**Information and Security**

***Final Project Report***

**Digital Assets Exchange Marketplace**

**Date**

***12-11-2022***

**Submitted to**

***Dr Fahad Samad***

**Table of Contents**

[***Project Overview*** 3](#_Toc121663504)

[***Project Cost Estimation:*** 4](#_Toc121663505)

[***Process Model Strategy*** 4](#_Toc121663506)

[***Tools, Libraries and Technologies Used*** 4](#_Toc121663507)

[***Use Case Diagram*** 5](#_Toc121663508)

[***Project implementation and work flow:*** 6](#_Toc121663509)

[User Verification and Digital Signature 6](#_Toc121663510)

[Data integrity using Digital Signature and IPFS: 8](#_Toc121663511)

[Owneship transfer and verification throough Digital Signatures and financing: 11](#_Toc121663512)

[Exploiting Smart Contract vulnerabilities through Reentrancy attack: 12](#_Toc121663513)

[***Project Testing*** 14](#_Toc121663514)

## ***Project Overview***

In this project we will develop a digital assets exchange platform and explore smart contact vulnerabilities. A digital asset is anything that is created and stored in digital form and is identifiable and has a value. Non-fungible tokens (NFTs) are the assets on a block chain that can be traded in a block chain system.

A user with an asset will convert into a digital asset and its NFT will be created. The user will post its asset on the website through which it could be traded with another user. A seller will provide an amount to trade their assets. The buyer will purchase the assets and provide the amount with Ethereum. The account created will provide public and private key address where public key address will be used as an identity and to transfer the amount through the block chain. A user registration and login system will be intact n the system for securing the user authentication.

When user uploads documents on block chain, they will be stored on IPFS that is a decentralized peer to peer storage system. To maintain the integrity of documents upload, strict security system is implemented using digital signatures to maintain the integrity of assets uploaded and to maintain the ownership to the right users.

Smart contract vulnerabilities are also explored in this project and one of the notable attack, Reentrancy attack and double spending attack are also implemented.

## ***Project Cost Estimation:***

Developer Cost = Rs. 500 per hour

Every Week: 5 hours of working

Per Week calculation: 500\*2\*2 = Rs. 2000

Contract Deployment Cost: 0.1 ether

## ***Process Model Strategy***

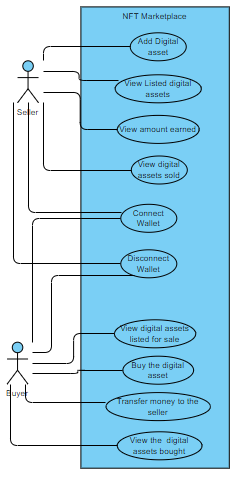
In this Digital asset exchange project we used the unified process model in our software development life cycle.

The system was developed using Agile Development methodologies using Extreme Programming framework which is the subset of unified process model.

## ***Tools, Libraries and Technologies Used***

* React js (Frontend)
* Node js (Backend)
* Material UI (Styling)
* Solidity (Smart Contract)
* ERC721, ERC721A, ERC720, Own able etc. libraries (Smart Contract)
* Jest (Smart Contract Testing)
* IPFS (Decentralized storage)
* Goerli (Test Net Block chain)
* Infura (JSON-RPC)
* React-Web-Provider
* Ether Scan
* Remix (IDE)
* VS Code (IDE)
* MetaMask (Wallet)

