Name : Kunal Rajesh Kumbhare

Prn : 120A3024

Branch : IT

3nd Year (6th semester)

Experiment No: 8

AIM: To test and deploy production ready Flutter App on Android

platform

THEORY:

Build and release an Android app

During a typical development cycle, you test an app using flutter run at the command line, or by using the **Run** and **Debug** options in your IDE. By default, Flutter builds a *debug* version of your app.

To publish an app to the Google Play Store, you need to prepare a *release* version of your app. Before publishing, you might want to put some finishing touches on your app. This may covers the following topics:

- Adding a launcher icon
- Enabling Material Components
- Signing the app
- Shrinking your code with R8
- Enabling multidex support
- Reviewing the app manifest
- Reviewing the build configuration
- Building the app for release
- Publishing to the Google Play Store Updating the app's version number
- Android release FAQ

Note: Throughout this manual, [project] refers to the directory that your application is in. While following these instructions, substitute [project] with your app's directory.

Adding a launcher icon

When a new Flutter app is created, it has a default launcher icon. To customize this icon, you might want to check out the flutter_launcher_icons package.

Alternatively, you can do it manually using the following steps:

1. Review the Material Design product icons guidelines for icon design.
2. In the [project]/android/app/src/main/res/ directory, place your icon files in folders named using configuration qualifiers. The default mipmap- folders demonstrate the correct naming convention.
3. In AndroidManifest.xml, update the application tag's android:icon attribute to reference icons from the previous step (for example, <application android:icon="@mipmap/ic_launcher" ...).
4. To verify that the icon has been replaced, run your app and inspect the app icon in the Launcher.

Enabling Material Components

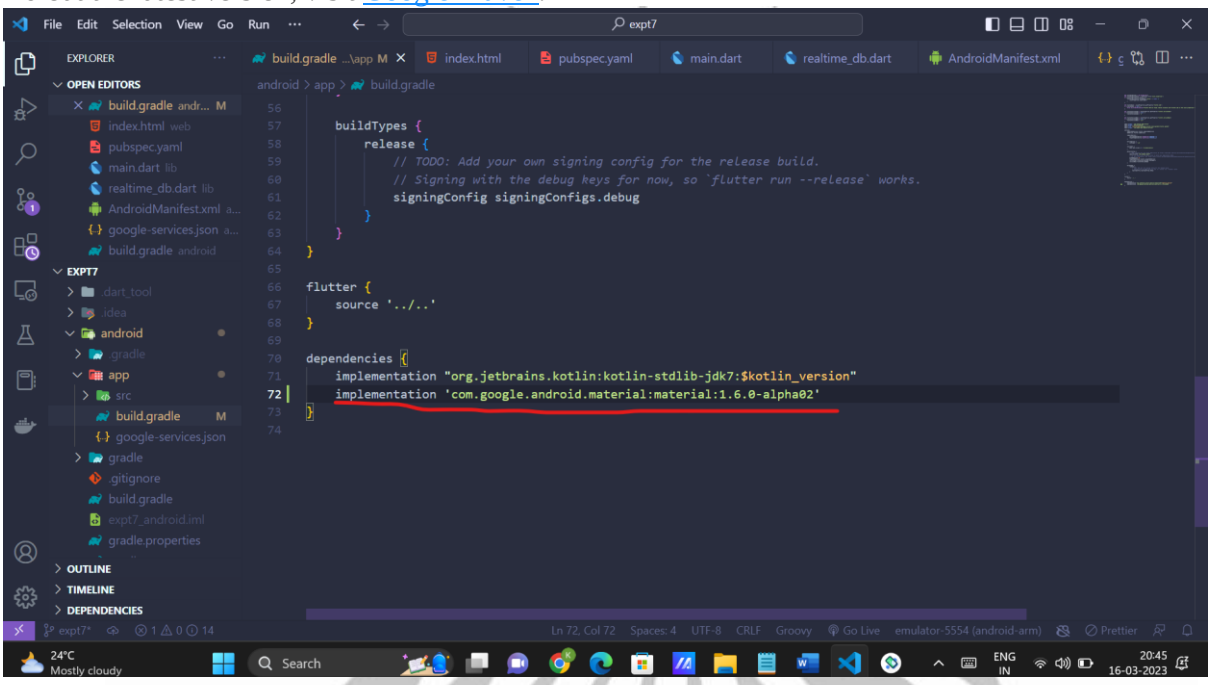
If your app uses Platform Views, you may want to enable Material Components by following the steps described in the Getting Started guide for Android.

For example:

1. Add the dependency on Android's Material in <my-app>/android/app/build.gradle:

```
dependencies {
   implementation 'com.google.android.material:material:<version>'
 }
```

To find out the latest version, visit Google Maven.

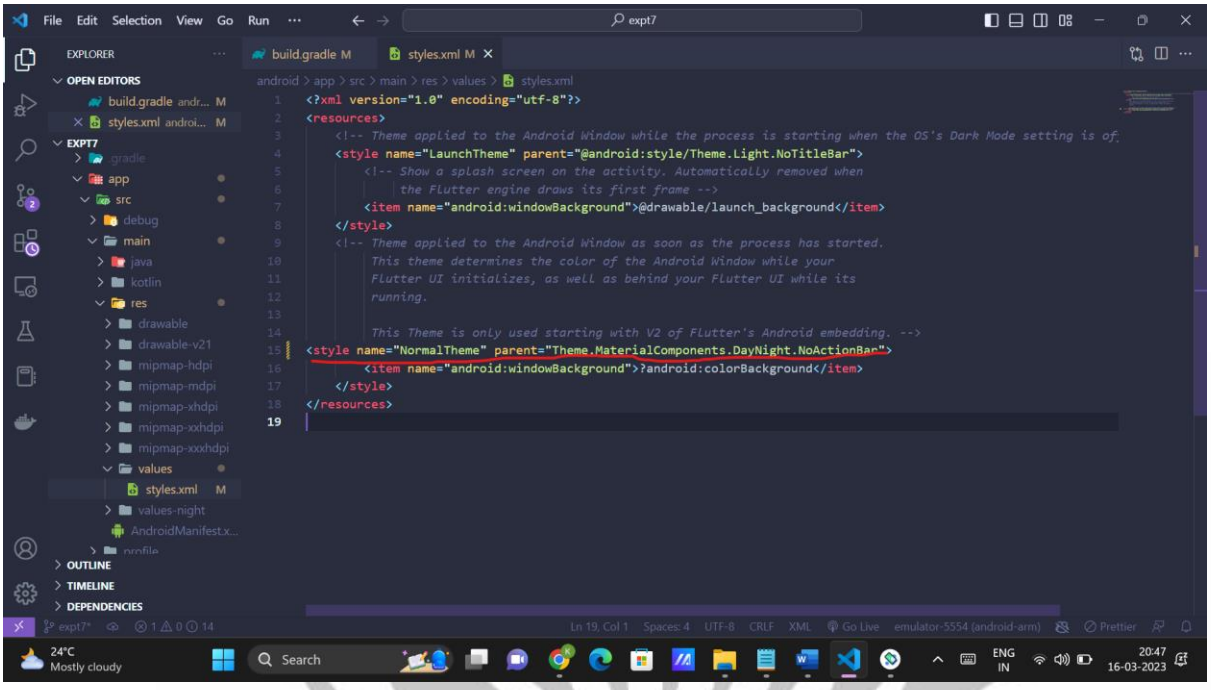


1. Set the light theme in <my-app>/android/app/src/main/res/values/styles.xml:

```
-<style name="NormalTheme" parent="@android:style/Theme.Light.NoTitleBar">
+<style name="NormalTheme" parent="Theme.MaterialComponents.Light.NoActionBar">
```

1. Set the dark theme in <my-app>/android/app/src/main/res/values-night/styles.xml

```
-<style name="NormalTheme" parent="@android:style/Theme.Black.NoTitleBar">
+<style name="NormalTheme" parent="Theme.MaterialComponents.DayNight.NoActionBar">
```



Signing the app

To publish on the Play Store, you need to give your app a digital signature. Use the following instructions to sign your app.

On Android, there are two signing keys: **deployment and upload**. The end-users download the .apk signed with the 'deployment key'. An 'upload key' is used to authenticate the .aab / .apk uploaded by developers onto the Play Store and is re-signed with the deployment key once in the Play Store.

- It's highly recommended to use the automatic cloud managed signing for the deployment key. For more information, see the official Play Store documentation.

