CS 360 - Mobile Architect & Programming

3-3 Project One - Inventory App Development Proposal

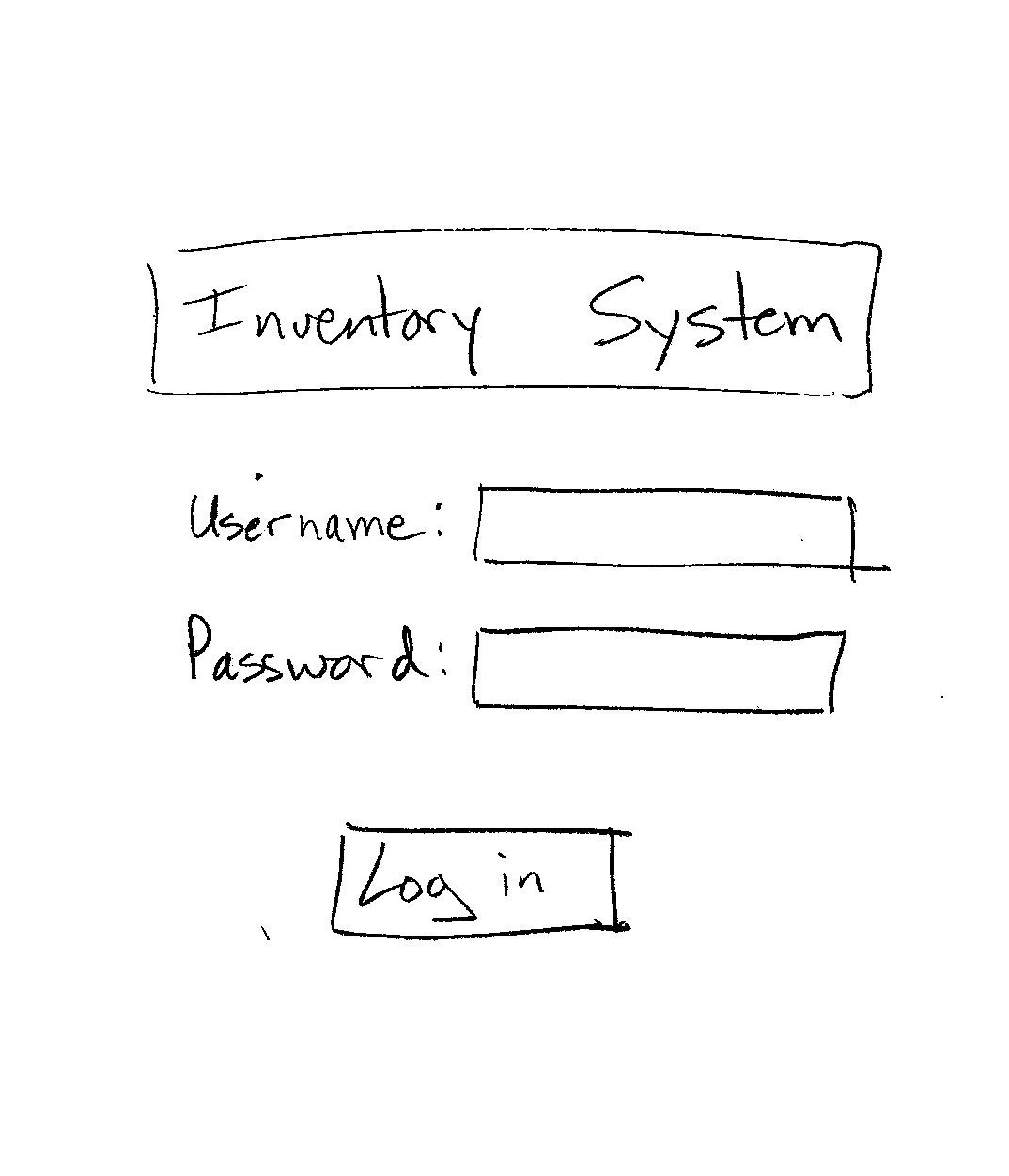
Thomas Vaughn

11/13/22

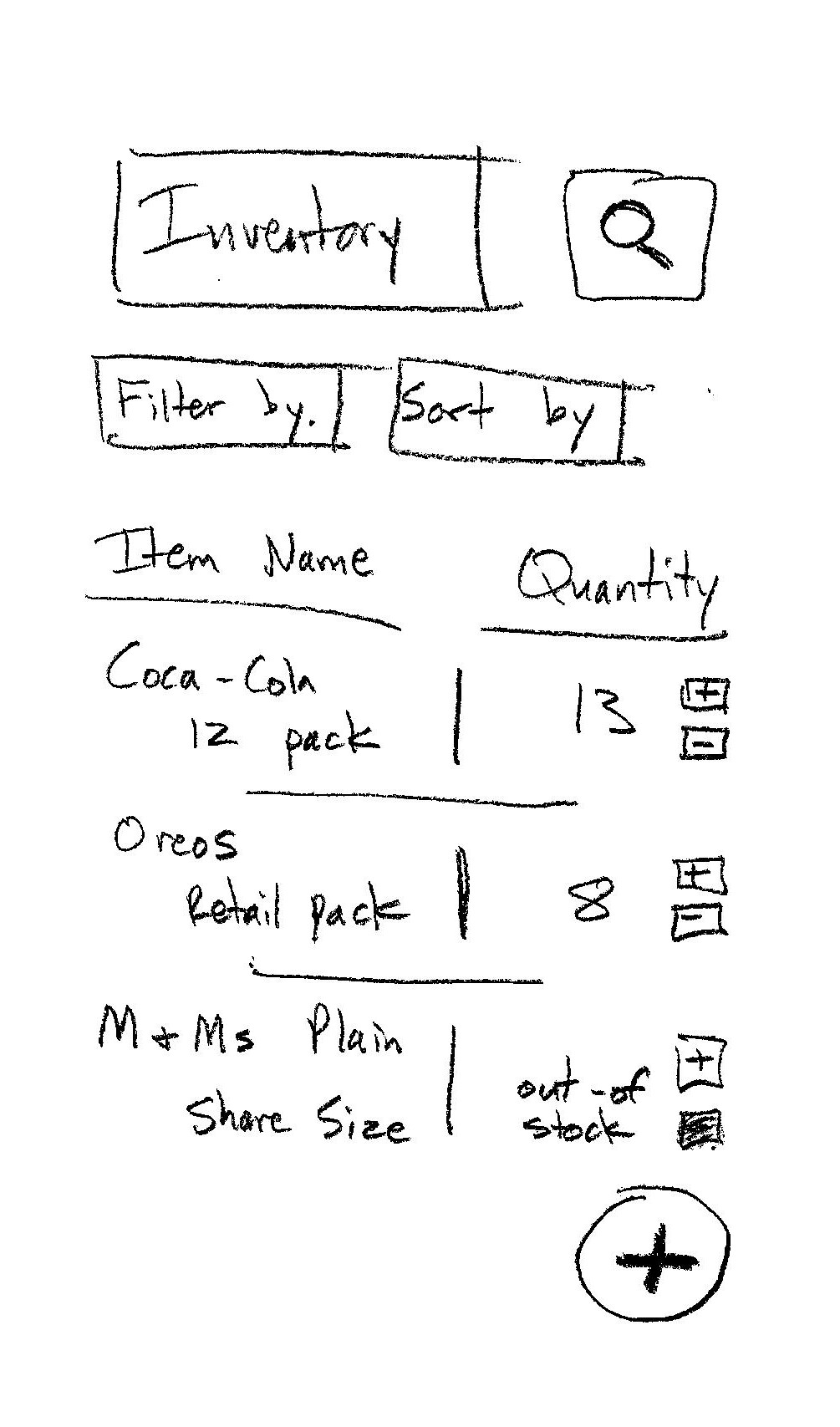
I have chosen to work on the Inventory Application the primary goal of which is to effectively and efficiently track items in a warehouse. This app will rely on a database with at least two tables, with one of the tables dedicated to logins and passwords and the other table dedicated to the inventory item data. The first primary requirement is a login screen, and we can handle that with a pair of EditText components prompting the user for their username and password and then a button for logging in. The second primary requirement is a screen which displays the inventory item data in a grid. We can use RecyclerView and the LinearLayoutManager or GridLayoutManager to not only provide the user with a useful visual arrangement of data but also to improve the performance of data retrieval and display to the user. As for the data model itself, depending on the amount of data and additional requirements, we might be able to use the Room persistence library. We can also create buttons or other interactable widgets for adding new items to the inventory and for removing items from the inventory. We would also need a button or comparable widget to update the quantity of items in the inventory. Then we could use a toast message and possibly a push notification to notify the user when the quantity of any item is reduced to zero. And to provide additional functionality to help users meet their goals, we would also implement a Search screen using EditText components, and we would implement either additional screens or possibly PopupWindows for the sort, filter, and add / remove item functionality. Efficient access to the inventory data and an effective implementation of the required functions as proposed above should meet the user’s needs quite well, thus meeting the goals of the development project.

