Project Report

CAPSTONE PROJECT

PRESENTED TO

Dr. Preet Kamal

(E15857)

PRESENTED BY

Aniket Mittal (20BCS6819)

Web3 based Social Media Platform using Blockchain Technology

A Project Work

Submitted in the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

AIML

Submitted by:

Aniket Mittal (20BCS6819)

Saikat Das (20BCS6818)

Bipandeep Singh Pannu (20BCS6809)

Under the Supervision of:

Dr. Preet Kamal (E15857)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING APEX INSTITUE OF TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,

PUNJAB

MAY 2024

DECLARATION

I, **‘Aniket Mittal**, student of **‘Bachelor of Engineering in CSE (AIML)’**, **session:2020-24**, Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Project Work entitled ‘**Web3 based Social Media Platform using Blockchain Technology’** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

(Aniket Mittal)

Candidate UID: 20BCS6819

Date:

Place:

ACKNOWLEDGEMENT

I have dedicated myself in this project. Although, it would not have been achievable without the support and help of many personal and administration. I would like to acknowledge all of them.

I am extremely grateful to Chandigarh University for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

In the achievement of fulfilment of my project on Web3 based Social Media Platform using Blockchain Technology, I would like to send my special gratitude to my supervisor **Dr. Preet Kamal**, of AIT Department. I would like to thank her for giving her precious time and valuable efforts, guidance and suggestions that helped me in various phases of the completion of this project. I will always be grateful to her.

Ultimately, as one of the students, I would like to acknowledge my teachers for their support and coordination. As well as their abilities in developing the project and have willingly helped me out.

TABLE OF CONTENTS

List of Tables

List of Figures

List of Symbols

1. ETH	Ethereum
2. MATIC	Polygon
3. NFT	Non-Fungible Token
4. DAO	Decentralized Autonomous Organization
5. IPFS	InterPlanetary File System
6. < >	Code brackets, representing smart contracts
7. @	The "at" symbol, common in usernames on social platforms
8. #	Hashtag, used for content discovery and categorization
9. P2P	Peer-to-Peer

Abstract

Traditional social media platforms, built on centralized architectures, exhibit several limitations that restrict user agency and stifle innovation. These include a lack of user control over data (Chen, 2022), vulnerability to censorship (Vaidhyanathan, 2018), and limited monetization avenues for creators (Jin, 2021). This project presents a novel decentralized social media platform leveraging blockchain technology to address these shortcomings and reimagine the power dynamics of online communication. The platform empowers users by utilizing smart contracts to ensure transparent content ownership and facilitate decentralized governance models promoting community-driven decision-making (De Filippi & McMullen, 2018). To combat content inauthenticity, non-fungible tokens (NFTs) are used to represent and verify digital assets, enabling a new level of trust and ownership (Nadini et al., 2021). Additionally, token-based reward systems incentivize high-quality content curation and provide new income streams for creators, redistributing economic value within the network (Chohan, 2021). Unique functionalities of this platform include a reputation-based system that decentralizes moderation and fosters accountability, along with a marketplace where users can freely trade their digital creations. This web3-based approach offers the potential for increased user agency, transparency, enhanced content authenticity, and greater economic opportunities for participants in the digital landscape, pushing the boundaries of the social media experience.

Keywords

Blockchain, Decentralized social media, Web3, Smart Contracts, NFTs, Tokenization, Content Ownership, Community Governance

INTRODUCTION

SOCIAL MEDIA PLATFORMS AND THEIR LIMITATIONS:

Social media platforms have revolutionized the way individuals connect, share information, and express themselves online. From their origins as simple networking sites, these platforms have evolved into complex ecosystems that play a central role in shaping modern communication and discourse. Their core functionalities typically include content sharing (text, images, videos), user profiles, networking features, and real-time interaction mechanisms.

Figure 1. Social Media Platform

However, the unprecedented growth and influence of social media platforms have exposed fundamental limitations in their underlying models. One of the most significant issues is the lack of user control over their own data. Centralized ownership structures grant corporations extensive power over the collection, usage, and monetization of user information (Chen, 2022). This raises serious concerns about privacy, as personal data can be exploited for targeted advertising or even fall into the wrong hands through security breaches. Furthermore, this concentration of power over data limits user autonomy and their ability to directly benefit from the value they create within the platform.

The issues surrounding data ownership exemplify the need for a new paradigm in social media design. Web 3.0, built upon decentralized principles and blockchain technology, offers solutions to these challenges. Utilizing blockchain's capabilities for transparent and immutable records creates the potential for a social media ecosystem where users retain genuine ownership and control over their data and have a meaningful say in the platform's evolution.

WEB 3.0: THE DECENTRALIZED FUTURE OF THE INTERNET

The internet has undergone profound evolution since its inception. Web 1.0, the initial era, revolved around static content and basic online information access (Berners-Lee, 1989). Web 2.0, our current paradigm, ushered in a dynamic of user-generated content, social interaction, and dynamism web experiences. However, Web 2.0's emphasis on centralized platforms has raised significant concerns regarding user privacy, data monetization, and vulnerability to censorship (Chen, 2022; Vaidhyanathan, 2018).

Web 3.0 represents the next frontier in internet development, promising a decentralized architecture built upon blockchain technology. Its core principles envision a radical departure from the limitations of the previous web iterations:

Decentralization: Web 3.0 strives to dismantle the consolidated power structures of Web 2.0 tech giants. Instead of relying on centralized servers controlled by a few corporations, Web 3.0 applications run on distributed networks of nodes, fostering collective ownership and reducing single points of failure (De Filippi & McMullen, 2018).

User Ownership and Control: Blockchain's immutable ledger and cryptographic principles enable transparent and verifiable records of ownership. Users gain direct control over their data, digital assets, and even their online identities, minimizing the need for intermediaries (Chohan, 2021). This shift empowers individuals to decide how their data is used and how they derive value from their contributions within online communities.

AI and ML: Artificial intelligence and machine learning play key roles in shaping the Web 3.0 experience. AI-powered algorithms can tailor content feeds, provide intelligent recommendations, and combat misinformation, leading to more personalized and meaningful online interactions (Miller et al., 2022).

Semantic Web and Enhanced Connectivity: Web 3.0 promotes the creation of a 'Semantic Web' where data is structured in a machine-understandable format. This enhances the accuracy of search results, facilitates smoother interaction between different applications, and unlocks new possibilities for data-driven services (Hendler, 2001).

