Short-Term Memory System (STeMS)
Deployment and Operations Guide
By AlphaSoft

Team Members

Awasthi, Aaditya

Blavat, Oleksiy

Bond, Matthew

Carter, Mackenzie

Mahbobi, Sayed shah

McAllister, Charlie

McCool, Max

McLaughlin, Taylor

Powers, Michael

Weaver, Daniel

Revision History

Date	Version	Description	Author
22 July 2023	1.0	Initial submission to Client	AlphaSoft
5 August 2023	2.0	Updated to match current state of project.	AlphaSoft

Table of Contents

Figures & Tables

1. Introduction

1.1. Purpose

This Deployment and Operations Guide describes how to configure a developer system so they may maintain and enhance the code. It includes instructions on how to retrieve and compile the code for all related projects, as well as how to deploy the Short-Term Memory System (STeMS). There are 3 separate code bases that work together to make up STeMS: the ConvoBuddy app, the browser extension service (BESie), and the browser extension. Each will be treated separately in this document unless otherwise specified.

By providing exact instructions for each step needed to get, compile, and deploy the code, this Deployment and Operations Guide will ensure that every STeMS developer has the same environment. This, in turn, will help prevent version conflict issues, delays due to lost tribal knowledge, and expediate on-boarding of new members of AlphaSoft.

1.2. Intended Audience

This document is intended to be used by the AlphaSoft Team as a developer reference for how to do things with the project code. It could be used when an existing developer needs to setup a new machine or when a new developer is hired. The AlphaSoft Quality Assurance Team may also use this reference to assist in the creation of test environments or for back-end testing. Future developers of STeMS will find this reference invaluable for quickly working with the codebase.

1.3. Project Documents

This Deployment and Operations Guide will be included in the documentation portion of the project deliverables. All documentation created throughout the project lifecycle is done so with the intent to assist in the understanding, implementation, and maintenance of both the mobile application and browser extension.

The following documents will be included as part of the project deliverables.

Document	Version	Date
Project Plan (PP)	4.0	5 August 2023
Software Requirements Specification (SRS)	4.0	5 August 2023
Technical Design Document (TDD)	3.0	5 August 2023
Software Test Plan (STP)	3.0	5 August 2023
Programmer Guide (PG)	2.0	5 August 2023
Deployment and Operations Guide (DOG)	2.0	5 August 2023
Software Test Report (STR)	1.0	5 August 2023
User Guide (UG)	1.0	5 August 2023
Traceability Matrix (TM)	1.0	5 August 2023

Table 1: Project Documents

1.4. Acronyms, Definitions, and Abbreviations

Throughout this Deployment and Operations Guide, a variety of terms and acronyms specific to the proposed application are used. For clarity, their definitions are provided below:

Term	Definition
AI	Artificial Intelligence
API	Application Programming Interface, or a way that difference applications interact with each other
App	A program that is included on the App User's mobile device
BESie	Back End Service
Browser Extension	An extension to either the Chrome or Safari browser, which works with the App to provide specific functionality
Conversation	A recording the App User made and saved within the app
ConvoBuddy	The app under development by this project
DOM	Document Object Model
Flutter	A software framework for developing cross-platform mobile applications
HTTP	Hypertext Transfer Protocol
iOS	iPhone Operating System
JSON	JavaScript Object Notation
Mobile Device	A smart phone, tablet, or some other portable computer with either the iOS or Android operating system
PG	Programmer's Guide
PP	Project Plan
Deployment and Operations Guide	A document that explains how to deploy the program
SRS	Software Requirements Specification
STeMS	Short-Term Memory System, the name of this project
STP	Software Test Plan
STT	Speech to Text
System	The complete STeMS system, which includes the App, the Brower Extension, and the Browser Extension Service
TDD	Technical Design Document
TM	Traceability Matrix. A document that traces defects and test cases back to their requirement.

Term	Definition
TMGR	Transmogrifier
TR	Test Report
UG	User Guide
UI/UX	User Interface/User Experience
URL	Uniform Resource Locator, typically a path to files on a computer

Table 2: Acronyms, Definitions and Abbreviations

1.5. Organization of Document

This document has six sections: Introduction, Source Control, ConvoBuddy App, BESie, Browser Extension, and Troubleshooting. The Introduction gives an overview of this document, the intended audience, and other helpful information to understand the rest of the document. The Source Control section is about accessing and understanding the organization of our source control for the code. This helps developers keep current on patterns so the code stays organized.

The next three sections are for the three different parts of STeMS, the app, the web service, and the extension. All three of these sections will cover how to set up and configure a developer's machine so that the developer can work with the code, compile it, and deploy it.

The last section, Troubleshooting, covers all issues that prior developers and testers have encountered and how they were solved. This helps other developers and testers quickly overcome issues they are experiencing.

2. Source Control

2.1. Account Setup

1. You must have an existing GitHub account
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.

2.2. Repositories

1. Main repository can be found at https://github.com/umgc/summer2023.
2. The app code can be found at https://github.com/umgc/summer2023/tree/development/talker-mobile-app.
3. BESie can be found at https://github.com/umgc/summer2023/tree/development/BESie.
4. The form-filler extension can be found at https://github.com/umgc/summer2023/tree/development/form-filler-extension.
5. The automation tests can be found at https://github.com/umgc/summer2023/tree/development/automation.

2.3. Organization

GitFlow will be used, which means the main branch is clean and the development branch is for staging, intermediary builds, and testing. All feature development will branch out of the development branch. All pull requests that go into main must be pulled down locally, built, and executed locally to ensure only runnable code makes it into main.

3. ConvoBuddy App

3.1. Machine Setup

3.1.1. Requirements

3.1.1.1. OS

- Windows 10 or later (64-bit), x86-64 based
- macOS, version 13.0 (Ventura) or later.

3.1.1.2. Hardware

- Disk Space: 10 GB
- Memory: 4 GB

3.1.2. Program Installations

3.1.2.1. Git

1. Download from https://git-scm.com/download/.

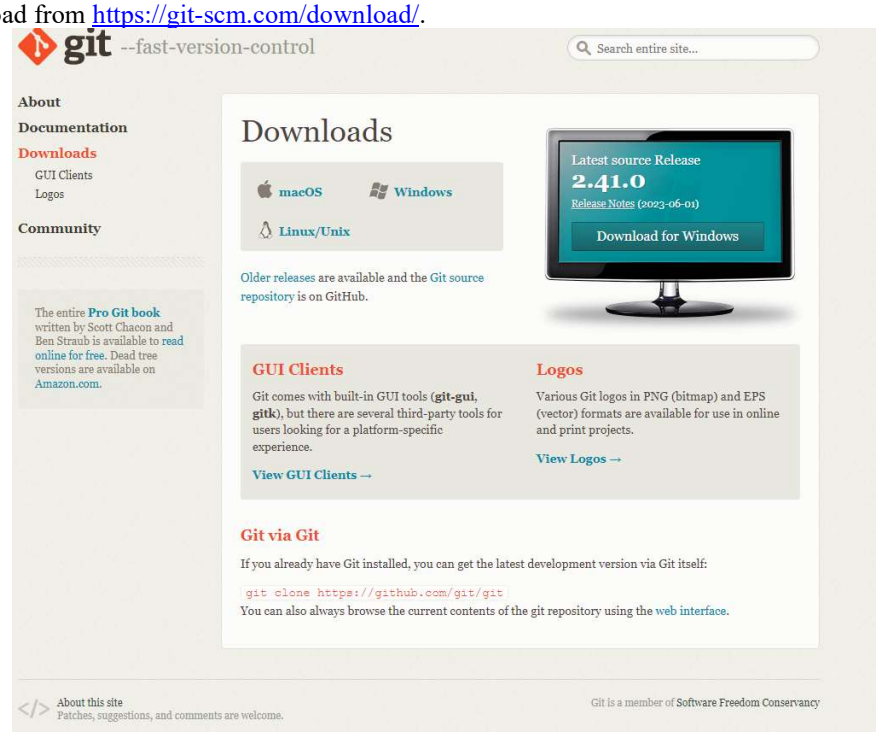


Figure 3: Git Website Download Options

   a. Select macOS, Windows, or Linux/Unix.
   b. Follow the instructions on the OS specific page for downloading and/or installing.
2. Installation for Windows (if applicable). Use the default values except as listed below:
   a. The default editor should be set to Notepad++ or Visual Studio Code (not VIM).

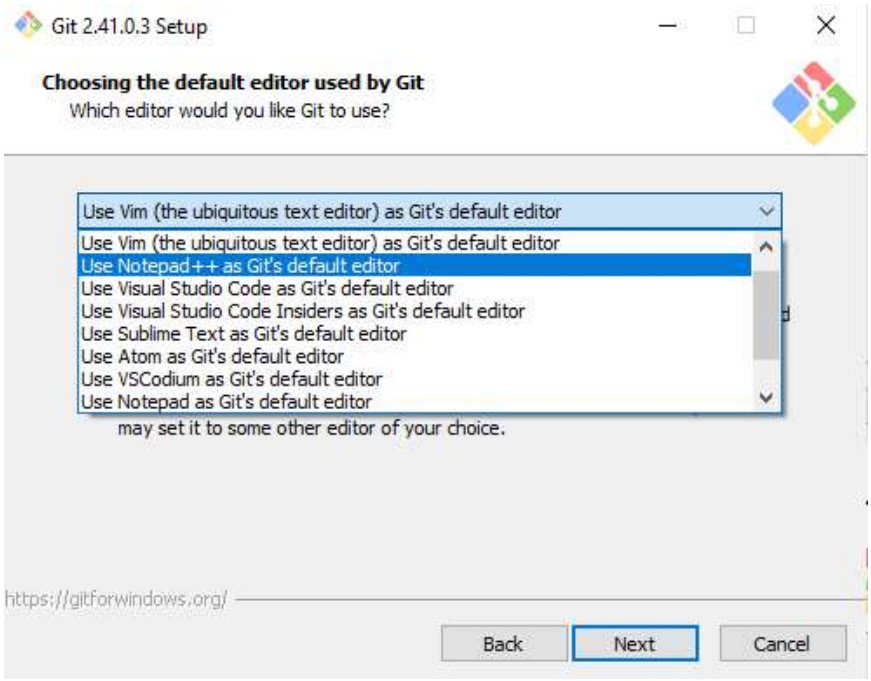


Figure 4: Git - Default Editor

b. The name of the initial branch should be overridden to "main" (case sensitive).

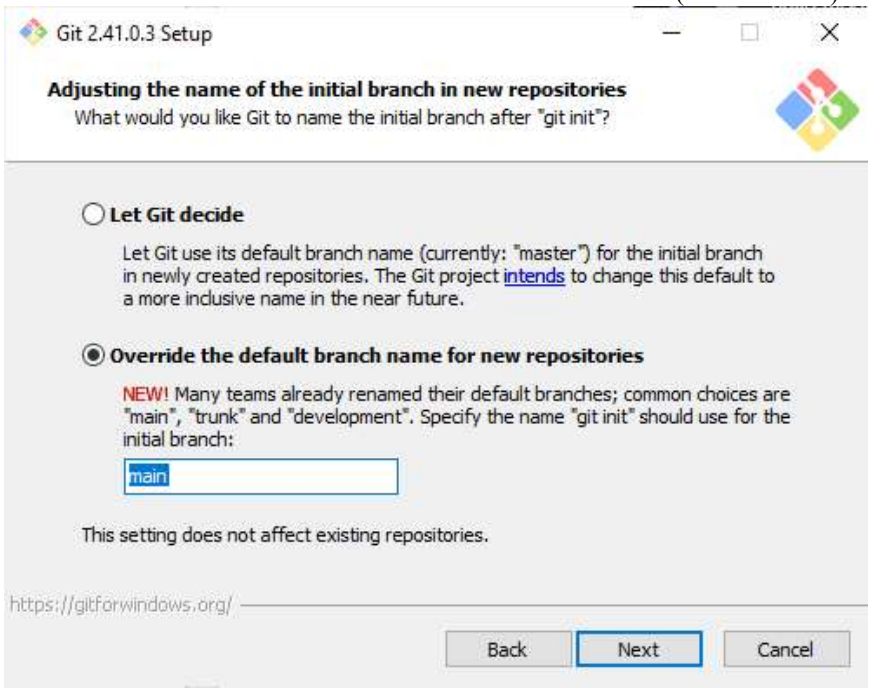


Figure 5: Git - Initial Branch Name

c. Adjusting your PATH environment: Make sure "Git from the command line and also from 3rd-party software" is selected. This is required for Flutter to install properly.

