Wireless Programming

COMP-304
Fall 2018

Review of Lecture 11

- Android Services
  - use the **Service** class.
  - Override **onBind** to bind an activity to a service, **onStartCommand** starts when the service is started, and **onDestroy**,when the service is stopped.
  - Declare the service in manifest file:

```
<service android:name=".MyService"
/>
```

- Use the startService() method, like this:

startService(**new Intent(getBaseContext(), MyService.class));**

- Use the stopService() method to stop the service:

stopService(**new Intent(getBaseContext(), MyService.class));**

Review of Lecture 11

- Performing Long-Running Tasks in a Service:
- Create an **inner class that extends the AsyncTask** class
- Override:
  - **doInBackground**()
  - This method is executed in the background thread and is where you put your long-running code.
  - call the publishProgress() method to report progress
  - This invokes **onProgressUpdate**() method

- Getting the results:
  - **onPostExecute()** — This method is invoked in the UI thread and is called when the doInBackground() method has finished execution

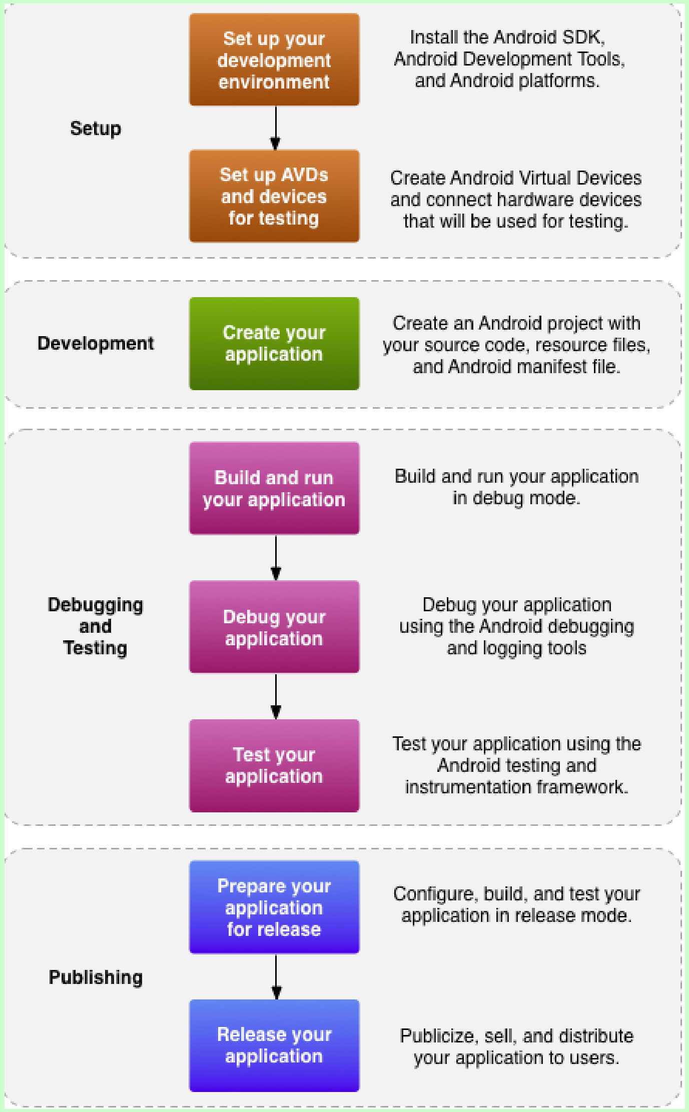
```
try {
        new DoBackgroundTask().execute(
        new
        URL("http://www.google.com/somefil
        es.pdf"),
        new
        URL("http://www.learn2develop.net/s
        omefiles.pdf"));
        } catch (MalformedURLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
```

Publishing Android Apps

Objectives:

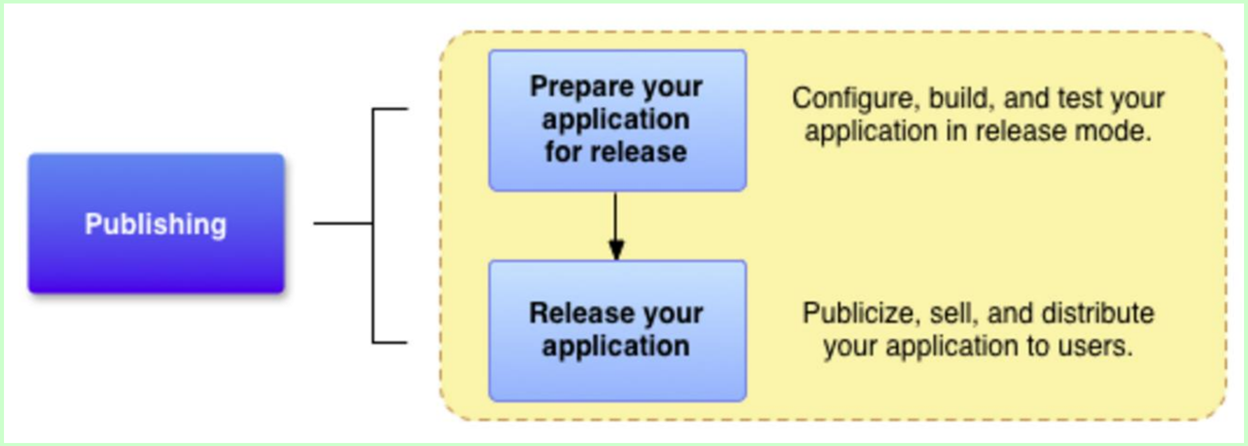
- ❑ Explain Android **deployment** features.
- ❑ **Digitally sign** and deploy APK files.
- ❑ Explain **publishing process**.

The development process for Android Apps


Setup
Set up your development environment
Install the Android SDK, Android Development Tools, and Android platforms.
Set up AVDs and devices for testing
Create Android Virtual Devices and connect hardware devices that will be used for testing.
Development
Create your application
Create an Android project with your source code, resource files, and Android manifest file.
Debugging and Testing
Build and run your application
Build and run your application in debug mode.
Debug your application
Debug your application using the Android debugging and logging tools
Test your application
Test your application using the Android testing and instrumentation framework.
Publishing
Prepare your application for release
Configure, build, and test your application in release mode.
Release your application
Publicize, sell, and distribute your application to users.

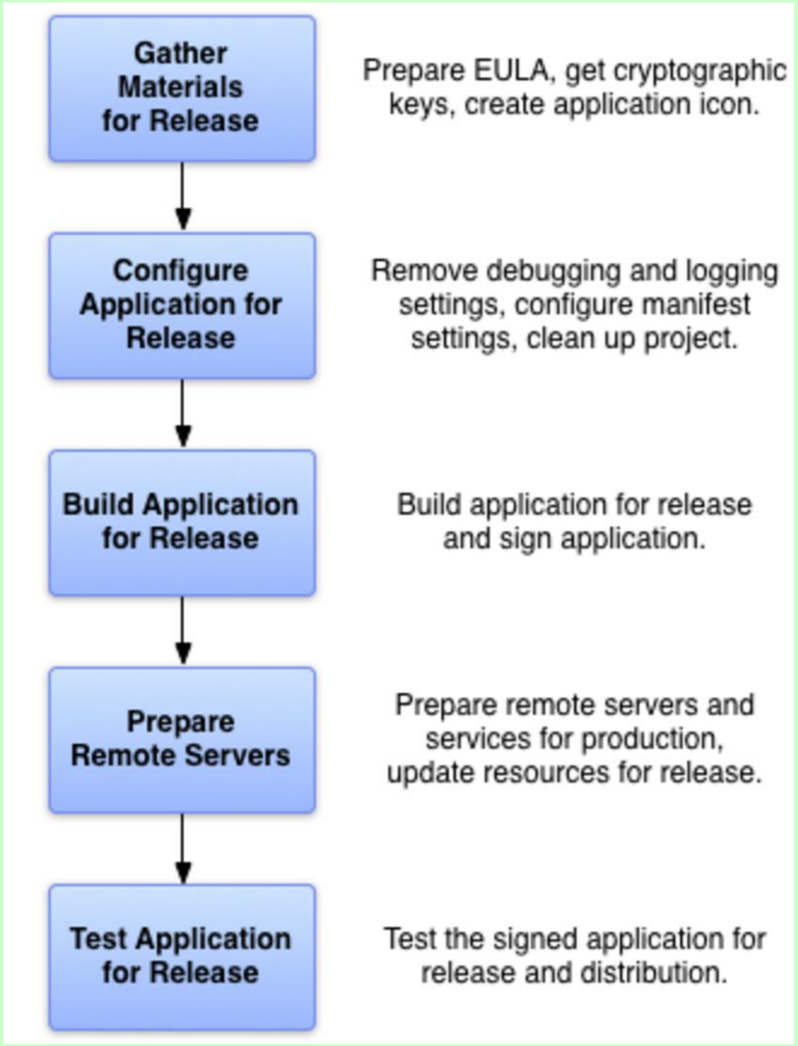
Publishing Android Apps

- Publishing an Android application includes two main tasks:
  - **Prepare** the application for release
    - **build a release version** of your application, which users can download and install on their Android-powered devices.
  - **Release** the application to users
    - **publicize, sell, and distribute** the release version of your application to users.


Publishing
Prepare your application for release
Configure, build, and test your application in release mode.
Release your application
Publicize, sell, and distribute your application to users.

Preparing App for Release

- Five main tasks:


Gather Materials for Release
Prepare EULA, get cryptographic keys, create application icon.
Configure Application for Release
Remove debugging and logging settings, configure manifest settings, clean up project.
Build Application for Release
Build application for release and sign application.
Prepare Remote Servers
Prepare remote servers and services for production, update resources for release.
Test Application for Release
Test the signed application for release and distribution.

Versioning Your Application

- If you are planning to publish your application on the Google Play, the AndroidManifest.xml file must have the following attributes:
  - android:**versionCode** (within the <manifest> element) - An integer value that represents the version of the application code, relative to other versions.
  - android:**versionName** (within the <manifest> element) - string value that represents the release version of the application code, as it should be shown to users.
  - android:**icon** (within the <application> element)
  - android:**label** (within the <application> element)

https://developer.android.com/studio/publish/versioning.html

Versioning Your Application

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.inika.transittest" >
            android:versionCode=”1”
            android:versionName=”1.0” >
    <uses-sdk
        android:minSdkVersion="12"
        android:targetSdkVersion="15" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

API Level requirements

- Add a **<uses-sdk>** element in the application's manifest, with one or more of these attributes:
  - **android:minSdkVersion** - the **minimum version** of the Android platform on which the application will run, specified by the platform's API Level identifier.
  - **android:targetSdkVersion** - the API Level on which the application is designed to run.
  - **android:maxSdkVersion** - the **maximum version** of the Android platform on which the application is designed to run, specified by the platform's API Level identifier.

API Level requirements

- To define the version information for your app, set values for the version settings in the **Gradle build files**.
  - These values are then merged into your app's manifest file during the build process.
- If your app defines the app version directly in the <manifest> element, the **version values in the Gradle build file will override the settings in the manifest**.
- Remove these attributes from the <manifest> element and define your version settings in the Gradle build files instead.

Signing Your Apps

- An Android Package (**APK**) file is the **file format used for installing software on the Android operating system**.
- Android requires that all **APKs be digitally signed with a certificate before they can be installed**.
- You can sign an app in debug or release mode:
  - You sign your app in debug mode during development and in release mode when you are ready to distribute your app.
  - The Android SDK generates a certificate to sign apps in debug mode.
  - To sign apps in **release mode**, you need to **generate your own certificate**.

