



TECHNOLOGY SOLUTIONS

CogniOpen Software Application

Deployment and Operations Guide

Project Name: CogniOpen Software Application
Document: Deployment and Operations Guide (Runbook)
Version: 2.0
Date: November 7, 2023
Prepared By: Team A

Team Lead / Project Manager:
Vincent Galeano
Dream Team Technology Solutions (DTTS)
vgaleano1@student.umgc.edu

Client:
Dr. Mir Assadullah
University of Maryland Global Campus (UMGC)
mir.assadullah@faculty.umgc.edu

Leads Sign-off Sheet

Key Reviewer	Version	Date	Signature
Vincent Galeano Team Lead / Project Manger	1.0	28/Oct/2023	<i>Vincent Galeano</i>
Kavon Johnson Lead Technical Writer	1.0	28/Oct/2023	<i>Kavon Johnson</i>
Zach Bowman Lead Business Analyst	1.0	28/Oct/2023	<i>Zach Bowman</i>
David Bright Architect / Lead Software Developer	1.0	28/Oct/2023	<i>David Bright</i>
Juan Torres-Chardon Lead UI/UX Designer	1.0	28/Oct/2023	<i>Juan Torres - Chardon</i>
Laura Hamann Lead Test Engineer	1.0	28/Oct/2023	<i>Laura Hamann</i>
Vincent Galeano Team Lead / Project Manger	2.0	07/Nov/2023	<i>Vincent Galeano</i>
Kavon Johnson Lead Technical Writer	2.0	07/Nov/2023	<i>Kavon Johnson</i>
Zach Bowman Lead Business Analyst	2.0	07/Nov/2023	<i>Zach Bowman</i>
David Bright Architect / Lead Software Developer	2.0	07/Nov/2023	<i>David Bright</i>
Juan Torres-Chardon Lead UI/UX Designer	2.0	07/Nov/2023	<i>Juan Torres - Chardon</i>
Laura Hamann Lead Test Engineer	2.0	07/Nov/2023	<i>Laura Hamann</i>

Revision History

Date	Version	Description	Author
25/Sep/2023	0.1	Document created	DTTS
28/Oct/2023	1.0	Milestone 3 version	DTTS
07/Nov/2023	2.0	Milestone 4 version	DTTS

Table of Contents

<i>Leads Sign-off Sheet</i>	2
<i>Revision History</i>	3
<i>Table of Contents</i>	4
<i>List of Figures & Tables</i>	6
1 Introduction	7
1.1 Purpose	7
1.2 Intended Audience	7
1.3 Technical Project Stakeholders	7
1.4 Document Conventions & Organization	8
1.4.1 Document Conventions	8
1.4.1.1 Terminology	8
1.4.1.2 Formatting	8
1.4.1.3 Lists	8
1.4.1.4 Hyperlinks.....	8
1.5 Project Document Suite	8
1.6 References	9
1.7 Terms, Abbreviations, & Acronyms	10
2 Development Team Workflow	11
2.1 Source Control	11
2.1.1 Account Setup	11
2.1.2 Repositories	11
2.2 Branching Strategy	11
3 Mobile Application	12
3.1 Philosophy.....	12
3.2 Features	12
3.3 Packages.....	12
4 Software Installation	13
4.1 Machine Setup	13
4.1.1 Get the Flutter SDK	13
4.1.2 Android setup.....	14
4.1.3 Set up your Android device	14
4.1.4 Set up the Android emulator.....	15
4.1.5 Agree to Android Licenses	15

4.1.6	Windows setup	16
4.2	Set up a text editor (VS Code).....	16
4.2.1	Install VS Code	16
4.2.2	Install the VS Code Flutter extension.....	16
4.2.3	Validate your VS Code setup	17
5	Preparations for Use	18
5.1	Source Control	18
5.1.1	Cloning a repository.....	18
5.2	Setup the .env file	18
5.2.1	Getting the AWS access key	18
5.3	Compiling	19
6	Deployment	20
6.1	Deployment.....	20
6.1.1	Emulator.....	20
6.1.2	Physical phone	20
7	Automation Testing	26
7.1	GitHub Actions	26
7.1.1	Setup	26
7.1.2	Execution	26
7.1.3	Results	27
8	Troubleshooting	29
8.1	Not recording audio in an emulator	29
8.2	Not recording video in an emulator	29
8.3	Database not loading	31

List of Figures & Tables

Table 1: Technical Project Stakeholders	8
Table 2: Project Document Suite.....	9
Table 3: Terms, abbreviations, and acronyms	10
Figure 1: Flutter Doctor – needs improvement.....	13
Figure 2: Flutter Doctor - working	16
Figure 3: Installing Flutter.....	17
Figure 4: Cloning a Git Repository.....	18
Figure 5: Turn on Developer Mode	21
Figure 6: Enabled Developer Mode	21
Figure 7: Developer Options	22
Figure 8: Allow USB Debugging.....	23
Figure 9: Check Allow USB debugging.....	24
Figure 10: Changing device	24
Figure 11: Changing device.....	25
Figure 12: Device changed	25
Figure 13: Automated Testing Configuration File	26
Figure 14: Test Results.....	27
Figure 15: Test Failed	27
Figure 16: Detailed Test Results	28
Figure 17: Enable Microphone.....	29
Figure 18: Device “Advanced Settings”	30
Figure 19: Error screen in flutter	31
Figure 20: Wipe Data of virtual device	31

1 Introduction

1.1 Purpose

Deployment and Operations Guide (Runbook) explains how CogniOpen is brought into one's computing environment. CogniOpen's dependencies are comprehensively listed. The document describes exactly the steps that one must complete to connect to remote cloud services. Some of these features require access keys. The runbook delineates how to obtain the codes. Mobile app installation, along with any quirks, is clearly presented.

1.2 Intended Audience

This document is meant to inform people who want to use CogniOpen. It does not postulate development philosophies. Nor does it trumpet so-called "industry best practices" to drive sales. Somebody who wants to develop, test, operate, document, or extend CogniOpen should complete the actions described in the order presented. The reader should be computer literate. However, he/she may not have any experience with web services or databases.

1.3 Technical Project Stakeholders

The successful development and deployment of the CogniOpen Software Application involves collaboration among various technical project stakeholders. The roles and responsibilities of these stakeholders are defined as follows:

Role	Responsibility
Client	Dr. Mir Assadullah, representing the client's interests and requirements for the CogniOpen Software Application.
Team Lead / Project Manager	Vincent Galeano, who oversees project management and team coordination.
Lead Technical Writer	Kavon Johnson, responsible for technical documentation, including this Programmer Guide.
Lead Business Analyst	Zach Bowman, involved in requirements analysis and business processes.
Architect / Lead Software Developer	David Bright, responsible for software architecture and development leadership.
Lead UI/UX Designer	Juan Torres-Chardon, focuses on user interface and user experience design.

Role	Responsibility
Lead Test Engineer	Laura Hamann, responsible for software testing and quality assurance.

Table 1: Technical Project Stakeholders

1.4 Document Conventions & Organization

Throughout this guide, we adhere to consistent terminology and naming conventions to enhance clarity and comprehension. Key terms and their definitions are provided in Section 1.7, "Terms, Abbreviations, & Acronyms."

1.4.1 Document Conventions

1.4.1.1 Terminology

Throughout this guide, we adhere to consistent terminology and naming conventions to enhance clarity and comprehension. Key terms and their definitions are provided in Section 1.7, "Terms, Abbreviations, & Acronyms."

1.4.1.2 Formatting

Text formatting is used consistently to convey information:

Bold Text: Used for emphasizing important points, section headings, and UI elements.