Create an upload keystore

If you have an existing keystore, skip to the next step. If not, create one by either:

- Following the Android Studio key generation steps
- Running the following at the command line:

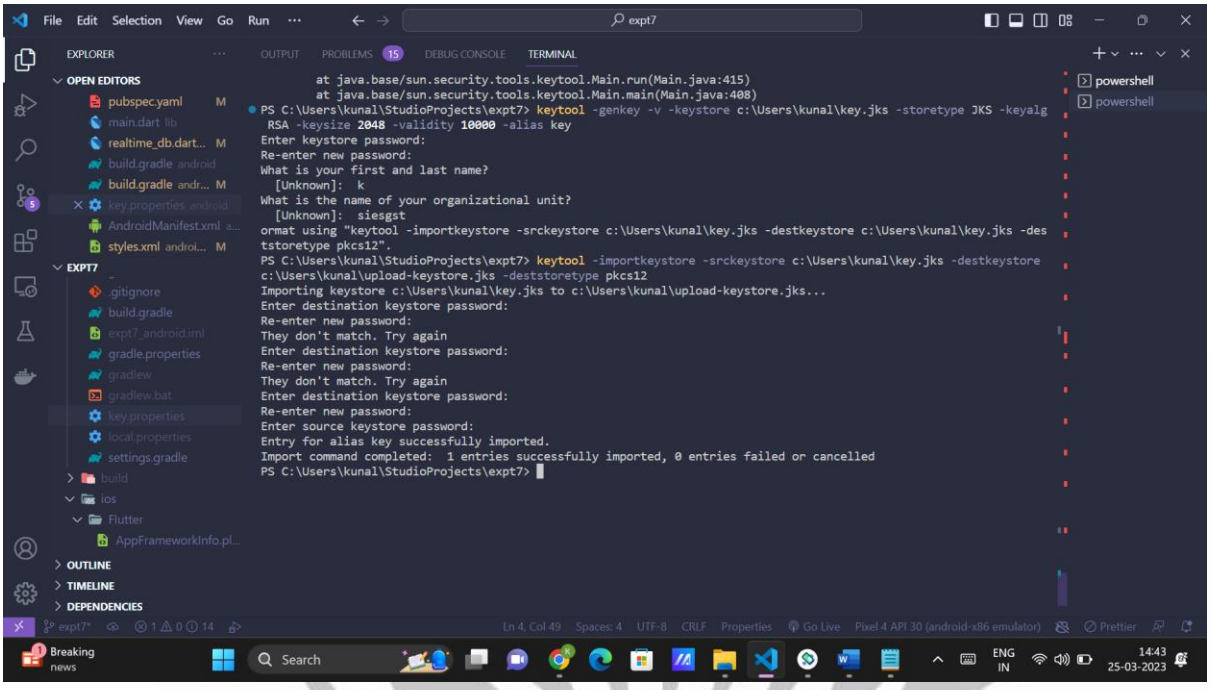
On Mac/Linux, use the following command:

```
keytool -genkey -v -keystore ~/upload-keystore.jks -keyalg RSA -keysize 2048 -validity 10000 -alias upload
```

On Windows, use the following command:

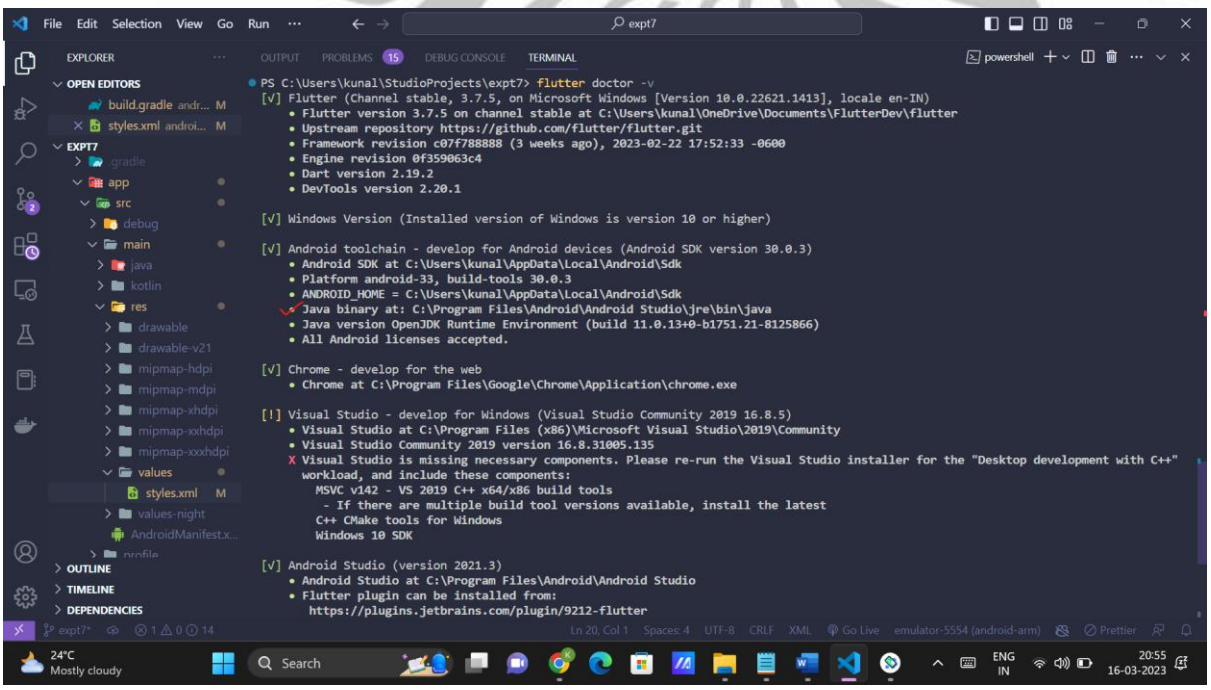
```
keytool -genkey -v -keystore c:\Users\USER_NAME\upload-keystore.jks -storetype JKS -keyalg RSA -keysize 2048 validity 10000 -alias upload
```

This command stores the upload-keystore.jks file in your home directory. If you want to store it elsewhere, change the argument you pass to the -keystore parameter. **However, keep the keystore file private; don't check it into public source control!**



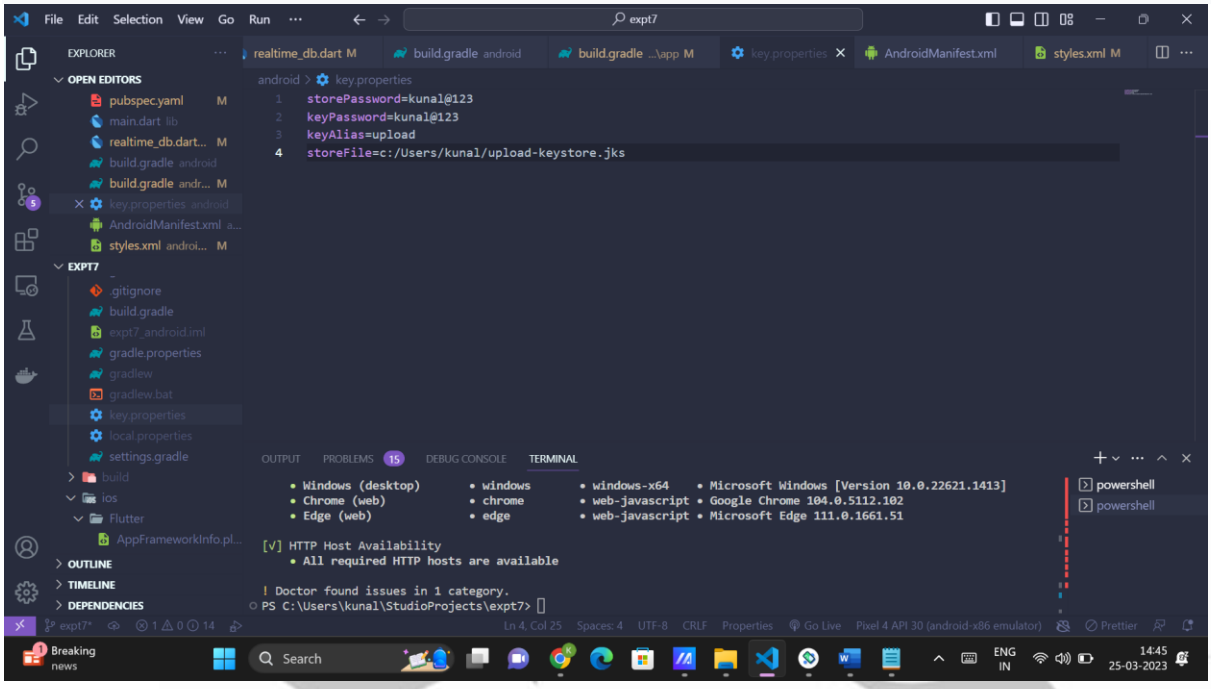
Note:

- The keytool command might not be in your path—it's part of Java, which is installed as part of Android Studio. For the concrete path, run flutter doctor -v and locate the path printed after 'Java binary at:' then use that fully qualified path replacing java (at the end) with keytool. If your path includes space-separated names, such as Program Files, use platform-appropriate notation for the names. For example, on Mac/Linux use Program\ Files, and on Windows use "Program Files".
- The -storetype JKS tag is only required for Java 9 or newer. As of the Java 9 release, the keystore type defaults to PKS12.



Reference the keystore from the app

```
storePassword=<password from previous step>
keyPassword=<password from previous step> keyAlias=upload
storeFile=<location of the key store file, such as /Users/<user name>/upload-keystore.jks>
```



Create a file named [project]/android/key.properties that contains a reference to your keystore:

Configure signing in gradle

Configure gradle to use your upload key when building your app in release mode by editing the [project]/android/app/build.gradle file.