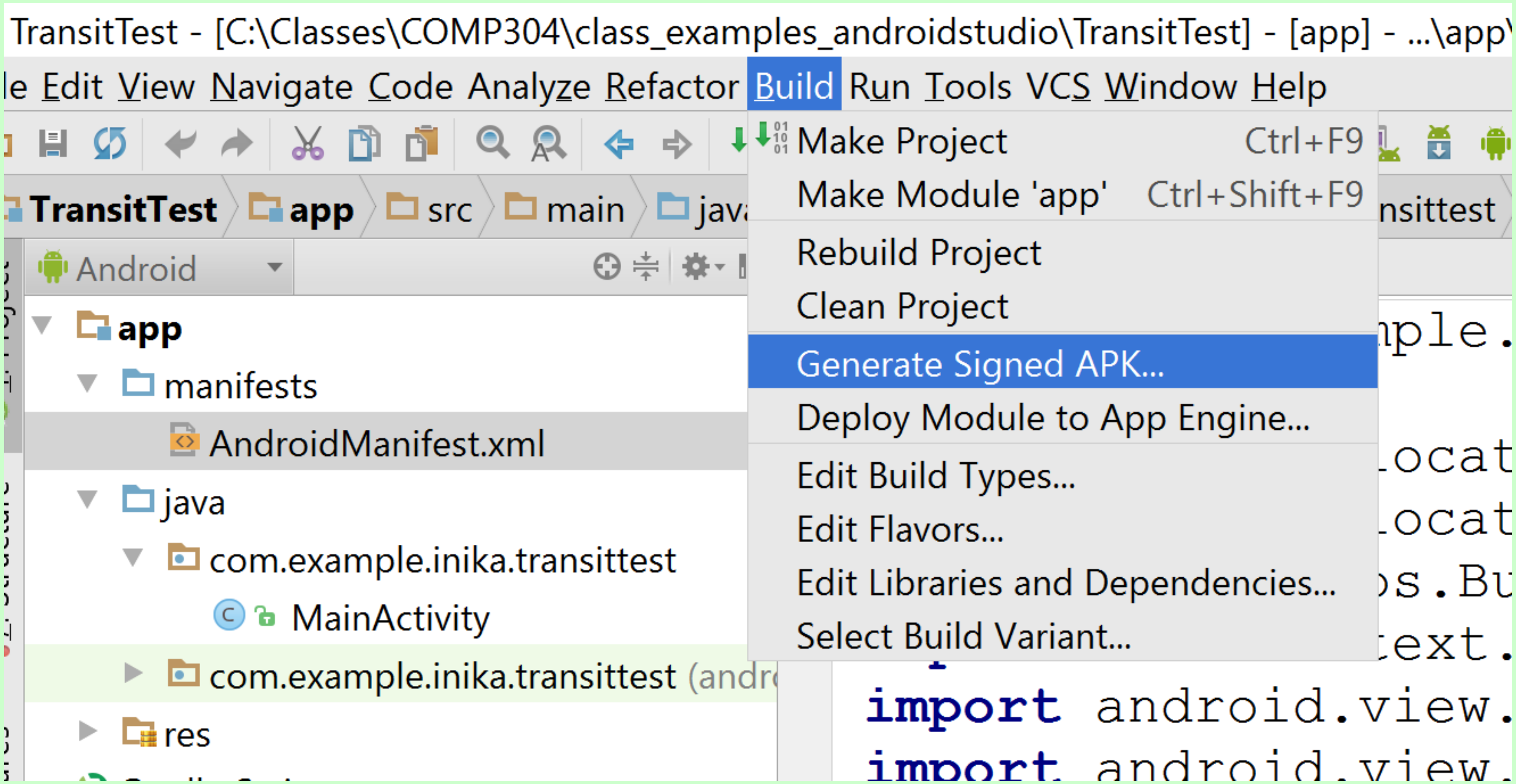
https://developer.android.com/studio/publish/app-signing.html

Digitally Signing Your Android Applications

- In release mode, you sign your app with your own certificate:
  - Create a **keystore** - a binary file that contains a set of private keys.
    - You must keep your keystore in a safe and secure place.
  - Create a **private key** - represents the entity to be identified with the app, such as a person/developer or a company.
  - Add the signing configuration to the build file for the app module – Android Studio does that.

