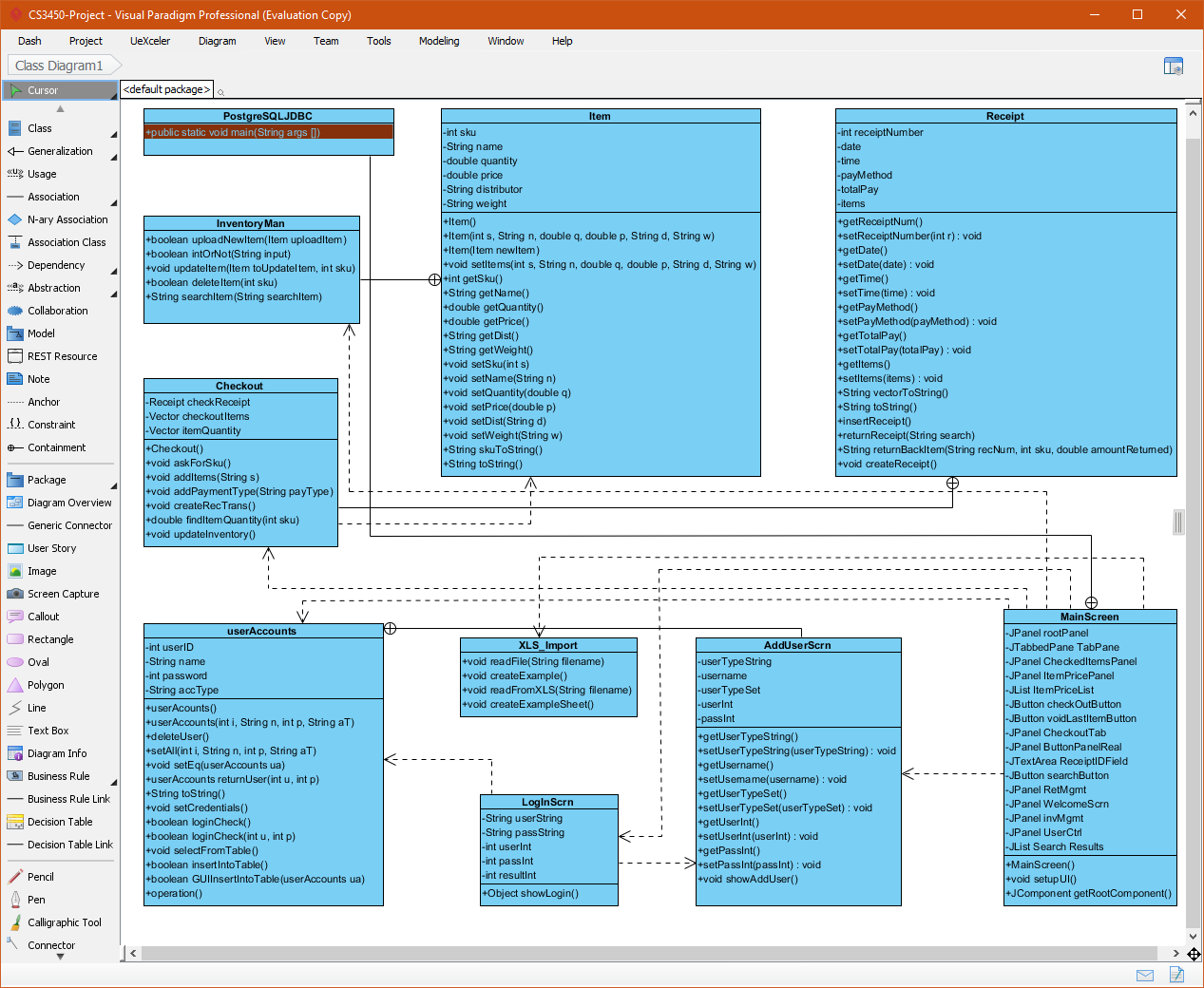
**Mr.Smith Grocery Application:**



**Class Descriptions:**

* AddUserScrn:
  + Creates a JFrame that receives input from the user to create a new instance of a user based on information from the user.
* Checkout:
  + Creates an instance of a receipt and a vector of items. This class handles the main checkout functionality such as: adding items, adding payment method, updating inventory, and finding quantity of items, all of which are processed through the SQL database.
* InventoryMan:
  + Allows the user to upload, edit, and delete items from the SQL database. Also includes functionality for other classes that call InventoryMan.
* Item:
  + Creates an instance of an object called item that contains a sku, name, quantity, price, distributor, weight. Has functionality that is utilized when other functions call Item.
* LoginScrn:
  + Allows the class that calls this function the ability to view the login screen at boot when the application starts. Allows for the checking of credentials upon login.
* MainScreen:
  + Main controller for all graphical interfaces and button objects. Contains instances of other classes that are tied to the button action listeners.
* PostgresSQLJDBC:
  + The main controller for the mainscreen class that calls mainscreen to display for the applications. Main controller for all functionality within the application.
* Receipt:
  + A class object for creating and manipulating information for a receipt object which contains a receipt number, date, time, pay method, total pay, and items list.
* XLS\_Import:
  + A class that handles the instances of importing from an excel spreadsheet and creating report sheets for the user.

