CS542 Project Progress Report

**<Database Application>**

**---** Stock Market Simulation System

Submitted in partial fulfillment of the Course CS542 - Database Management Systems

Students Involved in this project:

Chengjiao Yang

Jian Qiao

Zheng Yang

**Abstract**

This article mainly introduces our database management system - stock market simulation system progress, describes the function and effect of our application systems, and the detail system design (including database design, programming design, interface design, etc.). The issues encountered and the research materials studied were analyzed and explained in this article, as well as the approach of system evaluation and teamwork. In addition, this article describes the current completeness and the progress of our system developing, the remaining part of the development are well-planned and division of labor in this article.

**Keywords**

Database Management, Project Progress, Stock, Approach, Division

**Contribution**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Responsible | My team has discussed my progress, and I'm satisfied with the amount of effort that everyone has put in so far |
| Chengjiao Yang | Captain, Main program developer, Tester | Push the whole project and organize the group meeting every week; Determine and develop system frame and related document content, chief developer. |  |
| Jian Qiao | Document organizer, Main program developer, Tester | Take charge of composing, organizing and modifying documents; Determine and develop system frame and related document content, chief developer. |  |
| Zheng Yang | Presenter, Database designer, Program Assistant, QA | Write and Present the final PPT; Design of MySQL database and ER chart and related document content; Assistant in related development and documents. |  |

preface

[1. Overview 4](#_Toc404257952)

[1.1 Project purpose 4](#_Toc404257953)

[1.2 Core functionalities 4](#_Toc404257954)

[1.3 Current progress 4](#_Toc404257955)

[2. Background material 5](#_Toc404257956)

[2.1 Tools we choose 5](#_Toc404257957)

[2.2 Design pattern 6](#_Toc404257958)

[3. Our approach 6](#_Toc404257959)

[3.1 General approach 6](#_Toc404257960)

[3.2 Design and its justification 7](#_Toc404257961)

[3.3 Implementation and system detail 11](#_Toc404257962)

[3.4 Approach to solve issues 12](#_Toc404257963)

[3.5 Validation of your approach 13](#_Toc404257964)

[4. Lessons learned 13](#_Toc404257965)

[5. Member contribution 14](#_Toc404257966)

[6. Conclusion 14](#_Toc404257967)

[7. Plan and schedule for remainder of project 15](#_Toc404257968)

[Appendix 1 Database Dictionary 16](#_Toc404257969)

[Appendix 2 SQL for TABLES 17](#_Toc404257970)

[Appendix 3 Test Data 18](#_Toc404257971)

# 1. Overview

## 1.1 Project purpose

The purpose of stock market simulator is that attempts to reproduce or duplicate some or all features of a live stock market on a computer so that a player may practice trading stocks without financial risk. Our system is used for educational purposes to teach potential stock traders and future stock brokers how to trade stocks. It can also be used for entertainment purposes and to engage in fantasy trading competitions.

## 1.2 Core functionalities

The system can update the information of market in real time and the simulation of purchasing and selling of stocks is fulfilled by transaction simulation. And also, every activated user is allow to manage his account. The personal profile contains information like account balance, hold of shares and recent orders. By using this system, users are able to experience real stock market activities and get control of his stock business work easily.

## 1.3 Current progress

After several weeks’ work, we have already finished the construction of basic frame and the final goal of outcome. And most part of programming is finished right now. The total application development and system test will be finished on time, and more debug works should be done to make sure the system goes well.

**Finished**

Database: The design of database is done. By building on MySQL, database is accessible by the system. The handshake of database and the program is all set now, and test result of the program seems perfect to show the successful merging of database and system.

Programming: Factory model frame is accomplished as well as the basic class of application is settled. Now the app is ready to present stock and personal information.

UI: The system, now, can show the user the whole list of companies and stock information in market. The personal profile is also available, such as personal information, hold stocks, total balance, and price of each stock.

**Unfinished**

Database constrains: Since there are a lot of constrains within entities, and some more situations should be considered, like the limit of zero balance purchase, and overflow of buying shares from a company.

Price refresh: The real-time update of stock price and market information are still under construction, after which, the market info and stock price will change automatically.

Function of purchasing: Users is still not allowed to buy or sell his shares, and every order should be recorded by the system.

# 2. Background material

## 2.1 Tools we choose

**MySQL**  
In our project, we deployed MySQL. The advantage of using MySQL:

1) MySQL is an open source database system. Hence it can be downloaded and used by the developer for free.

2) MySQL occupies very less disk space.

3) MySQL can be easily installed in all major operating systems like Microsoft Windows, Linux, Mac OS and every easy to configure.

4) MySQL is best suited for small and medium applications.