1. Add the keystore information from your properties file before the android block:

```
def keystoreProperties = new Properties()
def keystorePropertiesFile = rootProject.file('key.properties')
if (keystorePropertiesFile.exists()) {
    keystoreProperties.load(new FileInputStream(keystorePropertiesFile))
}

android {
    ...
}
```

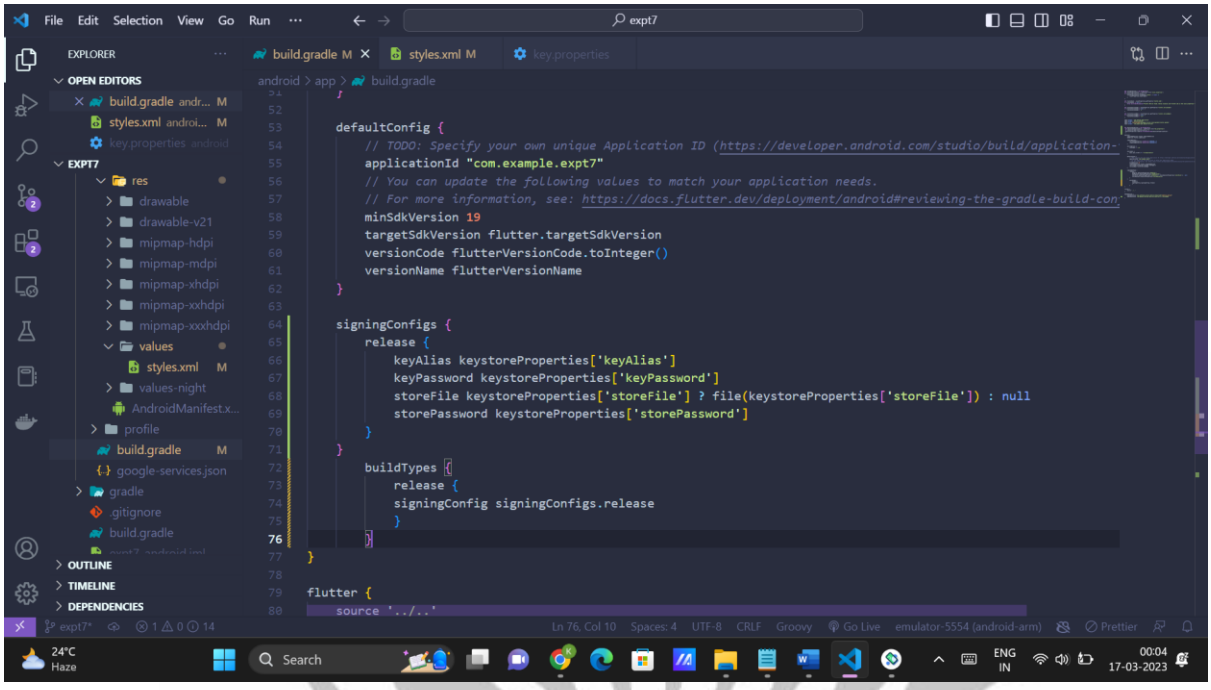
   Load the key.properties file into the keystoreProperties object.

2. Find the buildTypes block:

```
buildTy
    release {
        // TODO: Add your own signing config for the release build.
        // Signing with the debug keys for now,
        // so `flutter run --release` works.
        signingConfig signingConfigs.debug
    }
}
```

And replace it with the following signing configuration info:

```
signingConfigs {
    release {
        keyAlias keystoreProperties['keyAlias']
        keyPassword keystoreProperties['keyPassword']
        storeFile keystoreProperties['storeFile'] ? file(keystoreProperties['storeFile']) : null
        storePassword keystoreProperties['storePassword']
    }
}
buildTypes {
    release {
        signingConfig signingConfigs.release
    }
}
```



Release builds of your app will now be signed automatically.

Shrinking your code with R8

R8 is the new code shrinker from Google, and it's enabled by default when you build a release APK or AAB. To disable R8, pass the --no-shrink flag to flutter build apk or flutter build appbundle.

Note: Obfuscation and minification can considerably extend compile time of the Android application.

Enabling multidex support

When writing large apps or making use of large plugins, you may encounter Android's dex limit of 64k methods when targeting a minimum API of 20 or below. This may also be encountered when running debug versions of your app via flutter run that does not have shrinking enabled.

Flutter tool supports easily enabling multidex. The simplest way is to opt into multidex support when prompted. The tool detects multidex build errors and will ask before making changes to your Android project. Opting in allows Flutter to automatically depend on androidx.multidex:multidex and use a generated FlutterMultiDexApplication as the project's application.

Note: Multidex support is natively included when targeting min sdk 21+.

Reviewing the app manifest

Review the default App Manifest file, AndroidManifest.xml, located in [project]/android/app/src/main and verify that the values are correct, especially the following:

application

Edit the android:label in the application tag to reflect the final name of the app. uses-permission

Add the android.permission.INTERNET permission if your application code needs Internet access. The standard template does not include this tag but allows Internet access during development to enable communication between Flutter tools and a running app.

Reviewing the build configuration:

Review the default Gradle build file (build.gradle) located in [project]/android/app and the local.properties file located in [project]/android to verify the values are correct, especially the following values in the defaultConfig block: In build.gradle file applicationId

Specify the final, unique (Application Id) appid compileSdkVersion

Specify the API level Gradle should use to compile your app. For more information, see the module-level build section in the Gradle build file.

buildToolsVersion

If you're using Android plugin for Gradle 3.0.0 or higher, your project automatically uses the default version of the build tools that the plugin specifies. Alternatively, you can specify a version of the build tools.

In local.properties file flutter.versionCode

& flutter.versionName

Specify the internal app version number, and the version number display string. You can do this by setting the version property in the pubspec.yaml file. For more information, see the version information guidance in the versions documentation.

flutter.minSdkVersion & flutter.targetSdkVersion

Specify the minimum API level and the target API level on which the app is designed to run. For more information, see the API level section in the versions documentation.

Building the app for release

You have two possible release formats when publishing to the Play Store.

- AAB (Android App bundle) - preferred
- APK

Build an app bundle

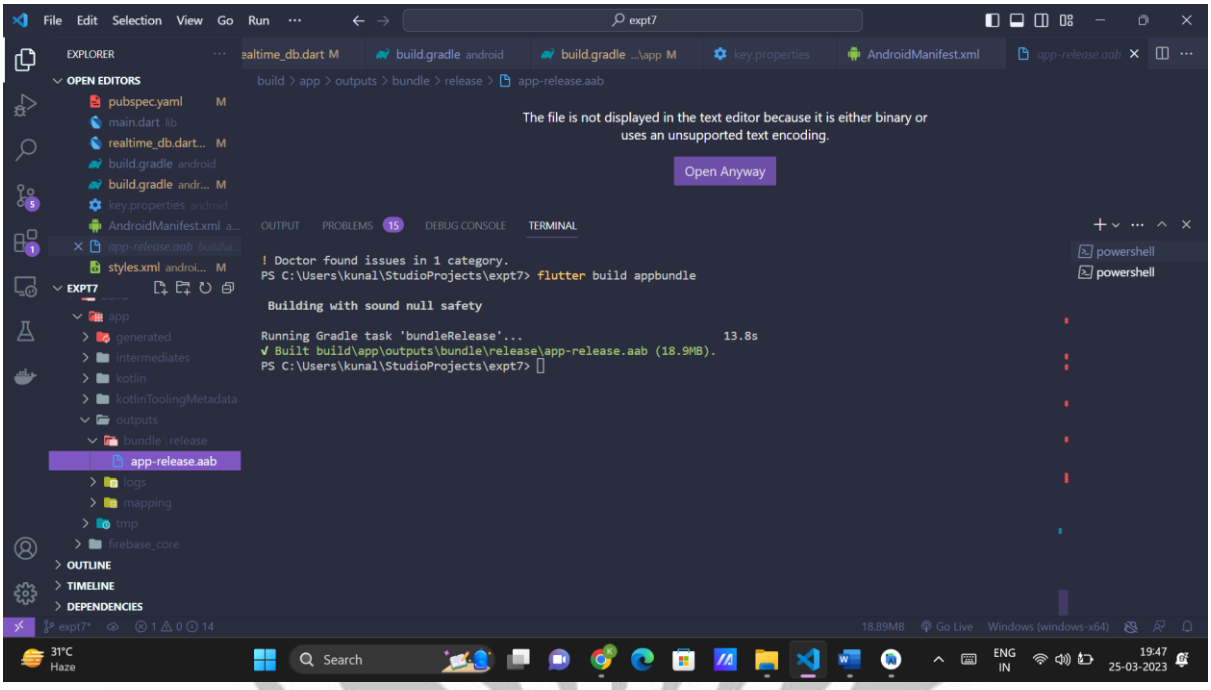
This section describes how to build a release app bundle. If you completed the signing steps, the app bundle will be signed. At this point, you might consider obfuscating your Dart code to make it more difficult to reverse engineer. Obfuscating your code involves adding a couple flags to your build command, and maintaining additional files to deobfuscate stack traces.