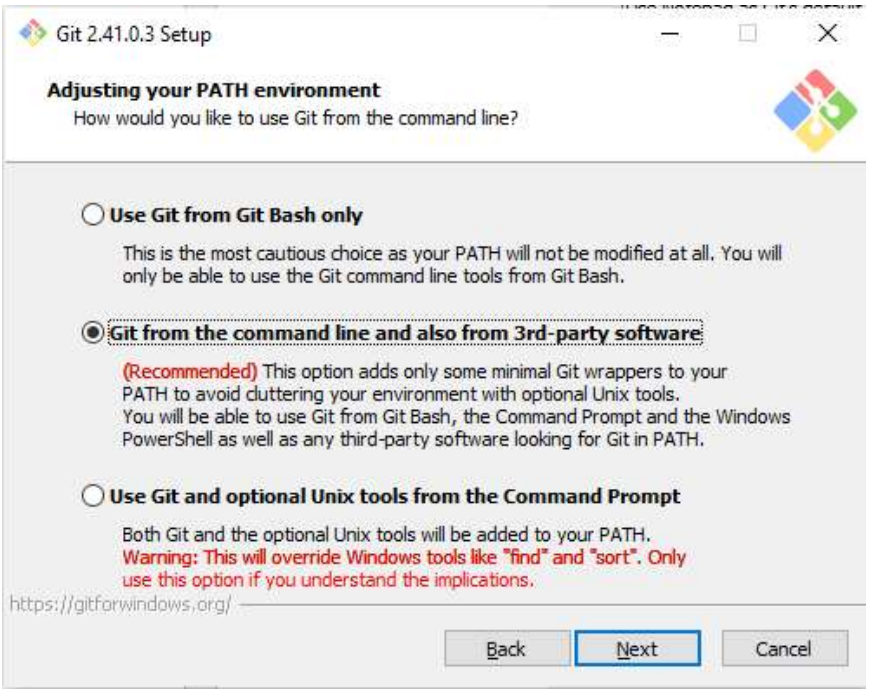


Figure 6: Git - Adjust PATH Environment

d. Select “Use bundled OpenSSH”

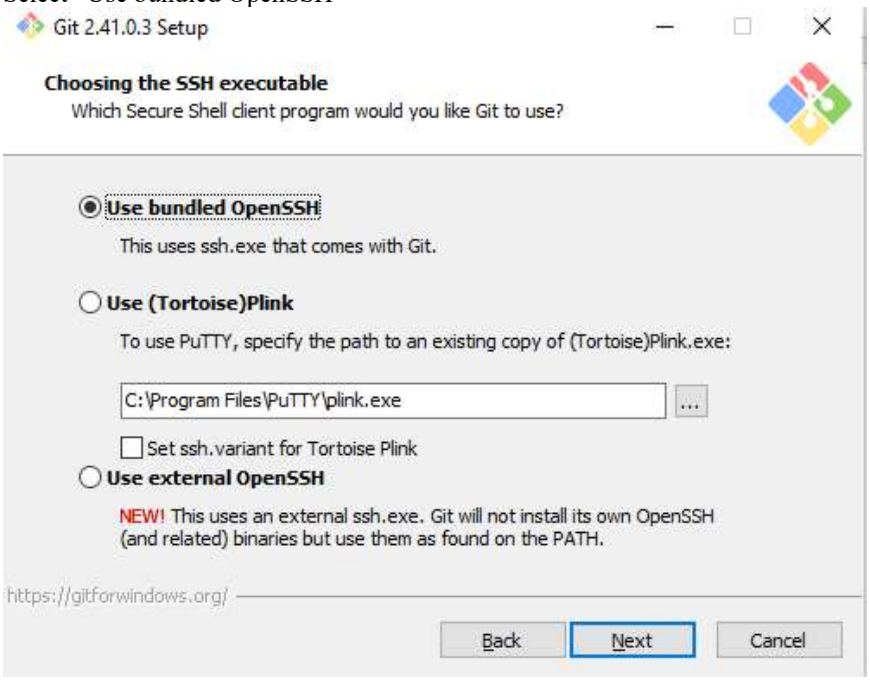


Figure 7: Git - Choose SSH Executable

e. Select “Use the openSSL library”

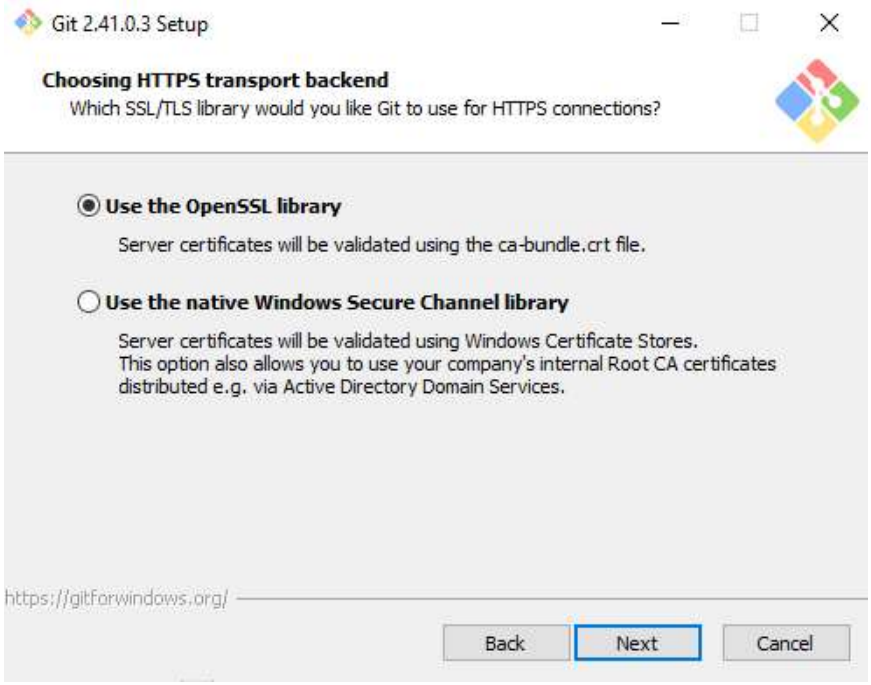


Figure 8: Git - Choose HTTPS Transport Backend

f. Select “Checkout Windows-Style, commit Unix-style line endings”

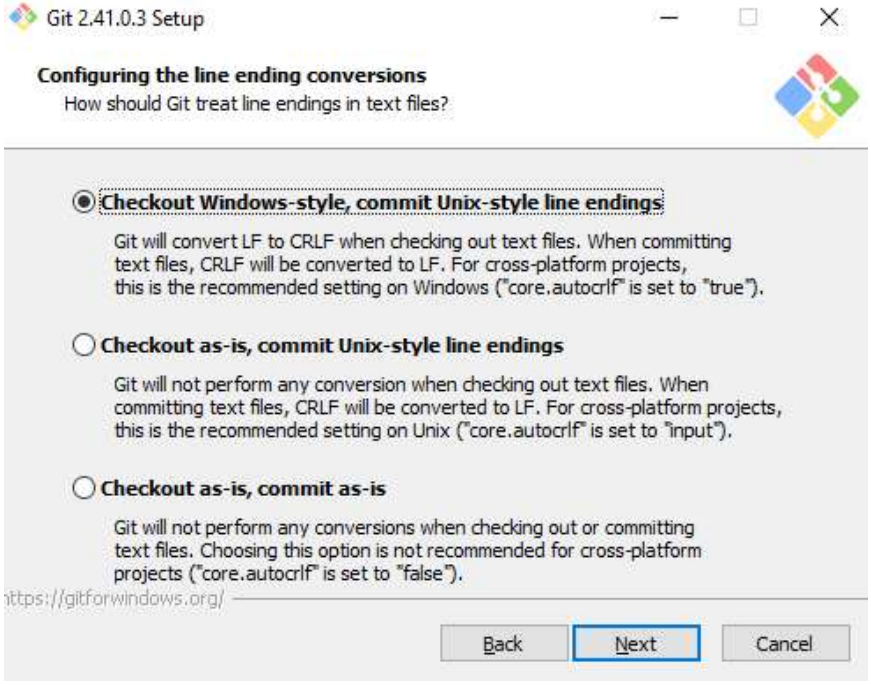


Figure 9: Git - Line Ending Conversions

g. Select “Use MinTTY”

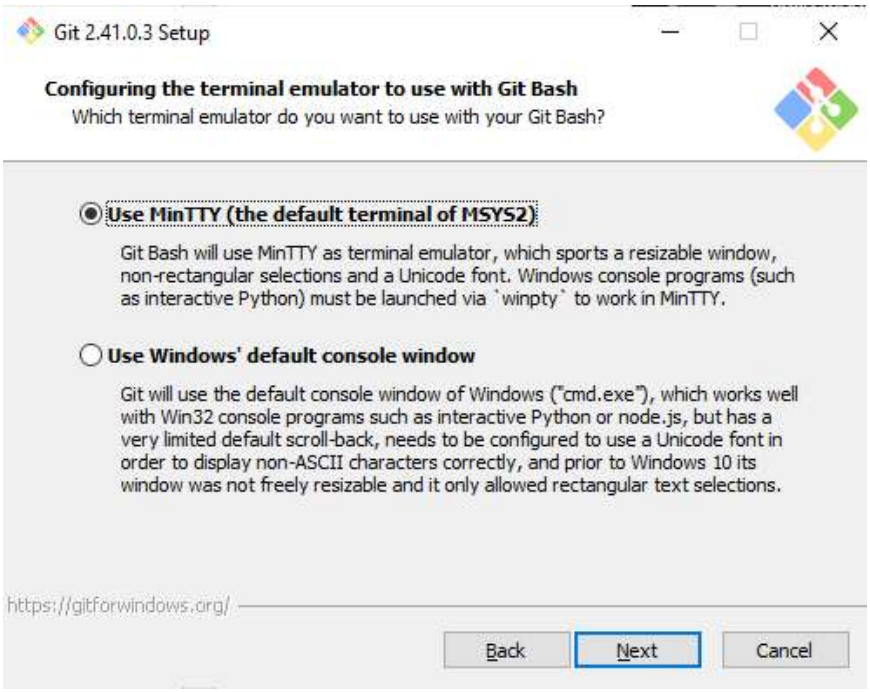


Figure 10: Git - Choose Terminal Emulator

h. Select “Default (fast-forward or merge)

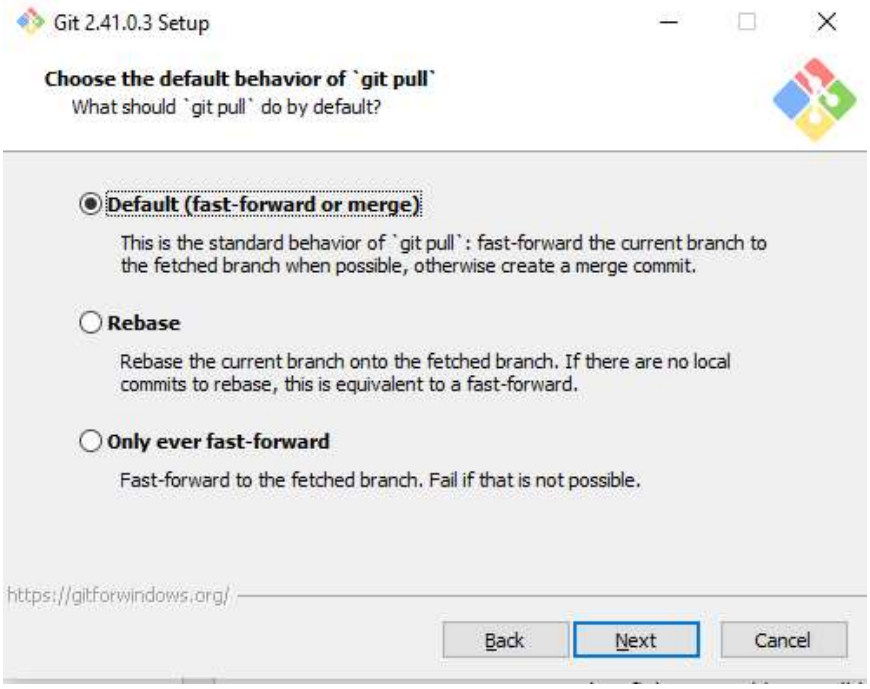


Figure 11: Git - Default Behavior of 'git pull'

i. Select “Git Credential Manager

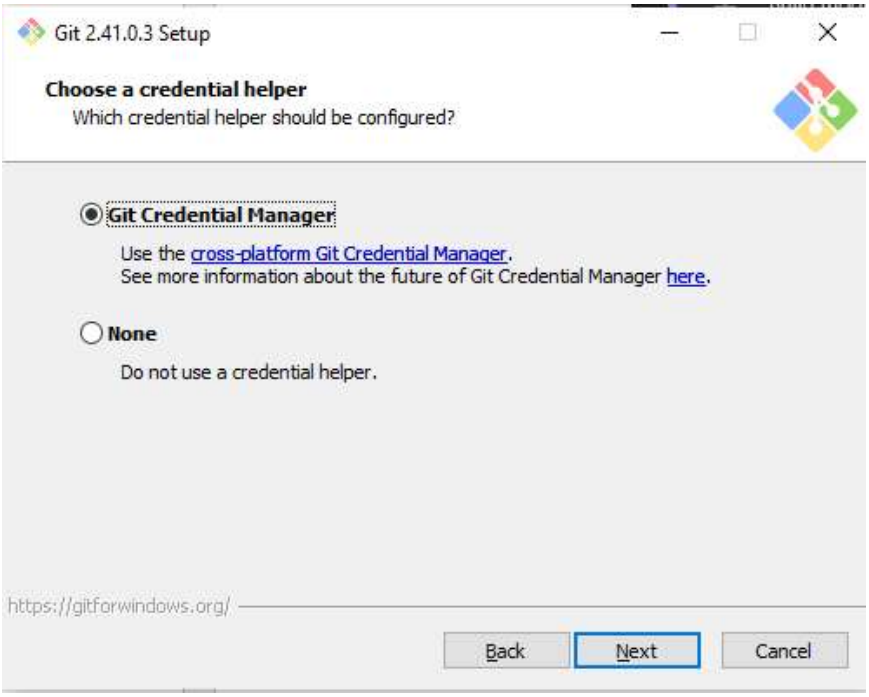


Figure 12: Git - Set Credential Manager

j. Check “Enable file system caching”