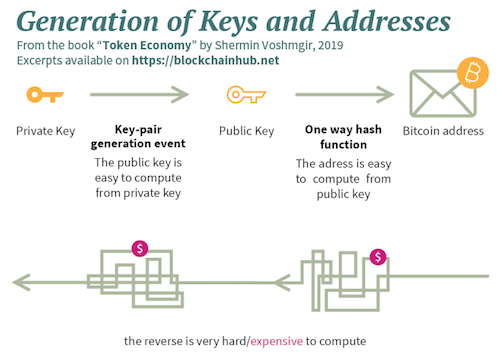
## ***Use Case Diagram***



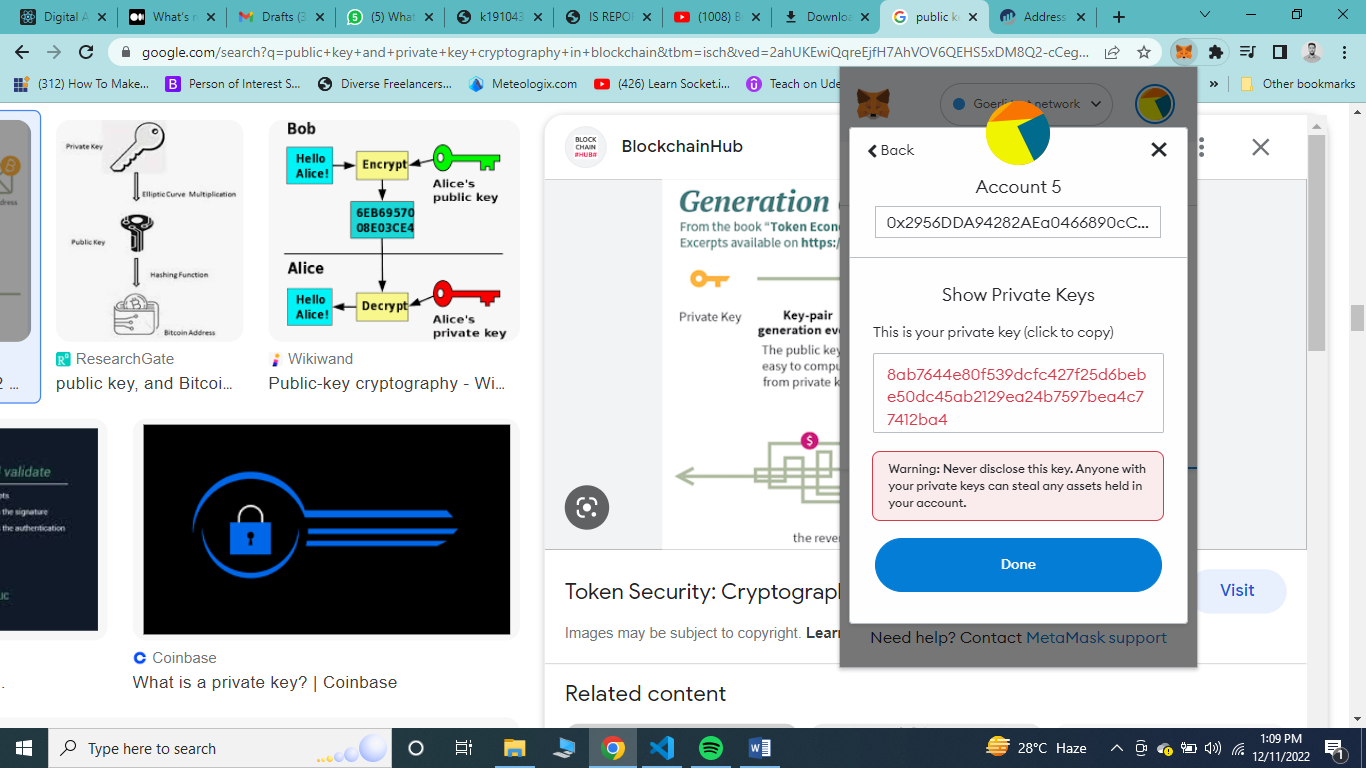
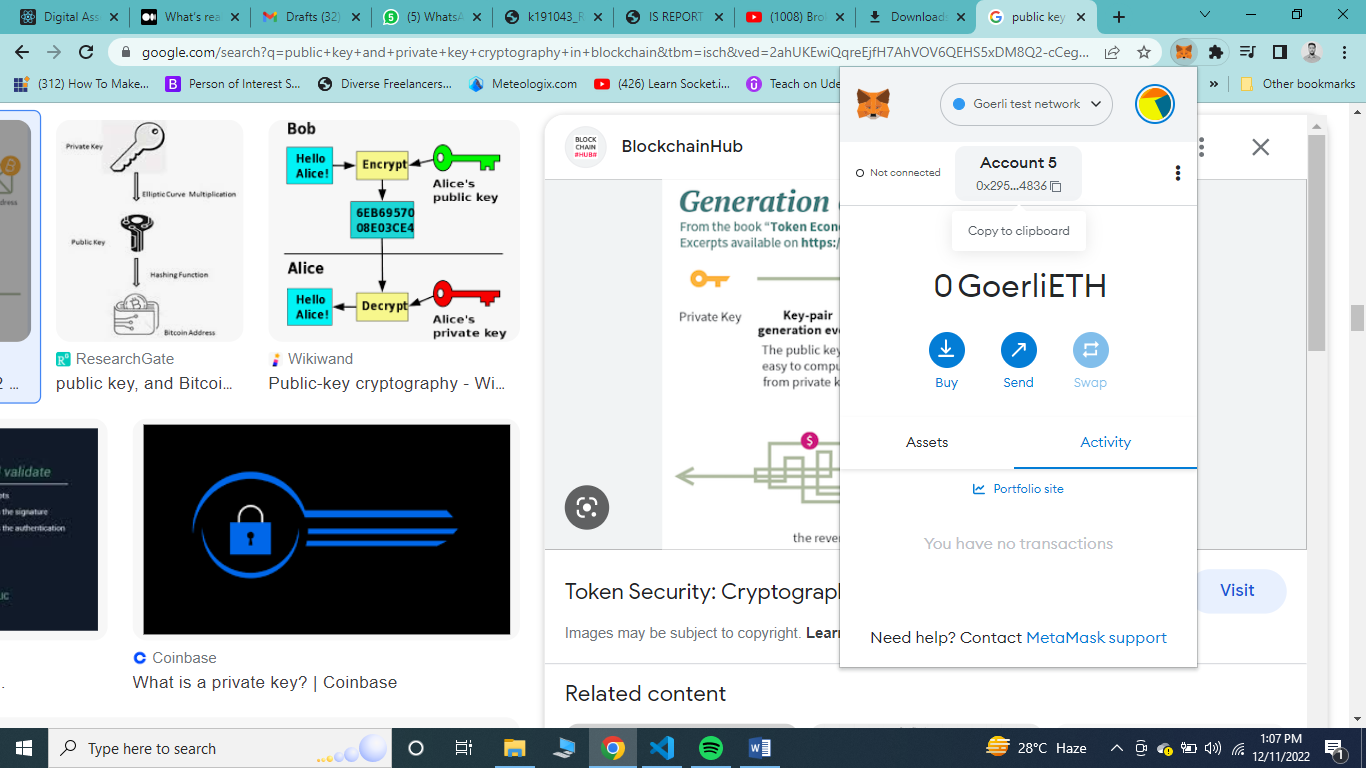
## ***Project implementation and work flow:***