From the command line:

1. Enter cd [project]
2. Run flutter build appbundle (Running flutter build defaults to a release build.)

The release bundle for your app is created at [project]/build/app/outputs/bundle/release/app.aab.

By default, the app bundle contains your Dart code and the Flutter runtime compiled for armeabi-v7a (ARM 32bit), arm64-v8a (ARM 64-bit), and x86-64 (x86 64-bit).



Test the app bundle

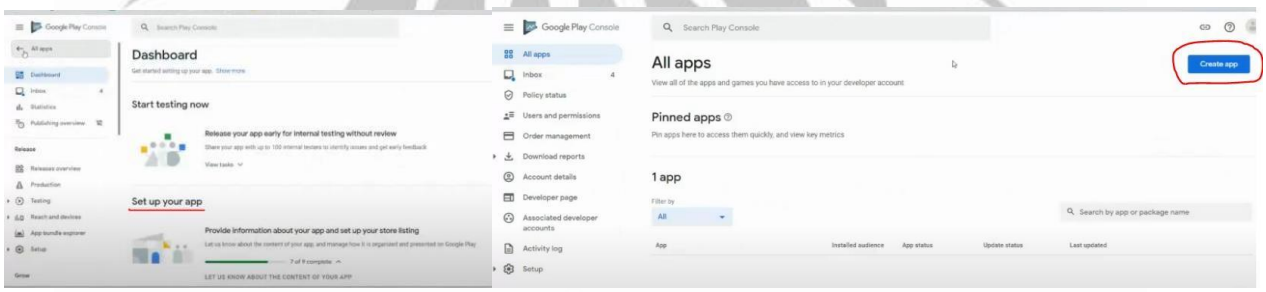
An app bundle can be tested in multiple ways—this section describes two.

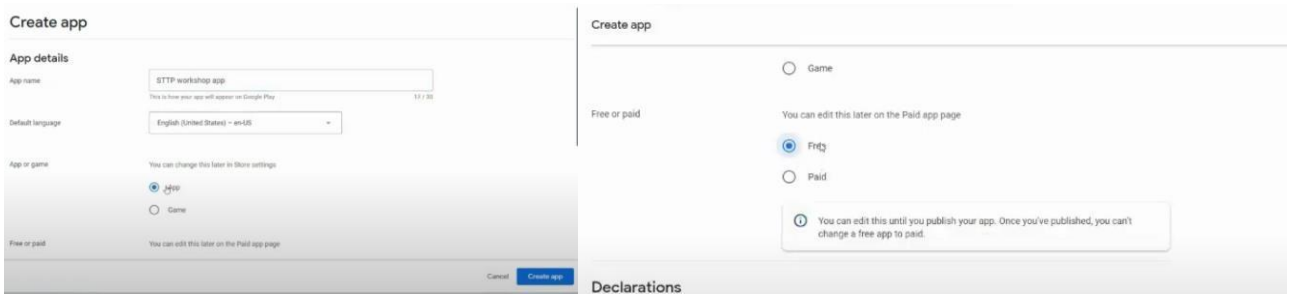
Offline using the bundle tool

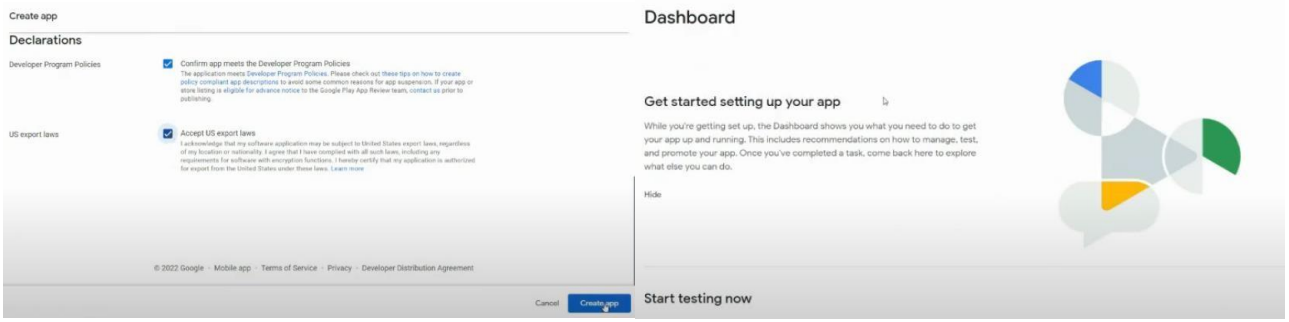
1. If you haven't done so already, download bundletool from the GitHub repository.
2. Generate a set of APKs from your app bundle.
3. Deploy the APKs to connected devices.

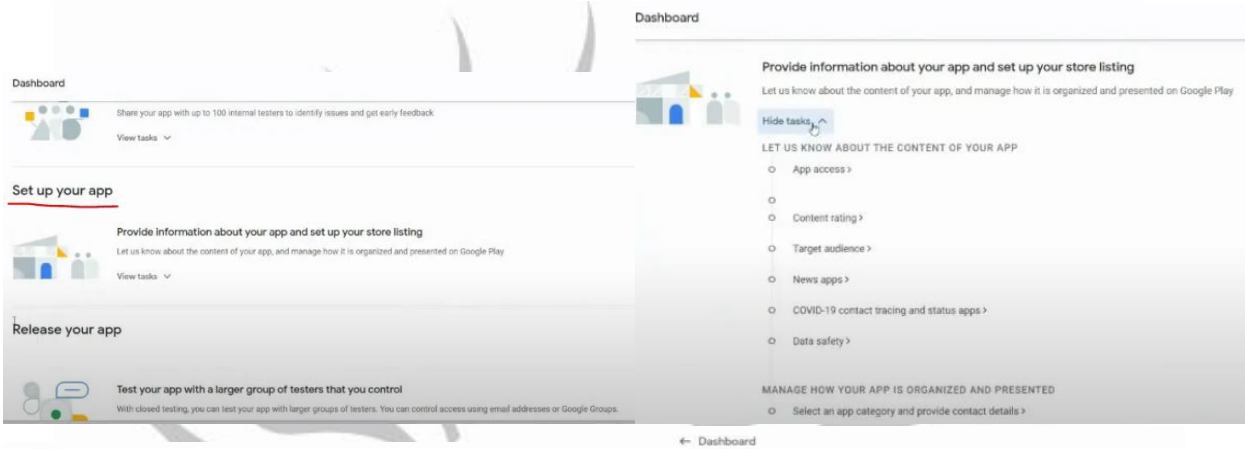
Online using Google Play

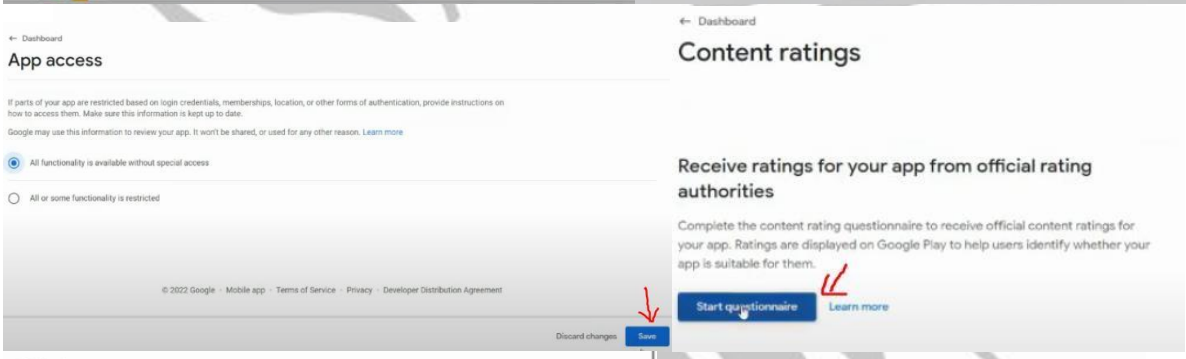
1. Upload your bundle to Google Play to test it. You can use the internal test track, or the alpha or beta channels to test the bundle before releasing it in production.
2. Follow these steps to upload your bundle to the Play Store.

