Miniproject1 Design Document

Group members:
graa - Rafaella Graña
awood - Andrew Wood
johnas - Johnas Wong

General Overview:

Welcome to our mini project. As a user, you will have access to our buy-and-sell application via a command line interface. Consult the diagram included to see the general overview of the application.

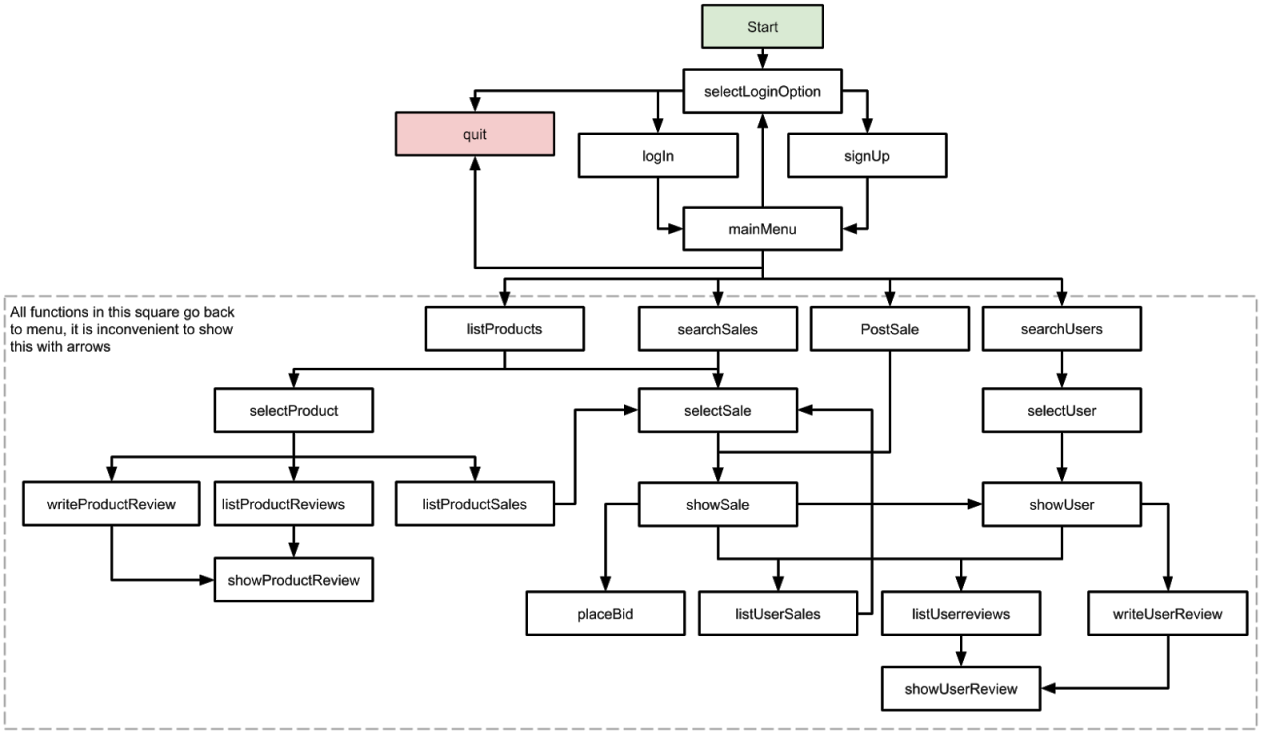
At the start, the user is asked whether they want to log in (have an existing account) or sign up (create a new account). Afterwards, the user is redirected to mainMenu(), the general function that redirects the user to all smaller functions. Here is a list of the main functions and a general description of them.

1. List products: when selected will show you all the products that featured in active sales. You will be able to select a product or the sale it belongs to, and see its information
2. Search for Sales: when selected will prompt you to enter keywords to use to search for specific sales. It will return all matching results. You can then select a sale and see its information
3. Post a Sale: when selected will ask you to give information about the sale you want to post. It will ask you to enter a product number optionally, this is for when you want to sale something that already exists in the app records
4. Search for Users: when selected will prompt you to enter keywords to use for searching for specific users. You can then select a user and see its information
5. Log out: goes back to SelectLogInOption()
6. Quit: ends the app

Note, all functions (except functions that ask for numbers) redirect back to the mainMenu() if the user inputs “9”. The reason 9 was chosen is because it is an uncommon digit to use and will usually never show up as a real input for most prompts.

