features, Packages, and Plugins 8](#_Toc117984500)

[3.1.1 Features 8](#_Toc117984501)

[3.1.2 Packages 8](#_Toc117984502)

[3.1.3 Plugins 8](#_Toc117984503)

[4 Software Installation 9](#_Toc117984504)

[4.1 Android Studio 9](#_Toc117984505)

[4.2 Flutter and Dart 12](#_Toc117984506)

[4.3 Android Emulator 13](#_Toc117984507)

[4.4 GitHub Desktop 14](#_Toc117984508)

[5 Prepare Mobile Application for Use 15](#_Toc117984509)

[5.1 Cloning the GitHub Repository 15](#_Toc117984510)

[5.2 Run the Flutter Application 15](#_Toc117984511)

[5.3 Setting up an Android Device 15](#_Toc117984512)

[5.4 Setting up an iOS Device 17](#_Toc117984513)

[6 Deployment, Testing and Release of the Mobile Application 20](#_Toc117984514)

[6.1 Google Play Store (Android) 20](#_Toc117984515)

[6.1.1 Deploying to Google Play Store (Android) 20](#_Toc117984516)

[6.1.2 Promoting Releases 21](#_Toc117984517)

[6.1.3 Viewing Errors with Release 22](#_Toc117984518)

[6.2 App Store Connect (iOS) 23](#_Toc117984519)

[6.2.1 Deploying to App Store Connect (iOS) 23](#_Toc117984520)

[6.2.2 Testing 28](#_Toc117984521)

[7 Troubleshooting 33](#_Toc117984522)

[7.1 Emulator Freezing 33](#_Toc117984523)

[7.2 Emulator Not Responding 33](#_Toc117984524)

[7.3 Out of Memory 34](#_Toc117984525)

[7.4 Dependency Errors 34](#_Toc117984526)

[Appendices 35](#_Toc117984527)

[Appendix A – Acronyms and Abbreviations 35](#_Toc117984528)

[Appendix B – Enabling Screen Reader on Android Device 36](#_Toc117984529)

[Appendix C – Enabling Screen Reader on iOS Device 37](#_Toc117984530)

# Introduction

The following Deployment and Operations Guide (DOG) will detail a set of instructions for the process of installing and deploying the MailSpeak application designed as part of the UMGC Fall 2022 Cohort. The guide will detail how to install the application on a user’s device and provide instructions of each step to configure the application for use.

## Purpose

This Deployment and Operations Guide (Runbook) describes how the system will be deployed and transitioned to a Production environment for all installation sites. Specifically, the Deployment Plan:

* Describes the major activities for deploying the system, including the objective of each activity, required resources, key personnel, and completion criteria.
* Presents the schedule for deploying the system.
* Describes system security features and security requirements for system deployment.
* Identifies required implementation support, including personnel, hardware, software, facilities, and materials.
* Designates the production sites at which the system will be deployed and describes how the system will be installed at the production sites.
* Presents the schedule for developing and testing recoverability prior to production implementation.

## Intended Audience

The intended audience for the deployment and operations guide (DOG) is the technical project team and stakeholders. This document will provide necessary guidelines on what is included within the application, how to deploy the application, and how to test the application.

## Scope

The scope of this document details the processes involved with deploying the application from the developer’s local environment all the way to the production environment, as well as the steps to configure these environments to properly run the application for testing and usage. This document is limited to only mobile application development via iOS and Android as those are the current target platforms for this iteration of product development.

## Project Documents

This DOG is part of a set of essential documents created to adequately manage, control and deliver the MailSpeak 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 1.4.1.

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 Development and Operations 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>

## Identify Stakeholders

Table 1.6.1 - 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) |

# Development Team Workflow

## Description of Deployment

Using GitHub, developers will branch from the repository to develop the features to be added to the previously developed MailSpeak application. Once development and unit testing has completed on the feature branch, the developer will submit a pull request, which must be reviewed by two other members of the team. Upon completion of this pull request, a CI/CD pipeline for iOS and Android will be run using Github Actions. The pipeline will assign a Build Identification Number and populate any secrets defined in the repository, setup Flutter and install any dependencies, and then generate the desired artifact(s). Once the artifact is built, the pipeline will run unit tests that must succeed before the application can be released to the test track for the AppStore/PlayStore (developer branch only). All stages must complete successfully, in addition to needing the approval of two team members, before a Pull Request can be merged into the developer branch. Once merged into the developer branch, the CI/CD pipelines will be triggered again and release the artifact to their respective store for Pre-Production testing.

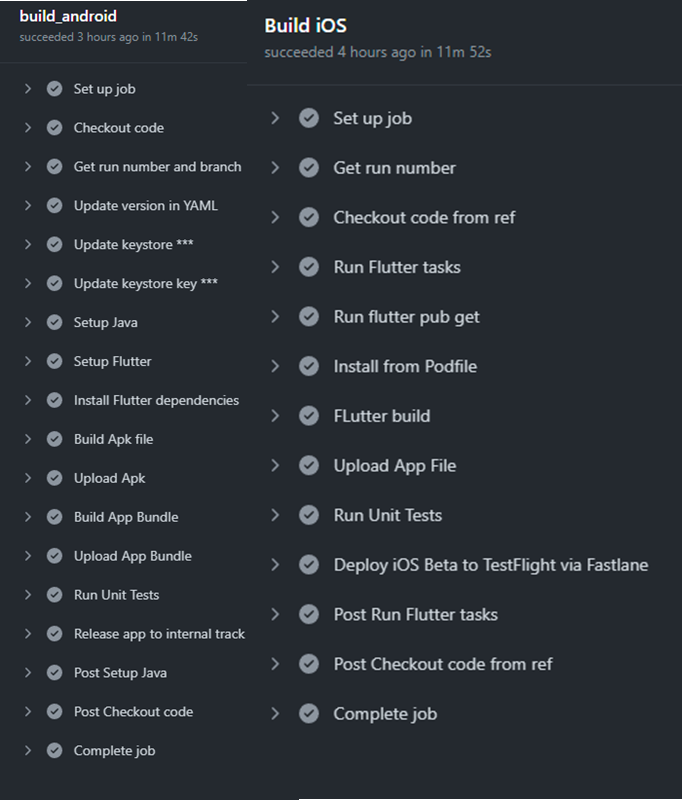


Figure 2.1.1 - CI/CD Pipelines for GitHub Actions

## Points of Contact

Table 2.2.1 - Points of Contact

| Name | Project Role |
| --- | --- |
| Mir Assadullah | Professor, Program Manager |
| Roy Gordon | Project Mentor |
| Robert Wilson | DevSecOps Mentor |
| Robert Dixon | Project Sponsor |
| 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 | SE II |
| Sarah Johnson | SE I |
| Shane Knowles | DevSecOps Engineer (Principal) |
| Tatiana Kozhevnikova | Product Owner |
| TraMel Perry | Principal Software Engineer (SE) |

## Deployment Schedule

All features will be completed by the end of the third milestone (10/29/2022), with individual feature testing being completed as features are completed. A final testing event will be completed for each iOS and Android by the fourth milestone (11/05/2022).

# Mobile Application

## Features, Packages, and Plugins

### Features

* **Search:** Users will have the ability to search their mail items by date range and/or keyword to find a mail piece they are looking for
* **Chatbot:** Users will be able to use a chatbot to more easily navigate the application.
* **Accessibility:** Users will be able to use the gesture- and/or voice- driven commands, as well as the native voice-to-text and text-to-speech capabilities to navigate the application.