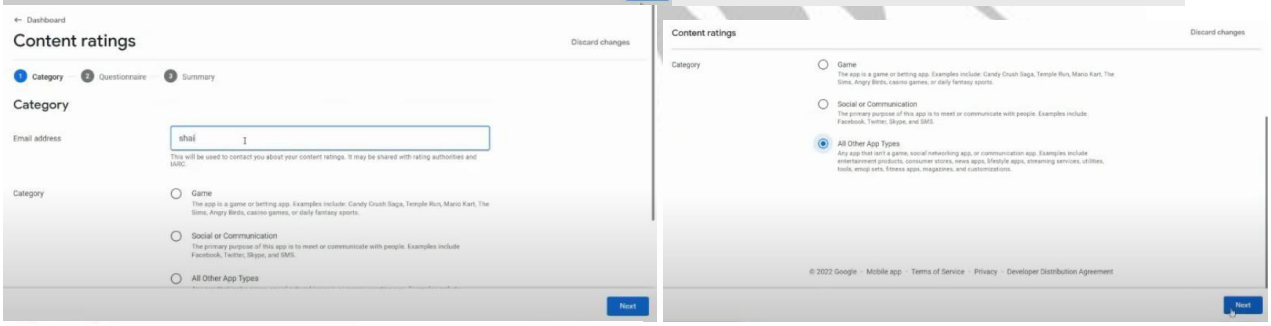


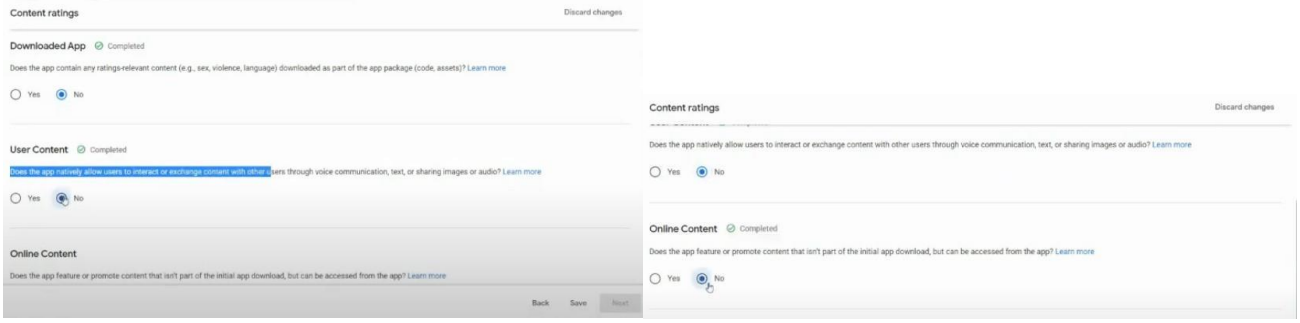


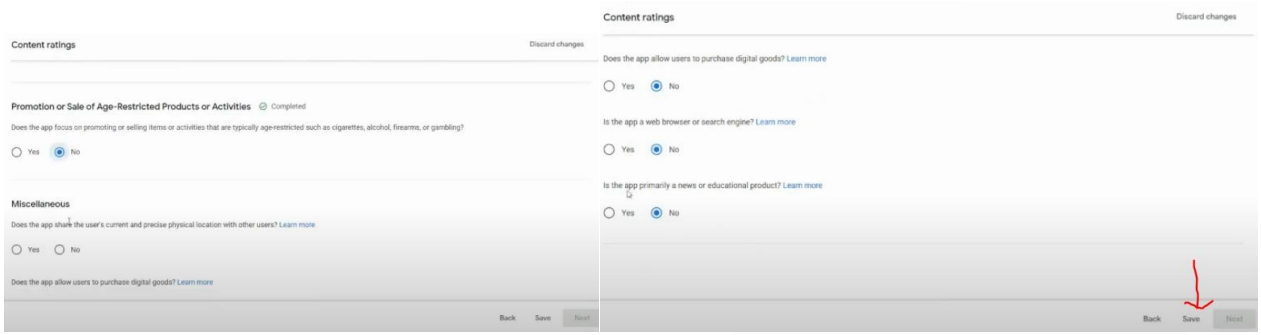

Create app
Declarations
Developer Program Policies
Confirm app meets the Developer Program Policies
US export laws
Accept US export laws
Cancel
Create app
Dashboard
Get started setting up your app
While you're getting set up, the Dashboard shows you what you need to do to get your app up and running. This includes recommendations on how to manage, test, and promote your app. Once you've completed a task, come back here to explore what else you can do.
Hide
Start testing now

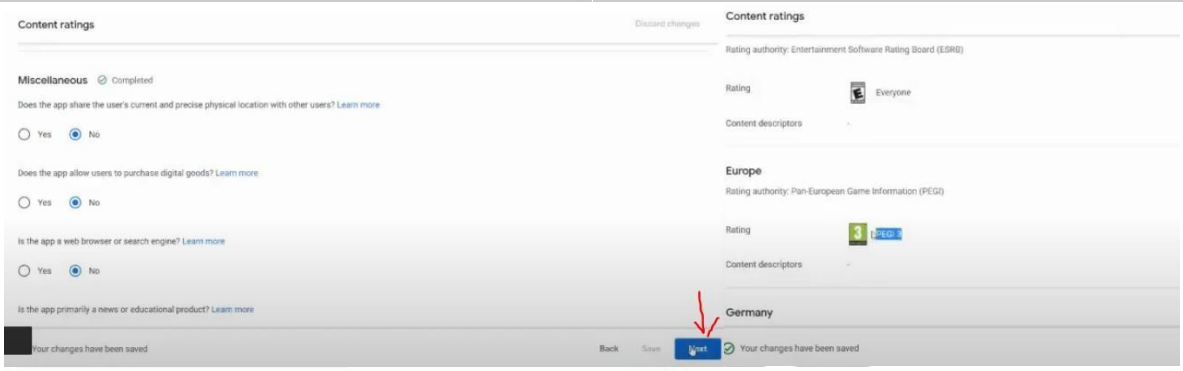

Dashboard
Share your app with up to 100 internal testers to identify issues and get early feedback
View tasks
Set up your app
Provide information about your app and set up your store listing
Let us know about the content of your app, and manage how it is organized and presented on Google Play
View tasks
Release your app
Test your app with a larger group of testers that you control
With closed testing, you can test your app with larger groups of testers. You can control access using email addresses or Google Groups.
Dashboard
Provide information about your app and set up your store listing
Let us know about the content of your app, and manage how it is organized and presented on Google Play
Hide tasks
LET US KNOW ABOUT THE CONTENT OF YOUR APP
App access
Content rating
Target audience
News apps
COVID-19 contact tracing and status apps
Data safety
MANAGE HOW YOUR APP IS ORGANIZED AND PRESENTED
Select an app category and provide contact details

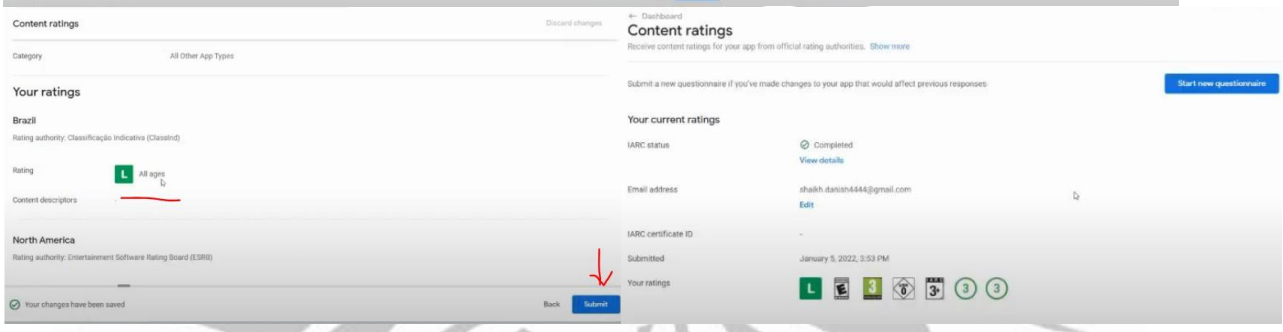

Dashboard
App access
If parts of your app are restricted based on login credentials, memberships, location, or other forms of authentication, provide instructions on how to access them. Make sure this information is kept up to date.
Google may use this information to review your app. It won't be shared, or used for any other reason. Learn more
All functionality is available without special access
All or some functionality is restricted
Discard changes
Save
Dashboard
Content ratings
Receive ratings for your app from official rating authorities
Complete the content rating questionnaire to receive official content ratings for your app. Ratings are displayed on Google Play to help users identify whether your app is suitable for them.
Start questionnaire
Learn more

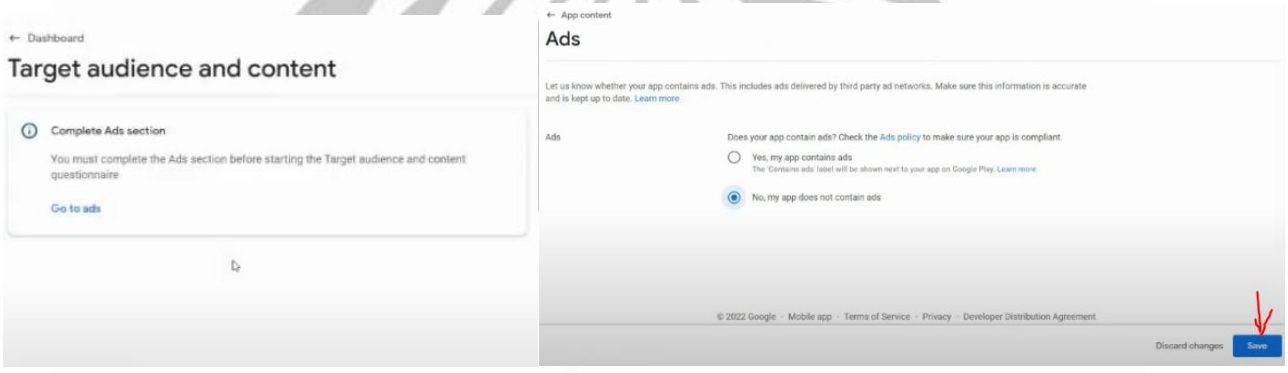

Content ratings
Discard changes
Category
Questionnaire
Summary
Category
Email address
shai
Game
Social or Communication
All Other App Types
Next
Content ratings
Discard changes
Category
Game
Social or Communication
All Other App Types
Next

