People's Democratic Republic of Algeria

Abou Bekr Belkaid University - Tlemcen

Faculty of Sciences

Department of Computer Science

Final study thesis for obtaining a Master's degree in Computer Science

Option: Software Engineering

Subject

Viral Campaign Blockchain-Powered Ads: Design & Implementation

Prepared by:

Baba Bendermel Houssam Eddine

Presented on June 25, 2024 before the jury composed of

Prof. Azeddine CHIKH	President (University of Tlemcen)
Dr. Amine BELHOCINE	Examiner (University of Tlemcen)
Dr. Salim ZIANI CHERIF	Supervisor (University of Tlemcen)
Mrs. Asma Ismail	Guest (Punchword)

Academic Year: 2023 - 2024

بسم الله الرحمان الرحيم

"وَأَن لَّيْسَ لِلإِنسَانِ إِلاَّ مَا سَعَى ۝ وَأَنَّ سَعْيَهُ سَوْفَ يُرى "

[النجم:38-41]

First of all, I am thankful to God Almighty for granting me the health, strength, and patience to complete my graduation project. I am profoundly grateful to Almighty Allah for guiding me through this journey.

Acknowledgments

This work was made possible thanks to Punchword and the opportunity they provided me to be part of the team. I am grateful for the chance, experience, and their trust.

I would like to express my deep gratitude and warmest thanks to my supervisor Mr. Ziani-Cherif Salim, for his guidance, and advice, and for the time he devoted to me during the development of this work

Additionally, I would like to express my sincere gratitude to our CTO, Asma Smail, for all the support, guidance, and knowledge that I have gained from her mentorship.

I am honored by the presence of Prof. Azeddine CHIKH Professor, who gave me the great honor of accepting the presidency of my dissertation jury

I also thank Dr. Amine BELHOCINE for agreeing to examine this work and being part of the Jury.

Dedication

Glory to my dear parents for their patience and support during these years of study, and to my dear brothers and my sister for their encouragement.

Thanks to my friends who stood beside me, and to those with whom I spent countless hours.

Finally, to anyone who has contributed to this memoir directly or indirectly, to all, I say

THANK YOU…

Table of Contents

Table of Figures

List of Tables

I. General Introduction

CONTENT

I.1. Project Context

This master's thesis explores an innovative application of blockchain technology within the context of a social media platform called Punchword. Developed by the company Punchword, where I currently work as an Android developer, this platform merges the classic features of social media with the unique capabilities of blockchain. Punchword is not just another social media platform; it includes an NFT marketplace and leverages blockchain to enhance user engagement and digital content ownership.

The focus of this thesis is on introducing and evaluating a novel feature within Punchword, termed "Punchword Viral Campaigns." This feature combines the creation of advertisements with Non-Fungible Tokens (NFTs), leveraging blockchain to facilitate transparent and secure viral marketing campaigns. Companies can launch a Viral Campaign by creating challenges on the Punchword platform, prompting users to submit their ad designs as NFTs. These challenges foster a competitive environment where users can earn rewards in the platform's native cryptocurrency, Punchy. The "Punchword Viral Campaigns" feature acts like an Influencer's Marketplace, enabling advertisers to set challenge rules through smart contracts, which automatically enforce reward distribution based on predefined criteria. Influencers participate by creating and minting NFTs, with winners selected based on engagement metrics such as likes, comments, and shares. This integration ensures a transparent, secure, and engaging platform for both advertisers and influencers, leveraging blockchain's capabilities to revolutionize digital marketing.

I.2. Thesis Organization

This thesis is organized into several chapters. The first chapter, Blockchain Basics, lays the foundation by introducing blockchain, defining its key concepts, and tracing its historical development. It also contrasts blockchain with traditional databases, explaining the distinct advantages and functionalities of each. This chapter further delves into the mechanics of how blockchain operates, covering essential components such as nodes, blocks, transactions, wallets (including public and private keys), smart contracts, and tokens, with specific attention to ERC-20 and ERC-721 standards. This foundational knowledge sets the stage for the detailed exploration of the project.

The second chapter focuses on the Project Conception and Design. It will elaborate on the development and integration of the "Punchword Viral Campaigns" feature within the Punchword platform. This chapter includes detailed diagrams and models illustrating the system architecture, workflow, and interactions between various components.

In the third chapter, "Implementation of the Punchword System," we first present the various tools and technologies chosen for realizing our system with the desired outcomes (Android, Solidity, etc.). We then describe the overall architecture used to develop the app (MVVM), and also how to

connect the mobile app with the blockchain smart contract using the web3j library, and how that works. Finally, we show screenshots that demonstrate the user experience of each part of the "Viral campaigns.".

II. Blockchain Basics

CONTENT

II.1. Introduction

Blockchain technology has emerged as a revolutionary force in the world of digital transactions and data management. Originally conceptualized as the underlying architecture for Bitcoin, blockchain has since evolved far beyond its initial use-case.

This chapter aims to provide a comprehensive overview of blockchain technology, setting the stage for a deeper exploration of its components, functionalities, and implications.

II.2. Definition

Blockchain is a distributed network technology that securely records transactions across a network of computers, making it difficult to alter, hack, or manipulate the system. It's most commonly associated with Bitcoin, a popular cryptocurrency that uses blockchain technology, for data storage, validation, and security. It consists of a series of blocks, each containing transaction data, a timestamp, and a cryptographic hash of the previous block, forming a chain of data. Once a transaction is recorded on a block, it cannot be retroactively changed without altering all subsequent blocks, ensuring the immutability of the data. [1] [2]

Key characteristics of blockchain include decentralization, transparency, and cryptographic security. Participants in the network collectively maintain and validate the blockchain, eliminating the need for a central authority. Transactions are added to the blockchain through a consensus mechanism, where network participants agree on the validity of new blocks.[1] [2]

In summary, blockchain is a revolutionary technology that enables secure, transparent, and decentralized recording and verification of digital transactions, offering a new level of trust and efficiency in various sectors.

II.3. Brief history of blockchain Development

Blockchain technology has a rich history that spans several decades. The concept of blockchain was first described in 1991 by research scientists Stuart Haber and W. Scott Stornetta, who aimed to introduce a computationally practical method for timestamping digital documents. However, the blockchain as we know it today was launched in January 2009 along with the associated cryptocurrency, Bitcoin, by the anonymous person or group known as Satoshi Nakamoto. [3]

Before Nakamoto's work, there were decentralized databases and blockchain-like systems. For instance, David Chaum, a doctoral candidate at the University of California at Berkeley, outlined a blockchain database in 1982. Chaum's work was not specifically designed to support digital currencies, but it provided the groundwork for future advancements. [3]

Nakamoto's innovation was the addition of the Bitcoin proof-of-work consensus mechanism for validating data blocks, which was outlined in the Nakamoto research paper. This mechanism allowed for the creation of a decentralized, peer-to-peer electronic cash system. [4]

The early years of blockchain were closely tied to the development of Bitcoin, with the two terms often being used interchangeably. However, as the technology evolved, blockchain began to separate from Bitcoin, and its potential uses expanded beyond digital currency. In 2014, Ethereum was established, introducing the concept of smart contracts, which will be explained in detail later [II.6.5], and further expanding the possibilities of blockchain technology. [3]

Today, blockchain is known for its security, immutability, traceability, and transparency, making it a viable alternative to traditional methods of conducting business and individual transactions. Its applications have grown, and more applications using blockchain technology are entering the public domain. Examples include the Brave browser and Ethlance, a freelancing platform.

II.4. Difference between blockchain and traditional database

A blockchain differs significantly from a traditional database in several key aspects. Blockchains are decentralized, meaning no single authority controls the data; instead, network participants collectively maintain and validate the blockchain. In contrast, traditional databases are centralized, with a designated authority controlling access and modifications to the data. One of the features of a blockchain is its immutability: once data is added to the chain, it cannot be altered or deleted. Conversely, data in traditional databases can be modified or deleted by authorized parties, such as the database administrator. Transparency is another distinguishing feature; blockchains are highly transparent, with all transactions visible to network participants, whereas traditional databases have lower transparency, with access and visibility controlled by the central authority. Data integrity is inherently strong in blockchains due to their cryptographic underpinnings, ensuring that data remains secure and unaltered once recorded. In traditional databases, while data integrity can also be high, it relies heavily on access controls and management policies to maintain security. [5]

However, blockchains tend to have slower transaction speeds due to the consensus mechanism required to add new blocks, whereas traditional databases are faster as they don't require consensus for every transaction. When it comes to cost, blockchains often incur higher costs, particularly due to the significant energy consumption required for mining and maintaining the network. This energy consumption is a notable problem, raising concerns about the environmental impact of blockchain technology. Cryptography plays a crucial role in both systems, but it is fundamental to blockchain, providing security and trust without a central authority. [6]

That comparison would help us better understand the upcoming points that we'll discuss later in the thesis.

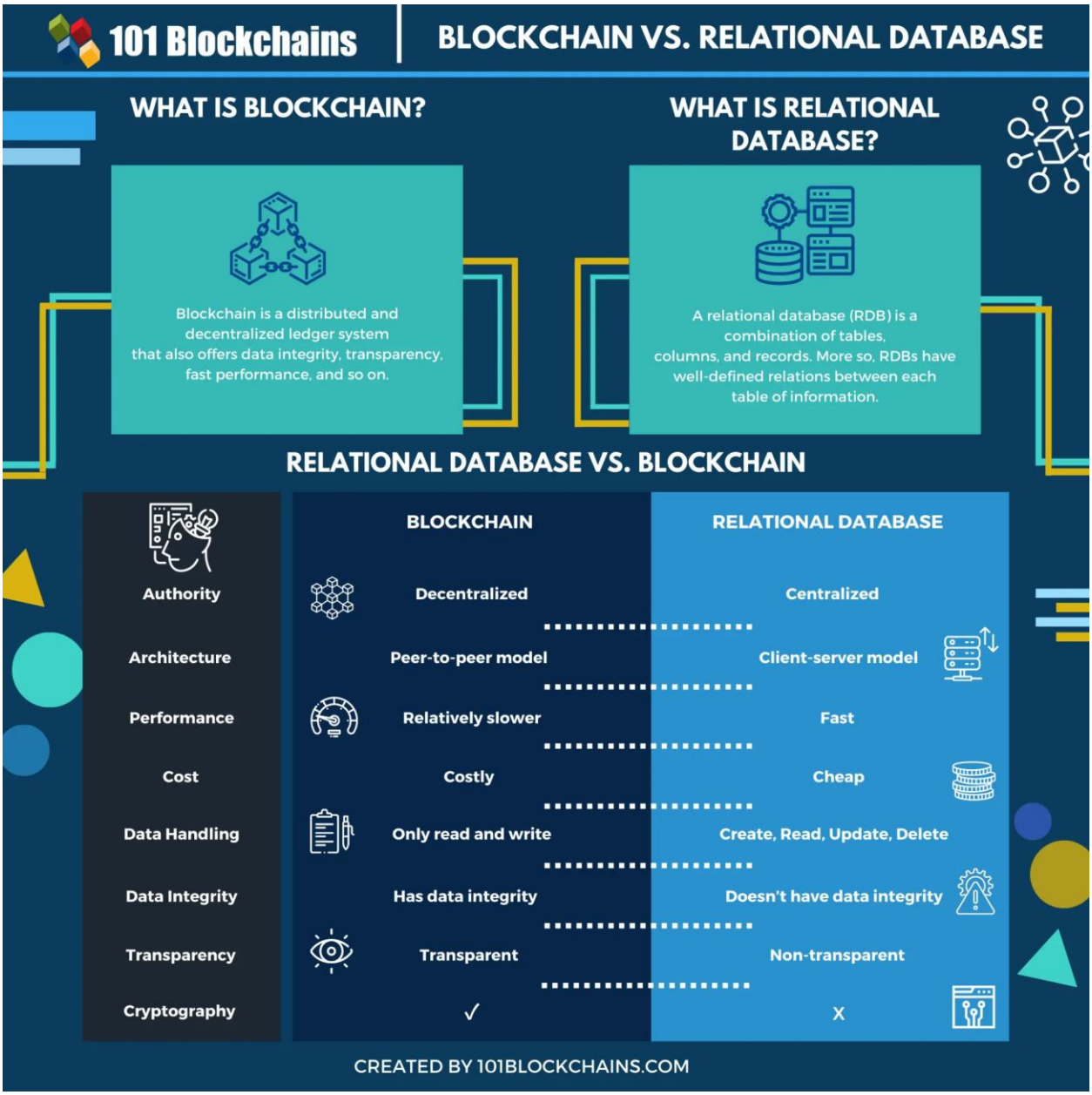

FIGURE II.1 - Comparison between a blockchain and a traditional database [6]

II.5. How Blockchain Works

Blockchain is a decentralized, distributed digital ledger that records transactions across many computers in a network. When a user or node initiates a new transaction, such as a cryptocurrency transfer or execution of a smart contract, the transaction request is broadcasted to all nodes in the peer-to-peer network. These nodes, often called miners or validators, perform checks to verify that the transaction is valid according to the blockchain's predetermined rules and consensus protocol. This validation process prevents invalid transactions from being added to the blockchain. [7]

Once validated, the transaction is grouped together with other valid transactions into a new block of data. This block also contains a reference to the previous block in the chain, known as the "previous block hash," creating an immutable link between them. To add this new block to the existing blockchain, the nodes must reach consensus on its validity through a consensus mechanism like Proof-of-Work (PoW) or Proof-of-Stake (PoS). [7]

In PoW, used by Bitcoin, miners/validators compete to solve a complex cryptographic puzzle, with the winner getting to add a new block and earn rewards. In PoS, used by Ethereum implementations, validators stake/lock/hold their coins. Unlike Proof-of-Work, which requires solving complex puzzles, PoS relies on validators' stakes to decide who gets to add the next block. The more coins a validator stakes, the higher their chances of being selected. [8]

This method (POS) is better, especially when it comes to energy efficiency, because not all nodes will be working and only one will get the reward, but rather one will be selected and will get it reward. This reward is also called gas fees, and they are calculated based on the computational effort required and the user's willingness to pay. These fees help regulate network activity and incentivize validators to maintain the blockchain. [8]

