

# Lesson 12: App Testing and Publishing

- Testing and Feedback Your App..... 12-1**
  - Setting up a Test Environment..... 12-5
  - Usability Testing by Participants..... 12-5
  - Starting Your Test Session ..... 12-6
  - Analyzing Your Test..... 12-7
  - Firebase Test Lab ..... 12-7
- Publishing Android App on Google Play Store..... 12-7**
  - Preparing your app for release..... 12-8
  - Publish App on Google Play Store ..... 12-13

## Testing and Feedback Your App

After completing all the steps of designing an Android app which satisfies users' needs and businesses with complete content and ease to access and use, we come to the important step which is testing and evaluating your application.

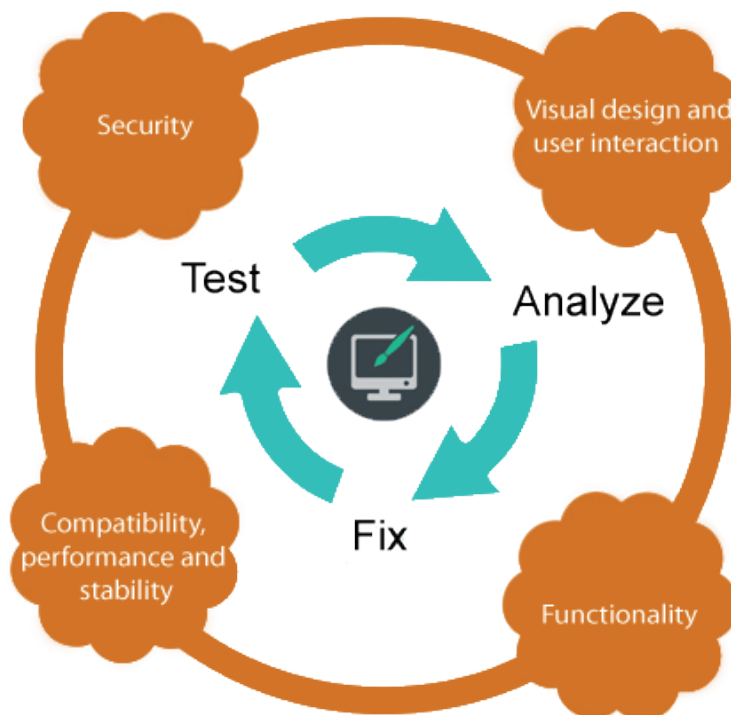
Testing the released version of your application helps ensure that your application runs properly under realistic device and network conditions. Ideally, you should test your application on at least one handset-sized device and one tablet-sized device to verify that your user interface elements are sized correctly and that your application's performance and battery efficiency are acceptable.

Android users expect high-quality apps. App quality directly influences the long-term success of an app—in terms of installs, users' ratings and reviews, engagement, and user retention.

You should assess the core aspects of quality in your app, through a compact set of quality criteria and associated tests. All Android apps should meet these criteria.

Before publishing your apps, test them against these criteria to ensure that they function well on many devices, meet Android standards for navigation and design, and are prepared for promotional opportunities in the Google Play store.

Android apps must meet the following essential quality criteria:



### 1- Visual design and user interaction

These criteria ensure that your app provides standard visual design and interaction patterns where appropriate, for a consistent and intuitive user experience as follows:

- a- The app does not redefine the expected function of a system icon (such as the Back button).
- b- The app does not replace a system icon with a completely different icon if it triggers the standard user interface behavior.
- c- If the app provides a customized version of a standard system icon, the icon strongly resembles the system icon and triggers the standard system behavior.



- d- The app does not redefine or misuse Android UI (user interface) patterns, and this means that icons or behaviors could be misleading or confusing to users.
- e- Consistent navigation is an essential component of the overall user experience. Nothing frustrates users more than a basic navigation that behaves inconsistently and in unexpected ways.

The Up button is used to navigate within an app based on the hierarchical relationships between screens. For instance, if screen A displays a list of items and selecting an item leads to screen B (which presents that item in more details), then screen B should offer an Up button that returns to screen A.

If a screen is the topmost one in an app (that is, the app's home), it should not present an Up button.



The system **Back** button is used to navigate, in reverse chronological order, through the history of screens the user has recently worked with. It is generally based on the temporal relationships between screens, rather than the app's hierarchy.

When the previously viewed screen is also the hierarchical parent of the current screen, pressing the **Back** button will have the same result as pressing an Up button which is a common occurrence. However, unlike the Up button, which ensures that the user remains within your app, the **Back** button can take the user back to the Home screen, or even to a different app.

All workflow diagrams are dismissible using the **Back** button and pressing the **Home** button at any point navigates to the Home screen of the device.

f- The app uses notifications only to indicate a change in context related to the user personally (such as an incoming message), or exposes information/controls related to an ongoing event (such as music playback or a phone call).

## 2- Functionality

These criteria ensure that your app provides the expected functional behavior.

a- The app requests only the absolute minimum permissions that it needs to support core functionality.

b- The app does not seek permissions to access sensitive data (such as Contacts or the System Log) or services that can cost the user money (such as the Dialer or SMS), unless related to a core capability of the app.

c- The app functions normally when installed on an SD card.

d- Audio does not play when the screen is off, unless this is a core feature (for example, the app is a music player).

e- Audio does not play behind the lock screen, unless this is a core feature.

f- Audio does not play on the home screen or over another app, unless this is a core feature.

g- Audio resumes when the app returns to the foreground, or there's what indicates to the user that playback is in a paused state.

h- The app supports both landscape and portrait orientations (if possible). Minor changes in content or views are acceptable.

i- The app uses the whole screen in both orientations and does not letterbox to account for orientation changes.

j- The app correctly handles rapid transitions between display orientations without rendering problems.

k- The app should not leave any service running when the app is in the background, unless related to a core capability of the app.