Digitally Signing Your Android Applications

- In Android Studio, we can choose Generate Signed APK... option from Build menu

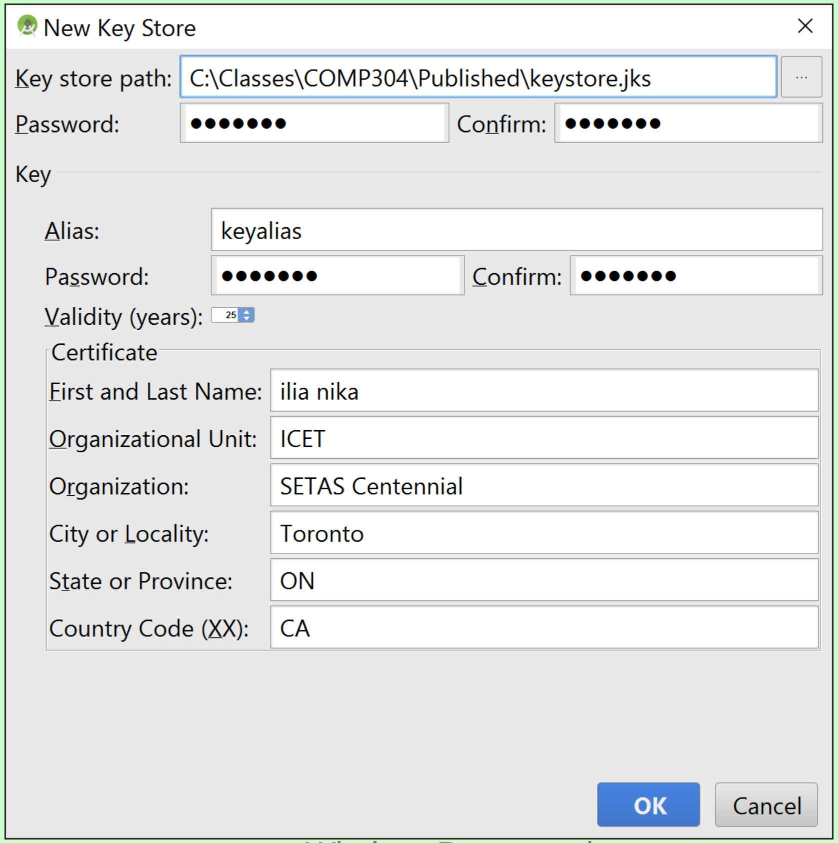


TransitTest - [C:\Classes\COMP304\class_examples_androidstudio\TransitTest] - [app] - ...\app\

le Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
TransitTest app src main java
Android
app
manifests
AndroidManifest.xml
java
com.example.inika.transittest
MainActivity
com.example.inika.transittest (andr
res
Make Project Ctrl+F9
Make Module 'app' Ctrl+Shift+F9
Rebuild Project
Clean Project
Generate Signed APK...
Deploy Module to App Engine...
Edit Build Types...
Edit Flavors...
Edit Libraries and Dependencies...
Select Build Variant...

Creating a new Keystore

- On the Generate Signed APK Wizard window, click Create new to create a new keystore - provide **keystore path, password** to protect the private key, a **validity period** for the key, minimum **25 years**, **key alias** and other information:



Key store path:	C:\Classes\COMP304\Published\keystore.jks		
Password:	●●●●●●●	Confirm:	●●●●●●●

Key

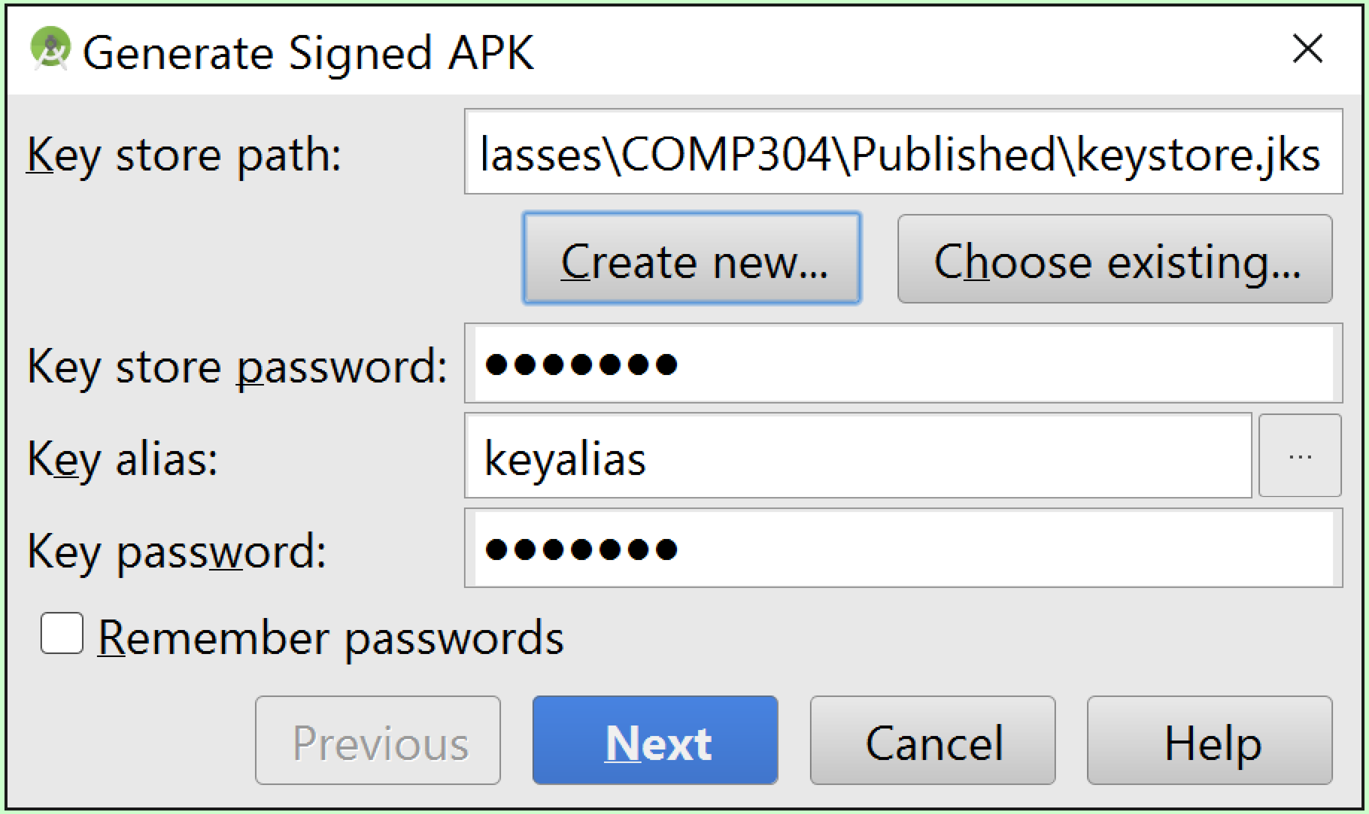
Alias:	keyalias		
Password:	●●●●●●●	Confirm:	●●●●●●●
Validity (years):	25		

Certificate

First and Last Name:	ilia nika
Organizational Unit:	ICET
Organization:	SETAS Centennial
City or Locality:	Toronto
State or Province:	ON
Country Code (XX):	CA

Creating a new Keystore

- Enter a path to save your new keystore
- Enter a password to protect the keystore