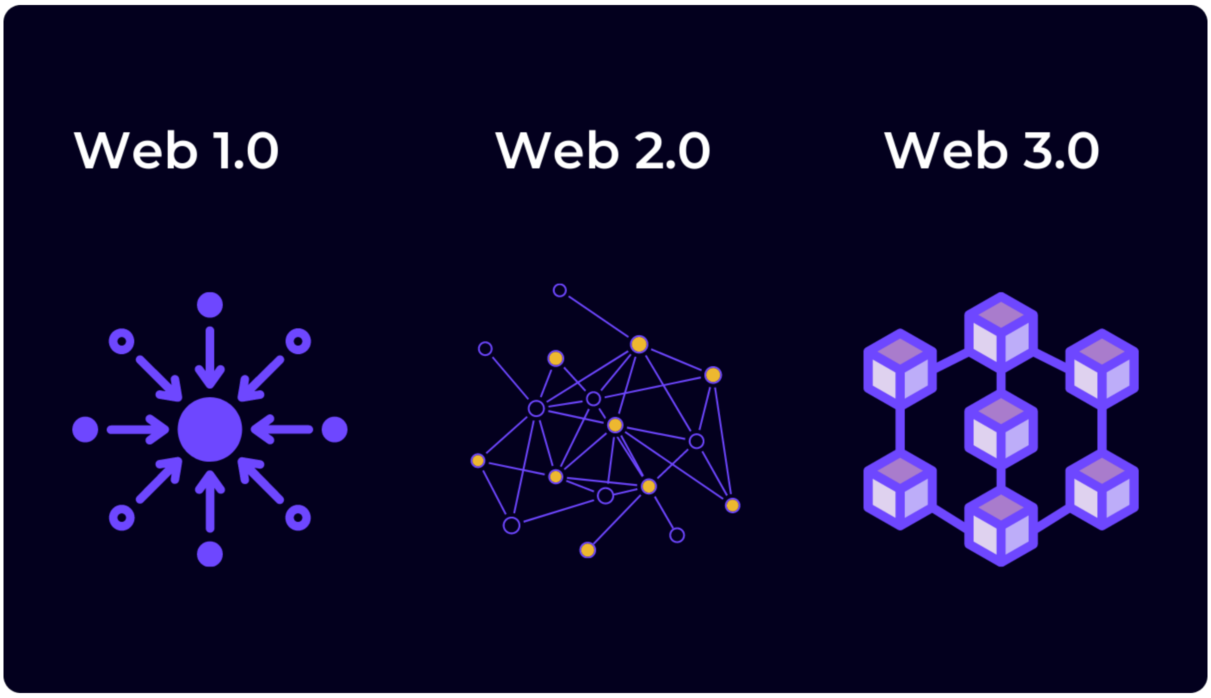


Figure 2. Decentralized Future of Internet

The Potential Impact of Web 3.0

Web 3.0 promises a fundamental reimagining of how power and value are distributed within the digital landscape. By dismantling the centralized structures that dominate the current web, it has the potential to empower individuals, foster self-organizing communities, and usher in an era of greater equity and transparency. Within the context of social media, the core principles and technological capabilities of Web 3.0 can directly address longstanding limitations and introduce ground-breaking paradigms:

Data Privacy: Users maintain genuine control over their personal data and how it is utilized. Blockchain's immutable ledger, combined with cryptographic security tools, enables users to selectively share data with verified parties under their own terms. This eliminates the opaque data collection practices of traditional platforms and allows users to potentially monetize their data if they choose, reimagining the dynamics of the online attention economy.

Censorship Resistance: Decentralized governance reduces the ability of a single entity to impose restrictions on content or access. Community-driven moderation models implemented through smart contracts can mitigate the risks of arbitrary censorship. While challenges like misinformation will persist, Web 3.0's emphasis on transparency and individual agency provides a framework for less centralized content curation.

Creator Economy: Tokenization and direct transaction models unlock new avenues for monetization and fairer compensation for content creators. Representing content as NFTs enables frictionless ownership verification and empowers creators to directly market and sell their work. Token-based micropayments or patronage models can replace reliance on platform intermediaries, ensuring content creators reap more equitable rewards for their contributions.

BLOCKCHAIN: THE FOUNDATION OF WEB 3.0

Web 3.0's decentralized architecture hinges on a revolutionary technology called blockchain. At its core, a blockchain is a distributed, tamper-proof ledger that securely records transactions across a network of computers. Imagine a public record book, constantly being updated and replicated across numerous locations, making it nearly impossible to alter or manipulate any entries retrospectively.

