ECE 651: Foundations of Software Engineering
Deliverable 3
Group 3

FREEX
(Free currency trade APP)

Ke Chen 20456026(kchen)

Zhechen Du 20340480(z6du)

Shangru Li 20637274(s488li)

Hongming Wang 20667749(h539wang)

Zhao Zhang 20689699(z664zhan)

Status Report

After deliverable 2, our group has met several times. We were focusing on combining individual works and simulated the user’s actions to test our system. Also, we tried to complete data transmissions and storage logically .

Client-side Android Development

We have made some improvements for Androids end both functional implementations and UI design after deliverable 2. We finish the functional architecture design for user Interface in order to achieve data transmission between androids end to backends based on backends data layer design. Moreover, we tried to connect between client side and backends several times for “login”,”sign up” and “deposit” simulations, and it works well so far. In addition, we integrated web API for our real time exchange rate calculator design, to achieve this function, we applied the URL and API key for future use. Also, for our face to face trade, we finish the QR code generation and QR code scanner activities. In the next two weeks, we are going to optimize the user interface and continue to connect to backends by simulating all functional operations; for instance, buy in, sell out,face to face trade,etc.

Connector Layer

For this layer, we implement the connect functions between the Androids Ends and the Data Access Object layer based on JSON. Also we put all the logical part for the data operations in this layer. For now, we completed the register, login, password change, deposit, withdrawal functions. This layer gets request from android ends, uses functions is passed to the next layer and then return data to android ends. In next steps, we will finish all the functions to handle different actions from android ends and test these new functions.

Core Trade Algorithm

We have created the framework for trade algorithm. Three concurrent high speed h2 memory databases are used for real time trading. Currently, the curreny-pool and the sort pool databases are implemented, we have created rules that is used to update the pool. Special algorithm was used to speed up the access time. Furthermore, the algorithm used to match trade is also implemented. We are able to add, delete, exchange using the database. Next step, we will implement the connection between the databases.

Data Access Object layer

In the backend, the data layer was optimized, we changed the structure of the relational database, we also optimized the data type for each attributes. For the common data access object layer, data operations was optimized to help make the database more precise. We use hibernate4 as framework, and apache DBCP as the database connections pool. By now we have finished the data access object component and begin to test each functionality. The challenge here is using java reflection and generics to create a common component for all entities. After building the database, we used uniform distribution and normal distribution to generate data and simulate hundreds of users account information, balance status and transaction history in R. Also, we use the MD5 hash to encrypted the generated data for database. Next step, we would combine the service layer with data access object layer and finish all transaction and logic about this system.

Demo description

We are going to present our current project procedure. We made some progresses based on our original architecture design. We tried to connect between client-side and backend several times, and simulated some functional operations successfully; therefore, we will show how our data transmission works by simulating the “login” and “deposit” operations. Since the most challenge and unique part of our project is our self-designed algorithm; thus, we will explain how we will integrate this algorithm to our architecture design and how to use this algorithm to meet users’ trading demands. Also, we will show our androids end user interface design on the cellphone. In addition, we will explain our encryption and decryption processes for the dependency non-functional property, and some future design ideas for the adaptability non-function property.

FreeX

By: Ke Chen, Zhechen Du, Shangru Li, Hongming Wang, and Zhao Zhang (Group 3)

UNIVERSITY OF WATERLOO

Background

- Exchange between users can be challenging
- Platform charges high transaction fee
- Secure mobile platform are very rare to find
- Most of the platform are not intuitive to use

UNIVERSITY OF WATERLOO

Goal

- User friendly platform
- Secure transaction
- Low exchange rate
- No transaction fee
- Dynamically expand platform

UNIVERSITY OF WATERLOO

Functional Properties

Core Function: TRADE

- Deposit
- Withdrawal
- Buy desired currency (Urgent demand)
- Sell holding currency (Provide own interest rate and waiting for trade)
- Face to face trade (By generate and scan the QR code)

Note: Using the self-designed algorithm to support currency exchange trade.