### Packages

Adding a package dependency to the application:

1. Open the pubspec.yaml file inside the app folder and add the package name under dependencies
2. Install it:
   1. From the terminal run: “flutter pub get”.  
      **OR**
   2. From Android Studio, click “Packages Get” in the action ribbon at the top of the pubspec.yaml file.
3. Add the corresponding import statement in the Dart code
4. Restart the application if necessary

### Plugins

Installing Flutter and Dart plugins

**Windows or Linux:**

1. Start Android Studio
2. Open plugin preferences (File > Settings > Plugins)
3. Select “Marketplace”, select the Flutter plugin
4. Click “Install”

**MacOS:**

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

# Software Installation

## Android Studio

Android Studio is the official IDE for Google’s Android Operating System (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 Central Processing Unit (CPU) architecture, 2nd generation Intel Core or newer, minimum 8 Gigabytes (GB) Random Access Memory (RAM), minimum 8 GB available disk space, minimum 1280 x 800 screen resolution.

Packages:

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

Installation:

Graphical user interface, application

Description automatically generated

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 software development kit (SDK) packages.

**MacOS:**

System Requirements: MacOS 10.14 (Mojave) or higher, Advanced RISC (Reduced Instruction Set Computer) Machine (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 disk image (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: Linus for Chrome OS, Intel i5 or higher recommended, minimum 8 GM RAM, minimum 4 GB available disk space, minimum 1280 x 800 screen resolution.

Packages:

1. Graphical user interface, application

   Description automatically generatedChrome

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”.

## Flutter and Dart

Flutter is an open-source UI SDK created by Google, used to develop cross platform applications for Android, iOS, Linux. MacOS, Windows, Google Fuchsia and the web using a single code base. Dart is a programming language, also developed by Google, designed for web and mobile applications.

**Flutter:**

**Windows:**

System Requirements: Windows 10 or later, 1.64 GB available disk space, Windows PowerShell 5.0 or newer, Git for Windows 2.x, with the “Use Git from Windows Command Prompt” option.

Installation:

Graphical user interface, application

Description automatically generated

1. Download the latest stable release of the Flutter SDK
2. Extract the zip file and place the contained “flutter” in the desired installation directory
3. Update the path to run flutter commands from the Windows console
   1. From the search bar, enter ‘env’ and select “Edit Environment Variable for your account’
   2. Under “User variables” check if there is a “Path”
      1. If there is, append the full to path to “flutter/bin” using a “;” as a separator
      2. If there isn’t, create a new user variable named “Path” with the full path to “flutter/bin”
4. Close and reopen any console window for changes to take effect.
5. From a console window, run “flutter doctor” to find any dependencies you need to complete setup
   1. Check output for other software needed to be installed or further tasks to perform (in bold)
6. Once missing dependencies have been installed, run “flutter doctor” again to confirm.

Detailed instructions: <https://docs.flutter.dev/get-started/install/windows>

**MacOS:**

System Requirements: macOS, 2.8 GB available disk space, git

Installation:

1. Download the latest stable release of Flutter SDK
2. Extract the file in the desired location
3. Add flutter tool to path



1. Run “flutter doctor” to see if there are any dependencies that need to be installed
   1. Check output for any other software needed or further tasks to perform
2. Confirm all dependencies have been installed by again running “flutter doctor”

Detailed instructions: <https://docs.flutter.dev/get-started/install/macos>

**Linux**:

System Requirements: Linux (64-bit), 600 MB available disk space, tools: bash, curl, file, git 2.x, mkdir, rm, unzip, which, xz-utils, zip

Installation:

1. Download the latest stable release of Flutter SDK
2. Extract the file in the desired location  
   
3. Add flutter tool to path



1. Run “flutter doctor”
2. Installed needed dependencies found in output

Detailed instructions: <https://docs.flutter.dev/get-started/install/linux>

**Chrome OS:**

System Requirements: Chrome OS (64-bit) with Linux (Beta) turned on, 600 MB available disk space, tools: bash, curl, git 2.x, mkdir, rm, unzip, which, xz-utils

Installation:

1. Download the latest stable release of Flutter SDK
2. Drag and drop the downloaded file from “Downloads” to “Linux Files”
3. Extract the file in the desired location  
   
4. Add flutter tool to path  
   
5. Run “flutter doctor”
6. Install needed dependencies found in output

Detailed instructions: <https://docs.flutter.dev/get-started/install/chromeos>

## 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:

A screenshot of a computer

Description automatically generated with medium confidence

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.

## GitHub Desktop

GitHub will be used as the code repository.

System Requirements: Windows 7 (64-bit) or later, MacOS 10.12 or later

Installation:

1. Download the current version of GitHub from <https://desktop.github.com/>
2. In your computer’s “Downloads” folder, double-click the “GitHubDesktopSetup” file (Windows) or the “GitHubDesktop” zip file, unzip, and then click “GitHubDesktop” file
3. Follow setup instructions

# Prepare Mobile Application for Use

## Graphical user interface, text, application Description automatically generatedCloning the GitHub Repository

To clone the GitHub repository:

1. Navigate to <https://github.com/umgc/fall2022>
2. Click the green “Code” button
3. Select the “HTTPS” option
4. Click “Download ZIP”

## Run the Flutter Application

To run the cloned flutter application on a device:

1. Unzip cloned application code
2. Open Android Studio
3. Import the project (File > New > Import Project)

Graphical user interface, text, application

Description automatically generated

## Setting up an 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 > Build Number” and tap the “Build Number” option 7 times.

Graphical user interface, application

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.

Graphical user interface, qr code

Description automatically generated

## Setting up an 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.

Setup:

1. Open the applications ios workspace in Xcode. This is found in “ios/Runner.xcworkspace”.  
   Graphical user interface, text, application, email

   Description automatically generated
2. Connect the iOS device to the computer using a USB.
3. Click on the “run destination menu” in the toolbar and select the connected device.  
   Graphical user interface, application

   Description automatically generated
4. In the project navigation, select the “Runner” target, then select the “Signing & Capabilities” tab.
5. Check the “Automatically manage signing” checkbox.
6. For “Team” select the Development Team.

Graphical user interface, application

Description automatically generated

1. If necessary, provide a unique “Bundle Identifier”, e.g. com.example.fall2022.

Graphical user interface, application

Description automatically generated

1. Under the “General” tab, update the application’s display name in the “Identity” section.

Graphical user interface, application

Description automatically generated

1. Click the “Run” button to build and deploy the application to the connected device.
2. The first time the application is deployed to the device, the application must the trusted by the device:
3. On the device, navigate to the “Settings > General > VPN & Device Management” section
4. Under “Developer Apps”, click “Apple Development”, then “Trust Apple Development”.
5. To enable developer mode, go to “Settings > Privacy & Security > Developer Mode” and toggle developer mode on.

A screenshot of a computer

Description automatically generated with medium confidence

# Deployment, Testing and Release of the Mobile Application

## Google Play Store (Android)

### Deploying to Google Play Store (Android)

The CI/CD pipeline allows building and publishing to the Google Play Store as a draft for review and testing by the specified team. Once the test team has been configured, they can be assigned to a draft and follow the provided link to download the application onto their devices from the Google Play Store.

