



Deployment and Monetization

**IT6306 - Mobile Application
Development**

Level III - Semester 6

Overview

- This lecture will elaborate on the following key points,
 - Android application deployment on Google Play Store
 - App Publishing process
 - Monetization options

Intended Learning Outcomes

At the end of this lesson, you will be able to;

- Describe the app publishing process
- Apply knowledge to determine a suitable option to publish Android app on google play store
- Determine a suitable monetization strategy

List of sub topics

1. App publishing / Deploying

- 1.1. App releasing strategies
- 1.2. Prepare for release
- 1.3. Versioning the app
- 1.4. Signing the app
- 1.5. Uploading the app
- 1.6. Choosing the right monetization strategy
- 1.7. Google Play's subscription platform
- 1.8. Using google AdMob

1.1 App Publishing/ Deploying

Importance of publishing your app

- After developing an application, the next step is to publish it for its users
- There are three main advantages of publishing your application
 - Reach a wider audience
 - Generate revenue
 - Gain user feedback

App publishing process

- Publishing is the process that makes your Android app available to users
- There are three high level steps in publishing process
 - Prepare the app for release
 - Release the app to users
 - Maintain and update your app

1.1 App Releasing Strategies

Three main strategies to release an app

- Releasing your app can be done via
 - an app marketplace such as Google Play
 - on your own website
 - by directly sending it to a user

Release through an App Marketplace

- The broadest reach is achieved through an app marketplace
- Google Play is the premier marketplace for Android apps for a large global audience
- Apps can be distributed through any app marketplace, and multiple marketplaces can be utilized

Releasing Apps on Google Play

- Google Play is a robust publishing platform providing various developer tools
- It offers features such as in-app billing and app licensing, along with numerous end-user community features

Preparing for Release

- Prepare promotional materials like screenshots, videos, graphics, and promotional text
- Configure options and upload assets
 - Set countries, listing languages, price, app type, category, and content rating
- Publish the release version of your app

Release through a Website

- Release your app on your own website or server, including on a private or enterprise server

Steps:

- Prepare your app for release
- Host the release-ready APK file
- Provide a download link to users

Note:

- The user must have their settings configured to allow installation from unknown sources

