

Criterion B: Solution Overview

Inputs/Outputs:

User Login UI

Input	Data Type	Example
Existing username	String	“sam234”
Existing user password	String	“samisgreat787”
New username	String	“dorothy999”
New user password	String	“0819237doro”
Confirm userpassword	String	“0819237doro”
Output	Data Type	Example
Error Handling	String	‘Username already exists’

Enter New Transaction UI

Input	Data Type	Example
Buy/Sell	Boolean	isBuy == true/false
Stock Name	String	“PTT”
Data of Transaction	String	“9 January 2019”
Number of Shares in Transaction	Integer	“500”
Price	Double	“2.81”
Dividends per Unit	Double	“0.60”
Output	Data Type	Example
Error	String	“Sale invalid”

Stock Specific Statistics UI

Input	Data Type	Example
Display Information For (Stock Name)	String	“Apple”
(Theoretical Sell) Stock Name	String	“Apple”
(Theoretical Sell) Number of Shares	Integer	“1000”

(Theoretical Sell) Price	Double	“50.60”
Output	Data Type	Example
Total Shares Owned	String	“Sale invalid”
Total Money Invested	Double	“400000”
Current Average Price	Double	“47.50”
Total Dividends to Date	Double	“5067.92”
(Theoretical Sell) Profit	Double	“6009.87”

Stock Specific Statistics UI

Input	Data Type	Example
Display Stock	String	“Apple”
Sort by	String	“Date”
Order to sort	String	“Recent First”
Output	Data Type	Example
Transactions Table	ArrayList<Transaction>	n/a (Output data on table)

Word Count: 0

Prototyping Process

Initial Prototype (First Draft + Client Annotation)

1. NAME

2. DATE

3. PRICE/UNIT

4. Dividend

5. AMOUNT/UNIT

BTENTIAL

ADD WARRANT

Share rights

Share give away

ENTER NEW

SUMMARY

TABLE

WHAT IFS

EXISTING NAME

Drop down (not textfield)

NEW STOCK

TF

DATE

DAY

MONTH

YEAR

DIVIDEND

TF

(Bakt per share)

AMOUNT

TF

(SHARES)

PRICE

TF

(Bakt per share)

☒

BUY

☒

SELL

error checking (cannot tick both)