Italic Text: Employed for highlighting variable names, file paths, and placeholders.

`Monospace Font:` Indicates code snippets, file names, and inline code references.

1.4.1.3 Lists

Lists are presented in two formats:

Numbered Lists: Used for sequential steps or procedures.

Bulleted Lists: Used for items without a specific order.

1.4.1.4 Hyperlinks

Hyperlinks to external resources, documents, or web pages are provided for additional information and context. They are displayed in blue and are clickable when viewed digitally.

1.5 Project Document Suite

This TDD is part of a suite of project documents that collectively provide comprehensive project documentation. The suite includes:

Document	Version	Date
Project Plan (PP)	4.0	07/Nov/2023
Software Requirements Specification (SRS)	4.0	07/Nov/2023
Technical Design Document (TDD)	3.0	07/Nov/2023

Document	Version	Date
Test Plan (TP)	3.0	07/Nov/2023
Programmer Guide (PG)	2.0	07/Nov/2023
Deployment and Operations Guide (Runbook)	2.0	07/Nov/2023
User Guide (UG)	1.0	07/Nov/2023
Test Report (TR)	1.0	07/Nov/2023

Table 2: Project Document Suite

Additional project documents, including the UG and TR, will be developed as the project progresses.

1.6 References

Anas, M. (2023, July). *dart_openai 4.0.0*. https://pub.dev/packages/dart_openai

Arnott, E. (2023, June 23). *How to create a runbook template for devops (with examples)*. <https://www.blameless.com/blog/runbook-template-devops>

Blackberry. (2019, March 7). *Celebrating 35 years of Blackberry an Icon of the Past and Present Future*. <https://blogs.blackberry.com/en/2019/03/celebrating-35-years-of-blackberry-an-icon-of-the-past-present-future>

GitHub. (n.d.). GitHub Actions Documentation - GitHub Docs. Docs.github.com. <https://docs.github.com/en/actions>

Google. (n.d.). *Building user interfaces with Flutter*. <https://docs.flutter.dev/ui>

PagerDuty. (n.d.). *What is a runbook?*. <https://www.pagerduty.com/resources/learn/what-is-a-runbook/>

Roux, A. (n.d.). *sqflite 2.3.0*. <https://pub.dev/packages/sqflite>

SQLite. (2023, July). *What is SQLite?* <https://www.sqlite.org/index.html>

UMGC. (2023). *Previous projects*. SWEN 670: Software Engineering Capstone, University of Maryland Global Campus (UMGC). <https://umgc-cappms.azurewebsites.net/previousprojects>

Walker, A. (2023, March). *Deployment runbooks (aka runsheets) explained*. <https://www.enov8.com/blog/deployment-runbooks-aka-runsheets-explained/>

1.7 Terms, Abbreviations, & Acronyms

Term	Definition
App	Application
AI	Artificial Intelligence
DDL	Data Definition Language
DML	Data Manipulation Language
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
NLP	Natural Language Processing
RDBMS	Relational Database Management System
Runbook	Deployment and Operations Guide
SQL	Structured Query Language
VM	Virtual Machine

Table 3: Terms, abbreviations, and acronyms

2 Development Team Workflow

This section explains where the code is stored, how to access it to make changes, and how to control the changes being made to the main and development branches.

2.1 Source Control

Getting set up in GitHub, how to access the branch repositories (what they mean). That kind of stuff.

2.1.1 Account Setup

1. You must have an existing GitHub account to commit changes.
 - a. Follow the instructions at this page to create one:
<https://github.com/signup>
2. Contact the class professor for an invite to UMGC's GitHub projects. The professor will need your student email address and GitHub username.
3. After you get access, the code can be reached at the links below.

2.1.2 Repositories

1. Main repository can be found at: <https://github.com/umgc/fall2023>.
2. The application code can be found at:
<https://github.com/umgc/fall2023/tree/development/cogniopenapp/lib>
3. The automation tests can be found at:
<https://github.com/umgc/fall2023/tree/development/cogniopenapp/test>

2.2 Branching Strategy

Main is the clean version of what is being pushed as a stable product. Development is used as the intermediary branch between active development and stable releases. New features are branched from “development”, and code that has passed local testing is submitted via pull requests to development.

3 Mobile Application

Smartphones have revolutionized the way the world does business, socializes, and stays healthy. CogniOpen is a facet of this amazing technical innovation. It is challenging to program mobile software technology, but necessary to enable the convenience of anywhere, anytime, anyhow availability.

3.1 Philosophy

Building and maintaining CogniOpen is difficult but very rewarding. Flutter applications (Apps) are comprised of modular units called “Widgets”. A widget has a definition of which variables ought to be displayed where on the screen. Specific values are calculated from instances of objects evaluated at runtime by the Virtual Machine (VM). There is a rich widget repository for many common tasks, such as location, camera, microphone, and buttons. Human actions, for instance tapping the screen, are captured with event-based callback functions.

3.2 Features

CogniOpen has four pillars of computation. They are audio recording, photographs, videos, and Natural Language Processing (NLP). The system receives input and stores input from the smartphone’s screen, file system, camera, microphone, and Global Positioning System (GPS). Artificial Intelligence (AI) distills the data for its essence. That information is put into tags received via an Internet connection. It is persisted in a local Relational Database Management System (RDBMS).

CogniOpen dynamically constructs Structured Query Language (SQL) commands to interact with the RDBMS. SQL commands that create tables and specify their types of columns execute when the application is initializing. This kind of statement gets classified as Data Definition Language (DDL). During CogniOpen being used, the system retrieves and modifies a table’s cellular content. This form of SQL is classified as Data Manipulation Language (DML).

3.3 Packages

Production and test databases come courtesy of `sqlite` and `sqlite_common_ffi`, respectively. They are implementations of the SQLite specification. It is a lightweight and flexible tool used in innumerable computer applications. NLP arrives from `dart_openai`. It makes OpenAI Hypertext Transfer Protocol (HTTP) calls.

Dart Packages are listed as dependencies in CogniOpen’s metadata. Working with the system, one must expect to update to the newest versions of these wonderful libraries. That poses both an opportunity and a risk. New abilities provided by third-party authors could either improve, or break, CogniOpen. Probably it will be required. This is because computer hardware inventions will inevitably push the proverbial envelope.

4 Software Installation

4.1 Machine Setup

To get started with Flutter, follow the detailed installation for Windows OS provided in this wiki here.

4.1.1 Get the Flutter SDK

1. Download the [Flutter SDK](#). For other release channels, and older builds, check out the [SDK archive](#).
2. Extract the .zip file to a desired installation location (avoid the C:\Program Files directory; prefer %USERPROFILE\flutter). It is about 2GB, so have some space available on the drive of choice.
3. Update the PATH environmental variable to run Flutter commands from the Windows console. (append ";{installation_location}\flutter\bin" to the %PATH% environmental variable).
4. Run "flutter doctor" from the console window to see if there are any platform dependencies needed to complete setup. This command checks your environment and displays a report of the status of your Flutter installation. Check the output carefully for other software you might need to install or further tasks to perform. Once you have installed any missing dependencies, you can run the flutter doctor command again to verify that you've set everything up correctly.

```
C:\Users\david>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.13.1, on Microsoft Windows [Version 10.0.19045.3324], locale en-US)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[X] Android toolchain - develop for Android devices
    X Unable to locate Android SDK.
      Install Android Studio from: https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK components.
      (or visit https://flutter.dev/docs/get-started/install/windows#android-setup for detailed instructions).
      If the Android SDK has been installed to a custom location, please use
      `flutter config --android-sdk` to update to that location.

[✓] Chrome - develop for the web
[!] Visual Studio - develop Windows apps (Visual Studio Community 2019 16.11.0)
    X Visual Studio is missing necessary components. Please re-run the Visual Studio installer for the "Desktop
      development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
          - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[!] Android Studio (not installed)
[✓] VS Code (version 1.81.1)
[✓] Connected device (3 available)
[✓] Network resources

! Doctor found issues in 3 categories.
```