1.2 Prepare for Release

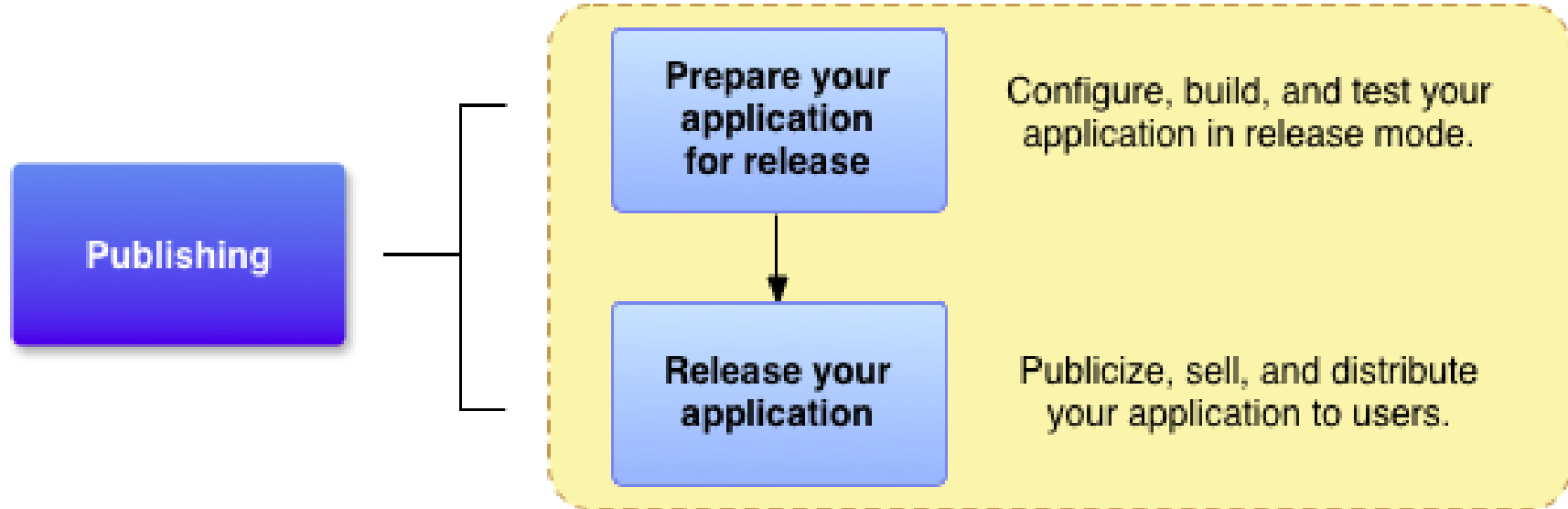
Preparing for Release

Key steps:

- Configure Gradle for release
- Sign your app
- Optimize your app's resources
- Review the app manifest
- Test your app

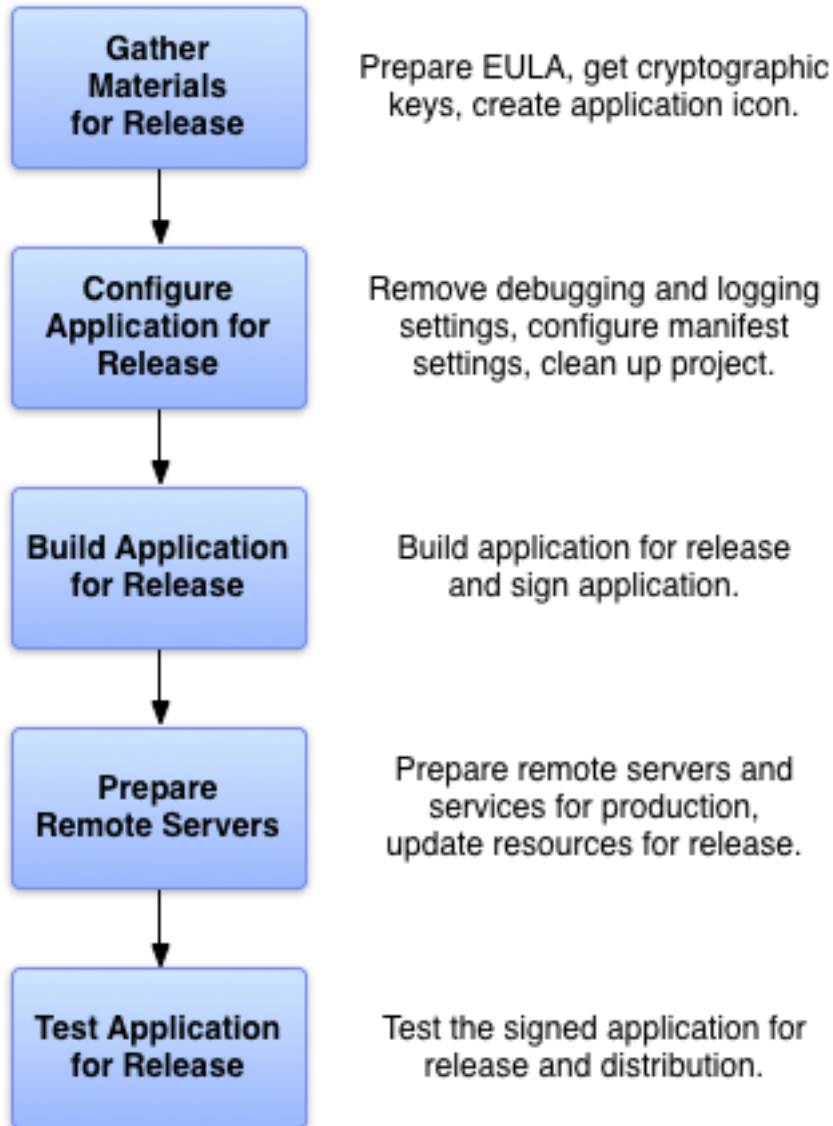
Prepare for Release Step

- Preparing step for release is the first step in publishing the app



Steps of Publishing an Application

Tasks to complete in prepare for release step



Five main tasks to prepare the app for release

1.3 Versioning the App

Versioning

- Versioning is a critical component of your app upgrade and maintenance strategy
- Version information to protect against downgrades

