Ohio University
Fall 2011
CS456/556
Software Engineering

POST-JOBS TEAM DOCUMENT

Project Name: Let's trade;

Status: Submitted for Final;

Last modified on: - 11/21/2011;

Created on: - 09/28/2011;

Revision number: 106;

Instructor: Dr. Chang Liu

Team Members: Ashwini Naik, Bibo Shi, Rayan Alsubail, Xin Ye

Abstract

The project includes developing a game where in users can buy and sell stocks and they compete with other users playing the game. They use virtual money for the trade. The game will be deployed as an app in the Android and iOS cell phones. The project is being done for the 5 Wide Company of Cincinnati by taking the initial requirements from them. In all, there are 4 teams trying to implement the app under the supervision and guidance of Dr. Chang Liu.

Change History:

Date	Member	Changes done
09-28-2011	Bibo	Created the team document including: --Cover Page. --Bug Cycle --Back-end server Protocol
09-28-2011	Ashwini	Merged the team requirements into one document. Added the table of contents and Original Project description.
09-29-2011	Ashwini	Added the team log section, 1st meeting record, change history.
10-03-2011	Bibo	Refined the team document based on 5a&b. Revised the cover page. Added the required sections.
10-05-2011	Xin	Refined the Team Log section. Documented the team process reflection. Designed three use cases.
10-05-2011	Rayan	Designed eight use cases. Combined the other use cases.
10-07-2011	Bibo	Added the latest meeting log; Added the section of partial analysis; Refined the team document, as in use case diagram.
10-09-2011	Rayan	Edited some use cases & followed Dr. Chang comments: -- Refine use case table -- Rename 2 admin use cases. -- Add actors' definition. -- Reformat 2 user use cases.
10-11-2011	Rayan	Added the team log for today's meeting; Added Analysis Model part & combined other member's jobs. Refined the Analysis part format.
10-18-2011	Ashwini	Added to the latest team log. Added the milestones. Added the TeamWatch screenshot of the team.

10-26-2011	Bibo Shi	Revise the doc based on Dr. Liu's feedback: 1) comment the first API as the temporary; 2) code: format 3) actor symbol 4) sequence diagram (haven't done yet, will ask Ashwini to do) 5) Milestone date 6) table of contents Add new team log; Add deployment mode section, no content yet.
10-29-2011	Rayan	Update the latest team log. Update the change history.
10-29-2011	Ashwini	Added the deployment model diagram and did changes to the document.
10-29-2011	Xin	Added the screenshots for the milestones 1 and 2.
11-01-2011	Rayan	Update the team log and change history in the document.
11-03-2011	Ashwini	Update the team log and add the screenshots for milestone3.
11-08-2011	Xin	Updated the team log.
11-10-2011	Bibo	Updated the team log and change history, added the screenshots for milestone4.
11-21-2011	Bibo	Complete the final version.

Table of Contents:

Index of Tables:

Index of Figures:

Index of Screenshots:

I. Post-Jobs Team Log:

Team meeting hours:

--Tuesday:-5pm

--Friday:-4pm

Team Information:

Name	***Ashwini***	***Bibo***	***Rayan***	***Xin***
Cell	740 856 5669	740 591 4240	202 413 5687	740 591 1297
Email	Ashwinikkhh @gmail.com	darrylbobo @gmail.com	rayan65 @gmail.com	xinxin8179 @hotmail.com
IDE	Xcode 3.2.2	Xcode 4.2	Xcode 3.2.6	Xcode 3.2.6
SDK	iOS 4.3	iOS 4.3	iOS 4.3	iOS 4.3
Device	iphone4	iphone4	iPad	iphone 3gs
PC	Macbook	iMac	iMac	Macbook
OS	OS 10.6.8	OS 10.6.8	iMac OS 10.6.6	OS 10.6.8

Table i. Team Information

Team Structure:

Each week, one of the members will be a team manager and another will be a recorder. The manager will be responsible for coordinating that week's tasks and the meeting. Recorder will be responsible for documenting the team log and refining the document. The two roles rotate among the 4 members for every meeting.

Team Meeting Process Reflection:

Role	***Obligation***	***People***
Manager	1. Manage the meeting. 2. Coordinate with each other regarding the project and assignment. 3. Assign tasks to every member.	Take turns in each meeting.
Recorder	1. Record the meeting logs. 2. Refine the team log section and the document.	Take turns in each meeting.
Team Member	1. Discuss and contribute in the meeting. 2. Work for the project and assignment.	Ashwini, Bibo, Rayan, Xin

Table ii. Team Process

Ashwini, Bibo, Rayan and Xin are all team members, they take turns(clockwise rotation) to be the manager or recorder. For example, if Ashwini is the manager and Bibo is the recorder in one meeting, then Bibo would be the manager and Rayan would be the recorder in next meeting, and Rayan would be the manager and Xin would be the recorder in the meeting next meeting that follows.

What worked: The team manager can help coordinate every one's work and improve the efficiency of the team meeting and team process. The team recorder can help record team logs and refine team logs, thus help make tasks and issues clear for the team. Every team member would be assigned tasks from the manager, everyone would be clear of his/her tasks.

What didn't work: The recorder felt hard to record all the details of every one's tasks in the meeting, and the recorder felt hard to do the record work and the do the discussion at the same time in the meeting.

Pros and cons of different team structures:

1. For a team with no manager or recorder, the pros is that the work flow would be easy because everyone can just decide what he or she want to do. However, the disadvantage of this structure is that the task distribution would be difficult and the decision making would be very difficult because no one has the power to distribute tasks and assign tasks to others.

2. For a team with a fixed manager, the advantage is that the task distribution would be very easy because the manager would take charge of this issue and other members just need to wait for the order from the manager. Also the work flow and decision would be easy. However, the disadvantage of this team structure is that other members except the manager would be passive to the team work.

3. For our team structure, the advantage is that task distribution would be very easy because the manager would take charge of this issue and other members just need to wait for the order from the manager. But the disadvantage is that the work flow would be a little complex because everyone takes turn to be the manager or recorder. However, everyone can have the chance to be the manager and be more active and more obsessive to team works. The manager also helps improve the efficiency and effectiveness of decision making.

In sum, we decide to set a team manager and a team recorder, and everyone takes turn to be the manager or recorder. This team structure and process help improve the efficiency and effectiveness of decision making and task distribution. Even though the work flow would be a little complex, but everyone would be more active and obsessive to team works.

Meetings Log:

--Week 10, Thursday, 11/10/2011, 5pm, Vital Lab, Room 201

People Present—All members

Issues:

1. Discussed the team assignment given and who should do which part.

Tasks:

- Xin
    - What I have done: Worked for meeting milestone4 use cases.
    - What I plan to do: Do the write-up for Testing and Analysis part of the assignment.
- Ashwini
    - What I have done: Checked out the server code.
    - What I plan to do: 1. Write the User's Guide to be put in the app
      Do the Settings page of the app.
- Rayan (Manager)
    - What I have done: Worked for meeting milestone4 use cases.
    - What I plan to do: Work on improvising the app and putting the document in the app.
- Bibo (Recorder)
    - What I have done: Worked for meeting milestone4 use cases.
    - What I plan to do: Design the paper presentation for the final demo.

--Week 10, Tuesday, 11/8/2011, 5pm, Vital Lab, Room 201

People Present—All members

Issues:

1. Discussed the server side code for the use cases of milestone4.
2. Each member of the team was assigned work as part of the team assignment.

Tasks:

- Xin (Recorder)

- What I have done: Linked the app with the server.
- What I plan to do: Wrap up the server side coding and update the team log.

➢ Ashwini

- What I have done: Updated the team document.
- What I plan to do: Check out the server code.

➢ Rayan

- What I have done: Fixed server code problems.
- What I plan to do: 1. Working for milestone4 functions.
  2. Work on the graph for the stocks.

➢ Bibo (Manager)

- What I have done: Wrapped up with the server code for User Info and leader board modules.
- What I plan to do: Do modifications required for milestone4.

--Week 9, Thursday, 11/3/2011, 5pm, Vital Lab, Room 201

People Present—All members

Issues:

1. Discussed the server side code for the various functionalities.
2. Each member of the team was assigned work.

Tasks:

➢ Xin (Manager)

- What I have done: Did the coding for certain functionalities.
- What I plan to do: Improve the code to make the app get information from the server.

➢ Ashwini (Recorder)

- What I have done: Tried to link View Markets with server.
- What I plan to do: Update the team document and do write-up for milestone3.

➢ Rayan

- What I have done: Make the buy and sell functions work and updated the team document.
- What I plan to do: Work on server code.

➢ Bibo

- What I have done: Linked the app with the server for getting the user info.
- What I plan to do: Improve my code.

--Week 9, Tuesday, 11/1/2011, 5pm, Vital Lab, Room 201

People Present—All members

Issues:

1. Discussed what all should be done for milestone3.
2. Each member of the team was assigned work.

Tasks:

➢ Xin

- What I have done: Wrote the code to link the app to the server for few functions.
- What I plan to do: Complete the code part for milestone3.

➢ Ashwini (Manager)

- What I have done: Checked out server code.
- What I plan to do: Linking of the server to the app for the View Markets module.

➢ Rayan (Recorder)

- What I have done: Dealt with the functions regarding the stocks.
- What I plan to do: Make the buy and sell functions work and update the team log.

.

➢ Bibo

- What I have done: Updated the team document.
- What I plan to do: Work for milestone3, linking he app with the server for getting the user info.

