CS 6360 – Term Project: Fall 2022

I. OVERVIEW

Motivated by the recent popularity of non-fungible tokens (NFTs[1]), a good friend of yours would like to set up a local shop that can buy/sell NFTs for investors in Dallas based on Ethereum blockchain[2]. She asked your help to develop a web based NFT trading application. In particular, she wants you to create convenient and easy-to-use software for NFT traders who are trying to buy and sell NFTs. To help your friend, you offer to develop a web based software system called NTS (NFT 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:
NTS will be used by the traders to directly buy and sell NFTs.

- Each trader 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).
- Each trader is assumed to have unique Ethereum address used for trading NFTs.
- For regulatory reasons, it is important to retrieve the city and zip code information for each trader easily.
- Each trader is assigned to one of two different levels based on his or her past transaction volume. Once a trader 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 trader is classified as "Silver". This classification will be updated monthly.
- Each NFT has unique token id (see ethereum uint256 type), address of the Ethereum smart contract used for keeping track of the NFT, and the name (e.g., CryptoKitties).
- When a trader wants to execute a transaction, the trader logs into the online system and specifies the NFT smart contract address, and the specific token the trader wants to buy from that address. Of course, system needs to verify the client's identity by asking the client to enter a password, and check whether the trader has enough fiat currency (i.e., USD) or Ethereum in his/her account.
- The trader should be able to see the NFTs he/she owns and their current market price in USD and Ethereum. If the trader wants to sell a NFT, the system should check whether the trader already owns the NFT that he/she is trying to sell.
- The trader also needs to specify whether he or she wants to pay the commission for the transaction in Ethereum or fiat currency. Based on the trader's choices, the system places the order. The system calculates the transaction commission based on the trader's

[1] https://www.investopedia.com/non-fungible-tokens-nft-5115211
[2] https://ethereum.org/en/nft/

classification. If the transaction fee is paid in Ethereum, the system automatically adjusts the amount of Ethereum 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 Ethereum prices. For example, see [3] on how to get this information dynamically in your app. This is critical because Ethereum prices can fluctuate a lot.

- The value of the transaction in Ethereum, the date of the transaction, the commission paid, the commission type, the NFT address, NFT toked id, the seller Ethereum address and the buyer Ethereum address should be stored separately for each transaction.
- From time to time, clients will transfer money/Ethreum to their account so that they can buy more NFTs. For each payment transaction, you need to store the amount paid, the date, type of the payment and the id related to the trader who submitted the payment, and the payment address (e.g., bank account number of fiat currency, Ethereum address for Ethereum payments). In your application, you may assume that all the payment transactions will be successful.
- In some cases, traders may want to cancel certain payment and NFT transactions. Although the system should allow such cancellations up to 15 mins after the transaction submission, logs should be stored for such cancelations for auditing purposes.
- You should allow a trader to search his/her transaction history.
- 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 15th.

STEP 1. (10%) 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. (10%) 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 MySQL or Postgresql.

[3] https://docs.cloud.coinbase.com/sign-in-with-coinbase/docs/api-prices /

STEP 3. (70%) 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.

Submit a single ZIP file containing the following items by **December 5th 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 five pages. Please use single space, one inch margins, and Times New roman 11 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 to 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. Please 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 very 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.
- Source code will be tested using software plagiarism tools including the other submissions. Plagiarized work is very easy to detect, and all necessary measures will be taken to identify and penalize such behavior.

GOOD LUCK!