**Eclipse**

Also we use eclipse as the IDE to develop our application. Why we use eclipse:

1) Eclipse support windowbuilder pulg-in, a very powerful tool, that makes java graphics programming very easy.

2) Eclipse support egit pulg-in. Our team use Github to share code and synchronize our workspace so that we can easily combine our code together and make it work.

**MySQL Workbench**

We also deployed MySQL Workbench, a visual database design tool that allow us to maintain and change our table and data easily. For us, the biggest advantage of using MySQL Workbench is that we do not need to type SQL on terminal over and over again when we want to change the structure of tables we designed. It makes our development more efficient

**Swing**

Initially, we plan use AWT to develop our project, but right now we decided use Swing since Swing was developed to provide a more sophisticated set of GUI components than AWT. Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT.

## 2.2 Design pattern

**Factory design pattern**

We use factory design pattern to develop our project. Factory design pattern is used to create objects and it provides loose coupling and high cohesion. Factory pattern encapsulate object creation logic which makes it easy to change it later when you change how object gets created or you can even introduce new object with just change in one class.  
Benefit of using Factory design patternin Java is that it encourages consistency of code since every time object is created using Factory rather than using different constructor at different client side.

Code written using Factory design pattern in Java is also easy to debug because you have a centralized method for object creation and every client is getting object from same place.

In our project we build a class called DAOfactory. It contains a lot object-get functions, these function will returns objects, which used to store data that we select from MySQL such as stock information to client's side. Therefore, client's side just need to create a DAOfactory objects to get every objects it need.

# 3. Our approach

## 3.1 General approach

We are a small team with only three team members, and the application is also java-based lightweight development, plus the relatively short period of developing time, we should take a relatively simple and efficient approach to complete the project. Therefore, we adopted agile development approach which is suitable for our team situation to carry out project development and collaboration.

As we know, the agile development has several features:

1) Individual and interactive is higher than processes and tools;

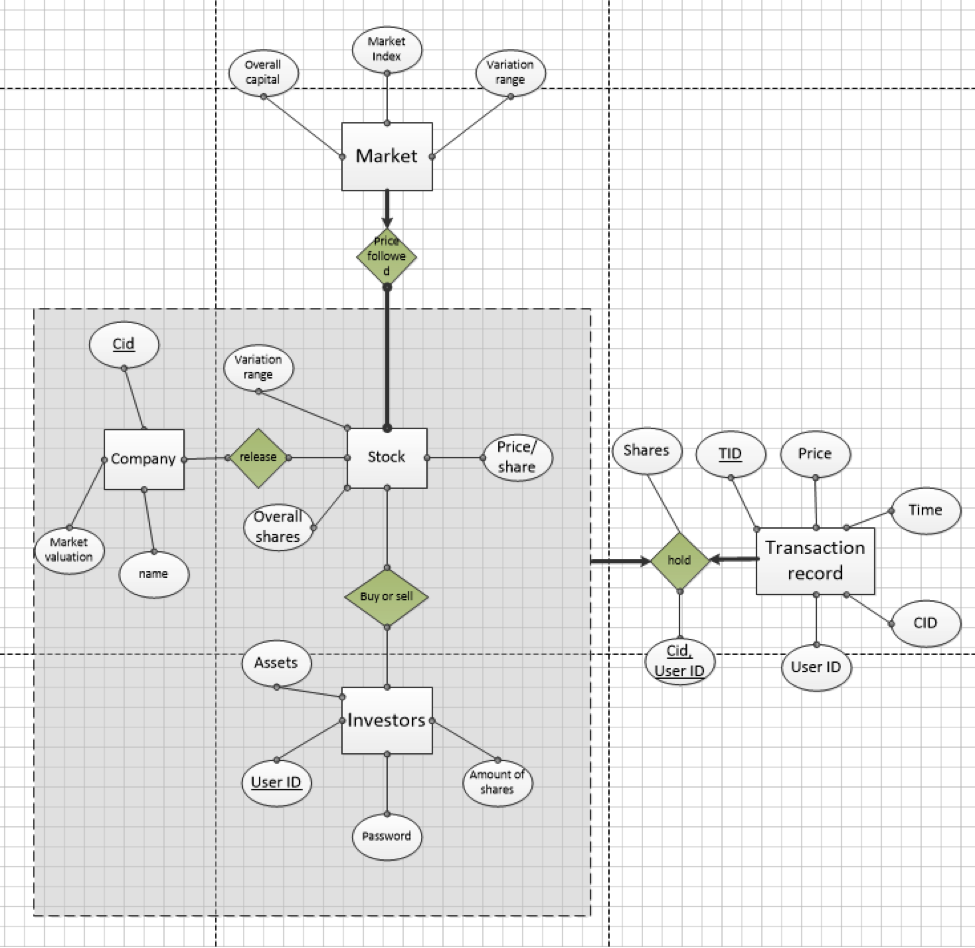
2) Runnable software is higher than exhaustive documentation;