OF CATEGORIES

→ FUN

→ COMMODITIES

1. AMOUNT (IN UNITS)

2. TOTAL MONEY SPENT

3. AVERAGE PRICE PER SHARE

4. All Transaction of Stock

5. DIVIDENDS IN PERCENT

6. DIVIDENDS IN BAHT

FUNCTION EXTRA

SELECT TIME FRAME

add sort by categories

Stock A

Total units owned: _____

Current average price: _____

Enter current price: TF

NET PROFIT: _____

calculate

TO TRANSACTION TABLE

100

100

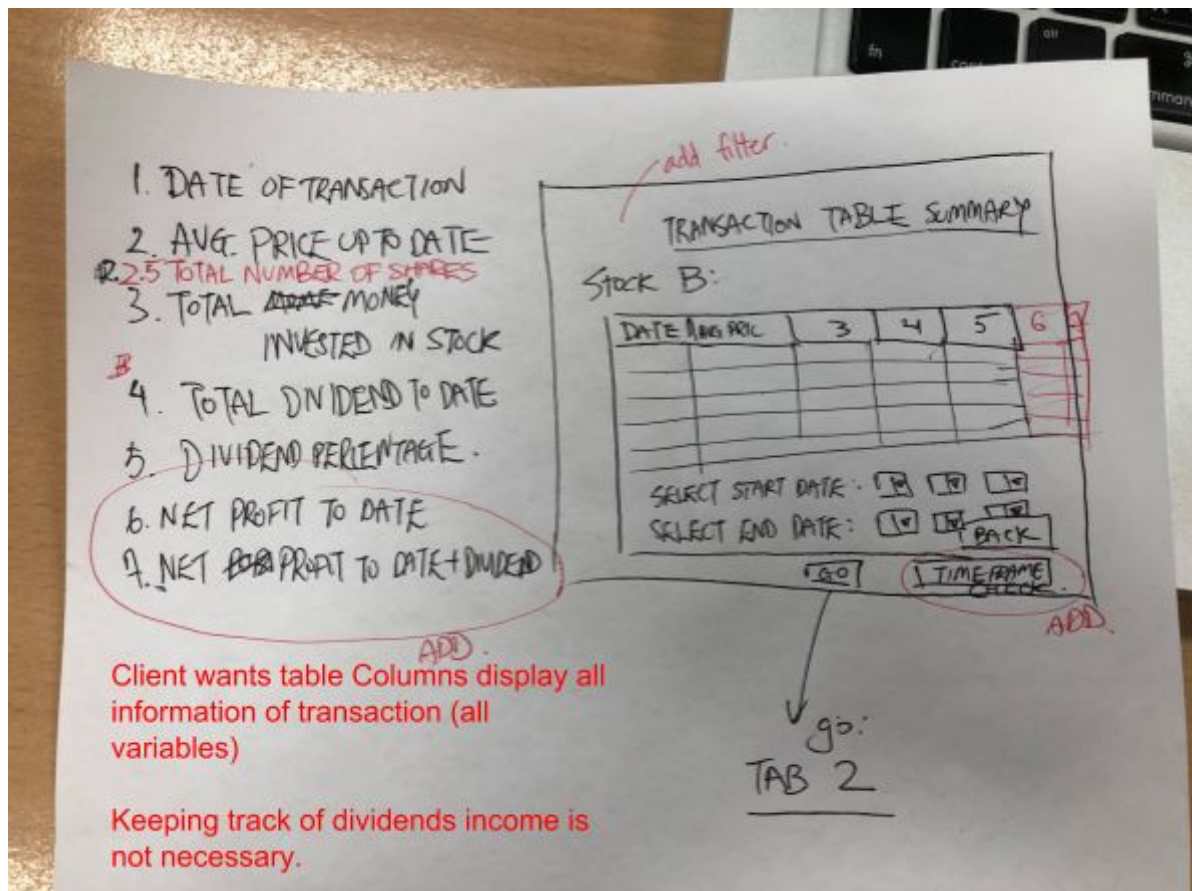
100

100

200

100

100



Prototype 2 Example

File Edit Help

User Login/New User

Enter New Transaction

Stock-Specific Statistics

Transactions Table

Returning User

Username

Password

New User

Username

Password

Confirm Password

File Edit Help

User Login/New User Enter New Transaction Stock-Specific Statistics Transactions Table

Type of Transaction ☐ Buy ☐ Sell

Stock Name

Date

Number of Shares

Dividends

Price

Enter Transaction

File Edit Help

User Login/New User Enter New Transaction Stock-Specific Statistics Transactions Table