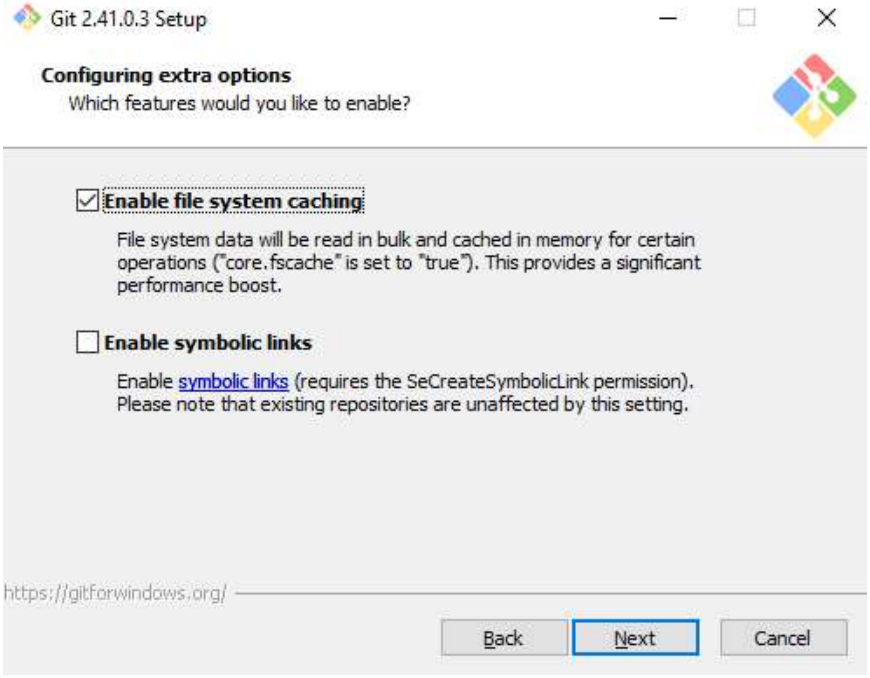


Figure 13: Git - File System Caching

k. Click Next to finish the installation.

3.1.2.2. Visual Studio

1. Download the community edition from https://visualstudio.microsoft.com/downloads/.
    a. Edition 2022, preferably version 17.6.0
2. Run the downloaded file.
3. Make sure to install the "Desktop Development with C++" workload.

Figure 14: Visual Studio's Workloads – Desktop Development with C++

4. For all other workloads, only include if already used.
5. Click "Install while downloading" in the bottom right corner.

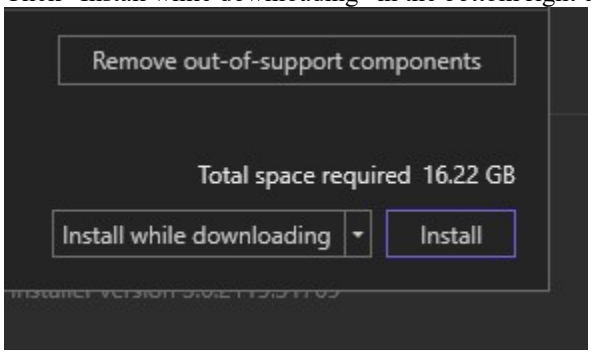


Figure 15: Visual Studios - Install While Downloading

6. When done, launch it and choose your personal preferences.

3.1.2.3. Xcode (mac OS only)

1. For M1 and M2 chip architecture, install Rosetta interpreter using the following command: sudo softwareupdate --install-rosetta --agree-to-license
2. Install xcode using the following command and follow the system prompts: sudo softwareupdate --install-rosetta --agree-to-license

3.1.2.4. Android Studio Flamingo

1. Download from https://developer.android.com/studio/.
2. Installation

a. Install all components to the default location.
b. Check “Start Android Studio” and click Finish.

3. The Android Studio Setup Wizard will run.
    a. Select “Standard” installation type.

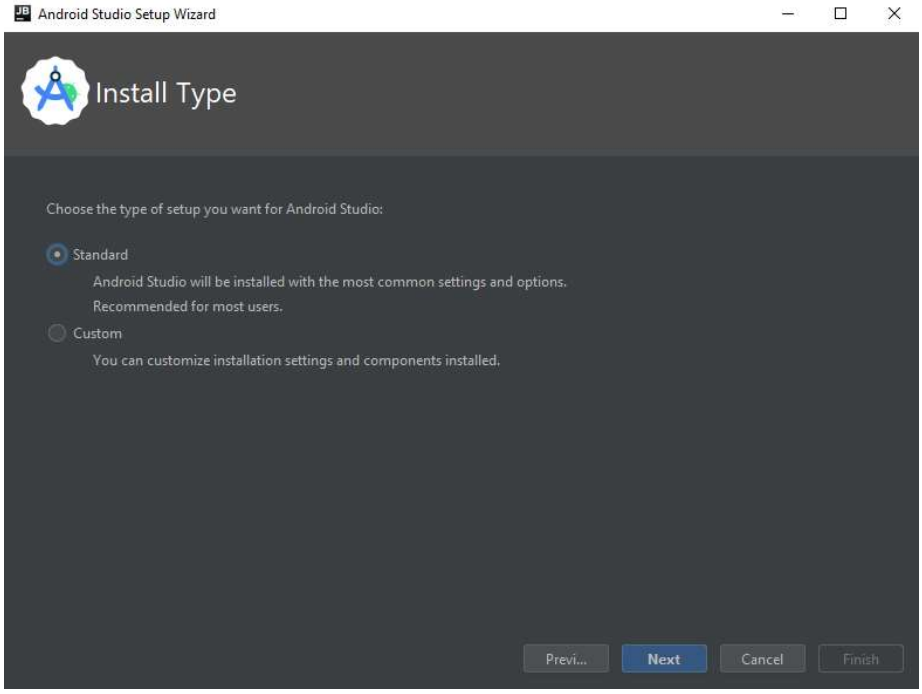


Figure 16: Android Studios - Standard Install

    b. Select everything possible.
    c. Accept the license agreements.
4. Install the command line tools.
    a. Once the main screen of Android Studio appears, click the “More Actions” drop-down list (blue text, bottom center) and select “SDK Manager”

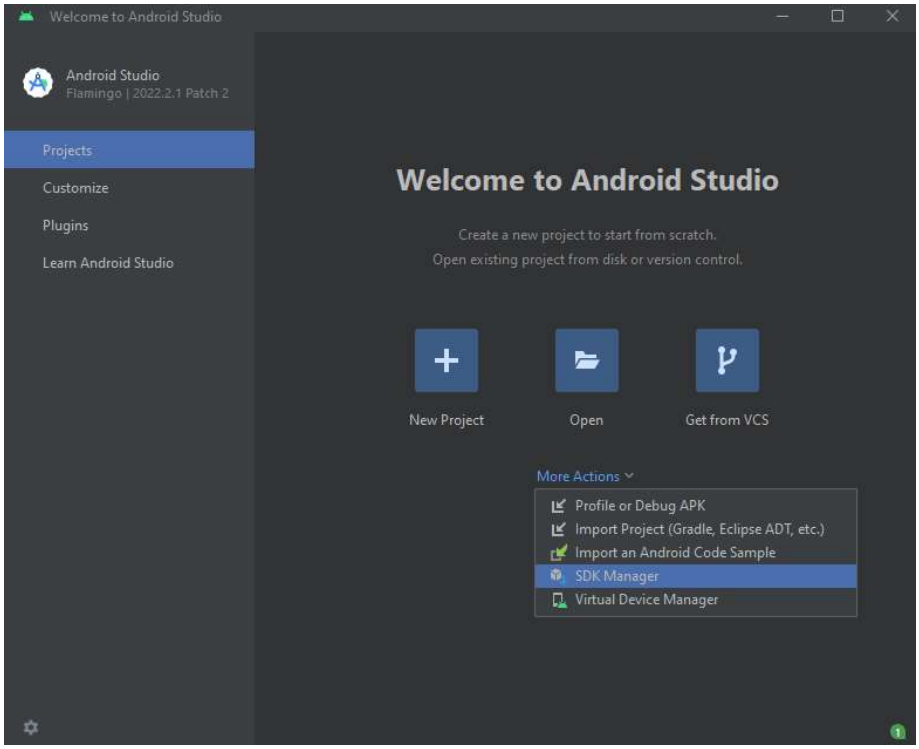


Figure 17: Android Studios - Open SDK Manager

b. Select the SDK Tools tab.
c. Check “Android SDK Command-line Tools (latest)”

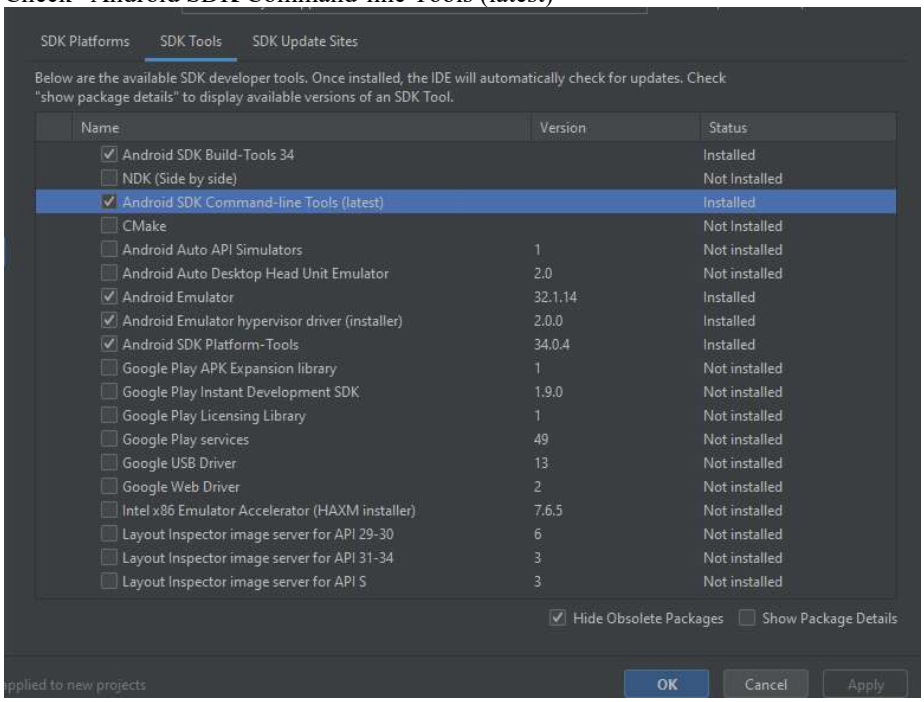


Figure 18: Android Studios - Command-line Tools

d. Click OK and then confirm the change (OK on pop-up dialog).

5. Click Finish when done.
    a. Close the app when finished.

3.1.2.5. Flutter

1. Download from https://docs.flutter.dev/get-started/install/.
    a. Select macOS, Windows, or Linux/Unix.

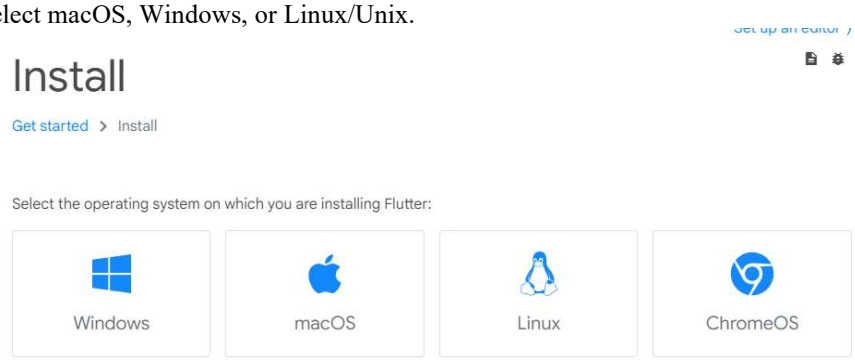


Figure 19: Flutter - Installation Options

    b. Follow the instructions on the OS specific page for downloading and/or installing.
2. Tips for Windows Installation
    a. Get the Flutter SDK

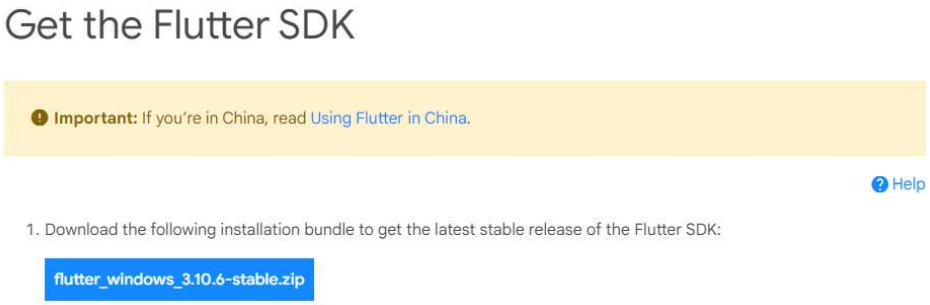


Figure 20: Flutter - Get Flutter SDK

    b. Download ZIP file from URL.
    c. Extract the contents of the ZIP file to C:\src\. This means that you should see a full path of \C:\src\flutter\bin\flutter.bat

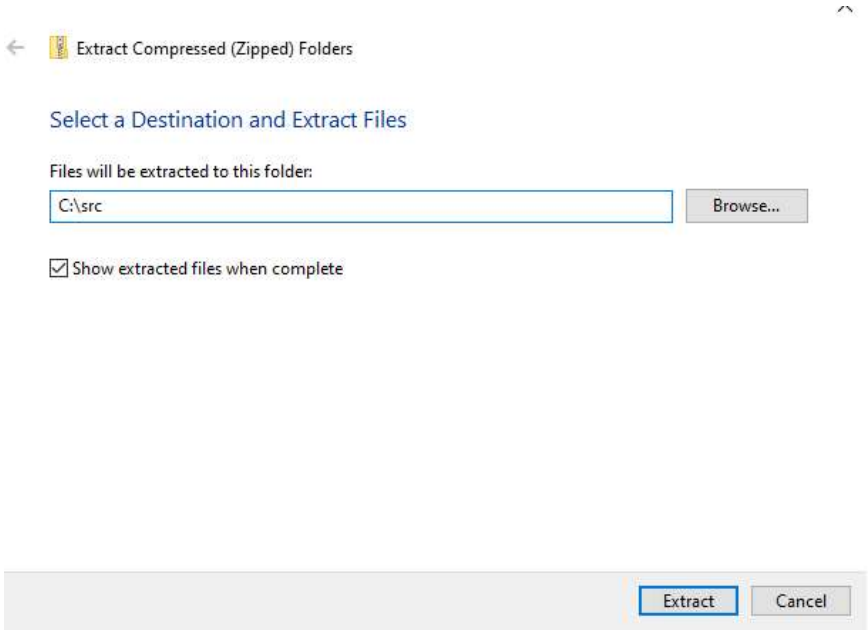


Figure 21: Flutter - Extract to C:\src

d. Update your System Environmental Variable "Path" to add the location of Flutter's bin folder.

