

Deployment and Operations Guide

(CogniOpen - Nurturing Memory Wellness for Cognitive Impairment)

Submitted to:

University of Maryland Global Campus in partial fulfillment of the course Software Engineering Capstone (SWEN 670)

Submitted By: Team B

Fall Semester 2023

Version 1.0

October 28th, 2023

Department of Software Engineering

University of Maryland Global Campus (UMGC)

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Issue Date | Changes | Change Made By |
| 1.0 | 09/25/2023 | Initial Version | Team B |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[1. Introduction 4](#_Toc149235169)

[**1.1** **Purpose** 4](#_Toc149235170)

[**1.2 Intended Audience** 4](#_Toc149235171)

[**1.3 Project Documents** 4](#_Toc149235172)

[**1.4 Acronyms, Definitions, and Abbreviations** 5](#_Toc149235173)

[**1.5 Document Overview** 6](#_Toc149235174)

[**1.6 Document Scope** 6](#_Toc149235175)

[2. Development Team Workflow 6](#_Toc149235176)

[**2.1 Description of Deployment** 6](#_Toc149235177)

[**2.2 Points of Contact** 7](#_Toc149235178)

[**2.3 Primary Objectives** 8](#_Toc149235179)

[3. Software Installation 9](#_Toc149235180)

[**3.1 Android Studio** 9](#_Toc149235181)

[**3.2 Flutter and Dart** 12](#_Toc149235182)

[**3.2.1 Windows Installation** 13](#_Toc149235183)

[**3.2.2 Mac Installation** 14](#_Toc149235184)

[**3.3 Android Emulator** 15](#_Toc149235185)

[**3.4 Test Your Development Environment** 17](#_Toc149235186)

[**3.5 GitHub Desktop** 20](#_Toc149235187)

[**3.6 Common Challenges** 21](#_Toc149235188)

[4. Prepare Mobile Application for Use 21](#_Toc149235189)

[**4.1 Cloning GitHub Repository** 21](#_Toc149235190)

[**4.2 Run the Application** 23](#_Toc149235191)

[**4.3 Using the .env File** 25](#_Toc149235192)

[**4.4 Getting the OpenAI ChatGPT API Key** 25](#_Toc149235193)

[**4.5** **Getting the AWS Access Key** 25](#_Toc149235194)

[5. Test the Mobile Application 26](#_Toc149235195)

[**5.1 Automation Testing** 27](#_Toc149235196)

[**5.2 Manual Testing** 29](#_Toc149235197)

[6. Troubleshooting 31](#_Toc149235198)

[**6.1 Troubleshoot Android Studio** 31](#_Toc149235199)

[**6.2 Troubleshoot Android Emulator** 32](#_Toc149235200)

# Introduction

This Deployment and Operations Guide (DOG) provides a detailed set of instructions on installing and deploying CogniOpen, a revolutionary mobile application designed to provide extensive support to and aid people experiencing cognitive impairment. CogniOpen features a user-friendly interface and a comprehensive range of functionalities that enhance cognitive capacities, foster emotional well-being, and promote a sense of belonging within a nurturing community. Developed using Dart and Flutter, the application offers a fast and native-grade application experience across multiple platforms, including iOS and Android.

## **Purpose**

The purpose of this DOG is to provide a comprehensive understanding of CogniOpen's features, functionalities, and operational procedures, including installation, configuration, security, monitoring, and troubleshooting. It is designed to educate its intended audience with the knowledge needed to effectively deploy and operate CogniOpen. By following the clear and concise instructions provided in this guide, you can ensure the optimal performance and maximum benefit of CogniOpen for its users.

## **1.2 Intended Audience**

The intended audience for this DOG includes Information Technology (IT) professionals, application developers, healthcare providers, and individuals interested in supporting their cognitively challenged loved ones. The guide is designed to be accessible to both technical and non-technical audiences, providing clear and concise instructions for effective deployment and operation of CogniOpen.

## **1.3 Project Documents**

This document is one part of the deliverables for the CogniOpen project. The other documents are listed in the table below with the most recent version of each at the time of editing this Deployment and Operations Guide.

|  |  |  |
| --- | --- | --- |
| Document | Version | Date |
| Project Plan (PPL) | 1.2 | 10/28/2023 |
| Software Requirements Specification (SRS) | 1.2 | 10/28/2023 |
| Technical Design Document (TDD) | 1.1 | 10/28/2023 |
| Test Plan (TP) | 1.1 | 10/28/2023 |
| Programmer Guide (PG) | 1.0 | 10/28/2023 |
| Deployment and Operations Guide (DOG) | 1.0 | 10/28/2023 |
| Software Test Report (STR) | - | - |
| User Guide (UG) | - | - |
| Traceability Matrix (TM) | - | - |

