CMPUT 291 Mini Project 1 Design Document

by Daylen Pacheco and Helen Aquino

A. Overview of System

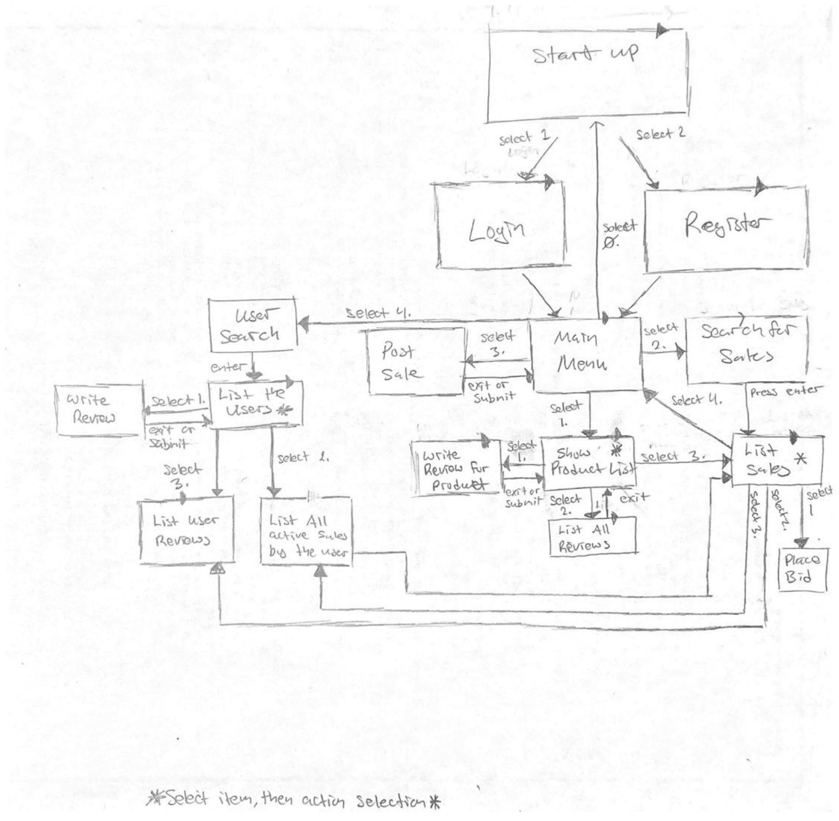

Figure 1: Flow Diagram showing the different actions the user can take to traverse the system.

The program first brings users to a start-up screen, where they are prompted to either log in as an existing user or register as a new user. Once logged in/registered, the user is taken to the main menu, where they can choose one of four actions:

1. Show the list of products in the database: If this action is selected, the user is shown the product listing. The user can then select a particular product from the listing and choose one of the following actions:
    a. Write a review for that product
    b. Show all the existing reviews for that product
    c. List the sales associated with that product: The user can then select a particular sale from the results and choose one of the following actions:
        i. Place a bid on the selected sale.
        ii. List all the active sales placed by the seller of that particular sale.
        iii. List all the reviews made about the seller of that particular sale.
2. Search for a sale: If this action is selected, the user is prompted to type in one or more keywords to search for sales that match either the sale description or the product description if the sale is associated to a product. The user can then select a particular sale from the results and choose one of the following actions:
    a. Place a bid on the selected sale.
    b. List all the active sales placed by the seller of that particular sale.

    c. List all the reviews made about the seller of that particular sale.
3. Post a new sale: The user is prompted to add the required details of the sale, then is taken back to the main menu.
4. Search for another existing user: The user is prompted to enter keywords that are in either the requested user's name/email. From the results, the user can select one of the following actions:
    a. Write a review for the selected user
    b. List all active listings of the user
    c. List all the review made about the selected user

The user can perform any of the following four main actions in any particular order. When finished, the user can log out and exit the program.

