Erik Jonsson School of Engineering and Computer Science

Department of Computer Science

CS 6360 – Database Design – Fall 2022 Project Final Report

By

Shreeprasad Anant Sonar – SXS210482

Kevan Apurvabhai Mehta – KAM210015

Varun Potluri- VXP200035

Satish Varma Udimudi – SXU220002

Rakesh Jampala – RXJ210049

Under the Supervision of

Dr. Murat Kantarcioglu

Professor

Zihe Song

Teaching Assistant

1. Overview

Project description

In this project, we implemented a web based NFT trading application that can support buying and selling NFT's with ease. The trader can choose to buy or sell the NFT's by paying a minimal commission to the system. From time to time, trader can transfer money/Ethereum to their account so that they can buy more NFTs. In this system traders can cancel certain payment and NFT transactions, system will allow such cancellations up to 15 mins after the transaction submission. For this system we have a manager who can see what all transactions performed by traders for daily, weekly, and monthly total transactions based on the dates entered by the manager. For the front end we are using a typescript based Angular Framework and for backend we are using a python-based application framework called flask. For storing traders' data and transaction data we are using MySQL database.

2. Requirement Analysis

Hardware:

1. Processor: Core I3/I5/I7
2. RAM: 4GB or more
3. Hard disk: 160 GB or more

Software:

1. Operating System: Windows 7/8/10
2. Program language: TypeScript, Python, HTML, JavaScript, SCSS
3. Web Application frameworks: Flask, Angular
4. VS Code

Required modules:

1. Flask api's, PyJWT, Angular dependencies, Angular DevDependencies, MySQL

3. Team members work distribution

Name	Net ID	Work distribution
Shreeprasad Anant Sonar	sxs210482	• Devised Frontend Architecture and implemented services for user authentication and data retrieval, Implemented Authentication Guards for restricted access to protected information • Developed Homepage, Login, Signup, Trade/Buy, Header Components with reusable user-friendly UI while contributing to the development of Dashboard component, Sell component • Established all the API calls needed for communication between frontend and backend • Bug fixing and code refinement
Kevan Apurvabhai Mehta	kam210015	• Implemented all backend Api's to implement all requirements given in the project report which include getting all NFT details, Trader NFT details, Trader details, getting ETH value dynamically, storing trader's NFT and fiat transaction details in the database, calculating the membership based on the trader's transaction, showing trader transactions history, getting all the transaction logs based on the given dates by manager and canceling the transactions performed by the trader. • Bug fixing and code refinement
Varun Potluri	vxp200035	• Responsible for design and structure of the application • Acted as a project lead/coordinator • Designed the database and developed appropriate SQL quires to create, view, insert and drop the table required by the project. • Contributed to the development various front-end components like trader's dashboard, buying, selling and cancel pages. • Contributed to the development of various backend api's like cancel, buying and selling.
Satish Varma Udimudi	SXU220002	• Designed and Developed NFT Dashboard for trader, traders NFT Selling page, Transaction Cancel page and Manager Dashboard pages in the frontend. • Bug fixing and code refinement
Rakesh Jampala	RXJ210049	• Bug fixing and code refinement
Common Contributions		• Build project basic structure • Drafting Report

		• Contributed and participated in various brainstorming sessions to design and implement all the given requirements.