**Architecture of Mr.Smith Grocery Application:**

**Model-View-Controller Pattern:**

With the architecture of Mr.Smith Grocery Application, It utilizes the model-view-controller pattern. The way the classes are divided in the project are in a fashion that there is a section for the main controls the application, a section for the basic structure of the application, and a section for the design/view of components of the app. Some instances that follow the model portion of the pattern consist of the: Receipt, and Item. These classes utilize the model pattern due to the fact that they are a main base structure and a foundation that is built upon in some of the other classes, for instance, the InventoryMan class utilizes and contains an instance of the item class and utilizes the functions/methods that are within the item class. The classes that implement the view pattern within the MVC include: LoginScrn, MainScreen, and AddUserScrn. These classes utilize this view pattern due to the fact that they control the implementation and structure of how GUI is displayed, what buttons go where, and how they operate. Lastly, the functions that include the controller pattern implementation are: PostgreSQLJDBC, MainScreen, and InventoryMan. These classes implement from the original userAccount and Item classes and either control the button layout, the button input and output, or functionality within the view classes. These classes make constant calls to the other controller classes for their functionality.

**Multi-Layer Pattern:**

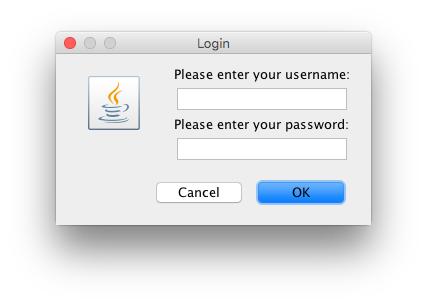
Inside the architecture of Mr.Smith Grocery Application, the classes are designed in another way that can utilize the multi-layered pattern, in which there are several classes that need to be tested individually and each class can be grouped into different modules for a group of services. In the application, the main layers consist of: GUI, Inventory control, Receipt Control, and other. Each layer plays a vital role in which each is attributed to another class in another way that they utilize the functions called in each other.

