ECE 499 Project
Resonance
ERC-1155 Tokens for Music Artists

by

Sailesh Bechar
sbechar@uwaterloo.ca
20717409

Spring 2022
University of Waterloo

Project Supervisor
Behkish Nassirzadeh

Co-reader
Dr. Sebastian Banescu

Table of Contents

Abstract

Every day, up-and-coming music artists around the world struggle with the decision of signing their rights and royalties away to a record label, or to stay independent and build up a passionate following through their own means. As the technology of the internet continues to evolve, the latter path poses increasing potential to new artists. Fungible and non-fungible tokens within Smart Contracts enable a deeper level of engagement between the artist and their fans. They can capture more of each dollar their fans spend to support them, something which was previously unattainable through legacy distribution. In conjunction with Chainlink's Verifiable Random Function, this project adds additional complexity to make each piece of artwork unique, so that it holds a special place in the hearts of every fan. Each NFT also contains a popularity score, calculated on-chain using a FIFO data structure, to reflect the rate at which fans mint new tokens. This metric is designed in a way to reward artists who have yet to hit the mainstream.

Chapter 1

Introduction

1.1 Motivation

The popularization of fungible tokens created with ERC-20 [1] and non-fungible tokens created with ERC-721 [2] has open the gates for creators and fans to gain previously untapped value. Artists with a larger following such as Kings of Leon [3], Nas [4] and The Weeknd [5] are releasing their latest music projects as NFTs, paving the way for artists with a smaller following to follow in their footsteps. These growing, up-and-coming music artists struggle to make a living in the face of financial obstacles posed by music labels and streaming services. The most popular streaming platform Spotify, pays $0.00274 per stream [6]. From this, music labels will additionally take over 80% of the royalties [7]. Using Chainlink's Decentralized Oracle Network along side the ERC-1155 token standard within Smart Contracts, this project will help music artists create meaningful relationships with their fans and allow them to move closer to financial independence.

1.2 Problem Statement

Music artists are consistently met with financial obstacles when trying to grow and build their fan base. From the record labels that trap young artists with egregious loans, known as “advances”, on top of taking 80-90% of all album sale and performance royalties [8]. To the streaming services that only give back pennies to the artist for a song with hundreds of plays [6]. What both of these obstacles have in common is the centralized big pocket acting as the middle man, obscuring the relationship the artist has with their audience. They create a hurdle which obstructs the artist’s ideal objective, to create art their fans will enjoy. It becomes increasingly more difficult for the artist to achieve this, when they are constantly distracted with the stress of, “How I financially make my dreams a reality?” More often then not, they will sacrifice the quality of their work to conform to the formula laid out by record labels and streaming platforms, who bear the fiduciary responsibility to maximize shareholder value [9].

1.3 Proposed Solution

This paper proposes leveraging ERC-1155 tokens to enable small artists to gain financial independence, while building a strong relationship with their fan base. This is achieved through creating a variety of tokens of fixed quantity. A token with a quantity of 1, is essentially non-fungible, while a token with any other quantity is essentially fungible.

The fungible aspect of this contract will be a currency for the artist, as every token is identical. There will be a fixed amount initially created, and then distributed to their fans, as seen fit by the artist. This currency can serve many purposes, as seen with the fungible ERC-20 token standard, including functioning as votes for community decisions in a Decentralized Autonomous Organization (DAO)[10]. In addition, more value can be

created for the artist by reducing the supply of the fungible currency as the total amount of tokens is limited and unable to be partitioned.

When an artist creates a piece of music, being a song or an album, they want to create as many identical licences to use the artwork as possible. This suitably translates to a fungible token. However, to create additional value the artist can add a unique element to their artwork for every token, making them non-fungible. By introducing scarcity through Chainlink's Verifiable Random Function (VRF), the music's artwork can be dynamically chosen with a given probability. Additionally, the artist's current popularity score can be computed on-chain and added to each token. Value is provided to both parties, by adding a dimension of collecting and trading to the artist's music. Furthermore, imprinting an on-chain, immutable score of the artist's popularity can serve as cryptographic proof for how long super-fans have been supporting their favorite artist and the role they may have played in their successes.

1.4 Summary of the Implementation

The solution is implemented by the integration of the ERC-1155 Multi-Token standard for Smart Contracts, Chainlink's Verifiable Random Function, and on-chain computation leveraging a FIFO data structure.

The ERC-1155 Multi-token standard is used for creating both fungible and non-fungible tokens in one Smart Contract. Chainlink's VRF controls the frequency of how the fungible tokens are transferred to users and is used to assign a provable scarcity to the non-fungible tokens. In addition, a score of the artists current popularity is assigned to each non-fungible token at the time of minting. This score is calculated using a FIFO data

structure which captures the notion of `frequency of tokens minted` using the block number difference of each token from the token minted before it.

1.5 Contribution

1. Designed and Implemented Smart Contract to allow artists to create fungible tokens representing a currency and non-fungible tokens of their artwork with an assigned scarcity and popularity score
2. Designed example artwork of an artist's currency and various album scarcities
3. Implemented a Python script to upload the tokens' metadata to IPFS, compatible with OpenSea's marketplace
4. An experimental evaluation of the project using Brownie's testing suite which includes an integration test of the complete token generation workflow and a unit test of the correctness of popularity scores between artists

1.6 Background

A Blockchain is a distributed ledger that is commonly used in a decentralized form to create cryptocurrencies such as Bitcoin and Ethereum. As transactions accumulate on this ledger, independent nodes verify their legitimacy using a consensus protocol and append these transactions onto a new block. The blocks become chained together, resulting in the previous blocks being immutable. [11]

A Smart Contract is a computer program that is stored on the Blockchain. Being distributed and immutable, any node on the network should be able to run this program

and arrive at the same conclusion. As a result, the code can mediate the exchange between two or more parties without the need for a centralized authority. [12]

Artists are currently using the ERC-721 standard to create non-fungible tokens (NFTs) within Smart Contracts, for the Ethereum Virtual Machine (EVM). Throughout 2020 and 2021, this method has found tremendous success for digital artists, allowing fans to own their artwork on the Blockchain and to also gain utility from such ownership. Prior to NFTs, the ERC-20 standard within Smart Contracts has been widely used to create fungible tokens. These tokens can provide a variety of utility such as functioning as a currency, asset, lottery ticket or in-game reputation points [1].

The ERC-1155 token standard within Smart Contract aims to combine the fungible and non-fungible standards into a multi-token standard [13]. This enables the creation of both types within a single Smart Contract. When a user wants to create another token within the deployed Smart Contract, they do not have to pay the same gas fees as deploying a new smart contract. Instead, only a fraction of the gas fee is required to carry out this transaction. These fees originate from the nature of proof-of-work Blockchains. In this framework, "miners" execute significant computational work to cryptgraphically prove the validity of each new block. Since there are limited computational resources, the user pays a premium in the form of a gas fee to the miners to prioritize the execution of their transaction.[14]

A Decentralized Autonomous Organization (DAO) is a structure in which, every member has a common goal that attempts to serve the best interest of the entity (in this case, the artist). Only token holders are allowed to vote and all activity is posted on the Blockchain. [10]

Blockchains should use decentralized oracles to interface with the outside world. Chainlink is a standard that enables Smart Contracts to access any external data source through their Decentralized Oracle Network. Decentralized oracle networks (DONs) enable the creation of hybrid smart contracts, where on-chain code and off-chain infrastructure are combined to support decentralized applications (dApps) that react to real-world events and interact with traditional existing systems. This eliminates the vulnerability of a single-point of failure seen in a centralized oracle and maintains the security of the decentralized Blockchain. Chainlink's VRF ensures randomness is preserved when nodes are reaching consensus to push a new block, proving the scarcity of an asset. [15]

InterPlanetary File System (IPFS) is a distributed system of storing files. This makes the system more resilient, harder to censor and can have faster speeds if the content is hosted on a node closer to the user. Content is addressed with `/ipfs/` followed by a unique hash pointing to the directory, file or other form of data. The hash is generated based on the content, so if a user tries upload a duplicate, they will just be pointed to the preexisting content already hosted. A user also has the ability to `pin` the data on their machine, ensuring that the content will remain hosted if another node goes down. [16]

Pinata is a service for `pinning` IPFS content. There are many upsides such as speed, up time, space and cost, however, adds some degree of centralization, holding it to the same downsides as other centralized cloud providers. [17]

Brownie is a Python framework that leverages the `Web3.py` library to allow for easy Smart Contract development on the EVM. Accessing accounts, previously deployed contracts

and changing networks is simple to use with Brownie's command line interface. Also, PyTest, a testing framework is integrated to enable assertions and debugging. [18]

Chapter 2

Implementation

2.1 Problem Explanation

Small, up-and-coming music artists are constantly struggling to make ends meet financially. Rather than taking a loan from a record label and signing their soul away, many artists are now going independent with the age of the internet. The adoption of fungible and non-fungible digital assets stored on the Blockchain is gaining popularity as artists are finding new ways to engage their fans. This project gives super-fans the opportunity to collect a unique and rare version of their favorite artist's work, that contains a receipt of the artist's popularity at the time of acquisition.

2.2 Design/ Solution

The solution involves integrating both the ERC-1155 Multi-Token standard and Chainlink Verifiable Random Function (VRF) into a single Solidity smart contract.

When the contract is first deployed, it is funded with at least 2 LINK. This is Chainlink's

funglible ERC-20 token that is then transferred to Chainlink's Subscription Manager, to handle all VRF transactions. Once a unique subscription ID has been created, the contract is added as an approved VRF consumer. This results in the contract able to use the LINK held by this Subscription Manager to request random words, the cost being approximately 0.3 LINK. Additionally, 1000 of a token with `id=0` is minted to the contract, representing the fungible currency belonging to the artist.

When a user is ready to mint a non-fungible token (NFT), the first transaction to occur is to the Subscription Manager, to generate 2 random words. Once fulfilled, the random words are stored on-chain for subsequent token generation.

The next component of the minting process is generating a popularity score for the artist, at the time of NFT creation. Instead of gathering the popularity data of an artist from a centralized data source such as Spotify, this project aims to preserve the decentralized nature of the various Blockchain technologies and calculate their popularity on-chain, leveraging a FIFO data structure. To conserve gas, this FIFO should require as little memory and computational resources as possible. In Solidity, this can be achieved using a mapping datatype, storing key-value pairs. Each element in the queue stores the difference in the block number of the most recently minted NFT from the block number of the second most recently minted NFT, up to a size of 50. The mapping has a storing, deleting and lookup time complexity of $O(1)$ and a memory complexity of $O(n)$, representing the storage of each block number delta. Additionally, the storage capacity has an upper bound of the maximum size of the queue. To compute each element of the queue of block number deltas, only the previous block's number is required to be stored. When a new NFT is minted, the block number delta is computed by the difference of the current block number, from the stored previous block's number. The delta is then enqueued and the

previous block number is overwritten with the current block number.

```
uint256 avg_delta = delta_sum / delta_queue.length();
return (15000000 / avg_delta) + (
        item_counter * 1000000 / avg_delta
    );
```

Figure 2.1 Code snippet of popularity calculation

The intuition behind this formula is to capture the rate at which tokens are being minted. This can be approximated by computing the average block number delta of the whole queue. This is implemented by first storing a moving sum of the block number deltas in the queue. When a new block number delta is enqueued, it is then added to the sum. Subsequently, when a block number delta is dequeued, upon reaching the queue's max capacity, this delta is subtracted from the sum. As a result, the average block number delta can easily be computed by dividing the sum of block number deltas in the queue, by the number of elements in the queue.

This calculation only occurs when the length of the queue is greater than 2. If the queue did not have a fixed capacity, the FIFO data structure could be replaced by a single `uint256`. However, since we only want to include 50 block number deltas in our calculation, the oldest deltas should be subtracted from the sum as the queue reaches the storage limits. Without this constraint, we could continue adding the latest delta into the sum, effectively averaging the block number deltas since the initial mint. This is unfavorable in this project's use case as an artist's popularity does not change linearly. Including block number deltas from far in the past may serve as outliers and not as accurately reflect the artist's current popularity.

To capture this notion of token mint rate, the formula involves keeping the average block number delta of the whole queue, in the denominator. This quantity is essentially `1/time = frequency`, meaning the time between token mints is inversely proportional to the token's popularity score. As floating point precision is unsupported by the latest version (v0.8.0) of the Solidity language, the token mint rate is multiplied by a factor of 15 000 000.

There is an additional factor to the equation which includes the total number of tokens minted. This factor is included to differentiate small artists, from larger artists. For example, if an artist has had a significant amount of tokens minted and then their fans become less interested in their tokens, the total amount of items minted will have less weight in the score. If an artist is very popular, meaning the time between token mints is very low, every additional item minted will have a larger increase to the popularity score. It is this paper's assumption that the lower the popularity score, the more valuable the token will have, incentivizing users to support artists who are less well known. This factor is then scaled by a factor of 1 000 000 to remove the need for floating point precision, and to have the correct weighting to balance out the token mint rate.

Using the first random word provided by the Subscription Manager, the following distribution is created.

```
enum Scarcity{COMMON, RARE, ULTRA_RARE}
uint256 randomNum = randomWord[0] % 500;
if (randomNum < 1){
  return Scarcity(2);
}
else if (randomNum < 50){
  return Scarcity(1);
}
else{
  return Scarcity(0);
}
```

Figure 2.2 Code snippet of NFT scarcity distribution

Where there is a 1 in 500 (0.2%) chance the NFT will have a Ultra Rare scarcity, a 1 in 10 (10%) chance of Rare scarcity, leaving a 449 in 500 (89.8%) chance of Common scarcity. In production, the artist would be able to customize this distribution to more appropriately fit the size of their audience.

When an NFT is minted there is also a chance that the minter will be transferred one token of the artist's fungible currency. This is accomplished by means of the second random word provided by the VRF Subscription Manager. Here, there is only a 50% chance this token will be transferred to the minter.

```
uint256 randomDrop = randomNums[1] % 2;  // 50% chance
if (randomDrop < 1) {
  // Artist currency has ID = 0
  _safeTransferFrom(address(this), msg.sender, 0, 1, "");
}
```

Figure 2.3 Code snippet of transferring artist's currency

Once the users are finished minting NFTs, the metadata files are uploaded to InterPlanetary File System (IPFS). Leveraging Pinata's API, the JSON files are pinned to permanently remained cached on Pinata's nodes. Attributes generated on-chain such as popularity and scarcity are written to the metadata file to be displayed on the OpenSea marketplace.

```
metadata_template = {
    "name" : "Dancing with Divine Mother",
    "description" : "Dance through life knowing Divine
        Mother is ever-guiding you back to Her",
    "image" : "",
    "attributes" : [
      {
          "trait_type":"popularity",
          "value" : 0
        },
        {
          "trait_type":"Scarcity",
           "value" : ""
        }
  ]
}
```

Figure 2.4 Code snippet of metadata

Each metadata file also specifies an image that will be displayed in the marketplace. These images are stored prior using Pinata's IPFS API, corresponding to their respective scarcity.

Figure 2.5 Example artwork of artist's currency

Figure 2.6 Example artwork of common NFT

Figure 2.7 Example artwork of rare NFT

Figure 2.8 Example artwork of ultra-rare NFT

Chapter 3

Experiments

The Python framework Brownie was used to deploy the smart contract, and execute all on-chain functions. Brownie leverages Web3.py, a framework for interacting with Ethereum, to execute all EVM interactions, in a high-level, user-friendly way.

Parameters required for Chainlink's VRF were added into a `brownie-config.yaml` file. This allows the Python script to fetch chain-specific addresses programatically, based on the network currently in use. This file also includes compiler remappings for importing GitHub libraries and accessing `.env` files for wallet private keys.

Two main functions were tested using Brownie's built-in testing framework, leveraging PyTest. The first function was an integration test for the full end-to-end workflow of NFT creation. The second function was testing the correct relative popularity scores of three different artists' NFTs, where each artist has their own distinct level of popularity.

3.1 Set Up

The preparation of the first experiment begins with deploying the smart contract. Rinkeby was the chain chosen for the testing on the Ethereum Blockchain because it is the only testnet compatible with Chainlink's VRF. When initializing, a new subscription to Chainlink's VRF is created, returning an `id` for subsequent interaction. In addition, 1000 of fungible tokens representing the artist's currency is minted to the contract. Next, the contract is funded with 2 of Chainlink's ERC-20 token, LINK. The contract then transfers the LINK to the VRF Subscription Manager. This manager is responsible for all random word generation and controls permissions for can generate and consume random words.

The second experiment involves an identical process, except repeated three times. This is for the three artists, whose popularity will be tested, each with their own associated smart contract.

3.2 Procedure

The first experiment, representing the integration test, begins by generating 2 random words. The words are of type `uint256` and are stored on-chain. Then, the `mint_nf_artwork()` transaction is called in the smart contract. The NFT representing the artist's music is subsequently minted and given a scarcity based on the specified distribution. The scarcity is emitted as an event to compare in the next portion of the test. In addition, one token is transferred to the contract caller, with a probability of 50%. Finally, the subscription to the VRF Subscription Manager is cancelled, with the remaining LINK transferred to the contract deployer's address.

The second experiment involves a similar process, however, with additional complexity

since three contracts will be tested against each other. Each contract is first deployed and then cleaned up before deploying the next contract. When cleaning up, the unused LINK from the previous Subscription Manager can be retrieved to use for the next contract's Subscription Manager. When an NFT is minted, its popularity score is emitted as an event. Each contract mints 5 NFTs and stores the popularity scores in a list in the test. To distinguish the contract popularities, the second artist waits for 60 seconds after each individual mint of their total 5 NFTs. The third artist waits for 180 seconds between mints. The expected result is the longer the period of time is between each mint transaction, the lower that NFTs popularity score will be of the artist.

3.2.1 The Results

To test the accuracy of the integration test, the two generated random words are asserted against the same logic as executed in the contract. As the first random word determines the scarcity, the test also applies the same operations to verify the scarcity generated, is the same. Rather than using the same implementation of the scarcity distribution used in production, the test uses a uniform distribution to more easily distinguish between the three different possible values.

```
tx_mint = resonance.mint_nf_artwork({"from":deployer})

scarcity = tx_mint.events["scarcityAssigned"]["scarcity"]
assert resonance.randomNums(0) % 3 == scarcity

deployer_balance = resonance.balanceOf(
                            deployer, 0, {"from":deployer}
                        )
calc_currency_transfered = 1 if resonance.randomNums(1)
                                % 2 < 1 else 0
assert calc_currency_transfered == deployer_balance
```

Figure 3.1 Assertions of Main Functionality of Integration Test

To test if the artist's fungible currency was transferred to the caller of the mint transaction, the second generated random word has the same logic applied. Built into the ERC-1155 standard `balanceOf()` conveniently returns the balance of any `itemID` associated with any address. In addition, the clean up functionality is tested so that once the subscription is cancelled. The deployer's address should have more LINK than before the clean up was initiated.

```
link_before = interface.LinkTokenInterface(
config["networks"][network.show_active()]["link_token"])
    .balanceOf(deployer, {"from": deployer})
tx_cancel = resonance.cancelSubscription(
        deployer, {"from":deployer}
    )
tx_cancel.wait(1)

link_after = interface.LinkTokenInterface(
config["networks"][network.show_active()]["link_token"])
    .balanceOf(deployer, {"from": deployer})

assert link_after > link_before
```

Figure 3.2 Assertions of Clean Up in Integration Test

The second experiment compares the values of popularity between each of the three contracts. As the delay between mint transactions is constant for each of the artists, it is expected that each element `i` of `artist_1` should be greater than the popularity score of element `i` of `artist_2`, which should then be greater then the popularity score of element `i` of `artist_3`. In the case where there are less than 2 prior mint transactions, each of the artists will share the popularity score of 0.

```
for i in range(5):
        assert a1_popularity[i] >= a2_popularity[i]
        assert a2_popularity[i] >= a3_popularity[i]
```

Figure 3.3 Assertions of Popularity Scores Between Artists

An example output of the popularity scores is given here. We can see the descending order of the popularity as the time between mint transactions increases.

```
Artist 1 Popularity: [0, 0, 2250000, 2375000, 2500000]
Artist 2 Popularity: [0, 0, 1500000, 1583333, 1666666]
Artist 3 Popularity: [0, 0, 900000, 950000, 1000000]

Artist 1 Popularity: [0, 0, 2250000, 1727272, 2000000]
Artist 2 Popularity: [0, 0, 1500000, 1583333, 1666666]
Artist 3 Popularity: [0, 0, 900000, 826086, 909090]
```

Figure 3.4 Examples of Popularity Scores Between Artists

Chapter 4

Limitations

In production, IPFS may not be a feasible solution for hosting ERC-1155 token metadata. This is because such metadata files are accessed by their Uniform Resource Identifier (URI) in the structure of `https://token-cdn-domain/{id}.json`. where id is a hex address padded to 64 characters. When a file is uploaded via IPFS, it is accessed through a unique hash identifier that includes its directory structure. The relationship between the file and the folders its located in is static, meaning uploading a new file to a directory, would treat this directory as unique and generate a new hash identifier. In real-world applications, this approach is inefficient as it would require re-uploading the folder containing every metadata file each time a new NFT is minted.

This project used only Rinkeby as the testnet for Ethereum testing. This is due to Chainlink's VRF only being compatible with the Rinkeby chain. Unfortunately, Rinkeby has become deprecated by the Ethereum team and has moved on to other testnets such as Goerli and Sepolia. This inhibits accurate testing of the smart contract on the Ethereum mainnet, which runs on a Proof-of-Work (PoW) protocol.

The integration test also does not test the deploying of the metadata to IPFS and subsequently the tokens displaying on the OpenSea marketplace. This is due to this functionality being independent of that which the smart contract is responsible for and requires a more complex testing suite to ensure feasibility, such as integrating with the framework Selenium to verify HTML elements.

This project currently does not wish for the token to become regulated by the SEC, thus the fungible currency should not pass the Howey test [19]. However, if the token were to be officially registered as a security, a fungible token acting as a security for the artist may potentially provide tremendous value.

Chapter 5

Conclusions

Often, music artists financially struggle to support themselves, especially during the time in their careers when they are still building a loyal and adamant following. Instead of signing a one-sided deal with a record label, many artists are choosing to remain independent and grow using technologies such as Smart Contracts. This paper proposed using the ERC-1155 Multi-token standard for Smart Contracts, in addition to Chainlink's Verifiable Random Function to create more dimensions to their artwork. Albums can now have different levels of scarcity and be imprinted with the current popularity of the artist, at the time of creation. The decentralized nature of these technologies keep these qualities of the artwork free from the tampering of any centralized authority. The popularity score is calculated on-chain and uses efficient data structures to minimize gas fee.

Going forward, this project would be benefited to be deployed on a low-gas fee blockchain such as Polygon. When an artist is small, their fans do not want to spend hundreds of dollars to support them, especially when the bulk of that cost is sent to the network miners, executing the transactions on-chain. To address the static nature of IPFS, an extension can be used called InterPlanetary Name System (IPNS). It solves this issue by creating an

address that can be mutable, by changing how the files are referenced. Instead of Rinkeby, Polygon's Mumbai can be used to simulate the Polygon Blockchain, also supported by Chainlink's VRF. Finally, a more comprehensive testing suite can be used to verify the metadata successfully being uploaded to a marketplace.

References

[1] V. Buterin, F. Vogelsteller, Eip-20: Token standard, 2015. URL: https://eips.ethereum.org/EIPS/eip-20.

[2] D. Shirley, W. Entriken, Eip-721: Non-fungible token standard, 2018. URL: https://eips.ethereum.org/EIPS/eip-721.

[3] S. Hissong, Kings of leon will be the first band to release an album as an nft, 2021. URL: https://www.rollingstone.com/pro/news/kings-of-leon-when-you-see-yourself-album-nft-crypto-1135192/.

[4] A. Hennis, Earn lifetime royalties from grammy-award-winning artist nas, 2022. URL: https://marketrealist.com/p/how-to-buy-nas-nft/.

[5] R. Perper, The weeknd's limited-edition nft collection raises over $2 million usd, 2022. URL: https://hypebeast.com/2021/4/the-weeknd-nft-collection-song-artwork-2-million-nifty-gateway-auction.

[6] M. Kristiansen, How much does spotify pay artists per stream, 2021. URL: https://homestudioideas.com/how-much-does-spotify-pay-artists/.

[7] P. Schwartz, Ponterio; Levenson, 3 reasons why musical artists are getting underpaid, 2021. URL: https://www.splaw.us/blog/2021/11/3-reasons-why-musical-artists-are-getting-underpaid/.

[8] K. Cornell, The artist; record label relationship – a look at the standard "record deal", 2017. URL: https://www.tunecore.com/blog/2017/05/artist-record-label-relationship-look-standard-record-deal-part-1.html.

[9] D. Andersson, How record labels screw artists, 2020. URL: https://elevenbstudios.com/how-record-labels-screw-artists/.

[10] N. Reiff, Decentralized autonomous organization (dao), 2022. URL: https://www.investopedia.com/tech/what-dao/.

[11] A. Hayes, Blockchain explained, 2022. URL: https://www.investopedia.com/terms/b/blockchain.asp.

[12] J. Frankenfield, Smart contracts: What you need to know, 2022. URL: https://www.investopedia.com/terms/s/smart-contracts.asp.

[13] A. Cooke, W. Radomski, Eip-1155: Multi token standard, 2018. URL: https://eips.ethereum.org/EIPS/eip-1155.

[14] ledger, What are gas fees?, 2021. URL: https://www.ledger.com/academy/blockchain/what-are-gas-fees.

[15] Chainlink, What is an oracle in blockchain? " explained: Chainlink, 2021. URL: https://chain.link/education/blockchain-oracles.

[16] I. Docs, What is ipfs?, 2022. URL: https://docs.ipfs.io/concepts/what-is-ipfs/.

[17] K. Tut, What is an ipfs pinning service?, 2020. URL: https://medium.com/pinata/what-is-an-ipfs-pinning-service-f6ed4cd7e475.

[18] B. Hauser, Introducing brownie, 2020. URL: https://iamdefinitelyahuman.medium.com/introducing-brownie-a763859409ca.

[19] N. Reiff, Howey test, 2021. URL: https://www.investopedia.com/terms/h/howey-test.asp.