Project Three: Inventory App Launch Plan

Francisco Sousa

SNHU

CS 360

Alex Benavente

10/19/2025

**App Description**

The Inventory Tracker App was developed to help users manage stock levels efficiently while learning how to integrate core Android features such as authentication, data persistence, and notifications. The app enables users to create secure accounts, log in, and maintain an inventory of items. Data is stored in an SQLite database that supports full CRUD operations, users can add, view, update, or delete items directly within the app.

The application also includes a notification system using the Android SmsManager API. When an item’s quantity drops below a threshold (for example, fewer than five units), the app sends an automated low-stock SMS alert. If the user denies the SMS permission, the rest of the app continues functioning without interruption. This design demonstrates proper permission handling and ensures a resilient user experience (Android, 2015).

The project follows Android industry standards, including Material Design 3 interface elements, resource-based string management, and consistent naming conventions for improved readability and maintainability (Google, 2023).

**App Icon & Store Listing**

The proposed app icon will feature a simple, clean purple box or storage bin graphic, symbolizing organization and inventory management, set against a white or light lavender background to match the app’s interface theme. The store description will emphasize simplicity and utility: “Track your inventory effortlessly. Add, edit, and manage items from one screen, and receive SMS alerts when stock runs low.” The listing will include screenshots of the login screen, inventory grid, and SMS permission prompt to clearly display key functionality.

**Android Version Compatibility**

The app is built in Android Studio Hedgehog using Java with the target SDK 34 and minimum SDK 26 to ensure compatibility with Android 8.0 and higher. This range was chosen because it allows testing on current Android emulators while maintaining backward compatibility with most active devices (Android, 2015).

Testing was performed using the built-in Android Emulator configured for API level 34. During debugging, several issues were addressed, including fixing package-path mismatches in DatabaseHelper.java, resolving null pointer checks for input fields, and removing deprecated SmsManager.getDefault() calls by switching to the recommended getSystemService(SmsManager.class) approach for Android 12+ (IBM Developer, n.d.).

**Permissions**

The app requests only the permissions necessary for its functionality:

* android.permission.SEND\_SMS: Required to send low-stock text alerts to the user.
* android.hardware.telephony (optional feature flag): Ensures SMS compatibility on devices with telephony capabilities.

No other permissions (such as RECORD\_AUDIO or ACCESS\_FINE\_LOCATION) are requested, following Android best practices for privacy and security (Android, 2015). The manifest was verified for correctness, and Android Studio warnings related to restricted SMS usage were acknowledged as acceptable for non-production educational use.

**Monetization Plan**

The Inventory Tracker App is designed as a free educational utility, but a future release could explore simple monetization strategies:

* Ad-supported version: Lightweight banner ads for optional revenue.
* One-time paid version: A $0.99 premium tier with ad-free experience and cloud sync.

Monetization is not implemented in the current version but can be added later if needed.

**Testing & Distribution**

Testing was performed iteratively using the Android Emulator. Each feature, login, database CRUD, and SMS permission was tested through real user interactions. Key debugging milestones included:

* Fixing callback errors and resolving mismatched package declarations.
* Addressing ObjectID and NoneType crashes in earlier versions.
* Verifying account creation flow, including input validation for username and password fields.
* Adding a two-step “Register” flow in LoginActivity to improve clarity when users create new accounts.
* Testing both permission outcomes (granted and denied) to ensure the app behaves consistently.

The final version was verified by building and running a signed APK, which confirms successful compilation with no runtime exceptions. For distribution, the app can initially be shared via direct APK download or internal testing on the Google Play Console’s closed testing track before public release (Android, 2015).

**Maintenance & Updates**

Future updates will focus on user interface refinement, potential integration with Firebase or a cloud-hosted database, and expanded notification options such as email alerts. Additional planned improvements include:

* UI redesign for tablet responsiveness.
* Enhanced error handling and logging.
* Regular updates to maintain compatibility with newer Android versions and SDK changes.

Code comments, consistent naming conventions, and modular design principles make the application maintainable and extensible (IBM Developer, n.d.). These best practices follow recommendations from Android Developer documentation, Material Design guidelines, and references such as Plotly’s Dash documentation and Giamas (2022), which emphasize modular coding and data structure design for maintainability (Dash Documentation & User Guide | Plotly, n.d.; Google, 2023; Giamas, 2022).

**References**

Android. (2015). *Permissions Overview | Android Developers.* Android Developers. <https://developer.android.com/guide/topics/permissions/overview>

Dash Documentation & User Guide | Plotly. (n.d.). *Dash.plotly.com.* [https://dash.plotly.com](https://dash.plotly.com/)/

Giamas, A. (2022). *Mastering MongoDB 6.x.* Packt Publishing Ltd.

Google. (2023). *Material Design.* Material Design. <https://m3.material.io/>

IBM Developer. (n.d.). *Developer.ibm.com.* [https://developer.ibm.com/tutorials/j-perry-writing-good-java-code](https://developer.ibm.com/tutorials/j-perry-writing-good-java-code/)/