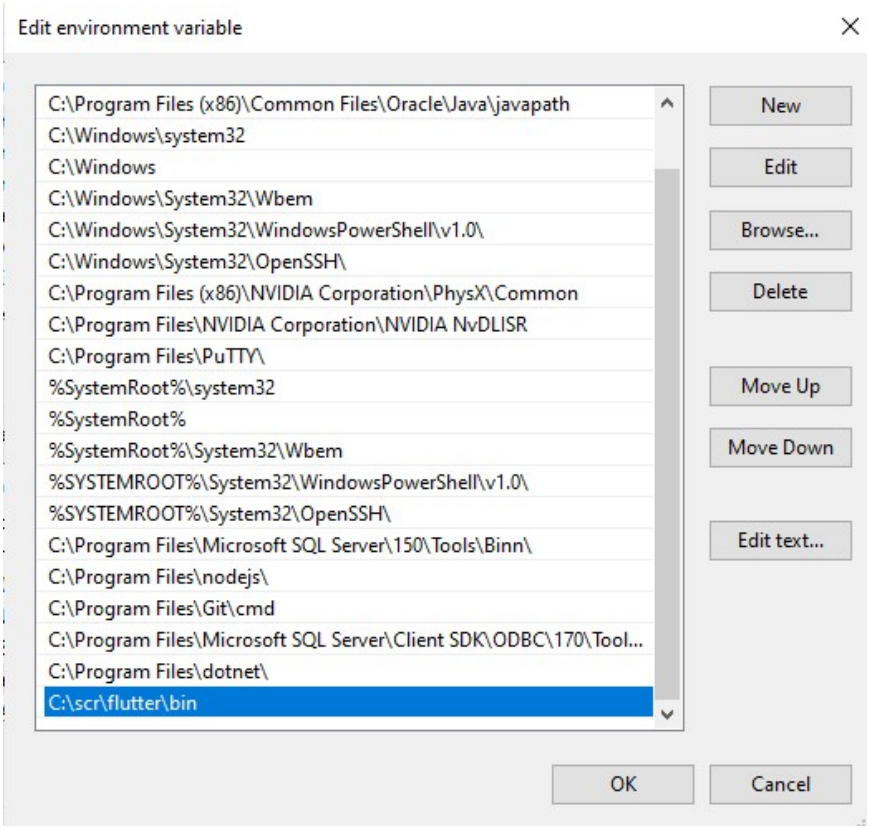


Figure 22: Flutter - Update PATH

e. If you did not have Dart installed before, then skip everything in the blue box.

f. Run Flutter Doctor

```
Run "flutter help" to see all available commands.

Want to use an IDE to interact with Flutter? https://flutter.dev/ide-setup/

Want to run the "flutter" command from any Command Prompt or PowerShell window?
Add Flutter to your PATH: https://flutter.dev/setup-windows/#update-your-path

================================================================================

C:\Users\Taylor>flutter doctor

 A new version of Flutter is available!

 To update to the latest version, run "flutter upgrade".

Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.10.5, on Microsoft Windows [Version 10.0.19045.3208], 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 for Windows (Visual Studio Community 2022 17.6.5)
[√] Android Studio (version 2022.2)
[√] Connected device (3 available)
[√] Network resources

• No issues found!
```

Figure 23: Flutter - Flutter Doctor

g. To run the Flutter console, browse to C:\src\flutter\ and run flutter_console.bat

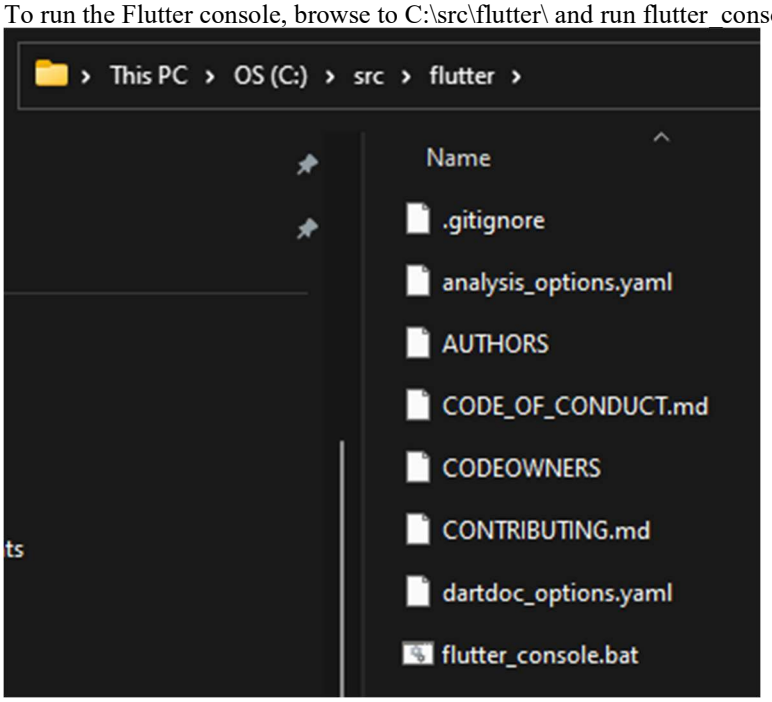


Figure 24: Flutter - Flutter Console BAT File

h. Only in the console, you can run all the commands that the webpage gives you.
i. Fix anything listed. By following all of these instructions, you might not have anything to fix except the licenses (see below).

3. Tips for Mac OS Installation
   a. Download Flutter SDK from https://docs.flutter.dev/get-started/install/macos/.
   b. Extract the files to the desired location.

c. Add the extracted path to your system's path using your terminal.
d. Note, the following command adds Flutter to the current terminal session only, if you would like to add it permanently to your path, you need to modify your .bash_profile, .bashrc, or .zshrc, file depending on your terminal setup.
e. Navigate to the root folder where the Flutter SDK was extracted to.
f. Use the following command: export PATH="$PATH:`pwd`/flutter/bin"
g. Verify Flutter is set up correctly
    i. Run: Flutter Doctor.
    ii. Correct anything that does not have a green checkmark. Common issues are the Android SDK missing, the Xcode not fully installed, and the license having not been accepted.

4. Setup your Android device
    a. This is not recommended unless you have a throw-away device (one that you will not mind doing a factory reset on it regularly).
5. Setup the Android emulator
    a. A more developer-friendly way to test a mobile app.
    b. The Device Manager icon is in Android Studio under the same "More Actions" drop-down list you used for the SDK Manager. It is called "Virtual Device Manager" now.

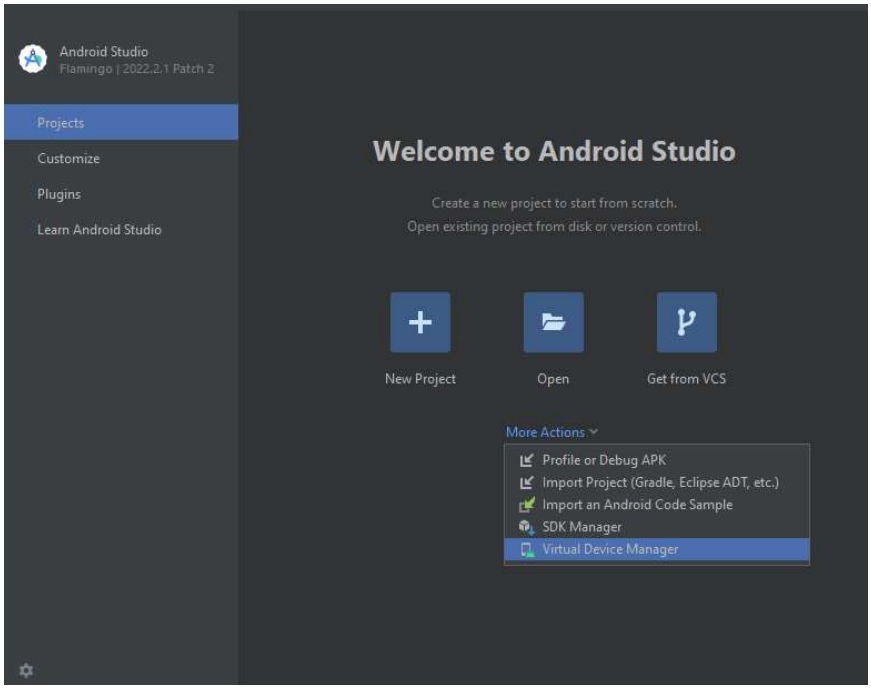


Figure 25: Android Studio - Virtual Device Manager

c. Choose the "Phone" device definition (1080 width x 1920 height).

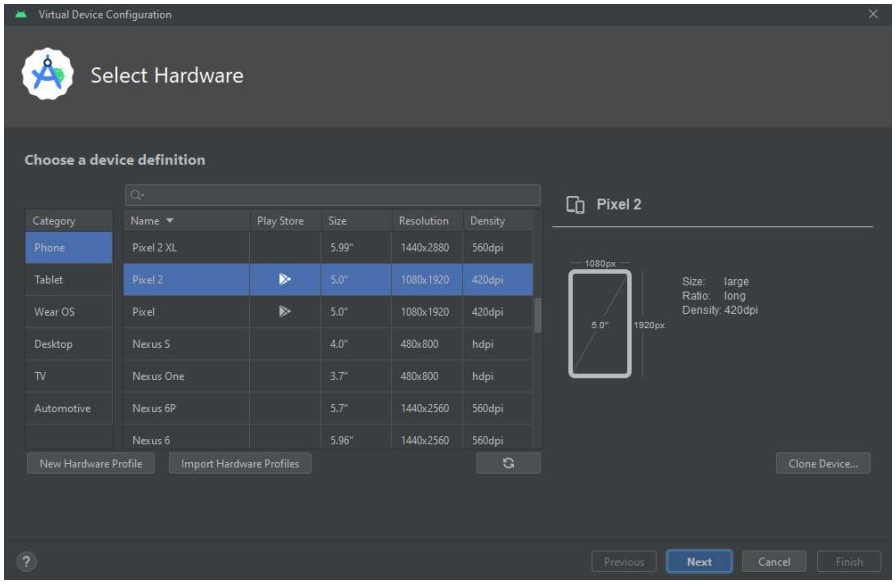


Figure 26: Android Studio - Phone Device Definition

d. Choose the system image with API level 31 (the first API level with x86_64 ABI).

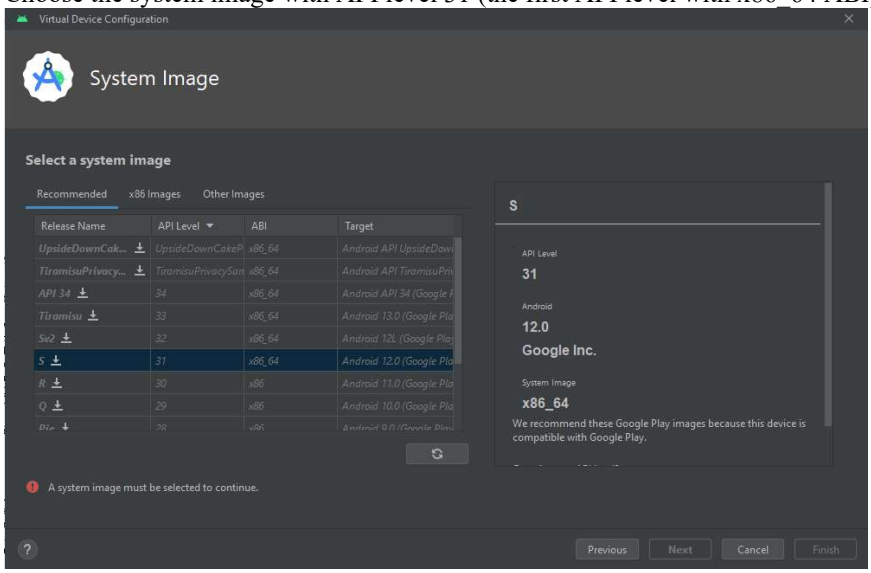


Figure 27: Android Studio - System Image

e. Click the download icon to download the image. A separate window opens for the download. When done, click Finish to close this download window.