3) Customer collaboration is higher than contract negotiation;

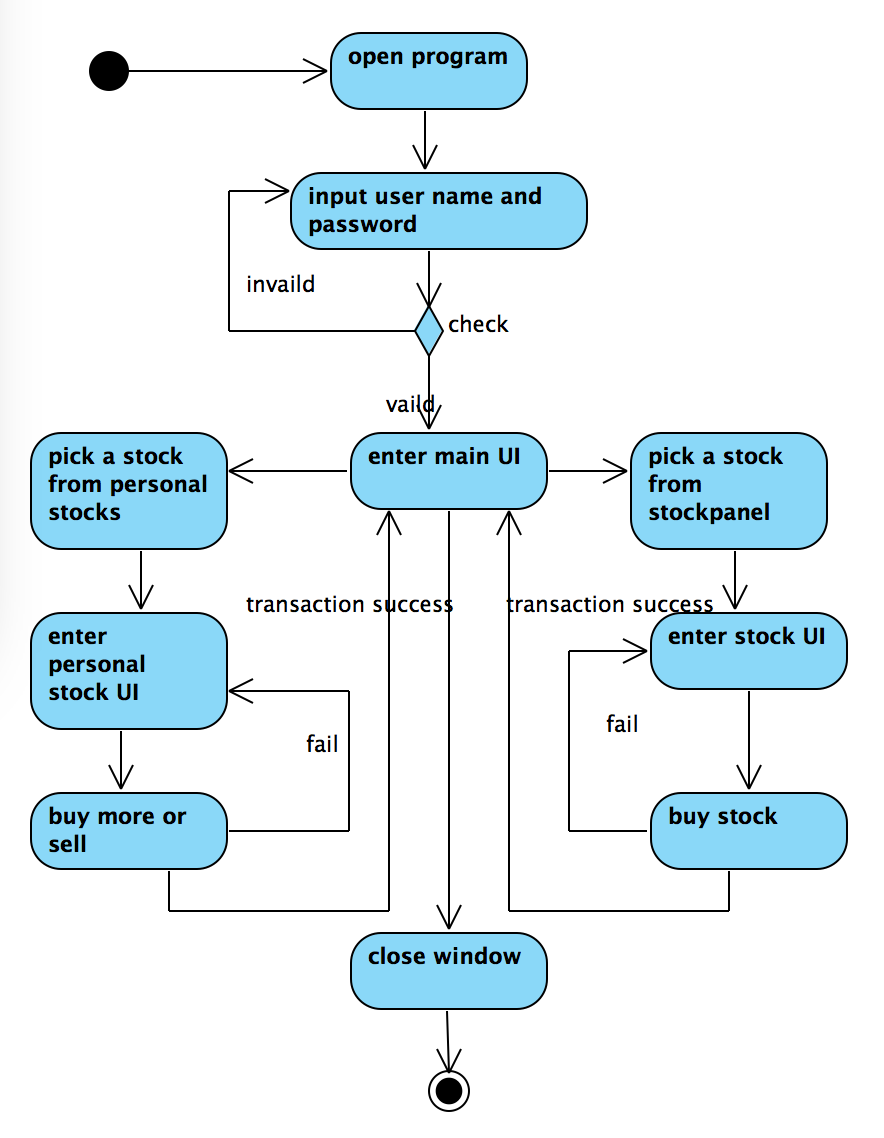
4) Respond to changes in higher than following a plan;

So apart from the usual communication, we have routine meetings every Monday to discuss the current situation and the difficulties encountered in progress, through discussion, let each member fully understand the system requirements and functionalities. On application development aspect, we analyzed and listed the main function point, priority to complete the main functions and user-interfaces, and submit the runnable program code in a short period, then gradually develop incremental functionality based on the main function. Agile development advocate simplicity, on the premise of team member understand, we use the simple graph and face-to-face discussion to design the system architecture, such as UI design, it is roughly drew on the paper to show the expected results. Because simplicity and incremental development, even the time conflict or requirements change, our team can adjust and continue the progress quickly.