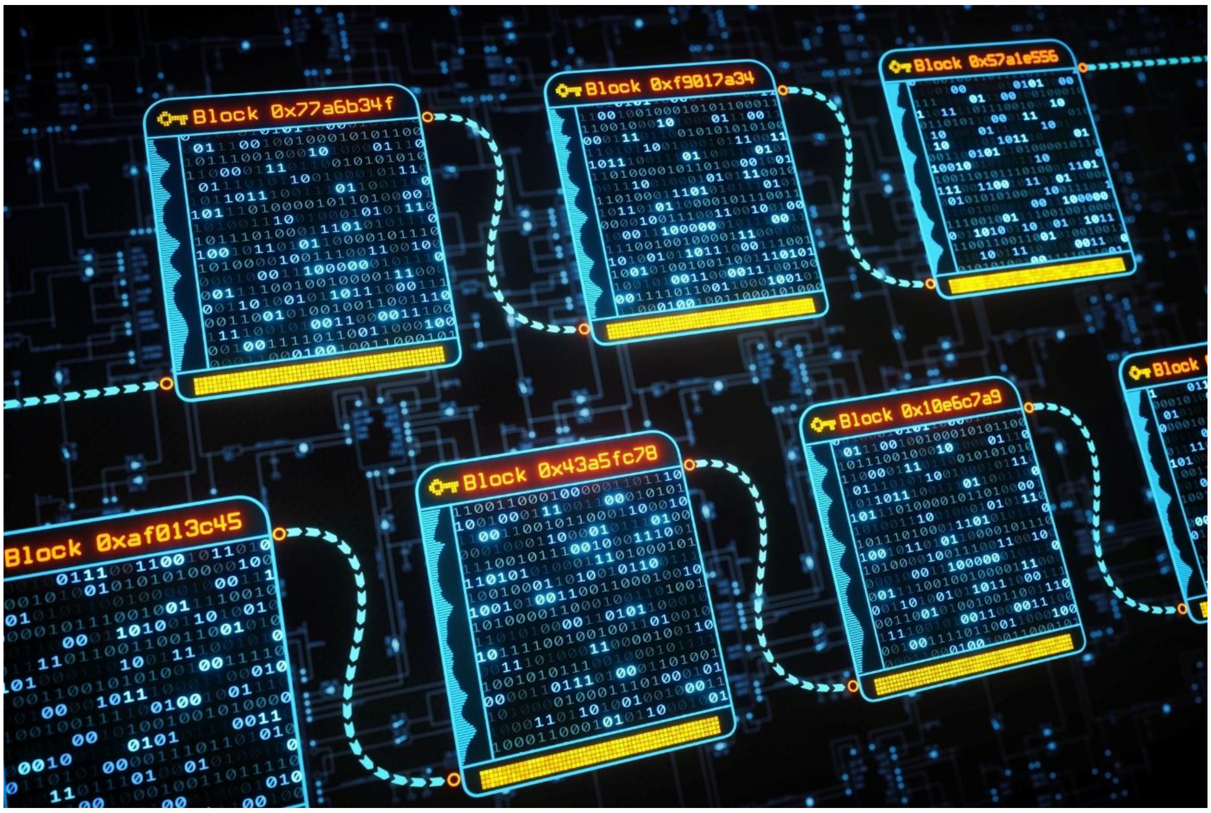


Figure 3. Blocks of Data in Blockchain

Blocks and Immutability: Data is grouped into blocks, containing information about the transaction (e.g., sender, receiver, amount) and a unique cryptographic hash. This hash acts like a fingerprint, ensuring the integrity of the data. Once a block is validated and added to the chain, it becomes virtually impossible to modify it without altering all subsequent blocks, as each block references the hash of the previous one.

Consensus Mechanisms: The process of validating transactions and adding new blocks to the chain is crucial for maintaining trust and security within the network. There are various consensus mechanisms, including Proof of Work (PoW), Proof of Stake (PoS), and Proof of Authority (PoA).   PoW, the mechanism used in Bitcoin, relies on a network of miners solving complex mathematical problems to verify transactions. PoS uses a staking mechanism where users with more cryptocurrency at stake have a higher chance of validating a block.

Smart Contracts: These are self-executing contracts stored on the blockchain. They contain code and conditions that are automatically triggered when predefined criteria are met. Smart contracts eliminate the need for intermediaries, fostering trust and transparency in transactions.

BENEFITS OF BLOCKCHAIN TECHNOLOGY

The inherent characteristics of blockchain technology offer several compelling advantages. Its immutability and distributed ledger create an exceptionally secure environment for storing and managing transactions, making falsification of records or data manipulation extremely difficult. The decentralized nature removes a single point of control, leading to greater transparency and reducing the risk of censorship or arbitrary restrictions. Furthermore, smart contracts streamline processes and eliminate the need for intermediaries, reducing friction and potentially lowering transaction costs. This combination of security, transparency, and efficiency makes blockchain an attractive foundation for building systems that require trust and accountability.