*For example*, the app should not leave services running to maintain a network connection for notifications, to maintain a Bluetooth connection, or to keep the GPS powered-on.

l- When the app resumes from the recent app switcher, the app returns the user to the exact state in which it was when last used.

m- When the app resumes after the device wakes from a state of sleep (locked), the app returns the user to the exact state in which it was last used.

n- When the app is re-launched from Home or All Apps, the app restores the app state as closely as possible to the previous state.

o- When pressing the Back key, the app gives the user the option of saving any app or user state that would otherwise be lost on back-navigation.

### 3- Compatibility, Performance and Stability

These criteria ensure that apps provide the compatibility, performance, stability, and responsiveness expected by users.

a- The app does not crash, force close, freeze, or otherwise function abnormally on any targeted device.

b- The app loads quickly or provides onscreen feedback to the user (a progress indicator or similar cue) if the app takes longer than two seconds to load.

c- The app runs on the latest public version of the Android platform without crashing or losing core function.

d- The app targets the latest SDK by setting the target SDK value in order to minimize the use of any platform-provided compatibility fallbacks.

e- The app is built with the latest SDK by setting the compile SDK value.

f- Music and video playback is smooth which means there are no crackle, stutter, or other artifacts, during normal app use and load.

g- The app displays graphics, text, images, and other UI elements without noticeable distortion, blurring, or pixelation. The app provides high-quality graphics for all targeted screen sizes and form factors.

h- The app displays text and text blocks in an acceptable manner. Make sure that there is sufficient spacing between texts and surrounding elements.

### 4- Security

These criteria ensure that apps handle user data and personal information safely:

a- All private data is stored in the app's internal storage.

b- All data from external storage is verified before being accessed.

c- All intents and broadcasts follow best secure practices. Intents that contain data and payload are verified before use.

d- No personal or sensitive user data is logged to the system or app-specific log.

e- All network traffic is sent over SSL.

- f- All libraries, SDKs, and dependencies are up to date.
- g- JavaScript is disabled in all Web Views (unless required).
- h- The app does not dynamically load code from outside the app.

## Setting up a Test Environment

To assess the quality of your app, you need to set up a suitable hardware or emulator environment for testing.

The ideal testing environment would include a small number of actual hardware devices that represent key form factors and hardware/software combinations currently available to consumers. It's not necessary to test the app on every device that's on the market — rather, you should focus on a small number of representative devices, even using one or two devices per form factor.

If you are not able to obtain actual hardware devices for testing, you should set up emulated devices (AVDs) to represent the most common form factors and hardware/software combinations.

To go beyond basic testing, you can add more devices, more form factors, or new hardware/software combinations to your testing environment. You can also increase the number or complexity of tests and quality criteria.

## Usability Testing by Participants

Usability tests identify areas where users struggle with an app and thus help you make recommendations for improvement. The goal is to better understand how real users interact with your Android app and to improve the product based on the results. The primary purpose of a usability test is to improve a design. In a typical usability test, real users try to accomplish typical goals, or tasks, with a product under controlled conditions. UX designers, UI designers, and development team members watch, listen, collect data, and take notes. Since usability testing employs real customers accomplishing real tasks, it can provide objective performance data, such as time on task, error-rate, and task success. There is also no substitute for watching users struggle with or have great success in completing a task when using an app. This observation helps designers and developers gain empathy with users and encourage them to think of alternative designs that better support tasks and workflow.

You should check the usability test by participants before publishing your app on Google Play store or Apple store because your purpose is to gain users' satisfaction.

If these users start using your app and find something that does not work or is not designed properly, your app will get negative reviews level on Google Play store or Apple store. If this happens, you will then need to put a lot of effort to change this feedback and sometimes negative reviews may lead to the end of a project because other users would prefer to use other apps with positive reviews.



Your research participants must be able to represent your target group or end users; otherwise, your results will not translate into something you can use.

To recruit participants that will represent real users for your app later, you have to take into consideration many factors when making this choice.

In most projects, you need to consider the age group of your participants, their geographical location, and if there is any specific type of experience they might have that may be suitable.

## Starting Your Test Session

When you start your test session, take into consideration the following steps:  
Welcome your participant and make them comfortable.

- Use a script to help you remember what you need to do and say.
- The participants should read the task scenarios and begin working on the scenario step by step.
- The participants should write comments, errors and completion (success or failure) on each task.
- The participants should register the time needed to complete each task and the total time to achieve the full scenario.
- The participants should be asked to give their opinion about the way they used to find items or navigation tools and if they were satisfied with the navigation method used to move from one task to the next.



You should ask participants after completing the task scenario the following questions:

- How did you find the app workflow?
- How did you find yourself with this type of app? Were you familiar with it?
- What is your opinion about the theme used?
- Did you find what you need easily?
- If this app was your app, what would you add or remove?
- Did you find a guide or any other way to tell you how to use this app?
- Are you satisfied to use this app for the purpose for which it is created?
- Do you still remember the purpose of the app? This question is asked to make sure that participants are serious in their testing tasks.

## Analyzing Your Test

Analyzing is a three-step process:

1. Identify exactly what you observed
2. Identify the causes of any problems
3. Determine Solutions

If you don't want or don't have enough time for this test step, you may delegate the testing step to a company that provides this service. This way, you can get a detailed report about your app and how you may improve the performance and suggest solutions for problems if there are any. Google Firebase provides a test lab service.

## Firebase Test Lab

Firebase Test Lab is a cloud-based app-testing infrastructure. With one operation, you can test your Android app across a wide variety of devices and device configurations as well as see the results—including logs, videos, and screenshots—in the Firebase console.

For more information about the Firebase test lab, check the following link:

<https://firebase.google.com/docs/test-lab>

## Publishing Android App on Google Play Store

After you complete testing, analyzing and fixing your app, it is the time to publish it on Google Play Store.