--Week 8, Thursday, 10/27/2011, 5pm, Vital Lab, Room 201

People Present—All members

Issues:

1. Discussed the linking of the app to the server.
2. Each member of the team was assigned work.

Tasks:

- Xin
  - What I have done: Did the server side code for Buy Stock and Sell Stock classes.
  - What I plan to do: 1. Work on how to link the Server to the app.

- Ashwini
  - What I have done: Modified the screen designed by me.
  - What I plan to do: 1. Check out linking of the server to the app for the next milestone.
    2. Design the Deployment model.
- Rayan (Manager)
  - What I have done: Modified the User interface screens for the "View Stocks" module (removed the Price label).

    What I plan to do: Work with Xin regarding the server code for the buy and sell functions.

- Bibo (Recorder)
  - What I have done: Designed the screens for User Info and Leaderboard.
  - What I plan to do: Do the write-up for milestone2 and update the team document.

    .

--Week 8, Tuesday, 10/25/2011, 5pm, Vital Lab, Room 201

People Present—All members

Issues:

1. Discussed the functionalities that should be completed as part of milestone2.
2. Each member of the team was assigned work as part of the team assignment.

Tasks:

- Xin (Recorder)
  - What I have done: Did the server side code for registration of user.
  - What I plan to do: 1. Work for milestone2 and update the team document.

- Ashwini

- What I have done: Designed the screen for View Markets module.
- What I plan to do: 1. Modify the previously done screens.

➢ Rayan

- What I have done: Design the User interface screens for the "View Stocks" module
- What I plan to do: Modify the User interface screens for the "View Stocks" Module following the Professor's comments.

➢ Bibo (Manager)

- What I plan to do: Start working on the user interface screens for User Info and leaderboard.

--Week 7, Tuesday, 10/18/2011, 5pm, Vital Lab, Room 201

People Present—All members

Issues:

2. Discussed the server side code for the login and logout module of the app and checked out the PHP script on the server.
3. Each member of the team was assigned work as part of the team assignment.

Tasks:

➢ Xin (Manager)

- What I have done: Did the server side code for login and logout modules.
- What I plan to do: 1. Do the write-up for milestone1.
  2.Get the permission from the lab administrator for the database.

➢ Ashwini (Recorder)

- What I have done: Finished the design of sequence diagram.
- What I plan to do: 1. Update the team document with the latest changes
  2. Design the User interface for "View Markets" module.

➢ Rayan

- What I have done: Refined the team document.
- What I plan to do: Design the User interface screens for the "View Stocks" module.

--Week 6, Tuesday, 10/11/2011, 5pm, Vital Lab, Room 201

People Present--All members

Issues:

1. Discussed http API options $ the possible ways of connection, choose "ASIHTTPRequest".
2. Assign team homework to each member.

Tasks:

➢ Ashwini (Manager)
- What I have done: Finished the design for menu structure with Bibo.
- What I plan to do: 1. Be responsible of drawing a sequence diagram.
  2. Finish the "Buy & Sell from server" team homework on the device side.

➢ Bibo
- What I have done: 1. Finished the design for menu structure with Bibo.
  2. Added the team log & refine the document.
- What I plan to do: Be responsible of identifying the attributes & responsibilities for two classes (Stock, Buy Stock).

➢ Rayan (Recorder)
- What I have done: 1. Found some ways for using the database in device.
  2. Refined the document by following Dr. Chang comments.
- What I plan to do: 1. Add what I recorded to the team log.
  2. Combine the other members' works, refine the document.

➢ Xin
- What I have done: 1. Finish the "Buy & Sell from server" team homework on the server side.
  2. Searched on how to establish server-device connection & provide members with helpful sources.
  3. Tested the menu structure.
- What I plan to do: 1. Identify all entity classes.
  2. Identify a boundary & an object class.

--Week 5, Friday, 10/07/2011, 4pm, Vital Lab, Room 201

People Present--All members

Issues:

1. Discuss and decide the menu structure for project, choose "TAB" way.

2. Assign team homework to each member.

Tasks:

- ➢ Ashwini
  - What I have done: 1.Fished the use case diagram and two use cases.
  - What I plan to do: 1. Be mainly responsible to finish the design for menu structure with Bibo.
- ➢ Bibo (Recorder)
  - What I have done: 1. Changed icon of the let's trade project.
    2. Wrote two use cases.
  - What I plan to do: 1. Be mainly responsible to finish the design for menu structure with Ashwini.
    2. Change the document according this meeting.
- ➢ Rayan (Manager)
  - What I have done: 1. Finished 8 use cases and combined with the other member's cases.
  - What I plan to do: 1. Be responsible to figure out how to use the database in device.
    2. Be responsible to research on the next team homework, server-device hello.
    3. Also refine the menu structure.
- ➢ Xin
  - What I have done: 1. Refined the team log section.
    2. Added splash feature to the project.
    3. Wrote 3 use cases.
  - What I plan to do: 1. Also refine the menu structure.

--Week 5, Tuesday, 10/04/2011, 5pm, Vital Lab, Room 201

People Present--All members

Issues:

1. Design the use cases of the project.

2. Assign use case writing tasks to all members.

Tasks:

- ➢ Ashwini

- What I have done: 1.Add text in the about window for the HelloWorld2 project assignment.
- What I plan to do: 1. Draw use case diagram.
  2. Write two use cases.

➢ Bibo (Manager)
- What I have done: 1. Refine the project document for assignment5a.
- What I plan to do: 1. Change icon of the let's trade project.

  2. Write two use cases.

➢ Rayan
- What I have done: 1.Create the HelloWorld2 project and create an issue regarding it.
- What I plan to do: 1. Write 8 use cases.

➢ Xin (Recorder)
- What I have done: 1.Create trunk, branches and tags folders on the svn repository.
- What I plan to do: 1. Refine the team log section
  2. Add splash feature to the project.
  3. Write 3 use cases.

--Week 4, Thursday, 09/29/2011, 7pm, Vital Lab, Room 201

People Present--All members

Issues:

1. Establish the team process.
2. Assign works of HelloWorld2 project to everyone.
3. Discuss of how to refine the project document.

Tasks:

➢ Ashwini(Recorder)
- What I have done: 1. Merged the requirements of all the team members and formed one single project description document.
- What I plan to do: 1. Contribute for the HelloWorld2 project assignment.

➢ Bibo
- What I have done: 1. Created the Team Document as specified by the Professor in assignment 4.
- What I plan to do: 1. Contribute for the HelloWorld2 project assignment.

- Rayan
    - What I have done: 1. Did some study on iphone application development and came up with some ideas.
    - What I plan to do: 1. Create the HelloWorld2 project and create an issue regarding it.
- Xin(Manager)
    - What I have done: 1. Helped set up the development environment in the lab.
    - What I plan to do: 1. Contribute for the HelloWorld2 project assignment.

II. Original Project Description

Development Project Description Document

5 Wide, LLC

STOCK EXCHANGE APP

The purpose of the project is to create a smart phone application on both iPhone and Android Platforms that allows users to compete to see who can be the most profitable day trader on the virtual stock exchanges of the world. The functionality of the yahoo stock exchange app will be a good reference for this project. A key difference being that the app designed by the students will include other stock exchanges from around the world or even created by the students as opposed to offering just the NY stock exchange.

There are no requirements regarding graphical look and feel beyond serving the functional purpose of the app. Students are free to design look and feel as they please, create their own name for the app and even the create their own commodities and stock exchanges. Students are encouraged to research the imagery of the NY stock exchange and other stock exchanges of the world as well as apps such as yahoo!stocks, scottrade, etrade, etc for ideas.

PROJECT CONTACTS:

Vincent Cordo +1.973.580.7470 vincentjcordo@gmail.com

Chris Nye +1.513.600.3403 chris.j.nye@gmail.com

COMMUNICATIONS

- Vincent & Chris will visit the class at least twice during the quarter and be available for a weekly or bi-weekly conf call. This should be noted in the timeline schedule worked on by the students in the requirements phase of this effort.
- Further questions regarding the project will be collected weekly by Chang Liu and forwarded to Vincent and Chris to avoid repeats of questions

LICENSING

- Licensed such that only 5Wide has the right to distribute the source code or use the source code in business.

DELIVERABLES

1. Software application and associated files necessary for submission in the appropriate application store for each end device.
2. Software source code, project files, and any files necessary to rebuild the application.
3. Detailed documentation of the necessary steps to recreate the software build.
4. Software user documentation.
5. Include the suite of test cases to ensure that the requirements are met. (unit tests or functional tests, whatever is appropriate)

SOFTWARE REQUIREMENTS

1. To be designed as native applications for Android and iOS.
2. Will use the lowest possible API version necessary to accomplish the requirement set forth in this document.
3. GUI is to be designed for a normal definition and high definition display on hand held devices.
4. Must function in portrait orientation.
5. All images used in the software shall be changeable without the need for a recompile.