Version Settings – Gradle Build File

```
android {  
    namespace 'com.example.testapp'  
    compileSdk 33  
  
    defaultConfig {  
        applicationId "com.example.testapp"  
        minSdk 24  
        targetSdk 33  
        versionCode 1  
        versionName "1.0"  
        ...  
    }  
    ...  
}
```

Version Code

- Positive integer used as an internal version number
- Helps determine whether one version is more recent than another
- Higher numbers indicate more recent versions
- Not the version number shown to users

Role of **versionCode** in Android System

- The Android system uses the **versionCode** value to protect against downgrades
- Prevents users from installing an APK with a lower **versionCode** than the version currently installed on a device

Usage of versionCode in App Releases

- The first version of an app is released with versionCode set to 1
- Monotonically increase the value with each release, regardless of whether the release constitutes a major or minor release
- The versionCode value does not necessarily resemble the app release version that is visible to the user

Version Name

- A string used as the version number shown to users
- The value is a string so that you can describe the app version as a string

<major>.<minor>.<point> (Eg: 1.1.1)

- The versionName is the only value displayed to users

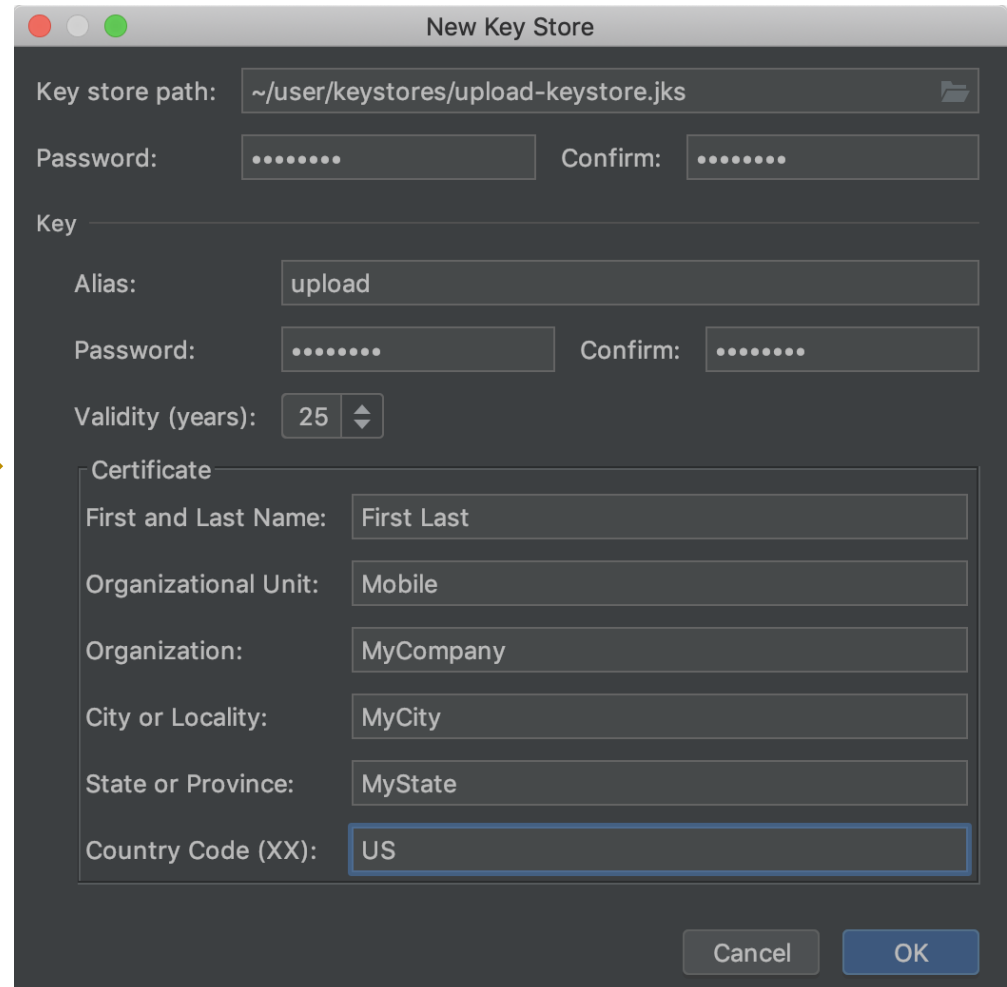
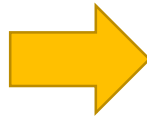
1.4 Signing the App (Android Studio)

