Term Project Report for CS6360.004, Fall 2022, Dr. Murat Kantarcioglu

Li, Desong [DXL180019]
Lumpkin, Michael Zayne [MZL190000]
Potdar, Siddhi Mahesh [SMP220001]
Sharma, Tanya [TXS220004]
Singh, Priya [PXS220067]

Our NFT Transaction System (NTS) was crafted over the course of a semester, to facilitate efficient trading while demonstrating both theoretical and practical knowledge of databases, particularly relational databases using the Structured Query Language (in our case, SQL). Our NTS takes the form of a website. Obviously, the basic web technologies of HTML, CSS, and JavaScript have been used; these have been augmented with the use of MySQL for our database management system (DBMS) and Flask, a simple Python web framework which is able to generate dynamic, responsive pages from basic HTML/CSS templates. The MySQL database is hosted on a cloud server through Amazon Web Services.

Our database schema includes eight tables of varying degrees of complexity and with various relationships between them. These tables, broadly, represent users, addresses, NFTs, and various types of transactions both within and across the boundaries of our system. Users can add new NFTs to the marketplace, make offers on NFTs, accept offers, view their owned NFTs, manage account information, add outside currency, and view their transaction history. Administrators are granted access to a manager dashboard, which shows the number of transactions and the total commission received across all transactions within several timeframes (daily, weekly, monthly, and custom).

I: NFT Transaction System (NTS) Relational Database Schema

1: Relation Schema

Referring the ER Diagram [1], the relational schema is as follows:

Trader(client_id, first_name, last_name, phone_number, email, cell_no, level, fiat_balance, ethereum _balance, ethereum_address, login)

Address(client_id, street_address, city, state, zip_code)

Payment(payment_id, client_id, payment_address, payment_type, date, amount_paid, cancelled)

NFT(token_id, name, ethereum _address, client_id)

User(login, password)

Manager(manager_id, login)

Transaction(transaction_id, ethereum_value, ethereum_buyer_address, ethereum_seller_address, token_id, ethereum_nft_address, commission_type, commission_paid, date)

Offers(nft_id, ethereum_balance, buyer_id, seller_id)

II: Software Architecture & Code Overview

Technologies used:

- Python
    - Flask
    - MySQL Connector

- HMTL, CSS
    - Bootstrap

- JavaScript, AJAX

- MySQL Workbench
    - AWS

Web page functionality:
Referring to the Website Flow Diagram [2], the page functionality is as follows:

Signup Page

- The trader shall be able to enter the required information displayed on the registration page.
- The application will save the new trader into the database with a unique client id and Ethereum address.

User Login Page

- The trader shall be able to login to their account by entering their username and password.
- The application will search the user table in the database to verify that the entered username and password match.
    - The application will push this trader in session and redirect the trader to their home page.

Profile Page

- The application will display the trader information.
- The trader shall be able to edit their account information. i.e., personal, address, balance, and login.
    - Edit User Info, Address, and Login Info
        - The trader shall be able to change any of their personal information.
        - The application will update the change in database and reflect it on profile page.
    - Edit balance
        - The trader shall be able to transfer money/Ethereum to their account by entering the type of payment, payment address, and amount of money/Ethereum they want to transfer.
        - The application will insert a new payment history into the payment table, with a unique payment id and a user id that references the current trader. The trader's balance will be updated accordingly.
    - View Payment History
        - The application will display the payment history.
        - The trader shall be able to cancel any payments that last no longer than 15 minutes.

    - The application will update the statue of any canceled payment in the payment table and update the trader's balance accordingly.

Manager Dashboard

- The manager shall be able to view aggregate information for daily, weekly and monthly total transactions based on the dates entered.
- The application will query the transaction table in the database to find the total commission paid and the total number of transactions based on daily/weekly/monthly or date range.

Home Page

- The trader can see a list of available NFTs to invest in, with their prices and names.
- A buy button routes to the specific NFT the trader is interested in buying, and the transaction is taken forward from there.
- A navigation bar at the top helps to navigate through the webpage, allowing access to different functionalities of the trading platform.