- Data structure: Hash Table, ADT Stack

Functional Properties

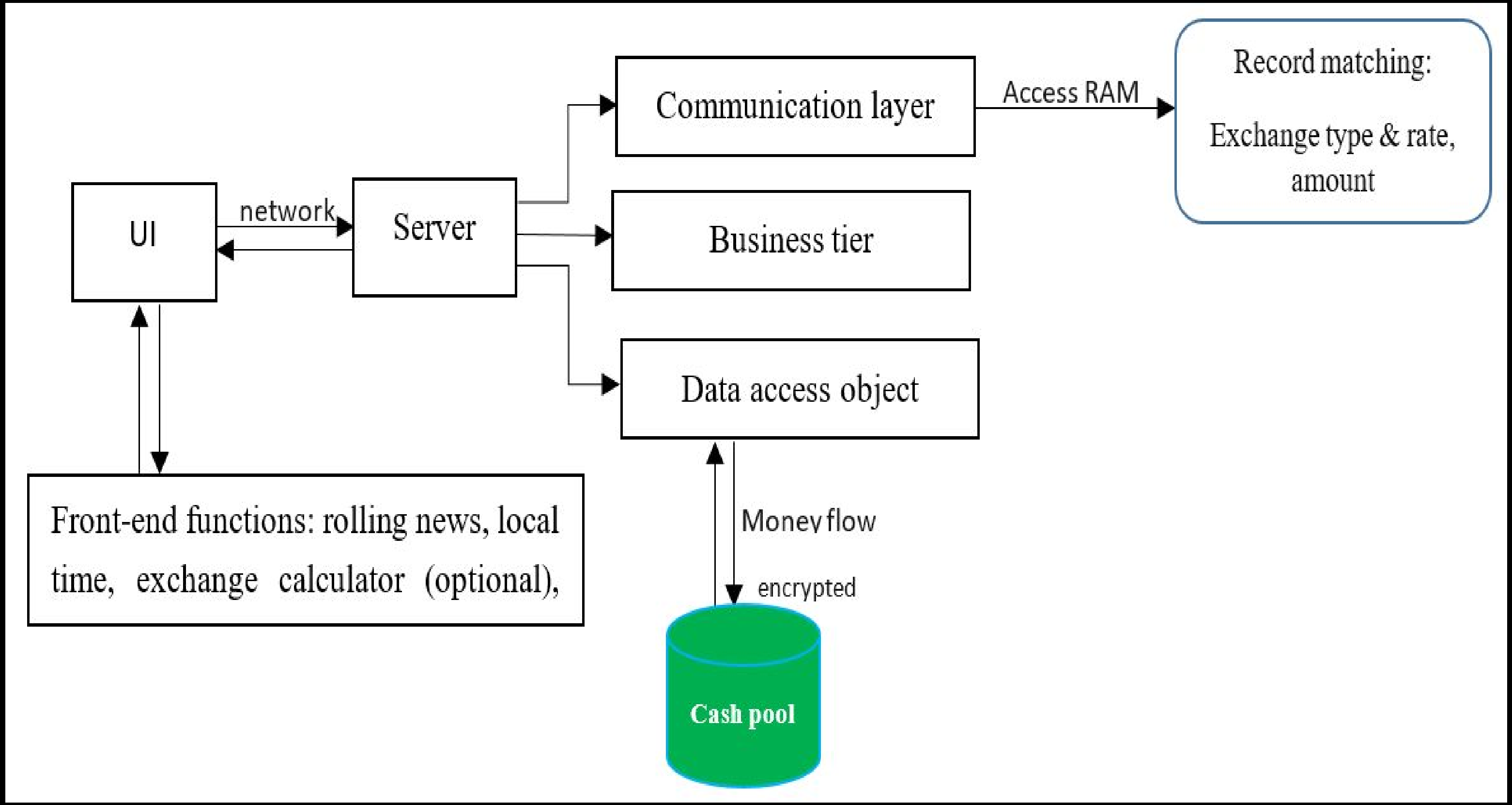
Accessibility

- Store and track each user’s trading history
- Check user accounts balance
- Real-time exchange rate calculator (Web API)
- Create notification to inform user breaking news