## 3.2 Design and its justification

3.2.1 Database design  
 

3.2.2 Activity diagram  
The activity diagram shows the workflow when a user use our program.

  
  
3.2.3 UML-Class Diagram

We are using MVC and Factory design pattern, it is a Muti-Level class structure. We list part of the class as example to generate this UML. The basic structure shows below:

**Front-end**

Entry Layer: ui.view

User Interface Layer: ui.login\_frame; ui.TotalInfo; ui.User\_Frame; ui.controller

Logical Processing Layer: ui.stockPanel, ui.UserInformation

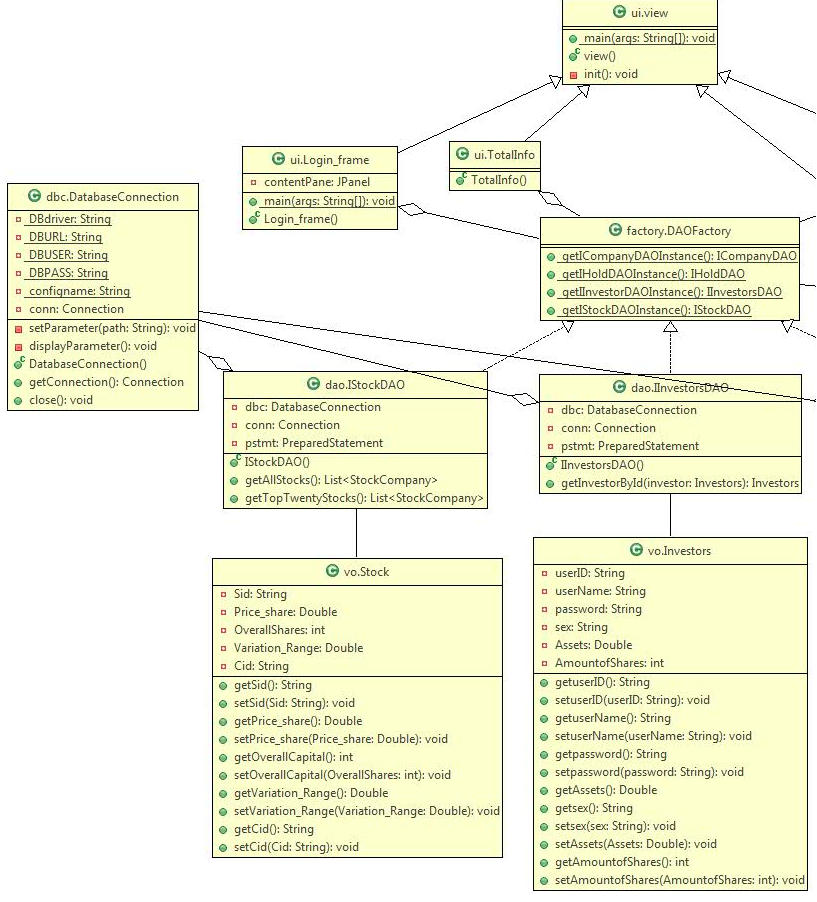
**Back-end**

Object Factory Layer: factory.DAOFactory

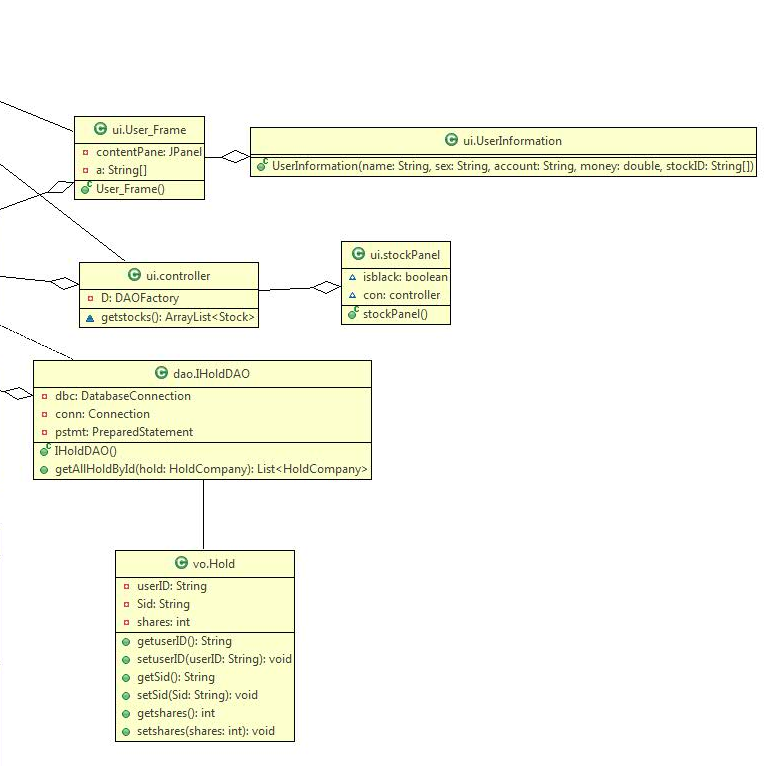
JDBC Interface Layer: dbc.DatabaseConnection