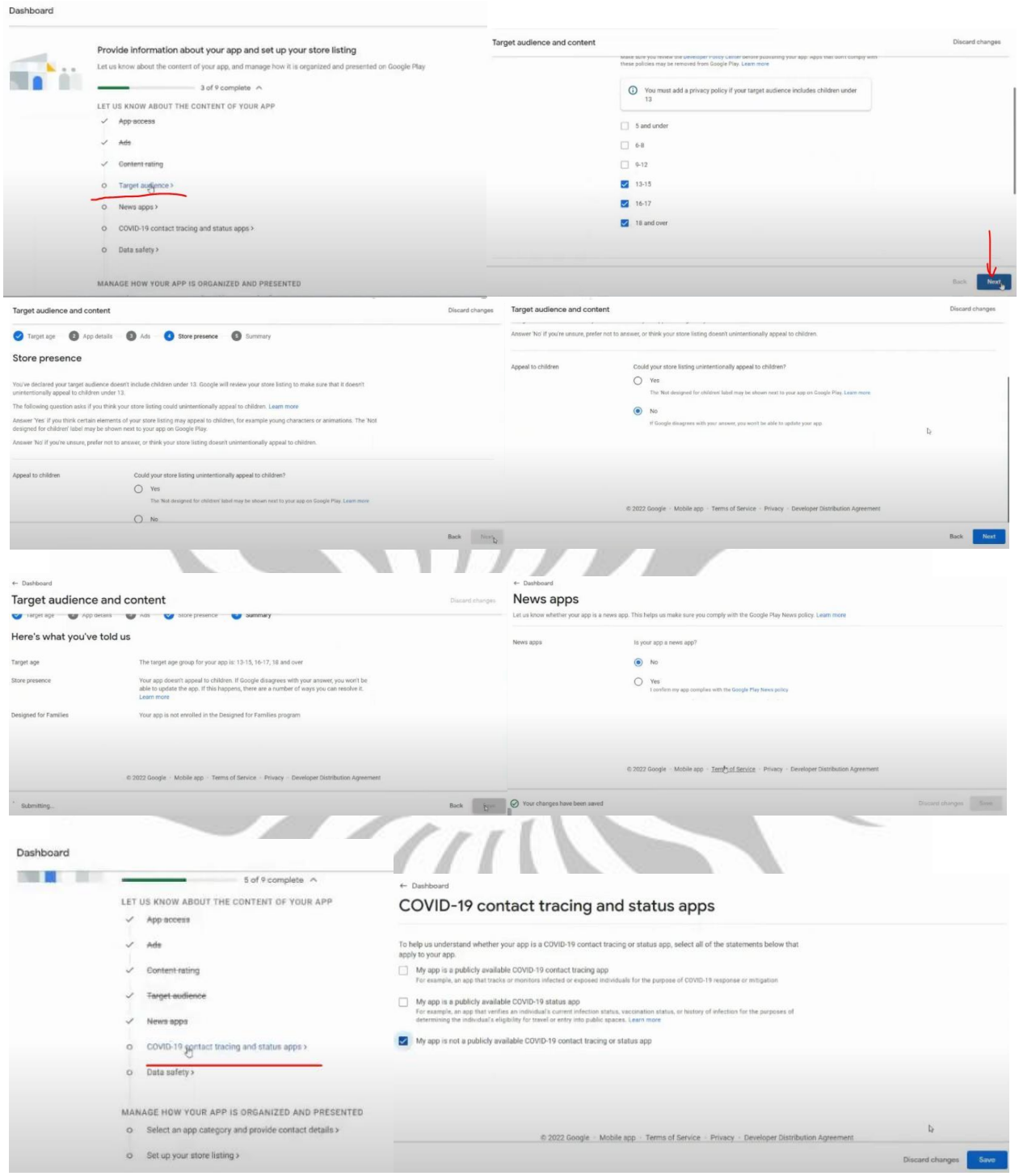

Content ratings
Discard changes
Downloaded App
Completed
Does the app contain any ratings-relevant content (e.g., sex, violence, language) downloaded as part of the app package (code, assets)? Learn more
Yes
No
User Content
Completed
Does the app natively allow users to interact or exchange content with other users through voice communication, text, or sharing images or audio? Learn more
Yes
No
Online Content
Does the app feature or promote content that isn't part of the initial app download, but can be accessed from the app? Learn more
Back
Save
Next
Content ratings
Discard changes
Does the app natively allow users to interact or exchange content with other users through voice communication, text, or sharing images or audio? Learn more
Yes
No
Online Content
Completed
Does the app feature or promote content that isn't part of the initial app download, but can be accessed from the app? Learn more
Yes
No

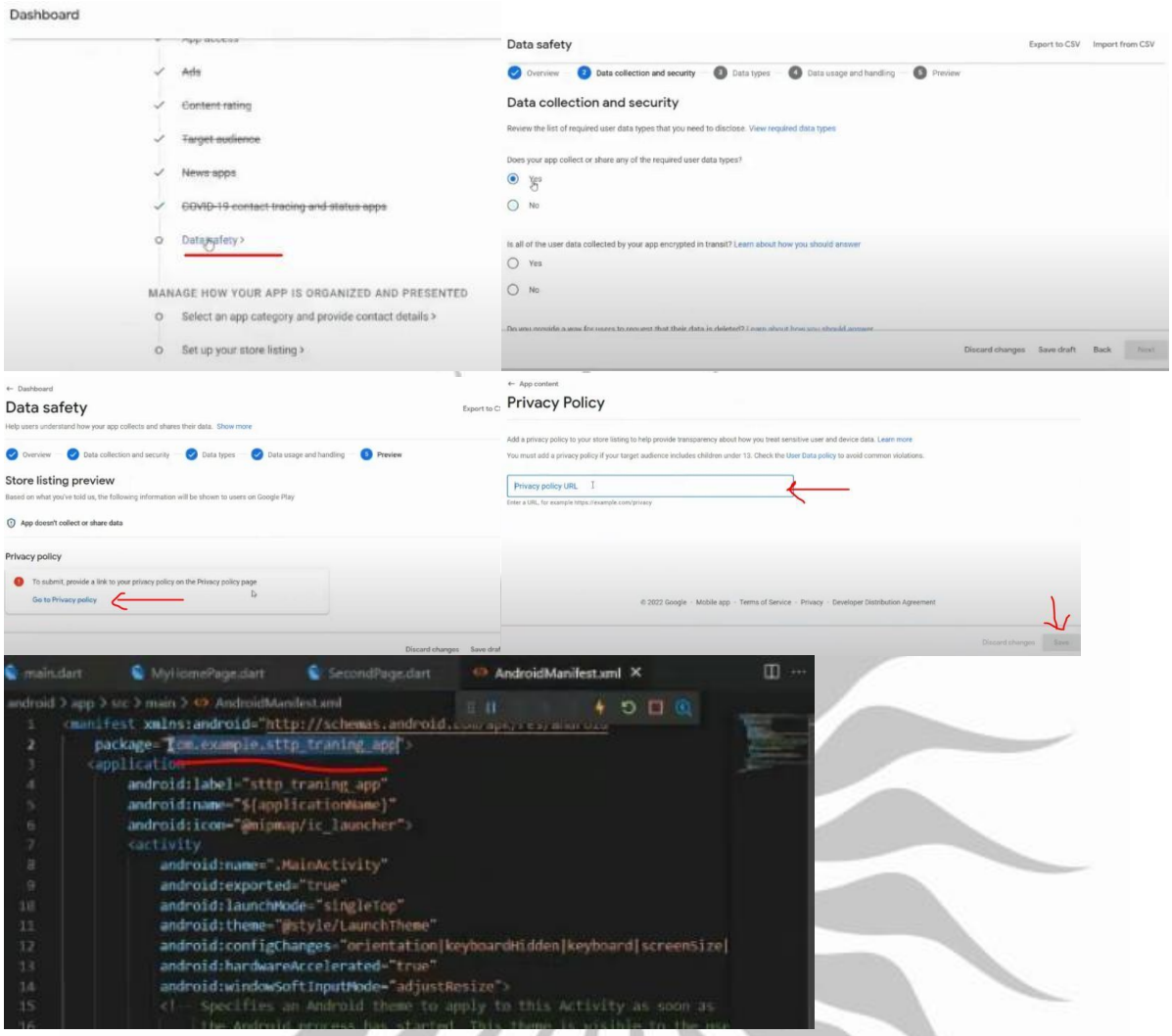

Content ratings
Discard changes
Promotion or Sale of Age-Restricted Products or Activities
Completed
Does the app focus on promoting or selling items or activities that are typically age-restricted such as cigarettes, alcohol, firearms, or gambling?
Yes
No
Miscellaneous
Does the app share the user's current and precise physical location with other users? Learn more
Does the app allow users to purchase digital goods? Learn more
Back
Save
Next
Content ratings
Discard changes
Does the app allow users to purchase digital goods? Learn more
Is the app a web browser or search engine? Learn more
Is the app primarily a news or educational product? Learn more
Back
Save
Next


