CMPE 275 Section 1 Fall 2020 Term Project - DirectExchange

*Last updated: 12/09/2020 (status: Published)*

Interactional remittance can be expensive and time consuming, and the often coupled need of currency exchange makes the situation worse - banks can charge a high exchange rate, plus possibly inconvenient requirements that can further slow down the exchange, e.g., requiring the remittance in currency A to stay in the account for 48 hours before it can be exchanged to current B. It is very often that the need to remit and exchange money goes both ways. For example,  Alex in the US needs to send 10K USD to his parents in India, and Bharat in India needs to send 750K INR to his sister in the US. Instead of going through the traditional costly international remittance and exchange route, Alex could send the money to Bharat’s sister, and Bharat sends the money to Alex’s parents, both domestically. This has motivated the service of P2P currency exchange, for which there are multiple service providers now. In this term project, we build such a P2P currency remittance and exchange service, with the focus on technical design, implementation, testing, and, to a lesser extent, end-to-end user experience. It is generally out of the scope of this academic project to consider regulation and legal compliance, or actual financial feasibility.

The service needs to be hosted in the cloud, and you also need to provide a client app, web app (preferred),  mobile, or both. There are no programming language or technology requirements for UI implementation. The primary language you use for server implementation, however, *must* be Java. You do not have to use Spring, but you need to exercise the principles, patterns, and methodologies that we have discussed in class, including DI, AOP, MVC, ORM, and transactions. You *must* use a relational database, e.g., MySQL, Amazon RDS (supports many popular relational databases), MSFT SQL Server, or Google Cloud SQL.

If any feature described below is unclear or ambiguous, and you fail to get a clear answer from the instructor or TA, you can use your best judgment to interpret it and add the missing details, provided that you clearly document and explain your reasoning in your project report.

Feature specification

1. User Registration and Authentication
   1. User registration info
      1. A user needs to provide the following info upon registration.
         1. Username - a valid email address. Must be unique and cannot be changed. Beside identification with the service, it is used for verification, notification, and communication purposes too. Generally invisible to other users through the service’s UI.
         2. Nickname - alphanumeric text only, with no other characters allowed. Editable. Still must be unique within the service. Visible to all DirectExchange members.
   2. You must support Oauth/OpenID for user sign-up and authentication; specifically, you need to support signing in with Facebook and Google. In addition, you ***must*** also support the option of letting the user set up a “local” password in DirectExchange with his email as the user name.
   3. Account verification
      1. No matter which registration option is used, local account with own password, or signing in with Facebook or Google, DirectExchange must send an email to the user with a verification code or link.
      2. The user needs to use that verification code or link to verify his account registration.
      3. Until the account is verified, a user cannot use any feature in DirectExchange.
   4. It is OK to use libraries like [Firebase Authentication](https://firebase.google.com/docs/auth/?utm_campaign=Firebase_announcement_awareness_general_en_05-18-16_&utm_source=gdev&utm_medium=site) to assist user auth.
   5. Bank account setup
      1. To register a bank account in DirectExchange, a user must provide the following information
         1. Bank name
         2. Country // the country that this account is operating in with free domestic remittance.
         3. Account number
         4. Owner’s name
         5. Owner’s address
         6. Primary currency
         7. Supports sending, receiving, or both
      2. A user must set up at least two bank accounts, each belonging to a different country, before he can post or accept any exchange offer.
2. Prevailing Rates
   1. DirectChange publishes the prevailing exchange rates that users can optionally rely on for currency exchange.
   2. Ideally, these rates need to be updated daily. For simplicity, you take a snapshot of the exchange rates between the minimal set of currencies as given below.
   3. Minimal set of currencies supported must include
      1. EUR
      2. GBP
      3. INR
      4. RMB
      5. USD
   4. You must have a dedicated UI page to show the prevailing rates, which can be easily accessible through a top level menu.
3. Post an exchange offer
   1. A verified user meeting the bank requirements above can post offers.
   2. An exchange offer must contain the following attributes
      1. Source country
      2. Source currency
      3. Amount to remit (in the source currency)
      4. Destination country
      5. Destination currency
      6. Exchange rate
         1. Exchange rate is defined by the amount of the destination currency that each source currency is worth.
         2. The exchange rate can either be specified as PrevailingRate, or a specific real number.
      7. Expiration date
      8. Whether to allow counter offers. Default to true.
         1. When yes, another user can propose a different remit amount at the same exchange rate. The proposed mount must be within 10% of the original amount, up or down.
      9. Whether to allow split exchanges. The default should be true.
         1. A split exchange allows more than two parties to participate in the exchange.
         2. Your service must at least support 2:1 split exchanges, where A and B jointly take C’s offer.
   3. An offer can be changed by the owner if no one has accepted it or made a split proposal.
4. Browse and filter offers
   1. A verified user can browse the active offers - he does not have to post an offer first before he is able to browse offers.
   2. Each offer must show all the attributes as defined above, the owner’s nickname, plus the reputation stars of the owner.
   3. When clicking on an offer, an offer detail page should be shown, which contains all the offer’s attributes, the user name of the owner, and the reputation stars of the owner.
   4. Clicking on the reputation stars of a user should show the user’s transaction histories, including aborted and completed transactions. For each transaction, only the first two letters of other parties’ user name should be shown.
   5. A verified user can filter the offers by source currency, source currency amount, destination currency, and destination currency amount.
   6. Paging should be supported, with each page up to 10 offers.
5. My offers page
   1. A user should be provided with a view to see his owner offers, with open offers listed first.
   2. Each offer must show all the attributes as defined above, including offer status as well (open, CounterMade, InTransaction, fulfilled, or expired). The state CounterMade means its owner has used this offer to make a counter offer from other offer(s).
6. Offer acceptance or counter offer
   1. A user can directly accept an offer, which brings the exchange into transaction mode. Or,
   2. If the offer allows a counter offer, the user can propose a counter offer with a different remit amount, using the original offer’s source currency. The proposed amount must be within 10% of the original. This would notify the user who made the original offer, which can either reject or accept the counter offer. If the counter offer is accepted, it enters the transaction mode.
7. Auto matching and offer split
   1. In the My Offers page, a user must be provided with a button (or any other easy means) to find matching offers for the current offer selected.
      1. In the matching offers view, we should show single matches first, then split matches.
      2. The user should be given the option to exclude split matches through a check box or filter dropdown menu.
      3. The matching offers (within its own bucket of single or split) should be sorted by the absolute value of the amount difference percentage, and the percentage, +5%, -3%, etc.
      4. Suppose the current offer is A, split matching offers should include all B and C, such that
         1. A ≅ B + C, or
         2. A + B ≅ C