Table 1: Project Documents

## **1.4 Acronyms, Definitions, and Abbreviations**

|  |  |
| --- | --- |
| Term | Definition |
| ADO | Azure DevOps |
| API | Application Programming Interface |
| App | A program that is included on the User’s mobile device |
| AWS | Amazon Web Services |
| HIPPA | Health Insurance Portability and Accountability Act |
| HTTPS | Hypertext Transfer Protocol Secure |
| IAM | Identity and Access Management |
| IDE | Integrated development environment |
| iOS | iPhone Operating System |
| Mobile Device | A smart phone, tablet, or some other portable computer with either the iOS or Android operating system |
| MVC | Model View Controller |
| OS | Operating System |
| REST | Representational State Transfer |
| SDK | Software Development Kit |
| TDD | Technical Design Document |
| UI | User Interface |
| UMGC | University of Maryland Global Campus |
| UX | User Experience |

Table 2: Definitions, Acronyms, and Abbreviations

## **1.5 Document Overview**

This document provides an in-depth overview and guide for the development team and other stakeholders involved in the mobile application project. It begins with an introduction to the project's objectives and scope, followed by a detailed explanation of the development team workflow. The document then goes on to provide step-by-step instructions for software installation, mobile application preparation, and testing. Finally, it includes a troubleshooting section to address any potential issues that may arise during the development process.

## **1.6 Document Scope**

This document covers only the technical components assigned to Team B. Discussions on features and components assigned to other teams will be detailed in their respective DOGs.

# Development Team Workflow

The purpose of the development team’s DOG workflow is to provide a structured process for deploying and operating CogniOpen. The workflow ensures that all necessary steps are taken to deploy the application successfully and minimize the risk of errors and issues during the deployment and operation process.

## **2.1 Description of Deployment**

The CogniOpen development team has chosen to use GitHub for their source control and code management needs. The team has implemented a branching structure to prevent code conflicts and ensure collaboration between the two teams: Team A and Team B. This structure has proven to be successful, and a review process has also been established for transparency and fairness. The CogniOpen application code is stored on GitHub using Git, and the repository name is “umgc/fall2023”. To contribute, team members branch out from the development branch and merge back before integrating with the other team's code in the joint development branch. Once stakeholders approve and verify the code, it will be merged into the class and main branches. For a visual representation of the workflow, please see the diagram below:

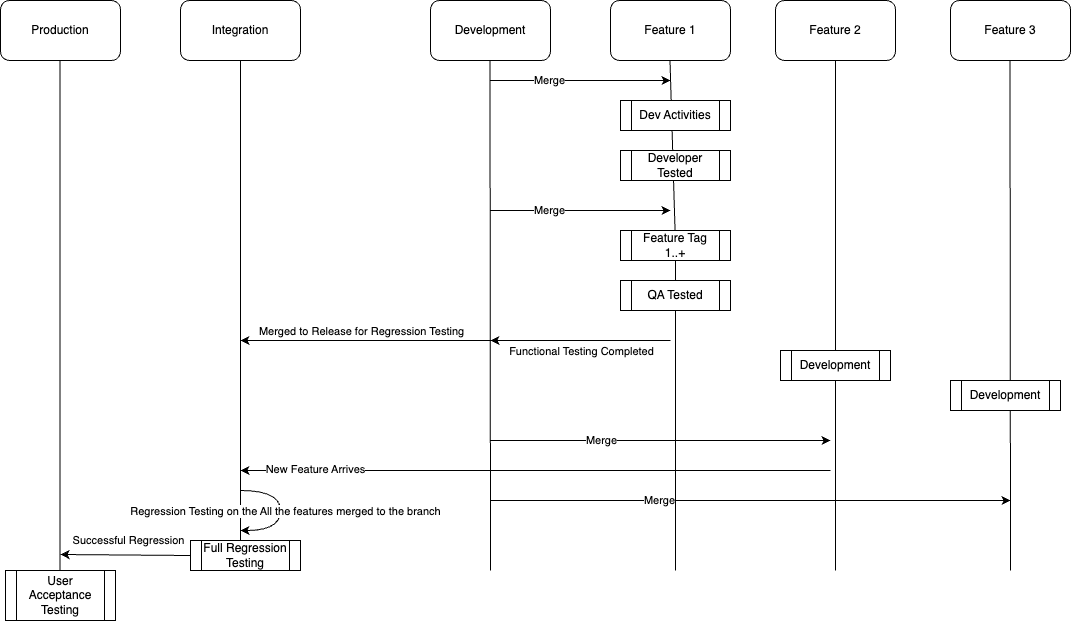


Figure 1: Branching Structure

## **2.2 Points of Contact**

|  |  |
| --- | --- |
| Name | Project Role |
| Dr. Mir Assadullah | Client/Professor |
| Roy Gordon | Project Mentor |
| Robert Wilson | DevSecOps Mentor |
| Edward Devine | Project Manager |
| Abebe Natea | Software Engineer |
| Alexis Shannon | Software Engineer |
| Eyerusalme Gebrehiwot | Software Engineer |
| Gabriel Gomes | Software Engineer |
| John Hamilton | Software Engineer |
| Malachi Jamison | Software Engineer |
| Sean Mirani | Software Engineer |
| Zachary Cappella | Software Engineer |

Table 3: Points of Contacts

## **2.3 Primary Objectives**

The primary tasks of deployment for the CogniOpen application are critical to its success. These tasks form the foundation of the deployment process, and they require careful planning and execution to ensure that the application runs smoothly. The tasks of deployment for the CogniOpen application are:

1. Project planning and documentation: The first step is to plan the project and document the requirements for the application. This includes defining the scope of the project, identifying the stakeholders, and documenting the functional and non-functional requirements.