Content ratings
Miscellaneous
Completed
Does the app share the user's current and precise physical location with other users? Learn more
Does the app allow users to purchase digital goods? Learn more
Is the app a web browser or search engine? Learn more
Is the app primarily a news or educational product? Learn more
Your changes have been saved
Back
Save
Next
Content ratings
Rating authority: Entertainment Software Rating Board (ESRB)
Rating
Everyone
Content descriptors
Europe
Rating authority: Pan-European Game Information (PEGI)
Rating
Content descriptors
Germany
Your changes have been saved


Content ratings
Category
All Other App Types
Your ratings
Brazil
Rating authority: Classificação Indicativa (ClassInd)
Rating
All ages
Content descriptors
North America
Rating authority: Entertainment Software Rating Board (ESRB)
Your changes have been saved
Back
Submit
Dashboard
Content ratings
Receive content ratings for your app from official rating authorities. Show more
Submit a new questionnaire if you've made changes to your app that would affect previous responses
Start new questionnaire
Your current ratings
IARC status
Completed
View details
Email address
shaikh.danish4444@gmail.com
Edit
IARC certificate ID
Submitted
January 5, 2022, 3:53 PM
Your ratings


Dashboard
Target audience and content
Complete Ads section
You must complete the Ads section before starting the Target audience and content questionnaire
Go to ads
App content
Ads
Let us know whether your app contains ads. This includes ads delivered by third party ad networks. Make sure this information is accurate and is kept up to date. Learn more
Ads
Does your app contain ads? Check the Ads policy to make sure your app is compliant.
Yes, my app contains ads
The 'Contains ads' label will be shown next to your app on Google Play. Learn more
No, my app does not contain ads
© 2022 Google · Mobile app · Terms of Service · Privacy · Developer Distribution Agreement
Discard changes
Save


Dashboard
Provide information about your app and set up your store listing
Let us know about the content of your app, and manage how it is organized and presented on Google Play
3 of 9 complete
LET US KNOW ABOUT THE CONTENT OF YOUR APP
App access
Ads
Content rating
Target audience
News apps
COVID-19 contact tracing and status apps
Data safety
MANAGE HOW YOUR APP IS ORGANIZED AND PRESENTED
Target audience and content
Discard changes
You must add a privacy policy if your target audience includes children under 13
5 and under
6-8
9-12
13-15
16-17
18 and over
Back
Next
Target age
App details
Ads
Store presence
Summary
Store presence
You've declared your target audience doesn't include children under 13. Google will review your store listing to make sure that it doesn't unintentionally appeal to children under 13.
The following question asks if you think your store listing could unintentionally appeal to children. Learn more
Answer 'Yes' if you think certain elements of your store listing may appeal to children, for example young characters or animations. The 'Not designed for children' label may be shown next to your app on Google Play.
Answer 'No' if you're unsure, prefer not to answer, or think your store listing doesn't unintentionally appeal to children.
Appeal to children
Could your store listing unintentionally appeal to children?
Yes
The 'Not designed for children' label may be shown next to your app on Google Play. Learn more
No
If Google disagrees with your answer, you won't be able to update your app.
© 2022 Google · Mobile app · Terms of Service · Privacy · Developer Distribution Agreement
Dashboard
Target audience and content
Here's what you've told us
Target age
The target age group for your app is: 13-15, 16-17, 18 and over
Store presence
Your app doesn't appeal to children. If Google disagrees with your answer, you won't be able to update the app. If this happens, there are a number of ways you can resolve it.
Learn more
Designed for Families
Your app is not enrolled in the Designed for Families program
Submitting...
News apps
Let us know whether your app is a news app. This helps us make sure you comply with the Google Play News policy. Learn more
Is your app a news app?
No
Yes
I confirm my app complies with the Google Play News policy
Your changes have been saved
Discard changes
Save
5 of 9 complete
App access
Ads
Content rating
Target audience
News apps
COVID-19 contact tracing and status apps
Data safety
MANAGE HOW YOUR APP IS ORGANIZED AND PRESENTED
Select an app category and provide contact details
Set up your store listing
COVID-19 contact tracing and status apps
To help us understand whether your app is a COVID-19 contact tracing or status app, select all of the statements below that apply to your app.
My app is a publicly available COVID-19 contact tracing app
For example, an app that tracks or monitors infected or exposed individuals for the purpose of COVID-19 response or mitigation
My app is a publicly available COVID-19 status app
For example, an app that verifies an individual's current infection status, vaccination status, or history of infection for the purposes of determining the individual's eligibility for travel or entry into public spaces. Learn more
My app is not a publicly available COVID-19 contact tracing or status app



Build an APK

Although app bundles are preferred over APKs, there are stores that don't yet support app bundles. In this case, build a release APK for each target ABI (Application Binary Interface).

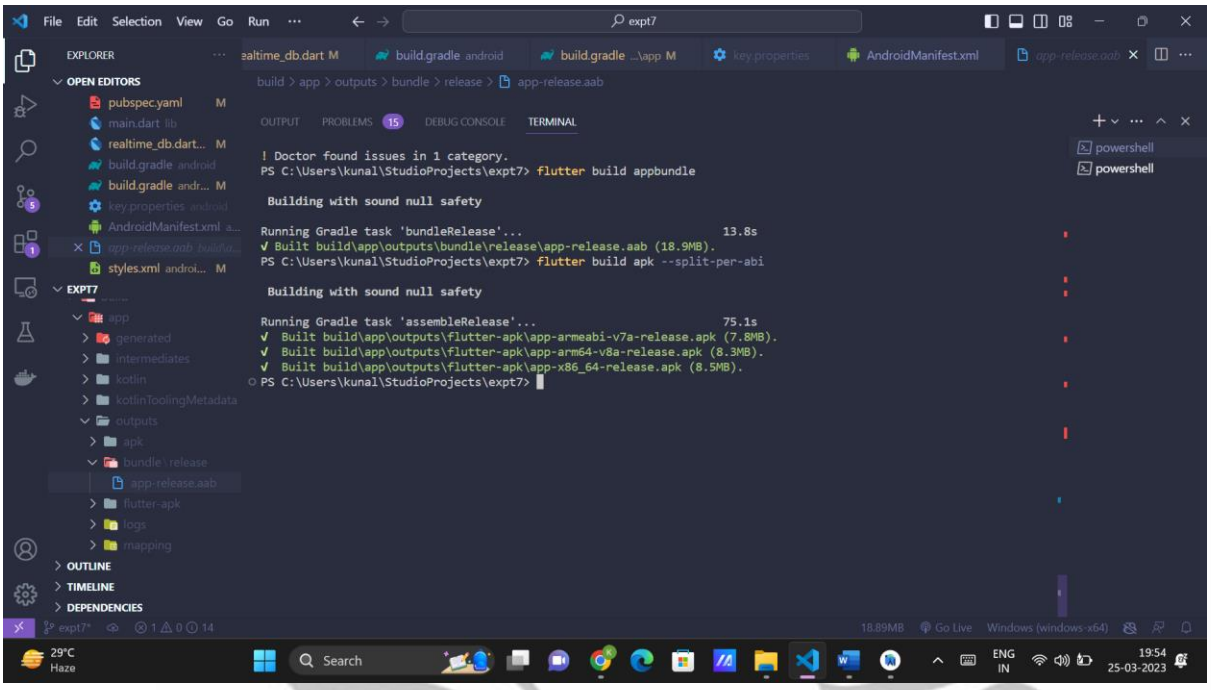
If you completed the signing steps, the APK will be signed. At this point, you might consider obfuscating your Dart code to make it more difficult to reverse engineer. Obfuscating your code involves adding a couple flags to your build command.
From the command line:

1. Enter cd [project]
2. Run flutter build apk --split-per-abi
   (The flutter build command defaults to --release.) This

command results in three APK files:

- [project]/build/app/outputs/apk/release/app-armeabi-v7a-release.apk
- [project]/build/app/outputs/apk/release/app-arm64-v8a-release.apk
- [project]/build/app/outputs/apk/release/app-x86_64-release.apk

Removing the --split-per-abi flag results in a fat APK that contains your code compiled for *all* the target ABIs. Such APKs are larger in size than their split counterparts, causing the user to download native binaries that are not applicable to their device's architecture.