Publishing is the general process that makes your Android applications available to users.

When you publish an Android application you perform two main tasks:

- You prepare the application for release. During the preparation step you build a release version of your application, which users can download and install on their Android-powered devices.
- You release the application to users. During the release step you publicize, sell, and distribute the release version of your application to users.



## Preparing your app for release

Preparing your application for release is a multi-step process that involves the following tasks:

### 1- Configuring your application for release.

You need to remove the Log methods such as `Log.v()`, `Log.d()`, `Log.i()`, `Log.w()`, and `Log.e()` which are used by the Android developer in the MainActivity file to make a troubleshooting and know how the app runs its code.

You should also provide values for the:

`android:versionCode` and `android:versionName` attributes which are located in the **AndroidManifest.xml** file as illustrated in the code below:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/
android"
    package="com.androidatc.lesson11_googlemap"
    android:versionCode="1"
    android:versionName="1.0">
```

Also, these attributes values must be the same in the **build.gradle** (Module: your\_app\_name) file as illustrated in the figure below. In the future when you want to release a new version of your app, you must increase the `versionCode` by 1 (Integer number). This means the `versionCode` should be 2 for the next version or updated of your app.



```
7  android {
8      compileSdk 30
9
10     defaultConfig {
11         applicationId "com.androidatc.lesson11_googlemap"
12         minSdk 29
13         targetSdk 30
14         versionCode 1
15         versionName "1.0"
```

If you in the future make a new version of your app or added an update to your app, you must change the version code of your app from 1 to 2 or any another integer number. Then, upload your app update to the Google Play store. Your app users who were already using an old version of your app will receive a notification message telling them that there is a new version of the app and it is ready for download.

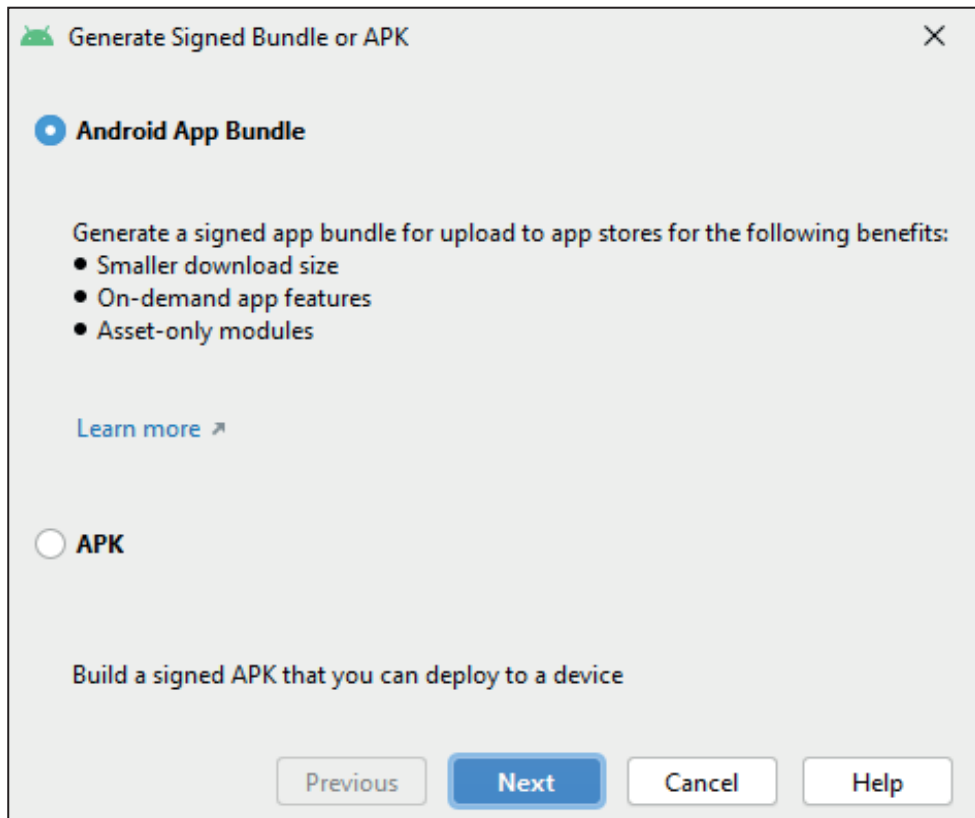
### 2- Building and signing a release version of your application.

The files which will upload to the Google Play store is a signed file that has APK extension. To create this file using Android Studio, follow the steps below:

a- Open Android Studio, from the menu bar, click: **Build → Build Bundle (s) / APK (s) → Build APK(s)**.

b- Click **Build** → **Generate Signed Bundle / APK...**

You should get the following dialog box. Select: **Android App Bundle**, then click **Next**.



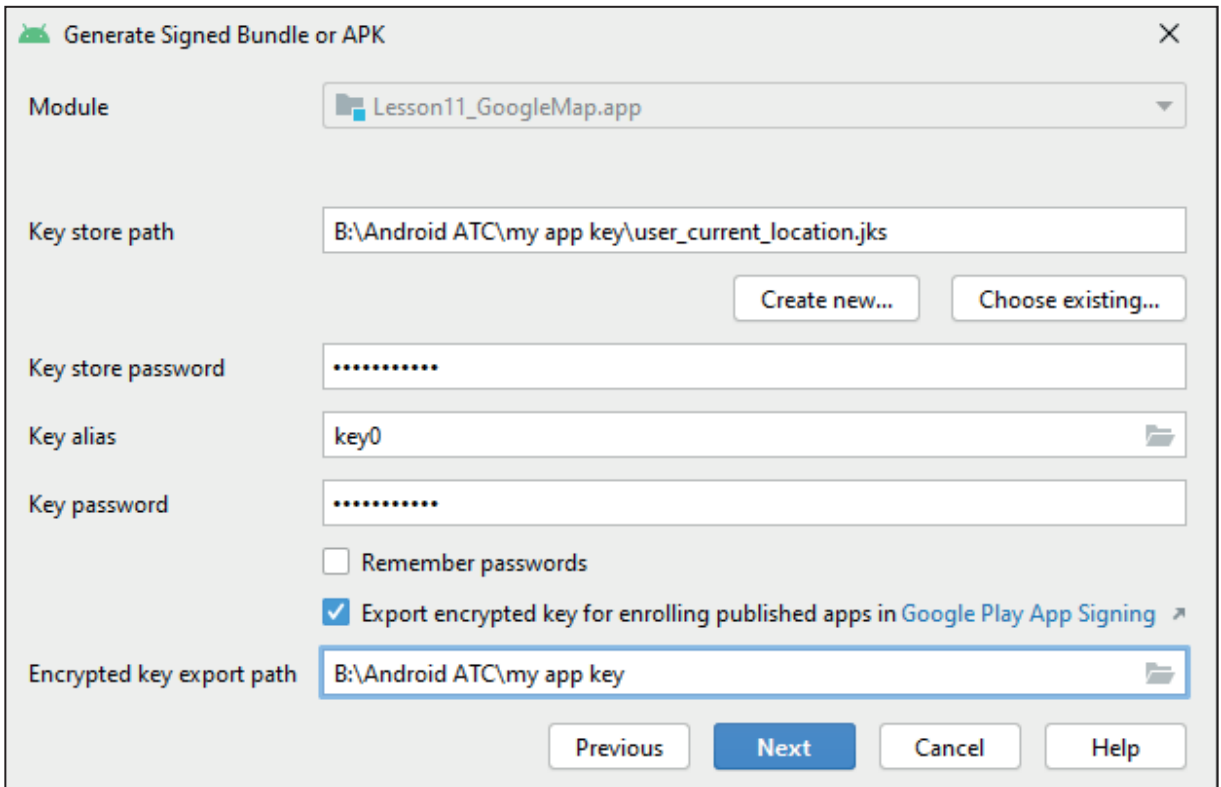