To configure the list of tests, navigate to Internal Testing on the Google Play Console and select the Testers tab as shown in Figure 6.1. Selecting the team will assign them to the test application making it available for download. Selecting the arrow to the right of the row will allow management of the team members.

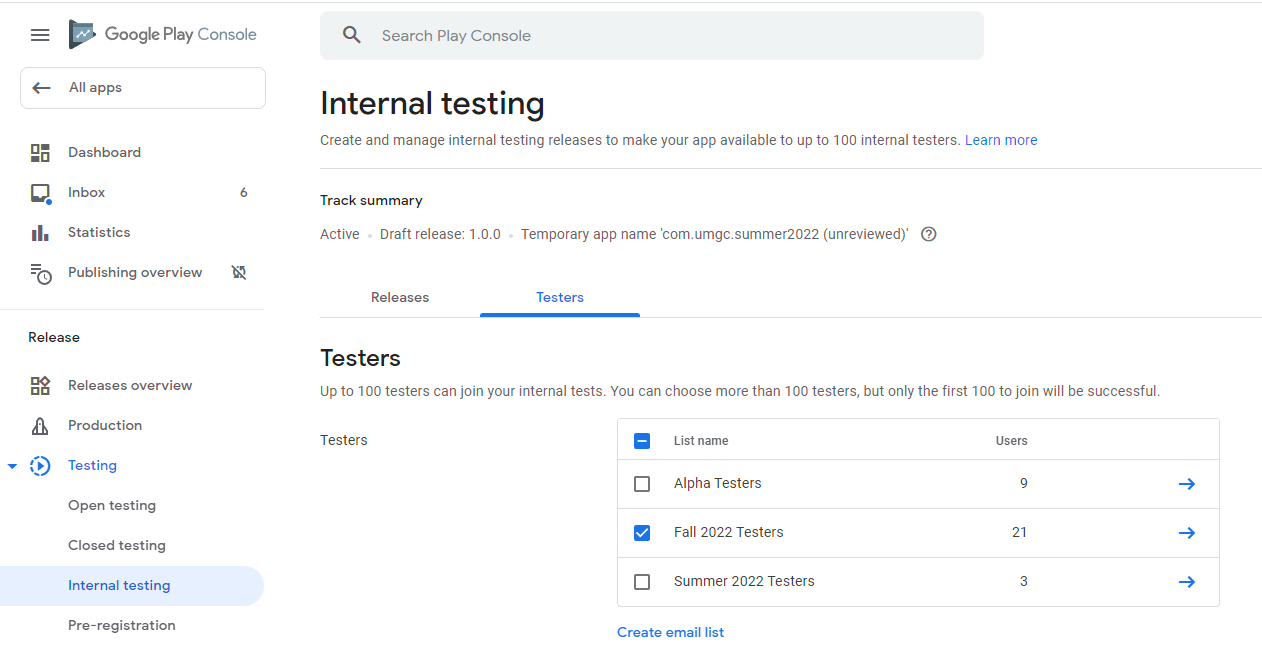


Figure 6.1.1 - Google Play Console Tester Specification

In order to get the CI/CD pipeline automatically releasing to the Google Play Console, a few steps must be followed. Note: Some of these steps may already be completed, but use this list to follow if issues arise with the current process.

* Generate Java Key Store (jks) file for signing the application (<https://developer.android.com/studio/publish/app-signing#generate-key>)
* Add the generated jks file to the repository (android/app/) and point to it in the key.properties file (android/key.properties) via the storeFile parameter
* Add the specified password in GitHub Secrets for KEYSTORE\_PASS and KEYSTORE\_KEY\_PASS (this values get set during the CI/CD pipeline)
* Using pc-api credentials for the Google Play Console service account (must be provided by account owner, in this case the professor), set the SERVICE\_ACCOUNT\_JSON secret
* Run the GitHub pipeline to release a draft version of the application. If there are any pending drafts they may need to be discarded (especially if new service account credentials are being used)
* Once the draft is set for internal testing, a link can be sent to the approved testers so they can download the application via the Google Play Store and begin testing

### Promoting Releases

After testing has completed on the internal track, the release can be promoted to closed alpha testing, and then onto open testing/release tracks.

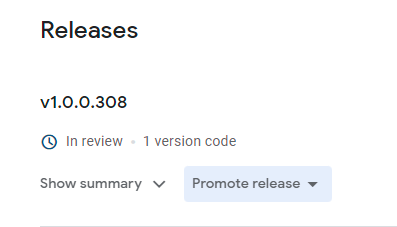


Figure 6.1.2 - Promote Release

### Viewing Errors with Release

Viewing the Pre-launch report will display any issues with the release that must be fixed before a production promotion can be performed.

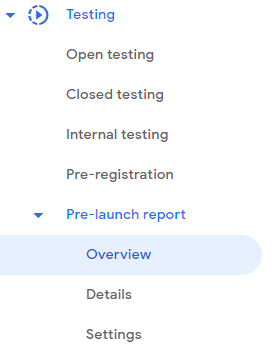


Figure 6.1.3 - Pre-launch report

Issues can be reviewed on this overview page to get more detailed descriptions as shown in Figure 6.1.3

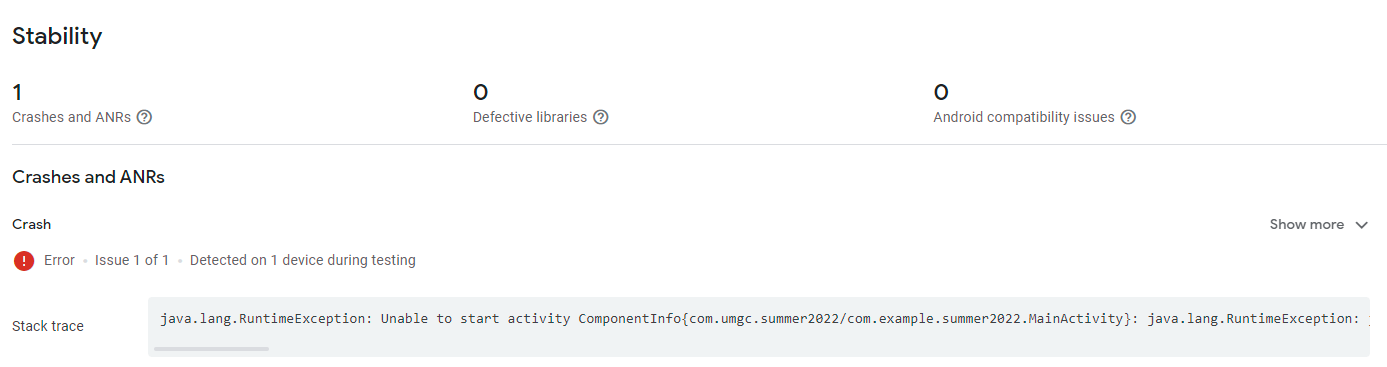


Figure 6.1.4 -Fixing issues

## App Store Connect (iOS)

### Deploying to App Store Connect (iOS)

Deploying an iOS app to App Store Connect will allow to:

* Invite users for testing on TestFlight
* Release the app on the App Store

To deploy an iOS app to the App Store Connect enrollment into the paid type of the Apple Developer Program is required. This will allow the creation of the provisioning profiles and certificates for app signing.

