Publishing an Android App on Google Play Store

Submitted by: Muhammad Mohsin (232202022)
Course: Android App Development
Instructor: Sir Uzair
Submission Date: November 12, 2025

Contents

1. Introduction

This report explains how to prepare, sign, and publish an Android application on the Google Play Store using Android Studio and the Google Play Console. It also demonstrates how to manage versioning in Android, generate a signed release build, and simulate the release process using a simple application.

2. Learning Objectives

By completing this assignment, the following learning objectives are achieved:

- Understand versionCode and versionName management in Android projects.
- Learn how to generate and sign an Android App Bundle (AAB) or APK file.
- Understand the Google Play Console workflow and publishing steps.
- Prepare a technical report using LaTeX format explaining the full publishing process.

3. Part 1: APK Versioning

3.1. What are `versionCode` and `versionName`?

- **versionCode:** An integer value representing the internal version of your app. Each release must have a higher versionCode so Play Store recognizes it as an update.
- **versionName:** A user-readable string (e.g., "1.0" or "2.1.3") that shows the release version to users.

3.2. Why are they important?

They help the Play Store and users track updates. The Play Store uses `versionCode` to detect new releases; users see `versionName` to know which version they are running.

3.3. What happens if you don't increase the `versionCode`?

Uploading a new app version with the same `versionCode` as the previous one will be rejected by the Play Console.

3.4. Where can you find and edit these values?

In `build.gradle (Module: app)`:

```
android {
    defaultConfig {
        applicationId "com.example.myapp"
        minSdkVersion 24
        targetSdkVersion 34
        versionCode 2
        versionName "1.1"
    }
}
```

4. Part 2: Generating a Signed Build

4.1. What is a `.jks` (keystore) file and why is it critical?

A keystore (`.jks`) contains cryptographic keys used to sign your app. It ensures only the developer can publish updates. Losing it means you cannot update your app on Play Store.

4.2. Difference between `.apk` and `.aab` files

- **APK:** Installable file containing all code and resources.
- **AAB:** Recommended bundle format; Play Store generates optimized APKs per device.

4.3. Steps to generate a signed release in Android Studio

1. Open your project in Android Studio.
2. Click **Build → Generate Signed Bundle / APK**.
3. Choose **Android App Bundle (.aab)** or **APK**.
4. Create or select an existing keystore (`.jks`) file.
5. Fill key alias, passwords, and certificate details.
6. Choose **release** build variant.
7. Click **Finish**. Signed file will appear in `app/release/`.

4.4. Precautions for keystore file and passwords

- Keep `.jks` file and passwords safe.
- Losing them means you cannot publish updates.

- Never share credentials.

5. Part 3: Publishing to Google Play Store

5.1. Requirements for a Developer Account

- Google account.
- One-time $25 registration fee.
- Acceptance of Google Play policies.

5.2. Steps to publish the app

1. Go to `https://play.google.com/console`, sign in.
2. Click "Create App" and fill app info.
3. Complete forms: privacy policy, data safety, ads, content rating.
4. Upload signed `.aab` in Production → Create Release.
5. Add release notes, choose countries and pricing.
6. Review and submit for rollout.
7. Wait for review (3–5 days).

5.3. Common errors / rejections

- Incorrect versionCode → always increment for updates.
- Policy violations → comply with Play policies.
- Invalid Data Safety form → be accurate.
- Missing assets → provide screenshots, icons, feature graphics.

6. Part 4: Practical Task Simulation

Sample "Hello World" app created.

6.1. Assigning versionCode and versionName

```
versionCode 1
versionName "1.0"
```

6.2. Generating signed release

1. Build → Generate Signed Bundle / APK.
2. Created new keystore: `mohsin-key.jks`.
3. Entered alias and password.
4. Generated signed `.aab` successfully.

7. Conclusion

This assignment covered all steps to prepare, sign, and publish an Android app — emphasizing versioning, signing, keystore security, and Play Console workflow. Following these steps reduces rejection risk and ensures a smooth release.

8. References

- Google Play Console Documentation: `https://developer.android.com/distribute`
- Google Play Policy Center: `https://play.google.com/about/developer-content-policy/`
- Android Developers – Build and Sign Your App: `https://developer.android.com/studio/publish/app-signing`

Appendix: Screenshots of Simulation

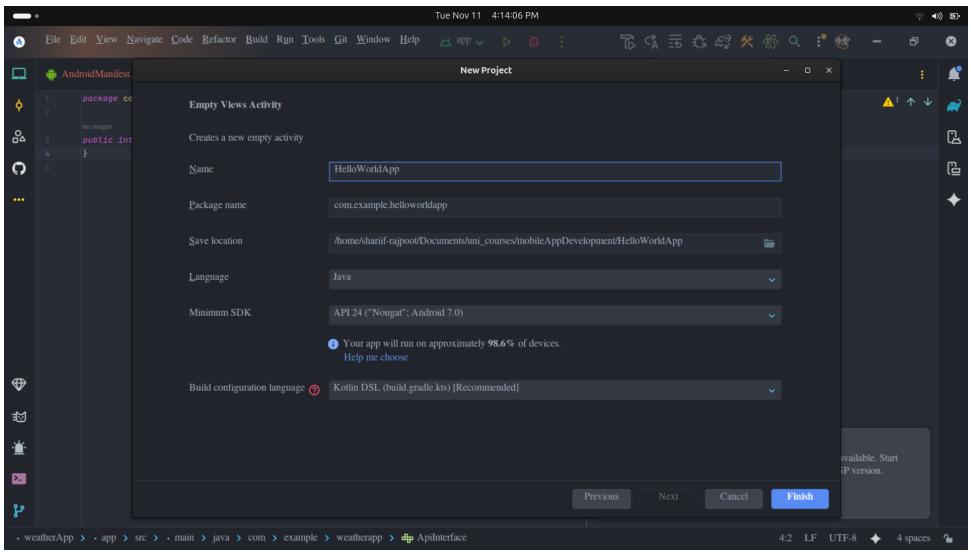

Figure 1: Start with the app building

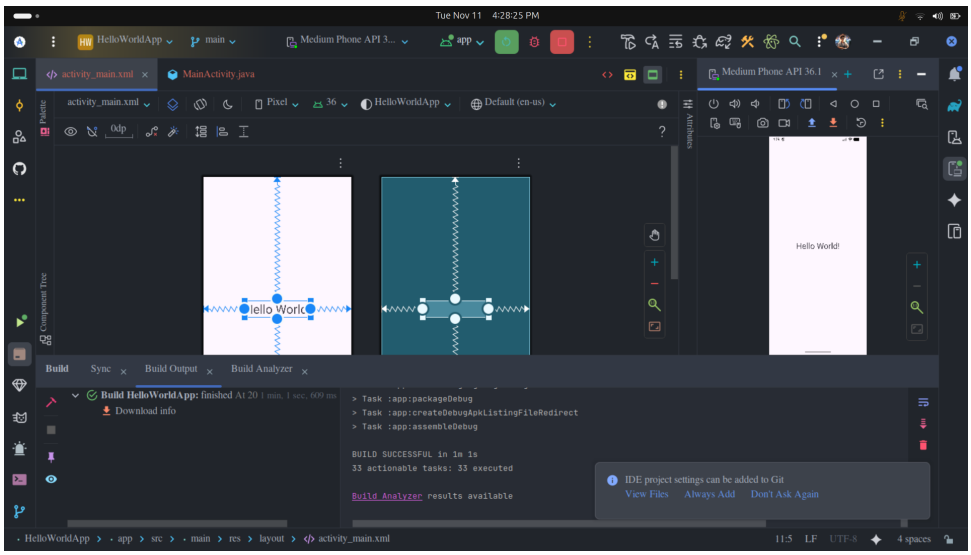

Figure 2: Gradle Building

```
android {
    namespace = "com.example.helloworldapp"
    compileSdk = 36

    defaultConfig {
        applicationId = "com.example.helloworldapp"
        minSdk = 24
        targetSdk = 36
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }
```