4. Relational Schema, Assumptions, Implementation, and code review

1. **Relational Schema (Appendix A)**
2. **Assumptions**

   To develop this NFT Trading system we have made some assumptions. The assumptions are as follows:
   - We have assumed that an NFT can be owned by only one individual at a time. Therefore, there is no fractionalization of the NFT ownership.
   - When the trader wants to sell a particular NFT, he/she can sell the NFT to the trading company and the trading company will pay the appropriate price to the trader and the system will display this NFT in the marketplace so the other trader who are interested in buying this can buy the NFT.
   - When a trader buys a particular NFT we store this transaction log in the database table NFT_TRANSACTION. When the user sells the NFT we update the same transaction log status. By this way we limit the usage of database storage. Even though the status was updated to sold we still calculate this transaction to determine the membership type.
   - We are assuming that all Fiat/ETH transactions are successful.
   - In this project to update the status of each transaction we are using different words. Those words are defined here:
     - Bought and Success – This mean that the trader bought the NFT, and the transaction was successful
     - Sold and Success – This means that the trader sold the NFT, and the transaction was successful.
     - Bought and Cancel – This means that the trader bought the NFT and cancelled the transaction within 15 minutes. There are stored for Audit purposes.
     - Sold and Cancel – This means that the trader initially sold the NFT and cancelled this transaction within the 15-minute window. We assumed that "Sold and Cancel" status is equal to "Bought and Success". We updated it to "Sold and Cancel" because if we update it to "Bought and Success" the system this it was bought recently and there is a chance to cancel the transaction. So, we assumed this assumption.
3. **Implementation**
   - First, we have the landing/home page where a trader can sign-up or login into his existing account. If the trader is new to the system, he sign-up by providing his details that are listed under TRADER and ADDRES tables (see Appendix A). After the trader sign-up, the system will assign a random id which will act as a primary key when combined with email. Now trader can login to the system.
   - After logging in, on the dashboard we will display all the NFT's that trader own, the fiat amount he has in his trading account, the Ethereum count he/she own, total value of the Ethereum as per the mark and the current market value of the Ethereum. If the trader wants to add more fiat currency or ETH to his/her account, they can add by clicking on "add more" button the dashboard. To add more fiat/ETH's the trader need to specify either he/she wants to add ETH or Fiat, then the trader need to provide the count of ETH or value of fiat in USD, account number and routing number.

- After the trader submits the transaction, the system logs the transaction into the database, and we update the values in trader table. Since the trader now has enough Fiat/ETH in his/her account they can buy NFT's available in marketplace. To buy NFT's the trader can click on "add more NFT" button under trader NFT's or they can navigate to trading page from the header of the dashboard. If the trader chooses a particular NFT to buy, the system asks the trader how they want to pay the commission.
- When the trader specifies, frontend sends the request to backend api where it calculates the commission based on traders' membership type and check if the trader has enough ETH/fiat to buy and pay the commission. By checking and confirming, the system processes the transactions and logs the details into the table NFT_TRANSACTION with all details of the transaction (See Appendix A). The trader can cancel the payment within 15 minutes from the time of submitting the transaction. If trader cancel the transaction, we will reflect the changes such as traders ETH count, commission paid, transaction status and NFT table so that the system can again display the NFT in marketplace.
- If the trader didn't cancel the transaction, they can sell the NFT anytime. If the Trader wants to sell a particular NFT he/she can navigate to selling page by clicking "sell NFT" button. After navigating to the sell page, they can specify the NFT they want to sell, and the system will check if the trader own the NFT. After verifying the system process the transaction by updating the traders ETH count, seller address of the NFT and owner of the NFT. Even now the system should allow the trader to cancel this transaction and update the status of the transaction and reflect other changes being made when we performed sold transaction.
- The trader can see all the transactions they performed.
- The manager can view all the transaction logs by specifying form and to date.

4. <u>Overview of the code</u>

a. <u>Frontend</u>

- Frontend of the project is built/developed on Angular Framework using Node.js runtime environment. The name of the project folder is "NFT-transaction-system" with all the code in "src/" folder. The "src/" folder contains primary HTML, TS, SCSS files with other components in "src/app/" folder containing routing module for all the routes to specific component, app.component has selectors for header and router outlet, auth and data services and authentication guard. Auth and data services contains api calls consumed by all the components in the app folder for validation and data retrieval purposes.