Application Logical Layer: dao.IStockDAO; dao.IInvestorsDAO; dao.IHoldDAO

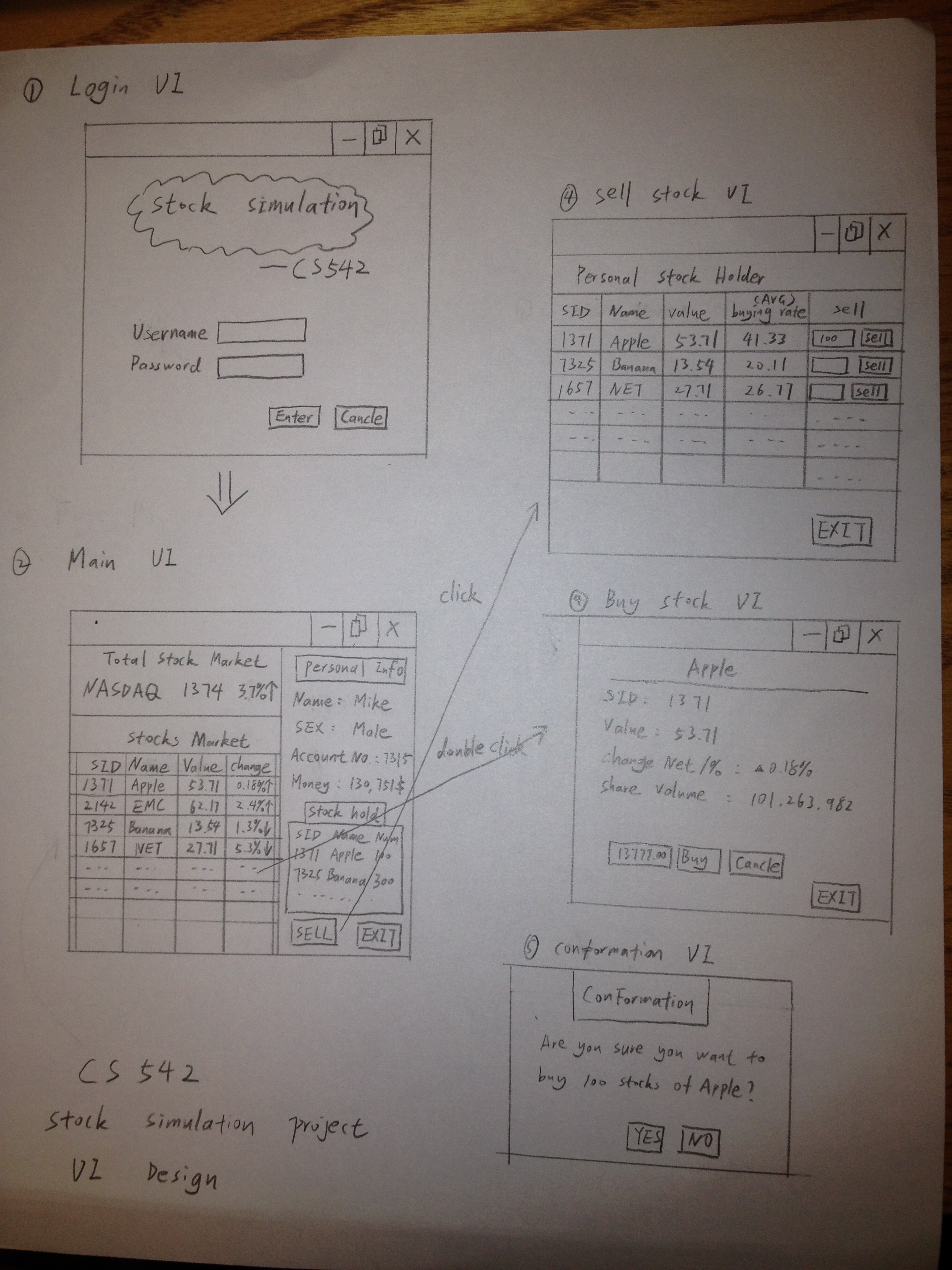
Data Model Layer: vo.Stock; vo.Investors; vo.Hold



