INSTITUTE OF SPACE TECHNOLOGY
KICSIT, Kahuta Campus

App Development Assignment

Name: Usman Shahid

Serial No: 232202007

Class: BSCS

Semester: 6th

Instructor: Sir Uzair

1 Introduction

In this report, I have explained everything I learned about preparing and releasing an Android app — from managing version codes to generating a signed release and understanding the Google Play publishing process. When I started this assignment, I had only a basic idea about Android builds, but while exploring Android Studio, I discovered how important these technical details are for real-world app publishing. The goal of this report is not just to repeat theory but to present what I personally understood, what each step means, and why it matters if you ever want to upload an app to the Play Store.

2 Part 1 — APK Versioning (Understanding `versionCode` and `versionName`)

2.1 What are `versionCode` and `versionName`?

In Android apps, every version has two numbers that describe it:

- **versionCode:** A positive integer that changes with every release. This number is for the system, not for users. The Play Store uses it to check if an update is newer.
- **versionName:** A string (like "1.0", "1.1.2" or "2.0-beta") that is shown to users. It's for humans to easily identify the version they are using.

I think of `versionCode` as something only Google Play and Android devices care about, while `versionName` is like the label users see on the screen.

2.2 Why they are important

Both are important because without proper versioning, updates cannot be distributed properly. If you upload a new build without increasing the `versionCode`, Play Console will reject it saying it's not a newer version. Even if your app code has changed, the store won't accept it because it compares `versionCode`, not the actual files.

This system prevents mistakes like pushing the wrong build or overwriting older versions. It ensures that updates move forward in sequence.

2.3 Where to edit these values in Android Studio

These values are found inside the `build.gradle` file at the app module level. When I opened it in Android Studio, I found a section like this:

```
defaultConfig {
    applicationId "com.example.myapp"
    minSdk 21
    targetSdk 34
    versionCode 3
    versionName "1.2.0"
}
```

Here, increasing `versionCode` to the next number (for example, from 3 to 4) is enough before generating a new release.

2.4 My Reflection

At first, I didn't really understand why the Play Store was so strict about version codes, but after trying to build the same version twice, I got the error that said the version was already uploaded. That's when I realized how this small detail saves developers from confusion. It also made me appreciate why version control and numbering systems are so essential in professional app development.

3 Part 2 — Generating a Signed Build

3.1 What is a `.jks` (keystore) file and why it matters

A keystore file is like a digital identity card for your app. It holds your private key and is used to "sign" the app bundle or APK, proving that it truly came from you. When users install an update, Android checks if the app is signed with the same key. If it's different, it refuses to install the update. That's why losing your keystore means losing your ability to update your app.

In short: The keystore guarantees authenticity, ownership, and security of the app. It should always be stored carefully, just like a password or private document.

3.2 Difference between .apk and .aab files

There are two main release formats:

- **APK (Android Package):** The classic installable file format. You can send it directly to others or sideload it on a phone.
- **AAB (Android App Bundle):** A newer, more advanced format introduced by Google. Instead of containing all versions of the app, it lets Play Store build optimized APKs for each user's device automatically.

I personally prefer AAB now because it is the required format for the Play Store and makes the final app size smaller.

3.3 Steps I followed to generate a signed build

I followed these steps carefully in Android Studio:

1. I opened my project.
2. I went to **Build → Generate Signed Bundle / APK**.
3. I selected **Android App Bundle (AAB)** since it's now the standard.

4. I chose to create a new `.jks` file and entered a strong password, alias, and certificate details.

5. I saved it in a secure folder on my laptop (and backed it up on a USB drive).

6. I selected the "release" build type and finished the process.

After a few moments, Android Studio generated my signed file and showed the location in the output folder.

3.4 Precautions and Safety Tips

- Always remember your keystore password. Without it, you cannot sign updates for the same app.
- Don't share the keystore file with anyone, even friends. It's your developer identity.
- Keep multiple backups in different safe places.
- Avoid storing the keystore in your GitHub or public repositories.

3.5 My Experience

The first time I generated a signed AAB, I actually got confused between keystore and key alias, but after reading Android documentation, it started to make sense. It was also interesting to see how a digital signature works — it's not visible in the app itself, but it silently guarantees authenticity behind the scenes.

4 Part 3 — Publishing to Google Play

4.1 Creating a Google Play Developer Account

To publish apps officially, a developer account is required. The setup is simple but includes a few steps:

- You need a Google account.
- You must pay a one-time registration fee (currently around $25).
- You fill out basic information such as developer name, contact email, and phone number.
- You accept Google's Developer Distribution Agreement and policies.

Once this account is active, you can create your first app entry and start preparing it for publishing.

4.2 Main Steps in the Publishing Process

Here is a summary of the publishing process as I understood it:

1. Create a new app entry in the Play Console.
2. Fill in details like the app's name, category, short and full description.
3. Upload your signed AAB file under the "Production" release section.
4. Add screenshots, an app icon, and feature graphics.
5. Complete all policy forms, including privacy policy and data safety.
6. Set the price (free or paid) and choose countries for availability.
7. Review everything and click "Submit for Review".

After submission, Google runs automatic checks, followed by a short review before the app becomes live.

4.3 Common Errors and How to Avoid Them

I noticed that many apps get rejected because of small mistakes. Here are some examples:

- **Privacy policy missing:** Even basic apps need one, especially if they request permissions like camera or storage.
- **Inaccurate description:** Don't promise features that are not in the app.
- **Inappropriate permissions:** Request only the permissions your app actually needs.
- **Policy violations:** Make sure the app content doesn't violate Google's family or content policies.

4.4 Personal Understanding

Publishing an app is not just about uploading a file — it's about being responsible as a developer. Google expects us to be clear about what the app does and how it handles user data. This part really taught me the professional side of development, where communication and compliance matter just as much as coding.

5 Part 4 — Practical Simulation

For the practical part, I made a small "Hello World" app just to test everything properly. It was a simple project but helped me practice versioning and signing steps in a real environment.

5.1 Version Code and Name Setup

Inside `app/build.gradle`, I set:

```
versionCode 3
versionName "1.2.0"
```

I then rebuilt the project and verified that the values appeared correctly in the generated bundle.

5.2 Signed AAB Generation

After signing the bundle, I found the file in:

`app/build/outputs/bundle/release/app-release.aab`

That confirmed the process was done correctly. The file size was smaller than expected because AABs are more optimized than normal APKs.

5.3 My Thoughts After Doing It

Before doing this, the Play Store publishing process seemed very technical and complicated. But after following each step once, I realized it's quite logical. The idea is simple: version your app correctly, sign it securely, provide transparent information, and you're good to go.

6 Final Reflection and Learning Summary

This assignment taught me more than I expected. I not only learned how Android manages versions and builds but also understood how professionalism works in app publishing. Each step — from versioning to signing — has a purpose, and skipping any one of them can cause real trouble later.

Some important takeaways I'll always remember:

- **Never forget to increase versionCode.**
- **Always back up your keystore safely.**
- **Keep your app information honest and clear.**
- **Follow Play Store policies carefully to avoid rejection.**

Overall, this assignment connected technical practice with real-world application. It made me realize that every Android developer, even a beginner, must learn these details before releasing anything publicly. Now, I feel much more confident about how to publish an app professionally and securely.