Owned NFTs Page

- The page shows a list of NFTs owned by the trader, with their names and market value.
- A functionality to check available offers on the trader's NFT is also provided via this page.

Buy Page

- The trader shall be able to view the information such as its price in USD, Ethereum Value and the name of the seller for the selected NFT.
- The application will query the NFT table in the database to fetch the required tuple.
- The application makes an api call to min-api.cryptocompare.com/data/price?fsym=ETH&tsyms=USD to fetch the current exchange rate.

User Validation Page

- Right before checking out to buy the NFT, to help in protecting sensitive data, the user's credentials are validated once again.
- If erroneous credentials are provided the user is routed back to the home page.

Sell Page

- The trader will be able to add new NFTs to sell and will be able to set a starting market value for his NFT.
- The application will query to insert a new NFT data row which will also include the owner's Ethereum address and will generate a unique token id for this NFT.
- The application will then show the owned NFT page, including the newly added NFT.

Checkout Page

- The trader will be able to see if the transaction made to buy an NFT was successful or not, in the payment mode of their choice.
- The trader will also be able to see the final amount of being deducted from their account to buy the NFT of their choice.
- A functionality to route back to the home page is also provided.

Offer Page

- The trader will be able to see all the offers made for the NFTs he owns along with the offered price and the address of the buyer who made the offer.
- The trader will be able to sell his NFT by accepting the offer and the NFTs ownership details will update accordingly

Transaction History Page,

- The trader shall be able to view all the transactions done by him/her.
- The trader can also cancel the transaction within 15 mins of the transaction.

Gold-Silver Level Update

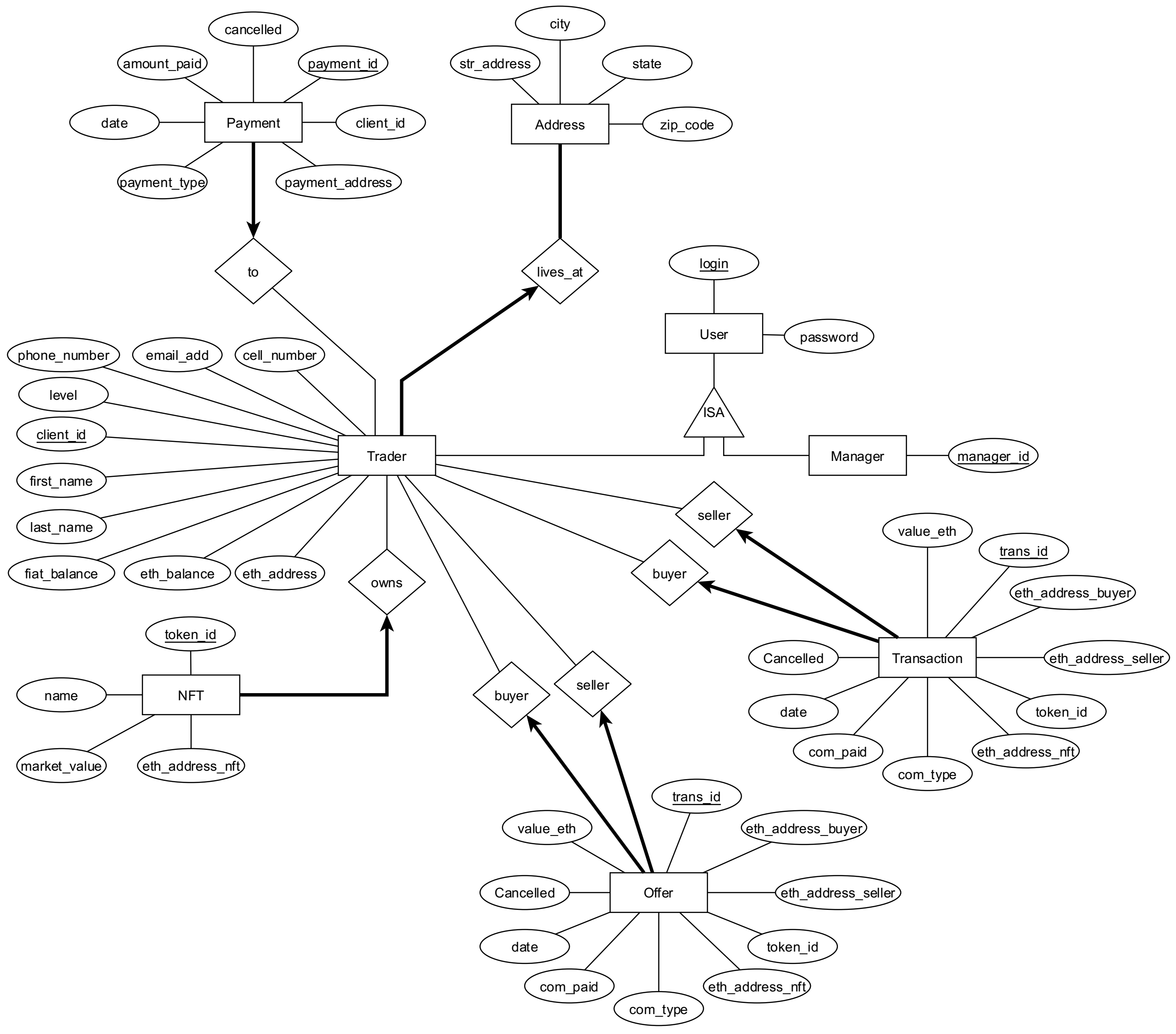
- A MySQL Procedure has been added namely ChangeLevel() which checks the total transaction made by a trader in the form of buy or sell in the past month and update the LEVEL from Gold to Silver or Silver to Gold according to it

