University of Maryland Global Campus

Presented By:

Team A: Reshawna Sampson, Barry Gartrell, Ananya Srinivasan, Stanley De Jesus, Sheena Johnpeter, Arnold Detoito

DEPLOYMENT AND OPERATIONS GUIDE  
(Runbook)

USPS Informed Delivery App- Visually Impaired

Table of Contents

[1. Introduction 3](#_Toc110975932)

[1.1 Purpose 3](#_Toc110975933)

[1.2 Intended Audience 3](#_Toc110975934)

[1.3 Technical Project Stakeholders 3](#_Toc110975935)

[1.4 Definitions, Acronyms, and Abbreviations 4](#_Toc110975936)

[2. Mobile Application 5](#_Toc110975937)

[2.1 Features, Packages, Plugins- 5](#_Toc110975938)

[2.1.1 Features 5](#_Toc110975939)

[2.1.2 Packages 6](#_Toc110975940)

[2.1.3 Plugins 7](#_Toc110975941)

[3. Software Installation 9](#_Toc110975942)

[3.1 Android Studio Installation 10](#_Toc110975943)

[3.2 Flutter and Dart System Requirements 10](#_Toc110975944)

[3.3 Dart and Flutter Plug-ins 11](#_Toc110975945)

[3.4 Android Emulator 14](#_Toc110975946)

[3.5 Visual Studio Code – Mac Install 21](#_Toc110975947)

[3.6 Dart and Flutter Plug-ins 23](#_Toc110975948)

[3.7 Xcode (iOS) 24](#_Toc110975949)

[3.8 Flutter and Dart SDKs 24](#_Toc110975950)

[3.9 GitHub Desktop setup 30](#_Toc110975951)

[4. Prepare the Mobile Application for Use 30](#_Toc110975952)

[4.1 Google Cloud Vision API setup 30](#_Toc110975953)

[4.2 Google ML Kit Barcode Scanning Plugin setup 39](#_Toc110975954)

[4.3 USPS Address Validation API setup 41](#_Toc110975955)

[4.4 Cloning GitHub repository 44](#_Toc110975956)

[4.5 Run the Flutter application 47](#_Toc110975957)

[5. Testing the Mobile Application 50](#_Toc110975958)

[5.1 Testing Objective 50](#_Toc110975959)

[5.1.1 Unit tests 50](#_Toc110975960)

[5.1.1.1 Google Cloud Vision 50](#_Toc110975961)

[5.1.1.2 Digest Email Parsing 52](#_Toc110975962)

[5.1.1.3 USPS Address Verification 53](#_Toc110975963)

[5.1.1.4 QR Code/Barcode Scanning 54](#_Toc110975964)

[6. Troubleshooting 55](#_Toc110975965)

[6.1 Issues configuring Flutter or Dart SDKs 55](#_Toc110975966)

[6.2 Emulator not responding 55](#_Toc110975967)

[6.3 Dependency Errors 57](#_Toc110975968)

[6.4 Known Issues and Solutions 60](#_Toc110975969)

[6.4.1 Google Vision Service Account - Violation of Google Cloud User Agreement 60](#_Toc110975970)

[7. Appendices 64](#_Toc110975971)

[7.1 References 64](#_Toc110975972)

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Date | Description | Approved By |
| 1.0 | 7/28/22 | Initial Document | Reshawna Sampson (PM) |
| 1.0 | 8/9/22 | Revised | Reshawna Sampson (PM) |
|  |  |  |  |

# Introduction

Installing and deploying the USPS Informed Delivery Vision Assistance Application is covered in depth in this Team Arch’s Deployment and Operation Guide (Runbook). Additionally, the documentation provides step-by-step instructions that show how the whole team members – including the Project Manager, Developers, Business Analysts, and Testers worked on, completed/accomplished, and implemented the entire application as a class during Summer 2022 at the University of Maryland Global Campus (UMGC). In addition to this document, there are a suite of documents that include the Project Plan, Software Requirements Specification, Technical Design Document, and Programmer’s Guide that can be used in conjunction with this document.

* 1. Purpose

The Deployment and Operation Guide (Runbook) is intended to lay out the process required for installing and deploying the USPS Informed Delivery Vision Assistance Application to a user’s device, as well as to provide illustrations of those actions. This document assists in outlining each step needed to complete a successful installation.

* 1. Intended Audience

Software developers, technical stakeholders, and future programmers who are interested in installing or deploying the application are the intended audience for this Runbook guide. This document’s goal is to give instructions that any user may easily follow by thoroughly outlining each stage of the installation procedure. Users may learn more about Flutter in depth by taking the [“Flutter Crash Course for Beginners 2021 - Build a Flutter App with Google's Flutter & Dart”](https://www.youtube.com/watch?v=x0uinJvhNxI&t=2407s) course offered by *Academind* on YouTube.

* 1. Technical Project Stakeholders

The table below shows the technical project stakeholders for the USPS Informed Delivery Vision Assistance Application:

**Table 1**

*Project Stakeholders*

| **Name** | **Role** |
| --- | --- |
| Dr. Mir Assadullah | Professor |
| Roy Gordon | Project Mentor |
| Robert Wilson | Project Mentor |
| Bereket Kelemu | Overall Project Manager |
| Reshawna Sampson | Team Arch’s Project Manager |
| Stanley De Jesus | Lead Developer |
| Barry Gartrell | Developer |
| Arnold Detoito | Developer/Tester |
| Ananya Srinivasan | Developer/Business Analyst |
| Sheena Johnpeter | Business Analyst/ Tester |

* 1. Definitions, Acronyms, and Abbreviations

The table below shows the most used acronyms/abbreviations and their definitions:

**Table 2**

*All Acronyms and Abbreviations*

| **Acronyms and Abbreviations** | **Definitions** |
| --- | --- |
| PM | Project Manager |
| BA | Business Analyst |
| UI | User Interface |
| OS | Operating System |
| iOS | iPhone Operating System |
| API | Application Program Interface |
| VM | Virtual Machine |
| CE | Code Editor |
| GCP | Google Cloud Services Platform |
| HI | Hardware Interface |
| IDE | Integrated Development Environment |
| SI | Software Interface |
| UML | Unified Modeling Language |
| UMGC | University of Maryland Global Campus |
| QR | Quick Response |

# Mobile Application

* 1. Features, Packages, Plugins-
     1. Features
* **Simplicity**: The USPS Informed Delivery Vision Assistance Application’s user interface (UI) is well-organized, user-friendly layout, and simple to use.
* **Flexibility**: Compatible with Android and iOS operating systems.
* **Android and iOS Devices Support:** Offering mobile applications that are cross-platform or multi-platform is one of the finest ways to attract customers. With this feature, users will have flexibility in when and how they can access the program.
* **Security**: Provides users the security they need to deal with storing sensitive and personal data.
* **Login:** Logins will be available for Visually Impaired users, allowing them to choose which features they user has access to.
* **Search Options:** Ensures that the application allows users to search for the information they have entered.
* **Audible feature:** From the informed delivery daily digest, the application audibly reads the following:
* Logos
* Type or read the handwritten text to find the sender and recipient information
* Reads Quick Response (QR) codes or barcodes on mail pieces. In addition, it also encourages users to visit those websites for more information while they are using the application.
* **Voice Command Feature:** It is trained to recognize the user’s voice. When a new USPS delivery digest email is received, the application is programmed to alert the user and offer voice command action choices.
* **Mic button Feature:** It will recognize the user’s voice and distinct phrases, which will cause the program to do tasks.
* **Scan an image feature:** Through the phone’s camera, this feature is programmed to scan a piece of mail and read the contents aloud.
  + 1. Packages

Both the Flutter and Dart ecosystems allow the usage of common packages made by other programmers, allowing for the quick creation of applications without having to start from scratch. Programmers may quickly develop an application using resources offered by other programmers with the aid of ***Flutter packages.*** These packages are available on [**pub.dev,**](https://pub.dev/) where developers often make their work downloadable. On this page, individuals can find a lot of packages and dependencies that work with and may be utilized with Flutter.

**Installing a Package Dependency Into an App:**

**Depend on it:**

* Open the ***pubspec.yaml*** file from within the application and add ***css\_colors:*** under dependencies.

**Install it:**

* From the terminal: ***Run flutter***pub get. **OR**
* **From Android Studio/IntellJ:** At the top of pubspec.yaml, select packages to open the action ribbon.
* **From VS Code:** Click ***“Get Packages”*** located on the right side of the action ribbon at the top of pubspec.yaml.

**Import it**

* In the Dart code, add a similar import statement.

**Stop and restart the application, if necessary**

* Platform-specific code (Kotlin/Java for Android, Swift/Objective-C for iOS) that is included in a package must be included into your application.
* A complete restart of the application might be necessary to prevent problems like ***MissingPluginExecption*** while using the package because hot reload and hot restart only update the Dart code.

Note: An easy reference of these processes may be found under the [Installing tab](https://pub.dev/packages/css_colors/install) on every package page on pub.dev.

The main **Google ML Kit packages** utilized by the USPS Informed Delivery Vision Assistance Application are listed in the table below:

Note: Refer<https://github.com/bharat-biradar/Google-Ml-Kit-plugin/tree/master/packages/google_mlkit_barcode_scanning> for a complete list of ML Kit Barcode Scanning packages.

2.1.3 Plugins

Depending on the platform being utilized, the installation steps for the Flutter and Dart plugins differ:

**macOS Installation:**

For macOS, utilize the guidelines listed below:

1. Start ***Android Studio*** first
2. Launch the plugin option by selecting ***“Plugins”*** in Android Studio (Android Studio >Preferences>Plugins).
3. After choosing the Flutter plugin, select ***Install.***
4. To install the Dart plugin, select ***Yes*** when prompted.
5. Select ***Restart*** when requested.

**Windows Installation**

For Windows, utilize the guidelines listed below:

1. Open ***plugin*** preferences.
2. Search for the ***necessary*** plugin.
3. Read and ***“Apply”*** the privacy notice.
4. Select the ***“OK”*** button.
5. To make modifications take effect when installation is finished, select ***“Restart.”***

**Table 3**

*Plug-ins, Packages and Descriptions*

| **Packages** | **Description** |
| --- | --- |
| Google\_ml\_kit | A Flutter plugin to use all APIs from Google’s standalone ML Kit for mobile platforms. |
| Google\_mlkit\_barcode\_scanning | A Flutter plugin to use Google’s ML Kit Barcode Scanning to read data encoded using most standard barcode formats. |
| Global\_configuration | A Flutter package for managing different configurations by merging them together and making them available everywhere inside the app via a singleton. |
| http | A Flutter package that contains a set of high-level functions and classes that make it easy to consume HTTP resources. It's multi-platform, and supports mobile, desktop, and browser. |
| image\_gallery\_saver | A Flutter plugin project for saving images to the gallery, iOS needs to add the following keys to your Info.plist file. |
| image\_picker | A Flutter plugin for iOS and Android for picking images from the image library and taking new pictures with the camera. |
| permission\_handler | A Flutter plugin that provides support for a cross-platform (iOS, Android) API to request permissions and check their status. |
| json\_annotation | A Flutter plugin for defining the annotations used by json\_serializable to create code for JSON serialization and deserialization. |
| path\_provider | A Flutter plugin for finding commonly used locations on the filesystem. Supports Android, iOS, Linux, macOS, and Windows. Not all methods are supported on all platforms. |
| googleapis | A Flutter library that provides support for accessing Google APIs described through the API discovery service. |
| googleapis\_auth | A Flutter package that provides support for obtaining OAuth2 credentials to access Google APIs. |
| enough\_mail | A Flutter library that provides support for IMAP, POP3 and SMTP for email developers. |

# Software Installation

The user will be guided by the instructions while they set up the development environment on Windows and macOS. This section will outline the core procedures to assist the user in selecting the appropriate tools. Please visit the official websites for each of the systems mentioned below for detailed step-by-step instructions.