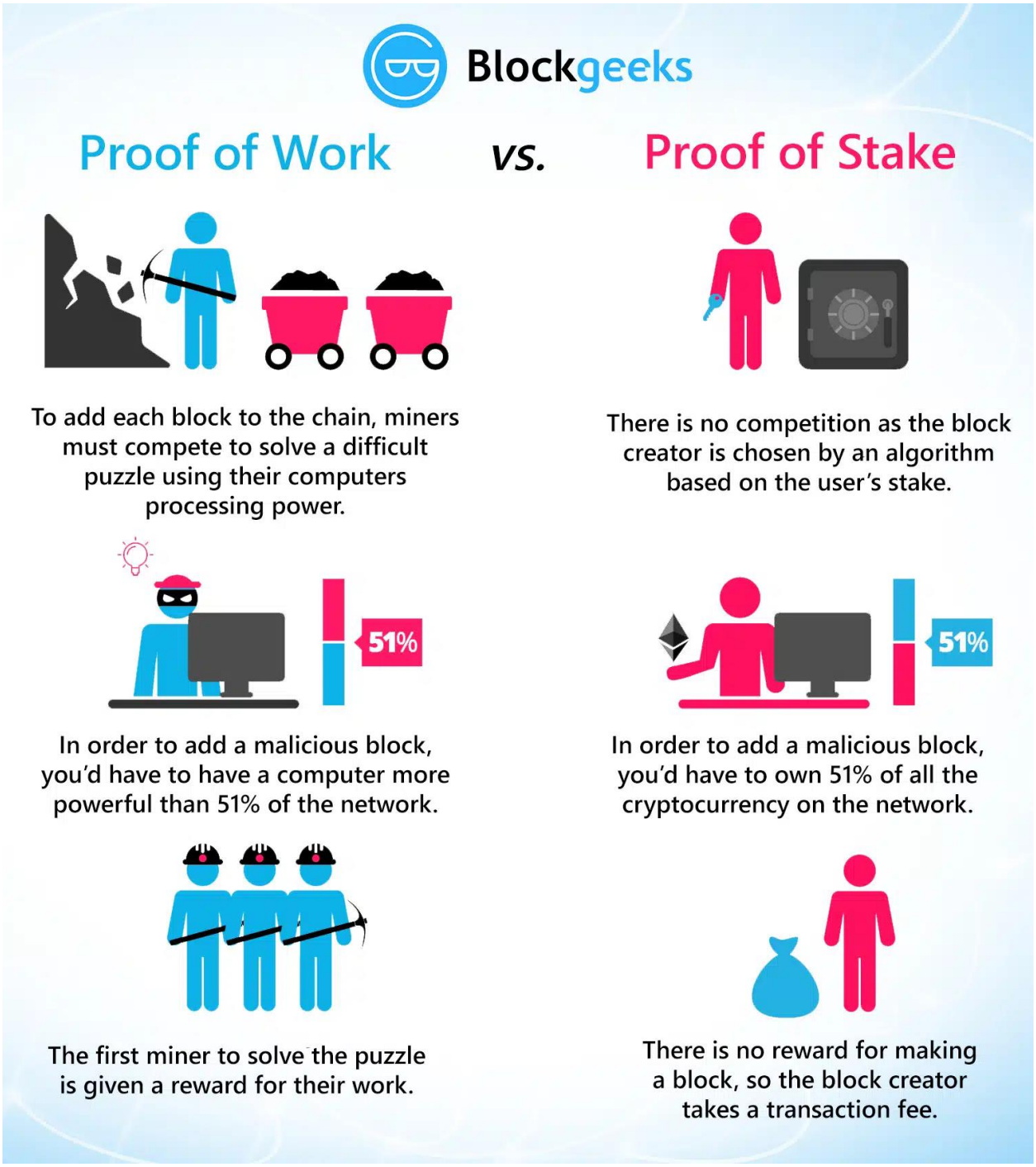

FIGURE II.2 - Comparison between a POS and POW [8]

After consensus is reached, the new validated block is appended to the existing blockchain across all nodes in an irreversible and immutable manner. The block's data cannot be altered retroactively without redoing the work for all subsequent blocks and gaining consensus from the network again. The updated blockchain, with the new block added, is then propagated and synchronized across

all nodes in the network, ensuring each node maintains a full copy of the blockchain, creating a decentralized, distributed ledger. [7]

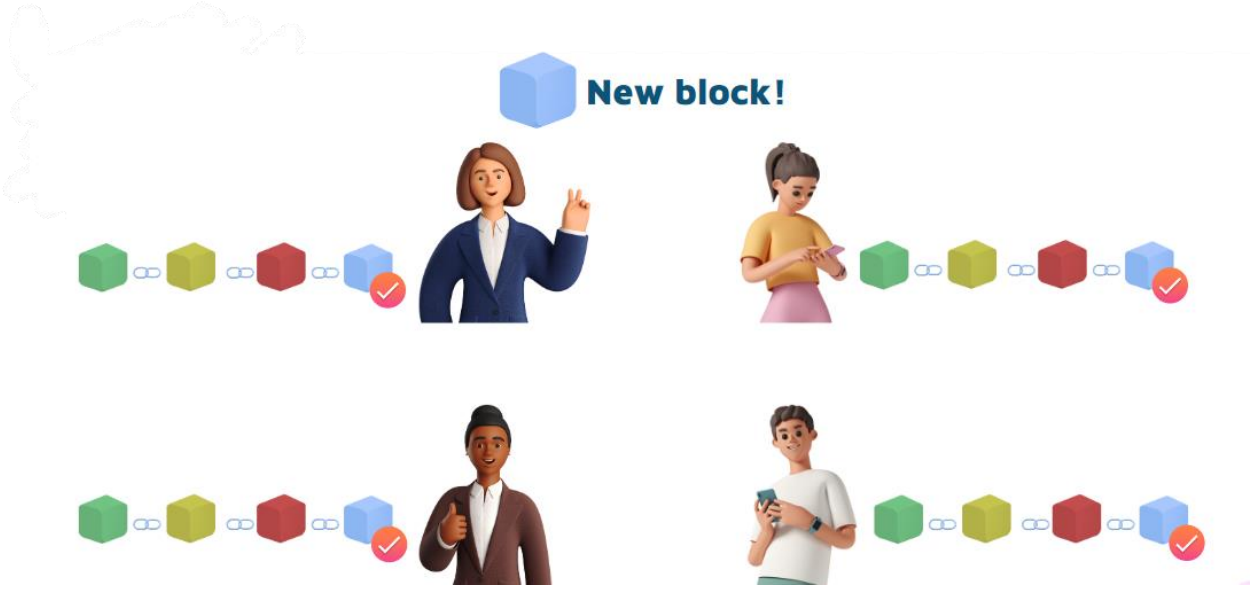

FIGURE II.3 - Add a new block to the blockchain

The transparency and immutability of the blockchain are key security features. Any node or user can independently verify the validity of transactions and the state of the blockchain by following the chain of cryptographic hashes from the latest block all the way back to the first (Genesis) block.

A cryptographic hash serves as a unique identifier for each block in a blockchain. It's like an "ID" or a "fingerprint" of the block's data. This hash is generated using a cryptographic hash function, which takes the block's data as input and produces a fixed-size string of characters as output. [10]

This combination of decentralization, cryptographic hashing, consensus mechanisms, and an immutable ledger allows blockchain networks to operate in a trustless manner without a central authority, while maintaining data integrity and transparency across the distributed system.

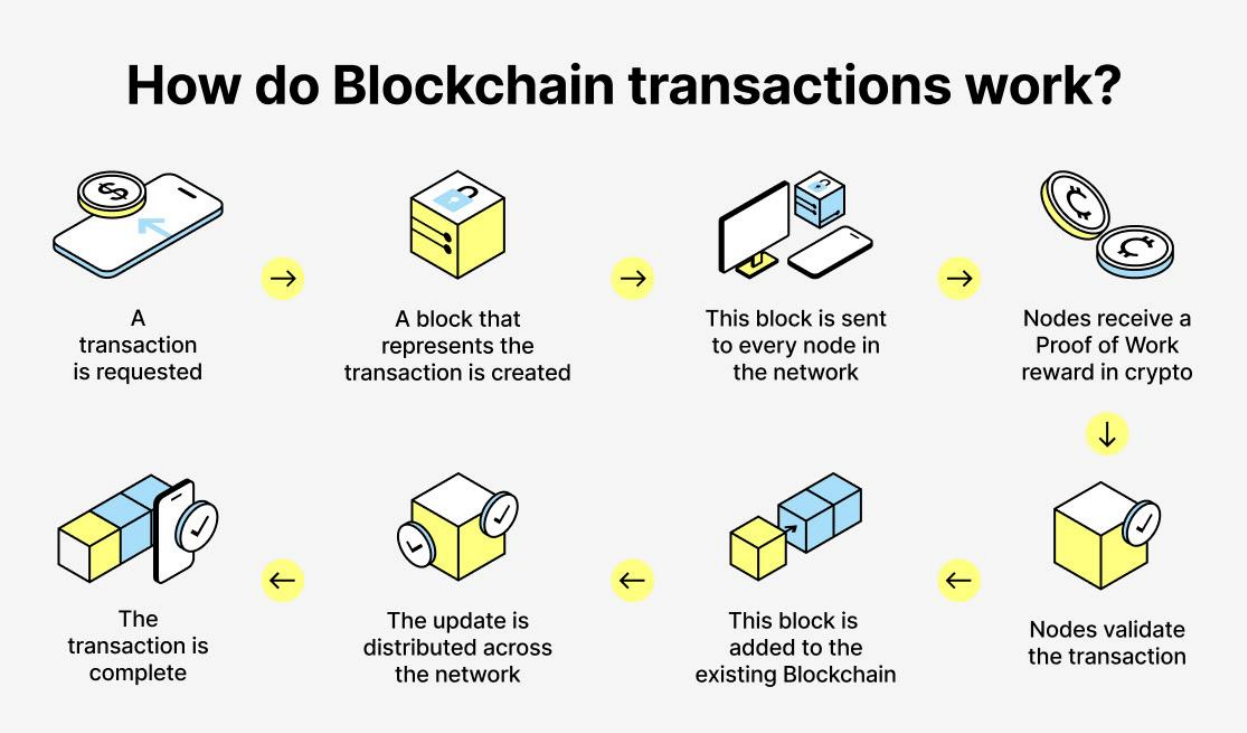

FIGURE II.4 - How do blockchain transactions work? [9]

II.6. Components of Blockchain

In the evolving landscape of digital technology, blockchain operates through a complex interplay of various components, each playing a crucial role in maintaining its integrity and functionality. Understanding these components is essential to grasping how blockchain technology works and the diverse applications it supports.

II.6.1. Nodes

Nodes are individual computers that participate in the blockchain network, each playing various roles in maintaining and securing the system and performing functions various, and they can be categorized into 4 main types: [11]

- Full nodes store the entire blockchain, verify all transactions and blocks, and ensure the network's security and integrity.
- lightweight nodes store only a subset of the blockchain, typically just the block headers, and rely on full nodes for transaction validation.
- Mining nodes participate in the creation of new blocks by solving complex cryptographic puzzles (proof of work) or validating transactions (proof of stake).
- validator nodes, specific to proof-of-stake networks, validate transactions and propose new blocks.

In summary, nodes validate transactions and blocks, maintain the blockchain, and participate in consensus mechanisms to add new blocks.

II.6.2. Blocks

Blocks, on the other hand, are the building units of the blockchain, containing a set of verified transactions and forming the immutable ledger (a blockchain).

Each block contains The data stored inside the block depending on the type of the blockchain. The bitcoin's blockchain for example stores the details about the transaction here such as the sender, the receiver, and the amount of coins. A block also has a hash man can think of it like a fingerprint. It identifies a block and all of its contents and it's always unique just like a fingerprint. The third element inside each block is the hash of the previous block. This effectively creates a chain of blocks. And it's this technique that makes a blockchain so secure. [12]

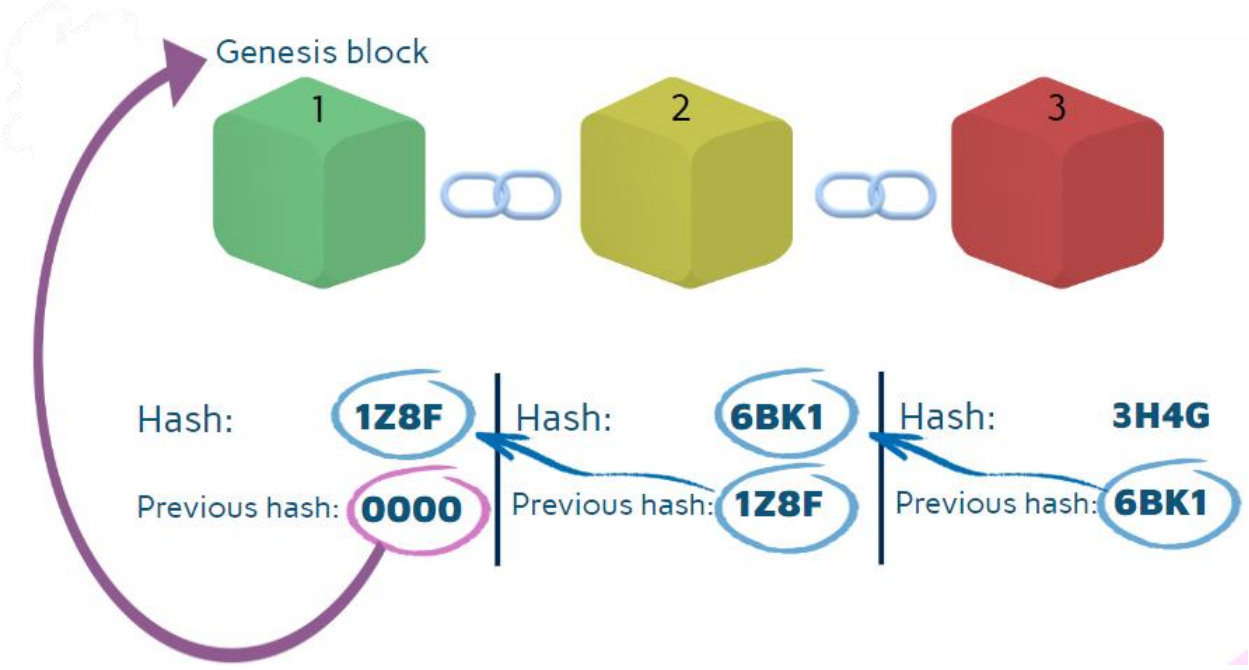

FIGURE II.5 - Blocks hashes in the Blockchain

II.6.3. Transactions

Transactions represent the fundamental interactions within a blockchain network, such as transferring digital assets, recording data, or executing smart contracts.

Each transaction contains information like the sender's address, recipient's address, amount, timestamp, and a digital signature for authentication. [13]

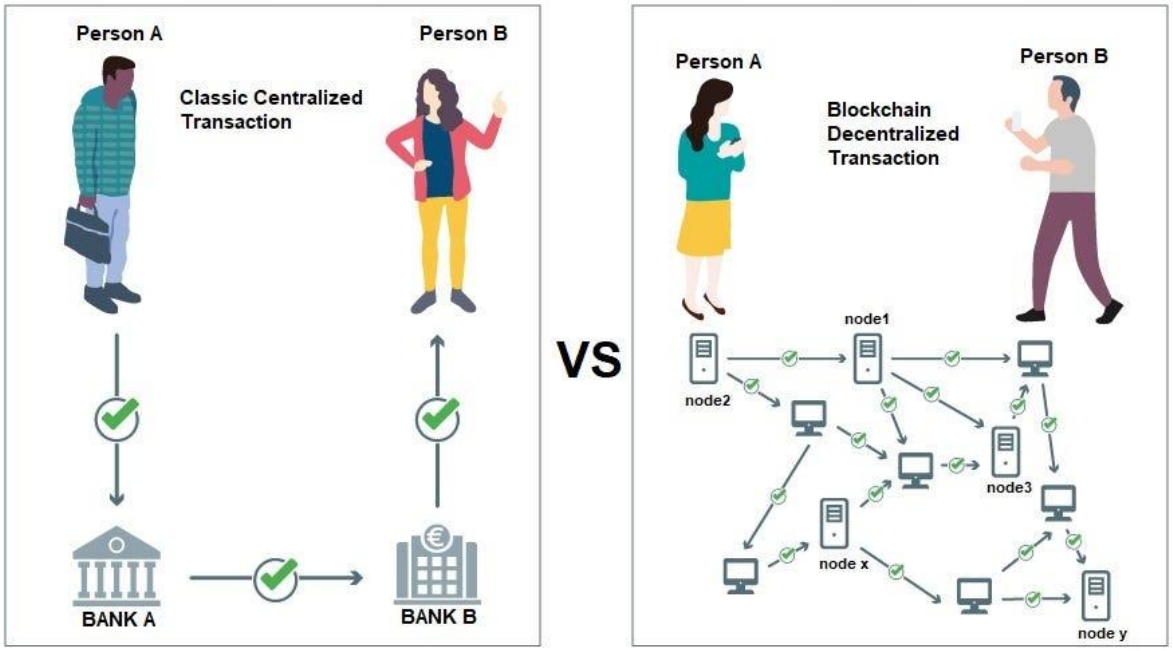

FIGURE II.6 - The difference between a normal transaction and a transaction in the blockchain

II.6.4. Wallets

Wallets are used to store cryptocurrency and interact with the blockchain network.
They use public and private key pairs for security and privacy. [14]

II.6.4.1. public keys

A public key is a cryptographic code that allows users to receive cryptocurrency transactions. It is shareable and acts as an address for sending funds to a wallet.

Public keys are used to receive transactions and are visible to all users in the network. They are like an account number that uniquely identifies a wallet.

Public keys can be freely shared without compromising security, as they only allow others to send funds to the wallet. [15]

II.6.4.2. Private keys

A private key is a secret code that grants ownership and control over the funds associated with a public address. It should be kept confidential at all times.
Private keys are used to prove ownership and spend cryptocurrency funds. They are crucial for signing transactions and accessing wallet balances securely.

Private keys should never be shared with anyone, as they provide total control over the associated funds. Losing a private key can result in the loss of access to the wallet and its contents. [15]

Criteria	Public Key	Private Key
Ownership	Publicly owned	Privately owned
Accessibility	Open to everyone	Restricted access
Security	Impossible to lose as it is in the public domain	Easy to lose if carelessly stored
Low risk if lost, as they are meant to be shared.	Low risk if lost, as they are meant to be shared.	High risk if lost, as it results in the loss of access to the wallet and its contents.
usage	Used to receive transactions and uniquely identify a wallet.	Used to sign transactions and access wallet balances securely.

TABLE II.1 - Comparison between private and public keys

Smart wallets came in 2 types mainly, hardware and software wallets:

II.6.4.3. Hardware Wallet

Hardware wallets, also known as cold wallets, provide offline storage for your cryptocurrency keys. They often resemble USB devices, although paper-based versions also exist, where the public and private keys are printed on paper. These wallets are highly regarded by investors for their security, as being offline minimizes hacking risks. However, they are not without their own risks, such as physical loss or theft.

You may ask, "If it's offline, how would a person be able to access the blockchain?" When we say "offline," we mean that the private key is stored offline inside the hardware. When a user wants to perform a transaction, it typically comes with companion software or applications that allow users to interact with the blockchain and authorize transactions. That software will then use the private key to sign the transaction, and the transaction will not be stored anywhere else online by any means. [14]

II.6.4.4. Software Wallet

Software wallets, also known as hot wallets, work like online bank accounts and are usually linked to a cryptocurrency exchange, offering user-friendly interfaces. Their introduction has played a significant role in making cryptocurrency accessible to the general public. There are different types of software wallets, each with distinct usage methods.

Desktop wallets involve downloading software to your computer. Alternatively, mobile applications are available for accessing your wallet on a smartphone or directly through the web. While software wallets provide convenience and easy access, they are connected to the internet, making them susceptible to hacking and private key theft. This connectivity raises the risk of exposure to cyber threats.[14]

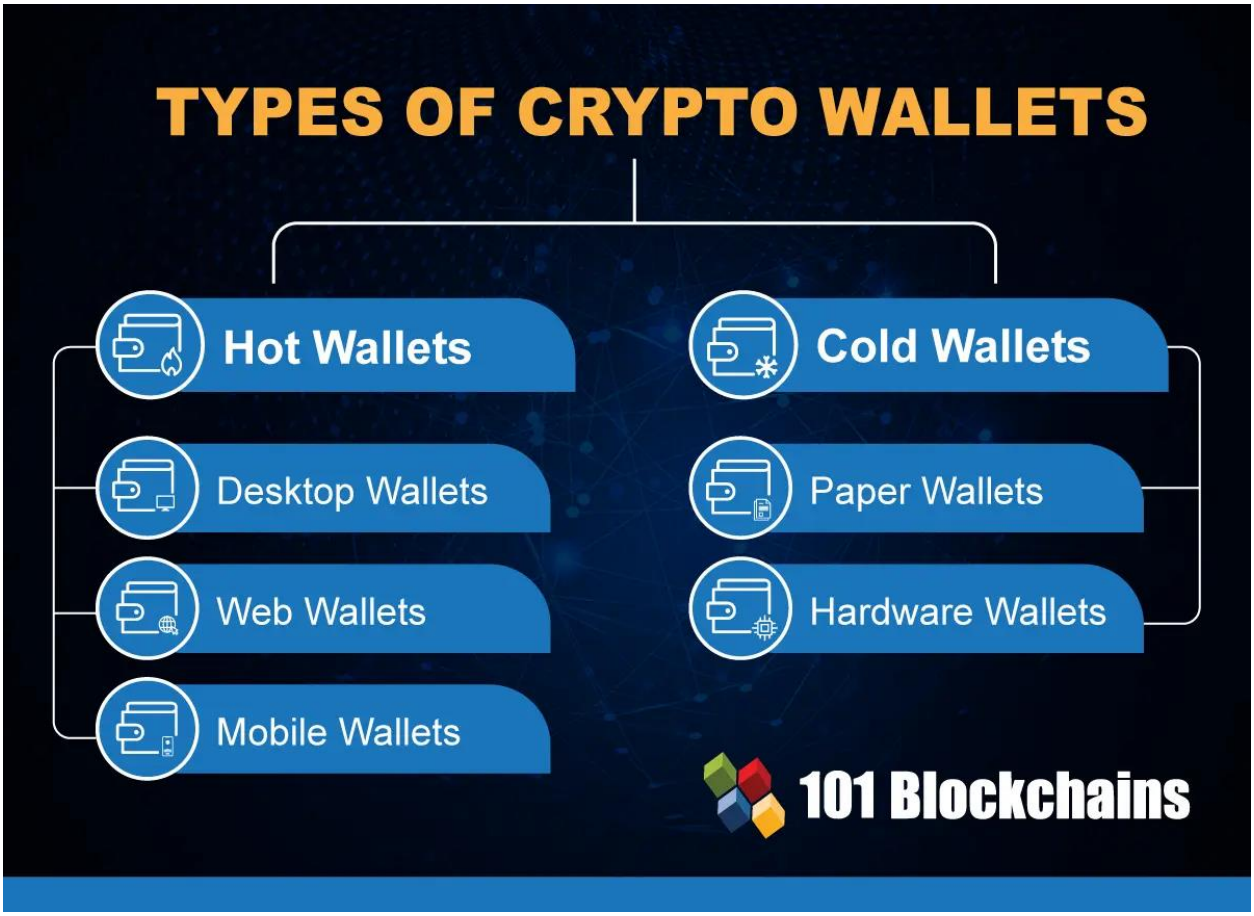

FIGURE II.7 - Smart wallet types [14]

II.6.5. Smart Contracts

Smart contracts are self-executing digital contracts stored on a blockchain that automatically enforce the terms of an agreement between parties. They are computer programs that run when predetermined conditions are met, eliminating the need for intermediaries and manual processing.

Smart contracts are similar to real-world contracts, but they are completely digital. In fact, a smart contract is a small computer program that is stored within a blockchain. Smart contracts can help you exchange money, property, or anything of value in a transparent and conflict-free way, all while avoiding the services of a middleman. [16]

For example, if a renter were to rent an apartment from a landlord, the transaction could be conducted using blockchain technology by paying in cryptocurrency. The renter would receive a receipt, which is stored in a virtual contract. The landlord would then provide a digital entry key by a specified date. If the key doesn't arrive on time, the blockchain releases a refund to the renter. If the landlord sends the key before the rental date, the system holds it, releasing both the payment to the landlord and the key to the renter when the rental date arrives. The system operates on an if-then premise and is verified by hundreds of nodes in the blockchain, ensuring a faultless delivery.

In this scenario, if the landlord gives the key, the landlord is sure to be paid, and if the renter sends a certain amount of cryptocurrency, they receive the key. The document is automatically canceled after the paid-for time has elapsed, and of course, the code cannot be changed because it is on the blockchain.

II.6.6. Token

A token in the context of blockchain and cryptocurrency is a digital asset that represents value or utility on a blockchain network. Created through smart contracts on platforms like Ethereum, tokens have defined properties, functionalities, and distribution rules, such as name, symbol, and total supply. These tokens can be transferred between addresses, facilitating peer-to-peer transactions. They serve various purposes, including payments, access to services, voting rights, and ownership representation. Tokens also interact with decentralized applications (dApps) and other smart contracts, enhancing their utility within the blockchain ecosystem. Notable standards for tokens include ERC-20 and ERC-721, which specify how tokens should function and interact on the Ethereum network. [17]

Before we get into the details of both tokens it’s essential to first explain what an NFT

II.6.6.1. NFTs

An NFT, or non-fungible token, is a type of digital asset that represents ownership or proof of authenticity of a unique item or piece of content. Unlike cryptocurrencies such as Bitcoin or Ethereum, which are fungible and can be exchanged on a one-to-one basis, NFTs are distinct and cannot be exchanged on a like-for-like basis due to their unique attributes. Each NFT contains metadata that distinguishes it from other tokens, including information about the asset it represents, such as its title, creator, and characteristics. This uniqueness and verifiable ownership make NFTs particularly valuable for digital collectibles, art, gaming assets, and other unique creations in the digital realm. [19]

II.6.6.2. ERC-20 Tokens

ERC-20 tokens are fungible tokens that follow a specific standard on the Ethereum blockchain. As mentioned earlier, a token is created as a smart contract, and by following this standard, it adheres

to specific functions and properties. Man can think of it as an interface containing specific variables and functions.
the main and most important variables in that interface are:

- **Name**: The name of the token (e.g., "PunchyToken").
- **Symbol**: The symbol representing the token (e.g., "PNCH").
- **Total Supply**: The total number of tokens that will ever exist.

The most important functions are:

- **BalanceOf**: A function that provides the number of tokens held by a specific address.
- **Transfer**: A function that allows the transfer of tokens from one address to another.
- **TransferFrom**: A function that allows a smart contract to transfer tokens on behalf of the token owner.
- **Approve**: A function that allows a token owner to approve another address to spend tokens on their behalf.
- **Allowance**: A function that provides the number of tokens that an owner allows another address to spend on their behalf.

When you create an ERC-20 token, you inherit from this interface and set specific variables. You can also modify some functions inherited from the interface. This standardization allows different ERC-20 tokens to work together seamlessly since they operate in the same way. ERC-20 tokens are compatible with each other and can be managed by the same wallet, traded on the same exchanges, and interact with the same decentralized applications (dApps) and smart contracts. This interoperability is a key advantage of the ERC-20 standard, enabling a cohesive environment for various digital assets and utilities. They are commonly used for cryptocurrencies, utility tokens, and other fungible digital assets. Each ERC-20 token is interchangeable and equal in value to another token of the same type, much like regular coins that are all the same and can be divided into smaller pieces. These tokens can interact with various applications and smart contracts on the blockchain, enhancing their utility and integration within the Ethereum ecosystem. [18]

II.6.6.3. ERC-721 Tokens

ERC-721 tokens, again it's also standard but unlike their fungible counterpart ERC-20, they are used to represent non-fungible assets on the Ethereum blockchain, adhering to a distinct standard. Each ERC-721 token is unique and indivisible, making it ideal for representing ownership or uniqueness of digital or physical assets such as digital collectibles, real estate, artwork, or in-game items. Before proceeding further.

when creating an ERC-721 token, same thing as the ERC-20 there is an interface that is the standard, that should be inherit, and it’s main components are:
Variables:

- name: A descriptive name for a collection of NFTs in this contract (e.g., "Punchy NFT").
- symbol: An abbreviated name for NFTs in this contract (e.g., "PNCH_NFT").

- totalSupply: The total number of NFTs tracked by this contract.
- ownerOf: Mapping from token ID to the owner's address.
- balanceOf: Mapping from owner address to the number of tokens they own.
- tokenURI: A mapping from token ID to a URL or other identifier for the token's metadata.

Function:

- **balanceOf**: Returns the number of NFTs owned by a given address.
- **ownerOf**: Returns the owner of the specified token ID.
- **safeTransferFrom**: Safely transfers the ownership of a given token ID to another address. If the target address is a contract, it must implement onERC721Received to accept the transfer.
- **transferFrom**: Transfers the ownership of a given token ID to another address. The caller is responsible to confirm that the recipient is capable of receiving NFTs or else they may be permanently lost.
- **approve**: Approves another address to transfer the given token ID, on their behalf.
- **getApproved**: Gets the approved address for a single NFT.
- **setApprovalForAll**: Approves or removes an operator. Operators can transfer any token owned by the sender.
- **isApprovedForAll**: Returns if the operator is allowed to manage all of the assets of the owner.

Unlike ERC-20 tokens, which are interchangeable and equal in value, ERC-721 tokens have distinct characteristics, allowing for differentiation between individual tokens. While ERC-20 tokens are commonly used for transactions and utility within the Ethereum ecosystem, ERC-721 tokens excel in representing ownership and uniqueness, opening up new possibilities in gaming, digital art, and asset tokenization. These tokens can be stored in compatible Ethereum wallets and traded on platforms supporting non-fungible tokens (NFTs), enabling collectors and creators to buy, sell, and showcase unique digital assets with verifiable ownership on the blockchain. Additionally, ERC-721 tokens can interact with decentralized applications (dApps) and smart contracts, allowing for innovative use-cases such as decentralized marketplaces, tokenized assets, and provably rare digital collectibles. [18]

II.7. Conclusion

In this chapter, we provided an introduction to blockchain technology, defining its core principles and exploring its historical development. We compared blockchain with traditional databases, highlighting the key differences in decentralization, immutability, transparency, speed, data integrity, cost, and cryptography. We also introduced the main components of a blockchain system, including nodes, blocks, transactions, wallets, smart contracts, and tokens, detailing their roles and functionalities within the network. This foundational understanding sets the stage for deeper exploration of blockchain's applications and implications in the subsequent chapters

III. Project Conception

CONTENT

III.1. Introduction

This chapter delves into the project's conceptualization, outlining the core functionalities and system interactions. We begin by exploring the use-case diagram to visualize the actors and their interactions with the system. Following this, sequence diagrams illustrate specific scenarios: creating a challenge, forming a team, and participating in a challenge. Finally, a class diagram will be presented to illustrate the system's objects and their relationships.

III.2. Use-case diagram

The Use-case diagram describes the interdependence between our Punchword application and the actor by identifying the user's needs and everything the app must do for the user specifically in in the viral Campaign functionality. Below is the use-case diagram defining the main actions performed by the user in our system.

Use-cases:

Create a challenge: In order to create a challenge or launch a viral campaign, a user must be a punchy token holder, which means they need to have a minimum amount of punchy tokens in their wallet. Once this requirement is met, the user can proceed to create the challenge.

Creating a team: Users can initiate the creation of a team and invite people to join their team, and the one who creates the team will be set as an admin.

Accepting a team invitation: Once a user creates a team, the invited members will receive a notification to confirm their joining and become official members of the team.

Explore the Challenges: Any user can browse through the challenges available on the platform, either in their feed or by using the explore feature.

View Participations: Users can check out the participations published on the platform by other users, either in the feed or through the explore feature.

Participate in a Challenge: Users can join any challenge published on the platform, either as a team or individually. To participate, they first need to create an NFT of the piece of art they want to enter.

View Teams in the Profile: Users can access the teams they're part of, either as a member or as an admin, in their profile.

Note: To avoid repetition, we have not included the 'authenticate' use-case in this diagram, but it is still used in our software solution.

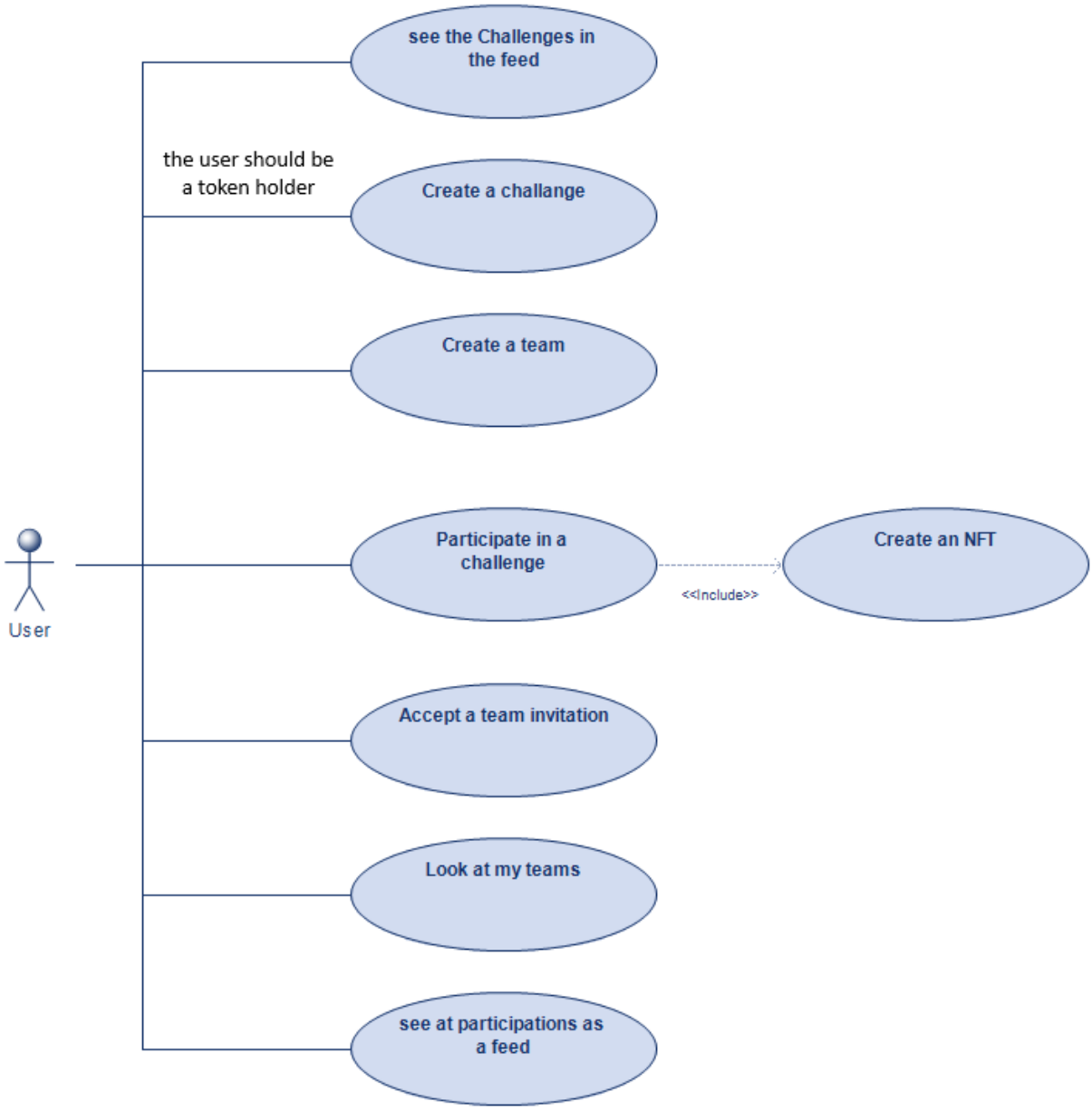

FIGURE III.1 - Use-case diagram

III.3. Sequence diagrams

III.3.1. Create a challenge

In order for a user to create a challenge or launch a viral campaign, they must first be a holder of punchy tokens, meaning they should have a minimum amount of punchy tokens in their wallet. If not we redirect the user to the main screen, Once this condition is met, the user can enter all the

required information to create a challenge, including the number of winners, the prize for each position (which can be specified individually or by range, for example, between 1st and 5th place will get 100 PUNCH and the rest will get 50 PUNCH), as well as the registration deadline and the announcement date. and then present it in a post. After that, the user will need to perform two transactions. The first transaction allows the smart contract to use the total prize amount provided by the user so that when it's time to select the winner, the smart contract can automatically send the money to the winners from the challenge creator, this transaction will be a call for the Allow function in the Punchytoken smart contract. The second transaction involves calling the createChallenge function from the smart contract, this transaction will be the call to the function CreateChallenge in the Challenge smart contract.

Once the challenge is created, we'll have a Firebase Cloud Function running in the background that gets triggered each time when a new challenge is created in the Firestore database. This function will schedule another function to be called at the announcement date to carry out the transactions. This scheduled function will then call the endChallenge function from the Challenge smart contract, which is responsible for selecting the winners, sending the prizes, and closing the challenge.

We need the Firebase Cloud Function because the interactions with the teams are not directly available in the blockchain, but rather in the Firestore database. Therefore, this cloud function will retrieve the interactions for each post of each participant in the specific challenge and send it to the blockchain within the transaction to pick a winner.

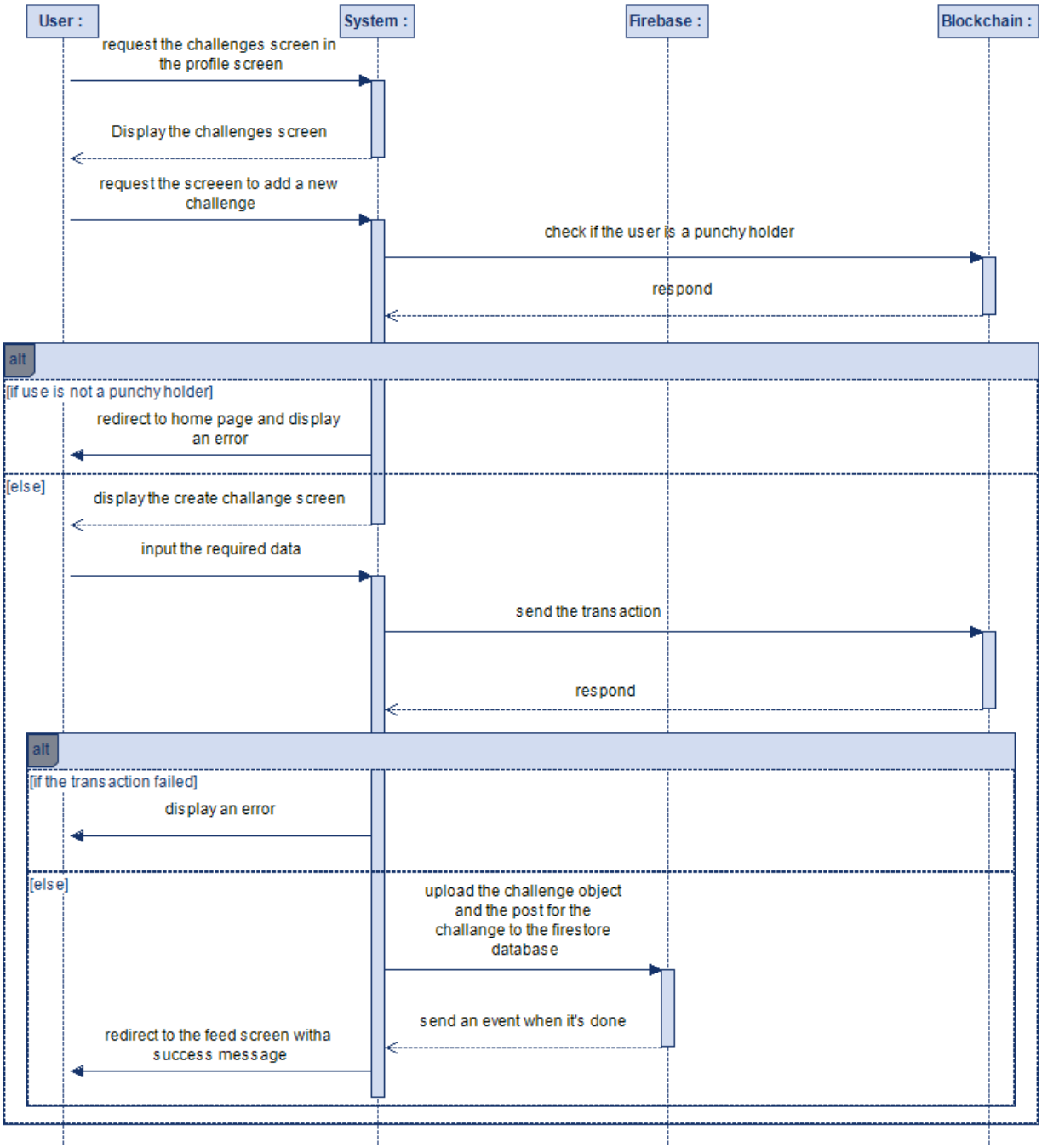

FIGURE III.2 - Sequence diagram - Create a challenge

III.3.2. Create a team

Users have the ability to create teams for participating in challenges. To do this, they first select the members to invite and then determine the share of the winnings that each member will receive. These shares will be used for any future challenge that the team participates in. For example, one user may receive 40% of the prize while another receives 30%. Once the shares are decided, the

transaction that called the function CreateTeam from the team smart contract is performed on the blockchain. After the team is successfully created, invite notifications are sent to all selected users.

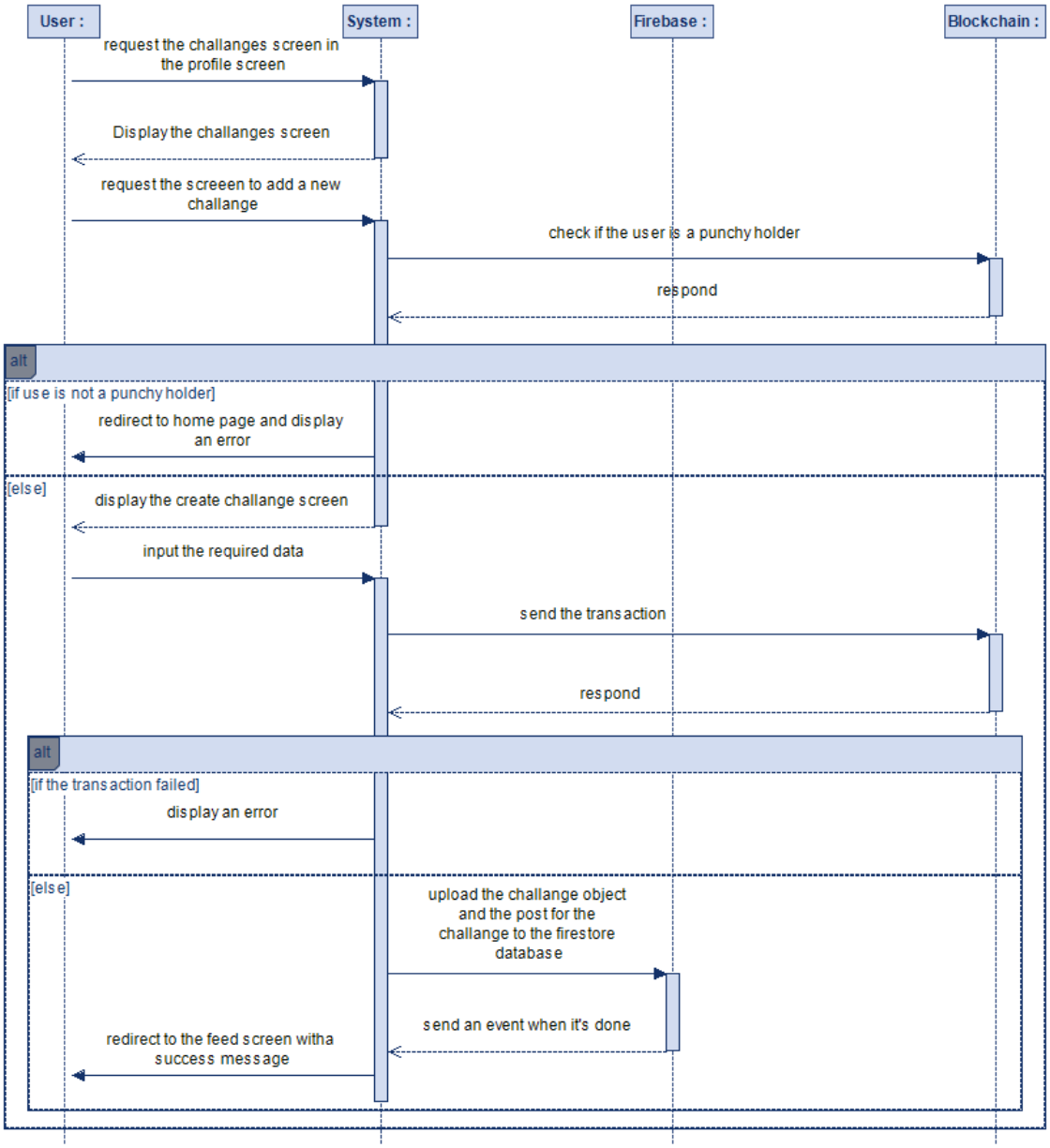

FIGURE III.3 - Sequence diagram - Create a team

III.3.3. Accept a team invitation

Once a user creates a team, all the selected members will receive a notification asking them to confirm their acceptance to the team. This confirmation will be recorded in the blockchain. When a user accepts an invite, they will need to perform a transaction by calling the acceptTeam function from the Teams smart contract.

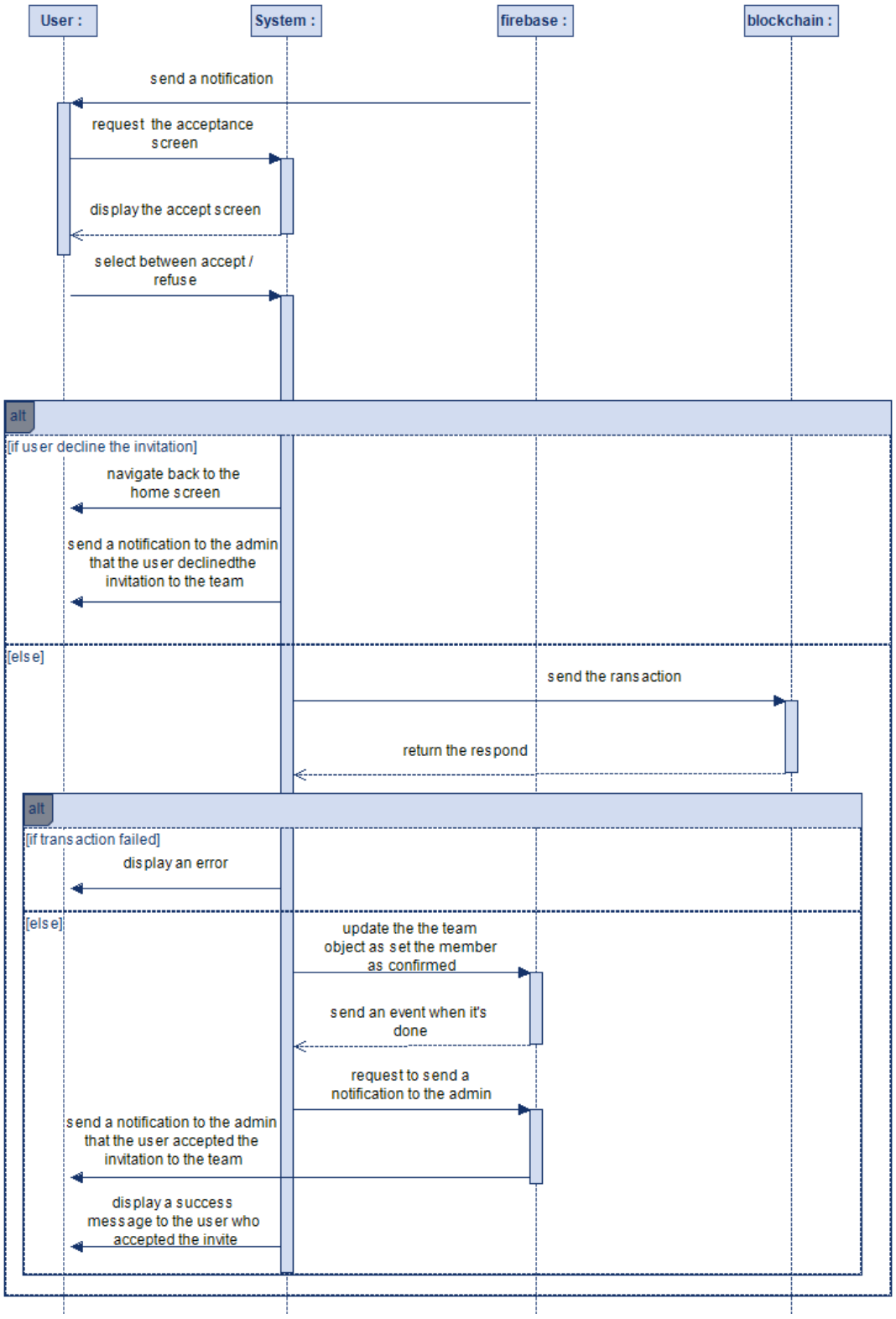

FIGURE III.4 - Sequence diagram - Accept a team invitation

III.3.4. Participate in a challenge

Once the user finds a challenge that interests them, they have two options: they can participate individually or with a team.

When the user decides to participate with a team, the system will only display the teams of which they are an admin and where all members have accepted the invitation.

In both cases, whether individually or as a team, the user must first create an NFT for the piece of art they intend to use, and that by performing the transaction to mint the NFT.

Next, they'll need to perform a transaction that allows the smart contract to transfer the NFT on their behalf. So that at the end if they will be one of the winners, the smart contract will transfer their NFT to the challenge creator, making them the owner.

Finally, they can use the NFT to execute the transaction required to participate in the specific challenge. By calling one of two functions, either "participateIndividually" or "participateAsTeam" from the Challenge smart contract.

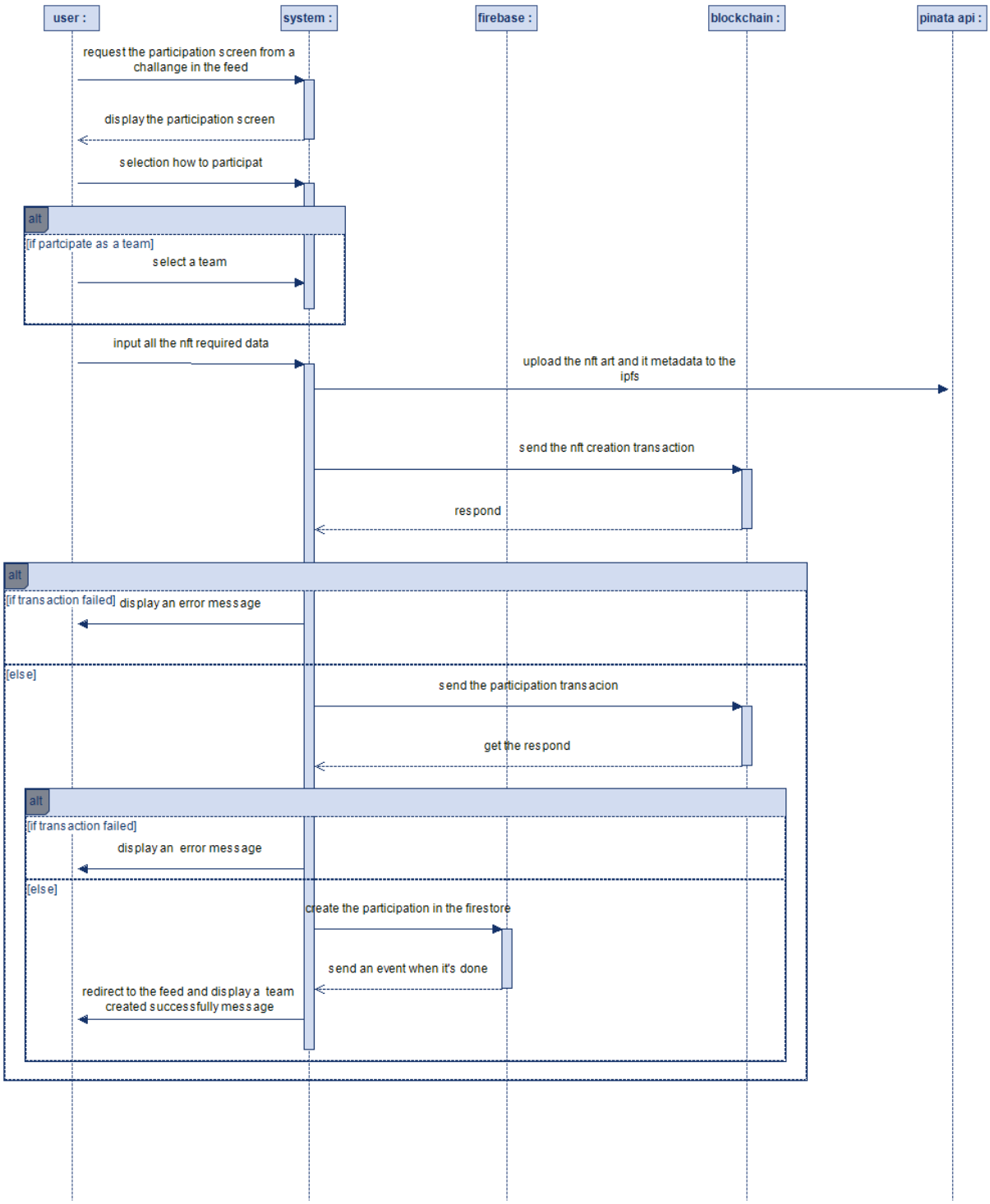

FIGURE III.5 - Sequence diagram - Participate in a challenge

III.4. Class diagram

Here we display each element of the Viral Campaign feature as a class with attributes that help them work together. It includes a total of 5 classes. Each of these classes has attributes that allow each instance to be identified or categorized.

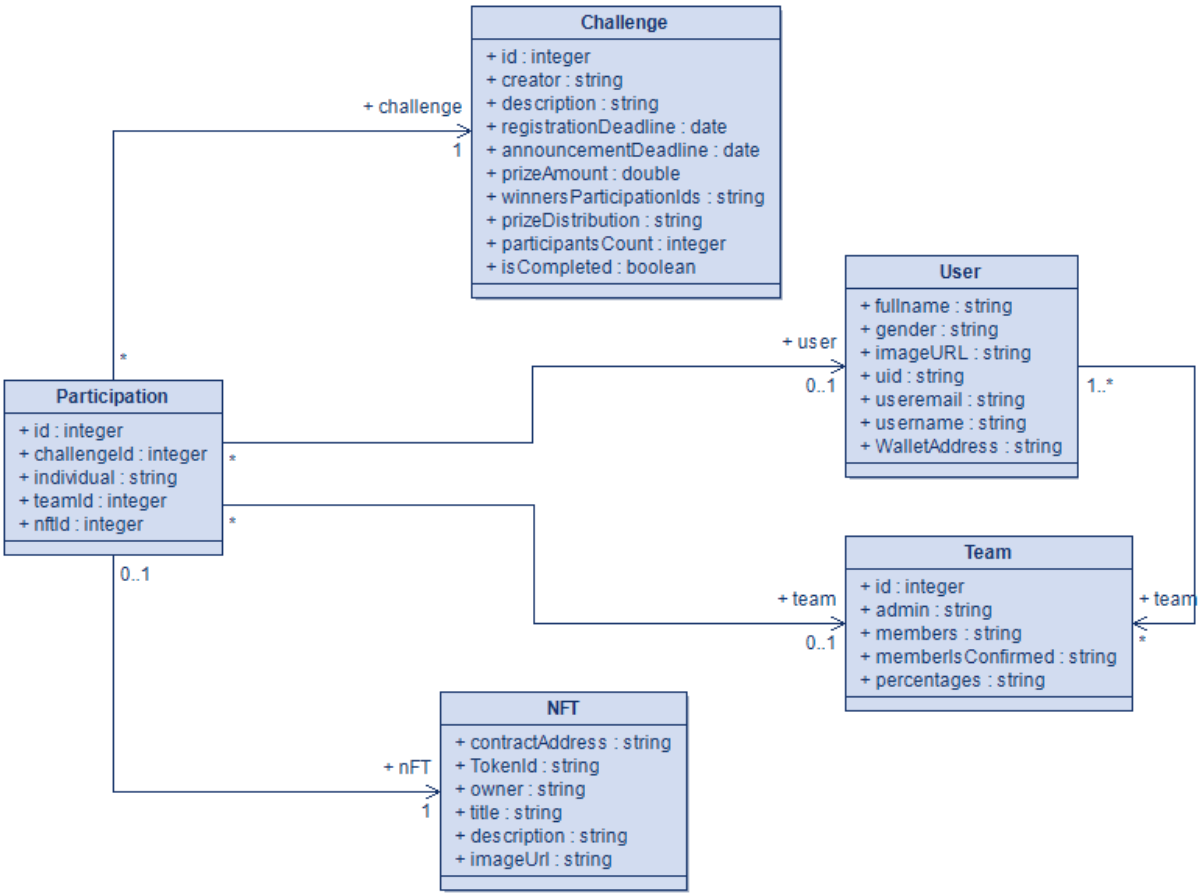

FIGURE III.6 - Class diagram

III.5. Conclusion

Throughout this chapter, we have presented the design of our project using all necessary UML diagrams to help us better understand the feature and how it works. In the next chapter (Project Implementation), we will describe the tools and technologies used, explain the code architectures, demonstrate how the entire system works together, and showcase the interfaces of our feature.

IV. Project Implementation

CONTENT

IV.1. Introduction

Our project involved two key components: developing the Viral Campaign functionality for the Punchword Android app and creating smart contracts.

This chapter will delve into each element of the viral campaign, explaining its purpose and responsibilities.

IV.2. Tools and technologies used

FIGURE IV.1 - Used Technologies

IV.2.1. Android Studio

An integrated development environment (IDE) specifically designed for building Android apps. It includes tools for coding, debugging, testing, and deploying apps. Think of it as a workbench with all the tools you need to build an Android house. [20]

IV.2.2. Remix Ide

From writing to deployment, Remix is a one-stop shop for building Ethereum smart contracts. This versatile toolset caters to both learners and experienced developers. [21]

IV.2.3. Github

A popular version control system and hosting platform for software development projects. It allows developers to track changes, collaborate on code, and share projects publicly or privately. Think of it as a central storage unit for your project files with version control. [22]

IV.2.4. FireBase

A platform from Google that provides backend services for mobile and web apps. It offers features like authentication, databases, cloud storage, cloud functions, and analytics. Imagine it as a set of pre-built tools you can integrate into your app to handle things like user logins and data storage.

We chose Firebase because, on top of being easy to use, it offers a comprehensive set of tools for backend development. Here are the main ones we used in our project:

- **Firebase Authentification:** This service manages user authentification and supports various sign-in methods, such as email/password and Google sign-in. It ensures secure user access and management. [24]
- **Firebase Firestore:** This real-time NoSQL database stores user data, campaign details, and engagement metrics. [25]
- **Firebase Storage:** We use this service to securely store media files in the cloud. It integrates seamlessly with Firestore, allowing efficient data management and retrieval. [26]
- **Cloud Functions:** These serverless functions handle backend logic, such as triggering events based on database changes or managing complex workflows that require interaction between the Android app and the blockchain. [27]

IV.2.5. Kotlin

A modern programming language specifically for Android development, Kotlin is known for being concise, safe, and interoperable with Java. Created by Google, the same company that owns the Android Operating System, Kotlin is the best technology to use if your goal is to build a scalable, fast, and well-optimized Android app. [28]

IV.2.6. Jetpack Compose

A new UI framework for building user interfaces in Android apps, created and officially supported by Google. It uses a components style for composing UI elements, making it easier to create complex, reusable, and dynamic layouts. [29]

IV.2.7. Solidity

A programming language specifically designed for writing smart contracts, which are self-executing contracts stored on a blockchain. [30]

IV.3. System Architecture

we delve into the system architecture of the Punchword platform, detailing the frameworks and architecture utilized in its development. The architecture is divided into three primary components:

the Android application, the blockchain integration, and the backend services managed through Firebase.

IV.3.1. Android Application

The Android part of the Punchword employs a multi-module architecture combined with Clean Architecture principles. This structure enhances the scalability and maintainability of the application by clearly separating concerns into distinct layers and modules, let’s explain each one of those in detail.

IV.3.1.1. Clean Architecture

The goal of using this Architecture is to divide our code into 3 layers, these layers are Data, Doain, and Presentation/UI. So let’s go through these step by step and understand what these layers actually contain.

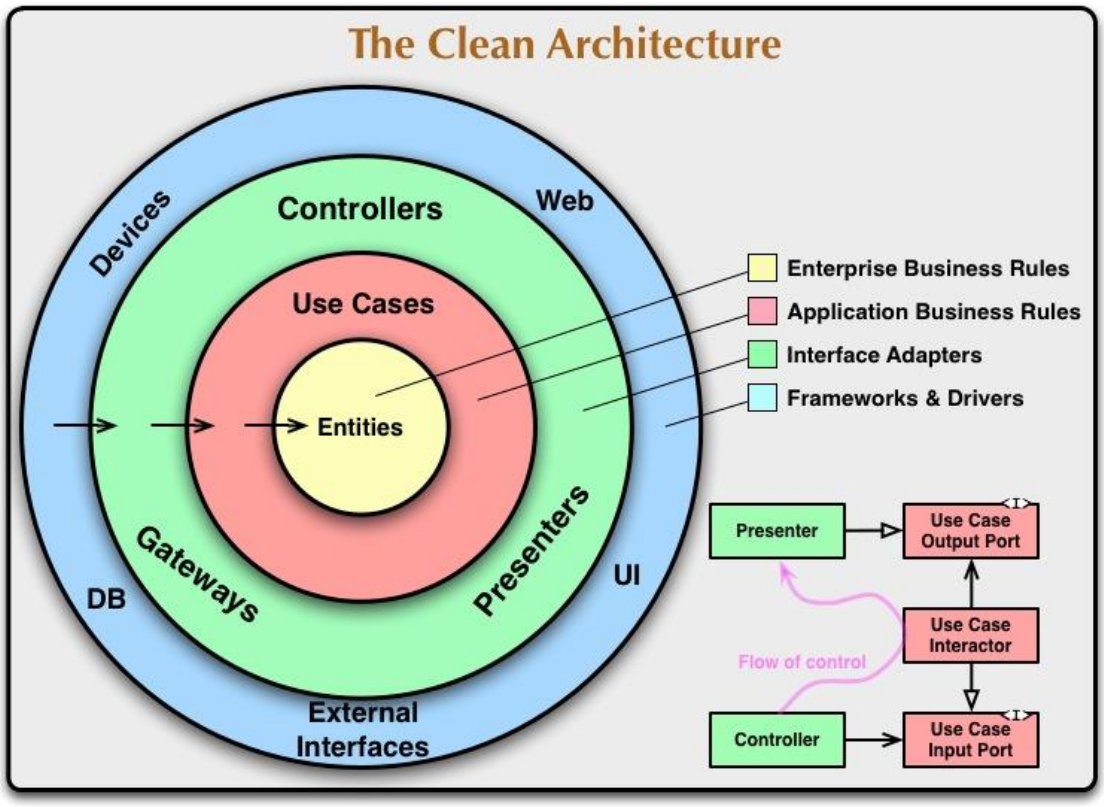

FIGURE IV.2 - The Clean Architecture [31]

- **Data Layer**

Let's start with the Data layer, which is the easiest to understand because its name clearly indicates its contents. It includes database implementations, API implementations, preferences, local

databases, and third-party providers that communicate with remote services. In our case, we have Firebase and Web3j, both of which we'll discuss shortly. These are the types of things that belong in our data layer.

Another job of the data layer is what is called mappers, mappers are a key concept of clean architecture. They take database entities (objects saved in the database) and DTO objects (Kotlin representation of a JSON response, obtained from an API), and transform them into a unified data class. This unified class will be explained in more detail when we discuss the domain layer, making it easier to work with these classes.

- **Domain Layer**

The Domain Layer serves as the middle layer of the architecture, with two primary responsibilities. Firstly, it houses the business logic, in the form of "Use-Cases." A Use-Case is essentially a class that performs a single function. For example, the use-case for creating a team would involve actions such as retrieving the wallets of the selected members, executing the team creation transaction, creating the team object in Firebase, and sending each member an invitation notification. It's important to note that calls to Firebase and blockchain transactions or any kind of remote services are defined in the data layer as functions in the repository class.

Secondly, the Domain Layer contains the Model classes, which are a combination of entities and DTO classes, or other types of classes. These model classes do not necessarily need to originate directly from our data layer. Another way to think about a model is as a class that represents a data type, for example, we have Team, Challenge, User, etc. so it’s a class that holds data that we can also easily later use in our presentation layer to show the data that we need to show

- **Presentation/UI Layer**

This will be the final layer in our clean architecture, the Presentation layer. It is used to present something to the user in the form of a user interface (UI). In the end, it will contain our composable if we use Jetpack Compose, and if we use XML, it will contain fragments and activities, and in both cases, we’ll have viewModels.

The ViewModel is considered the presenter. Its job is not to contain the business logic but to call the use-cases from the domain layer and map the result to the UI state. The UI will then observe that state.

Why do we use these Use-Cases and call them from the ViewModel instead of directly writing the business logic in the ViewModel? There are two reasons for that. First, it ensures that we don’t have huge viewModels containing excessive business logic and code, which can make them hard to read. Typically, we have a single ViewModel per screen, so this approach helps prevent them from becoming overly complex. Second, it makes our code a lot more readable because we usually give those use-cases very natural names. For example, a use-case class name would be

"SearchUser." You don’t have to be a genius to understand what this class does. If there is a new member in the team and they take a look at the use-case packages, they'll quickly understand what these classes contain and what kind of things a user is able to do in an app.

And one important note here, In the clean architecture, every layer can access the domain layer, but the domain layer is the only one that connects the presentation and data. The presentation layer should not have access to the data layer, and vice versa. [31]

IV.3.1.2. Multi-module architecture

This approach involves dividing the app into multiple Gradle modules using a "Layered-Feature Modularization" strategy. For example, in the case of a social media app, we would have modules for Messaging, Feed, and Viral Campaigns. Each of these modules would be further divided into three sub-modules following the clean architecture (Data, Domain, Presentation). This approach allows for easy reuse of a single feature and helps in work management by enabling separate teams to work on different features without interfering with each other's work. Additionally, it helps keep our modules smaller as we don't have all the feature code in a single module. This multi-module architecture also leads to faster build times due to multi-threading in Gradle and the ability to rebuild only the modules that have actually changed. Furthermore, it promotes code reusability, allowing for easy export and implementation of modules in other projects.

So the project will contain the following modules:

- **App Module:** The entry point of the application, containing the main application logic and UI.
- **Feature Modules:** Separate modules for each major feature (e.g., user profile, NFT marketplace).
- **Core Module:** Contains shared utilities, base classes, and common resources.

Additionally, we will implement Clean Architecture Sub-Modules, consisting of three modules: data, domain, and presentation. Each feature module will contain these sub-modules, following the clean architecture principles we discussed earlier. [32]

IV.3.2. Smart Contacts

To develop the smart contracts, we used the **Foundry** framework. It's a comprehensive solution for smart contract development using Solidity. With Foundry, you can manage your dependencies, compile your project, run tests, and even deploy your smart contract directly using command lines. Additionally, it allows you to interact with the chain from the command line and via Solidity scripts. [33]

IV.3.3. Connecting Android App to Blockchain

We need to figure out how to interact with our smart contracts deployed on the blockchain. To do this, we are using Web3j, which is a comprehensive Java library for working with Ethereum. Since Kotlin can interact with Java, we have no issues in that regard. This library enables communication between our Android application and the Ethereum blockchain, allowing the app to interact with smart contracts and perform blockchain operations. It establishes a connection to an Ethereum node, which can be a local node (like Ganache or Anvil) or a remote node (like Infura or a private node). Then, it allows us to load and interact with deployed smart contracts by providing their addresses and Application Binary Interface (ABI) definitions. [34]

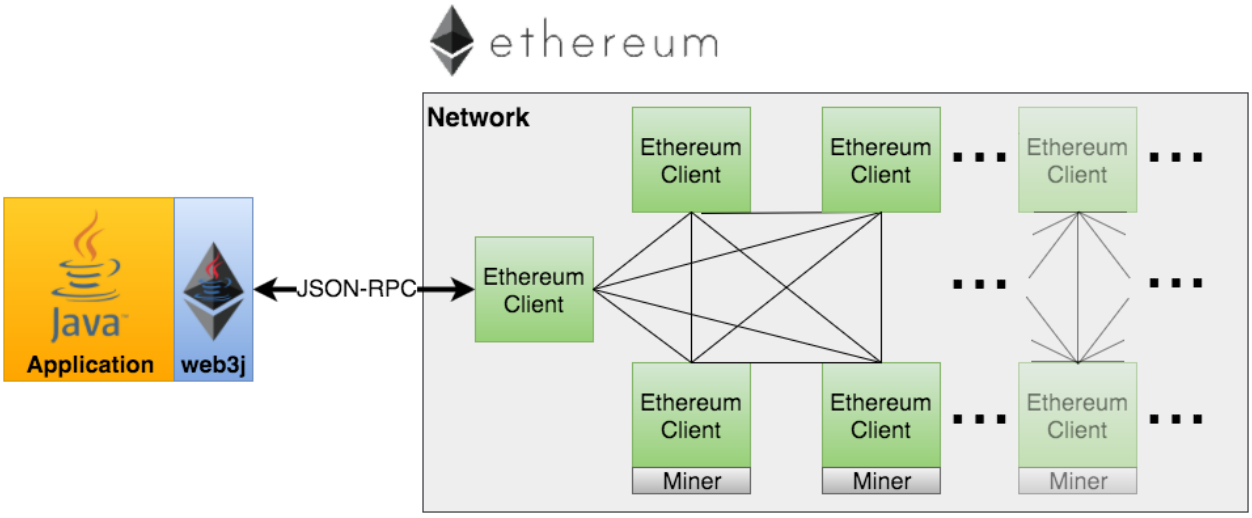

FIGURE IV.3 - Web3j connection between the blockchain and the Java app [34]

IV.3.4. FireBase

We initially planned to create a social media platform based on the Blockchain. However, we realized that not all functionalities should be fully decentralized. For example, it doesn't make sense for users to pay gas fees for actions like posting an image or sending a message. To address this, we decided to use a combination of Blockchain and a database. This also allows us to quickly access data, as reading from the Blockchain can be slow at times. Additionally, data written to the Blockchain cannot be edited or deleted. Therefore, when a user creates a team, we store that information in our Firestore database. This way, we can retrieve the team information quickly, providing a better user experience.

An important use case involved the use of Firebase cloud functions and cloud tasks to automate the conclusion of a challenge. We couldn't finish the challenge directly from the smart contract because the posts are saved in the Firestore database. This meant that in order for the smart contract to choose a winner when the announcement date arrived, it needed to access our Firestore database

to retrieve the posts from the participants of a specific challenge, which is not possible. To work around this, we created a Firebase cloud function that will be triggered each time a new challenge document is created. This function will then retrieve the announcement date of the challenge document and schedule an HTTP trigger using the cloud tasks. This would involve another HTTP function being called automatically at a specific time, in our case, at the announcement date. This last function would then retrieve all the participations of the challenge id, get the interactions (likes, comments, shares) for each post from each participation, and then send this data along with the challenge id to the blockchain by executing a transaction and calling the "endChallenge" function from our challenge smart contract. This "endChallenge" function would then determine the winner based on the reach and send the prize to each winner.

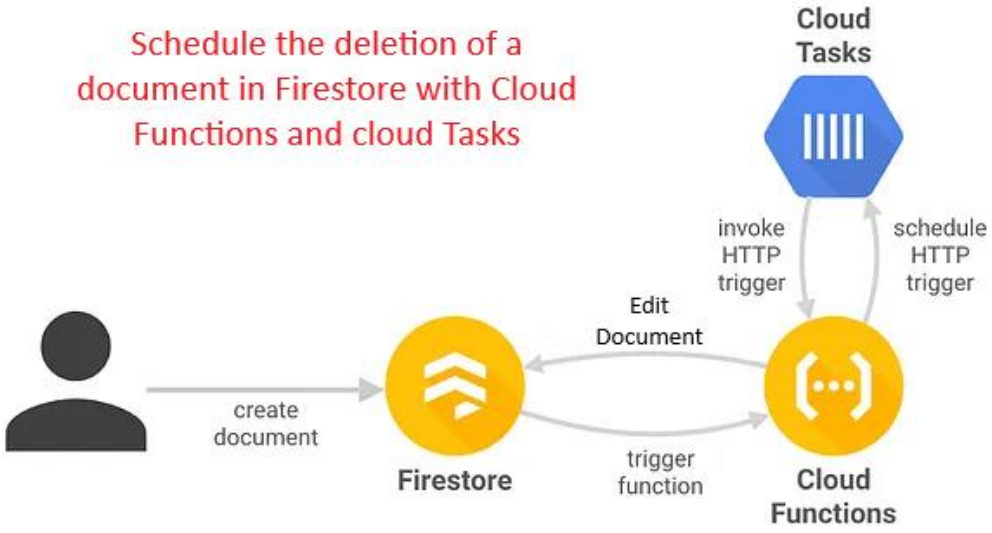

FIGURE IV.4 - Cloud Tasks with Firebase Cloud Function\

IV.4. User Experience

IV.4.1. Create Challenge

- From the profile screen, the user can access the general viral campaign from the bottom navigation bar

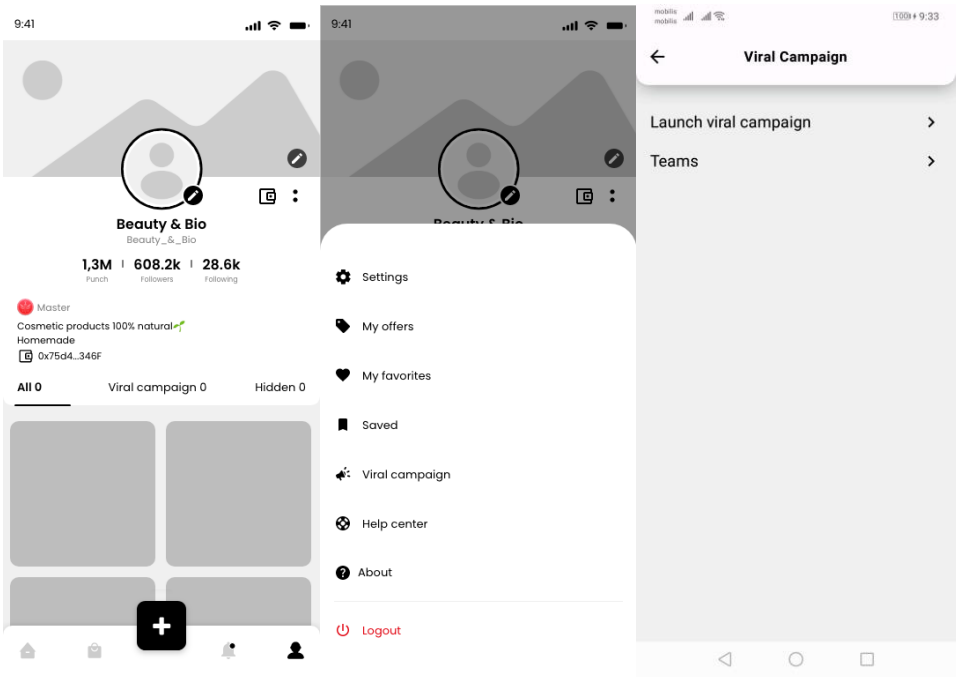

FIGURE IV.5 - Create challenge, Profile screens

- After that, the user will need to enter the required information: the number of winners, the prize for each position, as well as the registration deadline, and the announcement date.

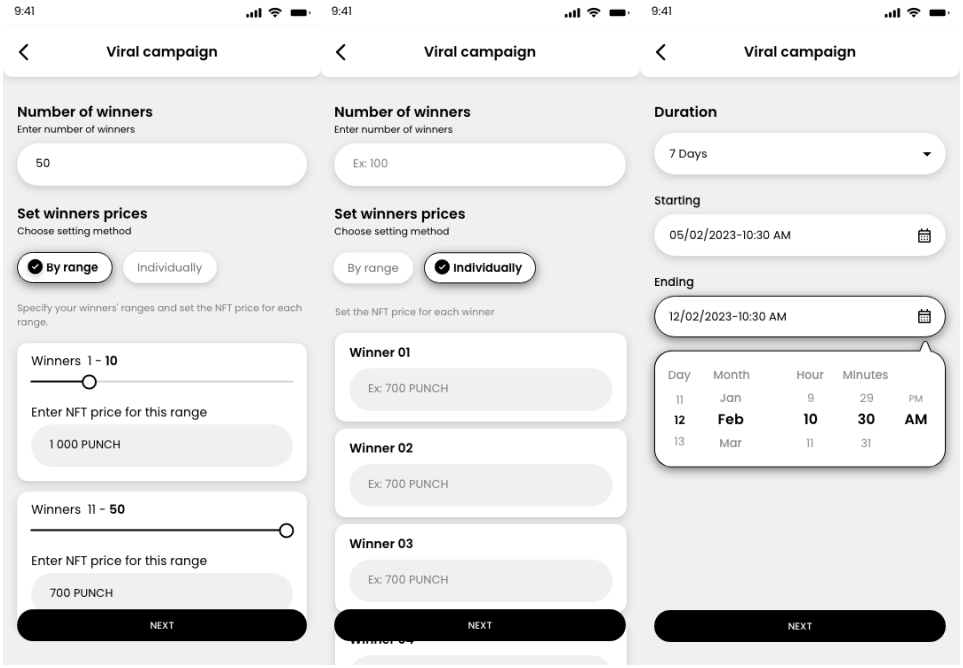

FIGURE IV.6 - Create Challenge, data input screens

- Then the user must create a post that will be displayed in the feed.

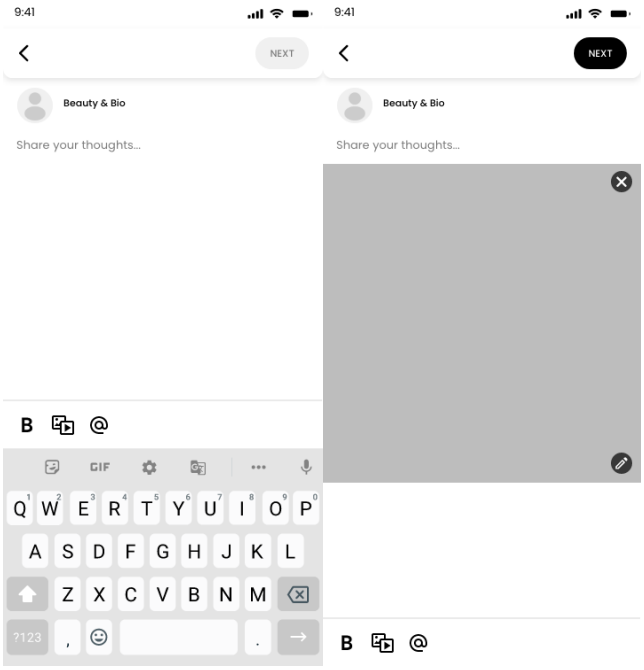

FIGURE IV.7 - Post creation screen

- As a last step, the transaction price screen will be displayed. The user will need to confirm the transaction by entering their wallet password.

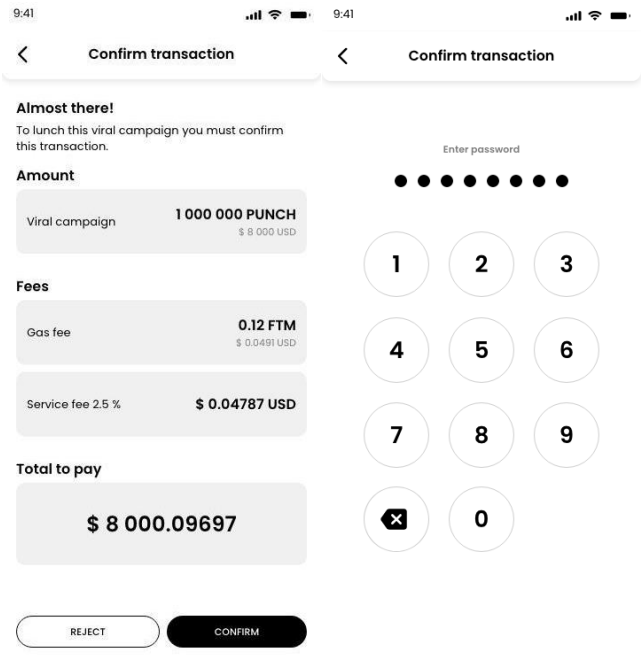

FIGURE IV.8 - Create a Challenge, transaction, and feed screens

- Once the transaction is successfully completed, they will be redirected to the main screen.

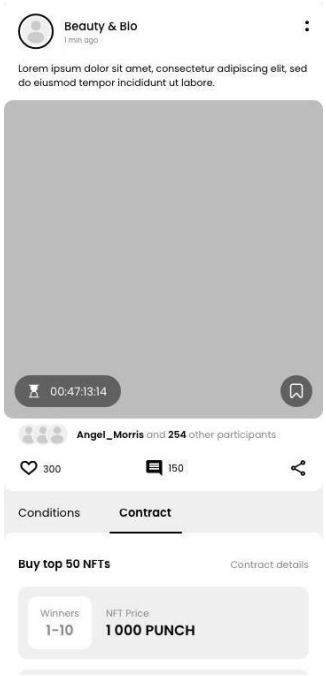

FIGURE IV.9 – Feed Screen

IV.4.1. Create Team

- You can access all your teams and create a new team on the viral campaign screen.

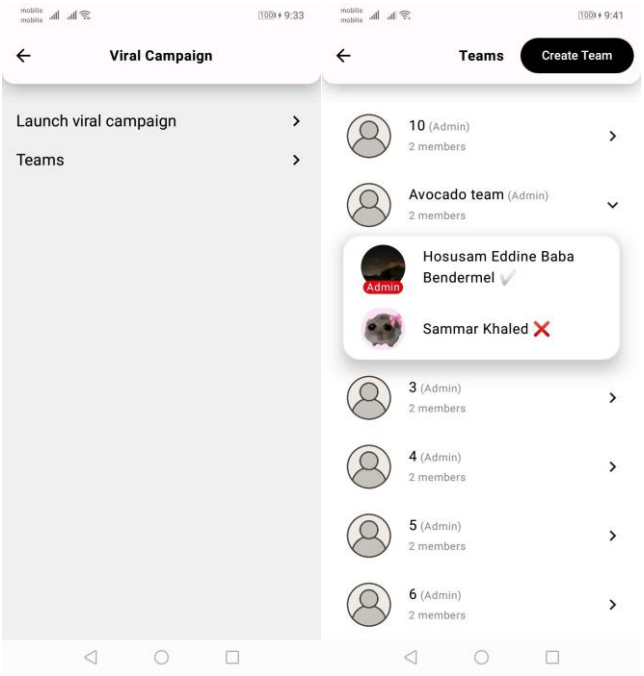

FIGURE IV.10 - Profiles team, profile screens

- The user will need to enter a name and upload an image for the team's identity. After that, they can select the members they want to invite and specify the share for each of them.

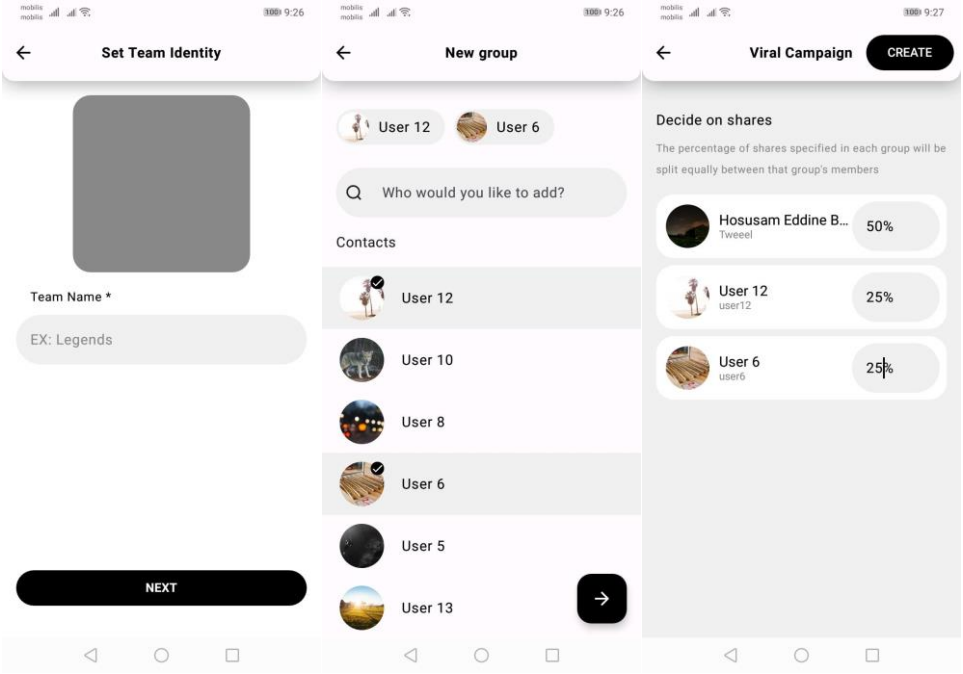

FIGURE IV.11 - Create a Challenge, team data input screens

- As a final step, they'll have to complete the transaction by entering their password.

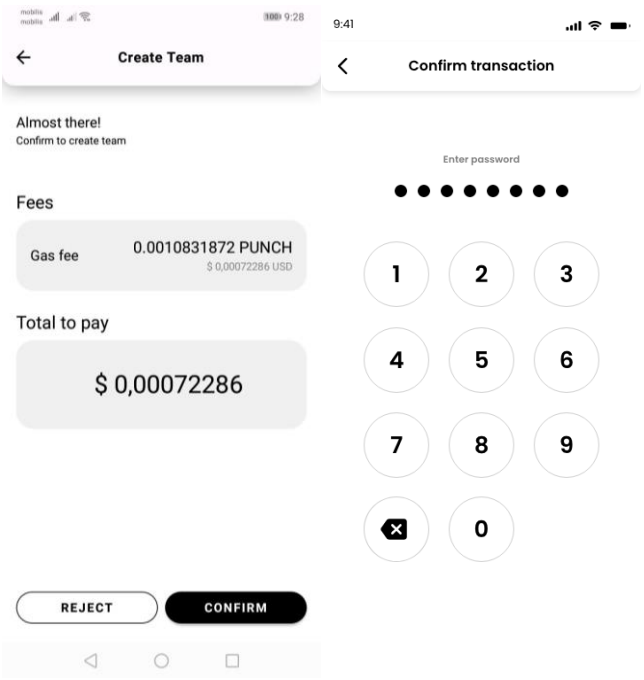

FIGURE IV.12 - Create a challenge, transaction screens

IV.4.1. Accept team Invitation

- When invited, users can accept by clicking "join group" after receiving a notification.

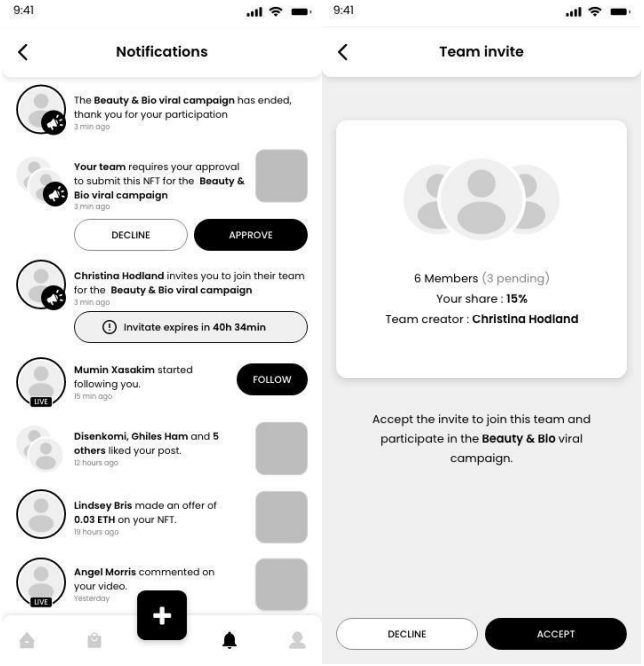

FIGURE IV.13 - Accept team invitation, notification, and team presentation screens

- As a final step, they'll have to complete the transaction by entering their password.