UNIVERSITY OF WATERLOO

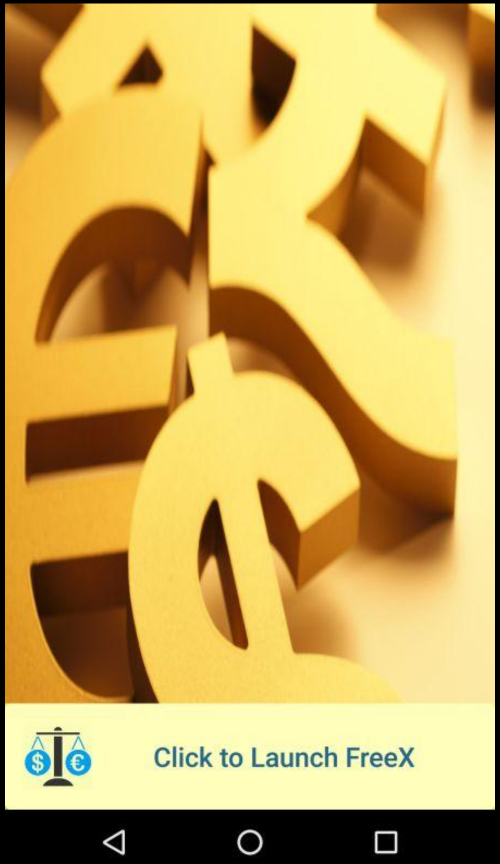
Architecture

- Model + View + Control
- Frontend: Android
- Backend: Java Web
- Spring IoC management
- Server: Apache Tomcat

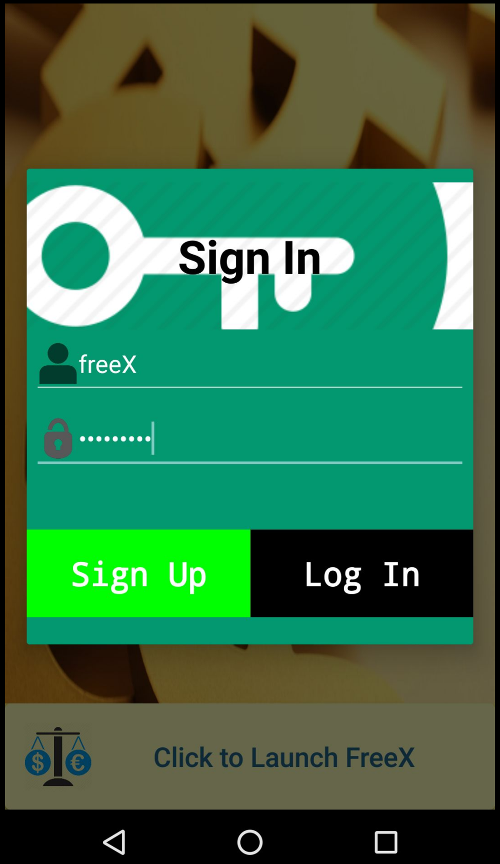


Client-side Android End

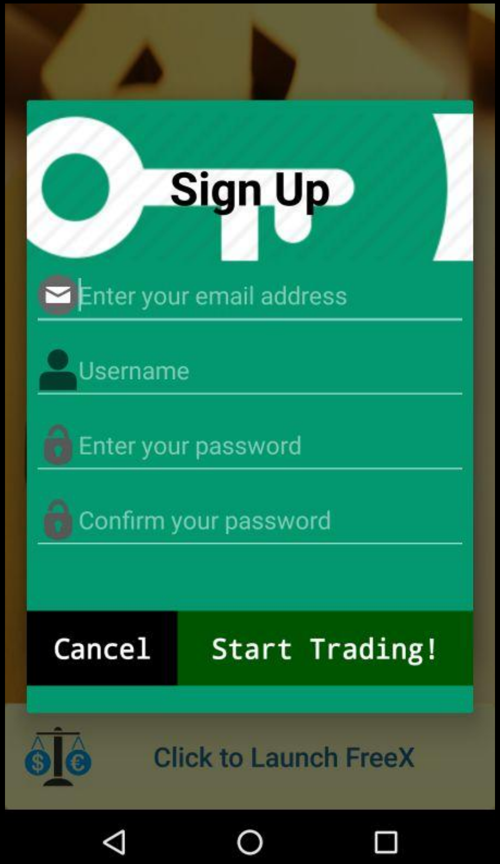
Corresponding to the functional properties, following pages are designed in Client-side Android end:



The welcome page with a launch button



Sign In Page

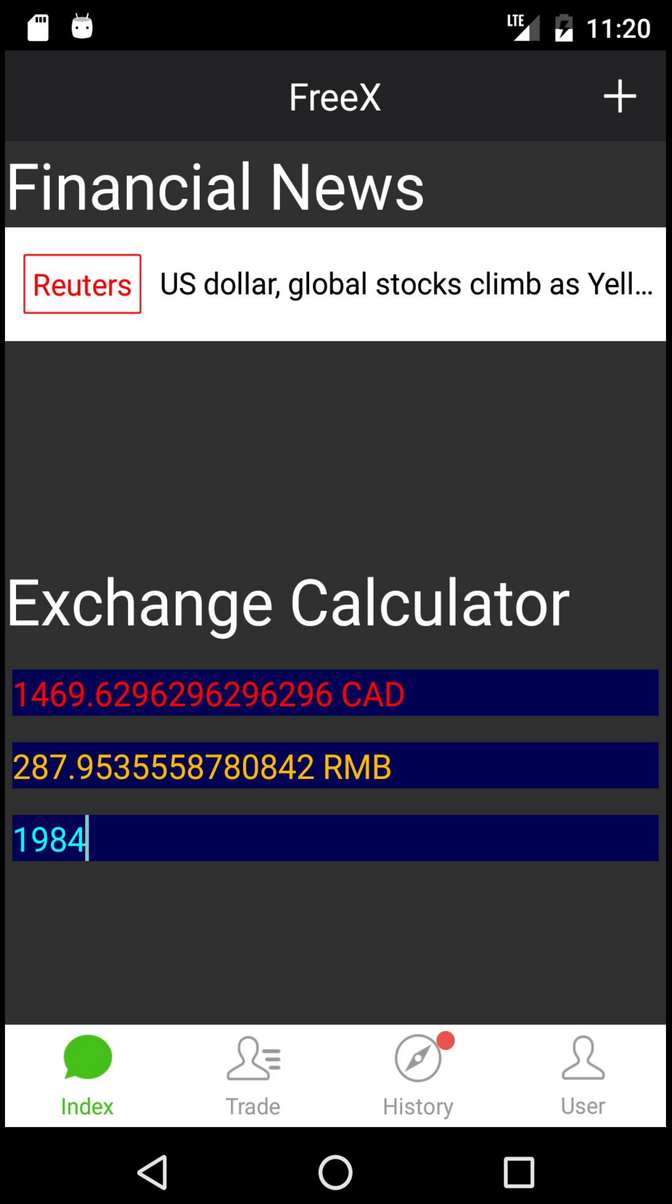


Sign Up Page

UNIVERSITY OF WATERLOO

Client-side Android End

- Wechat-like fragment-based UI. Index page contains a news flipper and an exchange rate calculator.



11:20

FreeX

Financial News

Reuters US dollar, global stocks climb as Yell...