Display Information For Display

Total Shares Owned

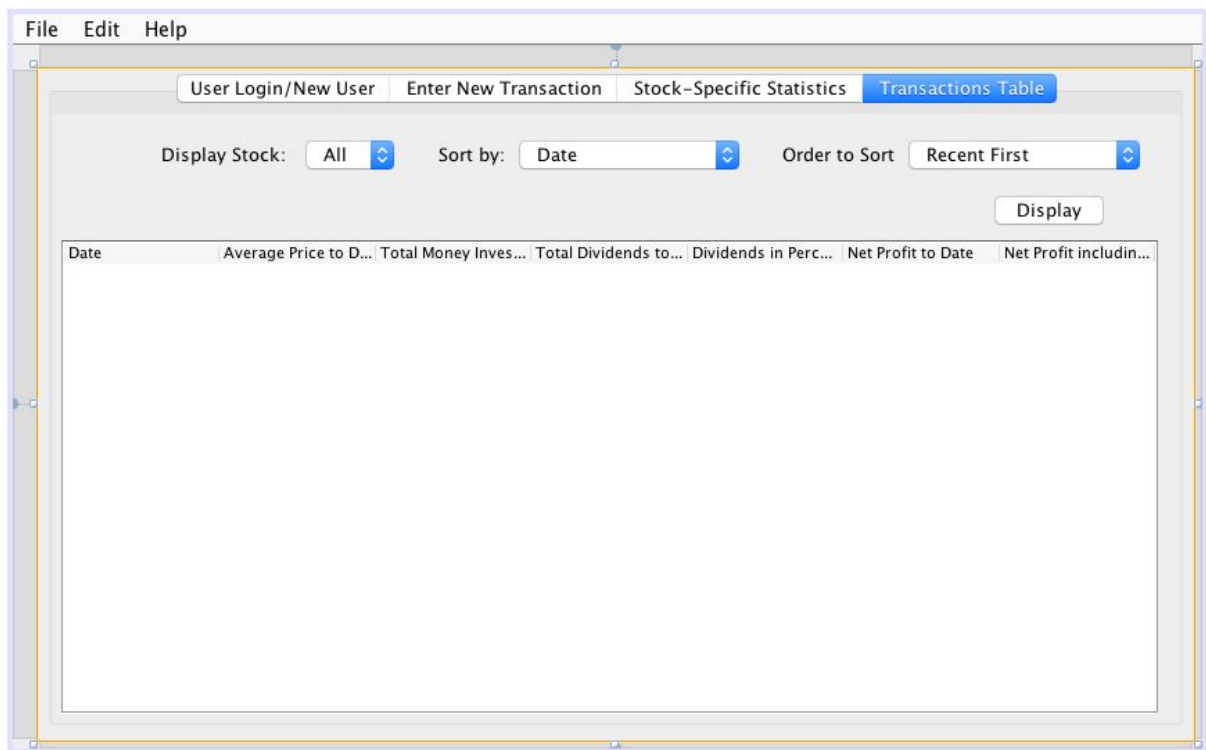
Total Money Invested

Current Average Price

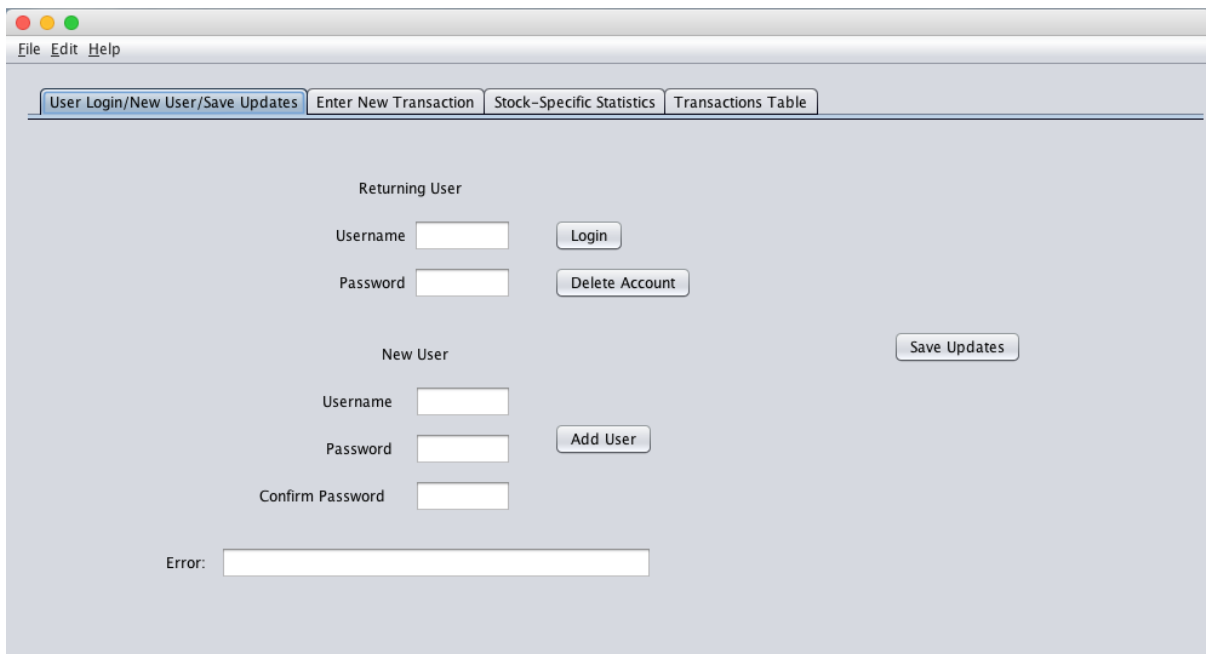
Total Dividends to Date

Sell (Amount) at (Price)

Calculate Profit Profit:



Prototype Final Stages



File Edit Help

User Login/New User/Save Updates Enter New Transaction Stock-Specific Statistics Transactions Table

Type of Transaction ☐ Buy ☐ Sell

☐ New Stock ☐ Existing Stock

Stock Name

Date

Number of Shares (Integer)

Dividends per Share (Number to 2 decimal places)

Price (Number to 2 decimal places)

Error

File Edit Help

User Login/New User/Save Updates Enter New Transaction Stock-Specific Statistics Transactions Table

Display Information For

Total Shares Owned

Total Money Invested

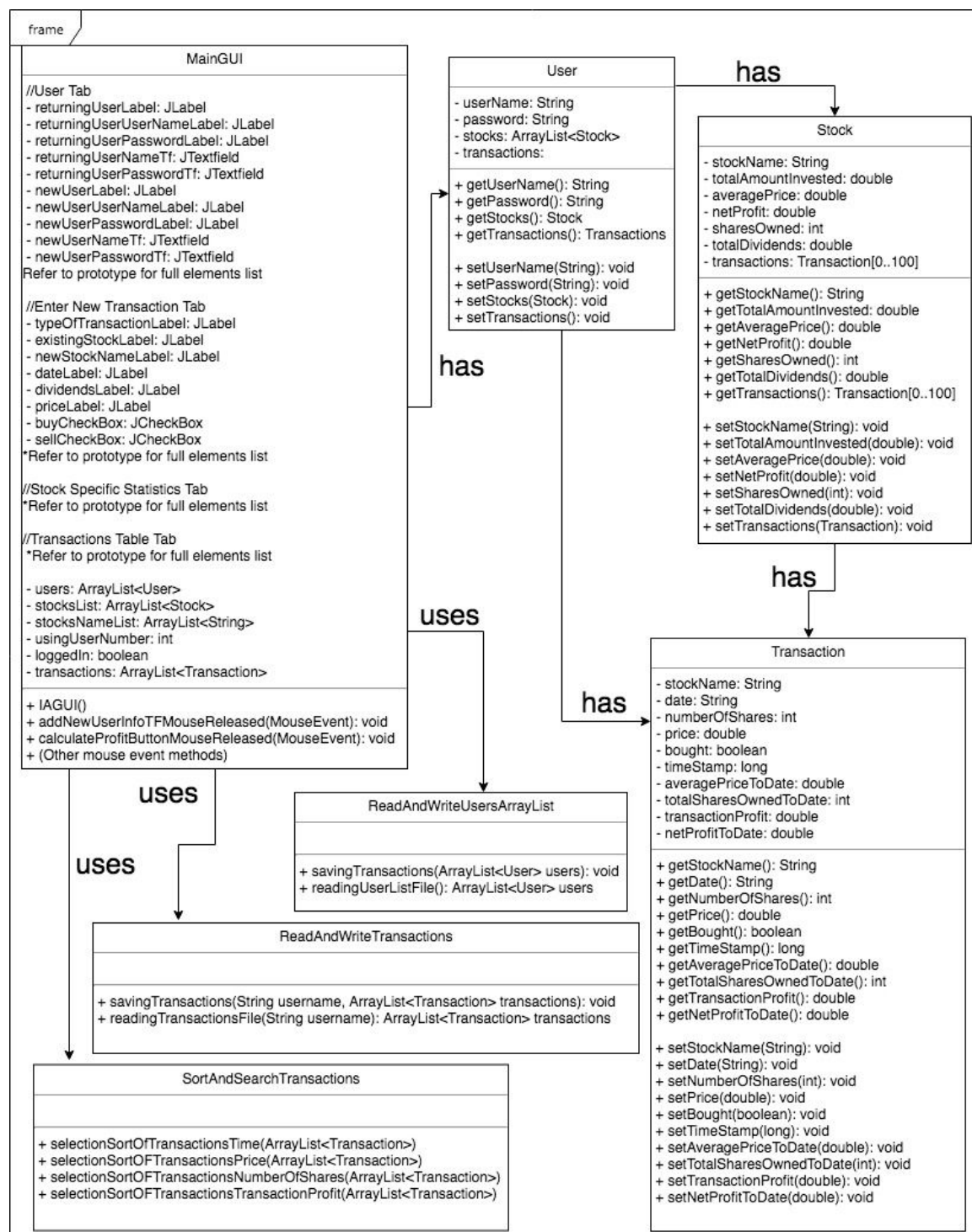
Current Average Price

Total Dividends to Date

Sell (Stock Name) (No. of Shares) at (Price)

Profit:

Class Diagram:



*In the process of programming, it was more appropriate to skip the stock aggregation step and have user have transactions instead of user has stocks which has transaction. The UML was created before the program was made. In program, stocks class is not utilized, and initially User and Transaction class did not have a direct aggregation relationship, but in the programming process, it was easier to just have user have transactions (1 step aggregation), rather than have User have stocks have transactions (2 step aggregation).

