The Chinese University of Hong Kong
Department of Systems Engineering and Engineering Management

(Financial Technology)

FTEC4007 - Project Final Report

Course: FTEC4007 Introduction to Blockchain and Distributed Ledger Technology
Project Title: Equity Crowdfunding Smart Contract (Programming Practice)
Group: 10
Year: 2023 - 2024

Student Name: Cheung Kwong Tai, Nico 1155142517
Lo Yi Chun, Elroy 1155142579
Kin HO, Dino 1155159599

Table of Content

1. Background

1.1 Current Crowdfunding

Crowdfunding platforms have exploded in popularity over the past few years, revolutionizing the way small businesses and individuals obtain funding for their ventures. These platforms provide entrepreneurs with unique opportunities to bring their innovative ideas to life while fostering a sense of community and support (Xu, Liang, Huang, & Davison, 2016). Well-known platforms in this space include Kickstarter and Patreon, each with its own unique approach to crowdfunding.

Kickstarter allows businesses and individuals to raise funds for their projects and provide benefits or end products based on investors' investment levels. Patreon, on the other hand, uses a subscription model where creators reward customers with exclusive content such as images, videos, or articles (QuickBooks Canada Team, 2021). Hence, the investment return of crowdfunding can be in many forms.

These platforms not only provide the means to gain financial support, but also foster vibrant communities of supporters who are deeply invested in the success of these projects. Through crowdfunding, individuals and businesses can escape traditional funding constraints, embrace innovation, and build meaningful connections with their audiences.

1.2 Challenges of Current Crowdfunding

Through extensive research, we identified three key challenges associated with existing crowdfunding platforms. First, a major concern is liquidity risk. This is due to the lack of a secondary market for investors to redeem the rewards they receive into cash. For example, if an investor receives unlisted stock or company merchandise as an award, the lack of a market to sell or trade these assets may limit their liquidity.

Secondly, another important issue is disclosure risk. Many crowdfunding platforms struggle to provide adequate information about where funds are flowing, creating uncertainty for potential investors. If there is no transparency into how funds raised are used and distributed, investors may be hesitant to contribute their financial resources (Kim, Park, Pan, Zhang, & Zhang, 2022).

Finally, a noteworthy issue is the trust factor associated with current crowdfunding platforms. These platforms often lack official penalties if project owners fail to live up to their promises. A lack of accountability can lead to a loss of investor confidence (Vromen, Halpin, & Vaughan, 2022). For example, creators on platforms like Patreon may not face any consequences for failing to deliver on their promises, such as not releasing videos every week as originally promised.

2. Smart Contract Solution

In response to the identified issues, we implemented a novel approach to incorporating equity crowdfunding into smart contracts. This innovative solution solves liquidity, disclosure and trust issues prevalent in traditional crowdfunding platforms.

First, we reward investors with equity non-fungible tokens (NFTs), giving them tokenized equity that can be traded on popular NFT marketplaces such as OpenSea, OKX NFT Marketplace and other secondary markets (PR Newswire Association LLC, 2021). Unlike startup equity trading, these existing NFT marketplaces offer a certain degree of liquidity for trading equity NFTs. It provides an option to investors to convert received NFT rewards into cash or other digital assets, enhancing the overall liquidity of the crowdfunding ecosystem.

Second, to ensure transparency, we leverage the inherent nature of blockchain technology to record all project and transaction details on a public decentralized ledger (Hua, X., 2019). In order to mitigate the risk of fraud, the smart contract requires crowdfunding project creators to provide comprehensive supporting information and evidence, which is publicly accessible for viewing. Moreover, this transparent ledger allows investors to track the flow of funds, providing them with comprehensive transaction information on how their investments are used. By eliminating information asymmetries, we enhance investor confidence and mitigate the disclosure risks associated with traditional crowdfunding platforms.

Finally, we implemented automatic reward distribution and refund mechanisms within the smart contract framework to solve the trust issue. Once the funding goal is reached, the smart contract automatically sends NFT rewards to the respective investor's digital wallet address. In the event of a funding campaign failure, the smart contract ensures that the funds are promptly returned to all the investors. This eliminates the possibility of project owners failing to meet their commitments, ensuring timely and seamless reward distribution. By implementing this automated process, we enhance the trustworthiness and eliminate the need for manual intervention, providing a reliable and efficient way to deliver on our promises with no third-party risk.

To enable these advancements, we developed two smart contracts (Appendix 7.3) —one designed specifically for crowdfunding operations and the other specifically for the management of NFT assets. The seamless interaction between these two smart contracts enables a streamlined and automated crowdfunding process.