2. Branching out from the class repository: The developer should branch out from the class repository to create a separate repository for the application. This ensures that any changes made to the codebase do not affect the original repository.

3. Setting up the development environment: The next step is to set up the development environment by installing the necessary tools, languages, frameworks, emulators, and plug-ins required for the application development.

4. Requirement identification and documentation: The developer should identify the requirements for each function of the application and document them. This ensures that the development process stays on track and that the final product meets the requirements.

5. Researching on each requirement and requirement breakdown into functions: The developer should research each requirement and break it down into smaller functions. This helps in identifying the dependencies between the functions and ensures that the development process is efficient.

6. Development of each function into the codebase: Once the requirements have been identified and broken down into functions, the developer can start developing the code for each function. This should be done in an iterative manner, with each iteration being tested thoroughly.

7. Testing throughout the iteration: The developer should test the application at each iteration to ensure that it meets the requirements and functions as expected. This includes unit testing, integration testing, and acceptance testing.

8. Execution and revision: Once the application has been developed and tested, the developer should execute it and revise it if necessary. This includes identifying any bugs or issues and fixing them before deployment.

9. Deployment of the application and completion: The final step is to deploy the application to the production environment. This includes setting up the necessary infrastructure, configuring the deployment environment, and setting up a continuous integration, continuous deployment (CI/CD), pipeline. Once the application has been deployed, the developer should monitor it regularly to ensure that it is functioning properly.

# Software Installation

Software installation is a crucial part of the software deployment and operation process. It involves the process of installing software onto a computer or server so that it can be used by end-users. Proper installation of the software is essential for its functionality, performance, and security. In this section, we will explore the different aspects of software installation, including the types of software installation, the components of the installation process, and the challenges that developers may face during installation.

## **3.1 Android Studio**

Android Studio is a development environment for creating Android applications. It is an official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. Android Studio provides developers with tools to create visually appealing and functional applications, and it includes an emulator for testing apps on virtual devices. It also has support for Dart, Java, Kotlin, and C++ programming languages, as well as tools for debugging, testing, and optimizing application performance. The latest version is Android Studio Giraffe | 2022.3.1, which was released on September 25, 2023.

**3.1.1 Windows Installment**

System Requirements for Windows:

**A screenshot of a computer

Description automatically generated**

Figure 2: Windows System Requirements

**Download:**

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

To install Android Studio on Windows, follow these steps:

If you downloaded an .exe file (recommended), double-click to launch it.

If you downloaded a .zip file:

1. Unpack the .zip.
2. Copy the android-studio folder into your Program Files folder.
3. Open the android-studio > bin folder.
4. Launch studio64.exe (for 64-bit machines) or studio.exe (for 32-bit machines).
5. Follow the Setup Wizard in Android Studio and install any recommended SDK packages.

The following video shows each step of the setup procedure for the recommended .exe download:

**Installation Videos:**

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

A screenshot of a computer

Description automatically generated

Figure 3: Windows Example

**3.1.2 Mac Installment**

System Requirements for Mac: A screenshot of a computer

Description automatically generated

Figure 4: Mac System Requirements

**Download:**

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

To install Android Studio on your Mac, follow these steps:

1. Launch the Android Studio DMG file.
2. Drag and drop Android Studio into the Applications folder, then launch Android Studio.
3. Choose whether to import previous Android Studio settings, then click OK.
4. Complete the Android Studio Setup Wizard, which includes downloading the Android SDK components that are required for development.

**Installation Videos:**

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

A screenshot of a computer

Description automatically generated

Figure 5: Mac Download Example

## **3.2 Flutter and Dart**

Flutter is a framework for developing mobile applications that is open source and was created by Google. It allows developers to build high-performance, high-fidelity apps for mobile, web, and desktop, all from a single codebase. Flutter uses the Dart programming language, which was also created by Google. Dart is a statically typed programming language that compiles native code, allowing Flutter apps to run smoothly on different platforms. Together, Flutter and Dart provide developers with a powerful set of tools to create beautiful and responsive apps that can run seamlessly on multiple devices.

### **3.2.1 Windows Installation**

System Requirements for Windows:

* Operating Systems: Windows 10 or later (64-bit), x86-64 based.
* Disk Space: 2.5 GB (does not include disk space for IDE/tools).
* Tools: Flutter depends on these tools being available in your environment.
  + Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10)
  + Git for Windows 2.x, with the Use Git from the Windows Command Prompt option.

**Download:**

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

To install Flutter software development kit (SDK) Studio on your Windows, follow these steps:

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

2. Extract the zip file and place the contained flutter in the desired installation location for the Flutter SDK (for example, *%USERPROFILE%\flutter, D:\dev\flutter*).

3. Update your path: If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the PATH environment variable:

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.