Figure 1: Flutter Doctor – needs improvement

4.1.2 Android setup

1. Download and install [Android Studio](#).
2. Start Android Studio and go through the 'Android Studio Setup Wizard'. This installs the latest Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools, which are required by Flutter when developing for Android. Alternatively, open Android Studio and navigate to "SDK Manager" to manually configure the Android Studio.
 - For latest Android SDK, navigate to **Languages & Frameworks > Android SDK**. In the **SDK Platforms**, select "Android API 34"
 - For Android SDK Command-line Tools, **Languages & Frameworks > Android SDK**. in the **SDK Tools**, select "Android SDK Command-line Tools (latest)"
 - For Android SDK Build-Tools, **Languages & Frameworks > Android SDK**. in the **SDK Tools**, select "Android SDK Build-Tools 34"
 - Once all are selected, select "Apply" and follow any on-screen steps/guides.
3. Run "`flutter doctor`" to confirm that the Flutter SDK has located your installation of Android Studio. If Flutter cannot locate it, run `flutter config --android-studio-dir=<directory>` to set the directory that Android Studio is installed to.

4.1.3 Set up your Android device

To prepare to run and test your Flutter app on an Android device, you need an Android device running Android 4.1 (API level 16) or higher.

1. Enable **Developer options** and **USB debugging** on your device. Detailed instructions are available in the [Android documentation](#).
2. [Optional] To leverage wireless debugging, enable **Wireless debugging** on your device. Detailed instructions are available in the [Android documentation](#).
3. **Windows-only:** Install the [Google USB Driver](#).
4. Using a USB cable, plug your phone into your computer. If prompted on your device, authorize your computer to access your device.
5. In the terminal, run the `flutter devices` command to verify that Flutter recognizes your connected Android device. By default, Flutter uses the version of the Android SDK where your AVD tool is based. If you want Flutter to use a different installation of the Android SDK, you must set the `ANDROID_SDK_ROOT` environment variable to that installation directory.

4.1.4 Set up the Android emulator

To prepare to run and test your Flutter app on the Android emulator, follow these steps:

1. Enable [VM acceleration](#) on your machine.
2. Launch **Android Studio**, click on **Device Manager** icon, and select **Create Device** under **Virtual** tab...
 - In older versions of Android Studio, you should instead launch **Android Studio > Tools > Android > AVD Manager** and select **Create Virtual Device....** (The **Android** submenu is only present when inside an Android project.)
 - If you do not have a project open, you can choose **3-Dot Menu / More Actions > Virtual Device Manager** and select **Create Device...**
3. Choose a device definition and select **Next**.
4. Select one or more system images for the Android versions you want to emulate, and select **Next**. An *x86* or *x86_64* image is recommended.
5. Under Emulated Performance, select **Hardware - GLES 2.0** to enable [hardware acceleration](#).
6. Verify the AVD configuration is correct, and select **Finish**. For details on the above steps, see [Managing AVDs](#).
7. In Android Virtual Device Manager, click **Run** in the toolbar. The emulator starts up and displays the default canvas for your selected OS version and device.

4.1.5 Agree to Android Licenses

To prepare to run and test your Flutter app on the Android emulator, follow these steps:

Before you can use Flutter, you must agree to the licenses of the Android SDK platform. This step should be done after you have installed the tools listed above.

1. Open an elevated console window and run the following command to begin signing licenses. `flutter doctor --android-licenses`
2. Review the terms of each license carefully before agreeing to them.

3. Once you are done agreeing with licenses, run `flutter doctor` again to confirm that you are ready to use Flutter.

```
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.13.1, on Microsoft Windows [Version 10.0.19045.3324], locale en-US)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop Windows apps (Visual Studio Community 2019 16.11.0)
[✓] Android Studio (version 2022.3)
[✓] VS Code (version 1.81.1)
[✓] Connected device (3 available)
[✓] Network resources

• No issues found!
```

Figure 2: Flutter Doctor - working

4.1.6 Windows setup

These are additional requirements for developers setting up on an Windows environment.

For Windows desktop development, a Visual Studio (2019 or 2022) workload is required in addition to the Flutter SDK. [Visual Studio 2022 IDE](#) [Visual Studio Build Tools 2022](#) When installing Visual Studio or only the Build Tools, you need the “Desktop development with C++” workload installed for building windows, including all of its default components.

4.2 Set up a text editor (VS Code)

4.2.1 Install VS Code

[VS Code](#) is a code editor to build and debug apps. With the Flutter extension installed, you can compile, deploy, and debug Flutter apps.

To install the latest version of VS Code, follow Microsoft’s instructions: [Install on Windows](#)

