Module 7: Project 3

Shannon Musgrave

Computer Science Dept. of Southern New Hampshire University

CS-360 Mobile Architecture & Programming

Professor Krauss

02/23/2025

To ensure a successful app launch, it's crucial to utilize robust testing tools and pre-launch reports to identify and address technical issues before release. The app can be shared with internal testers for initial quality assurance checks, allowing developers to gather early feedback from trusted individuals through closed testing. Expanding to open testing on Google Play then enables private feedback from a larger group. Beyond testing and quality assurance, following specific guidelines can further enhance the launch's success. According to play.google.com, “your app's store listing is your first opportunity to make a good impression with users and is crucial for app discovery” (Release | Google Play Console, n.d.). To improve this app's launch, recommendations include increasing discoverability with graphic assets, screenshots, and videos, ensuring accurate categorization and ratings, and launching a pre-registration campaign to boost interest. Moreover, optimizing the app’s description and keywords can improve search rankings and engaging with potential users on social media platforms can also create buzz and drive downloads. To ensure a smooth launch, the app must adhere to guidelines and developer policies to avoid removal from Google Play. New apps and updates are mandated to target Android 10 (API level 29) or higher.

The app icon plays a vital role in any application's launch because it conveys ideas and impressions to users as they browse through numerous download options. For example, if an app store lists three different inventory management apps, the icon is as critical as the description in attracting users. For this application, the best representation would be an icon resembling a spreadsheet, suggesting an organized and professional outcome. A visually appealing icon can also leave a memorable first impression, making it easier for users to recognize and recall the app later. Consistency between the icon's design and the app’s overall branding helps establish trust and professionalism. Alongside the icon, a concise yet well-worded description will reassure users that the app delivers value while remaining courteous, efficient, and respectful of privacy.

Deciding which Android version to target involves balancing two key considerations. One factor is the number of users who can download and use the application. Targeting the newest version, such as API 35, limits access to less than one percent of Android users at the time of this writing. Conversely, selecting a very old version can be detrimental; older targets restrict access to modern features and components, such as the newer RecyclerView or sensors like ambient light and proximity detectors. Additionally, older versions may lack support for critical modern security features, which could compromise the application’s safety. For this project, the chosen target is API 29, Android 10.0, which accommodates 81.2 percent of users at the time of writing. This choice strikes a balance between user reach and access to modern features, while also enabling the application to leverage the tools required for optimal performance and an enhanced user experience.

The Inventory Maker application is designed to request only the permissions necessary to deliver its promised features. It requests permission to send SMS text messages, though this is optional; the telephony feature’s required flag is set to false, allowing users to use the app without consenting to this permission. This approach maximizes user freedom and flexibility. The SMS permission enables the app to send low-inventory alerts when an item’s quantity reaches zero, but users must opt in and the app provides clear instructions for disabling this feature if they change their mind, ensuring politeness and minimal intrusiveness. When an item’s quantity hits zero, the app checks if the feature is permitted and sends the message if allowed; otherwise, it silently skips the function and continues. Since the app avoids requesting unnecessary permissions it builds user trust and protects privacy.

According to Buildfire.com, “91 percent of all app downloads were unpaid in 2013” (10 Ways to Effectively Monetize Your Mobile App, 2016). Most users avoid purchasing apps outright due to the availability of free or cheaper alternatives. Therefore, the plan to monetize this app involves incorporating advertising. This strategy allows users to download the application for free while ads generate revenue. Common ad types include full-screen, notification, capture form, advanced overlay, and banner ads. Using a mix of these ensures revenue while keeping the app free. Advertising also broadens accessibility for users unwilling to pay upfront and provides steady income for ongoing development and improvements. To further enhance user satisfaction, the app will offer customizable ad settings, allowing users to adjust the frequency and type of ads they see. Additionally, partnerships with reputable brands for sponsored content can diversify revenue streams without compromising user trust. The app will also periodically review ad performance metrics to ensure they align with user expectations and industry standards. Finally, continuously monitoring user feedback will help optimize ad placement and maintain a positive user experience.

**References**

*10 Ways to Effectively Monetize Your Mobile App.* (2016, April 15). BuildFire. <https://buildfire.com/ways-monetize-mobile-app/>

Release | Google Play Console. (n.d.). Play.google.com. <https://play.google.com/console/about/guides/releasewithconfidence/>

‌

‌

‌