Overview:

This project utilizes a simple command line UI that allows the user to interact with a persistent SQLite database containing the information based on an auction/sales system in which they may view the sales, reviews, bids and products and perform actions with such. Throughout the application the user will be prompted to enter inputs by the application to which they should type the requested input into the terminal and press confirm to finish and send the input. Most inputs will follow the following types:

1. A single digit (1..9) for "Enter an action:" prompts - Most menu options are displayed with a number before their description, this number corresponds with what should be inputted to perform the action in the menu.

2. Requested field to check/add to the database (Eg. "Enter email:", "Enter product:") - The user is expected to enter the requested information based on their personal information/what was previous provided. These requests vary but are typically detailed by the program through the input string. Most of said inputs are restricted by a check either for character distinctions or character limits. The program's *searching* functions are case-insensitive and will match inputs with data.

3. The input "return" - Entering "return" under most circumstances will allow the user to return to the previous menu/action. This "return" can be utilized to cancel an action currently being performed.

Any output provided from the program will be formatted with the column names of for each element at the top, using '|' as separators having each object printed on their own separate row.

Software design:

Assmuptions: the bid, pid and sid are assumed to be any alpha-numeric character that is limit to a capacity given by the scheme of the database.

(1) *Main function:* The main function connects and disconnects the database, prints out a menu for the user and calls each function. At the start of the program, the user will be asked to enter the name of the database to connect to. Once connected the user has the option to sign up as a new user or login. After logging, the main menu for interacting with the database will be provided allowing the following interactions.

(2) *List products:* Lists all products that have active sales associated to them. For each product that qualifies, the product id, description, number of reviews, average rating and number of sales associated are listed in descending order of the number of active sales. From this listing, the user may perform the following:

   I. Write a product review
   II. List all reviews of the product
   III. List all active sales of the product

(3) *Search for sales:* The user may enter one or more keywords and the system will retrieve all active sales with said keyword in either the sales description or product description.

(4) *1-2 Follow-up:* From the listings of either action 1 or 2, the user may select a sale which will provide more details about the selected sale. The following actions will be enabled:

   I. Place a bid on the selected sale
   II. List all active sales of the seller
   III. List all active reviews of the seller

(5) *Post a sale:* The user enters a product id (optional), sale end date and time, sale description, condition and a reserved price (optional) which will then be posted as a sale in the current database.

(6) *Search for users:* The user enters a keyword which will then return all user profiles that have the keyword in either name or email. Writing a review will overwrite a previous review on that user. From the set of users, the following actions are enabled:

   I. Write a review
   II. List all active listings of the user
   III. List all reviews of the user

Testing Strategy:

All testing was done using the test data in the data.sql file which was inputted into the database “project1.db”. The functions were tested individually as the program was being written. Once the functions where completed, more testing was done using the test database. The correctness was judged by manually examining the results from the program and comparing with the data to ensure the code’s correctness.

Break-down:

ducay - Succesfully managed to complete functionalities #2:Search for sales and #3:1-2 Follow up. woldegio - Succesfully managed to complete functionalities #3:Post a sale and #4:Search for users. Ojohnson - Succesfully managed to complete functionalities of the login/signup and #1:List products. The group members shared files through Github in order to ensure efficient collaboration as well as to efficiently distribute the work load. The group met up three times for an average of two hours either in-call or in-person to discuss the project. Each group member spent approximately 6 hours over the course of a week completing their work load.