Exchange Calculator

1469.6296296296296 CAD

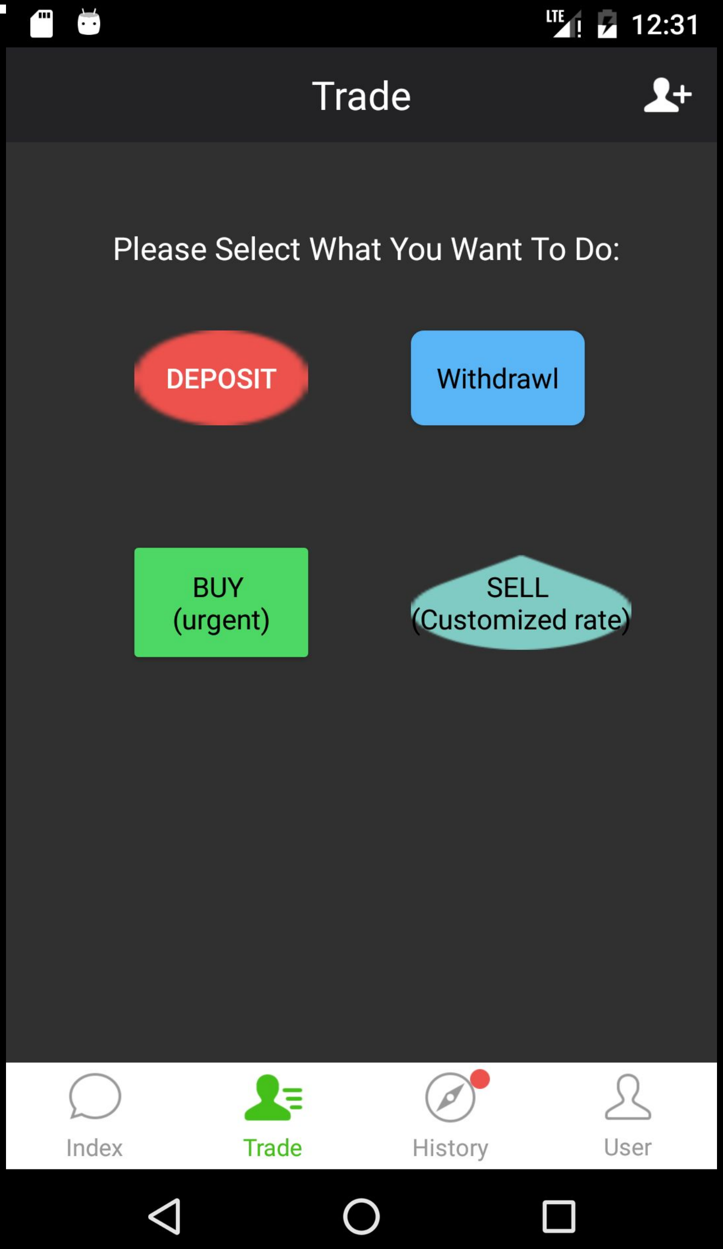
287.9535558780842 RMB

1984

UNIVERSITY OF WATERLOO

Client-side Android End

- Four types of transactions supported: deposit, withdrawal, sell and buy.
- Click any button to jump to the corresponding trade page, UI components are enabled or disabled according to the type that is selected



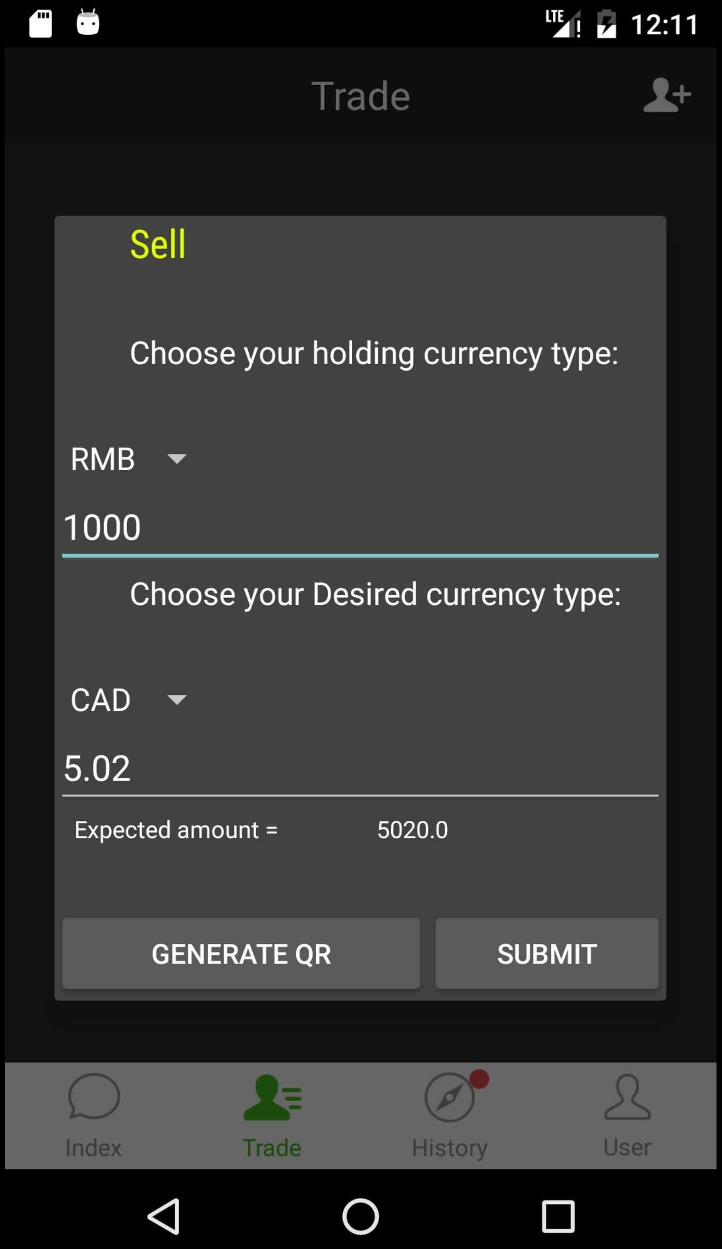
Please Select What You Want To Do:

DEPOSIT

Withdrawl

BUY (urgent)

SELL (Customized rate)



Sell

Choose your holding currency type:

RMB

1000

Choose your Desired currency type:

CAD

5.02

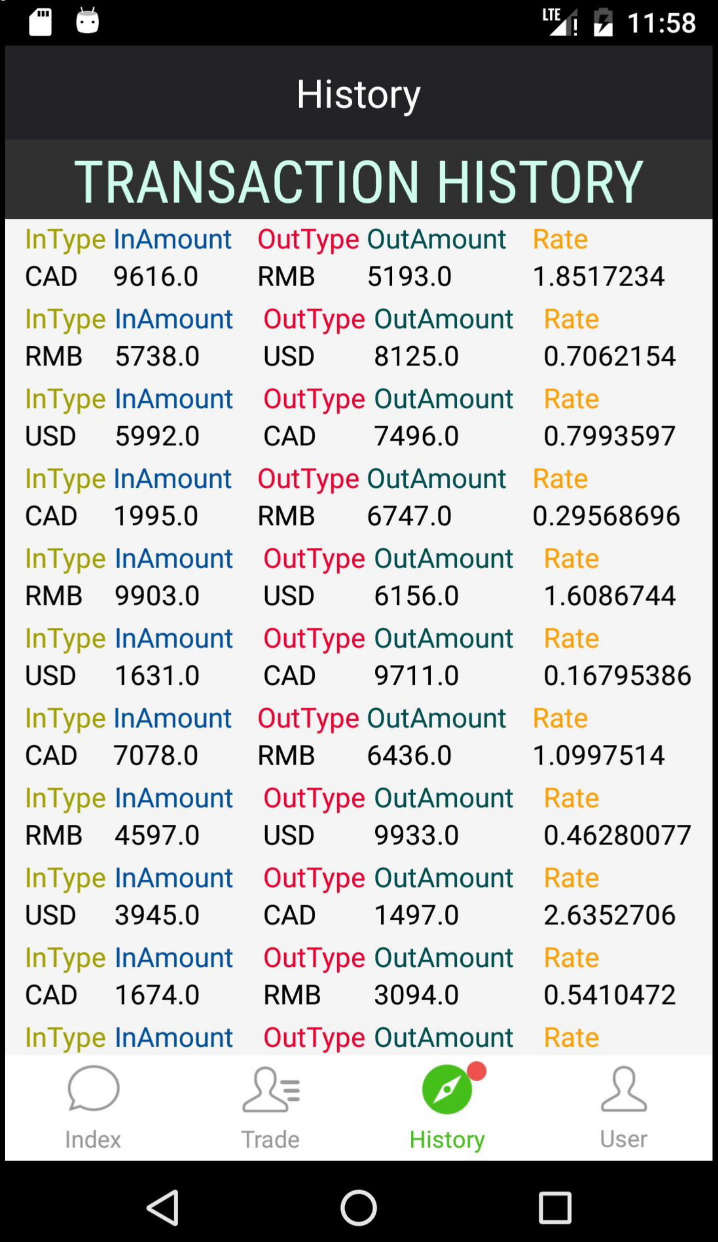
Expected amount = 5020.0

GENERATE QR

SUBMIT

Client-side Android End

- User can fetch all items of transaction history he/she involved from the server.
- User can also check his/her balances of the account and other general information.