c- Android devices only allow installing apps that are signed with a digital certificate. This certificate is self-signed and is used to detect the author of the app you need to give your Android app a digital signature. This app signature approves that you are the owner of this app, and only you have the permission to publish and update it on Google Play store. Use the following instructions to sign your app. Your app is signed by a private key that is retained by the developer. Signing your application is an obligatory task before releasing it to users. In this step, you must sign your app by entering the information below about your app and password which you will use later if you want to modify any of these information in the future. Click **Create new...** button.

The screenshot shows the 'Generate Signed Bundle or APK' dialog box. The 'Module' dropdown is set to 'Lesson11\_GoogleMap.app'. The 'Key store path' field is empty, with 'Create new...' and 'Choose existing...' buttons to its right. The 'Key store password', 'Key alias', and 'Key password' fields are also empty. There is an unchecked checkbox for 'Remember passwords' and a checked checkbox for 'Export encrypted key for enrolling published apps in Google Play App Signing'. The 'Encrypted key export path' is set to 'C:/Users/ATC/Desktop'. At the bottom are 'Previous', 'Next', 'Cancel', and 'Help' buttons.

d- As illustrated in the dialog box below, select the key store path and the key file name. Enter your information, then click **OK**.

The screenshot shows the 'New Key Store' dialog box. The 'Key store path' is 'B:\Android ATC\my app key\user\_current\_location.jks'. The 'Password' and 'Confirm' fields are filled with dots. Under the 'Key' section, the 'Alias' is 'key0', and the 'Password' and 'Confirm' fields are also filled with dots. The 'Validity (years)' is set to 25. The 'Certificate' section contains the following information: 'First and Last Name' is 'Alex Alex', 'Organizational Unit' is 'Keys', 'Organization' is 'Android ATC', 'City or Locality' is 'Toronto', 'State or Province' is 'Ontario', and 'Country Code (XX)' is 'CA'. The 'Country Code (XX)' field is highlighted with a blue border. At the bottom are 'OK' and 'Cancel' buttons.

e- As illustrated below, select the **Encrypted key export path** and click **Next**.



Generate Signed Bundle or APK

Module: Lesson11\_GoogleMap.app

Key store path: B:\Android ATC\my app key\user\_current\_location.jks  
Create new... Choose existing...

Key store password: .....

Key alias: key0

Key password: .....

☐ Remember passwords

☒ Export encrypted key for enrolling published apps in Google Play App Signing ↗

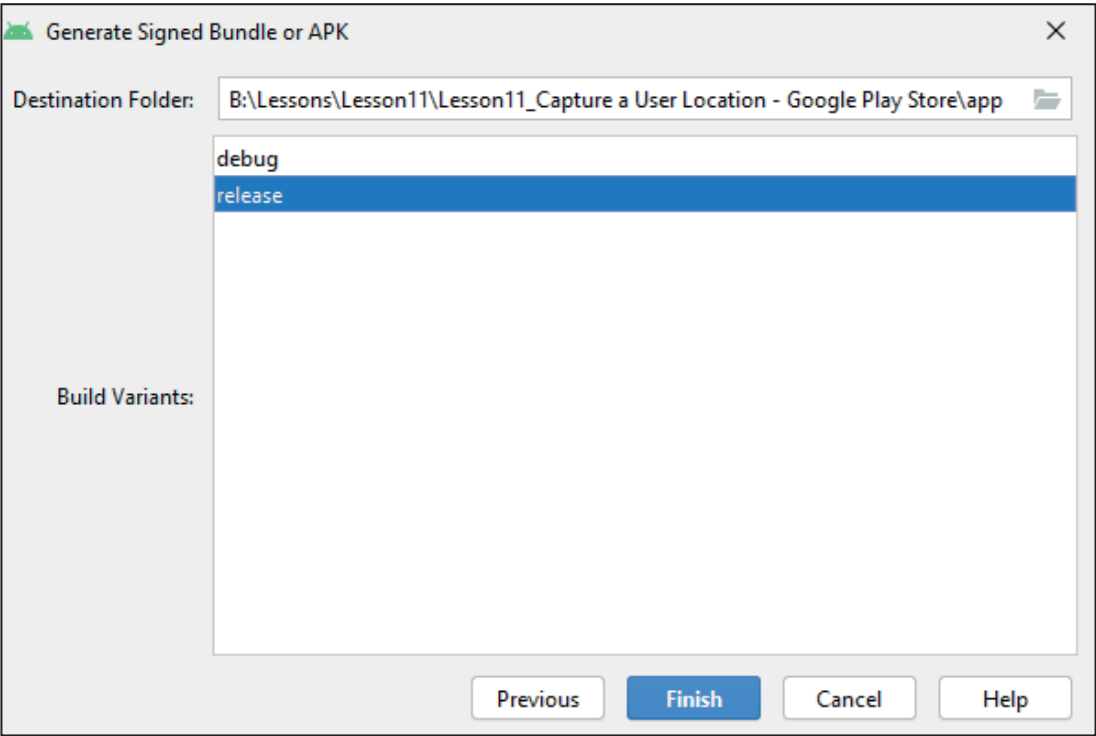
Encrypted key export path: B:\Android ATC\my app key

Previous Next Cancel Help

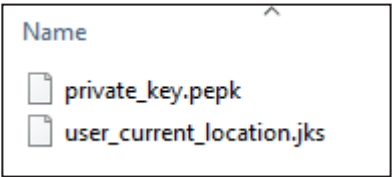
f- This step is very important because now you will create the **App bundle file** which will be uploaded later to the Google Play store. An *Android App Bundle* (file name: **app-release.aab**) is a publishing format that includes all your app's compiled code and resources (images, animations, videos, etc...) and defers from APK generation and signing to Google Play.

