SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER SCIENCE

DBet

BLOCKCHAIN AND DISTRIBUTED LEDGER TECHNOLOGIES

Professors:
Claudio Di Ciccio

Students:
Giovanni Pica
1816394
Alessandro Torri
1835715

Academic Year 2022/2023

Contents

1 Introduction **2**
1.1 Main Responsibilities 2
1.2 Outline of the report 2

2 Background **3**
2.1 History 3
2.2 Applications 4

3 Presentation of the context **6**
3.1 Brief explanation of the margin 6
3.2 Aim of DBet 7
3.3 Type of Blockchain 7

4 Software Architecture **9**
4.1 Front-End 9
4.2 Back-End 9
4.3 UML Class Diagram 10
4.4 UML Use Cases Diagram 10
4.5 UML Components Diagram 11
4.6 UML Collaboration Diagram 12
4.7 UML Activity Diagrams 13

5 Implementation **16**
5.1 GUI 16
5.2 Contract 18
5.3 Server 20

6 Conclusions **21**
6.1 Issues & Limitations 21
6.2 Future Works 21

References **22**

1 Introduction

DBet is a Decentralized Application implemented in **Solidity** that allows users to perform sport bets on the blockchain. In this way there are **no third-party bookmakers**, meaning that the user doesn't leak personal information and also he **doesn't lose the margin**. If a gambler wins enough bets, he can buy (with a special token that will be presented later) some fancy **NFT**s that he can then sell on the market for ETH.

1.1 Main Responsibilities

Each of us worked equally for the front-end programming, front-end communication and Solidity Development and among the things made individually there are:

1. *Giovanni Pica*: software-architecture;
2. *Alessandro Torri*: server-side programming, scraping.

1.2 Outline of the report

The report is structured like this:

- **Background**: in which are described hystory, main concepts, application domains and some ideas for the blockchain in general.
- **Presentation of the context**: in which is described the idea of the Dapp with an in-depth explanation of DBet.
- **Software Architecture**: diagrams to describe the main components of the architecture, the Smart Contract, usage scenarios, activities concerning the tokens.
- **Implementation**: how this application is implemented, the code, the GUI interfaces and some demos.
- **Conclusions**: a brief recap, knows issues and limitations and some future works.

2 Background

A **Blockchain** can be defined as an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way. A transaction is the operation in which there is the transfer of crypto assets (e.g. ether) from an account A to an account B. A ledger is an ordered collection of transactions that are grouped in blocks; so, as the word says, the blockchain is a chain of blocks that are linked to their predecessor thanks to hash functions. To reach consensus the blockchain has particular algorithms. Some key characteristics [1] of this technology are:

- **Ledger**: append only ledger to provide full transactional history. Unlike traditional databases, transactions and values in a blockchain are not overridden.
- **Secure**: cryptographically secure, the data within the ledger is attestable.
- **Shared**: the ledger is shared amongst multiple participants. This provides transparency across the node participants in the blockchain network.
- **Distributed**: the blockchain can be distributed. By increasing the number of nodes, the ability for a bad actor to impact the consensus protocol used by the blockchain is reduced.

2.1 History

This technology has a recent impact but it has also a history [2] [3] dating back 30 years. One of the first cryptocurrency was **eCash** invented by David Chaum in 1995 that was the owner of the DigiCash company. This currency emphasizes on anonimity as a key benefit and in fact not even the government was able to decipher encrypted eCash transfers. Chaum was unable to convince banks to support the project and without an internet infrastructure to support peer-to-peer transactions and only exchanges, the project failed. DigiCash declared bankruptcy in 1998. In 1991 Stuart Haber and W. Scott Stornetta describe a "cryptographically secured chain of blocks" in 1991, this is the first time where an electronic ledger was studied. In 1998 there was another digital cash called **b-money** invented by Wei Dai that is based on the minting of money through computationally expensive puzzle; this was the first time where the term **mining** was cited. In 2004 Hal Finney invent **Reusable Proof-of-Work** (RPOW) [4], it was intended as a prototype for a digital cash based on Nick Szabo's theory of collectibles. RPOW was a significant early step in the history of digital cash and was a precursor to Bitcoin. Finally, in 2008, there is the outstanding paper published pseudonymously by Satoshi Nakamoto that describe **Bitcoin**: a peer-to-peer electronic cash system [5] that works like a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership. The consensus algorithm to decide which is the next block is called **Proof Of Work** that is in brief a computationally expensive puzzle and when a node find a POW it broadcasts it to all nodes. In 2011 there were other new coins like **Litecoin** or **Namecoin**, in 2012 **Peercoin** and a memecoin called **Dogecoin**. After that new ideas of utilizations for the blockchain came up. The new evolution was to create a programmable blockchain for application development and in 2015 **Ethereum** [6] was invented, a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions. "Ether" is the main internal crypto-fuel of Ethereum. In this blockchain, there are two types of accounts: externally owned accounts, controlled by private keys, and contract accounts, controlled by their contract code. An externally owned account has no code, and one can send messages from an externally owned account by creating and signing a transaction; in a contract account, every time it receives a message its code activates, allowing it to read and write its internal storage and send other messages or create contracts in turn. The actual consensus protocol is called **Proof Of Stake** and the difference with POW is that the validators explicitly stake capital in the form of ETH into a smart contract on Ethereum. The idea is that the validators act as a committee to propose and add new blocks and it is more convenient for them to behave according to the rules (if they do so they get a reward, otherwise they could incur in slashes of their stake).

2.2 Applications

The application developed with the usage of Smart Contracts are called **Decentralized Application** (Dapps) and some interesting areas are:

- **Finance**: applications that focus on building out financial services using cryptocurrencies. They offer the likes of lending, borrowing, earning interest, and private payments – no personal data required.
- **Arts and Collectibles, NFTs**: applications that focus on digital ownership, increasing earning potential for creators, and inventing new ways to invest in your favourite creators and their work.
- **Gaming**: applications that focus on the creation of virtual worlds and battling other players using collectibles that hold real-world value.
- **Decentralized Technology**: applications that focus on decentralizing developer tools, incorporating cryptoeconomic systems into existing technology, and creating marketplaces for open-source development work.

- **Voting**: applications that allow users to vote online with anonimity.
- **Gambling**: applications that have any gambling products that include blockchain elements. For on-chain gambling apps (gambling dapps), fairness and transparency are their main advantages as most gambling dapp contracts are open-sourced and built on the blockchain. And these gambling dapps do not require registration or sign-in, making it very convenient for users.
- **Social Media**: social media platforms that are built on the blockchain. Two of the main advantages are censorship-resistance and reward incentives.

3 Presentation of the context

Betting on football events has become very popular in our days with the arrival of web betting applications. A user can interact directly with the odds of a bookmaker and bet on a particular event just by going to the application on the smartphone or the web interface and insert its login credentials and its wallet. However, when the user uses these websites/apps, he leaks personal informations to the bookmaker; furthermore, he bets in an unfair setting because of the margin [7] the bookmaker puts on the odds. These two reasons were the ones that inspired our idea; indeed, implementing a betting system on the blockchain can remove both of these problems.