4. Run flutter doctor: From a console window that has the Flutter directory in the path (see above), run the following command to see if there are any platform dependencies you need to complete the setup: *C:\src\flutter>flutter doctor*

A screenshot of a computer

Description automatically generated

Figure 6: Flutter Download Example (Windows)

### **3.2.2 Mac Installation**

System Requirements for Windows:

* Operating Systems: macOS, version 10.14 (Mojave) or later.
* Disk Space: 2.8 GB (does not include disk space for IDE/tools).
* Tools: Flutter uses git for installation and upgrade.

**Download:**

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

To install Flutter SDK Studio on your Mac, follow these steps:

1. Download the installation bundle to get the latest stable release of the Flutter SDK from the above link.

2. Extract the file in the desired location. For example:

*cd ~/development*

unzip ~/Downloads/flutter\_macos\_3.13.8-stable.zip

3. Open (or create) the rc file for your shell. Typing echo $SHELL in your Terminal tells you which shell you’re using. If you’re using Bash, edit $HOME/.bash\_profile or $HOME/.bashrc. If you’re using Z shell, edit $HOME/.zshrc. If you’re using a different shell, the file path and filename will be different on your machine.

Add the following line and change [PATH\_OF\_FLUTTER\_GIT\_DIRECTORY] to be the path of your clone of the Flutter git repo:

export PATH="$PATH:[PATH\_OF\_FLUTTER\_GIT\_DIRECTORY]/bin"

Run source $HOME/.<rc file> to refresh the current window, or open a new terminal window to automatically source the file.

Verify that the flutter/bin directory is now in your PATH by running: echo $PATH

4.. Add the flutter tool to your path: export PATH="*$PATH:`pwd`/flutter/bin"*

5. Run flutter doctor: *flutter doctor*

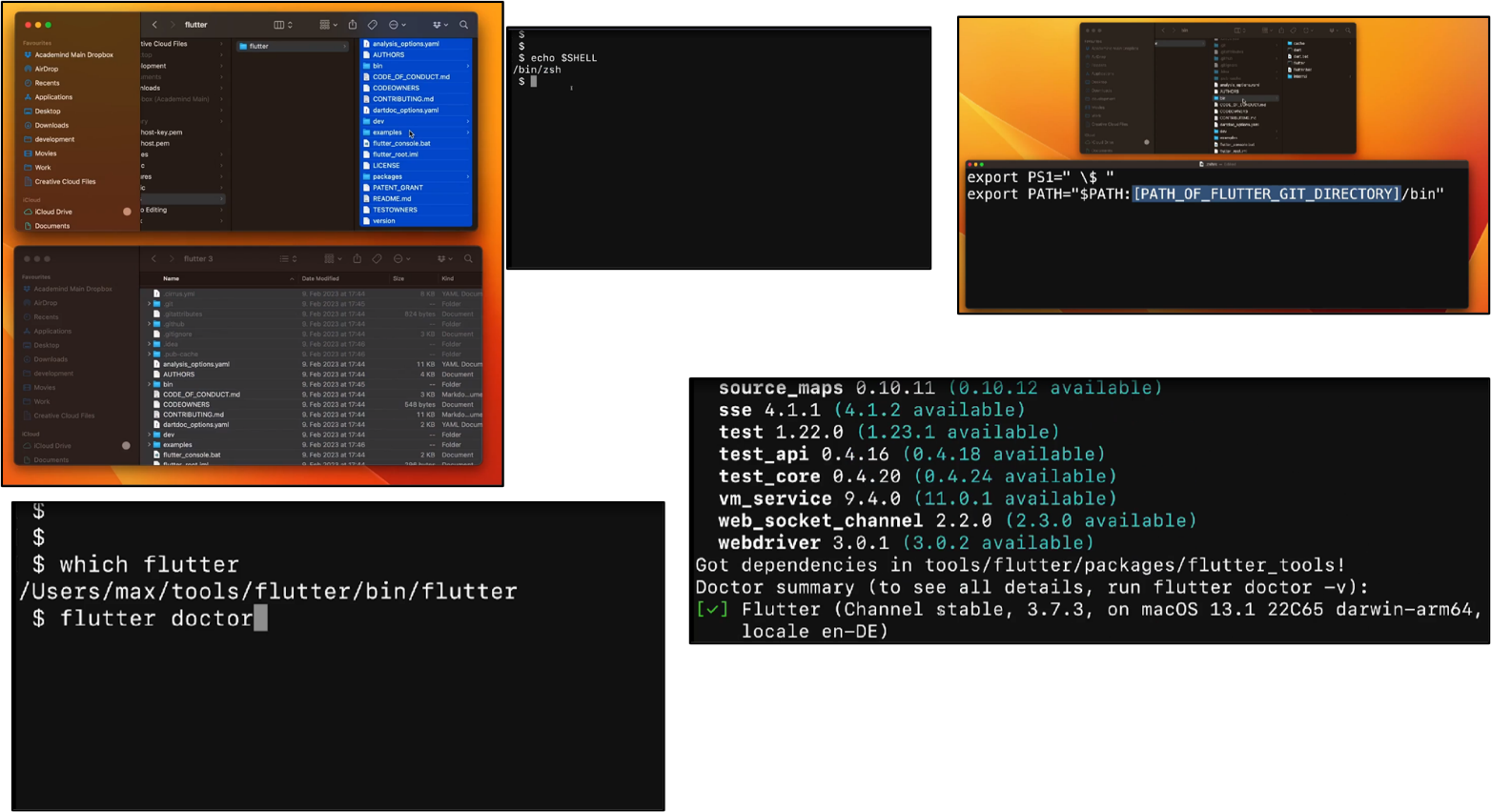


Figure 7: Flutter Download Example (Mac)

## **3.3 Android Emulator**

Here are the steps to creating and running an Android emulator:

1. Open Android Studio and go to the "Android Virtual Device (AVD) Manager" by clicking on the "AVD Manager" icon in the toolbar or by selecting "AVD Manager" from the "Tools" menu.