General steps to deploy an app for testing and further release include:

* Create a record for a new app in the App Store Connect
* Using XCode to archive and upload the app to the App Store Connect

#### Create a record for a new app in the App Store Connect

To create a record for the app in the App Store connect go to <https://appstoreconnect.apple.com/> and log in with Apple ID that was used to enroll into the Apple Developer Program or Apple ID that was added to the account in a specific role.

On the main page select “My Apps” (Figure 6.2.1)

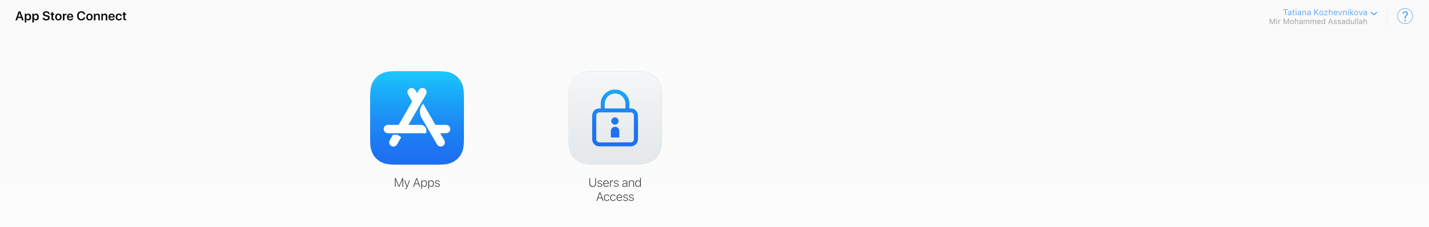


Figure 6.2.1 - Main Page App Store Connect

On the App page click “+” and select “New App” as shown in Figure 6.6

Graphical user interface, application

Description automatically generated

Figure 6.2.2 - Apps Management

In the “New App” dialog populate all required fields (Figure 6.2.3).

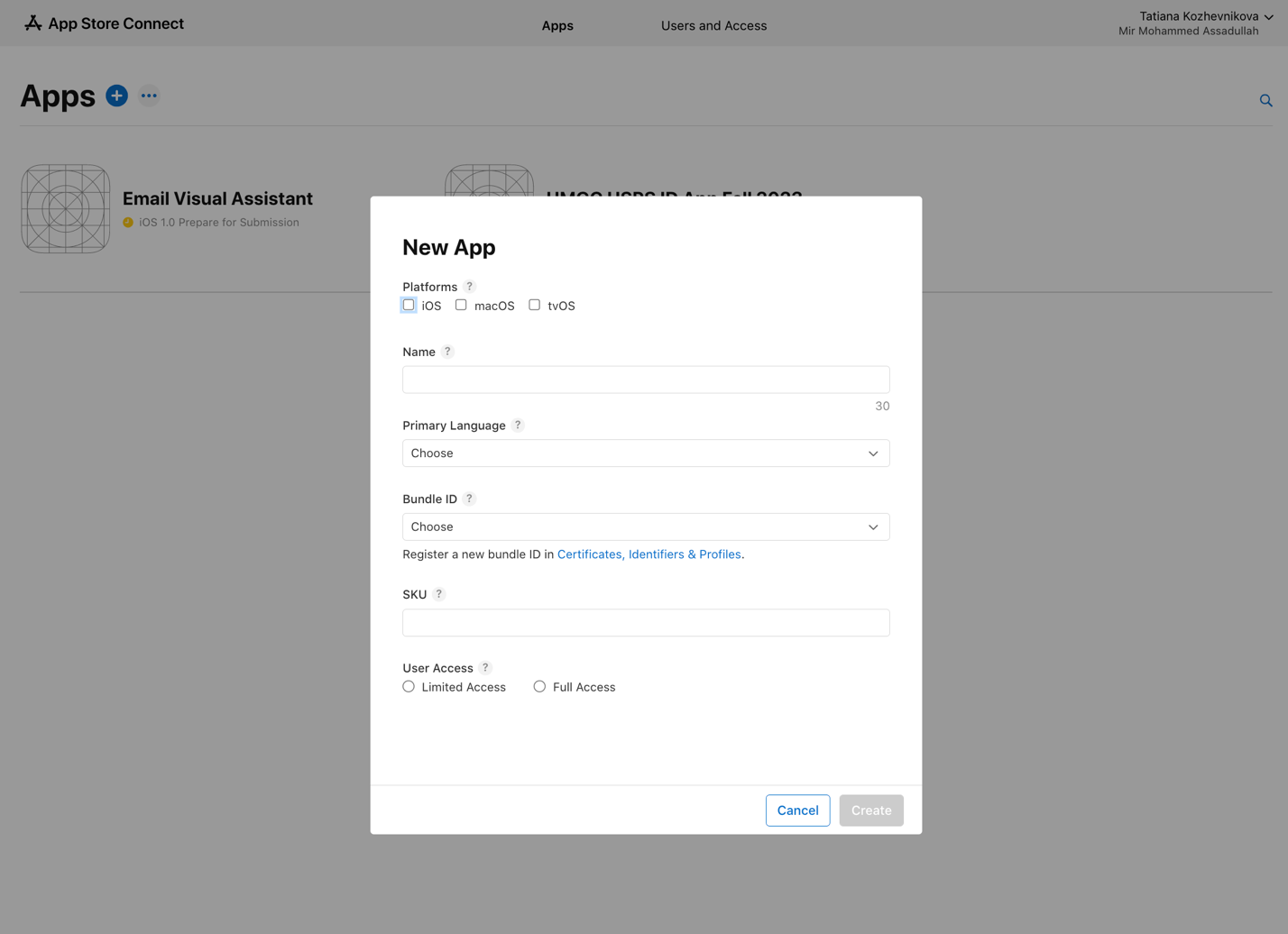


Figure 6.2.3 - Create New App

The new app record is created, and all information can be accessed by clicking on the app (Figure 6.2.4)