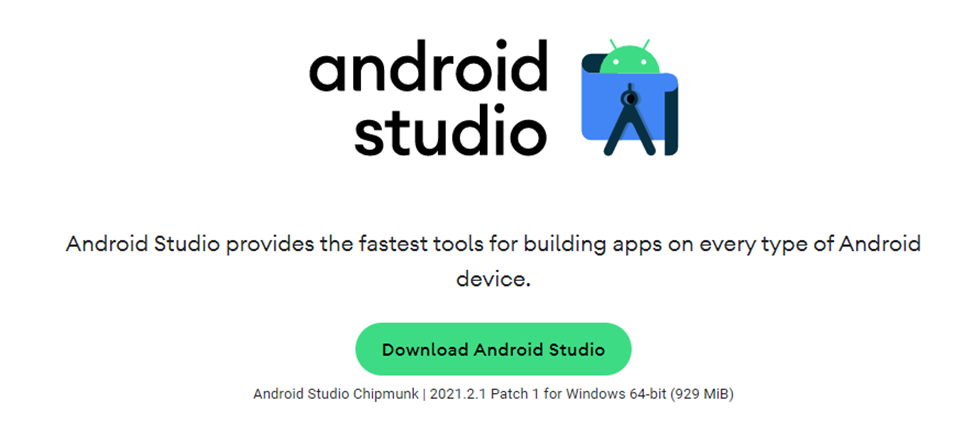
* 1. Android Studio Installation

**Download and Installation:**

* **Step 1:** Download the most recent version of Android Studio for your operating system from<https://developer.android.com/studio>.

**Figure 1**

*Android Studio Installation*



* **Step 2:** Follow the official steps for your Operating System at<https://developer.android.com/studio/install> to install Android Studio on your computer.
* **Step 3:** Following a successful installation of Android Studio, you may proceed to install the additional tools needed to launch Android Studio, including Flutter, Dart, and the Android Emulator.
  1. Flutter and Dart System Requirements

The following list names the system requirements needed for the development environment for Windows and MacOS.

Windows:

* Operating system: Windows 10 or later (64-bit), x86-64 based.
* Disk space: 1.64 GB.
* Tools: Windows PowerShell 5.0 or newer OR Git for Windows 2.x OR command Prompt.

MacOS:

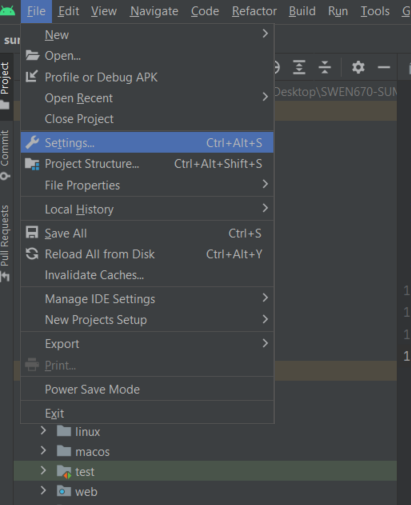
* Operating system: macOS.
* Disk space: 2.8 GB.
* Tools: git is used for flutter installation and upgrade. It is recommended to install Xcode that has git.
  1. Dart and Flutter Plug-ins

This section details instructions for installing Flutter and Dart Plug-ins on Android Studios.

Step 1: With a blank project open, navigate to File 🡪 Settings

**Figure 2**

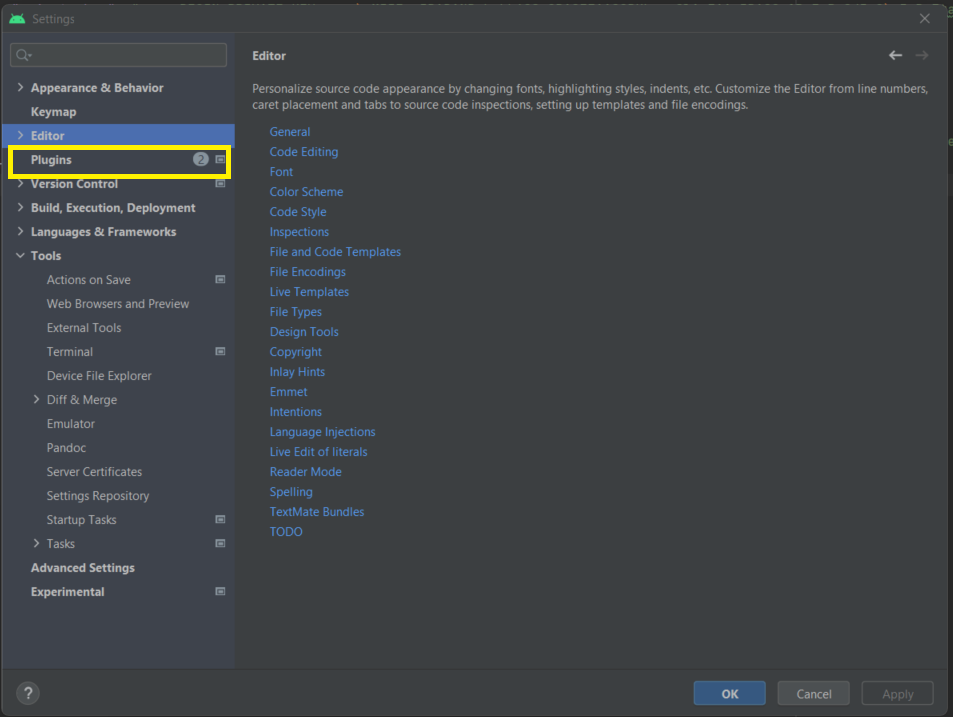
*File Tab*



Step 2: On the Settings screen navigate to the Plug-ins tab.

**Figure 3**

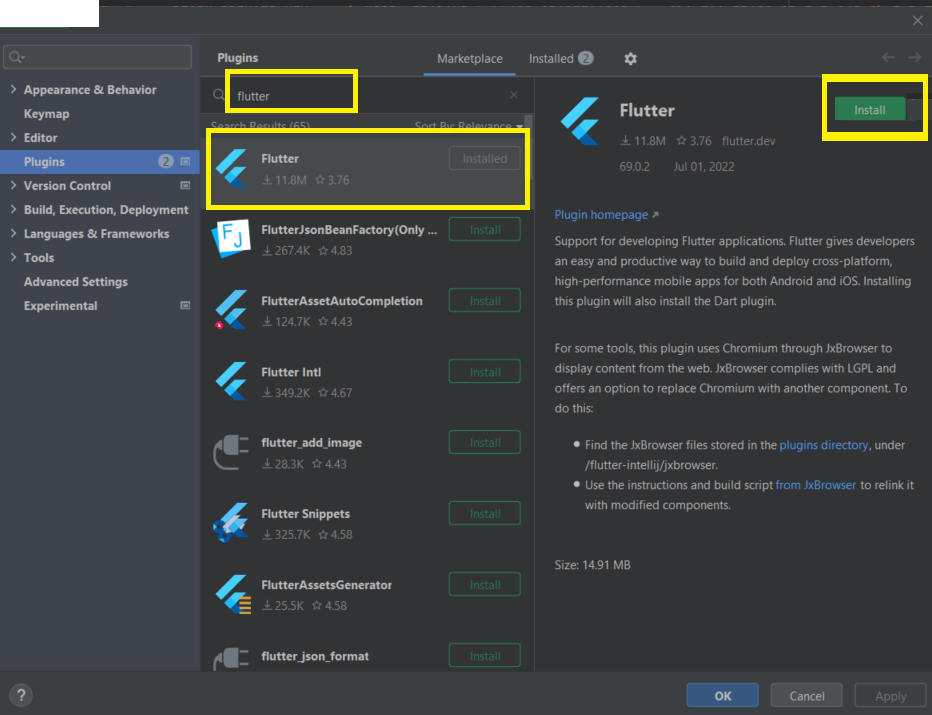
*Settings Plug-ins tab*



Step 3: On the Plug-ins screen, in the search bar type in flutter and choose the first one. Then click the install green button on the right to install the plug-in.

**Figure 4**

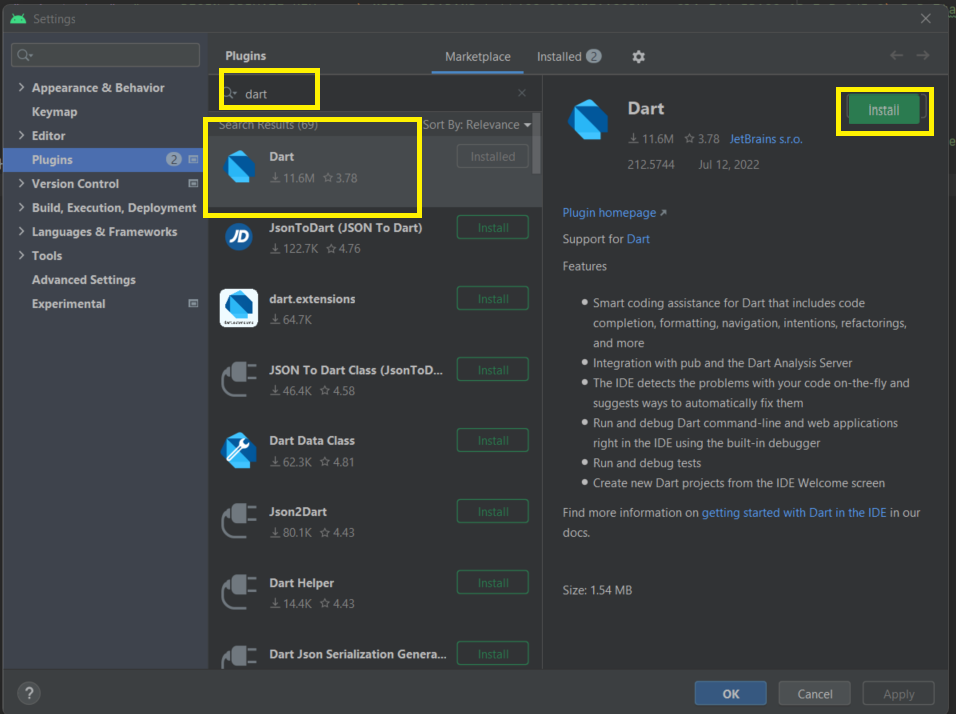
*Install Flutter Plug-in*



Step 4: After the Flutter plugin is installed, if you are not prompted to install Dart plugin, search “dart” in the search bar and install the first one.

**Figure 5**

*Install Dart Plug-in*



After both have been installed, click “OK” on the bottom right. Close and reopen Android Studios.

* 1. Android Emulator

Setting up Android Emulator

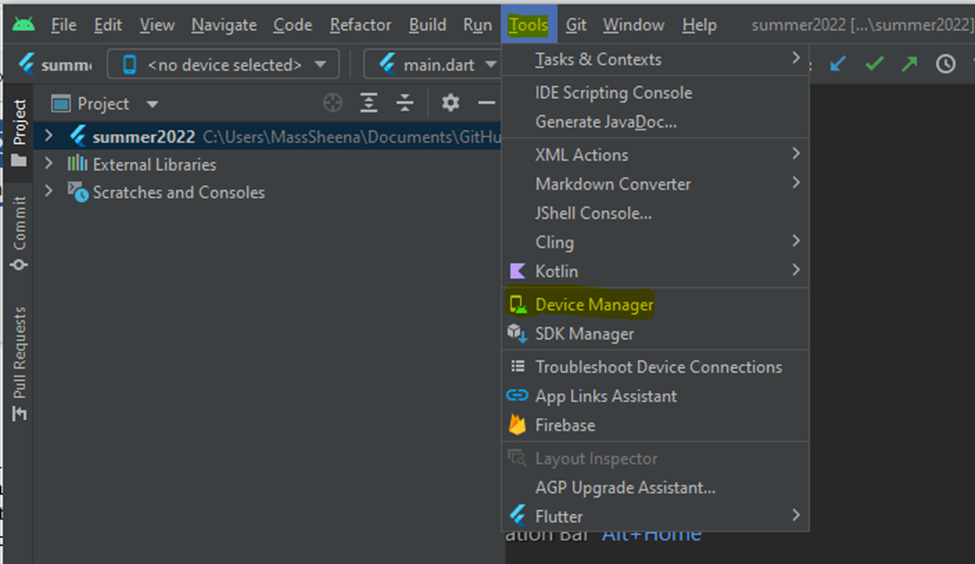
**To launch the Android Emulator, follow the below steps**

**Step 1**: On your computer, enable [**VM acceleration**](https://developer.android.com/studio/run/emulator-acceleration)**.**