2. Click on the "Create Virtual Device" button.

3. Choose the device definition you want to use by selecting a category and then selecting a device from the list. You can also create a custom device definition by clicking on the "New Hardware Profile" button.

4. Choose the system image you want to use by selecting a tab and then selecting an image from the list. You can also download additional system images by clicking on the "Download" link.

5. Click on the "Next" button.

6. Configure the settings for the emulator, such as the name of the emulator, the orientation, and the startup orientation.

7. Click on the "Finish" button.

8. In the toolbar, select the AVD that you want to run your app on from the target device menu.

9. Click Run. The emulator might take a minute or so to launch for the first time.

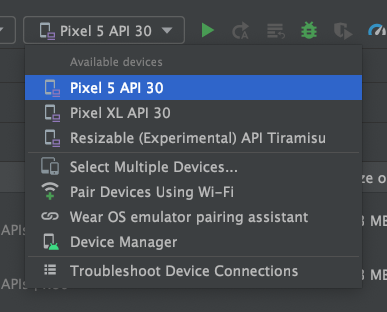


Figure 8: Selecting an emulator

To interact with the Android emulator, you can use your computer mouse pointer to simulate your finger on the touchscreen. This allows you to select menu items, input fields, and click buttons and controls.   
Figure 12 shows common features with their description.

A screenshot of a computer

Description automatically generated

Figure 9: Flutter features

## **3.4 Test Your Development Environment**

It is recommended to test your Android app on a physical device before releasing it to users. This guide provides instructions on how to prepare both your development environment and Android device for testing and debugging using an Android Debug Bridge (ADB) connection.

Before you can start debugging on your device, decide whether you want to connect to the device using a USB cable or Wi-Fi. Then do the following:

1. On the device, open the Settings app, select Developer options, and then enable USB debugging (if applicable).
2. Set up your system to detect your device:

macOS: No additional configuration required.

Windows: Install a USB driver for ADB (if applicable).

**3.4.1 Connect to your device using USB**

When you're set up and plugged in over USB, click Run in Android Studio to build and run your app on the device.

You can also use *adb* to issue commands, as follows:

* Verify that your device is connected by running the *adb devices* command from your *android\_sdk/platform-tools/* directory. If connected, you'll see the device listed.
* Issue any *adb* command with the *-d* flag to target your device.

**3.4.2 Connect to your device using Wi-Fi**

Android 11 and higher supports deploying and debugging your app wirelessly from your workstation via Android Debug Bridge (ADB).

To connect to your device, follow these steps:

1. Open Android Studio and select **Pair Devices Using Wi-Fi** from the run configurations menu (see Figure 13). The Pair devices over Wi-Fi dialog appears, as shown in Figure 14.

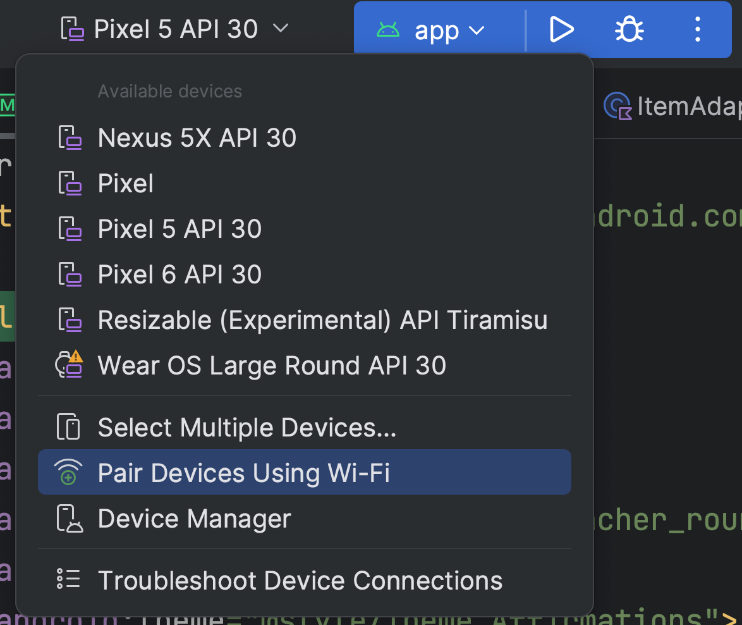


Figure 10: Device pairing using Wi-Fi

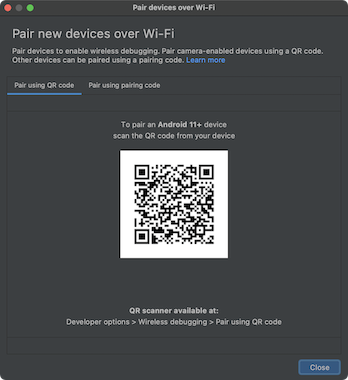


Figure 11: Pair new device QR code example

1. Enable developer options on your device.
2. Enable debugging over Wi-Fi on your device.

Screenshot of a pixel phone showing the Wireless debugging
      systems setting.

