# Criterion B

Final Prototype & Prototyping Process

After the interview, it was evident that most of the suggestions were centralized around bettering the search abilities and visual appeal of the database. Improvements referenced included:

- Adding a new feature that used the brand in the inventory and profit/loss list, suggesting better decisions for speculation. (Figure 4 and 5)
- Adding a date bought feature while selling (Figure 3)
- Add brand and collaboration for the buying and selling page. (Figure 2 and 3)
- Putting in a visual component that illustrates the results of the profit/loss (Figure 5)
- Searchability would benefit the user in the speculation as discussed in the interview. (Figure 4 and 5)
- Most of the other corrections were focused on the language and formatting of the language in the application. (Figure 1)

(Appendix A)

Final Prototype
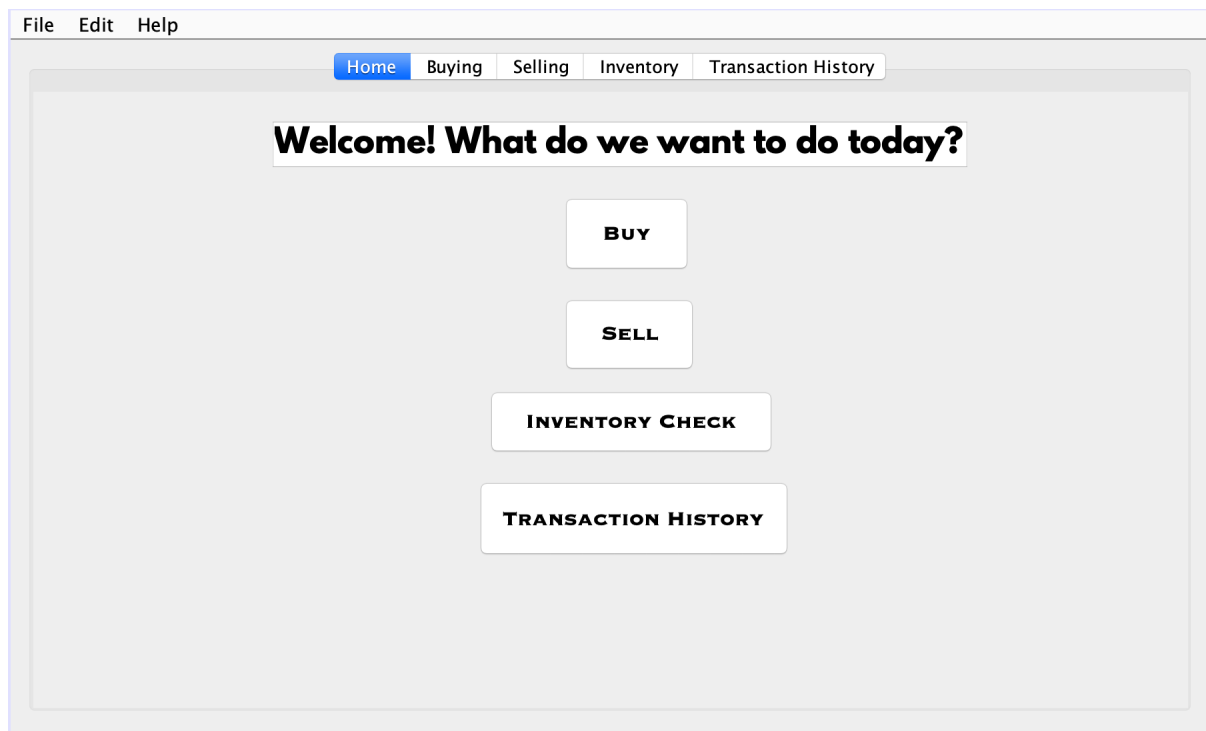
(Figure 1)

File   Edit   Help

Home | **Buying** | Selling | Inventory | Transaction History

Name of the Shoe: [   ]

Brand: [   ]

Collaboration: [   ]

Buying Price (USD $): [0] ↕

Gender: [-----] ↕

Size (US): [0] ↕

Date Bought: [DD] / [MM] / [YYYY]

[OK]

(Figure 2)

File   Edit   Help

Home   Buying   **Selling**   Inventory   Transaction History

Name of the Shoe: [   ]