**Step 2:** On the Android Studio main screen, select “**Tools**”, then “**Device Manager**” in the dropdown.

**Figure 6**

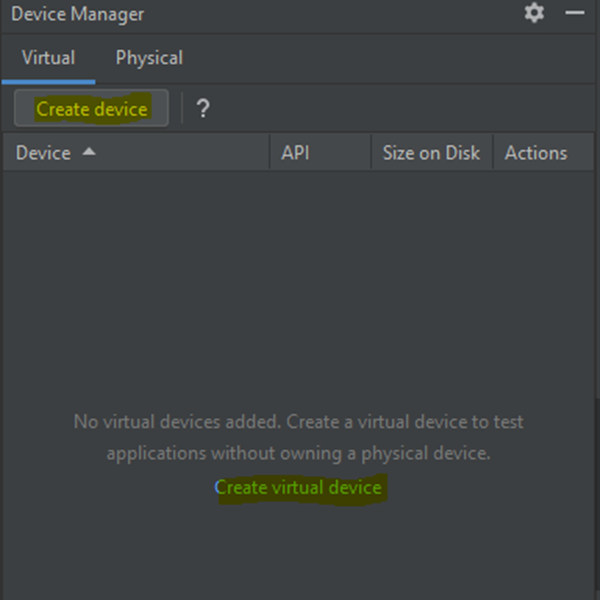
*Tools Tab*

****

**Step 3:** Select **“Create Device.”**

**Figure 7**

*Device Manager – Create a Device*

****

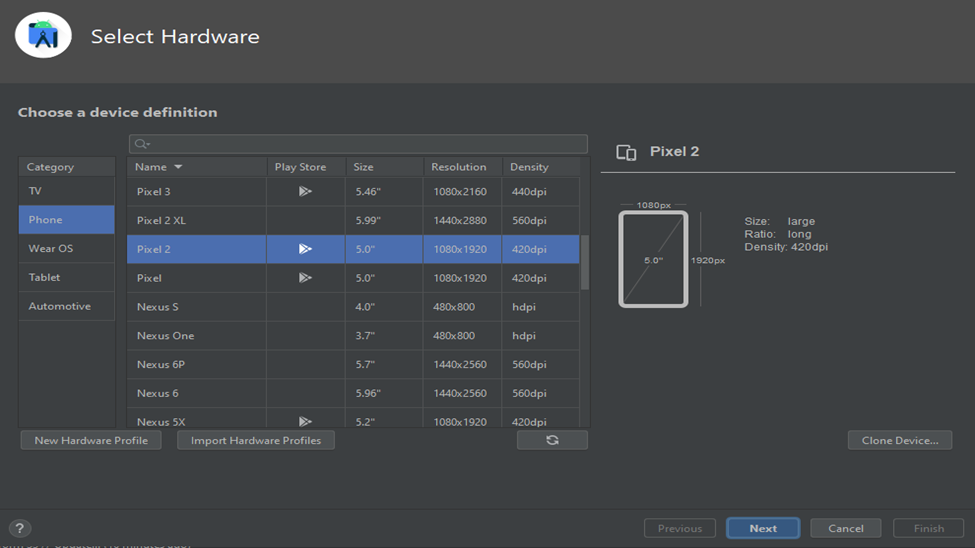
**Please note:**

* In older versions of Android Studio, select Android Studio >Tools> Android>AVD Manager and select Create Virtual Device (only while in an Android project is the Android submenu visible).
* Choose Configure>AVD Manager and select Create Virtual Device if you do not have a project open.

**Step 4:** Select Hardware and choose “**Next**.”

**Figure 8**

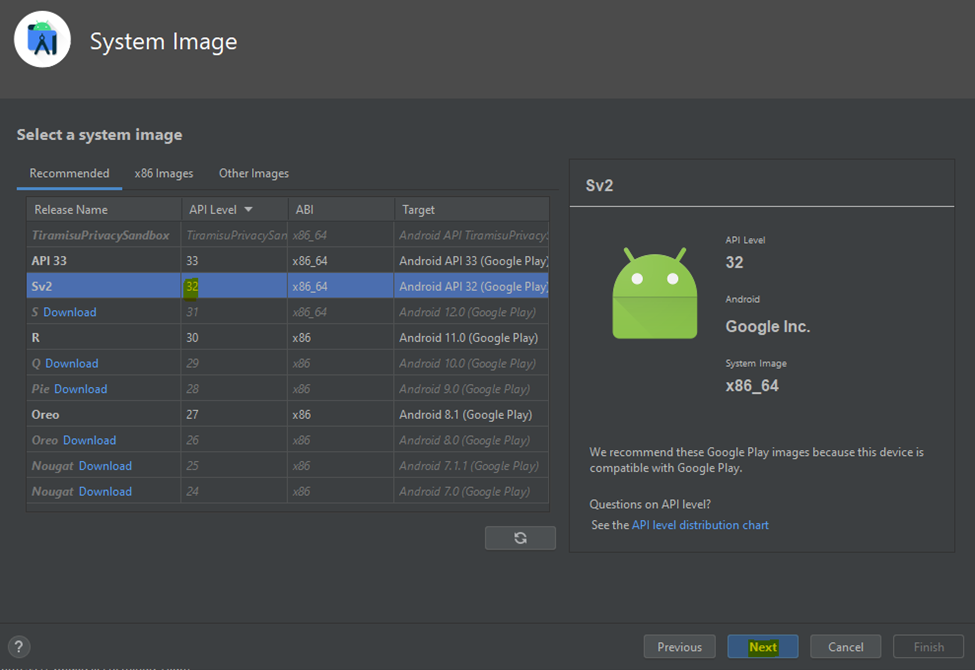
*Choose Device Settings and Type*

****

**Step 5:** Choose a system image for the Android version you would like to emulate and select “Next.” **Recommendation: X86\_64 image.**

**Figure 9**

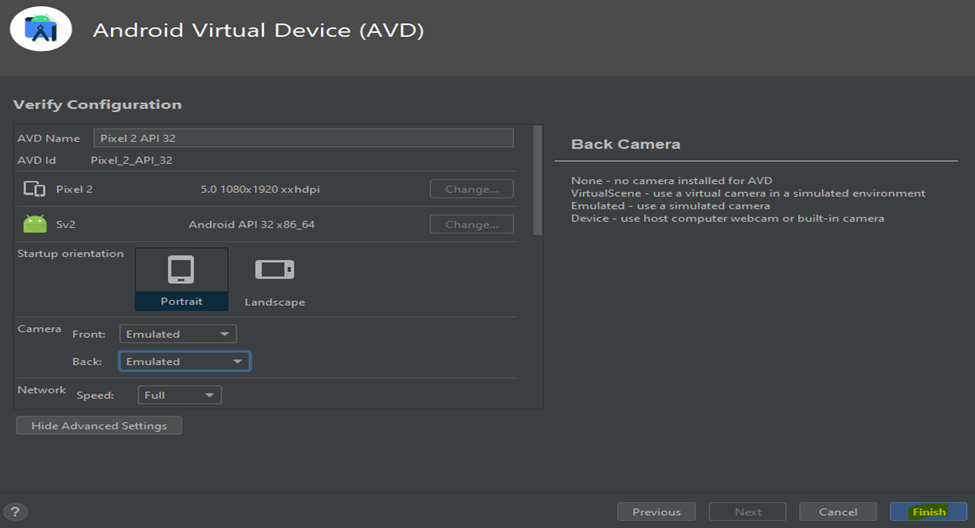
*Choose Android API Level*

****

**Step 6**: Select “**Finish**” after making sure the Android Virtual Device Configuration is accurate.

**Figure 10**

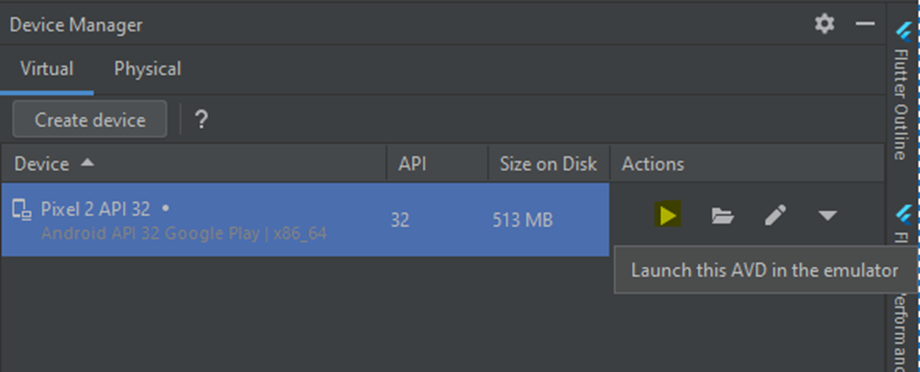
*Finish Creating Device Emulator*

****

**Step 7:** In Android Virtual Device Manager, Select the **“Launch this AVD in the emulator”** option.

**Figure 11**

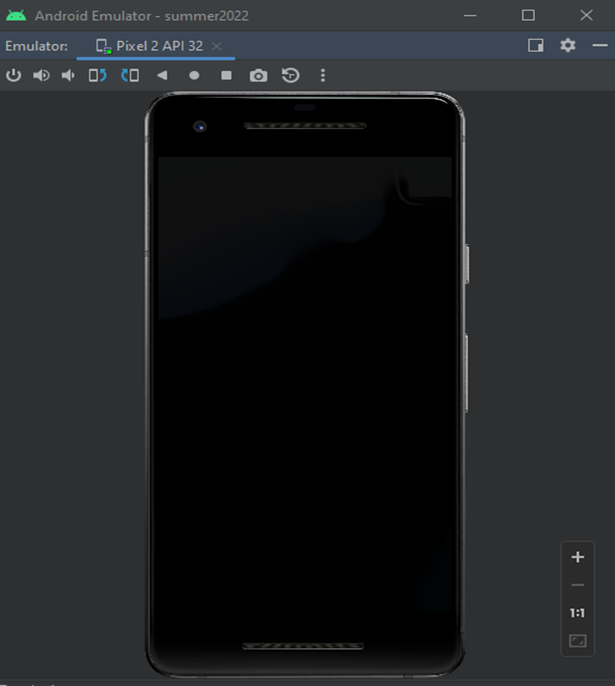
*Launch the Emulator*

****

**Step 8:** The emulator starts up and shows the standard canvas for the device and the Operating System version you have chosen.

**Figure 12**

*New Emulator*

****

* 1. Visual Studio Code – Mac Install

This section covers the installation of the Visual Studio Code on Mac.

1. Go to the Visual Studio Code homepage, [https://code.visualstudio.com](https://code.visualstudio.com/Download), and click on the "Download Mac Universal" button to download the latest version. By default, the download file will be saved in the "Downloads" folder in the Finder app on the Mac.

**Figure 13**

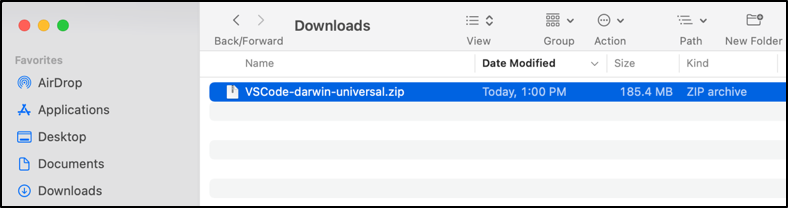
*Visual Studio Installation*



1. Locate the downloaded Visual Studio Code zip file on the Mac and double-click to extract the archive content.

**Figure 14**

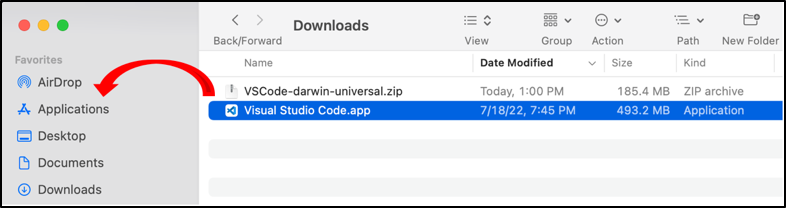
*Download Zip Location*



1. Drag and drop the "Visual Studio Code.app" to the "Applications" folder to make it available in the Mac Launchpad.

**Figure 15**

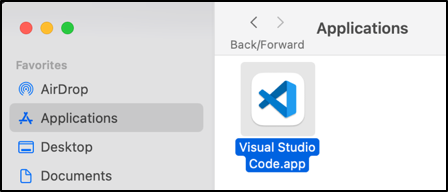
*Drag and drop the Download*



1. The Visual Studio Code is now installed and set up on the Mac. To open the Visual Studio Code from the "Applications" folder, double-click on the "Visual Studio Code.app" icon.