Figure 3: Assigning VersionName and VersionCode

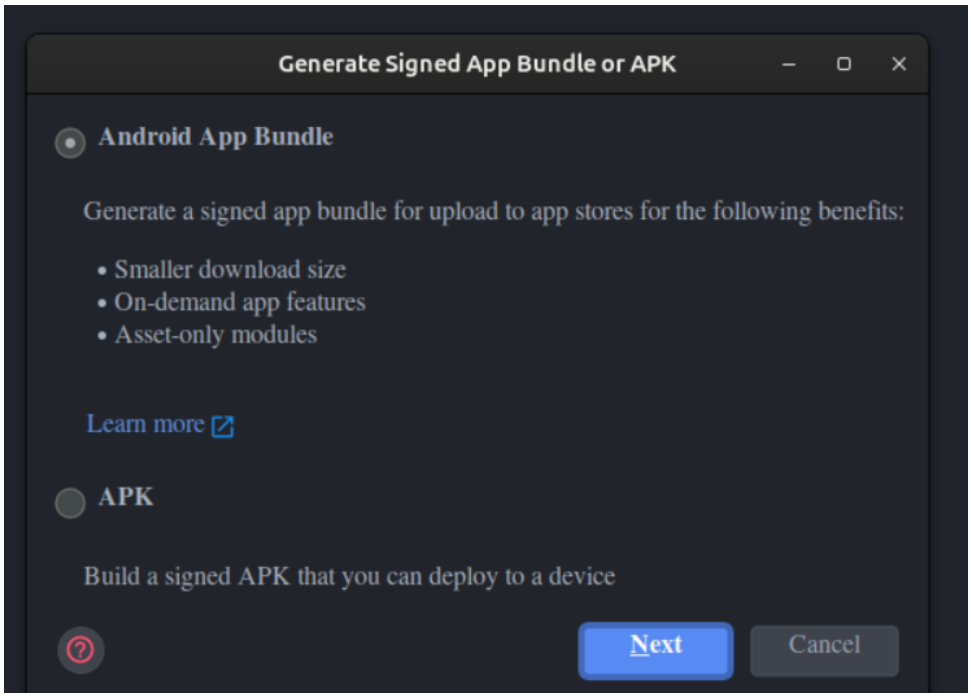

Figure 4: Generate Signed APP Bundle/APK

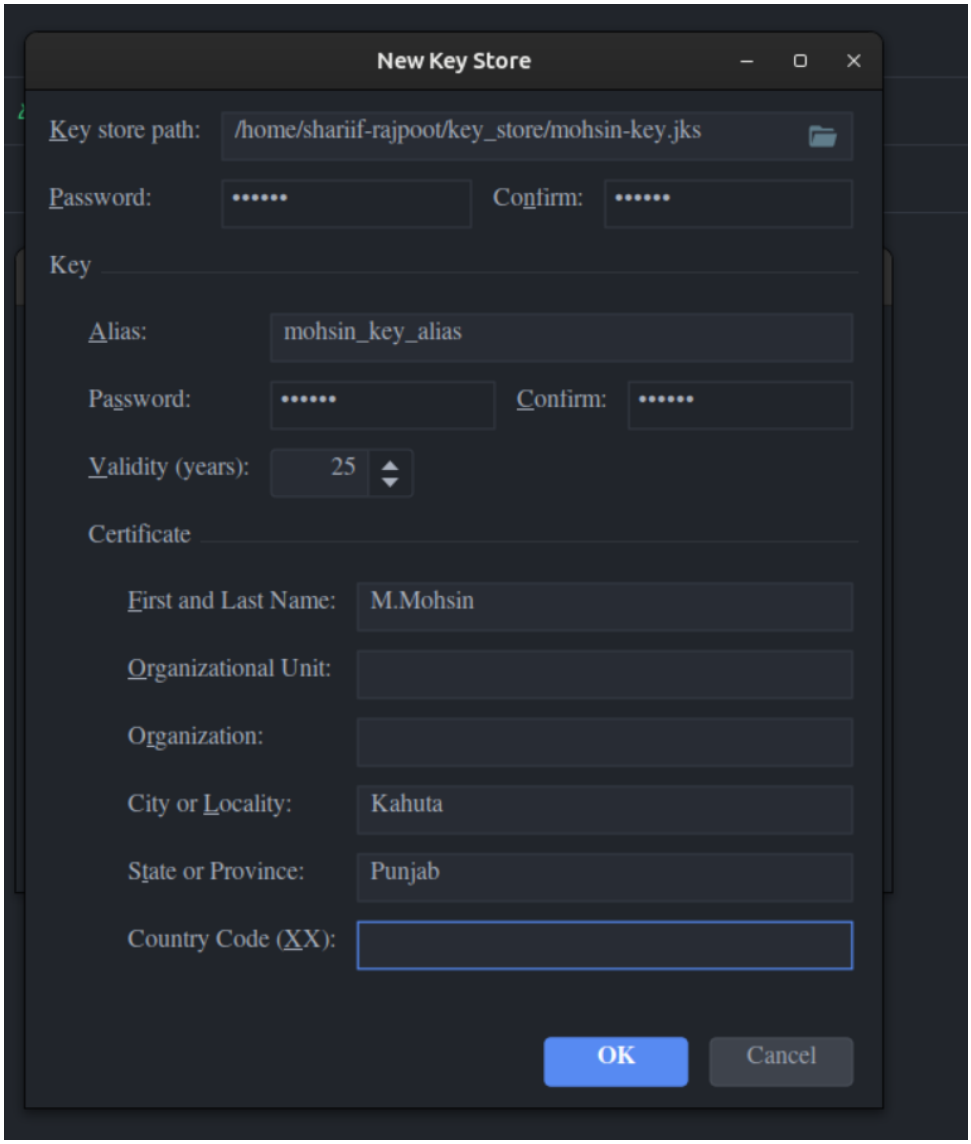

