

Project

CS6360 - Introduction to Database Design

by Dr. Murat Kantarcioglu

ER Diagram:

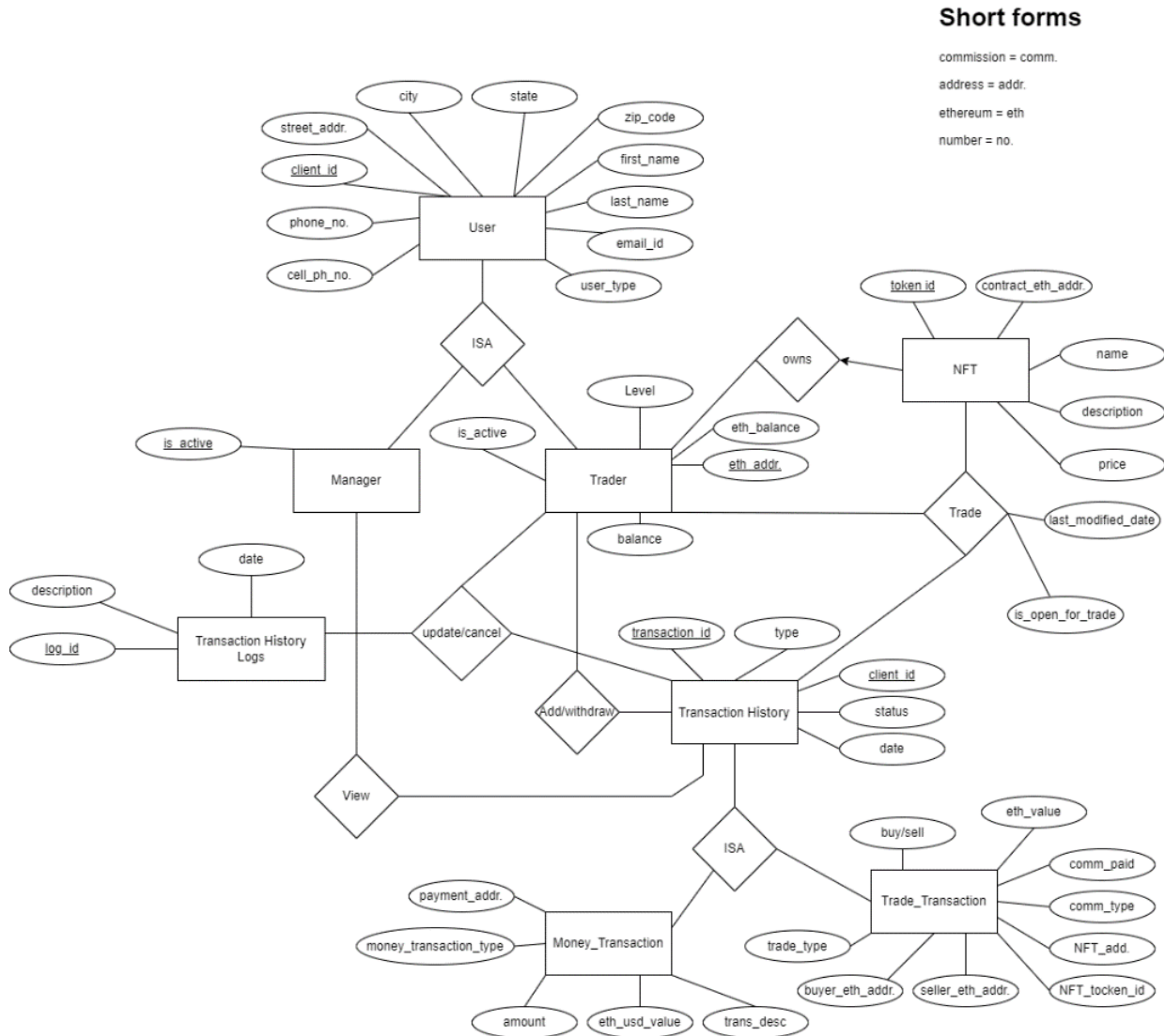


Figure 1. ER Diagram

Description:

The entity-relationship model consists of 8 entities with 4 relationships, in total. It has been designed and follows all the points made in the project's assignment. The hierarchies of the Diagram are the following:

1. The Trader and the Manager entities inherit all the attributes of the User.
2. The Trade_Transaction and the Money_Transaction inherits the attributes of the Transaction History.

Following criteria are met:

- ✓ 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 a 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, the 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 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 token 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/Ethereum 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 payments and NFT transactions. Although the system should allow such cancellations up to 15 mins after the transaction submission, logs should be stored for such cancellations 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.

Notes: A Trader can trade NFTs among other Traders. Each Trade is recorded in the Transaction History table. Furthermore, not only each Trader can update/cancel the existing Transaction that has been executed, but also can add/withdraw money (USD/fiat). Also, we store transactions that have been canceled for audit purposes.

Software Tools:

Tools and IDEs used for the project's development:












	Tool/IDE	Comments
	Spring Tool 4.0	Developing Spring-based enterprise applications.
	MySQL	Relational database management system
	Docker	Used for OS-level virtualization to deliver software in packages
	Java	Language used for backend development
	Postman	Design, build, test and iterate their APIs
	Spring Boot	Java framework used for rapid application development
	Jenkins	Reliably build, test, and deploy their software
	Google Cloud Platform	Cloud computing services that run on the same infrastructure
	React	Building tool for user interface
	VS Code	Code editing & development
	GitHub	Distribution, tracking & management of the code development

Table 1. IDE/Tools

The Integrated Development Environment (IDE) of the backend is the Spring boot. The development of the code built with this tool in order to create the desired application. We used the IDE of the Postman to set up the server to apply the developed requests and finally test our implementation. For the distribution, tracking and managing the development and the flow of the code we used GitHub.

- **Backend Code:**

src/main/java: Folder contains all the following packages

- com.utd.nts.controller: contains the main code which controls the function calling.
- com.utd.nts.service: contains abstractions & declarations.
- com.utd.nts.service.impl: implementation of the declared functions (business logic, validation).
- com.utd.nts.repository: use entity to interact with the DB
- com.utd.nts.pojo: contains I/O mapping

- **Frontend Code:**

src: contains the following

- App.js: contains the routings and mapping of the functions.
- Components: Folder contains the declared implementations

TEAM:

Name	Contact
Navaneeth Kumar Buddi	navaneethkumar.buddi@utdallas.edu
Indupriya Chegiredy	indupriya.chegiredy@utdallas.edu
Christos Vasileiou	christos.vasileiou@utdallas.edu
Harshavardhini Sridhar	harshavardhini.sridhar@utdallas.edu
Dibyanshi Singh	dibyanshi.singh@utdallas.edu

Note: To get more information about setting up the application, please get advised by the corresponding READ.md files of backend or frontend.