**Figure 16**

*Visual Studio App*



* 1. Dart and Flutter Plug-ins

1. On VS Code, click on the “extensions” icon on the left

**Figure 17**

*Visual Studio Code Side Bar*



2. VS code will open a screen with an extension marketspace window.

3. On search, look for dart and flutter. They are both from Dart Code.

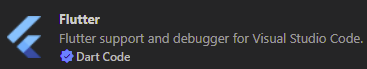
**Figure 18**

*Dart Plug-in*



**Figure 19**

*Flutter Plug-in*



4. Click the install button on the details window for **each**

**Figure 20**

*Install*



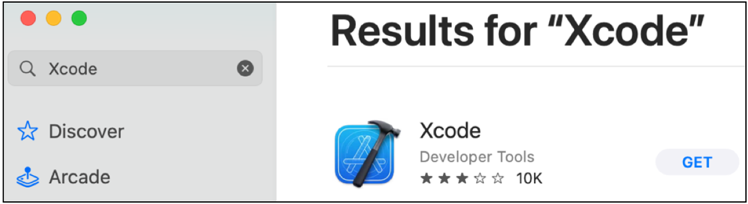
3.7 Xcode (iOS)

To develop a Flutter application in an iOS platform, the Xcode integrated development environment must be installed on the user’s Mac.

1. Install the latest version of Xcode. The easiest way is to download Xcode from the App Store. From the App Store, search "Xcode" and click the "Get" button to install Xcode to the user’s Mac.

**Figure 21**

*XCode Install*



1. Configure the Xcode command-line tools to use the newly installed version of Xcode by running the following from the Terminal or command line interface.

$ sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer

$ sudo xcodebuild -runFirstLaunch

This is the correct path for most cases when the user wants to use the latest version of Xcode. If the user needs to use a different version, specify that path instead.

1. Run the following in a Terminal window or command line interface to sign the Xcode license agreement.

$ sudo xcodebuild -license

* 1. Flutter and Dart SDKs

Install Flutter

1. Download the following installation bundle to get the latest stable release of the [Flutter SDK](https://docs.flutter.dev/development/tools/sdk/releases)
2. Extract the zip file and place the contained flutter in the desired installation location for the Flutter SDK.
   1. Avoid file locations that will have special characters and/or spaces.
3. To run flutter commands with the windows console you will need to update your computer’s path value.
   1. From the Start search bar, enter ‘env’ and select Edit environment variables for your account.

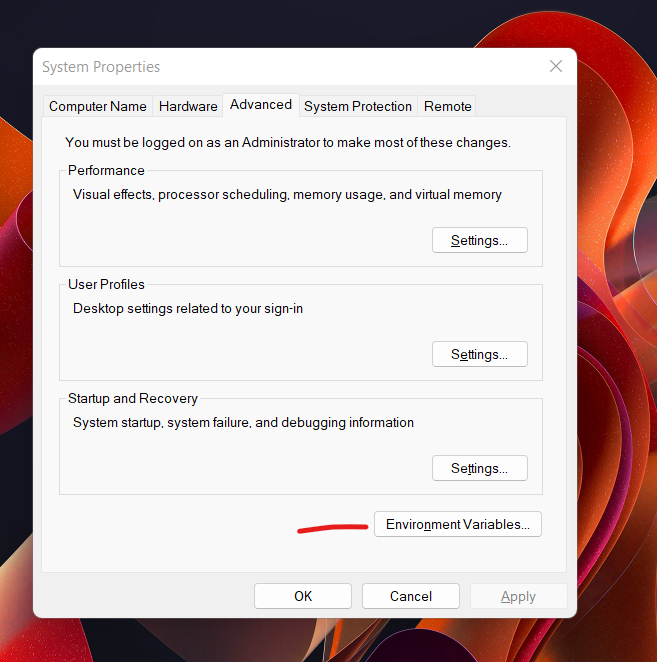
**Figure 22**

*Search for Environment Settings*



**Figure 23**

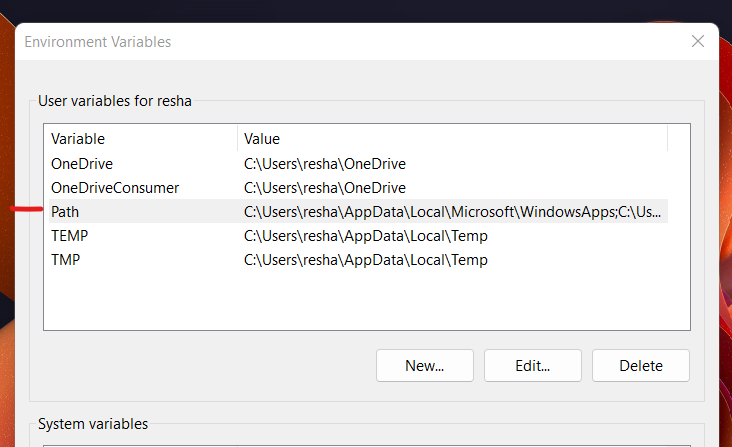
*System Properties – Environment Variables*



* 1. Under User, variables check if there is an entry called Path.
     1. If the entry exists, append the full path to flutter\bin using, as a separator from existing values.
     2. If the entry doesn’t exist, create a new user variable named Path with the full path to flutter\bin as its value.

**Figure 24**

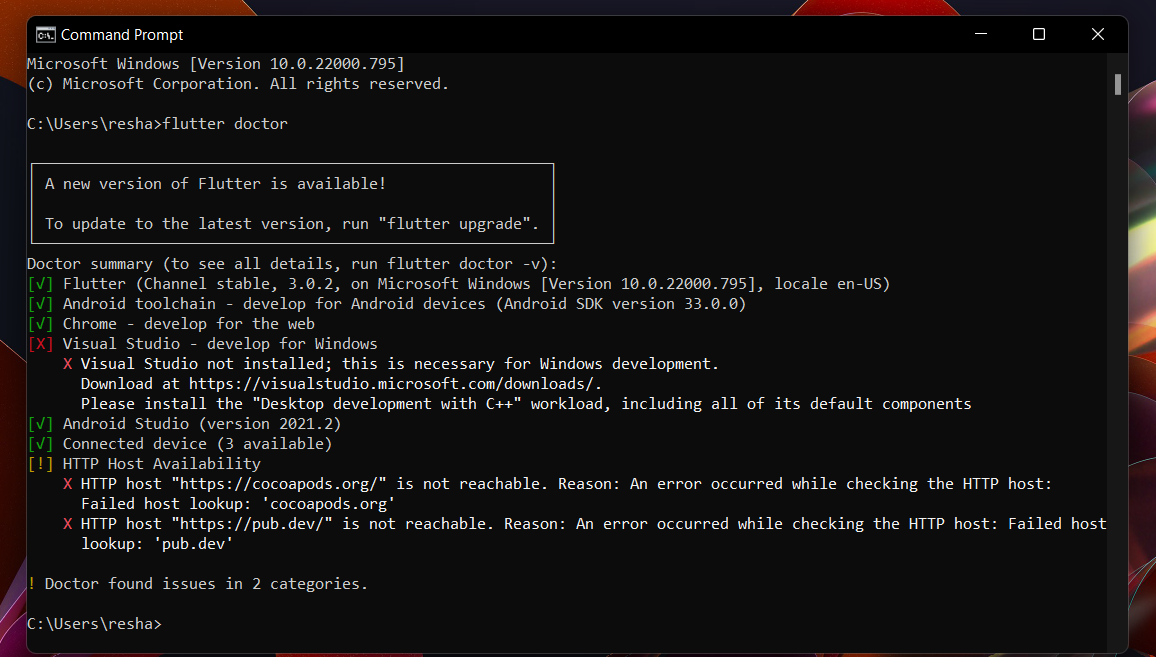
*Environment Variables*



1. To verify that flutter is completely installed run “flutter doctor” from the command line.

**Figure 24**

*Run flutter doctor in Command Prompt*



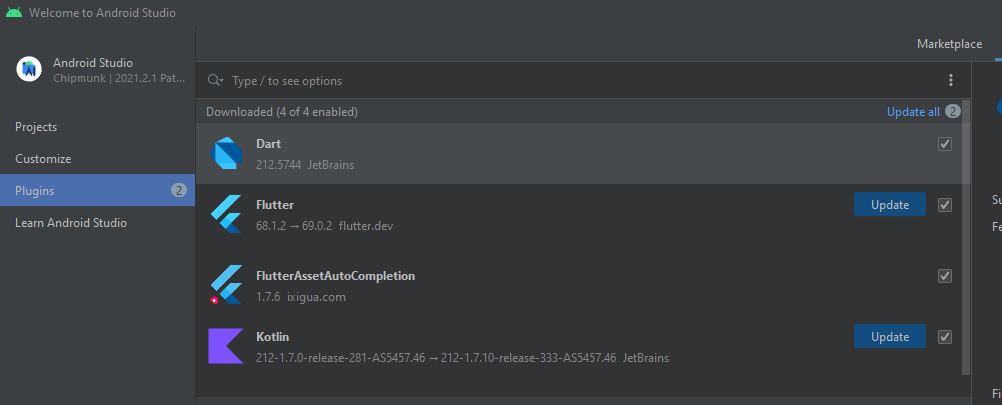
**Install Dart**

1. To install Dart you will need to download the Dart SDK from [here](https://dart.dev/get-dart/archive).
2. Once downloaded extract the SDK to a folder similar to the Flutter SDK.
3. Once extracted, you will need to add the Dart SDK to your computer’s route value like with Flutter.

Once both are installed you will need to make sure that Flutter and Dart plugins are enabled on your Android studio.

**Figure 25**

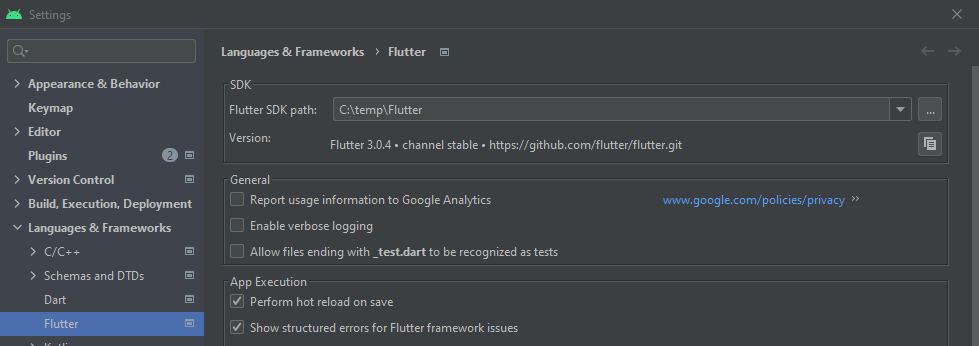
*Dart Plug-in Installation*



Once those are enabled you will need to path your Android Studio to the Dart and Flutter SDKs under settings.

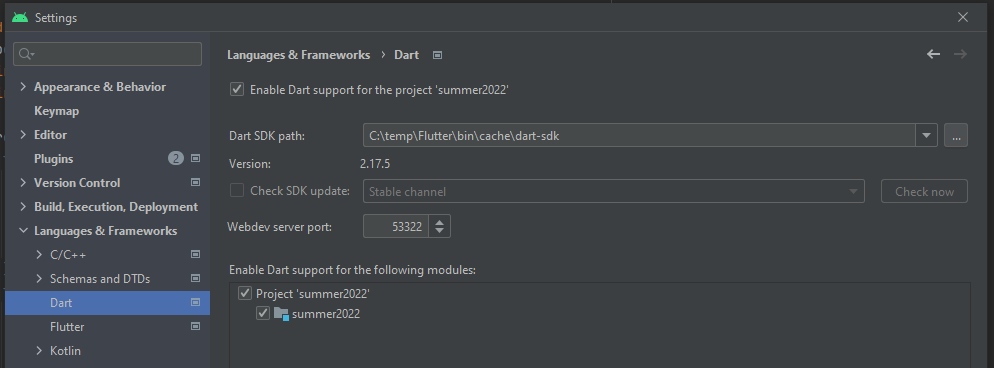
**Figure 26**

*Path to Flutter SDK*



**Figure 27**

*Path to Dart SDK*



* 1. GitHub Desktop setup

1. Install Git from [here](https://git-scm.com/downloads).
2. Download and install GitHub Desktop from [here](https://desktop.github.com/).
3. Sign in or create an account that is associated with your GitHub repo.
4. Clone down your repo to a file path where you want to store the application.

# Prepare the Mobile Application for Use

* 1. Google Cloud Vision API setup

1. Go to console.cloud.google.com.
2. On the top left, click the down array right next to the Google Cloud Platform.

**Figure 28**

*Google Cloud Platform*



1. Click on new project on the top right of the new window. Add a project name and click create
   1. For Project name, use Vision

**Figure 29**

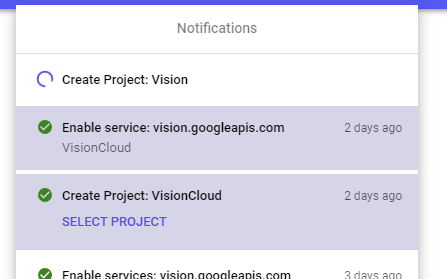
*Creating a new project*



1. It will take a minute for it to be created. You can see the progress under notifications. Once complete, select the project. You can also select it via figure 31.