3. Execution Logic

3.1 Control Flow

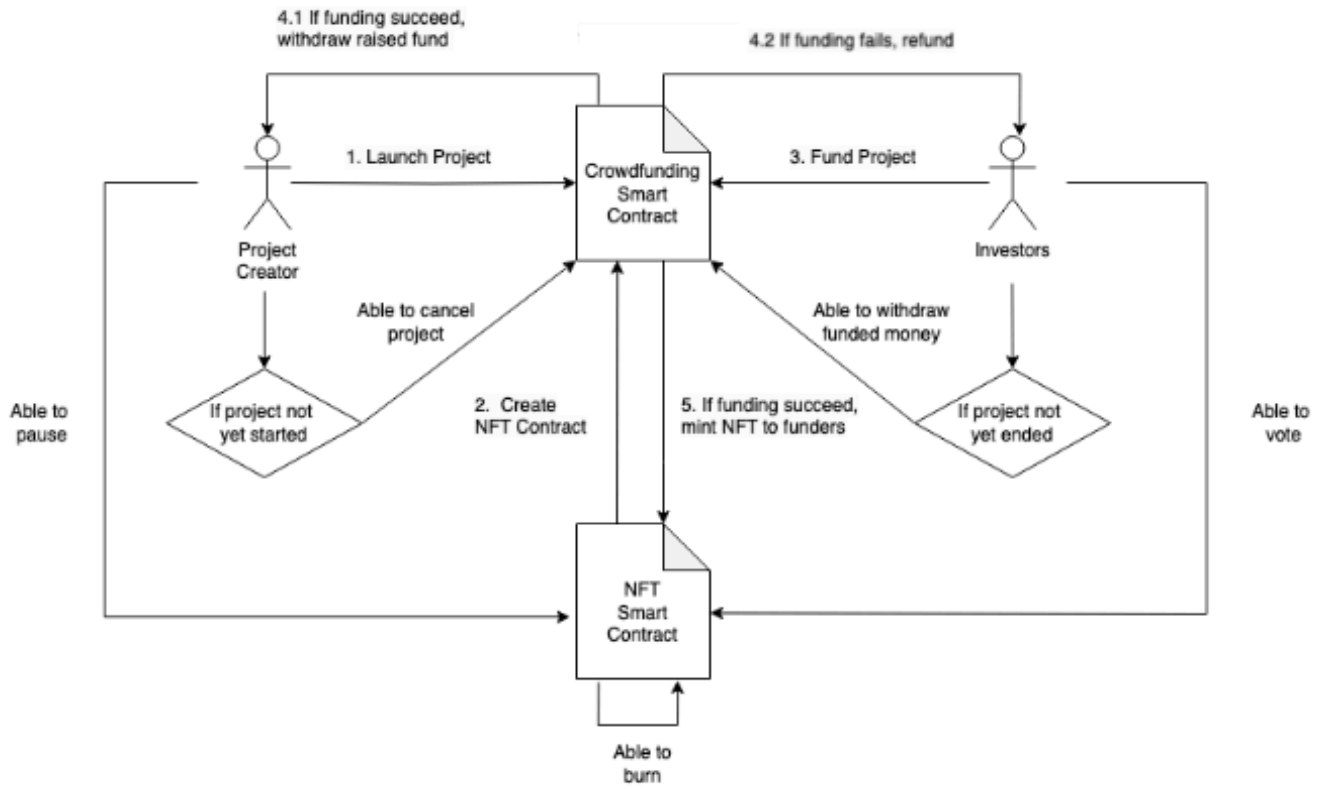

Figure 1: Control Flow Graph

The application caters to two user groups: project creators and investors. It is designed to handle concurrent execution of multiple projects, without any restrictions on the number of projects that can be created by the creators or funded by the investors.

Figure 1 illustrates between both user groups and the smart contracts in a crowdfunding project. The process begins with the project creator launching a new project in the crowdfunding contract. When a new project is created, a corresponding NFT smart contract is generated associated with the project. After that, investors will be able to see all the project information available on the public blockchain and fund the projects they prefer. Notably, the project creator and investors need to pay gas fees whenever they launch and fund the project.

For each funding transaction, the crowdfunding contract verifies whether the funding amount exceeds the predefined target funding amount. If the funding is successful, the contract proceeds to distribute equity NFTs to the funders and activate the withdrawal function for the project owner. For every wei invested by the funder, an NFT is minted to them.

A NFT will be minted to the funder for each wei the funder invested. However, if the funding fails, the contract will refund the investment to the funders. After that, all the information of the NFT minted, funding transactions and withdraw transactions will be also stored on the public blockchain.

Apart from the standard execution flow mentioned, several additional features have been implemented in both contracts. The crowdfunding contract allows project creators to cancel their project before the funding campaign begins if they decide not to proceed. Similarly, investors have the option to withdraw their investment as long as the project has not yet ended or achieved its funding target. As for the NFT contract, project creators can pause and unpause the NFT contract anytime they desire. It is useful to temporarily disable transferring NFTs during maintenance or in response to security concerns.

Moreover, NFT holders can initiate voting procedures to engage in the decision-making process of the NFT ecosystem. Each NFT represents a unit of voting right, allowing the NFT community to collaboratively decide on various matters that can be implemented on the contracts. It includes decisions related to the dividend distribution mechanism, token burning system, modification of the funding targets and so on. However, the enforceability of such voting decisions in real-life matters is limited since it requires interfacing with the corporate action in the physical world.

3.2 Data Flow

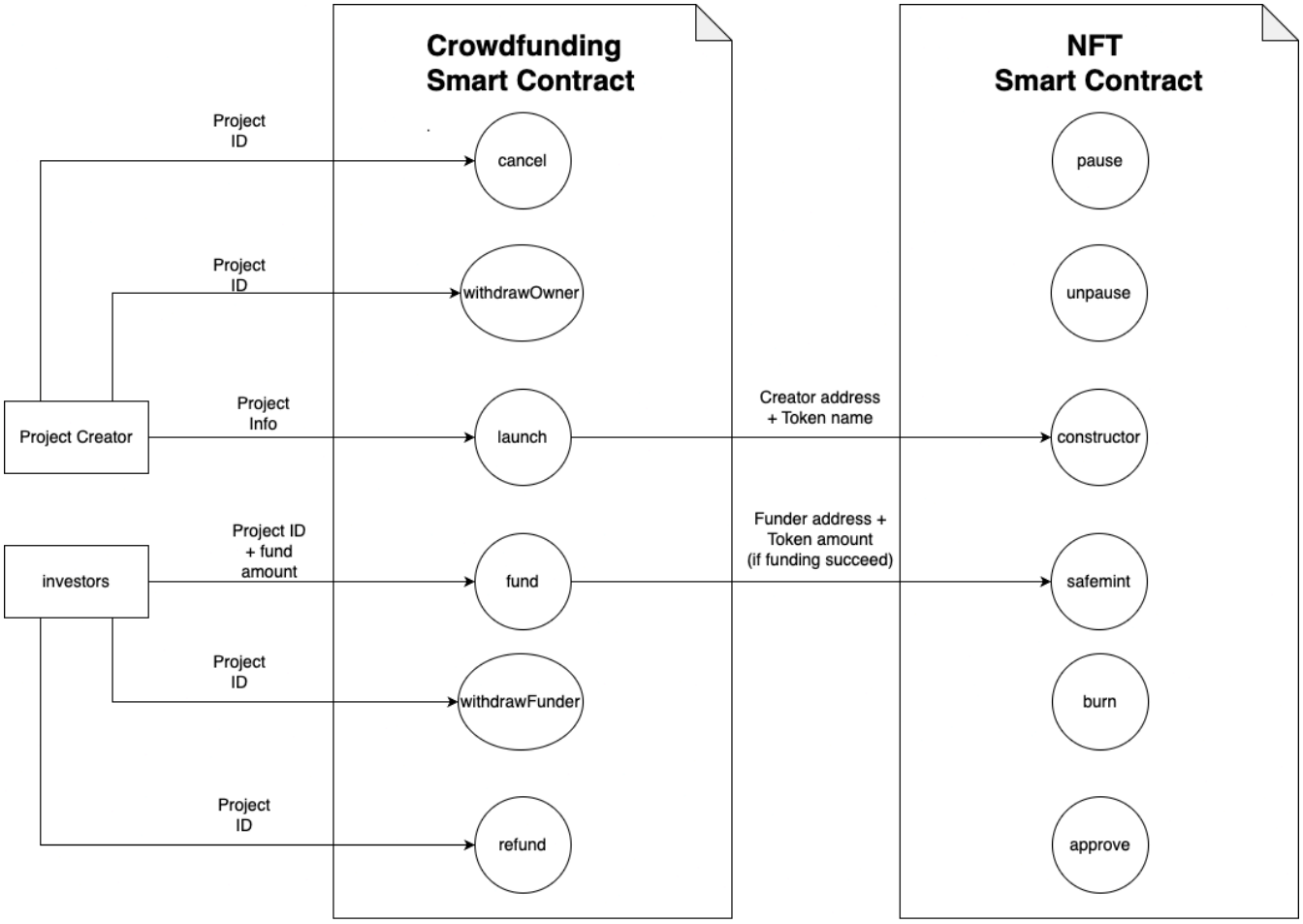

Figure 2: Data Flow Diagram

In Figure 2, the data flows between users and the functions in two smart contracts is illustrated. To launch a crowdfunding project, the project creator must provide the necessary project details in the *launch* function. Whenever project creators and investors interact with the crowdfunding smart contract, they are required to input project ID to execute the desired function. The figure also shows how *launch* and *fund* functions in the Crowdfunding smart contract trigger the functions in the NFT smart contract.