Figure 5: New Key Store

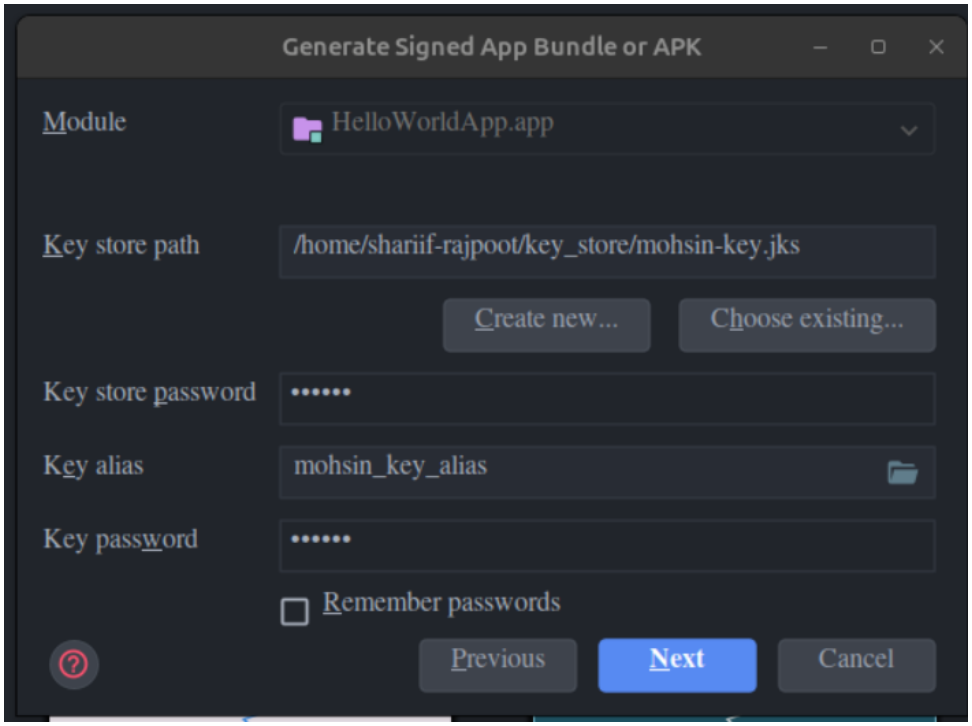

Figure 6: Generate Signed APP Bundle/APK

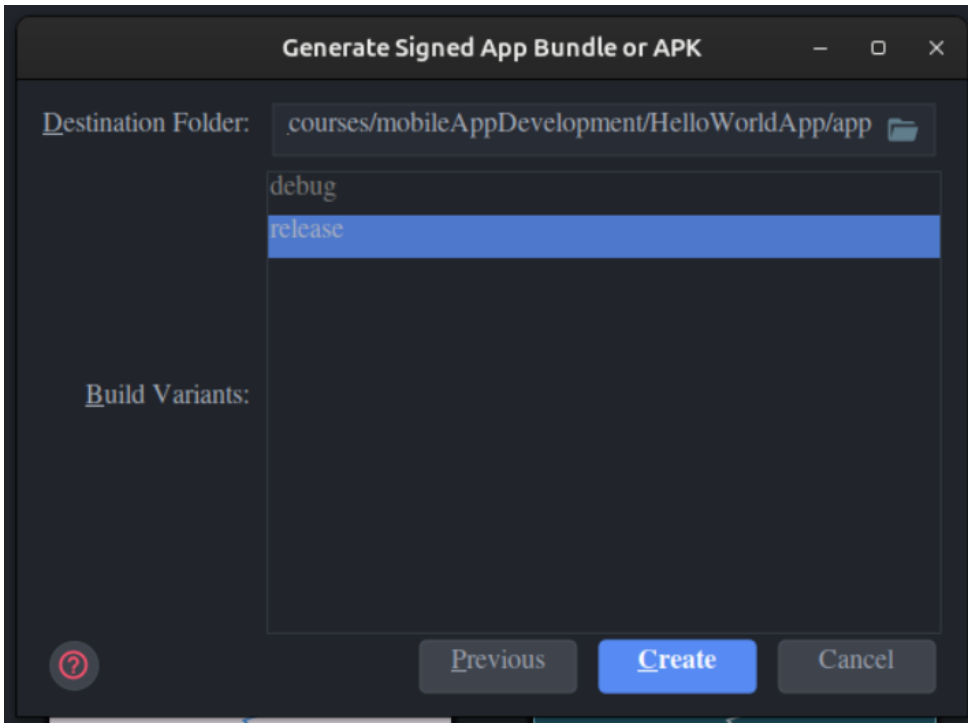