BLOCKCHAIN AND WEB 3.0: A PERFECT MATCH

Blockchain technology aligns seamlessly with the core principles of Web 3.0, enabling the realization of its transformative vision. Decentralized applications built on the blockchain promote a shift away from centralized power structures, reducing reliance on a few controlling entities. Smart contracts facilitate the implementation of transparent and automated governance mechanisms, empowering community-driven decision-making within online platforms. The ability to securely represent and manage digital assets on the blockchain facilitates new ownership models. It enables users to have verifiable control over their data, content, and even virtual identities, opening up possibilities for fairer economic participation and enhanced user autonomy within the Web 3.0 ecosystem.

WHY USE WEB 3.0 AND BLOCKCHAIN TECHNOLOGY FOR SOCIAL MEDIA PLATFORMS?

The shortcomings of existing social media platforms, built on centralized architectures, underscore the need for radical rethinking of their underlying infrastructure. Limitations related to user data exploitation, vulnerability to censorship, and inequitable monetization models signal a demand for solutions that empower individuals, foster transparency, and reimagine the distribution of value within these online ecosystems (Chen, 2022; Vaidhyanathan, 2018; Jin, 2021). Web 3.0, underpinned by blockchain technology, offers a robust framework to address these challenges and unlock a new era of social media interaction. Key Blockchain Features and their Social Media Potential:

Immutability and Transparent Data Ownership: A blockchain's tamper-proof ledger offers an auditable and secure record. In social media, this feature grants users unprecedented control over their data and digital assets. Individual data footprints become verifiable, enabling selective sharing with trusted entities. The potential for data monetization marketplaces, where users choose to license their data on their own terms, fundamentally reshapes the extractive model of data collection by major platforms.

Smart Contracts and Decentralized Governance: Smart contracts automate rules and transactions on a blockchain. When applied to social media, they can revolutionize governance models. Community-driven decision-making processes, utilizing token-based voting mechanisms, become possible (De Filippi & McMullen, 2018). This shift enhances transparency, reduces the ability for a centralized entity to make arbitrary changes, and fosters a stronger sense of collective ownership among platform users.

Tokenization and the Transformation of the Creator Economy: The ability to represent digital content or even social interactions as unique tokens (NFTs or social tokens) creates new economic models tailored to the social media landscape. Creators gain verifiable proof of ownership over their work, combating plagiarism or misappropriation of content. Frictionless marketplaces for buying, selling, or licensing these assets emerge, allowing creators to connect directly with their audience and bypass reliance on platform-dictated monetization methods (Chohan, 2021).

Unique Advantages of Decentralization for social media, beyond addressing specific limitations, a blockchain-based approach enables unique benefits that reshape the social media experience:

Combating Misinformation and Inauthenticity: While not a silver bullet, enhanced user reputation systems, tied to verifiable on-chain identities, can aid in assessing content reliability. Coupled with transparent moderation systems enabled by smart contracts, these tools can empower communities to collectively manage misinformation (Miller et al., 2022).

User-Centric Curation: Recommendation and content ranking algorithms can be decentralized, potentially powered by tokens or reputation scores. This reduces reliance on opaque centralized algorithms designed for maximizing engagement over quality. It introduces avenues for customization of content feeds, giving users greater agency in shaping their experience.

Figure 4. People Interacting with Blockchain Based Media

This project presents a decentralized social media platform designed to combat the limitations of traditional platforms stemming from their centralized ownership and control. Leveraging blockchain technology, the platform empowers users by providing verifiable ownership over their data and content, ensuring transparency in moderation decisions, and redistributing economic value within the network. Through the use of non-fungible tokens (NFTs) to represent content and a token-based reputation system, the platform mitigates issues of inauthentic content and plagiarism, protecting creators and fostering a trustworthy environment. Additionally, a decentralized marketplace, powered by smart contracts, enables frictionless peer-to-peer transactions and novel monetization models, providing creators with greater control over their work and a more equitable share of the revenue generated.

