IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 2 May 2024, accepted 25 June 2024, date of publication 3 July 2024, date of current version 12 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3422530

RESEARCH ARTICLE

Non-Fungible Token Enhanced Blockchain-Based Online Social Network

SHRUTI JADON, KARTHIKEYA BHAT, KARTHIKEYA R. JENNI, KRISHNA VEDANTHA, LIKITH R R, AND PRASAD B. HONNAVALLI, (Member, IEEE)
Department of Computer Science and Engineering, PES University, Bengaluru 560085, India

Corresponding author: Shruti Jadon (shrutijadon@pes.edu)

This work was supported by PES University.

ABSTRACT In the current digital landscape, almost everyone is on social media or various social media platforms. People use social media for a plethora of purposes, which include staying connected with friends and family, accessing information and updates about ongoing events, entertainment, networking with professionals, expressing themselves to a wide range of users, promoting businesses, joining online communities and engaging in various activities which has led to an increase in the consumption and usage of online social networks (OSN). One of the reasons for such a growth is their features such as ubiquitous access, on-demand service, friendship networks, user engagement strategies like recommendation engines, etc. However, there are various limitations to the current approach, such as the centralization of control, lack of data ownership, poor access control, fake news, bot accounts, censorship, digital rights management issues, etc. To address these limitations, a paradigm shift is necessary. This paper aims to develop a social media application where every post can be converted to a Non-Fungible Token (NFT) and be sold to earn money. Interplanetary File System (IPFS) is used as the decentralized storage. Algorithms for all the functionalities of the applications are given along with an algorithm for a reputation score for every user and their posts in social media are also proposed.

INDEX TERMS Online social network, distributed social network, blockchain, non fungible tokens, IPFS, Ethereum.

I. INTRODUCTION

The way the modern world appears and operates has been greatly influenced by social media applications. This has created new avenues for becoming well-known. A case in point is the ascent of American YouTuber James Stephen Donaldson, popularly known as "MrBeast". He is well-known for his high-production, fast-paced YouTube videos with intricate tasks and substantial prizes. Over 211 million people have subscribed to Mrbeast's YouTube channel [1]. Social media applications enable access-controlled or subscriber/publisher models to facilitate social interaction between users and content consumption. Security threats also develop in tandem with rising demand. This implies that security measures need to be given more attention to lessen potential dangers [2]. It is imperative to develop a robust infrastructure that facilitates the utilisation of blockchain technology. The social media platforms ought to be designed to fully utilise all of the benefits and capabilities that blockchain technology has to offer. The foundation of current social networking applications is a centralised data centre. Centralised systems come with advantages and disadvantages. While centralised server deployments give organisations more control over user data, customers often complain about not having enough ownership of their data. Second, rather than having their terms and conditions decided democratically, users have to accept a set that is drafted to serve the organization's interests. Centralised servers are also susceptible to server outages. Servers may be subject to denial of service attacks of various scales, which frequently lead to poor user experiences. Income reductions for content creators on social media apps are managed by the organization's laws. Data breaches and insider misbehaviour might be the result of such a design. Current social media applications have also

The associate editor coordinating the review of this manuscript and approving it for publication was Aneel Rahim.

caused issues like copyright violations and privacy concerns. This research presents a novel method of a social media application that combines a thorough reputation system with an NFT marketplace. This combination offers a more secure substitute for conventional centralised methods, with the goal of revolutionising user interaction, content monetization, and digital ownership.

A. SCOPE OF THE PAPER

This study aims to explore the development and integration of a blockchain-based social media application with a dedicated NFT marketplace. A formula and an algorithm for a reputation score for every user and their posts on social media are provided. Algorithms for social media operations like Registration of the Users, retrieving user's information, uploading a post, retrieving posts and liking and disliking a post are also provided. Other algorithms for the NFT Marketplace operations like creating a token, creating a market item, reselling a token, creating a market sale item and retrieving the NFTs are mentioned.

B. MOTIVATION

The study is motivated by several key factors:

- Addressing the shortcomings of the social media apps that are already in use, such as centralization, poor data ownership, data breach susceptibility, prolonged outages, and low user revenues.
- Identifying a gap in the existing research, which frequently approaches NFTs and social media applications as separate concepts without considering how NFTs may be integrated into social media platforms.
- Developing a social networking application that allows each post to be turned into an NFT and sold for profit.

C. ORGANIZATION OF THE PAPER

The paper is structured into eight sections for a comprehensive exploration.

- **Section I:** Introduction provides an overview of the existing challenges in current social media networks.
- **Section II:** Background delves into the fundamentals of blockchain, NFTs and OSNs. It explores the potential collaboration of these elements to enhance safety and security, content monetization, transparent data governance, improved content curation, and the prevention of fake news.
- **Section III:** Related Works provides a broader overview of relevant research conducted under various schemes
- **Section IV:** Architecture introduces the proposed OSN architecture, integrating with blockchain and NFT marketplace.
- **Section V:** Reputation score for users and posts explains the algorithm used to calculate a user's reputation score and the algorithm used to calculate the reputation score of a post.
- **Section VI:** Implementation explains the various algorithms used in the proposed OSN.
- **Section VII:** Testing explains the test cases written to test the proposed application.
- **Section VIII:** Results explains the Ganache events emitted and the user interface of the proposed application.
- **Section IX:** Conclusion wraps up the paper's findings.

TABLE 1. Abbreviations.

Term	Abbrevation
CDN	Content Delivery Network
ERC	Ethereum Request for Comment
IEEE	Institute of Electrical and Electronics Engineers
IPFS	InterPlanetary File System
NFT	Non-Fungible Token
OSN	Online Social Network
P2P	Peer-to-Peer
PoS	Proof of Stake
PoW	Proof of Work

A visual representation of the paper's organization is depicted in Figure 1, and Table 1 includes the abbreviations used throughout the paper. This structured approach aims to provide a thorough and organized exploration of the complex intersection between blockchain, NFTs, and social media applications.

II. BACKGROUND

This section provides an overview of blockchain and its usage. Furthermore, a discussion on NFT is also provided along with a discussion about OSN and their limitations.

A. BLOCKCHAIN AND ITS USES

First introduced by Bitcoin in 2008 [3], the main idea was to encrypt binary sequences in electronic files using a cryptosystem that would guarantee their non-precedence and resistance to manipulation [4]. Blockchain technology, or cryptographically secure distributed ledger technology, has attracted a lot of interest from academic and scientific communities lately. It makes it easier for data and transactions to be securely and irrevocably recorded over a decentralised network of servers [5], [6], [7]. A block is formed from a collection of transactions on the blockchain and forwarded to each other node, which serves as a validator node, in the manner of a distributed ledger [8]. The validator nodes come to an agreement and validate the block through various algorithms, namely Proof-of-Work (PoW), Proof-of-Stake (PoS), etc. [4], [9]. The block is appended to the blockchain once consensus is reached.

Every block in blockchain mainly comprises of two parts. They are the block header and the block body [2]. The metadata is present in the block header, consisting of block version, timestamp, nonce, Merkle tree root hash and parent block hash. The block body consists of all the transactions that were to be added into the block and the transaction counter. The very first first that is present is called the Genesis Block [10]. Typically, the genesis block has hardcoded

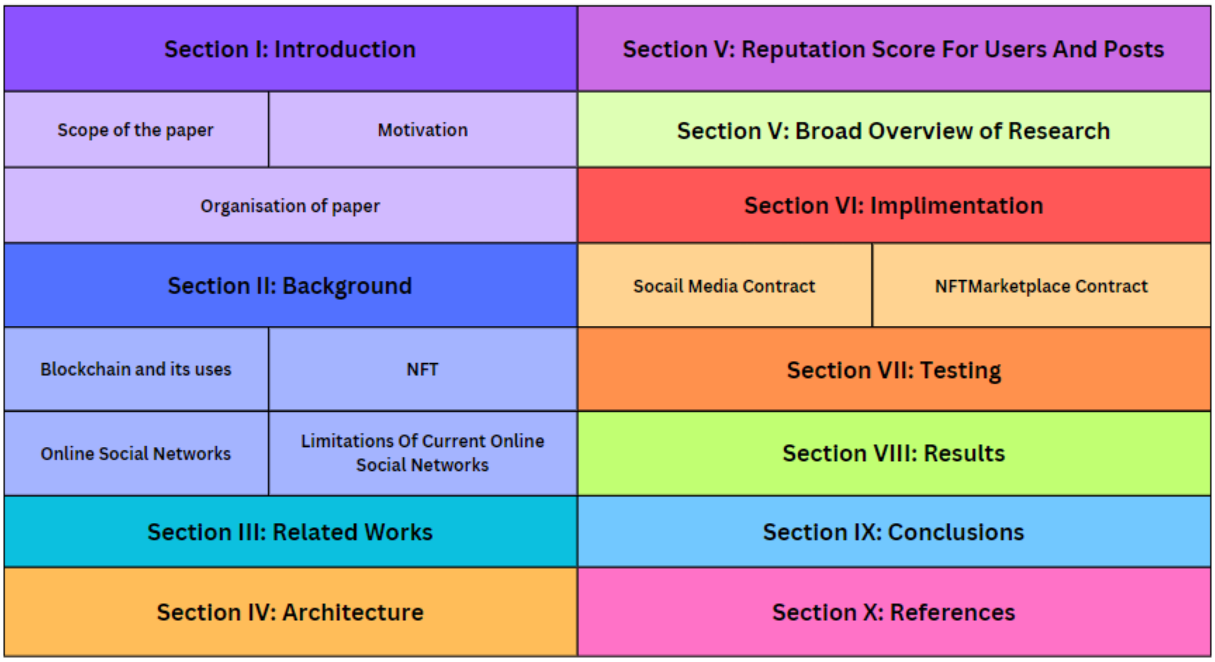


FIGURE 1. Organisation of the paper.

information defining key features of the blockchain, such the name of the network, its objectives, and occasionally an author's statement. Block header, hash, timestamp, and other blockchain-related properties are included, along with Merkle root, nonce value, and other features.

Transactions or data records are difficult to alter or delete once they have been added to the blockchain. This is because each block contains the hash value of the previous block [11]. Hence, modifying one of these blocks changes its hash which can be easily detected by subsequent blocks.

Every node stores a copy of the entire blockchain. This increases the transparency of data and its usage. The number of blocks downloaded by each node in a normal blockchain system can vary depending on several factors like blockchain type, size, specific blockchain protocol, and the node's role within the network. Some nodes might lack the resources to store a copy of the entire blockchain. In such cases, mobile nodes connect to a nearby node with a copy of the blockchain and, in turn, only store the header values of each block. Though the nodes usually do not declare their identities, the transactions are visible, which promotes trust among the participants.

It is possible to facilitate easy interaction between two parties by utilizing blockchain technology through a peer-to-peer (P2P) model, eliminating the need for a third party. Using a P2P protocol, blockchain enables all members of a network to share contacts, allowing agreement on a consensus method [4].

Beyond its initial association with cryptocurrencies, blockchain technology finds applications in a wide range of fields. Numerous industries could benefit from the introduction of blockchain, including supply chain management, financial services, healthcare, real estate, voting systems, and many more. Blockchain can improve security, expedite operations, save costs, and promote better efficiency and confidence in transactions by avoiding the need for intermediaries [12].