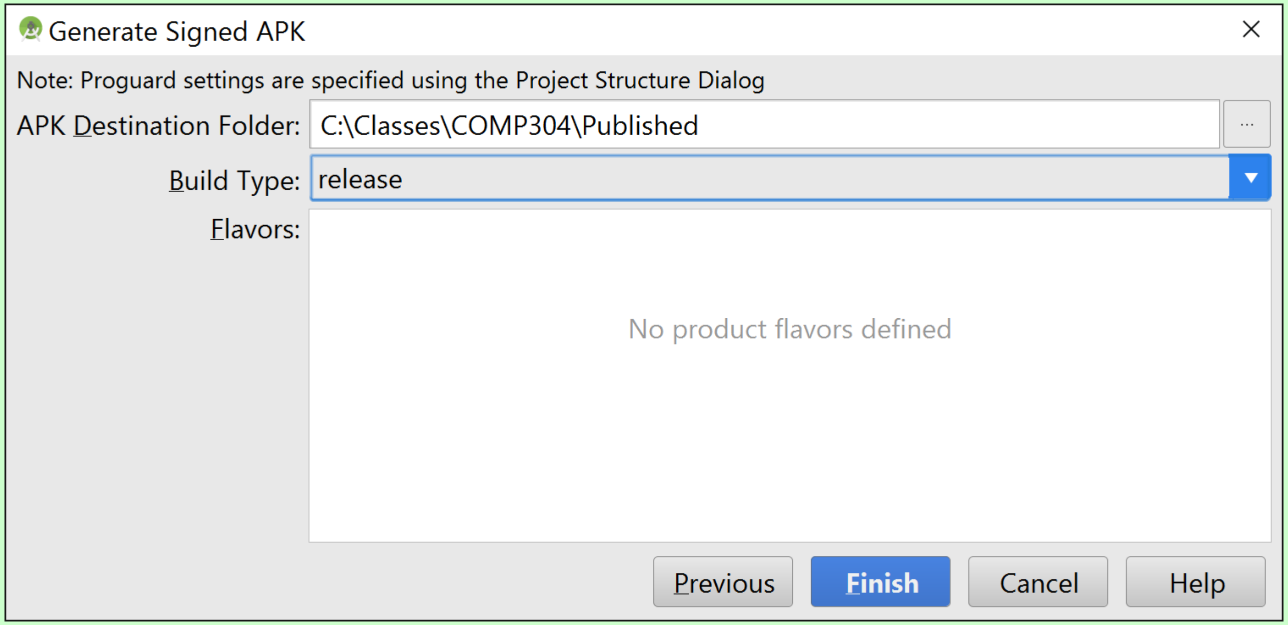
Generate Signed APK

Key store path:	lasses\COMP304\Published\keystore.jks
Key store password:	••••••••
Key alias:	keyalias
Key password:	••••••••

☐ Remember passwords

Creating APK file

- ❑ Store the destination APK file



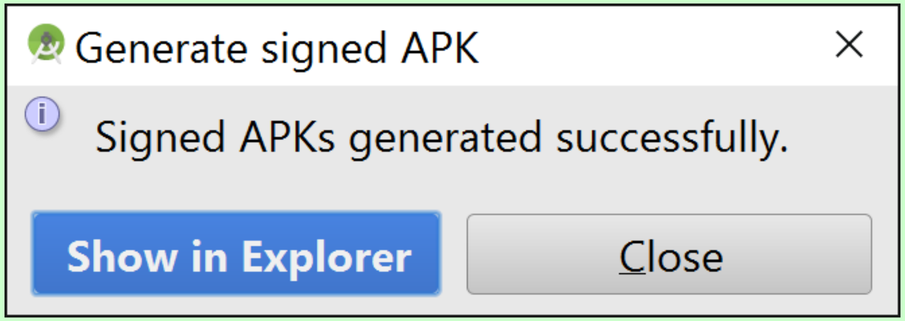
Generate Signed APK

Note: Proguard settings are specified using the Project Structure Dialog

APK Destination Folder: C:\Classes\COMP304\Published

Build Type: release

Flavors: No product flavors defined



Generate signed APK

Signed APKs generated successfully.

Automatically Signing Your App

- In Android Studio, you can configure your project to sign your release APK automatically during the build process:
  - On the project browser, right click on your app and select **Open Module Settings**.
  - On the *Project Structure* window, select your app's module under *Modules*.
  - Click on the **Signing** tab.
  - Select your keystore file, enter a name for this signing configuration (as you may create more than one), and enter the required information.

DEPLOYING APK FILES

- ❑ Use one of the following three methods to deploy an Android app to an Android device:
  1. Deploying manually using the adb.exe tool
  2. Hosting the application on a web server
  3. Publishing through the Google Play

Using adb.exe

- You can deploy an Android app to emulators and devices using the adb.exe (Android Debug Bridge) tool (located in the platform-tools folder of the Android SDK).
- To install the application to an emulator/device (assuming the emulator is currently up and running or a device is currently connected), use the following command:

```
adb install “C:\Temp\app-release.apk”
```

- You can view the devices currently connected to your computer by using the devices option with adb:

```
adb devices
```

Using a Web Server

- Copy the signed UITest.apk file to your web app root directory
- Create a new HTML file named index.html with the following content:

```
<html>
<title>UITest application</title>
<body>
Download the Where Am I application <a
    href="UITest.apk">here</a>
</body>
</html>
```

- You may need to register a new MIME type for the APK file. The MIME type for the .apk extension is application/vnd.android.package-archive

Using a Web Server

- For installation over a web server, you need to configure your Android emulator/device to accept applications from non-Market sources.
  - In the Settings application, click the Security item and scroll to the bottom of the screen.
  - Check the “Unknown sources” item.
  - You will be prompted with a warning message.
  - Click OK.
  - Checking this item will allow the emulator/device to install applications from other non-Market sources (such as from a web server).

Publishing on the Google Play

- A better way is to host your application on the Google Play, a Google-hosted service that makes it very easy for users to discover and download (i.e., purchase) applications for their Android devices.
- Users simply need to launch the Google Play application on their Android device in order to discover a wide range of applications that they can install on their devices.

Publishing Checklist

1. Understand the Publishing Process
2. Understand Google Play Policies
3. Test for Core App Quality
4. Determine Content Rating
5. Determine Country Distribution
6. Confirm Overall Size
7. Confirm Platform and Screen Ranges
8. Decide Free or Priced
9. Use In-app Billing
10. Set Prices for your Products
11. Start Localization
12. Prepare Promotional Graphics, Screenshots, and Videos
13. Build the Release-ready APK
14. Plan a Beta Release
15. Complete the Store Listing
16. Use Google Play Badges and Links
17. Final Checks and Publishing
18. Support Users after Launch

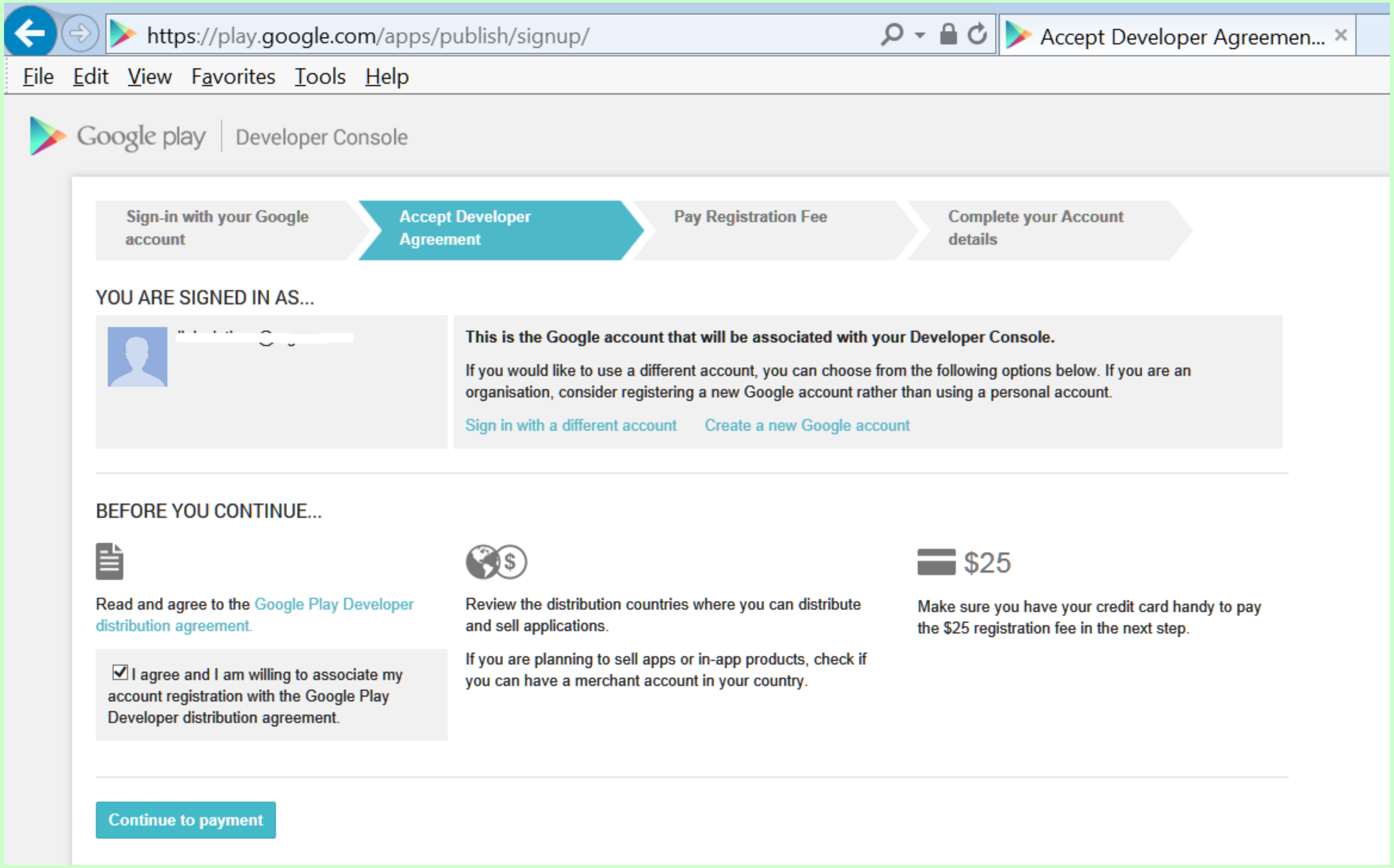
Testing Your App

- Google's Cloud Test Lab https://developers.google.com/cloud-test-lab helps you test your app across a wide range of devices.

Creating a Developer Profile

- ❑ The first step toward publishing on the Android Market is to **create a developer profile** at https://play.google.com/apps/publish/signup/#.
- ❑ For this, you need a Google account (such as your Gmail account).
- ❑ Once you have logged in to the Google Play, you first create your developer profile

Creating a Developer Profile


https://play.google.com/apps/publish/signup/
Accept Developer Agreemen...
File Edit View Favorites Tools Help
Google play | Developer Console
Sign-in with your Google account
Accept Developer Agreement
Pay Registration Fee
Complete your Account details
$25
Continue to payment

YOU ARE SIGNED IN AS...

[redacted]

This is the Google account that will be associated with your Developer Console.

If you would like to use a different account, you can choose from the following options below. If you are an organisation, consider registering a new Google account rather than using a personal account.

Sign in with a different account Create a new Google account

BEFORE YOU CONTINUE...

Read and agree to the Google Play Developer distribution agreement.

