Blockchain Application Development

CSE-526

Project phase 2 submission

ChronosCapsule

Ankita Sharma
50464503
as488@buffalo.edu

Diptangshu De
50466657
diptangs@buffalo.edu

1 Issues addressed

Authenticity: By using a temporal moment capture NFT, the ownership and validity of the captured moment may be confirmed on the blockchain, helping to thwart fraud and guarantee that the moment is actually special and precious.
Scarcity: The NFTs' scarcity may be preserved by producing a finite number of time moment capture NFTs for each instant, which can assist to raise their value and attractiveness.
Preservation: By using blockchain technology, it is possible to ensure that the moment is kept in a safe, decentralized manner, preventing the loss or erasure of the original event or memory.
Access: The usage of time moment capture NFTs can also offer greater access to historical events or moments that might otherwise be challenging or impossible to access, especially for people who are physically unable to attend or experience the moment.
Fundraising: Time moment capture NFTs can also be used to raise money for occasions or causes, with the sales revenues going to the occasion or cause.

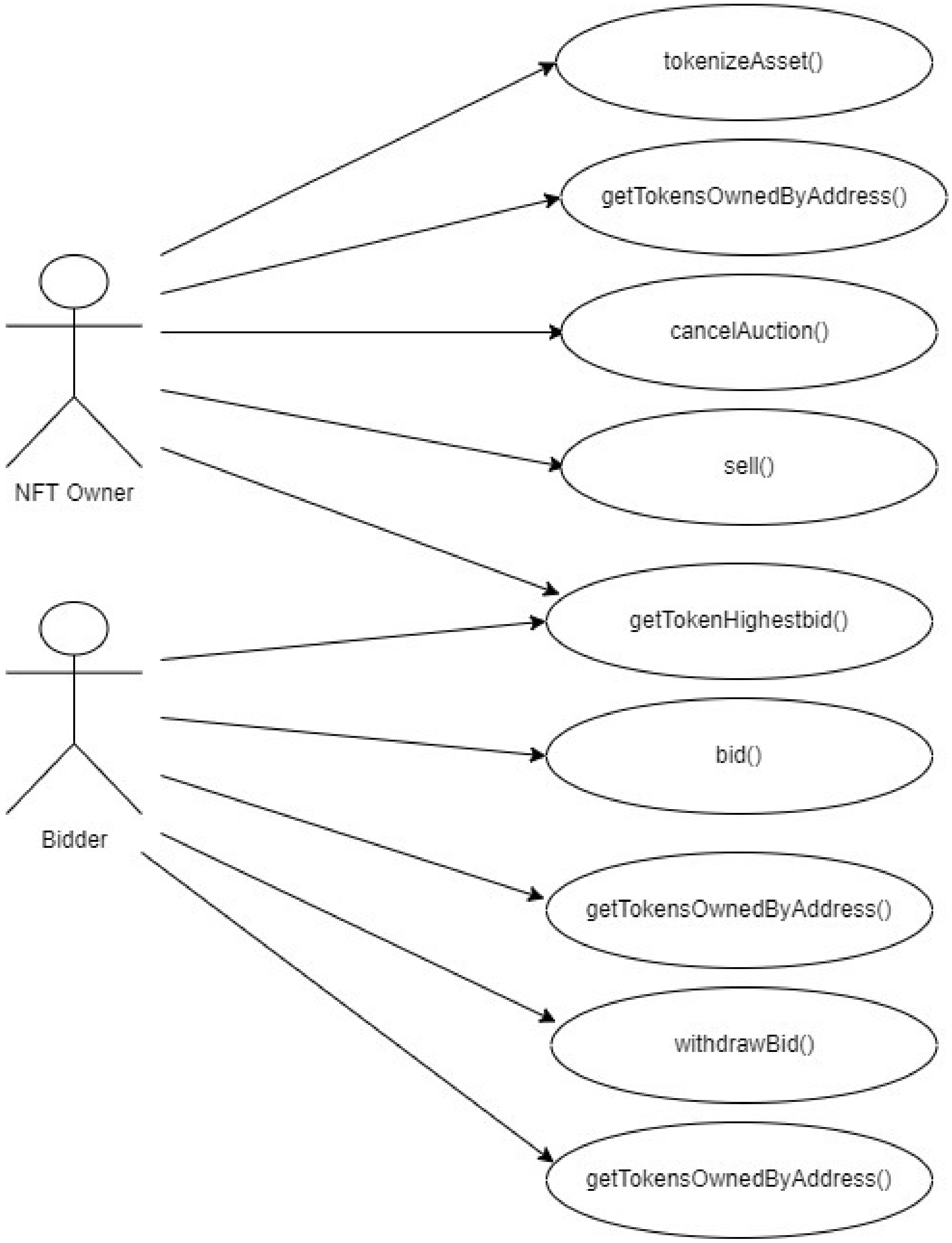
2 Abstract

A time-lapse photograph A single instant in time is preserved on the blockchain via the digital collectible known as NFT. With the use of these NFTs, historical memories and events can be captured and preserved, and a new type of collectible can be created as well as a way to support causes and events. These can be made and owned by anyone, and they are particularly helpful for documenting significant occasions like historical occurrences or defining moments in one's life. On the blockchain, there are time moment capture NFTs that may be accessed and traded internationally. By utilizing blockchain technology, it is possible to verify the validity and ownership of the moment that was captured, making these treasures one of a kind.