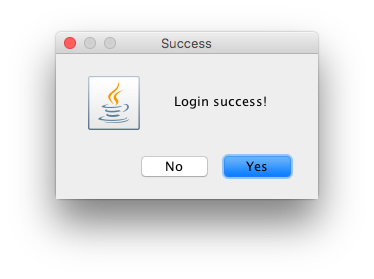
**Client-Server Pattern:**

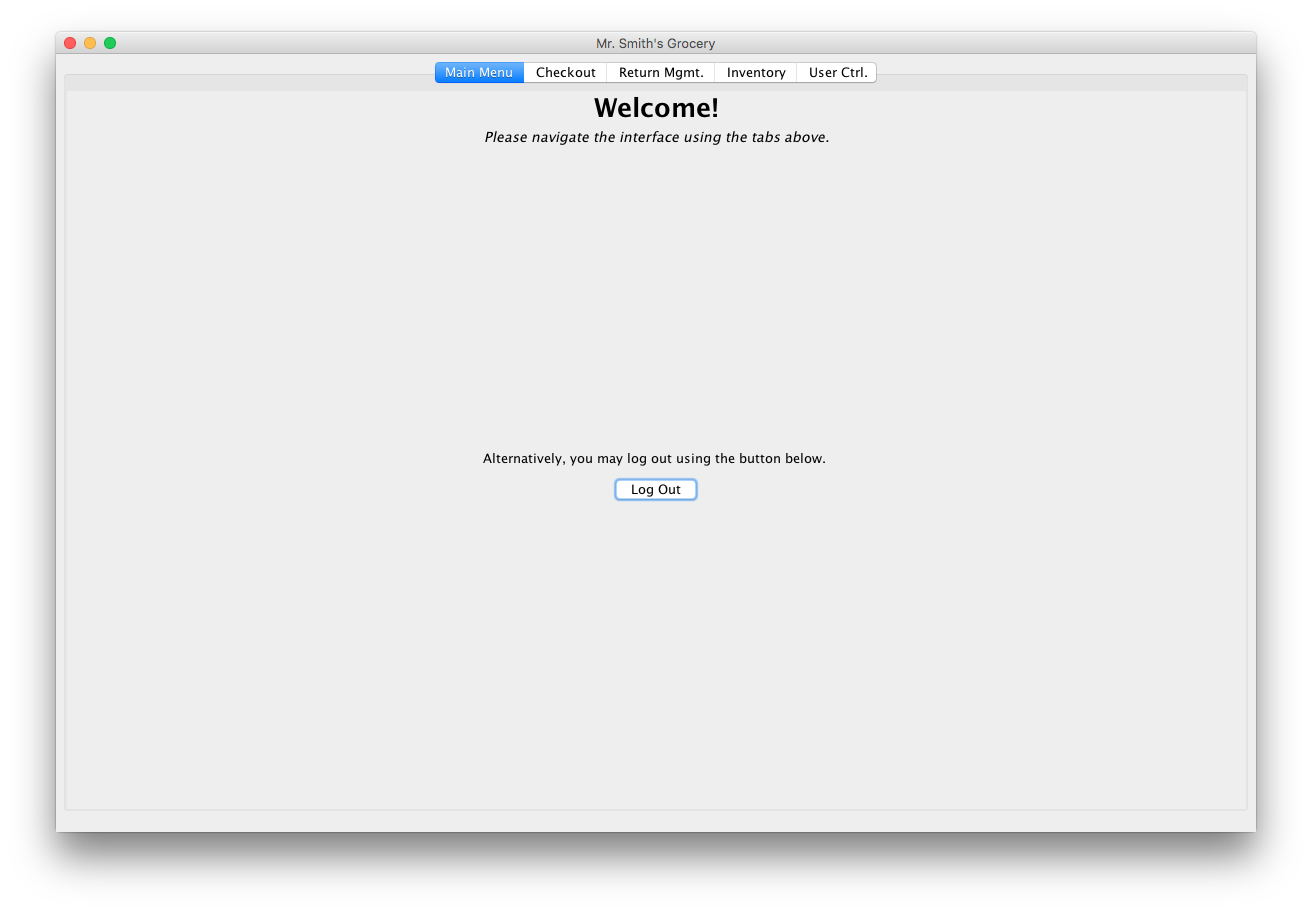
Mr.Smith Grocery Application utilizes the Client-Server pattern in only a few instances that include: Login into the server for user account access, and creating/updating/deleting items, receipts, and users from the server. Each classes throughout the application utilizes this pattern in a way that they make calls to the SQL database and return or create queries to this database. These classes all make calls to the server client and receive input and output for both the login services to client accounts. The main example is that the InventoryMan class makes constant requests to the server for creating and getting Item info for its methods that are implemented in its class.