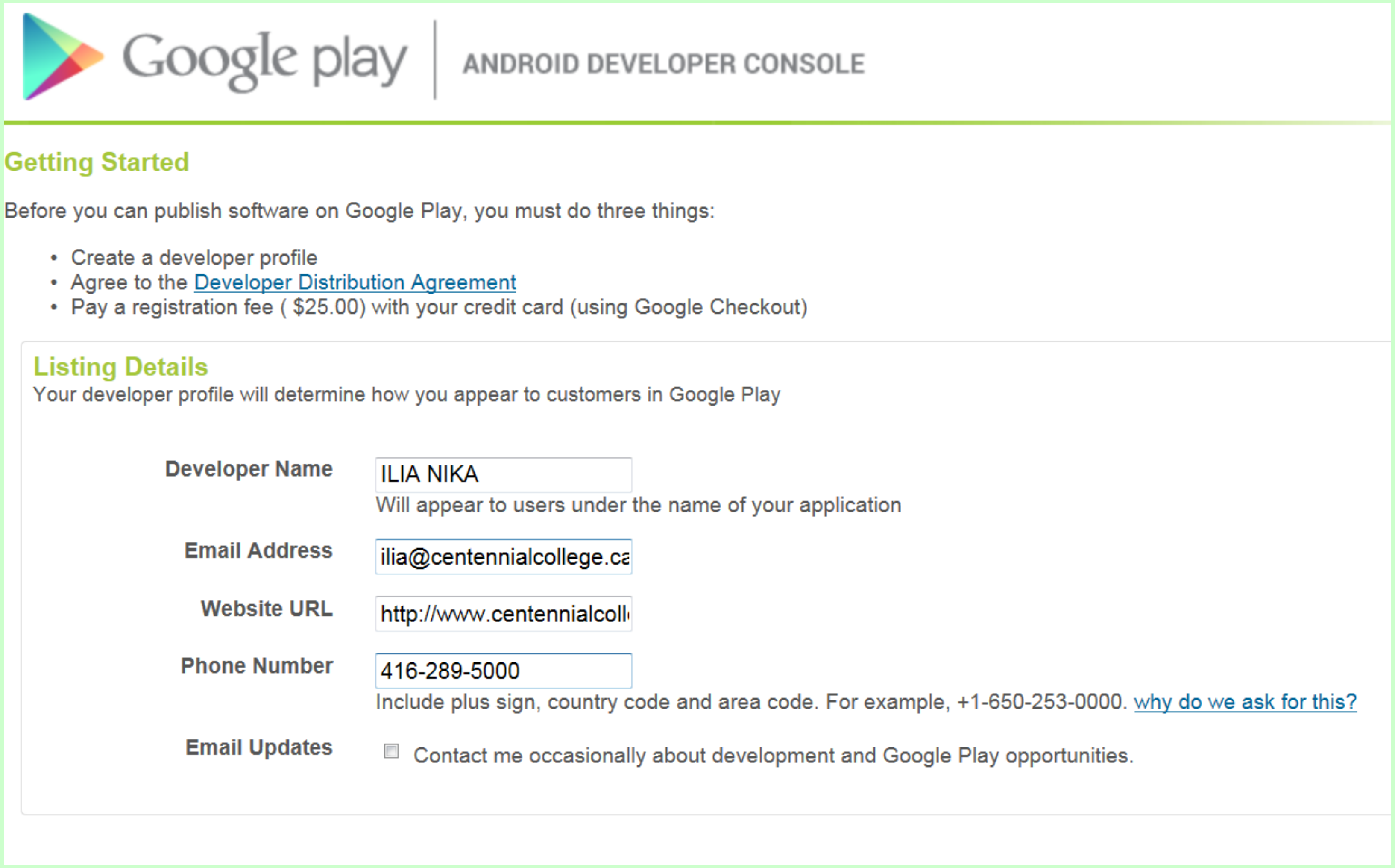
☑ I agree and I am willing to associate my account registration with the Google Play Developer distribution agreement.

Review the distribution countries where you can distribute and sell applications.

If you are planning to sell apps or in-app products, check if you can have a merchant account in your country.

Make sure you have your credit card handy to pay the $25 registration fee in the next step.

Creating a Developer Profile



Getting Started

Before you can publish software on Google Play, you must do three things:

- Create a developer profile
- Agree to the Developer Distribution Agreement
- Pay a registration fee ($25.00) with your credit card (using Google Checkout)

Listing Details

Your developer profile will determine how you appear to customers in Google Play

Developer Name	ILIA NIKA Will appear to users under the name of your application
Email Address	ilia@centennialcollege.ca
Website URL	http://www.centennialcoll
Phone Number	416-289-5000 Include plus sign, country code and area code. For example, +1-650-253-0000. why do we ask for this?
Email Updates	☐ Contact me occasionally about development and Google Play opportunities.

Creating a Developer Profile

- ❑ You need to pay a one-time registration fee, currently U.S. $25.
- ❑ Click the Google Checkout button to be redirected to a page where you can pay the registration fee.
- ❑ After paying, click the Continue link.
- ❑ Next, you need to agree to the Android Market Developer Distribution Agreement.
- ❑ Check the “I agree” checkbox and then click the “I agree.
- ❑ Continue” link