B. NFT

A significant development in blockchain technology, NFTs have redefined digital ownership and identity [13]. Non fungible tokens (NFTs) are unique digital assets that function as proof of ownership or legitimacy for a particular good. Any of these items—a tweet, an artwork, a collectible, a virtual location—might be eligible [14], [15]. Unlike fungible cryptocurrencies like Bitcoin or Ethereum, which can be exchanged one-to-one, NFTs are indivisible and have certain qualities that make them irreplaceable and singular [16]. Tokenization is essential for stopping fraud and protecting NFTs from being duplicated or falsified. As each token is unique and can be traced back to its originator, the digital asset's legitimacy and trustworthiness are increased. Investors and collectors benefit from this increased value [17].

Moreover, smart contracts streamline royalty payments to creators whenever NFTs are resold. This ensures fair compensation for artists as their creations appreciate in worth. Ultimately, NFT tokenization transforms the buying, selling, and ownership of digital assets. It establishes a secure, transparent platform for creators and buyers to transact, safeguarding the integrity and authenticity of the digital art market.

NFTs may use permissioned, public, or open blockchain networks as their foundational technology. The ownership records linked to NFTs can be made transparent, unchangeable, and secure in this way. Each NFT includes an individual digital signature that may be used to confirm its ownership, authenticity, and metadata [16]. To standardize non-fungible tokens, this paper uses ERC721 to define an API interface for smart contract implementation on the Ethereum blockchain and to outline the necessary functionalities for such a contract [18].

The capacity of NFTs to provide a mechanism for confirming the ownership and proof of digital assets is one of its main advantages. An NFT's complete transaction history and ownership changes may be traced and validated publicly using blockchain technology, promoting transparency and lowering the possibility of fraud or counterfeiting [13]. This particular feature has revolutionized the art sector, as NFTs allow artists to tokenize and sell their digital works to customers directly, bypassing intermediaries [19]. An NFT collection dubbed CryptoPunks is one example of this. A collection of 10,000 distinct digital items measuring 24 by 24 are known as CryptoPunks, and they are offered for sale on the Ethereum blockchain [16]. They appear on a variety of platforms, including Bloomberg, CNBC, and the New York Times. 1.1 million ETH, or 2.79 billion USD, had been sold overall as of the time the research was drafted, with the lowest valued NFT in the collection being 42 ETH, or 139,212.78 USD [20], [21]. Certain considerations pertain to the environmental consequences of blockchain networks, especially regarding energy usage, which is substantial. Additionally, matters concerning copyright, intellectual property rights, value, and the enduring viability of NFT markets will continue to be subjects of exploration in forthcoming research [16].

C. ONLINE SOCIAL NETWORKS

Online Social Networks, commonly referred to as social media platforms, are internet-based services that allow individuals to create profiles, establish connections with others, and share various types of content. These platforms have revolutionized global communication and interaction [22]. They serve as a primary medium for information dissemination, opinion exchange, and personal sharing [23]. At the heart of Online Social Networks lie connectivity and social engagement. Users can form networks, known as "friends," "followers," or "connections," with other users on the platform. These connections may be symmetric or asymmetric, depending on the specific platform. For instance, Facebook employs symmetric connections, reflecting mutual friendship ties where users establish bi-directional connections indicating mutual agreement. On the other hand, Instagram's network structure enables unilateral linkages between users through asymmetric connections. This design feature allows users to connect with others unilaterally, without necessitating reciprocity or consent. Through these connections, users can engage in various forms of communication, including sharing posts, commenting, liking, and messaging other users. Posts on these platforms encompass personal updates, photos, articles, news, videos, and other multimedia content. The real-time sharing and consumption of content have positioned Online Social Networks as vital sources of information, entertainment, and news for a vast user base. Network effects are another crucial aspect of these platforms. The network's value grows as more users join, expanding the pool of potential connections and content. This, in turn, boosts engagement, interaction, and the overall growth of Online Social Networks. These dynamics have fueled the success and widespread adoption of platforms like Facebook, Twitter, Instagram, and LinkedIn.

Despite their numerous benefits, Online Social Networks present several challenges and considerations. Due to their centralized structure, data can be accessed, traded, or misappropriated without the owner's direct oversight. The Cambridge Analytica scandal stands out as a prominent example involving Facebook, a leading social network. Approximately 87 million users engaged with a Facebook application designed to collect data from users of the app and their friends' profile information. Cambridge Analytica leveraged this data for political purposes, sparking widespread concern. Such incidents underscore the privacy risks associated with contemporary Online Social Networks [24], [25]. The collection, utilization, and potential misuse of user data have attracted scrutiny and ongoing discourse. Issues like misinformation, the impact of these platforms on mental health and well-being, among others, have prompted significant societal and ethical deliberations.

D. LIMITATIONS OF CURRENT ONLINE SOCIAL NETWORKS

OSNs have greatly impacted our daily lives, yet they come with limitations. Some of these are outlined below:

1) CENTRALIZED DATA STORAGE

Data in current OSNs is stored centrally, making it vulnerable to server outages and cyber-attacks, leading to data theft and leaks [26].

2) LIMITED USER CONTROL

Users lack ownership of their published data, resulting in reduced value of original content and potential censorship by organizations [26].

3) CHALLENGE IN CONTROLLING FAKE NEWS AND HARMFUL CONTENT

The rapid spread of fake news due to the lack of ownership associated with posts makes it difficult to prevent dissemination [27].

4) PRIVACY CONCERNS

OSNs pose a significant threat to privacy as platforms and third parties can collect and misuse personal information without user consent [28].

5) USER REVENUE

Users have limited opportunities to earn revenue from their data, as traditional social media platforms predominantly benefit from advertisements [29].

III. RELATED WORKS

Chen and Cho [23], have proposed a decentralized blockchain-based social network. The Tendermint platform has been used as the blockchain platform and IPFS as the storage. The architecture here has 3 main parts which are - the user application, blockchain and storage. Users can communicate with the blockchain through the user application. Wallet handles all security-related tasks, including managing and storing private keys and signing transactions. Private keys are kept on the user's device. When a transaction is received by the blockchain, it first verifies that it is a tweet action before immediately deducting a predetermined amount of tokens from the user to stop resource misuse and spam tweets. The tweet content is then posted to IPFS, and the CID (which is a cryptographically-generated name) and information like user address, tweet title, and gas used are stored in the blockchain for further querying. The publication of tweets ends when nodes commit the transaction, at which point the user receives the outcome. Every tweet has a hash ID that allows it to be uniquely identified in the blockchain and managed further. Comments on the tweet can be published using the hash ID of the tweet. Likes and dislikes on the tweets can be done using upvotes and downvotes. Also, there is a Decentralised Autonomous Organisation (DAO) structure which has two parts - the management team and the users. The management team members are selected at regular intervals and the users vote to select the members of the management team. The users select the management team through voting and operations like tweets and comments management and proposals for DAO operation rule changes are done by the management team. However, the proposals are voted on by all the users. The proposal to delete a tweet can be brought up and put to vote. If the number of votes for the proposal is more than the ones against it, the tweet is deleted automatically. A user can also be impeached using the voting process. One of the future scopes as mentioned in this study is that a user-friendly interface has to be implemented instead of a command line interface.

Authors in [26] propose a blockchain-based architecture for social media networks. Authors suggest that employing a blockchain-based solution makes the system more efficient. The framework consists of three layers. The first layer is the blockchain layer which functions as a control service for OSN, supporting various blockchain types based on application needs. Data is transparent but encrypted for privacy, with a hash value ensuring integrity and basic identification information stored in the blockchain, while detailed user data resides in the storage layer for efficiency. The second layer is the application layer which serves as a user-centric client, facilitating interaction with the blockchain and storage layers for online social activities. Comprising modules like blockchain manager (BM), user manager (UM), cache manager (CM), interaction manager (IM), and storage manager (SM), it manages communication with the blockchain layer, user registration, encryption/decryption operations, and network communications. The CM optimizes efficiency by caching data locally, allowing encrypted storage in the storage layer, and enhancing the user experience during platform or device transitions. The last layer is the storage layer, which utilizes a distributed storage service from third-party providers and ensures resilience against single points of failure. It stores encrypted user data, supports key-value storage similar to DHT and Dynamo, and employs attribute-based fine-grained encryption for varied friend permissions, guaranteeing data authenticity and integrity through hash values. The authors make use of two contracts, a User Registration Contract (URC) and a User Management Contract (UMC). URC handles user registrations and UMC handles details such as friends, posts etc. Every user has her own UMC, which gets created when the user registers into the platform for the first time.

The authors in [30] present a decentralized reputation system for the users in their proposed NFT Marketplace. The value ranges from 0 to 5 and every user initially has a value of 0. If a seller's reputation is below 2.5, then they would have to pay collateral to be able to make any new purchases on the NFT Marketplace. This collateral is returned after the user's reputation reaches 2.5 or above. Buyers that have a low reputation will also have reduced visibility of NFTs that have been put up for sale or auction. This is done as an incentive for people to be honest and gradually increase their reputation score in the marketplace. Users rate each other for every transaction of NFT sale. Every new rating received by a user is used to recalculate the reputation score, i.e., update the existing reputation score.

IV. ARCHITECTURE

Figure 2 explains the flow of social media combined with blockchain. It also explains the integration of the NFT marketplace with the social media application. Any content created by a user is stored in a decentralized storage like IPFS, which in turn communicates with a content delivery network (CDN) and provides a hash value for the content stored. A user's registration and login happens through a wallet which stores a pair of keys for authentication. The hash generated for each of the uploaded posts by the decentralized storage is stored on the blockchain, which is tamper-proof. Hence, the content created by the user cannot be tampered with by other persons. These posts can be later retrieved by other users for viewing, liking and disliking them in the same

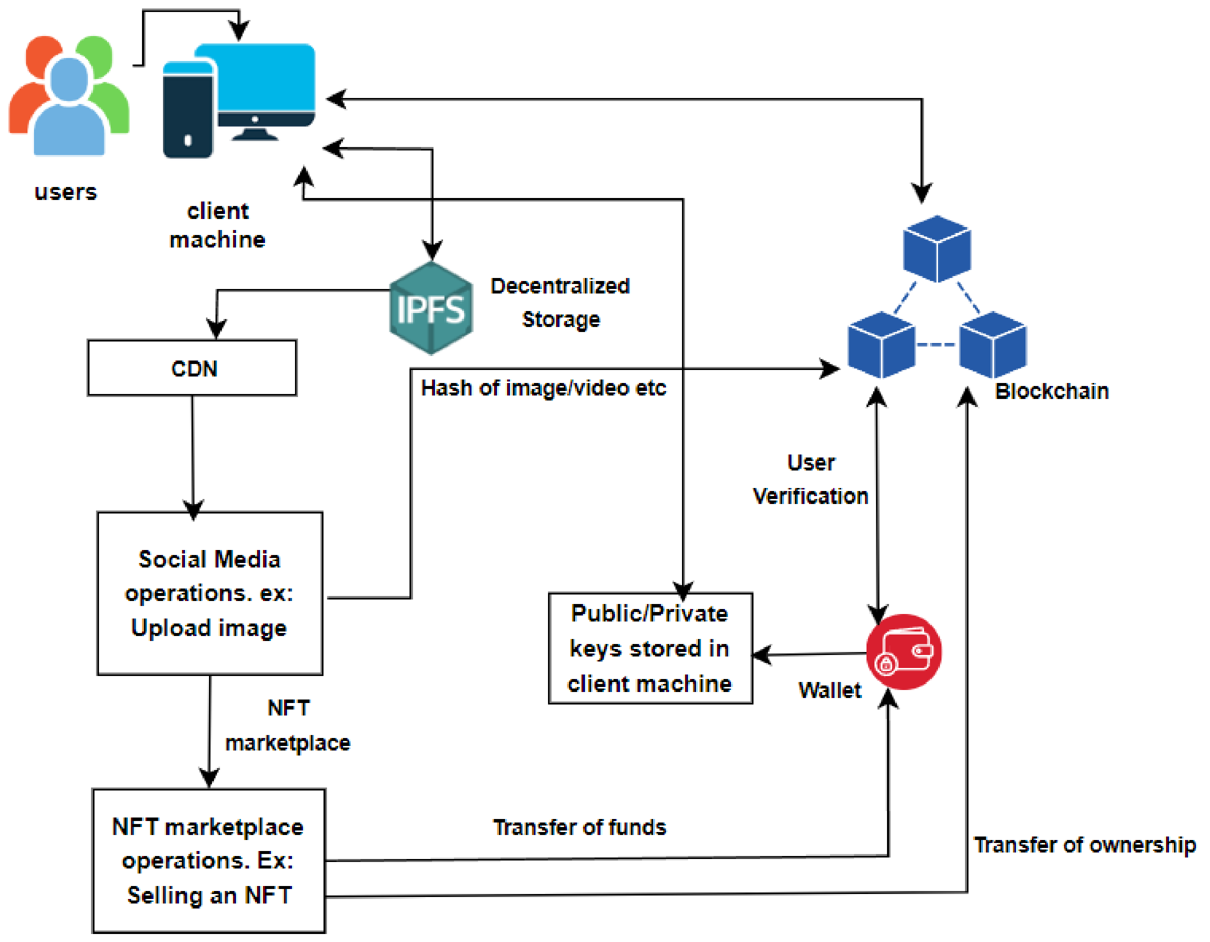


FIGURE 2. Proposed architecture.

way as in the current social media applications. Whenever the post is liked or disliked, it is stored on the blockchain as a new transaction and the necessary changes are made in the contract. On the NFT marketplace, the user can convert their posts into NFTs which can be then sold on the marketplace to other users on the same platform. These NFTs can also be resold on the application after it has been sold by the original post owner.

Figures 3 and 4 explain the sequence of operations that occur in the application, both the social media and the NFT marketplace. The first step that a user must take is to register for the application. On registering, the registration data is stored in the registration contract. Next, the management contract is created for the user.

Whenever the user uploads a post to the application, it is stored in IPFS and the hash of the post is stored in the user's management contract so that the post can be retrieved later to view the post by all users.

The user can also like/dislike other posts uploaded by all the users on the application. When a user likes/dislikes a post, the management contract of the post's user will be updated and the new reputation for the post, its user and the users who have liked/disliked will be calculated.

The users can convert their posts into NFTs and sell them on the marketplace. To do this, the user sends a request to the marketplace smart contract to create a token. The marketplace smart contract responds with the NFT of the post. The NFT is stored in the IPFS and the hash of the NFT is provided back to the user to access the NFT in the future. Then the NFT marketplace also receives the hash to display the new NFT that has been put up for sale. When a buyer wants to buy the NFT on sale in the marketplace, the user must transfer the tokens to the NFT marketplace to receive the NFT. These tokens are then transferred to the seller of the NFT. After the successful transfer of tokens from the buyer to the seller, the buyer is granted ownership of the NFT.

V. REPUTATION SCORE FOR USERS AND POSTS

The Reputation Score is a value assigned to every user to determine the user's quality of activity in the application. It helps the other users to determine how good the user is while buying NFTs from that user. This is done with the aim of incentivizing engagement of the users in both social media and the NFT marketplace areas of the application. The user reputation can change based on three different user actions i.e., liking or disliking a post, creating a post which others

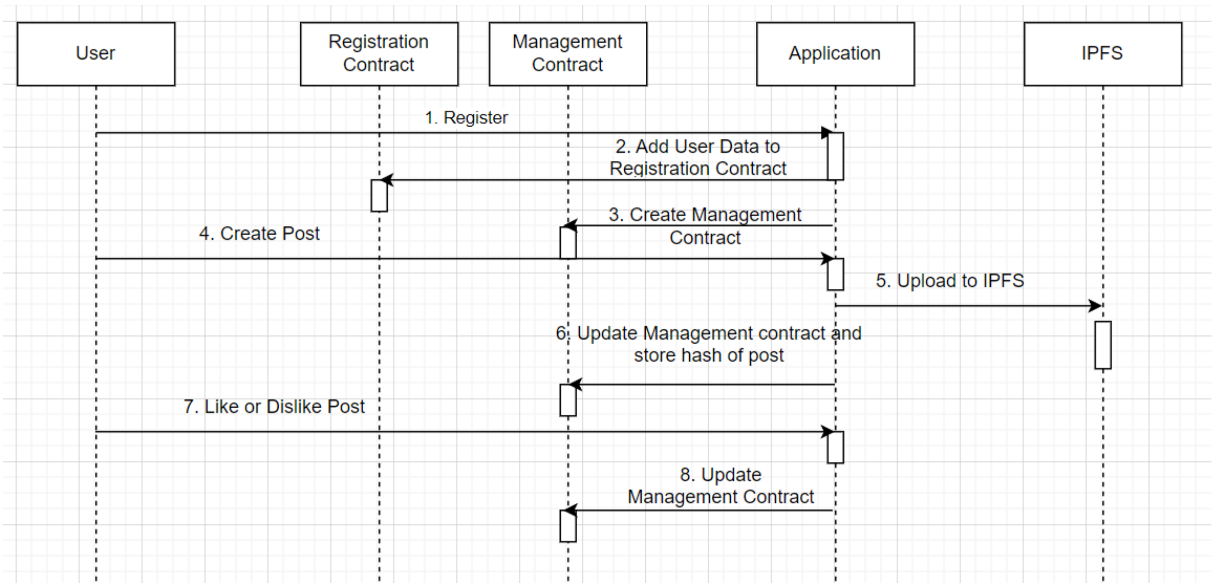

User
Registration Contract
Management Contract
Application
IPFS
1. Register
2. Add User Data to Registration Contract
3. Create Management Contract
4. Create Post
5. Upload to IPFS
6. Update Management contract and store hash of post
7. Like or Dislike Post
8. Update Management Contract

FIGURE 3. Sequence diagram for social media workflow.

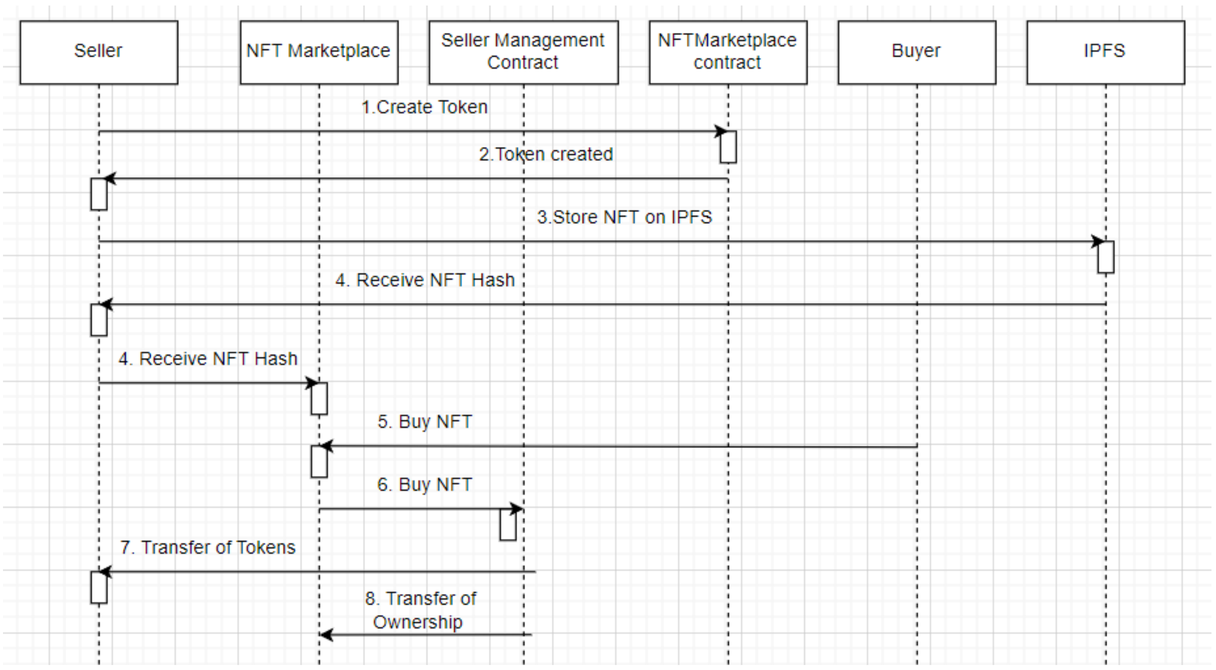


FIGURE 4. Sequence diagram for NFT Marketplace workflow.

will like or dislike and selling NFTs to other users on the application.

As shown in algorithm 1, the post reputation is calculated as the difference between the reputations of the users who have liked the post and the users who have disliked the post. This calculation is only done if the total number of users who have liked and disliked the post crosses the minimum threshold of thirty users. This is done to ensure that the post has statistical significance or not for its reputation to be calculated. This score is added to the post creator's reputation.

The user's reputation can vary based on the posts he/she has liked or disliked. There are two values, a post_ceil value and a post_floor value. If the post's reputation goes beyond post_ceil or below post_floor, the users who have liked or disliked the post will have a change in their reputation value. If the post's reputation goes beyond post_ceil and the user has liked the post, the user reputation will increase

Algorithm 1 Post Reputation

```
Input: posts, postID
1. posts - holds the unique identifier assigned to the post
2. postID - is a mapping of Post to unique identifiers.
Post is a structure with image url (url), number of likes
received (likesCount), number of dislikes received
(dislikesCount), timestamp, owner address (owner),
description of the image (description), handle of the user
(handle) as its attributes
Start
if $len(post.likes) + len(post.dislikes) > 30$ then
    for $like\_user\ in\ post.likes$ do
        $like\_rating \leftarrow like\_rating + like\_user.reputation$
    end
    for $dislike\_user\ in\ post.dislikes$ do
        $dislike\_rating \leftarrow$
        $dislike\_rating + dislike\_user.reputation$
    end
    $return\ like\_rating - dislike\_rating$;
else
    $return$ 0
end
End
```

by 0.5. For the same post, if a user has disliked it, then the user reputation will reduce by 0.5. Similarly, if the post's reputation goes below post_floor and the user has disliked the post, the user reputation will increase by 0.5. For the same post, if a user has liked it, then the user reputation will reduce by 0.5. The same can be seen in algorithm 2.

