**ECE 651: Foundations of Software Engineering**

**Project Proposal**

**Group 3**

Zhechen Du 20340480(z6du)

Ke Chen 20456026(kchen)

Shangru Li 20637274(s488li)

Hongming Wang 20667749(h539wang)

Zhao Zhang 20689699(z664zhan)

**Introduction**

In today’s world, online currency exchange has become more accessible to regular people. However, we still found it hard to trade with other people. For example, the exchange rate can be extremely high for traders in certain countries. We want to develop an online platform that allows trader to trade directly with other traders at their own preferred rate. Since including every currency is almost impossible to complete for a two-month project, we decided to scale down the platform to include just three currencies: USD, CAD, and RMB. However, our architecture should be able to handle the expansion if project is to be continued beyond this course. The exchange will be able to perform four basic functions: buy, sale, deposit, and withdraw. The end user will have the option to trade between three currencies. Due to time constraints, he/she will only be able to operate the exchange on Android devices powered by Java. The backend of the platform will also be developed in Java with MySQL as the database. A flowchart of the exchange can be found on in the Appendix

The online trade system is a very complex and interesting problem for computer architecture. First, there are few strict requirements. For example, security and dependability are essential for such system. Second, the platform has to be intuitive and user friendly. Third, the system must have the ability to process large amount of data in real time. All the things listed above is a design challenge for system architecture. By completing this project, we hope that we can apply all knowledge we learned from the class to our project.

**Environment**

The project would be divided into two parts. The server would be set up on a desktop platform, and the client would be set up on the Android platform.

Considering about the requirement of concurrency, desktop should be a comprehensive platform which is able to handle concurrent users trade requests. Users could make a transaction with different currency, and they also could deposit or withdraw from their own accounts. However, these operations are all requested from clients and would be processed in the server which is the backend of the program. That means the server on the desktop platform requires effective architectures and algorithms. Comparing with other running environments, such as mobile, cloud, or many others, desktop has advantages on processing high concurrency, not only for its high stability, it is also economic for the beginning of a program and convenient for the testing. Besides, trade program needs high security level to ensure the safety of the transaction data, and desktop could enforce the security from different aspects such as encryption of database, login session limitation, and transaction time limitation. Based on our idea, the security would be enabled not only in program but also in database.

Desktop platform is also capable for the future extension. In this project, the aspects of scalability may include more type of currency, commission, chain transaction, and API for other applications.

To sum up, in the backend of this project, C/S framework would be preferred, and Java Enterprise Edition would be used to build a backend system on the desktop platform. The system will process all requests from clients, and send the operations results to the users. The communication between server and clients would be the network. For the transaction data persistence, MySQL 5.7 database would be considered as the data container. Apache Tomcat 8.0 free edition would be the server to load the program. In addition, Maven would play an important role of loading components which we need.

The clients of this project would be implemented on Android 6.0 platform. As a frequently used application, the client should be very accessible. Developing a mobile application that runs in Android OS meets the requirement of portability. Users can trade with others conveniently from anywhere the network is available. Furthermore, Android 6.0 has high compatibility with smartphones in the current mainstream market, which is a plus for distribution of our product. The version 25.2.2 of Android SDK tools is chosen because its stability in development and performance when testing the software in the emulator. The whole client system would be developed in Android Studio 2.2, the newest official Android IDE. It helps build all components orderly, also provides tools to catch problems related to version compatibility and usability.

The graphical user interface would be designed in the client. It contains a basic login page and a transaction page to receive the input data from the user. If a user fails to enter the corresponding password correctly for three times, the account would freeze immediately. Under normal conditions, the data would be processed by using a basic encryption algorithm for security reasons. After that, the client would send a request to inform the server of data transmission. As for scalability, it’s simple to add some more types of currency in the front end. More to-be-developed functions can be integrated in the interface level.

**Functional Properties**

Our mobile application is to focus on currency trade market. Hence, our goal is to build a reliable system which is easy to operate for users who need to do the currency exchange. This system provides a platform for users to trade their money at the best market exchange rate. As a result, the functional properties of our system for users could be listed as following:

1.1 Create personal security account and users login/ logout functions

1.2 Deposit the currency holding in users’ accounts.

1.3 Withdrawal the currency demanded from users’ accounts and returns the money from this system to users’ bank accounts.