4.2.2 Install the VS Code Flutter extension

1. Start VS Code.
2. Open a browser and go to the [Flutter extension](#) page on the Visual Studio Marketplace.

3. Click Install. Installing the Flutter extension also installs the Dart extension.

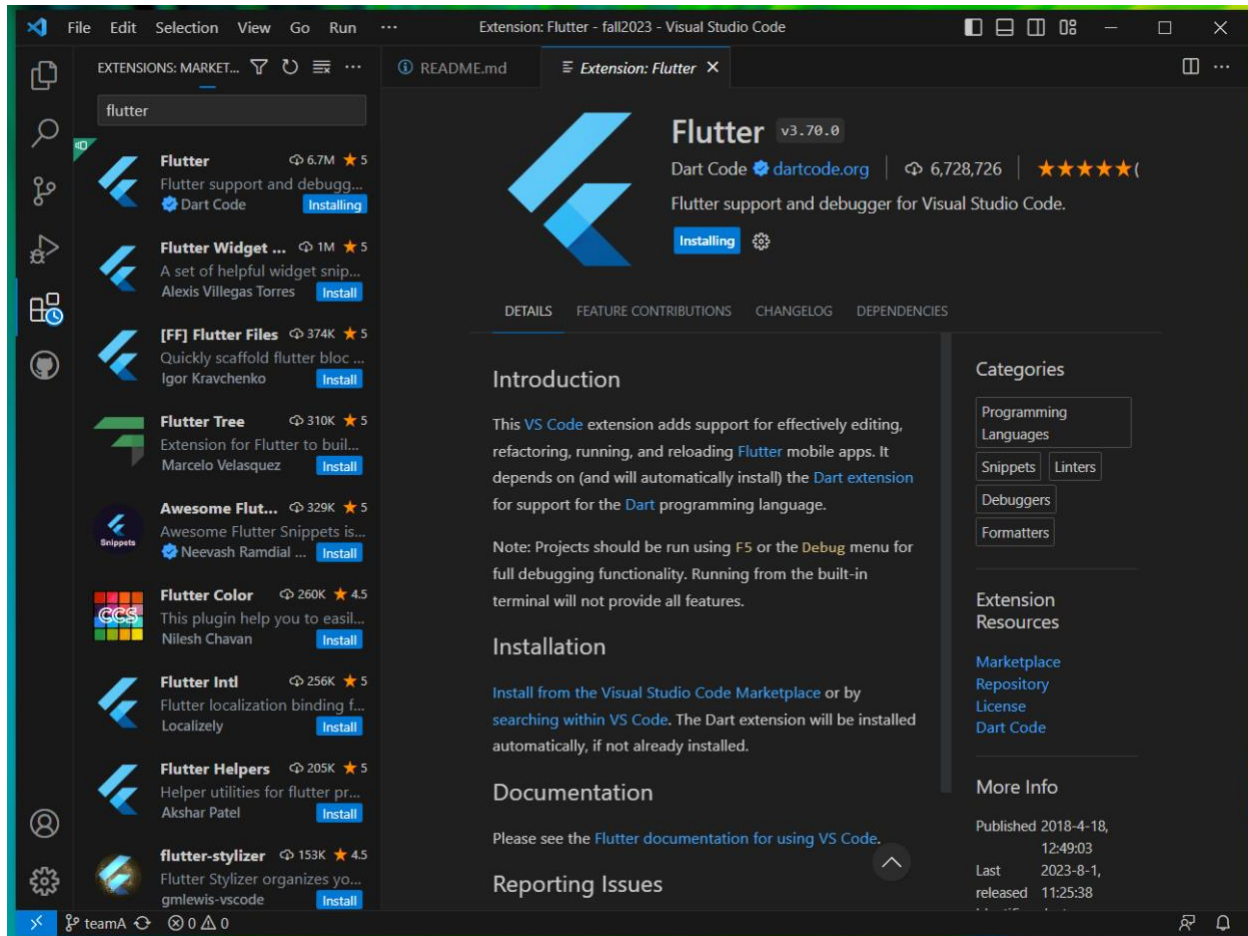


Figure 3: Installing Flutter

4.2.3 Validate your VS Code setup

1. Go to **View > Output**. You can also press Control / Command + Shift + U.
2. In the dropdown on the upper right of the **Output** panel, select **flutter (flutter)**.
3. Go to **View > Command Palette....** You can also press Control / Command + Shift + P.
4. Type doctor.
5. Select the **Flutter: Run Flutter Doctor**. Flutter Doctor runs and its response displays in the **Output** panel.

5 Preparations for Use

5.1 Source Control

5.1.1 Cloning a repository

If you have not already downloaded the “Git” extension for VS Code, do so now.

1. Go to the “Source Control” view and select “Clone Repository”.

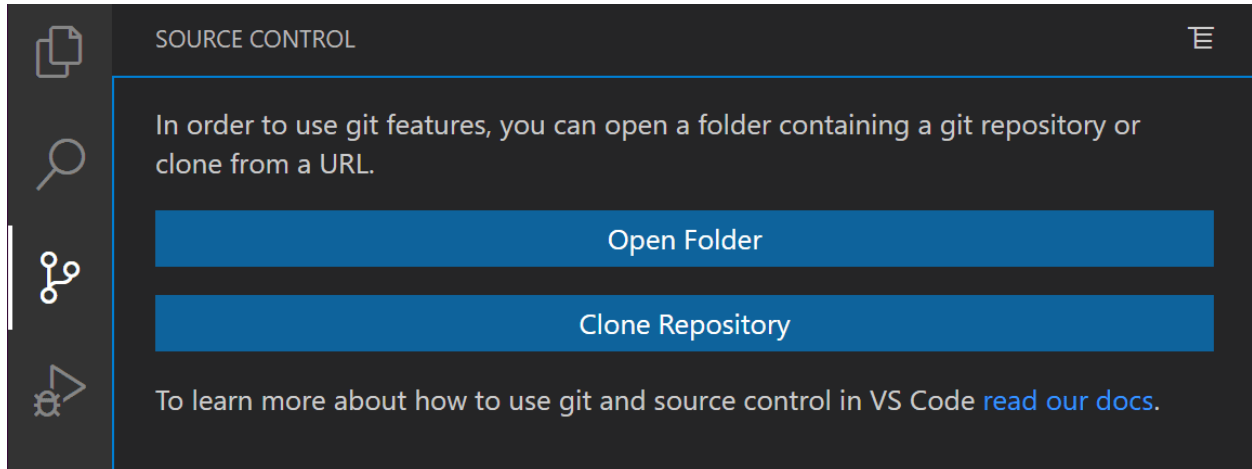


Figure 4: Cloning a Git Repository

2. Enter the GitHub repository URL.
3. “Git: Create Branch” to create a new branch. “Git: Checkout” to pull down an existing branch.

5.2 Setup 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.

5.2.1 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", "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.

Likewise be sure to always stop the custom labels (the project versions) object detection service when not in use for development/testing. That incurs a hefty charge once exceeding the free tier.

5.3 Compiling

TBD how to compile the application

1. Run `flutter pub get` to install dependencies
2. Run `flutter emulators --launch <emulator_id>` to start the emulator device. For example, `flutter emulators --launch Pixel_5_API_34`
 - Use `flutter emulators` to get a list of emulator ids.
3. Run `flutter analyze` to compile the application before deploying to the emulated environment.

6 Deployment

6.1 Deployment

The application is primarily deployed to Android devices, both emulated and physical. The commands to deploy these applications to either emulated or physical are detailed in the sections below.

6.1.1 Emulator

1. Run `flutter emulators --launch <emulator_id>` to start the emulator device. For example, `flutter emulators --launch Pixel_5_API_34`
 - Use `flutter emulators` to get a list of emulator ids.
 - Run `flutter run` to run the application in the emulated environment.

6.1.2 Physical phone

1. Turn on Developer Mode on the Android device
 - Go to Settings > About phone
 - Scroll down to Build number
 - Tap Build number seven times
 - Once developer options are activated, you will see a message that reads, 'You are now a developer!'

