CS/SE 6360 – Term Project: Fall 2021

I. OVERVIEW

Motivated by the recent fluctuation in Bitcoin prices, a good friend of yours would like to set up a local shop that can buy/sell Bitcoin for investors in Dallas. She asked your help to develop a web based Bitcoin trading application. In particular, she wants you to create convenient and easy-to-use software for Bitcoin traders who are trying to buy and sell Bitcoin for their clients who do not understand Bitcoin dynamics well. In addition, she wants you to enable more sophisticated clients to invest directly using the web based application. To help your friend, you offer to develop a web based software system called BTS (Bitcoin Transaction System) that leverage the relational DBMS technology for data storage and querying.

II. PROJECT DESCRIPTION

Based on your interactions with your friend, you gathered the following pieces of information:
BTS will be used by the traders to buy and sell Bitcoin for their clients and clients to directly buy and sell Bitcoin.

- Each client has a unique client id generated by the system, a name (first and last), a phone number, a cell-phone number, an e-mail address, and an address (including street address, city, state, and zip code).
- For regulatory reasons, it is important to retrieve the city and zip code information for each client easily.
- Each client is assigned to one of two different levels based on his or her past transaction volume. Once a client makes more than $100K in trades (buy or sell) in the previous month, the client is classified as a "Gold" customer and is charged a different commission rate for the next month's transaction. Otherwise, the client is classified as "Silver". This classification will be updated monthly.
- When a client wants to execute a transaction, the client logs into online system and specifies the amount of Bitcoin he/she wants to buy or sell. Of course, system needs to verify the client's identity by asking the client to enter a password, and check whether he has enough fiat currency (i.e., USD) in his/her account. If the client wants to sell Bitcoin, the system should automatically check if the client has enough Bitcoin stored by the company to satisfy the client's request.
- The client also needs to specify whether he or she wants to pay the commission for the transaction in Bitcoin or fiat currency. Based on the client's choices, the system places the order. The system calculates the transaction commission based on the client's classification. If the transaction fee is paid in Bitcoin, the system automatically adjusts the amount of Bitcoin left in the customer account. On the other hand, if the customer chooses to pay the commission in fiat currency, the system must automatically compute

the fee based on current Bitcoin prices. For example, see [1] on how to get this information dynamically in your app. This is critical because Bitcoin prices can fluctuate a lot.

- The value of the transaction (e.g., the value of the Bitcoin bought or sold), the date of the transaction, the commission paid, and the commission type should be stored separately for each transaction.
- From time to time, clients will transfer money to so that they can buy more Bitcoin. For each payment transaction, you need to store the amount paid, the date, and the information related to the trader who accepted the payment.
- In some cases, traders may want to cancel certain payment and Bitcoin transactions. Although the system should allow such cancellations, logs should be stored for such cancelations for auditing purposes.
- Also, you need to allow a trader to 1) issue transactions for a client, 2) search the client history for specific client based on name, address and etc.
- You should also provide an interface for the manager that can give aggregate information for daily, weekly and monthly total transactions based on the dates entered by the manager.

Please note that this document does not claim to be complete. Many design issues need careful analysis. Some of these include: how user history will be stored, what attributes entities should have, how users are upgraded to "gold" category. Yet, given this description, filling in the blanks should not be too hard.

III. Project Deliverables

PRELIMINARY STEP. Form a group of size 4-5 (no more than 5, no less than 4) and notify the TA before October 30th.

STEP 1. (25%) Draw the ER/EER diagram, and then convert it to a relational schema. Indicate primary keys, foreign keys and any other constraints. Follow the notation used throughout the textbook. Clearly specify any assumptions you make and your rationale.

STEP 2. (25%) Create database tables according to the relational schema in Step 1 and populate them with a reasonable number of tuples. Also include a series of 'drop table' queries that will drop all tables created. Submit the SQL commands as a **TXT** file.

Any line that is not part of an SQL command should be commented out. In other words, you need to submit an SQL script. Please notice that although SQL is a standard, query syntax differs by DBMS vendors. Your script should run on the campus SQL Server without any errors.

STEP 3. (50%) Design and implement a program for end-users. This web-based app should connect to the database and provide the functionalities discussed in the project description. The web page for this program should be simple and easy to use.

You may use any programming language and web programming framework you wish to use.

---

[1] https://www.coindesk.com/api/

Submit a single ZIP file containing the following items by **December 1st Midnight (CST):**

(1) Source code folder named source,
(2) A report that discusses the following topics: the design in Step 1, the software architecture, and overview of the code. Please put the ER diagrams in the appendix and refer them in your report. Other than the appendix that contains ER diagrams, the main discussion of the report should not be more than 10 pages. Please use single space, one inch margins, and Times New roman 12 fonts while writing your report.
(3) Readme file that describes in detail how to set up and run your software and links to all the dependencies.
(4) Sql script described in Step 2 above.

A demonstration schedule will be created for you to present your work by the TA.

IV. SUBMISSION GUIDELINES

Please read the information below carefully. Compliance with these guidelines is a vital part of the grading process.

- Please pay attention to submit your files in proper format, specified for each step. Only one submission per group is sufficient. Include names and NetIDs of all members in every file.
- Please do not submit at the last minute. Your clock most possibly is not synchronous with the elearning system clock. Unless there is a good reason (i.e., elearning system crash), only online submissions will be accepted. In order to encourage submitting partial work, multiple submissions are allowed. You can always retrieve your files, make necessary changes and re-submit until the due date.
- Please note that the TA is *not* a system administrator. Please direct all questions related to connection problems to department resources to cs-tech@utdallas.edu .
- Source code will be tested using software plagiarism tools including the past submissions. Plagiarized work is very easy to detect, and all necessary measures will be taken to identify and penalize such behavior.

GOOD LUCK!