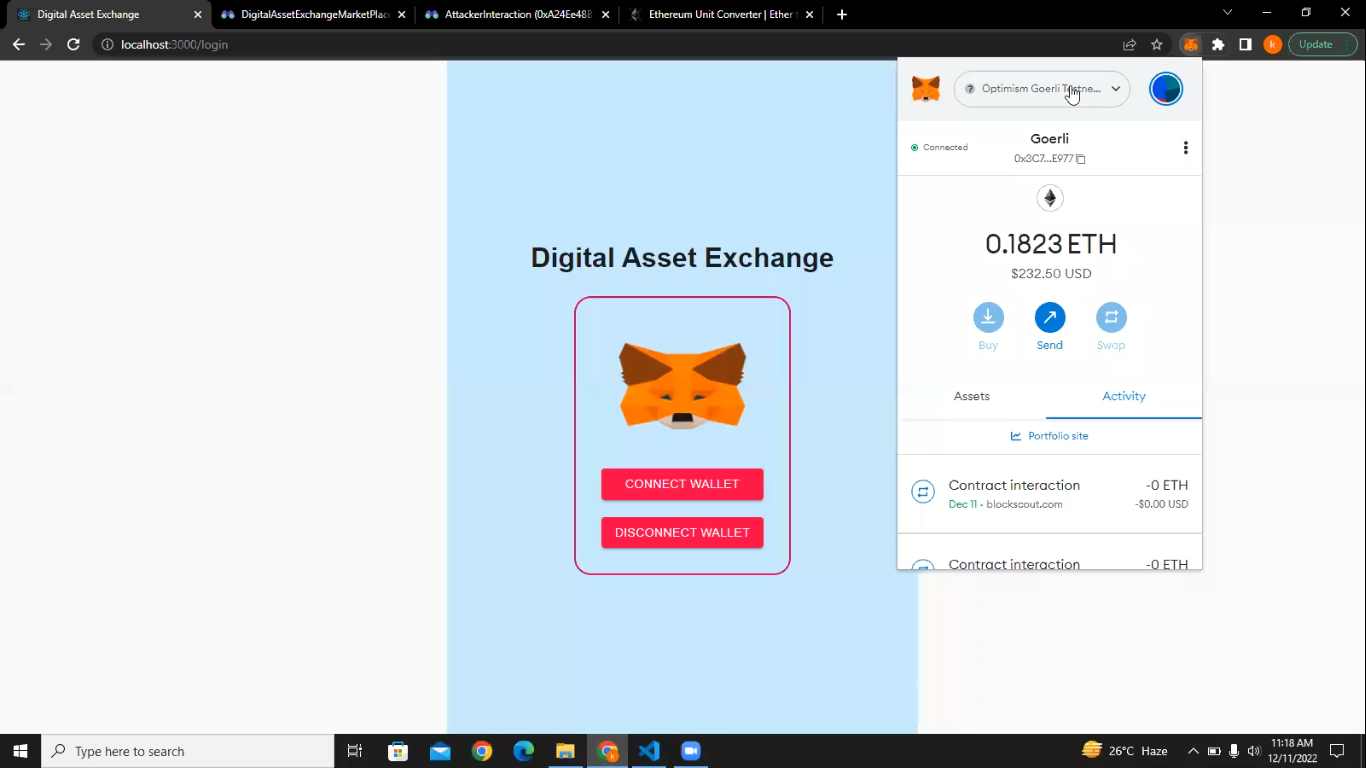
### User Verification and Digital Signature

We are using digital signatures for the verification of users in our application. Anyone who logs in to the application is required to connect its digital wallet to the website. We are using Meta mask for the management of digital signatures. User can create multiples accounts/addresses using Meta mask. Whenever a new account is requested by the user, a random seed phrase is generated at run time and a private key is generated from that seed phrase and also a public key that maps to that private key. This public address also represent the identity of user.



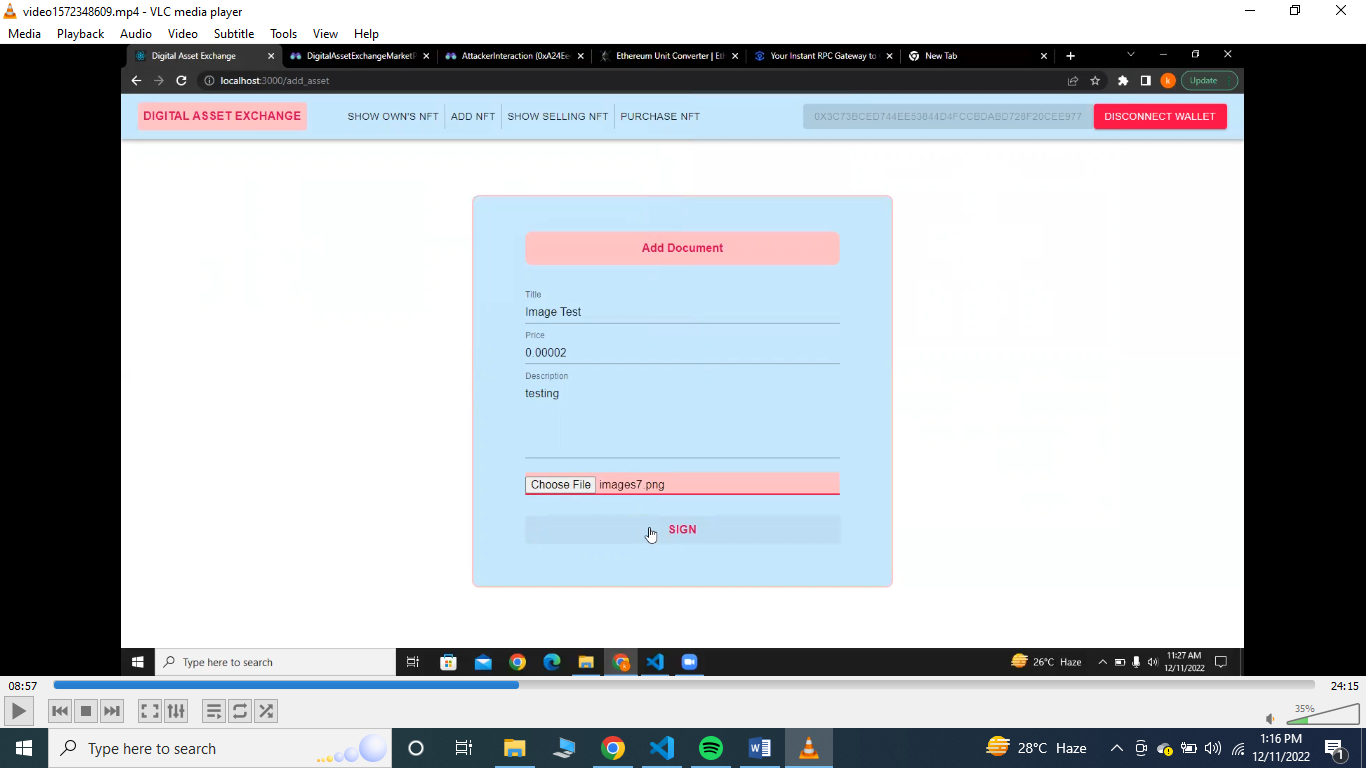
Whenever a connection to smart is to establish, Meta mask takes the message and private key of the particular address and returns a public address that is then used in smart contract for the validation of a user.

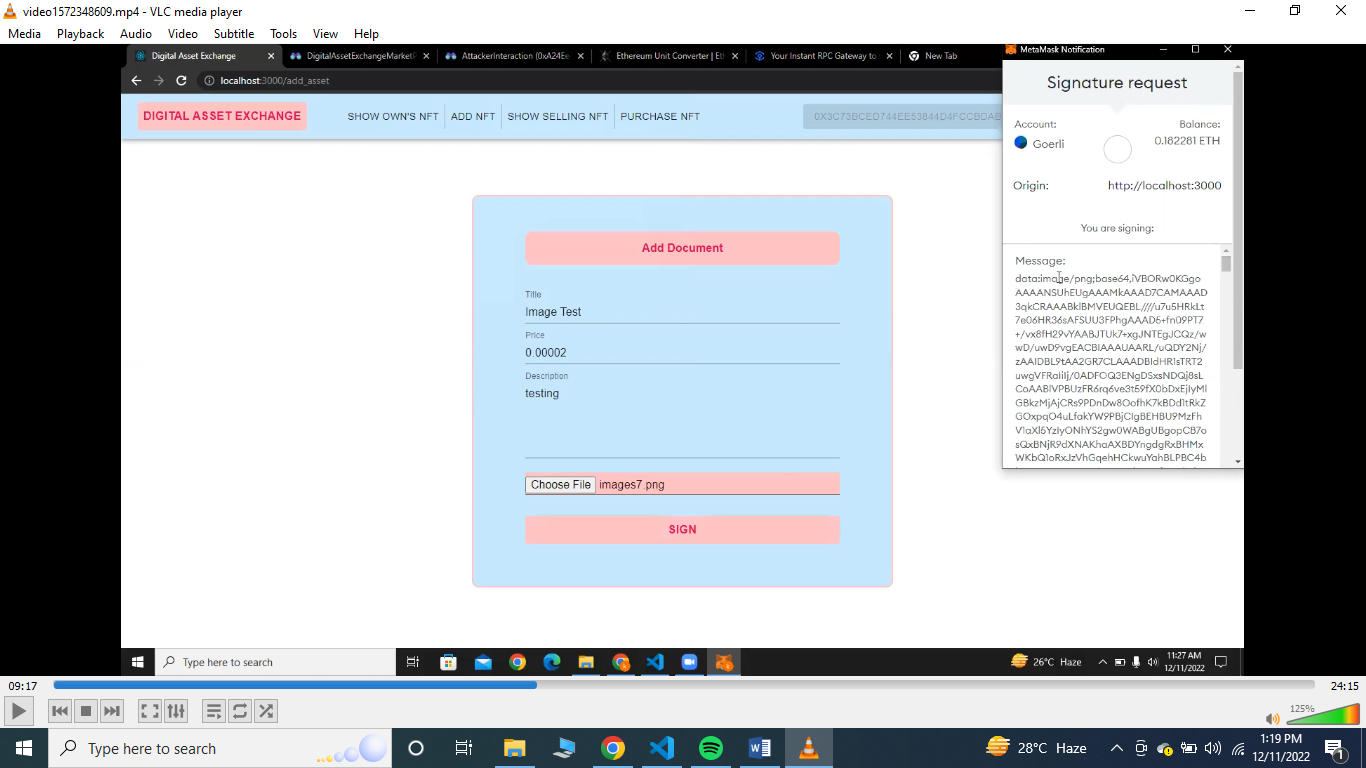


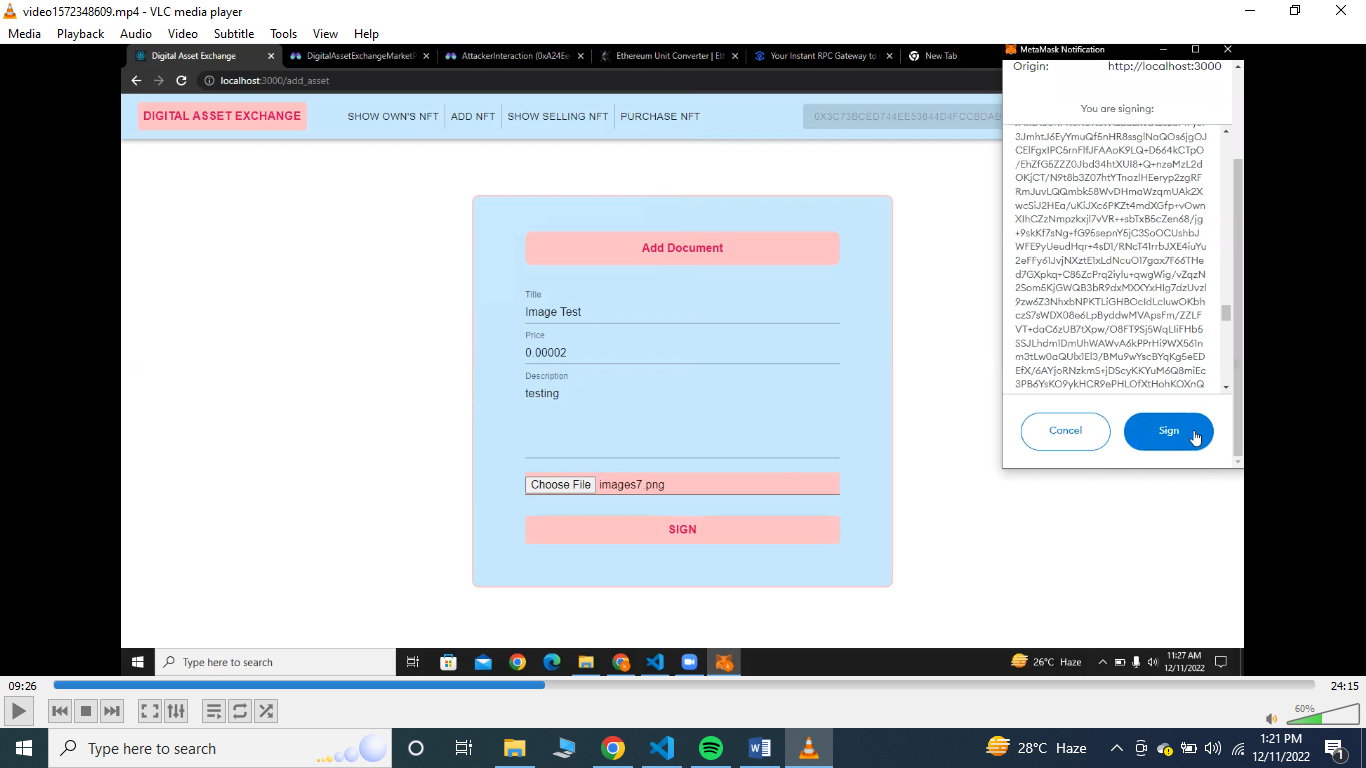


### Data integrity using Digital Signature and IPFS:

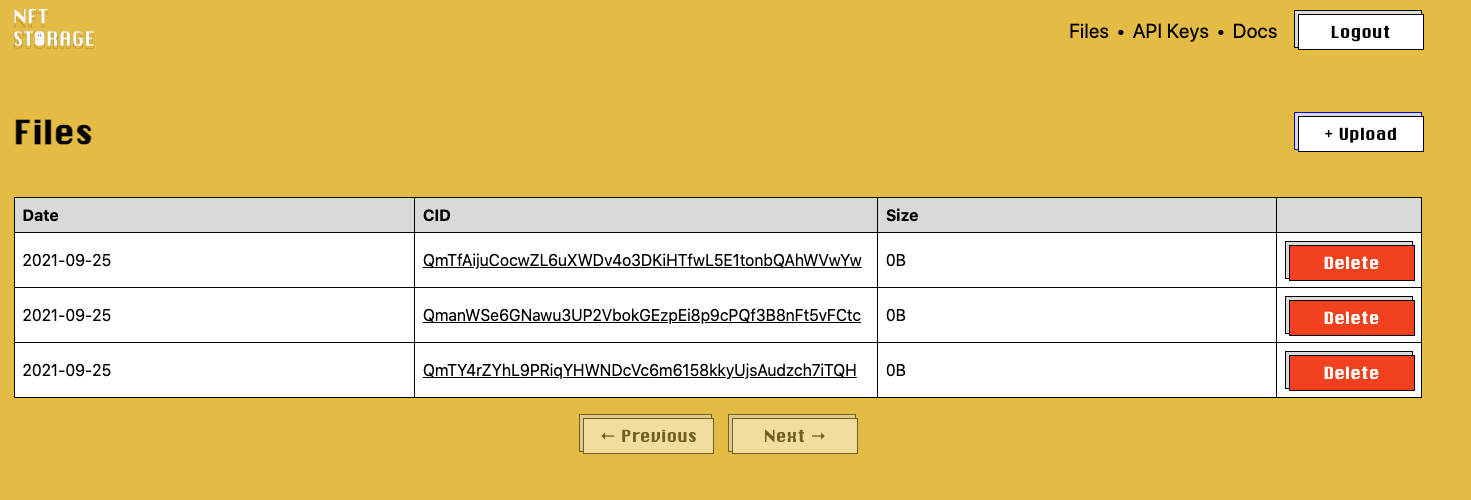
Once user connects it wallet to the application, it is redirected to the home screen. From home screen user can navigate to different section of the site. If user wants to upload its digital asset to the block chain and prove its owner ship, user can go to the upload document screen and uploads documents.



As soon as user uploads its image, we convert the uploaded image into base64 and sign the data using the user’s private key in order to get the digital signature public key.



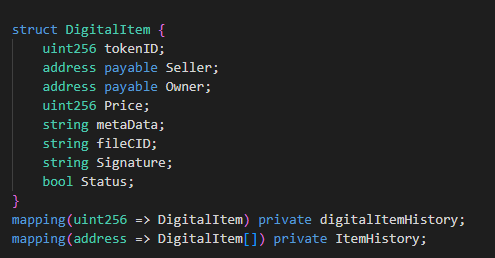
After signing the image, we upload the asset on to the IPFS which is a peer to peer decentralized storage service. After uploading on IPFS it returns a CID which is basically the hash of the assets uploaded on IPFS. This CID is also used to retrieve documents from the IPFS.



After we receive response from the IPFS, we add the CID and digital signature on the block chain using smart contract. This data is mapped over the public address of the user that we initially got from the MetaMask.

We added this functionality to ensure the integrity of assets being upload on IPFS. If the document have been changed while uploading on IPFS, we can verify this as we have uploaded the digital signature of original document on block chain .We will retrieve the malicious document from the IPFS and call the verify function, since the malicious document digital signature does not matches with the original document signature we will ensure that the document have not been changed and the integrity of the document is maintained.

Here is the implementation detail of the above mentioned concept in smart contract:



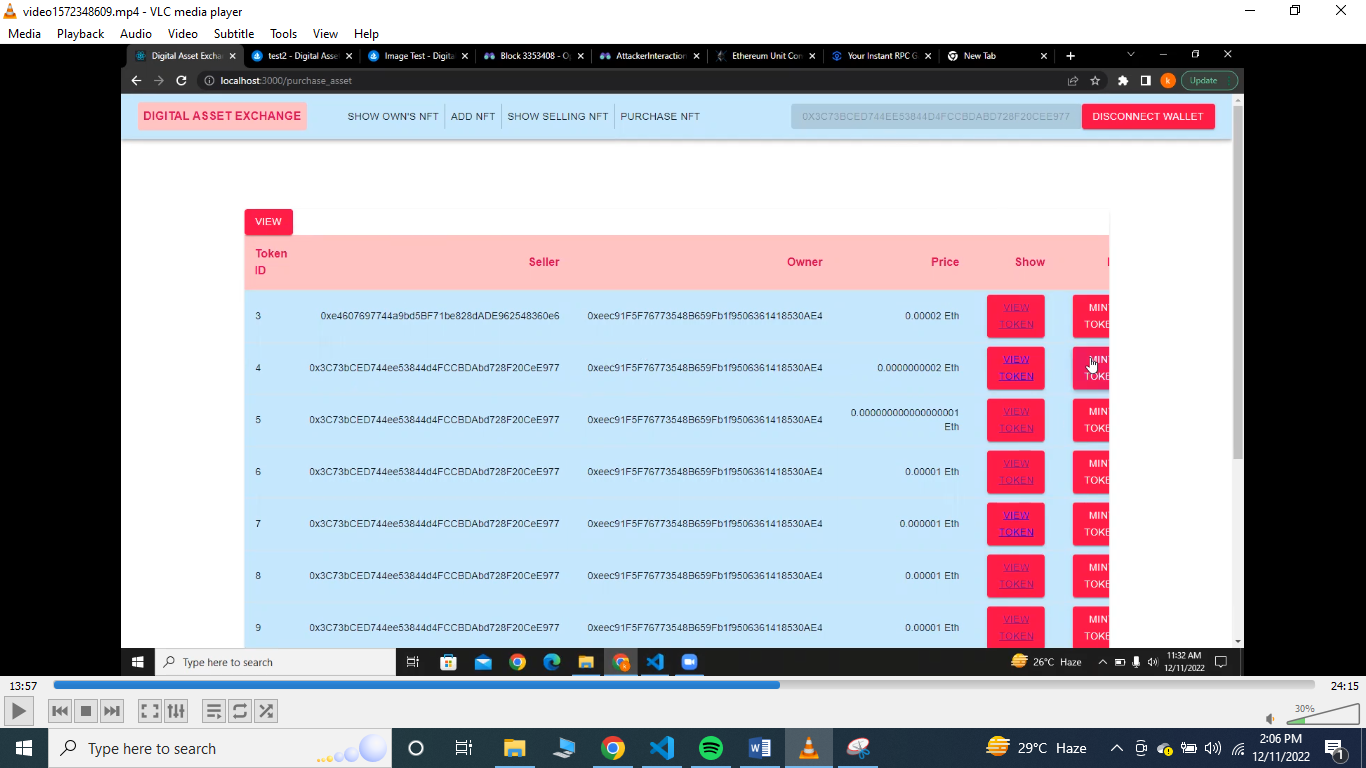
Here is the implementation detail of the signature verifcation on the application level:



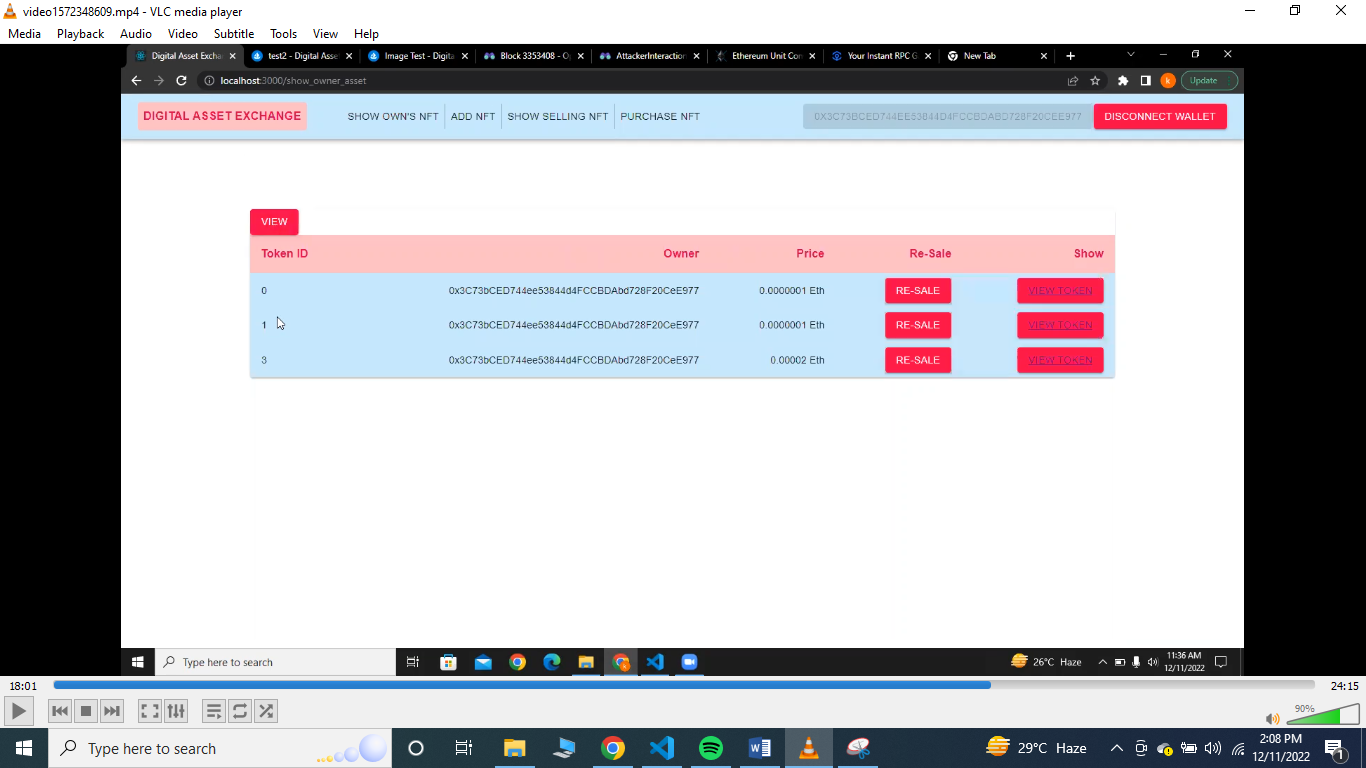
### Owneship transfer and verification throough Digital Signatures and financing:

Once document are uploaded, the seller can view it from its dashboard and the buyers can see the documents on the market place. Buyer can request for the purchase as per their willingness. Whenever a user buy the digital assets, the amount send buy the buyer is transferred to the owner of the document and once this transfer has been made, the ownership of the document is changed to the address public address of the buyer.

Here is the implementation of Market place screen:



Here is the implementation screen of the assets that the current user holds:



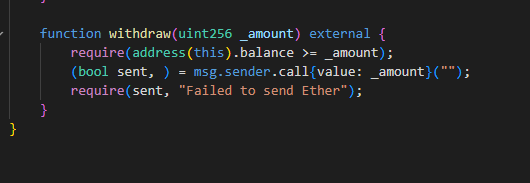
### Exploiting Smart Contract vulnerabilities through Reentrancy attack:

Since we are financing the transactions through ethers, the amount send buy the buyers reside in the contract. We have written an attacker contract which will instantiate the market place contract in itself and calls the withdraw function of the marketplace contract that is used to withdraw functions, and transfer those funds into itself.

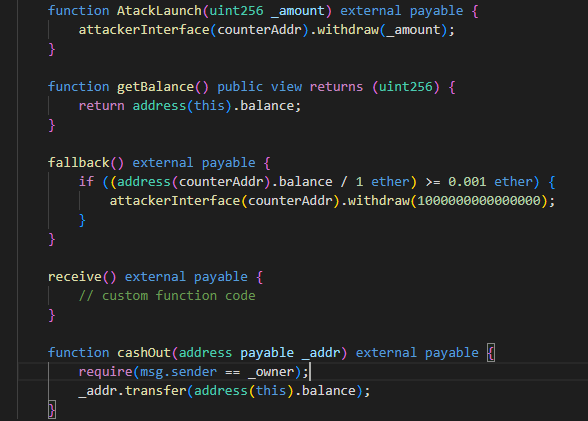
This attack is possible whenever a function that requires some funds movements with in itself is defined without using the modifier “payable”.

Modifier in smart contract are basically the pre-conditions that are checked and need to be true before the actual body of the function in executed.

Here is the withdraw function of the market place contract:



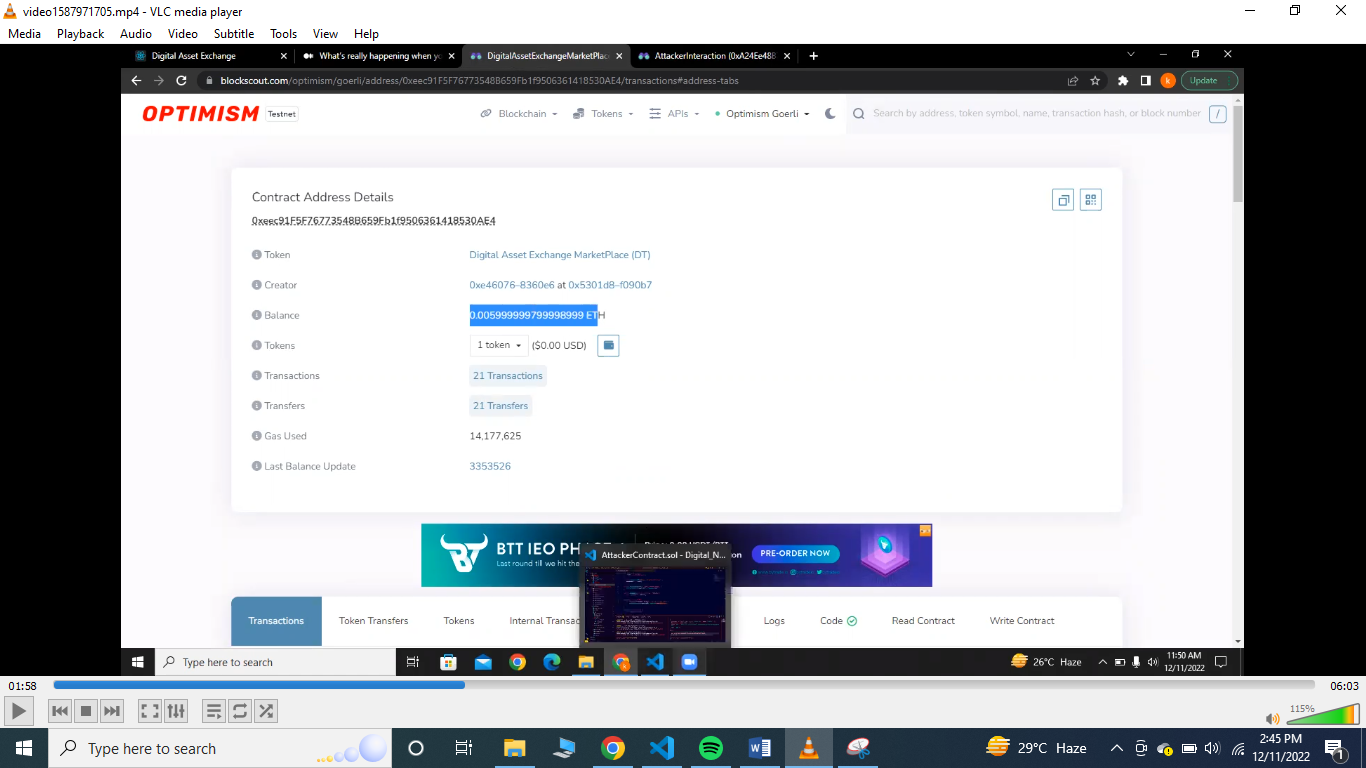
Here is the function which calls market place contract withdraw function very quickly from the attacker contract:



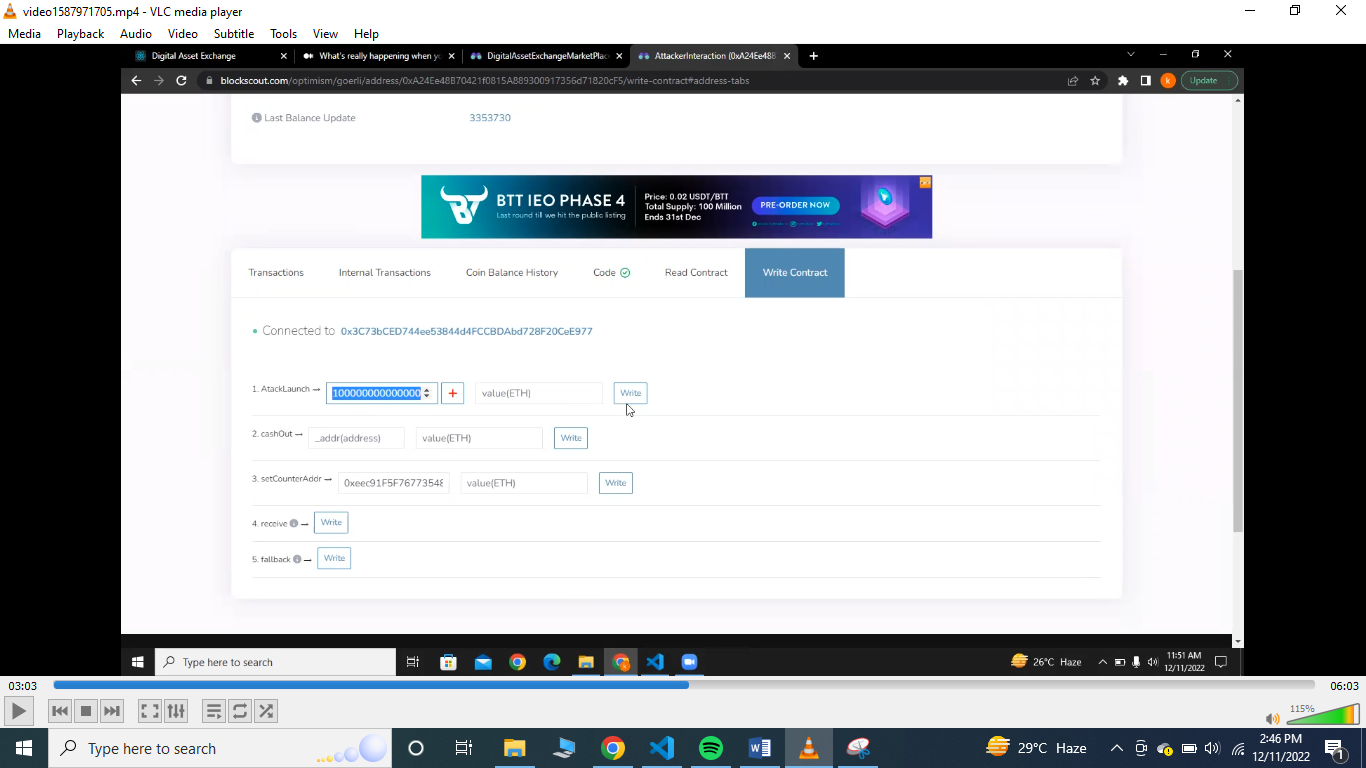
Since we were not using the payable what happening is that when the first call to withdraw s made, the value is not updated yet and the fallback function with in the attacker contract is called which again calls the attacking function.

In this way the Reentrancy attack is made on a smart contract.

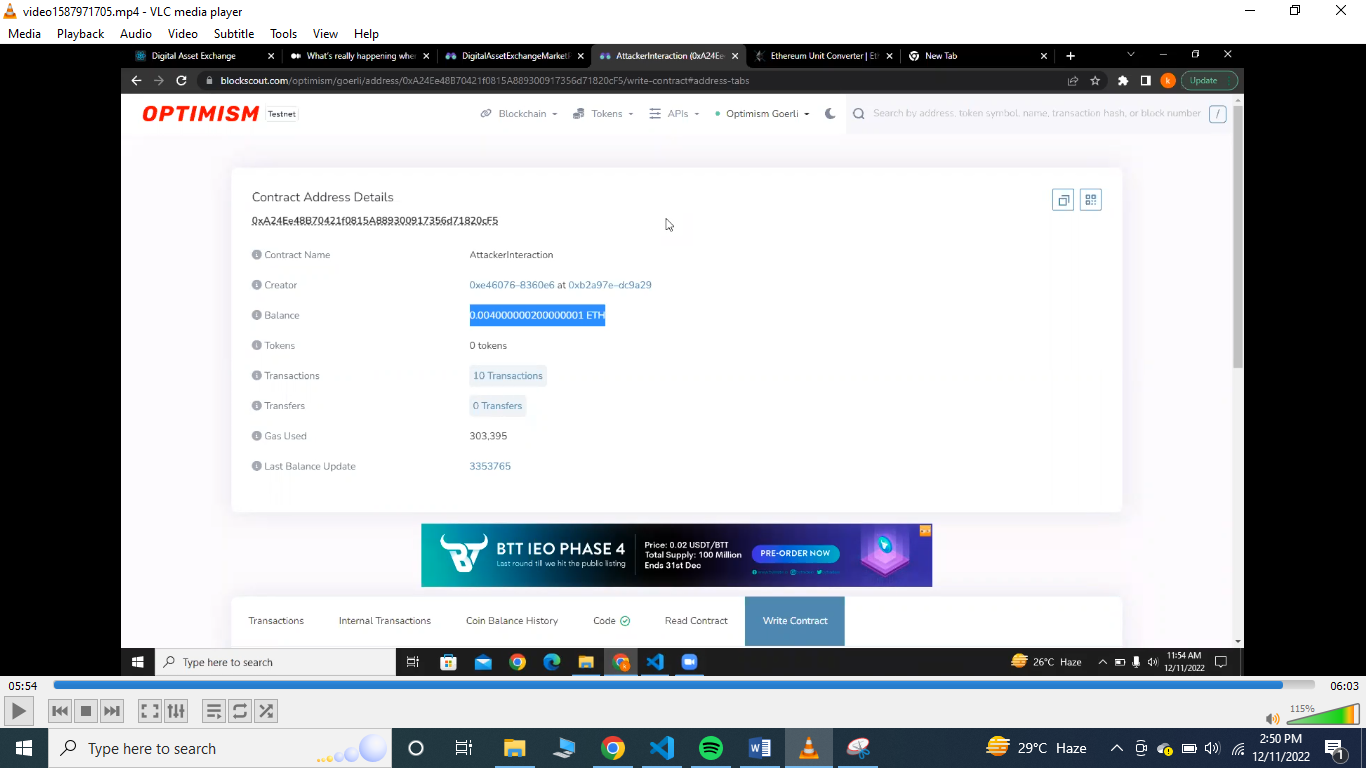
Here is the snapshot of current balacne present in marketplace smart contract before attack is launched



Here is the snapshot of initializing marketplace smart contract object in attacker’s contract and launching the attack



Here is the screenshot of the market place smart contract from where the current balance present is less as compare to the balance before the attack:



## ***Project Testing***

Smart contract was test using “mocha” which is the JavaScript library for writing the unit test cases for the smart contract.

Functional requirements of the project were tested through the manual testing performed by both of the group members.