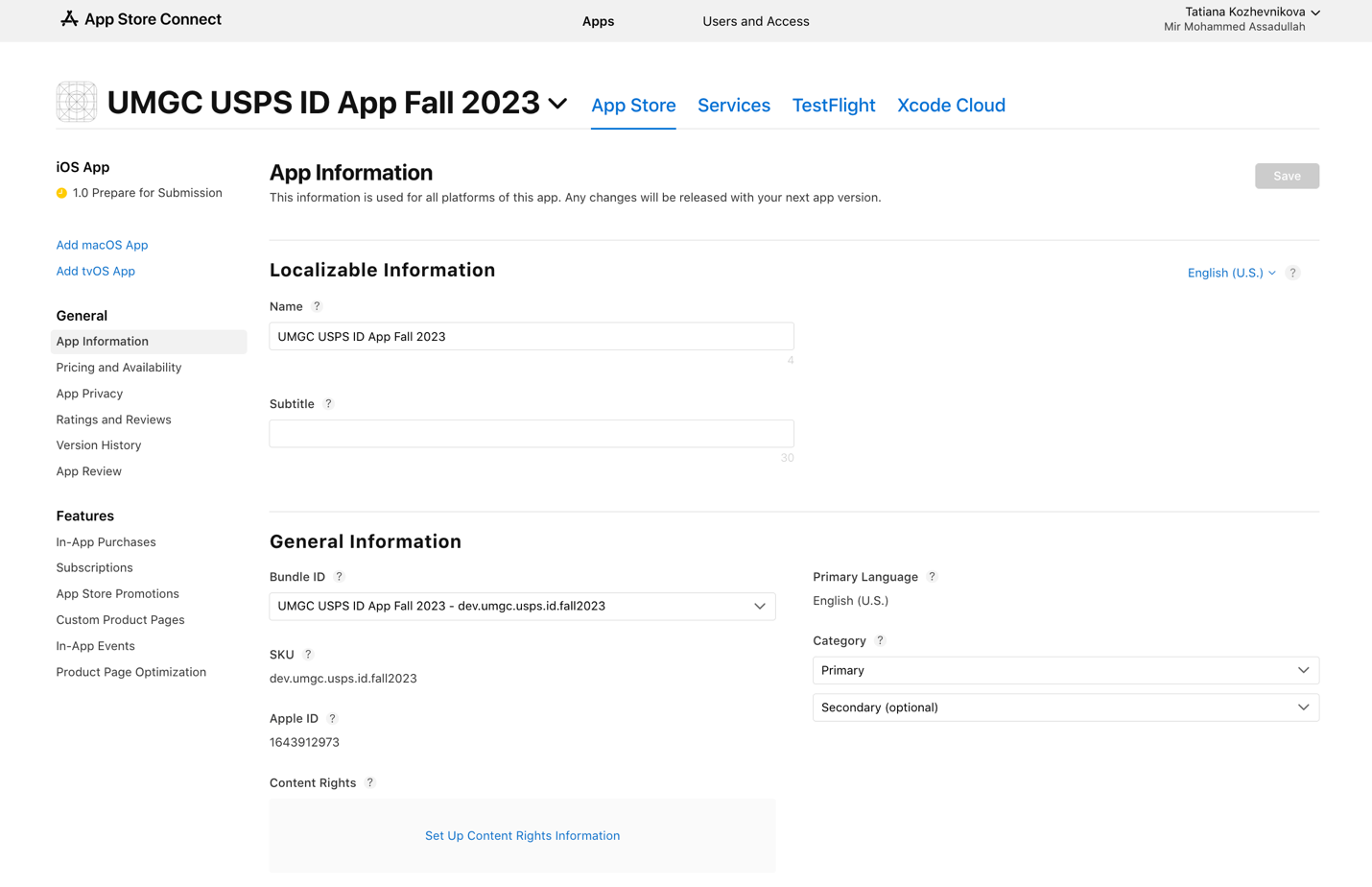


Figure 6.2.4 - App Details

#### Using XCode to archive and upload the app to the App Store Connect

Open the app in the XCode and connect a physical device. Select “Automatically manage signing”, choose a valid development account and confirm that “Bundle Identifier” matches the bundle ID from the app record in the App Store Connect (Figure 6.2.5). To properly update bundle ID change the corresponding value in Info.plist.

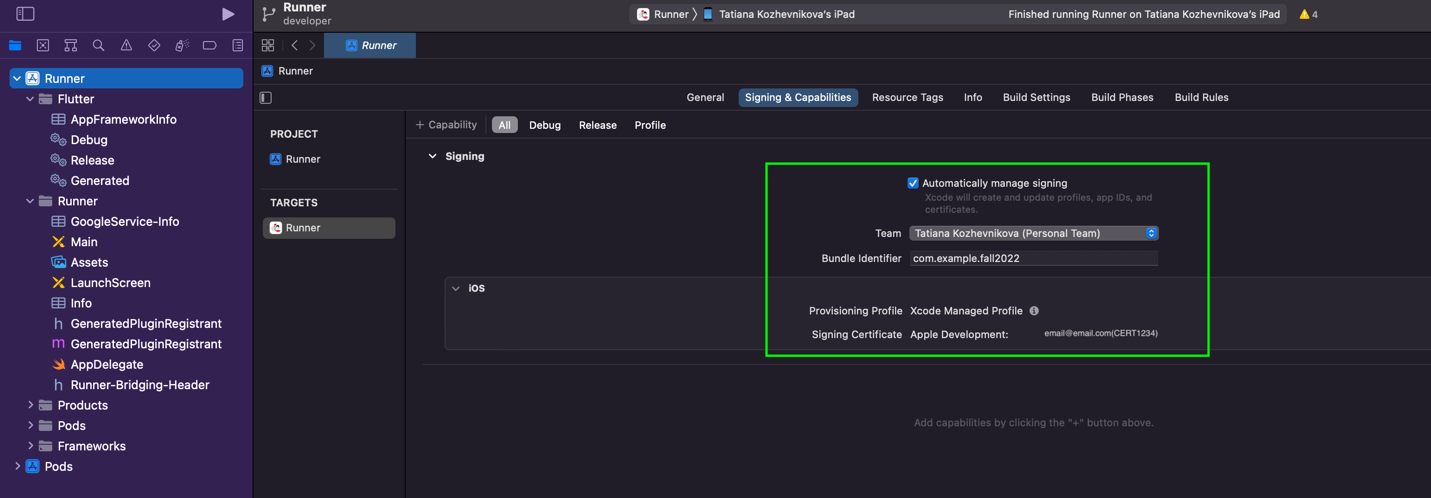


Figure 6.2.5 - Signing Profile Setup

In the top menu click “Product” and select “Archive” (Figure 6.2.6). The archiving process has been started, which may take a couple of minutes.

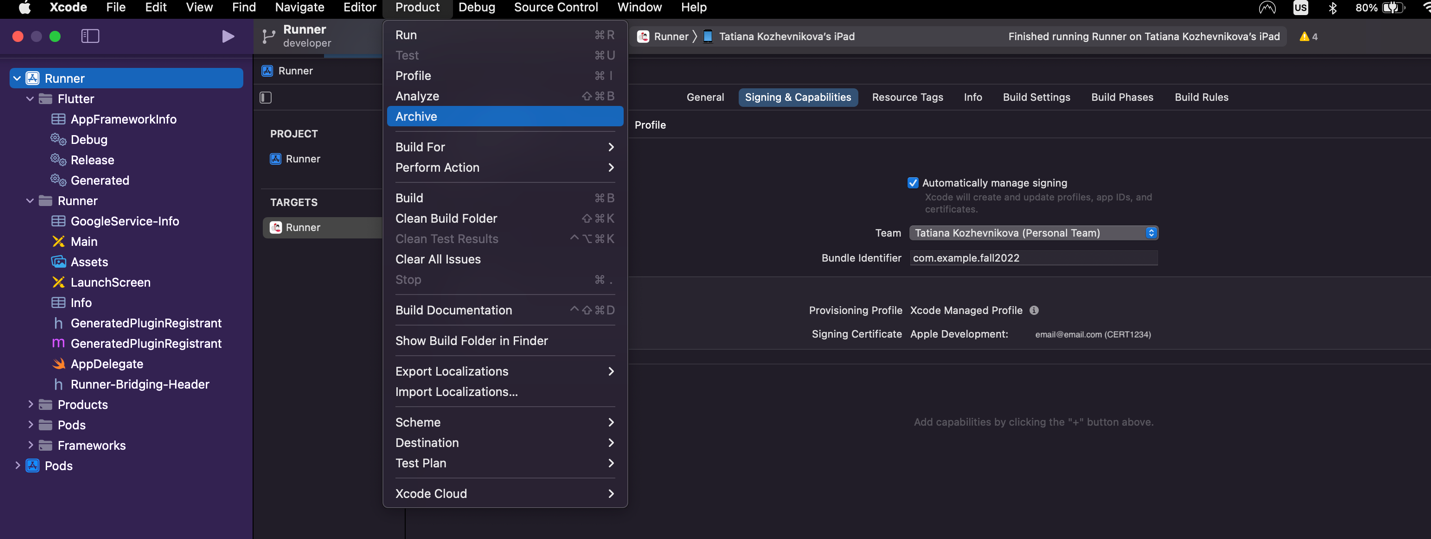


Figure 6.2.6 - App Archiving Process

Successfully completed archiving process will result in the creation of the archive file and will display it in the “Archives” window as shown in Figure 6.2.7