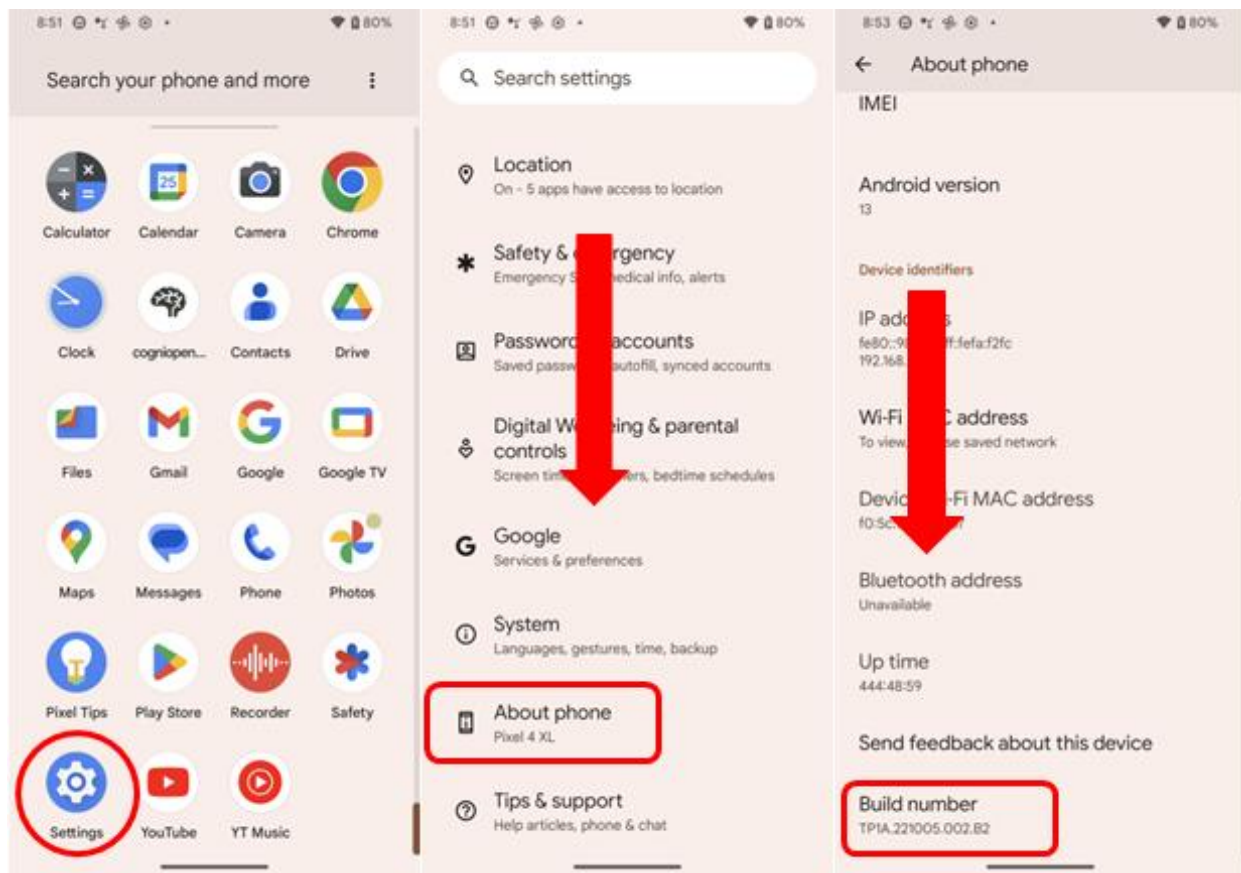


Figure 5: Turn on Developer Mode

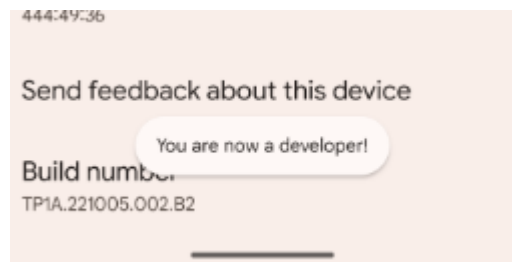


Figure 6: Enabled Developer Mode

2. Enable USB debugging

- Go to Settings > System > Developer options
- Scroll down and enable USB debugging
- Allow USB debugging by tapping on 'OK'

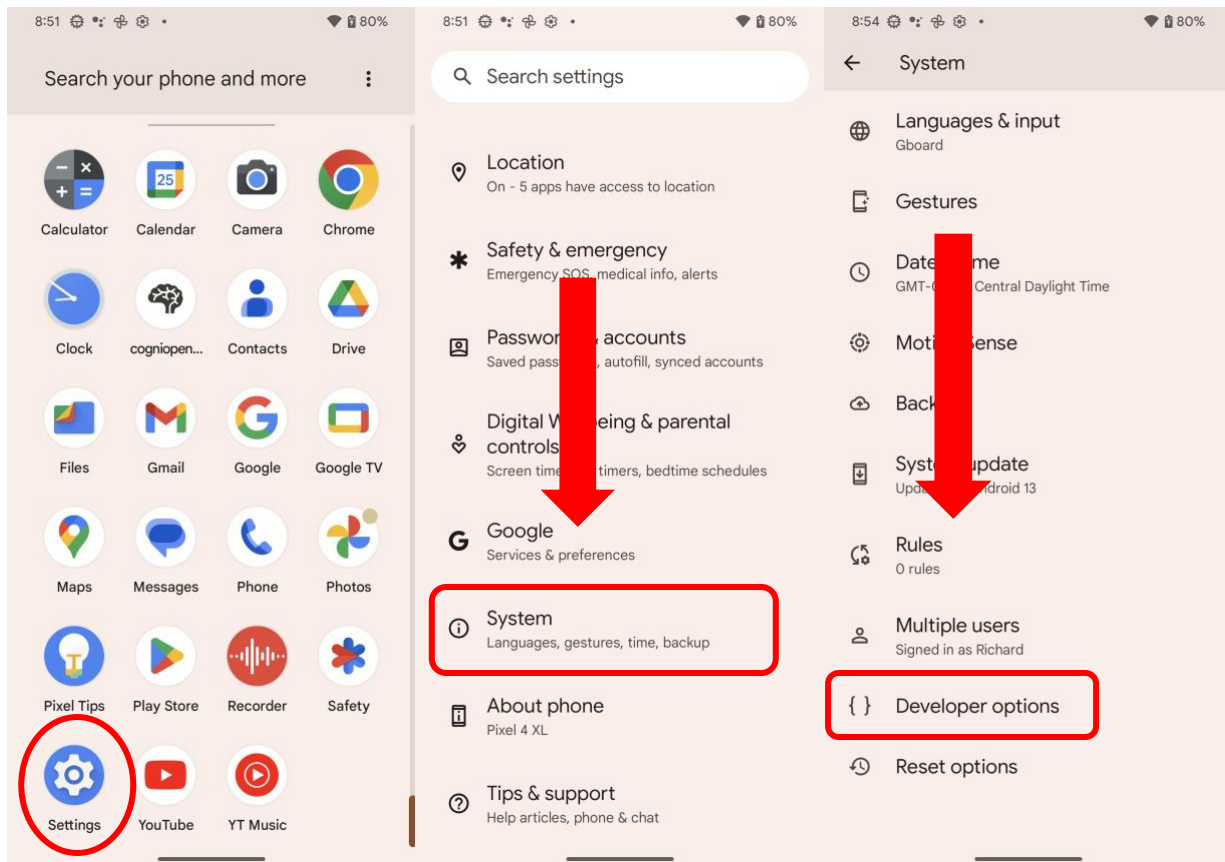


Figure 7: Developer Options

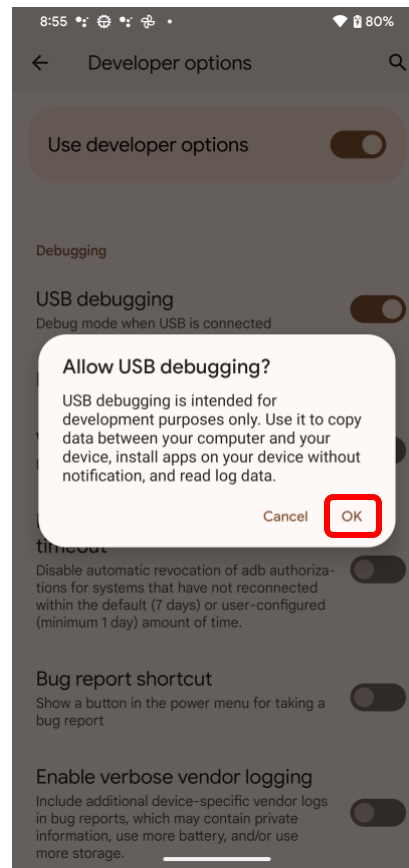


Figure 8: Allow USB Debugging

3. Connect the Android device into the computer
 - Allow USB debugging of the connected computer



Figure 9: Check Allow USB debugging

4. Select the Android device to run the application on
 - Click on the device selector on bottom right corner of Visual Studio Code
 - A selection of devices and emulators will appear
 - Click on your connected Android device