Part 1 UML-left part



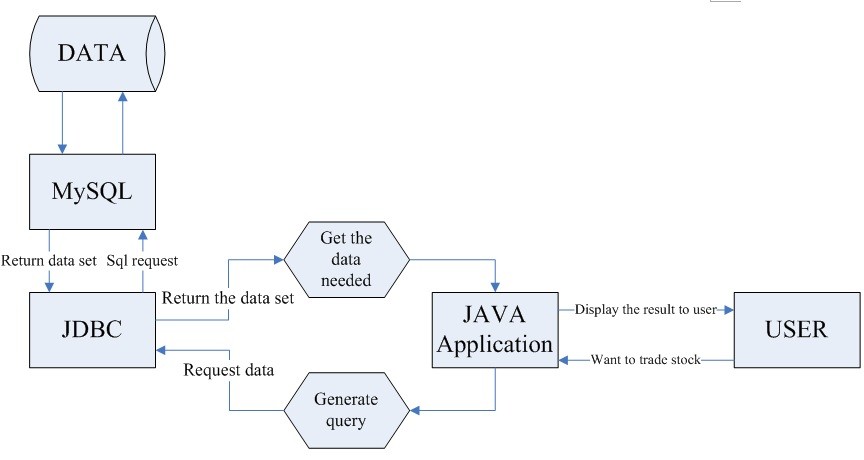
Part 2 UML-right part

3.2.4 UI design  


## 3.3 Implementation and system detail

**Architecture:**

We use java-based desktop application architecture, application connect to the MySQL database via JDBC. At the UI part, we use swing-based plugin WindowsBuilder for visualization development. The general system architecture is shown as below:



**Tools:**

We use lots of develop tools and assist tools such as:

Develop and DB tools: MySQL; Eclipse; Java1.7

Management tools: Github; Word online; MySQL Workbench 6.2 CE; phpAdmin

Plugin: Egit; Windowsbuilder; Swing; AmaterasUML; Fatjar

Library: forms-1.3.0; mysql-connector-java-5.1.29-bin

Those tools help us to complete the development task much faster and more efficient.

## 3.4 Approach to solve issues

Since we are a team, the project work is distributed to team members. Therefore, the cooperation and communication are very important for us. We schedule our meeting every week to make sure good teamwork, even we have one member living in Boston. We are still to meet each other by online chat.

We also have some technical problems when we using github. The synchronization of code, script and documents crash down when we are trying to update at same time. After several research online solutions, we solve the problem. To share ideas more efficiently, we apply word online for project building.

We select agile development as our method of programming, for its big advantage of small team project building. In this type, every team member plays very important role and performance of every single member determine the process of whole project.

## 3.5 Validation of your approach

This is our first time developing a java GUI program, and we have no experience about stocks. Therefore this is a very changeling project for us. Before implementing, we watched some tutorials and read some references to understand our problems and find ways to solve them.

For now, half of our project is working. Our program can read data (stock information and user information) from database and store them into a specified class. Also it can present these data to users. For example, user can see all stocks our database has and the stocks he/she holds. Every parts we have implemented works fine now.

The part we have not implemented is not working. For example the sell and buy features are not supported by our program currently, however we are working on this part. Also, we have not implemented the login feature yet, but this is the easiest part, we plan implement it at last stage.

We divided our project into several subproject. Before integrating these subproject, we conduct tests on each of them. For example for the Factory class, it is responsible for return data from database, but it was not connected database. Therefore, we used a Junit (a class simulate database) to test if the Factory can return our correct data.

After integrating our database, DAOFactory and UI, we have inserted 20 stocks and 5 valid user names into our database to test if our program can run correctly. It works fine now, our program can show us the stocks information and user information we expected on UI.

# 4. Lessons learned

**Programing and team work experience**

Some of us have experience about database-related project before, but it is our first time to make the desktop type program. Although the most popular desktop applications language is C++, use Java language to develop such application is also very interesting and a challenge to us, which gives us a new view and understanding of the development and system structure. Meanwhile, import the agile development approach into our group is also a big attempt and challenge. The project has not been completed yet, but progress is going very well. We will refer the experience of this project, using the same method for our future project.

**Design and tools**

We learned a lot from this project. In this project, we are trying to use new tools to solve our problem. In UI design, we apply Windowsbuilder, which gives us more convenient way to put on tags and buttons as we want according to our plan. We also use MVC for implementing user interface. Factory pattern is what we use and the good extending ability gives us more freedom to compliment whole application. Beside the technology, we do learned a lot about stock market. To make sure the system and database perfectly match the need of application user, the interface, database, even the color we use are all same to the common used stock business programs. That's what always happen when doing database design, since every database is based on some special topics and fields, there are lots of things we need to learn outside the developing skills.