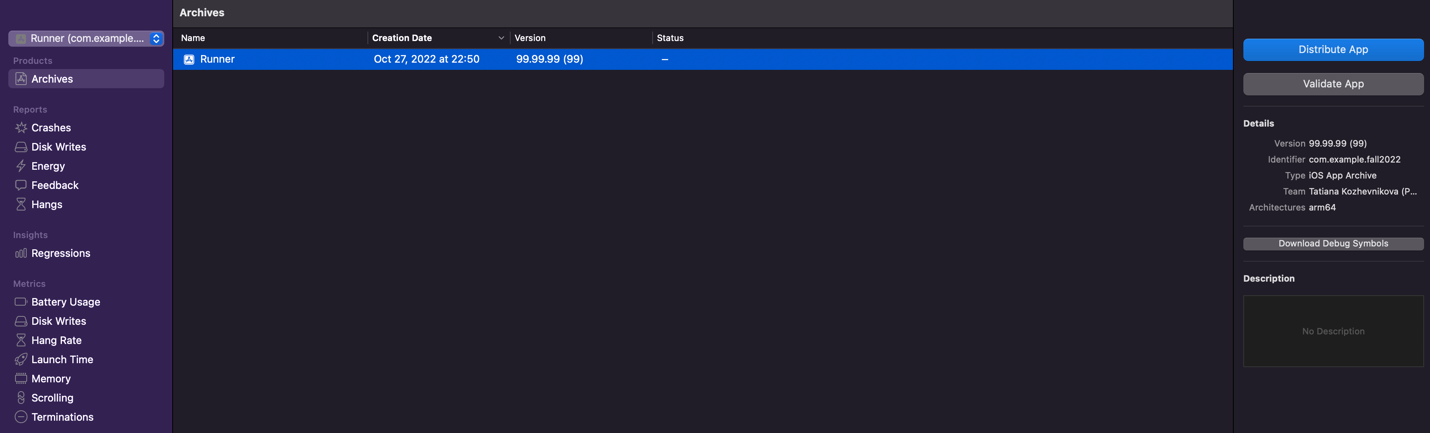


Figure 6.2.7 - Archives Ready for Distribution

Click the “Distribute App” button to proceed with the deployment process. A prompt with options for distribution methods are presented (Figure 6.2.8). Select “App Store Connect” for distribution on TestFlight and App Store.



Figure 6.2.8 - Finalizing Distribution: Method Selection

In the next prompt select “Upload” as the app destination, which will send the app to App Store Connect (Figure 6.2.9).

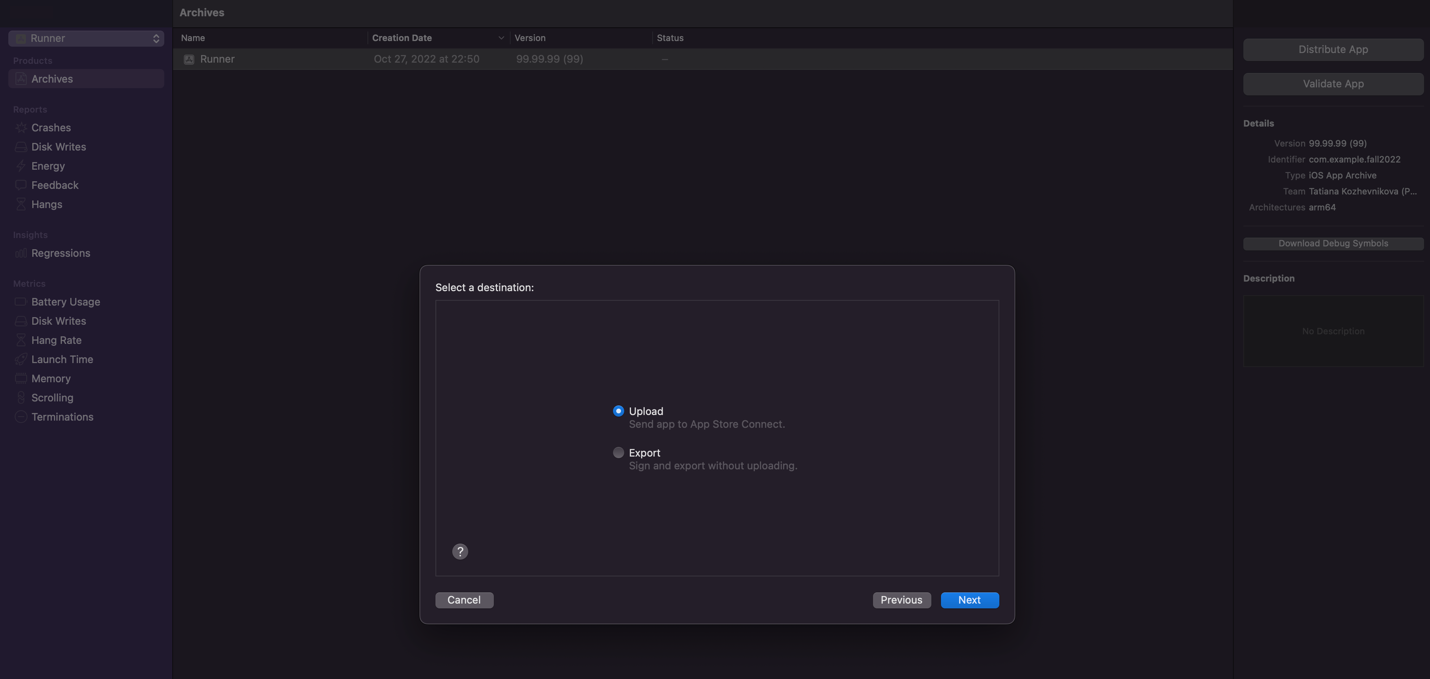


Figure 6.2.9 - Finalizing Distribution: Destination Selection

Further steps will be updated when the paid Apple Developer Program will be purchased. As current provisioning profile and certificate don’t allow App Connect Distribution.

### Testing

When the app is uploaded to the App Store Connect it can be distributed for internal testing via TestFlight. In the App Store Connect select the app navigate to the “TestFlight” tab and click “+” next to “Internal Testing” (Figure 6.2.10).

Graphical user interface, text, application, Teams

Description automatically generated

Figure 6.2.10 - Internal Testing Setup

Create new testing group and add testers as Figures 6.2.11 and 6.2.12 show respectively.