6. Development of the backend server supporting the game in PHP
7. API calls that talk using JSON.

PRODUCT REQUIREMENTS

1. Capability for multiple users interacting with the game on their phones at once.
2. On the server side, a web page will allow the developer to edit the following properties for each stock and stock exchange.
    a. Stock exchange properties:
        i. Stock exchange name
        ii. Stock exchange time zone
        iii. The name of the stocks traded in that stock exchange
        iv. A high point and low point setting for each stock
        v. A function that sets the rate at which a stock price "crashes" and "climbs". The function will be dependent upon the volume of shares of that particular stock sold in that stock market.
        vi. Current stock price – the functionality to directly edit the current stock price to simulate an instant “crash” or “climb”.
    b. Stock properties. – stock properties can override the properties set in stock exchanges by selecting the stock exchanges that will be overridden by the stock properties settings.
        i. Stock name
        ii. A “volume of shares sold” based pricing function that applies to one, several or all stock exchanges. Thus the stock price will have the option of operating as a function of the volume sold in the selected one, several or all stock exchanges.
        iii. A high point and low point setting for each stock that applies to the selected one, several or all stock exchanges.
        iv. Current stock price – the functionality to directly edit the current stock price to simulate an instant “crash” or “climb” that applies to the selected one, several or all stock exchanges.
        v. Each stock will have a volatility rating with 3 levels (high, med, low) based on the ratio of their high and low point
    c. Unlimited number of users, stocks and stock exchanges
    d. Capability to create new stocks to be traded on the market
    e. Capability to add or remove stocks to each stock exchange

    f. Capability to add new stock exchanges and remove old ones

3. A screen displaying a list of the stocks available on the user selected stock exchange.
    a. The list will include stock name, price, absolute $ change since open, % change since open and the volatility rating (assume that each stock exchange closes and instantly opens every day at 9am in the time zone of the stock exchange)
    b. The list is sortable by stock name A-Z, price and $/% change since open.
    c. The functionality to select the stock for purchase
    d. Search functionality to search for and select another stock exchange to be displayed
    e. A graph display similar to that of yahoo stocks is a “nice to have” but not a hard requirement.
4. Functionality for a user to select the number of shares of a stock that they'll purchase
5. Each user will have a username and password. User’s credentials are to be kept in the server. When a user logs in to the app they are really logging on the server and the app will display their purchase history and sell net gain/loss
6. A screen displaying the individual user’s current positions and net gain/loss.
    a. For stocks that have been purchased but not sold - stock name, purchase price, # of shares and current market price
    b. For stocks that have been sold - the net gain/loss for each sell
7. A winners/losers boards, displaying each user’s username and current net gain/loss $ and %
8. To maintain market activity in the case that there are a low number of users. The server will have “dummy” ay traders. These “dummies” will buy and sell stocks at a random frequency and thus at random prices

III. Revised Project description

[Blue Text: Modifications by team]

Development Project Description Document

5 Wide, LLC

STOCK EXCHANGE APP

The purpose of the project is to create a smart phone application (game application) on both iPhone and Android Platforms that allows users to compete to see who can be the most profitable day trader on the virtual stock exchanges of the world .,the users will trade with a default number of credits initially. The functionality of the yahoo stock exchange app will be a good reference for this project. A key difference being that the app designed by the students will include other stock exchanges from around the world or even created by the students as opposed to offering just the NY stock exchange.

There are no requirements regarding graphical look and feel beyond serving the functional purpose of the app. Students are free to design look and feel as they please, create their own name for the app and even the create their own commodities and stock exchanges. Students are encouraged to research the imagery of the NY stock exchange and other stock exchanges of the world as well as apps such as yahoo!stocks, scottrade, etrade, etc for ideas.

PROJECT CONTACTS:

Vincent Cordo +1.973.580.7470 vincentjcordo@gmail.com

Chris Nye +1.513.600.3403 chris.j.nye@gmail.com

COMMUNICATIONS

- Vincent & Chris will visit the class at least twice during the quarter and be available for a weekly or bi-weekly conference call. This should be noted in the timeline schedule worked on by the students in the requirements phase of this effort.

LICENSING

- Licensed such that only 5Wide has the right to distribute the source code or use the source code in business.
- The development should be under MIT license, but this is not the main goal of the project.
- It is supposed to be a free app i.e., no money will be charged for its installation and usage.

DELIVERABLES

1. Software application and associated files necessary for submission in the appropriate application store for each end device.
2. Software source code, project files, and any files necessary to rebuild the application.
3. Detailed documentation of the necessary steps to recreate the software build.
4. Software user documentation.
5. Include the suite of test cases to ensure that the requirements are met. (unit tests or functional tests, whatever is appropriate)

SOFTWARE REQUIREMENTS

1. To be designed as native applications for Android and iOS.
2. Will use the lowest possible API version necessary to accomplish the requirement set forth in this document.
3. GUI is to be designed for a normal definition and high definition display on hand held devices.

--The GUI design should be identical for both iphone and Android platforms.
--The GUI design should have no buttons or menu on Android, since iphone doesn't have those.

--It should be able to resize itself to any resolution.

4. Must function in portrait orientation.

   --Landscape orientation isn't required.

5. All images used in the software shall be changeable without the need for a recompile.
6. Development of the backend server supporting the game in PHP

   --Both of the iphone and android should use the same server.

   --A production server and a database, which holds a small amount of information of a few stocks are built and maintained by the students.

   --The database information is necessary to design test cases which help the customer validate the app**.**

7. API calls that talk using JSON.

PRODUCT REQUIREMENTS

1. Capability for multiple users interacting with the game on their phones at once.

   --One device can allow different users to play, but just one game at one time.

2. On the server side, a web page will allow the developer to edit the following properties for each stock and stock exchange.

   a) Stock exchange properties:
      i. Stock exchange name
      ii. Stock exchange time zone

         --Currently, different stock markets use the same time zone, with the same currency in dollars. And, the opening and closing time should be designed for future development, but no gap between them right now.

         --The price of a stock should be the same at the open and last close time. It is suggested to have the closing time at 9:00 AM and opening time just a few seconds after it.

iii. The name of the stocks traded in that stock exchange

iv. A high point and low point setting for each stock

v. A function that sets the rate at which a stock price "crashes" and "climbs". The function will be dependent upon the volume of shares of that particular stock sold in that stock market.

vi. Current stock price – the functionality to directly edit the current stock price to simulate an instant “crash” or “climb”.

b) Stock properties. – stock properties can override the properties set in stock exchanges by selecting the stock exchanges that will be overridden by the stock properties settings.

i. Stock name

ii. A “volume of shares sold” based pricing function that applies to one, several or all stock exchanges. Thus the stock price will have the option of operating as a function of the volume sold in the selected one, several or all stock exchanges.

--The game does not need deal with the dividend issue.

iii. A high point and low point setting for each stock that applies to the selected one, several or all stock exchanges.

iv. Current stock price – the functionality to directly edit the current stock price to simulate an instant “crash” or “climb” that applies to the selected one, several or all stock exchanges.

--The system can use any function for calculating the change of price of a stock in real time, the price of a stock can change linearly or it can change with a complex mathematical curve.

v. Each stock will have a volatility rating with 3 levels (high, med, low) based on the ratio of their high and low point

vi. Stocks bought in one market should be sold in that market only.

vii. Different stock exchanges can have the same stock with different prices.

c) Unlimited number of users, stocks and stock exchanges

--Unlimited implies only the amount of load the server will be able to handle, keeping the requirement of maintaining the min. response time in consideration.

d) Capability to create new stocks to be traded on the market

e) Capability to add or remove stocks to each stock exchange

f) Capability to add new stock exchanges and remove old ones

--Only the administrator can perform the above 3 functions.

g. No offline game is allowed.
--If there is no network connection, the game can still show information like “no network connection”.

3. A screen displaying a list of the stocks available on the user selected stock exchange.

    a. The list will include stock name, price, absolute $ change since open, % change since open and the volatility rating (assume that each stock exchange closes and instantly opens every day at 9am in the time zone of the stock exchange)
    b. The list is sortable by stock name A-Z, price and $/% change since open.
    c. The functionality to select the stock for purchase

    --The money used to purchase is virtual, not real.

    --All the stocks in this game should have a minimum volume constraint for buying which means that a player can only buy a stock when he/she buys more than the minimum volume of this stock.

    d. Search functionality to search for and select another stock exchange to be displayed.

    e. A graph display similar to that of yahoo stocks is a “nice to have” but not a hard requirement.

4. Functionality for a user to select the number of shares of a stock that they'll purchase
5. Each user will have a username and password. User’s credentials are to be kept in the server. When a user logs in to the app they are really logging on the server and the app will display their purchase history and sell net gain/loss

    --Only one game can be played per username at any point of time. The server validates the user’s credentials once he/she logs in.

    --Users play the game with each other not with the computer.

    --The new user will have to register for the game.

6. A screen displaying the individual user’s current positions and net gain/loss.
    a. For stocks that have been purchased but not sold - stock name, purchase price, # of shares and current market price

    b. For stocks that have been sold - the net gain/loss for each sell

7. A winners/losers boards, displaying each user's username and current net gain/loss $ and %

   --They will lose the game when they have no money or credits left and have sold of all the stocks he has. They can end the game in the middle. Both of these will to lead a new start of the game.

   --The user who has earned the maximum profit in the game will be declared as the winner.

8. To maintain market activity in the case that there are a low number of users. The server will have "dummy" ay traders. These "dummies" will buy and sell stocks at a random frequency and thus at random prices

   --1 to 2 dummies to be present who randomly keep buying and selling the stocks just to maintain the market activity.

IV. Issue life cycle

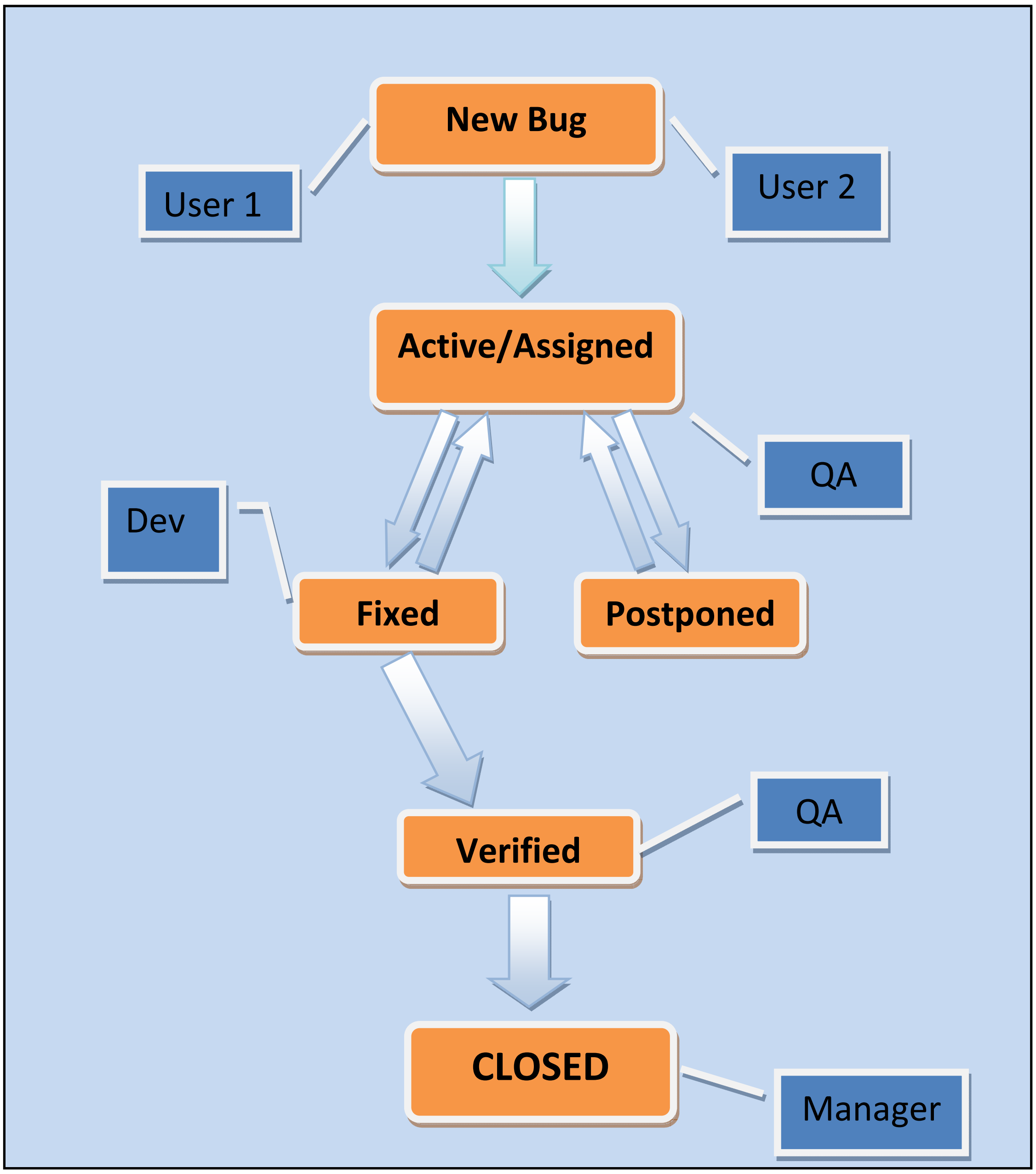


Figure i. Issue Lifecycle

Note: The above figure shows the bug life cycle in POST-Jobs groups. Each week, the four members will be assigned to the role of "User1", "User2","QA", and "Manager". Meanwhile, all of them four are also the "Dev". They will follow the rule that "only the manager can close the bug". And also, they will change the role each week, so that everyone has the chance to experience each role in the bug life cycle.

V. Standard API for the Backend Server

Server Spec – Collaborated

This document was started from the GLaDOS API doc and edited by collaboration between members of each of the four teams.

HTML requests passing JSON objects:

PHP for server side processing – JSON requests will be received, parsed, and executed, and JSON response will be echoed back.

PHP will interact with the MySQL database.

- App must be able to function with only the below implemented by the server.
- Server must provide all the below.

An HTML request (post) with title "request" is sent to a single php page, which calls other functions (separate pages) as appropriate (by operation – see below)

- server doesn't care if requests are http or https.
- error messages are specified per request
- all calls contain a variable "operation" which specifies the operation to complete. Each operation will have a function which will be called, with the rest of the json data passed as an argument.
- the login relationship between the app and the server depends on sessionId. On Login, the server returns a sessionId which is stored on the phone. Until the User Logs out, the session is maintained by the server and device. This is used to establish relationship on every mutable or secure call (user data).

Functions are UpperCamelCase, parameters are lowerCamelCase, types areUpperCamelCase.

All server calls must have the following fields:

- `operation(string)` // (containing the operation of the call)
- `data(array)` // contains each of the remaining parameters
- `sessionId(string)` //(only if the call requests or modifies user data)

All server response objects must have the following fields

- `status(enum { success, loggedOut, error})`
- `serverTime(datetime)`
- `string display_error` //error message for user, empty if none.

Format in JSON:

```
{
   “operation”:”op1”,
   “data”: {
              “param1”:“<param1val>”
              “param2”:“<param2val>”
            }
 }
```

Example: Function validateUserName:

```
{
   “operation”:”validateUserName”,
   “data”: {
              “username”:“<username>”
           }
}
```

Example: of return data:

```
{
  “response”: [“retparam1”:”retval1”]
}
```

Example: for validateUserName return:

```
{
   “response”: [“valid”:”<valid>”]
}
```

Server Operations:

Function - **Description**	**operation(string)=”operationname”** **and** **members of data[]** **name(type)[=defaultvalue]**	**Return data**
validateUniqueness - Check for unique or already set username or Email address	- operation(string)=”validateUniqueness” - username(string) - email(string)	-validUsername(bool) -validEmail(bool)
CreateUser - Creates a new user on the server	- operation(string)=”createUser” - username(string) - password(string) - email(string)	
EditProfile - Edits a user profile	- operation(string)=”editProfile” - username(string) - password(string) - newUsername(string) - newPassword(string) - newEmail(string)	
Login - Logs a user into the server	- operation(string)=”login” - username(string) - password(string)	- sessionId(string) - username(string) - email(string)
Logout - Logs out a user	- operation(string)=”logout” - sessionId(string)	- status(Status)
NewGame - Starts a new game	- operation(string)=”newGame”	- servertime(datetime)

for a user	- sessionId(string)	
BuyStock - Purchases stock for a user	- operation(string)=”buyStock” - sessionId(string) - stockId(string) - quantity(integer)	- purchasePrice(float) - status(Status) - servertime(datetime)
SellStock - Sells a user’s stock	- operation(string)=”sellStock” - sessionId(string) - stockId(string) - quantity(integer)	- salePrice(float) - status(Status) - servertime(datetime)
GetExchangeInfo - Returns info about a stock exchange	- operation(string)=”getExchangeInfo” - exchangeId(string)	- Info(ExchangeInfo)

GetExchanges -
 	
 Returns a list of exchanges
	- operation(string)=”getExchanges”	-
		 numExchanges(integer)
		
		 -
		 exchanges(ExchangeInfo[])

GetStockInfo - Returns info about a stock	- operation(string)=”getStockInfo” - stockId(string) - startTime(datetime) - endTime(datetime) - interval(int) (seconds)	- Info(StockInfo) - servertime(datetime)
GetStocks	- operation(string)=”getStocks” - exchangeId(string)	- stocks(StockInfo[]) - servertime(datetime)
GetCurrentStocks	- operation(string)=”getCurrentStock” - sessionId(string)	- stocks(StockHolding[]) - servertime(datetime)
GetSoldStocks	- operation(string)=”getSoldStock” - sessionId(string)	-stocks(StockHolding[]) - servertime(datetime)

GetGameLength	- operation(string)="getGameLength" - sessionId(string)	- gameLength(timespan) - servertime(datetime)
GetLeaderboard - Returns the leaderboard of top users	- operation(string)="getLeaderboard" - numUsers(int)	- topUsers(Ranking[]) - currentUser(Ranking) - servertime(datetime)
GetUsers - gets the number of users and logged in users	-operation(getUsers)="getUsers"	- registered(int) - loggedIn(int)

Table iii. Server operations

Custom types (represented as nested arrays in JSON)

Title	**- memberName(type)**
ExchangeInfo	- exchangeId(int) - longName(string) - shortName(string) - numStocks(integer) - open(bool) - timeZone(string)//human readable - opentime(time) - closetime (time)
StockInfo	- stockId(int) - exchangeId(int) - longName(string) - stockName(string) - volitilty(string) - currentPrice(float) - openingPrice(float)

	- priceHistory(PricePoint[])
Ranking	- username(string) - ranking(integer) - netWorth(float) - worthIncreaseToday(float)
StockHolding	- stockName(string) - shares(integer) - currentValue(float) - purchasePrice(float) - salePrice(float)=”0.0” (0 if not sold)
PricePoint	- serverTime(datetime) - price(float) (negative if no price data for this time)
Datetime	- time(float) (seconds since unix epoch)

Enums

Name	Type	Options
Status	string	- success - loginRequired - errorString - undefined //function is not defined on this server implementation Example: {“status”:”successful”}

		{“status”:”loginRequired”}
		
		{“status”:”error string - some errors....”}
		
		{“status”:”undefined”}

Table iv. Custom Types(Server Code)