Reputation can be gained by selling NFTs of the posts uploaded on social media. When another user buys a NFT, the seller of the NFT's user reputation will increase based on the given formula.

$$\begin{aligned} NFT\_rep &= 0.5 \cdot usr.rep + 0.3 \cdot post\_rep(post) \\ &\quad + 0.2 \cdot t\_map(s, b) \end{aligned}$$

The formula depends on three things, the reputation of the seller, the reputation of the post and the time difference between the previous transaction between the buyer and the seller on the NFT marketplace. The time difference is mapped to certain values as mentioned below. If the value of NFT_reputation goes below zero, the value will be made zero. This is done to act as an incentive to join the NFT marketplace and buy/sell NFTs. Algorithm 3 shows the calculation of the user's marketplace reputation.

The final user reputation is the sum of all the three above factors. All three factors decide the final user reputation. There are no floor or ceiling values for the user reputation. This is done to ensure fairness in the value and avoid artificially limiting or inflating the values. Having a floor or ceiling value might lead to users' activity not reflecting in their reputation score. This will also make users create a lot of posts, like/dislike accurately and sell NFTs to have a higher user reputation when compared to other users, creating a friendly competitive environment. Thus, it leads to the growth of a healthy and authentic community within the application.

Algorithm 2 User Post Reputation

```
Input: posts, posted, user
1. posts - holds the unique identifier assigned to the post
2. postID - is a mapping of Post to unique identifiers.
Post is a structure with image url (url), number of likes
received (likesCount), number of dislikes received
(dislikesCount), timestamp, owner address (owner),
description of the image (description), handle of the user
(handle) as its attributes
3. user - holds the Ethereum address of the user
Start
$post\_val \leftarrow post\_reputation(posts, postID)$
$reputation \leftarrow 0$ if
$len(post.likes) + len(post.dislikes) < -20$ then
    if $user\ is\ in\ post[postID].dislikes$ then
        $reputation \leftarrow reputation + 0.5$
    else
        $reputation \leftarrow reputation - 0.5$
    end
end
if $len(post.likes) + len(post.dislikes) > 20$ then
    if $user\ is\ in\ post[postID].likes$ then
        $reputation \leftarrow reputation + 0.5$
    else
        $reputation \leftarrow reputation - 0.5$
    end
end
$return\ reputation$;
End
```

Algorithm 3 User Marketplace Reputation

```
Input: posts, postID, seller, buyer
1. posts - holds the unique identifier assigned to the post
2. postID - is a mapping of Post to unique identifiers.
Post is a structure with image url (url), number of likes
received (likesCount), number of dislikes received
(dislikesCount), timestamp, owner address (owner),
description of the image (description), handle of the user
(handle) as its attributes
3. seller - holds the Ethereum address of the user selling
the NFT
4. buyer - holds the Ethereum address of the user buying
the NFT
Start
$reputation \leftarrow 0.5 * seller.reputation + 0.3 *$
$post\_reputation(posts, postID) + 0.2 *$
$time\_mapping(seller, buyer)$ if $reputation < 0$ then
    $return$ 0;
end
$return\ reputation$;
End
```

TABLE 2. Time mapping values.

Time	Value
0 - 1 hours	0
1 - 3 hours	1
3 - 6 hours	2
6 - 12 hours	3
12 - 36 hours	4
> 36 hours	5

VI. IMPLEMENTATION

A. SOCIAL MEDIA CONTRACT

1) REGISTRATION OF THE USERS

Algorithm 4 Registration of the Users

```
Input: handle, owner, caller, socialMediaAddress, usermap, handlemap
1. handle - holds the handle of the new user to be set
2. owner - holds the Ethereum address of the new user
3. caller - holds the Ethereum address of the function caller
4. socialMediaAddress - holds the Ethereum address of the Social Media Contract
5. usermap - holds the mapping of the encoded Ethereum address to the users
6. handlemap - holds the mapping of the handle to the users
Start
if $caller = socialMediaAddress$ then
    $UID \leftarrow keccak256encode(owner)$;
    if $UIDnotincurrentuserslist$ then
        $mc \leftarrow$ new instance of Social Media Management contract;
        $usermap[UID] \leftarrow User(UPubK : owner, handle : handle, contractPubK : address(mc))$;
        $handlemap[handle] \leftarrow User(UPubK : owner, handle : handle, contractPubK : address(mc))$;
    end
else
    Preview the error with the appropriate message and restore the previous contract state
end
if usermap[UID] is not set then
    Preview the error with the appropriate message and restore the previous contract state
end
End
```

Algorithm 4 manages the registration of new users. The keccak256 algorithm is used to obtain the hash of the new user's Ethereum address, which serves as a key in the user mappings. A new user is created using a new instance of the social media management contract. Each user has a unique instance of the social media management contract created and mapped. The user's Ethereum address, handle, and the address of the Social Media Management Contract are stored in the arguments UPubK, handle, and contractPubK respectively. If the user map is not set or the caller of the function is not the Social Media Contract, an error occurs and the contract's state is restored to its original state.

2) RETRIEVE USER INFORMATION FROM PUBLIC ADDRESS

Algorithm 5 Retrieve the User Information Using the user's Public Address

```
Input: caller, socialMediaAddress, usermap, owner
1. caller - holds the Ethereum address of the function caller
2. socialMediaAddress - holds the Ethereum address of the Social Media Contract
3. owner - holds the Ethereum address of the new user 4. usermap - holds the mapping of the encoded Ethereum address to the users
Start
if $caller = socialMediaAddress$ then
    $UID \leftarrow keccak256encode(owner)$;
    if UID in usermap then
        return (usermap[UID].UPubK, usermap[UID].handle, usermap[UID].contractPubK)
    else
        Preview error as user not registered with the given user address
    end
else
    Preview error with an appropriate message
end
End
```

This algorithm is designed to access user information associated with the user's Ethereum address. If the caller is not the Social Media Contract, an error message will be displayed. Once the specified condition is satisfied, the UID (which is the hash of the Ethereum address) is utilized to retrieve the user's information from the user map. Should the user not be registered, a pertinent message will be presented.

3) RETRIEVE USER INFORMATION FROM USER HANDLE

This algorithm above is designed to retrieve user information based on the user's handle. If the caller is not the designated Social Media Contract, an error message will be displayed. Once the specified condition is satisfied, the user's handle will be utilized to access their details from the handle map. Should the user not be registered, a pertinent message will be presented to the user.

4) UPLOADING A POST

The algorithm "Upload a Post" can be logically segregated into three parts. In the first part, we increment the *postCount* counter, following which we create a new variable of type *post* and populate it with *hashValue* of the post, *description*, *user handle*, *owner address* and *postCount* which are taken as arguments to the function. Finally, the posts mapping is updated and a new entry is made.

Algorithm 6 Retrieve the User Information Using User Handle

```
Input: caller, socialMediaAddress, handlemap, handle
1. caller - holds the Ethereum address of the function caller
2. socialMediaAddress - holds the Ethereum address of the Social Media Contract
3. handle - holds the handle of the new user to be set
4. handlemap - holds the mapping of the handle to the users
Start
if $caller = socialMediaAddress$ then
    if UID in handlemap then
        return (handlemap[handle].UPubK, handlemap[handle].contractPubK )
    else
        Preview error as a user with handle not registered
    end
else
    Preview error with an appropriate message
end
End
```

Algorithm 7 Upload a Post

```
Input: postCount, hashValue, time, description, handle, owneraddr
1. postCount holds the value of the total number of posts on the social media
2. hashValue holds the hash value returned by the IPFS after the image/video is uploaded
3. time hold the timestamp when the post was uploaded
4. description holds the description of the post given by the user
5. handle holds the social media handle of the user uploading the post
6. Owneraddr holds the metamask wallet address of the user uploading the post
Start
    $postCount \leftarrow postCount + 1$;
    $newPost$ ←new Instance of type post passing hashValue, time, description, handle,owneraddr as attributes
    $posts[postCount] \leftarrow newPost$;
End
```

5) LIKING A POST

This algorithm facilitates users in expressing their appreciation and support by liking a post. It requires the input of a postID, which serves as a unique identifier for the post. The posts are stored in a map where each post is associated with its respective identifier. A post is represented by a structure that contains additional information about the post. The algorithm functions by accessing the posts map using the provided postID to locate the specific post, and subsequently incrementing its likesCount by 1 to register the user's likes.

Algorithm 8 Liking a Post

```
Input: postID, posts
1. postID holds the unique identifier assigned to the post
2. posts is a mapping of Post to unique identifiers. Post is a structure with image URL (URL), number of likes received (likesCount), number of dislikes received (dislikesCount), timestamp, owner address (owner), description of the image (description), handle of the user (handle) as its attributes.
Start
    $posts[postID].likesCount \leftarrow$;
        $posts[postID].likesCount + 1$;
End
```

6) DISLIKING A POST

Algorithm 9 Disliking a Post

```
Input: postID, posts
1. postID holds the unique identifier assigned to the post
2. posts is a mapping of Post to unique identifiers. Post is a structure with image URL (URL), number of likes received (likesCount), number of dislikes received (dislikesCount), timestamp, owner address (owner), description of the image (description), handle of the user (handle) as its attributes.
Start
    $posts[postID].dislikesCount \leftarrow$;
        $posts[postID].dislikesCount + 1$;
End
```

The algorithm outlined above allows users to indicate their disagreement with a post by selecting the dislike option. It requires the input of postID, a distinct identifier for the post, which is stored in a mapping known as Posts. Every post is defined by a structure that includes supplementary details. The algorithm locates the precise post by utilizing the postID from the mapping, subsequently incrementing the dislikesCount attribute by 1 to record the user's dislike.

7) RETRIEVE ALL POSTS

The algorithm "Retrieve all posts" retrieves all posts in the contract. The contract maintains a counter called *postCount* which counts the number of posts. The contract maintains a posts mapping which maps *postCount* with the *post* structure. We then iterate over the posts mapping and populate an array of the type *post*. We then return this array.

B. NFTMARKETPLACE CONTRACT

1) CREATING A TOKEN

The Algorithm takes three inputs, tokenURI; which stores URI of the token, price; which is the price set at the time of creation, and _tokenIds which is counter to count the number of tokens in the marketplace. The new token ID can be calculated by incrementing the global variable which stores the count of the tokens (_tokenIds). The algorithm then calls the mint function defined in the ERC721 contract to mint the

Algorithm 10 Get All Posts

```
Input: postCount, posts
1. postCount holds the value of the total number of posts
on the social media
2. posts is a mapping of Post to unique identifiers. Post
is a structure with image URL (URL), number of likes
received (likesCount), number of dislikes received
(dislikesCount), timestamp, owner address (owner),
description of the image (description), handle of the user
(handle) as its attributes.
Start
    $returnArray[postCount + 1]$ (Array of type Post
    of size postcount+1)
    for $i \leftarrow 0$ to $postcount$ $increment$ by 1 do
        $returnArray[i] \leftarrow posts[i]$
    end
    return returnArray
End
```

Algorithm 11 Create a Token

```
Input: tokenURI, price, _tokenIds
1. tokenURI is the string type URI of the token
2. price is the price of the NFT
3. _tokenIds is the global count of the number of NFTs
in the marketplace
Start
    $\_tokenIds \leftarrow \_tokenIds + 1$;
    Call mint function of the ERC721 contract
    Call setTokenURI function of the ERC721
    contract
    return newTokenId
End
```

token. Lastly, the algorithm executes Algorithm 2 to create a market item after calling the setTokenURI function defined within the ERC721 contract.

