**Project One - Option 1: Inventory App.**

**Goals of the project.**

I travel quite a bit every year, and every time I do, I forget something. A simple inventory application will allow me and other users to log-in, create a username and password, create a new inventory, add or remove items from the inventory, and receive notifications when items have run out, such as toiletries or perishables. This will enable users to pack their belongings for travel, go grocery shopping, or just keep track of important items in an easy and quick manner.

**Users’ needs and preferences**.

Anyone who needs to keep track of their belongings, whether for travel, shopping, or keeping inventory at home or work, can benefit from utilizing a simple inventory application.

The primary goal of this inventory application is to display a quick visual reference for all items in inventory, and for the user to modify that inventory quickly and intuitively. Additionally, the application will send notifications when an items’ quantity reaches zero. This can help users either replenish the item or act upon receiving this information, depending on their goals.

Users already familiar with simple GUIs will have an easy transition into using the application because the mechanics, icons, and descriptions in the user interface will be rather generic, making it seamless to use.

**Application’s screens and navigation flow.**

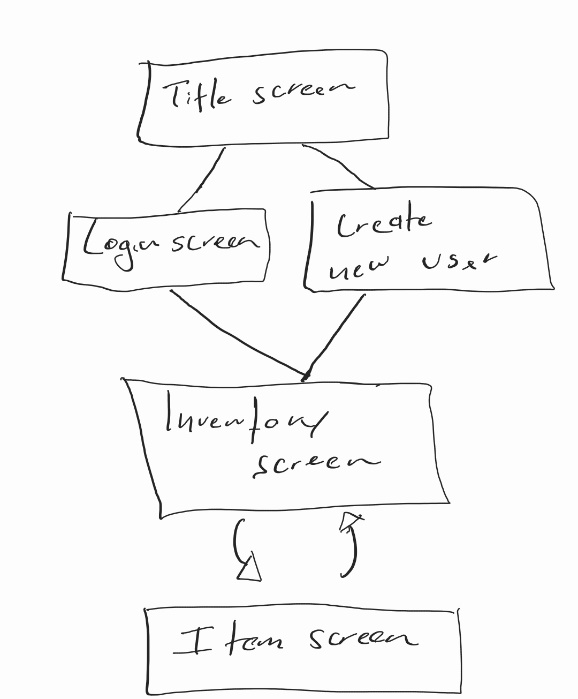
****

Figure 1- App navigation flow diagram.

The initial draft of the application contains five screens: A welcome title, login, create new user, inventory, and item screens.

A picture containing text, sketch, handwriting, drawing

Description automatically generated

The Welcome screen welcomes the user and has two buttons:

* Login: Sends existing users to “Login screen”.
* New User: Allows new users to create a username and password in the “Create new user” screen to protect their inventories from unauthorized access.

Figure 2- Welcome screen.

A close-up of a login form

Description automatically generated with medium confidence

Figure 3- Login screen.

The “Login screen” will have a title image, several view text objects to prompt the user for username and password input, as well as dynamic text to notify users when the incorrect password or username has been entered, and a button to log in.

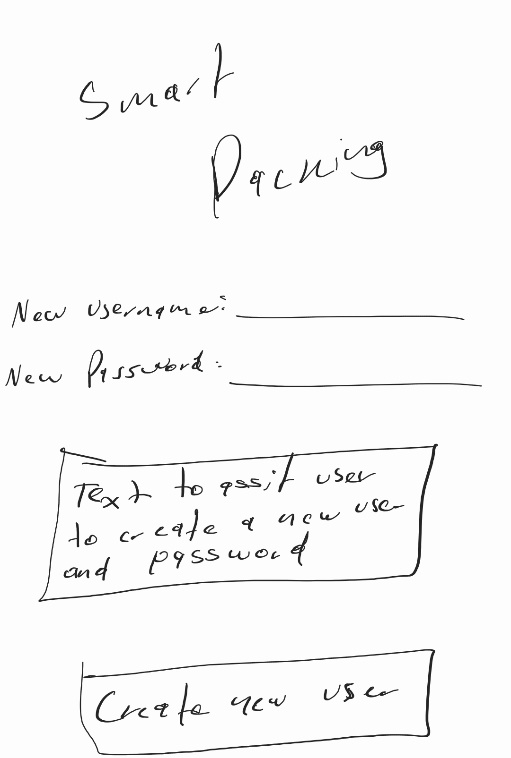


Figure 4- Create new user screen.

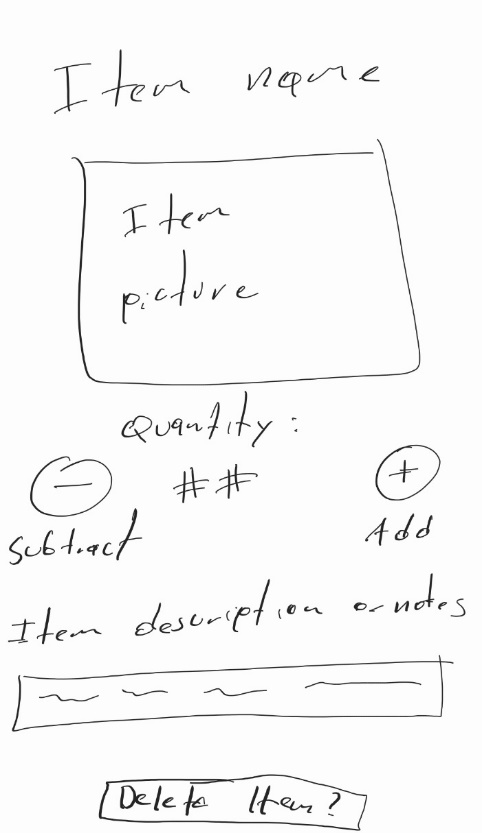
The “Create new user” screen is almost identical to the “Login screen” with the exception of the suggestions to prompt corrections from the user when entering an invalid username or weak password. Also, the button at the bottom creates the new user and logs in.

A picture containing text, handwriting, sketch, drawing

Description automatically generated

Figure 5- Inventory screen

The “Inventory screen will display the name of the inventory at the top, and divide the items displayed in a grid layout. Each item is selectable and will send the user to the “Item” screen for individual item modification. A button at the bottom right of the screen will enable users to add items to their inventory. A grid layout will enable users to see all items in their inventory to add items quickly.



The Item screen will display the name of the item at the top, a picture of the item below it, the quantity with a subtract button to the left and an add button to the right. A description of the item or notes can be entered in the lower section, as well as a delete item button.

Figure 6- Item screen.

**App functionality requirements.**

Text components used are TextView for most informational text and labeling, EditText for user input, including creating a username, password (concealed with textPassword input type), inventory and item creation. Button will use a standard button widget and for images, the ImageView widget should work. A simple ConstraintLayout will be used for most screens, and the GridLayout will be used for the inventory screen.

All these widgets will trigger their own methods to navigate from one screen to the next, pass data input from each view to the controller, then finally handled by the model, which will encapsulate most logic and calculations. A more concise list of methods will be created upon the initial layout of the user interface.