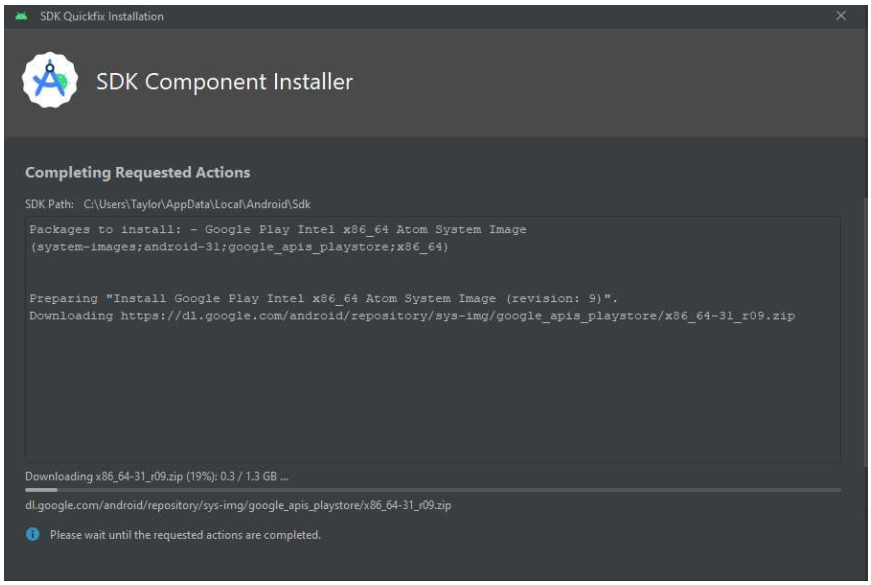


Figure 28: Android Studio - SDK Component Installer

f. The Next button should now be enabled. Click it to continue.

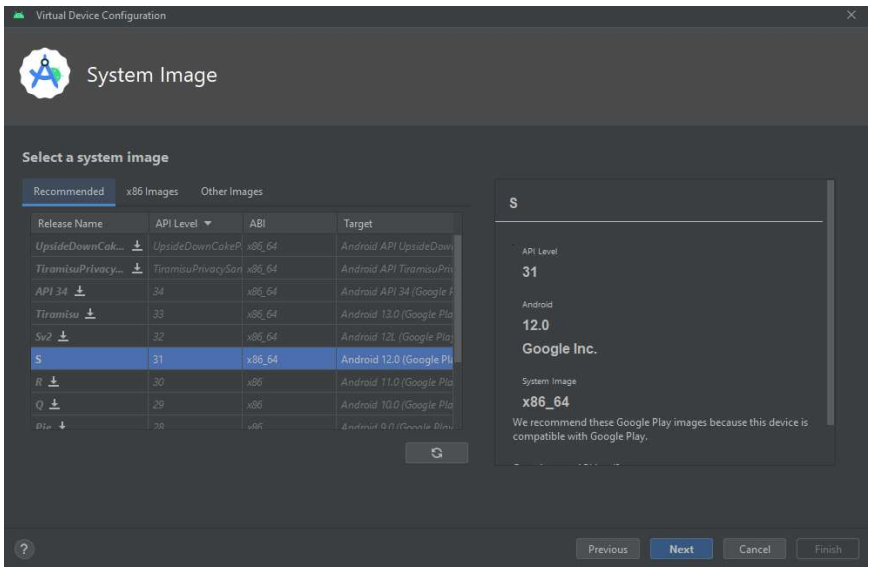


Figure 29: Android Studio - Next Button Enabled

g. On Android Virtual Device, leave default settings and click Finish.

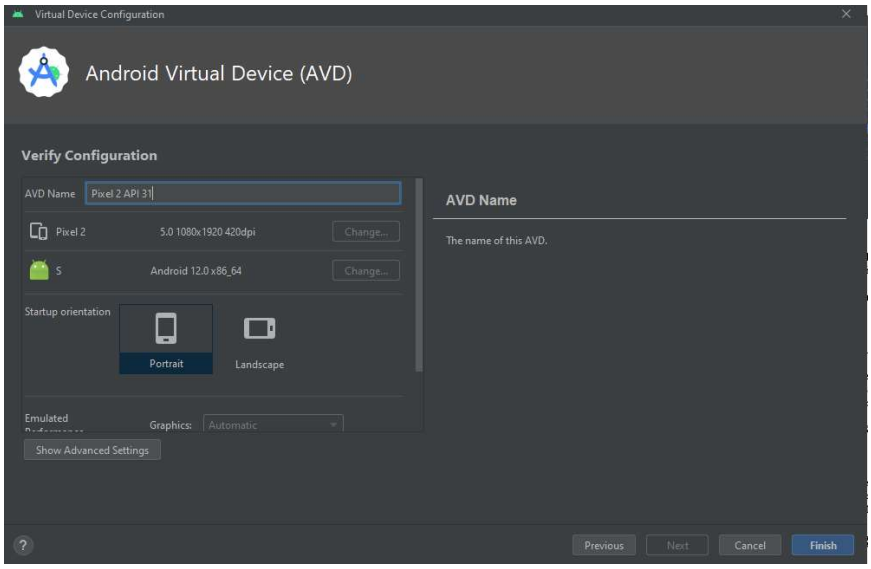


Figure 30: Android Studio - Finish Device Creation

h. You can now close the Virtual Device Manager window.

6. Agree to Android Licenses
    a. Again, make sure you are in the Flutter console (see above).

3.1.2.6. Dart SDK

1. Update the Dart SDK to the stable version 3.0.5.
2. Browse to https://dart.dev/get-dart/archive.
    a. Select the version and OS under the Stable Channel section of the page.

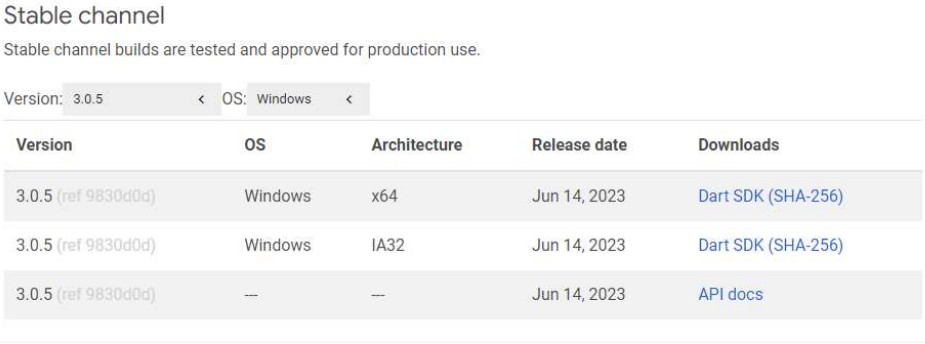


Figure 31: Dart - Stable Version Selection

    b. Click the Download link for the OS Architecture that matches your system.
    c. Save the ZIP file to a temporary location.
3. Extract the ZIP file.
4. Get/Find the location of your existing Dart SDK folder.
    a. If you installed Flutter as per above, then it will be at C:\src\flutter\bin\cache\dart-sdk\.
    b. Rename this folder.

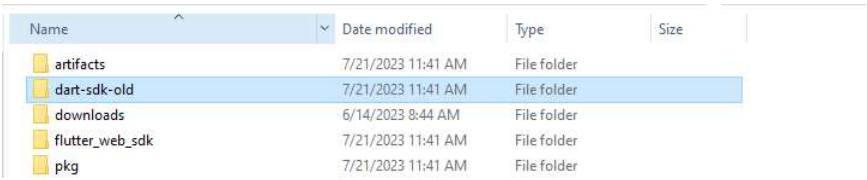


Figure 32: Dart - Rename Old SDK

5. Copy the contents of the ZIP file to the current location of your Dart SDK.
    a. When all done, the ZIP file's contents and the folder they are in will have the same path and folder name as the original Dart SDK folder.

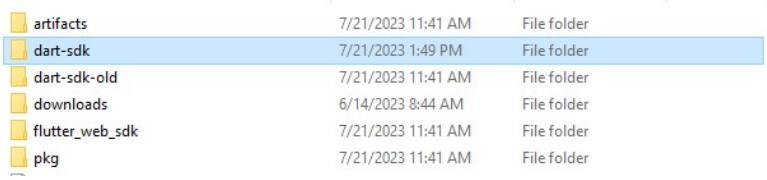


Figure 33: Dart - Rename New Folder

3.1.3. Program Setup and Configuration

3.1.3.1. Android Studio IDE Setup

1. Open Android Studio.
    a. If a project auto-opens, then close the project (File menu>Close Project).
2. On the left side, click Plug-ins.

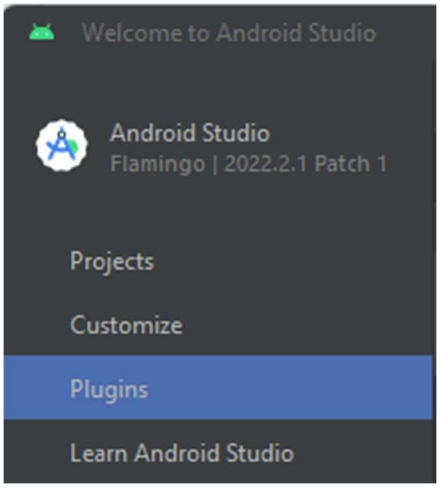


Figure 34: Android Studio's Plugins Location

3. Select the "Marketplace" tab at the top center.

Figure 35: Android Studio - Marketplace Tab

4. Install several plugins:
    a. Search for "Dart" by JetBrains and install it.

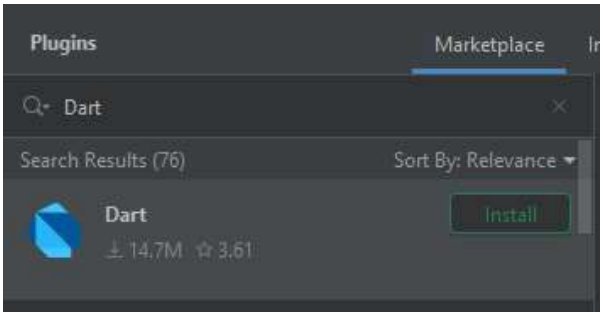


Figure 36: Android Studo - Install Dart Plugin

b. Search for “Flutter” by flutter.dev and install it.

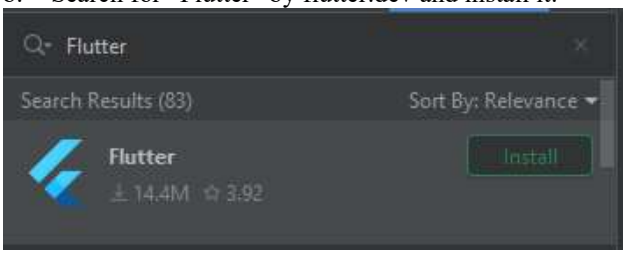


Figure 37: Android Studio - Install Flutter Plugin

5. Restart the Android Studio IDE.

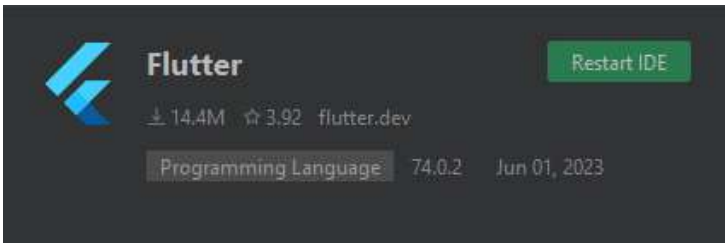


Figure 38: Android Studio - Restart IDE Post Install

6. On Mac OS only:
    a. Go to IntelliJ -> Settings -> plugins and search for “Flutter” in the plugins Marketplace.
    b. Install the plugin, reload IntelliJ.

3.2. Source Control

- Clone the “summer2023” repository.
- Branch the ConvoBuddy project from https://github.com/umgc/summer2023/tree/development/talker-mobile-app.
- If using the default locations in Windows, this project is under %appdata%\<user name>\StudioProjects\summer2023.
- All references to the project’s ConvoBuddy folder mean this location.

3.3. Compiling

1. In Android Studio, open the Terminal window.
2. Browse to the talker-mobile-app folder.
3. Run “flutter pub get” to get all dependencies.
    a. Dependencies can be found in pubspec.yaml under the dependencies section.

4. Run “flutter build apk --split-per-abi” to build the APK files for several different OS types (arm64, x86_x64, and armabi).

3.4. Deployment

3.4.1. Emulator

5. Make sure an emulator is open.
6. Do either of the below:
    a. Click the run/play button on your IDE.
    b. Run flutter run lib/main.dart in the terminal window.