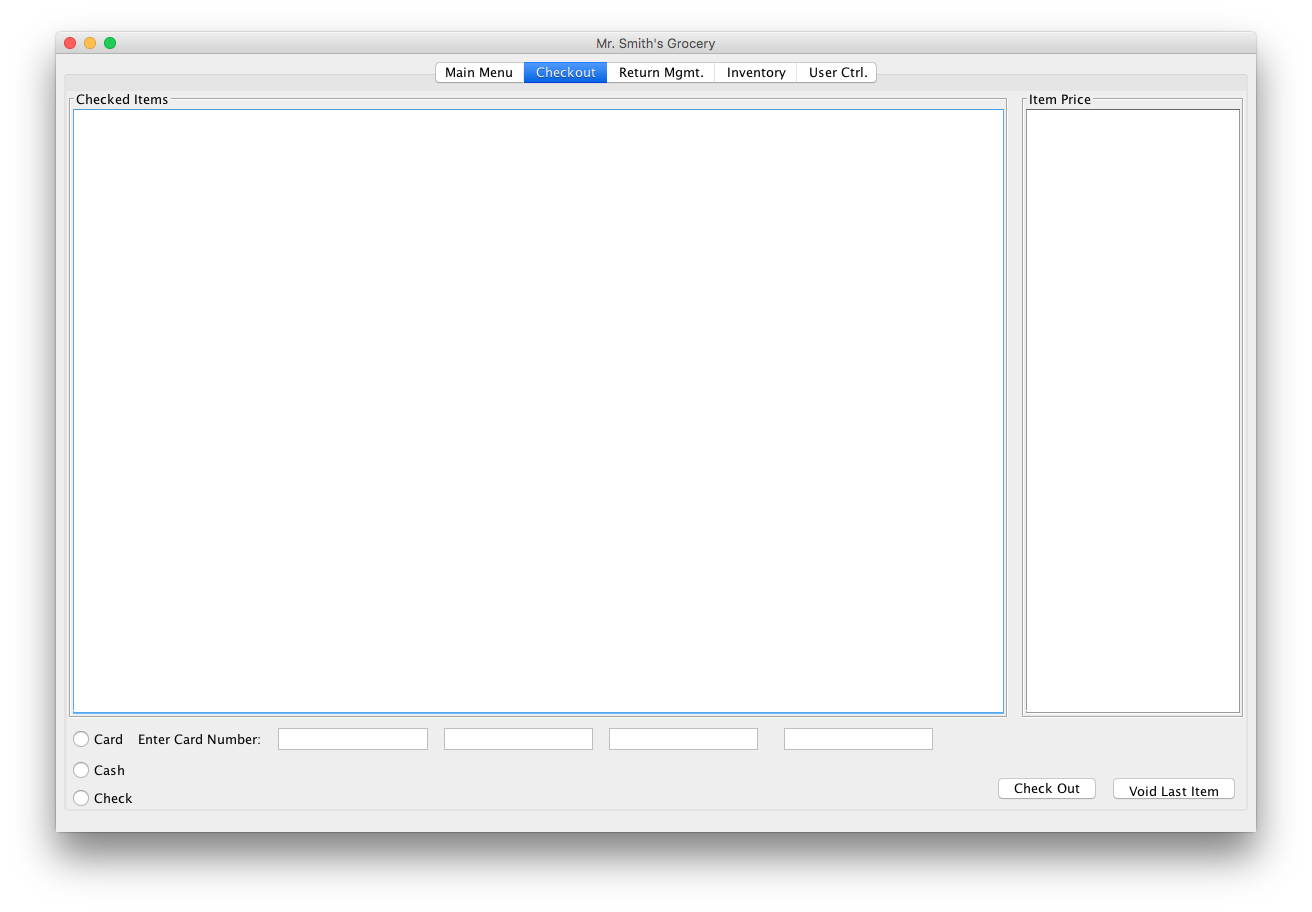
**Service-Oriented Pattern:**

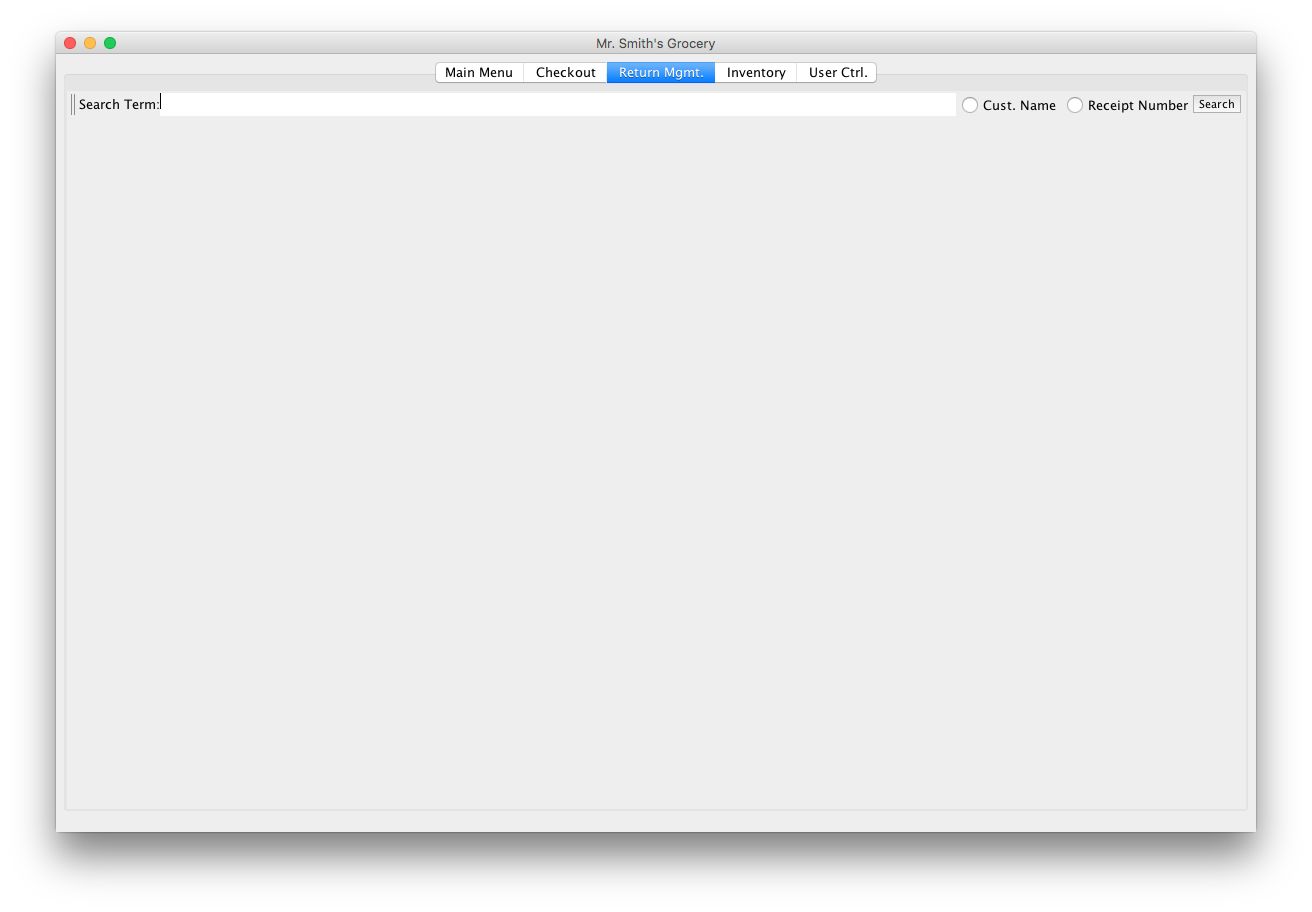
Mr.Smith Grocery Application doesn’t utilizes the service-oriented pattern due to the fact that the application doesn’t have services and websites throughout the web that the application access or runs through. Service-Oriented Architecture is described to provide to other components a software or communication protocol over a network. In a way, the SQL database that are associated with it are consider a communication process and application due to the database being able to remote to the application. In this case that TumblrSnap utilizes the service-oriented pattern, each class in a way allows for access to the network and connection to the SQL database allowing users to update and create items, receipts, and users.