In case 1, A should be within the range of -10% & +10% of (B+C).

In case 2, A should be within the range of -10% & +10% of (C-B).

1. If the user picks an exact matching offer (split or not), he can accept it, which brings all involved offers into the transaction mode.
2. If the user picks an approximate match, he can either accept the offer by changing his own offer into an exact match, or propose a counter offer. The system will present the counter offer to the other party of a single match, or the one with the bigger amount among  of the other two parties, if it’s a split offer. In the case Alice makes a counter offer to somebody else (say, Bob) based on her current offer A, A’s status becomes CounterMade, and A should either disappear in the browsing/filtering pages by others, or show up differently (e.g., with a status label, different color, etc. to make it clear that somebody else other than Bob cannot directly accept this after, Bob accepts or rejects this counter offer, or the counter offer expires).
3. Every counteroffer automatically expires if not accepted or rejected within 5 minutes. (The 5 minutes is too short in real-world, but convenient for testing.)
4. Transaction
   1. Once the parties enter the transaction mode, each gets a notice to fulfill the translation.
   2. Each involved party takes the action to transfer the according amount of money to DirectExchange.
   3. After getting all the funds from all the parties, DirectExchange distributes them to the according receiving account.
   4. All parties must transfer the money to DirectChange within 10 minutes, otherwise the transaction is cancelled.
   5. Offers in transaction modes, similar to countered and fulfilled/expired, cannot accept counter or be accepted like open offers.
5. Service fee
   1. For simplicity, we charge a service fee of 0.05%. (More details to be added)
   2. Service fee is charged by withholding 0.05% of the amount to be deposited into each receiving account. Suppose A and B are doing direct exchange, A is supposed to send x USD to B’s receiving account, and B to send y INR to A’s receiving account, then A and B’s receiving account will only get 99.95%\*y INR and 99.95\*x USD respectively.
6. Messaging and notification
   1. When user A is doing business with user B within DirectExchange, A can directly message B through within DirectChange. For simplicity, it is OK to let all actual messaging and notification be transported through email. For example, when Alice messages Bob through DirectExchange, it ends up sending an email to Bob through the system.
7. Reporting
   1. User transaction history report. The app must provide a Transaction History screen to report all the current user’s finished transactions for any given calendar month within the last 12 months, including the current month. In this screen, we should show date/time, source currency/amount, exchange rate, destinationation currency/amount (before service fee),  and service fee for each translation, and the total for each metric for all the reported transactions.
   2. System financial reports.