3.4.2. Physical Phone

3.4.2.1. Android

1. In Android Studio, open the Terminal window.
2. Browse to the talker-mobile-app folder.
3. Enter “flutter build apk --split-per-abi” and press enter.
    a. This could take around 5 minutes to run.
4. Physically connect your phone to the machine running Android Studio.
5. Using a file explorer, browse to <project folder>\talker-mobile-app\build\app\outputs\apk\release\.
    a. The “talker-mobile-app” folder is the same folder as in step 2 above.
6. Search online for which type of APK file your phone needs.
7. Copy the matching APK file to the phone.
8. On the phone, open the built in file explorer app and go to the location that the APK file was copied to.
9. Open the file to start the installation.
10. Security prompts will vary by OS and OS version.
11. Because the app is not signed, there might be additional checks and warnings.

3.4.2.2. IOS

1. To build and release the app for iOS, one must have:
    a. macOS
    b. Xcode installed
    c. An Apple developer license
2. Open a terminal window and browse to the talker-mobile-app folder.
3. Run “flutter build ipa”.
4. This results in an App Store app bundle (.ipa file) in folder ./build/ios/ipa.
5. Note: If not distributing to the App Store, the option “–export-method development” can be appended to the “flutter build ipa” command to use for development and/or testing.
6. Connect the iOS device to the Mac.
7. Open the App Store on the Mac.
8. Click “Add”
9. Select “Apps” from the drop down
10. Select “Choose from my Mac”.
11. Find the ipa file on the Mac, and then click Add.
12. Once the displayed loading bar completes, the IPA file will be ready for use on the device and the app is installed.

4. BESie

4.1. Machine Setup

4.1.1. Program Installation

4.1.1.1. Git

1. See Git for details.

4.1.1.2. Java IDE

1. The latest versions of VS Code, NetBeans, and Eclipse are all acceptable.
2. Maven is required. If needed, it can be downloaded from https://maven.apache.org/index.html.

4.1.1.3. Java SE Development Kit

1. Go to https://adoptium.net/temurin/releases/?version=20.
2. For Windows, download the MSI file.

Figure 39: MSI File Installation

3. Run the MSI file.
4. During the Custom Setup screen, the "Set JAVE_HOME variable" must be changed from a red X to " Will be installed on the local hard drive".

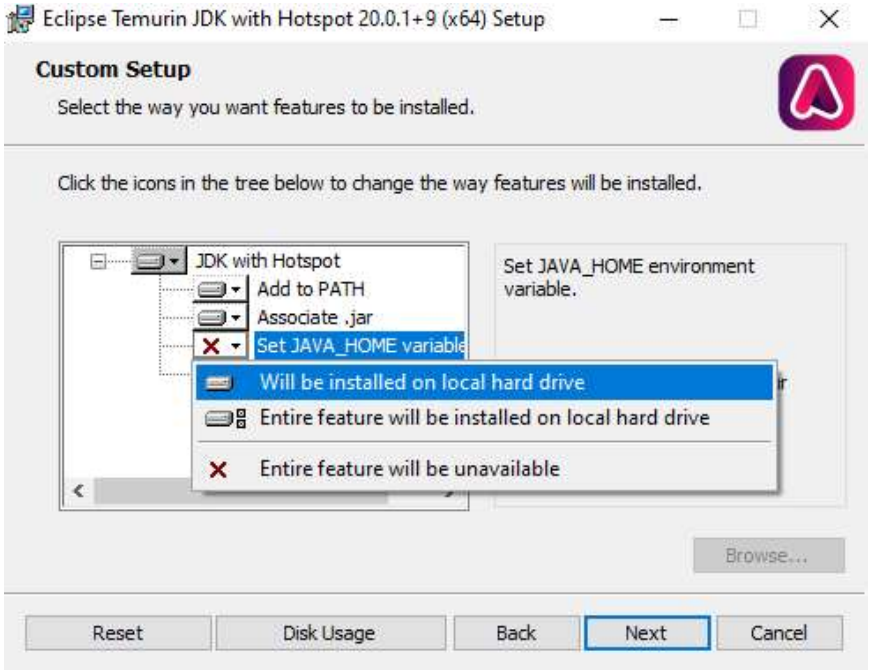


Figure 40: Temurin - Set JAVA_HOME Variable

5. Take all other default settings to finish the installation.

4.1.2. Program Setup and Configuration

1. Open Windows Command prompt
2. Type "java -version".
3. You should see "Temurin-20" in returned text.

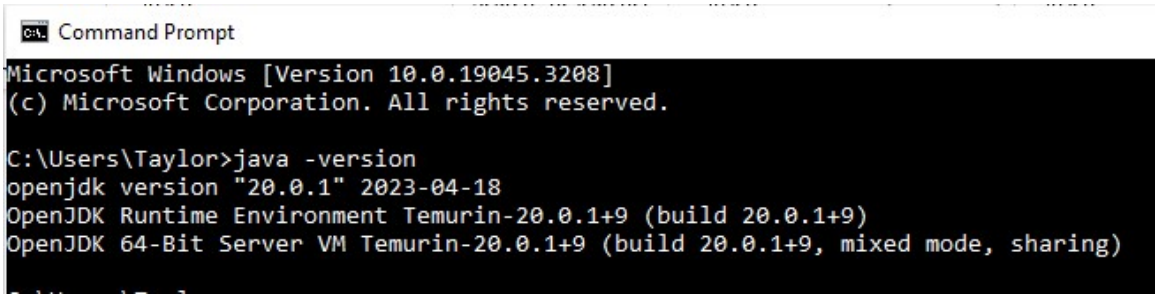


Figure 41: Temurin- Confirm Installation

4.2. Source Control

- Clone the “summer2023” repository.
- Branch the BESie project from https://github.com/umgc/summer2023/tree/development/BESie.
- If using the default locations in Windows, this project is under %appdata%\<user name>\StudioProjects\summer2023.
- All references to the project’s BESie folder mean this location.

4.3. Compiling

1. Open Windows Command prompt.
2. Browse to the project's BESie folder.
3. Run “mvnw clean package” to build the project.

4.4. Deployment

4.4.1. Dev Environment

1. Open Windows Command prompt.
2. Browse to the project's BESie folder.
3. Run “mvnw spring-boot:run”.

4.4.2. Testing Dev Deployment

1. Go to "http://localhost:8080/" in your browser.
2. Open Chrome's Developer tools (Ctrl-Shift-I) or via the menu:
    a. Click the kebob menu in the top right corner of Chrome.
    b. Select “More Tools”.
    c. Select “Developer Tools”.

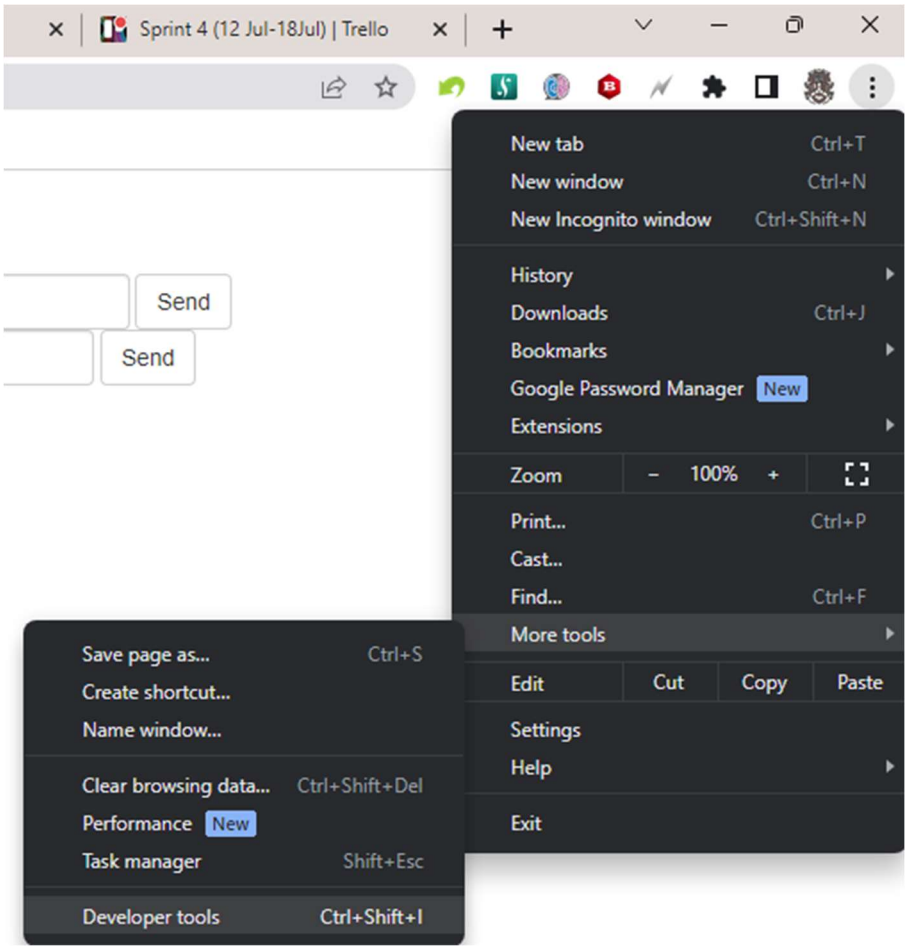


Figure 42: Menu Location of Chrome's Developer Tools

3. Go to the Console tab.
4. On the webpage click "Connect".
    a. Text should display in the Console tab showing "<<< CONNECTED", "Connected: CONNECTED" and ">>> SUBSCRIBE".
5. Type in anything into the "Send this to app/fill endpoint:" textbox and click Send.
    a. The payload should display under the horizontal bar.

4.4.3. Prod Environment

1. These instructions assume some familiarity with AWS.
2. Create an AWS AMI image according to the following specification. This image will contain a yum Linux repository to install any additional utilities, JDK 17, and secret keys to access BESie's GitHub repository.


```
{
  "Images": [
    {
      "Architecture": "x86_64",
      "CreationDate": "2023-07-16T10:43:40.000Z",
      "ImageId": "ami-067c82129896fe470",
      "ImageLocation": "791669188810/BESie",
      "ImageType": "machine",
```

```
        "Public": false,
        "OwnerId": "791669188810",
        "PlatformDetails": "Linux/UNIX",
        "UsageOperation": "RunInstances",
        "State": "available",
        "BlockDeviceMappings": [
          {
            "DeviceName": "/dev/xvda",
            "Ebs": {
              "DeleteOnTermination": true,
              "Iops": 3000,
              "SnapshotId": "snap-00b290b1e36f92fb0",
              "VolumeSize": 30,
              "VolumeType": "gp3",
              "Throughput": 125,
              "Encrypted": false
            }
          }
        ],
        "Description": "Machine that runs BESie for UMGC's Grad School Project",
        "EnaSupport": true,
        "Hypervisor": "xen",
        "Name": "BESie",
        "RootDeviceName": "/dev/xvda",
        "RootDeviceType": "ebs",
        "SriovNetSupport": "simple",
        "Tags": [
          {
            "Key": "Name",
            "Value": "BESie image"
          },
          {
            "Key": "application",
            "Value": "BESie"
          }
        ],
        "VirtualizationType": "hvm",
        "BootMode": "uefi-preferred",
        "ImdsSupport": "v2.0"
      }
    ]
}
```

3. From the EC2 console, go to Images > AMIs

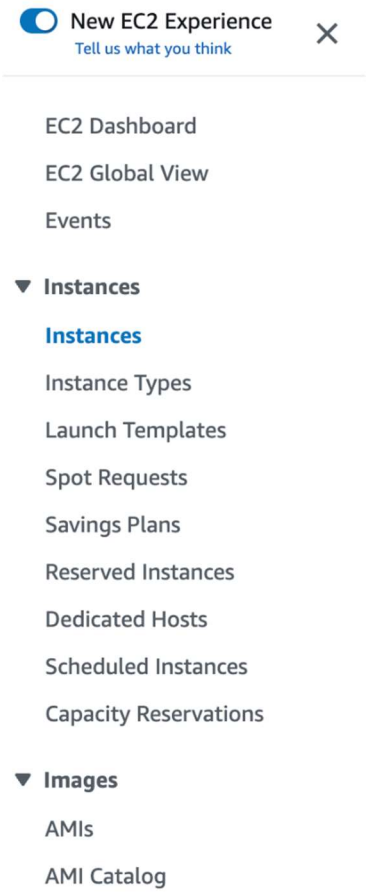


Figure 43: AWS's AMI Menu Location

4. Select "Launch Instance" from AMI and wait for the service to come up.

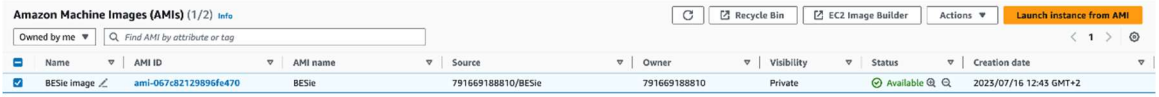


Figure 44: AWS's List of AMI's

5. Once the instance is ready, in the AWS console, go to running instance and copy the public IP.
6. http://<Public IP>:8080/ is how BESie can be accessed by supporting applications.

5. Browser Extension

5.1. Machine Setup

5.1.1. Program Installation

5.1.1.1. Text Editor

A text editor will be required to view and edit the source files. The preferred text editor is Visual Studio Code. The latest version should be downloaded from https://code.visualstudio.com/download. Visual Studio Code is lightweight and powerful as it is designed to assist developers. However, any text editor will do if needed.

5.2. Source Control

- Clone the "summer2023" repository.
- Branch the form-filler-extension project from https://github.com/umgc/summer2023/tree/development/form-filler-extension.
- If using the default locations in Windows, this project is under %appdata%\<user name>\StudioProjects\summer2023.
- All references to the project's browser extension folder mean this location.