Google Play uses your app bundle to generate and serve optimized APKs for each device configuration, so only the code and resources that are needed for a specific device are downloaded to run your app.

In this step and as illustrated in the dialog box below, select **release**, then click **Finish**.



The following figure displays the keys which you have created for releasing your app:



Also, the app bundle file has been created by default in the same app folder (App Name → app → release) as illustrated in the figure below:

View			
PC > New Volume (B:) > Lessons > Lesson11 > Lesson11_Capture a User Location - Google Play Store > app > release			
Name	Date modified	Type	Size
app-release.aab	2021-09-05 8:19 PM	AAB File	3,513 KB

**3- Testing the release version of your application.**

Before you distribute your application, you should thoroughly test the release version on at least one target handset device and one target tablet device.

**4- Updating application resources for release.**

You need to be sure that all application resources such as multimedia files and graphics are updated and included with your application or staged on the proper production servers.

## 5- Preparing remote servers and services that your application depends on.

If your application depends on external database or multimedia servers or services, you need to be sure they are secure and the production is ready.

Now, your app bundle file is ready for publish on Google Play store.


## Publish App on Google Play Store


To publish an Android app on Google Play store, you should create a publisher account which allows you to access the tools in Google Play Developer Console and eventually publish your Android applications. The account registration fees for Google Play are a one-time fee of \$25. There is no extra charge when you want to update your existing Android app or if you want to publish other Android apps in the future.

The following steps tell you how to register for a Google Play publisher account:

1- Go to: **<https://play.google.com/apps/publish/signup>**

Login using your Gmail account, then you will get the following figure:


 **Google Play Console**

 alex@androidatc.com

### Create a new developer account

The selected Google account will own this new developer account. If you are trying to join an existing developer please request an invite from your administrator.

If you are an organization, we recommend that you do not use a personal account to set up developer accounts. You can set up Google accounts using any existing email address. [Find out more](#)

 To create an account you'll need to pay a one-off \$25 registration fee. You may be asked to verify your identity using a valid ID to complete your account registration. If we can't verify your identity, the registration fee won't be refunded.

Public developer name \*

This will be visible to users on Google Play 9 / 50

Secondary contact email address \*