Figure 12: Wireless debugging

1. Tap **Wireless debugging** and pair your device:

* To pair your device with a QR code, select **Pair device with QR code** and scan the QR code, shown in figure 14.
* To pair your device with a pairing code, select **Pair device with pairing code** from the **Pair new devices over Wi-Fi** dialog. On your device, select Pair using pairing code. A six-digit code appears. Once your device appears on the Pair devices over Wi-Fi window, enter the six-digit code shown on your device and select **Pair**.

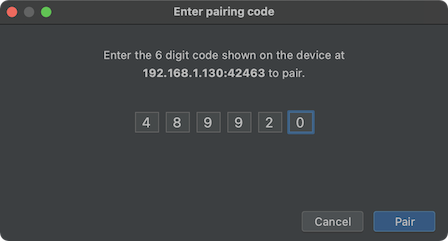


Figure 13: Pair device code example

1. After pairing, you can attempt to deploy your app to your device.

## **3.5 GitHub Desktop**

To download GitHub Desktop, follow these steps:

1. Go to the official GitHub Desktop website at <https://desktop.github.com>.
2. Click on the "Download for [your operating system]" button (Windows or Mac)
3. Once the download is complete, double-click on the downloaded file to start the installation process.
4. Follow the on-screen instructions to complete the installation.
5. Once installed, launch GitHub Desktop, and sign in with your GitHub account.
6. You can now use GitHub Desktop to clone, commit, and push code to your GitHub repositories.

A screenshot of a computer

Description automatically generated

Figure 14: GitHub Desktop download website

## **3.6 Common Challenges**

Common challenges that may arise when installing software like Flutter, Android Studio, and plugins include compatibility issues with the operating system, insufficient disk space, incorrect or incomplete installation of required software dependencies such as Java Development Kit (JDK), outdated graphics drivers, and conflicting software installations. Other challenges can include slow download speeds, interrupted installations due to network issues or power outages, and configuration errors during setup.

# Prepare Mobile Application for Use

In the previous section, you successfully downloaded and installed the required software, which includes Android Studio, Flutter & Dart, and GitHub Desktop. Also, the development environment was established by setting up the Android emulator. Upon the emulator's successful deployment, it is imperative to configure the mobile application to ensure seamless functionality, whether through your virtual or physical device simulation. The purpose of this section is to provide you with guidance on duplicating the code repository on your system.

## **4.1 Cloning GitHub Repository**

To replicate the GitHub repository on your system, please follow the steps outlined below:

1. Navigate to <https://github.com/umgc/fall2023>

A screenshot of a computer

Description automatically generated

Figure 15: SWEN 670 GitHub home page

1. Above the list of files, click Code.

A screenshot of a computer

Description automatically generated

Figure 16: Local file for cloning

1. To clone the repository using HTTPS, under "HTTPS", click the copy icon. See Figure 20 for details.

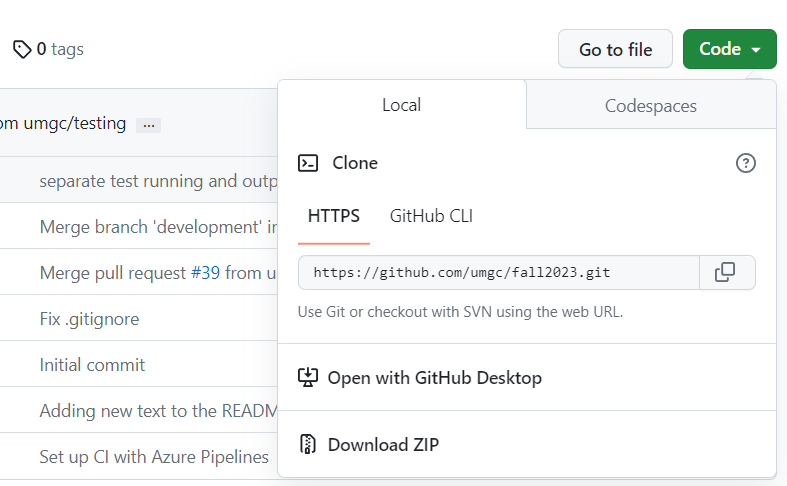


Figure 17: Location of copy icon

## **4.2 Run the Application**

To run an application using Android Studio, follow these steps:

1. Open Android Studio
2. Open Project from Version Control (File > New > Project from Version Control). See Figure 21.
3. Copy the HTTPS repository into URL. See Figure 22.
4. Click Clone.