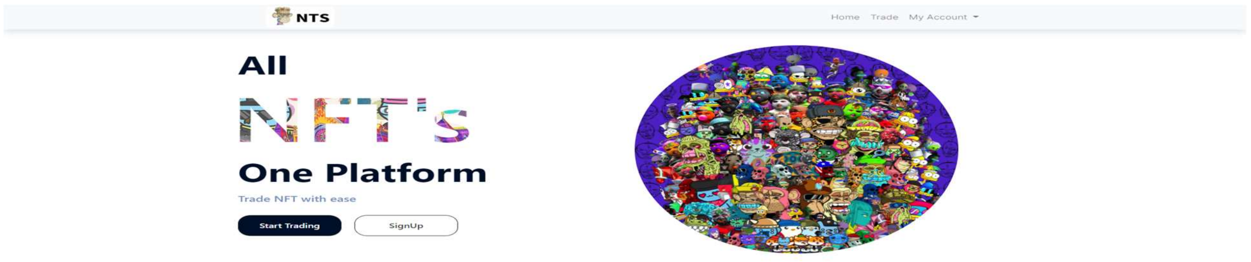
b. <u>Backend</u>

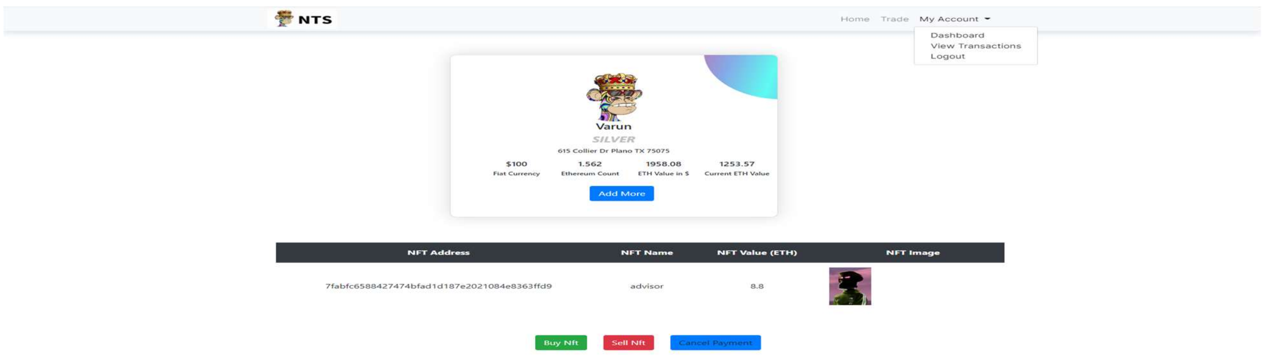
- Backend of the project is developed using Flask (a Python Framework). The "app.py" file is located inside Project folder called "Backend". It contains the entire project code. This file contains all the import statements for importing various libraries, APIs that have been built for all the necessary functionalities, dynamically retrieving values of ETH and other necessary logics are included in the file.

c. <u>Database</u>

- For the storing all the data logs generated we used MySQL. The script files are in the subfolder called "Database" in the parent folder "NFT-transaction-system". It contains script used to create database and associated tables, script to drop the tables and data such as commission rates, NFT's for the marketplace and manager credentials.