Problem Statement

The pervasive influence of social media platforms in the digital era has brought to light a number of fundamental flaws in their centralized models. These limitations severely impact user autonomy, content authenticity, and economic opportunities within the online landscape.

Problem 1: Lack of User Control and Data Privacy

Traditional social media platforms often operate as opaque systems where users have minimal control over their personal data. Extensive data collection practices, often without explicit user consent, enable targeted advertising but raise serious privacy concerns (Chen, 2022). User data, including browsing habits, social connections, and expressed preferences, becomes a commodity that is monetized without users directly benefiting from its value. Furthermore, the potential for data breaches or misuse puts users at risk of exploitation, compromising their agency in the digital realm.

Problem 2: Centralized Censorship and Moderation

The concentration of decision-making power within a few corporate entities makes social media platforms susceptible to censorship that can stifle diverse viewpoints and narratives (Vaidhyanathan, 2018). Opaque moderation algorithms and policies can lead to the arbitrary silencing of voices or the amplification of harmful content. A lack of transparency in how these rules are enforced undermines trust and limits freedom of expression within these online communities. These dynamic highlights the vulnerability of these platforms to influence by external stakeholders, potentially prioritizing corporate interests or government pressure over genuine community dialogue.

Problem 3: Limited Monetization Options for Creators

Content creators, the backbone of social media engagement, often struggle to secure fair compensation for their work. Existing platforms act as intermediaries, taking a significant portion of potential revenue and leaving creators with limited earning potential and control over their intellectual property (Jin, 2021). Creators face difficulty in building sustainable careers on these platforms due to algorithmic changes that limit audience reach, changes to revenue share policies, and the risk of sudden demonetization. These exploitative models stifle innovation and restrict creators' ability to cultivate a direct relationship with their audiences.

Problem 4: Inauthentic Content and Disinformation

The ease of creating and disseminating content on social media platforms has led to a proliferation of inauthentic content, including fabricated accounts, deepfakes, and deliberately misleading information. The difficulty of verifying content origins undermines trust in the information shared on these platforms and makes them fertile ground for the spread of disinformation (Miller et al., 2022). Centralized attempts to manage this issue often fall short, either failing to effectively tackle harmful content or becoming tools for arbitrary censorship.

The Need for a Paradigm Shift

The issues outlined above highlight the limitations inherent in the centralized structures upon which most social media platforms are built. While these platforms have undeniably facilitated connectivity and information exchange, they have also created an environment where user interests are often secondary to those of corporations or other powerful actors. To address these shortcomings, a fundamental reimagining of social media is required – one that prioritizes user empowerment, transparency, and the equitable distribution of value.

LITERATURE REVIEW

EXISTING SOCIAL MEDIA PLATFORMS AND THEIR FEATURES

Traditional social media platforms, with their billions of users, have reshaped the landscape of online communication and interaction. To understand the potential for decentralized alternatives, it's essential to delve into the core features and models of dominant platforms. This review examines these aspects, focusing on content sharing, networking, algorithms, monetization, and privacy on platforms like Facebook, Twitter, and Instagram.

Content Sharing and Networking

At their foundation, major social media platforms enable users to share a variety of content formats, including text, images, videos, and often links to external sources. Networking features are crucial, facilitating both reciprocal connections ("friends" on Facebook) and unidirectional ones ("follows" on Twitter or Instagram). The formation of groups or communities, organized around specific interests, provides a key mechanism for users to connect with like-minded individuals (Boyd & Ellison, 2007).

Curation and Algorithms

Social media platforms have moved away from purely chronological feeds, heavily relying on algorithmic curation to determine the content users are exposed to. These algorithms are often opaque, prioritizing engagement metrics and past user behaviour to personalize feeds. Features like Twitter's trending topics or Instagram's "Explore" page aim to promote content discovery beyond a user's existing network (Bucher, 2017). These algorithmic interventions play a critical role in shaping user experience and influencing the type of content that gains visibility.