A screenshot of a computer

Description automatically generated

Figure 18: Path to “Project from Version Control”

**A screenshot of a computer

Description automatically generated**

Figure 19: Repository URL clone screen

## **4.3 Using the .env File**

The .env file will store environmental (and other secret) variables used in the development and execution of the application functions. See the *temp-env* file with variables to fill in. For example, the ChatGPT access token variable will be stored here to access the ChatGPT API (used in various audio transcription related features). Likewise, the AWS S3 and Rekognition accessKeys are also stored in this file. This .env file IS NOT to be checked into the source code repo. Before committing code, please ensure that your .env file is not being pushed into the branches; similarly, ensure that the. gitignore file include  *\*.env* so that this .env file will never be uploaded to the repository.

## **4.4 Getting the OpenAI ChatGPT API Key**

The following outlines setting up a new key and adding it to the .env created with the instructions above.

1. Go to [https://openai.com](https://openai.com/) and Login or Sign up for a new account.
2. Click on API
3. At top right, click on Personal and then View API Keys menu item.
4. Click "+ Create new secret key" button
5. Give the key a name, relating to CogniOpen or SWEN670
6. The new secret key will appear in a text box with the copy button next to it.
7. Copy the key and paste it into your .env file.
8. Save the key somewhere like OneNote where it will be safe, you cannot retrieve this key once you close this dialog. You can, however, make more.
9. Click done.

## **Getting the AWS Access Key**

1. Go to <https://aws.amazon.com/> and select "Sign In" to sign in or create a new "Free Tier" account
2. Once you are at the console home, navigate to your account and then "Security credentials"
3. Select "Users" and then "Create user"
4. Give the user a name. Select "next" to move to "Step 2: Set permissions"
5. Select "attach policies directly". Then select the "AdministratorAccess-Amplify", "AmazonRekognitionFullAccess", "AmazonKinesisFullAccess", "AmazonS3FullAccess", and "AmazonTranscribeFullAccess" permission policies.
6. Select "Create user" in "Step 3: Review and create". The new secret key will appear in a text box with copy button next to it.
7. Copy the accesskey and paste it into your .env file.
8. Copy the secretKey and paste it into your .env file.
9. Save the key somewhere like OneNote where it will be safe, you cannot retrieve this key once you close this dialog. You can, however, make more.
10. Click done.

The Amazon Free tier ought to have enough space and video processing requests to complete development and testing. Just be aware of video fidelity and length when recording. Requests are pretty cheap afterwards, but the dollar amount can pile up if unaware.

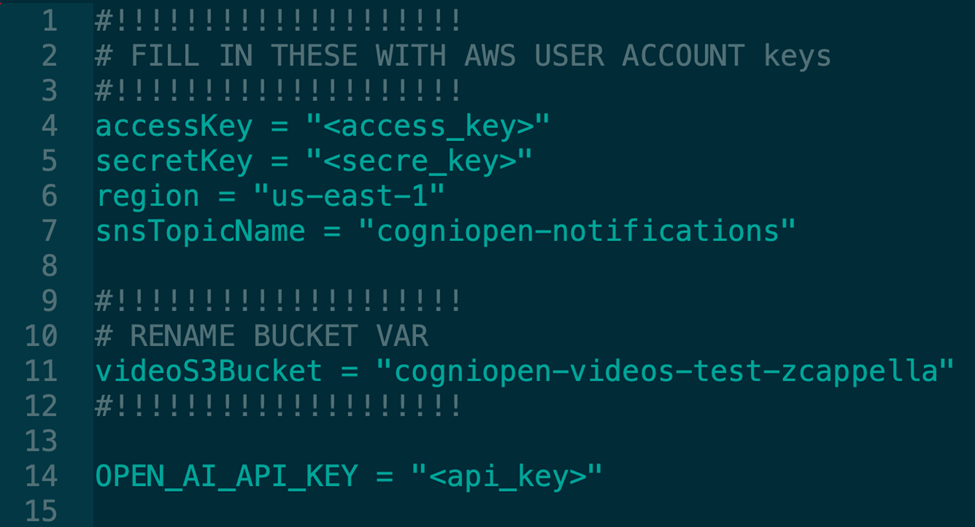


Figure 20: API Key Example

# Test the Mobile Application

## **5.1 Automation Testing**

Automation testing is the practice of writing code or leveraging third-party tools to automatically exercise components of an application. In the context of a Dart and Flutter application, this can be accomplished in the form of unit testing, widget testing, or integration testing. Unit testing provides a mechanism for testing a single component, function, algorithm, or class. Widget testing exercises the functionality of a single widget. Finally, integration tests combine the two methods to test larger application components.

Automated testing is used within the CogniOpen application, utilizing the baked in functionality of the Flutter testing suite of tools. To run the automated tests, perform the following actions:

1. Clone the repository.
2. Change directories to the folder where the code repository is checked out.
3. Navigate to the application folder – cogniopenapp.
4. Run flutter clean to clean the application folder.
5. Run flutter pub get to download all the application dependencies.
6. Run flutter test to run through the suite of the automated tests.

The output of flutter test will show one of the two outputs:

A blue background with white text

Description automatically generated

Figure 21: Pass test screen

If all of the tests are successful, flutter will display the above output.

