

## **Final-project Report**

### **Decentralized Application to lock NFTs**

#### **Group Members**

Michael Buchar .....	260831528
Dominic Chan .....	260904794
Ezra Gomolin .....	260926917
Ege Karadibak .....	260830803

#### **Project Advisor**

Dr. Majid Babaei, McGill - Faculty of Engineering (majid.babaei@mcgill.ca)

## **Abstract**

With the rise of NFTs also came many attackers against specific individuals in order to retrieve their valuable assets and capitalize off them. Inadvertently clicking on a hyperlink or approving a seemingly authentic transaction has resulted in significant financial losses ranging from hundreds to hundreds of thousands of dollars for numerous individuals. Our project holds immense significance and worth, as we aim to enhance the security measures surrounding people's assets, thus curtailing several of these malicious assaults. Our ultimate objective is to develop an all-inclusive decentralized application that enables users to interface with the blockchain via our web application. Our web app will display all their NFTs and enable users to lock them up in a smart contract that only releases them to their rightful owner after the time they requested has passed. The underlying drive of this project is to facilitate individuals with a safeguarded means of possessing their assets, which may hold considerable value.

## Table of Contents

Final-Project Report.....	1
Decentralized Application to lock NFTs.....	1
Group Members.....	1
Project Supervisor.....	1
Abstract .....	2
List of Abbreviations .....	4
Introduction/Motivation .....	5
Background .....	5
Blockchain.....	5
Ethereum and Gorell.....	6
Metamask Wallet.....	6
Smart Contracts.....	7
Gas Fees.....	8
NFT.....	8
React Js.....	9
Requirements/Problem .....	10
Design and Result .....	11
Impact on Society and on the Environment .....	16
Use of Non-Renewable Resources .....	16
Environmental Benefits .....	16
Safety and Risk .....	16
Benefits to Society .....	16
Report on Teamwork .....	17
Conclusion .....	18
References .....	20
Appendices .....	22

## **List of Abbreviations**

NFT – Non-Fungible Token

API - Application Programming Interface

SC - Smart Contract

UI - User Interface

## **Introduction/Motivation**

Since blockchain was invented and numerous cryptocurrencies and other tools started using this technology, it has become rather evident that this technology has enormous potential. Blockchain is highly popular and seems not to be going away anytime soon, so it is crucial to consider it an essential technology. Moreover, many people use this technology for malicious activities, and scams in the crypto space have skyrocketed in the past years [16].

Blockchain is widespread, and there are so many scams in the crypto space; we use NFTs and saw some problems in this technology. This project aims to increase the security aspects of NFTs for users from possible phishing attacks.

## **Background**

As blockchain technologies are growing at an exponential rate while still being so new it was important for us to learn a lot of theory in terms of why our project is important. There were several topics that we all had to learn which include what is a blockchain, why is the blockchain and NFT are important and what can it be used for. How does a smart contract work? The following section will be outlining these topics and important information for our project and for anyone just getting started to understand what our project is, how it works and how it is powered.

### *Blockchain*

A blockchain is essentially a public digital ledger that records transactions and ownership of digital assets [1]. It is a ledger that is duplicated and distributed across all systems on the network, making it a decentralized system. In other words, it is a financial book that notes down all records of transactions, and a copy of the book is given to everyone.

As such, this is one of the reasons that makes blockchains extremely useful, as it makes blockchains extremely hard to hack. This is because tampering a single ledger on a computer would simply change one of the many records on the whole network, and since the ledgers of the network would need to reach consensus for each transaction to occur, hacking just one of the ledgers on the network would not work. In fact, one would need to alter at least 51% of the blockchains in the system for the ledger to be altered [1]. In addition to their reputation for safety and security, blockchains have gained widespread popularity due to their ability to eliminate the involvement of intermediaries in transactions. This is accomplished through the use of smart contracts, which execute the entire transaction automatically and obviate the need for third-party intervention. Furthermore, the immutable nature of smart contracts on the blockchain ensures complete transparency of the process, as they are publicly visible and cannot be altered. Therefore, the use of blockchain also reduces the amount of time because all of the processes are streamlined and resources needed to complete each transaction, making transactions quick and efficient. As a result, these properties make blockchain technology very desirable for a variety of online applications, as it is a safe, secure and efficient way of conducting online transactions in a decentralized setting.

The fundamental mechanism behind blockchains involves the formation of a sequential chain of data blocks, wherein each transaction is recorded as a new block of data. These blocks are then disseminated across the network and subject to validation by every computer. Following successful validation, the blocks are added to the blockchain, and their positions become firmly fixed within the chain, owing to their interconnectivity with the previous and subsequent blocks[2]. Consequently, once added to the blockchain, these blocks become immutable and irrevocable.

### **Ethereum and Goreli**

Ethereum is a blockchain network that offers support for the seamless execution of smart contracts and is backed by its own cryptocurrency known as Ether. Conceptually, the Ethereum blockchain functions as a platform that enables the verification and execution of code, in the form of smart contracts, when triggered by transactions. By harnessing the potential of the blockchain, individuals are empowered to construct decentralized applications that operate within a global, distributed database, thus ensuring complete transparency and availability of information to all stakeholders. Notably, each transaction executed on the Ethereum blockchain incurs a cost in Ether, and a transaction fee is levied for every transaction. This fee is subsequently paid out to the miners who locate a spot on the public ledger (blockchain) for the transaction to be recorded. Transactions on the Ethereum blockchain record all pertinent details, such as the value of Ether transacted and the transfer of NFTs, including the identification of the public keys of the sender and receiver.

For our project we will be using the Goreli blockchain which is an identical copy of Ethereum's blockchain although its native currency Goreli Ether has no actual value and is used widely for testing and development of ethereum decentralized applications [4]. It is extremely easy to transfer our application to Ethereum once our application is tested and finalized as all it requires is publishing our smart contract on the Ethereum blockchain and having our web application trigger actions over there.

### **Metamask Wallet**

Metamask is a cryptocurrency wallet that is used to interact and trigger transactions on the blockchain. As metamask is widely used in the NFT space and it supports Ethereum and Goreli blockchain, this is the wallet that our users must have in order to use and interact with our application [5].

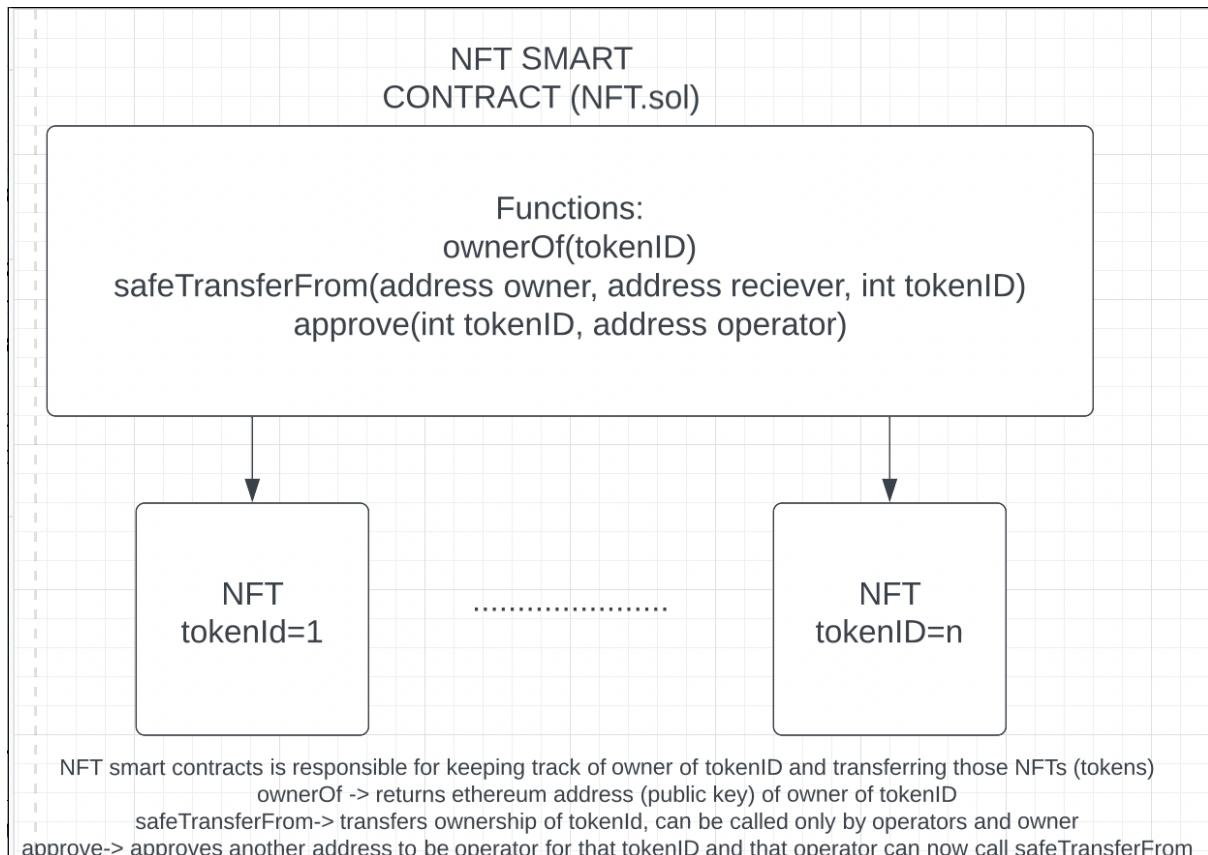
For a user to interact with our application they are required to have a metamask wallet and have the extension downloaded on their browser, some Goreli Ether to initiate transactions and at least one NFT to be locked up.

### **Smart Contracts**

A smart contract is a program that exists on the blockchain and is responsible for executing a given set of actions depending on certain requirements set by the author. These actions are processed via transactions on the blockchain which is a distributed database and cannot be altered, all transactions are recorded on the blockchain and are visible to anyone to see for all of eternity.

As our application is based on NFTs it is important that we understand how NFT smart contracts work although smart contracts are not limited to just NFTs the scope of our project requires us to understand NFT smart contracts.

In terms of NFTs, smart contracts are responsible for keeping track of ownership of a token and keeping track of who is able to transfer their ownership (based on ethereum public key). The contract is responsible for transferring ownership of NFTs when an owner decides to invoke that transaction. Another important aspect to a NFT smart contract is that an owner of an NFT can delegate transfer rights to another individual which allows that other individual to invoke the transfer function on their behalf [6]. Figure 1 below describes how a NFT smart contract works and is important as our smart contract will be responsible for triggering functionality from the original NFT smart contract.



*Figure 1: Architecture of NFT Smart Contract*

Aside from the actual NFT smart contract, there are several other practical uses for them. For example, if someone wants to create a marketplace for buying and selling NFTs a smart contract must be created. This smart contract can invoke the transfer function of the NFT contract if they have transfer rights of that token (delegated by the owner). With that system in play they can create a set of requirements within the smart contract that ensures that the buyer has paid the asking price of the seller. That the seller has delegated transfer rights to the marketplace contract and once all those requirements have been achieved then the smart contract will invoke the transfer of the NFT to the buyer.

It is quite important for us to understand how these types of smart contracts work as our application will also require the creation of a smart contract that executes based on a set of requirements that ensure security and trust over the user's assets as our contract will become the owner of these tokens.

for the time being and we must ensure that the tokens get sent back to the rightful owner. The design of our smart contract will be outlined in the design section. We will also end up creating a NFT smart contract in order to create some NFTs for all of our wallets to be able to test our application.

### Gas Fees

Gas fees are fees that are charged to anyone executing any transaction on the blockchain, whether sending cryptocurrencies to a different wallet or interacting with a smart contract [7]. When performing the transaction they can either execute or fail, regardless if a transaction fails the users gas fees are lost. A large reason for failure is when a user sends a transaction to a smart contract if they do not pass all the requirements the transaction will fail. Anyone is able to execute any transaction to a smart contract but if you do not meet the requirements in the smart contract it will be a waste of funds. That is a large reason why our applications front end must not allow the executions of smart contract functions that would lead to a failure.

### NFT

NFT stands for Non-Fungible Token. NFTs are unique cryptographic tokens that exist on the blockchain.

As the name states, NFTs are not fungible. This means that, unlike cryptocurrencies like Ethereum and Bitcoin, they cannot be traded equivalently. Each NFT has its metadata to differentiate it from others, making each and every one unique. Since each NFT is one of a kind, NFTs can be really valuable and expensive. One of the most expensive NFTs was sold for \$91.8M [8]. When you buy an NFT, even though you don't get ownership of the NFT, you get to hold, share and trade the original copy. Bought NFTs cannot be copied since each NFT has its own unique metadata, so creating a copy would result in another unique NFT, which is not the same as the other one.

In order to obtain an NFT, there are two possible routes: one can either go through the minting process, or one can buy the desired NFT from different marketplaces. Minting an NFT means publishing that unique NFT to the blockchain for it to be traded. The mint happens through a smart contract. Smart contracts assign the owner of the NFT and manage the transfer of NFTs. In the high level of things, when minting happens the code in the smart contract executes and a new block has been created for the new NFT. Then the validation happens through the smart contract again and after validation, the information is recorded into the blockchain.

In the case of the second option, there are many different NFT marketplaces in which trading can occur. NFT marketplaces are web platforms based on blockchain created for people to store and sell their NFTs. Just like there are auctions for real artwork, NFTs are sold by bidding so, the highest bid ends up owning the NFT when the specified time is over by now. There are numerous marketplaces available for people to trade and buy NFTs. One of the main, and largest, marketplaces for ETH blockchain is called OpenSea.

NFTs are utilized all around the globe and have so many different use cases. An NFT can represent anything from artwork to collectibles, to real estate, to tickets, even to real-world events, etc.

One of the main use cases of NFTs is to represent artwork. NFTs allow artists to create their artwork digitally and put it on auction through existing marketplaces. The ownership of an NFT cannot be changed. So when someone buys that NFT they will get the token hold, trade, or display.

Another use case of NFTs is through the metaverse. Metaverse is a virtually shared digital reality. Some believe that it will be the new iteration of the internet. NFTs can represent land in the metaverse and people can buy that NFT to own the specified land in the metaverse.

NFTs are also commonly used for people to collect collectible items. NFTs offered collectors a new way of collecting collectibles. Allowing collecting collectibles digitally, as each collectible is an NFT on its own. This offers easier accessibility than physical collecting to the collectors. It also minimizes possible damages and adds another security layer to the collectibles.

## *React JS*

The role of react is to handle the view layer of our application. React is an open-source javascript library developed by Meta (formerly known as Facebook) [9]. We will be using react for our web development in order to create an interactive user interface efficiently. Using react will let us write less code compared to just javascript which will help us implement the website more smoothly.

React lets us build a single-page application that loads only a single HTML document on the first request. Then, it updates the specific portion, content, or body of the webpage that needs updating using JavaScript [9]. This way our user doesn't have to reload the full webpage to get a new page each time a user makes a new request. Instead, React intercepts the request and only fetches and changes the sections that need changing without having to trigger a full page reload. This approach results in better performance and a more dynamic user experience [9].

React relies on a virtual DOM, which is a copy of the actual DOM. React's virtual DOM is immediately reloaded to reflect this new change whenever there is a change in the data state. After which, React compares the virtual DOM to the actual DOM to figure out what exactly has changed [9]. This way, react reflects the changes without requiring reloading the whole page.

## **Requirements/Problem**

Over the past year, NFTs have been rising in popularity. What may look like any digital art is being treated as a valuable asset due to the power of the blockchain and project founders providing digital and real-life utility through the technology. This has caused people to deploy large sums of money as an investment. NFT prices can range from free to hundreds of thousands and even millions of dollars. For example, the most expensive NFT ever sold went for 91.8 million dollars [8].

Whether or not you believe in the technology, it is evident that NFTs are here to stay, and people need to keep their assets safe. Many problems can arise with NFTs, such as false promises by project founders, causing people to invest in a scam. Aside from that, there are so many phishing scams that cause people to lose all their valuable assets because of a click of a link or interacting with a compromised smart contract. Our application is designed to prevent such scams by allowing users to store their assets in our custom build smart contract for as long as they desire. This makes our smart contract the temporary owner of the NFTs. Our application also has further benefits aside from our primary goal, including preventing people from selling their assets too early. As our contract only executes actions based on a predefined set of requirements, we are required to ensure reliability and trust.

There are several requirements that we are responsible for implementing in our contract as well as on our frontend application.

On the smart contract side, we must ensure that the individual who is locking their token must be the only person able to unlock it. We must ensure that the token is only able to be unlocked after the time they have set for locking has passed. We must ensure that our contract keeps track of any NFTs that any user locks up.

On the frontend side, we must ensure that users only have the option of locking and unlocking NFTs they own. Although our smart contract will prevent the execution of locking and unlocking tokens you do not own, it is essential our frontend does not allow for non-executable transactions, as any transaction on the blockchain costs gas fees, which will be wasted. We also must ensure that we display and send the accurate details of a user-owned NFT to our smart contract.

## Design and Results

For the design of our application we have used react.js framework for our front end, react was used as it allows for the use of existing Web3.js libraries to easily allow us to connect our website to the blockchain as well as easily interacting with external APIs. The following section will contain a walkthrough of how our application is designed.

A user first signs into our application using their ethereum public key. When a user clicks on sign in, it prompts metamask using Web3.js library and allows the user to connect to our website.

Once connected when a user wishes to view all their NFTs, our website uses Alchemy's API (a free API for NFTs) in order to retrieve all the NFTs that a user owns and display them as a gallery. Figure 2 shows a diagram outlining this process.

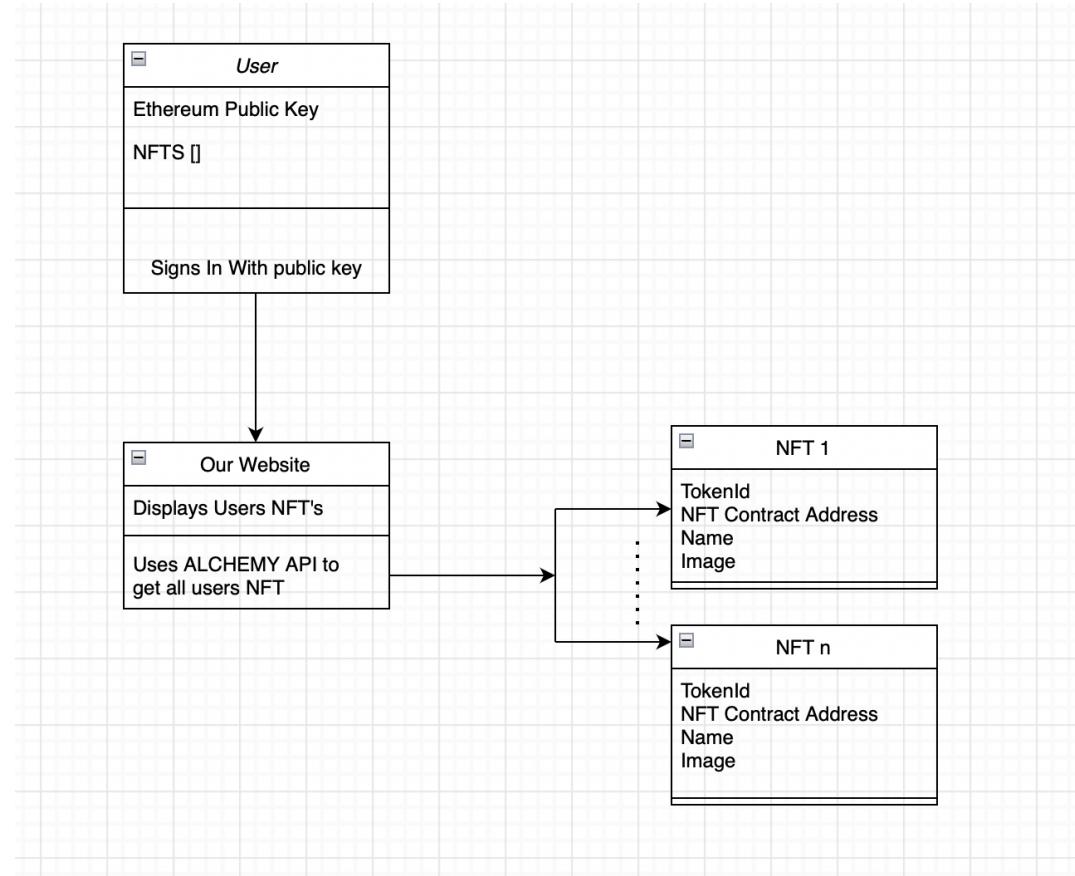


Figure 2: Process of signing in and displaying NFTs

The next step is for a user to choose a NFT, and input the number of days in which they wish to lock it for. Once the user confirms the relevant information is sent to our smart contract and the transaction gets executed, storing that chosen NFT in our smart contract for that period of time. Figure 3 outlines this process.

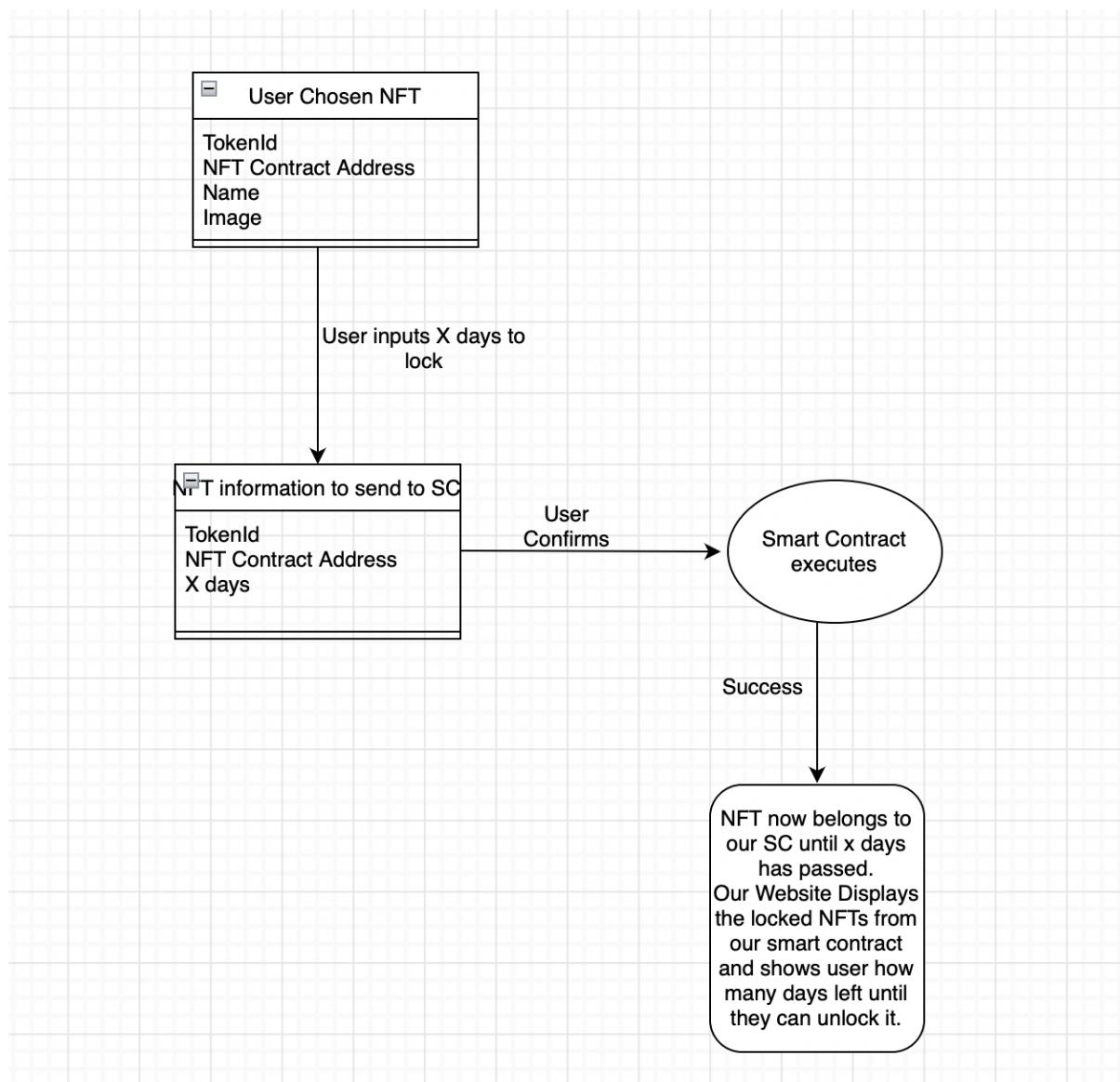


Figure 3: Process of locking a NFT

The next step is for a user to choose a NFT that is locked and return it to him after X days. Figure 4 shows this process.

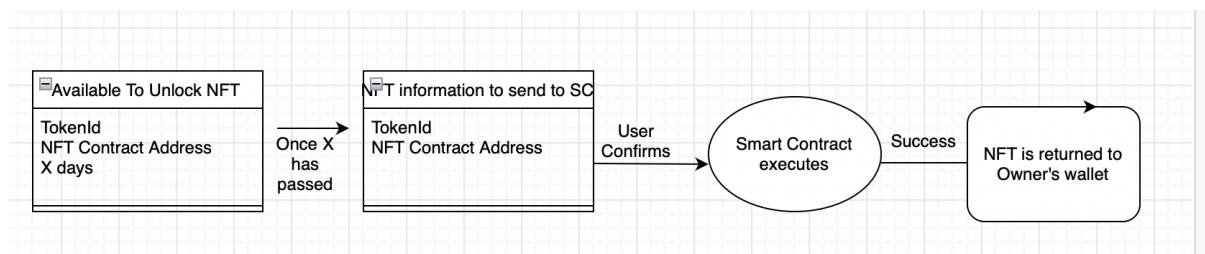


Figure 4: Process of unlocking a NFT

A user can sign in to our application with their metamask wallet and our website displays their NFTs as a gallery. The user can browse through this gallery and can input how many days they want to lock their NFT for. Everything is ready to be sent to the smart contract. Once the user hits the lock button metamask will be prompted open and the user will confirm the blockchain transaction, the result is the NFT gets transferred to our smart contract and is now displayed on the right side of the gallery (see appendix below). The architecture of our smart contract has been designed and deployed and is outlined below.

In order to make sure our contract is secure and safe for users our contract must keep track of several things. For each user we must create our own data structure and have a list of that data structure for each user to keep track of which NFTs they have staked, for how long and what NFT smart contract they belong to. Most staking contracts at the moment just have to store a list corresponding to tokenIDs that a user is staking because it is hardcoded into their contract which NFT collection (their own) can be stored in their contract.

Our contract has two main functionalities, one being locking a token and the second being withdrawing a token. Figure 5 outlines the design of our smart contract which will be developed next semester. Utilizing the Web3.js javascript library we will be able to send transactions to our smart contract from our front end.

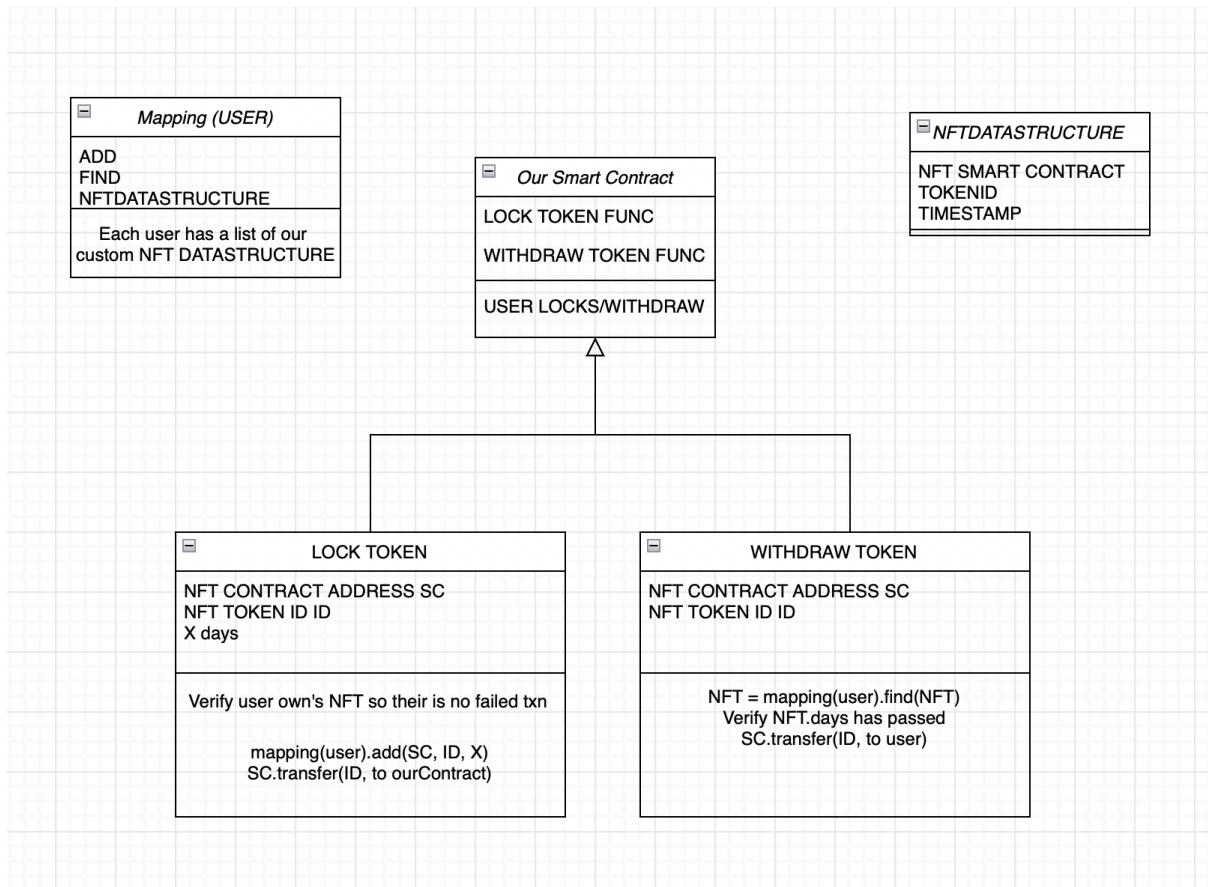


Figure 5: Smart Contract Design

For security reasons it is important that any NFT that a user withdraws can only be returned to the same person that locked it. Using a mapping between users and a list of our custom data structure it allows us to keep track of which NFTs belong to who.

*Our Contract (See Appendix for full source code)*

```
struct Asset721 {  
    uint256 tokenId;  
    uint256 withdrawDate;  
    IERC721 add;  
    string imageLink;  
}  
  
mapping(address => Asset721[]) public staked721;
```

The struct is our data structure we created which defines a NFT object. Within the struct we have the tokenId and ‘add’ which as mentioned above are required for transferring NFTs. The withdrawal date will be used to verify the time has passed when a user wants to withdraw their token. The imageLink is so we can display the token to our front end.

The mapping is a crucial part of the application as it maps a list of our custom data structure for every user (ethereum address). This is important and used for locking and unlocking NFTs by adding and removing NFTs to this list.

```
function stake721(uint256 _tokenId, uint256 _stakeFor, address _add,  
string memory _imageLink) public {  
    IERC721 nft = IERC721(_add);  
    nft.safeTransferFrom(msg.sender, address(this), _tokenId);  
    staked721[msg.sender].push(Asset721(_tokenId, block.timestamp +  
_stakeFor, nft, _imageLink));  
}
```

This is the locking functionality of our application, this function is responsible for transferring the NFT with all the important details to our smart contract, storing the NFT into that user’s mapping for later withdrawal.

```
function withdraw721(uint256 _tokenId, IERC721 _add) public {  
    Asset721 memory deleteFromPool;  
    Asset721[] storage assetsStaked721 = staked721[msg.sender];  
    for (uint i = 0; i < assetsStaked721.length; i++) {  
  
        //verifies asset you're looking for  
        if(assetsStaked721[i].tokenId == _tokenId &&  
assetsStaked721[i].add == _add){
```

```

    //verify time has been met
    require(block.timestamp > assetsStaked721[i].withdrawDate,
    "Time Not Up");

    //get address
    IERC721 add = assetsStaked721[i].add;

    //transfer token
    add.safeTransferFrom(address(this), msg.sender,
assetsStaked721[i].tokenId, "0x00");

    //remove from pool
    deleteFromPool = assetsStaked721[i];
    assetsStaked721[i] = assetsStaked721[assetsStaked721.length
- 1];
    assetsStaked721[assetsStaked721.length - 1] =
deleteFromPool;
    assetsStaked721.pop();
    break;
}
}

}

```

This function is responsible for unlocking a user's NFT. When a user selects an NFT to be unlocked this function iterates through the mapping and finds that NFT. It then verifies that the withdrawal date has passed. It then transfers the token back to the user and removes the NFT from the mapping.

## *Security Design*

Security is a very important component for our application, and when designing our smart contract we had to ensure it was easy to understand, robust and simple. The most important aspect of our application was to ensure that the NFT that a user locks up can only be returned to that user. This is really the only security concern and to ensure that it was important that we searched the mapping and added to the mapping of the user msg.sender. msg.sender in solidity is essentially the user that has submitted the transaction on the blockchain; this means that no one can ever remove or add NFTs to the mapping of anyone but themselves as that only occurs upon a transaction. In the future for our project to grow we will need to consider further security aspects in terms of expanding our product.

## **Impact On Society and Environment**

### *Use of Non-Renewable Resources*

Throughout the project, our team has been using emails for online communication with our supervisor, and we have been using zoom to meet among the team and also with our supervisor to give updates on the project. However, these technologies have a carbon footprint. A regular email with a photo/attachment can cause 50g of CO<sub>2</sub> emission [10]. The standard-definition video calls on zoom would emit 2.4g of CO<sub>2</sub> [11]. Since our project is about locking NFTs from the Ethereum blockchain, there will be a carbon footprint due to the NFT transactions. There are also varying views about how much energy is released through the production of NFTs. Digiconomist estimates that a single Ethereum transaction's carbon footprint is 33.4 kg CO<sub>2</sub> [12]. Artist and programmer Memo Akten estimates an NFT transaction has a carbon footprint of about 48 kg CO<sub>2</sub> [12]. Thus, the main environmental impact will be caused by NFT transactions.

### *Environmental Benefits*

For the sake of our project, we will be accepting to lock NFTs from the Ethereum blockchain. For each transaction –including locking to be accepted by the network, it must be included in a block and that block must be added to the chain by the validator. When Ethereum was first introduced, the intentions were to use proof of stake instead of proof of work like other cryptocurrencies to validate and add a new block to the chain[13]. The proof of work blockchains uses a consensus algorithm where blocks are validated by solving an energy-intensive computational puzzle [14]. The energy consumption depends on the complexity of the block that is being added to the chain. The computational power of a miner is measured by its hash rate. The difficulty of a block is proportional to the total hash rate of the network [14]. This might end in a high computational power which emits more greenhouse gasses. However, Ethereum is using proof of stake. This means that the creation of the native token Ether will not require energy-intensive computers. In theory, this will result in a 99.95% reduction in energy consumption by the Ethereum network [15]. This way Ethereum is more environmentally friendly compared to other cryptocurrencies. Thus, by only locking NFTs that exist on the Ethereum blockchain, we will make sure the transactions are environmentally friendly and sustainable in the long run.

### *Safety and Risk*

The main idea of our project is to offer more safety to our user's assets. There are a lot of people out there trying to scam people for investing in scam NFTs or sending phishing links to steal their assets. In order to prevent this from happening we offer to lock our user's NFTs, which means that the smart contract that we will implement will be the owner of that NFT. We will be taking full caution on the security of the system to provide a safe experience for our users.

### *Benefits to Society*

Our product offers great security measures for people that have NFTs. Our main goal is to offer an extra security layer that owns digital assets because there are a lot of people whose assets were stolen by hackers. One common way hackers steal NFTs is through deception; they trick an NFT holder into transferring their assets to them or sharing access to their digital wallet. This often happens in emails or direct messages. Someone with a fake profile might convince someone to transfer assets into a different digital wallet. Or they might send a phishing link the NFT owner clicks on, and then they share their private key [16]. However, even if someone falls for this, if an NFT is locked through our website hackers will not be able to steal it.

## **Report on team work**

Despite not knowing each other before this project started, our passion for software development brought us together to solve this unique and exciting blockchain challenge and create a website with various functionality similar to previous software development courses at McGill. We held bi-weekly virtual meetings, and every team member attended without exception. Every team member contributed equally to development and documentation so that everyone understood the project as a whole and its impact. As we all have different backgrounds and experiences, we need to fill the gaps in knowledge and support each other along the way. With all of us having worked with different frameworks and technologies, we understand that we have to help one another along the way by each determining our areas of expertise. On another note, we have never worked on a project of this scale to which deadlines, standards, as well as the project idea were based entirely on our own decisions, so it was a valuable experience for us, and we think we have done a great job keeping track of progress and meeting deadlines. Overall, each team member collaborated well and did not encounter any significant difficulties. The team dynamics have been more than satisfactory.

Ezra has worked with NFTs and blockchain over the past year and has an excellent technical understanding of the technology. He has also worked with smart contracts, a valuable experience for our project. Besides that, he had a lack of experience with frontend development but expanded his skills greatly in that area. Ezra has been our team's great leader and mentor since he knows the most about NFTs and smart contracts. Ezra set up the React application, worked on the architecture of how our software will work, and helped with the image gallery UI.

Ege has knowledge about NFTs and blockchain and developed this as a personal interest. He has bought and traded NFTs in the past; however, this is the first time he has got a chance to work on implementing anything related to blockchain. Ege has more experience in frontend development from his previous affairs, but this was an excellent chance for him to ameliorate his skills. He has led our frontend development efforts and has been a great overall resource for web development. Ege implemented the UI mockups in Figma and developed the UI for the home page.

Michael has never worked with React.js, nor has he traded any NFTs, but React has been on his radar for quite some time now, and he was glad he finally got to delve into it. This project accelerated his understanding of blockchain, NFTs, and smart contracts, which he is excited about. Michael learned about Alchemy API, Etherscan API, Opensea (NFT market), and Metamask (Cryptocurrency wallet). Moreover, he attended some YouTube tutorials online to ensure he had the same knowledge level as his teammates regarding this project. He implemented parts of the front end with Ege, worked on setting up the ETH address, and contributed to the logic in the backend as well.

Similarly to Michael, Dominic has not worked with NFTs or blockchains in general, so this is a new area of knowledge for him. This, coupled with his inexperience in frontend development, made this project a big challenge for him, but he quickly learned and developed skills in this area. Dominic self-learned React.js, NFTs, and blockchain to ensure that he had an adept and fundamental understanding of each of these topics before fully immersing himself in the project to be a valuable member and fully contributing to the team effort. Dominic worked on displaying the NFTs in our application, as well as worked on the image gallery UI.

## Conclusion

In the past two semesters, from the project proposal to the actual implementation of our project, we have managed to accomplish many tasks, fulfilling our goals and objectives throughout. Although we have faced a multitude of challenges and issues, with time and effort, we were able to work together as a team to overcome these problems, allowing us to ultimately finish and complete our project with a good and working solution.

For the first semester, we initially spent a lot of time conducting research on the subject matter, ensuring that everyone fully understood the key concepts and fundamentals of the project. This helped us in meticulously planning our tentative schedule and general tasks in the next year or so, giving us an idea of what was to come. Furthermore, before we started with the actual implementation of the project, a lot of time was spent on increasing our background knowledge and developing skills in using the technologies required for our project, such as different crypto APIs, react.js, javascript as well as smart contracts. Once it was ensured that the knowledge required to complete the project was attained, we started on the design of our project.

Since the design step of the project is integral to the implementation of the website, we did further research on existing website examples to gain inspiration from them, and after a period of time, we were able to come up with a general design along with the functionalities of each aspect of the website. The graphical designs and diagrams we created were able to illustrate exactly how we wanted each page to look and work respectively.

Moving on to the implementation, we first created the general framework for the website; a react.js application that consists of several pages with a crypto API connected to the site. Since without a proper fundamental working website, where the backend architecture works exactly as intended, our website would not be able to function correctly. As such, we prioritized building and constructing the basic frontend of the website, setting requirements and goals to be met and fulfilled before integrating it with the backend. As soon as the basic frontend framework works to expectation, we implemented the basic backend portion of the website, making the website look and function to that as entailed in the design diagrams.

After the christmas break, we moved on to further implement and develop the functionalities of our website in its backend. By self-learning Solidity, which is the programming language used for constructing and designing smart contracts for blockchains, we were able to eventually design and code our very own smart contracts fitted for our purpose, implementing the NFT storage functionality of our application and website. After undergoing trials and tests in order to benchmark and verify that the smart contract works to expected standards, we then deployed it onto an official blockchain. The deployed smart contract is then also integrated onto the website, successfully kickstarting the launch of our website. Note that throughout the semester, minor frontend and backend aspects of the website have been improved and fine tuned as well, ensuring that any integration, compatibility and user interface issues are resolved.

As we have reached the end of this project, all of us collectively have learned a lot in the past year from just completing this capstone project. With most of us being immersed in a completely new field in NFTs and Blockchains, not only were we able to develop and learn completely new skills and technical knowledge, but we were also able to experience and understand the fundamental process

behind conducting a project, including the intricacies of working together, planning ahead, completing tasks (on time) and doing documentation.

For our next steps, in order to expand the depth of this project as a whole, we will be considering the factors of scalability and security design in the project, as these are essential considerations for increasing the size and level of our project. As such, going forward we will consider incorporating the use of Model-Driven Engineering (MDE) techniques, as these techniques can be employed to help with complex design decisions, helping further overcome design, safety and security challenges that may exist in our current project. Furthermore, through the use of the Business Process Model and Notation (BPMN), concerns with the scalability of our project can also be addressed. By outlining the entire business process of the entire project with careful considerations and precise processes, the scalability of the project can be rapidly evaluated and improved using these tools.

## References

- [1] "What is blockchain and how does it work?," *Synopsys*. [Online]. Available: <https://www.synopsys.com/glossary/what-is-blockchain.html#:~:text=A%20blockchain%20is%20distributed,a%20timestamp%2C%20and%20transaction%20data>. [Accessed: 01-Dec-2022].
- [2] "What is blockchain technology? - IBM Blockchain," *IBM*. [Online]. Available: <https://www.ibm.com/topics/what-is-blockchain>. [Accessed: 01-Dec-2022].
- [3] J. Frankenfield, "What is ethereum and how does it work?," *Investopedia*, 20-Oct-2022. [Online]. Available: <https://www.investopedia.com/terms/e/ethereum.asp>. [Accessed: 01-Dec-2022].
- [4] "How to get testnet eth using a goerli faucet on ethereum," *Alchemy*. [Online]. Available: <https://www.alchemy.com/overviews/goerli-faucet>. [Accessed: 01-Dec-2022].
- [5] J. Cortez, "Metamask Cryptocurrency Wallet Review 2022," *Investopedia*. [Online]. Available: <https://www.investopedia.com/metamask-cryptocurrency-wallet-review-5235562>. [Accessed: 01-Dec-2022].
- [6] G. Andrew, "NFTs use 'smart' contracts-but what exactly are they?," *The Art Newspaper - International art news and events*, 17-Aug-2022. [Online]. Available: <https://www.theartnewspaper.com/2022/08/17/nfts-use-smart-contractsbut-what-exactly-are-they>. [Accessed: 01-Dec-2022].
- [7] J. Frankenfield, "Gas (ethereum): How gas fees work on the ethereum blockchain," *Investopedia*, 21-Nov-2022. [Online]. Available: <https://www.investopedia.com/terms/g/gas-ethereum.asp>. [Accessed: 01-Dec-2022].
- [8] JakeHaleee, "Top 10 most expensive nfts ever sold," *Dexerto*, 14-Oct-2022. [Online]. Available: <https://www.dexerto.com/tech/top-10-most-expensive-nfts-ever-sold-1670505/>. [Accessed: 22-Nov-2022].
- [9] D. Herbert, "What is react.js? (uses, examples, & more)," *HubSpot Blog*, 27-Jun-2022. [Online]. Available: <https://blog.hubspot.com/website/react-js>. [Accessed: 30-Nov-2022].
- [10] "Why your internet habits are not as clean as you think," BBC Future. [Online]. Available: <https://www.bbc.com/future/article/20200305-why-your-internet-habits-are-not-as-clean-as-you-think>. [Accessed: 30-Nov-2022].
- [11] Cibo, "Calculating the carbon footprint of zoom meetings " Cibo technologies," CIBO Technologies, 04-Nov-2021. [Online]. Available: <https://www.cibotechnologies.com/blog/calculating-the-carbon-footprint-of-zoom-meetings/>. [Accessed: 30-Nov-2022].

- [12] Bybit Learn, “What is the impact of nfts on the environment?,” Bybit Learn, 20-Apr-2022. [Online]. Available: <https://learn.bybit.com/nft/nft-environmental-impact/>. [Accessed: 30-Nov-2022].
- [13] T. Patil, “Proof of stake with casper the friendly finality gadget protocol for fair validation consensus in ethereum,” International Journal of Scientific Research in Computer Science, Engineering and Information Technology , 2018.
- [14] S. Marro and L. Donno, “Green NFTs: A Study on the Environmental Impact of Cryptoart Technologies,” May 2021.
- [15] K. Clinebell, “How green will ethereum 2.0 be?,” Investopedia, 04-Nov-2022. [Online]. Available: <https://www.investopedia.com/how-green-is-ethereum-2-0-6666266>. [Accessed: 30-Nov-2022].
- [16] E. Gobler, “How do people steal nfts? (and why does it seem to be so easy?),” Investor Junkie, 15-Aug-2022. [Online]. Available: <https://investorjunkie.com/nfts/how-do-people-steal-nfts/#:~:text=This%20often%20happens%20in%20emails,they%20share%20their%20private%20key>. [Accessed: 30-Nov-2022].

## Appendices (Smart Contract Code and UI)

### Code

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol";
import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract LockAssets is Ownable, ERC721Holder{

    struct Asset721 {
        uint256 tokenId;
        uint256 withdrawDate;
        IERC721 add;
        string imageLink;
    }

    uint256 public counter = 0;
    mapping(address => Asset721[]) public staked721;

    function stake721(uint256 _tokenId, uint256 _stakeFor, address _add, string
memory _imageLink) public {
        IERC721 nft = IERC721(_add);
        nft.safeTransferFrom(msg.sender, address(this), _tokenId);
        staked721[msg.sender].push(Asset721(_tokenId, block.timestamp + _stakeFor,
_nft, _imageLink));
        counter++;
    }

    function withdraw721(uint256 _tokenId, IERC721 _add) public {
        Asset721 memory deleteFromPool;
        Asset721[] storage assetsStaked721 = staked721[msg.sender];
        for (uint i = 0; i < assetsStaked721.length; i++) {

            //verifies asset your looking for
            if(assetsStaked721[i].tokenId == _tokenId && assetsStaked721[i].add ==
_add) {

                //verify time has been met
            }
        }
    }
}
```

```

        require(block.timestamp > assetsStaked721[i].withdrawDate, "Time Not
Up") ;

        //get address
        IERC721 add = assetsStaked721[i].add;

        //transfer token
        add.safeTransferFrom(address(this), msg.sender,
assetsStaked721[i].tokenId, "0x00");

        //remove from pool
        deleteFromPool = assetsStaked721[i];
        assetsStaked721[i] = assetsStaked721[assetsStaked721.length - 1];
        assetsStaked721[assetsStaked721.length - 1] = deleteFromPool;
        assetsStaked721.pop();
        break;
    }
}

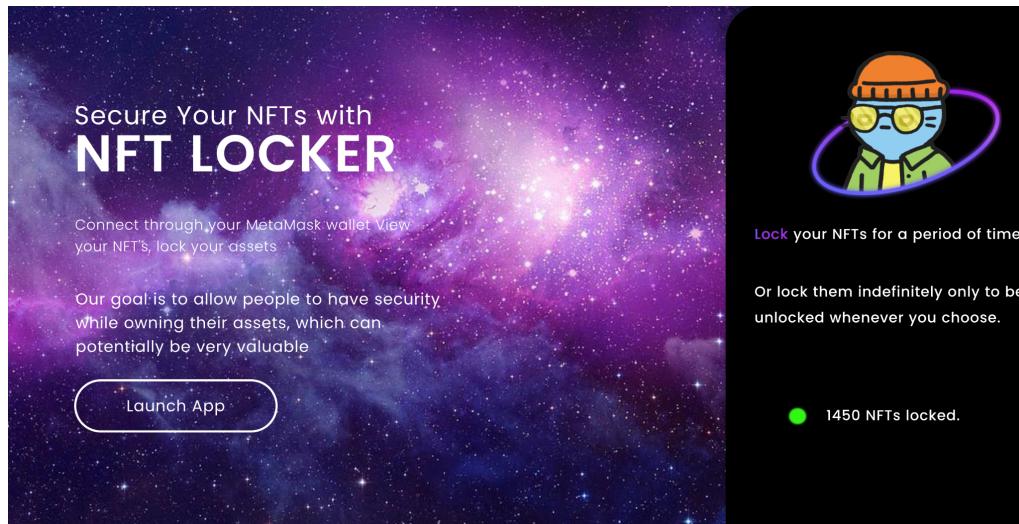
}

function getLockedAssetsForAddress(address addr) public view returns (Asset721[]
memory) {
    // Return the locked assets for the given address
    return staked721[addr];
}
}

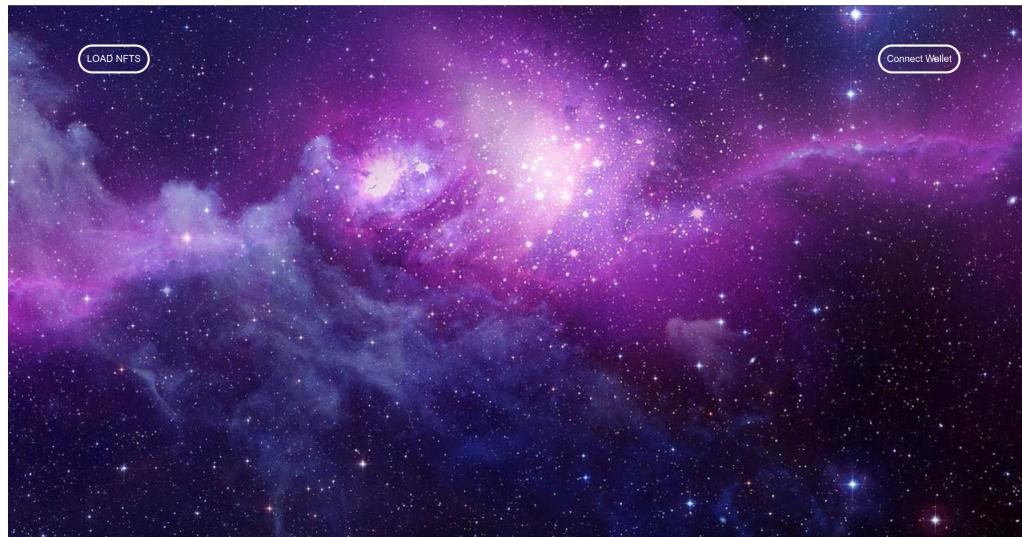
```

## UI

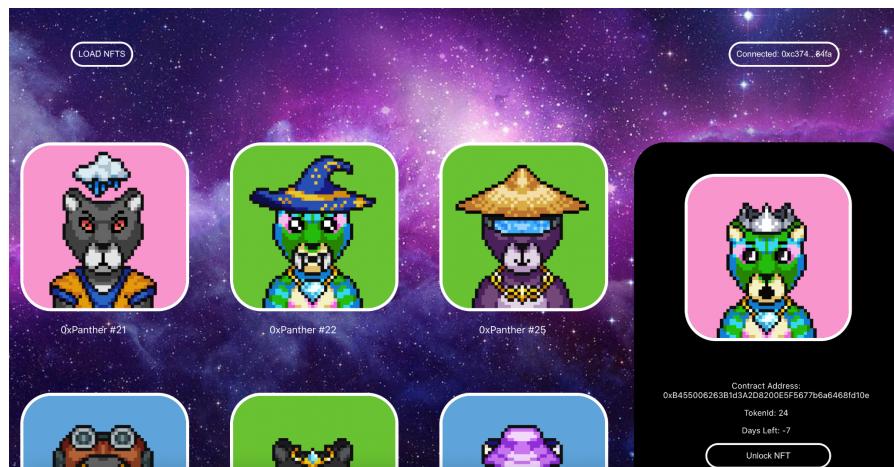
### Homepage



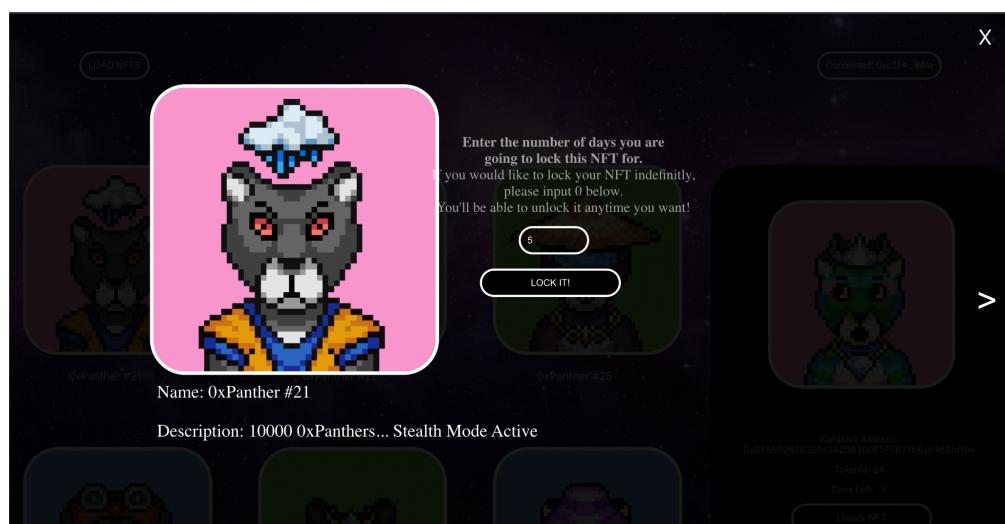
Initial Opened App:



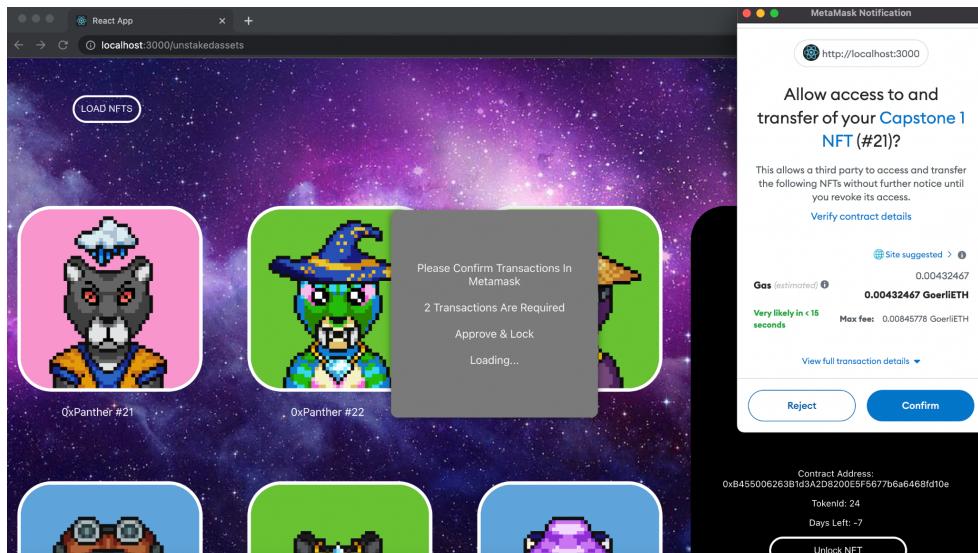
Gallery of locked and unlocked NFTs



Lock a specific NFT



## Lock NFT



## Unlock NFT