Detailed Overview:

Here we will describe the functionality of all methods located in the class Handler defined in Handler.py. These are in the above diagram and constitute the backbone of the flow of functionality in this app. At any point, a 9 as input returns back to the main menu. Each one of these functions calls at least one sql function in the sqlBackEnd.py in order to display information taken from the database.



1. selectLoginOption(): Prompts the user to select either logIn(), signUp() or Quit.
2. logIn(): Asks the user for email and password (password is hidden so when user types they won't see their own writing), checks if the email and password exist and are correct, then redirects to mainMenu()
3. signUp(): Asks the user to enter an email that has not been used before, then set a password, creates the new user and inserts it into the database. Then redirects to mainMenu()
4. mainMenu(): Continually gives the user 6 options: go to listProducts(), searchSales(), postSale(), searchUsers(), log out, or quit.
5. listProducts(): Shows the user all the products associated with an active sale. The prompts the user to choose whether to selectProduct() or a selectSale() based on their pid and sid respectively.
6. selectProduct(): prompts the user for a valid pid. Displays information about the product and offers the user three choices: writeProductReview(), listProductReview(), listProductSales().
7. writeProductReview(): Prompts the user to write a review for the previously selected product. Then redirects to showProductReview()
8. listProductReview(): Lists all reviews associated with that product, then prompts the user to select a valid review. Then redirects to showProductreview().
9. showProductReview(): Just displays the previously selected review.
10. listProductSales(): Lists all sales associated with the previously selected product. Then redirects to selectSale()

11. searchSale(): Prompts the user to input one or more keywords that will try to match any sales or sale description in the database. It shows all matching keywords and associated products.. Then gives the user the choice to selectSale() or selectProduct()
12. selectSale(): Prompts the user to the give a valid sid and directs to showSale()
13. postSale(): Prompts the user to fill information about a new sale and then redirects to showSale()
14. showSale(): displays information about the previously selected sale then asks the user to either placeBid(), listUserSales(), listUserReviews() and showUser()
15. placeBid(): shows the user the highest bid on the sale currently and prompts for a new bid higher than the highest bid so far then returns to mainMenu()
16. listUserSales(): lists all the sales associated with the previously selected user then returns back to mainMenu()
17. listUserReviews(): lists all the reviews associated with the previously selected user then prompts the user to select a specific review and then showUserReview()
18. showUserReview(): displays information about the previously selected review the returns back mainMenu()
19. SearchUsers(): Prompts the user to input one or more keywords that will match the name, city, or email of a user. Then asks the user if they want to selectUser() or search again
20. selectUser(): Prompts the user to select a user based on its email. The redirects to showUser()
21. showUser(): Displays information about the previously selected user and gives the user three options: writeUserReview(), listUserReviews(), listUserSales(),
22. writeUserReview(): Prompts the user to provide data for the new review, stores the review in the database then returns to mainMenu()

Testing Strategy

For this app, the Handler class was designed first based on the flow diagram. It was designed with print statement as “fake queries” to replace where the actual sql queries would go. This way, we could ensure the flow and functionality of the program worked and did not redirect the user to the wrong method. Second, we tested each query separately via terminal or script, this way we could make sure the values it returned were correct. Then, both were combined and user tested. We tried to enter values that would break the app like “NULL” or duplicate primary keys which we knew would give an error if we had bad code. This allowed us to see if any checks or assertions were not being properly implemented. The use of a debugger was very useful for the last stage since, without it, it would have impossible to figure out which of all the numerous functions was causing errors. We did not really keep track of every single bug, mostly because they were, in the most part, very quick to fix like syntax errors or out of index errors. One of them which took the longest to fix was the one that occurred when the SelectLoginOption() and mainMenu() methods kept returning back to each other whenever the

user wanted to quit. Query errors were surprisingly easy to fix, it was just a matter of reading the documentation.

Group work break-down Strategy

The need for always keeping an updated version was very strong. So we decided to use github as a way to keep our project in line. The use of commits also helped to see who was doing more contributions to the assignment than others. We only met face to face once for the project. Time management was difficult, so we used google hangouts while instead of in person meetings. We also had a groupchat where we assigned each other more functionalities and asked questions whenever we felt stuck.

Work division:

- Handler.py (all functions): Rafaella, Andrew, Johnas
- sqlBackEnd.py: Johnas, Andrew, Rafaella
- Main.py: Rafaella
- Report (including diagram): Rafaella
- Testing: Ongoing and fully collaborative
- README.txt: