Development Proposal – Inventory App

As previously discussed, our stakeholder is looking for an app to track items within their warehouse. This document will discuss the goals, users, features, and desired functionality of the app in order to align with the stakeholders' expectations for the app.

Goals

The application will be used to track inventory within a warehouse. The application must be simple and intuitive so as to facilitate a smooth transition integrating the application into the warehouses current system. We've identified the following key elements that will be useful in implementing the apps required features:

- A login/sign up screen to login and/or register new users
- A grid-based view that displays all inventory items
- A database containing both inventory and user credential tables

The available inventory within a warehouse is constantly changing, thus the app should provide the user the ability to dynamically alter inventory by adding or removing items from the system and increasing or decreasing the amount of available inventory. A user must also know when a particular inventory item is out of stock. This will be addressed by adding a notification feature that alerts a user when any inventory items stock count drops to zero.

User Assumptions

Potential users of this application share the common interests of knowing where a particular piece of inventory is, knowing how many are in stock, and knowing if stock has run out. With this in mind we can break down potential users into four categories: warehouse workers, administrators, vendors, and customers. Warehouse workers and administrators would naturally have higher privilege in terms of adding, removing, and updating inventory. Vendors should only be authorized to view and update a subset of inventory, specifically what they are responsible for bringing in and out of the warehouse. Customers, similarly, only need to view a subset of inventory for the sake of having visibility of their orders across the supply chain, possibly through a secure API to prevent unintended access.

UI Features & Design

The user interface of the inventory application will consist of 5 screens: a user login screen, a registration screen, an inventory display screen, an add new inventory screen, and a messaging permissions prompt (for SMS based notifications of low inventory). Naturally users

will begin at the login screen with the option to either register or enter their credentials to login.