1.4 The system provides the current exchange rate as a reference for users.

1.5 The system provides customer service for users by email.

1.6 Trade (We have the detailed explanation from user scenario section)

1.6.1: User's set their exchange currency amount. The algorithm will calculate a feasible exchange rate and amount for user to decide.

1.6.2: Users can sell the holding currencies as the flexible amount. Our system holds their money and matches the buyer(s) who accept our feasible exchange rate provided.

1.6.3: Users can buy the desired currencies as the flexible amount.

1.6.4: Users can establish an offline trade face to face. They negotiate the amount and exchange rate in person, and by scanning the barcode to complete the trade. Our database would update correspondingly as soon as the personal account information change shown online.

1.6.5 The database should use garbage collection. For example, automatically delete seller list that have been on the list for a long time.

1.6.6 Create multiple tables for single currency exchange listing; freeze some table for sorting and other change. The server should be able to handle real time large data process

1.7 Store and track each user's trading history, check user accounts balance.

1.9 Create the exchange rate calculator.

1.10 No transaction fee for trade

Optional Functional Properties if time is allowed:

2.1 Create financial news for trader to view.

2.2 Create live chat/ telephone service as the extend customer service.

2.3 Create notification to inform user breaking news.

2.4 Create captcha for secure system

2.5 Create the exchange rate tendency graph.

2.6 Create advertisement.

2.7 Create user guide

2.8 Create chain trading algorithm for better trade matching

2.9 Set accounts, change users’ telephone numbers or email addresses, update the password.

**Simulated scenarios: We provide the most challenging parts as two scenarios to illustrate our core functions of this system. To achieve the core functions, we need to build the efficiency algorithms to meet the user's demands, and design the quality database structure.**

Scenario 1: (functional properties 1.6.1, 1.6.2, 1.6.3)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USD to CAD seller list | | | CAD to USD seller list | | |
| ID | Exchange Rate(USD to CAD) | Amount | ID | Exchange Rate(USD to CAD) | Amount |
| 1 | 1:1 | 1000 | 1 | 1:0.9 | 1000 |
| 2 | 1:1.1 | 2000 | 2 | 1:0.8 | 2000 |
| 3 | 1:1.2 | 1000 | 3 | 1:0.7 | 1000 |

Table 1. Simple illustration of potential seller table list.

Note: For simplicity, both exchange rate are listed as USD to CAD

In the first scenario, the user wants to buy an uncertain amount of USD using 2000 CAD. He/she has to go to the BUY page and enter the desired currency and amount. In this case, it is 2000 CAD to USD. The algorithm will check USD to CAD seller list. From the sorted list, the lowest exchange rate is 1:1 with 1000 USD. The algorithm finds the user can trade 1000 CAD for 1000 USD. This information will be saved, and ID 1 will be freezed until further action. Then, the algorithm will check the second available ID, which is 2. Its exchange rate is 1:1.1 with 2000USD. The user can exchange 1000 CAD for 909 USD. The algorithm will add the two amounts 909+1000=1909 and then calculate the exchange rate, which is 1:1.047. The amount and exchange will be displayed on buyer’s screen. The buyer will have 10 seconds to decide whether to accept this offer. Meanwhile, the database will again freeze ID 2 until user made its decision. If user decide to buy, the database will be updated to the table below. If the user decides not to buy, he/she have to option to list their desired price on CAD to USD seller’s list.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| USD to CAD seller list | | | CAD to USD seller list | | |
| ID | Exchange Rate(USD to CAD) | Amount | ID | Exchange Rate(USD to CAD) | Amount |
| 1 | 1.1:1 | 1091 | 1 | 0.9:1 | 1000 |
| 2 | 1.2:1 | 1000 | 2 | 0.8:1 | 2000 |
|  |  |  | 3 | 0.7:1 | 1000 |

Table 2. Updated potential seller table list

Scenario 2: (functional properties 1.6.4)

This case shows a face-to-face trade example. In our daily life, it happens a lot that people need to exchange currencies with their friends or acquaintances. In order to ensure the trade can be done securely and flexibly, we will develop a face-to-face offline trade function in our application. For instance, if user A wants to exchange the currency with user B face to face by e-commerce method. The only thing they need to do is to open this app and set the amount and exchange rate which they both has agreed, and then one user just simply scans another user’s QR produced by this app. The system is able to figure out if this deal can be completed or not. If all the requirements have been satisfied, the trade will be accomplished and their currencies exchange will be done successfully.

