3-3 Project One

App Development Proposal

Alex Kastigar

7/20/25

CS-360-16014-M01 Mobile Architect & Programming

3-3 Project One: Inventory App Development Proposal

I’ve never built a full app before, but I’m excited by how this Inventory App could keep small warehouses and hobbyists from running out of stock. Here’s my plan in plain terms:

The primary goal is to prevent unexpected stockouts by giving users an easy way to sign in, view every item at a glance, adjust quantities on the fly, and get a notification the moment any item hits zero. To meet this goal, the app will need a local database with two tables, one for user credentials and one for inventory items, and four main screens: login, inventory grid, add/edit item, and settings for alerts.

I see three distinct user types. First, warehouse managers who need a dashboard view of all stock levels and proactive alerts so they can reorder before running out. Second, inventory clerks on the floor who perform rapid, frequent count updates as shipments arrive or orders ship; they’ll value embedded plus/minus buttons for one-tap adjustments. Third, occasional users, like event planners or hobbyists, who track only a handful of supplies and don’t want forced registrations or cloud fees; they need a local, no-frills tool. Across all types, I assume users prioritize speed, clarity, and reliable notifications over advanced analytics or mandatory syncing.

Drawing on Android’s Material Design Layout and Components guidelines, the UI will feel familiar and intuitive. At launch, LoginActivity presents two clearly labeled text fields and a single “Sign In / Sign Up” button, following Material input patterns for consistency. After authentication, the app lands in MainActivity, which uses a BottomNavigationView to switch between the Inventory and Settings tabs. In InventoryFragment, a RecyclerView displays each item as a CardView showing an optional icon, the item name, its current count, and small “-” and “+” ImageButtons in the card footer. A FloatingActionButton in the corner opens EditItemActivity, where an EditText for name and a numeric input for quantity fulfill item creation or editing. The SettingsFragment offers a SwitchCompat to toggle zero-stock alerts on or off, stored in SharedPreferences. Users move between screens simply by tapping tabs, the FAB, or card buttons, creating a smooth navigation flow that follows Android best practices.

Under the hood, the app follows a lightweight MVVM approach. A single Repository class mediates all data operations via Room, mapping two entities (User(id, username, passwordHash) and Item(id, name, quantity, iconUri)) to SQL tables. UI components ask the repository for data, for example “getAllItems” or “authenticateUser,” and the repository returns LiveData streams that InventoryFragment or AuthViewModel observe. When a user taps “+” or “–,” the fragment calls repository.updateQuantity(itemId, newCount), which writes to the database. Because LiveData is observed by the RecyclerView’s adapter, the list refreshes automatically, with no manual adapter juggling needed. On login, the repository either finds or creates a user record and then signals success to open MainActivity. For zero-stock alerts, the repository detects when an update sets quantity to zero and invokes Android’s NotificationManager on a pre-registered NotificationChannel so the user receives a push notification even if the app is backgrounded.

By mapping each screen’s UI components to clear data flows and following Material Design and Android quality guidelines, this Inventory App will deliver a cohesive, user-centered experience that keeps stock management simple, transparent, and reliable.

References:  
Google LLC. (n.d.). Design & Plan. Android Developers. Retrieved July 20, 2025, from <https://developer.android.com/design>