VI. Use Case Model

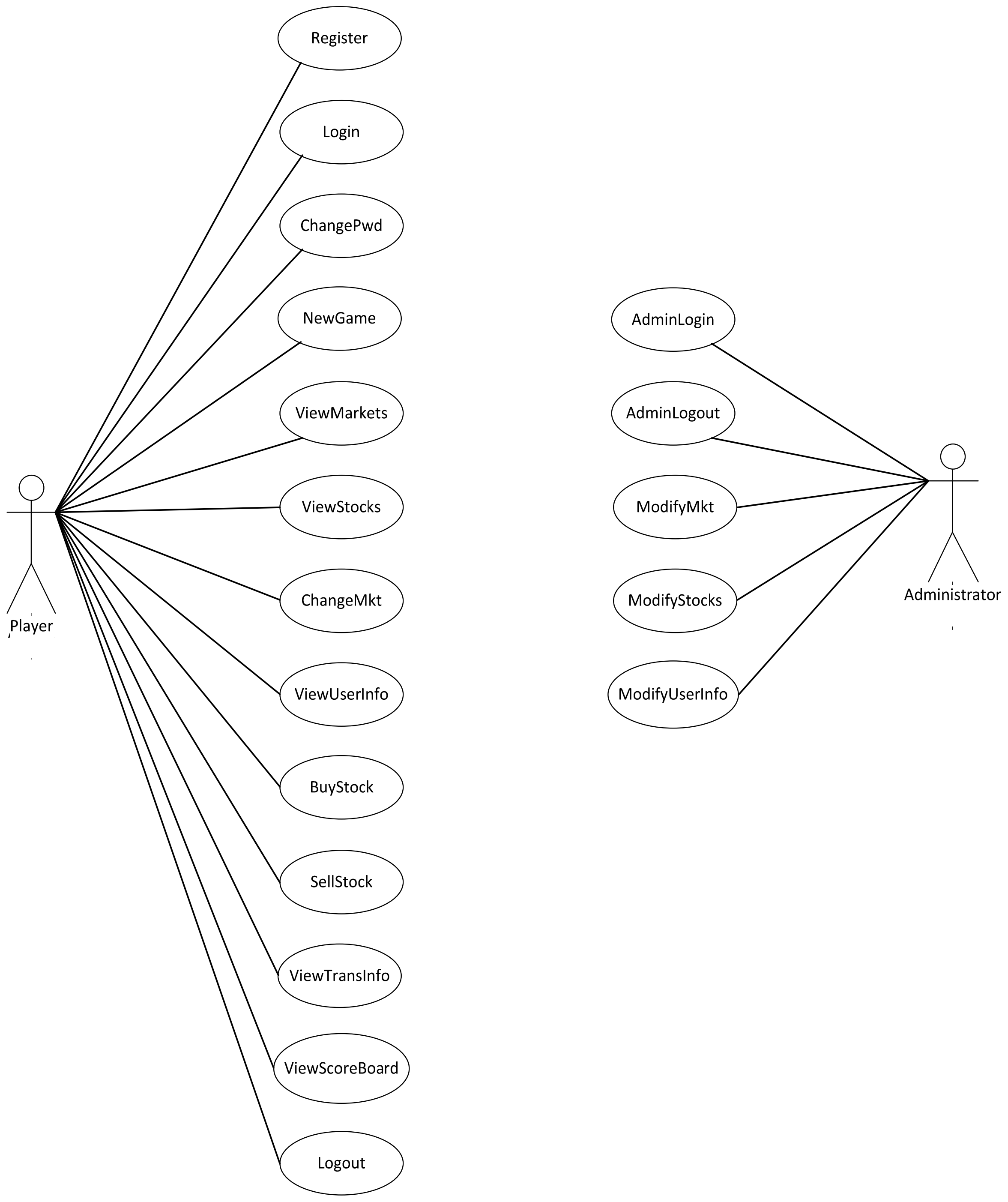


Figure ii. Use Case Diagram

1. Definition of Actors:

Player:

User is the player, the trader and the iOS user. User has the most part of the system. A user would play the game by buying & selling stocks and compete with other users on the amount of gain. User can register itself, create new game, view all markets, change the current market, view all stocks of the chosen market and can view its information as well.

Administrator:

The administrator is the admin of the game. An administrator can add, remove or edit markets & stocks. The administrator also would have the ability to edit the information of users or even delete a user. An administrator can only access the system from the server side and he/she has to be login in order to do its job.

2. Use Cases Descriptions:

Player Side:

Name: Register
Actors: Player
Priority: High
Entry Condition:
User opens the application in his phone
Exit Condition:
User successfully registers himself.
Flow of events:
1. User opens the application in his phone.
2. If he's a new user then he has to register himself to play the game.
3. He enters details in the registration form which includes his email id and choosing a password for his account.
4. Server accepts the registration and sends a message to the user saying he has successfully completed registering.

Special Requirements:
- Every user must compulsorily have an email account.

Name: Login
Actors: Player
Priority: High
Entry Condition:

User must already be registered.
Exit Condition:
User logs in successfully and begins the game.
Flow of events:
1. User enters the login page and gives his email id and password.
2. Server verifies the information and gives access to the user.
Special Requirements:
- The login must be processed by the server quickly within 30 seconds.

Name: NewGame
Actors: Player
Priority: Medium
Entry Condition:
The user runs out his or her balance or the user wants to restart the game with a new transaction history and default balance.
Exit Condition:
User enter into the game with a blank transaction history and default balance.
Flow of events:
1. User clicks the "New Game" button on the toolbar menu.
2. User's transaction history is cleared to blank.
3. User's current balance changes to the default value.
4. User will not be in the leader board in the current day.
Special Requirements:
--It takes no longer than five seconds for the user to enter into a new game.

Name: Logout
Actors: Player
Priority: Medium
Entry Condition:
The user wants to logout his or her current user account.
Exit Condition:
User logout his or her user account.
Flow of events:
1. User clicks the "Logout" button on the higher right corner of the toolbar.
2. A confirmation window will show up asking the user whether he or she decide to logout.
3. If user clicks the "No" button, then this confirmation window will disappear.
4. If user clicks the "Yes" button, then this confirmation window will disappear and the user logout his or her current user account.
5. The "Logout" button on the higher right corner of the toolbar changes to the "Login" button.
Special Requirements:
--It takes no longer than five seconds for the user to logout his or her user account.

Name: ViewScoreBoard
Actors: Player
Priority: Medium
Entry Condition:
The user has logged in.
Exit Condition:
None
Flow of events:
1. User clicks on “Leadership” button on the tab bar.
2. User can see the top ranked traders and its own rank.
3. User clicks any other button on the tab bar to get out.

Name: ViewUserInfo
Actors: Player
Priority: Medium
Entry Condition:
The user has logged in.
Exit Condition:
None
Flow of events:
1. User clicks on “My info” button on the tab bar.
2. User can see all of his/her information.
3. User clicks any other button on the tab bar to get out.

Name: View transaction history
Actors: Player
Priority: Medium
Entry Condition:
The user has logged in.
Exit Condition:
None
Flow of events:
1. User clicks on “My info” button on the tab bar.
2. User clicks on “Transactions history” button on the tab bar.
3. User can see his/her last transactions info.
4. User clicks “done” to get back to “My info” view.
5. User clicks any other button on the tab bar to get out
Quality Requirements:
- The system should list no less than five transactions.

Name: ChangeMkt
Actors: Player
Priority: Medium
Entry Condition:
The user has logged in.
Exit Condition:
None
Flow of events:
1. User clicks on “View Stocks” button on the tab bar.
2. User clicks on “View all Markets” button on the tab bar.
3. User can see a list of all markets.
4. User clicks on one of the markets to choose it.
5. The system shows a list of the chosen market’s stocks.
5. User clicks any other button on the tab bar to get out

Special Requirements:
- If there is no more than one market, the “View all Market” button should be disabled.

Name: ViewMarkets
Actors: Player
Priority: Medium
Entry Condition:
The user has logged in.
Exit Condition:
None
Flow of events:
1. User clicks on “View Stocks” button on the tab bar.
2. User clicks on “View all Markets” button on the tab bar.
3. User can see a list of all markets.
4. User clicks “done” to get back to “My Stocks” view
5. User clicks any other button on the tab bar to get out

Special Requirements:
- If there is no more than one market, the “View all Market” button should be disabled.

Name: ViewStocks
Actors: Player
Priority: Medium
Entry Condition:
The user has logged in.
Exit Condition:
None
Flow of events:

1. User clicks on “View Stocks” button on the tab bar.
2. User can see a list of the chosen market’s stocks.
5. User clicks any other button on the tab bar to get out

Special Requirements:
-The system should show the stocks within 30 seconds.

Name: BuyStock
Actors: Player
Priority: Medium
Entry Condition:
The user is viewing the special stock.
Exit Condition:
User’s current balance and transaction information is changed according to how much he bought.
Flow of events:
1. User clicks “Buy” button on “View Stock” tab.
2. A transaction page displays.
3. If user doesn’t want to make this transaction, he clicks “Cancel” button to get out.
4. If user is sure to make this transaction, he inputs the amount he wants to buy at the current price, and clicks “Deal” button.
5. If user has enough money to make this transaction, this transaction is successful, and user receives the message “Transaction Successful!”. Then, go to 7.
6. If user has not enough money to make this transaction, user receives the message “Transaction Fails. No enough money!” Then, go to 2.
7. User clicks “Done” button to get out of this transaction page.

Name: SellStock
Actors: Player
Priority: Medium
Entry Condition:
The user is viewing the specific stock.
Exit Condition:
User’s current balance and transaction information is changed according to how much he bought.
Flow of events:
1. User clicks “Sell” button on “View Stock” tab.
2. A transaction page displays.
3. If user doesn’t want to make this transaction, he clicks “Cancel” button to get out.
4. If user is sure to make this transaction, he inputs the amount he wants to sell at the current price, and clicks “Deal” button.
5. If user has enough amount of this stock to make this transaction, this transaction is successful. And user receives the message “Transaction Successful!”. Then, go to 7.
6. If user has not enough amount of this stock, or even he is not holding this stock,

user receives the message “Transaction Fails. You are not holding enough stocks!” Then, go to 2.