Graphical user interface, application, Teams

Description automatically generated

Figure 6.2.11 - New Testing Group Creation

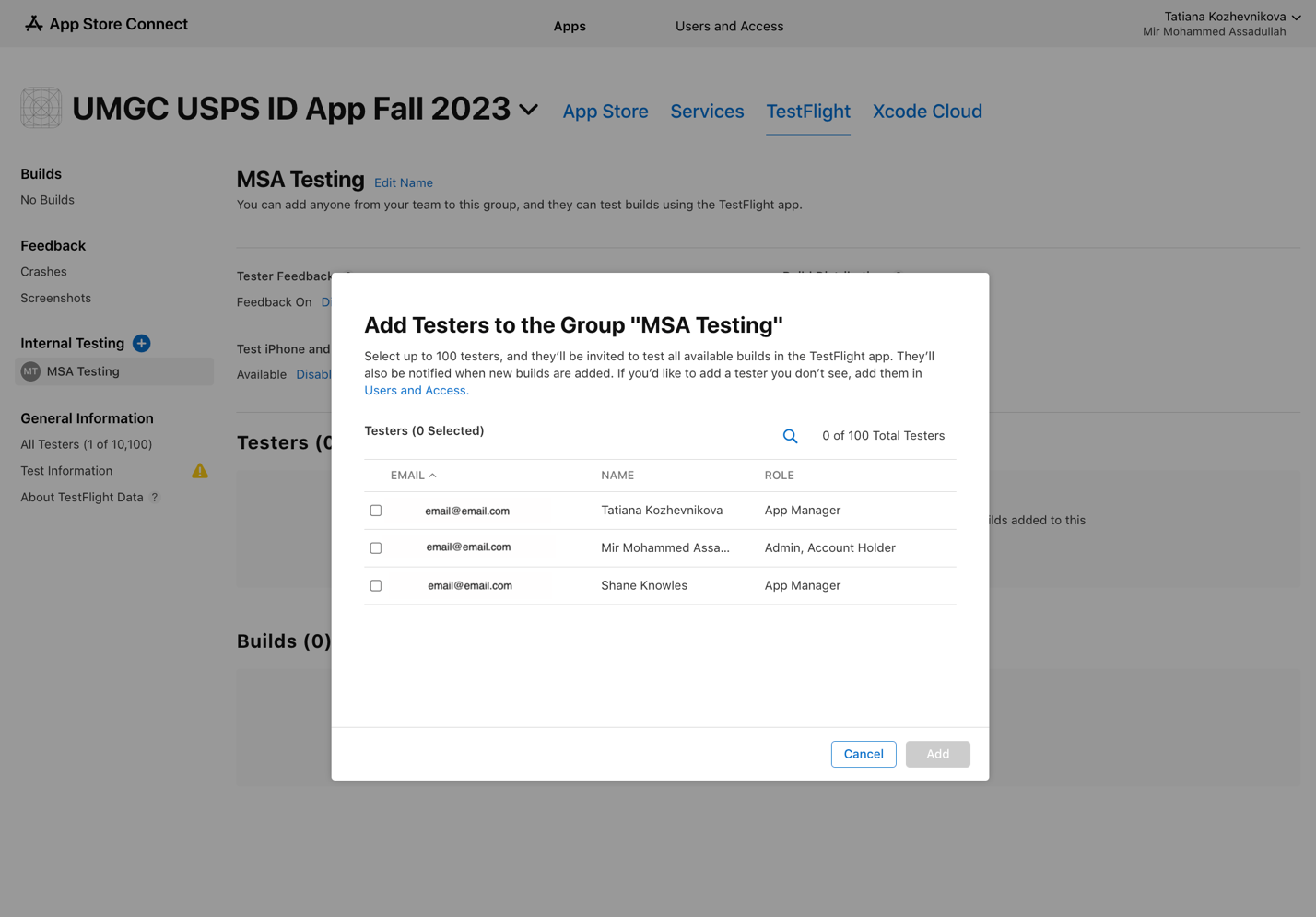


Figure 6.2.12 - Adding Tester to the Testing Group

Testers are added to the testing group (Figure 6.2.13), now they can download the app through TestFlight and will get automatic app updates whenever the new build is released.

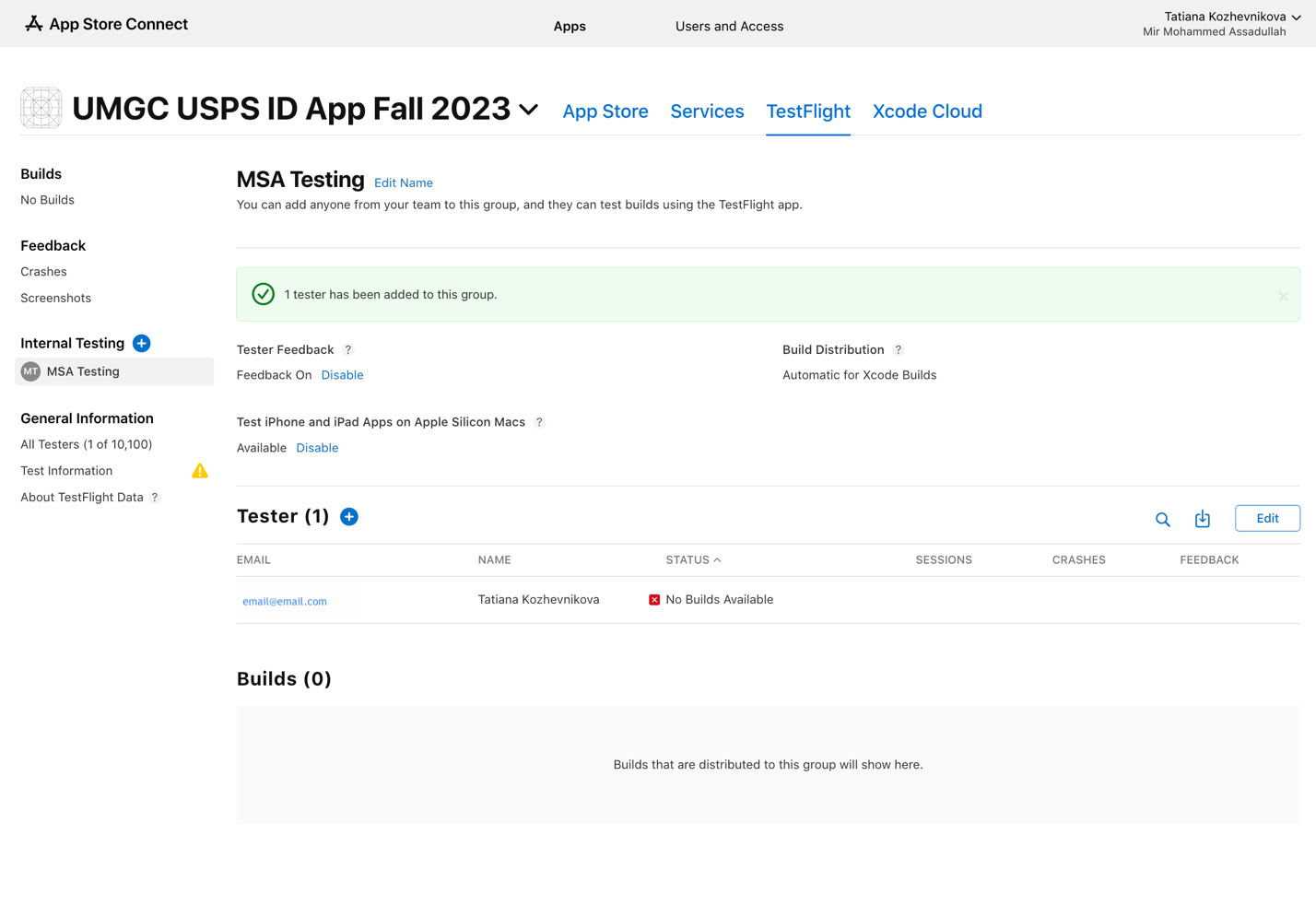


Figure 6.2.13 - Internal Testing Group View

# Troubleshooting

This section will cover the common issues users might run into as they are developing and testing the application.