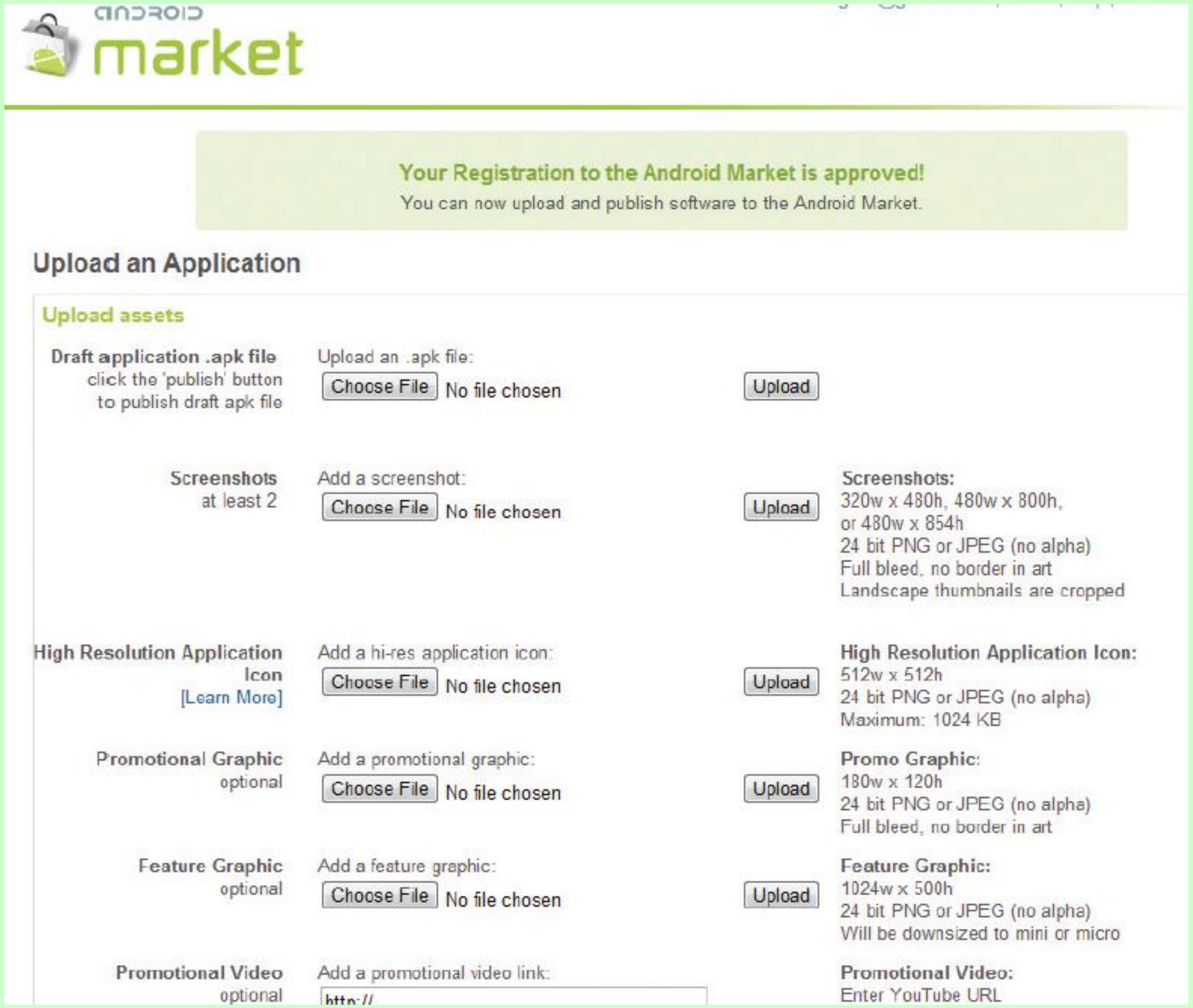
Submitting Your Apps

- After you have set up your profile, you are ready to submit your application to the Android Market.
- If you intend to charge for your application, click the Setup Merchant Account link located at the bottom of the screen.
- Here you enter additional information such as bank account and tax ID.
- For free applications, click the Upload Application link

Submitting Your Apps

- You will be asked to supply some information about your application.
- Among the information needed, the following are compulsory:
  - The application must be in APK format
  - You need to provide at least two screenshots.
    - You can use the DDMS perspective in Eclipse to capture screenshots of your application running on the emulator or real device.
  - You need to provide a high-resolution application icon.
    - This size of this image must be 512x512 pixels.
- The other information details are optional

Submitting Your Apps



android market

Your Registration to the Android Market is approved!
You can now upload and publish software to the Android Market.

Upload an Application

Upload assets

Draft application .apk file click the 'publish' button to publish draft apk file	Upload an .apk file: Choose File No file chosen	Upload	
Screenshots at least 2	Add a screenshot: Choose File No file chosen	Upload	**Screenshots:** 320w x 480h, 480w x 800h, or 480w x 854h 24 bit PNG or JPEG (no alpha) Full bleed, no border in art Landscape thumbnails are cropped
High Resolution Application Icon [Learn More]	Add a hi-res application icon: Choose File No file chosen	Upload	**High Resolution Application Icon:** 512w x 512h 24 bit PNG or JPEG (no alpha) Maximum: 1024 KB
Promotional Graphic optional	Add a promotional graphic: Choose File No file chosen	Upload	**Promo Graphic:** 180w x 120h 24 bit PNG or JPEG (no alpha) Full bleed, no border in art
Feature Graphic optional	Add a feature graphic: Choose File No file chosen	Upload	**Feature Graphic:** 1024w x 500h 24 bit PNG or JPEG (no alpha) Will be downsized to mini or micro
Promotional Video optional	Add a promotional video link: http://		**Promotional Video:** Enter YouTube URL

Submitting Your Apps

- The next set of information you need to supply includes the title of your application, its description, as well as details about recent changes (useful for application updates).
- You can also select the application type and the category in which it will appear in the Google Play.

Submitting Your Apps

- In the last dialog, you indicate whether your application employs copy protection, and specify a content rating.
- You also supply your website URL and your contact information.
- When you have given your consent to the two guidelines and agreements, click Publish to publish your application on the Google Play.

References

- Textbook (chap. 12)
- Reference textbook
- Android Documentation
- https://developer.android.com/studio/publish/index.html
- http://developer.android.com/tools/publishing/preparing.html
- http://developer.android.com/tools/publishing/app-signing.html
- http://developer.android.com/distribute/tools/launch-checklist.html
- *Android 6 for Programmers: An App-Driven Approach, Deitel & Deitel, 2016.*