2009

Stock Market Fantasy Game

Presented by:

Alex Sood, John Grun, Kevin Folinus, Chris Zalewski & David Meng.

2/20/2009

	John Grun	Kevin Folinus	Chris Zalewski	Alex Sood	David Meng	
Project management	25%	25%	25%	25%		
Sec.3: Customer Statement of Requirements		100%				
Sec.4: Glossary of Terms		100%				
Sec.5: Functional Requirements Specification				100%		
Sec.6: Nonfunctional Requirements			100%			
Sec.7: Domain Analysis	100%					
Sec.8: User Interface Design		50%	50%			
Sec.9: Plan of Work	100%					
Sec.10: References		100%				

Table of Contents

1. Customer Statement of Requirements:

In this project we are designing a stock market fantasy league in which users can compete against one another in the stock market. The final design will mimic real stock brokerage services for the self-directed investor, such as E-Trader. It will, however, not deal with real money and instead be a platform for individuals to test their investment prowess in the stock market against one another without risk. We are building an application to be run off of an individual's desktop, which will enable them to virtually participate against users on other stations. When a user joins the game they will have an account created specific to them. These users will be given a specific amount of virtual money when they first join. They will then use this money and make real time investments in the stock markets in an attempt to out invest their competition. A real time investment means that any stock purchased or sold will be done so at the current market value. To create a program that satisfies these user specifications there are three main components. These components can be broken down into three areas, one front-end and two back-end. The front-end specification is the user interface. This will enable the user to control his account and the investment decisions he wants to make. The projects two back-end items are the back-end database and the code to access the real time stock prices. With these three items in place users will be able to buy and sell securities via our electronic trading platform.

Front-End

User Interface

- Login / New User
- Account Management
- Trade Execution
- Stock Value Searches
- Stock Market News
- Leaderboard

Back-End Operations

Database

- Login/Password Check
- Stock Records
- Trading Support
- Trading Support
- Portfolio Evaluation

Stock Retriever

- Immediate Stock Retrieval
- Periodic Price Updates

The simplest aggregate of means by which users interact with a piece of software is a user interface. The user interface is the front-end to our application. In our program the first form this will take, when individuals fist access the stock market fantasy league, will be a login screen. They will be faced with two options, log on using an existing account or to create a new account. When the user types in an incorrect account name or password an error will be returned and the user will get to try again. If they choose to create a new account, their new account will be added to the database. The second user interface gives the users the ability to make stock interactions. The current plan is to give the user the ability to check a ticker price, buy, sell, view leader board, and view the current stocks they own.

A primary aspect to our application will be the back-end database. This relational database will use a schema in which two are connected through a primary key. The primary key will be the username.

DT_Login_PW

LOGIN	PASSWORD	VM
//Kevin	//Secret	//$$

DT_User_Info

TICKER	QUANTITY	LOGIN
//goog	//50	//Kevin

This database schema satisfies the user requirements by enabling individuals to invest with their personal account, keeping it separate from other users. With a primary key that is checked against a password we eliminate the possibility that our code has an error causing multiple users to accounts to get modified by the incorrect user. This functionality is inherently critical in a stock market virtualization for security purposes. The back-end database is a crucial user requirement that will never be visible to the user themselves.

A key tool in implementing the user requirements is the programs ability to retrieve real time stock prices, at intervals and upon request. This is our second back-end item in our application. Without this the program would not be able to run a realistic stock market virtualization. When a user enters a stock ticker they would like to buy, sell, or view, our application will return the real value from yahoo finance. In addition to this stock ticker pull from the internet, later builds of our code will attempt to incorporate RSS news feeds related to the ticker inquired to give additional information to the user.

These three items come together to create an environment that satisfies all of the customer requirements. When a user logs in they will have access to their specific stock history and account balance. When a user goes to use an option from the user interface both the database and stock price updater code will be used to create a seamless runtime virtual stock market environment satisfying all of the customer needs.

2. Glossary of Terms

1. Terms

Login Database – This is the database containing the users log in and password.

Stock Database – This is the database containing the stocks owned by individual investors.

Login Interface – The screen users use to log in or create a new account.

Functional Interface – The primary interface used to interact with the stock market.

Yahoo Finance Library – Library which parses Yahoo Finance with Tickers for stock values.

Transaction – A transaction is when a security is either purchased or sold.

Security – In our program this will only be used to represent stocks.

Buy – Purchasing a stock at the current market value.

Sell – Selling a stock at the current market value.

Trade – A securities transaction.

Ticker – The symbol used to represent a company in the stock market.

Quantity – The amount of shares of a stock bought or sold.

Virtual Wallet – The amount of virtual money a participant has to spend on securities.

3. Functional Requirement Specifications

Stakeholders:

Potential stakeholders in this project include people who enjoy online fantasy games, people who want to test various investment strategies before assuming the actual risk of investing, as well as anyone who wants to learn the basics of how investing works. Additionally, educators are potential stakeholders as they may wish to use this product to teach students about the stock market. The nature of this product also makes it an excellent spot for investment firms to place advertisements, so they hold a stake in this as well.

Actors and Goals:

Actor 1: New Player
Goal: To create a new account and start playing the game.
Use Case: Initiates Create Account (UC-1).

Actor 2: Finished Player
Goal: To quit playing the game.
Use Case: Initiates Remove Account (UC-2).

Actor 3: Player
Goal 1: To login to an existing player account.
Use Case: Initiates Login (UC-3).

Goal 2: To view the stocks and cash currently owned by this player.
Use Case: Initiates View Portfolio (UC-4).

Goal 3: To purchase shares of a stock.
Use Case: Initiates Buy Stock (UC-5).

Goal 4: To sell currently owned shares of a stock.
Use Case: Initiates Sell Stock (UC-6).

Goal 5: To view the current price of a stock.
Use Case: Initiates View Stock Price (UC-7).

Goal 6: To view additional information regarding a stock.
Use Case: Initiates View Stock Details (UC-8).

Goal 7: To compare the value of this player's portfolio to the highest valued portfolios in the game.
Use Case: Initiates View Leaderboard (UC-9).

Goal 8: To set price thresholds for the player to be automatically notified when a certain stock exceeds them.
Use Case: Initiates Set Alert Thresholds (UC-10).

Goal 9: To view an alert regarding a stock price.

Use Case: Participates in Send Alert (UC-11).

Goal 10: To logout of the current account.
Use Case: Initiates Logout (UC-12).

Actor 4: Account Database
Goal 1: To insure that a new account does not share a name with an existing account and add the new account to the database.
Use Case: Participates in UC-1.

Goal 2: To completely remove all information of the account being deleted.
Use Case: Participates in UC-2.

Goal 3: To verify that the username-password combination matches an existing account and grant access to that account.
Use Case: Participates in UC-3.

Goal 4: To provide accurate information regarding the contents and/or value of player portfolios.
Use Cases: Participates in UC-4 and UC-9.

Goal 5: To update information regarding the contents and/or value of a player's portfolio.
Use Cases: Participates in UC-5 and UC-6.

Goal 6: To provide accurate information regarding a stock price.
Use Case: Initiates Update Price (UC-13) and UC-11. Participates in UC-5, UC-6, and UC-7.

Actor 5: Cell Phone
Goal: To receive an alert regarding a stock price.
Use Case: Participates in UC-11.

Actor 6: Yahoo! Finance
Goal: To provide up-to-date information regarding stock prices.
Use Cases: Participates in UC-8, and UC-13.

Use Cases:

The final product will implement the following 12 use cases:

Create Account (UC-1): This use case allows a new user to create an account with a selected username and password. A new portfolio containing only cash is created for that account. UC-3 is also carried out automatically after the account has been created.

Remove Account (UC-2): This use case allows the user to delete the account to which he or she is currently logged in. All of the account information in the database is deleted, and UC-12 is automatically carried out.

Login (UC-3): This use case allows the user to access an account if he or she provides the correct username and password. In that case, UC-4 is carried out. If incorrect

login information is supplied, the system displays an error message to notify the user.

View Portfolio (UC-4) (sub-use case): This use case allows the user to view the contents and value of his or her portfolio. This use case is not directly selected by the user; it is automatically carried out when the user initiates UC-3, UC-5, or UC-6.

Buy Stock (UC-5): This use case allows the user to buy shares of a stock. The user enters the stock name and the number of shares he or she wishes to buy, and the total price of the purchase is calculated plus a commission equal to a small percentage of the price. If the user has enough money, the transaction is carried out; otherwise the system displays an error message to notify the user that there is not enough money. UC-4 is carried out upon completion of the transaction.

Sell Stock (UC-6): This use case allows the user to sell shares of a stock they currently own. The user enters the stock name and the number of shares he or she wishes to sell. If the user does not own that many shares of that stock, the system displays an error message; otherwise the income from the sale is computed minus a commission equal to a small percentage of the sale, and the transaction is carried out. UC-4 is carried out upon completion of the transaction.

View Stock Price (UC-7): This use case allows the user to view the current price of a given stock. The user enters the company name or ticker symbol for the stock they wish to track, and the price of the stock is displayed along with how much the price has changed in the day's trading. These numbers are automatically updated. If the user enters an invalid company name of ticker symbol, an error message is displayed.

View Stock Details (UC-8): This use case allows the user to view additional information about a stock. The user enters the company name or stock ticker symbol and is provided with an interactive display which the history of the stock price and or trade volume over a selected period of time. If the user enters an invalid company name of ticker symbol, an error message is displayed.

View Leaderboard (UC-9): This use case allows the user to see who has the highest valued portfolios. The user selects an integer, n, and the usernames and values corresponding to the highest n valued portfolios are displayed. If the user is not among the leaders, his or her rank will be at the bottom of the displayed.

Set Alert Thresholds (UC-10): This use case allows the user to set thresholds for selected stocks, so that he or she will be notified if the stock price crosses that threshold. The user enters a stock name along with a lower and/or upper threshold. The user must also enter a cell phone number for the alert to be sent to.

Send Alert (UC-11): This use case allows the user to be notified when alert thresholds are crossed. When the system finds that a threshold has been crossed, it sends an alert to the cell phone number provided by the user.

Logout (UC-12): This use case allows the user to log out of the account to which he or she is currently logged in. The user clicks a logout button, and the system returns to the login screen. However, screens opened with UC-7 and UC-8 will remain open.

Update Price (UC-13): This use case is initiated periodically by the Account Database in order to keep its list of stock prices up-to-date. Current stock prices are retrieved from Yahoo! Finance.

Why are Login and Logout use cases?: The action of logging in is not ordinarily considered a use case because the user typically has some other goal in mind and does not plan on just logging in. However, that is not necessarily the case with this system. A

competitive player who knows how quickly stock prices can change may wish to login just to ensure quick access to the account later. In this case, logging in is achieving a goal of the user by itself and should therefore be considered a use case. Similarly, logging out achieves the goal of a security-concious player to protect his or her account from unauthorized access.

We will now proceed with a more detailed description of use cases UC-1, UC-2, UC-3, UC-4, UC-5, UC-6, UC-7, UC-12, and UC-13.

Use Case UC-1: Create Account
Related Requirements:
Initiating Actor: New Player
Actor's Goal: To create a new account with a new portfolio.
Participating Actors: Account Database
Preconditions: None.
Postconditions: 1. A new account is created with the username and password supplied by the New Player.
2. A new portfolio containing only cash is created for the new account.
3. All postconditions of UC-3.

Flow of Events for Main Success Scenario:
1. System prompts the New Player for a username and password.
2. New Player enters a username and password.
3. System sends username to the Account Database.
4. Account Database confirms that the username is not taken.
5. System enters username and password into the Account Database.
6. System enters the starting amount of cash into username's account in the Account Database.
7. System displays account screen with buttons to initiate other use cases.
8. include:: View Portfolio (UC-4).

Flow of Events for Extensions:
2. New Player fails to enter either username or password.
    1. System displays error message telling New Player enter the missing information.
    2. Start over from Step 1.
4. Account Database finds that the username is taken.
    1. System displays an error message telling New Player that the chosen username is already taken.
    2. Start over from Step 1.

Use Case UC-2: Remove Account
Related Requirements:
Initiating Actor: Finished Player
Actor's Goal: To delete his or her current account.
Participating Actors: Account Database

Preconditions: 1. The Finished Player is logged in to an existing account.
Postconditions: 1. All information regarding the account is question is deleted from the Account Database.
2. All postconditions of UC-12.

Flow of Events for Main Success Scenario:
1. System prompts Finished Player for the account password.
2. Finished Player enters the correct password.
3. System sends the password to Account Database.
4. Account Database confirms that the password is correct.
5. System removes all account information from Account Database.
6. include:: Logout (UC-12).

Flow of Events for Extensions:
2. Finished Player enter incorrect password.
    1. System sends the password to Account Database.
    2. Account Database finds that the password is incorrect.
    3. System displays an error message telling Finished Player that the password was incorrect.
    4. System returns to account screen.

Use Case UC-3: Login
Related Requirements:
Initiating Actor: Player
Actor's Goal: To access an existing account.
Participating Actors: Account Database
Preconditions: 1. The Account Database is non-empty.
Postconditions: 1. The System displays the account screen which provides buttons for Player to initiate all other use cases except UC-1 and UC-4.
2. Also displayed on the account screen are the current contents and value of Player's portfolio.

Flow of Events for Main Success Scenario:
1. System prompts Player for a username and password.
2. Player enters a correct username and password.
3. System sends the username and password to Account Database.
4. Account Database confirms that the username exists and the password is correct.
5. System retrieves the contents of Player's portfolio from Account Database.
6. System displays account screen with buttons to initiate other use cases.
7. include:: View Portfolio (UC-4).

Flow of Events for Extensions:
2a. Player fails to enter either username or password.
    1. System displays an error message telling Player to enter the missing information.
    2. Start over from Step 1.

2b. Player enters nonexistent username or incorrect password.
1. System sends the username and password to Account Database.
2. Account Database finds that the username is nonexistent or the password is incorrect.
3. System displays an error message telling Player that the login information was incorrect.
4. Start over from Step 1.

Use Case UC-4: View Portfolio (sub-use case)
Related Requirements:
Initiating Actor: Player
Actor's Goal: To view the contents and/or value of his or her portfolio.
Participating Actors: Account Database
Preconditions: 1. Player has initiated either UC-3, UC-5, or UC-6.
Postconditions: 1. The contents and value of the portfolio is displayed on the account screen.

Flow of Events for Main Success Scenario:
1. System prepares a list of stocks owned and the number of shares owned of each stock.
2. System retrieves the current price of each stock from Account Database and calculates the value of the portfolio.
3. System displays the list of stocks, cash balance, and total value of the portfolio on the account screen.

Flow of Events for Extensions:

Use Case UC-5: Buy Stock
Related Requirements:
Initiating Actor: Player
Actor's Goal: To purchase shares of a stock.
Participating Actors: Account Database
Preconditions: 1. Player is logged into an existing account.
2. Player's cash balance is non-zero.
Postconditions: 1. Player's portfolio is updated with the new shares owned and cost of the transaction is deducted from the cash balance.
2. Changes to the portfolio are reflected on the account screen.

Flow of Events for Main Success Scenario:
1. System prompts Player for a stock name and a number of shares to buy.
2. Player enters a valid company name or ticker symbol and an affordable number of shares.
3. System attempts to retrieve the stock price from Account Database.
4. Account Database finds a match for the stock name and returns the price.
5. System calculates the total cost and adds the commission.
6. System verifies that this value is less than or equal to Player's cash balance.
7. System adds the shares to Player's portfolio and deducts the cost from Player's cash

balance.
8. System updates Account Database with the new values.
9. include:: View Portfolio (UC-4).

Flow of Events for Extensions:
2a. Player fails to enter either stock name or number of shares.
1. System displays and error message telling Player to enter the missing information.
2. Start over from Step 1.
2b. Player enters invalid company name or ticker symbol.
1. System attempts to retrieve the stock price from Account Database.
2. Account Database cannot find a match for the stock name.
3. System displays an error message telling Player that the stock name is invalid.
4. Start over from Step 1.
2c. Player enters valid stock name with an unaffordable number of shares.
1. System attempts to retrieve the stock price from Account Database.
2. Account Database finds a match for the stock name and returns the price.
3. System calculates the total cost and adds the commission.
4. System finds that this value is greater than Player's cash balance.
5. System displays an error message telling Player that he or she does not have enough money for this transaction.
6. Start over from Step 1.

Use Case UC-6: Sell Stock
Related Requirements:
Initiating Actor: Player
Actor's Goal: To sell shares of a stock he or she currently owns.
Participating Actors: Account Database.
Preconditions: 1. Player is logged into a existing account.
2. Player's portfolio shares of at least 1 stock.
Postconditions: 1. Player's portfolio is updated with the shares sold removed and the income from the transaction added to the cash balance.
2. Changes to the portfolio are reflected on the account screen.

Flow of Events for Main Success Scenario:
1. System prompts Player for a stock name and a number of shares to sell.
2. Player enters a valid company name or ticker symbol and a valid number of shares.
3. System verifies that Player owns the entered number of shares of the entered stock.
4. System retrieves the stock price from Account Database.
5. System calculates the total income and subtracts the commission.
6. System subtracts the shares from Player's portfolio and adds the income to Player's cash balance.
7. System updates Account Database with the new values.
8. include:: View Portfolio (UC-4).

Flow of Events for Extensions:

2a. Player fails to enter either stock name or number of shares.

1. System displays and error message telling Player to enter the missing information.
2. Start over from Step 1.

2b. Player enters a stock he or she does not own.

1. System fails to find the stock in Player's portfolio.
2. System displays an error message telling the Player that he or she does not own that stock.
3. Start over from Step 1.

2c. Player enters a number of shares greater than what he or she owns.

1. System finds that Player does not have enough shares of the stock.
2. System displays an error message telling Player that he or she does not own that many shares of the stock.
3. Start over from Step 1.

Use Case UC-7: View Stock Price

Related Requirements:

Initiating Actor: Player

Actor's Goal: To view the current price of a stock.

Participating Actors: Account Database

Preconditions: 1. The window for viewing stock prices has been opened.

Postconditions: 2. The current price of the stock and the change in the price for the current day are displayed in the viewing window.

Flow of Events for Main Success Scenario:

1. System prompts Player for a stock name.
2. Player provides a valid stock name.
3. System asks Account Database for the current price and change for the entered stock.
4. Account Database finds the stock on its list and returns the current price and change.
5. System displays the stock ticker symbol along with its price and change.
6. Steps 3-5 are repeated periodically to keep the displayed price and change current.

Flow of Events for Extensions:

2a. Player fails to enter a stock name.

1. System displays an error message telling Player to enter a stock name.
2. Start over from Step 1.

2b. Player enters an invalid stock name.

1. System asks Account Database for the current price and change for the entered stock.
2. Account Database fails to find the stock on its list.
3. System displays an error message telling Player that the entered stock name is invalid.
4. Start over from Step 1.

Use Case UC-12: Logout
Related Requirements:
Initiating Actor: Player
Actor's Goal: To log out of the current account.
Participating Actors: None.
Preconditions: 1. Player is logged into an existing account.
Postconditions: 1. The account screen is closed and the login screen is reopened.

Flow of Events for Main Success Scenario:
1. Player clicks logout button.
2. System closes the account screen.
3. System opens login screen prompting Player for username and password.

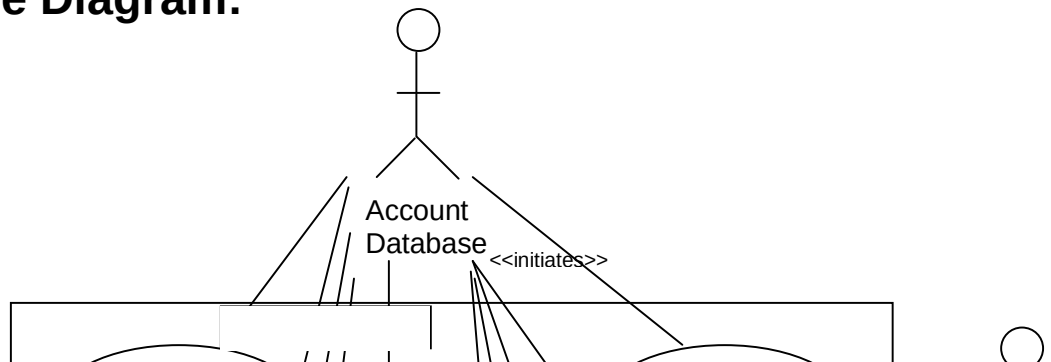
Flow of Events for Extensions: None

Use Case UC-13: Update Price
Related Requirements:
Initiating Actor: Account Database
Actor's Goal: To keep its list of stock prices up-to-date.
Participating Actors: Yahoo! Finance
Preconditions: 1. Account Database contains a non-empty list of stocks and their prices.
Postconditions: 2. The price of each stock is replaced with a new value from Yahoo! Finance.

Flow of Events for Main Success Scenario:
1. Account Database asks for the current price of the first stock on its list.
2. Sys retrieves the current price of that stock from Yahoo! Finance.
3. Account Database replaces its price for that stock with the new price.
4. Steps 1-3 are repeated for each stock on the list.

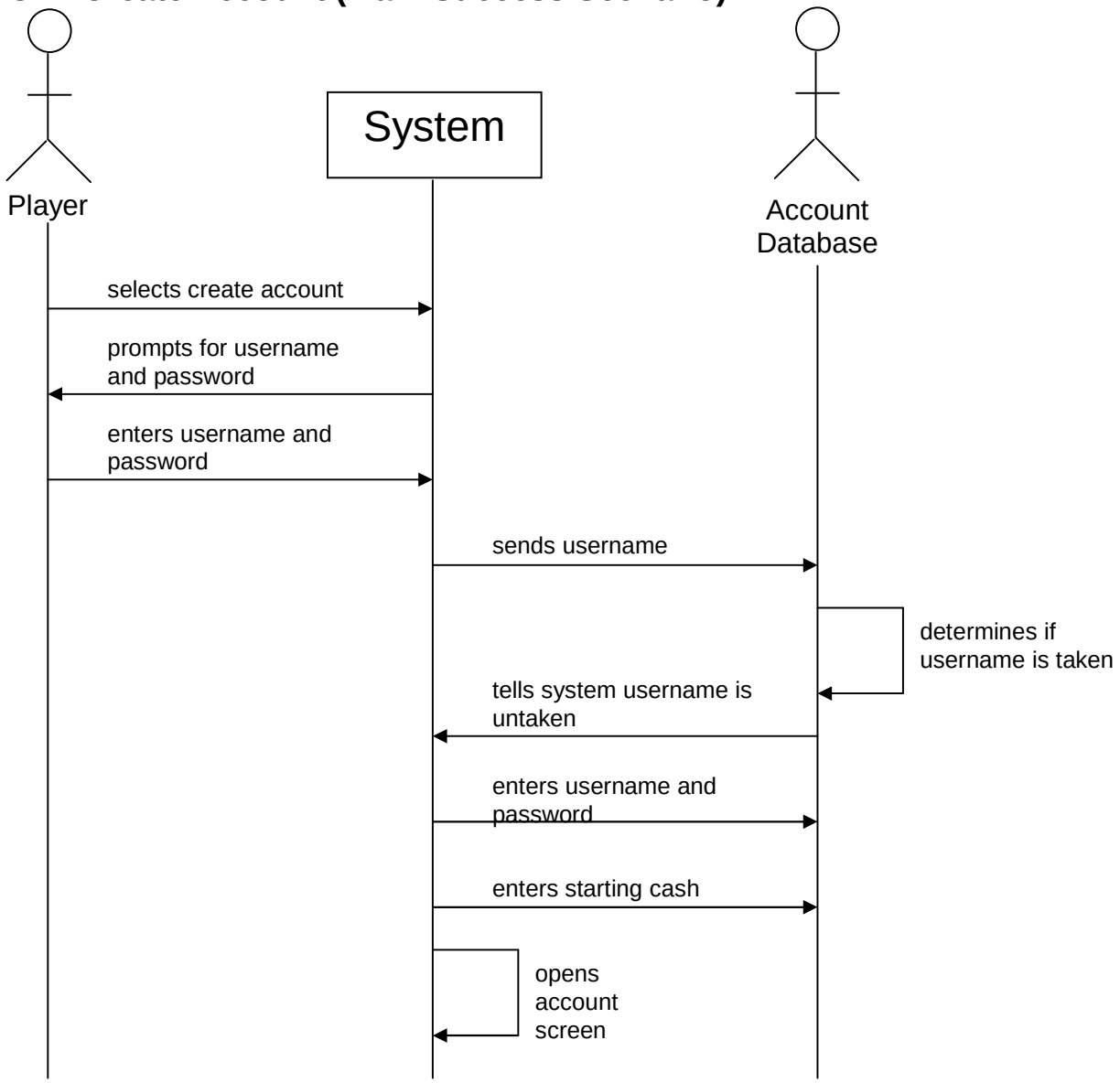
Flow of Events for Extensions: None

Use Case Diagram:

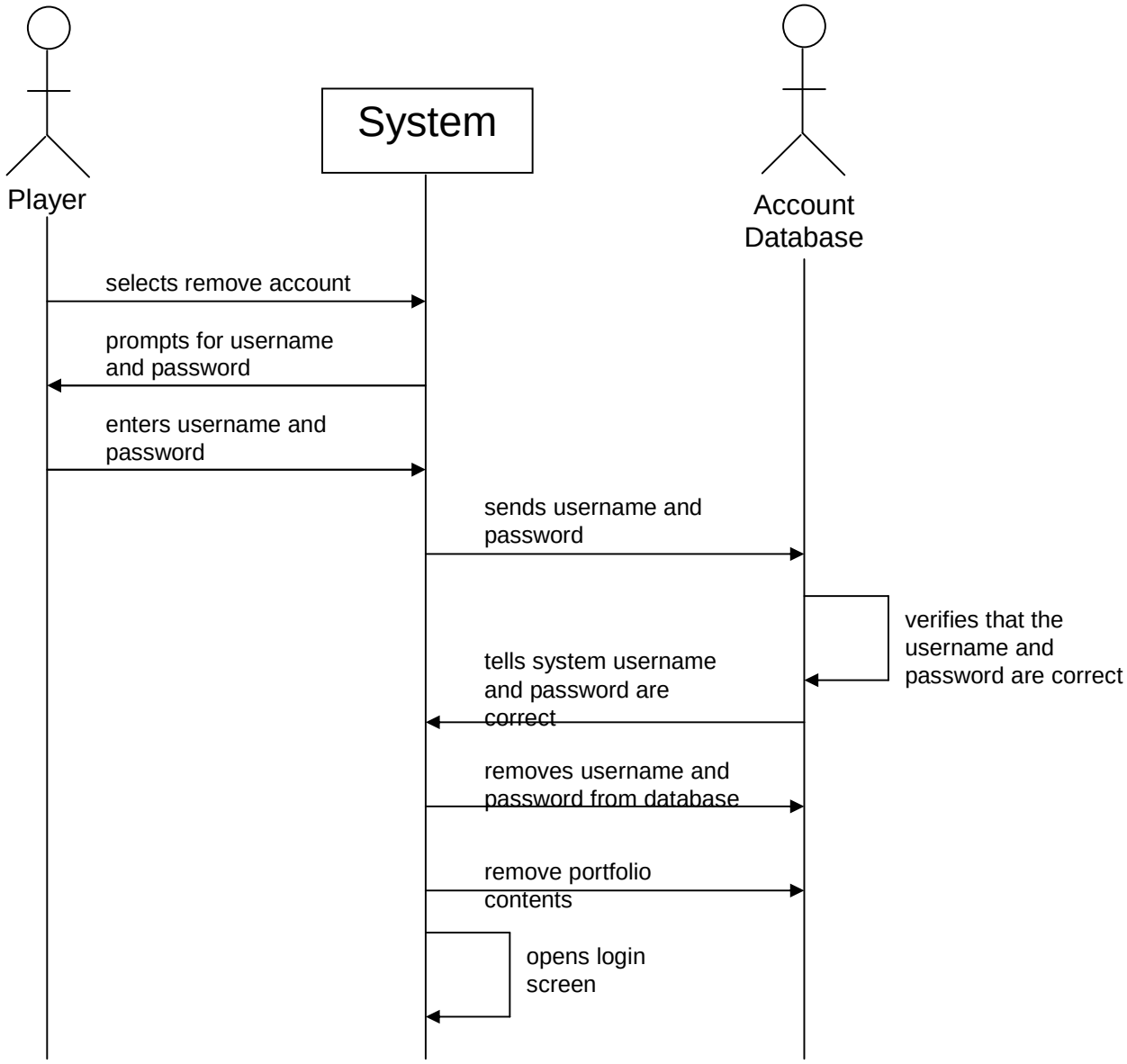


System Sequence Diagrams:

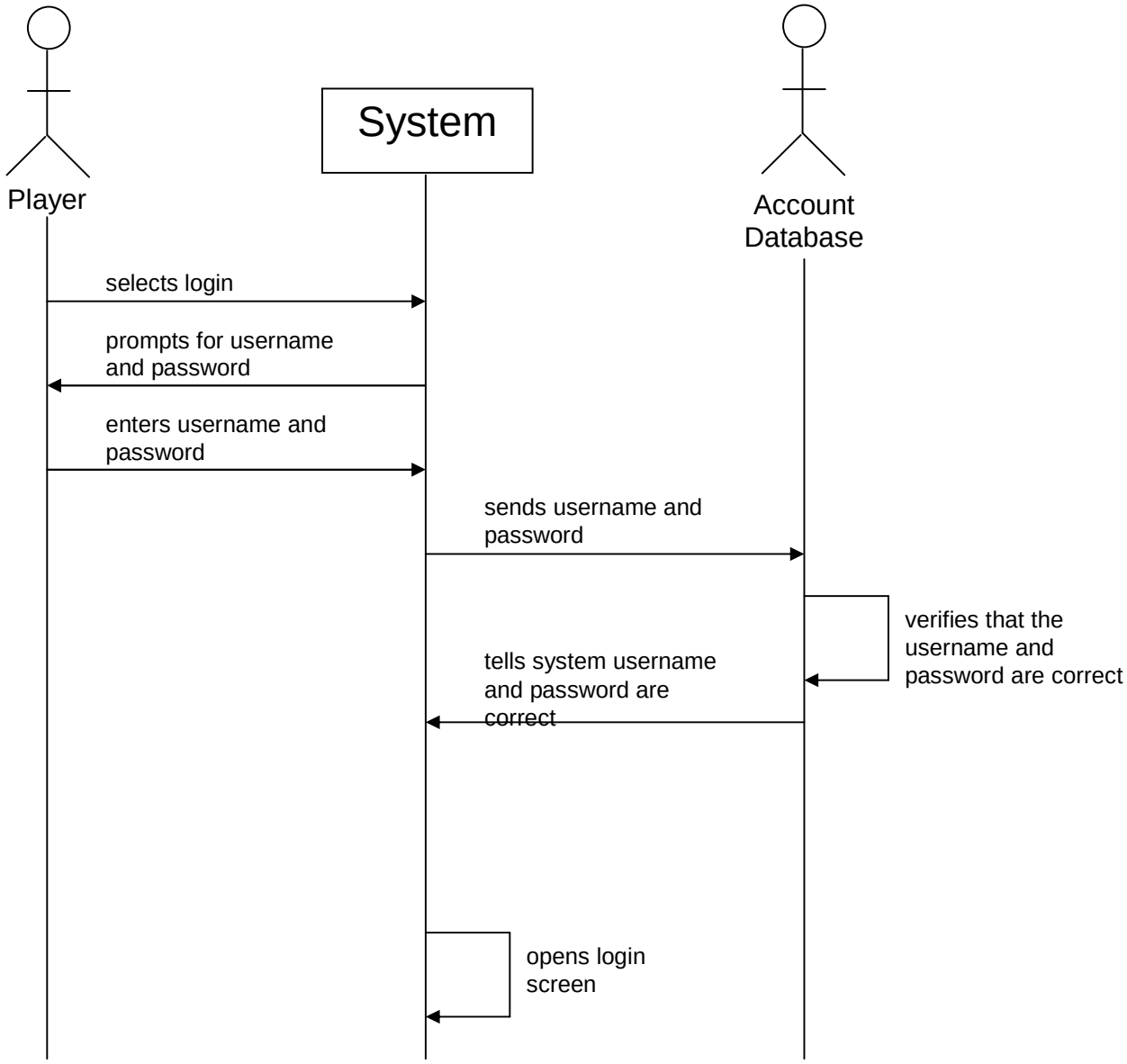
UC-1: Create Account (Main Success Scenario)



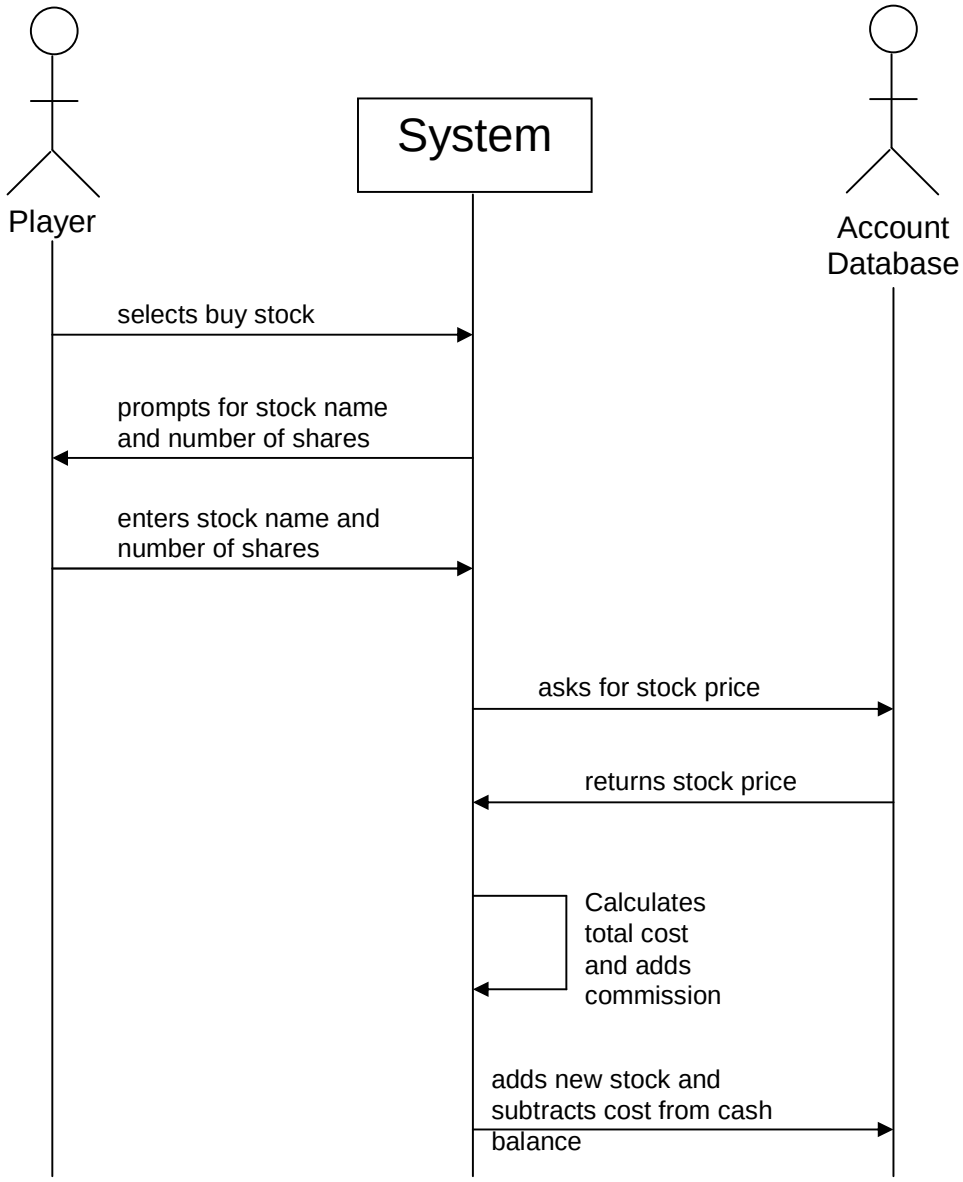
UC-2: Remove Account (Main Success Scenario)



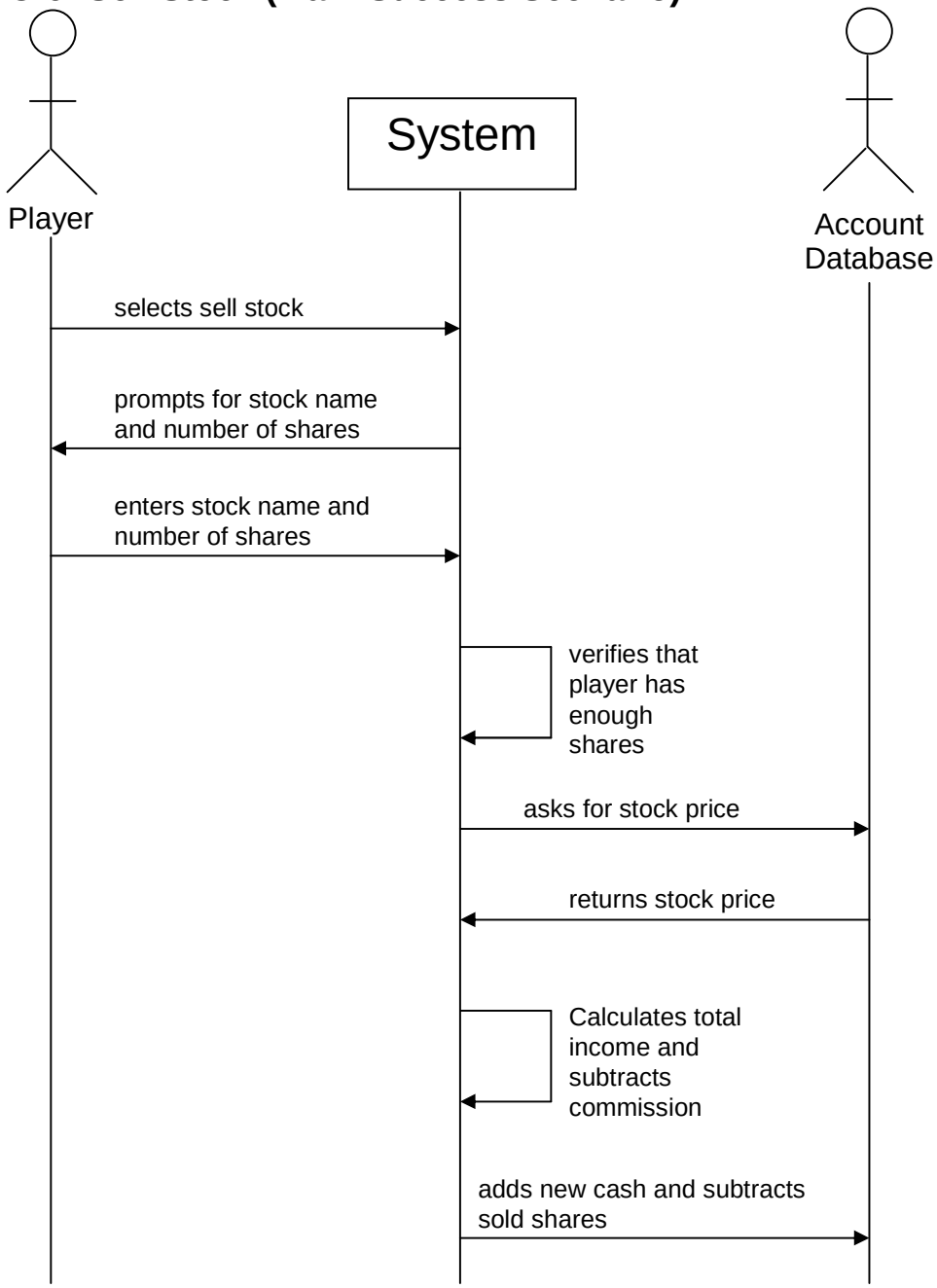
UC-3: Login (Main Success Scenario)



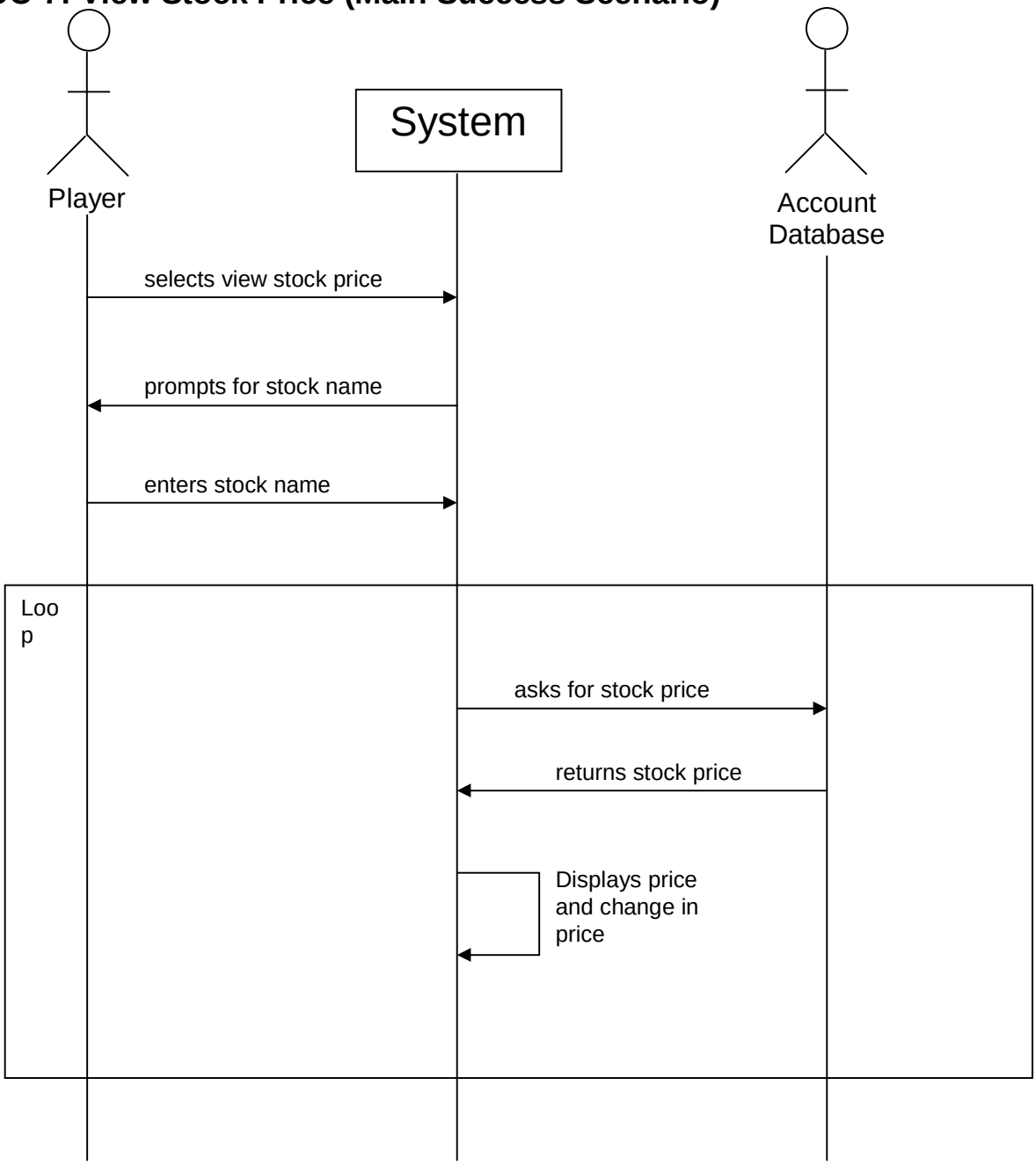
UC-5: Buy Stock (Main Success Scenario)


Player
System
Account Database
selects buy stock
prompts for stock name and number of shares
enters stock name and number of shares
asks for stock price
returns stock price
Calculates total cost and adds commission
adds new stock and subtracts cost from cash balance

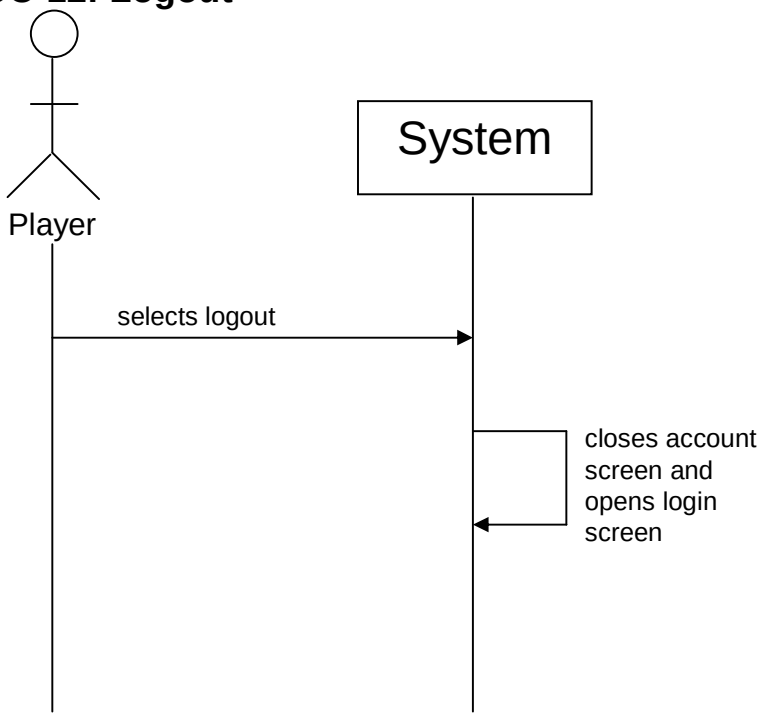
UC-6: Sell Stock (Main Success Scenario)



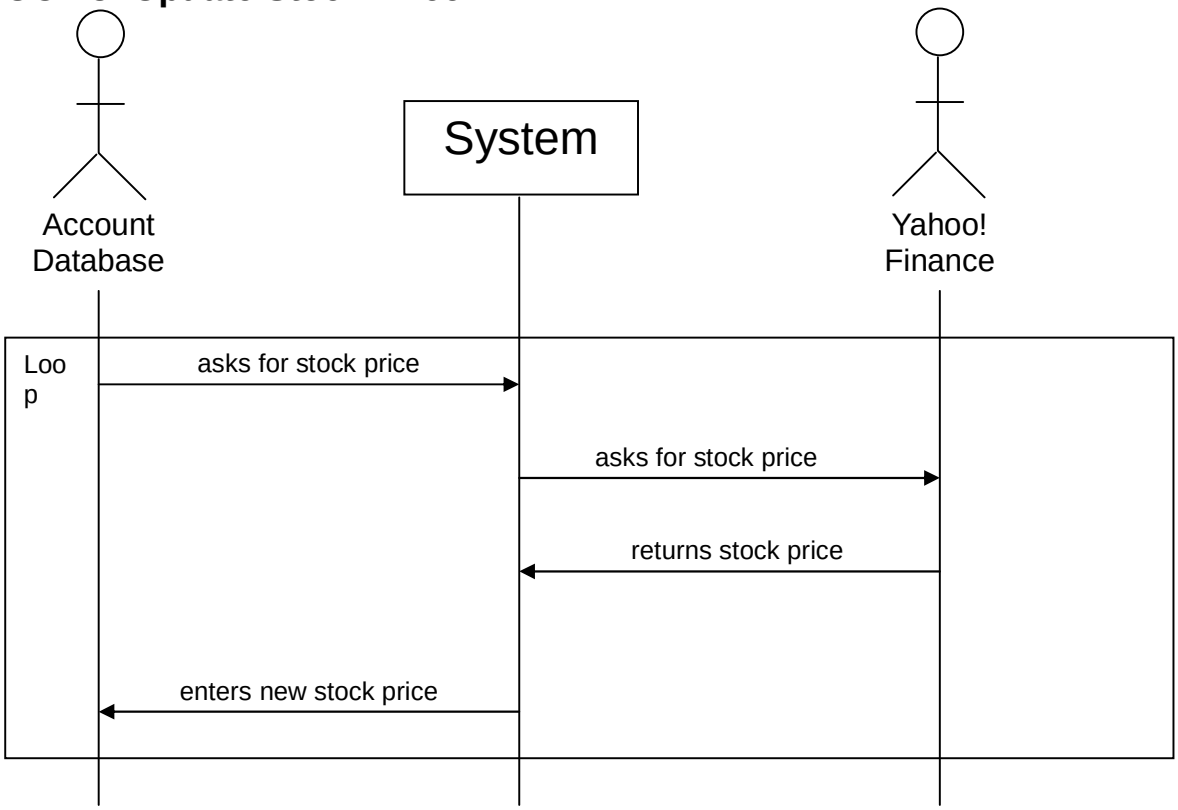
UC-7: View Stock Price (Main Success Scenario)



UC-12: Logout


System
Player
selects logout
closes account screen and opens login screen

UC-13: Update Stock Price



4. Nonfunctional Requirements

Functionality

FEATURE SET:

-One window with all options (provides easy access)
-Automated stock-ticker
-Automatically updated database
-Cell phone updates

CAPABILITIES

The program will be able to save and store portfolios automatically without the player requesting. Also, the stock tickers will automatically update in real time so there is no need constantly refresh it. It will be able to search any stock and send data to one's cell phone.

SECURITY

Logins with passwords are required, so one cannot simply change every player's data. Since the portfolios are stored in a data base, there is no way for other players to change information.

Usability

HUMAN FACTOR

The human factor is accounted for in our program, due to the minimal amount of clicks to get a player to their destination. There will be a very small amount of learning to figure out how to use the program. Most if not all of the buttons required for the program are on the main screen, so it will not be very hard to navigate through the program.

AESTHETICS

The program will have excellent readability and will look pleasing to the eye. It will be clear what each button's purpose is and there will only a few windows opened so there isn't much navigation required between windows.

CONSISTENCY

Since each option in the program is specific with familiar words, there will be no confusion in determining what each option does.

DOCUMENTATION

There will be instructions for the user to understand how to use the program and how the fantasy league works and the league's rules.

Reliability

FREQUENCY/SEVERITY OF FAILURE

There are a few possibilities of failures. One is if the internet connection of the player/user is interrupted or shut off. This could possibly hinder the player/user from viewing and/or changing their stocks. Their transactions may not have gone through, which if gone unnoticed can affect the portfolio of the player.

Another possible situation where a failure may occur is if the site, Yahoo! Finance, goes down. This is where the ticker information is extracted from in real time. If the site goes down, the players cannot see their updated stocks.

RECOVERABILITY

Since the player's most up to date portfolio is in a database, it is easy to recover in case of an error. If a transaction did not go through, there would be no way to recover the desired change in stocks.

ACCURACY

Since our program will update automatically, the stock prices will be accurate. Also, since the database updates when a player changes his or her portfolio, it will also be accurate for checking standings and one's own data.

Performance

Once searched, a stock will be constantly updated. Even when another stock is searched, the first stock and each one after will be constantly updated. This will help the speed of our program as well as the response time and resource consumption, since only some stocks will be updated instead of every one. Once the stock is found, it will auto update in the window until another stock is requested or the window is closed. Our database will be updated constantly as well, which can affect performance once it gets large.

Supportability

TESTABILITY

It is easy to create new data bases and mock players, so the program will have high testability.

EXTENSIBILITY

If new extensions are added, they should not be hard to implement. Changes in the user interface will have to be made to create new buttons if desired.

ADAPTABILITY

Since the program is based on the stock market, it contains terms widely used in the market. Therefore, a user will be able to adapt to the program easily since there aren't too many aspects to learn besides the fantasy league rules.

MAINTAINABILITY

The program should not be difficult to maintain. One difficulty may occur if the databases are changed to meet a new requirement, which can affect the way the code organizes the players' portfolios.

5. Domain Analysis

Concept Definitions

Concept Name	Responsibility	Type
Transaction	Contains all operations related to buying and selling of stock	D
Login	Controls all functions related to logging in, logging out and registering	D
Scoreboard	Displays the user's portfolio and the current top investors	D
Interface	All elements of the GUI	D
Fetch	Retrieves the current stock prices from the Internet (yahoo)	D
Database	Controls all functions of the SQL database including connecting to the server reading from the database and updating the database.	D
MYSQL	The databases that contains all data in the program including the login data and portfolio	K
Monitor	Monitors if a stock crosses a given threshold	D
Cell_Alert	Alerts an Investors cellphone	D

Association definitions

Association name	Association Descriptions	Location
Buy_Stock	Checks if a user has enough money to buy a stock, If yes the stock is purchased. The user's Virtual Wallet and portfolio are updated	UC-5
Sell_Stock	Sells the specified stock(if the user owns it) and updates the user's Virtual wallet and portfolio	UC-6
Price_Find	Displays the current price of a given stock	UC-7
Login	Logs a user in if they have an account in the database.	UC-1
Logout	Logs a user out	UC-12
Sign_Up	Creates a new user account	UC-1,UC-3
Leaderboard	Displays the names of the top investors, their rankings(in order) and their net worth	UC-9

Your_Data	Displays the user's current portfolio as well as net change	UC-4,UC-1,UC-5,UC-6
Connect_DB	Connects to the SQL database	ALL
Write_DB	Writes data to a specified database.	UC-1,UC-2,UC-5,UC-6,UC-13
Read_DB	Reads data from a specified database	ALL
Form1	GUI of the login screen	UC-1,UC-2,UC-3,UC-12
Form2	GUI of the Leaderboard and Your_Data	UC-4 -- UC-9
Timer	Timer to update the stock price database.	UC-13
Display_Top_Investors	Finds the top investors	UC-9
Update_DB	Updates the SQL database with the current price from the internet	UC-13
Download_SP	Downloads the current stock price from Yahoo!	UC-13
Remove_User	Removes user from the database and calls logout	UC-2
Send_Alert	Alerts investor that a something has passed the threshold	UC-10
Set_Alert	Function used to set the Alert_Threshold in Monitor	UC-10
Watch	Watches a stock for any change	UC-11
Call_Phone	Calls cellphone	UC-10

Attribute Name	Attribute Description
Login	The user names and the “primary key” in the database
PASSWORD	The user's password enabling them to access their account
VM	The current value of the user's assets
TICKER	Stock names
QUANITY	How many of a perticular stock the user owns
timeInterval	The amount of time between updating the database with the current stock prices
currentprice	The current price of stock
NetchangePortfolio	The net change in the user's portfolio
PriceChange	Price change of a given stock.
Logintrue	Used to determine if the login was successful
netprofit	User's net profit
alert_Thres	The threshold level specified by the investor for a given stock
Cell_number	Investor's cellphone number

Operation	**Create Account**
Preconditions	Check if the account name exist.
Postconditions	Create an new user account Set LOGIN,PASSWORD, and VM = $100000 Display Form2

Operation	**Remove Account**
Preconditions	Check if the account name exist.
Postconditions	If account exist remove data for LOGIN, PASSWORD, and VM Log User Out

Operation	**Login**
Preconditions	Check if LOGIN and PASSWORD are equal those stored in database
Postconditions	LOG user in Display Form2

Operation	**View Portfolio**
Preconditions	Call Your_Data
Postconditions	Display Form2

Operation	**Buy Stock**
Preconditions	Check if VM – currentprice*quantity is not less than 0
Postconditions	Update the database with the new values of VM, QUANTITY, and TICKER

Operation	**Sell Stock**
Preconditions	Check if the number of shares they wish to sell is not greater QUANTITY
Postconditions	Update the database with the new values of VM, QUANTITY, and TICKER

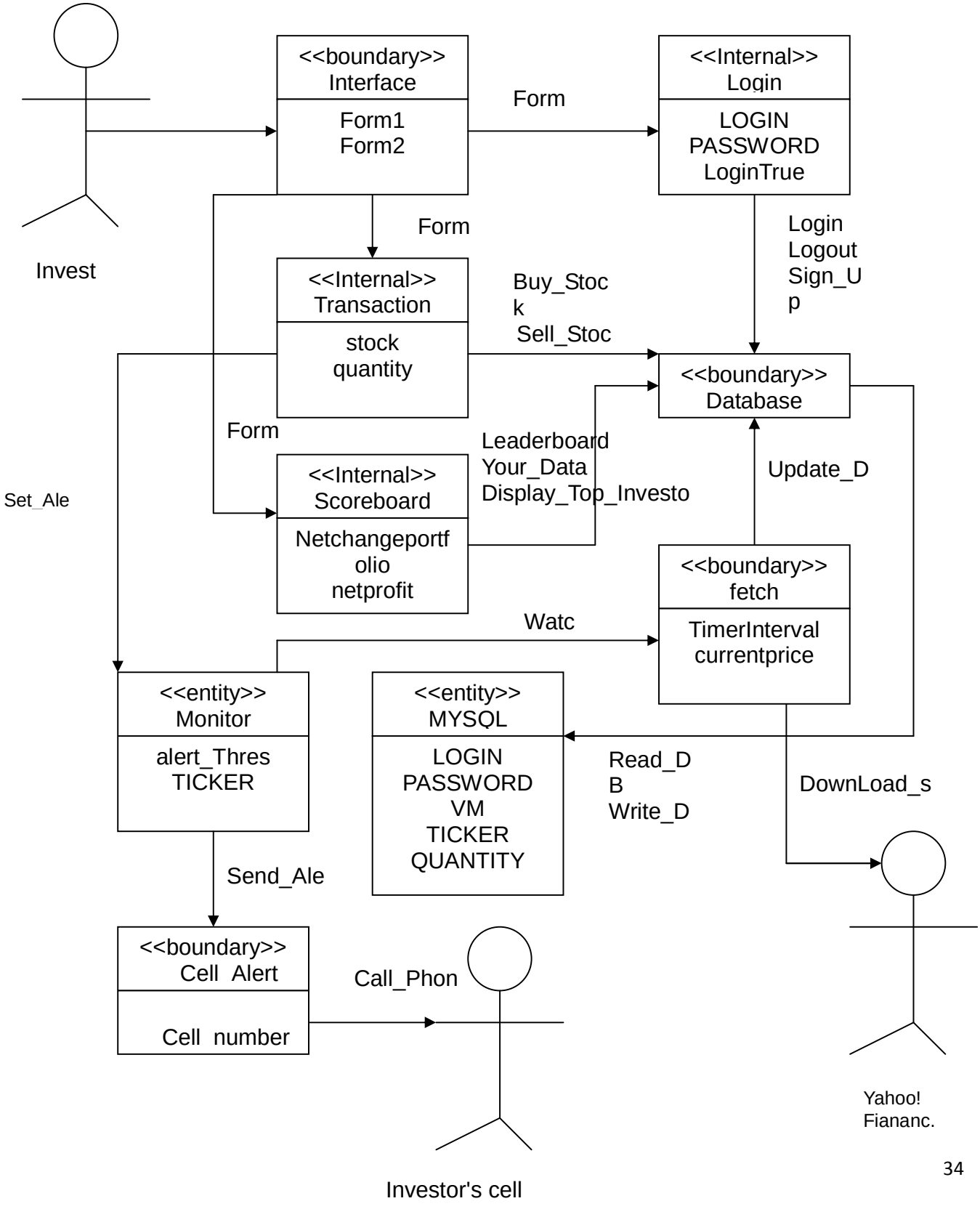
Operation	**View Stock Details**
Preconditions	Check if the entered name matches anything in TICKER
Postconditions	Display Form2 with the additional stock info

Operation	**View Leaderboard**
Preconditions	None
Postconditions	Display Form2 with the ranking information

Operation	**Logout**
Preconditions	None
Postconditions	Displays Form1(Login screen) User can exit from this point

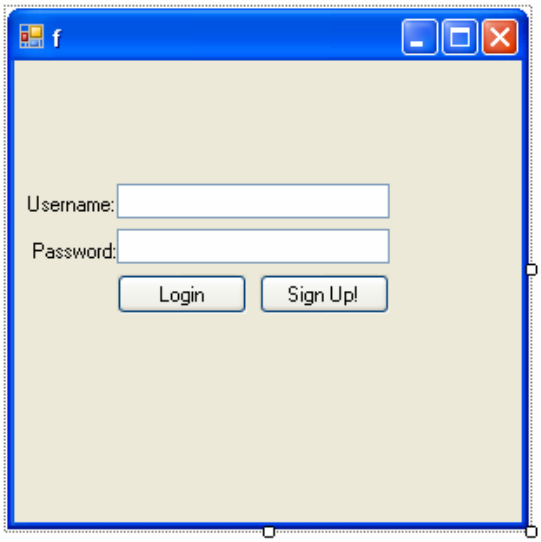
Operation	**Update Price**
Preconditions	Check timerInterval if has passed
Postconditions	Update stock prices in the database

Domain Model



6. User Interface

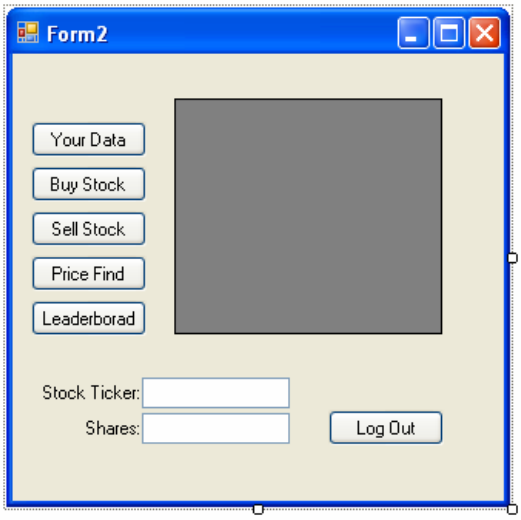
Login Screen Preliminary Design:



When a user first access our program they will encounter a login screen similar to the one displayed. They will then enter a username and password and login or click the sign up button. When the Login button is clicked the string typed into the username and password location will be checked against those existing in the database. The sign up button will then trigger another interface where people can create a new account. We will then check if this is a viable option for their account name and if so create the account.

Once the user has successfully navigated the login screen the second user interface will appear.

Main User Interface Preliminary Design:



The second user interface has the primary functionality of our program. With a successful login the user will now be able to use this interface to interact with their personal account. The empty box will display the stocks they currently own, their virtual wallet, and the net value of their account.

Button Functionality

Price Find: Searches the stock ticker string and will return the price of that stock.

Buy Stock: Checks if your fund are sufficient and if so purchases the specified number of shares.

Sell Stock: Checks if you own the stock and if so they are sold returning the value to your virtual wallet.

Leader Board: Changes the display from your data to a leader board.

Your Data: Returns the display to your data.

Log Out: Returns the user to the log in screen.

User Effort Estimation:

Buy Stock/Sell Stock
NAVIGATION
1 click
DATA ENTRY
a. Click to text window
b. Type stock's ticker
c. Type quantity of shares

Ticker Finder
NAVIGATION
1 click
DATA ENTRY
a. Click to text window
b. Type stock's ticker

Player Data/Leaderboard
NAVIGATION
1 click

Login
NAVIGATION
1 click

DATA ENTRY

a. Click to text window “Username”
b. Type Username
c. Click to text window “Password”
d. Type Password
e. Click “Login”

Signup

NAVIGATION

1 click

DATA ENTRY

a. Click to text window “Username”
b. Type new Username
c. Click to text window “Password”
d. Type new Password
e. Click “Confirm”

7. Plan of Work

Attached in email.

10. References

Brend Bruegge & Allen H. Dutoit. Object-Oriented Software Engineering. Upper Saddle River: Pearson Education, Inc., 2001.

Paul T. Tymann & G. Michael Schneider. Modern Software Development Using Java. Pacific Grove, Thomson, 2004.

Dan Pilone & Neil Pitman. ULM 2.0 In a Nutshell. Sebastopol, O'Reilly Media, Inc., 2005.

Cay S. Horstmann & Gary Cornell. Core Java: Volume 1 – Fundamentals. Santa Clara, Sun Microsystems Inc., 1959.

Wikipedi. Wikimeda Foundation. www.wikipedia.org

Microsoft Developer Network. Microsoft. Msdn.microsoft.com/en-us/default.aspx