Generate an upload key and keystore (Android Studio)

- In the menu bar of Android studio,
 - click Build > Generate Signed Bundle/APK
- In the Generate Signed Bundle or APK dialog,
 - select Android App Bundle or APK and click Next
- Below the field for Key store path,
 - click Create new

Generate an upload key and keystore

On the New Key Store window, provide the information shown



The 'New Key Store' dialog box contains the following fields and values:

- Key store path: ~/user/keystores/upload-keystore.jks
- Password: Confirm:
- Key:
 - Alias: upload
 - Password: Confirm:
 - Validity (years): 25
- Certificate:
 - First and Last Name: First Last
 - Organizational Unit: Mobile
 - Organization: MyCompany
 - City or Locality: MyCity
 - State or Province: MyState
 - Country Code (XX): US

Buttons: Cancel, OK

Keystore

- Key store path:
 - Select the location where your keystore should be created
 - Also, a file name should be added to the end of the location path with the .jks extension
- Password:
 - Create and confirm a secure password for your keystore

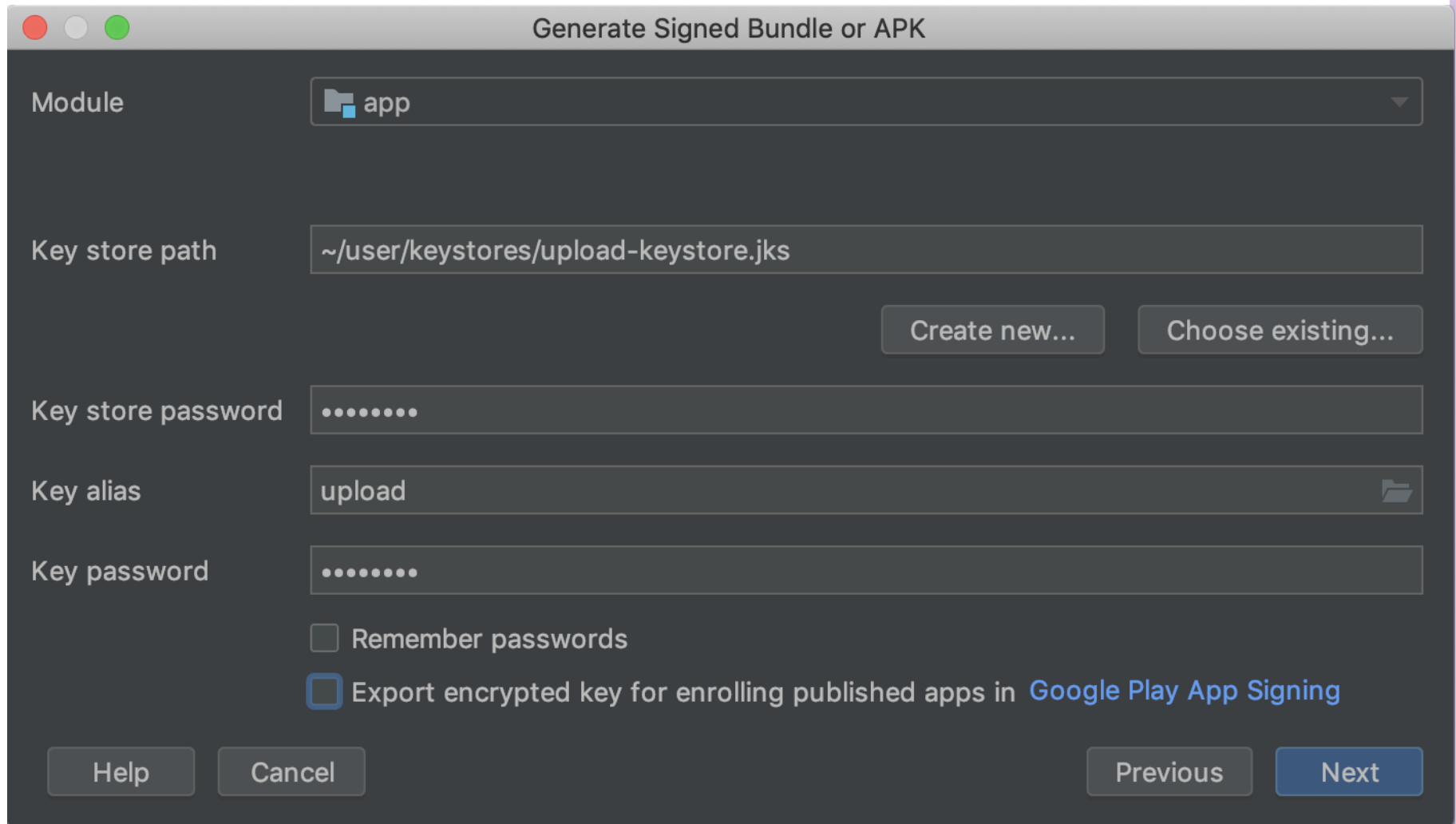
Key

- **Alias:** Enter an identifying name for your key
- **Password:** Create and confirm a secure password for your key
- This should be the same as your keystore password
- **Validity (years):** Set the length of time in years that your key will be valid
- **Certificate:** Enter some information about yourself for your certificate
- Once you complete the form, click OK

Sign the App with the Generated Key (Android Studio)

- click Build > Generate Signed Bundle/APK
- In the Generate Signed Bundle or APK dialog,
 - select either Android App Bundle or APK and click Next
- Select a module from the drop down
- Specify the path to your keystore, the alias for your key, and enter the passwords for both
- Click Next.

Sign the App with the Generated Key (Android Studio)



The screenshot shows the 'Generate Signed Bundle or APK' dialog box in Android Studio. The dialog has a title bar with standard macOS window controls (red, yellow, green buttons) and the title 'Generate Signed Bundle or APK'. The main area contains several fields and buttons:

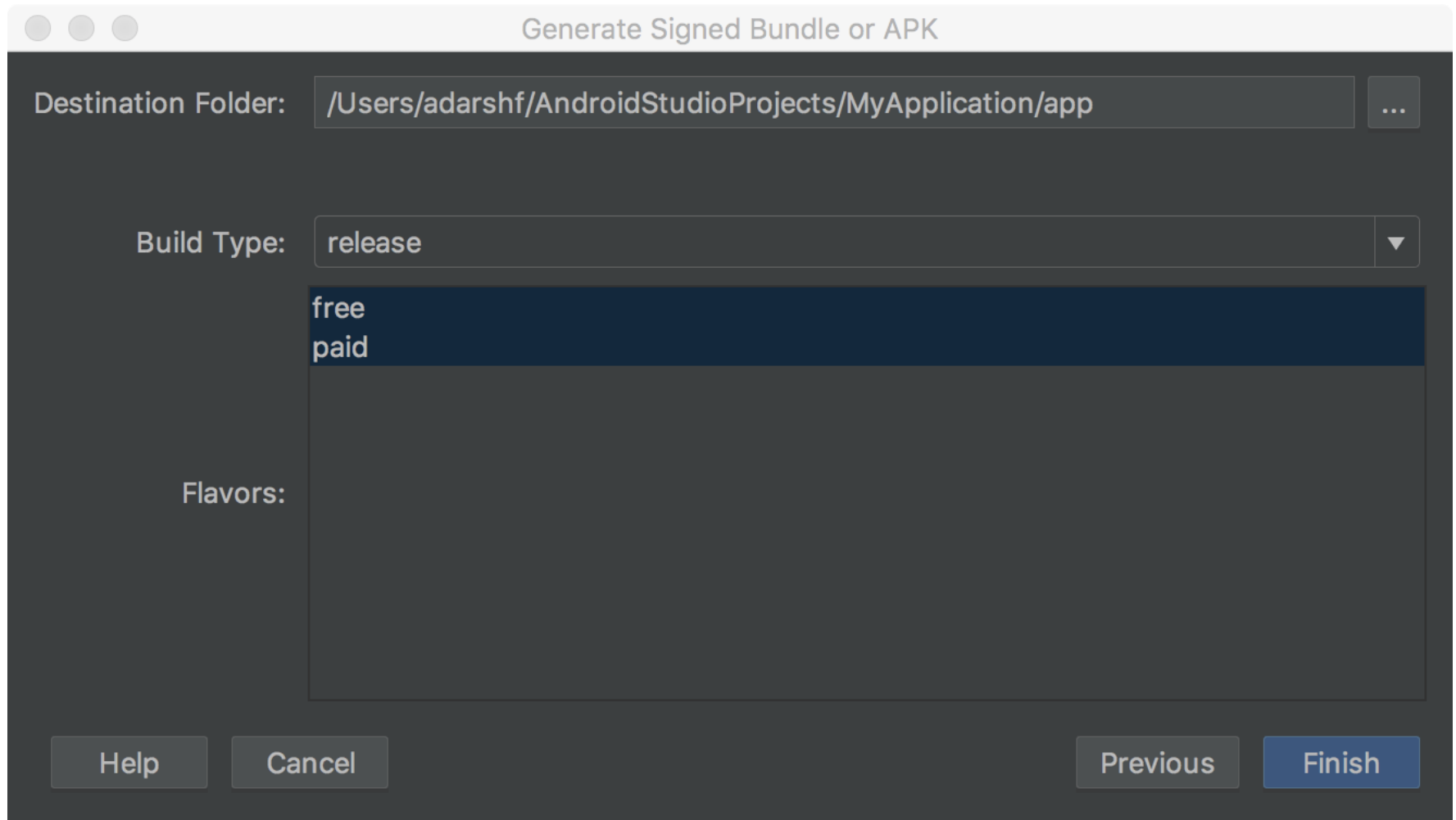
- Module:** A dropdown menu showing 'app' with a folder icon.
- Key store path:** A text field containing '~/.user/keystores/upload-keystore.jks'.
- Key store password:** A text field with masked characters (dots).
- Key alias:** A text field containing 'upload' with a folder icon.
- Key password:** A text field with masked characters (dots).
- Buttons:** 'Create new...' and 'Choose existing...' are located below the 'Key store path' field.
- Checkboxes:** 'Remember passwords' and 'Export encrypted key for enrolling published apps in Google Play App Signing' are located below the 'Key password' field.
- Navigation Buttons:** 'Help', 'Cancel', 'Previous', and 'Next' are located at the bottom.

Sign your app with the upload key

Sign the App with the Generated Key (Android Studio)

- In the next window,
 - select a destination folder for your signed app
 - select the build type
 - choose the product flavor(s) if applicable
- If you are building and signing an APK, you need to select which Signature Versions you want your app to support

Sign the App with the Generated Key (Android Studio)



Generate a signed version of your app for the selected product flavors

1.5 Uploading the App

Prerequisites for Uploading the App

- Enroll in Play App Signing (mandatory since August 2021)
- Ensure your app meets Google Play's size requirements (150 MB or less compressed download size)
- Prepare to upload your app to the Play Console

Testing Your App Internally

- Different ways to share your app internally for testing
- **Firebase App Distribution**
 - Allows you to deploy any kind of build and distribute it to a list of users
 - Useful for distributing builds from a continuous integration system
- **Play Console's internal app sharing tool**
 - Faster to deploy compared to alpha or beta tracks
 - Provides access to services like Subscriptions, In-App purchases, and ads
 - Undergoes Play Console signing and shrinking, providing a close representation of what end-users receive

Updating Your App Bundle

- Increase the version code in the base module for updates
- Build and upload a new app bundle
- Google Play generates updated APKs with new version codes and serves them to users as needed