7. User clicks “Done” button to get out of this transaction page.

Name: ChangePwd
Actors: Player
Priority: Low
Entry Condition:
The user wants to change his or her current password for the current user account.
Exit Condition:
User’s password of the current user account has been changed.
Flow of events:

1. User clicks the “User Account” button on the toolbar menu.
2. User clicks the “Change Password” button in the “User Account” window.
3. User input a new password in the “New Password” textbox.
4. User input the new password in the “Verify the new password” textbox.
5. User clicks the “Commit” button.
6. A window showing “Please re-enter your new password” will show up if the password in the “Verify the new password” textbox does not match the password in the “New Password” textbox.
7. The user’s password is changed to the new password if the password in the “Verify the new password” textbox matches the password in the “New Password” textbox.
8. The current window of the application turns to the default main window.

Special Requirements:
--The password is combined with either character or number.
--The length of the password is neither shorter than 4 numbers nor longer than 16 numbers.

Administrator side:

Name: ModifyMkt
Actors: Administrator
Priority: Medium
Entry Condition:
The admin has logged in.
Exit Condition:
None
Flow of events:

1. User clicks “Modify Markets” button in the administrator site.
2. User can choose of a list of all available markets and “new market” option. If the user chooses market go to step (3). If the user chooses “new market” jump to step (5).
3. System shows all the chosen market’s details with the ability to edit them.
4. User edits all information that needs to be edited. Jump to step (6)
5. User enters all new market information.

6. User clicks “update” button.
7. System adds the new information and refreshes the site.

Name: ModifyStocks
Actors: Administrator
Priority: Medium
Entry Condition:
The admin has logged in.
Exit Condition:
None
Flow of events:
1. User clicks “Modify Stocks” button in the administrator site.
2. User chooses one of a list of all available markets.
3. User can choose of a list of all available stocks and “new stock” option. If the user chooses stock go to step (4). If the user chooses “new market” jump to step (6).
4. System shows all the chosen stock’s details with the ability to edit them.
5. User edits all information that needs to be edited. Jump to step (6)
6. User enters all new stock information.
7. User clicks “update” button.
8. System adds the new information and refreshes the site.

Name: ModifyUserInfo
Actors: Administrator
Priority: Medium
Entry Condition:
The admin has logged.
Exit Condition:
None
Flow of events:
1. User clicks “Modify Users” button in the administrator site.
2. User can choose of a list of all available users accounts and “new user” option. If the user chooses a user account go to step (3). If the user chooses “new user account” jump to step (5).
3. System shows all the chosen user account’s details with the ability to edit them.
4. User edits all information that needs to be edited. Jump to step (6)
5. User enters all new user account information.
6. User clicks “update” button.
7. System adds the new information and refreshes the site.

Name: AdminLogin
Actors: Administrator
Priority: High

Entry Condition:
The administrator has not logged in.
Exit Condition:
The administrator logged in.
Flow of events:
1. User enters admin’s username & password.
2. User clicks “login” button.
3. If the username & password are correct goes to step (4). Otherwise jump to step (5).
4. The system will show the administrator site.
5. The system will show an error message and ask the user to enter the information again.

Special Requirements:
-User can only have three wrong entries; after that, the system should block the user from trying for 15 minutes or more.

Name: AdminLogout
Actors: Administrator
Priority: High
Entry Condition:
The administrator has logged in.
Exit Condition:
The administrator has not logged in.
Flow of events:
1. User clicks “logout” button in the administrator site.
2. The system gets a confirmation from the user to log him/her out. Otherwise jump to step (4).
3. The system goes to the “administrator login” site.
4. The system will go back to the administrator site.

VII. Partial Analysis Model

1. Object Definition:

	Object	Attributes & Responsibilities
Entity Classes	User	
	User Account	
	Stock Exchange	
	Stock	name:String shortName:String code:Integer currency:String marketCapitalization:Float share:Integer currentPrice:Float openPrice:Float historyPrice:Array lowestPricetradingday:Float highestPricetradingday:Float - - - - - - - - - - getInput() update()
	System Administrator	

Boundary Classes	Buy/Sell order form	getInput()
		sendOutput()
		return()

Control Classes	Process a Buy Order	

Table v. Object definition

2. Sequence Diagram:

BuyStock class:

After opening the “View Markets” page in the app, the user will be able to see a “Buy” button. On pressing it the process of buying stocks begins.

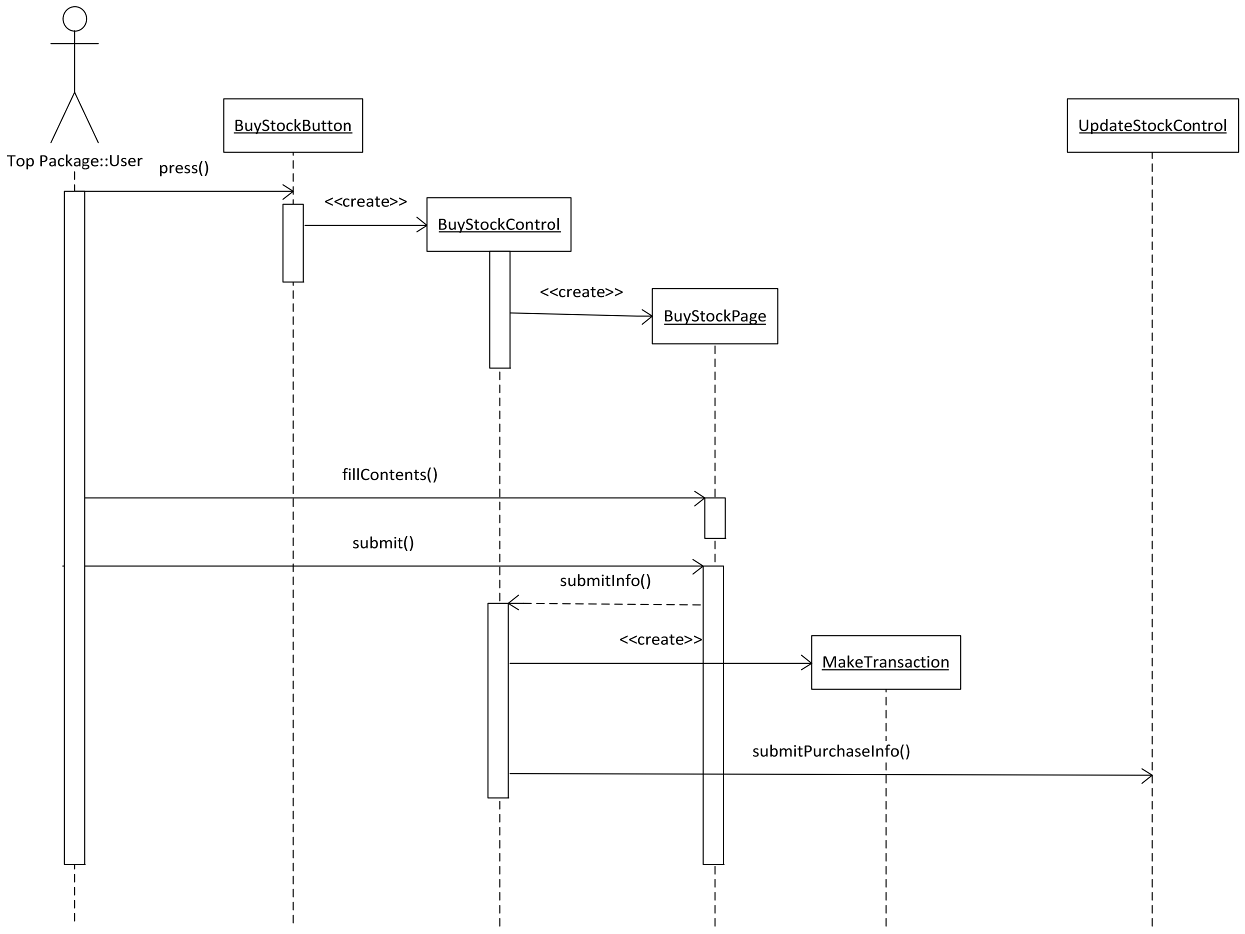


Figure iii. Sequence Diagram

Boundary classes:

- BuyStockButton
- BuyStockPage

Control class:

- BuyStockControl
- UpdateStockControl

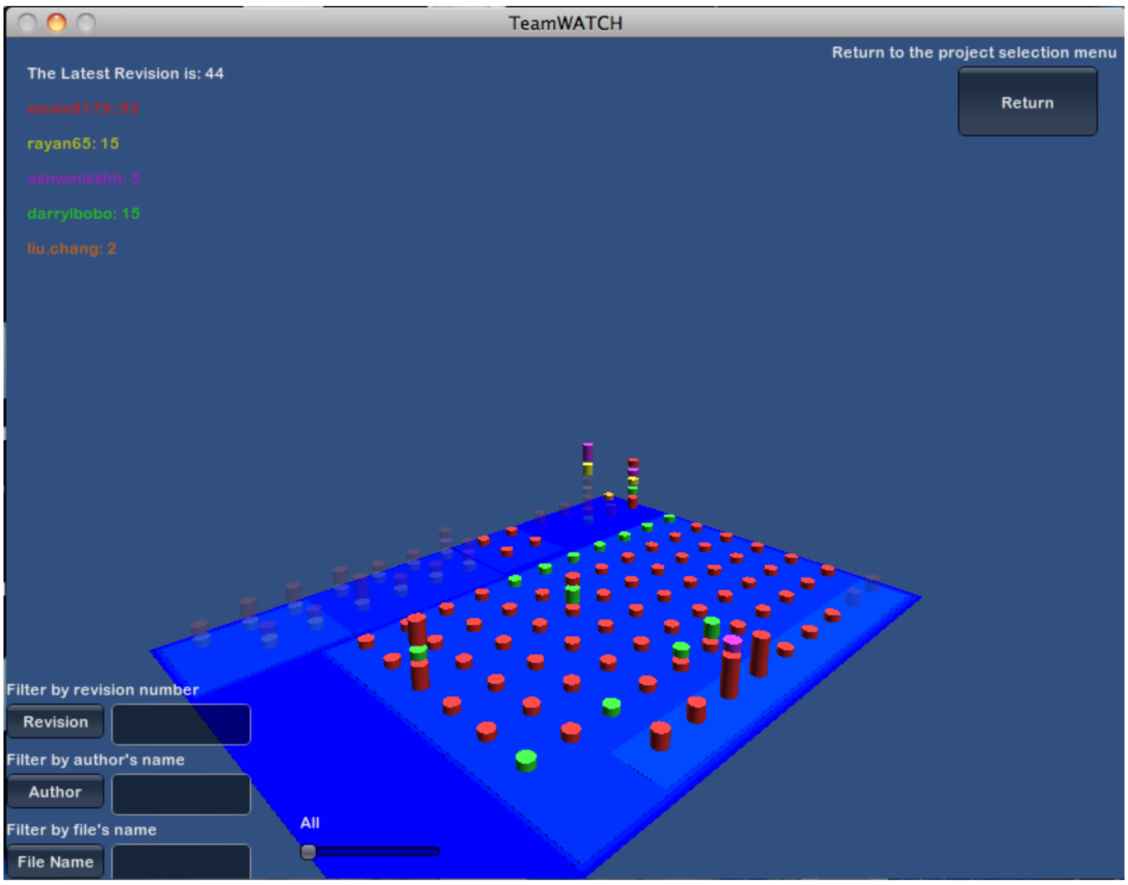
Functions:

- press(): User presses the BuyStock button whenever he wishes to buy stocks.
- fillContents(): User fills in details like number of stocks he wants to buy etc..
- submit(): User submits the details and finalizes his purchase.

VIII. Implementation

--Milestones:

1. Implement the login/logout system, and design user interfaces of “View Stock” use case and “Change Stock Exchange” use case. (Due: 10-19-2011)
   TeamWatch screenshot for Milestone1:
   Status: Done



The Latest Revision is: 44

xinxin8179: 93

rayan65: 15

ashwinikkhh: 5

darrylbobo: 15

liu.chang: 2

Return to the project selection menu

Filter by revision number

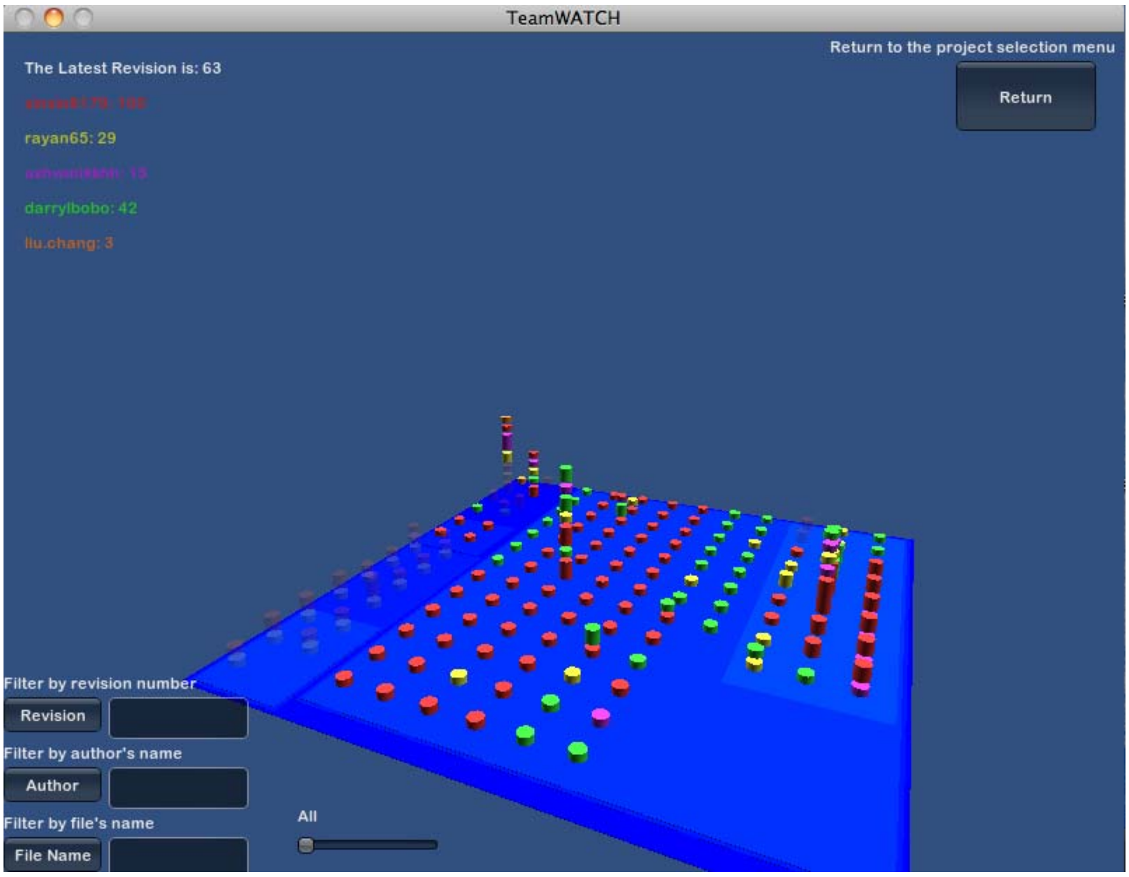
Filter by author's name

Filter by file's name

Screenshot i. Milestone1

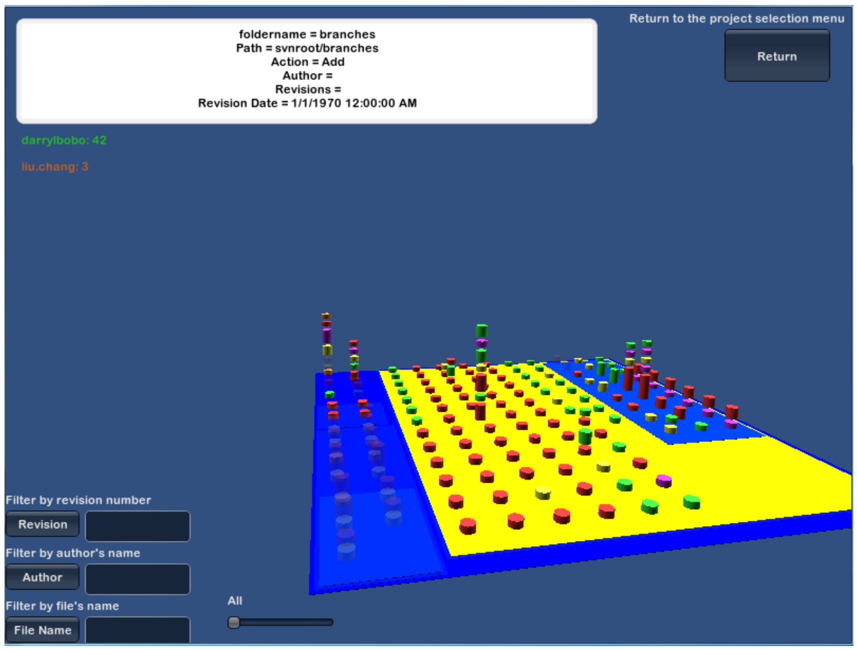
2. Create stock information table, stock exchange information table, user information table on the backend server database, complete the “View Stock” use case and “Change Stock exchange” use case. (Due:10-26-2011)