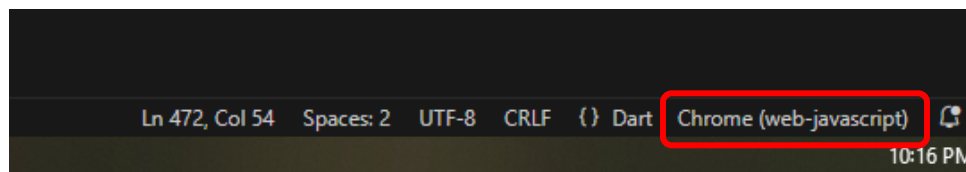
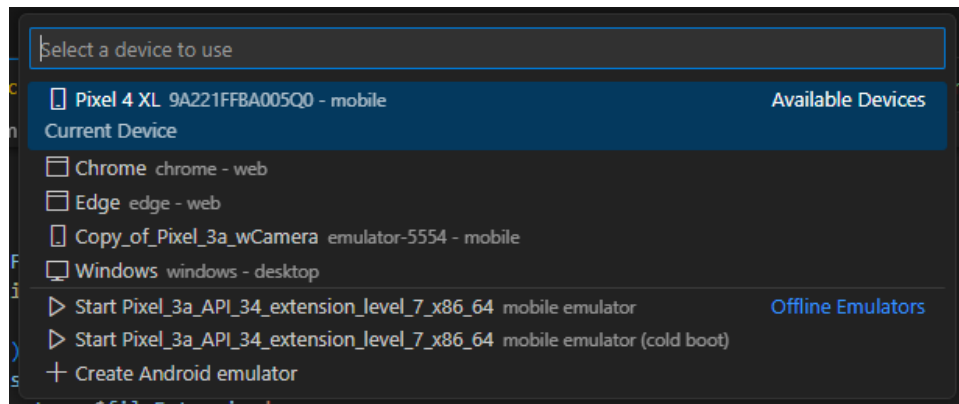
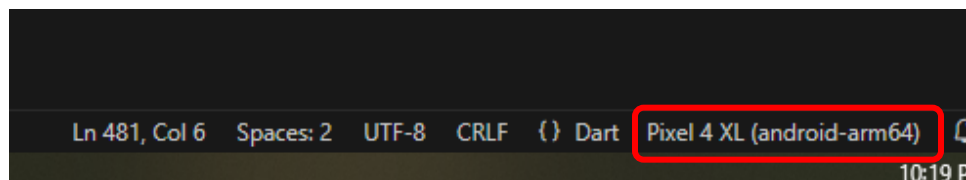


Figure 10: Changing device

*Figure 11: Changing device**Figure 12: Device changed*

5. Run `flutter run` to run the application on your Android device

7 Automation Testing

This section will document the details of the automation testing including how it was implemented and where and how to access the test results.

7.1 GitHub Actions

The automation testing and results are setup using GitHub's CI/CD pipeline. To view configuration and results, navigate to the Actions tab under the fall2023 GitHub project. The link is here: [Android CI · Workflow runs · umgc/fall2023 \(github.com\)](https://github.com/umgc/fall2023/actions/workflows/android-ci-cd.yml)

7.1.1 Setup

The automation testing is set up using the *android-ci-cd.yml* file listed on the GitHub Actions page. The file is listed below with details on what each step is doing for clarity.

```

1  name: Android CI
2
3  on:
4    push:
5      branches: [ "main", "development", "testing" ]
6    pull_request:
7      branches: [ "main", "development", "testing" ]
8
9  jobs:
10   build:
11
12     runs-on: ubuntu-latest
13
14     steps:
15       - run: echo "The job was automatically triggered by a ${ github.event_name } event."
16       - run: echo "This job is now running on a ${ runner.os } server hosted by GitHub!"
17       - run: echo "The name of your branch is ${ github.ref } and your repository is ${ github.repository }."
18       - run: echo "The ${ github.repository } repository
19         has been cloned to the runner."
20       - name: "Checkout repository code"
21         uses: actions/checkout@v4
22       - name: "Setup Java"
23         uses: actions/setup-java@v1
24         with:
25           java-version: 11.x
26       - name: Create .env file
27         run: touch .env
28         working-directory: ./cogniopenapp
29       - name: "Setup Flutter"
30         uses: subosito/flutter-action@v2
31         with:
32           channel: "stable"
33       - name: "Install Flutter dependencies"
34         run: flutter pub get
35         working-directory: ./cogniopenapp
36       - name: "Run Unit Tests"
37         run: flutter test
38         working-directory: ./cogniopenapp

```

trigger running of the jobs below on push and pull_requests to the main, development and testing branches of the GitHub project

operating system setup using the latest version of ubuntu

checkout the associated repository and branch

sets up java using the Azul Zulu OpenJDK version 11.x

creates an empty .env file needed to build the application

setups the flutter environment using the latest stable version

installs all needed dependencies for the application

runs all tests

Figure 13: Automated Testing Configuration File

7.1.2 Execution

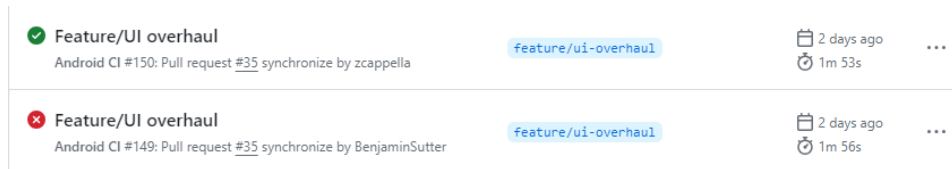
The automation testing is triggered on push and pull-requests from the main, development and testing branches of the GitHub project. Under the GitHub

Actions tab, there will be a blue box indicating which branch triggered the automation testing.

7.1.3 Results

The automation testing results can be viewed under the GitHub Actions tab, a link to the page can be found here: [Workflow runs · umgc/fall2023 \(github.com\)](https://github.com/umgc/fall2023/actions/workflows/android-ci-cd.yml)

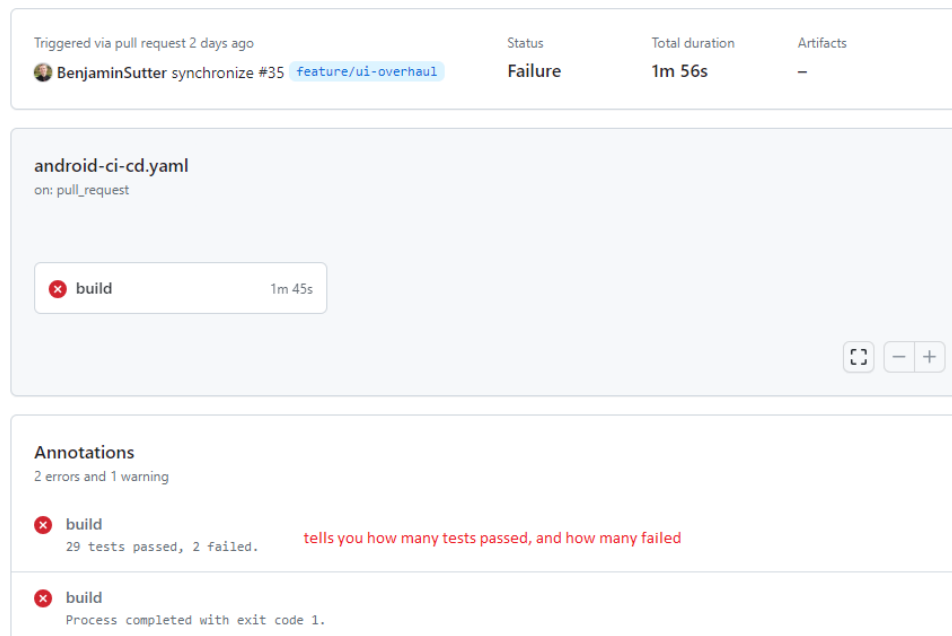
Each new trigger of the automation testing is called a workflow run. Using the image below for clarity. From the Actions page each workflow will have a summary item which displays whether all tests have passed (the green check indicates all tests passed, the red x indicated tests have failed), name of the branch triggering the result (feature-ui-overhaul), what type of action triggered the automation testing (Pull request #35 synchronize), and when it was run (2 days ago) and for how long (1m 53s).