3 Digital assets and tokens

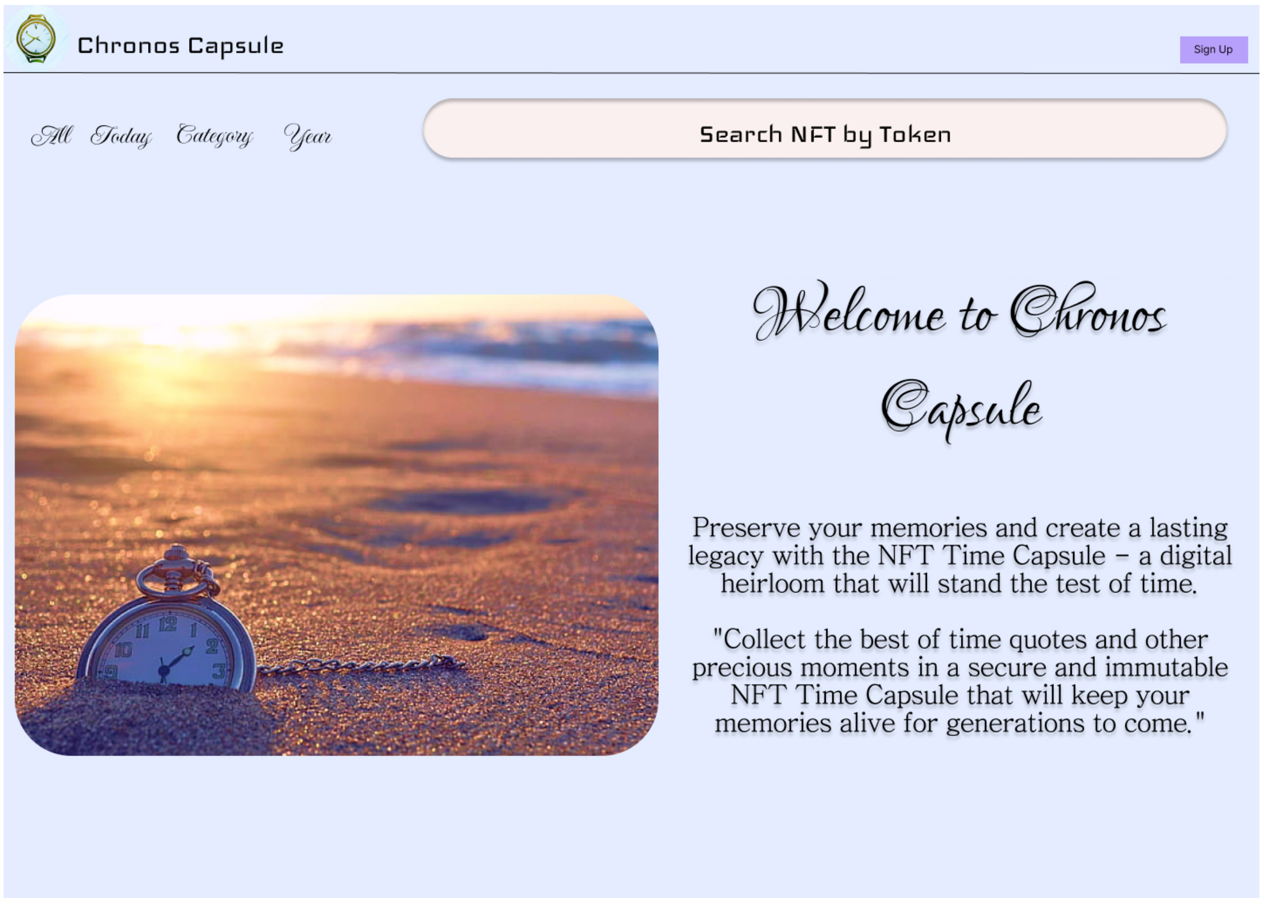
Any digital file that depicts a particular instant in time can serve as the digital asset for a time moment capture NFT. In this instance, we've chosen to employ photographs. The non-fungible token (NFT), which is a distinct digital asset that is maintained on the blockchain and signifies ownership of the digital asset, may be used as the token for a time moment capture. NFTs are unique digital assets that can represent any form of unique digital asset, including images, videos, audio files, and other types of media. They are created via smart contracts on blockchain networks like Ethereum. NFTs are distinctive because they each possess a distinct set of qualities that set them apart from other NFTs. Cryptographic hashing algorithms are used to generate a unique code that represents the NFT and its metadata, enabling its uniqueness. NFTs can be easily exchanged between people and purchased, sold, and traded in marketplaces built on blockchain technology. The ownership and authenticity of each moment can be confirmed and maintained on the blockchain by utilizing NFTs as the token for time moment capture NFTs. By doing so, fraud may be avoided and every moment can be guaranteed to be genuine unique and priceless as a collection. Because they have a long history of being used to record and retain memories, photos were chosen as the digital asset for time-moment capture NFTs. Photographs serve as a visual record of a certain point in time and have the capacity to stir up strong feelings and memories in individuals who view them. Photographs are a realistic option for usage in NFTs since they are simple to digitize and put on the blockchain as a digital asset. They can be used to represent a wide range of diverse occasions, from historical events to personal milestones, and are simple to share and access on a worldwide scale. Moreover, it is simple to digitize and store photos on the blockchain: Photos can be readily stored on the blockchain as an NFT as a digital asset. This makes it possible to provide a distinct and verifiable record of ownership and validity for each moment that is filmed. Photos may also be quickly shared and accessed from anywhere in the globe because they are in a digital format.

4 Use case diagram



Here's a description of the cases that are being dealt by the NFT:
tokenizeAsset(): By minting a fresh ERC721 token, the tokenizeAsset method creates a new asset. It requests information on the asset's description, URL, base price, and auction expiration date. Using this data, it constructs a new Asset struct and designates the sender as the asset's owner. After that, it uses the ERC721. mint method to produce a new token and returns the ID of that token.
getAllTokenIds(): This function returns an array of all the token IDs that have been minted so far.
getTokenHighestBid(): This function takes in a token ID and returns the current highest bid for that token.
bid(): With this feature, we make a bid on an object. When a token ID and a bid amount are entered, the system checks to see if the amount is higher than the highest bid currently in place and if the auction has not yet ended. If the bid is higher, it changes the asset's highest bid and highest bidder and transfers the prior highest bid to the previous highest bidder's bids mapping.
withdrawBid():With this function, the owner can cancel a previous bid they made on an asset. It accepts a token ID, confirms that the sender is not the top bidder at the moment, and, if they have placed a bid, sends the winnings back to them.
sell(): In order to sell an asset to the highest bidder right now, utilize this function. The highest bid amount is transferred to the asset's owner after the system receives a token ID, verifies that the sender is the owner of the asset and that the auction has ended. The highest bidder is then given the token.
getBidCount(): This function takes in a token ID and returns the number of bids that have been made on the asset.
cancelAuction(): The asset is returned to the owner if an auction is canceled using this function. The asset is deleted and the token is burned once it receives a token ID, verifies that the sender is the owner of the asset and that no bids have been placed on it, and then deletes the asset.

5 Wireframes



Preserve your memories and create a lasting legacy with the NFT Time Capsule – a digital heirloom that will stand the test of time.

"Collect the best of time quotes and other precious moments in a secure and immutable NFT Time Capsule that will keep your memories alive for generations to come."

Figure 1: This is the introductory page of the application.

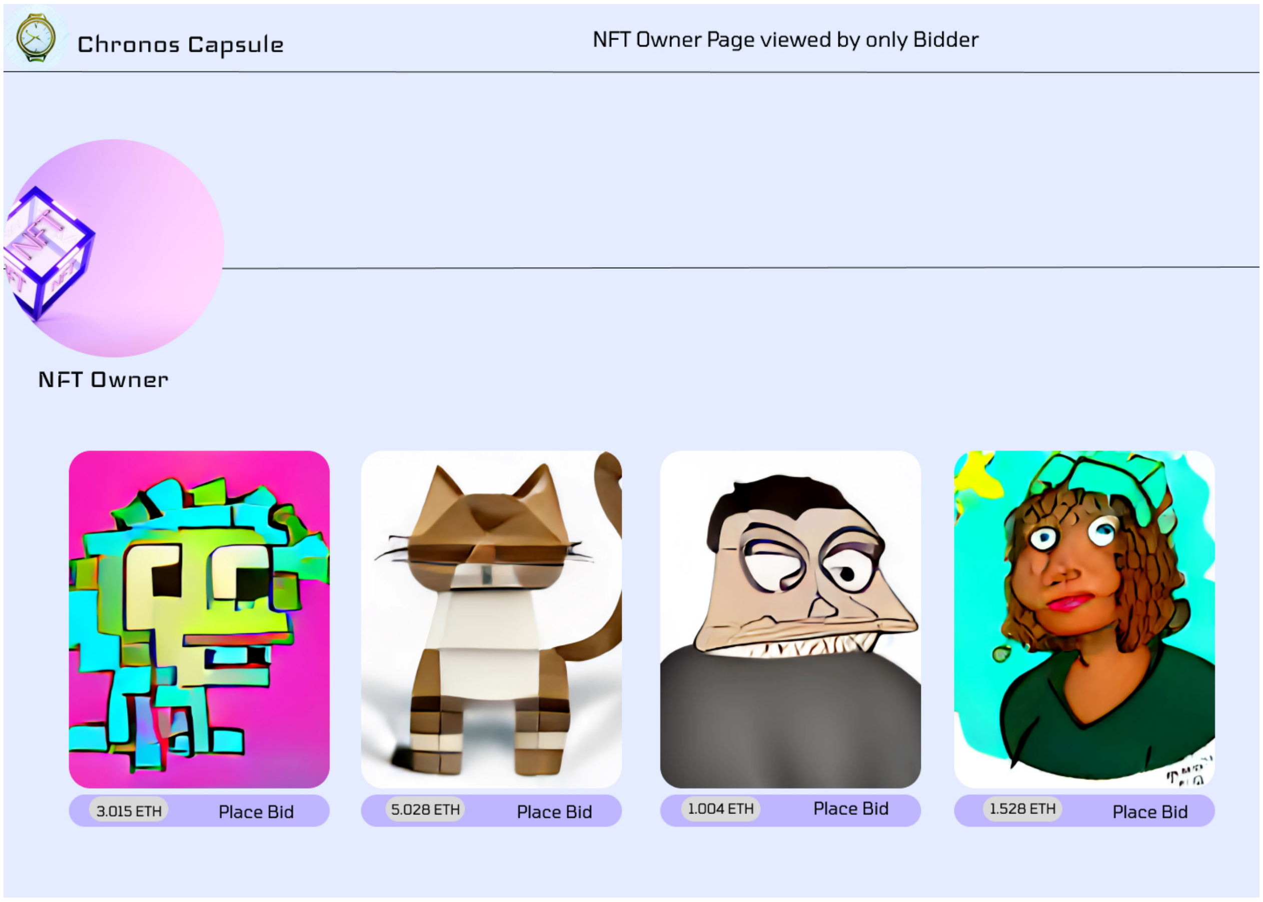


Figure 2: This is the biding page, on which the bid can be placed.

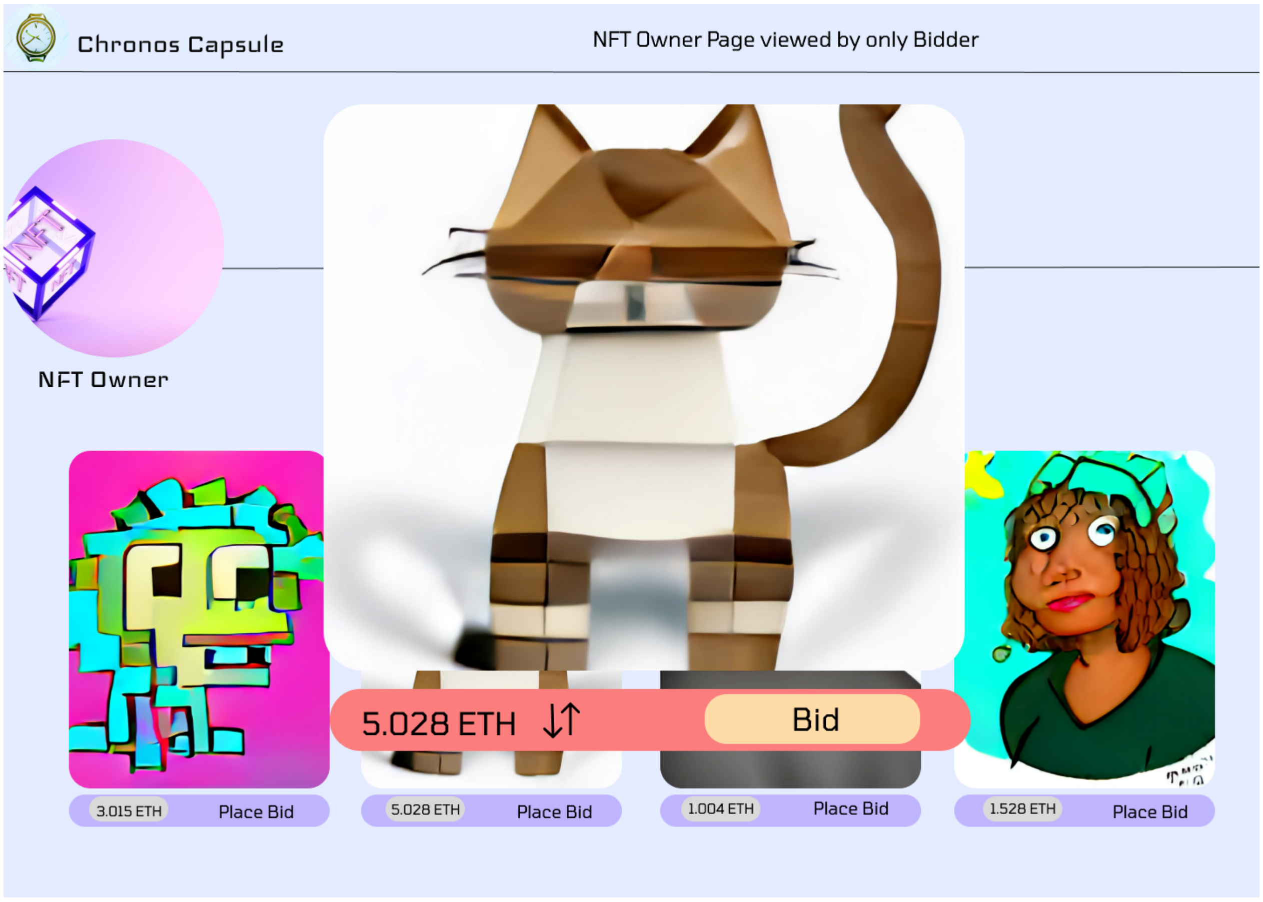


Figure 3: This is the second stage of the biding page.

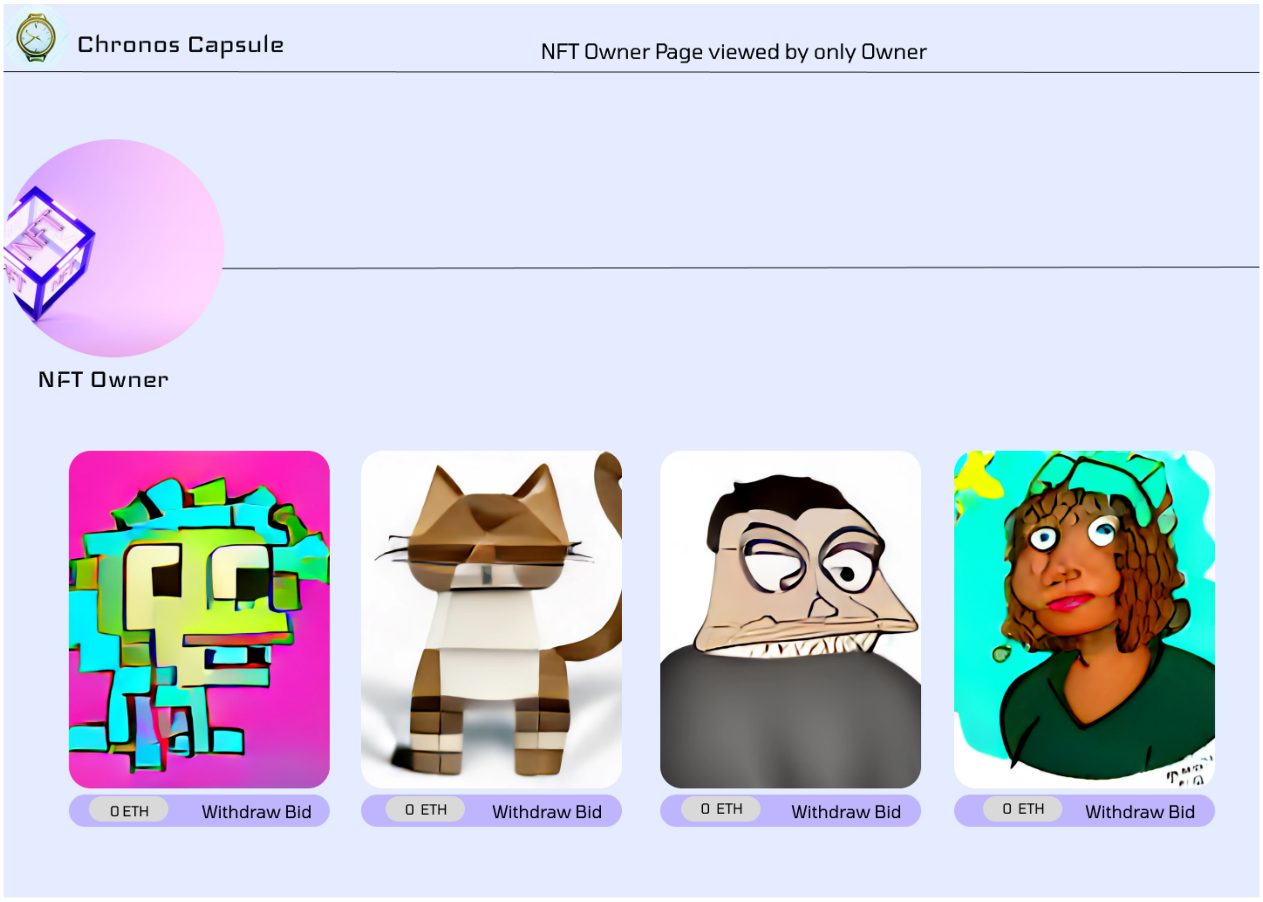
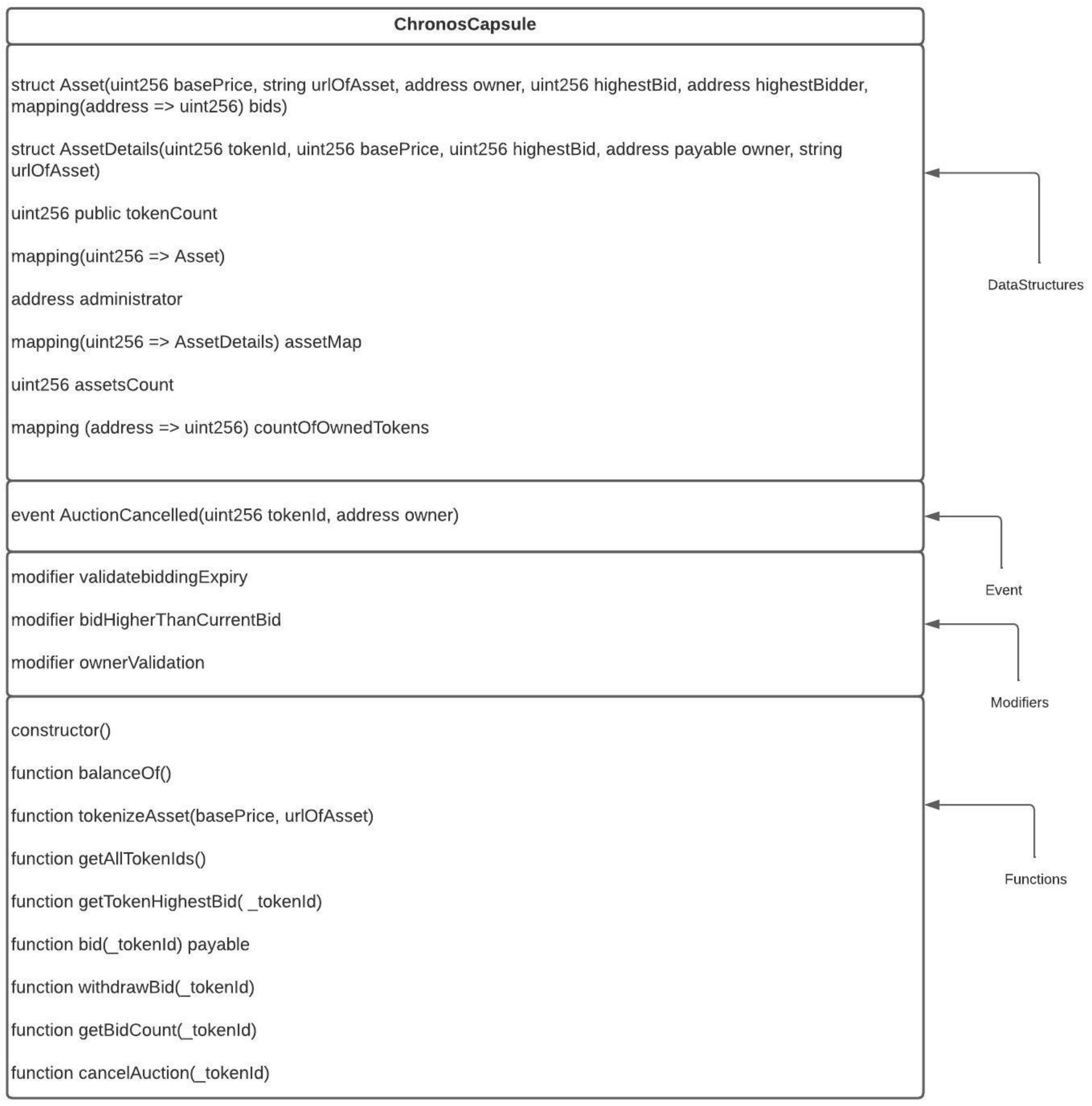


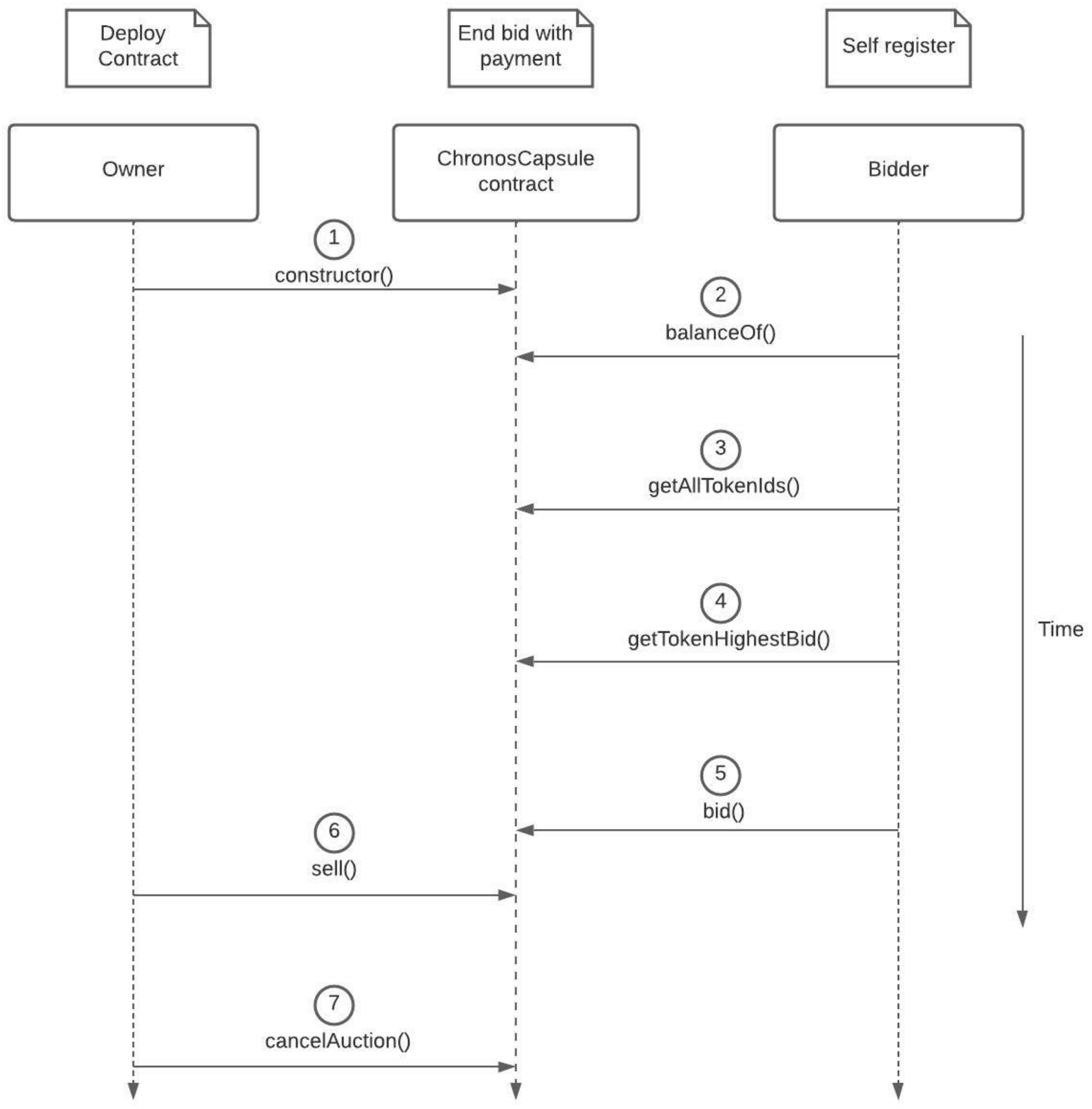
Figure 4: The page which only owner can access and use to withdraw bid.

6 Smart Contract diagram

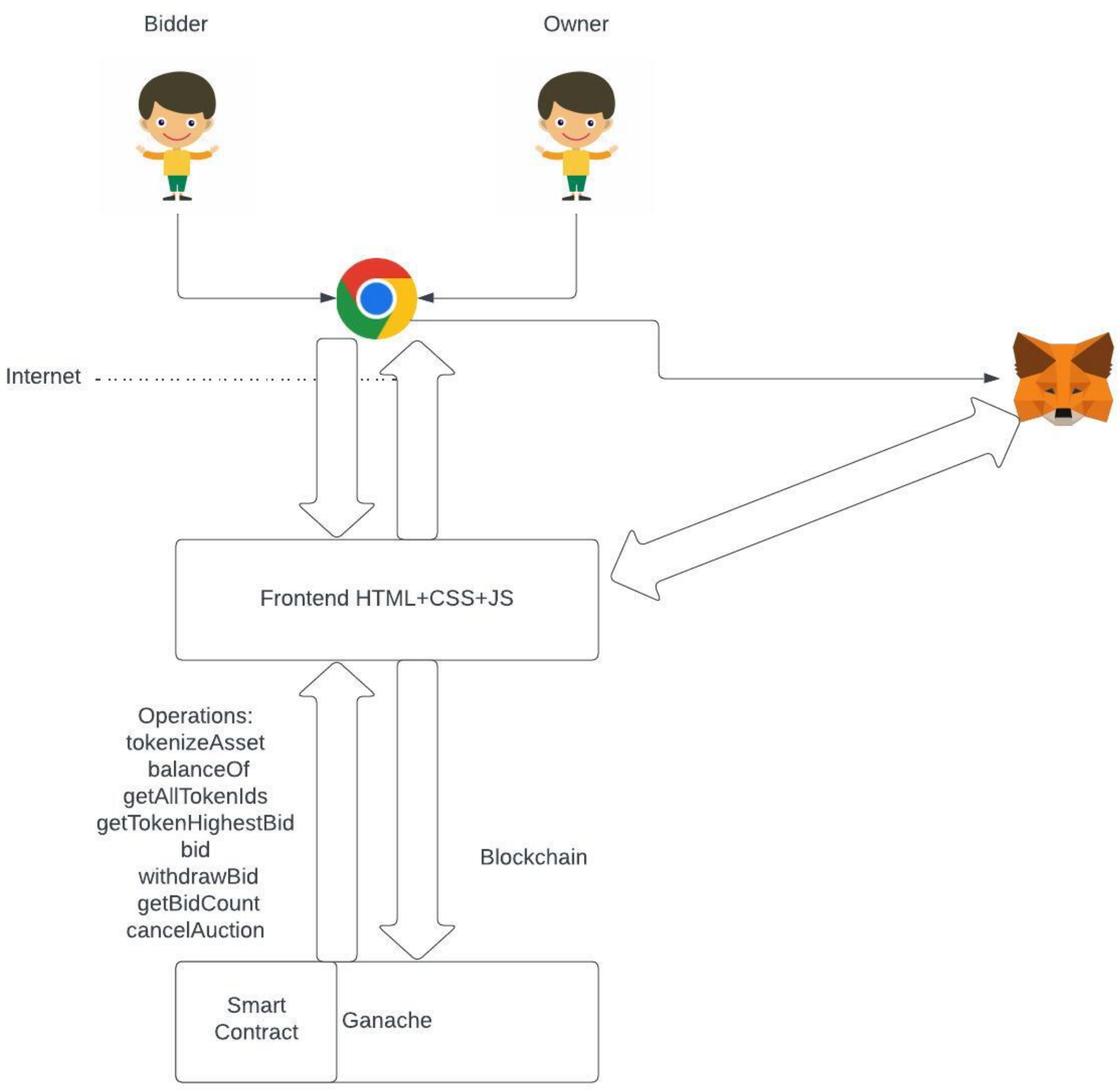


ChronosCapsule
struct Asset(uint256 basePrice, string urlOfAsset, address owner, uint256 highestBid, address highestBidder, mapping(address => uint256) bids)
struct AssetDetails(uint256 tokenId, uint256 basePrice, uint256 highestBid, address payable owner, string urlOfAsset)
uint256 public tokenCount
mapping(uint256 => Asset)
address administrator
mapping(uint256 => AssetDetails) assetMap
uint256 assetsCount
mapping (address => uint256) countOfOwnedTokens
event AuctionCancelled(uint256 tokenId, address owner)
modifier validatebiddingExpiry
modifier bidHigherThanCurrentBid
modifier ownerValidation
constructor()
function balanceOf()
function tokenizeAsset(basePrice, urlOfAsset)
function getAllTokenIds()
function getTokenHighestBid(_tokenId)
function bid(_tokenId) payable
function withdrawBid(_tokenId)
function getBidCount(_tokenId)
function cancelAuction(_tokenId)
DataStructures
Event
Modifiers
Functions

7 Sequence diagram



8 Architecture diagram



9 Deployment steps

Step 1:: Open the ChronosCapsule-DApp directory, go to ChronosCapsule-Contract directory and run CMD.
Step 2:: Run command 'truffle compile' to compile the contract.
Step 3:: Run command 'truffle migrate –reset' to deploy the contract. (Note: Don't forget to run Ganache on port 7545 before contract deployment)
Step 4:: Go back and then go to ChronosCapsule-App directory and run CMD there.
Step 5:: Run command 'npm install'.
Step 6:: Run command 'npm start'.
Step 7:: The application will be running on localhost:3000.

10 References

1. Professor's lecture slides
2. https://docs.soliditylang.org/en/v0.8.19/
3. https://www.figma.com/
4. https://app.diagrams.net/
5. https://www.overleaf.com/
6. https://stackoverflow.com/
7. https://ethereum.org/en/developers/docs/
8. https://remix.ethereum.org/
9. https://metamask.io/

11 Approved by

Nilkumar Dhamecha