[Note: For this feature, it is fine to assume that the current user is an admin and has access to the transaction history across the system, as the reports are for the system] The app must provide a System Financial screen to report the total number of completed transactions, total remitted amount is USD, total service fee, and total number of uncompleted transactions for any given calendar month within the last 12 months, including the current month.

1. A three person split transaction is still considered as a single transaction.
2. The total remitted amount is calculated in USD, using the system’s prevailing rate *at the time of the transaction*. For any transaction, each transferred amount will be counted once for. For example, in a two-party transaction where A sends x USD and B sends y INR, the total amount will be x + y\*(the relevant prevailing exchange rate at the time of the transaction).
3. An uncompleted transaction means the parties enter the transaction mode, but not all parties transfer the money to DirectExchange within the required time.
4. Reputation ~~and Reviews~~
   1. There are six reputation ratings, N/A, 1-5 stars. N/A is for a user that has never entered transaction mode.
   2. For any transaction, if a user did not transfer the money within the given time limit, the transaction is called at-fault transaction.
   3. The rating # of star is calculated by:

Round ((1- (# of at-fault transactions) / (# of entered transactions)) \* 4) + 1

For example, Alice entered 10 transactions, and has 3 at fault, then her

 reputation is  4 stars. If she has 7 at fault instead, it becomes 2 stars.

1. Once a transaction is cancelled or completed, the according user’s reputation is updated right away, and subsequent requests for showing reputations should reflect these changes.

~~Bonus Features~~

Please note this is the end of functional specification.

User Interface and Experience

In addition to the functional features, we want the app to have proper and intuitive UI/UX design, navigate naturally, run robustly and smoothly.

Grouping

This term project is group based, with group size up to four people. Once the project plan is submitted, group membership cannot be changed.

Source Code Management

You are recommended to use a Source Control Management (SCM) system to manage your team’s source code. This can be a private Bitbucket repository or your local git. During the grading of the term project, you may be asked to provide commit history or any other document to help evaluate each team member’s contribution.

Cheating Policy

Your app must be built by yourself, and cannot be based on the code base of any existing app. If you used any code not written by yourself, it must be clearly documented in your README.TXT file, unless it is part of publicly available libraries. *If your app is already used to serve the requirements of any other class, it will not be accepted by this class*. In the case any form of cheating is confirmed, you will get an F grade for this class.

Deliverables and Grading

The project is worth 31 points in total. The actual *due dates* of the deliverables will be specified in Canvas.

**Group Formation (1 point)**

Please submit your project name and group formation info through Canvas.

**Project Presentation and Demo (5 points)**

To be presented in class.

* The presentation should cover introduction, high-level design, and major features with screenshots. Time limit: 3 minutes.
* You must also do a live demo. The guideline for how to do demos is to be announced. Time limit: 3 minutes.
* Grading will be based on the successfulness of the demo, the content and clarity of the slides, and the delivery of the presentation

The presentation slides must be submitted through Canvas as a PDF file.

**Project Report (5 points)**

The report needs to cover the following topics.

1. Motivation and introduction of your app
2. High level and component level design
3. Technology choices
4. Description of features with final screenshots
5. Testing plan executed and results
6. Lessons learned and possible future work

You are recommended not to exceed 20 pages, but you will not be penalized just because the report is too long or too short, as long as the level of coverage for the required topics is reasonable and clear. The report must be submitted through Canvas as a PDF file.

**Project App (20 points)**

Note: the instructions for submission are still *subject to change*. Please refer to submission instructions in Canvas.

1. You must submit all your source code / resource files through Canvas
2. Features correctness, stability, performance, choice of technology and implementation are worth 18 points; **all** grading will be done through your UI - **no** partial grades will be given for any feature not exposed or untestable in UI.
3. User experience is worth 2 points
4. You need to keep your app live for at least a week before we finish grading
5. README.TXT, including
   1. The names, email IDs, and students IDs of the members
   2. The URL to access your app
   3. Any other instructions necessary for the TA to grade the app
   4. Build instructions