Three user types have been identified for this app, and they are as follows: a Sales Representative, an Inventory Control Specialist, and a Purchasing / Replenishment Specialist. Each user type would have goals tied to their specific role in the organization with some crossover in between. A Sales Representative would want to know which items are currently in stock in order to focus on available products. In order to achieve that goal, they would likely need to view, filter, sort, and search the data grid to see which items were in stock. An Inventory Control Specialist would also want to be able to view the items which are in stock, but they would also want to add new items, remove items, and update item quantities. They would need the ability to view, filter, sort, and search the data grid, and they would also need easy ways to add, remove, and update the quantities for items in the inventory. These additional functions would allow Inventory Control Specialists to effectively update the inventory data to match the actual items in stock. A Purchasing / Replenishment Specialist would want to view, filter, sort, and search the data grid, but they would be especially concerned with the zero quantity notifications since it is their responsibility to purchase the items which are stored in the warehouse and sold by the sales reps. Like the other user types, they would need to view, filter, sort, and search the data grid, but receiving zero quantity notifications in a clear and effective manner will help meet their goals of maintaining stock in the warehouse.

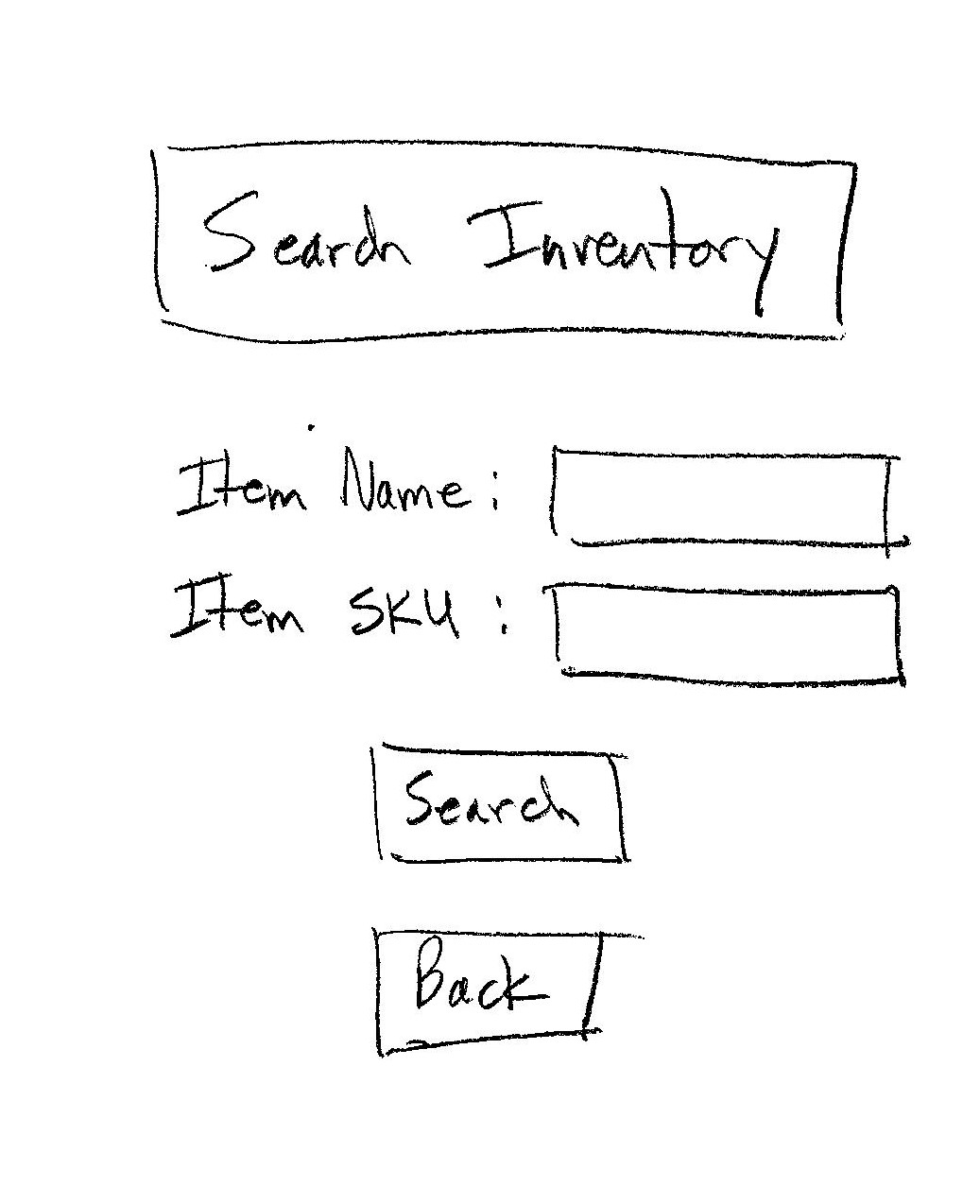
When a user first opens the app, they will be presented with the Login Screen. There will be an EditText for their username and another EditText for their password. After they enter a valid username / password combination, they can click the Login button to proceed.



Upon successful login, the app will present the user with the Inventory Screen where the user can view inventory data in a grid. The Inventory Screen will use a RecyclerView to retrieve and display data quickly and efficiently. It will be determined during the early stages of development whether the Inventory Screen will be implemented using a LinearLayout or GridLayout, but the primary requirements are that the Inventory Screen will display the inventory item data in a grid. The screen will include a search button, plus and minus buttons next to each item in order to increase or decrease the quantity, and a floating action button toward the lower right of the screen where the user can add a new item or completely remove an existing item from the inventory. The mockup below uses an ‘Add’ drawable for the add / remove button, but we may use a different drawable. Over the course of development, we will explore implementations of Sorting and Filtering functionality, although these functions may need to wait for a later release.



Although not a primary requirement of the app, it seems extremely likely that a user would want to be able to search for products in the inventory system, so we plan to include that functionality. We would include a search button on the Inventory Screen which would allow the user to navigate to the Search Screen. The Search Screen would then include an EditText for the item’s name and an EditText for the item’s SKU / Item Number. The logic would only require one of the fields to be populated but not both, which would allow a user to still search the inventory even if they only knew the item name or only knew the item number.



For all screens implemented in the app, we will follow Android Design and Quality Guidelines, placing an emphasis on consistency, reliability, and performance. For example, all screens will be implemented to maintain state data after configuration changes such as changing from portrait to landscape orientations as well as designing the primary screens of the app with defined portrait and landscape views, see Visual Experience / UI and Graphics / VX-U1 of Android Developers’ *Core App Quality*.

The functional app requirements will be represented as clearly defined activities in the app with specialized methods designed to handle the interaction between activities. The moment the user opens the app, they will be presented with the login activity. The login activity will be tied to two EditTexts used to collect user input. The data from the user will then be passed to a validateUser method to query the login table in the database. Upon successful authentication, the app will proceed to the Inventory Screen. The data displayed in the inventory grid will be retrieved through a readAllData method to run a select query against the inventory table in the database. The readAllItems method will be called in the InventoryActivity’s onCreate method. Updating item quantities will be managed through UI elements like buttons and PopupWindows or possibly separate activities which will be tied to method calls to run update queries in the inventory table in the database. Accessing the add and remove item functionality will be similarly handled through buttons and PopupWindows or separate activities, but they will be tied to method calls to run insert and delete queries in the inventory table. The SearchActivity will prompt the user for input using EditText components, and that user data will be passed to a readItem method to run a select query against the inventory table. The app will then load the Inventory Screen displaying item data based on the results of the query. Lastly, the app will also monitor item quantities and send both a Toast and a push notification to all relevant user-types when an item quantity reaches zero.

By implementing the features and UI elements outlined in this proposal, we will develop an app which meets functional requirements in a reliable, efficient, and user-focused way, equipping multiple user types to meet their unique business goals and therefore adding significant value to the larger business organization.

References

*Core app quality*. (n.d.). Android Developers. https://developer.android.com/docs/quality-guidelines/core-app-quality