A screenshot of a computer program

Description automatically generated

Figure 22: Flutter test screen

If there are errors with the unit tests, flutter will present the user with the issues that need to get resolved. Additionally, when code is submitted for a pull request in GitHub, each automated test is executed in a pipeline that will block merging if there are errors. Below is a picture of the output of a successful run.

A screenshot of a computer program

Description automatically generated

Figure 23: Successful Flutter test screen

## **5.2 Manual Testing**

The manual testing process involves a human tester who follows predefined test cases to ensure that the software application meets the business requirements and performs as expected. It is a time-consuming and meticulous process that requires attention to detail and a good understanding of the software application's functionality. The expected results for the manual testing provide a comprehensive overview of how the CogniOpen application will be utilized, including its core functions and the components that drive its functionality. The CogniOpen application is designed to serve as a comprehensive memory wellness solution for individuals dealing with cognitive impairment. Figure 27 ensures that the application meets the highest standards of quality and performance.

|  |  |  |
| --- | --- | --- |
| General Testing | iPhone Results | Android Results |
| 1. **User Login:** Users can securely log in to the application using their registered credentials, ensuring data privacy and security. |  |  |
| 1. **User Application Registration:** New users can create accounts to access the application's features. |  |  |
| 1. **Home Interface:** The central hub where users can access various features and navigate through the application's features. |  |  |
| 1. **Virtual Assistant Interface:** A conversational interface designed to assist users with their memory wellness activities, answer questions, and provide support. |  |  |
| 1. **Record Audio:** Users can record audio content, such as voice notes or reminders, to aid in memory retention. |  |  |
| 1. **Record Video:** Video recording functionality for capturing important moments or instructions. |  |  |
| 1. **Photo Gallery:** A gallery feature that allows users to organize and browse photos, enhancing memory recall through visual cues. |  |  |
| 1. **Video Gallery:** A gallery feature for storing and playing back videos, promoting engagement, and reminiscence. |  |  |
| 1. **Audio Gallery:** A gallery for organizing and replaying audio recordings, assisting users in remembering spoken information. |  |  |
| 1. **Recent Conversations:** A repository of commonly asked questions and answers to support users in memory-related inquiries. |  |  |
| 1. **Menu Interface:** Navigation and settings options to customize the user experience. |  |  |
| 1. **Profile Interface:** User profile management for updating personal information and preferences. |  |  |
| 1. **Guided Tour:** A feature that guides users through the application's functionalities and provides an interactive tutorial. |  |  |

Table 3: Manual testing expected results

# Troubleshooting

## **6.1 Troubleshoot Android Studio**

If you're experiencing problems with Android Studio, there are several troubleshooting steps you can take to resolve them. Here's a list of troubleshooting steps for Android Studio:

1. Check your internet connection - A stable internet connection is crucial to work on Android Studio properly. If you're experiencing issues, check your internet connection to ensure its strong and stable. You can also try disconnecting and reconnecting your internet to see if that solves the problem.
2. Restart Android Studio - Sometimes, minor issues can be fixed by simply restarting the program. To do this, close Android Studio completely and then reopen it.
3. Update Android Studio - Make sure you have the latest version of Android Studio installed. Updates often include bug fixes and new features that can help to resolve issues. To check for updates, go to Help > Check for Updates.
4. Clear cache and data - Over time, cached data can build up in Android Studio, which can cause issues. To clear the cache and data, go to Settings > Apps > Android Studio > Storage, then select Clear Cache and Clear Data.
5. Invalidate caches and restart - Cached data can also cause issues with Android Studio. To clear these cached data, go to File > Invalidate caches / Restart.
6. Check for conflicting plugins - If you have any plugins installed, they may be causing conflicts with Android Studio. Try removing any plugins that you're not using or that you suspect may be causing issues.
7. Reset Android Studio to default settings - If you've tried all the above steps and are still experiencing issues, you may need to reset Android Studio to its default settings. To do this, go to File > Manage IDE Settings > Restore Default Settings. Keep in mind that this will remove any customizations you've made to Android Studio.

## **6.2 Troubleshoot Android Emulator**

Here's a list of troubleshooting steps for the Android Emulator:

1. Emulator is running slow - if the emulator is running slow, try increasing the amount of RAM allocated to it. You can do this by going to Settings > Advanced > Memory and increasing the RAM size.
2. Emulator not responding - If the emulator is not responding, try closing it and reopening it. If that doesn't work, try restarting Android Studio or your computer. You can also try disabling any plugins that may be causing conflicts with the emulator.
3. Emulator freezing - If the emulator freezes, try reducing the screen resolution or turning off animations in the developer options. You can also try updating the graphics card driver or using a different emulator.
4. Out of memory error - If you receive an out of memory error, try increasing the heap size. You can do this by going to the Run Configuration and adding the -Xmx512m flag to the VM options.
5. Check for conflicts with antivirus software - some antivirus software can interfere with the emulator. Try disabling your antivirus temporarily to see if that solves the problem.
6. Wipe data - if all else fails, you can try wiping the data from the emulator. To do this, go to Settings > System > Advanced > Reset options > Erase all data (factory reset).