Figure 7: Generate Signed APP Bundle/APK

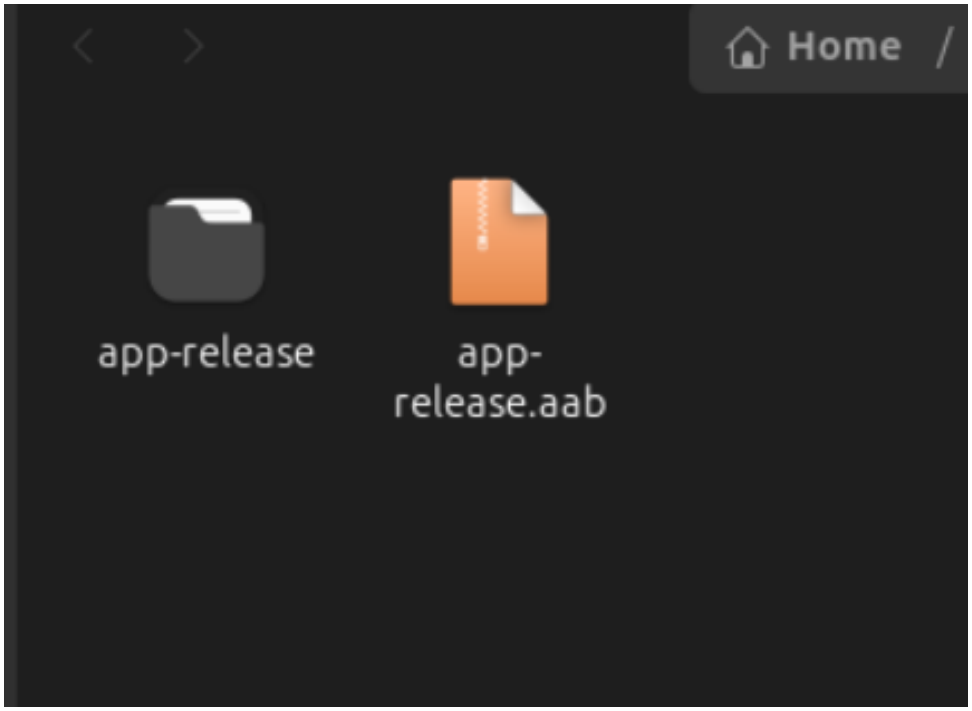

Figure 8: Generated signed .aab file in app/release folder