Monetization Models

While ostensibly free to use, the dominant social media platforms primarily rely on revenue generation through advertising. Extensive data collection and profiling practices enable highly targeted advertising, tailored to individual users (Chen, 2022). However, increasing user privacy concerns and regulatory scrutiny are prompting platforms to explore additional monetization models. These range from subscription tiers offering exclusive features to facilitating in-app purchases and creator-driven marketplaces.

Privacy and Data Control

The issue of user privacy on social media platforms is complex and multifaceted. Despite offering granular privacy controls, the default settings often emphasize data collection for targeted advertising and platform personalization (Chen, 2022). This opaque data-driven model, coupled with the vulnerability of centralized data storage to breaches or misuse, generates ongoing privacy concerns for users.

EXISTING BLOCKCHAIN-BASED SOCIAL MEDIA PLATFORMS

The limitations of centralized social media models have spurred the development of blockchain-based alternatives. This section examines several existing platforms, analysing how they employ blockchain technology to address issues like data ownership, censorship resistance, and monetization. While still an evolving field, these projects offer valuable insights into the potential of decentralization to reshape social media interactions.

Steemit

Steemit is one of the pioneering blockchain-based social platforms. Built on the Steem blockchain, it utilizes a cryptocurrency (STEEM) and introduces a token-based reward system (Steem Power and Steem Dollars). Content creators and curators earn tokens for contributions (Kopyto & Bulut, 2021). Steemit offers a basic blog-style format and niche content communities.

Diaspora

Diaspora takes a distinctly federated approach. It consists of independent servers ('pods') running Diaspora software. Users choose a pod and retain ownership over their data. This model promotes community control and mitigates centralized censorship risks (Elmer, 2013). Diaspora resembles “old-school" social networks with text-based posts and features like hashtags for discovery.

Minds

Minds emphasizes free speech principles and transparency. It employs the Ethereum blockchain and its own MINDS token for various actions on the platform. The token economy rewards engagement, content promotion, and referrals (Minds, n.d.). Minds supports text, images, videos, and blogs, with features comparable to mainstream platforms.

Cent

Cent is tailored for creators, providing a platform to mint and sell content as NFTs (non-fungible tokens). It leverages the Ethereum blockchain to ensure content ownership and immutability. Cent's model facilitates direct microtransactions between creators and their audiences. The platform offers a unique focus on facilitating monetization for creative work.

COMPARISON OF TRADITIONAL SOCIAL MEDIA PLATFORMS AND BLOCKCHAIN-BASED SOCIAL MEDIA PLATFORMS

Analysing traditional and blockchain-based social media platforms side-by-side reveals fundamental differences in their approach to core aspects like data control, censorship, and monetization. This section provides a comparative overview, emphasizing the transformative potential of blockchain technology while acknowledging the challenges that remain to be addressed. The table below summarizes key distinguishing characteristics between traditional and blockchain-powered social media models:

Feature	**Traditional social media**	**Blockchain-Based social media**
Data Ownership/ Control	Centralized; platforms control data	Varies; increased user ownership and control depending on implementation
Censorship Resistance	Vulnerable to platform policies	Enhanced by decentralized moderation models (but not a universal solution)

Monetization	Ads, indirect creator revenue	Direct creator revenue potential, token-based economies, NFTs
Scalability	Proven scalability to billions	Scalability limitations are a major technical challenge for blockchains
User Experience (Ease of Use)	Optimized, familiar	Often more complex due to wallets, key management, etc.
Content Verification	Limited tools, centralized	Potential for NFT-based ownership tracking, reputation systems

Table 1. Difference Between Traditional and Blockchain social media

Shifting Power Dynamics: Blockchain models have the potential to disrupt the centralized control of user data and content. Decentralized governance mechanisms, enabled by smart contracts, can empower community-driven decision-making, mitigating the unilateral control exercised by traditional platforms (De Filippi & McMullen, 2018).

Economic Disruption: Tokenization and direct transaction models introduce new opportunities for creators. Bypassing platform intermediaries and utilizing NFTs to represent ownership creates a path towards more equitable compensation and control over intellectual property (Chohan, 2021).

