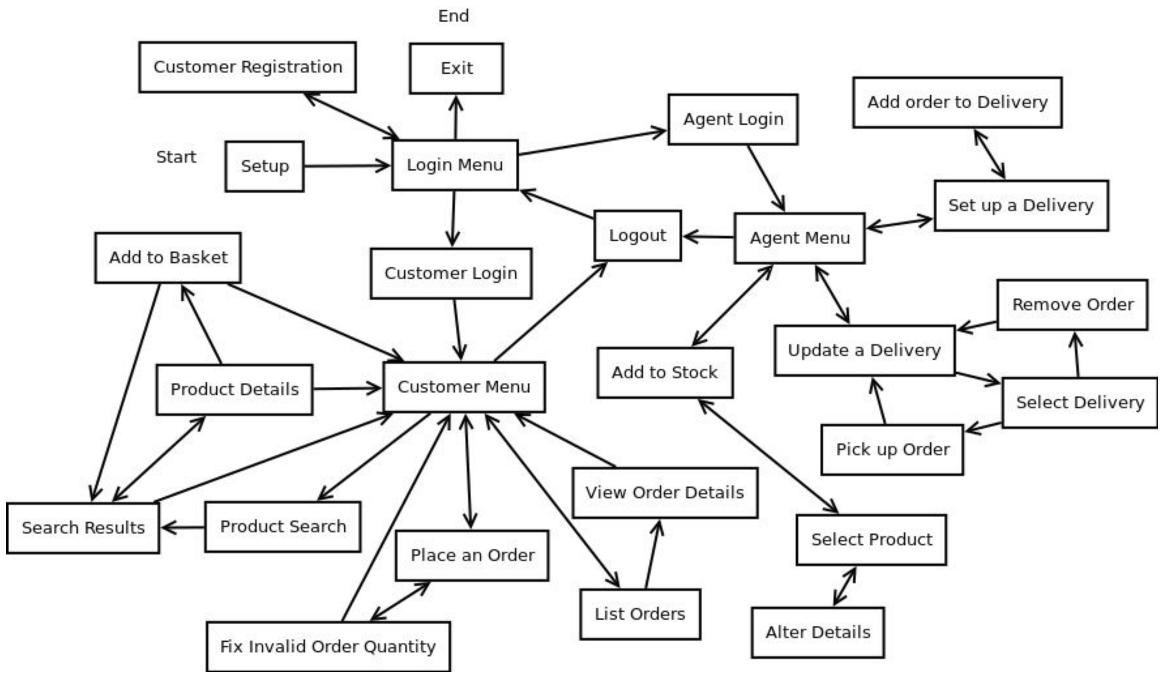
CMPUT 291 Mini-project 1: Design Document

Kevin de Haan, 1464775; Nicholas Bombardieri, 1468911;
Group 93

General Overview:

Usage instructions:
Unzip all files to an appropriate folder, then add .sql or .txt files containing the data you would like to use to the directory. Ensure that you have python3 installed on the system, and type ‘python3 main.py’ in order to run the program.

To use this software, enter the file locations of all relevant SQL commands in order to set up the tables and databases, then follow the prompts as appropriate. A flow diagram has be provided below to map the possible menu navigations.



Software Design:

The general design of our system is a state machine contained within an ‘Interface’ class. The class is initialized, and then the primary operational function is started with a call to interface.run(), which will display the next state required based on the return statements of the last function executed. Each menu state is an individual method within the interface class that proceeds to another (or the same) state based on user input. Invalid input is handled by printing an error message and then re-running the current state with the same settings as previous. Data is passed between method (when applicable) as attributes of the main class. Some other methods have been created to minimize code duplication, and are used within the state methods without directly displaying as a new ‘menu’ page. When the user chooses to exit the program, the run() function breaks out of its continuous loop and the program exits.

The major methods are as follows:

run():

    State machine framework, handles input errors and state transitions

login_menu():

    Displays login menu and handles navigation to registration and logins. Program exit point

register():

    Allows user to register as a new customer. Will refuse duplicate CIDs

customer_login():

    State for customer login. Will refuse incorrect passwords or nonexistent CIDs

agent_login():

    State for agent login. Will refuse incorrect passwords or nonexistent CIDs

logout():

    Logs out user or agent

exit():

    Exits the program

customer_menu():

    Displays options and handles customer state selection.

agent_menu():

    Displays options and handles agent state selection.

search_products():

    Takes user keyword input and stores search results. Leads to search_results if any exist

search_results():

    Displays search results according to relevancy and allows selection of products to view product_submenu. Handles pagination of results

product_details(): (Not a state)

    Used in search_results to print information about a product

product_submenu():

    Displays detailed information on a product with the stores that carry it. Allows for adding to basket

place_order():

Places an order for all items in the customer’s basket. Prompts the user to alter the quantity of an item if the store does not have enough in stock

list_orders():

Lists all orders placed by currently logged in user. Allows user to view individual orders

set_delivery():

Allows an agent to create a delivery and add orders to it

update_delivery():

Allows an agent to update an existing delivery

add_stock():

Allows an agent to add stock to selected products

Testing Strategy:

Due to some functional issues with the python3 module unittest, we did not implement automatic unit testing.

Our testing strategy was to treat parts of the assignment design specifications as individual user stories, and ensure that each one could be completed successfully. We created testing data that fit each specific case and went through the program in the manner described, making sure there were no unexpected results.

Group Work Strategy:

Kevin implemented the state machine and ‘Interface’ class, as well as the basic menu states (login, register, customer menu, agent menu, etc). He also created the state for customer product searching and the product submenu. Documentation (including this design document) was also completed by Kevin. Kevin also attempted to implement unit testing but did not create any significant functional code from it.

Nick designed the code for placing an order, listing orders, setting up deliveries, updating deliveries, and adding products to stock, as well as all relevant sub-methods for their operation.

Each member of the group contributed to testing, though Nick probably did slightly more.

In general, each member created an equivalent amount of the SQL code.

Estimated time spent:

Kevin: 12.5hrs

Nick: 12hrs

Github was used for coordinating work and maintaining our codebase, and we communicated with Discord when not working physically nearby.