Decentralized Retail Store application using Ethereum

By Adam Tache, Gabriel Chen, Vishnutej Mylavarapu

1. **Introduction**

A normal application has clients and a server, the clients send requests to the server, the server then talks to a database to insert, delete and fetch information or when transactions are performed and respond to the clients. The database layer is not visible to the clients. There could be a few situations in which it could be helpful to have secure access to publicly visible database

1. When the server is down, any information is not available.
2. The server/admin has full control over all the information. Movement of data is not possible.

This is where decentralized applications come into picture. There is no centralized server to which all the clients send their requests to. Any client that wishes to communicate with the decentralized application needs to have the full copy of block chain on their system. “Block chain is an enormously powerful shared global infrastructure that can move value around and represent the ownership of property”.

1. **Application Details**

Our application is based on smart contracts that run on Ethereum. Block chains in Ethereum store data, code for smart contracts and run code on Ethereum Virtual Machine (EVM). The smart contracts are built using Solidity.

We have built a console based retail store which works on silent auction bidding. Similar to an actual retail store where we have sellers who add the product to the store and buyers who purchase the item from the store, this application has store owner who can add products to the store and users who can purchase the products by bidding on them. Currently, the application supports adding products to the store, bidding on products and finally displaying who the winners of the bids for various products are.

The store owner can add products he wants to auction. He lists a minimum price for each of the products which is not disclosed. If none of the bids have a price that is greater than or equal to the minimum price of the product, the product is not sold to any user. If the product is sold, then it is removed from the store.

The users bid on any of the products in the store they like with an amount after looking at the product description. Each of the products have a bid timer which turns on after the first bid by any user. After the timer ends, the application picks a winner for each of the products.

The winners of the bids are based on highest bids and the balance of ether in the wallets of the user. If the winner of the bid does not have the required balance to purchase the item, the winner of the bid is the next highest bid. The amount is then reduced from the winner’s wallet and added to the store owner’s wallet. At the end, the winner of the bid is displayed.

1. **Application Framework**
2. **How to run the application**
3. **Test scenarios**
4. Add a product as the store owner.
5. Try to add the product as a user.
6. Bid on a product as a user.
7. Try to bid on a product as the store owner.
8. Bid with amount more than what the user has in the wallet.
9. Have two users bid on the same item with same amount, when they have enough balance in their wallets.
10. Have two users bid on the same item with same amount, when one of them does not have enough balance in their wallets.
11. Try to bid on non-existing product.
12. Try to bid on a product with very small amount.
13. **References**

<https://medium.com/@mvmurthy/ethereum-for-web-developers-890be23d1d0c>

<https://ethereum.org/>