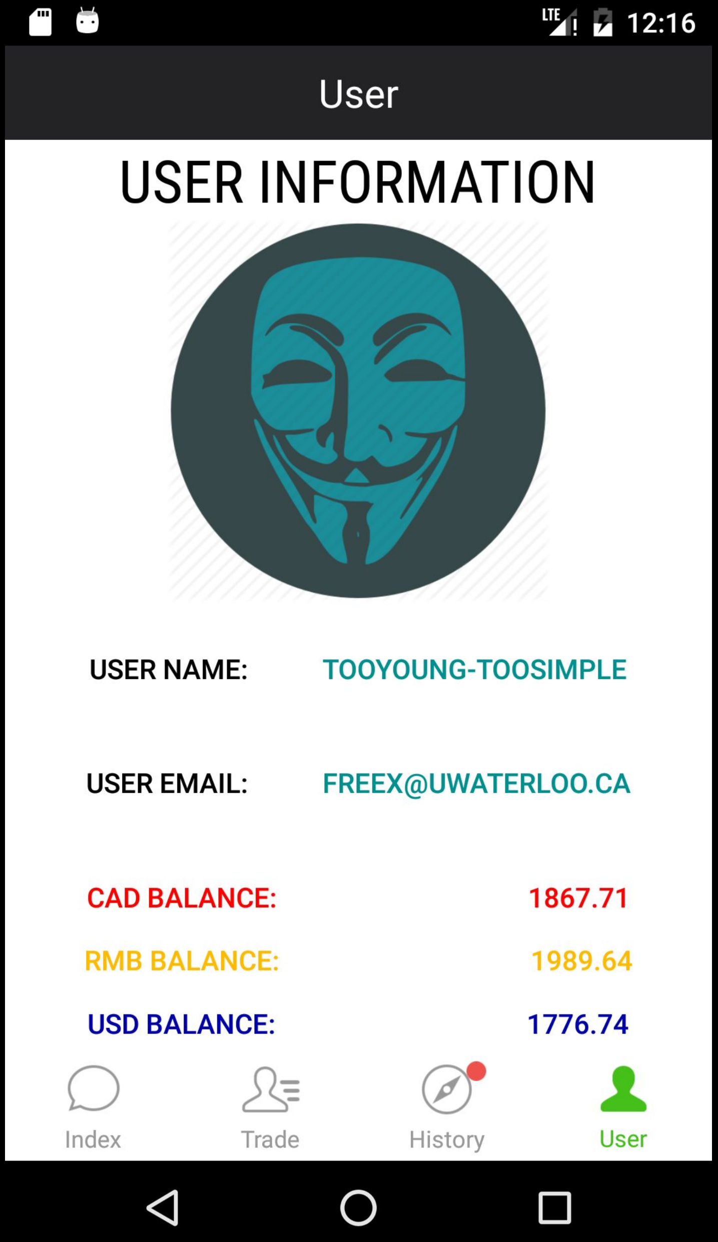

11:58

History

TRANSACTION HISTORY

InType	InAmount	OutType	OutAmount	Rate
CAD	9616.0	RMB	5193.0	1.8517234
RMB	5738.0	USD	8125.0	0.7062154
USD	5992.0	CAD	7496.0	0.7993597
CAD	1995.0	RMB	6747.0	0.29568696
RMB	9903.0	USD	6156.0	1.6086744
USD	1631.0	CAD	9711.0	0.16795386
CAD	7078.0	RMB	6436.0	1.0997514
RMB	4597.0	USD	9933.0	0.46280077
USD	3945.0	CAD	1497.0	2.6352706
CAD	1674.0	RMB	3094.0	0.5410472

InType InAmount OutType OutAmount Rate


12:16

User

USER INFORMATION

USER NAME:	TOOYOUNG-TOOSIMPLE
USER EMAIL:	FREEX@UWATERLOO.CA
CAD BALANCE:	1867.71
RMB BALANCE:	1989.64
USD BALANCE:	1776.74

Database & Data Access Object(DAO)

- MySQL + Apache DBCP + Hibernate4
- Common Object Database Operation Class
- Factory Design Pattern
- Hibernate4 encapsulated the original operations for accessing database