Summary - Publishing to Google Play

- Create a Google Play Developer account
 - One-time registration fee
- Set up a Google Play Console account
 - Manage apps and their distribution
- Create a store listing
 - Add app title, description, and graphics
- Add content rating
 - Fill out the questionnaire
- Set up pricing & distribution
 - Free or paid, countries, device categories
- Roll out your app
 - Choose release track (alpha, beta, or production)

1.6 Choosing the right Monetization Strategy

Prerequisites for Uploading the App

- There are five popular app monetization strategies
 - Free and Paid App Versions Model
 - Free App with In-app Purchases Model
 - Free App with Subscription Model
 - Paid App Model
 - Partnership Model

Free and Paid App Versions Model

- Offer both free and paid versions of your app
- Monetization:
 - Limit features in the free version to encourage upgrades or include in-app advertising
- Benefits:
 - Free version attracts users; potential monetization through upgrades or ads

Free App with In-app Purchases Model

- Provide a free app with basic features; offer advanced or premium features via in-app purchases
- In some cases, patience or frequent engagement allows users access to premium features for free
- Some features may be exclusively obtainable through in-app purchases

Free App with Subscription Model

- Free-to-download app with limited access to content/services
- Subscription required for full benefits
- Commonly used in service-oriented or content-centric apps
- Helps build an initial user base and cultivate paid subscribers

Paid App Model

- Offer apps exclusively in a paid version, providing unique value or functionality
- Commonly used in productivity apps

Note:

- App should offer substantial value to build a user base and generate revenue

Partnership Model

- Concept: Leverage a popular app's niche audience for sponsorship opportunities.
- Think about the target user base and the brands that might be interested in this audience.
- Monetization through brand sponsorship, partnership, or acquisition could be a good choice for large user bases.

1.7 Google Play's subscription platform

Subscription Platform

- A tool allowing developers to offer users access to content and features over a set time period
- Key Features of Google Play Subscriptions
 - Automatic billing on a recurring basis
 - Subscription management for users
 - Free trial and introductory price support

Benefits of Using Subscription Platform

- Recurring revenue and customer retention
- Enhanced user engagement and app loyalty
- Greater understanding of user behavior through subscription analytics

Subscription Types

- One-time purchase:
 - Users pay once to access content or features
- Recurring subscription:
 - Users pay on a regular basis to access content or features

Note:

Google Play handles renewals and allows grace periods for lapsed payments, keeping users subscribed while payment issues are resolved

Pricing and Free Trials

- Developers have the flexibility to set their subscription pricing and length of free trial periods
- Introductory pricing can be set to attract new subscribers



1.8 Using Google AdMob

What is Google AdMob?

- AdMob is a mobile advertising platform by Google
- Helps app developers monetize their apps through in-app advertising
- Provides insights into user behavior to enhance app marketing and monetization strategies

How AdMob Works

- Display ads within your app using the AdMob platform
- Get paid whenever users interact with ads
- Leverage Google's vast advertising network for diverse, high-quality ads

Integrating AdMob into Android App

- Sign up for an AdMob account
- Add your app to the AdMob console
- Install the Google Mobile Ads SDK
- Configure ad units and place them in your app

AdMob Ad Formats

- **Banner Ads:**
 - Standard, rectangular ads that can be placed in various locations within your app
- **Interstitial Ads:**
 - Full-screen ads that appear at natural breaks or transition points in your app
- **Rewarded Ads:**
 - Ads that reward users for watching short videos or interacting with playable ads
- **Native Ads:**
 - Ads that match the look and feel of your app for a seamless user experience

AdMob Metrics

- **Impressions:**
 - Number of times an ad is displayed in your app
- **Clicks:**
 - Number of times users click on the ads
- **CTR (Click Through Rate):**
 - The ratio of users who click on an ad to the number of total users who view the ad
- **eCPM (Effective Cost Per Mille):**
 - The estimated earnings per thousand ad impressions

Summary

App publishing

Publishing is the process of allowing the users to access the application

Monetization

Choosing the right monetization approach will help to achieve the desired goals

Mobile
Advertising
Platform

AdMob is one of the popular and easy to use monetization option. You can use different types of adds withing your app