## Emulator Freezing

Perform a cold boot:

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

It is recommended to perform a cold boot after any type of troubleshooting is performed to clear the RAM of internal data.

## Emulator Not Responding

Wipe emulator data and return it to its original state:

1. If emulator is running, stop/exit the emulator.
2. In Android Studio, open the “Android Manager” by either clicking the icon in the top ribbon or click on “Tools > Android Manager”.
3. Select the emulator that has frozen and click the down arrow in the “Actions” column.
4. Select “Wipe Data” and perform a cold boot

## Out of Memory

Free up virtual memory and increase assigned memory:

1. Close all unnecessary applications running in the background.
2. In Android Studio, open the General Settings: “File > Settings”.
3. Go to the memory Settings: “Appearance & Behavior > System Settings > Memory Settings”
4. Change the IDE max heap size to desired amount (2048 MB is recommended).
5. Click “Apply” then “OK”.
6. Restart Android Studio.

## Dependency Errors

Clear Flutter build files and re-import libraries:

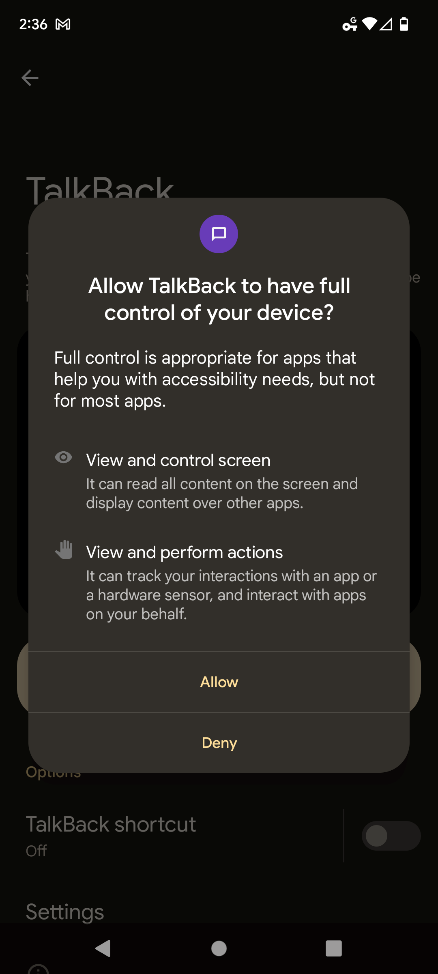
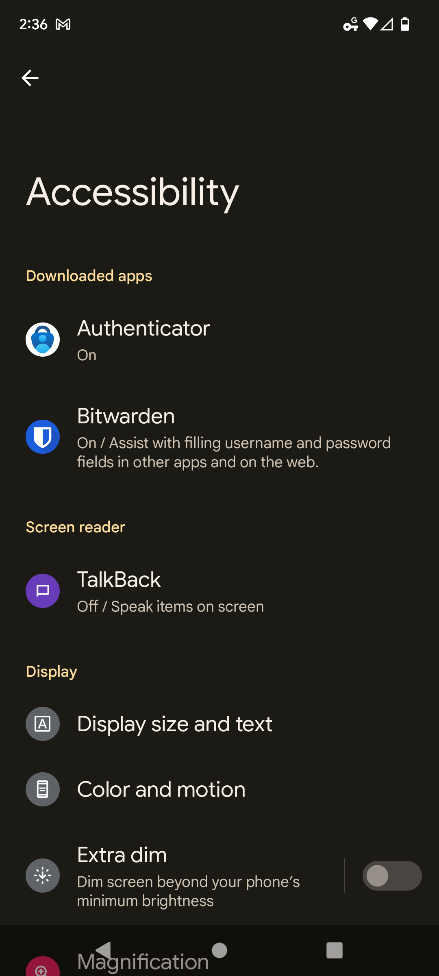
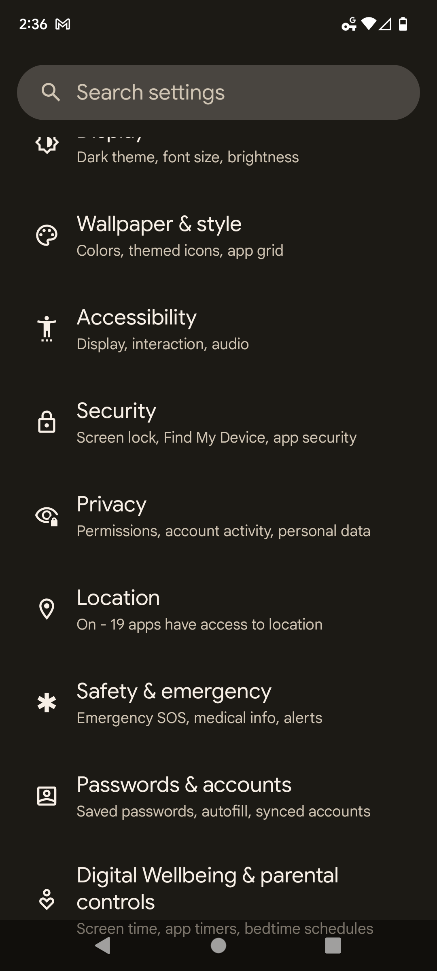
1. Open Flutter terminal
2. Navigate to directory of the Flutter project
3. To clear dependencies, run “flutter clean” command
4. Re-import the libraries to get the correct dependencies by running Flutter “Get dependencies” and “Upgrade dependencies”.

Appendix A – Acronyms and Abbreviations

| Term | Meaning |
| --- | --- |
| ADB | Android Debug Bridge |
| ARM | Advanced RISC Machine |
| AVD | Android Virtual Device |
| CI/CD | Continuous Integration/Continuous Deployment |
| CPU | Central Processing Unit |
| DMG | Disk Image |
| DOG | Development and Operations Guide |
| DevOps | Development and Operations |
| DevSecOps | Development, Security & Operations |
| GB | Gigabytes |
| ID | Identification |
| IDE | Integrated Development Environment |
| iOS | iPhone Operating System |
| MacOS | Macintosh Operating System |
| MS | Microsoft |
| MSA | MailSpeak Application |
| OCR | Object Character Recognition |
| OS | Operating System |
| PM | Project Manager |
| PMP | Project Management Plan |
| QR | Quick Response |
| RAM | Random Access Memory |
| RISC | Reduced Instruction Set Computer |
| SDK | Software Development Kit |
| SE | Software Engineer |
| SWEN | Software Engineering |
| TBD | To Be Determined |
| UMGC | University of Maryland Global Campus |
| USB | Universal Serial Bus |
| USPS | United States Postal Service |

Appendix B – Enabling Screen Reader on Android Device

* Open the Settings app
* Tap ‘Accessibility’ > ‘TalkBack’
* Enable ‘Use TalkBack’
* Go through tutorial until completion



Appendix C – Enabling Screen Reader on iOS Device

* Go to ‘Settings’ > ‘Accessibility’ > ‘VoiceOver’
* Turn feature On/Off
* Alternatively, you can activate Siri and say ‘Turn on VoiceOver’