Chronological Development:

1. Interview Client - Create GUI
 - a. First interview
 - i. Create tabbed pane for user login for security purposes
 - ii. Create 2nd tabbed pane for entering transactions
 - iii. Create 3rd tabbed pane for display specific stock statistics
 - iv. Create 4th tabbed pane to display custom transaction table
 - b. Second interview
 - i. Rework GUI, change information to display on transactions table to suit clients needs/add new features users might think of
2. Create template classes
3. Create sorting class
4. Programming
 - a. User login tab
 - i. Create username
 - ii. Login
 - iii. Delete account
 - b. Add transaction tab
 - c. Display specific stocks tab
 - d. Display transactions details table tab
 - e. Error Exceptions
5. Create classes for sorting methods/functions
 - a. Methods for sort according
 - i. Date (1 for min to max, 1 for max to min (2 in total))
 - ii. Price (1 for min to max, 1 for max to min (2 in total))
 - iii. Number of shares (1 for min to max, 1 for max to min (2 in total))
 - iv. Transaction profit (1 for min to max, 1 for max to min (2 in total))
 - b. Sorting algorithm

```
Loop for int i from 0 to size of arraylist transactions - 1
    minIndex = i
    Loop for int j (= i + 1) from 0 to size of arraylist transactions
        If(transactions.get(i) < (transactions.get(minIndex)
            minIndex = j
        End if
        If minIndex != i
            Transaction temp = transactions.get(i)
            set transactions.get(i) = transactions.get(minIndex)
            set transactions.(minIndex) = temp
        End if
    End loop
```

6. Program methods for saving methods (reading and writing information)
 - a. Reading and writing transactions class
 - i. Method for saving transactions

- ii. Method for retrieving saved transaction data
- b. Reading and writing users information class
 - i. Method for saving stocks name list
 - ii. Method for retrieving saved stock name list data
 - iii. Method for saving users list
 - iv. Method for retrieving saved users list
- 7. Implement saving methods into program
- 8. Execute testing plan to test program
 - a. Fix errors
 - b. Program missed error handlings
- 9. Final interview with client
 - a. Instruct client on how to use program
 - b. Receive feedback
 - c. Fix unaccounted error handling issues

Word Count: 0 (Bulleted List)

Testing Plan:

Testing plan, ideally, will go through essential success criterias outlined in Criterion A document.

- Check the login page to see if read and write of users list and transactions ArrayList is working properly.
 - Test using the program to see if there are errors that causes program to stop running.
- Check if transactions entry work as it should.
 - Enter transaction entry, but instead of the textfields clearing as the actual program would work (in anticipation of a new transaction entry), it would display the variables of transaction itself onto the appropriate textfield using the `transaction.get(MostRecent).getVariable()` method to check whether input of variable functions properly.
- Check if the variables of the transactions that require calculations work properly. Check if display table is working properly.
 - By entering one buy transaction and one sell transaction and checking whether the data on the display table is correct, specially those that require calculations within setting the transaction parameters such as average price to date, number of shares to date and net profit.
- Check if information displayed on stock specific statistics functions properly
 - Enter two buy transactions with the same stock name.
 - Look at the information on the last transaction of the stock on the display table.
 - Display stock specific statistics on stock specific statistics page and see if data matches that of last transaction listed on transaction table of that stock.
- Check if the theoretical selling extra feature in the stock specific statistics tab is working properly.
 - Enter a buy transaction, check the statistics of specific stocks.
 - Enter made up price and amount to sell and check if value matches manual calculation.
- Check if the sorting of the data in the transaction table works.

- Enter transactions with different values of price, date, number of shares and transaction profit.
- Check save function.
 - Click the save button and go check if files are created to store transactions and users list data.
- Check the read function
 - After saving some users and transactions, relaunch program and login and see whether saved transactions show up on program correctly.
- Error Checking
 - //User Login Tab
 - Entering any data before login
 - Error message: Login Required
 - Create account with a username already used
 - Create account with 'name', try to create another account with 'name' as username. Result: Error message
 - Enter confirm password different to new password when making new user.
 - Error message: New Password and Confirm Password does not match
 - Create account without username
 - Error message: Please enter username
 - Enter username that doesn't exist
 - Error message: Invalid Username, username does not exist.
 - Login with wrong password
 - Error Message: Incorrect password, please try again.
 - //Enter Transactions Tab
 - Leave any combination of stock name, number of shares, dividends per share and price textfield blank.
 - Error Message: Please enter all transaction information
 - Enter transaction without checking buy or sell
 - Error message: Please select either buy or sell
 - Enter transaction without checking existing stock or new stock
 - Error message: Please check either 'existing stocks' or 'new stock' check box
 - Enter a sell transaction that sells more shares of a stock than owned
 - Error message: Selling more shares than owned
 - //Stock Specific Statistics Tab
 - Display stock that does not exist in database
 - Error Message: No data for stock

Word Count: 150 (Bulleted List not included)

Document Total Word Count: 247

Criterion A + B Word Count: 646