TeamWatch screenshot for milestone2:-→
Status:Done



Screenshot ii. Milestone2

3. Implement buy stock function and sell stock function. (Due: 11-02-2011)
   Status: Done.
   Screenshot:



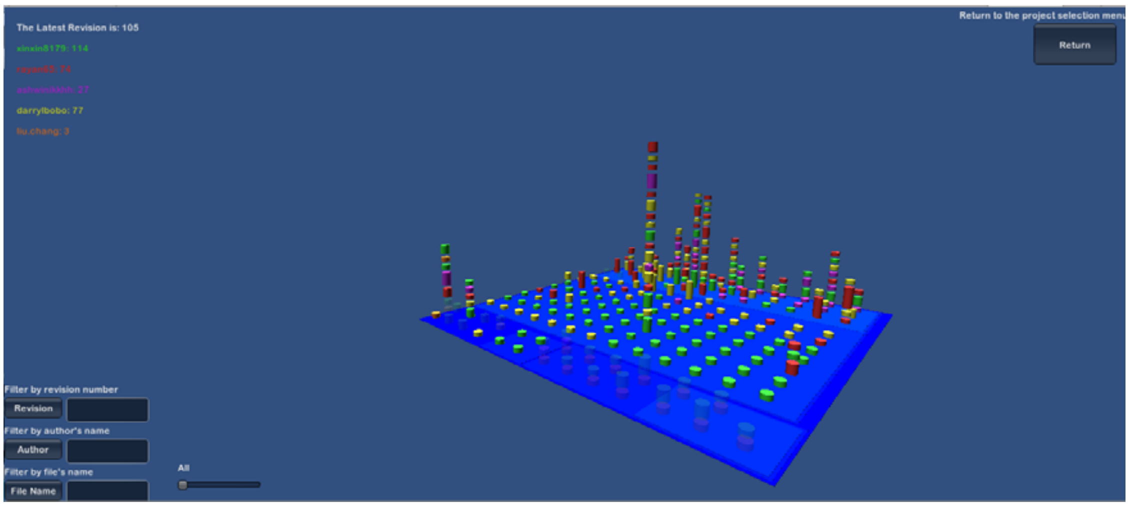
foldername = branches
Path = svnroot/branches
Action = Add
Author =
Revisions =
Revision Date = 1/1/1970 12:00:00 AM

darrylbobo: 42

liu.chang: 3

Screenshot iii. Milestone3

4. Complete the whole game and make a presentation on class. (Due: 11-09-2011)
   Status: Done.
   Screenshot:



The Latest Revision is: 105

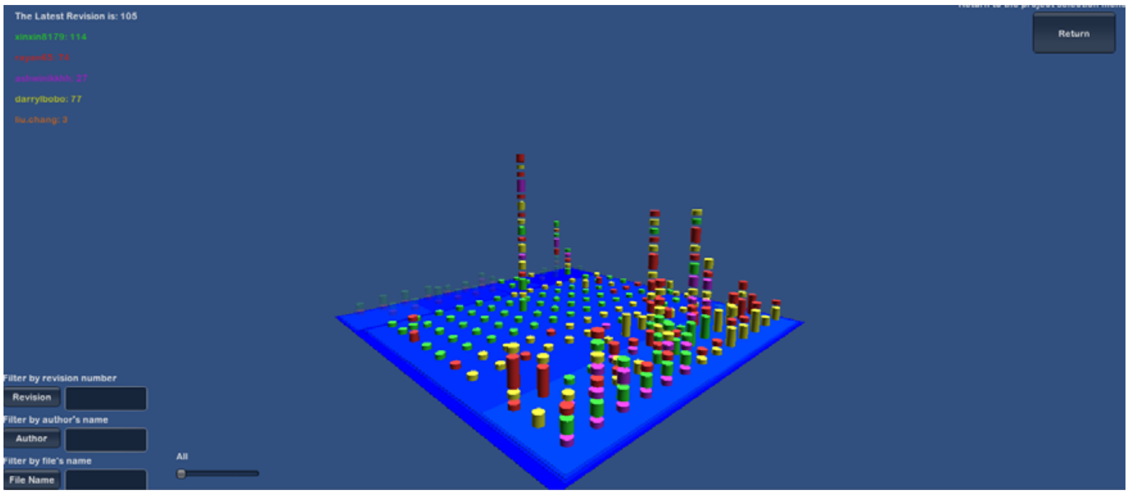
xinxin8179: 114

rayan65: 74

ashwinikkhh: 27

darrylbobo: 77

liu.chang: 3



Screenshot iv. Milestone4

For screenshot, please refer to user's guide.

IX. Deployment Model

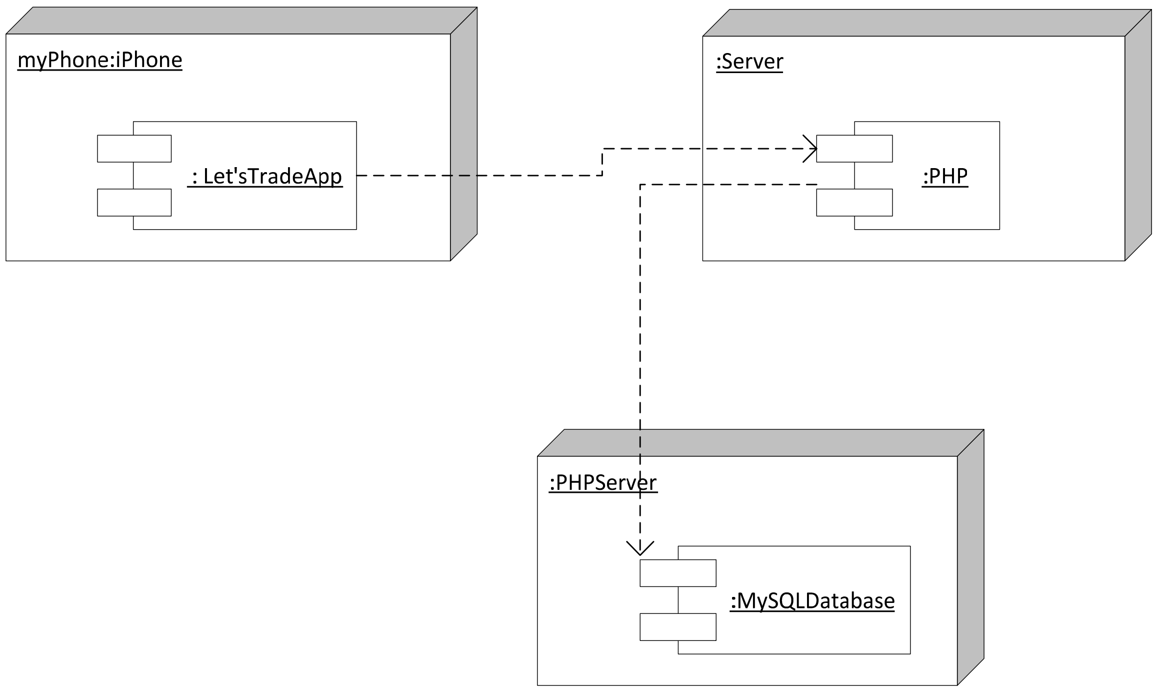


Figure iv. Deployment Model Diagram

--Detailed view of the Server component:

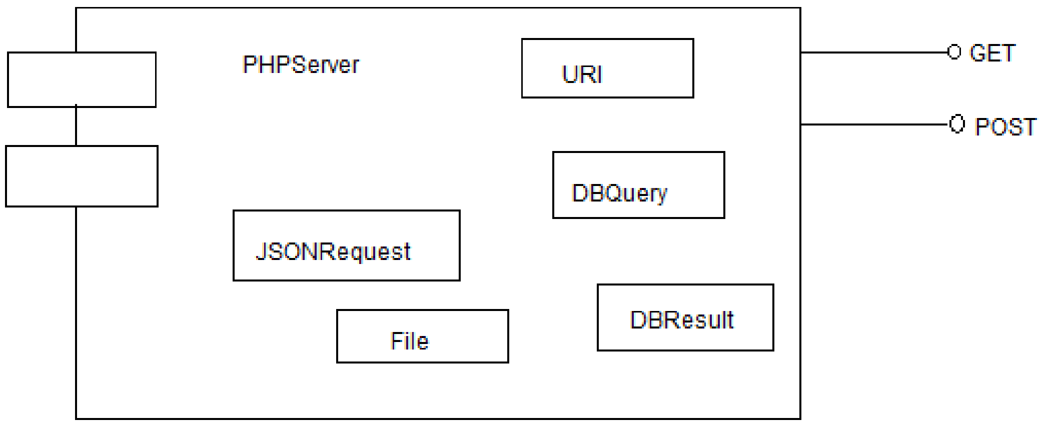


Figure v. Detailed view of Sever in Deployment Model

The user with the device sends a request, which is serviced by the PHP server. This server accesses the Database to provide the user with the required information.

X. Test and Analysis

Function for test

The function for test is the "buy stock" function.

Test coverage criterion

Apply conditional coverage as the test coverage criterion.

Test case and test result

Test Case	Result	Satisfied?
Buy 1000 volumes of a stock "BAC" with a current price which is $5, the user's current balance is $10000.	Return "Success". The current balance is now $5000. A transaction record of buying 1000 "BAC" is shown in the "View Transaction" window.	Yes
Buy 2000 volumes of a stock "BAC" with a current price which is $5, the user's current balance is $10000.	Return "Success". The current balance is now $0. A transaction record of buying 2000 "BAC" is shown in the "View Transaction" window.	Yes
Buy 3000 volumes of a stock "BAC" with a current price which is $5, the user's current balance is $10000.	Return "Success". The current balance is now $0. A transaction record of buying 2000 "BAC" is shown in the "View Transaction" window.	Yes
Buy 0 volume of a stock "BAC" with a current price which is $5, the user's current balance is $10000.	Return "Success". The current balance is now $10000. A transaction record of buying 0 "BAC" is shown in the "View Transaction" window.	Yes

Table vi. Test Cases

Time and Space Analysis:

The time complexity is *O (1) or* constant time.

The space complexity is *O (1)*.

Test Harness

A separate test harness is unnecessary.

Alternative solution

There is no need to make any alternative solution.

Known Issue:

Please refer to: https://dev.5widellc.com/redmine for detailed issues we solved.

XI. Acknowledgements

Post Jobs team would like to thank Dr. Liu for the help during the development of “Let’s Trade”.

We also appreciate the help from Mingy uGu, Qing Zhu for her suggestion after using “Let’s Trade”.

And, thanks very much for Ying Zhong’s help of organizing the server.

Thanks to stackoverflow.com, gen-x-design.com, and allseeing-i.com and the users’ suggestion there.

XII. References

1. https://dev.5widellc.com/redmine

2. http://www.rapidbbs.cn/

3. http://developer.apple.com/

4. http://www.gen-x-design.com

5. http://allseeing-i.com