# 5. Member contribution

|  |  |
| --- | --- |
| Task | Finish-by |
| System architecture | Whole team |
| Application architecture | Whole team |
| Database architecture | Whole team |
| UI design | Whole team |
| Database detail design | Zheng Yang |
| Database dictionary | Zheng Yang |
| Database create statement | Zheng Yang |
| Test Data | Zheng Yang, Chengjiao Yang |
| VO Class | Zheng Yang |
| Application logic Class | Chengjiao Yang |
| JDBC connection Class | Chengjiao Yang |
| Configuration file | Chengjiao Yang |
| Main UI Frame Class | Jian Qiao |
| Personal info UI and Logic Class | Jian Qiao |
| Stock info UI and Logic Class | Jian Qiao |
| Global market UI and Logic Class | Jian Qiao |
| Personal hold UI and Logic Class | Jian Qiao |
| Unit test | Jian Qiao, Chengjiao Yang |
| Integration test | Zheng Yang |
| Code management | Jian Qiao, Chengjiao Yang |
| Project management | Chengjiao Yang |
| Document integration correction | Jian Qiao |
| Project Progress Report | Whole team |
| Project Proposal Report | Whole team |
| Meeting organizer | Chengjiao Yang |

# 6. Conclusion

Up to now, our project goes well along the plan. Every team member is very strictly obey the determined schedule and do their own job perfectly. We have finished about 1/2 of the whole work and rest of them are planned to be done on time. Since we have good management of the process, the project steps are accomplished very well. Our next step is to finish some special constrains of entities and complete the rest program functions, and test the whole system. There is no doubt that our project will perform well as what we expected in the plan.

# 7. Plan and schedule for remainder of project

We plan to conduct evaluations on following aspects: Understandability, Learnability and Integrity. At Integrity aspect, we will run our program by ourselves to test if our program can provide the features we designed and how often it crash when we run with inappropriate actions. And try to find out some bugs. About understandability and Learnability, we plan to invite some classmates participate this evaluation. First, we will introduce our program to them to check whether they can understand what propose of our project or not. Next, we will invite them to operate our program with guild to check whether they are able to learn how to use its functions easily.

The following sheet presents our schedule for the remainder of project:

|  |  |
| --- | --- |
| WEEK | SCHEDULE |
| Week5 (11/14-11/20) | * Integrate the program framework and database * Program the application logic * Complete the project-progress report * Unit test |
| Week6 (11/21-11/27) | * Implement the login and update features then integrate it to our program * Perfect the database * Unit test |
| Week7 (11/28-12/04) | * Implement the sell and buy features and integrate it to our program * Perfect the UI * If we have more time we can implement some additional features * Unit test |
| Week8 (12/05-12/11) | * Complete the application * Integrity, Understandability and Learnability test |
| Week9 (12/11-12/15) | * Complete the project final report * Write PPT and prepare the presentation |

# Appendix 1 Database Dictionary

|  |  |  |
| --- | --- | --- |
| market |  |  |
| marketIndex | varchar | marketIndex represent the index determines trend of market price; |
| VariationRange | double | VariationRange represent the range of change in price of whole market; |
| OverallCapital | double | OverallCapital represent the whole capital a market hold in side; |
| stock |  |  |
| Price\_share real | double | Price/share represent stock value per share; |
| OverallCapital | Integer | OverallCapital represent total number of shares the company has; |
| Variation\_Range | double | Variation\_Range represent range of change in price of the stock; |
| company |  |  |
| Cid | varchar | represent unique identify code of a company; |
| MarketValuation | varchar | represent the value of a company in the market; |
| Name | varchar | company name |
| investors |  |  |
| userID | varchar | userID represent unique identify code of an investor; |
| password | varchar | password is the login permission code; |
| Assets | double | Assets represent total amount of money of the investor's account; |
| AmountofShares | integer | AmountofShares represent total amount of shares in the investor's account; |
| hold |  |  |
| userID | varchar | userID represent unique identify code of an investor make this order; |
| Cid | varchar | Cid represent unique identify code of a company whose stock in this order; |
| shares | integer | shares represent the amount of shares transacted in this order; |
| transactionrecord |  |  |
| TID | varchar | TID represent the unique code of a transaction; |
| userID | varchar | userID represent unique identify code of an investor make this transaction; |
| Cid | varchar | Cid represent unique identify code of a company whose stock in this transaction; |
| time | date | time represent when transaction is made; |
| price | double | price represent how much money is transacted in this transaction; |

# Appendix 2 SQL for TABLES

CREATE TABLE `company` (

`Cid` varchar(50) NOT NULL,

`MarketValuation` double DEFAULT NULL,

`Name` varchar(50) NOT NULL,

PRIMARY KEY (`Cid`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `hold` (

`userID` varchar(50) NOT NULL,

`Sid` varchar(50) NOT NULL,

`shares` int(11) DEFAULT NULL,

PRIMARY KEY (`userID`,`Sid`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `investors` (

`userID` varchar(50) NOT NULL,

`password` varchar(50) NOT NULL,

`Assets` double NOT NULL,

`AmountofShares` int(11) DEFAULT NULL,

`sex` varchar(10) DEFAULT NULL,

`userName` varchar(50) DEFAULT NULL,

PRIMARY KEY (`userID`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `market` (