2) CREATING A MARKET ITEM

Algorithm 12 Create Market Item

```
Input: price, newTokenId
1. price is the price of the NFT
2. newTokenId is the newly created token id
3. idToMarketItem is a mapping of the token id to the
type MarketItem
Start
if $price \geq 0$ then
    $idToMarketItem[tokenId] \leftarrow MarketItem($;
    $tokenId$ :tokenId, $seller$ : sender's address, $buyer$ :
    contract address, $price$:price, $sold$ :false );
    Call the transfer function of the ERC721 contract
else
    Price must be greater than 0 wei
end
End
```

The algorithm is designed to process three inputs: price, idToMarketItem mapping (which establishes the correlation between tokenIDs and the MarketItem structure), and newTokenId (representing the tokenId of the newly generated token). Following validation of the provided price, a new MarketItem structure is generated to capture pertinent token particulars such as ID and price. The seller ID is assigned to the caller's address, while the owner ID is designated as the NFT marketplace contract address. Subsequently, the algorithm triggers the transfer function of the ERC721 contract to effectuate the transfer of NFT ownership. Finally, an event is emitted to signify the successful establishment of the market item.

3) RESELLING A TOKEN

Algorithm 13 Resell Token

```
Input: price, newTokenId, idToMarketItitemSoldCount,
idToMarketItem
1. price is the price of the NFT
2. TokenId is the token id of the NFT
3. itemSoldCount is the count of the number of NFTs
sold
4. idToMarketItem is a mapping of the token id to the
type MarketItem
Start
if $idToMarketItem[TokenId].owner == caller\ of\ the\ contract$ then
    idToMarketItem[tokenId].sold ← false
    idToMarketItem[tokenId].price ←price
    idToMarketItem[tokenId].seller =←caller
    idToMarketItem[tokenId].owner
    ←NFTMarketplace contract address
    itemSoldCount←itemSoldCount - 1
    Call the transfer function of the ERC721 contract
else
    The caller of the contract is not the owner of the NFT
end
End
```

The algorithm requires three inputs: NFT price, Token ID, and a counter to track sold items in the marketplace. It first checks if the caller owns the NFT associated with the provided Token ID. If this condition is met, the algorithm updates market item details by marking it unsold, adjusting the resale price, assigning the caller as the seller, and setting the NFT marketplace contract address as the new owner. The itemSoldCount then decreases. Finally, the algorithm invokes the ERC721 contract's transfer function to transfer NFT ownership.

4) CREATE A MARKET SALE ITEM

The algorithm takes two inputs, a token ID and a counter to count the number of items sold. It starts by retrieving the seller's address from the market item associated with the given TokenId. Then, it updates the market item details, setting the new owner as the caller of the function (the

Algorithm 14 Create a Market Sale

```
Input: TokenId, itemSoldCount, idToMarketItem,
itemSoldCount
1. price is the price of the NFT
2. tokenId is the token id of the NFT
3. itemSoldCount is the count of the number of NFTs
sold
4. idToMarketItem is a mapping of the token id to the
type MarketItem
Start
address seller ← idToMarketItem[tokenId].seller
idToMarketItem[tokenId].owner ← Caller of the
contract
idToMarketItem[tokenId].sold ← true
idToMarketItem[tokenId].seller =←caller of the contract
itemSoldCount←itemSoldCount + 1
Call the transfer function of the ERC721 contract
End
```

buyer), marking the NFT as sold, and designating the seller as the caller. The itemSoldCount is then incremented to reflect the successful sale. Finally, the algorithm calls the transfer function of the ERC721 contract within the seller's contract to complete the transfer of ownership.

5) RETRIEVE UNSOLD MARKETPLACE ITEMS

Algorithm 15 Fetching the Marketplace Items Which Are Unsold

```
Input: itemCount, IDtoMarketItem, itemSoldCount
1. itemCount is the total number of NFT's in the
marketplace
2. itemSoldCount is the count of the number of NFTs
sold
3. idToMarketItem is a mapping of the token id to the
type MarketItem
4. salecount is the number of sold NFT's in the
Markeplace Start
unsolditemcount ← itemCount - salecount
items← new Instance of Marketitem of size
unsolditemcount
for $i \leftarrow 0$ to total number NFT's increment by 1 do
    if idToMarketItem[i + 1].owner == contract
    address then
        currentId ← $i + 1$;
        currentItem of type MarketItem←
        idToMarketItem[currentId];
        items[currentIndex] ← currentItem
        currentIndex ← currentIndex + 1
    end
end
return items
End
```

The function is invoked by the application to display all unsold Non-Fungible Tokens that are currently available in the marketplace. A data structure called MarketItems is instantiated to store these NFTs. Finally, the MarketItems array is retrieved and returned as the output.

6) RETRIEVE ALL SOLD AND UNSOLD NFTS IN THE MARKETPLACE

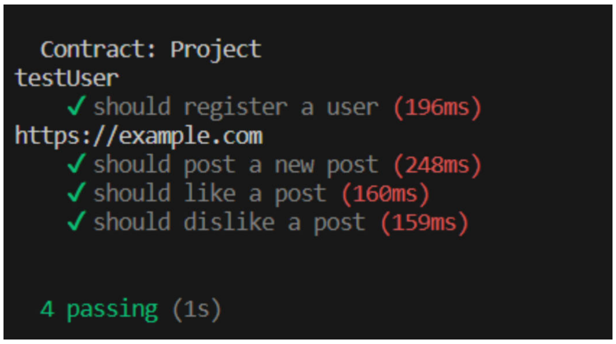
Algorithm 16 Fetch All the Sold and Unsold NFT's in the Marketplace

```
Input: itemCount, IDtoMarketItem
1. itemCount is the total number of NFT's in the
marketplace
2. idToMarketItem is a mapping of the token id to the
type MarketItem
Start
items← new Instance of Marketitem of size itemCount
for $i \leftarrow 0$ to total NFTs increment by 1 do
    currentId ← $i + 1$;
    currentItem of type MarketItem←
    idToMarketItem[currentId];
    items[currentIndex] ← currentItem
    currentIndex ← currentIndex + 1
end
return items
End
```

Algorithm 17 Fetch All the NFT's Owned by the User

```
Input: itemCount, IDtoMarketItem
1. itemCount is the total number of NFT's in the
marketplace
2. idToMarketItem is a mapping of the token id to the
type MarketItem
Start
useritemCount ← 0 for $i \leftarrow 0$ to total number NFT's
increment by 1 do
    if idToMarketItem[i+1].seller == payable(user's
    address) then
        useritemCount ← useritemCount + 1
    end
end
items ← new Instance of Marketitem of size
useritemCount
for $i \leftarrow 0$ to itemCount by 1 do
    if idToMarketItem[i + 1].seller == payable(user's
    address) then
        currentId ← $i + 1$;
        currentItem of type MarketItem←
        idToMarketItem[currentId];
        items[currentIndex] ← currentItem
        currentIndex ← currentIndex + 1
    end
end
return items
End
```

The function is invoked by the application to display all Non-Fungible Tokens available in the marketplace, regardless



```
  Contract: Project
testUser
    ✓ should register a user (196ms)
https://example.com
    ✓ should post a new post (248ms)
    ✓ should like a post (160ms)
    ✓ should dislike a post (159ms)


  4 passing (1s)
```

FIGURE 5. **Test case results for social media contract.**

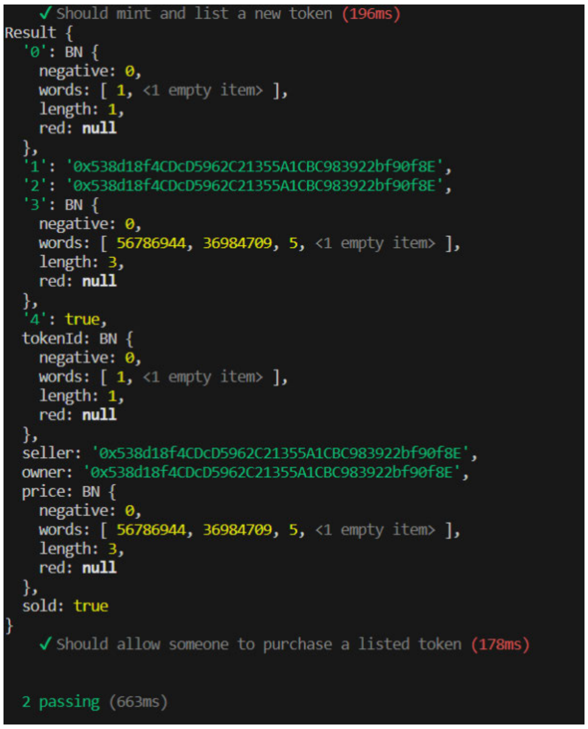
of their sales status. A data structure named MarketItems is initialized to store the NFTs, encompassing all tokens within its collection. Subsequently, the MarketItems array is retrieved and returned as the output.

Algorithm 18 Fetch All the NFT's That the User Has Listed for Sale

```
Input: itemCount, IDtoMarketItem
1. itemCount is the total number of NFT's in the marketplace
2. idToMarketItem is a mapping of the token id to the type MarketItem
Start
$useritemCount \leftarrow 0$ for $i \leftarrow 0$ to $total\ number\ NFT's\ increment$ by 1 do
    if $idToMarketItem[i+1].seller == user's\ address$ then
        $useritemCount \leftarrow useritemCount + 1$
    end
end
$items \leftarrow$ new Instance of Marketitem of size useritemCount
for $i \leftarrow 0$ to $itemCount$ by 1 do
    if $idToMarketItem[i+1].seller == user's\ address$ then
        $currentId \leftarrow i + 1$;
        $currentItem$ of type MarketItem $\leftarrow idToMarketItem[currentId]$;
        $items[currentIndex] \leftarrow currentItem$
        $currentIndex \leftarrow currentIndex + 1$
    end
end
return $items$
End
```

7) RETRIEVE ALL THE NFTS OWNED BY THE USER

The function is invoked by the application to display all the non-fungible tokens acquired by the user in the marketplace. It computes the purchaseCount value for ascertaining the quantity of tokens procured. A MarketItems array is instantiated to store the essential non-fungible tokens, incorporating any unsold tokens. Subsequently, the MarketItems array is returned.



```
    ✓ Should mint and list a new token (196ms)
Result {
  '0': BN {
    negative: 0,
    words: [ 1, <1 empty item> ],
    length: 1,
    red: null
  },
  '1': '0x538d18f4CDcD5962C21355A1CBC983922bf90f8E',
  '2': '0x538d18f4CDcD5962C21355A1CBC983922bf90f8E',
  '3': BN {
    negative: 0,
    words: [ 56786944, 36984709, 5, <1 empty item> ],
    length: 3,
    red: null
  },
  '4': true,
  tokenId: BN {
    negative: 0,
    words: [ 1, <1 empty item> ],
    length: 1,
    red: null
  },
  seller: '0x538d18f4CDcD5962C21355A1CBC983922bf90f8E',
  owner: '0x538d18f4CDcD5962C21355A1CBC983922bf90f8E',
  price: BN {
    negative: 0,
    words: [ 56786944, 36984709, 5, <1 empty item> ],
    length: 3,
    red: null
  },
  sold: true
}
    ✓ Should allow someone to purchase a listed token (178ms)


  2 passing (663ms)
```