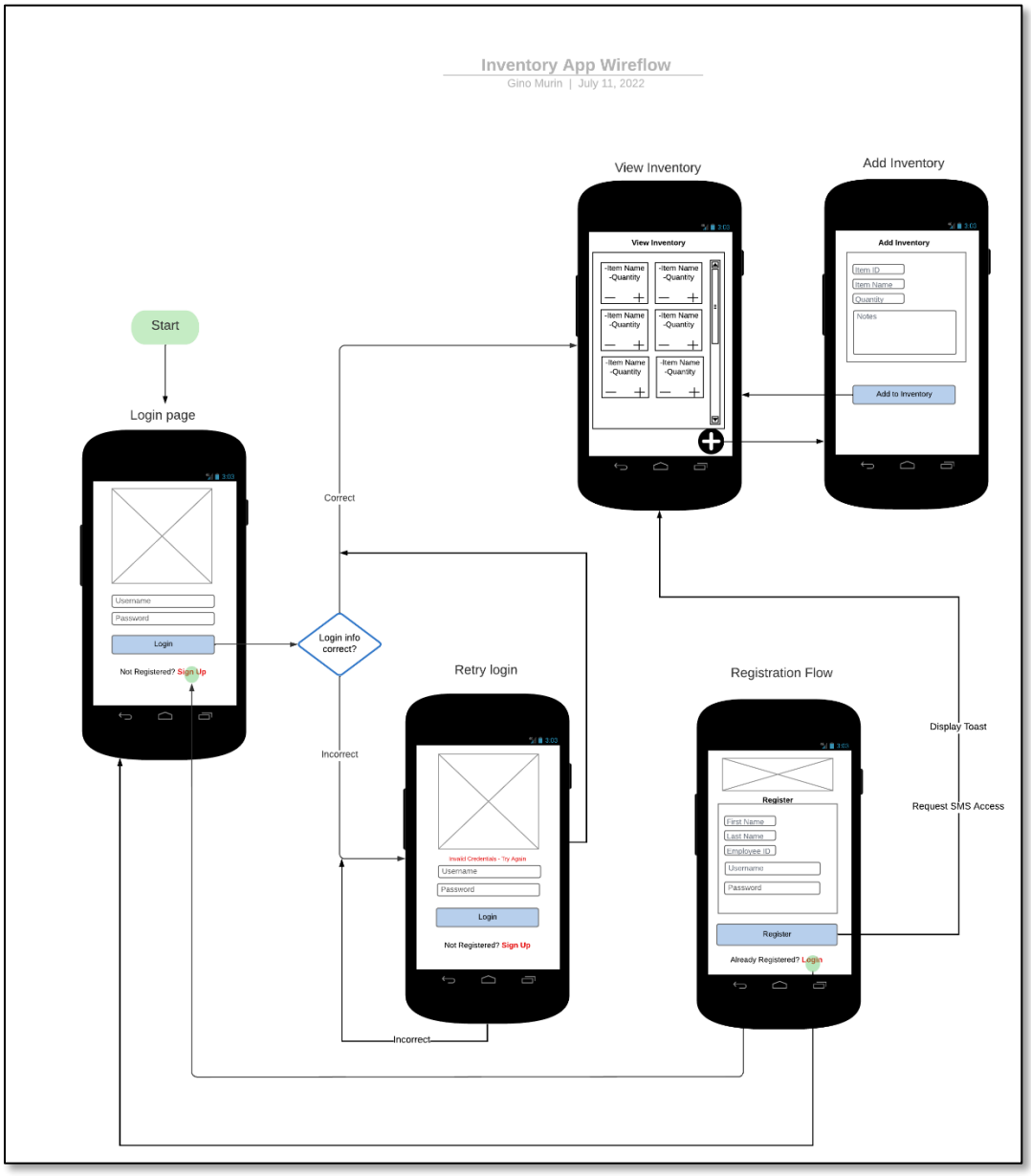

Figure 1-Wire flow Diagram of Proposed Login, Registration, and Inventory Pages

If the registration button is selected, the user will be led to the registration screen, where they will enter their new user information. Upon hitting the register button, the user will be led to a page that request to enable SMS permission. From here, a toast will display stating that the profile was created successfully while the user is led into the main inventory view.

The main inventory view provides a scrollable grid of card views, each displaying an inventory item along with its associated quantity. The user can add or remove units by hitting

the plus and minus buttons on each card. Hitting a plus or minus button will trigger an alert popup to specify the exact number to increment/decrement the inventory by. Pressing the plus at the bottom right hand of the view inventory view as shown in Figure 1, will lead the user into the add inventory view. This view consists of a group of text edit fields to specify item attributes and a submit button to add the item to inventory. Clicking this button will either trigger an error if the user entered incomplete information or it will generate a toast to confirm success while subsequently leading the user back into the main inventory view.

Coding Functional Requirements

The inventory application will utilize an MVC architecture. In MVC, the Model is responsible for handling business logic which in our case consists of our users, inventory, and the behaviors associated with each (adding, removing updating, etc.). The model will store and manipulate this data as necessary. The View is comprised of the user interface that we have discussed in detail above. The elements in the View are what the user sees and interacts with in order to send and receive data. The Controller acts as the intermediary between the View (UI) and the Model (business logic), listening for users to trigger events within the UI. When an event is triggered such as a button click or entering a text field, the Controller calls an associated function to handle the event.

As an example of MVC in action within our app, the login screen will have listeners attached to the username and password fields as well as the submit and register buttons. Entering text into the text fields will trigger the controller to save the state of these fields to be passed into the model once the click event that attached to the login button is triggered. The model will compare credentials against the database and pass the results back to the controller which will either prompt the view to either output an invalid credentials message or to display the main inventory view.

Summary

The stakeholder has defined a very specific set of goals to be achieved in implementation of this mobile app in their warehouse. The app should be simple, reliable and secure so as to meet the demand of their busy warehouse workers and other users. Implementation of the proposed features using and UI design should allow us to meet and exceed customer expectation.