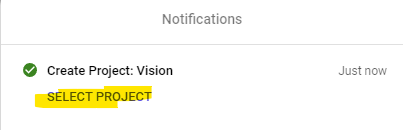
**Figure 30**

*Project Selection 1*



**Figure 31**

*Project Selection 2*



1. Next go to Navigation Menu on the top right -> APIs & Services -> Enabled API & Services.
2. On the top left, you should see Enable APIs and Service. Click on it.

**Figure 32**

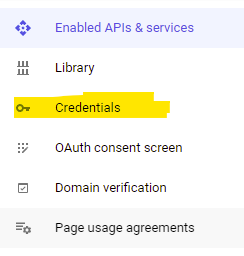
*Enable APIs and Services*



1. Use the search to look for “Cloud Vision API”, click on it and hit enable. It will bring you to the API/Service Detail Window.
2. Once done, you will need credentials to use with your project. In the service detail window, there will be a button for credentials on the left navigation window. (You can also go to the Navigation Menu button on the top left -> APIs and Services -> Credentials).

**Figure 33**

*Credentials*



1. Under the credentials window, click on Create Credentials -> Service Account.

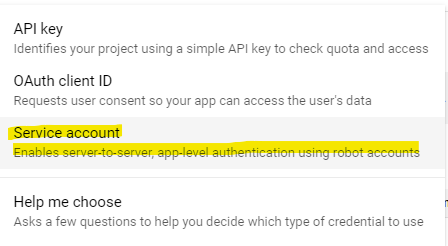
**Figure 34**

*Create Credentials*



**Figure 35**

*Select Service account*



1. For Step one, insert a Service Account name and click “Create and Continue”.
   1. For Service Account name, insert “Vision”.

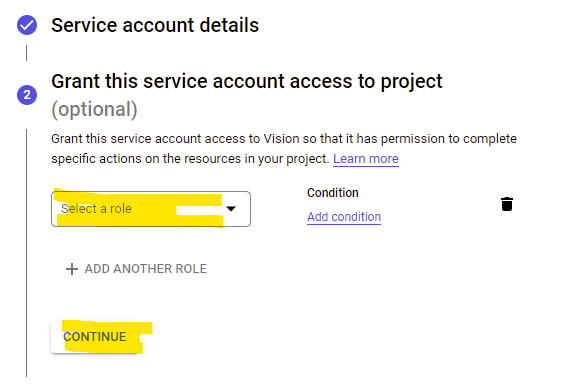
**Figure 36**

*Enter Service Account Details*  


1. Step 2 is to create a role. For simplicity, select “Owner”. This will be revisited later once everything is working and we need to lock things down. Now click next.

**Figure 37**

*Grant Permissions*



**Figure 38**

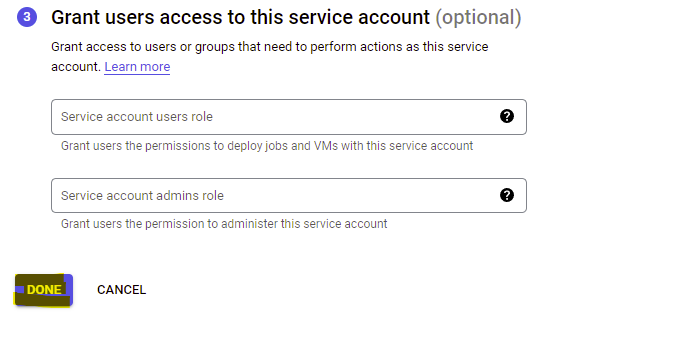
*Choose Owner for Role Permissions*



1. Step 3 will be skipped. Click on “Done

**Figure 39**

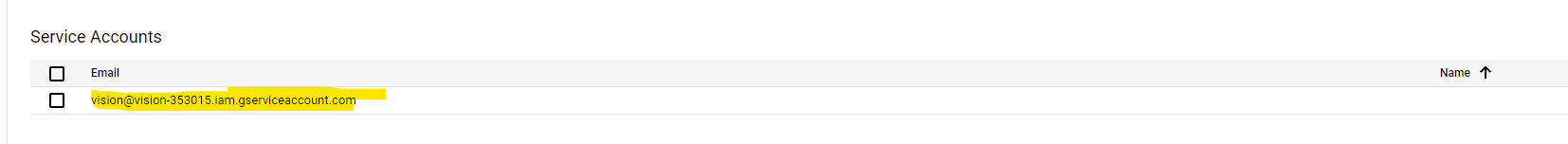
*Done*



1. Now that your role has been established, under Service Accounts, click the email address.

**Figure 40**

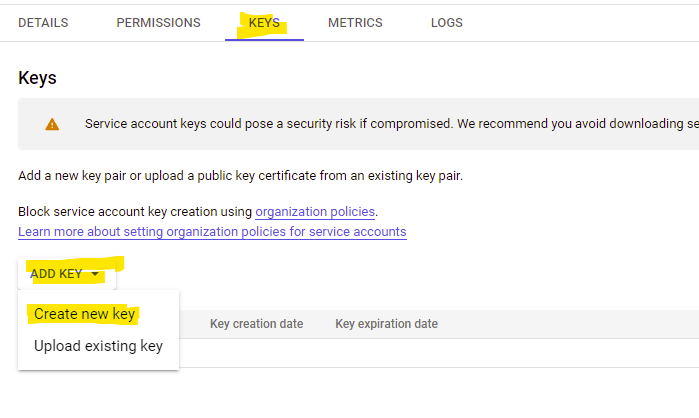
*Service Accounts created*



Click on Keys at the top, then the “Add Key” Button -> Create New Key.

**Figure 41**

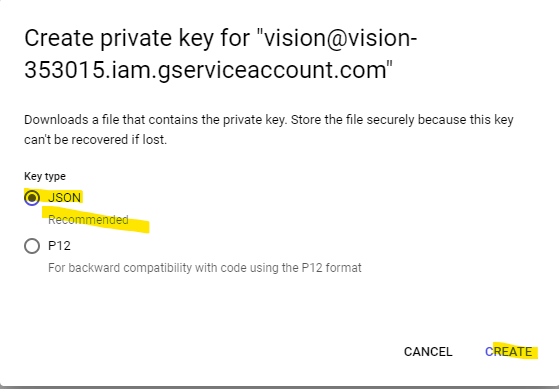
*Create New Key*



1. Click on json and then create. IT will automatically download the json file to your computer. Make sure to change the name of the file to “credentials.json”.

**Figure 42**

*Create private key credentials json file*



15. Enable Billing - Go to menu and billing.

**Figure 43**

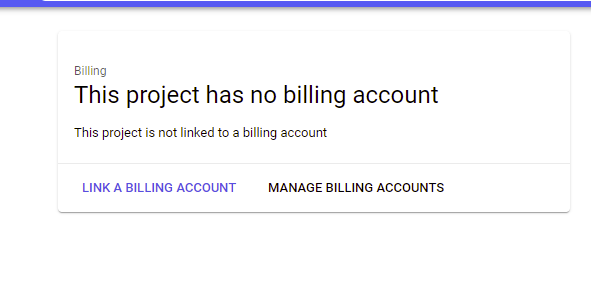
*Enabling Billing*



1. It will open the billing page. Click on Link a Billing Account.

**Figure 44**

*Link a Billing Account*



1. Follow the prompt until it brings you back to the original screen.
2. You are now good to go to use Google Cloud Vision API.
   1. Google ML Kit Barcode Scanning Plugin setup

This section will detail the process of setting up the Google ML Kit Barcode Scanning Plugin. While the entire ML Kit package (google\_ml\_kit) can be downloaded as an umbrella plugin for Flutter application development, each API within the package can be installed as a separate plugin for use. The Barcode Scanning API plugin name is google\_ml\_kit\_barcode\_scanning.

**Figure 45**

*Google ML Kit Plugin on pub.dev*

Graphical user interface, text, application, email

Description automatically generated

To install the plugin, the developer must have the Flutter SDK installed on their system and a Flutter application to run. Please refer to section 3.4 of the document for a detailed installation of the Flutter SDK.

To install Flutter SDK: <https://docs.flutter.dev/get-started/install>

To create a new Flutter application: <https://docs.flutter.dev/get-started/test-drive?tab=androidstudio>

The minimum Android SDK version required to use ML Kit Barcode Scanning is Version 21. Ensure that you have this SDK version or higher. Also, you may need to update the *build.gradle* file under android/app folder in the defaultConfig section with the minSdkVersion to ensure that the plugin runs successfully.

**Figure 46**

*MinSDKVersion*

Text

Description automatically generated

To install the google\_ml\_kit\_barcode\_scanning plugin:

1. In the command line interface (CLI) of your choice, navigate to the Flutter project directory if you are not in it already
2. Run the command:

flutter pub add google\_ml\_kit\_barcode\_scanning

which will add a dependency for the plugin in your project’s *pubspec.yaml* file.

**Figure 47**

*Dependency in pubspec.yaml*

Text

Description automatically generated

To use the plugin in your Dart code:

1. Once the dependency is added to the *pubspec.yaml* file, the plugin can be utilized in your Dart code for the application. The following import should be added to the Dart file before any classes are declared:

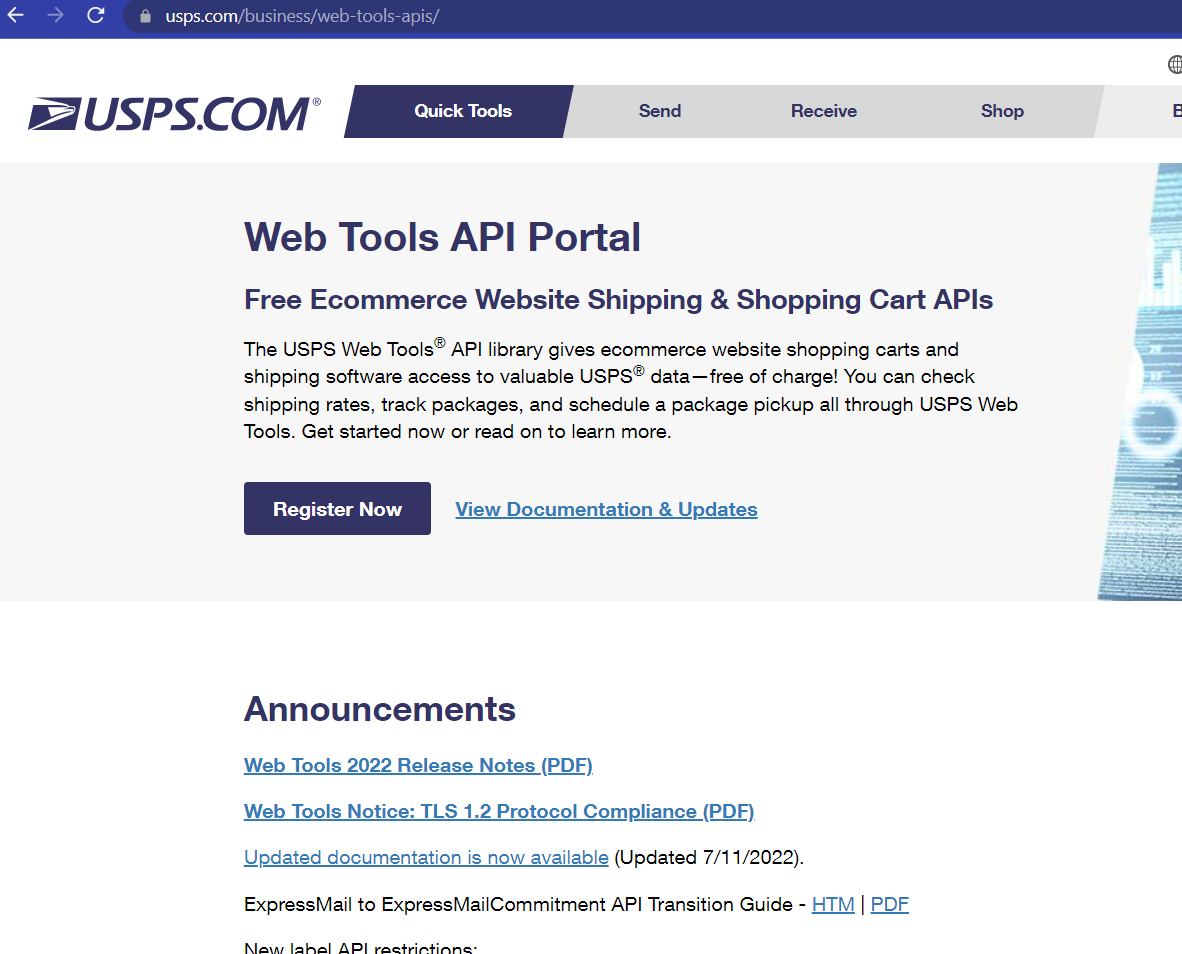
import 'package:google\_mlkit\_barcode\_scanning/google\_mlkit\_barcode\_scanning.dart';

* 1. USPS Address Validation API setup

This section will detail the process of registering an account with the USPS Web Tools API Portal. The following image illustrates the website https://www.usps.com/business/web-tools-apis/ landing page.