Install an APK on a device

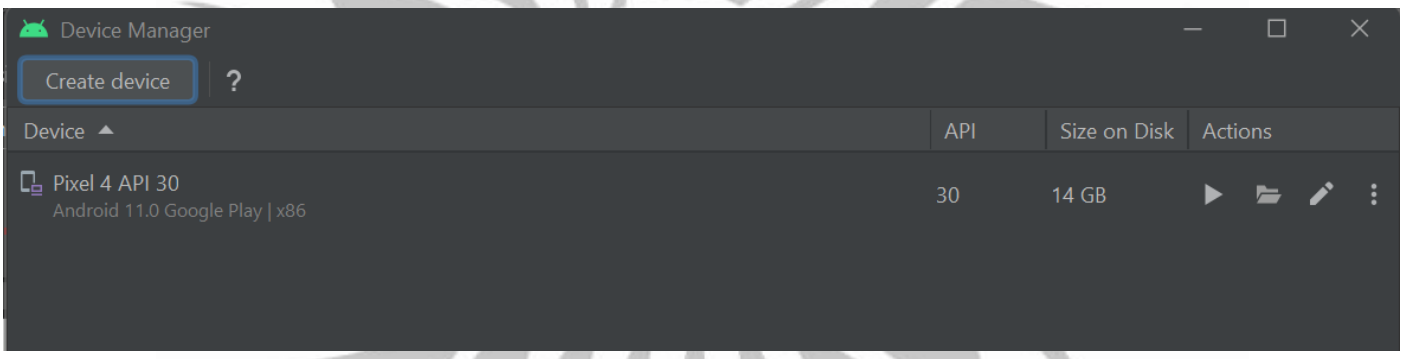
Follow these steps to install the APK on a connected Android device.

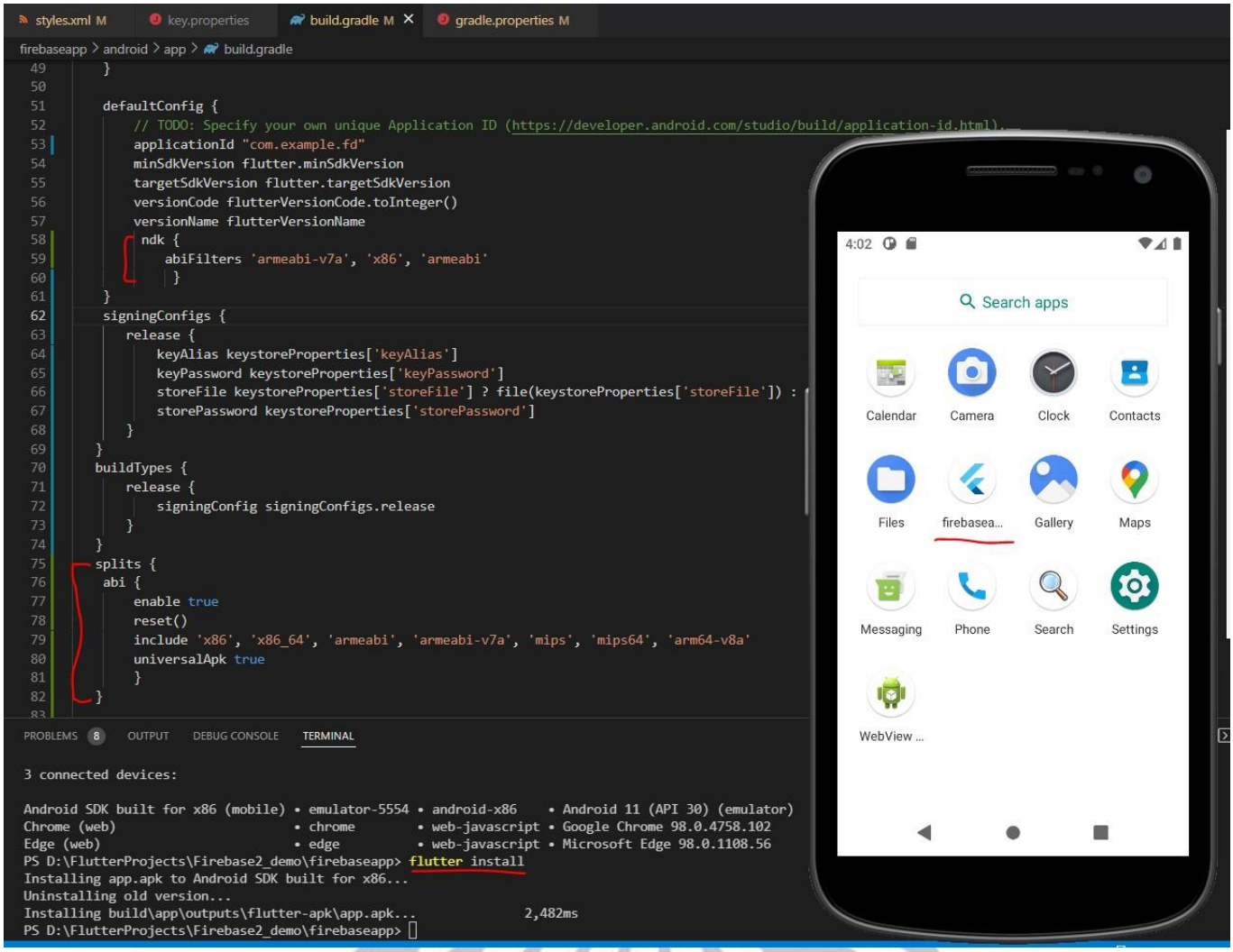
From the command line:

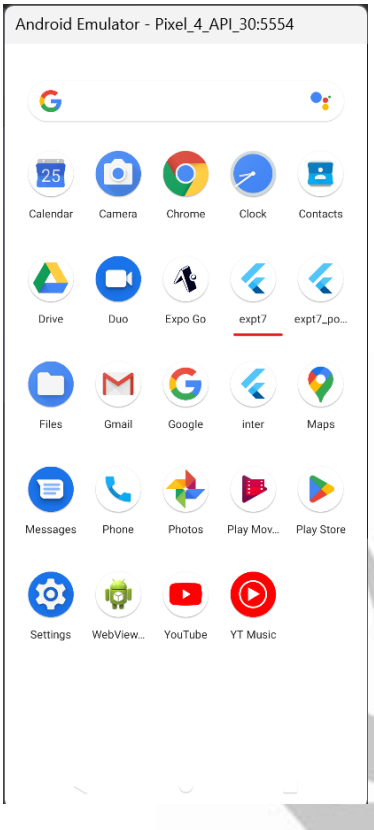
1. Connect your Android device to your computer with a USB cable.
2. Enter cd [project]. 3. Run flutter install.

```
PS D:\FlutterProjects\Firebase2_demo\firebaseapp> flutter install
Installing app.apk to Redmi Note 6 Pro...
Installing build\app\outputs\flutter-apk\app.apk...                18.8s
PS D:\FlutterProjects\Firebase2_demo\firebaseapp>
```

To install it on emulator, create an AVD with an x86 image from Android Studio:




styles.xml M
key.properties
build.gradle M
gradle.properties M
firebaseapp > android > app > build.gradle
defaultConfig {
// TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).
applicationId "com.example.fd"
minSdkVersion flutter.minSdkVersion
targetSdkVersion flutter.targetSdkVersion
versionCode flutterVersionCode.toInteger()
versionName flutterVersionName
ndk {
abiFilters 'armeabi-v7a', 'x86', 'armeabi'
}
}
signingConfigs {
release {
keyAlias keystoreProperties['keyAlias']
keyPassword keystoreProperties['keyPassword']
storeFile keystoreProperties['storeFile'] ? file(keystoreProperties['storeFile']) :
storePassword keystoreProperties['storePassword']
}
}
buildTypes {
release {
signingConfig signingConfigs.release
}
}
splits {
abi {
enable true
reset()
include 'x86', 'x86_64', 'armeabi', 'armeabi-v7a', 'mips', 'mips64', 'arm64-v8a'
universalApk true
}
}
PROBLEMS 8
OUTPUT
DEBUG CONSOLE
TERMINAL
3 connected devices:
Android SDK built for x86 (mobile) • emulator-5554 • android-x86 • Android 11 (API 30) (emulator)
Chrome (web) • chrome • web-javascript • Google Chrome 98.0.4758.102
Edge (web) • edge • web-javascript • Microsoft Edge 98.0.1108.56
PS D:\FlutterProjects\Firebase2_demo\firebaseapp> flutter install
Installing app.apk to Android SDK built for x86...
Uninstalling old version...
Installing build\app\outputs\flutter-apk\app.apk... 2,482ms
PS D:\FlutterProjects\Firebase2_demo\firebaseapp>
4:02
Search apps
Calendar
Camera
Clock
Contacts
Files
firebasea...
Gallery
Maps
Messaging
Phone
Search
Settings
WebView ...




```
splits
{
            // Configures multiple APKs based on ABI.
      abi {
            // Enables building multiple APKs per ABI.
            enable true
            // By default all ABIs are included, so use reset() and include to specify that we only
      // want APKs for x86, armeabi-v7a, and mips.          reset()
            // Specifies a list of ABIs that Gradle should create APKs for.
      include "x86", "x86_64", "armeabi-v7a", "arm64-v8a"
            // Specifies that we want to also generate a universal APK that includes all ABIs.
      universalApk true
          }
       }
```

Conclusion: Thus we have successfully able to test and deploy production ready Flutter App on Android.