We may use this to contact you in addition to the email associated with your Google account. It won't be visible to Google Play users.

Contact phone number \*

Include the + symbol, country code, and area code. We may use this to contact you, but it won't be visible to users on Google Play.

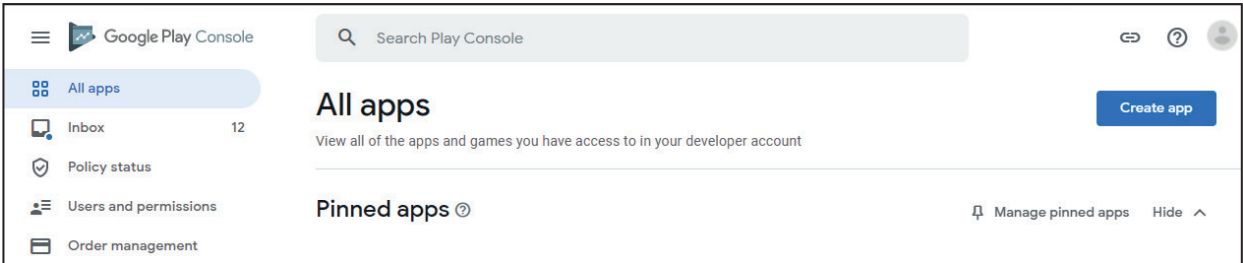
2- Scroll down this web page, in the **Developer agreement and Terms of Service** section, select **I confirm**, then click: **Create account and pay** button.

3- You will get the payment page. The Developer Registration Fee is 25 USD for a life time. If you want to continue publishing your app, you will be required to enter your credit card information. The following screenshots will give you enough knowledge about how to publish an Android app on Google Play store without the need to pay at this time being.

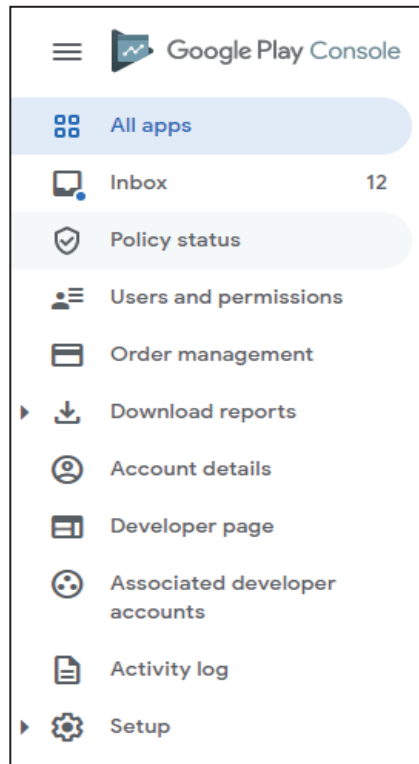
The screenshot shows a 'Complete your purchase' dialog box with a close button (X) in the top right corner. It lists the 'Developer Registration Fee' as '\$25.00'. Below this, there is a section titled 'Add credit or debit card' with a card icon. The form includes fields for 'Card number' (with a '#' symbol and icons for American Express, Discover, JCB, MasterCard, and Visa), 'MM / YY' (month/year), and 'CVC'. There is also a 'Cardholder name' field. Below the card fields is a 'Billing address' field with a location pin icon. At the bottom, there is a disclaimer: 'By continuing, you agree to the Google Payments [Terms of Service](#). The [Privacy Notice](#) describes how your data is handled.' and a blue 'Buy' button.

4-After completing the payment step, you will be asked to complete your account details. Here, you will fill out a form that verifies you about your name, email address, web site, and phone number.

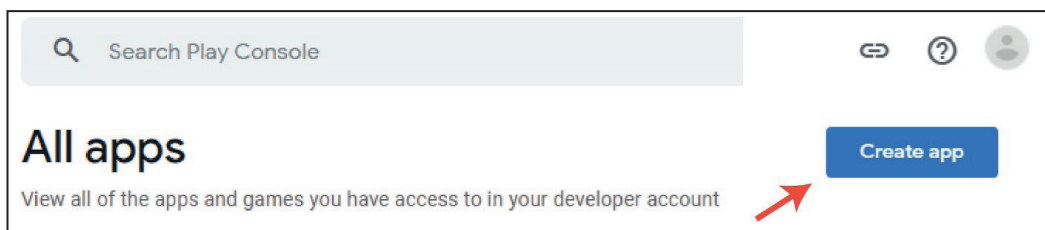
The following figure displays the Google Play store dashboard:



Also, the following figure displays the content of the Google Play console panel which exits in the left side:



5- To add your app to Google Play store, click : **Create app** button as illustrated in the following figure:



6- Then, as illustrated in the figure below, enter your app information (App name, default language, app type, and select free app), accept Google terms and conditions, then click **Create app** button.



### Create app

#### App details

App name

User's Location on Google Maps

This is how your app will appear on Google Play. It should be concise and not include price, rank, any emoji or repetitive symbols.

Default language

English (United States) – en-US

App or game

You can change this later in Store settings

☒ App


☐ Game

Free or paid

You can edit this later on the Paid app page


☒ Free

☐ Paid

 You can edit this until you publish your app. Once you've published, you can't change a free app to paid.

7- Now, you should enter more information about your app. As illustrated in the figure below, In the **Set up your app** area, click **App access**.

### 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

Hide tasks ^