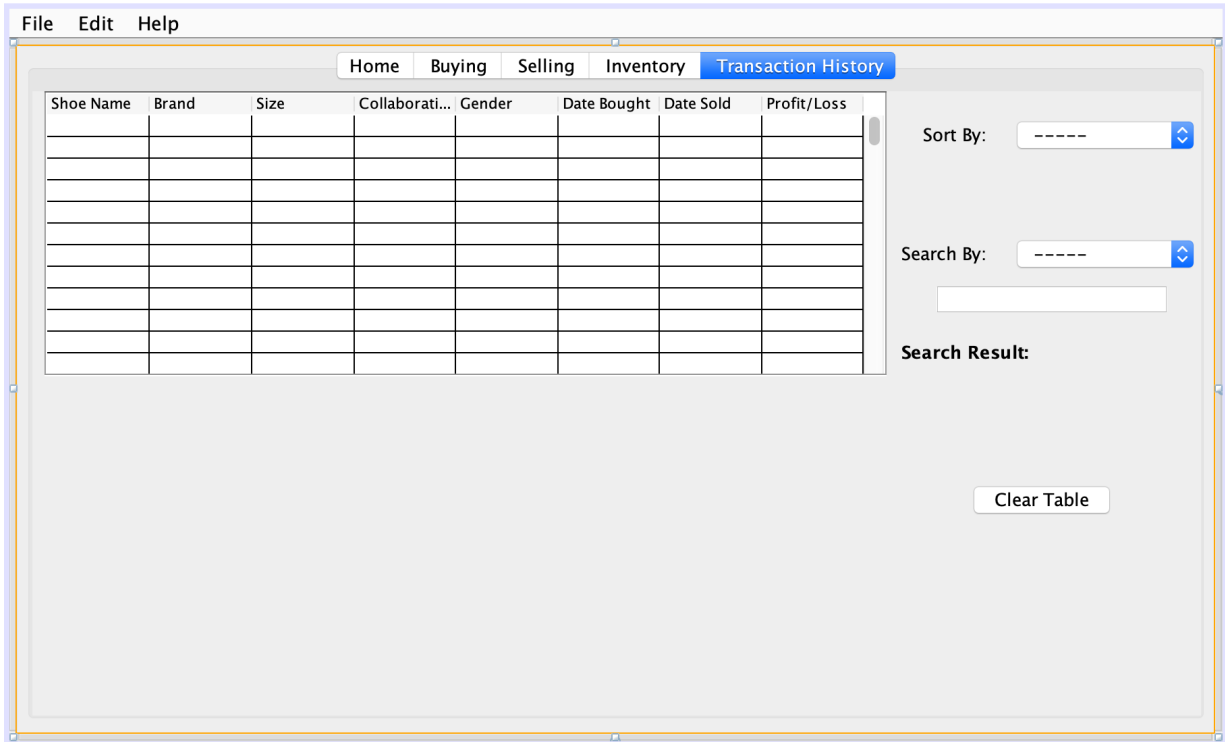Brand: [   ]

Collaboration: [   ]

Selling Price (USD $): [0 ⇕]

Gender: [----- ⇕]

Size (US): [0 ⇕]

Date Bought: [DD] / [MM] / [YYYY]

Date Sold: [DD] / [MM] / [YYYY]

[OK]

(Figure 3)

File   Edit   Help

Home   Buying   Selling   **Inventory**   Transaction History

Shoe Name	Brand	Size	Collaboration	Gender	Date Bought

Sort By: [----- ⇕]

[Refresh]

Search By: [----- ⇕]

[   ]

Search Result:

[Clear Table]

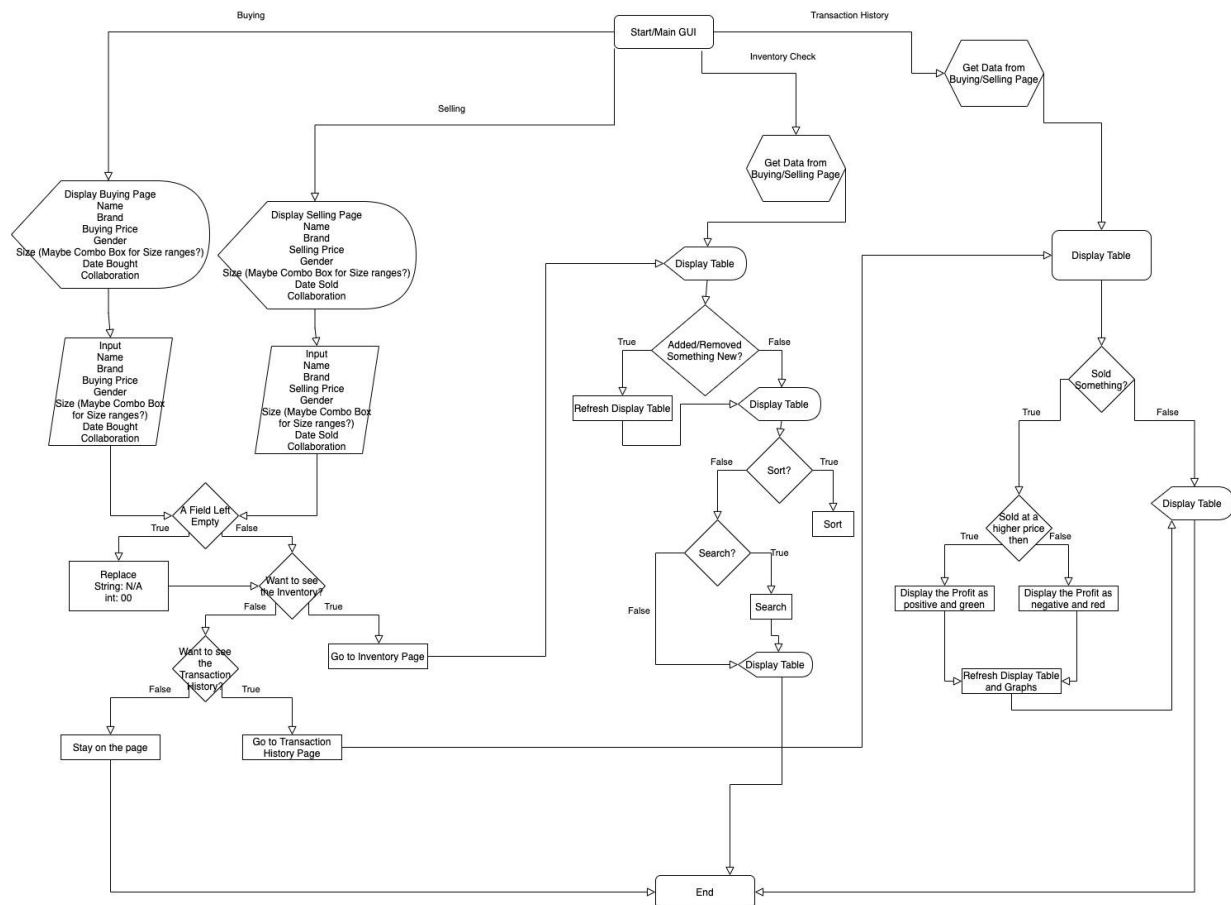(Figure 4)

(Figure 5)

Input and Output

Input (Labels and data required)	Data Type (String, int, double, etc.)	Normal Range etc. (Limitation/Boundaries of the int values or string, what can the input be or be in between)	Example
Page Directors	Boolean	In between the 4 choices	Buy, Sell, Inventory Check, Transaction History
Name of Shoe (Search by option is same)	String	Not left blank	Yeezy
Brand (Search by option is same)	String	Not left blank	Adidas
Price	int (What is it called	0 - ∞	350

(Buying/Selling)	if there is an up and down button?)		
Shoe Gender (Search by option is same)	String Array	Mens or Womens or Kids	Womens
Shoe Size (Search by option is same)	double	0.0 - 40.0	12.5
Date Bought/Sold	int	01 - 31 (Day) 01-12 (Month) After 2021 (Year)	01 / 01/ 2021
Collaboration (Search by option is same)	String	Not left blank	Off-White
Search by Options	String Array	Name or Brand or Shoe Gender or Shoe Size or Collaboration or Price	Name
Search by Search Bar	Depending on Category	Depending on the category, same as the ones listed above	Nike

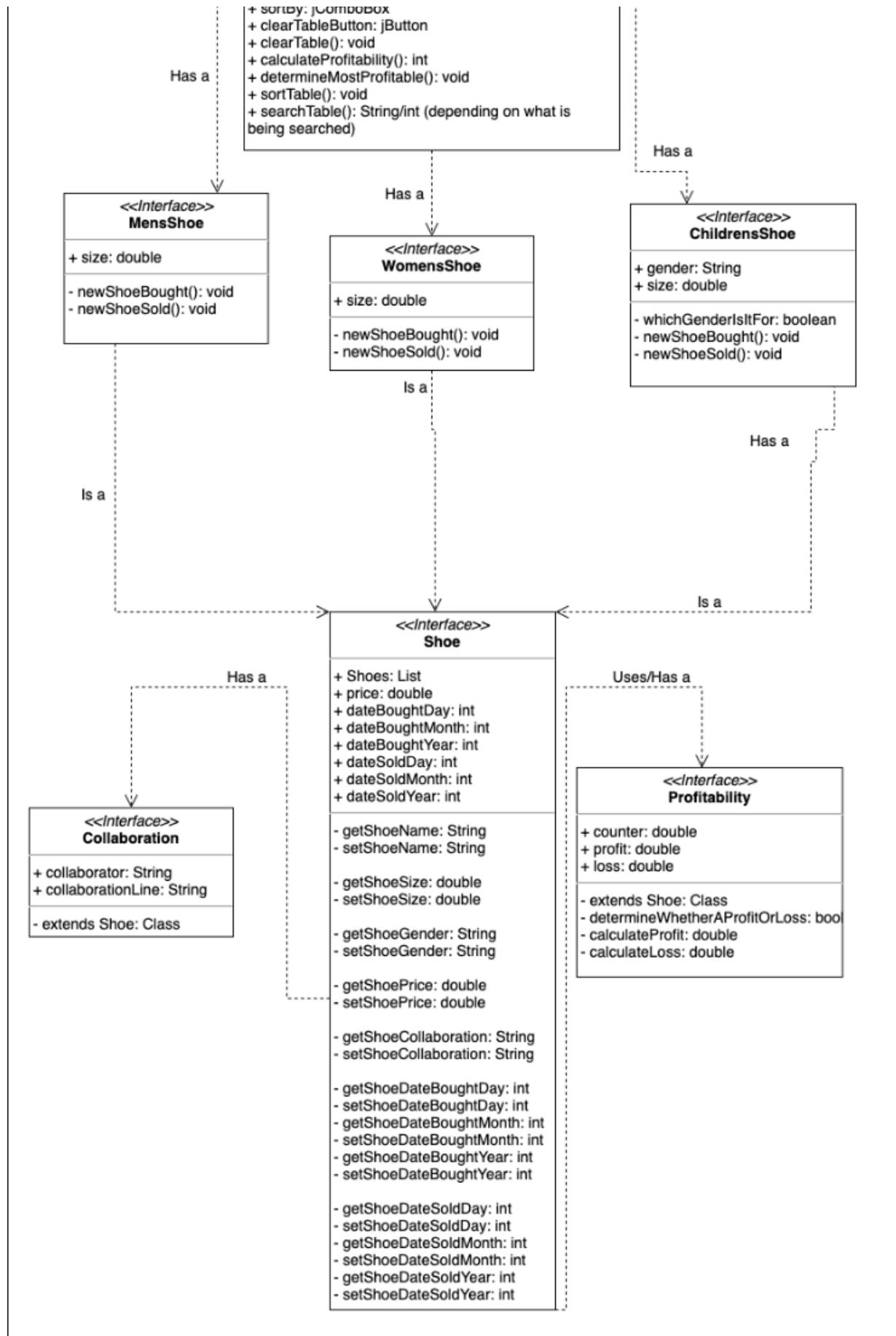| Output	Data Type	Normal Range etc.	Example
Search (Name)	String Array List	(Same as mentioned above for the specific category)	(Same as mentioned above for the specific category)
Search (Size)	double Array List	(Same as mentioned above for the specific category)	(Same as mentioned above for the specific category)
Search (Brand)	String Array List	(Same as mentioned above for the specific category)	(Same as mentioned above for the specific category)
Search (Gender)	String Array List	(Same as mentioned above for the specific category)	(Same as mentioned above for the specific category)
Search (Collaboration)	String Array List	(Same as mentioned above for the specific category)	(Same as mentioned above for the specific category)

Search (Price)	Int Array List	(Same as mentioned above for the specific category)	(Same as mentioned above for the specific category)
Table	(Refer back to each one as it was inputted)	All info per student displayed, in a JTable	Nike | Air Jordan | 11 | etc.

Processing Flowchart

Class/UML Diagrams

frame

<<Interface>>
Main GUI

+ buy: JButton
+ sell: JButton
+ inventory: JButton
+ transactionHistory: JButton

BuyingPane

+ shoeName: jTextField
+ shoeBrand: jTextField
+ buyingPrice: jSpinner
+ shoeGender: jComboBox
+ shoeSize: jSpinner
+ dateBoughtDay: jTextField
+ dateBoughtMonth: jTextField
+ dateBoughtYear: jTextField
+ collaboration: jTextField
+ confirmationButton: jButton
+ newShoeBought(): void

Selling Pane

+ shoeName: jTextField
+ shoeBrand: jTextField
+ sellingPrice: jSpinner
+ shoeGender: jComboBox
+ shoeSize: jSpinner
+ dateBoughtDay: jTextField
+ dateBoughtMonth: jTextField
+ dateBoughtYear: jTextField
+ dateSoldDay: jTextField
+ dateSoldMonth: jTextField
+ dateSoldYear: jTextField
+ collaboration: jTextField
+ confirmationButton: jButton
+ newShoeSold(): void

InventoryPane

+ inventoryTable: jTable
+ searchCategory: jComboBox
+ searchInput: jTextField
+ searchConfirmationButton: jButton
+ searchResultLabel: jLabel
+ searchResult: jLabel
+ refreshButton: jButton
+ sortBy: jComboBox
+ clearTableButton: jButton
+ clearTable(): void
+ sortTable(): void
+ searchTable(): String/int (depending on what is being searched)

TransactionHistoryPane

+ transactionHistoryTable: jTable
+ searchCategory: jComboBox
+ searchInput: jTextField
+ searchConfirmationButton: jButton
+ searchResultLabel: jLabel
+ searchResult: jLabel
+ refreshButton: jButton
+ sortBy: jComboBox
+ clearTableButton: jButton

- - Use >

<<Interface>>
SortAndSearch

+ counter: int
+ key: int/String
+ dateKey: int

- searchByName: String
- sortByName: String
- searchByBrand: String
- sortByBrand: String
- searchBySize: int
- sortBySize: int
- searchByCollaboration: Strin
- sortByCollaboration: String
- searchByPrice: int
- searchByDateBought: int
- searchByDateSold: int

+ clearTableButton: jButton
+ clearTable(): void
+ calculateProfitability(): int
+ determineMostProfitable(): void
+ sortTable(): void
+ searchTable(): String/int (depending on what is
being searched)

Has a

Has a

Has a

<<Interface>>
MensShoe

+ size: double

- newShoeBought(): void
- newShoeSold(): void

<<Interface>>
WomensShoe

+ size: double

- newShoeBought(): void
- newShoeSold(): void

<<Interface>>
ChildrensShoe

+ gender: String
+ size: double

- whichGenderIsItFor: boolean
- newShoeBought(): void
- newShoeSold(): void

Is a

Has a

Is a

Is a

Is a

Has a

Uses/Has a

<<Interface>>
Shoe

+ Shoes: List
+ price: double
+ dateBoughtDay: int
+ dateBoughtMonth: int
+ dateBoughtYear: int
+ dateSoldDay: int
+ dateSoldMonth: int
+ dateSoldYear: int

- getShoeName: String
- setShoeName: String

- getShoeSize: double
- setShoeSize: double

- getShoeGender: String
- setShoeGender: String

- getShoePrice: double
- setShoePrice: double

- getShoeCollaboration: String
- setShoeCollaboration: String

- getShoeDateBoughtDay: int
- setShoeDateBoughtDay: int
- getShoeDateBoughtMonth: int
- setShoeDateBoughtMonth: int
- getShoeDateBoughtYear: int
- setShoeDateBoughtYear: int

- getShoeDateSoldDay: int
- setShoeDateSoldDay: int
- getShoeDateSoldMonth: int
- setShoeDateSoldMonth: int
- getShoeDateSoldYear: int
- setShoeDateSoldYear: int

<<Interface>>
Collaboration

+ collaborator: String
+ collaborationLine: String

- extends Shoe: Class

<<Interface>>
Profitability

+ counter: double
+ profit: double
+ loss: double

- extends Shoe: Class
- determineWhetherAProfitOrLoss: bool
- calculateProfit: double
- calculateLoss: double

Development Plan

Creating a development plan would allow for a structured organization of what has to be done, and is not as affected by changes in programming, in comparison to algorithms.

1. Design MAIN GUI (1 week)
   a. Inventory
   b. Transaction History
      i. Calculating Profit and Loss
   c. Data Management (edit and remove inputted data)
      i. Buying
      ii. Selling
2. Create template classes (Shoe, etc.) (1 day), examples include:
   a. For Mens Shoe:

pseudocode for MensShoe

    extends Shoe
    intialize MENSSHOE
    MENSSHOE constructor method
    MENSSHOE takes in NAME, SIZE, BUYINGPRICE, SELLINGPRICE,
        SHOEDATEEBOUGHTDAY, SHOEDATEEBOUGHTMONTH,
        SHOEDATEEBOUGHTYEAR, DATEBOUGHT, SHOEDATESOLDDAY,
        SHOEDATESOLDMONTH, SHOEDATESOLDYEAR, COLLABORATION,
        BRAND, PROFITABILITY, GENDER, MENSSIZE

        super extends from SHOE (NAME, SIZE, BUYINGPRICE, SELLINGPRICE,
        SHOEDATEEBOUGHTDAY, SHOEDATEEBOUGHTMONTH,
        SHOEDATEEBOUGHTYEAR, DATEBOUGHT, SHOEDATESOLDDAY,
        SHOEDATESOLDMONTH, SHOEDATESOLDYEAR, COLLABORATION,
        BRAND, PROFITABILITY, GENDER)

        MENSSIZE = MENSSIZE

    GETMENSSIZE method
        return MENSSIZE

    SETMENSSIZE method
        set MENSSIZE = MENSSIZE

3. Create Sorting and Searching Class (3 days), examples include:
   a. For Sorting Collaboration:

ALPHABETSORTCOLLABORATION takes in the SHOES of LinkedList SHOES

      initialize variable TEMP for SHOES

      loop i from 0 to number of shoes in SHOES List

            loop j from i + 1 to number of shoes in SHOES List

                  if SHOECOLLABORATION at index i compared to SHOECOLLABORATION at index j is greater than 0

                  temp = object at index i for SHOES

                  set SHOE at index i to shoe at index j
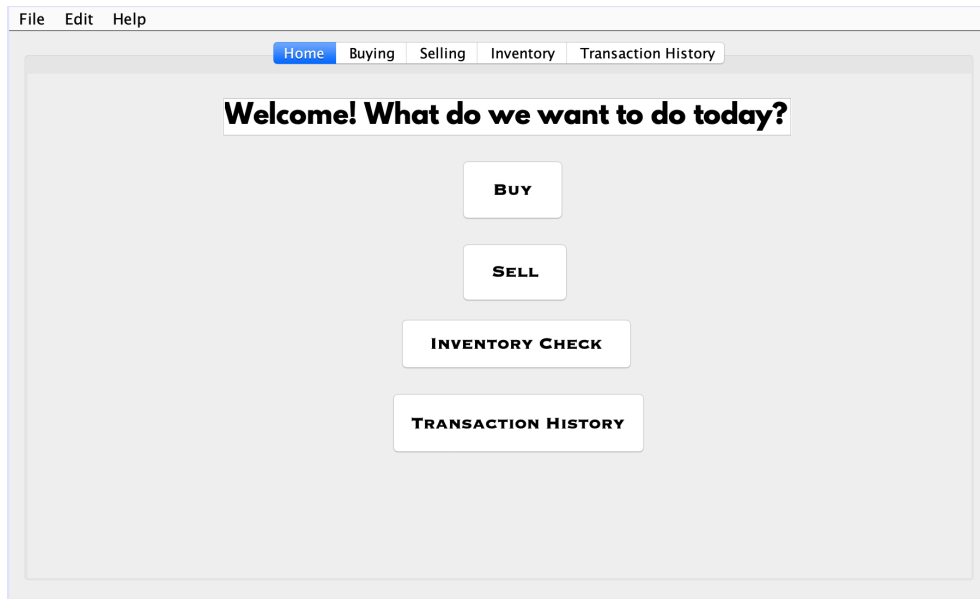
                  set SHOE at index j to shoe at temp

                  end if

            end loop

      end loop

4. Work on Programming of the Project (4 weeks)
   a. Create Error Handling Methods (JOptionPane) for any possible input or editing errors and possibly login page
5. Implement saving (3 days)
6. Perform testing plan to test program. (1 week)
   a. Make necessary changes in error
   b. Reflect on criteria of success
7. Final Interview client (1 day)
   a. Create a video instruction on how to use program
   b. Receive feedback and discuss for extensibilities and recommendations.

Testing Plan

File   Edit   Help

Home | Buying | Selling | Inventory | Transaction History

Welcome! What do we want to do today?

> Buy

> Sell

> Inventory Check

> Transaction History

Testing General Functionality	
Action To Test	**Method of Testing/Expected Result**
Buy Button	Go to Buying page
Sell Button	Go to Selling page
Inventory Check Button	Go to Inventory page
Transaction History Button	Go to Transaction History page
Testing Particular Correct & Erroneous Input	
Input	**Normal**
Selecting a page to go to	One of the options given

Buying and Selling Page

Testing General Functionality	
Action To Test	**Method of Testing/Expected Result**
OK Button	Add the shoe to the inventory table OR calculate the profit/loss of the shoe and add to the transaction history table

Testing Particular Correct & Erroneous Input			
Input	**Normal**	**Border**	**Abnormal**
Name	Any Name	Left Blank Show error warning and not taken in	Abnormally Long or has an int in the name
Brand	Any Brand	Left Blank Show error warning	Abnormally Long
Collaboration	Any Name	Left Blank Show error warning	Abnormally Long
Buying/Selling Price	Buying = Any Number Selling = Any Number	Negative Numbers 0 Show error warning	N/A
Gender	Non-initial titles	Try the first and last in the list.	Initial titles - User forgot to choose

			Show error warning and not taken in
Size	Any size	Negative Numbers 0 Show error warning	Abnormally huge number
Date Bought/Sold	Day = 1 - 31 Month = 1 - 12 Year = 2021 - ∞	Negative Numbers 0 Any number exceeding the limitations Show error warning	If no existing shoe when entering the date Show error warning of the shoe does not exist

File   Edit   Help

Home   Buying   **Inventory**   Transaction History

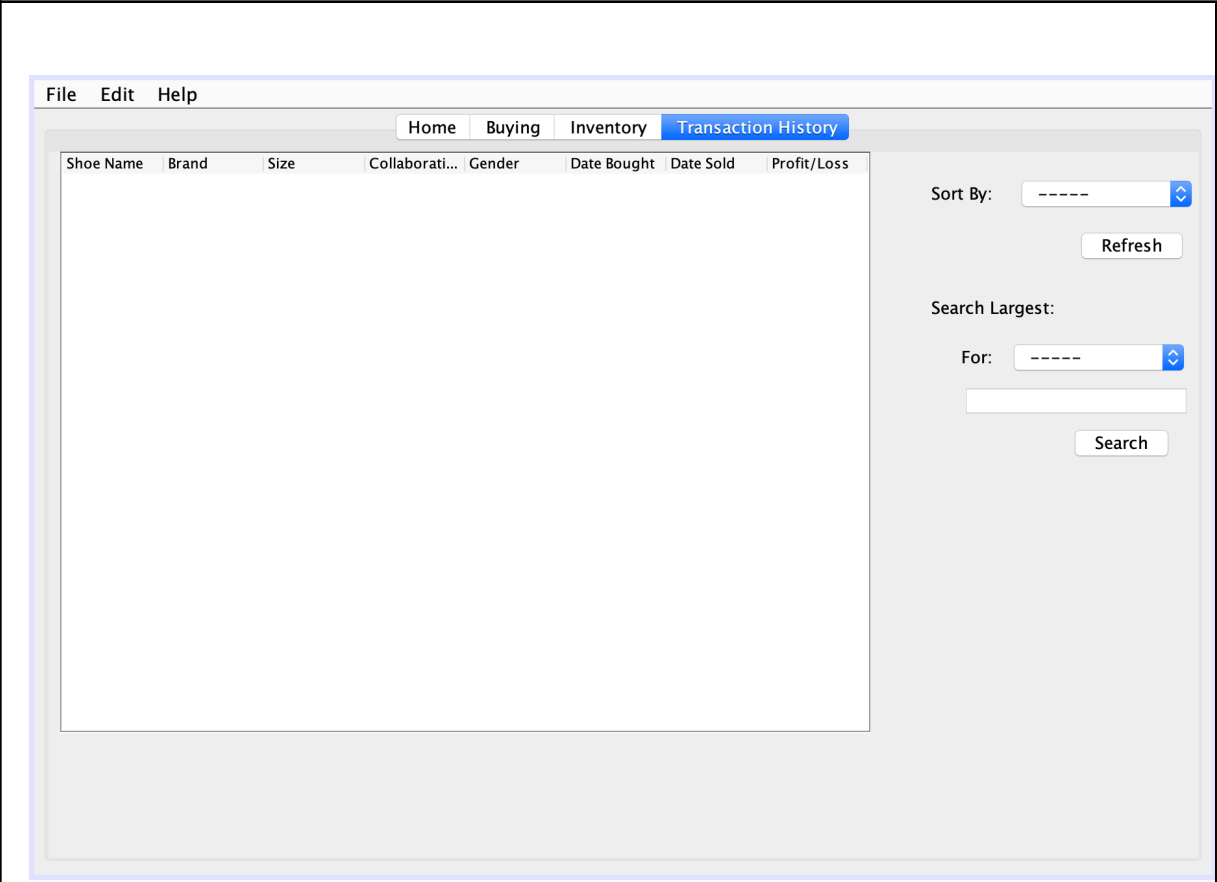| Shoe Name	Brand	Size	Collaboration	Gender	Date Bought

Sort By: ----- ◊

Refresh

Search By: ----- ◊

Search

Inventory Page

Testing General Functionality	
Action To Test	**Method of Testing/Expected Result**
Refresh Button	Click the refresh button. All shoes should be updated in a queue fashion and display only the shoes bought
Sort Refresh Button	Click the refresh button. All shoes should be ordered in a queue fashion following the attribute selected

Testing Particular Correct & Erroneous Input			
Input	**Normal**	**Border**	**Abnormal**
Table	All inputted data	No data entered for the specific column/row Show "Not Entered"/"ERROR"	N/A
Sort By Combo Box	Non-initial Title	Try the first and last in the list.	Initial titles - User forgot to choose Show error warning and not taken in
Search By Combo Box	Non-initial Title	Try the first and last in the list.	Initial titles - User forgot to choose Show error warning and not taken in
Search By Text Field	Something entered	Spelled incorrectly or shoe non-existent Show error warning	N/A
Searched Result	What is searched for	Not found Show "Shoe has been sold" or "Incorrectly	N/A

		Entered"	

File	Edit	Help		

Home	Buying	Inventory	Transaction History

Shoe Name	Brand	Size	Collaborati...	Gender	Date Bought	Date Sold	Profit/Loss

Sort By: -----

Refresh

Search Largest:

For: -----

Search

Transactions Page

Testing General Functionality	
Action To Test	**Method of Testing/Expected Result**
Refresh Table	Click the refresh button. All shoes should be updated in a queue fashion
Sort Refresh Button	Click the refresh button. All shoes should be ordered in a queue fashion following the attribute selected
Testing Particular Correct & Erroneous Input	

Input	**Normal**	**Border**	**Abnormal**

Table	All inputted and taken out data	No data entered for the specific column/row Show "Not Entered"/"ERROR"	N/A
Sort By Combo Box	Non-initial Title	Try the first and last in the list.	Initial titles - User forgot to choose Show error warning and not taken in
Search By Combo Box	Non-initial Title	Try the first and last in the list.	Initial titles - User forgot to choose Show error warning and not taken in
Search By Text Field	Something entered	Spelled incorrectly or shoe non-existent Show error warning or show no search result	N/A
Searched Result	What is searched for	Not found Show "Incorrectly Entered" or show no result	N/A

Word Count: 162

Appendix A:
Post Design Prototype Feedback: