TRADE FUN!

Group 6

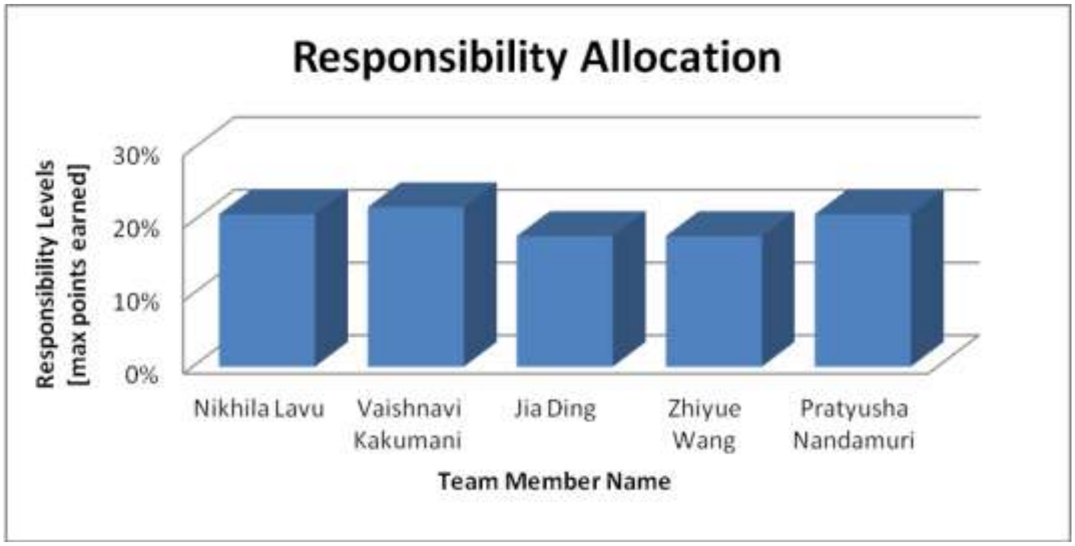
Jia Ding
Nikhila Lavu
Pratyusha Nandamuri
Vaishnavi Kakumani
Zhiyue Wang

Date of Submission: 11/4/2011
URL of the Project's Website:
https://sites.google.com/site/stocktradefun/

1. INDIVIDUAL CONTRIBUTIONS BREAKDOWN:

The effort breakdown for all team members is shown in the following responsibility matrix and responsibility allocation chart.

Responsibility Levels	Team Member Name				
	Nikhila Lavu	Vaishnavi Kakumani	Jia Ding	Zhiyue Wang	Pratyusha Nandamuri
Project management	30%	40%			30%
Cover Page and Individual Contributions Breakdown			100%		
Table of Contents		100%			
Interaction Diagrams				40%	60%
Class Diagram and Interface Specification		100%			
System Architecture and System Design	60%		40%		
Algorithms and Data Structures		100%			
User Interface Design and Implementation	100%				
Progress Report and Plan of Work		33.3%	33.3%	33.3%	
References				100%	



2. TABLE OF CONTENTS:

3. INTERACTION DIAGRAMS:

(a) List of Implemented Use Cases:

- Use Case 1: Login
- Use Case 2: Registration
- Use Case 3: View About Us
- Use Case 4: View About Me
- Use Case 5: Edit My Profile
- Use Case 6: View RSS Newsfeed
- Use Case 7: View My Game Portfolio
- Use Case 8: Go Home
- Use Case 9: Market Order - Buy
- Use Case 10: Market Order - Sell
- Use Case 11: View Trading History
- Use Case 12: View Help Documentation
- Use Case 13: View Trade Diary
- Use Case 14: View top five Gainers/Losers
- Use Case 15: View Most Active Companies
- Use Case 16: Delete Account
- Use Case 17: Logout

(b) Diagrams for Implemented Use cases:

Each interaction diagram that we depict below assumes that each page in our website is an object which communicates with the rest of our internal and external modules via function calls. The inputs necessary for the function calls are specified as parameters with the braces ‘()’.

INTERACTION DIAGRAM FOR USE CASE 1: LOGIN

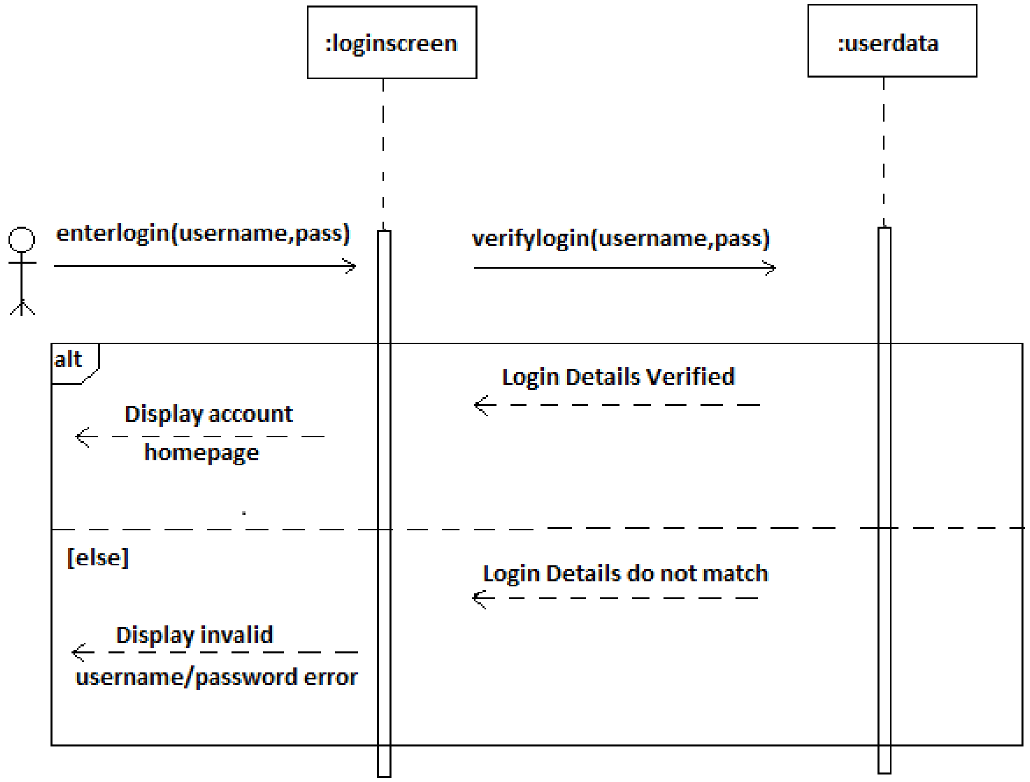

:loginscreen
:userdata
enterlogin(username,pass)
verifylogin(username,pass)
alt
Login Details Verified
Display account homepage
[else]
Login Details do not match
Display invalid username/password error

Fig 3.1 INTERACTION DIAGRAM FOR USE CASE 1: LOGIN

This interaction sequence diagram represents the **USE CASE 1: LOGIN**. The user initiates the login process by entering his username and password to the login screen. The login screen sends a ‘verifylogin’ query to the userdata table with username and password as input parameters. If the login details are found to be correct, a session is opened and the user is taken into his personalized account homepage. In case of a mismatch, ‘Invalid username/password’ error is displayed.

INTERACTION DIAGRAM FOR USE CASE 2: REGISTRATION

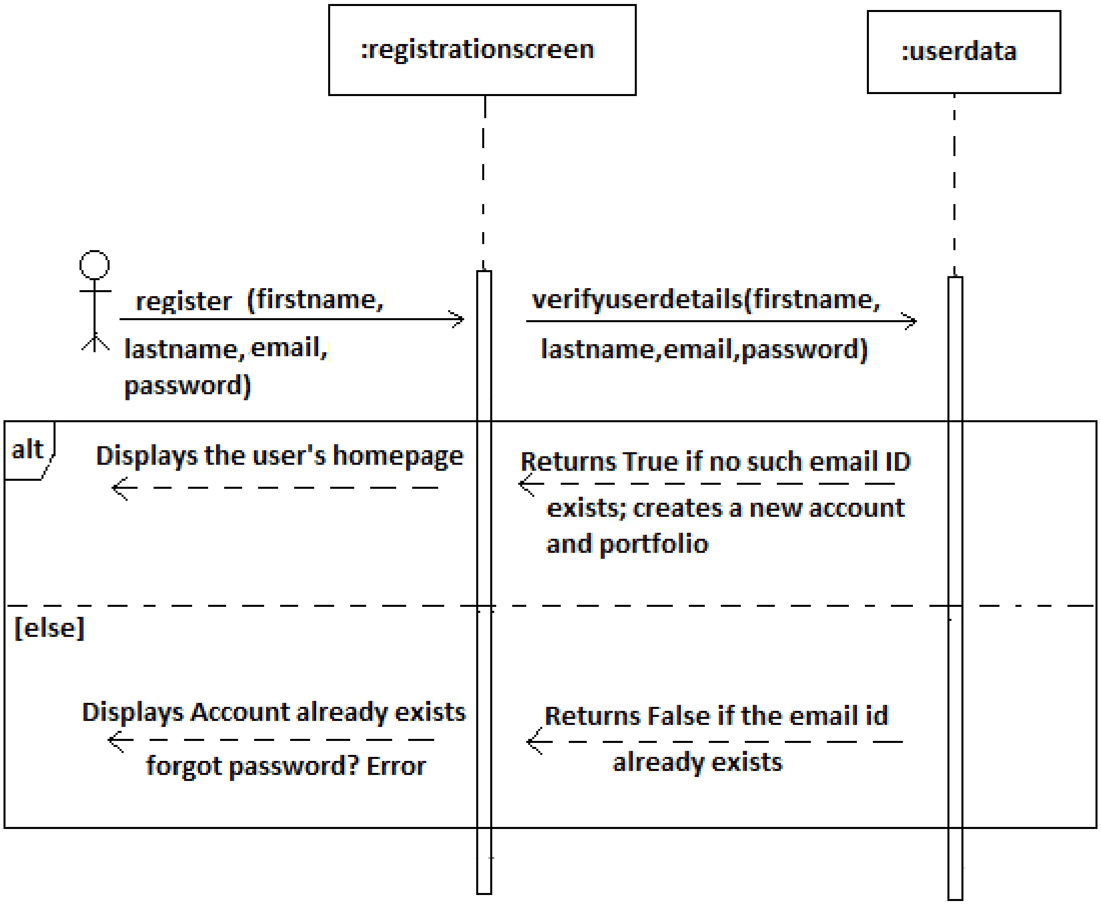

:registrationscreen
:userdata
register (firstname, lastname, email, password)
verifyuserdetails(firstname, lastname,email,password)
alt
Displays the user's homepage
Returns True if no such email ID exists; creates a new account and portfolio
[else]
Displays Account already exists forgot password? Error
Returns False if the email id already exists

Fig 3.2 INTERACTION DIAGRAM FOR USE CASE 2: REGISTRATION

This interaction sequence diagram represents the **USE CASE 2: REGISTRATION**. The User initiates the process by entering his firstname, lastname, email and password in the registration screen. The registration screen poses a ‘verifyuserdetails’ query to the userdata table to check if the user’s details already exist in the database. If no such email ID exists, the query returns true. A new account and a new portfolio are created for the user and he is directed to his personalized homepage. In case the email ID already exists in the database, the query returns false and the registration screen displays ‘Account already exists; Forgot Password?’ error.

INTERACTION DIAGRAM FOR USE CASE 3: VIEW ABOUT US

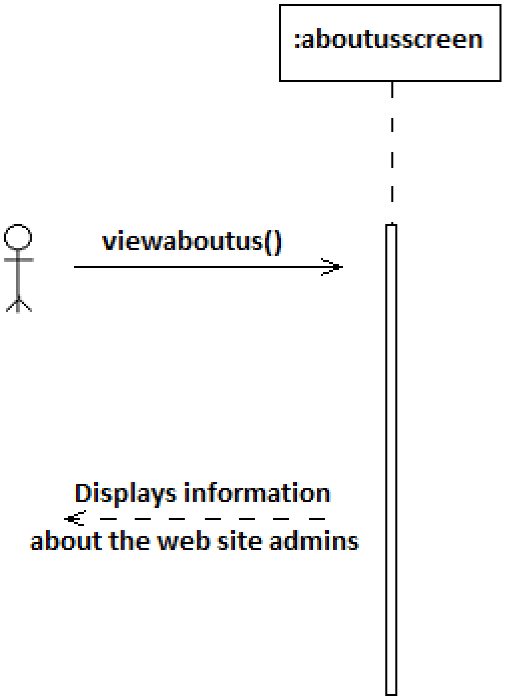


Fig 3.3 INTERACTION DIAGRAM FOR USE CASE 3: VIEW ABOUT US

This interaction sequence diagram represents the **USE CASE 3: VIEW ABOUT US**. The user initiates the entire process by using the ‘viewaboutus()’ function call. This prompts the about us screen to display information about the website and the website administrators.

INTERACTION DIAGRAM FOR USE CASE 4: VIEW ABOUT ME

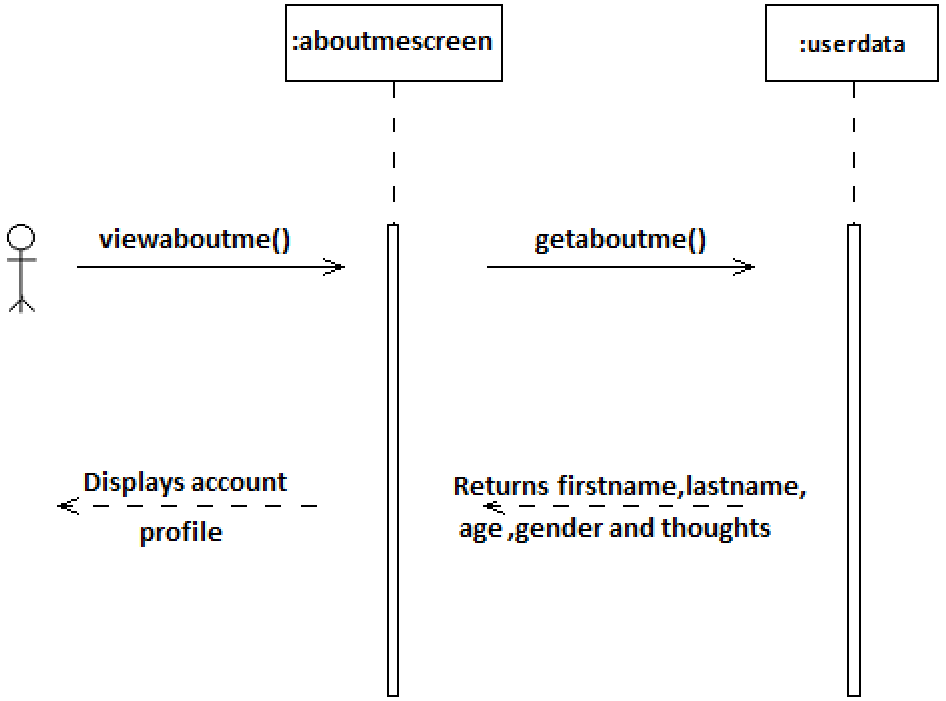


Fig 3.4 INTERACTION DIAGRAM FOR USE CASE 4: VIEW ABOUT ME

This interaction sequence diagram represents the **USE CASE 4: VIEW ABOUT ME**. When the user wants to check his profile information, he does it using the 'viewaboutme ()' function call to the about me screen. The page then communicates with the userdata table, gets the user information including his firstname, lastname, age, gender and the thoughts he put in his about me column and displays it for the user.

INTERACTION DIAGRAM FOR USE CASE 5: EDIT MY PROFILE

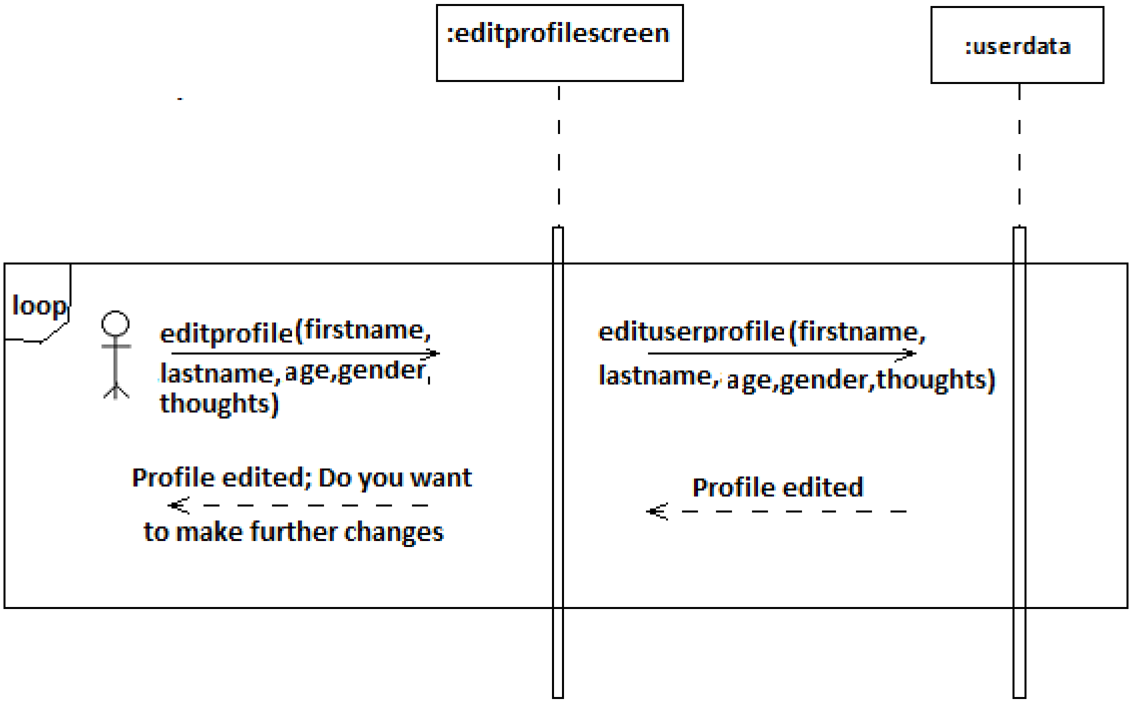

:editprofilescreen
:userdata
loop
editprofile(firstname, lastname, age,gender, thoughts)
edituserprofile (firstname, lastname, age,gender,thoughts)
Profile edited; Do you want to make further changes
Profile edited

Fig 3.5 INTERACTION DIAGRAM FOR USE CASE 5: EDIT MY PROFILE

This interaction sequence diagram represents the **USE CASE 5: EDIT MY PROFILE**. The user initiates this process using the ‘editprofile()’ function call. The columns he wants to edit will be the input parameters. The edit profile screen then instructs the userdata table to edit the user record according to the information provided by the user. Once the changes are made, the user is intimidated and he is allowed to make further changes. This process is a loop and continues until the user is satisfied with his profile.

INTERACTION DIAGRAM FOR USE CASE 6: VIEW RSS NEWSFEED

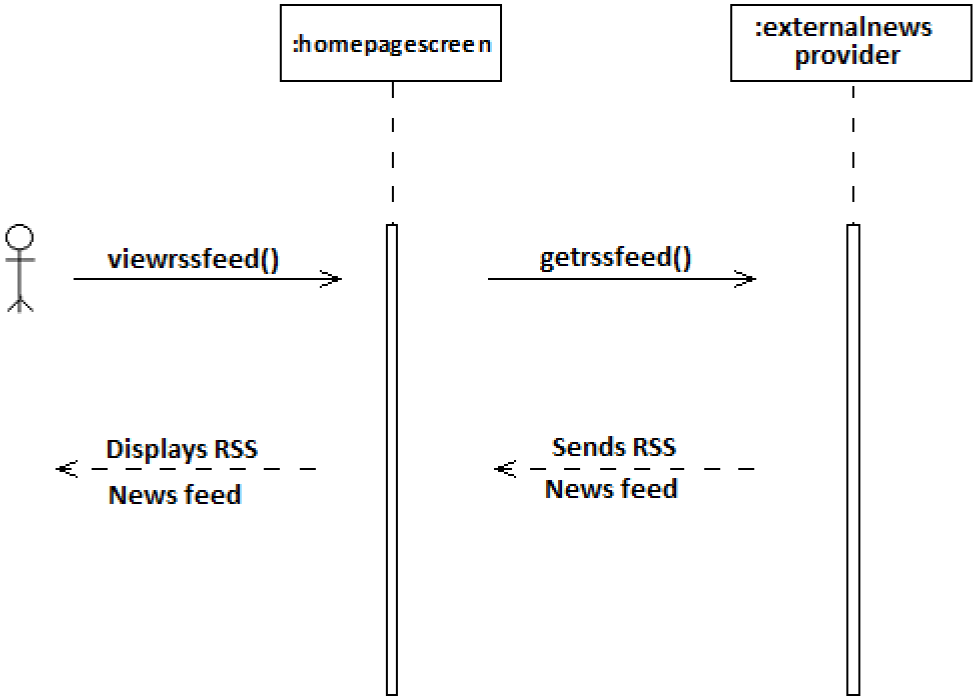


Fig 3.6 INTERACTION DIAGRAM FOR USE CASE 6: VIEW RSS NEWSFEED

This interaction sequence diagram represents the **USE CASE 6: VIEW RSS NEWSFEED**. When the user enters his homepage, the viewrssfeed() function prompts the home page screen to display the latest financial news for the user. The home page screen gets this information from an external news provider (in our case, it is CNBC news) and displays it for the user.

INTERACTION DIAGRAM FOR USE CASE 7: VIEW MY GAME PORTFOLIO

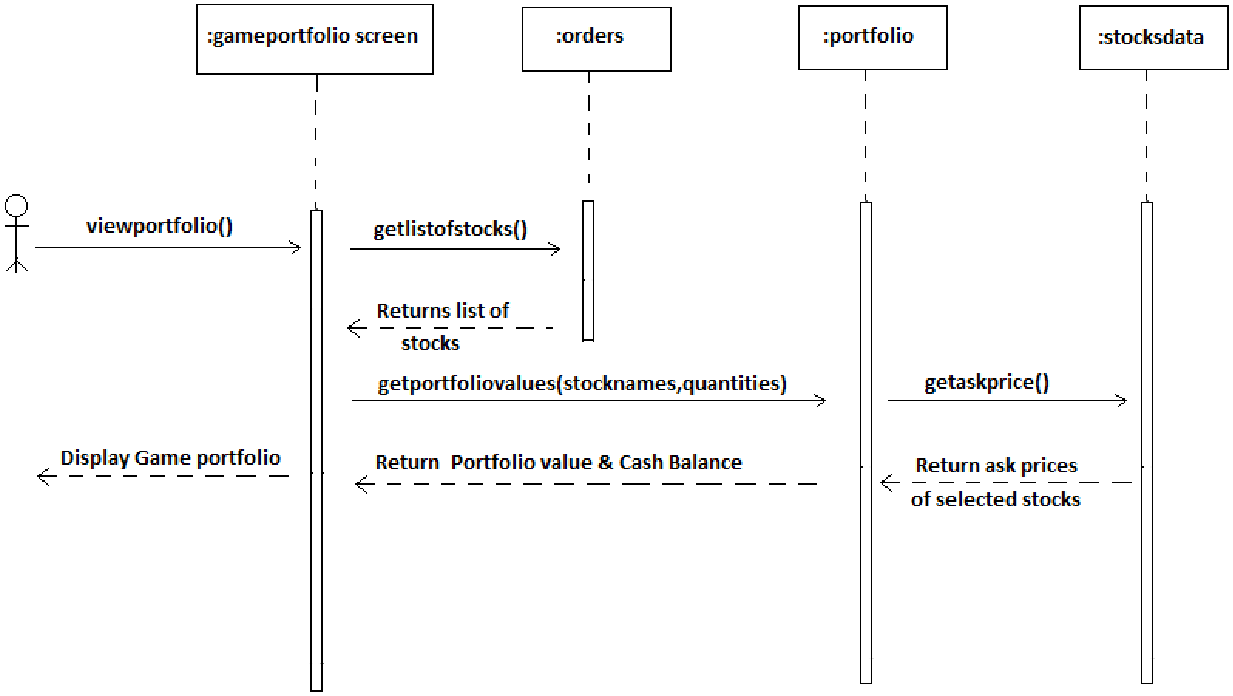


Fig 3.7 INTERACTION DIAGRAM FOR USE CASE 7: VIEW MY GAME PORTFOLIO

This interaction sequence diagram represents the **USE CASE 7: VIEW MY GAME PORTFOLIO**. The user initiates this process by sending the ‘viewportfolio’ function call to the game portfolio screen. The game portfolio screen then gets a list of all the stocks currently in the user’s portfolio from the orders table and then provides this information to the portfolio table. The portfolio table in turn gets the current ask prices for all these stocks and calculates the portfolio value. It then returns this portfolio value along with the user’s cash balance. These two values along with the list of current stocks in the portfolio are displayed for the user.

INTERACTION DIAGRAM FOR USE CASE 8: GO HOME

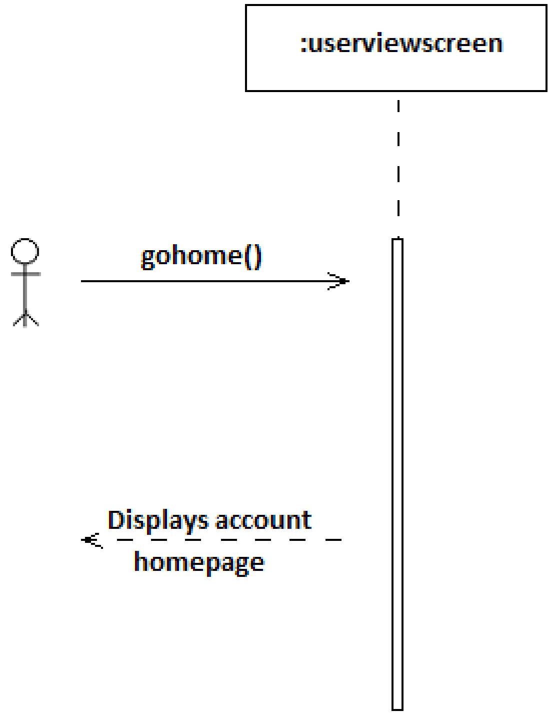


Fig 3.8 INTERACTION DIAGRAM FOR USE CASE 8: GO HOME

This interaction sequence diagram represents the **USE CASE 8: GO HOME**. Whenever the user wants to return to his homepage from a current page, he initiates the ‘gohome()’ function call. The user view screen, which is the current screen the user sees, responds to this function and directs him to his homepage.

INTERACTION DIAGRAM FOR USE CASE 9: MARKET ORDER - BUY

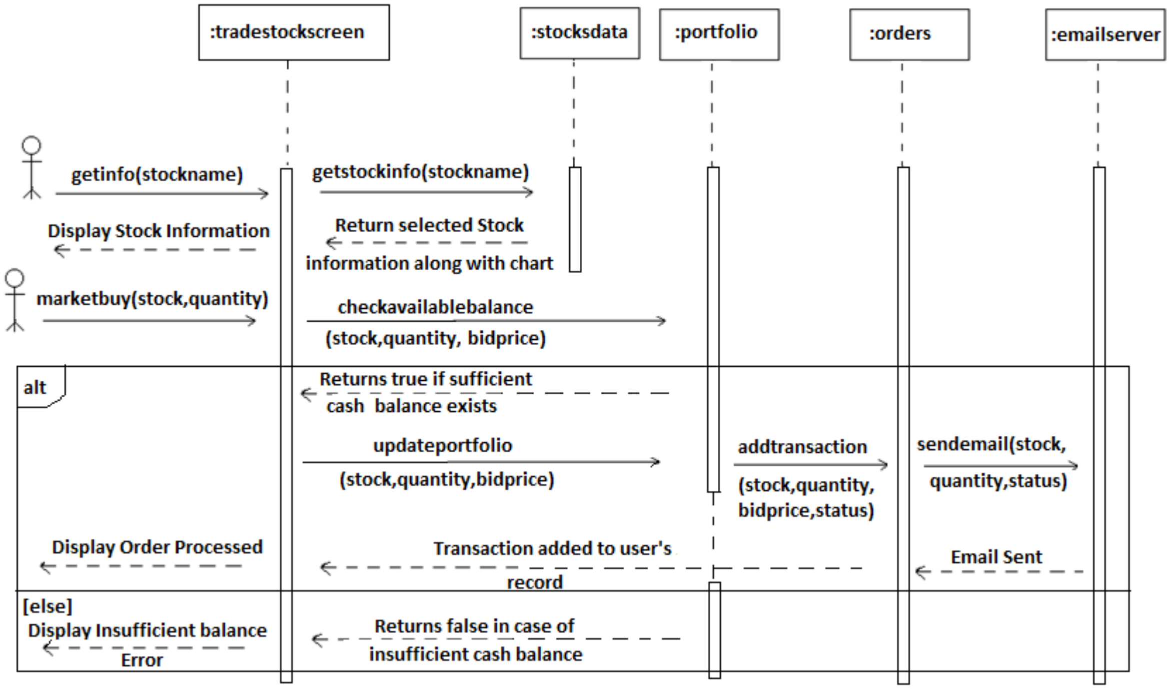

:tradestockscreen
:stocksdata
:portfolio
:orders
:emailserver
getinfo(stockname)
getstockinfo(stockname)
Display Stock Information
Return selected Stock information along with chart
marketbuy(stock,quantity)
checkavailablebalance (stock,quantity, bidprice)
alt
Returns true if sufficient cash balance exists
updateportfolio (stock,quantity,bidprice)
addtransaction (stock,quantity, bidprice,status)
sendemail(stock, quantity,status)
Display Order Processed
Transaction added to user's record
Email Sent
[else]
Display Insufficient balance Error
Returns false in case of insufficient cash balance

Fig 3.9 INTERACTION DIAGRAM FOR USE CASE 9: MARKET ORDER - BUY

This interaction sequence diagram represents the **USE CASE 9: MARKET ORDER - BUY**. When the user wants to buy a stock (market order), he initiates this process using the 'getinfo(stockname)' function call to the trade stocks screen where, 'stockname' is the stock he chooses to buy. The page then gets the current information about the stock like it's ask price, bid price etc. along with a five day chart from the stocks data table. The stocks data table fetches this information from Yahoo! Finance whenever a query is made.

The user can then decide on the quantity of the stocks based on the information he sees. 'marketbuy(stock,quantity)' then asks the portfolio table to check if the user has sufficient cash balance to make this transaction. If yes, the portfolio table is updated, the transaction is added to the orders table and an email notification is sent. If not, the user is informed about the insufficient cash balance scenario.

**We are currently fetching the NASDAQ stocks because Yahoo! Finance doesn't somehow give the ask and bid prices for NYSE stocks.*

**After 4pm, currently we are asking the users to make the trade next day rather than put it in pending, because the price may change when the market opens for the next day. We are also debating on giving the user a choice to keep his trade pending or to come back the next day for the trade.*

INTERACTION DIAGRAM FOR USE CASE 10: MARKET ORDER - SELL

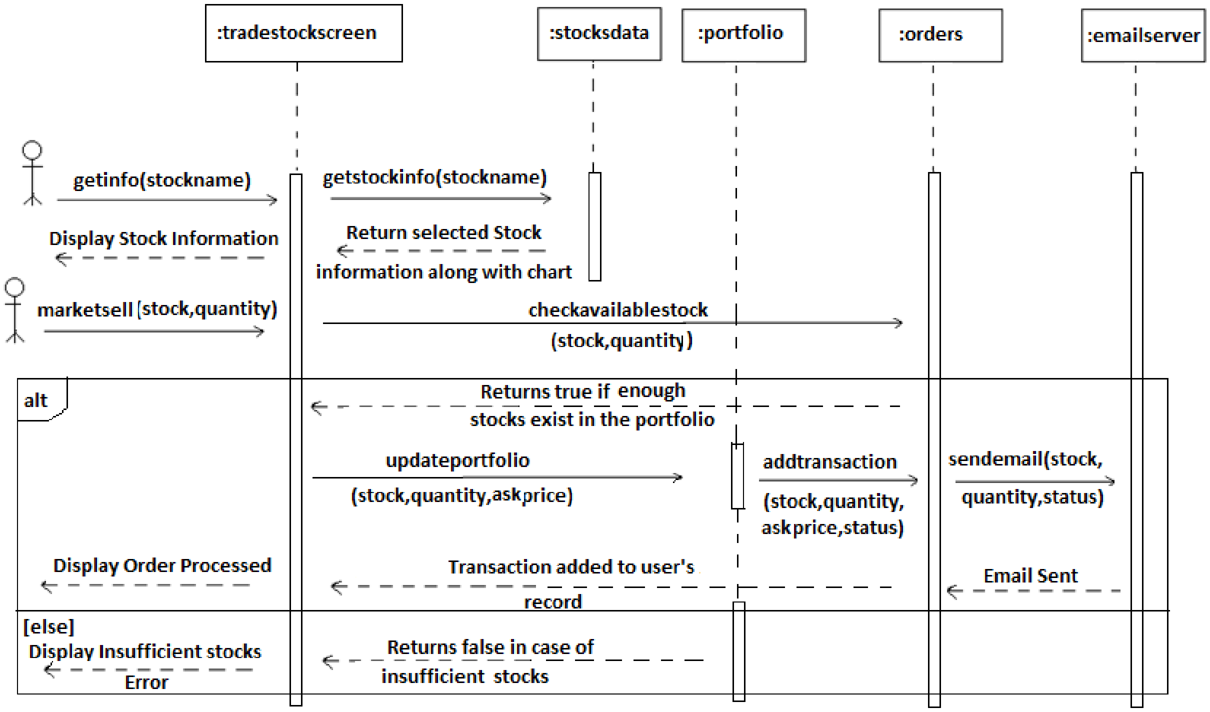

:tradestockscreen
:stocksdata
:portfolio
:orders
:emailserver
getinfo(stockname)
getstockinfo(stockname)
Display Stock Information
Return selected Stock information along with chart
marketsell [stock,quantity)
checkavailablestock (stock,quantity)
alt
Returns true if enough stocks exist in the portfolio
updateportfolio (stock,quantity,askprice)
addtransaction (stock,quantity, askprice,status)
sendemail(stock, quantity,status)
Display Order Processed
Transaction added to user's record
Email Sent
[else]
Display Insufficient stocks Error
Returns false in case of insufficient stocks

Fig 3.10 INTERACTION DIAGRAM FOR USE CASE 10: MARKET ORDER - SELL

This interaction sequence diagram represents the **USE CASE 10: MARKET ORDER - SELL**. When the user wants to sell a stock (market order), he initiates this process using the ‘getinfo(stockname)’ function call to the trade stocks screen where, ‘stockname’ is the stock he chooses to sell. The page then gets the current information about the stock like it’s ask price, bid price etc. along with a five day chart from the stocks data table. The stocks data table fetches this information from Yahoo! Finance whenever a query is made.

The user can then decide on the quantity of the stocks based on the information he sees. ‘marketsell(stock,quantity)’ then asks the orders table to check if the user has sufficient number of stocks to make this transaction. If yes, the portfolio table is updated, the transaction is added to the orders table and an email notification is sent. If not, the user is informed about the insufficient stocks scenario.

INTERACTION DIAGRAM FOR USE CASE 11: VIEW TRADING HISTORY

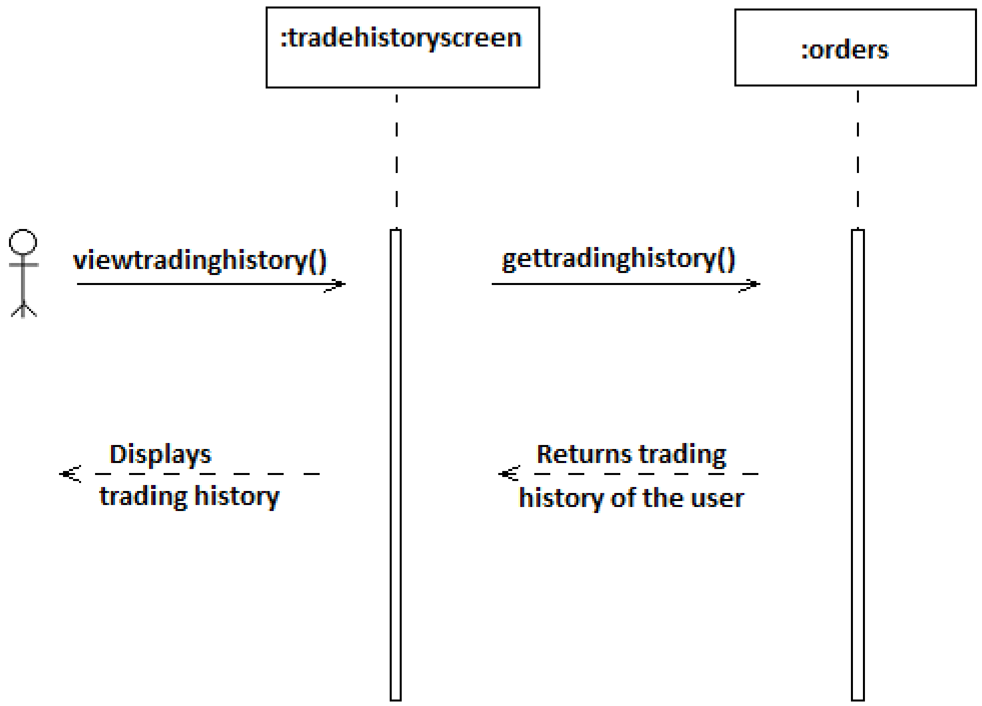


Fig 3.11 INTERACTION DIAGRAM FOR USE CASE 11: VIEW TRADING HISTORY

This interaction sequence diagram represents the **USE CASE 11: VIEW TRADING HISTORY.** When the user wants to see a history of his past transactions he can initiate this process using the 'viewtradinghistory()' function call to the trade history screen. The page then requests all the user history from the orders table and displays it to the user.

INTERACTION DIAGRAM FOR USE CASE 12: VIEW HELP DOCUMENTATION

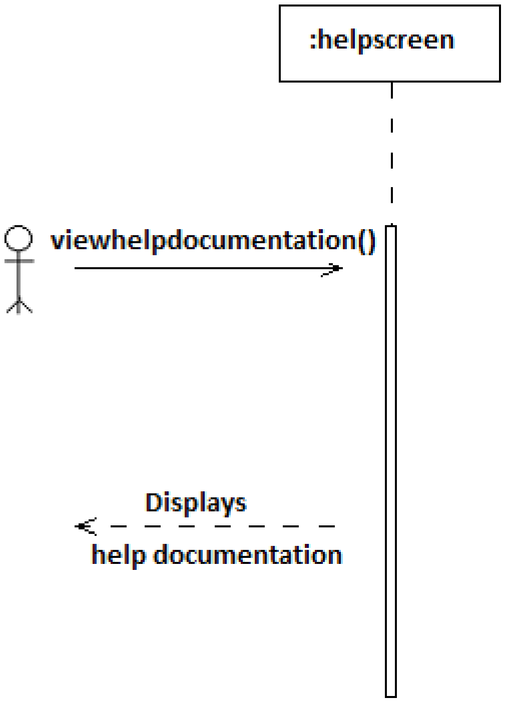


Fig 3.12 INTERACTION DIAGRAM FOR USE CASE 12: VIEW HELP DOCUMENTATION

This interaction sequence diagram represents the **USE CASE 12: VIEW HELP DOCUMENTATION**. The user can view the help documentation provided my making the 'viewhelpdocumentation()' function call to the help screen which then displays the required content.

INTERACTION DIAGRAM FOR USE CASE 13: VIEW TRADE DIARY

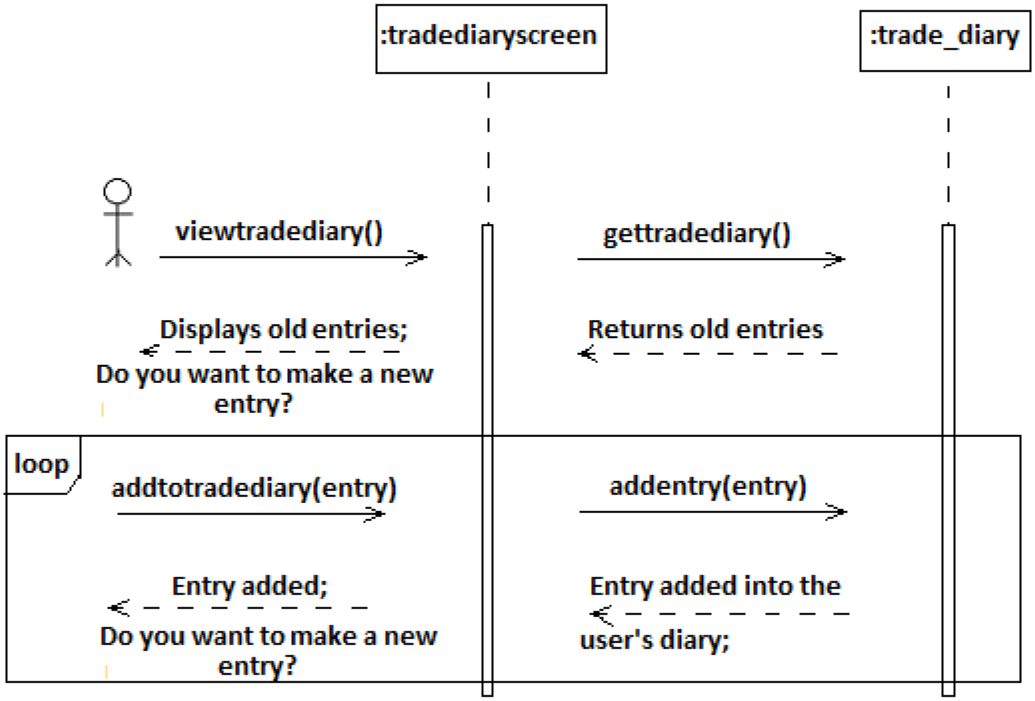


Fig 3.13 INTERACTION DIAGRAM FOR USE CASE 13: VIEW TRADE DIARY

This interaction sequence diagram represents the **USE CASE 13: VIEW TRADE DIARY**. The user initiates this process by making the ‘viewtradediary()’ function call to the trade diary screen. The trade diary page then requests the trade_diary table for the diary entries the user has previously made. It displays them and then asks the user if he wants to make a new entry. Once the user has made this new entry, it is sent to be stored in the trade_diary table as a new record and the user is asked if he wants to make another entry. This process continues in a loop as long as the user wants to make diary entries.

INTERACTION DIAGRAM FOR USE CASE 14: VIEW TOP FIVE GAINERS/LOSERS

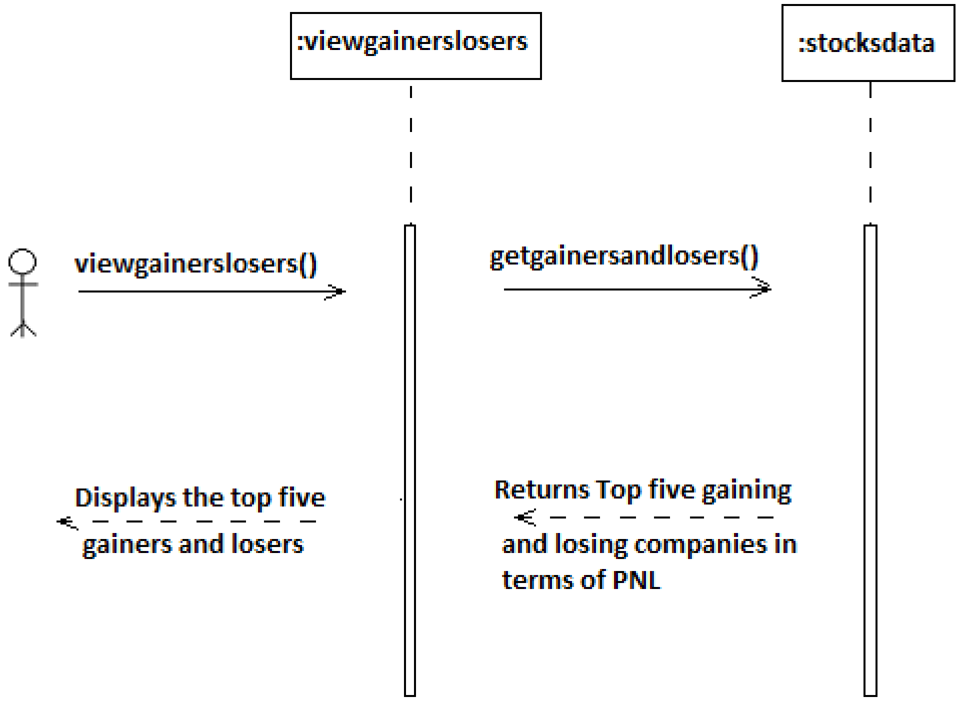

:viewgainerslosers
:stocksdata
viewgainerslosers()
getgainersandlosers()
Displays the top five gainers and losers
Returns Top five gaining and losing companies in terms of PNL

Fig 3.14 INTERACTION DIAGRAM FOR USE CASE 14: VIEW TOP FIVE GAINERS/LOSERS

This interaction sequence diagram represents the **USE CASE 14: VIEW TOP FIVE GAINERS/LOSERS**. The user can see a list of top five price-gaining and losing companies from our current database of 20 companies. For this, he needs to make the ‘viewgainerslosers()’ function call to the view gainers losers page. The page then sends a ‘getgainersandlosers()’ query to the stocksdata table. This function call returns the top five companies in the database that have the highest and lowest percentage change in the stock price. The percent change is calculated using the yesterday’s closing price and the price at which the latest trade has been made.

INTERACTION DIAGRAM FOR USE CASE 15: VIEW MOST ACTIVE COMPANIES

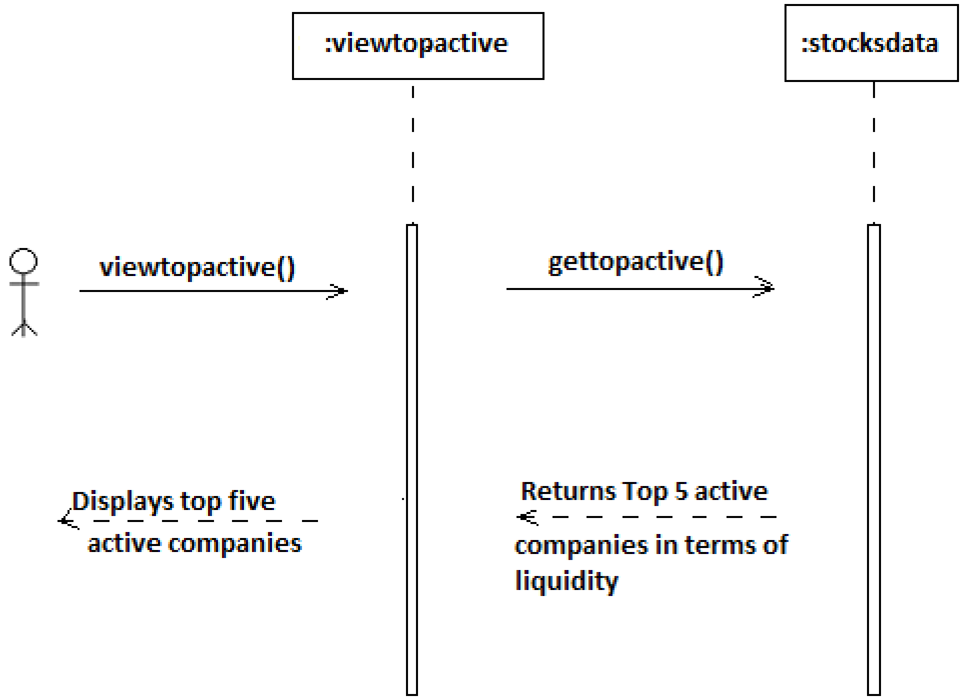

:viewtopactive
:stocksdata
viewtopactive()
gettopactive()
Displays top five active companies
Returns Top 5 active companies in terms of liquidity

Fig 3.15 INTERACTION DIAGRAM FOR USE CASE 15: VIEW MOST ACTIVE COMPANIES

This interaction sequence diagram represents the **USE CASE 15: VIEW MOST ACTIVE COMPANIES**. The user can see a list of top five active companies from our current database of 20 companies. For this, he needs to make the ‘viewtopactive()’ function call to the view top active companies page. The page then sends a ‘gettopactive’ query to the stocksdata table. This function call returns the top five companies in the database that have the highest liquidity (volume of stocks) which are then displayed.

INTERACTION DIAGRAM FOR USE CASE 16: DELETE ACCOUNT

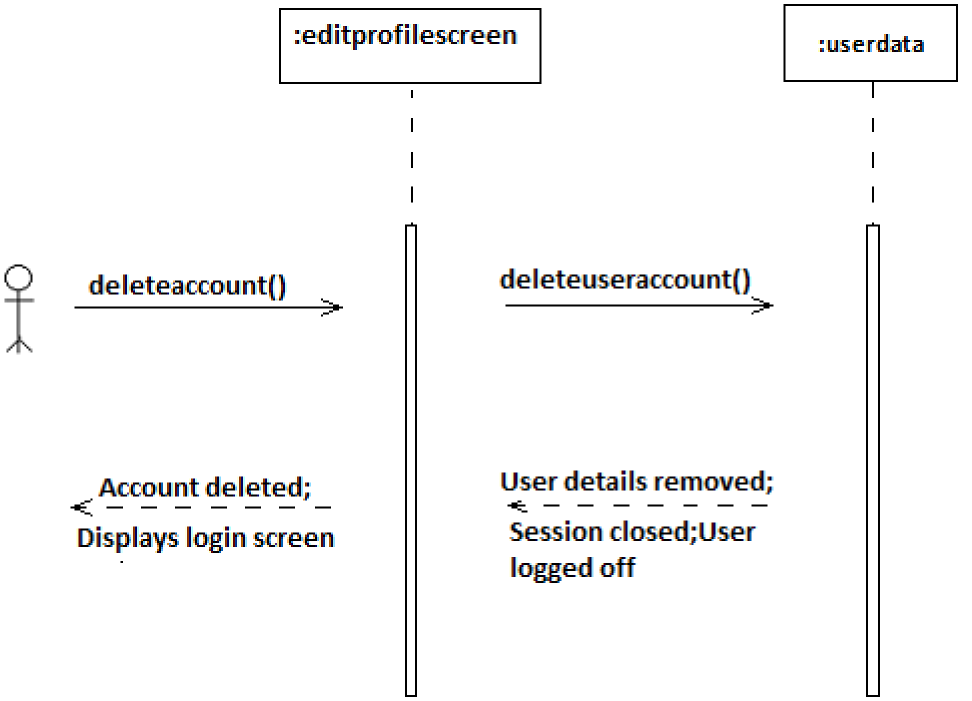


Fig 3.16 INTERACTION DIAGRAM FOR USE CASE 16: DELETE ACCOUNT

This interaction sequence diagram represents the **USE CASE 16: DELETE ACCOUNT**. The user initiates this process using the ‘deleteaccount()’ function call to the edit profile page. The page on receiving this call instructs the user data table to remove the user’s record from the table, close the current session and log him off. The user is then directed to the login screen.

INTERACTION DIAGRAM FOR USE CASE 17: LOGOUT

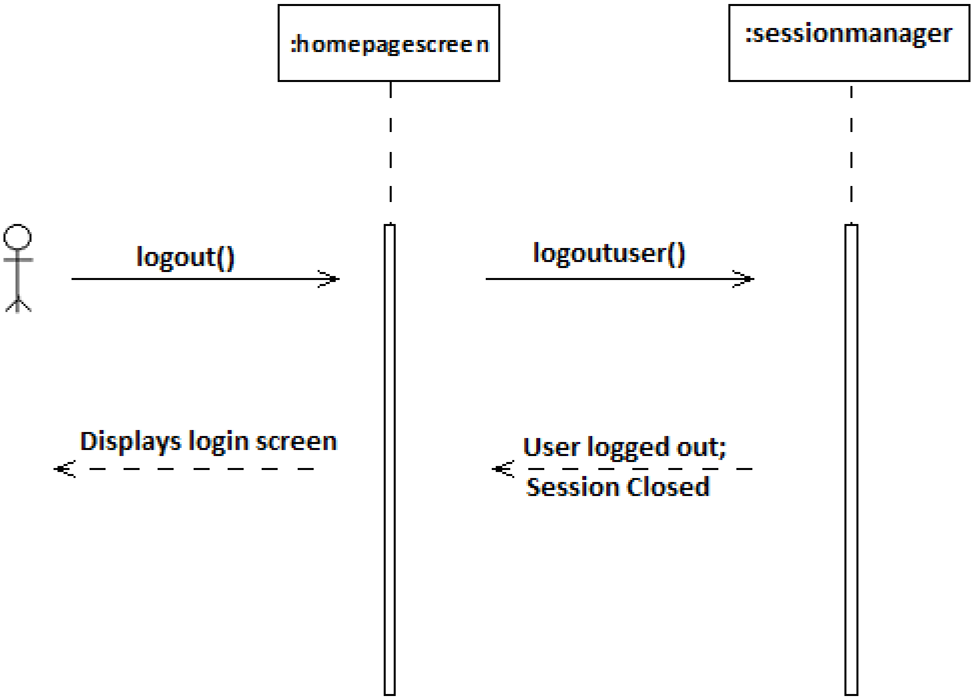


Fig 3.17 INTERACTION DIAGRAM FOR USE CASE 17: LOGOUT

This interaction sequence diagram represents the **USE CASE 17: LOGOUT**. When the user wants to log out, he goes to his homepage and makes a call to the ‘logout()’ function. The home page then instructs the session manager to close the session and log the user out. The user is then directed to the login screen.

4. CLASS DIAGRAM AND INTERFACE SPECIFICATION:

a) Class Diagram:

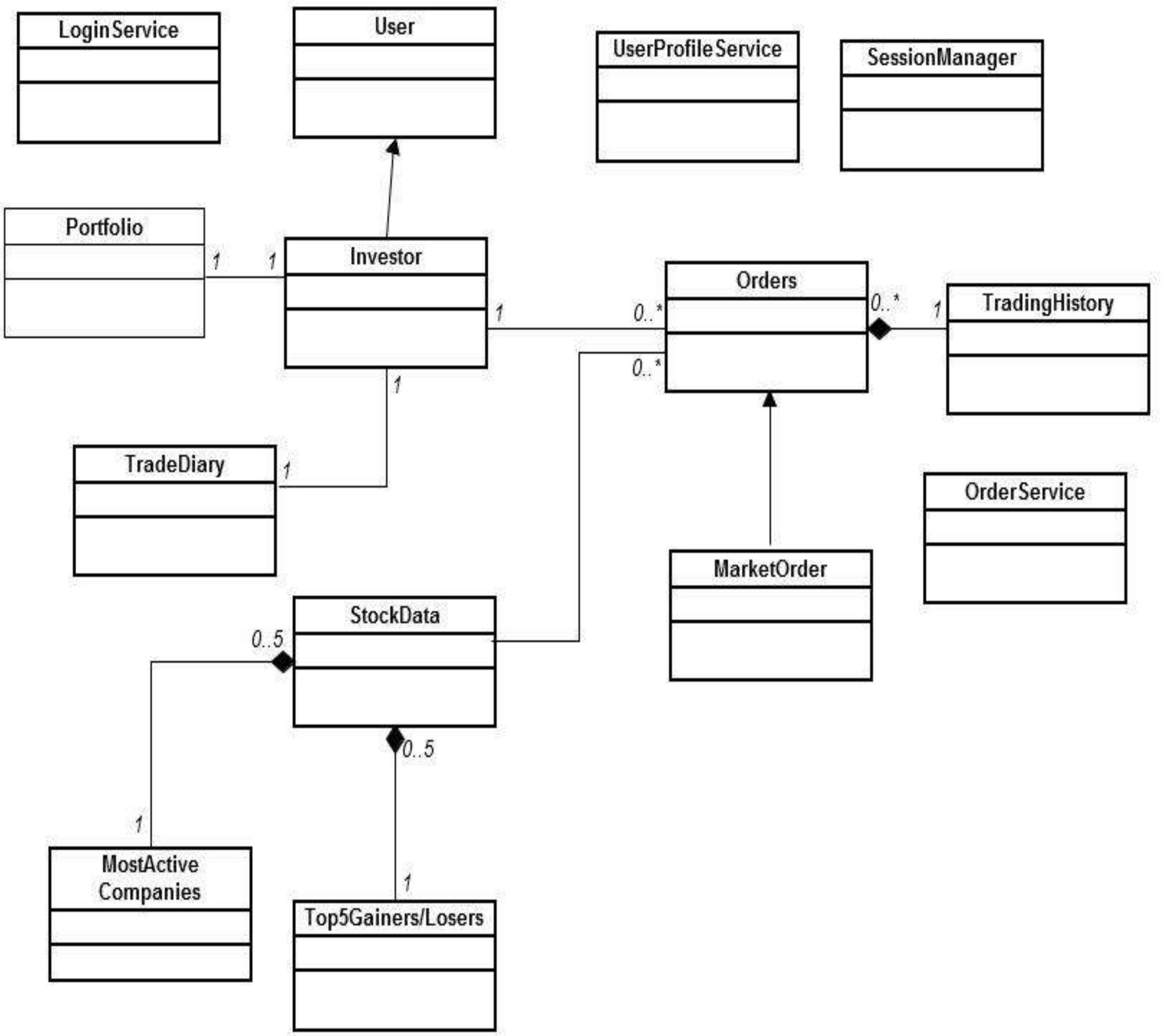


Fig 4.1 Class Diagram

b) Data Types and Operation Signatures:

1. User (Investor):
-userid: int
-firstname: String
-lastname: String
-aboutme: String
-email: String
-password: String

2. LoginService:
-email: String
-password: String
+verify_login (in email: String, in password: String): bool{Post-condition: user is authenticated and homepage is displayed. Alternate-scenario: user enters wrong username and password and is requested to enter them again. }

3. SessionManager:
-userid: int
-email: String
+is_valid_session(): bool
+getuserid (in email: String): int

4. UserProfileService:
+register (in user:User): void
+display_userdata(in userid: int):void{Post-condition: First name, Last name, Age, About-me is extracted from the database and displayed.}
+update_Profile(in user:User):void {Post-condition: Changed fields are updated in the userdata table in the database }
+delete_account(in userid: int):void {Post-condition: All information about the user is deleted in the database}

5. TradeDiary:
-Date: String
-Symbol: String
-Comment: String
+insert_comment(in Date: Date, in Symbol: String, in comment: String): void{ Post-condition: comment is inserted into the trade_diary table in the database}
+display_comment():void{Post-condition: all the comments of the user are extracted from trade_diary table in the database and displayed.}

6. Order: (Market Order)
-orderid: int
-company: String
-tickersymbol: String
-action: String
-Order_type: String
-quantity: int
-transaction_date: datetime
-Price: double
-Commission: double
-userid: int
-status: String

7. OrderService:
+insert_order(in order: Order):void
+get_orders(in userid: int): Array{Order}
+get_stockholdings(in userid:int): Array{Company, quantity}
+send_email(in email: String): void
+update_cashbalance (in quantity: int, in Price: double, in commission: double): void{Post-condition: cashbalance is increased or decreased depending on whether shares are bought or sold and is updated in the portfolio table in the database}

8. TradingHistory:
+extract_history(in userid: int): Order

9. StockData:
-symbol: String
-company: String
-ask: double
-bid: double
-stockchange: double
-stockvolume: double
+update_stockdata(): void{Post-condition: stockdata is extracted from yahoo finance and updated in the stockdata table in the database.}

10. Top5Gainers/Losers:
+extract_gainers(in stockdata: StockData): Array{company, stockchange}{Post-condition: Top 5 companies based on the change% are extracted from stockdata table and displayed}
+extract_losers(in stockdata: StockData): Array{company, stockchange}{Post-condition: Last 5 companies based on the change% are extracted from stockdata table and displayed}

11. MostActiveCompanies:

+extract_active(in stockdata: StockData): Array{company, stockvolume}{Post-condition: Top 5 companies based on the volume are extracted from stockdata table and displayed}

12. Portfolio:

-portfolioid: int
-cashbalance: double
-portfoliovalue: double
-userid: int
+update_portfoliovalue(in userid: int) }{Post-condition: Depending on the stock holdings of the user the portfolio value is updated in the portfolio table.}
+display_portfolio(in userid: int) }{Post-condition: Displays the portfoliovalue, cashbalance and stockholdings of the user.}

5. SYSTEM ARCHITECTURE AND SYSTEM DESIGN

a) Architectural Styles

Architectural style corresponds to the composition of the system along with communications involved in it. The architecture of our software system is not limited to a single architectural style, but is a combination of architectural styles that make up the complete system. This is because the system under consideration (TradeFun!) is an interactive and dynamic website. A few architectural styles that relate close to our system are explained in the following table

Architectural Style	Description(MSDN & Wikipedia)	Role in Trade Fun! website
Client/Server	Segregates the system into two applications, where the client makes requests to the server. In many cases, the server is a database with application logic represented as stored procedures. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.	Client/Server architectural style can be found in working of various applications of our website like during Login, Trading, Trading Diary, Portfolio display, Delete Account. This style acts as the back bone of our website as this a database driven website.
Component based Architectural style	Component-based architecture describes a software engineering approach to system design and development. It focuses on the decomposition of the design into individual functional or logical components that expose well-defined communication interfaces containing methods, events, and properties.	Trade Fun! Website contains different modules which are interconnected and interact with each other. For example, Trading diary and Portfolio are two individual applications which interact with each other for data. This makes our website to have Component based Architectural style
Event driven Architectural style	Event-driven architecture (EDA) is a software architecture pattern promoting the production, detection, consumption of, and reaction to events. An *event* can be defined as "a significant change in state"	A typical Web application design like Trade Fun! works with the user logging in(an event), asking for information(event), executing the trade(event) and logging out. Thus Trade Fun! has an event driven architectural style

b) Identifying Subsystems

The following package diagram depicts our subsystems. It shows the domains <<boundary>>, <<control>>, <<entity>> that were discussed in the domain analysis and the classes that were discussed in the class diagram.

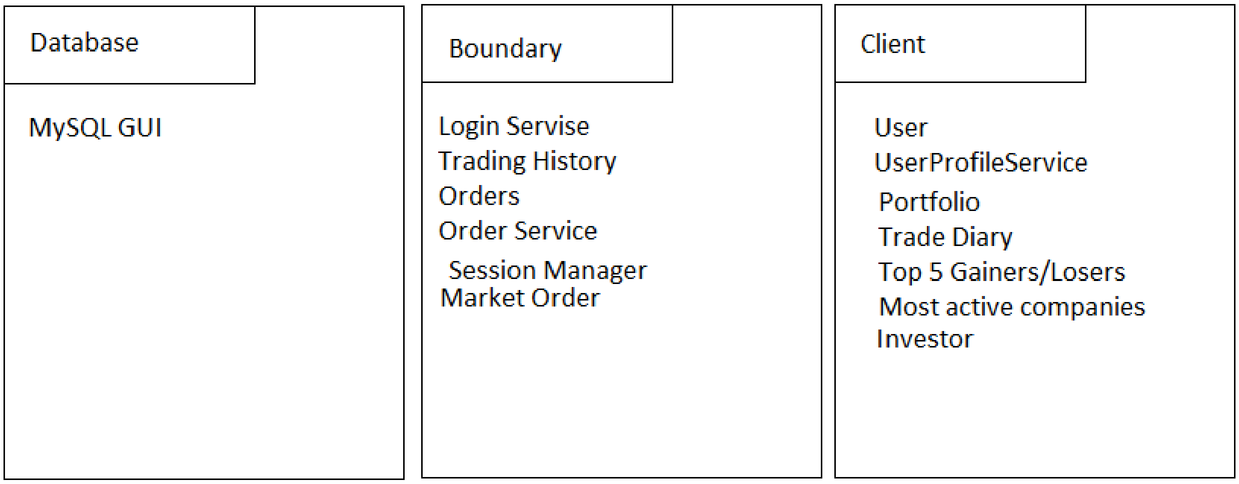


Fig 5.1 Package diagram of subsystems

c) Mapping Subsystems to Hardware

At the time of writing this report, our system runs on a single machine. Both the server and client reside on the same machine, which means, to play the game user should have database and web server setup on his/her machine. By the time of final release, we want to make sure that this is not required.

d) Persistent Data Storage

As our system is a register and play model, we are required to store the personal information of our registered users along with their individual portfolios and trading diaries. This was achieved by setting up a relation database called MySQL. In MySQL, at present we have a database called “test”. This has all the tables that are required for the smooth flow of the game. All the personal information of the users is stored in a table called “userdata” in which email will be a unique column. This was implemented so that no two users can have the same email (just like a username). “yahoo data table” is the table which stores the stock information and gets updated frequently by running a query.

These tables are interconnected either through the primary key (userid) or through email. Email was used to connect the tables because it is easy to store and update tables with this as email will be the session variable and all the data pertaining to that session can be sent to database tagged with the email.

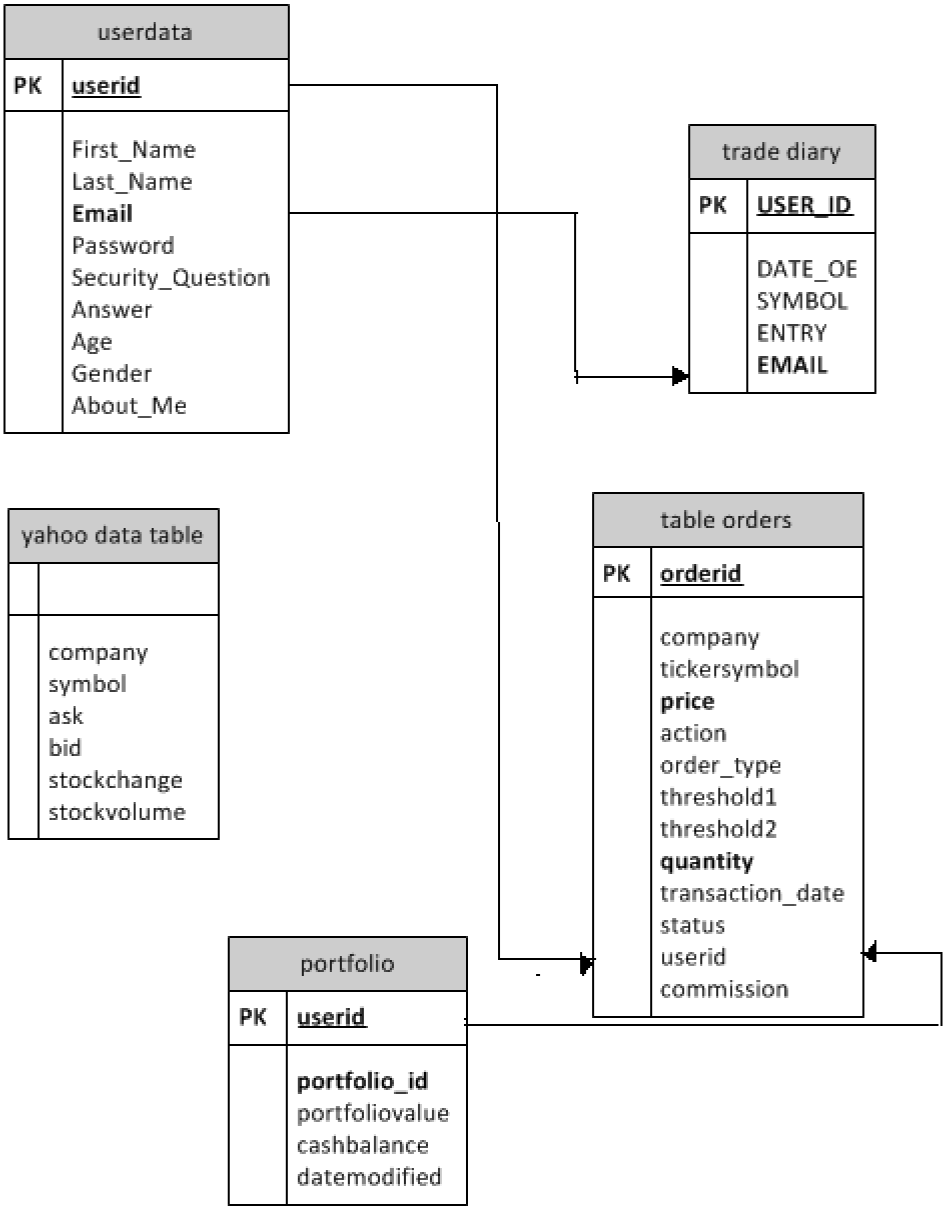


Fig 5.2 Database Schema

e) Network Protocol

Since our system is a web based application using PHP and MySQL, we will be running on an Apache server. PHP has built in a library for communicating with MySQL databases. It is much more efficient and faster than opening an ODBC connection. We will use this library for communications between the application and its backend MySQL database. The PHP-MySQL extension makes full use of the MySQL Client/Server Protocol. This is a powerful protocol and will facilitate all the server-to-database server connections. We choose MySQL not only because it is highly robust, but also it is completely free and enables us to invest resources in other facets of the system.

Also, our system connects to Yahoo! Finance using HTTP. We are using an HTTP protocol because it is the most commonly used protocol on the internet and it is a standard protocol in any server as well as browser. This option was especially attractive as it allows for great flexibility in design and high reliability. Any user who can run a web browser will be compatible with our system. By the way, HTTP is driven simply by the fact that HTTP URLs for stock quotes and charts on Yahoo! Finance are readily available.

f) Global Control Flow:

- Execution Orderness:

Most of our system is event-driven. Which means actions can be generated by any user at any point in time. Once an action is generated the system will respond accordingly, managed by the control structure. When no actions are being taken, the system will remain idle until user-interaction occurs. The advantage of event-driven is that it provides a simpler structure and waits in a loop for events, and every user can generate the actions in a different order. The difficulties arise when a sequence requires multiple steps to complete.

Some of screens in our system have two threads that run concurrently. For example, my portfolio and my trading history are both have two threads. One is event-driven, which we have mentioned above, the system responds when the user clicks a button or when an event occurs. The other is procedure-driven which periodically updates the display of the portfolio contents and transaction history with taking no inputs from the users.

- Time Dependency:

Until now we have not used any timers in our system. That is to say, when a user clicks trade stocks, my portfolio, top five gainers/losers and most active companies, the stock data table in the database is automatically updated with the data extracted from Yahoo! Finance and accordingly the actions are carried out with the updated data. Maybe we will add timers in the future demo.

- Concurrency:

Since our system is implemented via a server-side scripting language and we use Apache as our web server, it is, by nature, multithreaded. This is completely seamless to us and saves a lot of development time while making use of proven technologies. Apache, PHP, MySQL are all proven to be solid, enterprise caliber software. These are multithreaded and allow for many concurrent users and concurrent database queries.

g) Hardware Requirements

- Disk Storage**:** 100MB Available Hard Drive Space would satisfy our needs to hold user data in database.
- Operating System: the user can access to our website using Windows XP/Vista/7.
- Internet-LAN Connection with minimum bandwidth of 56Kbps are needed to connect to Yahoo! Finance and CNBC so that users can get the latest information.
- Screen Display-our website will be well presented if users use a computer with a display resolution of 1024x768 or greater.
- The system should be able to run PHP, MYSQL, java script and Apache HTTP server smoothly.

These requirements are not minimal, but are recommended for the best server experience.

6. ALGORITHMS AND DATA STRUCTURES:

a) Algorithms:

Top 5 Gainers/Losers:

The top 5 gainers/losers list gives a list of top 5 companies based on change (%) whose share values have increased and decreased.

Change in %:

Change is the difference between the last closing price and the current price (Last trade).Change percent signifies the profit per share made by the company.

Previous day’s closing price=p1

Last trade (current price) = p2

Change (%) = ((p1-p2)/p1)*100

(b) Data Structures:

Our system uses two types of data structures. Arrays are used in many parts of our system. The other data structures used are data tables loaded in the MYSQL database.

7. USER INTERFACE DESIGN

One of the main goals of Trade Fun! Website is to attract students and traders with little or no technical knowledge. In order to achieve this goal, from the beginning we made sure to keep the User Interface as simple and as easy to use as possible. The initial design for a general page that we planned was as follows:

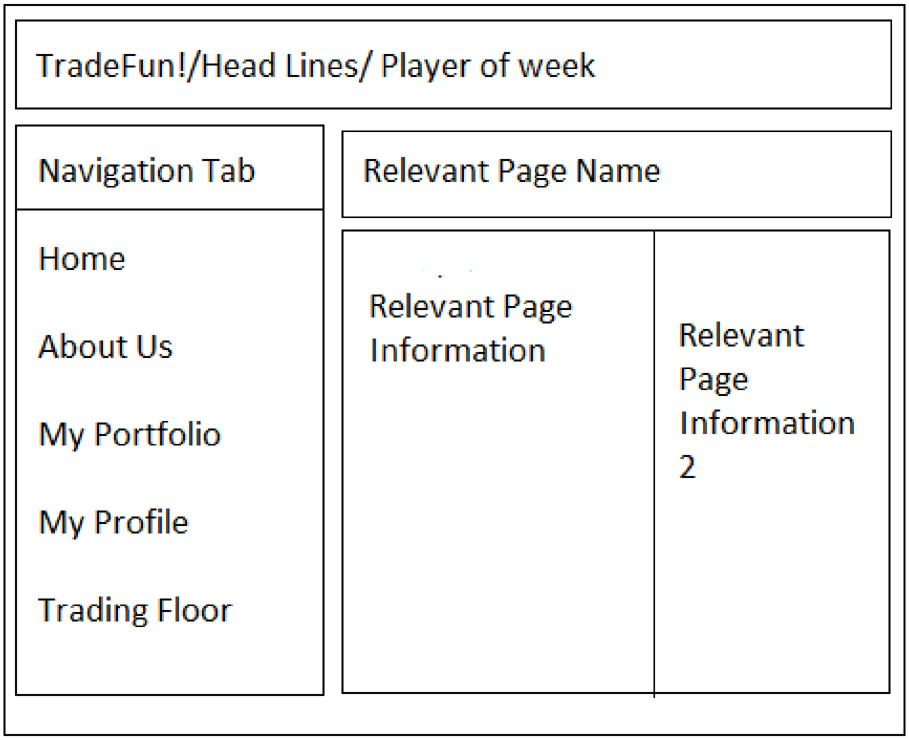
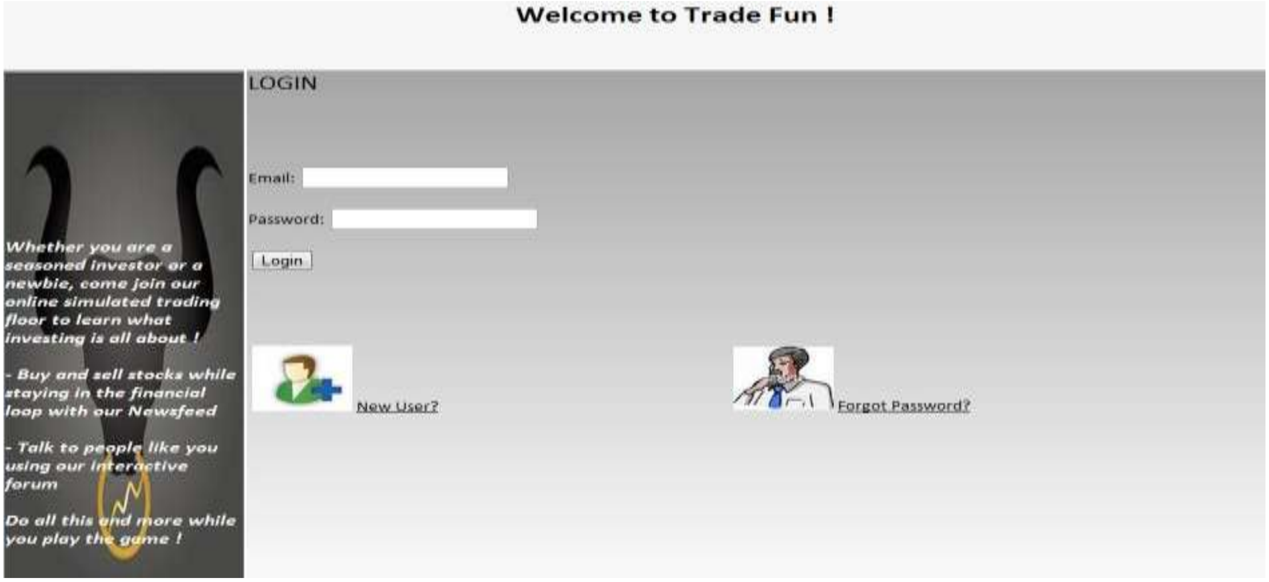

TradeFun!/Head Lines/ Player of week
Navigation Tab
Home
About Us
My Portfolio
My Profile
Trading Floor
Relevant Page Name
Relevant Page Information
Relevant Page Information 2

Fig 7.1 General page format

Keeping our initial promise, we decided to use the above design for all our pages. This decision was made after we found that the users feel comfortable if all the pages are of the same design/ pattern (as opposed to boredom attached to it). In order to achieve this goal, we are using iframes where all the new pages open in the frame at the center of the page as opposed to an entirely new page. This has two distinct advantages: it is easy to navigate through the pages- using the navigation bar on the left side of the page and it is easy to keep the session alive through all the pages (in addition to keeping the number of mouse clicks at minimum).

Our User Interface promises least number of mouse clicks to get the work done. The front page of Trade Fun! Has login along with links that redirects the user to “new user registration” and “forgot password”. There will be running scroll on the left side of the page with latest updates/news from Trade Fun! team.



Whether you are a seasoned investor or a newbie, come join our online simulated trading floor to learn what investing is all about !

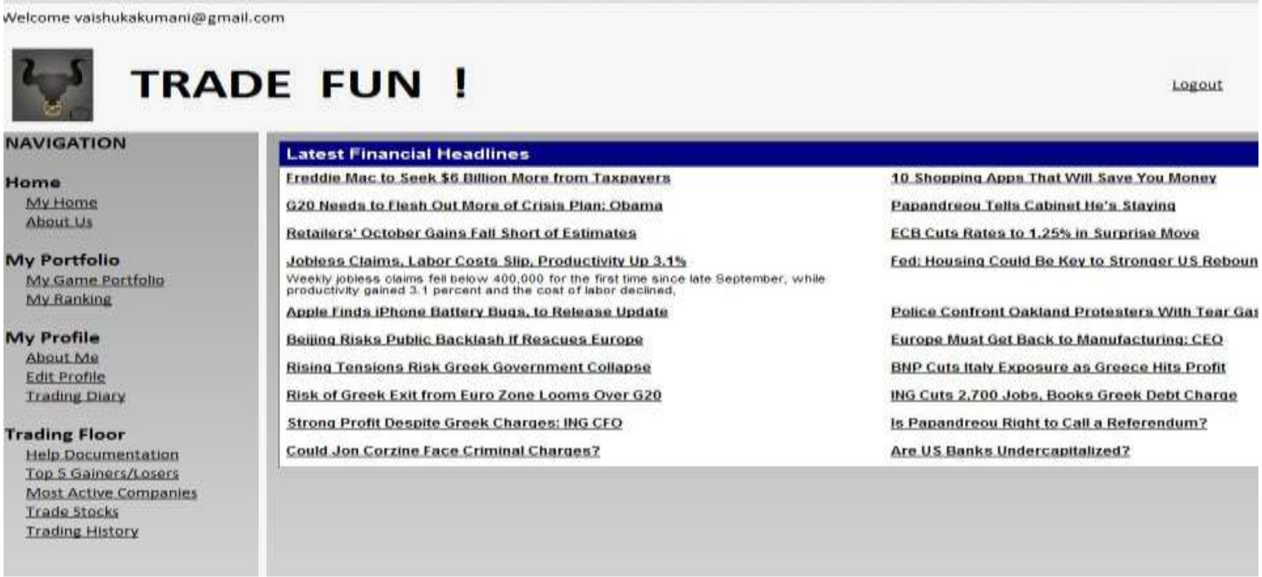
- Buy and sell stocks while staying in the financial loop with our Newsfeed

- Talk to people like you using our interactive forum

Do all this and more while you play the game !

Fig 7.2 Login Page

Once the User enters his/her credentials and system authenticates it, user will be redirected to the homepage which is shown below. Going along with the lines of general page description, home page has navigation on the left side of the page and an iframe at the center of the screen with latest financial/ world headlines from CNBC. Care has been taken not to include the entire CNBC website into our homepage as this may result in giving up the valuable space.



Freddie Mac to Seek $6 Billion More from Taxpayers

G20 Needs to Flesh Out More of Crisis Plan: Obama

Retailers' October Gains Fall Short of Estimates

Jobless Claims, Labor Costs Slip, Productivity Up 3.1%

Weekly jobless claims fell below 400,000 for the first time since late September, while productivity gained 3.1 percent and the cost of labor declined.

Apple Finds iPhone Battery Bugs, to Release Update

Beijing Risks Public Backlash If Rescues Europe

Rising Tensions Risk Greek Government Collapse

Risk of Greek Exit from Euro Zone Looms Over G20

Strong Profit Despite Greek Charges: ING CFO

Could Jon Corzine Face Criminal Charges?

10 Shopping Apps That Will Save You Money

Papandreou Tells Cabinet He's Staying

ECB Cuts Rates to 1.25% in Surprise Move

Fed: Housing Could Be Key to Stronger US Reboun

Police Confront Oakland Protesters With Tear Ga

Europe Must Get Back to Manufacturing: CEO

BNP Cuts Italy Exposure as Greece Hits Profit

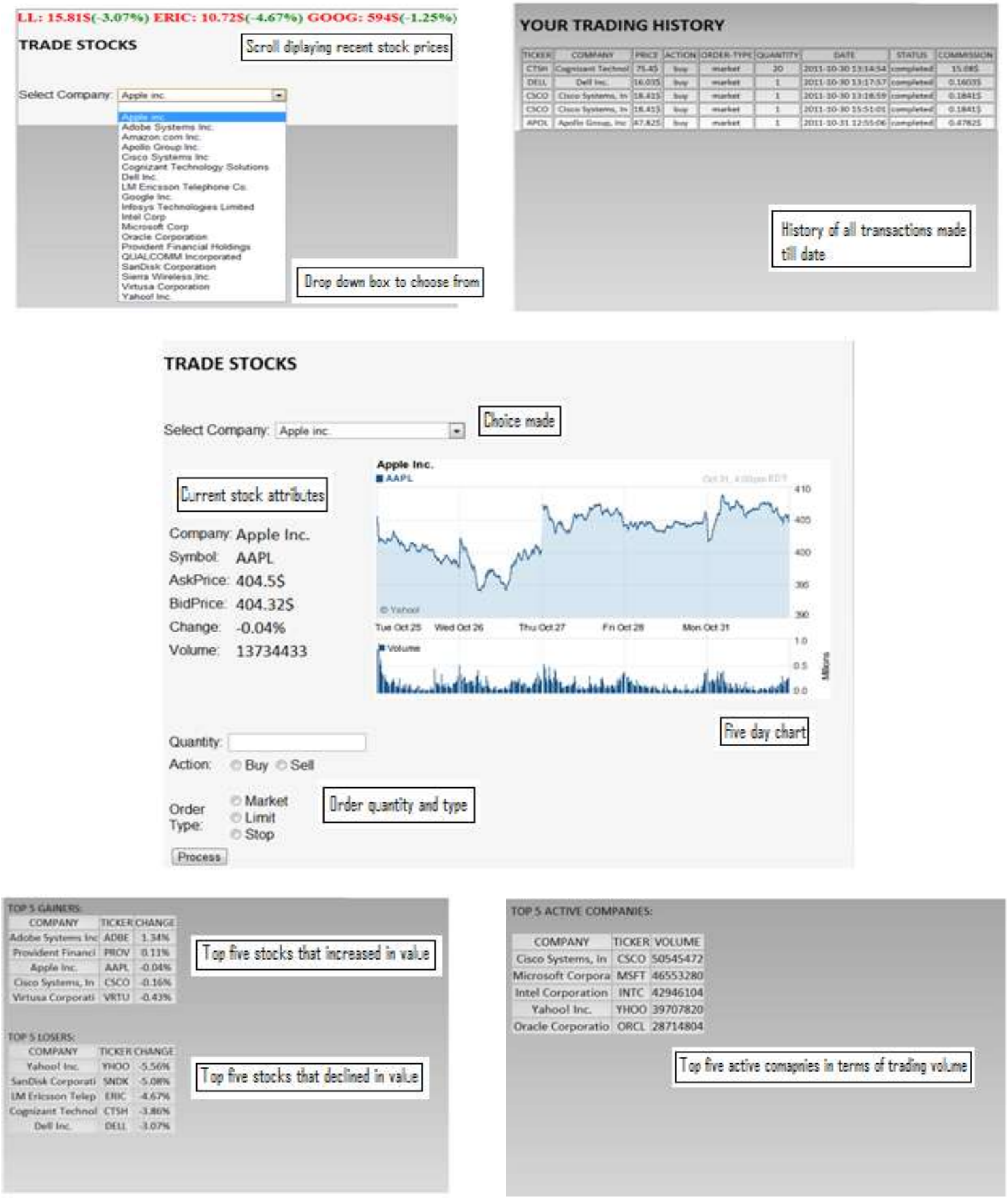
ING Cuts 2,700 Jobs, Books Greek Debt Charge

Is Papandreou Right to Call a Referendum?

Are US Banks Undercapitalized?

Fig 7.3 Homepage

From the homepage, user can navigate to the page of his/her choice using the navigation bar. For reference, trading page and portfolio are shown below. Again, please keep in mind that our main goal is to deliver an enterprise level trading league and hence we are concentrating more on the code and less preference is being given to flashy web designs.


Scroll diplaying recent stock prices
Drop down box to choose from
History of all transactions made till date
Choice made
Current stock attributes
Apple Inc.
AAPL
Oct 31, 4:00pm EDT
410
405
400
395
390
© Yahoo!
Tue Oct 25
Wed Oct 26
Thu Oct 27
Fri Oct 28
Mon Oct 31
Volume
1.0
0.5
0.0
Millions
Five day chart
Order quantity and type
Top five stocks that increased in value
Top five stocks that declined in value
Top five active comapnies in terms of trading volume
Process

LL: 15.81$(-3.07%) ERIC: 10.72$(-4.67%) GOOG: 594$(-1.25%)

TRADE STOCKS

Select Company: Apple inc.

- Apple inc.
- Adobe Systems Inc.
- Amazon.com Inc.
- Apollo Group Inc.
- Cisco Systems Inc.
- Cognizant Technology Solutions
- Dell Inc.
- LM Ericsson Telephone Co.
- Google Inc.
- Infosys Technologies Limited
- Intel Corp
- Microsoft Corp
- Oracle Corporation
- Provident Financial Holdings
- QUALCOMM Incorporated
- SanDisk Corporation
- Sierra Wireless Inc.
- Virtusa Corporation
- Yahoo! Inc.

YOUR TRADING HISTORY

TICKER	COMPANY	PRICE	ACTION	ORDER-TYPE	QUANTITY	DATE	STATUS	COMMISSION
CTSH	Cognizant Technol	75.4$	buy	market	20	2011-10-30 13:14:54	completed	15.08$
DELL	Dell Inc.	16.03$	buy	market	1	2011-10-30 13:17:57	completed	0.1603$
CSCO	Cisco Systems, In	18.41$	buy	market	1	2011-10-30 13:18:59	completed	0.1841$
CSCO	Cisco Systems, In	18.41$	buy	market	1	2011-10-30 15:51:01	completed	0.1841$
APOL	Apollo Group, Inc	47.82$	buy	market	1	2011-10-31 12:55:06	completed	0.4782$

TRADE STOCKS

Select Company: Apple inc.

Company: Apple Inc.
Symbol: AAPL
AskPrice: 404.5$
BidPrice: 404.32$
Change: -0.04%
Volume: 13734433

Quantity:
Action: ○ Buy ○ Sell

Order Type: ○ Market ○ Limit ○ Stop

TOP 5 GAINERS:

COMPANY	TICKER	CHANGE
Adobe Systems Inc	ADBE	1.34%
Provident Financi	PROV	0.11%
Apple Inc.	AAPL	-0.04%
Cisco Systems, In	CSCO	-0.16%
Virtusa Corporati	VRTU	-0.43%

TOP 5 LOSERS:

COMPANY	TICKER	CHANGE
Yahoo! Inc.	YHOO	-5.56%
SanDisk Corporati	SNDK	-5.08%
LM Ericsson Telep	ERIC	-4.67%
Cognizant Technol	CTSH	-3.86%
Dell Inc.	DELL	-3.07%

TOP 5 ACTIVE COMPANIES:

COMPANY	TICKER	VOLUME
Cisco Systems, In	CSCO	50545472
Microsoft Corpora	MSFT	46553280
Intel Corporation	INTC	42946104
Yahoo! Inc.	YHOO	39707820
Oracle Corporatio	ORCL	28714804

Fig 7.4 Trading Floor pages

8. PROGRESS REPORT AND PLAN OF WORK:

(a) Progress Report

List of Implemented Use Cases:

- UseCase-1: Login
- UseCase-2: Registration
- UseCase-3: View About Us
- UseCase-4: View About Me
- UseCase-5: Edit My Profile
- UseCase-6: View RSS Newsfeed
- UseCase-7: View My Game Portfolio
- UseCase-8: Go Home
- UseCase-9: Market Order- Buy
- UseCase-10: Market Order- Sell
- UseCase-11: View Trading History
- UseCase-12: View Help Documentation
- UseCase-13: View Trade Diary
- UseCase-14: View Top Five Gainers/Losers
- UseCase-15: View Most Active Companies
- UseCase-16: Delete Account
- UseCase-17: Logout

List of use cases currently being tackled:

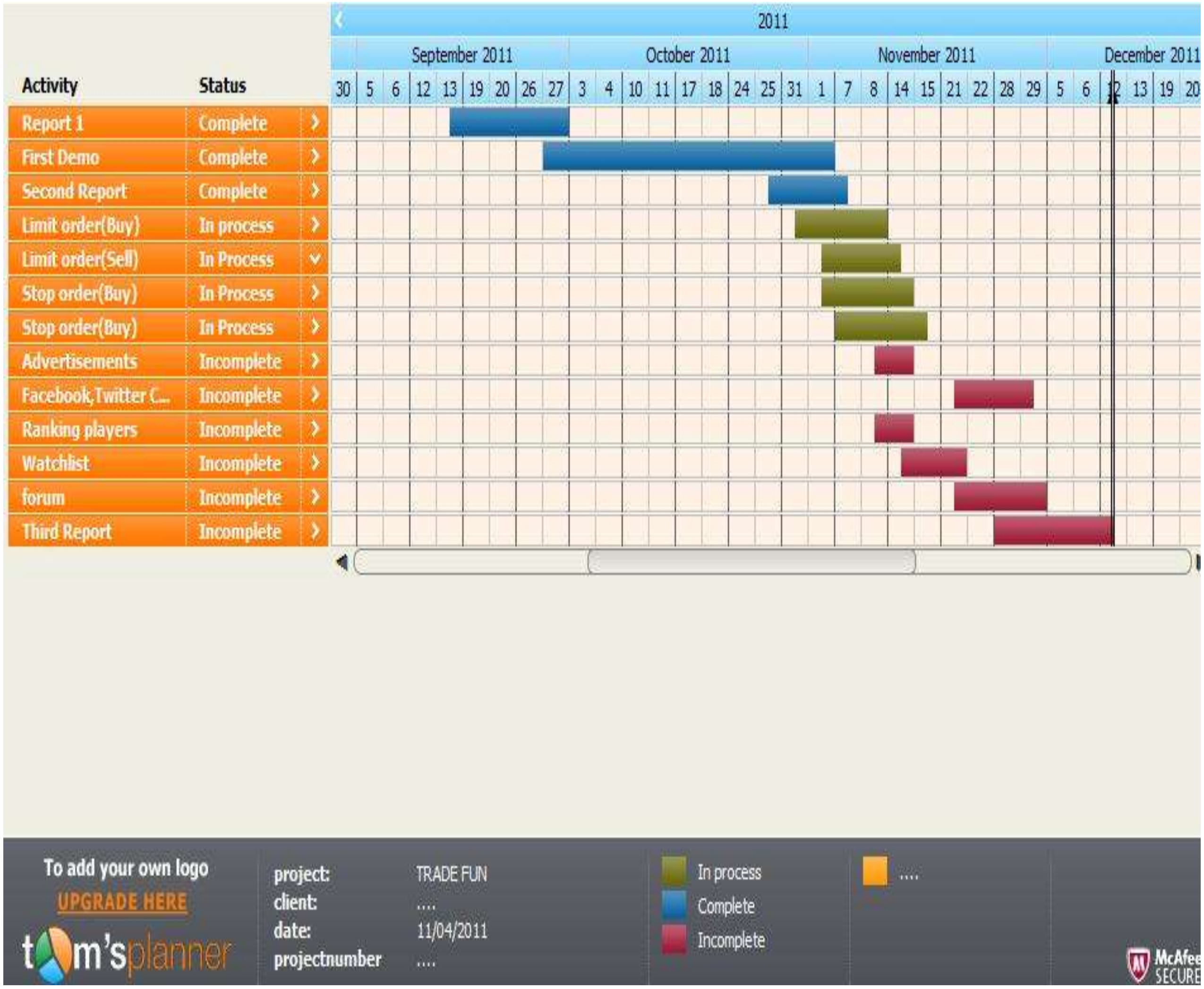
- UseCase-18: Rank players
  In this use case players are automatically ranked according to their portfolio values i.e on the value of their stock holdings
  .
- UseCase-19: Watchlist
  In this use case a user can keep track of their favorite stocks or the stocks which they want to observe before investing on them.

- UseCase-20: Forum
  We are trying to implement a forum where users can stay connected with each other and can post new threads and reply already posted threads.

- UseCase-21: Limit Order- Buy
- UseCase-22: Limit Order- Sell
- UseCase-23:Stop Order- Buy
- UseCase-24:Stop Order- Sell
- UseCase-25: Forgot Password
- UseCase-26: Connectivity with Facebook, Twitter

- UseCase-27: Advertisements

(b) Plan of Work



Activity	Status
Report 1	Complete
First Demo	Complete
Second Report	Complete
Limit order(Buy)	In process
Limit order(Sell)	In Process
Stop order(Buy)	In Process
Stop order(Buy)	In Process
Advertisements	Incomplete
Facebook,Twitter C...	Incomplete
Ranking players	Incomplete
Watchlist	Incomplete
forum	Incomplete
Third Report	Incomplete

project: TRADE FUN
client: ....
date: 11/04/2011
projectnumber ....

(c) Breakdown of Responsibilities:

- Vaishnavi Kakumani: Programming, testing of market order-buy, market order-sell, portfolio, trading history, top 5 gainers/losers, most active companies, E-mail notification modules.
- Nikhila Lavu: Programming, testing of login, registration, edit profile, trading diary, session management, RSS feed modules.

- Pratyusha Nandamuri: Design and validation of all modules.
- Jia Ding: help documentation module.
- Zhiyue Wang: Displaying userdata in About Me page.

Vaishnavi Kakumani does the integration and the testing of the integrated system.

Nikhila Lavu does most of project management and sets the meeting agenda and time limits.

9. REFERENCES

- Wikipedia: Stock market, http://en.wikipedia.org/wiki/Stock_market
- Wikipedia: Finance, http://en.wikipedia.org/wiki/Equity_(finance)
- Investopedia: http://www.investopedia.com/#axzz1XWQ25K7D
- Investopedia: Stock basics, http://www.investopedia.com/university/stocks/default.asp#axzz1XWQPiPfX
- Wall Street Survivor: https://www.wallstreetsurvivor.com/Public/Members/Login.aspx?ReturnUrl=%2fPrivate%2fTrading%2fTrade.aspx
- Up Down: http://www.updown.com/trade-stock
- UML tutorials and reference documents: http://www.uml.org/