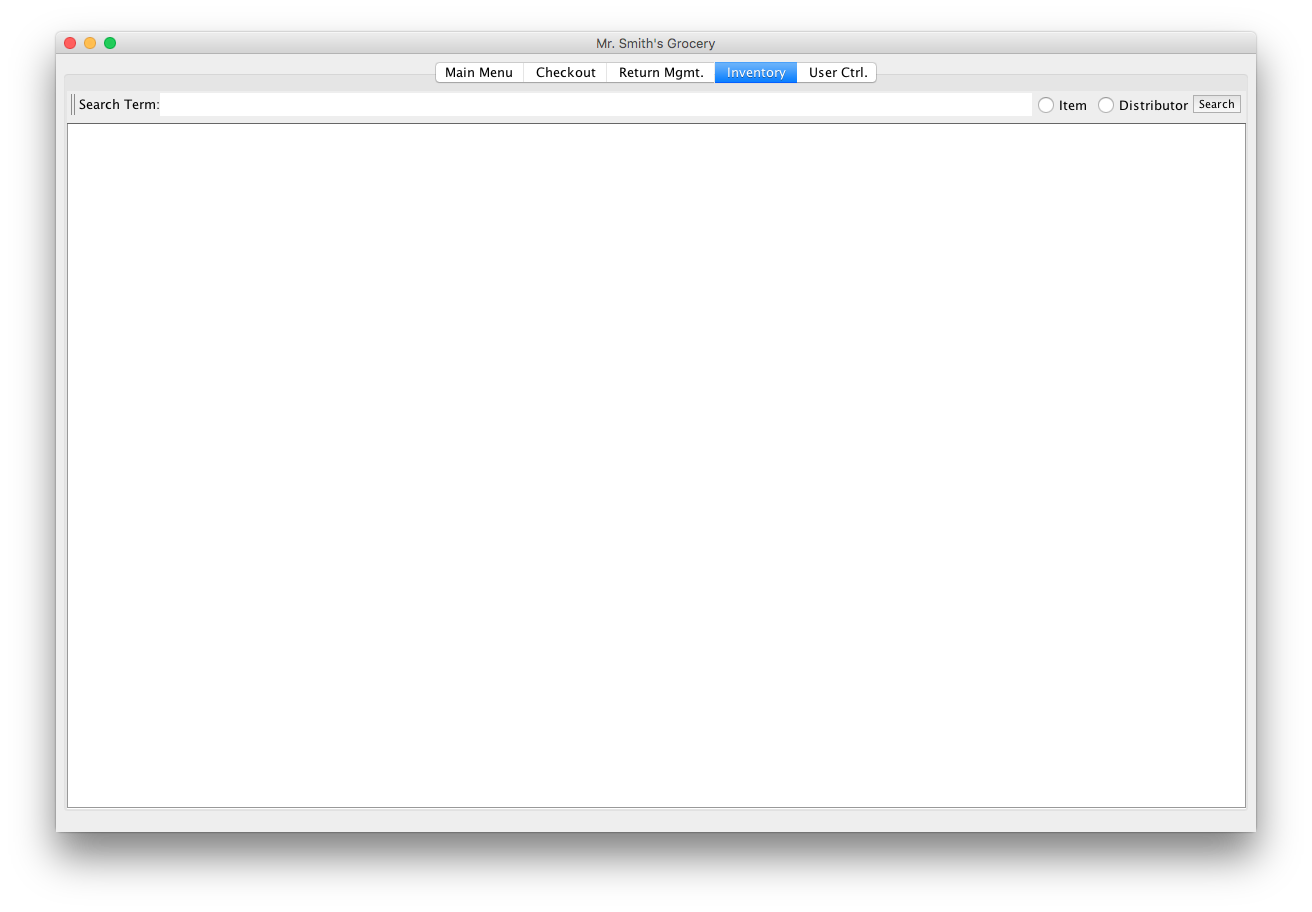
**User Interface Design(Screenshots):**

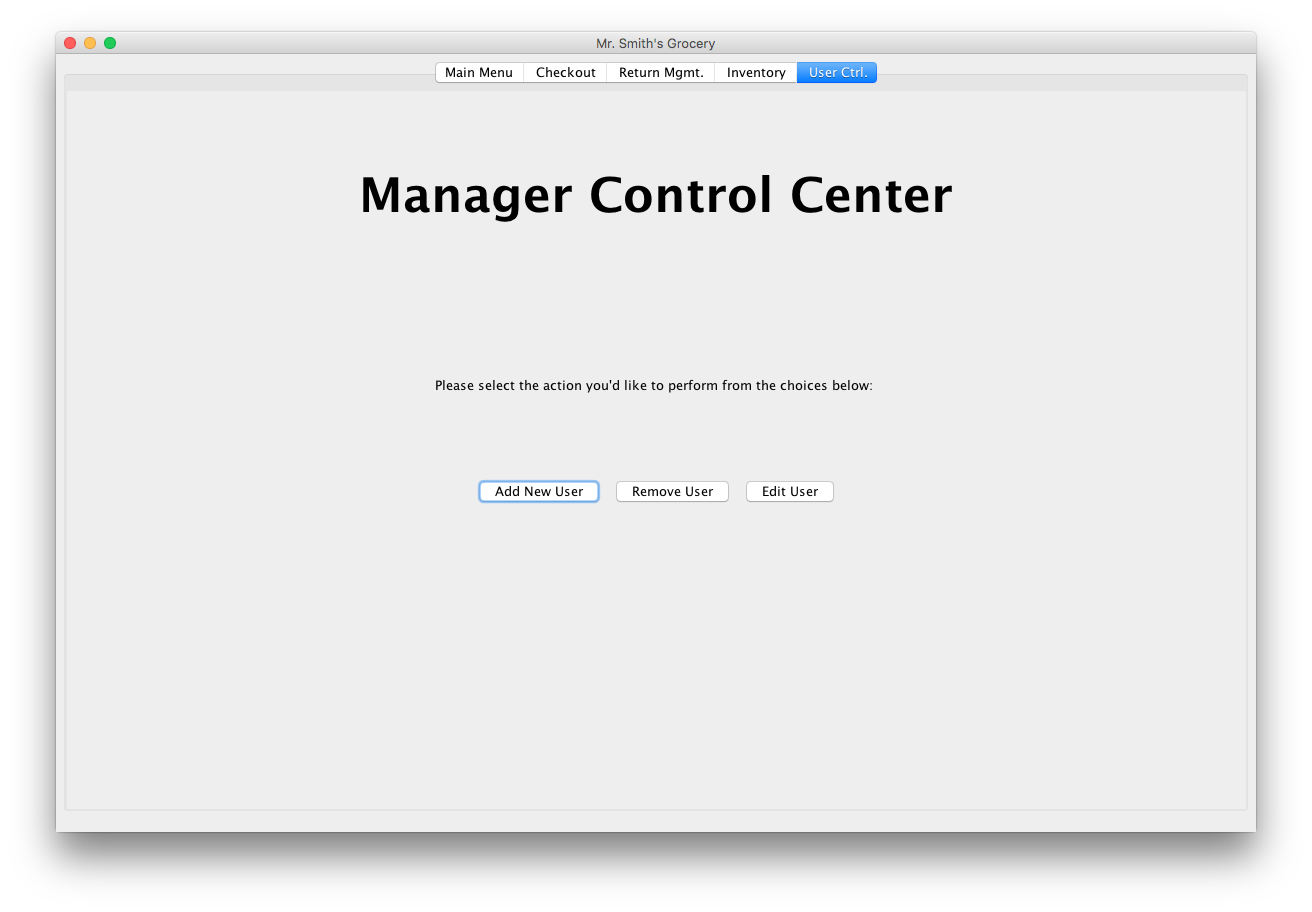






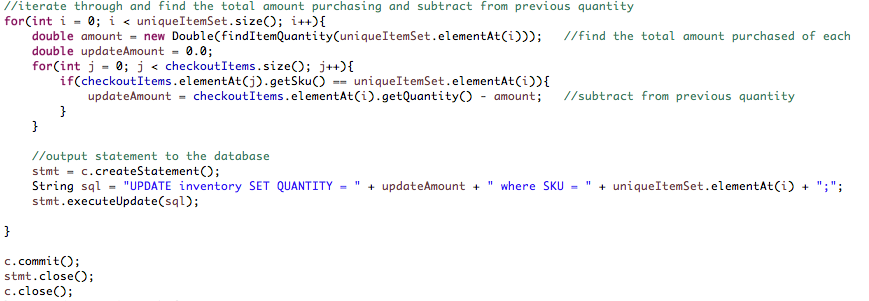






**Algorithm Design:**

This algorithm show below has key components that are consistent throughout the operations and functions in the application.



This functions is described to have the functionality of finding a unique set of items based on the sku numbers of each item. This equation/method finds the total amount of each item in a receipt item list and returns the quantity to be used for returning items off of a receipt and updating the inventory by adding back items returned.