5.3. Compiling

Not applicable because this is written in JavaScript and HTML.

5.4. Configuration

The browser extension must be configured to point to the correct instance of BESie (DEV or PROD). The default is for PROD. The below steps

1. In a text editor, open form-filler-extension/js/utils.js.
2. Find the serviceUrl constant.
3. Change the serviceUrl:
4. DEV => http://localhost:8080/ws
5. PROD => http://44.202.25.184:8080/ws
6. There are 2 constants, ServiceURL_DEV and ServiceURL_PROD, that should be used to set the value of ServiceURL.

5.5. Deployment

5.5.1. Developer Environment

1. In Chrome, enter chrome://extensions/ in the address bar.
2. Enable "Developer mode" in the top right corner.
3. First time steps:
4. Click Load unpacked and then select the folder where the extension is located. For example: "C:\Users\userid\Documents\summer2023\form-filler-extension".

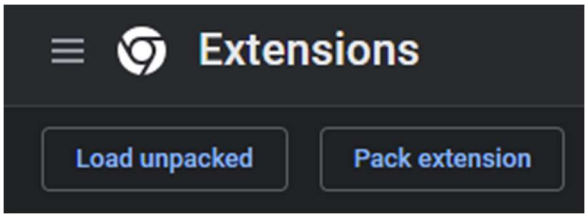


Figure 45: The "Load unpacked" Button

5. At this point, the extension should have been installed into your browser and you should see its icon in your browser's extensions toolbar.
6. Loading new code after first install:
7. Find ConvoBuddy in the list of extensions.
8. Click the reload button inside the extension's box. It looks like this

Figure 46: Browser Extension's Reload Button

5.5.2. Production Environment (Chrome Web Store)

1. Create a zip file of the browser extension. The zip file must contain the manifest file which must specify the following:
    a. Name
    b. Version
    c. Icons
    d. Description
2. Register as a Chrome Web Store Developer (first time only).
    a. Go to https://chrome.google.com/webstore/devconsole
    b. Sign in to your Google account
    c. Accept the Developer Agreement and Privacy policies
    d. Pay the $5 registration fee.
3. Set up your Web Store Developer Account (first time only).
    a. Go to the Account page.
    b. Type in your publisher/display name, email address, and physical address.
    c. Verify your email address.
4. Upload your Browser Extension.
    a. Go to the main page of the Chrome Developer Dashboard.
    b. Click Add New item.
    c. Click Choose file.
    d. Select your zip file.
    e. Click Upload.
5. Submitting the Browser Extension for publishing.
    a. On the Listing tab, fill in basic information about the extension such as name, description, category, language, and version number.
    b. On the Privacy tab, include declarations about privacy and security features.
    c. On the Distribution tab, you declare if the extension is free or contains in-app purchases. You also determine the extension visibility (public, unlisted, private).
    d. Click Submit for Review.
        i. You can choose to automatically publish the extension after it has passed the review.

6. Automation Testing

6.1. Machine Setup

6.1.1. Requirements

- Windows 10 or later (64-bit)

6.1.2. Program Installation

6.1.2.1. Android Studio

1. See Android Studio Flamingo for details.

6.1.2.1. Java IDE

1. See Java IDE for details.

6.1.2.2. Maven

1. Follow the instructions on https://phoenixnap.com/kb/install-maven-windows.
2. Completed all steps except substituting download Maven from the official site: https://maven.apache.org/download.cgi.
3. Make sure the verification step completes successfully:

```
C:\Users\Matt\StudioProjects\summer2023\automation\testng-appium-app>mvn --version
Apache Maven 3.9.3 (21122926829f1ead511c958d89bd2f672198ae9f)
Maven home: C:\Components\Development\Java\apache-maven-3.9.3
Java version: 20.0.1, vendor: Eclipse Adoptium, runtime: C:\Program Files\Eclipse Adoptium\jdk-20.0.1.9-hotspot
Default locale: en_US, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
```

Figure 47 – Maven Installation Verification

6.1.2.3. Java SE Development Kit

1. See Java SE Development Kit for details.

6.1.2.4. Appium Desktop App

2. Download the appropriate installer from https://github.com/appium/appium-desktop/releases for your machine's OS.
3. Run the downloaded file.
    a. The below screen shots are for the Windows EXE (Appium-Server-GUI-windows-1.22.3-4.exe)
    b. Use the default settings and click Next.

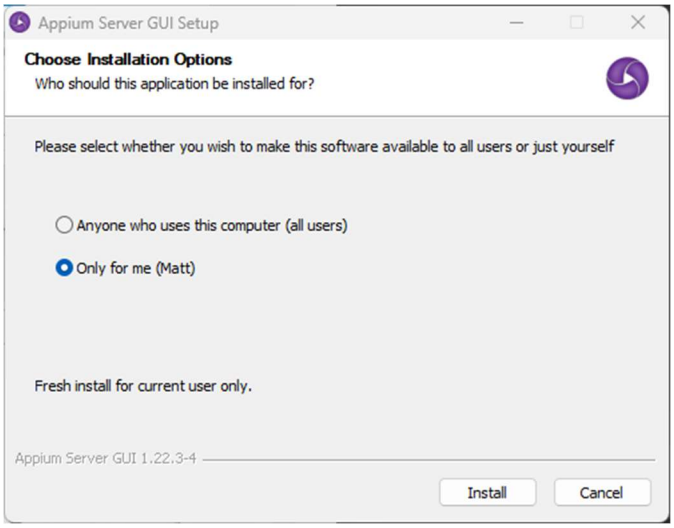


Figure 48 – Appium Server GUI Installer Options

c. Click Finish

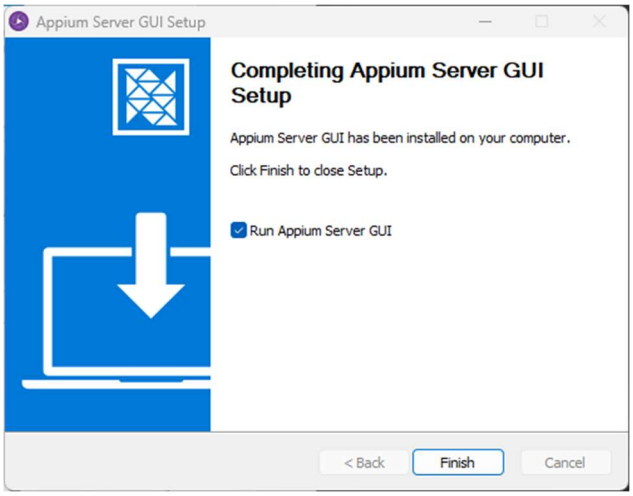


Figure 49 – Appium Server GUI Final Installation Window

6.1.3. Program Setup and Configuration

6.1.3.1. Android SDK

4. Set the ANDROID_HOME system variable.
5. This varies by OS and location that the Android SDK was installed.
6. Android Studio installs the SDK automatically.
7. Windows
    a. Open System Properties.

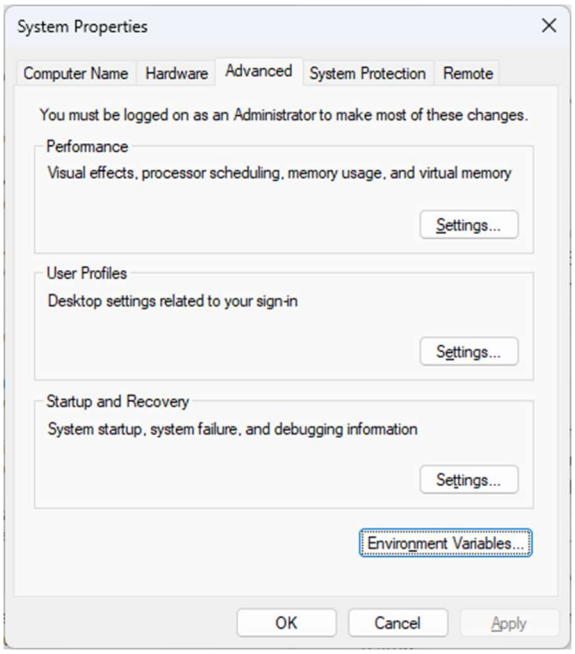


Figure 50 – Windows System Properties Dialog

b. Click Environment Variables.

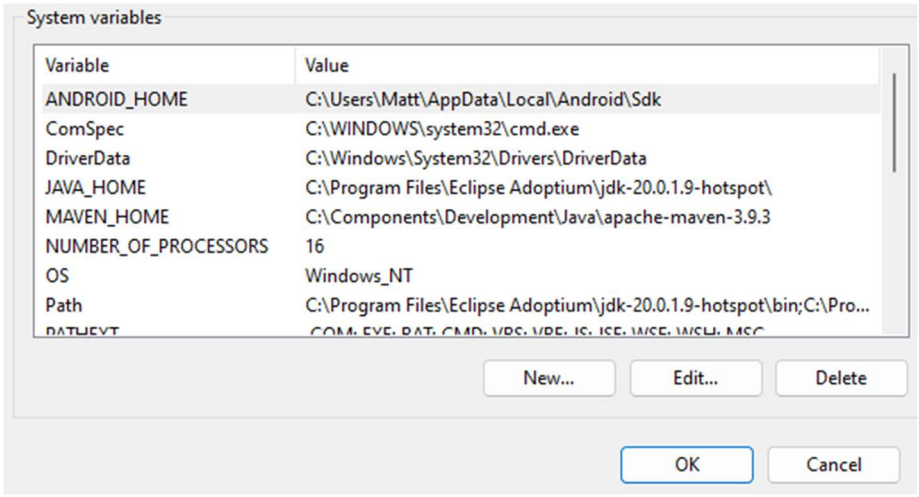


Figure 51 – Windows Environmental Variables Dialog

c. Click New.
   i. Enter “ANDROID_HOME” for “Variable name”.
   ii. Enter “C:\Users\<username>\AppData\Local\Android\Sdk” for Variable value. Replace <username> with your Windows user name.
   iii. Click OK.
d. Find “Path” in the System variables portion of the dialog.
e. Select “Path” and click Edit.

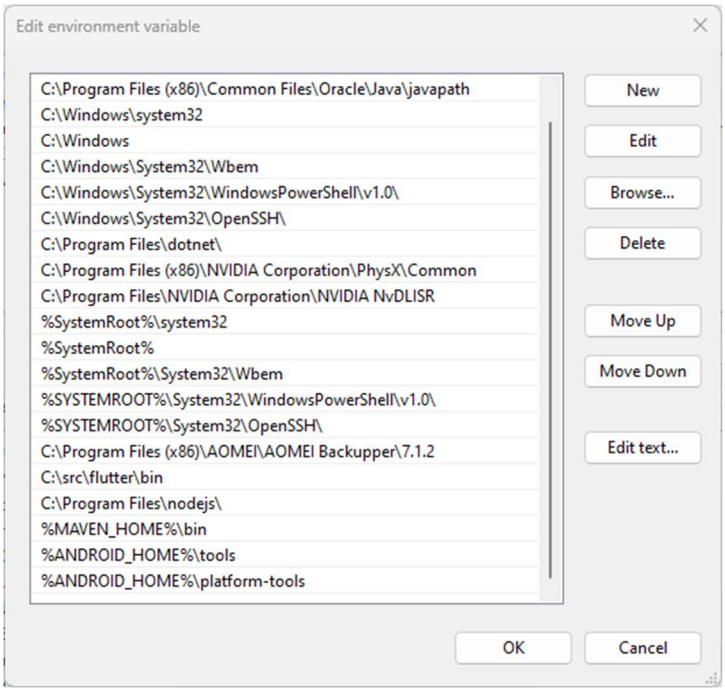
f. 

Figure 52 – Windows Edit Environmental Variables Dialog

i. Click New.
ii. Enter “%ANDROID_HOME%\tools”.
iii. Click New again.
iv. Enter “%ANDROID_HOME%\platform-tools”.
v. Click OK.

g. Click OK on the Environmental Variable dialog.
h. Click OK on the System Properties dialog.

6.1.3.2. Appium Server GUI

1. Launch the Appium program.
2. Use the default values and click “startServer”:

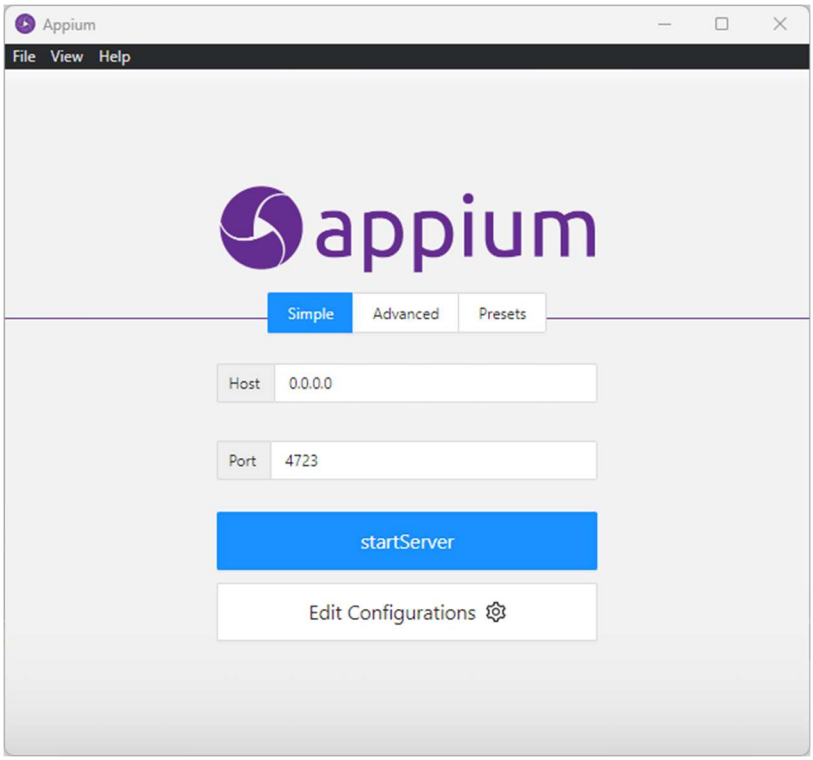


Figure 53 – Appium Server GUI Main Screen

a. If a Windows Defender security alert pop-up, select at least private network and click “Allow access”.

