CS542 Final 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

Yang Zheng

Abstract:

This article mainly introduces our database management system - stock market simulation system completion status, include the function and effect of our application systems, and the detail system design (including database design, programming design, interface design, etc.). All the issues encountered were fixed and the research materials studied were analyzed and explained in this article, as well as the approach of system evaluation and teamwork, the detail division of labor is clearly described in this article. In addition, this article describes the experience we have learned during the development and the future work we are planning to accomplish.

Keywords:

Stock Market, Approach, Division, DBMS, Web Services, REST, Agile Development

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 Potential user group

Since we have already update our system into online version, which enable people all around the world to use it. The only thing users need to do is to download a user application and connect to our database server and they can use the system and train themselves as they wished. Therefore, the potential user group of our system will be local and foreign learners who want to practice on stock purchasing. And only thing they need to do is to connect to the Internet and download the user application and press some buttons. The user can be individual or groups like financial college and stock analysis institutes.

2. Background material

2.1 DBMS tools

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.

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.

2.2 Develop tools

Eclipse

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

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

2) Eclipse support EGITplug-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.

Swing

Initially, we plan use AWT to develop our project, but we used Swing as the tool to develop our project 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.

Web Service

In our project, we develop the Restful Web Service which support the JSON format HTTP API.The term Web services describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDIopenstandards over an Internet protocolbackbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available. Used primarily as a means for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other's IT systems behind the firewall.

Unlike traditional client/server models, such as a Web server/Web page system, Web services do not provide the user with a GUI. Web services instead share business logic, data and processes through a programmatic interface across a network. The applications interface, not the users. Developers can then add the Web service to a GUI (such as a Web page or an executable program) to offer specific functionality to users.

Web services allow different applications from different sources to communicate with each other without time-consuming custom coding, and because all communication is in XML, Web

services are not tied to any one operating system or programming language. For example, Java can talk with Perl, Windows applications can talk with UNIX applications.

REST

REST stands for Representational State Transfer. (It is sometimes spelled "ReST".) It relies on a stateless, client-server, cacheable communications protocol -- and in virtually all cases, the HTTP protocol is used.

REST is an architecture style for designing networked applications. The idea is that, rather than using complex mechanisms such as CORBA, RPC or SOAP to connect between machines, simple HTTP is used to make calls between machines.

Restful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations.

REST is not a "standard". There will never be a W3C recommendation for REST, for example. And while there are REST programming frameworks, working with REST is so simple that you can often "roll your own" with standard library features in languages like Perl, Java, or C#.

JERSEY

In order to simplify development of RESTful Web services and their clients in Java, a standard and portable JAX-RS API has been designed. Jersey RESTful Web Services framework is open source, production quality, and framework for developing RESTful Web Services in Java that provides support for JAX-RS APIs and serves as a JAX-RS (JSR 311 & JSR 339) Reference Implementation.

Jersey framework is more than the JAX-RS Reference Implementation. Jersey provides its own API that extend the JAX-RS toolkit with additional features and utilities to further simplify RESTful service and client development. Jersey also exposes numerous extension SPIs so that developers may extend Jersey to best suit their needs.

CORS

In our project, we use CORS technology to allow the remote http access for the html client.

Cross-Origin Resource Sharing (CORS) is a W3C spec that allows cross-domain communication from the browser. By building on top of the XMLHttpRequest object, CORS allows developers to work with the same idioms as same-domain requests.

JSON

To minimum the data transmission along the network, reduce the pressure on the server side, we use the JSON format for the REST API.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language,Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

2.3 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 pattern in 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 server's 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. Your approach

3.1 General approach

Considering we are a small team with only three team members, and the project is a lightweight application(initially it's a java-based application, now it include web service, java desktop client, html client etc.), 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.

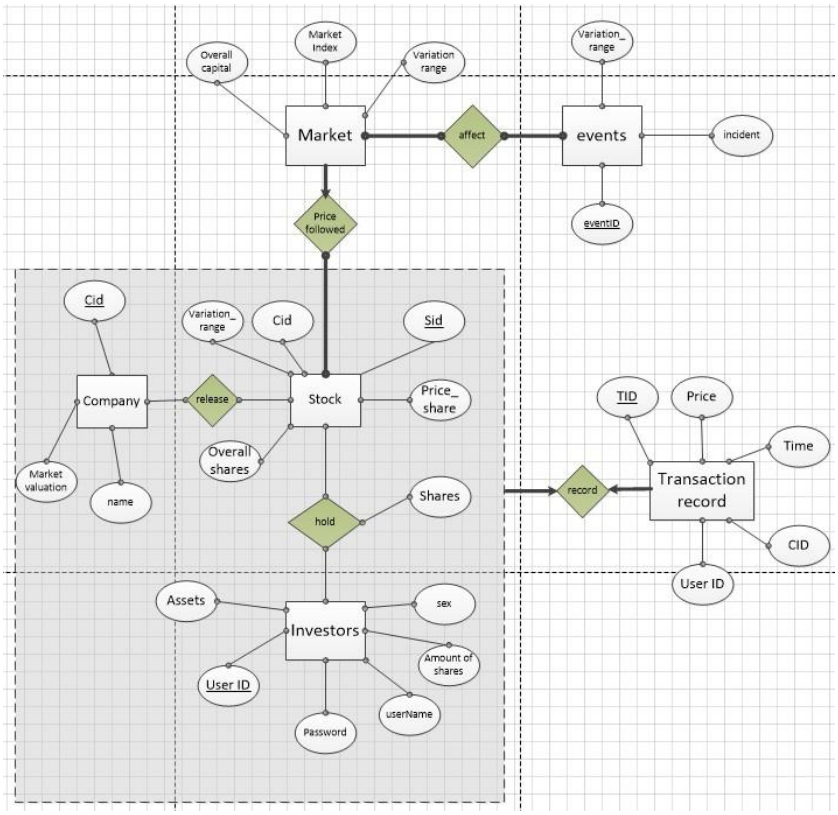
As we know, the agile development has several features:
1) Individual and interactive is higher than processes and tools

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 (our application has upgraded from the local java application to the Web Service architecture application). 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 (each teammate take different class, has different schedule), our team can adjust and continue the progress quickly.

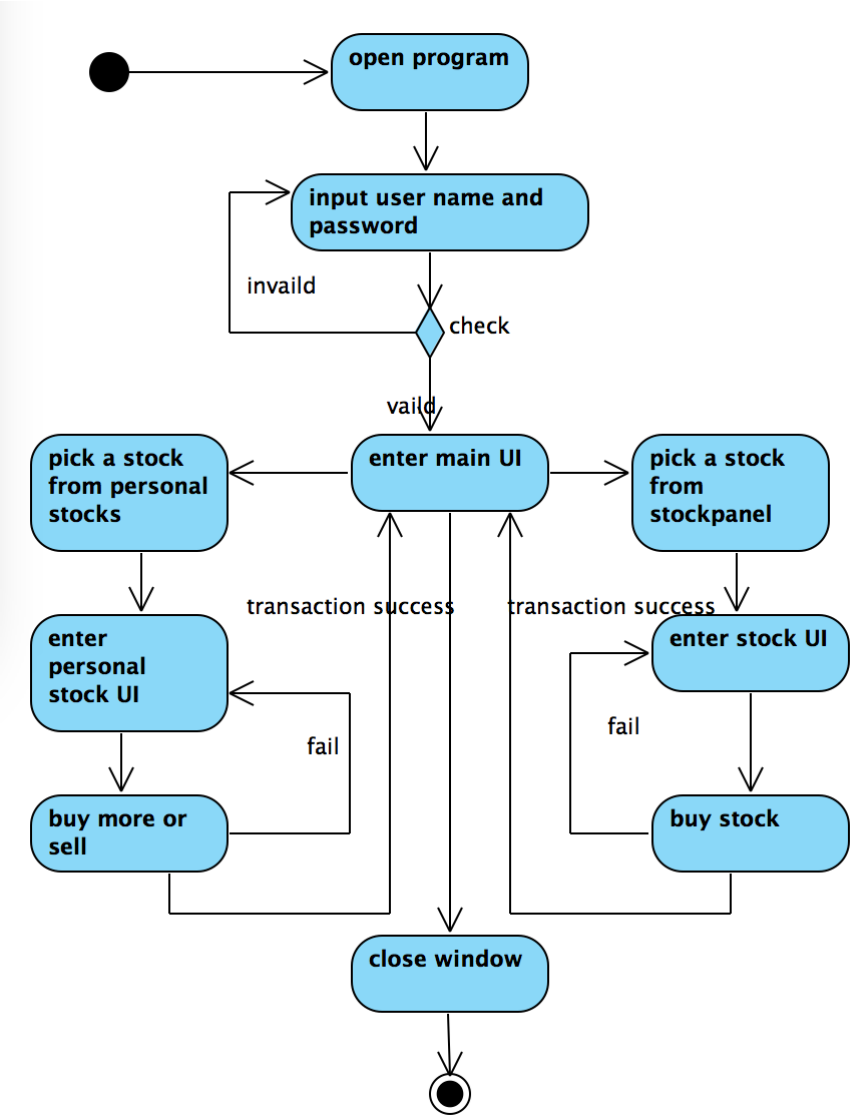
3.2 Design and its justification

3.2.1 Database design



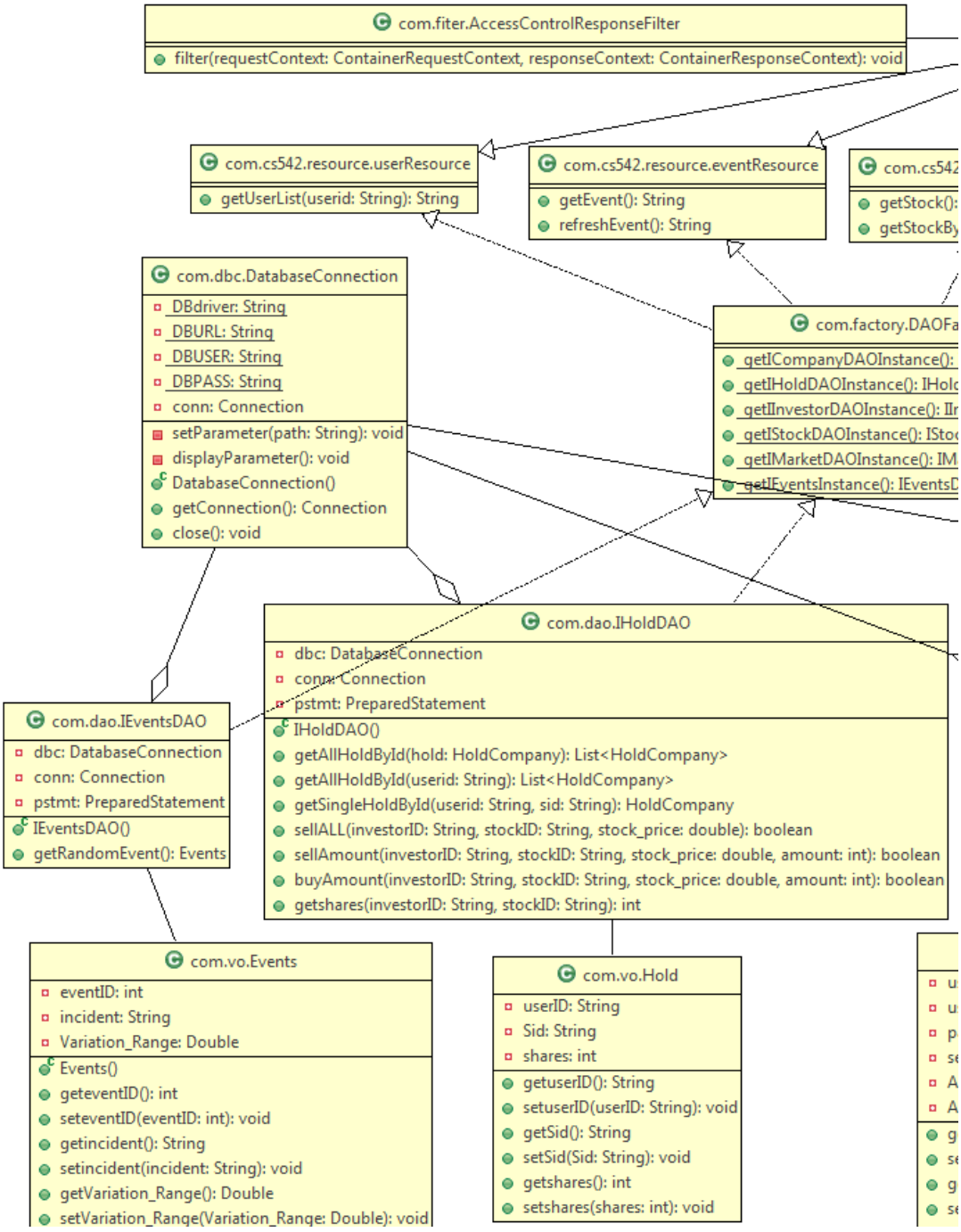
3.2.2 Activity diagram

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

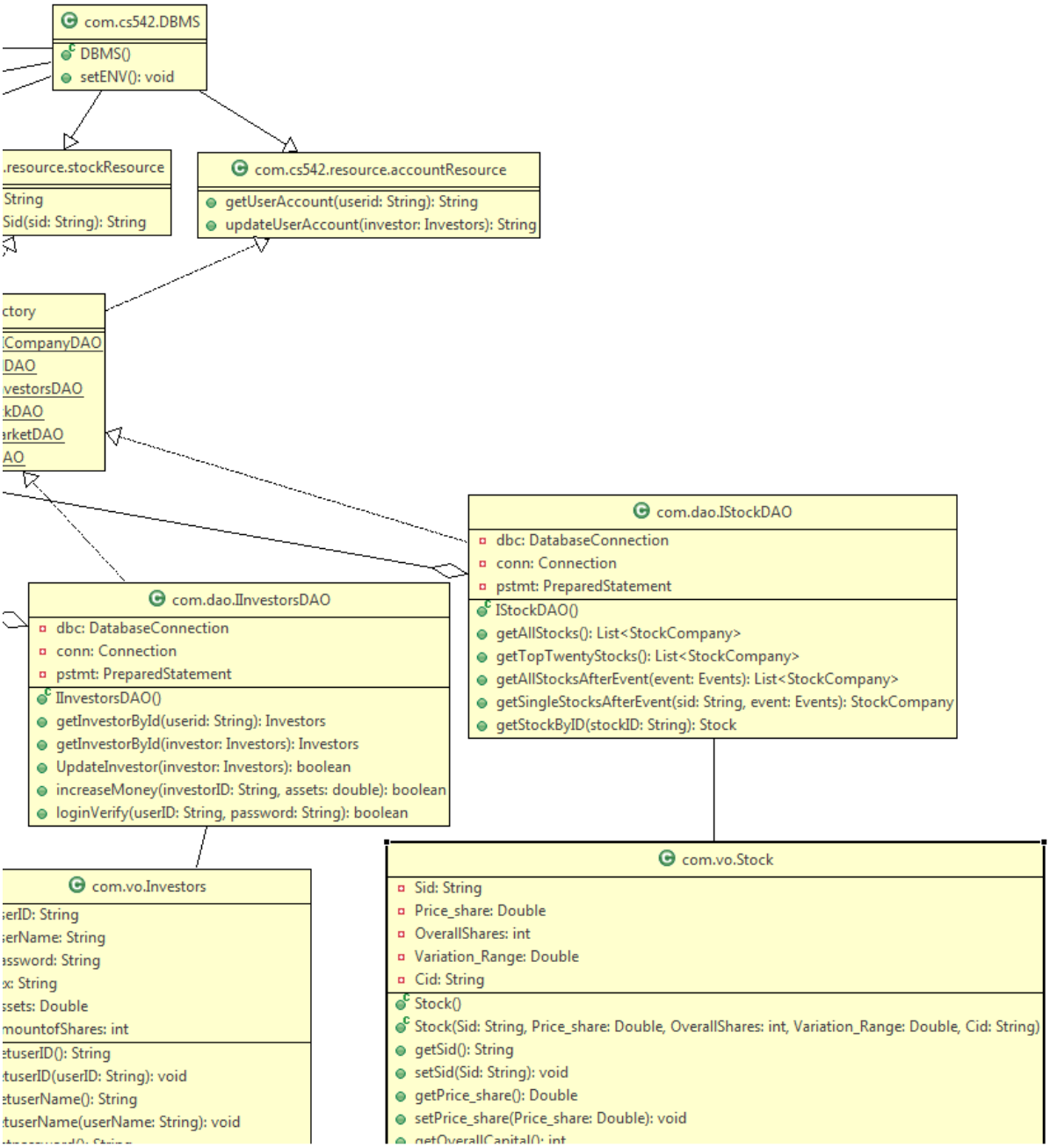


3.2.3 UML-Class Diagram

Our application is a REST Web Service architecture application, thus, it include the service side and the client side. The server side mainly accept the request, query from the database, handle the work flow and support the JSON API. We list part of the class as example to generate this UML. The basic structure shows below:



Server UML left part

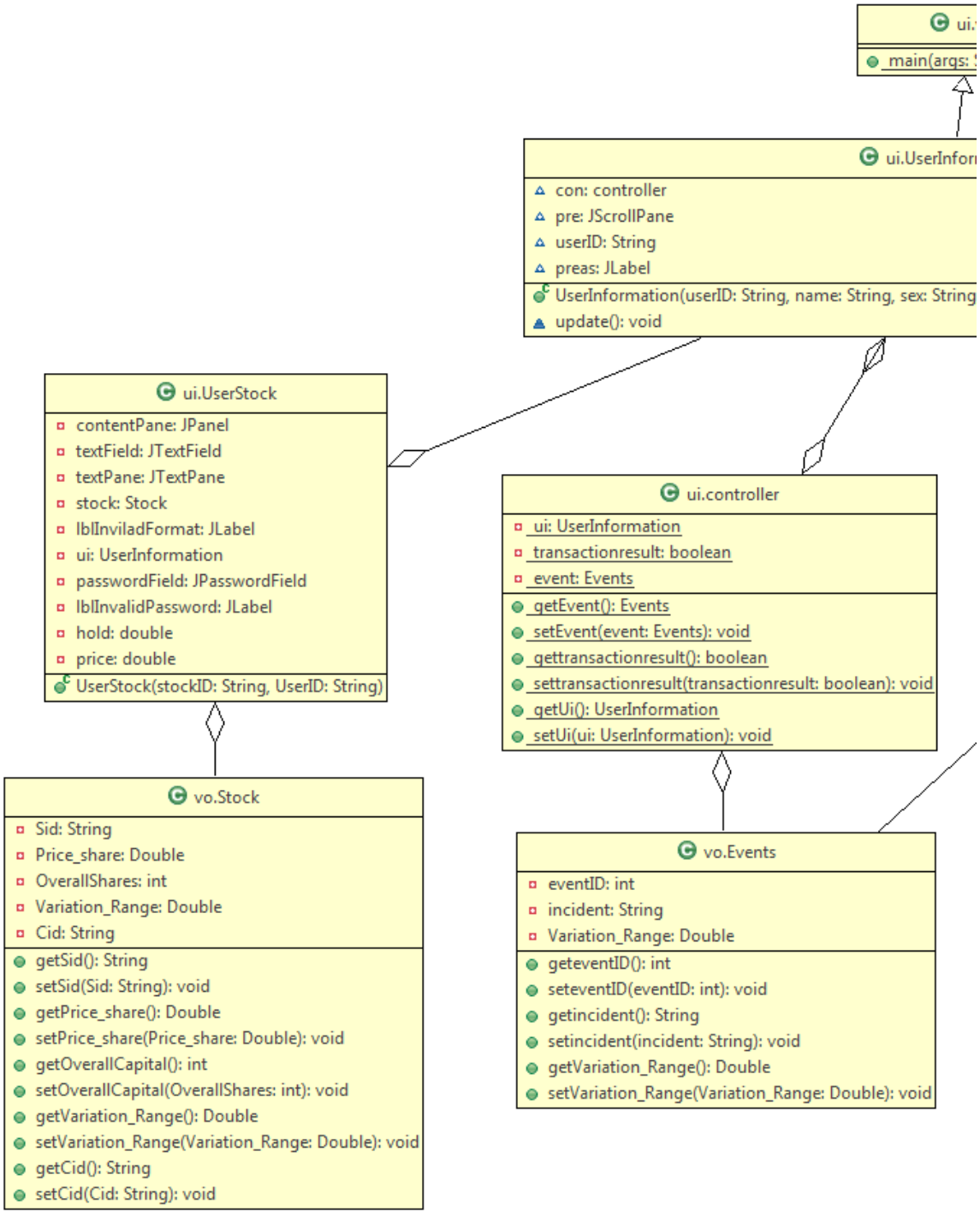


Server UML right part

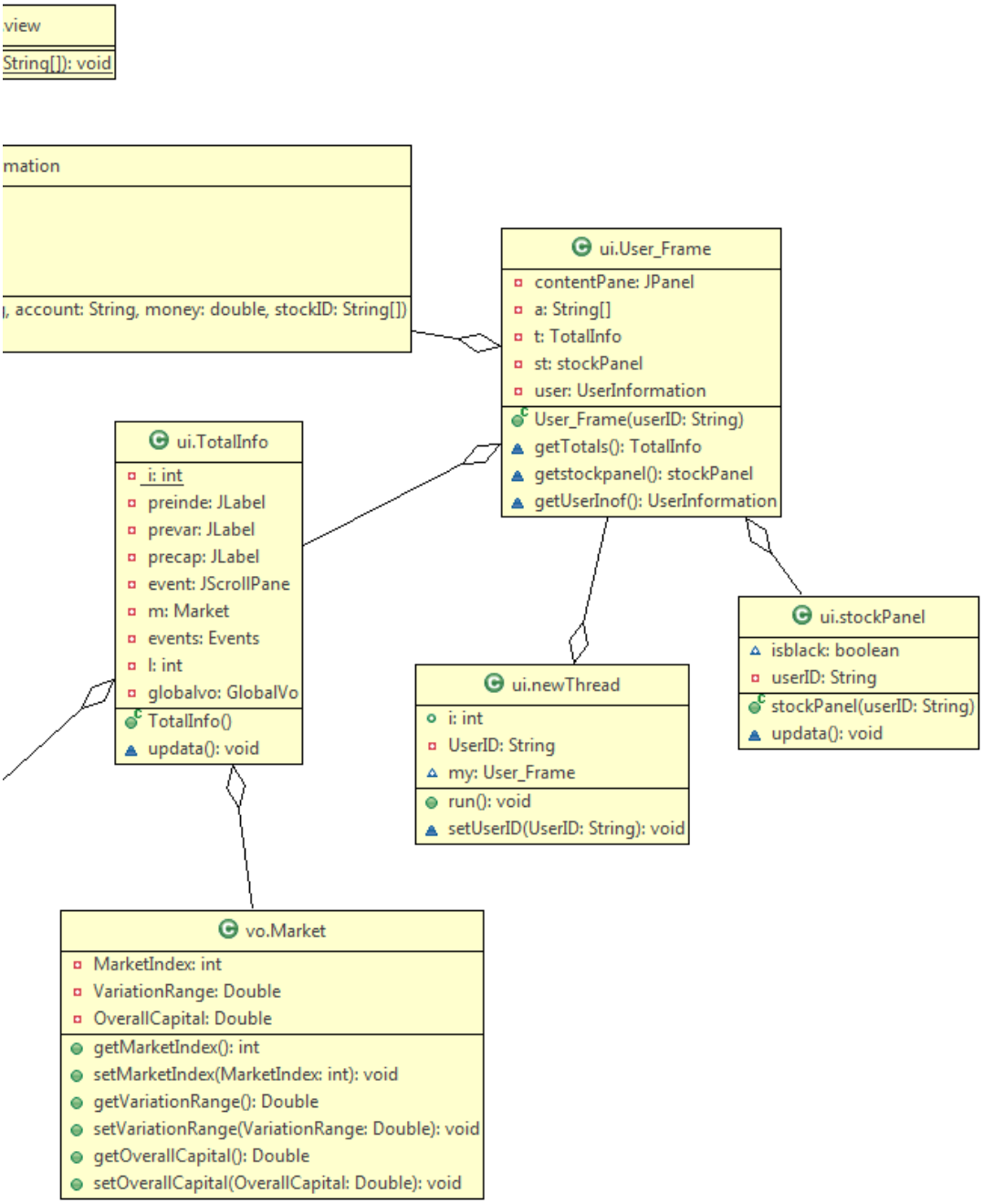
Server Description:

Server Setting Layer: com.cs542.DBMS; com.filter.AccessControlResponseFilter
URL Analyze Layer: com.cs542.resource.UserResource, etc.
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

We have multi-clients (desktop client, html client), here we use the java-based desktop client as example, the basic structure shows below:



Client UML left part



Client UML right part

Client Description:

Entry Layer: ui.view

User Interface Layer: ui.login_frame; ui.TotalInfo; ui.User_Frame; ui.controller

Logical Processing Layer: ui.stockPanel, ui.UserInformation

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

3.2.4 RESTFul API design

We know the HTTP has five verbs to operate the request, it is very like the SQL commands (The verbs inside the parentheses are the corresponding SQL commands)

- GET（SELECT）：get the data from server(one or more)
- POST（CREATE）：create an new resource in the server
- PUT（UPDATE）：update the resource in the server
- PATCH（UPDATE）：update part of resource in the server
- DELETE（DELETE）：delete the resource in the server.

Thus, here is the standard how we design the JSON API:

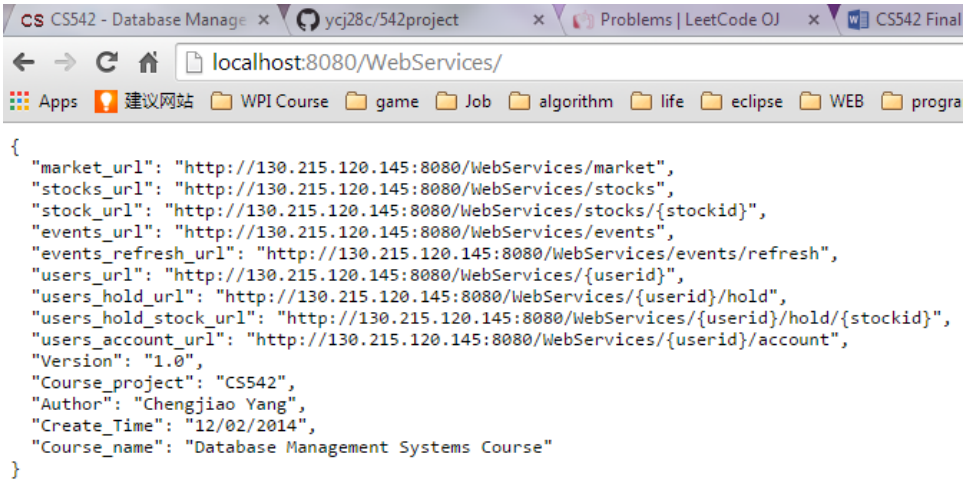
- It should use web standards where they make sense
- It should be friendly to the developer and be exportable via a browser address bar
- It should be simple, intuitive and consistent to make adoption not only easy but pleasant
- It should provide enough flexibility to power majority of the Enchant UI
- It should be efficient, while maintaining balance with the other requirements

Our JSON URL API example:

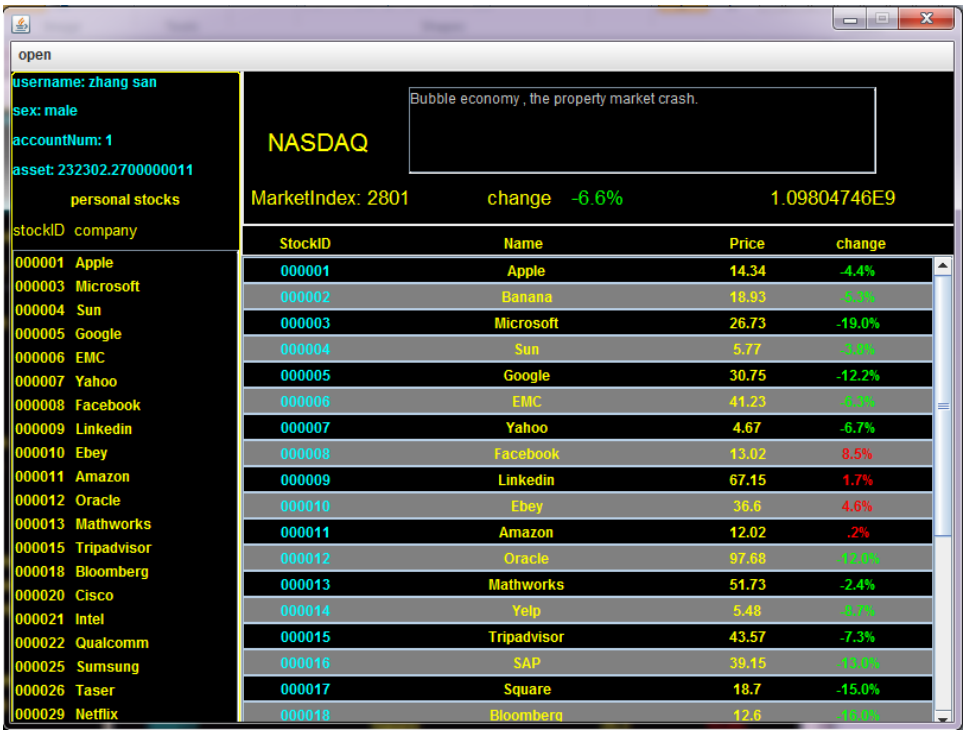
"market_url": "http://130.215.120.145:8080/WebServices/market",
"stocks_url": "http://130.215.120.145:8080/WebServices/stocks",
"stock_url": "http://130.215.120.145:8080/WebServices/stocks/{stockid}",
"events_url": "http://130.215.120.145:8080/WebServices/events",
"events_refresh_url": "http://130.215.120.145:8080/WebServices/events/refresh",
"users_url": "http://130.215.120.145:8080/WebServices/{userid}",
"users_hold_url": "http://130.215.120.145:8080/WebServices/{userid}/hold",
"users_hold_stock_url": "http://130.215.120.145:8080/WebServices/{userid}/hold/{stockid}",
"users_account_url": "http://130.215.120.145:8080/WebServices/{userid}/account",

3.2.5 UI design

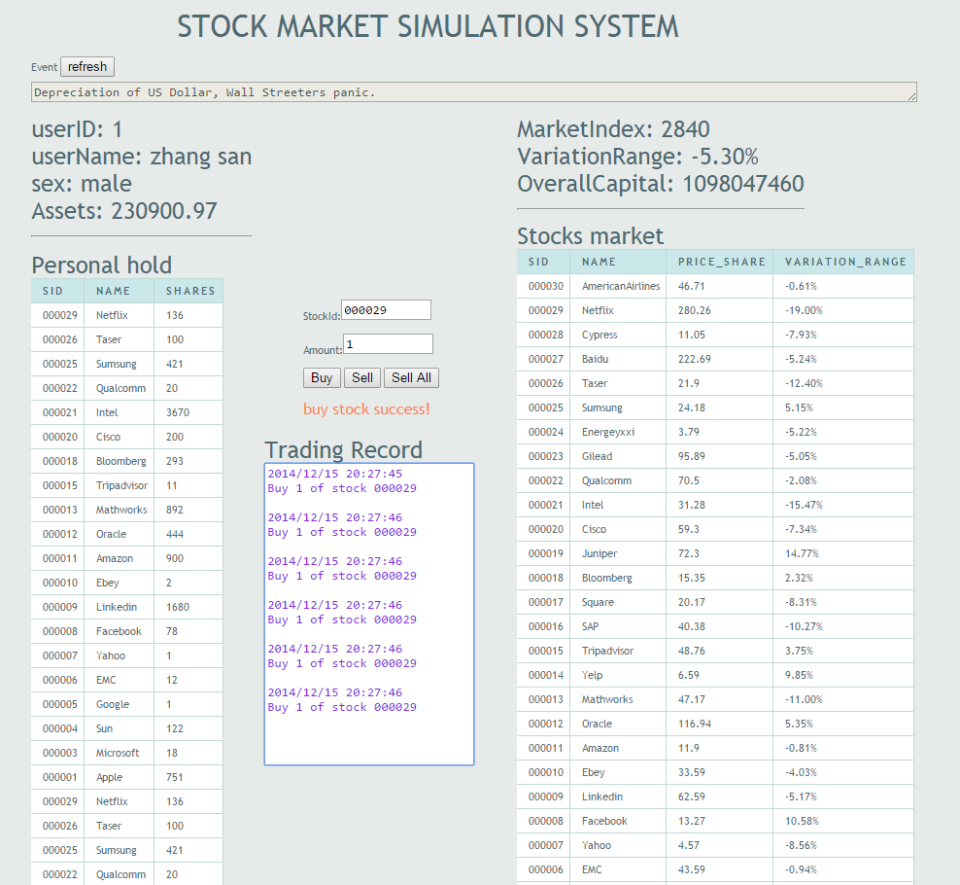
Web Service API UI Example:



JAVA Desktop Client UI Example:



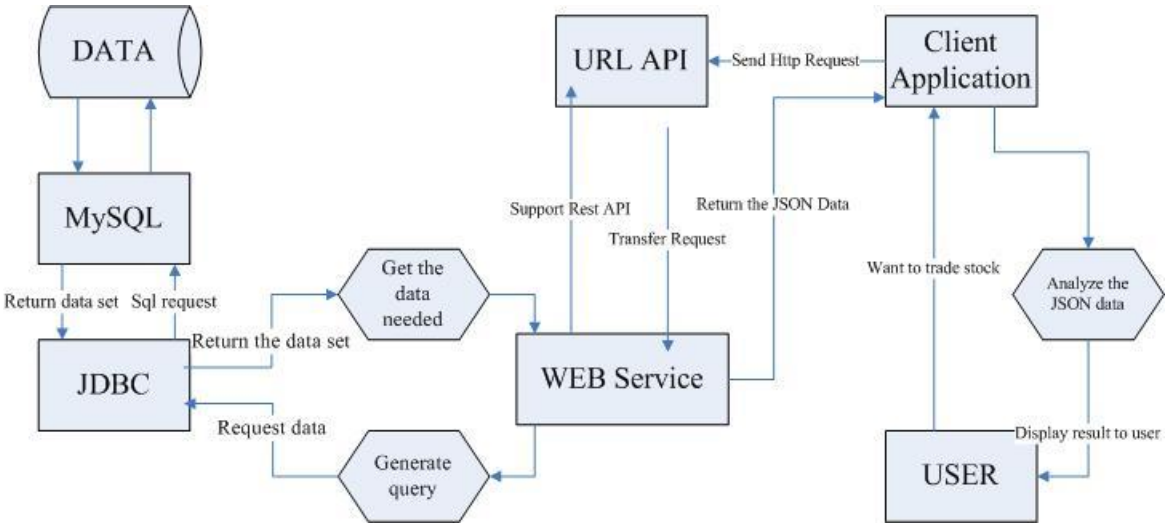
Html Client UI Example:



3.3 Implementation and system detail

Architecture:

We use Restful Web Service architecture. At server side, program connect to the MySQL database via JDBC, use Jersey lib to support JSON API URL. At Client side, there are two client, one is java-based desktop client, and we use swing-based plugin WindowBuilder for the UI part, visualization development. Another one is off-line version html client, which implement the JQUERY, CORS, CSS technologies. The general system architecture is shown as below:



Tools:

We use lots of develop tools and assist tools such as:
Develop and DB tools: MySQL5.7; Eclipse Juno; Java1.7
Rest WebService tools: Jersey2.13, glassfish4
Management tools: Github; Word online; MySQL Workbench 6.2 CE; phpAdmin; MAVEN
Eclipse Plugin: EGIT; Windowsbuilder; Swing; AmaterasUML; Fatjar
Library: forms-1.3.0; mysql-connector-java-5.1.29-bin; json20090211; jackson1.9.12; Gson2.2.4
Web tools: CSS2.0; html5; JQuery-1.8.2; Ajax; CORS

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

3.4 Ways to solve issues

The main issue is the teamwork issue. 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 still meet each other by online chat. We also 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.

About the resource share and code management aspect, we use Github for version control and material share, however, weencountered some technical problems when we were 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, what we do is keep the master version clear and runnable, doing update and changes in the fork version, then merge

them together. In addition, to share ideas more efficiently, we apply word online for project document building.

About the technical issues, here is the example how we solve the issue:We developed the local html client to access the service. However, this kind of access is cross domain access which violate the W3C standard, so we got error response hundreds of times until we know the issue is. To solve this kind of technical problem we had never met, what we did is first discuss with teammates on WeChat see if they know the solution, then ask other students weather they have the same problem before. Unfortunately, both way doesn't work. Now the only way left is to search online, and learn deeply for the cross domain standard. After doing lots of research online such as "crunchify.com", "stackoverflow.com", "developerscrappad.com", finally we got the idea (use CORS technology) and solve the problem.

In addition, after several meetings online and on site, we discussed the usability of our system, and tried to find some better style of functions to fulfill more needs from users. As the result of long time discussion and development, our newest version, Stock Market Simulation System Online comes to life. To solve the problem of widely used and trying to attract larger number of users, we apply the online approach to make our system more accessible for users all over the world. In our online version, the only thing users need to do is to download a user application from internet and connect it to our database server, then they can use its full function to train themselves.

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. Therefore to validate our program we did different test on different stages:

3.5.1Unit test

We divided our project into several sub-projects. Before we integrate these subproject we operated some tests. For example for the Factory class, it is responsible for return data from database, but it was not connected to UI we cannot make sure the data we retrieved is correct. Therefore, we do the unit test for the core java class we developed, here is a JUNIT test example, which test the iCompanyDAO work flow see if the company valuation matched.

```
public void testICompanyDAO() throws Exception{
        System.out.println("**testICompanyDAO**");
        Company company = new Company();
        company.setCid("11");
        company.setMarketValuation(1111.00);
        boolean result =DAOFactory.getICompanyDAOInstance().testMethod(company);
```

```
        boolean expResult = true;
        assertEquals(expResult, result);
}
```

3.5.2 Integration Testing

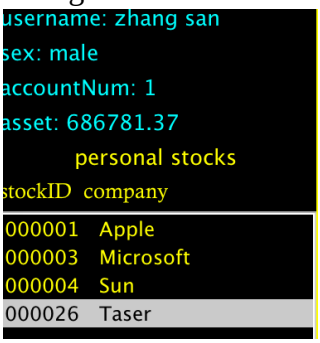
After us integrating these subprojects, we inserted 30 stocks and 5 valid user names into our database to test if any data we retrieved can display on the UI and to test the login systemcan get a match to our database.

3.5.3 System test

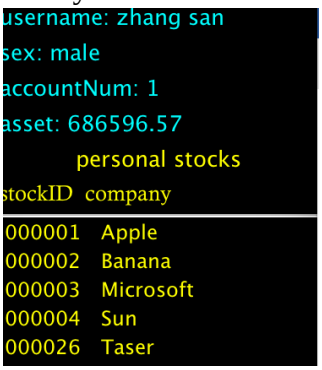
Function test

We have implemented all functions we designed, and we executed some function test to make sure everything function of our program can works the way we expected. The result of these test show us that every function works well. We can login to our program with valid ID and Password, the user's asset can update correctly. After the user buying or selling a stock, our system can update market stock price automatically every 10 sec without any error.

Here is an example of buying function:
Zhang san's asset and the stocks he hold



Hebuy stock banana



Both his asset and personal stocks have updated correctly.

Compatibility Test

Our project is a java program, we believe that our program can run on any system without error. We ran our program on Mac OSX and windows, it seem like everything works fine on these two system.

4. Lessons learned

Chengjiao Yang:

I had done with the database-related and web application project before, but it is my first time to implement the fashion technology REST to develop a Web Service. This is a complete new idea in my mind, thus I search lots of related information and make lots of attempts to make the server works. Luckily, finally it works, which connect everything together, now the server support the JSON API, accept the HTTP request, and support the CORS access. Actually, the most difficult thing is not developing, it is the designing the meaningful API which cost most the developing time. Meanwhile, import the agile development approach into our group is also a big attempt and challenge, because our project is team project. The project has perfectly completed, the progress is smooth and fun. I will refer the experience of this project, using the same method for my future project.

Jian Qiao:

I have took cs543 (graphics) and I have a little experience with developing a graphics program by using objective-c. Therefore, implement the UI is not difficult to me. But this my first time using java to develop a graphics program, it makes me have a better understanding on java. Meanwhile, it make me have a concept about c/s and MVC model. also this is my second time do project as team, I did not do my first team project so well, because that is our first time doing the teamwork, so we did not have the sense of teamwork , but for this time, so far so good. This project make me have more experience about teamwork. I believe I can benefit from this valuable experience in the future.

Yang Zheng:

Since I find out a better MySQL developing tool, I changed my coding style. Through this latest version of MySQL, the whole work can be done with straight ways. We can double check the building of tables and relations by more convenient way, and that improve the quality of our database structure. As well as new finding in database development, we also use more powerful online protocol to fulfill our target of user group. After our latest update of the system, the application, now, is able to deal with remote learning issues. From now on, everyone in the world with Internet connection is able to get access to our training system and this is, in some way, the original version of cloud technology which is very heated nowadays.

5. Member contribution

Task	Finish-by
System architecture	Whole team
Application architecture	Whole team
Database architecture	Whole team
UI design	Whole team
Rest API design	Whole team
Database detail design	Zheng Yang
Database dictionary	Zheng Yang
Database create statement	Zheng Yang
Database constraints	Zheng Yang
Database indexes	Zheng Yang
Database trigger	Zheng Yang, Chengjiao Yang
Test Data	Zheng Yang, Chengjiao Yang
VO Class	Zheng Yang, Chengjiao Yang
Application work flow Class	Chengjiao Yang
Event related valuation algorithm	Chengjiao Yang
JDBC connection Class	Chengjiao Yang
Configuration file	Chengjiao Yang
Web Service develop	Chengjiao Yang
Html Client develop	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
Automatic refresh function	Jian Qiao
Buy and sell UI and logic class	Jian Qiao
Application menu	Jian Qiao
Unit test	Jian Qiao, Chengjiao Yang
Function test	Zheng Yang, Jian Qiao
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
Project Final Report	Whole team
Meeting organizer	Chengjiao Yang

6. Conclusions

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. All the functions and plans are finished as expected. Since we have good management of the process, the project steps are accomplished very well. The whole system is already passed the test and debugging, which is now ready to use and show in the presentation. And we believe it will be a great choice for users who want to train his ability in stock market management.

7. Futurework

Although our stock simulator can simulate the basic function, I bet there is no client want to use it if we release this product. There are several parts we need to improve to make our program more useful.

1. The most important part is that the stocks we have are we designed, there is nothing to do with real stocks market, and the reason a user use stock simulator is that they want to be familiar with the real stock market without any risk. Therefore, in the future we could import the real stock data from yahoo to make our simulator like a real stock market.

2. Our program do not support Candlestick chart since we do not have enough time to implement it. However it is a crucial function to investors. Most of times professional investors buy or sell a stock by analyzing its Candlestick chart. Therefore, if we can continue with this project and we want to make it into a really useable system, we need to implement this function.

3. Our project do not support search function. As mentioned above if we can continue with this project we have to put this on schedule.

8. Reference

1) Classification of software testing, Chong Shi, 10-24-2012,
http://www.cnblogs.com/fnng/archive/2012/10/24/2737972.html
2) ECMA-404 the JSON Data Interchange Standard,
http://www.json.org/
3) Web services, Vangie Beal,
http://www.webopedia.com/TERM/W/Web_Services.html
4) Representational State Transfer (REST), Roy Thomas Fielding, 2000,
http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
5) Learn REST: A Tutorial, DR. M. ELKSTEIN,
http://rest.elkstein.org/
6) Jersey- RESTful Web Services in Java,
https://jersey.java.net/
7) HTTP access control (CORS), tfm, 08-27-2014,
https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS
8) Using CORS, Monsur Hossain, 10-29-2013,
http://www.html5rocks.com/en/tutorials/cors/
9) Stock market, Wikipedia,
http://en.wikipedia.org/wiki/Stock_market

9. Appendix

9.1 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;

9.2 Table create statement and constraints

```
--
-- Table structure for table `company`
--
DROP TABLE IF EXISTS `company`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `company` (
  `Cid` varchar(50) NOT NULL,
  `MarketValuation` double NOT NULL,
  `Name` varchar(50) NOT NULL,
  PRIMARY KEY (`Cid`),
  KEY `Cid` (`Cid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
```

```
-- Table structure for table `events`
--
DROP TABLE IF EXISTS `events`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `events` (
  `eventID` int(11) NOT NULL,
  `incident` varchar(256) DEFAULT NULL,
  `Variation_Range` double NOT NULL,
  PRIMARY KEY (`eventID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Table structure for table `hold`
--
DROP TABLE IF EXISTS `hold`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `hold` (
  `userID` varchar(50) NOT NULL,
  `Sid` varchar(50) NOT NULL,
  `shares` int(11) DEFAULT NULL,
  PRIMARY KEY (`userID`,`Sid`),
  KEY `Sid_idx` (`Sid`),
  KEY `userID` (`userID`),
  CONSTRAINT `Sid2` FOREIGN KEY (`Sid`) REFERENCES `stock` (`Sid`) ON DELETE
NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `userID2` FOREIGN KEY (`userID`) REFERENCES `investors` (`userID`)
ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Table structure for table `investors`
--
DROP TABLE IF EXISTS `investors`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = 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`),
  KEY `userID` (`userID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Table structure for table `market`
--
DROP TABLE IF EXISTS `market`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `market` (
  `marketIndex` int(11) DEFAULT NULL,
  `VariationRange` double DEFAULT NULL,
  `OverallCapital` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Table structure for table `stock`
--
DROP TABLE IF EXISTS `stock`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = 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`),
  KEY `Sid` (`Sid`),
  CONSTRAINT `Cid2` FOREIGN KEY (`Cid`) REFERENCES `company` (`Cid`) ON
DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
--
```

```
-- Table structure for table `transactionrecord`
--
DROP TABLE IF EXISTS `transactionrecord`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `transactionrecord` (
  `TID` varchar(50) NOT NULL,
  `userID` varchar(50) NOT NULL,
  `Cid` varchar(50) NOT NULL,
  `time` date NOT NULL,
  `price` double NOT NULL,
  PRIMARY KEY (`TID`),
  UNIQUE KEY `TID_UNIQUE` (`TID`),
  KEY `userID` (`userID`),
  KEY `Cid` (`Cid`),
  CONSTRAINT `Cid` FOREIGN KEY (`Cid`) REFERENCES `company` (`Cid`) ON DELETE
NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `userID` FOREIGN KEY (`userID`) REFERENCES `investors` (`userID`) ON
DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
```

