GitHub Username: TheBaileyBrew

Inventory Manager

Description

Inventory Manager is a simple Android app, intended for managing all your stock & inventory needs. This application allows users to create all the details of their inventory item, and manage levels with clean visual guides. The application stores data in a cloud server and can be accessed from multiple devices via unique user logins. Each user utilizing the app can be assigned a role and have access to different data or views.

Intended User

The intended user for this app is anyone who has a collection of items to manage in an inventory and wants to keep track of usage and be notified when it's time to reorder supplies again.

Features

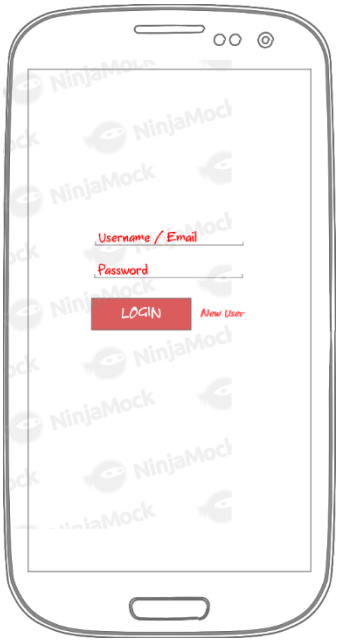
- Saves inventory levels

- If items have barcode, barcodes can be scanned and attached to individual items in inventory
- Visual representation of stock levels

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1 - LOGIN

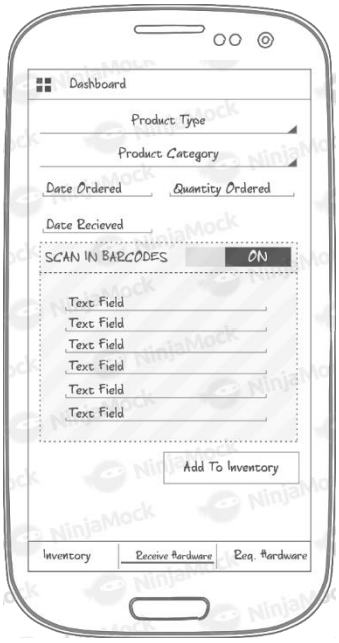


The Login Screen will be the start screen, and where all the network access begins. The users can login or create a new user if they've never used the application before. Each user will be validated against the Firebase Server.

Screen 2 - DASHBOARD

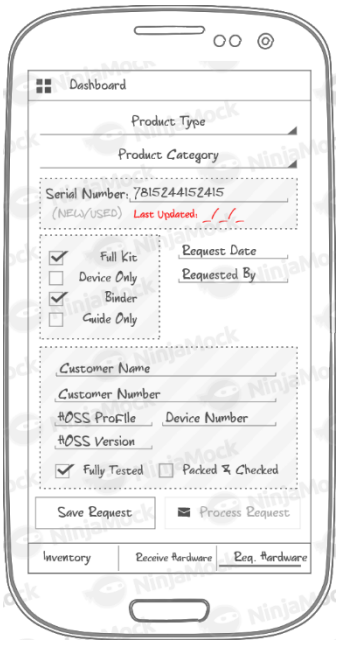
The Dashboard is the primary screen with basic details about the inventory itself. This view shows current quantities of all inventory items – along with a percentage of how much over minimum we have.

Screen 3 - RECEIVE INVENTORY



This view will act as a check-in for inventory. As new shipments are received they can be checked in and added to inventory. *Barcode scanning is a potential future enhancement*

Screen 4 - SEND EQUIPMENT

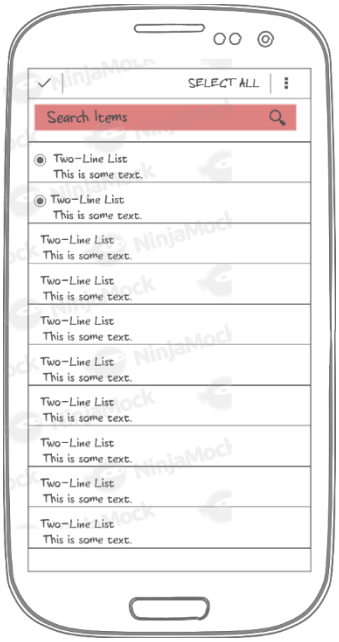


The Request screen will be used for checking out equipment. As an order is being processed, the report can be saved locally to the device and the inventory record altered. *Future Enhancement* Once the order is complete an email can be generated for a recap of the order.

Screen 5 - INDIVIDUAL CATEGORY INVENTORY

This screen will act as a history record of the item category in question. View will display current total, as well as new/refurbished. Last ordered and a recycler of past orders.

Screen 6 – INVENTORY AUDIT



As a service to maintain inventory, there will be the ability to search for and delete items that no longer exist in inventory. This is also a screen where changes can be made to individual items.

Key Considerations

How will your app handle data persistence?

Data for user details will be stored locally via Room. Data for inventory will be serviced by Firebase (with local Room replica for offline use).

Describe any libraries you'll be using and share your reasoning for including them.

Picasso (Displaying images)	V – 2.71828	Picasso is one of the simplest utilities for loading network images

FlowingDrawer	V – 2.0.0	Customizable Drawer-style fragment that hovers and animates with drag.
Android Process Button	V – 1.0.4	This is a button object with a builtin progress bar. Using this for the login component
Material-BottomNavigation	V – 2.0.1-rc1	For main fragment display navigation. In the Flowing Drawer, I've built the screen in sets of 3 or 4. When the grouping changes the bottom-navigation changes as well to allow for quick navigation through available fragments.
Zxing (Google Camera) **Potential**	Native implementation	Zxing is a camera extension that allows for barcode/qr scanning. This will be utilized when scanning in orders of supplies.

Describe how you will implement Google Play Services or other external services.

Firebase Storage, Authentication + Cloud Messaging

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

First, I need to prepare the framework designs, and work through the logistics of how my app should behave. The app will be written fully in Java and incorporate several external libraries for data modeling.

Task 2: Implement UI for Each Activity and Fragment

- Create the Login Activity + Logic
- Create the Main Dashboard
- Develop the UI for breakaway screens

Task 3: Work out backend logic

- Incorporate libraries and external resources
- Link Firebase
- Set up Room Instance + ViewModels

Task 4: Debug & test

- Test on multiple devices, validate onSave/onRestore