FIGURE 6. **Test case results for NFT marketplace contract.**

TABLE 3. **Gas prices for all functions.**

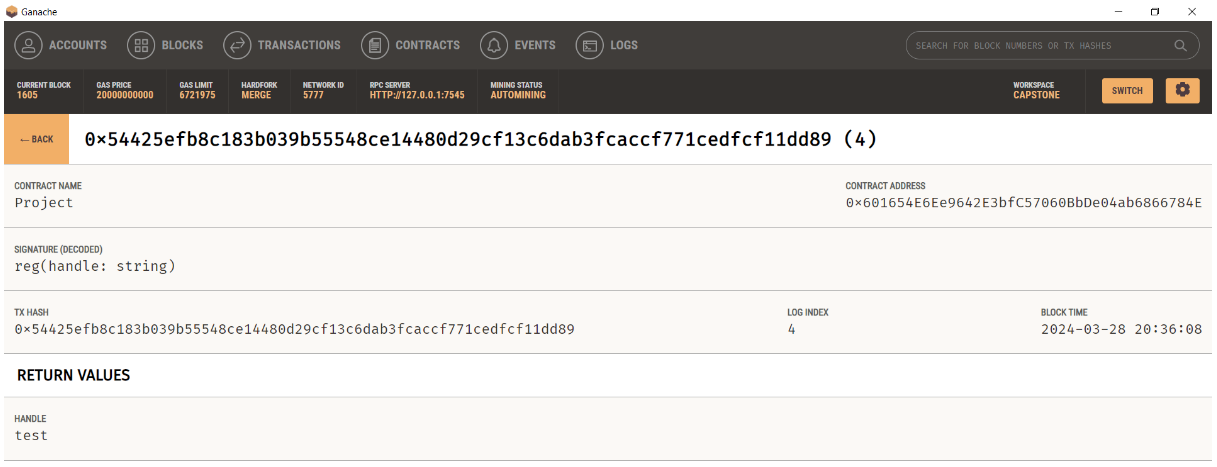
Function	Total Gas Price(in Ethers)
Fetching public wallet address of the user	0.04194304
User Registration	0.00327378
Upload a Post	0.00057302
Like a post	0.00007283
Dislike a post	0.00007272
Create NFT	0.00068571
Convert to NFT	0.00068577
Buy NFT	0.00016531

8) FETCH ALL THE NFT'S THAT THE USER HAS LISTED FOR SALE

The function displays all non-fungible tokens available for sale, calculating saleCount to show the number of tokens for purchase. It initializes a MarketItems array to store essential NFTs, ensuring any unsold tokens are included. The MarketItems array is then returned.

VII. TESTING

To demonstrate the proper working of the functions, we have written test cases that cover most of the functions that have been described. We have used node.js to call the solidity functions using chai and mocha libraries. Some non-functional requirements such as the time taken to run the function are also displayed as the output.



CURRENT BLOCK 1605 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE CAPSTONE

0x54425efb8c183b039b55548ce14480d29cf13c6dab3fcaccf771cedfcf11dd89 (4)

CONTRACT NAME
Project

CONTRACT ADDRESS
0x601654E6Ee9642E3bfC57060BbDe04ab6866784E

SIGNATURE (DECODED)
reg(handle: string)

TX HASH
0x54425efb8c183b039b55548ce14480d29cf13c6dab3fcaccf771cedfcf11dd89

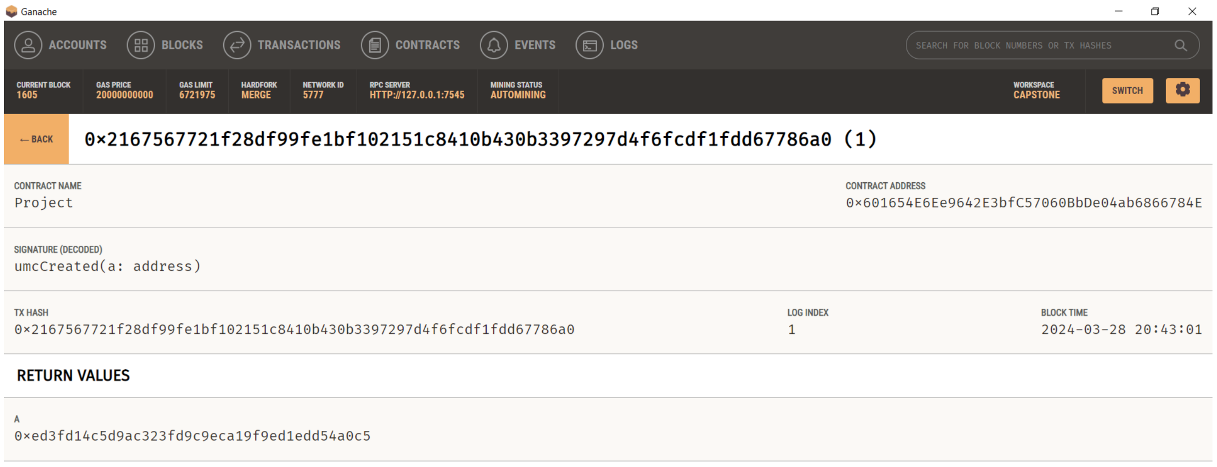
LOG INDEX
4

BLOCK TIME
2024-03-28 20:36:08

RETURN VALUES

HANDLE
test

FIGURE 7. Ganache event for successfully registering a user.



CURRENT BLOCK 1605 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE CAPSTONE

0x2167567721f28df99fe1bf102151c8410b430b3397297d4f6fcdf1fdd67786a0 (1)

CONTRACT NAME
Project

CONTRACT ADDRESS
0x601654E6Ee9642E3bfC57060BbDe04ab6866784E

SIGNATURE (DECODED)
umcCreated(a: address)

TX HASH
0x2167567721f28df99fe1bf102151c8410b430b3397297d4f6fcdf1fdd67786a0

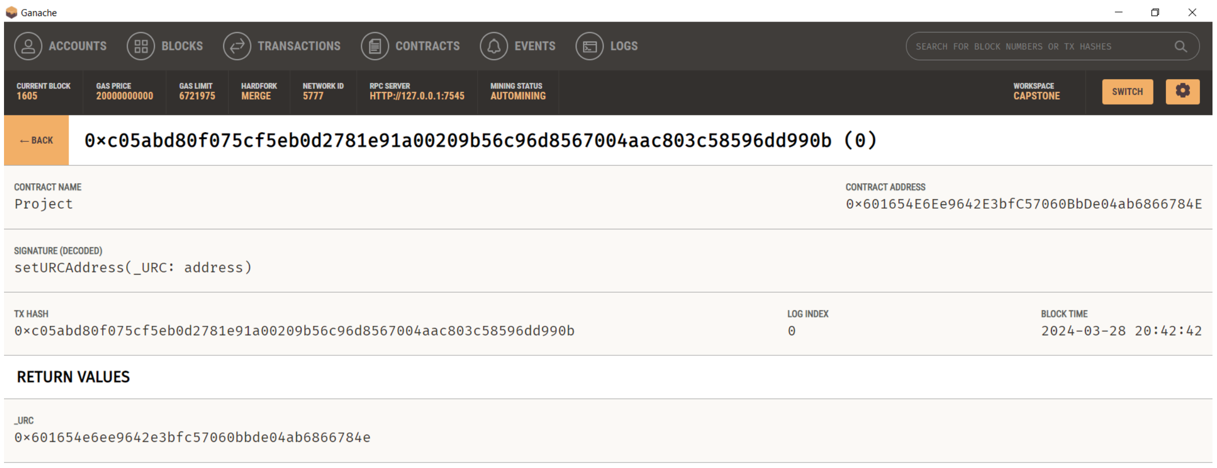
LOG INDEX
1

BLOCK TIME
2024-03-28 20:43:01

RETURN VALUES

A
0xed3fd14c5d9ac323fd9c9eca19f9ed1edd54a0c5

FIGURE 8. Ganache event for UMC contract creation.



CURRENT BLOCK 1605 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE CAPSTONE

0xc05abd80f075cf5eb0d2781e91a00209b56c96d8567004aac803c58596dd990b (0)

CONTRACT NAME
Project

CONTRACT ADDRESS
0x601654E6Ee9642E3bfC57060BbDe04ab6866784E

SIGNATURE (DECODED)
setURCAddress(_URC: address)

TX HASH
0xc05abd80f075cf5eb0d2781e91a00209b56c96d8567004aac803c58596dd990b

LOG INDEX
0

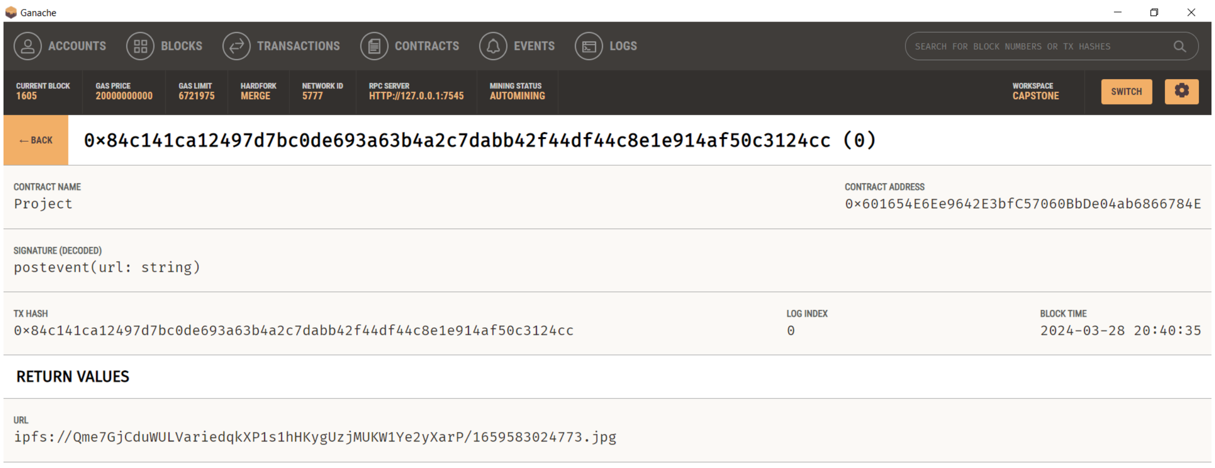
BLOCK TIME
2024-03-28 20:42:42

RETURN VALUES

_URC
0x601654e6ee9642e3bfc57060bbde04ab6866784e

FIGURE 9. Ganache event for setting the URC address.

The project contract, illustrated in Figure 5 includes four test cases. In the first one, we use a dummy handle to register a user by calling the register function in the project contract. We then check if the user is registered by calling Algorithm 3.