Error Page

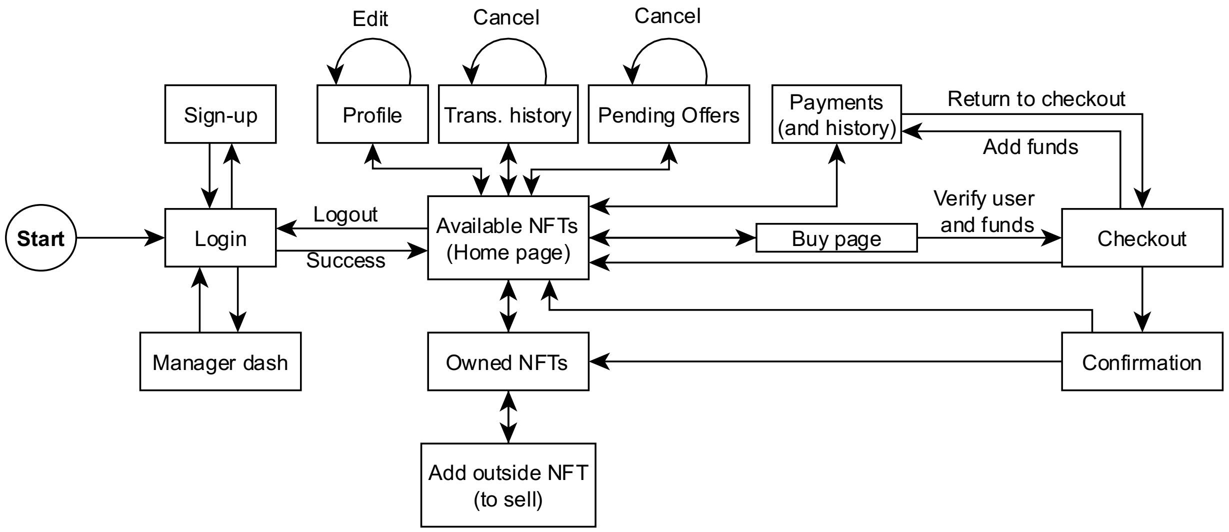
- A generic error page that is shown in case of exceptions.

Appendix

[1] ER Diagram



[2] Website Flow Diagram/Software Architecture



References:

[a] https://www.tutorialspoint.com/how-to-store-usernames-and-passwords-safely-in-mysql-database [a] https://www.tutorialspoint.com/how-to-store-usernames-and-passwords-safely-in-mysql-database

[b] https://getbootstrap.com [b]

[c] https://fontawesome.com [c]