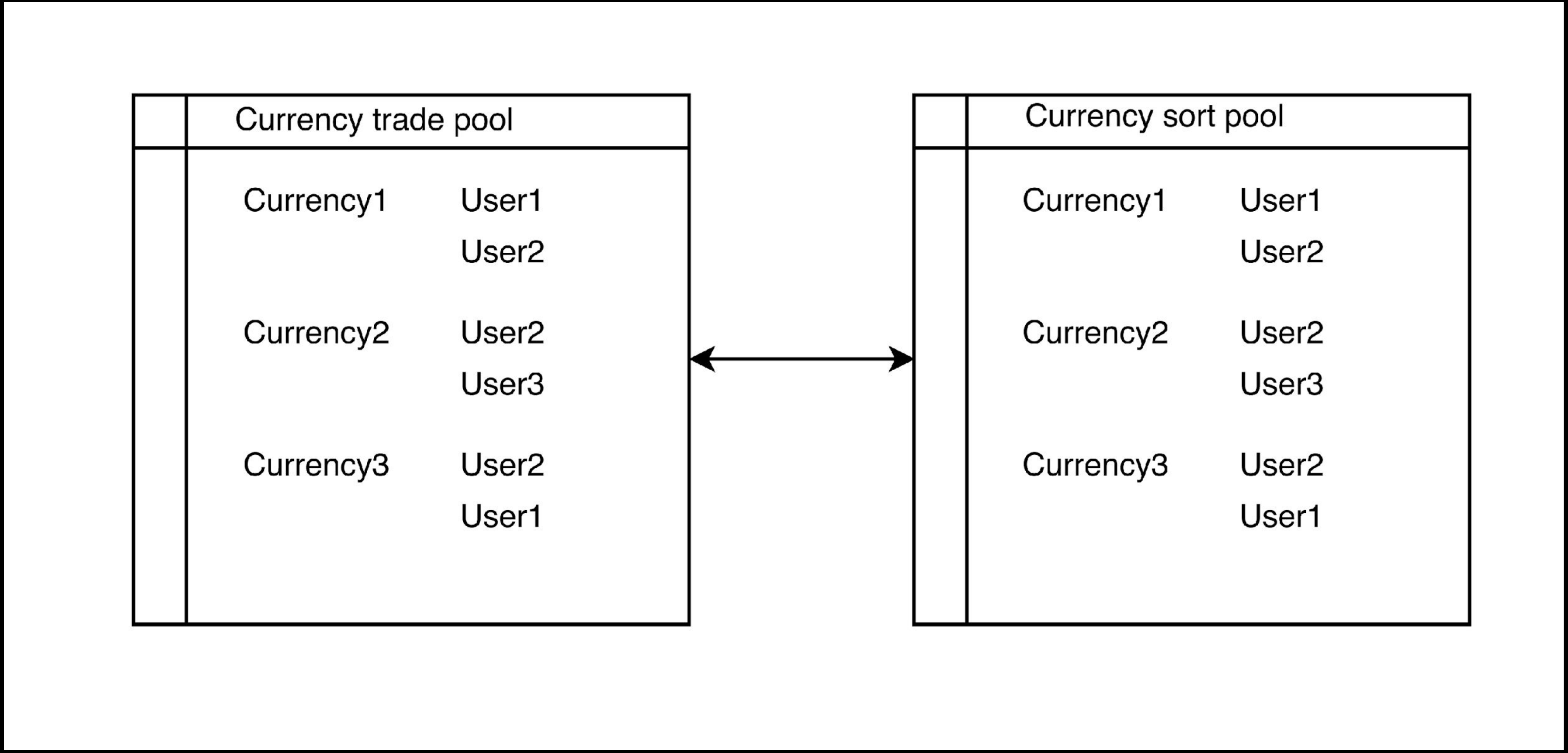
UNIVERSITY OF WATERLOO

Exchange Algorithm

- Given user desire exchange amount and currency, the algorithm quarry's database for the highest bidding exchange rate.
- The algorithm may quarry multiple trader if one trader can’t satisfy user amount.
- The total trade amount and exchange will be outputted to the user. The user will decide whether to accept the exchange rate and amount.

UNIVERSITY OF WATERLOO

Exchange Algorithm Database



Thank You !

UNIVERSITY OF WATERLOO