**Figure 48**

*USPS Web Tools API Page*

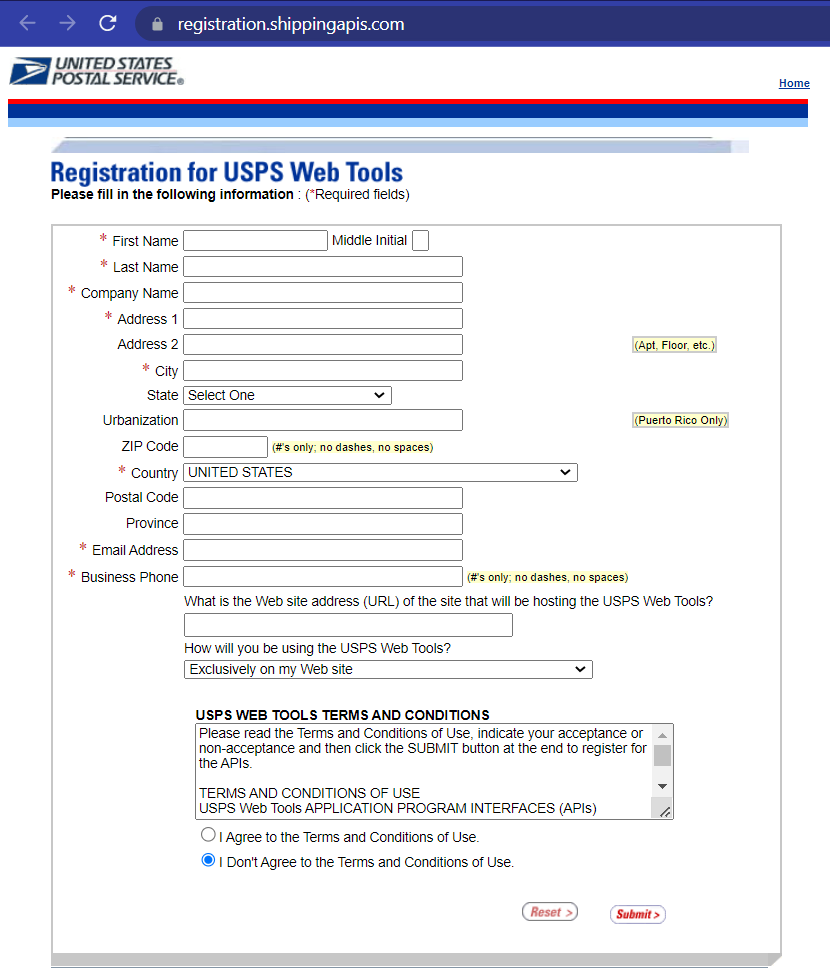


USPS requires an account to be registered with their portal to have access to any of their APIs.

Click the “Register Now” button and it will redirect to a registration form. The following image illustrates the form needed to register for a valid account.

**Figure 49**

*Registration Form for USPS Web Tools*

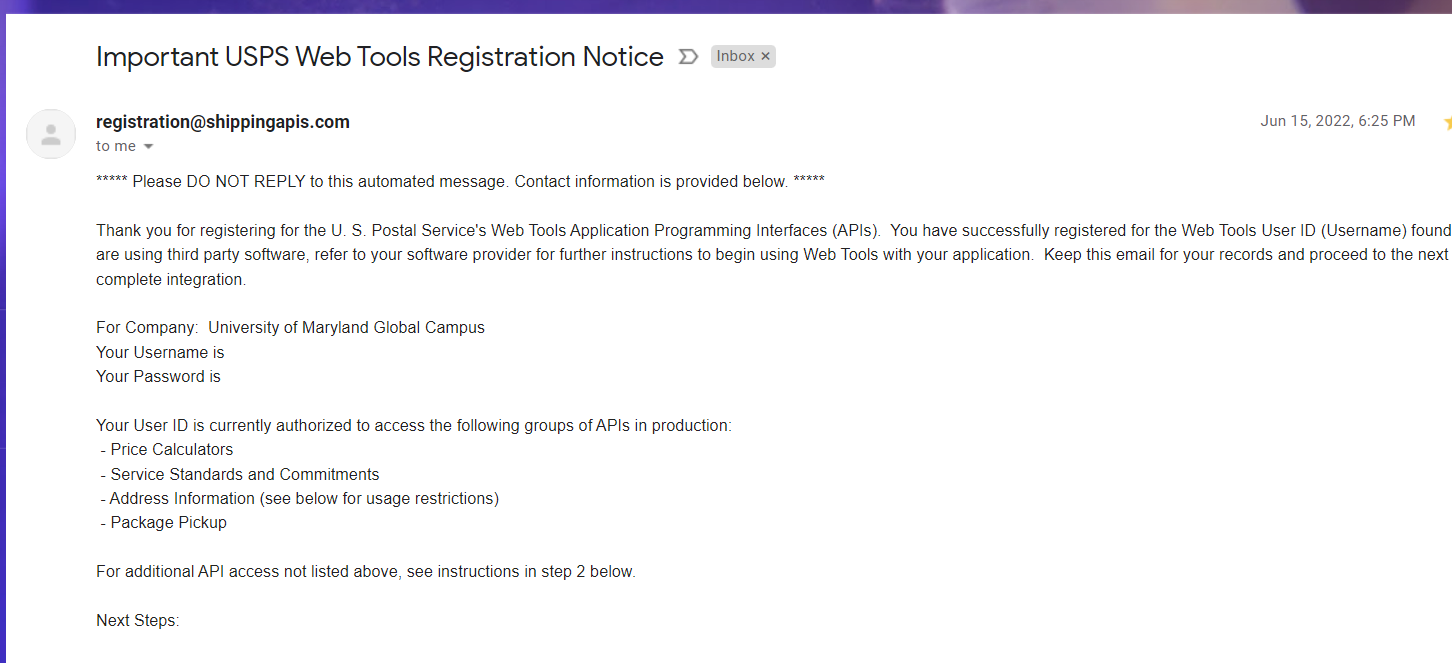


After filling out the form with the necessary information and submitting it, email notification of the registration will be sent to the email used to create the account.

This email will include information about the username and password needed to connect to their APIs and documentation for developers on the implementation of the APIs. The following images illustrate an example of the email notice.

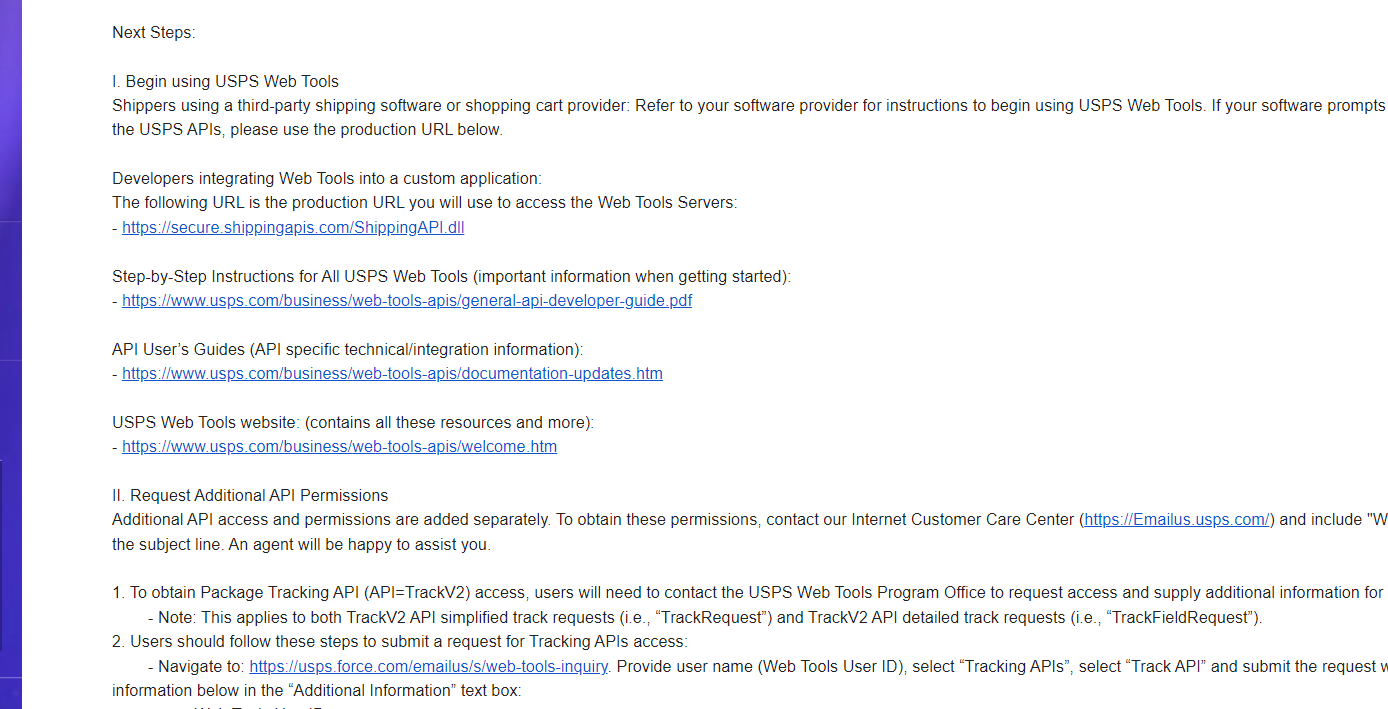
**Figure 50**

*Email Notice of Registration being complete*



**Figure 51**

*Email Notice of Registration*



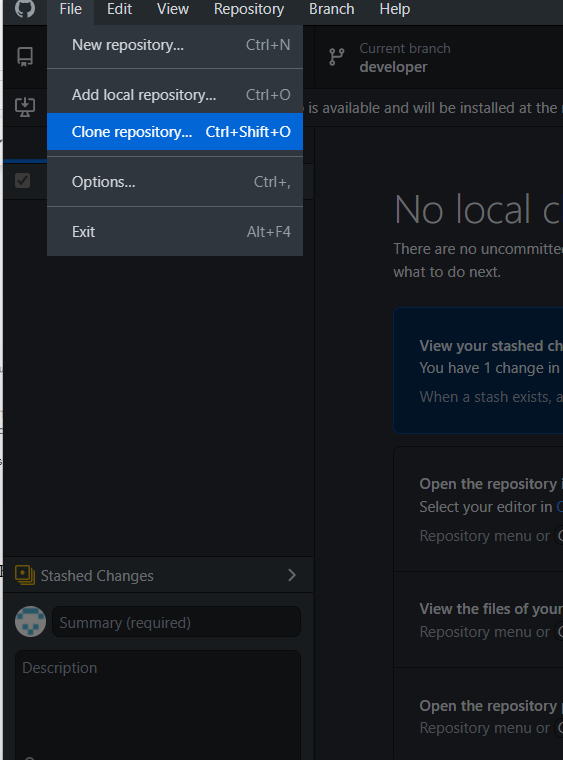
* 1. Cloning GitHub repository

This section gives step instructions on how to clone the repository from GitHub using GitHub Desktop.

Step 1: Open the GitHub Desktop app. Click the file and then choose “Clone repository”.