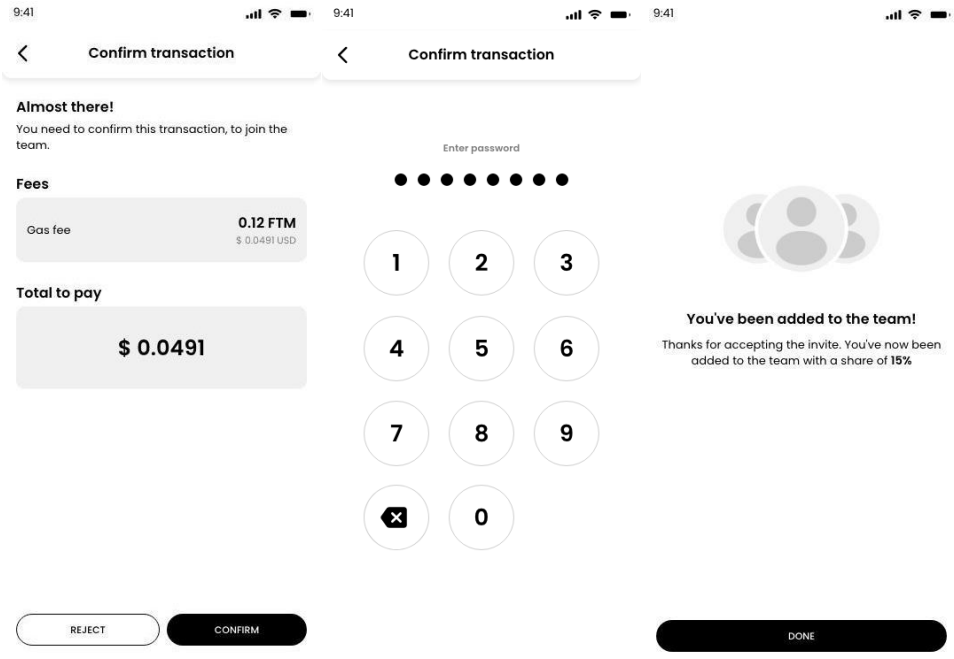

FIGURE IV.14 - Accept team invitation, transaction screens

IV.4.1. Participate in a challenge

- Once a user finds a challenge that interests them, they can click on the participate button and begin by selecting the participation type (Individual or Team).

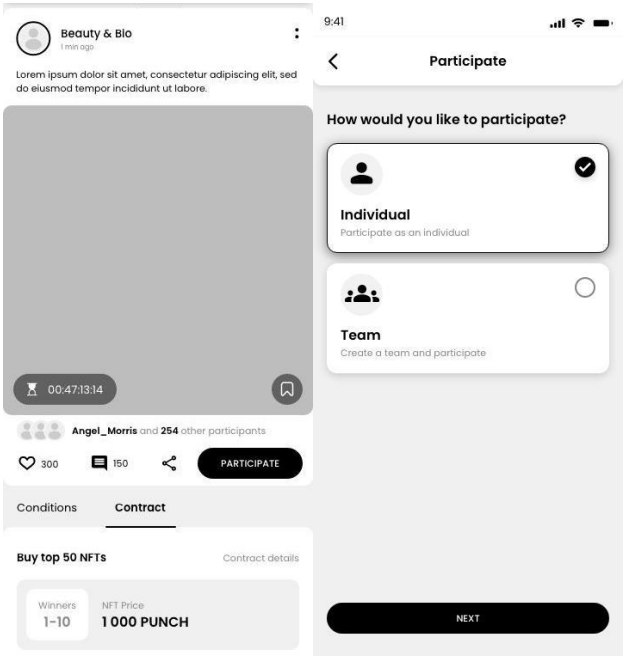

FIGURE IV.15 - participate in a challenge, feed, and participation type selection screens

- If they choose to participate, they must select a team to join. The screen will only display teams for which the user is an admin, and all invited members have accepted the invitation.

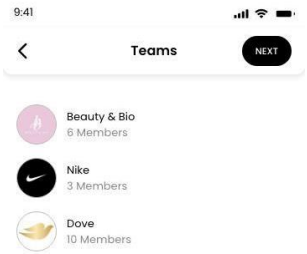

FIGURE IV.16 - Participate in a challenge, select a team screen

- Then the user has to enter the NFT information (the piece of art, its title, and the description).

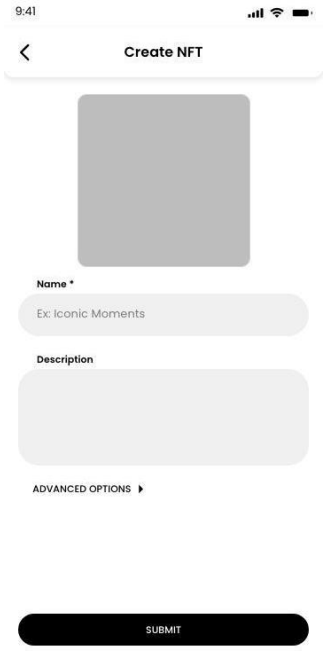

FIGURE IV.17 - participate in a challenge, create an NFT screen

- As a final step, they'll have to complete the transaction by entering their password.

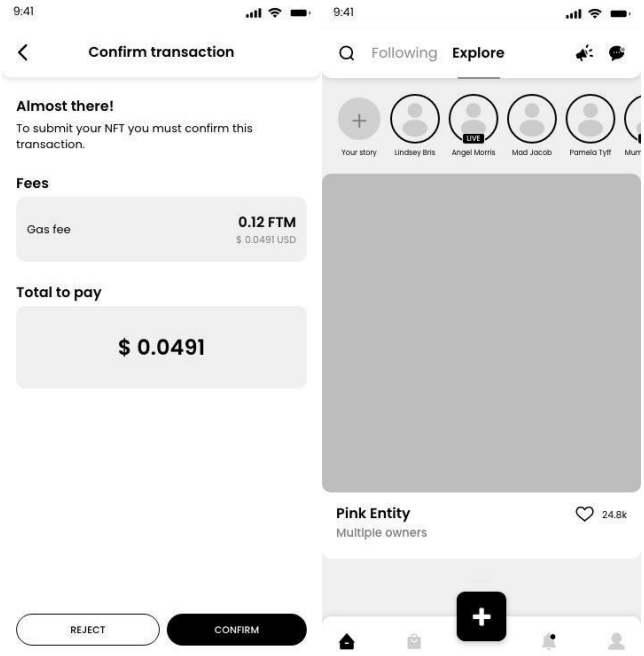

FIGURE IV.18 - participate in a challenge transaction

General Conclusion

This thesis explored the integration of blockchain technology into the Punchword social media platform, where I work as an Android developer. The focus was on the "Punchword Viral Campaigns" feature, which combines NFTs with viral marketing campaigns. Companies can create challenges on Punchword, where users submit NFT ads and compete for rewards in Punchy, the platform's cryptocurrency. This setup uses smart contracts to automate reward distribution based on engagement metrics like likes and shares, enhancing transparency and security for advertisers and influencers.

The first chapter covered blockchain basics, defining key concepts and contrasting blockchain with traditional databases. It detailed blockchain components such as nodes, blocks, transactions, wallets, smart contracts, and tokens, including ERC-20 and ERC-721 standards.

The second chapter detailed the design and development of the "Punchword Viral Campaigns" feature, with diagrams illustrating the system architecture and workflows.

The third chapter discussed the tools and technologies used (Android, Solidity), the app’s architecture (MVVM), and the integration of the mobile app with blockchain using the web3j library. It also showcased the user experience with screenshots of the Viral Campaigns feature.

At the conclusion of this project, I have gained valuable expertise in blockchain and mobile development. We are satisfied with the results, and as a future development task, we plan to include transaction sponsorship. This means that users will not have to pay the gas fees; instead, the fees will be sponsored and paid directly by us.

References

[1] Blockchain Facts: What is it, How it Works, and How It Can Be Used. consulted on 13/06/2024. URL: https://www.investopedia.com/terms/b/blockchain.asp

[2] What is blockchain? consulted on 25/05/2023. URL: https://www.ibm.com/topics/blockchain

[3] A timeline and history of blockchain technology. consulted on 25/05/2024. URL: https://www.techtarget.com/whatis/feature/A-timeline-and-history-of-blockchain-technology

[4] Bitcoin: A Peer-to-Peer Electronic Cash System. consulted on 13/05/2024. URL: https://bitcoin.org/bitcoin.pdf

[5] Blockchain vs Database: Understanding The Difference. consulted on 24/05/2024. URL: https://101blockchains.com/blockchain-vs-database-the-difference/

[6] Blockchain Vs Relational Database: What's The Difference? Consulted on 24/05/2024/ URL: https://101blockchains.com/blockchain-vs-relational-database/

[7] How Blocks Are Added to a Blockchain, Explained Simply. Consulted on 26/05/2024. URL: https://www.coindesk.com/learn/how-blocks-are-added-to-a-blockchain-explained-simply/

[8] Proof of Work vs Proof of Stake: Basic Mining Guide. Consulted on 08/06/2024. URL: https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/

[9] Was ist eine Blockchain-Brieftasche und wie funktioniert sie. Consulted on 13/06/2024. URL: https://www.cropty.io/blockchain-wallet

[10] Blockchain Blocks Explained: How Blocks Are Added To A Blockchain . consulted on 13/06/2024. URL: https://droomdroom.com/blockchain-blocks-explained/

[11] Types of blockchain nodes explained. Consulted on: 29/05/2024. URL: https://polymesh.network/blog/types-of-blockchain-nodes-explained

[12] What are the 3 Key Components Of The Blockchain Network? Consulted on 29/05/2024. URL: https://iglu.net/key-components-of-the-blockchain-network/

[13] Blockchain Explained. Consulted on 01/06/2024. URL: https://www.web3labs.com/blockchain-explained-what-is-a-blockchain-transaction

[14] Know Everything about Crypto Wallet. Consulted on 09/06/2024/ URL: https://101blockchains.com/crypto-wallets/

[15] What Are Public and Private Keys? Consulted on 03/06/2024. URL: https://www.gemini.com/cryptopedia/public-private-keys-cryptography#section-what-does-it-mean-to-digitally-sign-a-transaction

[16] Key Components of a Blockchain Network. Consulted on 03/06/2024/ URL: https://www.identity.com/key-components-of-a-blockchain-network/

[17] What is a token?. Consulted on 01/06/2024. URL: https://www.bitcoin.com/get-started/what-is-a-token/#what-is-a-token

[18] What is a token? Consulted on 02/06/2024. URL: https://www.coinhouse.com/learn/blockchain-technology/what-is-a-token/

[19] What is an NFT? (Non Fungible Tokens Explained). Consulted on 02/06/2024/ URL: https://whiteboardcrypto.com/what-is-a-nft/

[20] Meet Android Studio. Consulted on 20/05/2024. URL: https://developer.android.com/studio/intro

[21] REMIX PROJECT JUMP INTO WEB3. Consulted on 20/05/2024. URL: https://remix-project.org/?lang=en

[22] About GitHub and Git. Consulted on 20/05/2024. URL: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git

[23] Make your app the best it can be with Firebase and generative AI. Consulted on 20/05/2024/ URL: https://firebase.google.com/

[24] Firebase Authentication. Consulted on 20/05/2024. URL: https://firebase.google.com/docs/auth

[25] Cloud Firestore. Consulted on 20/05/2024. URL: https://firebase.google.com/docs/firestore

[26] Cloud Storage for Firebase. Consulted on 20/05//2024. URL: https://firebase.google.com/docs/storage

[27] Cloud Functions for Firebase. Consulted on 20/052024. URL: https://firebase.google.com/docs/functions

[28] Get started with Kotlin. Consulted on 20/05/2024. URL: https://kotlinlang.org/docs/getting-started.html

[29] Build better apps faster with Jetpack Compose. Consulted on 20/05/2024. URL: https://developer.android.com/develop/ui/compose

[30] Solidity. Consulted on 20/05/2024. URL: https://docs.soliditylang.org/en/v0.8.26/

[31] The Clean Architecture. Consulted on 21/05/2024. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

[32] Guide to Android app modularization. Consulted on 21/05/2024. URL: https://developer.android.com/topic/modularization

[33] Foundry Book. Consulted on 22/05/2024. URL: https://book.getfoundry.sh/

[34] Web3j. Consulted on 22/05/2024. URL: https://docs.web3j.io/4.11.0/

ملخص

ينصب تركيز هذا الماجستير على تنفيذ وتقييم ميزة "Viral Campaigns" في تطبيق Punchword، وهو تطبيق تواصل اجتماعي يعتمد على Blockchain. تدمج هذه الميزة الرموز غير القابلة للاستبدال (NFTs) مع استراتيجيات التسويق الفيروسي. فهو يمكّن الشركات من إطلاق تحديات حيث يقوم المستخدمون بإنشاء إعلانات تعتمد على NFT، ويتنافسون للحصول على مكافآت في Punchy، العملة المشفرة الأصلية للمنصة. تعمل العقود الذكية على أتمتة توزيع المكافآت بناءً على مقاييس المشاركة، مما يضمن الشفافية والأمان. من خلال الاستكشاف التفصيلي والتنفيذ باستخدام أدوات تطوير Android وSolidity للعقود الذكية.

الكلمات المفتاحية: البلوكتشاين، وسائل التواصل الاجتماعي ، NFT، الذكية العقود ، ERC-20، ERC-721، تطوير الاندرويد ، MVVM المعمارية الهندسة، Solidity، web3j.

Abstract

This master's focus is on the implementation and evaluation of the " Viral Campaigns" feature into the Punchword Application, a social media App based on the Blockchain. This feature merges Non-Fungible Tokens (NFTs) with viral marketing strategies.
It enables companies to launch challenges where users create NFT-based advertisements, competing for rewards in Punchy, the platform's native cryptocurrency. Smart contracts automate reward distribution based on engagement metrics, ensuring transparency and security. Through detailed exploration and implementation using Android development tools, and Solidity for smart contracts.

Keywords: Blockchain, Social Media, NFTs, Smart Contracts, ERC-20, ERC-721, Android Development, MVVM Architecture, Solidity, web3j.

Résumé

Ce mémoire de master se concentre sur la mise en œuvre et l'évaluation de la fonctionnalité "Viral Campaigns" dans l'application Punchword, une application de médias sociaux basée sur la blockchain. Cette fonctionnalité fusionne les jetons non fongibles (NFT) avec des stratégies de marketing viral. Elle permet aux entreprises de lancer des défis où les utilisateurs créent des publicités basées sur des NFT, en compétition pour des récompenses en Punchy, la cryptomonnaie native de la plateforme. Les contrats intelligents automatisent la distribution des récompenses en fonction des métriques d'engagement, assurant transparence et sécurité. À travers une exploration détaillée et une implémentation utilisant des outils de développement Android et Solidity pour les contrats intelligents.

Mots-clés : Blockchain, Plateforme de Médias Sociaux, NFT, Contrats Intelligents, ERC-20, ERC-721, Développement Android, Architecture MVVM, Solidity, web3j.