✔ Feature/UI overhaul Android CI #150: Pull request #35 synchronize by zcappella	feature/ui-overhaul	2 days ago 1m 53s	...
✖ Feature/UI overhaul Android CI #149: Pull request #35 synchronize by BenjaminSutter	feature/ui-overhaul	2 days ago 1m 56s	...

Figure 14: Test Results

To view more details on which tests passed and/or failed the user can click on the title “Feature/UI overhaul” next to the passing indicator to open up a details page on a particular run. It will display how many tests passed and how many failed, annotated below in the example.



Triggered via pull request 2 days ago

	Status	Total duration	Artifacts
 BenjaminSutter synchronize #35 feature/ui-overhaul	Failure	1m 56s	–

android-ci-cd.yaml
on: pull_request

✖ build 1m 45s

Annotations
2 errors and 1 warning

- ✖ build
29 tests passed, 2 failed. tells you how many tests passed, and how many failed
- ✖ build
Process completed with exit code 1.

Figure 15: Test Failed

For more details on the failed versus passing tests, click on the build text to open up the details. It will display a line with a green checkbox or red x next to each line which represents one test which was executed. The white carrot can be clicked to expand details on failing tests. Image below for clarity.

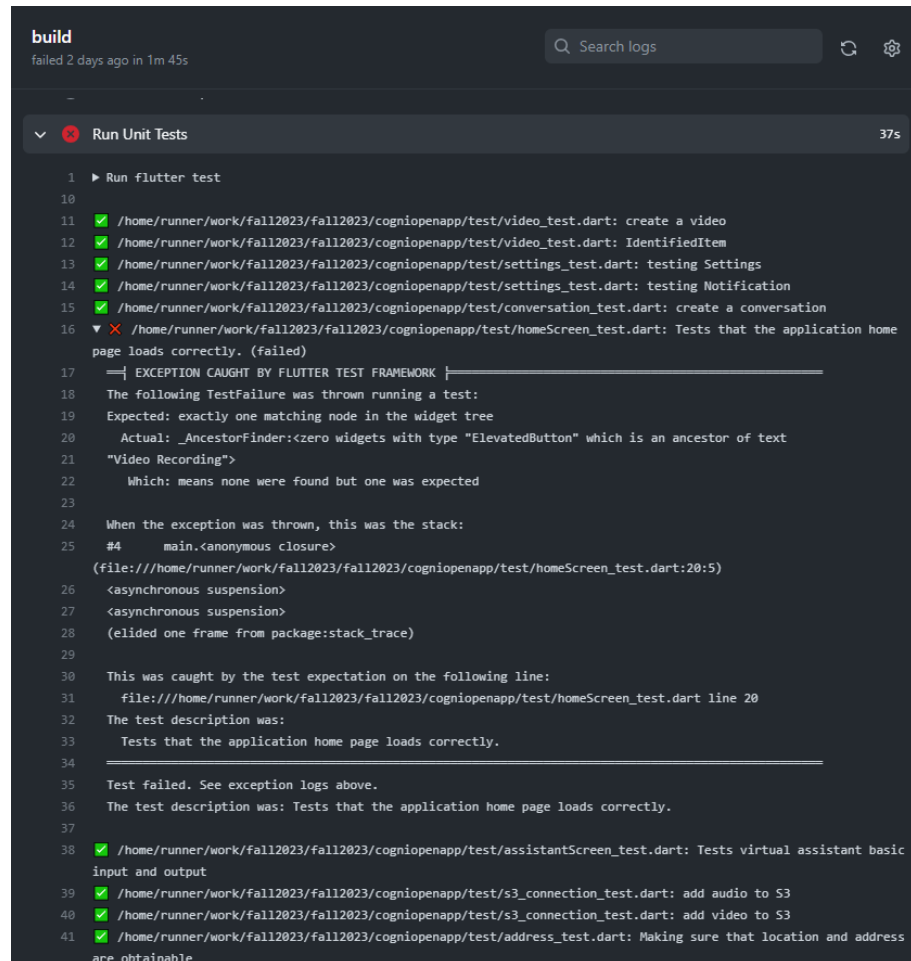


Figure 16: Detailed Test Results

8 Troubleshooting

8.1 Not recording audio in an emulator

1. Click on the 3 consecutive dots located on the right of the device.
2. On the popup Extended Controls window, click on the Microphone tab on the left.
3. Toggle 'Virtual microphone uses host audio input' to enable.
 - a. Toggling “virtual headset plug inserted” will enable all options.

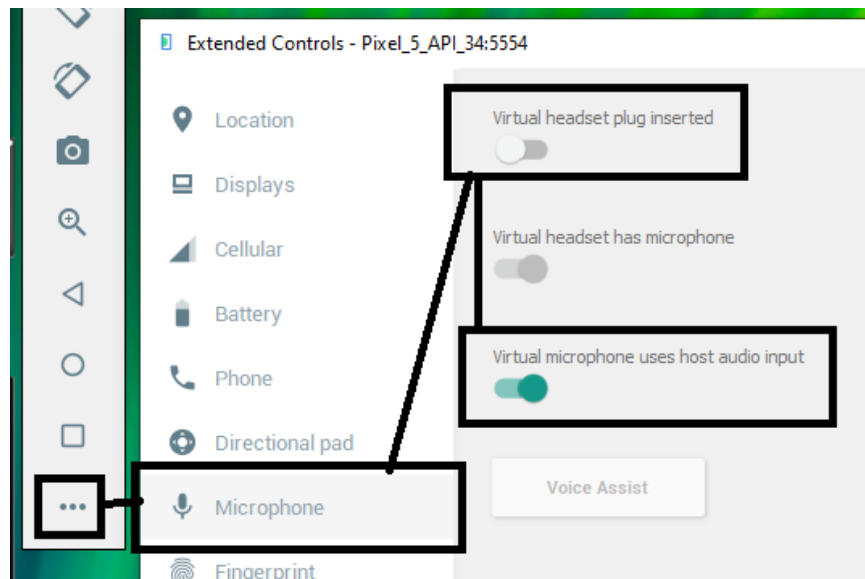
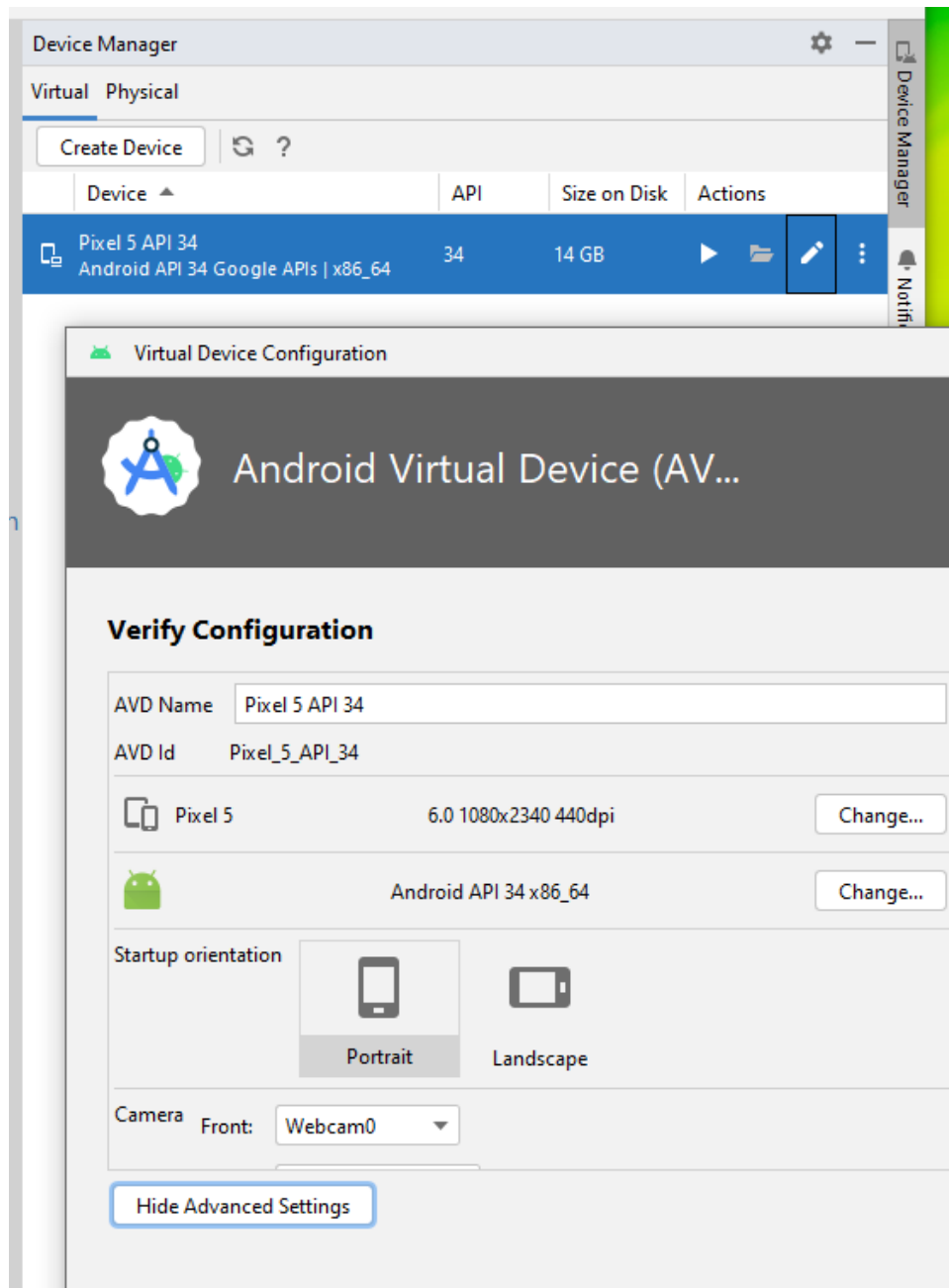