**Figure 52**

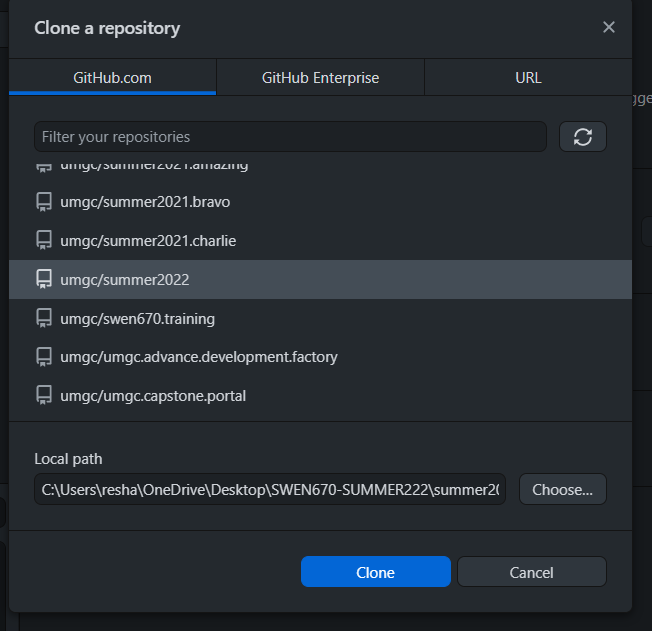
*Clone a Repository from GitHub Desktop*



Step 2: The following image shows the clone repository screen. The screen shows the repository that is available to be cloned. Click on the umgc-summer22 repository to be cloned, choose the file location to clone it to, and click clone.

**Figure 53**

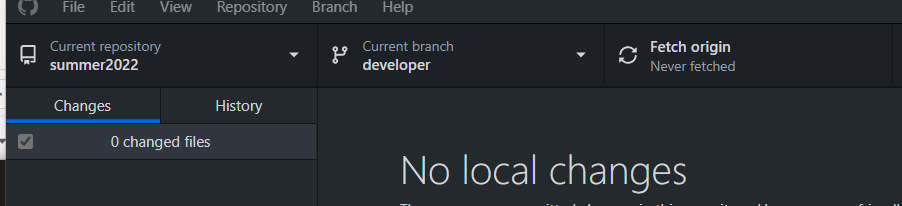
*Choose the Repo to Clone*



Step 3: After the cloning process is done, open Android Studio if that is your IDE of choice, and it will automatically sync with the developer branch. The developer branch is the default branch. The following images illustrate the default branch on the GitHub Desktop app and on Android Studios.

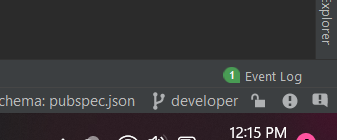
**Figure 54**

*Cloned Repository*



**Figure 56**

*Developer Branch*



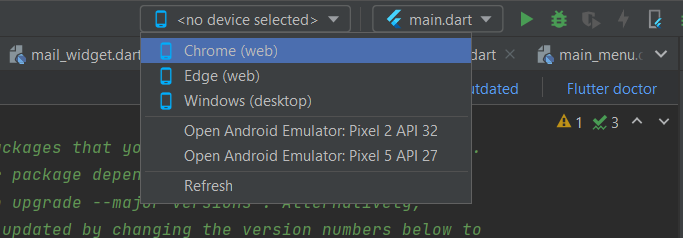
* 1. Run the Flutter application

To run the Flutter application: on Android Studios IDE

Step 1: Click on no device dropdown and choose open android emulator. An android phone emulator should load.

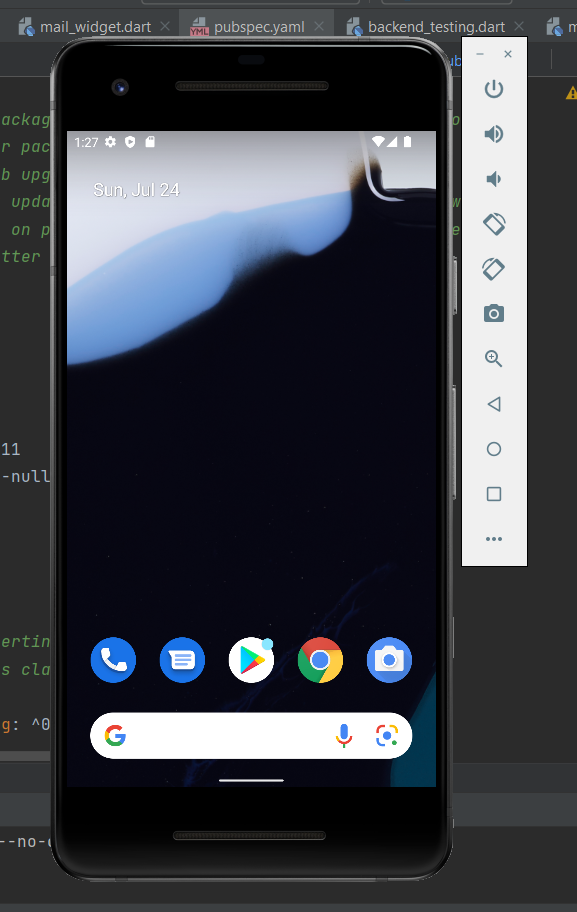
**Figure 57**

*Connection to an Android Emulator Device*



**Figure 58**

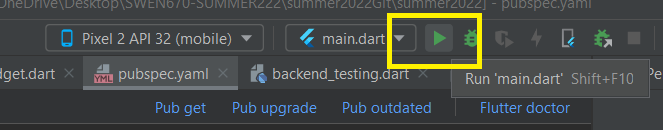
*Android Emulated Device*



Step 2: click “Run” the green arrow.

**Figure 59**

*Run the Application*



After step 3, the Futter application will begin building and the app should display on the android phone emulator. As depicted in the following image.

**Figure 60**

*USPS Informed Delivery Visual Assistance App*

Graphical user interface, text, application

Description automatically generated

# Testing the Mobile Application

* 1. Testing Objective
     1. Unit tests

5.1.1.1 Google Cloud Vision

**Table 4**

*Google Cloud Vision Tests*

| **Test** | **iPhone** | **Android** |
| --- | --- | --- |
| **The Application can access the credentials.json file in the assets folder.**  Success case: The application does not return an error. |  |  |
| **The application receives text results for processed mailpiece**  Success case: The application is able to access a stored mailpiece, send it to vision to be process and return the result as a AddressObject. The result using mail.test.01.jpg under the assets folder should match the expected object results. |  |  |
| **The Application sends mailpiece to detect logos.**  Success case: The application sends image to be processed by google vision using logo detector option and receive its outcome. The result using mail.test.01.jpg under the assets folder should match the expected object results. |  |  |
| **The Application uses the vision search function.**  Success case: The application uses the search function and returns the outcome as a MailResponse object. The object contains both addresses and logos found. The result using mail.test.01.jpg under the assets folder should match the expected object results. |  |  |
| **The Application uses the search function illegible image.**  Success: The List Objects within MailResponse (Address and logo) will both have a length of zero. |  |  |

5.1.1.2 Digest Email Parsing

Due to the nature of IMAP, it is not possible to unit test this without running the application.

**Table 4**

*Digest Email Parsing tests*

| **Functional Test** | **iPhone** | **Android** |
| --- | --- | --- |
| **The application must connect to the user’s email.**  Success case: The application is able to create an IMAP client with the user’s credentials. |  |  |
| **The application must retrieve the user’s USPS Daily Digest.**  Success case: The application is able to find and fetch the desired email using the IMAP client. |  |  |
| **The application must parse the information of the Daily Digest Email.**  Success case: The application is able to parse the information into an object to be consumed by the UI. |  |  |

5.1.1.3 USPS Address Verification

**Table 6**

*USPS Address Verification Tests*

| **Unit Test** | **iPhone** | **Android** |
| --- | --- | --- |
| **The application must accept an address string.**  Success case: The application is able to take a formatted address string. |  |  |
| **The application must create an XML for the USPS Web API.**  Success case: The application is able to create an XML for a given address. |  |  |
| **The application must connect to the USPS Web API.**  Success case: The application is able to verify the status of the USPS Web API. |  |  |
| **The application must determine if an address has been verified.**  Success case: The application determines if the address is verified based on the response from the USPS Web API. |  |  |

5.1.1.4 QR Code/Barcode Scanning

**Table 7**

*QR code/Barcode Scanning Tests*

|  |  |  |
| --- | --- | --- |
| **Unit Test** | **iPhone** | **Android** |
| **Scanner must recognize QR Code**  Success case: There is one parsed QR code in the list |  |  |
| **Scanner must not recognize QR Code**  Success case: There are no parsed barcodes in the list |  |  |
| **Scanner must recognize barcode**  Success case: There is one parsed barcode in the list |  |  |

* + 1. Integration tests

**Table 8**

*Integration Test*

|  |  |  |
| --- | --- | --- |
| **Functional Test** | **iPhone** | **Android** |
| **The application will open the Digest Page with the digest of the specified date.** |  |  |

# Troubleshooting

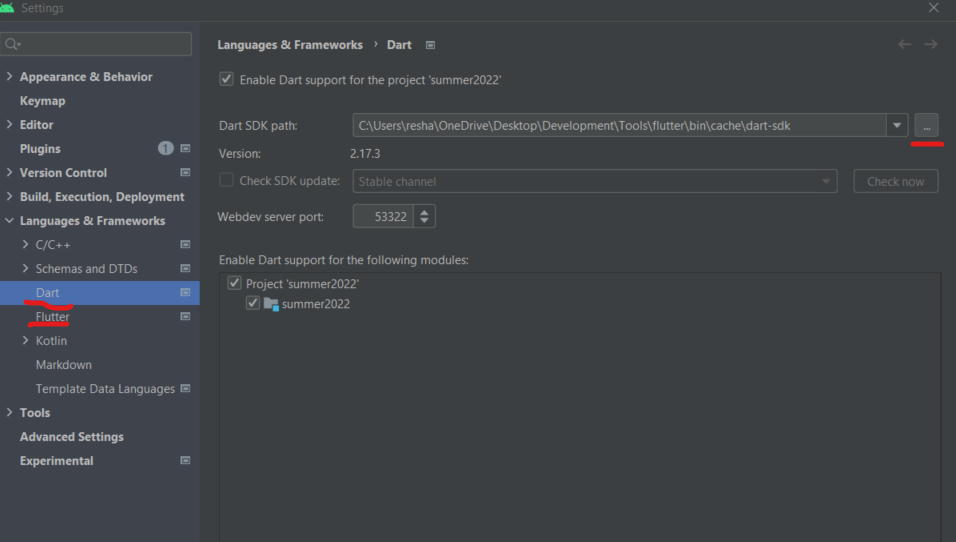
* 1. Issues configuring Flutter or Dart SDKs

If the IDE being used says it cannot detect the dart or flutter SDKs, it might need to be remapped to the location they were downloaded to. Usually, you can know if this issue occurred if the project won’t build and the main.dart file is not recognized. The following steps shows how to remap the SDKs.

Click “File” then “Settings” to remap the SDKS. Then rerun flutter clean in the terminal. The following image shows the settings dialog where you can select the path to the Dart or Flutter SDK.

**Figure 61**

*Settings Dart SDK Path*



* 1. Emulator not responding

In the case of an Android emulator not responding, it is encouraged to close the running app and the emulator first. If the user is using Android Studio as the IDE of their choice, they should navigate to the Device Manager screen and select the action menu for the device being used. From the action menu, the user should select “Cold Boot Now” to clear the device's RAM and restart the device.

**Figure 62**

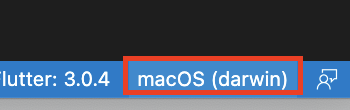
*Android Studio Virtual Devices Manager*



For users running the app in Visual Studio, the user can cold boot the device by selecting the device menu from the bottom toolbar.

**Figure 63**

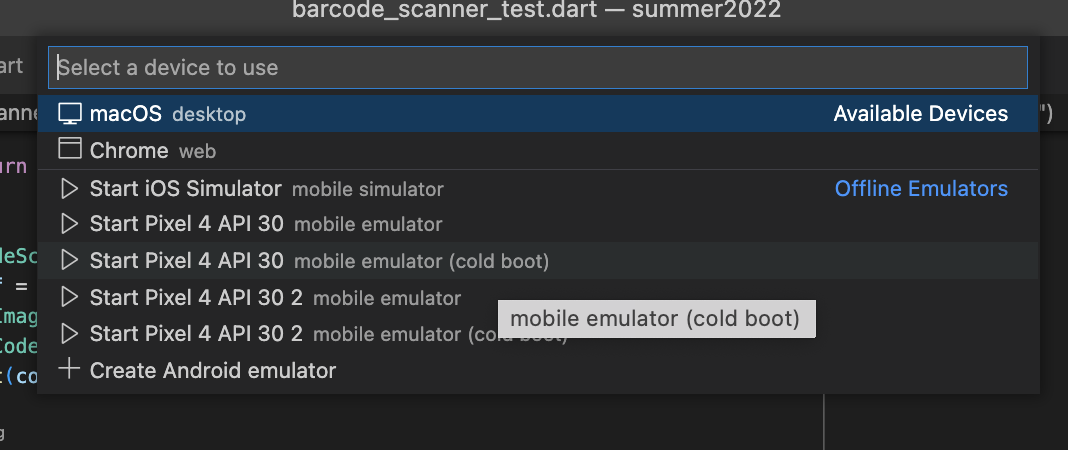
*Visual Studio Device Menu Button*



This button will launch a menu at the top of the IDE, which displays all the emulators and devices from which the user can run the application. To cold boot an emulator, please select the device that has “(cold boot)” next to its name.