3.1 Brief explanation of the margin

The odds of an event are defined as the inverse of their probability. This means that the bookmaker odds are the inverse of the probability estimated by it; However, contrary to what you would expect, if you take the inverse of the bookmaker odds of some events that are complementary, they do not sum up to 1; For instance, let's take the odds of the match Celta Vigo - Villareal by bet356. 1 (Celta Vigo wins) has odds 2.7, X (draw) has odds 3.1 and 2 (Villareal wins) has odds 2.8. The sum of their inverse is $\frac{1}{odd_1} + \frac{1}{odd_X} + \frac{1}{odd_2} = p_1 + p_x + p_2 = \frac{1}{2.7} + \frac{1}{3.1} + \frac{1}{2.8} = 0.37 + 0.32 + 0.35 = 1.04$. The margin of profit of the bookmaker is defined as this sum -1; so, in the above example the margin is $m = 1.04 - 1 = 0.04$. In our application, we removed the margin of profit with the following formula:

$$\hat{odd_1} = \frac{3 * odd_1}{3 - (odd_1 * m)} \tag{1}$$

Where 3 is basically the number of events that forms the partition (in our case 1,X and 2). Whit this formula the new odds are:

- $\hat{odd_1} = \frac{3*2.7}{3-(2.7*0.04)} = \frac{8.1}{2.892} = 2.8$
- $\hat{odd_X} = \frac{3*3.1}{3-(3.1*0.04)} = \frac{9.3}{2.876} = 3.23$
- $\hat{odd_2} = \frac{3*2.8}{3-(2.8*0.04)} = \frac{8.4}{2.888} = 2.91$

If we now try to recompute the margin with these new odds we get $\hat{m} = \left(\frac{1}{2.8} + \frac{1}{3.23} + \frac{1}{2.91}\right) - 1 = (0.357 + 0.309 + 0.343) - 1 = 1.009 - 1 = 0.009$ that, apart from floating point approximation, is basically 0. So, in our Dapp, the user can definitely bet without margin and fairly; if he bets 100 tokens on the 1 event he will get (in case of win of course) 100*2.8 = 280 tokens that are 10 tokens more that with the unfair odds; this may not seem a lot, but in some cases it can really do the difference.

3.2 Aim of DBet

DBet provides to users anonimity, because you are not leaking personal informations like credit card, name, surname and so on and it also provides the true odds for football events. To make the system both live and profitable for the contract owner(us) we decided also to create a NFT marketplace where users can get new NFTs and sell them to earn ether. To perform all of these stuffs the application works with three kind of tokens:

- **entryTokens**: these tokens are used only to perform bets and they can be minted in exchange with ether.
- **superTokens**: they are created as a reward for a winning bet and they can be used both to get new NFTs forged by the creator and to perform bets.
- **NFTs**: they are obtained with superTokens and after this operation the owner can decide to sell them for ether. They are generated by the creator each time that an event happens (e.g. Messi won the world cup) and they have also rarity (only a graphical and psychological aspect) which could be Bronze, Silver and Gold. The creator also decides how many superTokens are needed to get them.

3.3 Type of Blockchain

A blockchain is useful for the reasons explained earlier and the type of Blockchain that we use is a **Permissionless** one because it has some key benefits [8] for our purpose:

- **No central authority** that manages the network, any user can participate. No third party bookmaker.
- **Transparency** because every transaction is visible to any node.
- **Pseudo-Anonimity** because the user has an alphanumerical ID and so long as no real connection is drawn between that ID and the user, it's difficult to trace one from the other.

Other four reasons according to the Wüst & Gervais [9] Model (Figure 1) we have:

- **To store state**.
- **Multiple writers**.
- **No TTP (Trusted Third Party)**.
- **Unknown writers**.

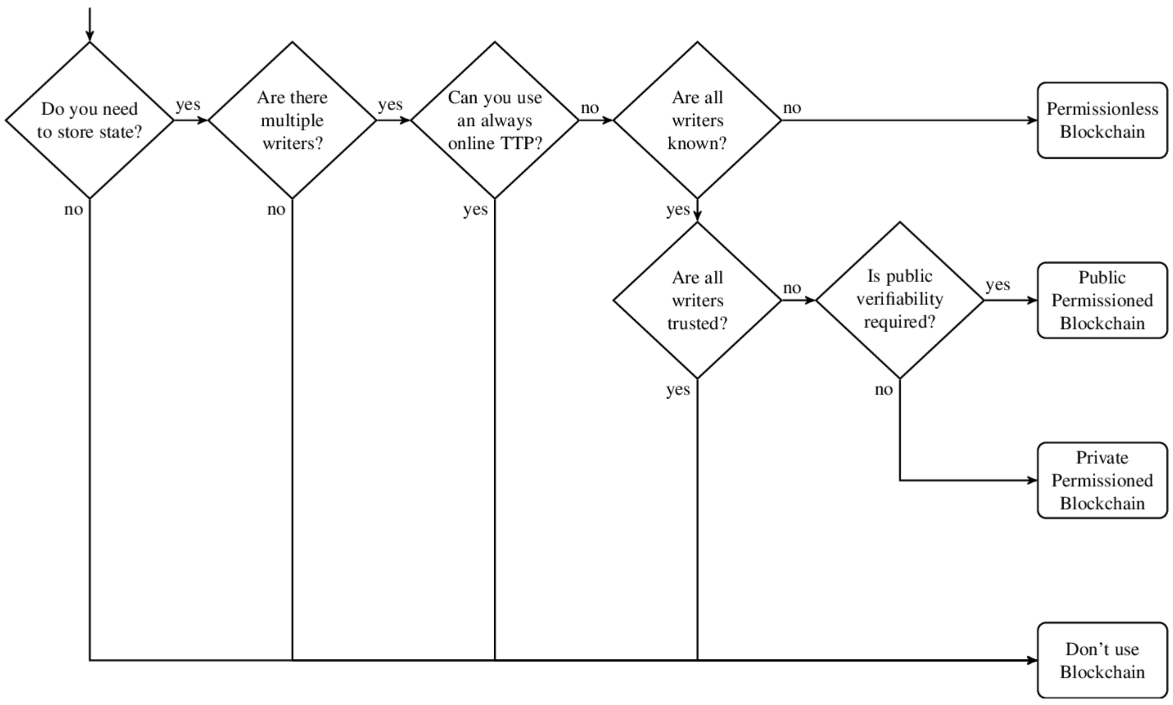

Do you need to store state?
yes
no
Are there multiple writers?
yes
no
Can you use an always online TTP?
no
yes
Are all writers known?
no
yes
Permissionless Blockchain
Are all writers trusted?
no
yes
Is public verifiability required?
yes
no
Public Permissioned Blockchain
Private Permissioned Blockchain
Don't use Blockchain

Figure 1: Do you need a Blockchain?

4 Software Architecture

In this section will be described the **front-end** and **back-end** with the usage of some UML diagrams. The front-end is simply the user interface, something that the user can see and can interact with; the back-end is the opposite, is the layer in which the data is accessed. This two parts work together to create the full user experience.

4.1 Front-End

The technologies used in the development of the user interface of our application are **HTML** which is a markup language, **CSS** which is a style language and **JavaScript** which is a programming language. HTML was used to create the web pages and CSS to change some style graphics with the help of **Bootstrap** which is an open source front-end development framework for the creation of websites and web apps. JavaScript was used to perform some particular scripts dynamically, for example to call the contract or to get the address or to add some graphical things like button or input texts.

4.2 Back-End

This part of the application is where the Smart Contract is defined and implemented in **Solidity** which is an object-oriented, high-level language for implementing smart contracts in the Ethereum Blockchain. The server implementation is in **Flask** that is a framework used for the development in python of web applications. Flask server is useful to store files and to communicate with the blockchain through web3 python library. In this way we can get from the blockchain information like the transaction data, useful in our case to get the bets and the NFTs with a script in python. The server is used also for scraping the match results and this is also a script written in python. To test our application we decided to use **Ganache**, useful to quickly fire up a personal Ethereum blockchain which you can use to run tests, execute commands, and inspect state while controlling how the chain operates.

4.3 UML Class Diagram

The Class Diagram is a diagram where all the content of a class is described, like attributes, relations and methods. In Figure 2 we can see the Smart Contract content and specify like attributes and some relations, in our case with NFT struct which contains all attributes of a NFT. And also there are methods.

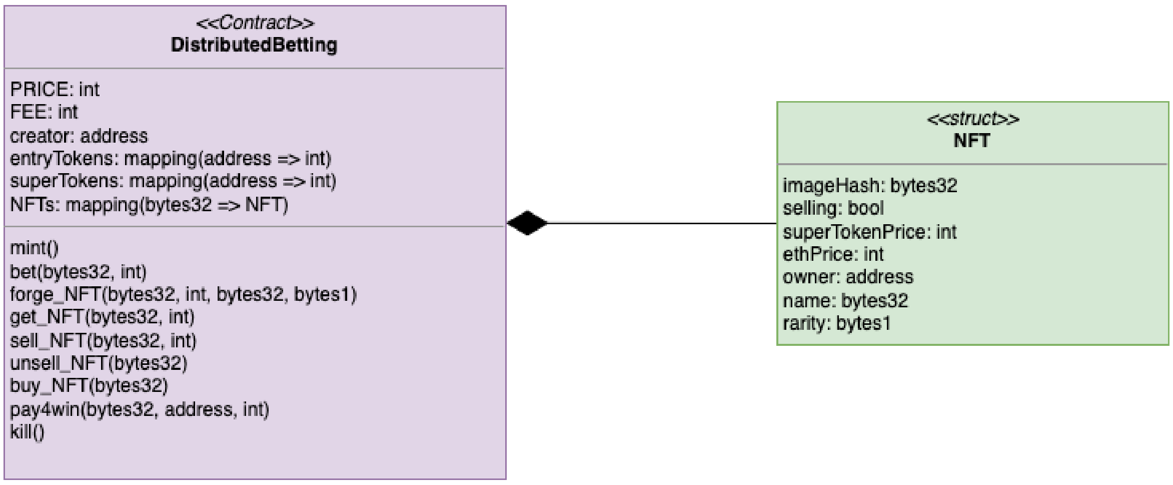


Figure 2: Class Diagram of the Smart Contract Distributed Betting

4.4 UML Use Cases Diagram

The Use Cases Diagram in Figure 3 is useful to illustrate the main usage scenario of the Dapp.

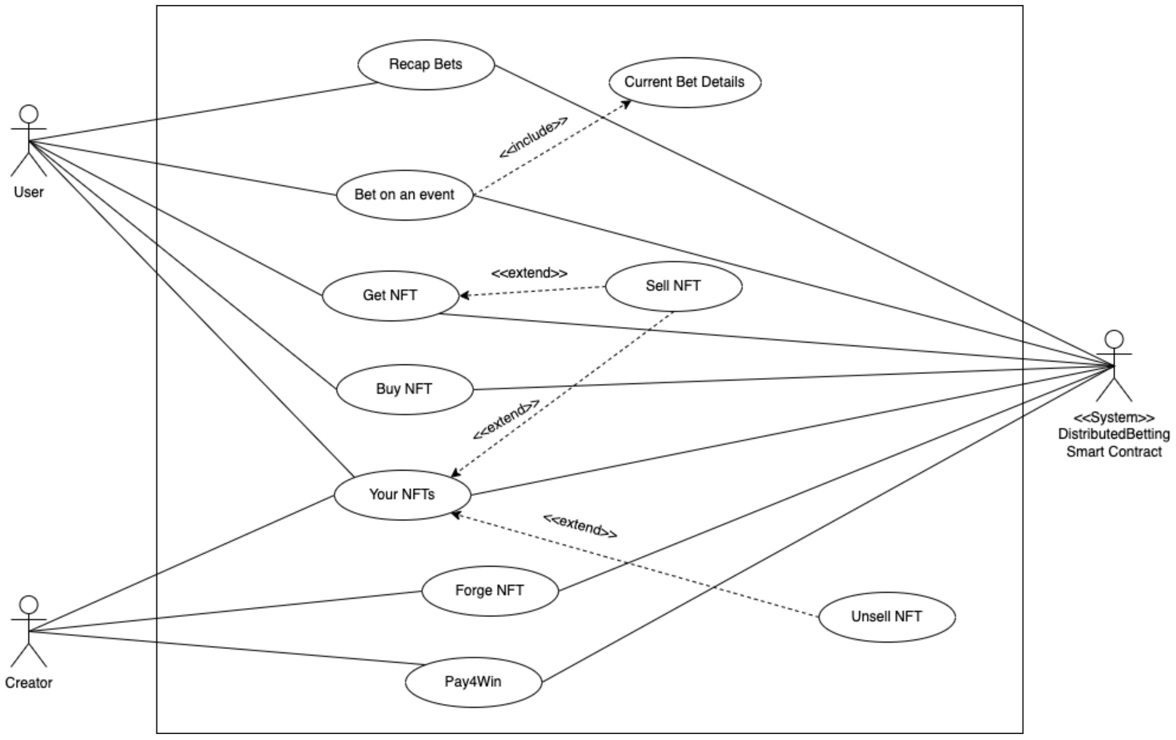


Figure 3: Use Cases

4.5 UML Components Diagram

The Components Diagram is used to describe which are the front-end and back-end modules and how they interact with each other. In Figure 4 we can see the main components used for the development of the software application. Some of the components were described in previous sections but let's talk about Metamask and Remix IDE. **Metamask** is a crypto wallet that works like an extension of your browser, is used to approve transaction and to switch wallets. **Remix IDE** is a no-setup tool with a GUI for developing smart contracts. Each of them support Ganache. Another important thing is the scraping part, that works with a script in python that fetch and extract information from a web page with the help of **Selenium** (a tool to automate browsers); to correctly work it needs the geckodriver for Firefox browsers and chromedriver for the Chrome ones. In our project the scraped page is `www.diretta.it` which is a site where there are all the match results and odds.

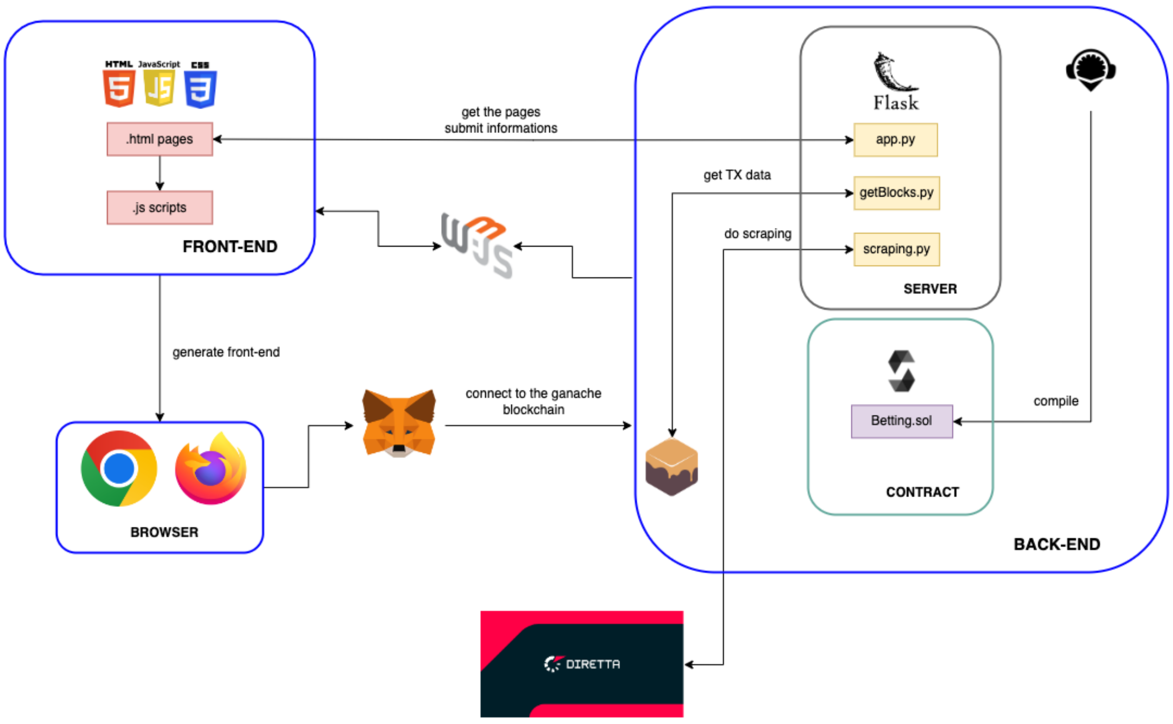


Figure 4: Components Diagram

So the main parts of the Dapp are:

- **.html** pages that interacts with the **.js** scripts and they are the GUI of the users.
- **app.py** is the main script of the Flask server used to store files, to get submitted information or to call the **getBlocks.py** that communicates with the blockchain or to call the **scraping.py** that performs scraping.
- **Betting.sol** contains the code of the Smart Contract.

4.6 UML Collaboration Diagram

The Collaboration Diagram is a diagram that describe the interactions between che components of a software. In Figure 5 there are the most interesting actions like Bet on a football event, Forge a new NFT, or buy a NFT.

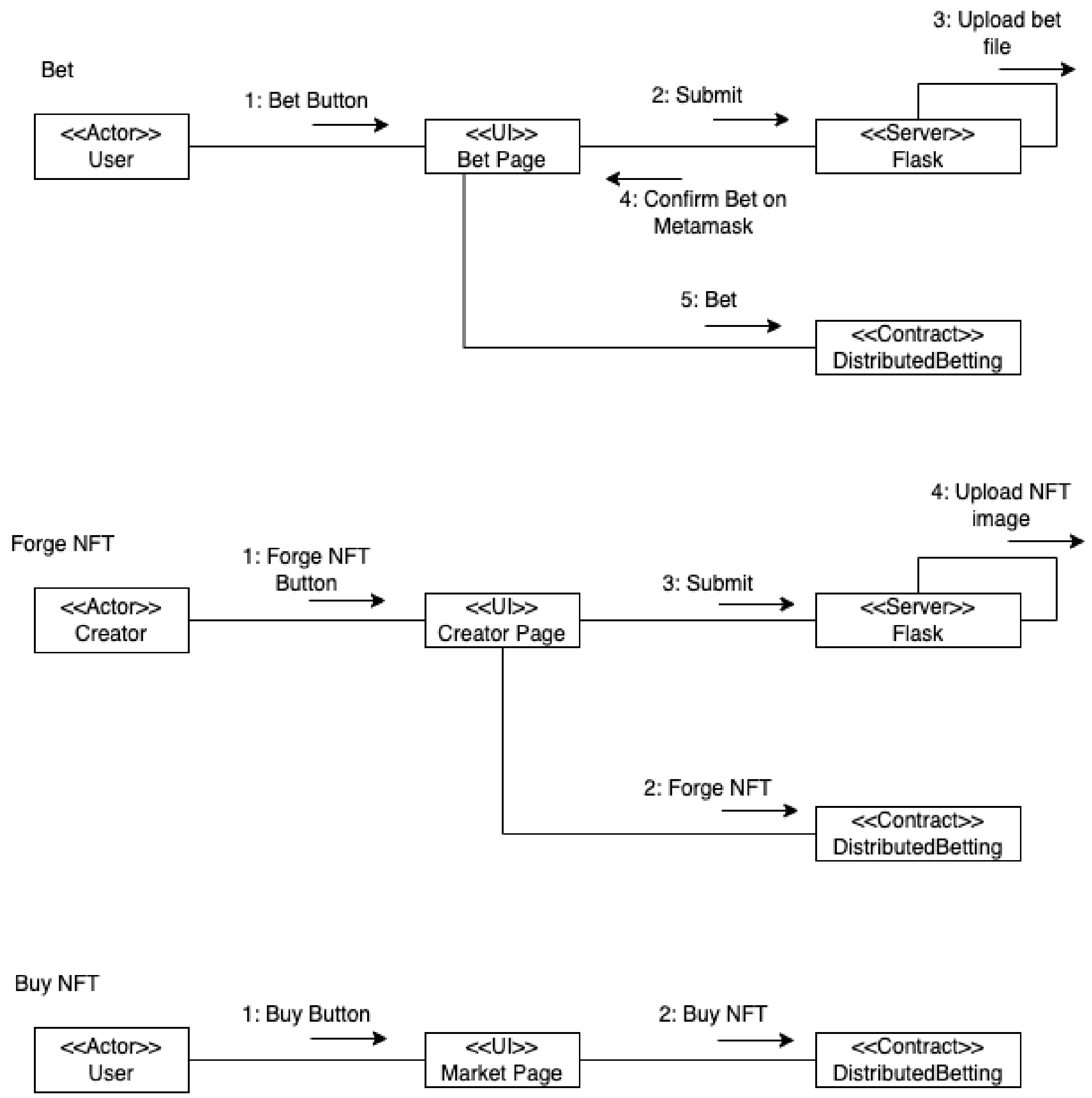


Figure 5: Collaboration Diagram

4.7 UML Activity Diagrams

The Activity Diagrams in Figure 6, 7, 8 describe how activities are coordinated to provide a service which can be at different levels of abstraction. In this case is used to see how the system works when one of the three token is given.

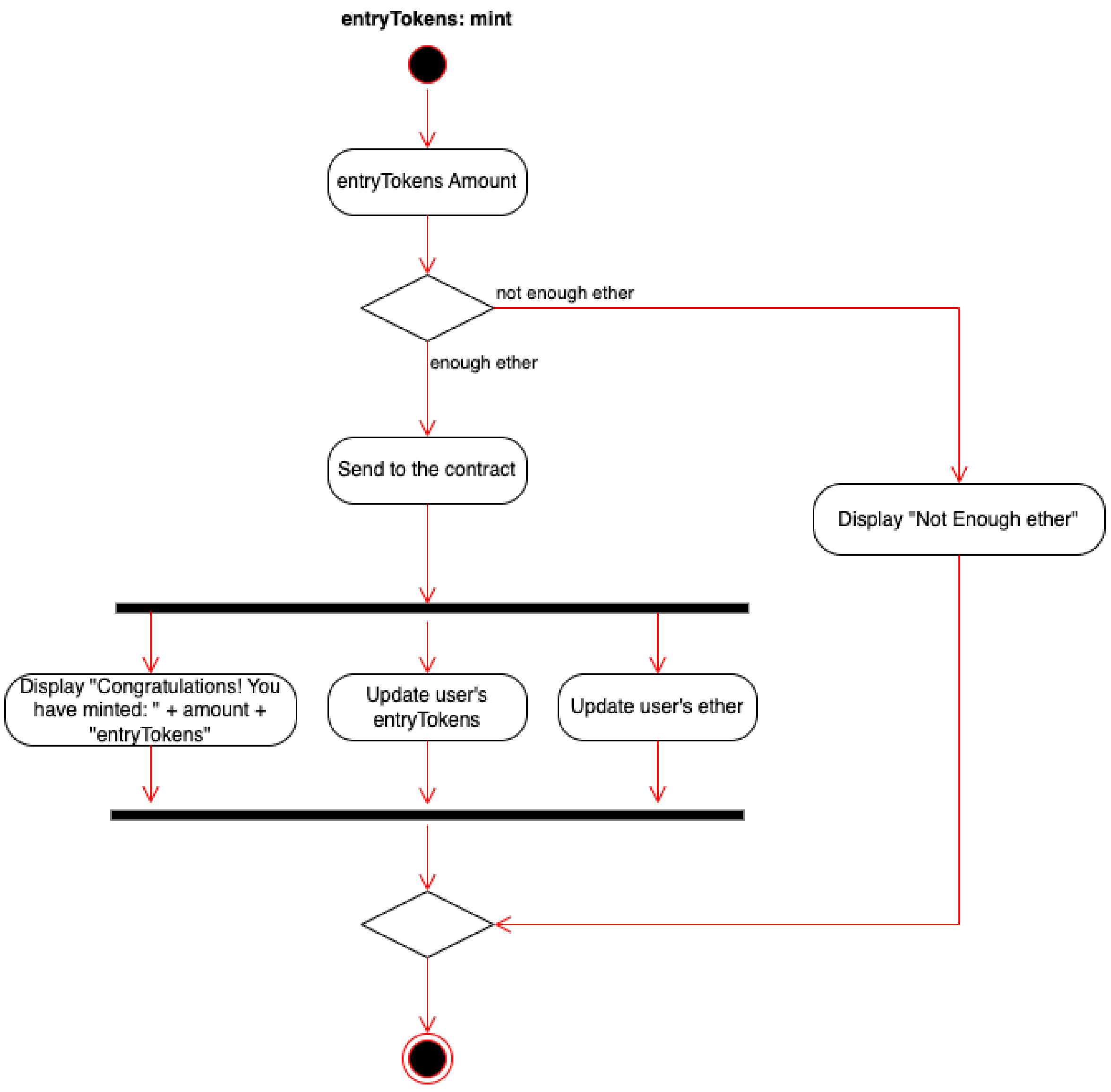

entryTokens: mint
entryTokens Amount
not enough ether
enough ether
Send to the contract
Display "Not Enough ether"
Display "Congratulations! You have minted: " + amount + "entryTokens"
Update user's entryTokens
Update user's ether

Figure 6: minting of entryTokens

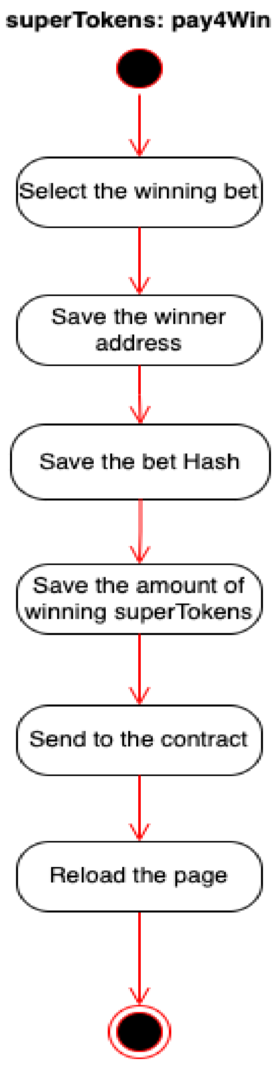


Figure 7: paying winning users with the generation of superTokens

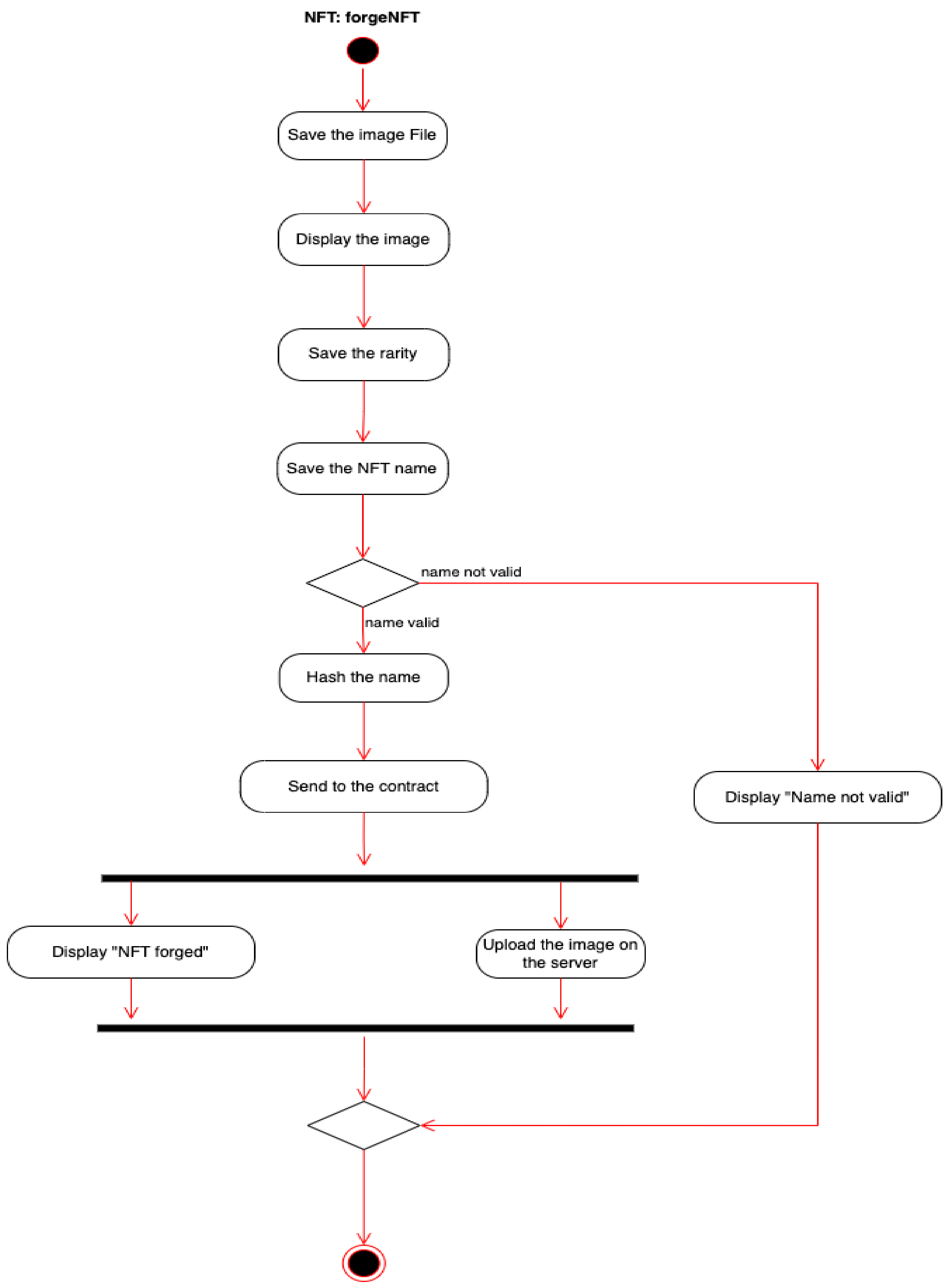
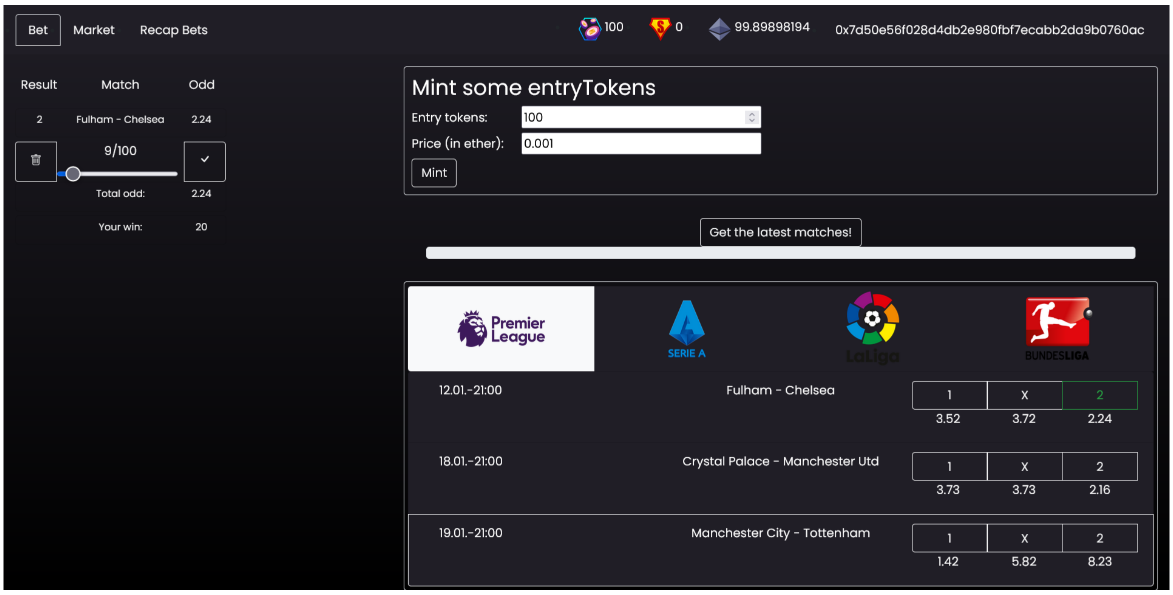


Figure 8: forge of new NFTs

5 Implementation

5.1 GUI



Result	Match	Odd
2	Fulham - Chelsea	2.24
	9/100	
	Total odd:	2.24
	Your win:	20

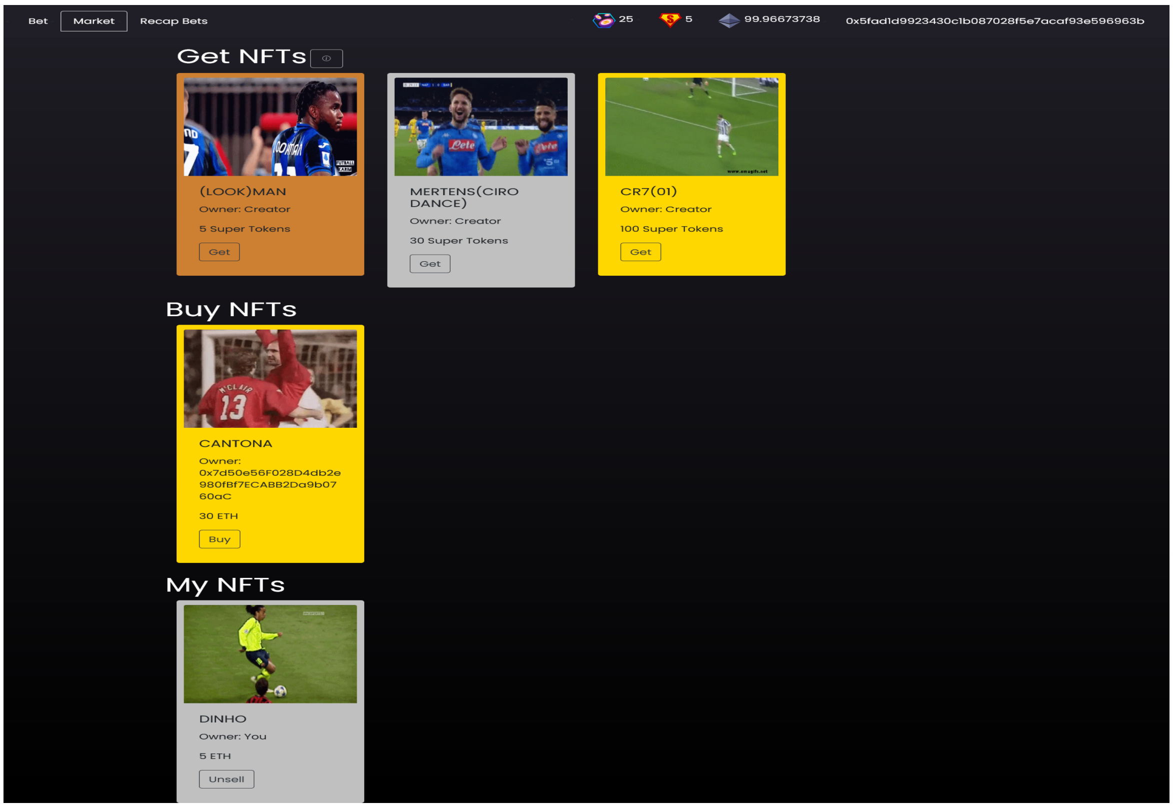
Mint some entryTokens

Entry tokens: 100

Price (in ether): 0.001

		1	X	2
12.01.-21:00	Fulham - Chelsea	3.52	3.72	2.24
18.01.-21:00	Crystal Palace - Manchester Utd	3.73	3.73	2.16
19.01.-21:00	Manchester City - Tottenham	1.42	5.82	8.23

Figure 9: Home page of the user. Here you can mint entryTokens or bet.



Get NFTs

(LOOK)MAN
Owner: Creator
5 Super Tokens

MERTENS(CIRO DANCE)
Owner: Creator
30 Super Tokens

CR7(01)
Owner: Creator
100 Super Tokens

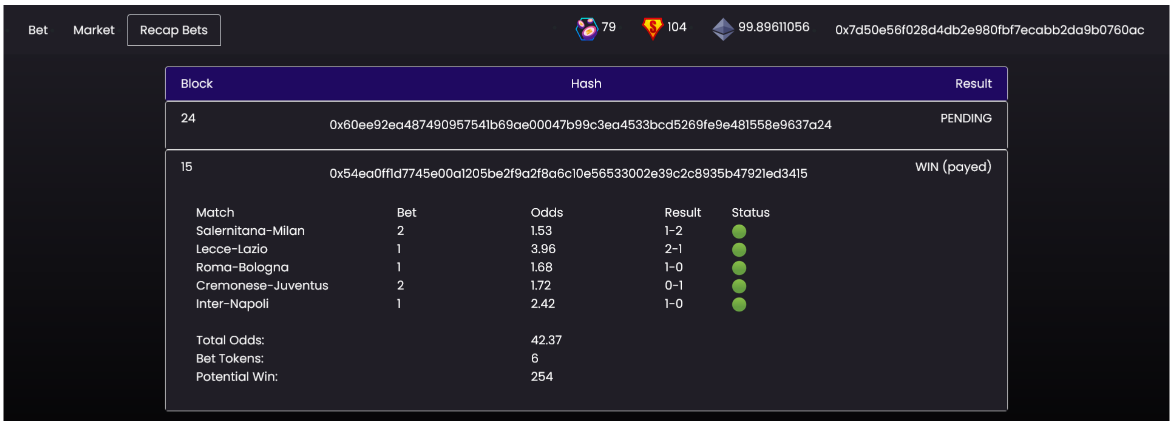
Buy NFTs

CANTONA
Owner: 0x7d50e56F028D4db2e980fBf7ECABB2Da9b0760aC
30 ETH

My NFTs

DINHO
Owner: You
5 ETH

Figure 10: This page is the NFT market. Here you can become the first owner of a NFT if you acquire it with superTokens. Otherwise, you can acquire them with ETH from other user (or sell yours).

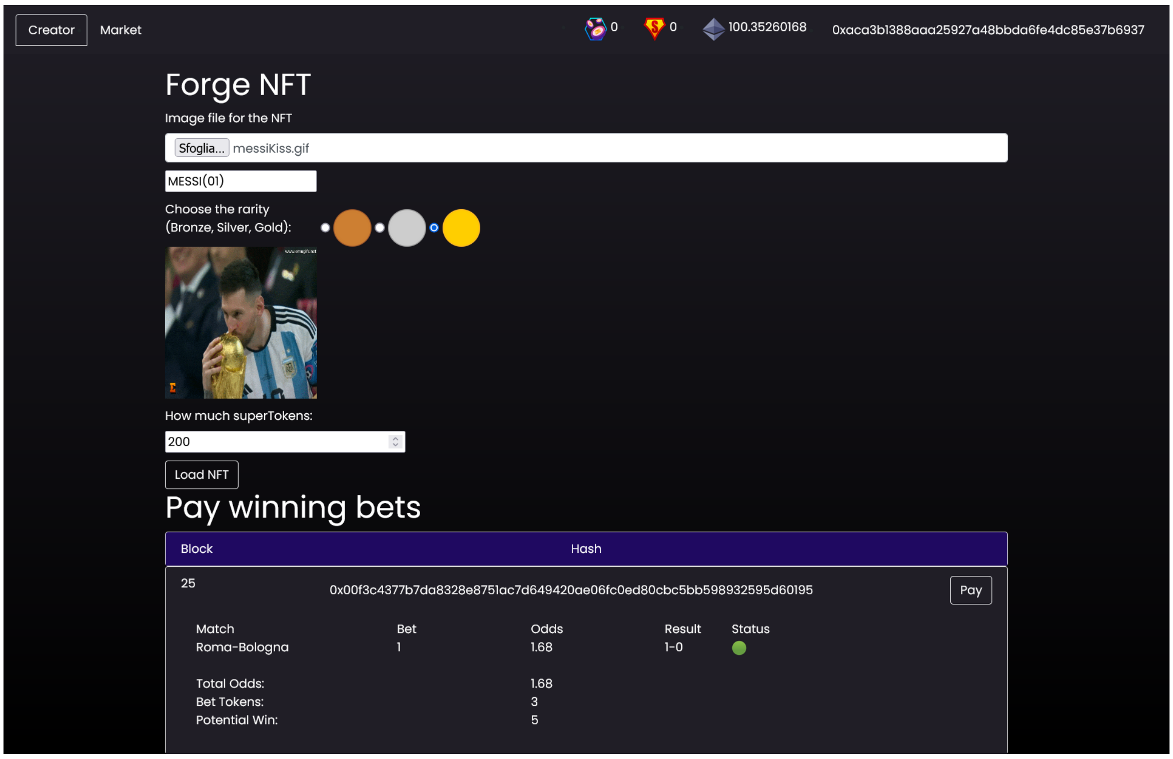


Block	Hash	Result
24	0x60ee92ea487490957541b69ae00047b99c3ea4533bcd5269fe9e481558e9637a24	PENDING
15	0x54ea0ff1d7745e00a1205be2f9a2f8a6c10e56533002e39c2c8935b47921ed3415	WIN (payed)

Match	Bet	Odds	Result	Status
Salernitana-Milan	2	1.53	1-2	●
Lecce-Lazio	1	3.96	2-1	●
Roma-Bologna	1	1.68	1-0	●
Cremonese-Juventus	2	1.72	0-1	●
Inter-Napoli	1	2.42	1-0	●

Total Odds:	42.37
Bet Tokens:	6
Potential Win:	254

Figure 11: Recap bets page. Here you can see if a bet is pending, a winning one or a losing one.



Forge NFT

Image file for the NFT: messiKiss.gif

MESSI(01)

Choose the rarity (Bronze, Silver, Gold):

How much superTokens: 200

Pay winning bets

Block	Hash
25	0x00f3c4377b7da8328e8751ac7d649420ae06fc0ed80cbc5bb598932595d60195

Match	Bet	Odds	Result	Status
Roma-Bologna	1	1.68	1-0	●

Total Odds:	1.68
Bet Tokens:	3
Potential Win:	5

Figure 12: This page is the creator page. Here the creator can forge new NFTs and set things like rarity, name, image, how much superTokens it costs. Also in this page the creator pays the winning bets.

5.2 Contract

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.0 <0.9.0;

contract DistributedBetting
{

    // the price of an entry token
    uint constant public PRICE = 1 * 1e15;

    // the percentage of the creator's fee
    uint constant public FEE = 10;

    // the creator of the contract
    address payable internal creator;

    // these tokens can be bought with ether and can be used to bet
    mapping(address => uint) public entryTokens;

    // these tokens can be used to buy NFT and to bet
    // (they can only be obtained winning bettings)
    mapping(address => uint) public superTokens;

    // Pay4win event
    event BetPayed(address winner, bytes32 bet, uint amount);

    // The fields of the NFT
    struct NFT
    {
        bytes32 imageHash; // the hash of the image of the card
        bool selling; // whether the owner is selling the card
        uint superTokenPrice; // the price in super tokens
        uint ethPrice; // the price in ether
        address owner; // the owner of the NFT
        bytes32 name; // the name of the NFT
        bytes1 rarity; // the rarity of the NFT
    }

    // stores all the NFTs using the hash of the image as a key
    mapping(bytes32 => NFT) public NFTs;

    constructor()
    {
        creator = payable(msg.sender);
    }

    // this function is used to mint entryTokens
    function mint() external payable
    {
        // you need to send at least PRICE eth to buy entryTokens
        require(msg.value >= PRICE, "You're a jerk!");

        // "buy" entryTokens
        entryTokens[msg.sender] += msg.value / PRICE;

        // give the money to the creator
        creator.transfer(msg.value);
    }

    // this function is used to bet on the blockchain
    function bet(bytes32 stake, uint amount) external
    {
        // you need to have enough tokens to bet
        require(entryTokens[msg.sender] + superTokens[msg.sender] >= amount,"You're a jerk, again!");
        // remove the entryTokens from the user account
        if (entryTokens[msg.sender] >= amount)
            entryTokens[msg.sender] -= amount;
        else // if the entryTokens are not enough, for the rest remove the superTokens
        {
            uint remaining = amount - entryTokens[msg.sender];
            entryTokens[msg.sender] = 0;
            superTokens[msg.sender] -= remaining;
        }
    }

    // this function creates a new NFT
    function forge_NFT(bytes32 NFT_hash, uint superTokenPrice, bytes32 NFT_desc_hash, bytes1 NFT_rarity) external
    {
        require(msg.sender == creator, "You are not the contract owner");
        require(NFTs[NFT_hash].imageHash == 0, "The NFT already exists");
        NFTs[NFT_hash] = NFT(NFT_hash, false, superTokenPrice,0,creator,NFT_desc_hash,NFT_rarity);
    }

    // this function is used to get an NFT from the creator (paying in superTokens)
    function get_NFT(bytes32 NFT_hash, uint ethPrice) external
    {
        // the NFT has never been got by anyone (with superTokens)
        require(NFTs[NFT_hash].owner == creator,"the NFT has never been got by anyone (with superTokens)");
        // the sender has enough super tokens to get the NFT
        require(superTokens[msg.sender] >= NFTs[NFT_hash].superTokenPrice,
        "the sender has not enough super tokens to get the NFT");
        // decrease the super tokens from the sender account
        superTokens[msg.sender] -= NFTs[NFT_hash].superTokenPrice;
        // make the sender the new owner of the NFT
        NFTs[NFT_hash].owner = msg.sender;
        // if the ethPrice is greater than 0, start selling the NFT for ether
        NFTs[NFT_hash].selling = (ethPrice != 0);
        if (NFTs[NFT_hash].selling)
            NFTs[NFT_hash].ethPrice = ethPrice;
    }

    // this function is used to put your NFT on the market (selling it for ethPrice ether)
    function sell_NFT(bytes32 NFT_hash, uint ethPrice) external
    {
        // you need to be the owner of the NFT
        require(msg.sender == NFTs[NFT_hash].owner,"you need to be the owner of the NFT");
        // set the price to what you decided and start selling it
        NFTs[NFT_hash].ethPrice = ethPrice;
        NFTs[NFT_hash].selling = true;
    }

    // this function is used to stop selling your NFT
    function unsell_NFT(bytes32 NFT_hash) external
    {
        // you need to be the owner of the NFT
        require(msg.sender == NFTs[NFT_hash].owner,"you need to be the owner of the NFT");
        // stop selling the NFT
        NFTs[NFT_hash].selling = false;
    }

    // this function is used to buy an NFT from another user (in ether)
    function buy_NFT(bytes32 NFT_hash) payable external
    {
        // the NFT is not being sold
        require(NFTs[NFT_hash].selling,"the NFT is not being sold");
        // you need enough ether to buy the NFT
        require(msg.value >= NFTs[NFT_hash].ethPrice,"you need enough ether to buy the NFT");
        // take the fee
        uint fee = msg.value * FEE / 100;
        creator.transfer(fee);
        // send the rest of the money to the old owner
        payable(NFTs[NFT_hash].owner).transfer(msg.value - fee);
        // the sender is now the new owner
        NFTs[NFT_hash].owner = msg.sender;
        // stop selling the NFT
        NFTs[NFT_hash].selling = false;
    }

    // this function (called only by the creator) pays a user who won a bet
    function pay4win(bytes32 stake, address winner, uint amount) external
    {
        // you need to be the contract owner
        require(msg.sender == creator, "You are not the contract owner");
        // pay the winner of the bet whose hash is stake
        superTokens[winner] += amount;
        // notify that the bet has been payed
        emit BetPayed(winner, stake, amount);
    }

    //selfdestruct the contract
    function kill() external
    {
        require(msg.sender == creator, "You are not the contract owner");
        selfdestruct(creator);
    }

}
```

5.3 Server

Our Flask server is not a simple web server to store pages and files and retrieve them to the user; Indeed, it also does two very interesting things:

1. It performs scraping (using the Selenium library) to obtain match results and odds; it basically automatically visits the `www.diretta.it` website and it reads the html of the page to get the informations it needs.

2. With the web3 python library, the server inspects the blockchain transactions to retrieve the existing NFTs and bets. Through this inspection, it can also be checked if a bet has already been payed (to avoid paying twice the same bet). Here there is the code snippet of the function that examines the transaction:

```python
def getTxData(start: int, end: int, function_selector: str, address: str = None) -> List[str]:
'''
Given a start and an end block, it returns a list of hashes
of the transactions that are calls to the function with the given selector.
If the address is not None, it returns only the hashes of the transactions
that are calls to the function with the given selector and are sent by the given address.
'''
hashes = []
# iterate through the blocks
for x in range(start, end+1):
    # get xth block
    block = w3.eth.getBlock(x, True)
    # iterate through the transactions of the block
    for transaction in block.transactions:
        # if the transaction is a call to the function of the contract
        if transaction['input'][:10] == function_selector
        and transaction['to'].lower() == config.CONTRACT_ADDRESS.lower():
            # if the address is None, add the hash to the list regardless of the sender
            if address == None:
                hash = "0x"+transaction['input'][10:74]
                hashes.append([hash, transaction['from'].lower(), x])
            # otherwise, check if the sender is the given address
            elif transaction['from'] is not None and transaction['from'].lower() == address.lower():
                # if the transaction is a call to the function
                hash = "0x"+transaction['input'][10:74]
                hashes.append(hash)
return hashes

```

6 Conclusions

In this project, we implemented a Dapp that allows user to bet on football events on the blockchain. We believe that this application could be more profitable for the users (they play with fair odds) but also for the owner of the contract (that gets a fee for every NFT sold).

6.1 Issues & Limitations

We found two types of issues & limitations:

1. Scraping limitation, meaning that if the site where we get the match results and odds crashes or gets hacked, we lose these informations or they are compromised. And also there is a time issue where the user has to wait the end of the scraping to get the latest matches. The first issue could potentially be limited by scraping odds and results from more than one website (to avoid the single point of failure).

2. Blockchain limitation, because we don't have an Oracle that pay the winning bets, but the creator does this. When the creator pays users it loses every time ether as transaction fee. There is also an age issue meaning that we do not know if the user has legal age.

6.2 Future Works

There are some future implementations that we have thought about:

- Pack Opening: a new page with the possibility to get NFTs packs where the user can obtain random NFTs with some probability proportional to the rarity. In this way the rarity is not more a graphical or psychological aspect but has importance.

- Distributed Results: as we said before, the match results could be retrieved from more than one site.

- Scraping Alternatives: we did not find a good alternative but we want to find something that works faster, for example websites APIs(but it's not free).

- Unfair Payment System: we should find an alternative to the fact that the creator has to pay transaction fees for each winning bet.

References

[1] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview, 2018-10-03 2018.

[2] Blockchain: A brief history of blockchain technology that everyone should read. https://kriptomat.io/blockchain/history-of-blockchain/. Accessed: 2023-01-10.

[3] History of bitcoin: How the blockchain ecosystem evolved. https://fourweekmba.com/history-of-bitcoin/. Accessed: 2023-01-10.

[4] Rpow - reusable proofs of work. https://nakamotoinstitute.org/finney/rpow/. Accessed: 2023-01-10.

[5] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009.

[6] Ethereum whitepaper. https://ethereum.org/en/whitepaper/#ethereum. Accessed: 2023-01-10.

[7] How and why to calculate bookmaker's margin? https://www.bettingwell.com/sports-betting-guide/maths-sports-betting/how-and-why-calculate-bookmakers-margin. Accessed: 2023-01-10.

[8] Permissionless vs. permissioned blockchains: Pros cons. https://www.1kosmos.com/blockchain/permissionless/. Accessed: 2023-01-10.

[9] K. Wüst and Arthur Gervais. Do you need a blockchain? *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54, 2018.