#### 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 >

8- Select All functional is available without special access, then click **Save**.

← Dashboard

## App access

If parts of your app are restricted based on login credentials, memberships, location, or other forms of authentication, 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

9- Then click the **Dashboard** on the left panel. You should get the following figure which displays that one task has been completed. You should complete all the other tasks for the Set up your app.

## 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

1 of 8 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 >

#### MANAGE HOW YOUR APP IS ORGANIZED AND PRESENTED

☐ Select an app category and provide contact details >

☐ Set up your store listing >

The following setting for Ads will appear. Click **Save**, then click **Dashboard**.

## 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

10- Move to the: **MANAGE HOW YOUR APP IS ORGANIZED AND PRESENTED**, and click: **Select an app category and provide contact details**. Set the following information, then click **Save**.

## Store settings

Manage how your app is organized on Google Play, and how users can contact you

\* – Required fields

### App category

Choose an application type, category, and tags that best describe the content or main function of your app. These help users discover apps on Google Play.

App or game \*

App

Category \*

Education

Tags

[Manage tags](#)

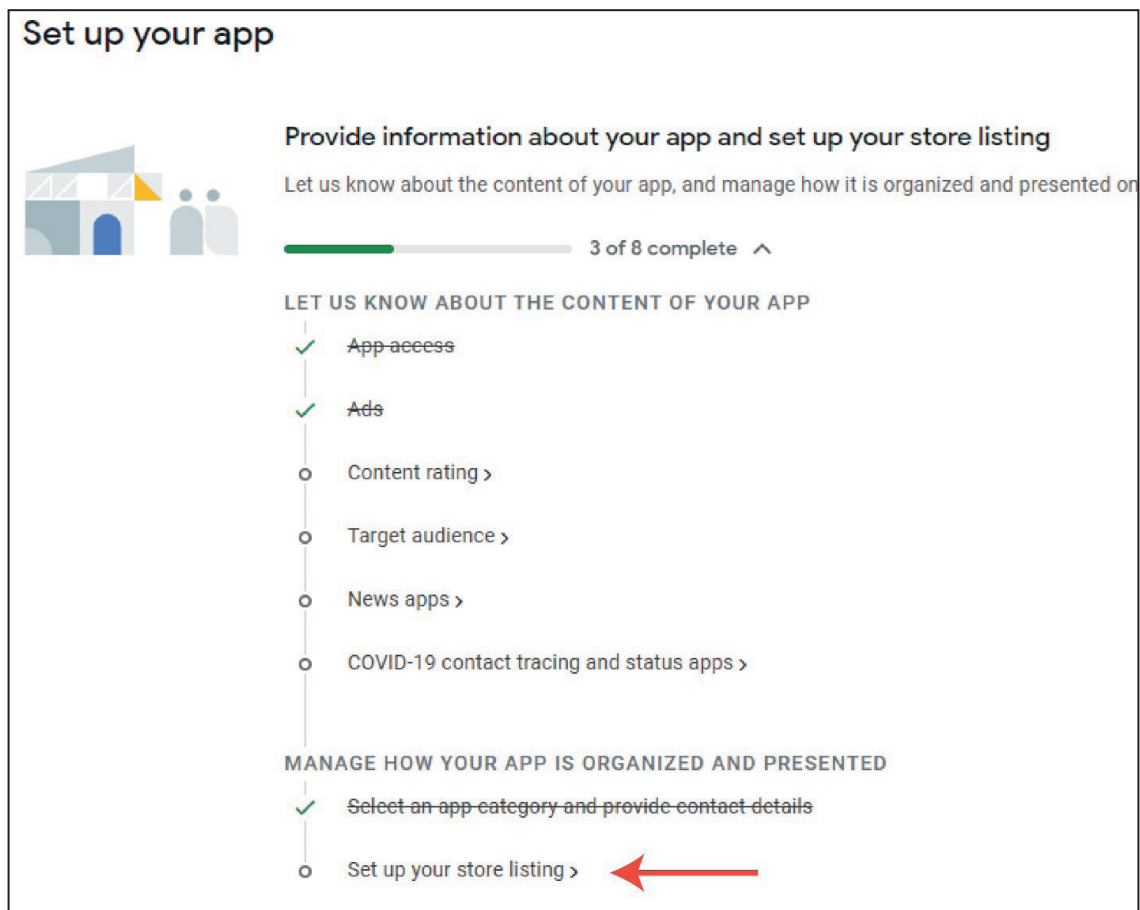
### Store listing contact details

This information is shown to users on Google Play

Email address \*

info@androidatc.com

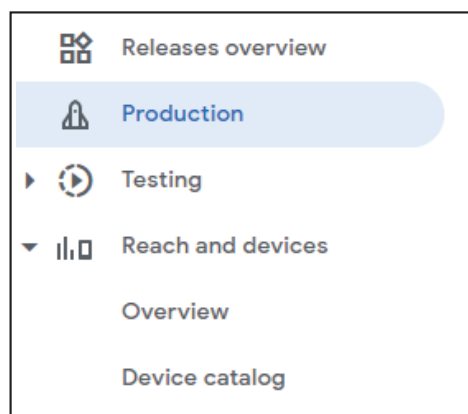
11- Click **Dashboard** on the left panel. Then, as illustrated in the following figure, click: **Set up your store listing**. This option is very important for the app release process.



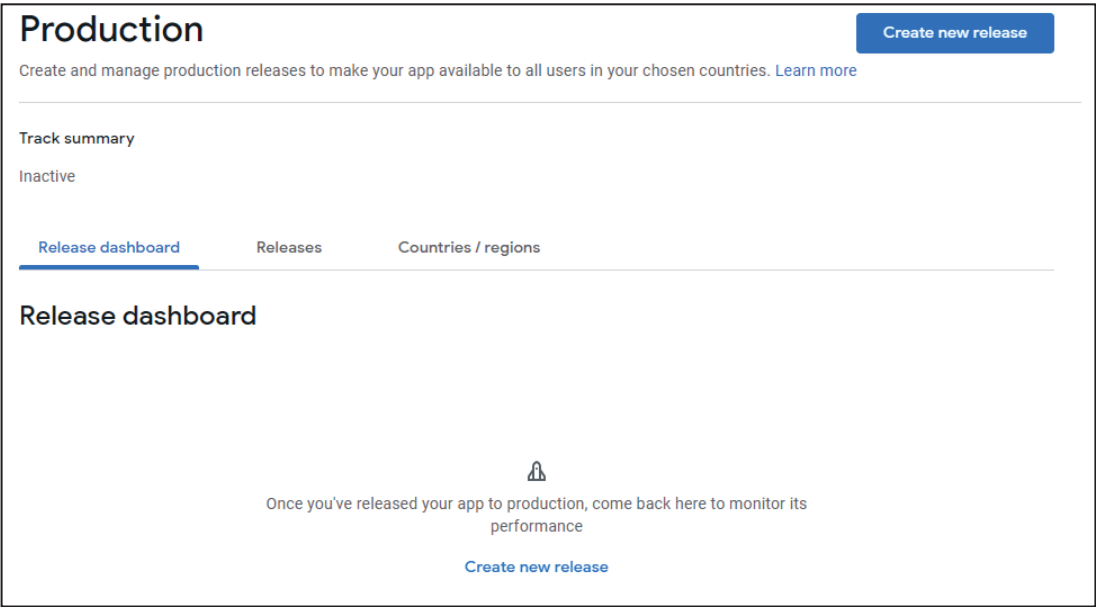
In this part, you should enter your app name, short and full descriptions about your app, app icon, screenshots about your app, and tablet screenshots. You may use Adobe Photoshop to customize your images to get the required images sizes. Then click **Save**.

Complete all the other tasks (Target audience, News apps, and COVID -19).

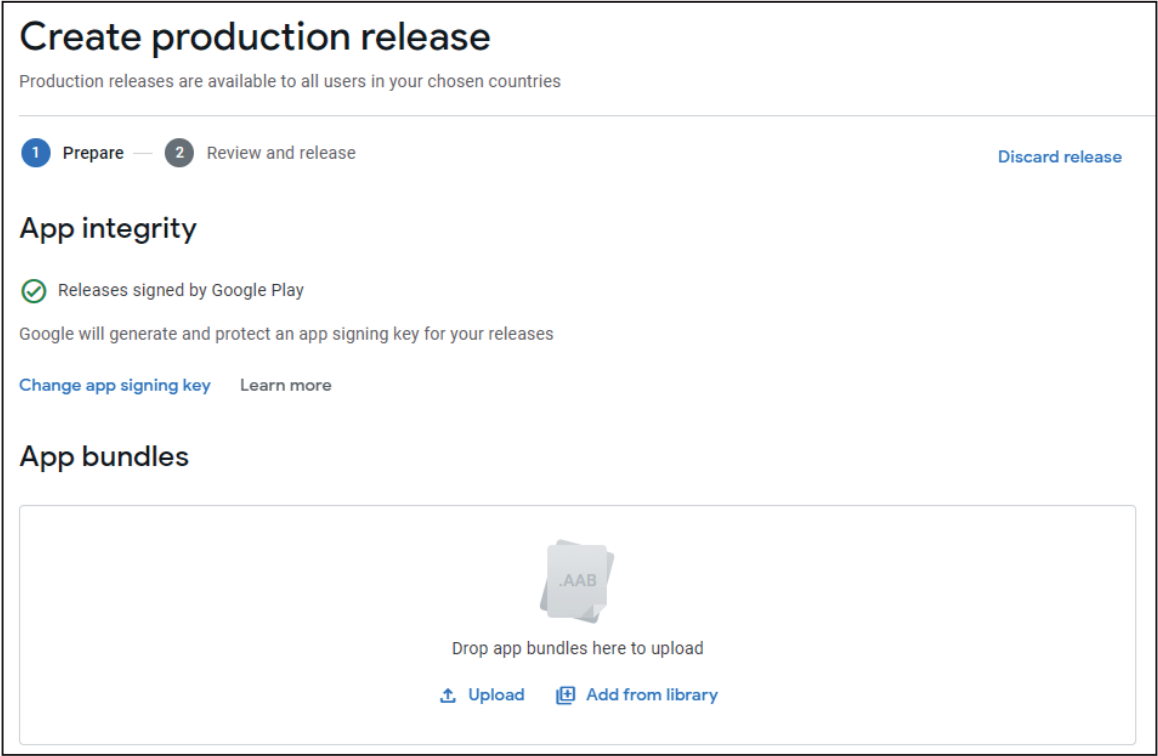
Now, click **Production** on the left panel of your web browser as illustrated in the figure below:



Click **Create new release** button.



In this step and as illustrated in the figure below, you will upload your app file: **app-release.aab** from your computer to the Google Play store.



Click **Upload** or drag and drop the file: **app-release.aab** to upload your app bundle to Google Play store. You should get the following figure which displays the version of your app.

# Create production release

Production releases are available to all users in your chosen countries

1

Prepare

2

Review and release

Discard release

## App integrity

Releases signed by Google Play