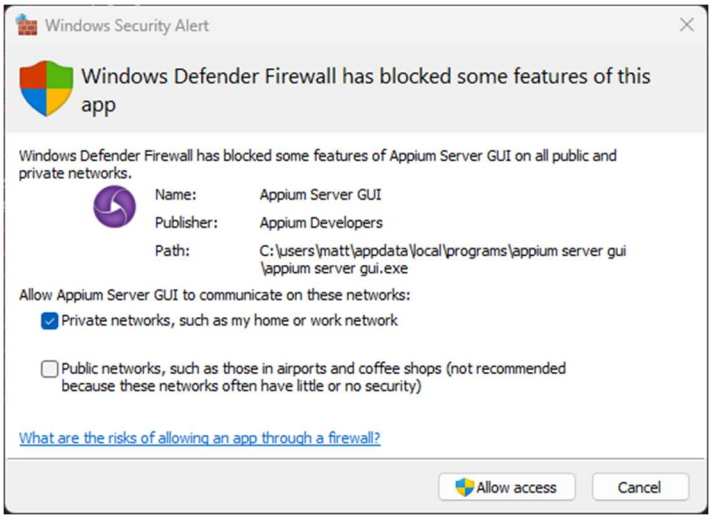


Figure 54 – Security Alert for Appium

b. The Appium service is now running:

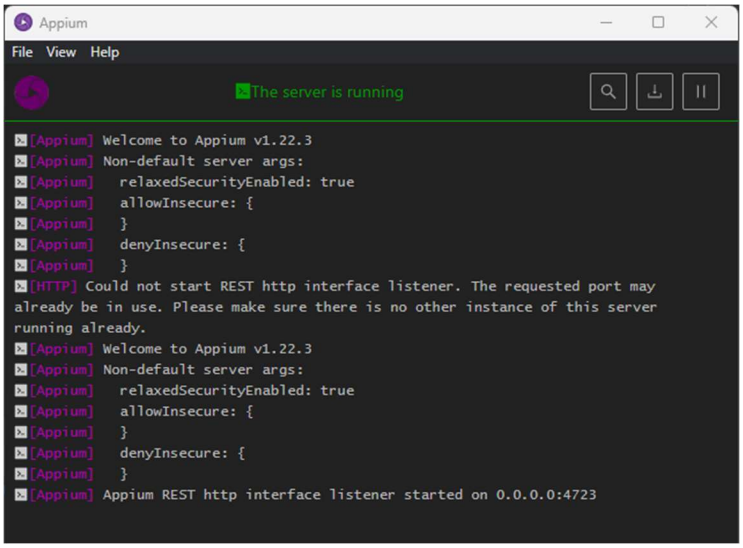


Figure 55 – Appium Server GUI Running

6.2. Source Code

- Clone the "summer2023" repository.
- Branch the automation project from https://github.com/umgc/summer2023/tree/development/automation.
- If using the default locations in Windows, this project is under %appdata%\<user name>\StudioProjects\summer2023.
- All references to the project's Automation folder mean this location.

6.3. Running Automation Tests

1. Start the Appium Server. See Appium Server GUI for details.
2. Open Android Studio.
3. Open Device Manager
    a. Run any Emulator setup for ConvoBuddy. See Deployment for details.
4. Open a Terminal window (Alt-F12).
    a. Go to the automation\testng-appium-app\android\testng-examples\ folder.
    b. Run "mvn clean install".
    c. In Android Studio, the script will compile and then run in the emulator.
5. Intellij only:
    a. Starting from the prior terminal window's location, go the \src\test\resources\com.talkerMobile\ subfolder.
    b. Open run-test.testng.xml.
    c. Right click on it and select Run.
    d. The script will start running in emulator.
6. Wait for the tests to complete.
7. Open the test report.
    a. Browse to …\summer2023\automation\testng-appium-app\android\testng-examples\target\surefire-reports\.
    b. Open index.html in a browser of your choice.
    c. The testing report will display:

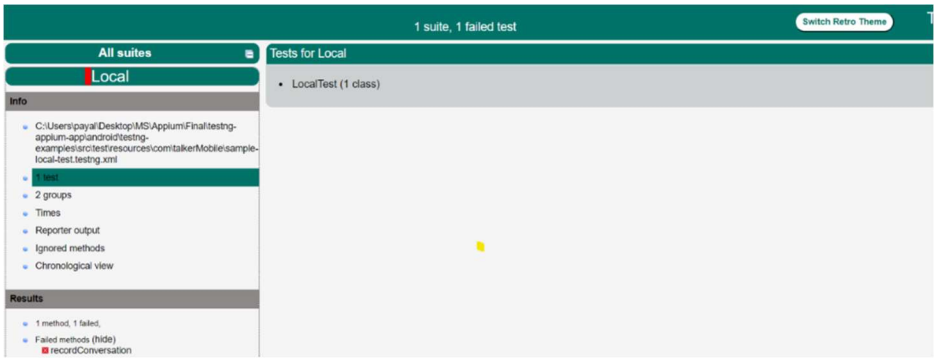


Figure 56 – Automation Testing Report

7. Troubleshooting

7.1. Android Studio Flamingo

1. If you get the error “Unable to detect Android SDK”, then possible fixes are:
    a. Resolve a firewall issue.
    b. Reboot the computer.

7.2. ConvoBuddy>Compiling

If “flutter run” returns “Error: No pubspec.yaml file found”, then change your directory to the sub-folder “talker-mobile-app” and try again.

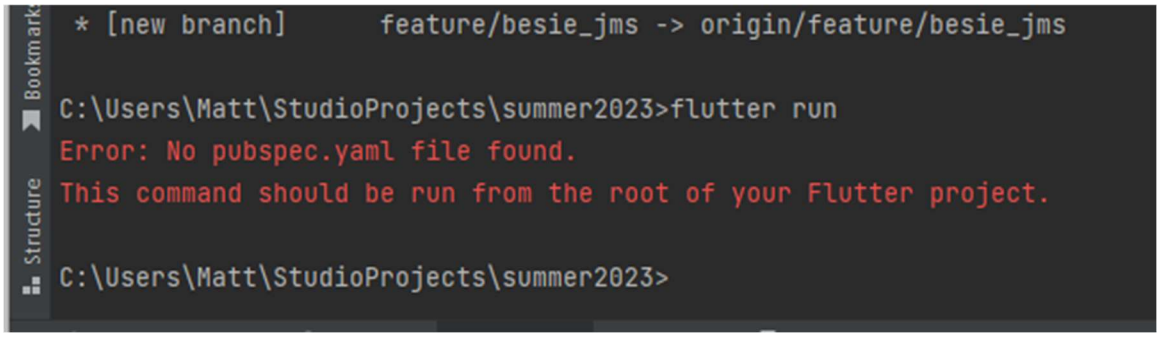


Figure 57: Screenshot of “no pubspec.yaml file found” error.

7.3. ConvoBuddy>Can not Record in Emulator

Emulators need their permissions set to have access to the host machine’s hardware. See screen shot for details of the below instructions.

1. From the emulator’s window, click the kebab menu to launch the emulator’s extended controls.
2. From the extended controls window, select Microphone on the left side.
3. Make sure “Virtual microphone uses host audio input” is turned on.
4. Click the X to close the extended controls window.
5. Recording should now work.

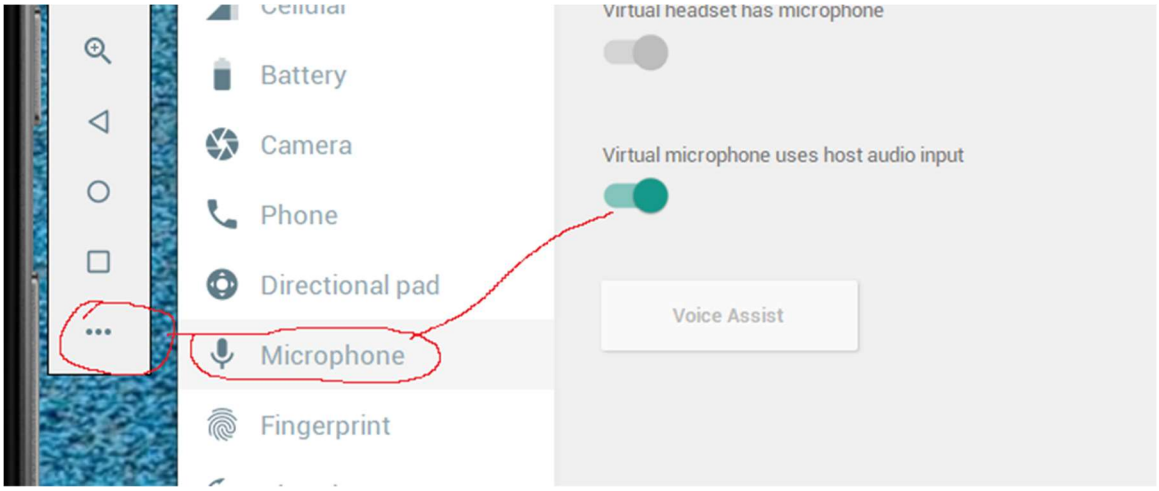


Figure 58: Extended Controls>Microphone setting

7.4. ConvoBuddy>Installation

When opening the APK file, it is possible to get pop-up warnings and messages that make it hard to install the app.

7.4.1. Unsafe App Blocked

1. A message could appear that says "Unsafe app blocked. Play protect doesn't recognize this app's developer".
2. Click “More” to see more options.
3. Click “Install Anyways” to continue the installation.

7.4.2. Developer Mode

1. The phone should be put into Developer Mode.
2. This can be found under Android’s Settings area.
    a. Exact location varies by OS version.

7.5. BESie>Deployment

If you get a pop-up showing the below screen shot (from Windows 11), select "Private networks, such as my home or work network" and click "Allow access".

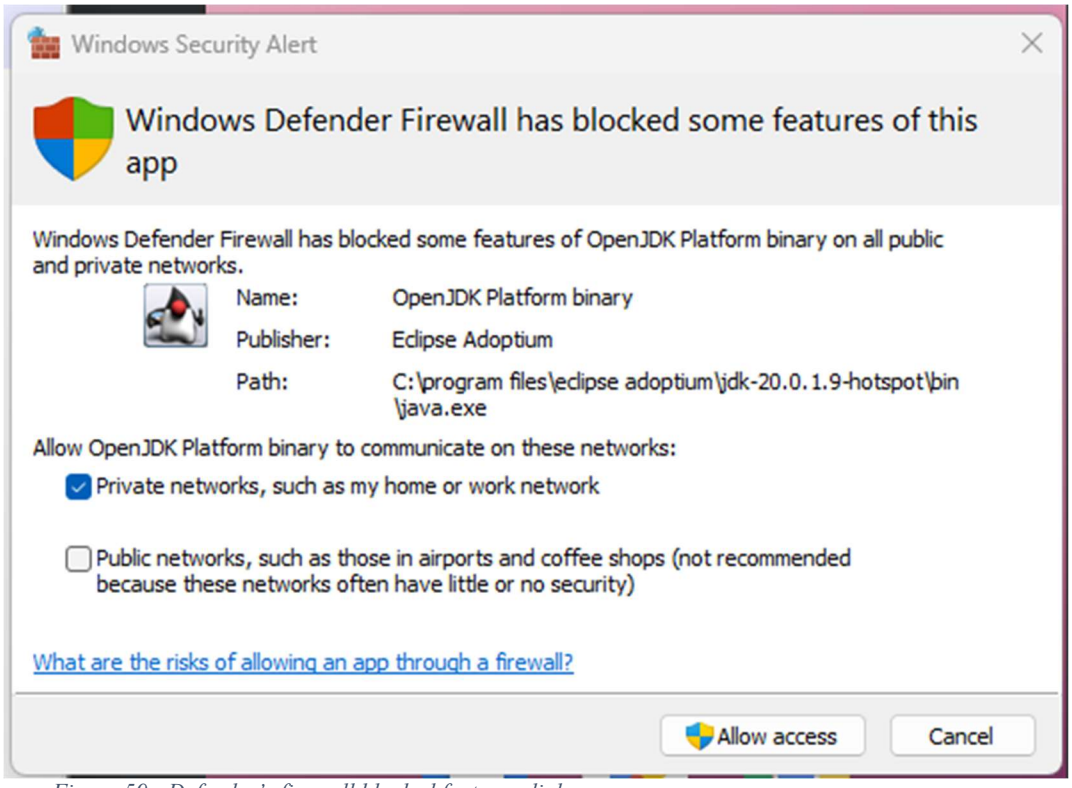


Figure 59: Defender’s firewall blocked features dialog