Figure 17: Enable Microphone

8.2 Not recording video in an emulator

1. Open Android Studio
2. In the Device Manager, select the “edit” icon on the emulated device
3. Select “Show Advanced Settings”
4. In the Camera section, select “Webcam0” from the drop-down menu for the Front camera, and “virtualscene” from the drop-down menu for the Back camera.

5. Select “Finish” to apply changes

*Figure 18: Device “Advanced Settings”*

6. If you are still experiencing issues, try to switch the back camera to “VirtualScene” from the drop-down menu next to “Back”.
7. Select “Finish” to apply changes.

8.3 Database not loading

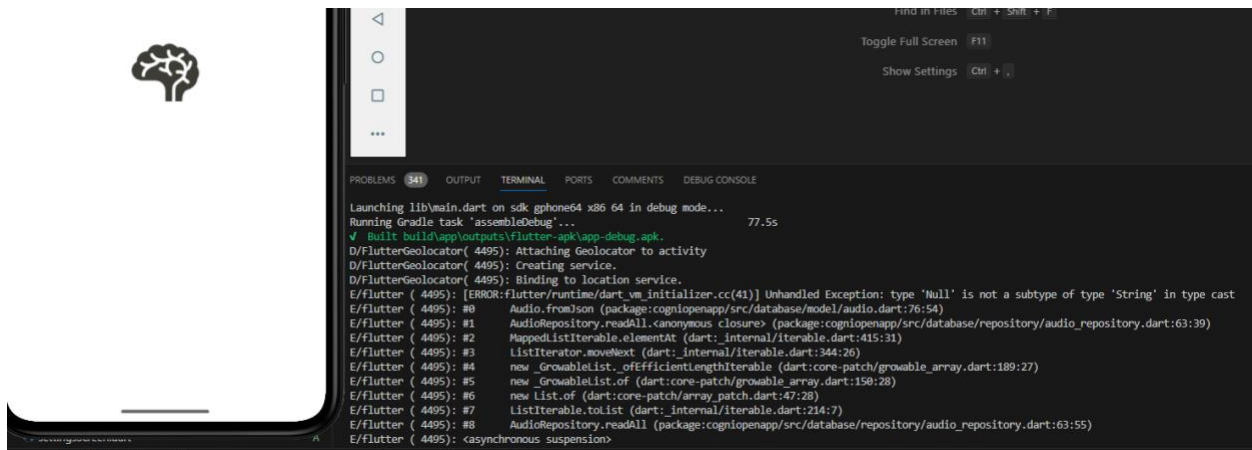


Figure 19: Error screen in flutter

- Open Android Studio
- In the Device Manager, select the “3 dot menu” on the device
- Select “Wipe Data”

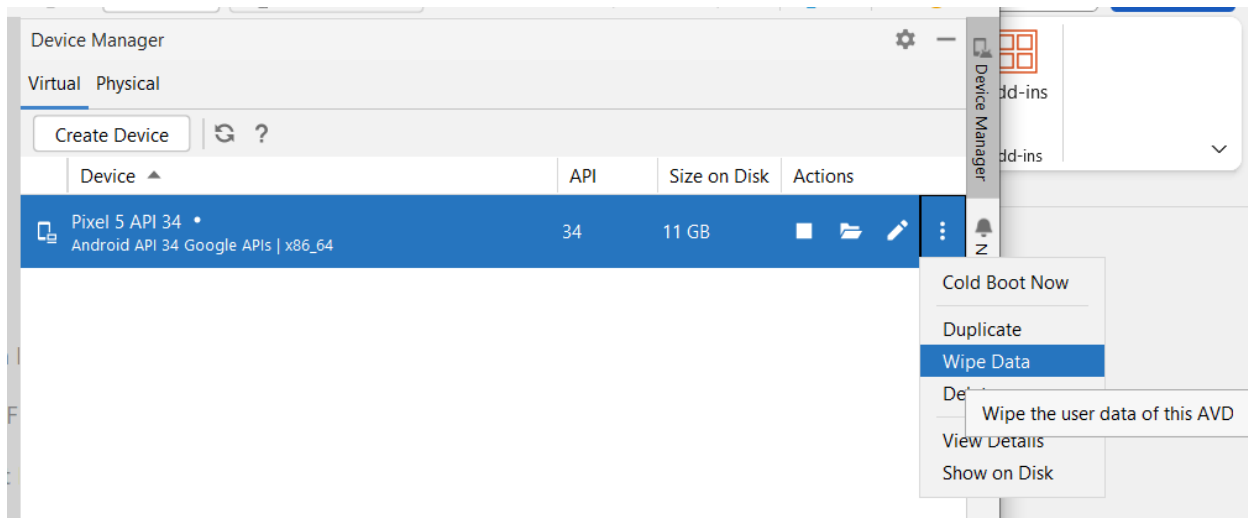


Figure 20: Wipe Data of virtual device