**Name**: CongS Stock Control

**Purpose**: Track, manage, and update inventory items from a mobile device

The goal of the application is to create a mobile solution that allows users to track item stock levels, manage restocking alerts, and maintain real-time visibility into inventory activity- whether it’s for warehouse, retail space or home office. The app will improve accuracy, reduce manual errors, and streamline logistics for individuals’ users and teams.

**Major Components**:

* Home/Dashboard: displaying stock overview and urgent alerts
* Item Detail View: view/edit information for each inventory item
* Add/Update Item Screen: Input new stock, edit quantity, or delete items
* Notifications Center: Sending alerts for low stock or new shipment arrivals
* Settings/ Profile: Custom user preferences, themes, and log out

**Core Functionality**:

* Add/ Edit/Deleted Inventory Items
* Set restocking thresholds
* Trigger alerts based on stock levels
* View real-time data summaries
* Optional Cloud Sync for multi-user access

The users of the application will be suitable to small/ big business owners or hobbyists, who want an easy way to track inventory and receive stock alerts, and inventory staff or admins who need deeper insights, multi-user access, and analytic tolls. The users benefit from a simple, user-friendly interface that enables fast data entry, clear visual alerts, secure login, and smoot navigation with minimal effort. With the application user-friendly, it will be easy to train each employee who will be in charge of inventory control.

Screens & Features for User-Centered Design

Screen Description Key UI Components

|  |  |  |
| --- | --- | --- |
| Login Screen | User authentication | Text fields, login button |
| Dashboard | Summary of items + alerts | Cards, icons, navigation bar |
| Item Detail | View/Edit one item | Text views, edit buttons |
| Add Item | Enter new inventory | From fields, save button |
| Notification Screen | View stock alerts | List view, actions buttons |
| Settings | Edit profile and preferences | Toggles, drop downs |

**Diagram Explanation**

The app starts with the **Login Screen**, which takes users to the **Dashboard** once authenticated. From the Dashboard, users can view specific items, add new items, check stock alerts. Users can update item information or cancel changes and settings are accessible from Dashboard or nav bar, allowing profile edits.

*Navigation Flow Diagram (High Level)*

***[Login Screen****]*

*[****Dashboard****]*

*[****Item Detail****] [****Add Item****] [****Notifications****]*

*[****Edit Item****] [****Save****/* ***Cancel****]*

*[****Back to Dashboard****]*

*[****Settings/Profile****]*

*\*Bottom nav bar allows fast switching between screens*

Key UI Components & Their Data Roles

Component Data Type Source/ Destination

|  |  |  |
| --- | --- | --- |
| TextView (Dashboard) | Displays quantity/ alerts | Retrieved from local DB |
| EditText (Add Item) | Accepts user/input | Sent to ViewModel |
| Button (Save Item) | Triggers add/update call | Invokes logic in ViewModel |
| ListView (Notifications) | Displays alert items | Fetched based on threshold check |
| Spinner (Settings) | Displays Options | Sets user preferences |

Functional Requirements in Code Design & UI Connectivity

To represent functional requirements, this inventory app uses a modular, event-driven architecture that connects user input, business logic, and persistence for data storage.

* Java classes model inventory items (Item.java) and handle logic through a **View Model** (InventoryViewModel.java)
* Layout Files (XML) define each screen’s structure and interactive components
* Calls between UI and backend use **listener methods** and **ViewModels functions** to update and retrieve data in real time
* Data storage is managed by a **Repository** using local storage tools like Room or SharePreferences

Data Flow Between Code & Screens: User Input 🡪 ViewModel 🡪 Repository 🡪 Database

UI Components by Screen & Their Data Roles

|  |  |  |  |
| --- | --- | --- | --- |
| **Screen** | **UI Component** | **Purpose** | **Data Source/Input** |
| Login | EditText (username)  Button (login)  TextView (Error msg) | Accept credentials  Authenticates user  Show login failures | User input  Auth method  ViewModel feedback |
| Dashboard | RecyclerView  FloatingActionBu  TextView (summary) | Displays item cards  Nav to add item screen  Total invent. Count or alerts | DB via ViewModel  UI Trigger  Aggregated DB Data |
| Add/Edit Item | EditText (name/qty)  Button (save/cancel) | User inputs new item data  Save or Discard changes | Manual input  Triggers Logic |
| Notifications | ListView  Button (resolve alert) | Displays low-stock alerts  Updates item status or triggers reorder | DB query results  DB update |
| Settings | Switch/Spinner  TextView | Changes user preferences (theme, threshold)  Displays current settings | SharedPreferences or DB  DBor Preferences |

Each component either ***accepts input*** (via form fields or toggles), ***display output*** (via text, list views, or alerts), and ***Triggers methods*** to move data between UI, ViewModel, and database layers.