3.3 Execution Demonstration

The mentioned execution logic can be tested on Sepolia Ethereum testnet, one of the popular testing environments for smart contracts. We chose Alchemy as the hosting platform and Etherscan to monitor deployed contracts. The step-by-step execution procedure is shown below:

1. Set up .env file to include the crypto-wallet private key, Alchemy API URL and Etherscan API Key

2. Deploy the Crowdfunding smart contract to Sepolia testnet (Appendix 7.1) by running deploy.js with command:

npx hardhat run scripts/deploy.js --network sepolia

3. Interact with Crowdfunding smart contract by running interact.js with command:

npx hardhat run scripts/interact.js --network sepolia

The result is shown in Figure 3. The script simulates the interaction between users and the smart contracts. First, the script launched a project with a funding target of 10 wei. This triggered the automatic creation of a NFT Contract (Appendix 7.2). Subsequently, the program funded the created project with 10 wei. As the funding target is reached, 10 NFTs were minted to the funder as shown in Figure 4.

```
(django-7ifw2e5Z) ➜  ftec4007 git:(main) ✗ npx hardhat run scripts/interact.js --network sepolia
Start launching new project: project-1713342315022
Project Created - ID: 5, name: project-1713342315022, owner: 0x8A1D0a624802E1d40ef6f348Aaf9766cf58e5f93, targ
etFund: 10, endDate: Fri, 17 May 2024 08:25:37 GMT
NFT Contract created, address: 0x329F56b127eAC6A842EfC065EaA933f74A109Af9
Start funding 10 to Project 5
Fund Success
Project 5: Funding met target amount, raised amount: 10 wei
Minted NFT to funders
```

Figure 3: Result of interact.js

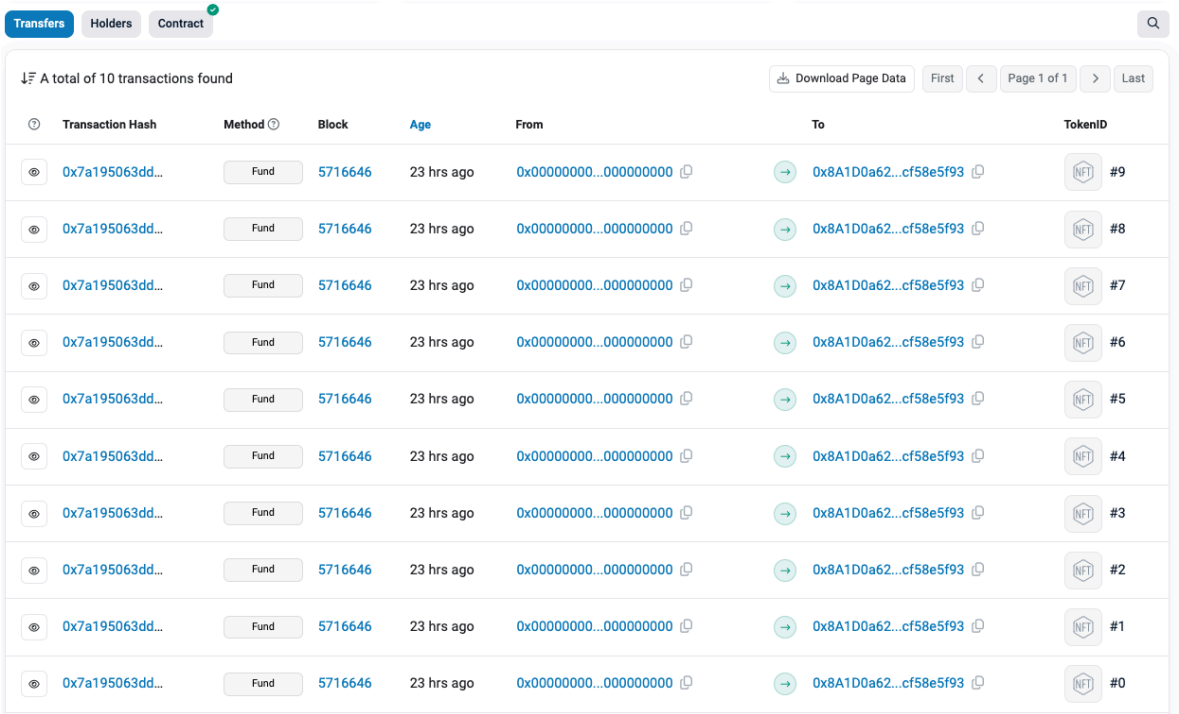

Figure 4: NFTs mint records

4. Crowdfunding Smart Contract

4.1 Contract Structure

The Crowdfunding smart contract is deployed under the MIT license and has been developing using Solidity version 0.8.20. This contract is specifically designed to replicate traditional crowdfunding platforms on blockchain, offering enhanced security, immutable record keeping, and transparent transactions. The crowdfunding contract includes several key features tailored to manage various aspects of crowdfunding campaigns efficiently.

This contract introduces an integration with the NFT contract, allowing the crowdfunding contract to trigger NFT minting based on specific interaction and milestones within the crowdfunding campaign. For example, NFTs can be minted as rewards for contributors once a project successfully meets its funding goal.

Key variables in the contract include **'projectCount'**, which maintains a count of all projects created. Each project within the contract is represented as a struct containing essential details such as project ID, its owner address, funding target, and the starting and ending timestamp of the crowdfunding campaign.

The contract also employs variables **'raisedFund'** to track the current amount of money raised and denote the funding goal. These variables help determine the success of a campaign and trigger further actions, such as fund withdrawals and NFT minting.

Moreover, the contract records and updates each contribution made by the backers by utilizing mappings **'funders'** and the array **'funderAddress'**. Ensuring accurate tracking of fund distribution and eligibility for receiving rewards like NFTs.

4.2 Crowdfunding Contract Events

To ensure all participants are kept up-to-date with the latest status of the projects and maintain the transparency of the platform, the Crowdfunding contract employs 7 blockchain events. Below is a table detailing each event, their triggers, and their purposes:

Event Name	Description	Trigger Condition	Purpose
Launch	Emitted when a new project is successfully created.	When the launch function is successfully executed.	providing essential details of a new project such as the project ID, owner, funding targets, and associated NFT.

Fund	Occurs each time a contribution is made to a project.	Upon a successful call to the fund function.	Signals a successful funding operation to the project
SuccessFund	Announced when a project reaches or exceeds its funding goal.	When funding meets or surpasses the target fund during or after a contribution.	Indicates that the project has been financially successful and can proceed to the next phases
OwnerWithdraw	Emitted when the project owner withdraws the raised funds post-success.	After the project owner successfully calls the withdrawOwner function and transfers out the funds.	Confirms that the funds have been transferred to the project owner, marking a key financial transition in the project's lifecycle.
Cancel	Logged when a project is canceled before it starts funding.	Triggered by the cancel function	Provides a clear indication that the project has been officially canceled
Refund	Emitted for each refund processed to a contributor when a project fails to meet its funding goals.	When a contributor successfully calls the refund function after a failed project.	Assures contributors that their funds have been returned, maintaining trust and accountability.
FunderWithdraw	Notified when a funder retrieves their contribution before the project ends.	Upon a successful execution of the withdrawFunder function.	Informs the network that a backer has withdrawn their funds, detailing the financial adjustments within the project's fund pool.

Table 1: Events of Crowdfunding Contract

4.3 Crowdfunding Contract Read-Only Functions

This part explores the read features of the Crowdfunding contract. These functions act as a gateway to the contract providing users with an unchangeable repository of project related data without impacting the blockchains status. It enables users to view contribution information and understand the funding environment.

4.3.1 getMyFund

The getMyFund function is an essential tool for enhancing user engagement and maintaining transparency within the crowdfunding platform. By calling this function, a user can find out exactly how much they have contributed to a specific project. This functionality not only allows participants to keep track of their financial commitments but also aids them in managing their investment portfolios efficiently.

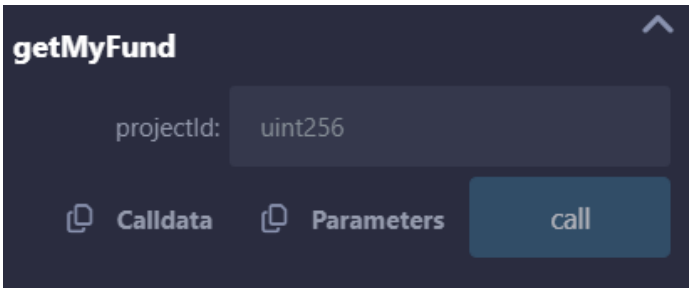

Figure 5: getMyFund Function Interface on Remix IDE

4.3.2 getRaisedFund

The getRaisedFund function plays a critical role in the crowdfunding ecosystem by providing real-time updates on the total amount of funds collected for a particular project. This function is pivotal for all stakeholders, including project creators, investors, and platform administrators, as it allows them to gauge the current success of fundraising efforts and make informed decisions based on collected financial insights.

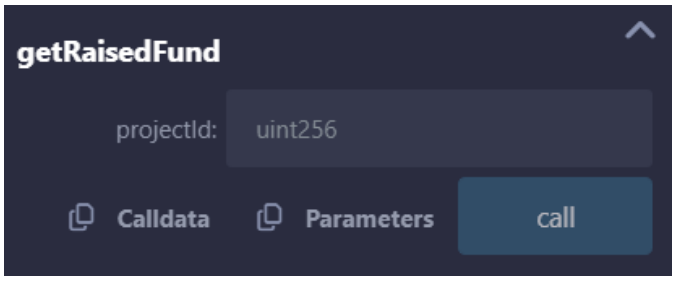

Figure 6: getRaisedFund Function Interface on Remix IDE

4.3.3 getDeadline

‘getDeadline’ is a straightforward yet crucial function within the Crowdfunding contract that retrieves the funding deadline for any given project. This function provides essential information about the time remaining until a project stops accepting funds, which is critical for both backers and project owners.

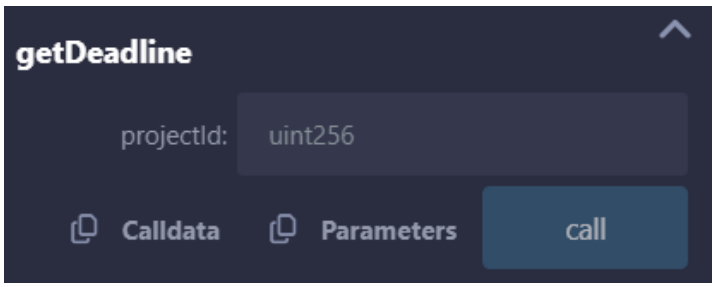

Figure 7: getDeadline Function Interface on Remix IDE

4.3.4 getFunders

The getFunders function is vital for maintaining high levels of transparency within the crowdfunding platform. It provides a detailed list of all individuals who have funded a particular project, along with the specific amounts each has contributed. This visibility is crucial as it allows all participants to verify who is supporting a project and to what extent.

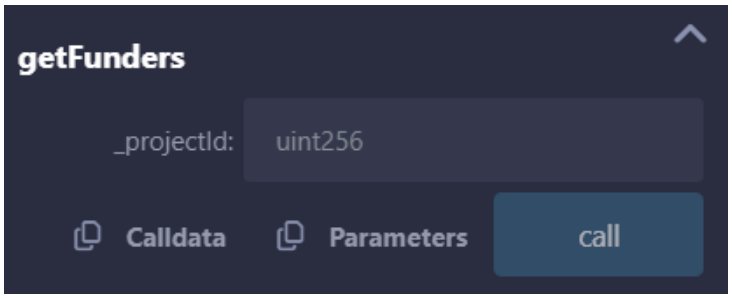

Figure 8: getFunders Function Interface on Remix IDE

4.3.5 isSuccess

Finally, the isSuccess function serves as a quick and efficient way to determine whether a project has met its funding objectives. By returning a simple true or false, this function tells users whether a project has successfully reached or exceeded its declared funding target, which is a key indicator of the project's viability and potential for realization.

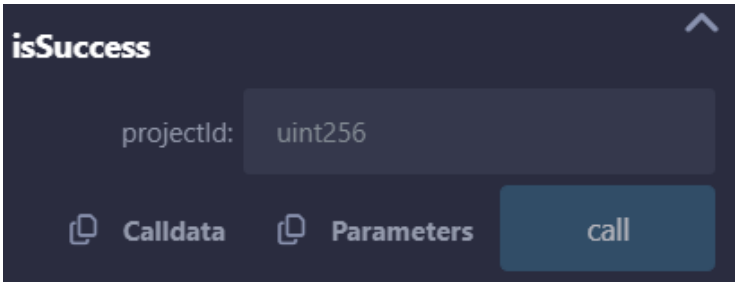

Figure 9: isSuccess Function Interface on Remix IDE

4.4 Crowdfunding Contract Update Functions

In the sections we'll delve into the features of the Crowdfunding smart contract. These functions serve as the engine that drives interaction and transactions. They create an adaptable platform where creators can gather backing supporters, can fund concepts and the entire crowdfunding process is managed efficiently and effectively.

4.4.1 launch

The launch function is pivotal as it enables users to initiate new crowdfunding projects on the platform. By calling this function, users can specify fundamental attributes of their project, including the project's name, description, the total funds needed (targetFund), and the project's duration. This function sets up the basic framework upon which the project operates, defining both its scope and operational timeline.

Key to this function's operation are several safeguards: it requires that all monetary values, including the target fund, be strictly positive, ensuring that the project has a legitimate financial goal. Additionally, the project's timeline must be logically consistent — the end time (endAt) cannot precede the start time (startAt), and the entire project duration must be confined within a maximum limit of two years from the current date. These conditions help maintain the platform's integrity and prevent misuse through unrealistic project setups.

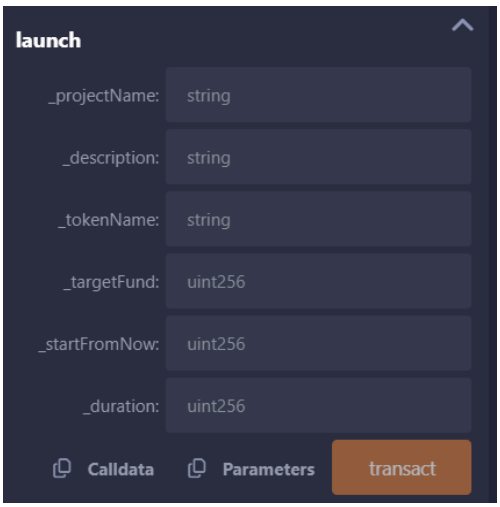

Figure 10: Launch Function Interface on Remix IDE

4.4.2 fund

The fund function is central to the crowdfunding contract's operation as it allows users to actively contribute funds to projects they wish to support. This function is essential not only for gathering the necessary financial resources needed by the project but also for enabling community participation and investment in potentially successful initiatives.

For contributions to be processed, the project must already exist within the system, and it must be actively seeking funds (i.e., within its predefined funding period). These requirements ensure that funds are only collected for valid, officially recognized projects that are in their fundraising phase.

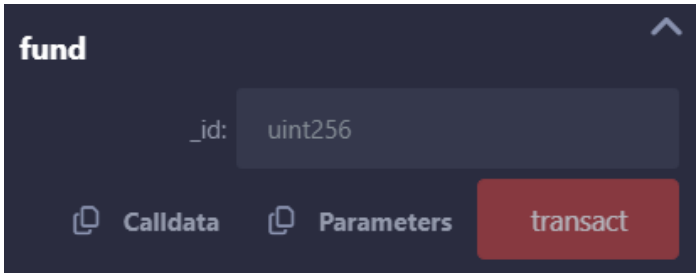

Figure 11: Fund Function Interface on Remix IDE

4.4.3 cancel

The cancel function provides a crucial control mechanism for project creators, allowing them to terminate their projects under specific conditions before any funding begins. This function is designed to give creators maximum flexibility and control over their projects, enabling them to step back without financial penalties if external circumstances or reconsiderations prevent the project's initiation.

To execute a cancellation, the project must not have received any funds yet (ensuring no financial obligations to backers), and it must not have officially started. Moreover, only the project's creator (owner) has the authority to cancel the project, which secures the function against potential misuse by unauthorized individuals.

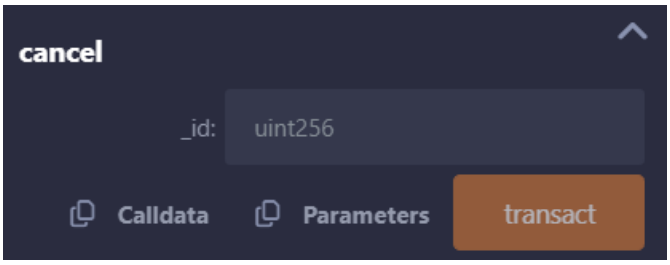

Figure 12: cancel Function Interface on Remix IDE

4.4.4 withdrawFunder

‘withdrawFunder’ allows backers to retract their contributions under certain conditions, providing them with a safety net if the project's prospects dim. This function is critical in maintaining trust and confidence among the platform's users, as it ensures that backers can recover their investments if the project fails to start on time or meet its funding objectives.

For a backer to withdraw funds, the project must either have not met its funding target or still be within its active funding period. Furthermore, only contributors who have actually funded the project can make withdrawal requests, which prevents fraudulent claims and ensures that only genuine backers can retrieve their contributions.

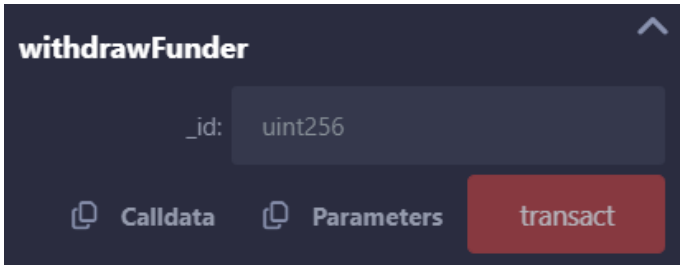

Figure 13: withdrawFunder Function Interface on Remix IDE

4.4.5 withdrawOwner

The withdrawOwner function allows project owners to access the funds raised upon successful completion of the crowdfunding campaign. This function is gated by stringent conditions to ensure that only legitimate withdrawals following successful campaigns are processed.

Specifically, the project must have achieved or exceeded its funding target, the funding period must have ended, and there must be no prior withdrawals by the owner. These criteria help safeguard the collected funds, ensuring they are used appropriately according to the crowdfunding goals and only disbursed once these goals are conclusively met.

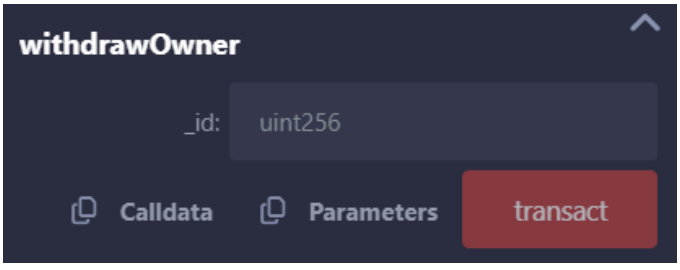

Figure 14: withdrawOwner Function Interface on Remix IDE

4.4.6 refund

The refund function is designed to uphold the crowdfunding platform's integrity by ensuring that backers are not financially disadvantaged if a project fails to secure enough support. By

enabling the return of contributions under specific conditions, this function plays a vital role in maintaining fair dealings within the platform.

To qualify for a refund, a project must not have reached its funding goal by the stipulated end date, ensuring that refunds are only issued for unsuccessful ventures. Additionally, there must be a record of the contribution from the user seeking a refund, which helps prevent fraudulent refund claims and ensures that only genuine backers receive their money back.

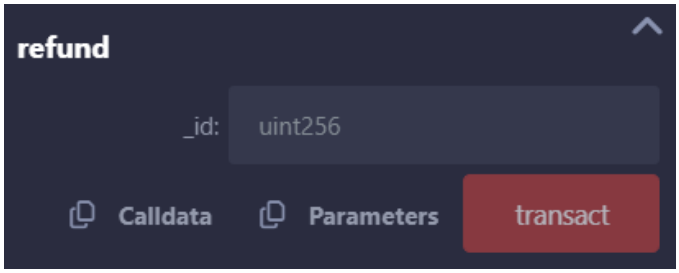

Figure 15: refund Function Interface on Remix IDE

5. NFT Contract

5.1 Contract Structure

The NFT smart contract is released under the MIT license and is designed to be compiled using Solidity version 0.8.20 or a higher compatible version.

Additionally, the contract is designed to be used in conjunction with the OpenZeppelin contract library (specifically version 5.0.0 or any subsequent version). OpenZeppelin Contracts is a trusted library that provides tested and secure implementations of common smart contract functions.

We utilize the ERC721 standard in smart contracts to implement functionally enhanced NFT contracts. This implementation takes advantage of various features and extensions provided by the OpenZeppelin contract library. These additional features include enumeration, suspension, ownership control, burning, cryptographic signature processing, and voting capabilities.

In the context of smart contract control, three key variables play a vital role. The first variable, "maxmint," serves as a means to control the maximum number of NFTs that can be minted within a given project. This variable allows project creators to set an upper limit, ensuring scarcity and value preservation within the NFT ecosystem.

The second variable, "currentMint," represents the current count of NFTs that have been minted and currently exist within the project. By tracking this variable, anyone can monitor the progress and growth of the NFT collection, enabling them to make informed decisions regarding future minting activities or adjustments to the project's parameters.

Lastly, the "tokenId" variable holds a unique identification number assigned to each NFT within the project. This identifier serves as a crucial reference for distinguishing and managing individual NFTs, facilitating their transfer, ownership verification, and interaction within the smart contract ecosystem.

5.2 NFT Contract Read-only Functions

5.2.1 balanceOf

This function provides participants with information on the token balance associated with a specific owner account.

By calling the function, users can get the exact number of tokens held by a specific account in the contract. This feature enables participants to monitor their token holdings and track balance changes.

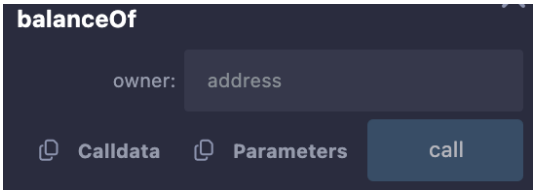

Figure 16: balanceOf Function Interface on Remix IDE

5.2.2 ownerOf

This function is designed to provide participants with information about the current owner of a given token.

It should be noted that this function has a requirement: tokenId must exist in the contract. This requirement ensures that the function runs reliably and avoids returning incorrect or inconsistent data.

By calling this function, users can obtain the legal owner of a token, allowing them to verify ownership, track token transfers, and conduct secure transactions.

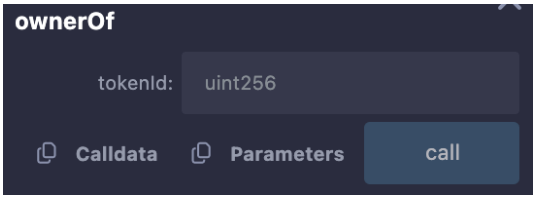

Figure 17: ownerOf Function Interface on Remix IDE

5.2.3 getTokenIdsByOwner

This feature allows participants to retrieve the token identifier associated with a given address.

By calling the function, the user can obtain the tokenId corresponding to the address within the contract. This feature provides a convenient way to verify token ownership, track specific tokens, and perform targeted actions based on tokenId.

The ability to retrieve the tokenId owned by a specific address enhances transparency and facilitates efficient token management. Participants can identify and interact with the tokens they own, enabling transactions and participation within the token ecosystem.

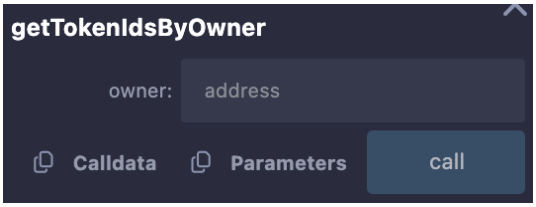

Figure 18: getTokenIdsByOwner Function Interface on Remix IDE

5.2.4 maxmint

This function provides participants with essential information regarding the upper limit or cap on the token supply.

By calling this function, users can obtain the maximum number of tokens that can be created or minted within the contract. This feature ensures transparency and clarity regarding the token issuance process.

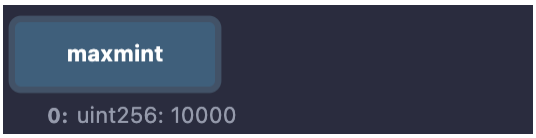

Figure 19: maxmint Function Interface on Remix IDE

5.3.5 getApproved

This function retrieves the approved address for a specific token ID. If the address has been approved, the function returns the approved address; otherwise, it returns a zero address to indicate that no address has been set for approval. Note that if the provided token ID does not exist in the contract, the function will revert.

By incorporating this functionality, the contract ensures transparency and accountability in tracking approved addresses associated with token IDs. This feature allows participants to easily verify and obtain information about the authorization address for token transfers.

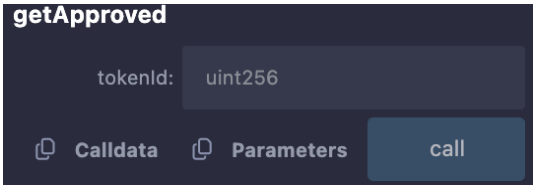

Figure 20: getApproved Function Interface on Remix IDE

5.2.6 isApprovalForAll

This function verifies the operator's approval status for token transfers made by a specific owner.

By calling the function, participants can confirm whether the specified operator is authorized by the owner to make token transfers on their behalf. This feature enhances transparency and allows users to make informed decisions regarding delegation of token transfer permissions.

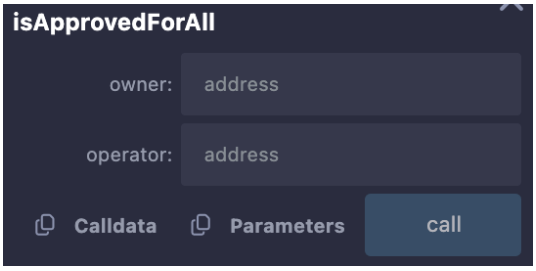

Figure 21: isApprovedForAll Function Interface on Remix IDE

5.3.7 paused

This function provides participants with a way to check the current status of the contract and determine whether its operations have been temporarily halted.

By calling this function, the user can obtain a true or false value indicating the contract's suspended status. This feature increases participants' transparency and awareness of the contract's availability and functionality.

Figure 22: paused Function Interface on Remix IDE

5.3.8 getVotes

By calling the function, participants can obtain real-time information about the voting rights held by specific accounts in the contract.

This feature is critical for transparency and accountability within voting systems. Let users clearly understand the influence and weight of their account in the decision-making process. By knowing how many votes they currently have, participants can make informed choices, participate in discussions, and actively contribute to the governance process.

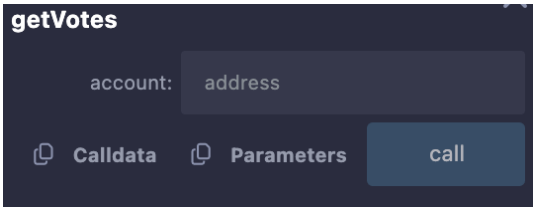

Figure 23: getVotes Function Interface on Remix IDE

5.3.9 totalSupply

This function provides participants with information about the total number of votes in circulation within the contract.

The ability to retrieve the current vote total increases transparency and accountability within the voting system. It allows participants to measure the size and impact of their votes, enabling them to make informed decisions based on the broader voting context.

Figure 24: totalSupply Function Interface on Remix IDE

5.3.10 tokenURI

By calling this function and providing the tokenId as input, users can obtain the URI that corresponds to the token within the contract.

The URI typically points to a location where additional metadata or media related to the token can be accessed, such as descriptions, images, or other relevant information.

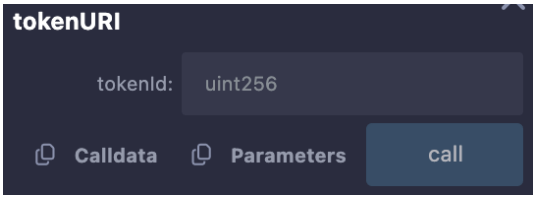

Figure 25: tokenURI Function Interface on Remix IDE

5.3 NFT Contract Update Functions

5.3.1 safeMint

This function empowers the project owner with the ability to mint a designated number of new NFTs and allocate them to a specific target address. However, there is an essential prerequisite for the successful execution of this function: the minted amount must not exceed the predefined "maxmint" value. If the minted quantity surpasses this maximum limit, the function will fail to execute.

During the minting process, the "safemint" function assigns a unique identification to each newly minted NFT. This ID serves as a distinct and non-duplicative identifier for each NFT, enabling precise tracking, ownership verification, and interaction with individual tokens within the smart contract ecosystem.

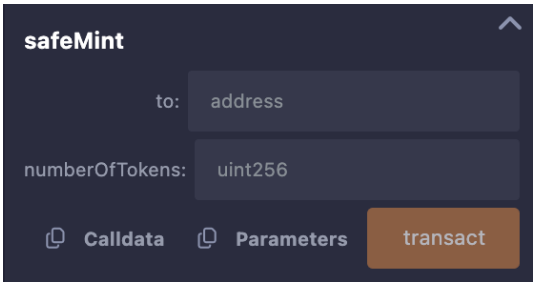

Figure 26: safeMint Function Interface on Remix IDE

5.3.2 safeTransferFrom

This function empowers users to safely transfer NFTs between addresses while specifying a unique "tokenId" for the particular token. This function ensures secure and controlled token transfers within the ecosystem. However, there is an important requirement for the caller of this function: they must either be the rightful owner of the NFT or possess explicit approval from the NFT owner.

By enforcing this ownership verification process, the function guarantees that only authorized individuals can initiate the transfer of NFTs.

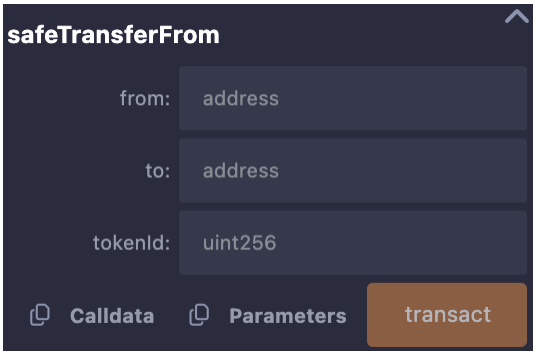

Figure 27: safeTranferFrom Function Interface on Remix IDE

5.3.3 approve

This function allows token owners to grant specified addresses permission to initiate transfers on their behalf. The approval process involves specifying the token ID and the address to which the approval is to be granted.

It's worth noting that setting an approved address to the zero address will actually revoke any previous approval. It is important to emphasize that only one address per token can be approved at any given time.

To ensure the security and integrity of the approval process, this function can only be called by the token owner himself or an approved operator. By limiting calling rights to authorized individuals, the contract ensures that only trusted parties have the ability to manage and transfer tokens on behalf of the owner.

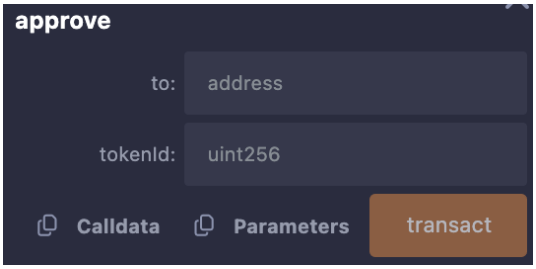

Figure 28: approve Function Interface on Remix IDE

5.3.4 setApprovalForAll

This feature allows setting or unsetting of approvals for specific operators. This approval gives the designated operator the authority to transfer all tokens owned by the sender of the transaction.

Once approved, the operator can initiate transfers of all tokens on behalf of the sender. This feature is particularly useful in scenarios where the sender wishes to delegate transfer rights to a trusted third party or smart contract. Conversely, when set approval is revoked, the operator's ability to transfer tokens on behalf of the sender is revoked.

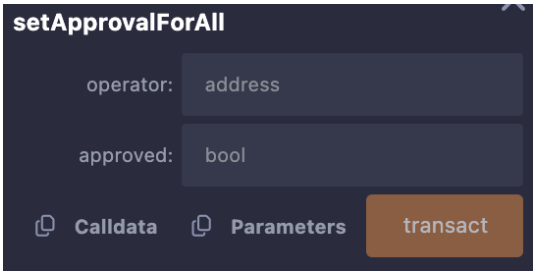

Figure 29: setApprovalForAll Function Interface on Remix IDE

5.3.5 burn

The purpose of this function is to permanently remove a specific token from circulation. If there are no tokens specified for destruction in the contract, the function will be restored, ensuring the validity of the operation.

The function facilitates inflation control and maintains the integrity of the token economy. For example, The project party can buy the NFT from users according to a desired price. By burning the NFT to reduce its supply and facilitate token inflation.

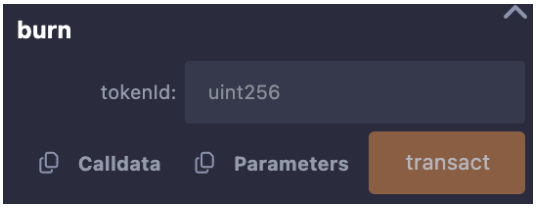

Figure 30: burn Function Interface on Remix IDE

5.3.6 pause/unpause

To address critical security issues, our NFT contracts are equipped with "pause" and "unpause" functionality. These functions provide the ability to temporarily stop the execution of a contract, serving as an important safeguard in situations such as bug fixes or potential attacks by malicious hackers.

The "pause" function is designed to suspend the functionality of the contract, effectively freezing all operations and preventing any further operations. This feature allows developers and administrators to quickly resolve any identified bugs or vulnerabilities, ensuring the stability and security of the contract.

Instead, the "Unpause" function serves as a counterpart to the "Pause" function. When called, it restarts the contract, thus resuming normal operation. This feature is critical to restoring the functionality of the contract once the necessary bug fixes or security enhancements are implemented.

Figure 31: pause and unpause Function Interfaces on Remix IDE

5.3.7 delegate

This function allows token holders to transfer their voting rights to another account of their choice.

By calling this function, participants can delegate their voting rights to the delegate, granting them the authority to vote on their behalf. This delegation mechanism promotes flexibility and participation in the governance process, as token holders can delegate voting rights to individuals or entities they deem to be knowledgeable or in their interest. The ability to delegate voting enhances the democratic nature of the governance system and promotes token holder representation.

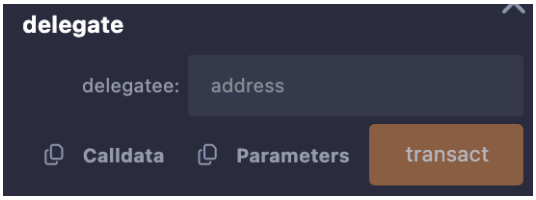

Figure 32: delegate Function Interface on Remix IDE

6. Possible Concerns

6.1 Regulatory Compliance

As our crowdfunding smart contract involves the offering of equity-related products, the business usually needs to be licensed in many jurisdictions. For example, according to the Securities and Futures Ordinance (SFO) in Hong Kong, equity crowdfunding businesses need to be licensed and regulated by the Securities and Future Commission (SFC). Failing to obtain the required license may result in regulatory actions, including lawsuits and fines, similar to the SEC's lawsuit against Telegram for its unregistered equity ICO (Baydakova, 2021).

6.2 Investor Protection

In crowdfunding equity smart contracts, the absence of intermediaries introduces risk of fraud and the participation of unqualified investors, resulting in investor loss. Robust investor protection measures, including KYC and accreditation, are crucial. On the other hand, our smart contracts try to collect and verify the project information and provide a higher degree of transparency to minimize the risk of fraud.

7. Reference

Baydakova, A. (2021, September 14). *Telegram is nearly done paying back ton investors, Eyes IPO Next*. CoinDesk Latest Headlines RSS. https://www.coindesk.com/markets/2021/04/12/telegram-is-nearly-done-paying-back-ton-investors-eyes-ipo-next/

PR Newswire Association LLC. (2021). NFT Glee Sells Out Bitcoin Bob NFT; Bitcoin Trading Algorithm Inside An NFT Now Has Waiting List And Robust Secondary Market. New York.

Hua, X. (2019). Industrial Management and Data Systems: Financial Technologies: Artificial Intelligence, Blockchain, and Crowdfunding. Bradford, West Yorkshire, England: Emerald Publishing Limited.

Keongtae Kim, Jooyoung Park, Yang Pan, Kunpeng Zhang, Xiaoquan (Michael) Zhang (2022) Risk Disclosure in Crowdfunding. Information Systems Research 33(3):1023-1041. https://doi.org/10.1287/isre.2021.1096

QuickBooks Canada Team. (2021, June 10). Which Crowdfunding Platform is Best for Your Small Business? [Blog post]. Retrieved from https://quickbooks.intuit.com/ca/resources/funding/which-crowdfunding-platform-is-best-for-your-small-business/?cid=ppc_ROW_SMB_QBO_HK_G_NB_Search_DSA&gad_source=1&gclid=CjwKCAjw5v2wBhBrEiwAXDDoJZ23Cq7_iXsuvcC3YOC53eYkos7V146_tpbCPbgQ9d630NDffF4iYBoCKAgQAvD_BwE&gclsrc=aw.ds

Vromen, A., Halpin, D., & Vaughan, M. (2022). Crowdsourced Politics: The Rise of Online Petitions and Micro-Donations. Springer Nature Singapore Pte. Ltd. https://doi.org/10.1007/978-981-19-4357-7

Xu, B., Liang, H., Huang, J., & Davison, R. M. (2016). Configurational paths to sponsor satisfaction in crowdfunding. Journal of Business Research, 69(2), 915-927.

8. Appendix

7.1 Deployed Crowdfunding Smart Contract on Sepolia Ethereum testnet
https://sepolia.etherscan.io/address/0xE02E90F6F0dB2fb72BCaa90cc2Fe614031008A4C

7.2 Deployed NFT Smart Contract on Sepolia Ethereum testnet
https://sepolia.etherscan.io/address/0x329F56b127eAC6A842EfC065EaA933f74A109Af9

7.3 Codebase Github Link
https://github.com/nicockt/ftec4007