Google will generate and protect an app signing key for your releases

## App bundles

.AAB

Drop app bundles here to upload

Upload

Add from library

app-release.aab

File type	Version	API levels	Target SDK	Screen layouts	ABIs	Required features
App bundle	1 (1.0)	29+	30	4	All	2 <div><div></div></div> <div></div>

Scroll down, enter your release details and click **Save**. Then click **Review Release**

## Release details

Release name \*

Release 01

10 / 50

This is so you can identify this release, and isn't shown to users on Google Play. We've suggested a name based on the first app bundle or APK in this release, but you can edit it.

Release notes

Copy from a previous release

<en-US>

This is my first release for this app. September, 2021

</en-US>

12- Click **Production** on the left panel. As illustrated in the figure below, click the **Countries/regions** tab. Then, click **Add countries / regions**.

# Production

Create and manage production releases to make your app available to all users in your chosen countries. [Learn more](#)

---


**Track summary**

Inactive • Draft release: Release 01 • 0 countries / regions • 0 installs

[Release dashboard](#)[Releases](#)[Countries / regions](#)

---

## Countries / regions




Add the countries / regions where you want your app to be available


[Add countries / regions](#)[Learn more](#)

Select the first choice (All Countries), then click **Add Countries/regions** button.


Click **Production** on the left panel again. Click **Edit release** button, then **Review release** to check if your app is ready for release on the Google Play store. If you got any warning or error message, this means there is some information that still need to be completed. You should get the following figure which displays that your app is ready for release. Click **Start rollout to Production**.

## Create production release

 We found some problems with your release


 Prepare


—

 2 Review and release


[Discard release](#)

### Errors, warnings and messages

 1 Warning

[Show more](#) 

### New app bundles

File type	Version	API levels	Target SDK	Screen layouts	ABIs	Required features
App bundle	1 (1.0)	29+	30	4	All	2 

### Release notes

Default language - en-US

This is my first release for this app. September, 2021


Review your release before rolling it out

[Edit release](#)

[Start rollout to Production](#)

You should get the following dialog box. Click **Rollout** to release your app.

### Rollout to production?



This release will be available to all users on Google Play in your chosen countries

Cancel

[Rollout](#)

Your app needs some time to appear for public on Google Play store. Normally, this period is less than two weeks.

This was the end of lesson 12 and the end of this Android application development course. Now, you are ready to apply for the Android certified application development exam (exam code: AND-X01). For more information about this exam or exam sample, check the Android ATC web site.