**Figure 64**

*Cold Boot Option*



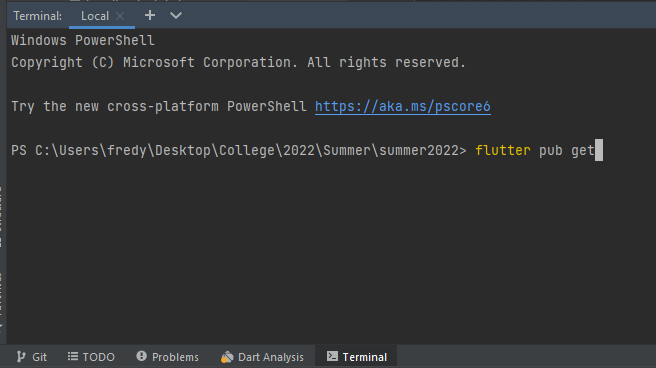
For users who are running an iPhone emulator through Xcode, it is recommended to first check if there are any issues with the build. The user should navigate to the “View” menu and under the “Navigators” option should “Show Issue Navigator” option. If no issues are visible with the build, then the user should close the emulator window and try to relaunch the device emulator again. If the issue persists, navigate to the “Device” menu and then select the “Erase All Content and Settings” option.

* 1. Dependency Errors

When a developer encounters errors with dependencies being changed this is generally due to changes in the pubspec.yaml file where packages and resources are organized. To remedy this issue a developer will need to run “flutter pub get” in the terminal to get the packages associated with the entries in the pubspec.yaml file.

**Figure 65**

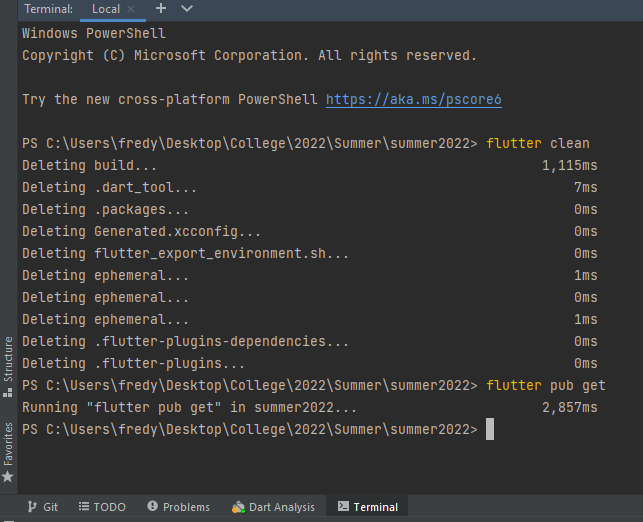
*Run flutter pub get*



If the issue persists the developer should run a “flutter clean” and then “flutter pub get” to try to resolve the issue.

**Figure 66**

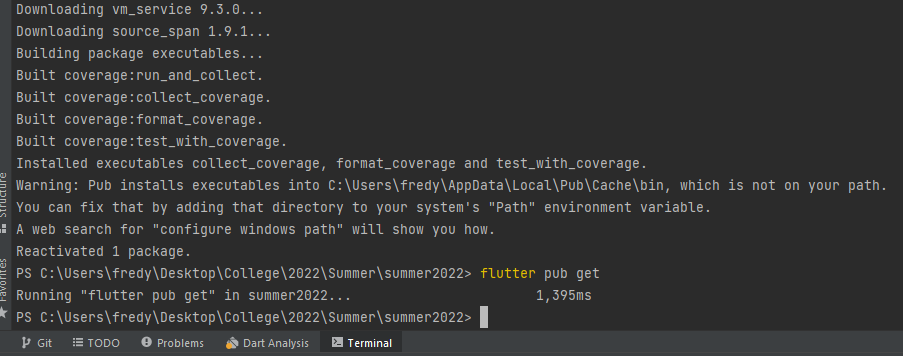
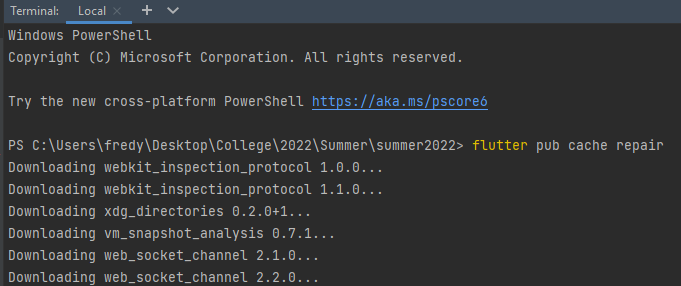
*Run flutter clean*



If neither resolves the issue the developer will need to run the “flutter pub cache repair” from the command line and then the “flutter pub get” command.

**Figure 67**

*Run flutter pub cache repair*



* 1. Known Issues and Solutions
     1. Google Vision Service Account - Violation of Google Cloud User Agreement

An issue that presented itself while developing was the suspension of the service account used to make connections to Google Vision API. Google assumes that it is one person per service account per IP address that will be making connections. For the purpose of this project this is fine. However, once the code was pushed to the developer branch and more developers were using one account credential, it got suspended. Some solutions to combat this include using Google OAuth 2.0 and to create an API Server to store the credentials on.

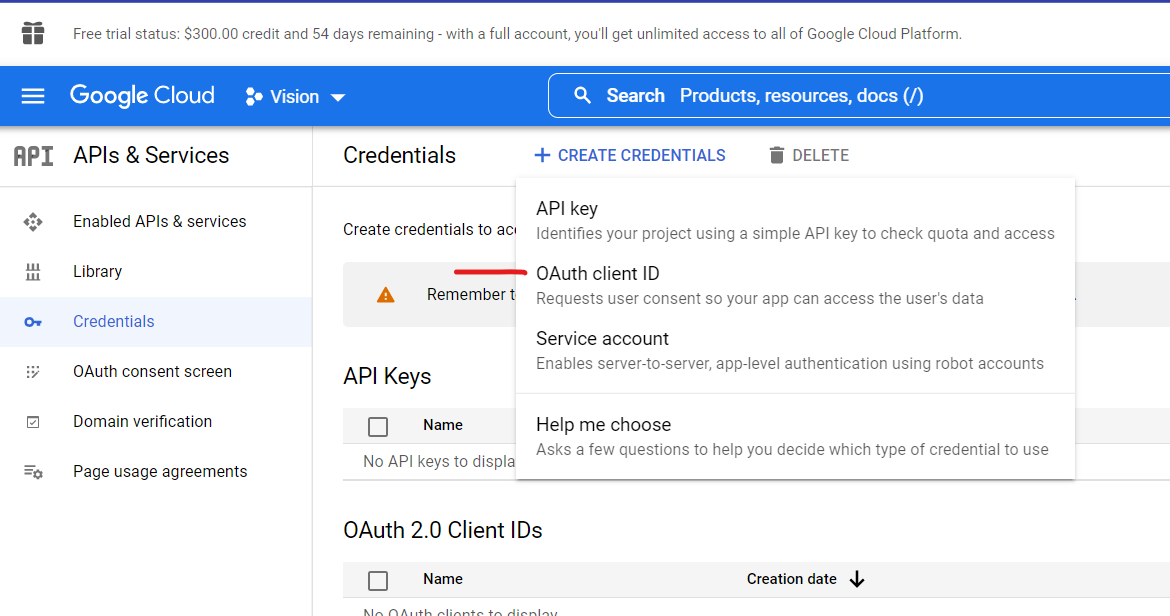
**Solution 1: Google OAuth 2.0**

Creating an OAuth credential is needed in order to make calls to Google APIs.

Go to Credentials page on the Google Cloud Console and click on create credentials. This will show options to select which credentials to create. Select eh OAuth client ID option and follow the instructions The image bellow shows this step.

**Figure 68**

*GCP Credentials Page*

.

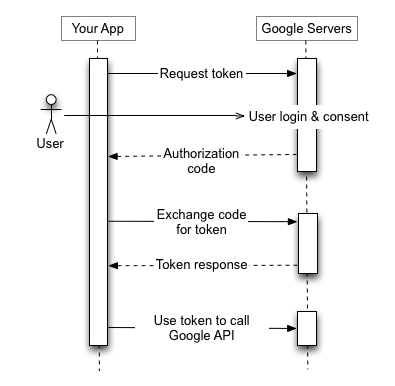
The following steps are used to make the connection calls to the Google APIs once a OAuth credential is created.

1. Request Token - oAuth client ID and client secret.
2. user redirected to Consent page to grant access to private data.
3. Authorization code - sent by Google for user to use service.
4. Use auth code for token.
5. Token Response (access/refresh) is provided to user.
6. Token is used to call Google API.

The image bellow depicts a high-level illustration of this process. More information can be found [OAuth 2.0 for Mobile & Desktop Apps | Google Identity | Google Developers](https://developers.google.com/identity/protocols/oauth2/native-app)

**Figure 69**

*UML of the OAuth 2.0*

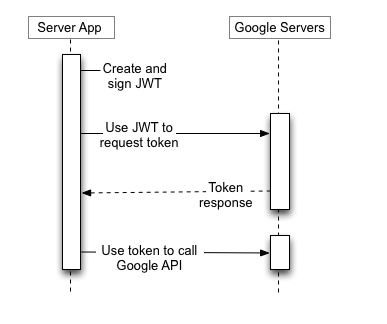
****

**Solution 2: API Server**

Another solution is to create an API Server to interact with Google Vision. On this server the service account credentials can be added and be used to interface with cloud services. This would be linked to one IP address, therefore preventing the violation of the Google Cloud User Agreement. The following image shows a high-level illustration of this process.

**Figure 70**

*UML of using the API Server*



# Appendices

* 1. References

*MacOS install*. Flutter. (n.d.). Retrieved July 12, 2022, from https://docs.flutter.dev/get-started/install/macos

*Web tools API portal*. USPS. (n.d.). Retrieved July 12, 2022, from https://www.usps.com/business/web-tools-apis/

*Windows install*. Flutter. (n.d.). Retrieved July 12, 2022, from https://docs.flutter.dev/get-started/install/windows

B. (n.d.). *Google-Ml-Kit-plugin/pubspec.yaml at master · bharat-biradar/Google-Ml-Kit-plugin*. GitHub. Retrieved July 27, 2022, from <https://github.com/bharat-biradar/Google-Ml-Kit-plugin/blob/master/packages/google_mlkit_translation/pubspec.yaml>

Academind. (2019, July 10). *Flutter Crash Course for Beginners 2021 - Build a Flutter App with Google’s Flutter & Dart*. YouTube. Retrieved July 27, 2022, from <https://www.youtube.com/watch?v=x0uinJvhNxI&t=2407s>

*css\_colors | Flutter Package*. (n.d.). Dart Packages. Retrieved July 27, 2022, from <https://pub.dev/packages/css_colors/install>

*Windows install* (n.d.) Flutter, Retrieved July 26, 2022, from <https://docs.flutter.dev/get-started/install/windows>

*Get the Dark SDK* (n.d.) Dart, Retrieved Jult 26, 2022, from <https://dart.dev/get-dart>

Google. (n.d.). *OAuth 2.0 for Mobile & Desktop Apps | google identity | google developers*. Google. Retrieved July 28, 2022, from https://developers.google.com/identity/protocols/oauth2/native-app

*MacOS install*. Flutter. (n.d.). Retrieved July 28, 2022, from https://docs.flutter.dev/get-started/install/macos

Purdue Writing Lab. (n.d.). *APA tables and figures // Purdue Writing Lab*. Purdue Writing Lab. Retrieved July 29, 2022, from https://owl.purdue.edu/owl/research\_and\_citation/apa\_style/apa\_formatting\_and\_style\_guide/apa\_tables\_and\_figures.html