CURRENT BLOCK 1605 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE CAPSTONE

0x84c141ca12497d7bc0de693a63b4a2c7dabb42f44df44c8e1e914af50c3124cc (0)

CONTRACT NAME
Project

CONTRACT ADDRESS
0x601654E6Ee9642E3bfC57060BbDe04ab6866784E

SIGNATURE (DECODED)
postevent(url: string)

TX HASH
0x84c141ca12497d7bc0de693a63b4a2c7dabb42f44df44c8e1e914af50c3124cc

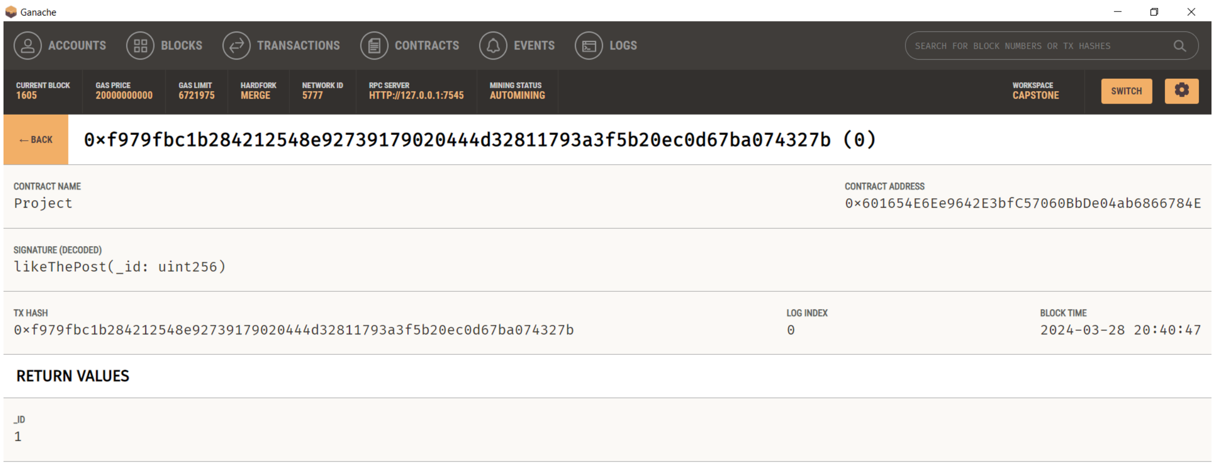
LOG INDEX
0

BLOCK TIME
2024-03-28 20:40:35

RETURN VALUES

URL
ipfs://Qme7GjCduWULVariedqkXP1s1hHKygUzjMUKW1Ye2yXarP/1659583024773.jpg

FIGURE 10. Ganache event for successful upload of a post by a user.



CURRENT BLOCK 1605 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE CAPSTONE

0xf979fbc1b284212548e92739179020444d32811793a3f5b20ec0d67ba074327b (0)

CONTRACT NAME
Project

CONTRACT ADDRESS
0x601654E6Ee9642E3bfC57060BbDe04ab6866784E

SIGNATURE (DECODED)
likeThePost(_id: uint256)

TX HASH
0xf979fbc1b284212548e92739179020444d32811793a3f5b20ec0d67ba074327b

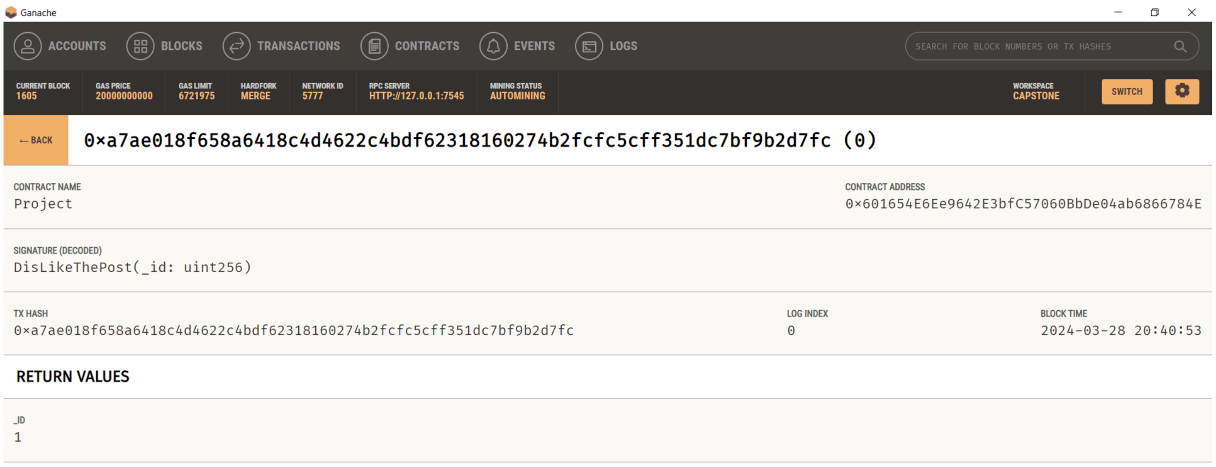
LOG INDEX
0

BLOCK TIME
2024-03-28 20:40:47

RETURN VALUES

_ID
1

FIGURE 11. Ganache event after liking of a post by a user.



CURRENT BLOCK 1605 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE CAPSTONE

0xa7ae018f658a6418c4d4622c4bdf62318160274b2fcfc5cff351dc7bf9b2d7fc (0)

CONTRACT NAME
Project

CONTRACT ADDRESS
0x601654E6Ee9642E3bfC57060BbDe04ab6866784E

SIGNATURE (DECODED)
DisLikeThePost(_id: uint256)

TX HASH
0xa7ae018f658a6418c4d4622c4bdf62318160274b2fcfc5cff351dc7bf9b2d7fc

LOG INDEX
0

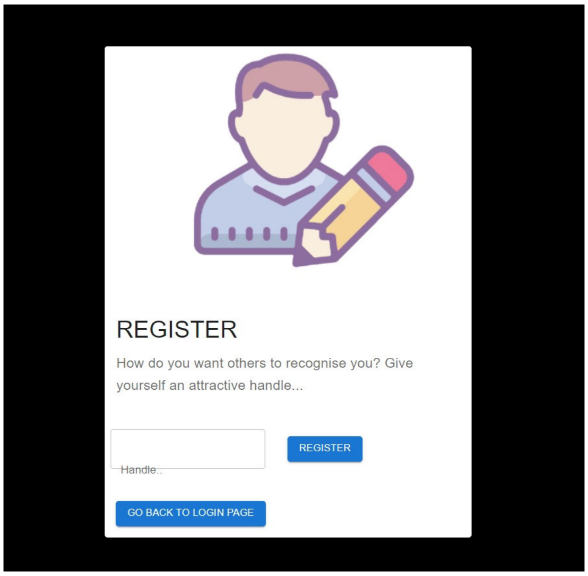
BLOCK TIME
2024-03-28 20:40:53

RETURN VALUES

_ID
1

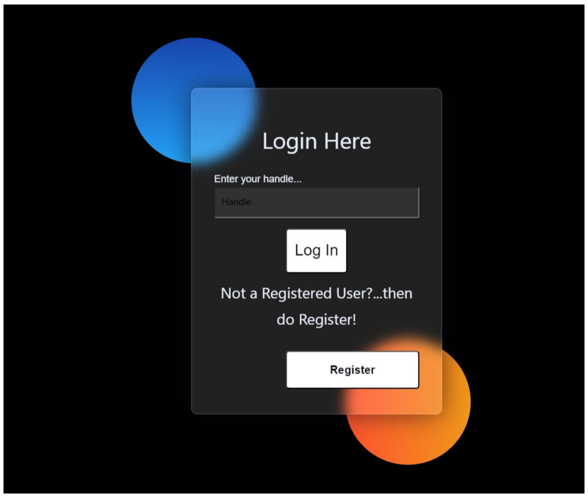
FIGURE 12. Ganache event after disliking a post by a user.

For the second test case, we upload a dummy post to IPFS, get a hash value, and use it, along with other details, to call the post function in the contract. We verify the upload by calling Algorithm 6. In the third test case, we take the post's hash as



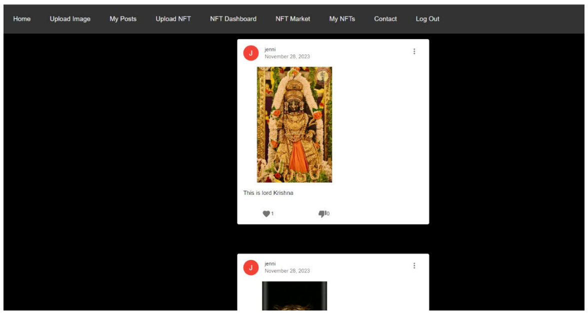
How do you want others to recognise you? Give yourself an attractive handle...

FIGURE 13. Page for registering a new user.



Not a Registered User?...then do Register!

FIGURE 14. Login page.



This is lord Krishna

FIGURE 15. Page for viewing all posts.

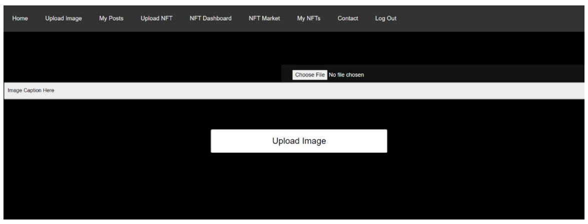


FIGURE 16. Page for uploading a new post.

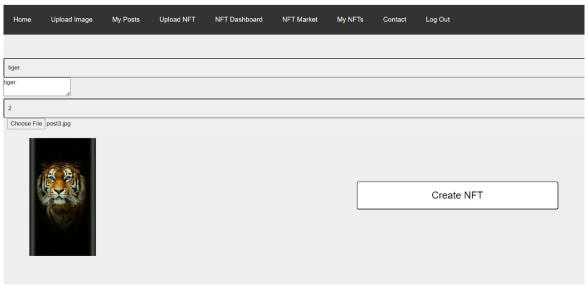


FIGURE 17. Page for creating a NFT.

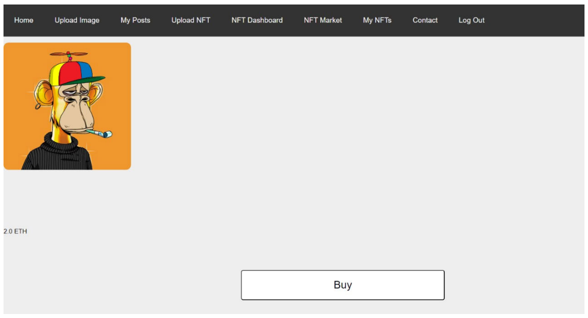


FIGURE 18. Page for buying NFT.

input and call the like function in the contract. We check if the likes count has increased by one. The fourth and final test case involves the dislike function, which performs a similar operation as the like function but decrements the likes count instead.

This contract took one second to run all the functions. The speed can be further improved if side-chains like polygon are used due to their ability to process higher transactions per second.

Figure 6 depicts two functional test cases for the NFT marketplace contract. The first test case creates a new token following Algorithm 11 and creates a Market Item following Algorithm 12. The second test case allows someone to purchase a new token by creating a market sale item following Algorithm 14.

VIII. RESULTS

The results include the events which are recorded on Ganache after each successful operation on the social media application. After an operation is executed, an event is emitted in the code which contains information to verify the operation.

Upon successful registration of a user, an event is emitted as shown in Figure 7, including the user's name in the return values. Subsequently, Figure 8 signifies the successful creation of a unique management smart contract for each user, with the contract address provided in the return values. Concurrently, Figure 9 denotes the emission of an event upon the creation of a Registration smart contract, a global contract shared among all users, with its address included in the return values. Upon successful user post submission, the application triggers an event as shown in Figure 10 containing the URL with the hash of the post stored in IPFS. Furthermore, Figure 11 depicts the event of a user liking a post, providing the updated number of likes in the return values, while Figure 12 shows the event of a user disliking a post, presenting the updated number of dislikes.

When a user opens the applications, they are initially directed to the login page of the application. If the user has an existing account, they can enter their user handle and log in to the application. They are verified on their wallet ID. In case the user does not have an existing account, they can register into the application. The user enters a handle and they are registered into the application with the entered handle and their wallet ID. Figure 13 and 14 show the registration and the login pages respectively.

On successful registration or login, the user will be redirected to the home page where all the uploaded posts in the application can be viewed. Each post will have its uploaded user handle, date of upload, the uploaded image, a caption and its likes and dislikes. On the top bar, the user can choose to view their posts, create a new post, create a new NFT, enter the NFT marketplace, view the user's owned NFTs or log out as seen in Figure 15.

To create a new post, the user must upload the file and enter a caption for the same as seen in Figure 16. Similarly, a new NFT can also be created by selecting the post to be converted into an NFT and entering the asking price for the NFT. This is shown in Figure 17.

In the NFT marketplace, users can view all the available NFTs for sale and the price for each of the NFT. Users can opt to buy any NFT on the marketplace as visible in Figure 18. The user will have to transfer the requested NFT price to the seller's wallet. After a successful transaction, the ownership of the NFT is now transferred to the user who bought the NFT.

Table 3 shows the gas price of various functions in our application. As can be seen functions like a post and dislike a post consume lesser gas price as compared to other functions since these are simple operations which just increment the like and dislike count respectively. For buying a NFT, the user has to pay the price of the NFT along with the gas price associated with that function.

IX. CONCLUSION

In this paper, a comprehensive framework for leveraging blockchain technology to enhance online social networks, with a focus on integrating non-fungible tokens (NFTs) and reputation scores has been presented. Our work addresses the limitations of current online social networks by introducing novel features such as NFT marketplaces and reputation-based user interactions.

Through the implementation and testing of our proposed system, we have demonstrated its feasibility and effectiveness in providing a secure and transparent environment for social media interactions. The use of blockchain ensures data integrity and decentralization, mitigating concerns related to privacy and censorship.

Furthermore, our experiments have showcased the efficiency of the implemented functionalities, with test results indicating successful user registration, post uploads, and interactions with NFTs. The integration of smart contracts has facilitated seamless transactions within the social media ecosystem, enhancing user experience and fostering trust among participants.

However, developing blockchain-based applications comes with it's own set of challenges. Ensuring scalability and performance can be difficult due to the intrinsic slower transaction speeds and the high computational costs associated with blockchain. Also, compliance to regulations across various jurisdictions can be complex, given the evolving legal landscape surrounding blockchain.

Overall, our research contributes to the ongoing discourse on blockchain applications in social networking and lays the foundation for future developments in this domain. By harnessing the power of decentralized technologies, we aim to empower users with greater control over their digital identities and foster a more inclusive and authentic online community.

REFERENCES

[1] S. DeWind, W. Geerling, G. D. Mateer, and K. Halfen, "The economics behind the billions: How Taylor swift and MrBeast can be used to teach economics," *SSRN Electron. J.*, 2023, doi: 10.2139/ssrn.4640484.

[2] B. Guidi, "An overview of blockchain online social media from the technical point of view," *Appl. Sci.*, vol. 11, no. 21, p. 9880, Oct. 2021.

[3] N. S. Bitcoin, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008.

[4] M. A. Hisseine, D. Chen, and X. Yang, "The application of blockchain in social media: A systematic literature review," *Appl. Sci.*, vol. 12, no. 13, p. 6567, Jun. 2022.

[5] R. Vairagade, L. Bitla, H. H. Judge, S. D. Dharpude, and S. S. Kekatpure, "Proposal on NFT minter for blockchain-based art-work trading system," in *Proc. IEEE 11th Int. Conf. Commun. Syst. Netw. Technol. (CSNT)*, Apr. 2022, pp. 571–576.

[6] L. Liu, W. Zhang, and C. Han, "A survey for the application of blockchain technology in the media," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 5, pp. 3143–3165, 2021.

[7] P. Freni, E. Ferro, and G. Ceci, "Fixing social media with the blockchain," in *Proc. 6th EAI Int. Conf. Smart Objects Technol. Social Good*, Sep. 2020, pp. 175–180.

[8] A. Guadamuz, "The treachery of images: Non-fungible tokens and copyright," *J. Intellectual Property Law Pract.*, vol. 16, no. 12, pp. 1367–1385, 2021.

[9] M. Pärssinen, M. Kotila, R. C. Rumin, A. Phansalkar, and J. Manner, "Is blockchain ready to revolutionize online advertising?" *IEEE Access*, vol. 6, pp. 54884–54899, 2018.

[10] T. Poongodi, R. Sujatha, D. Sumathi, P. Suresh, and B. Balamurugan, "Blockchain in social networking," in *Cryptocurrencies and Blockchain Technology Applications*. Hoboken, NJ, USA: Wiley-Scrivener, 2020, pp. 55–76.

[11] A. Qayyum, J. Qadir, M. U. Janjua, and F. Sher, "Using blockchain to rein in the new post-truth world and check the spread of fake news," *IT Prof.*, vol. 21, no. 4, pp. 16–24, Jul. 2019.

[12] P. Christodoulou and K. Christodoulou, "Developing more reliable news sources by utilizing the blockchain technology to combat fake news," in *Proc. 2nd Int. Conf. Blockchain Comput. Appl. (BCCA)*, Nov. 2020, pp. 135–139.

[13] A. C. Moreno, "NFT as a proof of digital ownership-reward system integrated to a secure distributed computing blockchain framework," M.S. thesis, UIS, Springfield, IL, USA, 2022.

[14] J. B. Cho, S. Serneels, and D. S. Matteson, "Non-fungible token transactions: Data and challenges," *Data Sci. Sci.*, vol. 2, no. 1, Dec. 2023, Art. no. 2151950.

[15] M. Rahrouh, W. Alayash, and M. Ghanem, "The potential application of NFT in the publishing industry; opportunities and challenges," in *Proc. Int. Arab Conf. Inf. Technol. (ACIT)*, Nov. 2022, pp. 1–5.

[16] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (NFT): Overview, evaluation, opportunities and challenges," 2021, *arXiv:2105.07447*.

[17] R. A. A. Mochram, C. T. Makawowor, K. M. Tanujaya, J. V. Moniaga, and B. A. Jabar, "Systematic literature review: Blockchain security in NFT ownership," in *Proc. Int. Conf. Electr. Inf. Technol. (IEIT)*, Sep. 2022, pp. 302–306.

[18] S. Casale-Brunet, P. Ribeca, P. Doyle, and M. Mattavelli, "Networks of Ethereum non-fungible tokens: A graph-based analysis of the ERC-721 ecosystem," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Dec. 2021, pp. 188–195.

[19] H. Bao and D. Roubaud, "Non-fungible token: A systematic review and research agenda," *J. Risk Financial Manage.*, vol. 15, no. 5, p. 215, May 2022.

[20] *CryptoPunks*. Accessed: Apr. 2, 2024. [Online]. Available: https://cryptopunks.app/

[21] M. Dowling, "Is non-fungible token pricing driven by cryptocurrencies?" *Finance Res. Lett.*, vol. 44, Jan. 2022, Art. no. 102097.

[22] B. Guidi and A. Michienzi, "The decentralization of social media through the blockchain technology," in *Proc. 13th ACM Web Sci. Conf.*, Jun. 2021, pp. 138–139.

[23] N. Chen and D. S. Cho, "A blockchain based autonomous decentralized online social network," in *Proc. IEEE Int. Conf. Consum. Electron. Comput. Eng. (ICCECE)*, Jan. 2021, pp. 186–190.

[24] B. Guidi, "When blockchain meets online social networks," *Pervasive Mobile Comput.*, vol. 62, Feb. 2020, Art. no. 101131.

[25] H. Berghel, "Malice domestic: The Cambridge analytica dystopia," *Computer*, vol. 51, no. 5, pp. 84–89, May 2018.

[26] L. Jiang and X. Zhang, "BCOSN: A blockchain-based decentralized online social network," *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 6, pp. 1454–1466, Dec. 2019.

[27] W. J. Tee and R. K. Murugesan, "Trust network, blockchain and evolution in social media to build trust and prevent fake news," in *Proc. 4th Int. Conf. Adv. Comput., Commun. Autom. (ICACCA)*, Oct. 2018, pp. 1–6.

[28] R. Ciriello, R. Beck, and J. Thatcher, "The paradoxical effects of blockchain technology on social networking practices," *SSRN Electron. J.*, 2018, doi: 10.2139/ssrn.3920002.

[29] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Secur. Privacy Workshops*, May 2015, pp. 180–184.

[30] H. R. Hasan, K. Salah, A. Battah, M. Madine, I. Yaqoob, R. Jayaraman, and M. Omar, "Incorporating registration, reputation, and incentivization into the NFT ecosystem," *IEEE Access*, vol. 10, pp. 76416–76433, 2022.

SHRUTI JADON received the B.Tech. degree from U.P. Technical University, the M.Tech. degree from Banasthali Vidyapith, and the Ph.D. degree from NIT Allahabad, in 2019.

Currently, she is an Associate Professor with PES University, Bengaluru, India. She has published many papers in various journals and conferences. Her research interests include real-time systems, cryptography, and blockchain.

KARTHIKEYA BHAT is currently pursuing the B.Tech. degree in computer science and engineering with PES University, Bengaluru, India.

He interned with Pronisi LLC, during the summer, in 2023, where he was a Backend Intern. He is a Software Intern with Ansys Software Pvt. Ltd. His research interests include distributed systems, blockchain, and AI.

KARTHIKEYA R. JENNI is currently pursuing the B.Tech. degree in computer science and engineering with PES University, Bengaluru, India.

He interned with Dover India, during the summer, in 2023, where he was a Software Development Intern. He is a Software Intern with Dover India. His research interests include blockchain, cloud computing, and database systems.

KRISHNA VEDANTHA is currently pursuing the B.Tech. degree in computer science and engineering with PES University, Bengaluru, India.

He is a Software Intern with Hakimo AI. His research interests include blockchain, cloud computing, and big data.

LIKITH R R is currently pursuing the B.Tech. degree in computer science and engineering with PES University, Bengaluru, India.

He is a Software Intern with Epsilon. His research interests include blockchain, software development, and database systems.

PRASAD B. HONNAVALLI (Member, IEEE) received the B.E. degree from UVCE, Bangalore University, and the M.B.A. degree from The University of Melbourne, Australia.

He is currently a Professor with PES University, Bengaluru, India. He is also the Director of the PESU Centre for Information Security, Forensics and Cyber Resilience (C-ISFCR) and PESU Centre for Internet of Things (C-IoT), with a focus on Security. He leads the teaching, research, executive education, industry partnership, and consultancy in his areas of focus. He has authored and co-authored several research papers in leading journals and conferences. His research is focused on various aspects of information security, such as network and cloud security, software and web security, mobile application security, cryptography and blockchain, the IoT, SCADA, and industry 4.0.