9.3 Test data

```
--
-- Dumping data for table `company`
--
LOCK TABLES `company` WRITE;
/*!40000 ALTER TABLE `company` DISABLE KEYS */;
INSERT INTO `company` VALUES
('1',0.5,'Apple'),('10',0.5,'Ebey'),('11',0.5,'Amazon'),('12',0.5,'Oracle'),('13',0.5,'Mathworks'),('14',0
.5,'Yelp'),('15',0.5,'Tripadvisor'),('16',0.5,'SAP'),('17',0.5,'Square'),('18',0.5,'Bloomberg'),('19',0.5,'
Juniper'),('2',0.5,'Banana'),('20',0.5,'Cisco'),('21',0.5,'Intel'),('22',0.5,'Qualcomm'),('23',0.5,'Gilead'
),('24',0.5,'Energeyxxi'),('25',0.5,'Sumsung'),('26',0.5,'Taser'),('27',0.5,'Baidu'),('28',0.5,'Cypress'),
('29',0.5,'Netflix'),('3',0.5,'Microsoft'),('30',0.5,'AmericanAirlines'),('4',0.5,'Sun'),('5',0.5,'Google'),
('6',0.5,'EMC'),('7',0.5,'Yahoo'),('8',0.5,'Facebook'),('9',0.5,'Linkedin');
/*!40000 ALTER TABLE `company` ENABLE KEYS */;
UNLOCK TABLES;
--
-- Dumping data for table `events`
--
LOCK TABLES `events` WRITE;
```

```
/*!40000 ALTER TABLE `events` DISABLE KEYS */;
INSERT INTO `events` VALUES (1,'A celebrity hung up, the stock market effected.',-0.03),(2,'Terrorist attacks in the United States , the World Trade was destroyed.',-0.1),(3,'The success of the six-party talks , a peaceful world.',0.04),(4,'Bubble economy , the property market crash.',-0.09),(5,'Chang E 100 spacecraft was successfully launched , the positive reaction.',0.05),(6,'Obama expands OPT visa program for foreign students, good news.',0.04),(7,'Food safety issue occur, people are angry.',-0.07),(8,'Chinese soccer team win the world cup, cheers!',0.06),(9,'Olympic Games month, go go go.',0.03),(10,'Depreciation of US Dollar, Wall Streeters panic.',-0.03),(11,'Large volcanic eruption in Yellowstone National park, western of United States are affected',-0.05),(12,'The new technology of pig keeping is invented, now people can eat cheap pork',0.03),(13,'New diplomatic relations between the Mars and the earth, go into the era of the Milky Way',0.07),(14,'New York blackout last for two weeks, Economic affected.',-0.04),(15,'Global economic recovery, employment rate increased.',0.05);
/*!40000 ALTER TABLE `events` ENABLE KEYS */;
UNLOCK TABLES;
--
-- Dumping data for table `hold`
--
LOCK TABLES `hold` WRITE;
/*!40000 ALTER TABLE `hold` DISABLE KEYS */;
INSERT INTO `hold` VALUES
('1','000001',111),('1','000002',111),('1','000003',234),('1','000004',111),('1','000005',1),('1','000006',1),('1','000007',1),('1','000008',1),('1','000009',1680),('1','000010',2),('1','000011',900),('1','000012',444),('1','000013',888),('1','000015',11),('1','000018',284),('1','000019',100),('1','000020',200),('1','000021',11),('1','000022',111),('1','000023',11),('1','000024',11),('1','000026',11),('2','000001',284),('2','000002',42),('2','000019',500),('3','000002',2342),('3','000003',235),('4','000001',235),('4','000003',551);
/*!40000 ALTER TABLE `hold` ENABLE KEYS */;
UNLOCK TABLES;
--
-- Dumping data for table `investors`
--
LOCK TABLES `investors` WRITE;
/*!40000 ALTER TABLE `investors` DISABLE KEYS */;
INSERT INTO `investors` VALUES ('1','111',433327.27,2000,'male','zhang san'),('2','111',40001,555554,'female','lucy'),('3','111',999999,3333,'male','Mike'),('4','111',34534,4444,'male','Ted'),('5','111',45234,5555,'male','Ralph'),('6','111',-45435,1234,'female','Cindy');
/*!40000 ALTER TABLE `investors` ENABLE KEYS */;
UNLOCK TABLES;
--
-- Dumping data for table `market`
--
```

```
LOCK TABLES `market` WRITE;
/*!40000 ALTER TABLE `market` DISABLE KEYS */;
INSERT INTO `market` VALUES (3000,0.1,1098047460);
/*!40000 ALTER TABLE `market` ENABLE KEYS */;
UNLOCK TABLES;
--
-- Dumping data for table `stock`
--
LOCK TABLES `stock` WRITE;
/*!40000 ALTER TABLE `stock` DISABLE KEYS */;
INSERT INTO `stock` VALUES
('000001',15,2000000,0.14,'1'),('000002',20,6666666,0.16,'2'),('000003',33,454353,0.19,'3'),('000
004',6,34634234,0.12,'4'),('000005',35,6345,0.16,'5'),('000006',44,646345,0.14,'6'),('000007',5,42
345,0.16,'7'),('000008',12,235523,0.18,'8'),('000009',66,34345,0.16,'9'),('000010',35,23623,0.15,'
10'),('000011',12,235623,0.13,'11'),('000012',111,124524,0.12,'12'),('000013',53,62356234,0.11,'
13'),('000014',6,24523,0.19,'14'),('000015',47,46236235,0.16,'15'),('000016',45,43345,0.13,'16'),('
000017',22,3735723,0.15,'17'),('000018',15,4632342,0.16,'18'),('000019',63,223535,0.18,'19'),('0
00020',64,34636235,0.12,'20'),('000021',37,13266085,0.16,'21'),('000022',72,8592866,0.2,'22'),('
000023',100,8427649,0.11,'23'),('000024',4,7959969,0.19,'24'),('000025',23,7807701,0.18,'25'),('
000026',25,7352702,0.17,'26'),('000027',235,389488641,0.15,'27'),('000028',12,378143686,0.13,'
28'),('000029',346,352721007,0.19,'29'),('000030',47,350031447,0.11,'30');
/*!40000 ALTER TABLE `stock` ENABLE KEYS */;
UNLOCK TABLES;
```

9.4 Trigger for transactionrecord

```
/*!50003 CREATE*/ /*!50017 DEFINER=`cs542`@`%`*/ /*!50003 TRIGGER
`cs542`.`hold_BEFORE_INSERT` AFTER INSERT ON `hold` FOR EACH ROW
insert into cs542.transactionrecord values (sysdate(),userID,(select cid from cs542.stock where
sid = New.sid), sysdate(), New.shares) */;;

/*!50003 CREATE*/ /*!50017 DEFINER=`cs542`@`%`*/ /*!50003 TRIGGER
`cs542`.`hold_BEFORE_UPDATE` BEFORE UPDATE ON `hold` FOR EACH ROW
insert into cs542.transactionrecord values (sysdate(),userID,(select cid from cs542.stock where
sid = OLD.sid), sysdate(), old.shares) */;;

/*!50003 CREATE*/ /*!50017 DEFINER=`cs542`@`%`*/ /*!50003 TRIGGER
`cs542`.`hold_BEFORE_DELETE` BEFORE DELETE ON `hold` FOR EACH ROW
insert into cs542.transactionrecord values (sysdate(),userID,(select cid from cs542.stock where
sid = OLD.sid), sysdate(), old.shares) */;;
```

9.5 SQL work flow logic

```
/* calculate how many male or female in our system */
SELECT count(*),sex FROM cs542.investors group by sex;

/* get random event */
SELECT eventID,incident,Variation_Range FROM events order by rand() limit 1;

/* get all stocks a user hold */
select c.name, a.shares, b.sid from (select sid, shares from hold where userid = ?) a, stock b,
company c where b.cid = c.cid and a.sid = b.sid;

/* sell all stocks of a user */
delete from hold where userID = ? and Sid = ?;

/* sell amount of stocks a user */
delete from hold where userID = ? and Sid = ?;
UPDATE hold set shares = shares-? where userID = ? and Sid = ?

/* buy amount of stocks a user */
insert into hold values(?,?,?);
UPDATE hold set shares = shares+? where userID = ? and Sid = ?;

/* get shares of a stock a user */
SELECT shares FROM hold where userID = ? and Sid = ?;

/* get user information */
select userName, sex, userID, Assets from investors where userID = ?;

/* increase money for a user */
update investors set Assets = Assets+? where userID = ?;

/* verify the login in function */
select 1 from investors where userID = ? and password = ?;

/* get the market information */
select marketIndex,VariationRange,OverallCapital from market;
```

```
/* list all the stocks information */
select a.sid,b.name,a.price_share,a.variation_range from stock a,company b where a.cid = b.cid;

/* list top 20 stocks information */
select a.sid,b.name,a.price_share,a.variation_range from stock a,company b where a.cid = b.cid
limit 20;

/* get stock by its sid */
SELECT Sid,Price_share,OverallShares,Variation_Range,Cid FROM stock where Sid = ?;
```

a