`marketIndex` char(50) DEFAULT NULL,

`VariationRange` double DEFAULT NULL,

`OverallCapital` double DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `stock` (

`Sid` varchar(50) NOT NULL,

`Price\_share` double DEFAULT NULL,

`OverallShares` int(11) DEFAULT NULL,

`Variation\_Range` double DEFAULT NULL,

`Cid` varchar(50) NOT NULL,

PRIMARY KEY (`Sid`),

UNIQUE KEY `Cid\_UNIQUE` (`Cid`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

# Appendix 3 Test Data

INSERT INTO `stock` (`Sid`,`Price\_share`,`OverallShares`,`Variation\_Range`,`Cid`) VALUES ('000001',15,2000000,0.4,'1');

INSERT INTO `stock` (`Sid`,`Price\_share`,`OverallShares`,`Variation\_Range`,`Cid`) VALUES ('000002',20,6666666,0.7,'2');

INSERT INTO `stock` (`Sid`,`Price\_share`,`OverallShares`,`Variation\_Range`,`Cid`) VALUES ('000003',33,454353,0.9,'3');

INSERT INTO `stock` (`Sid`,`Price\_share`,`OverallShares`,`Variation\_Range`,`Cid`) VALUES ('000004',6,34634234,0.2,'4');

INSERT INTO `stock` (`Sid`,`Price\_share`,`OverallShares`,`Variation\_Range`,`Cid`) VALUES ('000005',35,6345,1.1,'5');

INSERT INTO `stock` (`Sid`,`Price\_share`,`OverallShares`,`Variation\_Range`,`Cid`) VALUES ('000006',44,646345,0.4,'6');

INSERT INTO `investors` (`userID`,`password`,`Assets`,`AmountofShares`,`sex`,`userName`) VALUES ('1','111',5000,2000,'male','zhang san');

INSERT INTO `investors` (`userID`,`password`,`Assets`,`AmountofShares`,`sex`,`userName`) VALUES ('2','111',6666,3000,'female','lucy');

INSERT INTO `investors` (`userID`,`password`,`Assets`,`AmountofShares`,`sex`,`userName`) VALUES ('3','111',999999,3333,'male','Mike');

INSERT INTO `investors` (`userID`,`password`,`Assets`,`AmountofShares`,`sex`,`userName`) VALUES ('4','111',34534,4444,'male','Ted');

INSERT INTO `investors` (`userID`,`password`,`Assets`,`AmountofShares`,`sex`,`userName`) VALUES ('5','111',45234,5555,'male','Ralph');

INSERT INTO `investors` (`userID`,`password`,`Assets`,`AmountofShares`,`sex`,`userName`) VALUES ('6','111',-45435,1234,'female','Cindy');

INSERT INTO `hold` (`userID`,`Sid`,`shares`) VALUES ('1','000001',100);

INSERT INTO `hold` (`userID`,`Sid`,`shares`) VALUES ('1','000002',200);

INSERT INTO `hold` (`userID`,`Sid`,`shares`) VALUES ('1','000003',111);

INSERT INTO `hold` (`userID`,`Sid`,`shares`) VALUES ('2','000001',284);

INSERT INTO `hold` (`userID`,`Sid`,`shares`) VALUES ('2','000002',42);

INSERT INTO `hold` (`userID`,`Sid`,`shares`) VALUES ('3','000002',2342);

INSERT INTO `hold` (`userID`,`Sid`,`shares`) VALUES ('3','000003',235);

INSERT INTO `hold` (`userID`,`Sid`,`shares`) VALUES ('4','000001',235);

INSERT INTO `hold` (`userID`,`Sid`,`shares`) VALUES ('4','000003',551);

INSERT INTO `company` (`Cid`,`MarketValuation`,`Name`) VALUES ('1',0.5,'Apple');

INSERT INTO `company` (`Cid`,`MarketValuation`,`Name`) VALUES ('2',0.2,'Banana');

INSERT INTO `company` (`Cid`,`MarketValuation`,`Name`) VALUES ('3',0.5,'Microsoft');

INSERT INTO `company` (`Cid`,`MarketValuation`,`Name`) VALUES ('4',0.3,'Sun');

INSERT INTO `company` (`Cid`,`MarketValuation`,`Name`) VALUES ('5',0.4,'Google');

INSERT INTO `company` (`Cid`,`MarketValuation`,`Name`) VALUES ('6',0.5,'EMC');

INSERT INTO `company` (`Cid`,`MarketValuation`,`Name`) VALUES ('7',0.8,'Yahoo');

INSERT INTO `company` (`Cid`,`MarketValuation`,`Name`) VALUES ('8',1.1,'Facebook');