5. Usage of the application with screen shots





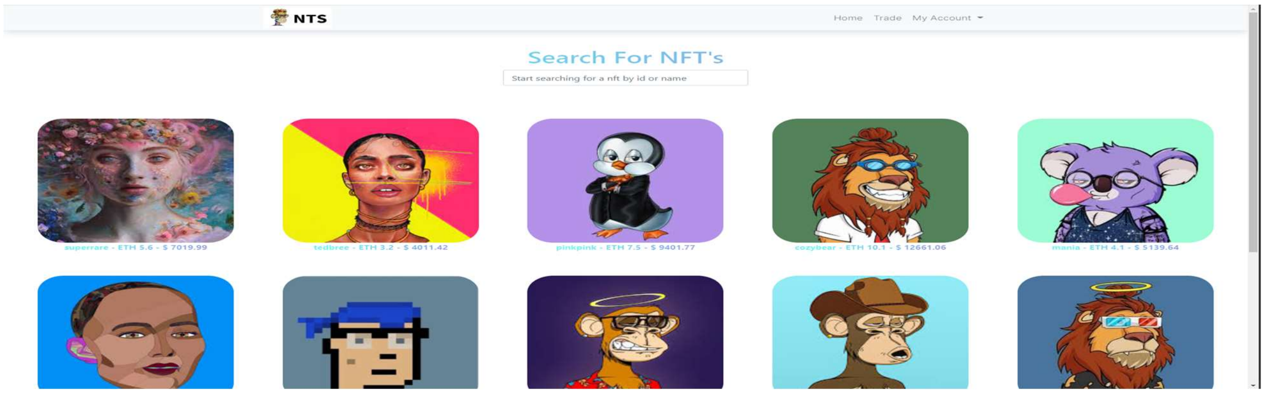
Varun

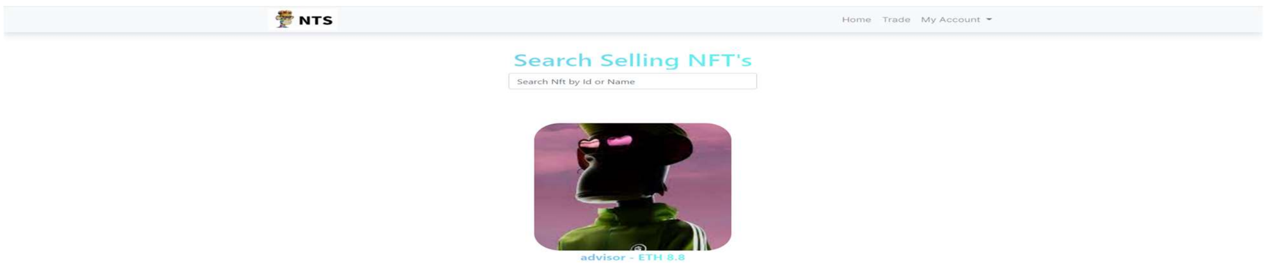
SILVER

615 Collier Dr Plano TX 75075

$100	1.562	1958.08	1253.57
Fiat Currency	Ethereum Count	ETH Value in $	Current ETH Value

NFT Address	NFT Name	NFT Value (ETH)	NFT Image
7fabfc6588427474bfad1d187e2021084e8363ffd9	advisor	8.8	


NTS
Home
Trade
My Account
Search For NFT's
Start searching for a nft by id or name
superrare - ETH 5.6 - $ 7019.99
tedbree - ETH 3.2 - $ 4011.42
pinkpink - ETH 7.5 - $ 9401.77
cozybear - ETH 10.1 - $ 12661.06
mania - ETH 4.1 - $ 5139.64



The Implementation recording is available in the submitted folder and online
https://tinyurl.com/CS6360projectdemo

<u>Appendix A</u>

Relational Schema:

Firstly, according to the requirements of the project and with purpose to simplify the relational model for this database, we have set the relation/tables conforming to 3NF Normalization. There is no transitive dependency of the non-prime attribute of a relation to the key attribute:

- The TRADER table has primary key t_id, email. The other attributes are non-prime and are directly functional dependent on the primary key.
- Since a TRADER can have multiple addresses but our system considered and accepts only one address for each trader. We have created a separate table for TRADER Addresses to make it easy for the system to retrieve information easily. The table ADDRESS has super key t_id and other non-prime attributes. t_id is the foreign key referencing t_id of TRADER table.
- Since a Trader can buy multiple NFT's and each NFT can be owned by only one user at a time we have a separate relation NFT table with NFT_id as primary key and t_id as foreign key referencing TRADER table. All the other attributes are directly functionally dependent on the primary key attribute.
- A Trader can buy multiple NFT's so they will be performing multiple transactions and each transaction is uniquely performed by one trader. To the relation between a TRDAER and NFT_TRANSACTION is many to one. In the table NFT_TRANSACTION, we have NFT_trans_id as primary key. t_id is a foreign key referencing TRADER table and NFT_id is a foreign key referencing NFT table. All the other attributes are directly functionally dependent on the primary key attribute.
- Similar to NFT_TRANSACTION table we have FIAT_TRANSACTION table where we store the transaction logs performed by trader to add either fiat or ETH to this trading account. In this we have ft_id as primary key to keep track of each transaction performed by trader. t_id is a foreign key referencing TRADER table. Here since a trader can perform multiple transactions and each transaction only performed by one trader, we have one to many relations. All the other attributes are directly functionally dependent on the primary key attribute.
- The MANAGER table has M_id as primary key. Manager will be able to access all the transaction logs performed by multiple traders. All the other attributes in the table are directly functionally dependent on the primary key attribute.

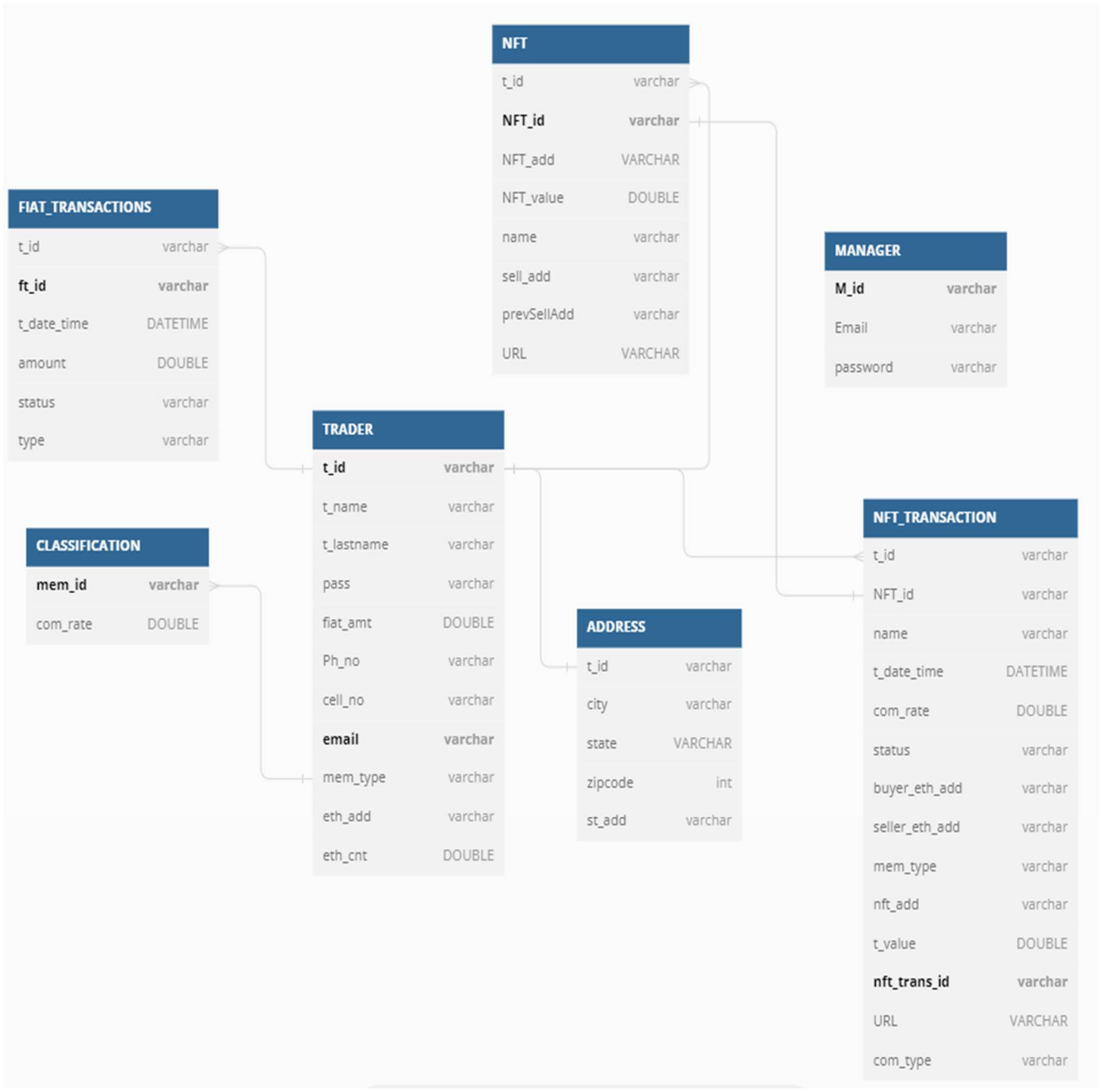
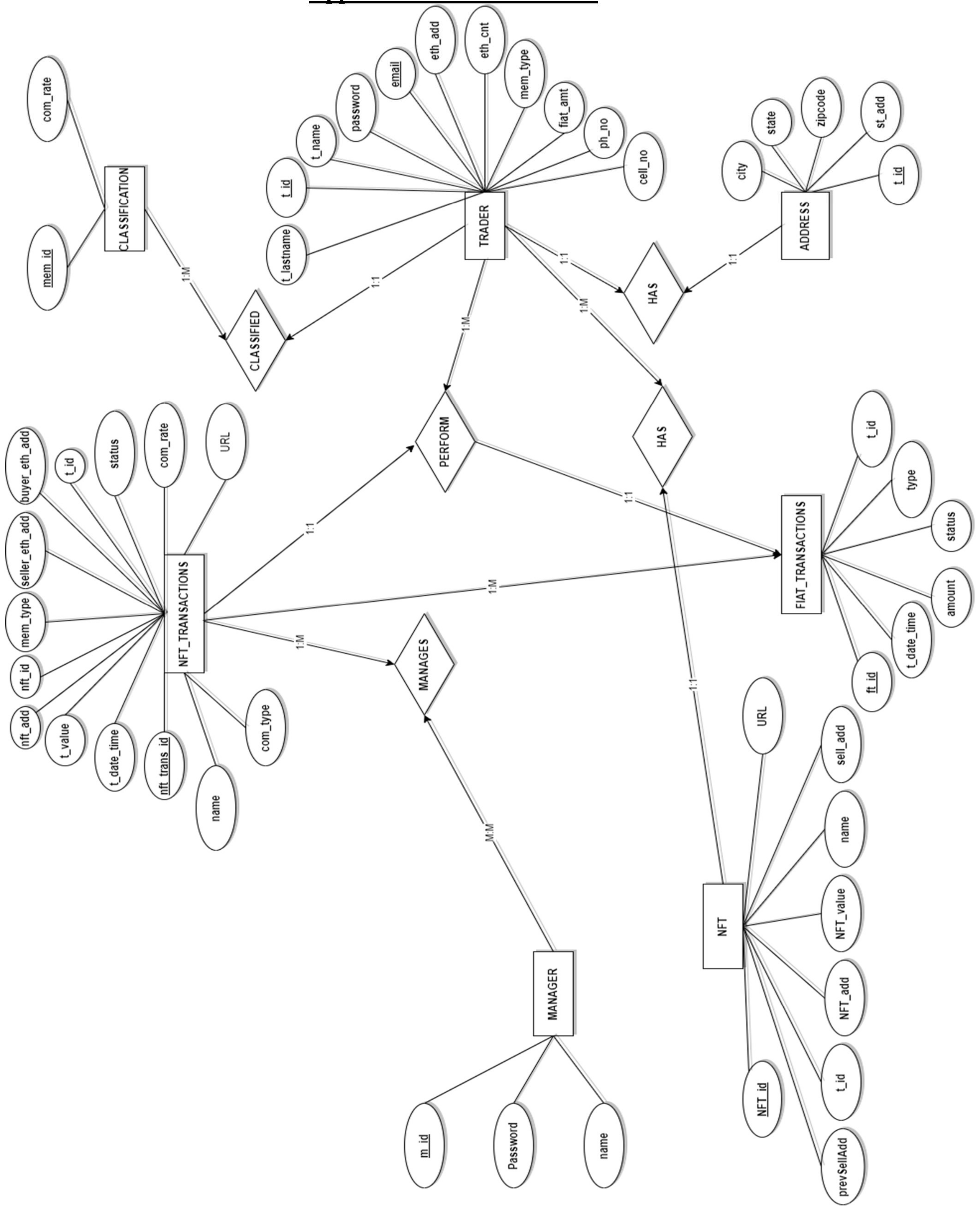


Fig 1. Normalized Relational Schema

Appendix B - ER DIAGRAM


CLASSIFICATION
com_rate
mem_id
TRADER
t_lastname
t_id
t_name
password
email
eth_add
eth_cnt
mem_type
fiat_amt
ph_no
cell_no
ADDRESS
city
state
zipcode
st_add
t_id
CLASSIFIED
HAS
1:M
1:1
1:1
1:1
1:M
1:M
PERFORM
HAS
NFT_TRANSACTIONS
buyer_eth_add
t_id
status
com_rate
URL
seller_eth_add
mem_type
nft_id
nft_add
t_value
t_date_time
nft_trans_id
name
com_type
1:1
1:1
1:M
1:M
MANAGES
FIAT_TRANSACTIONS
t_id
type
status
amount
t_date_time
ft_id
1:1
M:M
MANAGER
m_id
Password
name
NFT
URL
sell_add
name
NFT_value
NFT_add
t_id
NFT_id
prevSellAdd