B. Design Details

1. *main.py*; imports every other python file in the folder
    a. main(): connects to the database using filename, calls start() and if successful, carries onto mainMenu()
    b. mainMenu(): the main menu of the system; in a while loop the user is prompted to select an action and the function calls either list_products(), sale_search(), post_sale(), user-search(), or close_program(), depending on the selected action, or the loop finishes to take the user back to the startup page.
2. *database.py*; imports the User class from user
    a. connect(path)
    b. set_user(email, name, city, gender)
3. *startup.py*: startup page of the system
    a. start()
    b. login()
    c. register()
    d. getUser(email)
4. *user.py*: contains the User class used throughout the system. The User class has attributes for email, name, city, and gender, and the class has methods to return each of the attributes.
5. *external_funct.py*:
    a. clear_screen()
    b. close_program()
    c. place_bid(sID_choice, maxAmt): prompts the user to enter a bid amount for the selected sale ID, checks if the amount is larger than the current largest bid in the database, then continues to get the remaining values needed to insert a new bid onto the bids table
    d. get_sale_select(sID_choice): creates a query to select the specific details of the sale with the given sale ID. The following details are checked if they have None values: product description, number of reviews, and average product rating; if a sale has a None value for any of these details, the None value is replaced to inform there is no description/no reviews/no ratings for that sale. All the details are then printed out and returned.
    e. print_active_sale(rows): the results from active_sales() are firstly printed out before the user is prompted to select an index for the sale to be returned.
6. *users.py*:
    a. user_search(): in a loop, the user is prompted to enter keywords that match either the name or email of the user they're looking for, then a query is made to look for users that match those keywords. The results are then assigned to variable 'list' and passed as an argument for print_userSearch, before going back to the main menu.
    b. print_userSearch(list, search): prints the results of the previous query

7. *sales.py*:
    a. sale_select(sID_choice): the next actions the user can take are displayed and the user is prompted to choose an action. One of the functions place_bid(), user_active_sales() and print_reviews() are called depending on the choice the user makes.
    b. active_sales(pID_choice): Makes a query to list all active sales associated with the product, ordered based on the remaining time of their sale. print_active_sales() is called to return an index, and sale_select() is called for the next action choices.
    c. sale_search(): the user is prompted to enter keywords to be matched the sale descriptions through a query made, and the results are printed out. The function sale_select() is then called for the next action choices.
    d. post_sale(): prompts the user to enter the following information about the new sale: sale's product ID (optional), description, condition, reserved price. The remaining information is found (sid, email) and all the prompted info is inserted into the database as a new sale.
    e. get_datetime(): Prompts the user to enter a correct date (year-month-day) and time (hour-minute) format for a sales end date and time, strings the date and time together and returns the result
    f. generateSID(): returns a randomly generated sale ID
8. *products.py*:
    a. write_preview(): the user is prompted to fill in a rating and review text for the selected product before it fills in other required fields to insert the new product review into the previews table
    b. list_preview(pID_choice): creates a query to select all the product reviews related to the selected product ID
    c. list_products(): creates a query to select all the products with currently active sales and prints out the results. The user is then prompted to select an action out of the following given to them, and a function is called, depending on the action the user selects

C. Testing Strategy

The nature of our testing was to use a dataset that had a wide of different types of data that we could use to ensure that we were retrieving and displaying the correct data as per the project specifications. This was data such as sales that have already expired, sales without a pid, sales without a rprice, sales without both a pid and a rprice.

After confirming that our program had displayed the correct data as per our data set, we began testing and checking that what we were entering into our program was as per the specifications. As our program displayed data correctly, if we were checking that the data was added correctly then we knew that the data being entered would not cause any issues.

Bugs Found:
Numerous bugs that were found pertained to queries data that contained None/Null statements and trying to print those statements through a formatter.
Not checking the type or validity of the input was causing a lot of issues.

D. Groupwork Strategy

The group work was initially started with designing the structure of the program using the flow diagram that Daylen had created. From there the both of us began creating issues on a corporate github repository (https://github.com/CMPUT291W20/MiniProject1) for all the tasks that needed to be completed. From here we began assigning ourselves to issues that we felt we could/wanted to complete within a reasonable time; and if someone had stopped working on that particular issue they would assign themselves from the issue to show that they are no longer working on that issue and the other could pick up where they left off. Furthermore we were able to keep adding new issues for troubles that we were having as a place for both to keep track of current issues in the system.

Task Distribution:

Daylen (~25 hours):
- Startup: Login/Register for a user
- User Search
- All reviews for a user
- Write a review for a user
- Active sales for a user
- Active sales for a seller
- Create Dataset for testing
- Fixed bugs pertaining to queries
- Added Error checking to all inputs
- Refactoring/Modularizing the code

Helen (~14 hours):
- List active sale products
- Create product review
- List all reviews for a product
- Sale search
- Display active sales or products
- Place bid on sale