**Non-functional properties**

This project’s three main non-functional properties are security, scalability, and efficiency.

In the security part, the most important as well as basic secure feature is to ensure the safety of users' asset which includes both their own accounts and platform hold during transaction. That means this system should have the ability to protect the data in both database and dataflow. In database, all data should be stored in a well designed and effective way in order to not only reduce the redundancy but also increase the reaction rate. Moreover, some database level security standards should be considered and applied. In dataflow, system level criteria should be applied to maintain the safety of dataflow. For example, all users related operations such as login should be strictly controlled. Another significant secure point is time. Since most of the information in a trade system is related to time; for instance, the exchange rate, time is another point should be controlled. This system would restrict time in many functions to ensure the effectiveness and security of transaction.

Considering about the future functions extension, another non-functional property is scalability. Since the financial systems are mostly related with currency, and it may provide services through other systems, it is meaningful to design some interfaces which connect to others. Besides, currency types which provide to users should be able to modify for the various situations, and for the future benefit, transactions fees between users may be charged with different rates. Other features of scalability include extending the basic function such as chain exchange among many users, paying grocery shopping, and trade without net connection.

Finally, server efficiency is highly related to user experience. In order to handle real time trade data, we need to develop an efficient algorithm. First, we have to develop a sorting algorithm for potential seller list. Due to size of the data table, the seller list should not be updated in real time, we need to implement rules and method on how the table changes. Second, we have to implement a garbage collection algorithm; this could significantly reduce the table size. Lastly, we have to handle some of the database in RAM; this should be done with the concern of security.

**Appendix**

trade.png

Figure 1. Flowchart of basic user scenario.

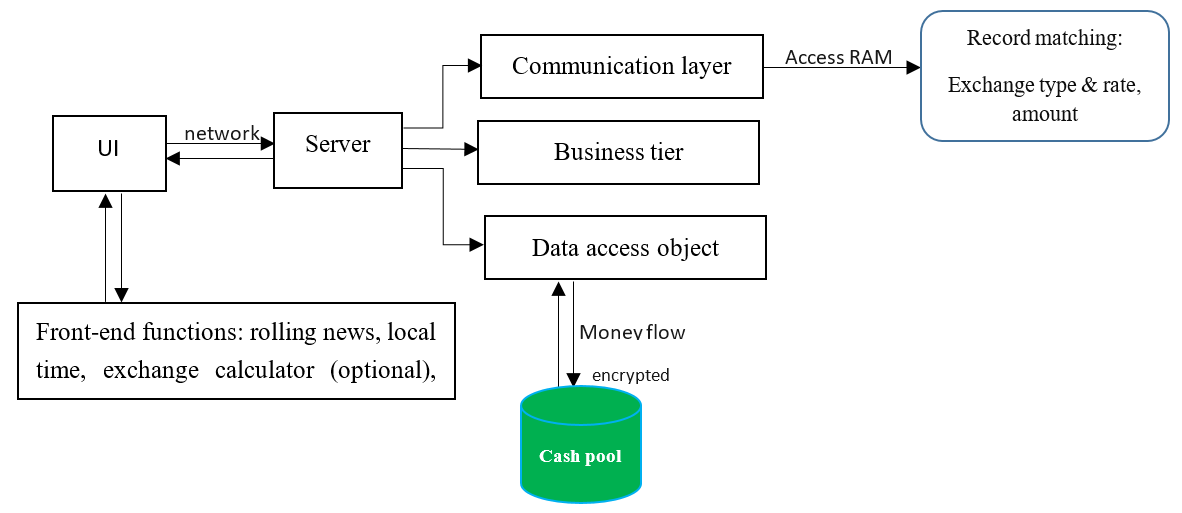


Figure 2. Structure design of the project.