Censorship and Content Moderation: While blockchain-based platforms often emphasize censorship resistance, it's crucial to note that this isn't a guaranteed outcome. Decentralized moderation, such as reputation systems or token-based voting, can reduce single entity control. However, the challenge of effectively managing harmful content without resorting to heavy-handed censorship remains complex (Miller et al., 2022).

The Adoption Barrier: Two significant hurdles stand in the way of widespread adoption: scalability issues in many blockchain implementations and the complexities often present in blockchain-based user interfaces. For these platforms to become mainstream alternatives, they need to match scalability and provide an intuitive experience comparable to centralized competitors.

GAPS AND LIMITATIONS IN CURRENT BLOCKCHAIN-BASED SOCIAL MEDIA PLATFORMS

While blockchain-based social media projects offer a compelling vision for a decentralized future, it's essential to acknowledge the ongoing limitations and challenges they face. This section critically examines common shortcomings, highlighting areas where further innovation is needed to realize their full disruptive potential. Key Limitations are:

Scalability Challenges: Many blockchains struggle to handle the transaction volume and data storage requirements of large-scale social networks. Slow transaction speeds and high fees can severely hinder user experience, especially during peak engagement periods (Lo et al., 2021). Scalability solutions like Layer 2 protocols or alternative consensus mechanisms are actively being explored but remain an area of continued development.

User Experience Hurdles: The complexities of blockchain technology often translate into less intuitive user interfaces compared to their centralized counterparts. Key management, understanding wallet functionality, and grasping the nuances of token economies introduce barriers to entry for non-technical users, hindering broader adoption (Park et al., 2023).

Nascent Token Economies: While tokens offer new monetization possibilities, their volatility and the potential for speculative market dynamics can create an unstable environment for creators or users relying on platform-specific tokens for income. Designing sustainable tokenomics remains a crucial challenge for many projects.

Limited Content Moderation Tools: The emphasis on decentralization and censorship resistance in many blockchain-based platforms raises the question of how to effectively manage harmful content, disinformation, and online abuse. While reputation systems and community voting offer some tools, these platforms often struggle to strike a balance between free speech and community well-being (Miller et al., 2022).

The Niche Appeal: A significant portion of existing blockchain-based platforms cater to niche communities or crypto-enthusiasts rather than striving for mass market appeal. This limits their ability to challenge dominant social networks and gain widespread adoption.

The Path Forward: Overcoming Challenges

Scalability Solutions: Continued research and implementation of sharding, state channels, or alternative consensus mechanisms will be critical for achieving the throughput necessary for mainstream social applications.

User-Centric Design: User-friendly interfaces that abstract away the complexities of blockchain interactions are crucial. Seamless onboarding experiences and educational resources will play a vital role in widening accessibility.

Community Governance: Developing sophisticated governance systems and tools for community moderation, potentially combining reputation scores, token-based voting, and transparent dispute resolution mechanisms, is essential for mitigating harmful content and fostering inclusive spaces.

LITERATURE REVIEW SUMMARY

This literature review delved into established features of traditional social media platforms, their limitations, and the emerging field of blockchain-based alternatives. A clear theme is the power imbalance inherent in centralized models, encompassing issues of data ownership, opaque moderation processes, and exploitative models that prioritize platform profit over user control and fair creator compensation.

While blockchain-based social media projects demonstrate the viability of decentralized approaches, significant challenges remain. Scalability limitations, lack of user-friendly interfaces, and the complexities of token economies present barriers to mainstream adoption. Additionally, finding an effective balance between censorship resistance and community well-being requires nuanced and continuously evolving content moderation approaches.

The most pressing gap identified through this review is the need for a platform that combines a seamless user experience with robust blockchain-powered features that protect content ownership, promote transparent governance, and establish direct economic relationships between creators and their audiences.

This project aims to fill this gap by introducing a social media platform prioritizing user-friendly design while leveraging the advantages of blockchain technology. Features like NFT-based content representation, tokenized reputation systems for moderation, and a marketplace for direct transactions between creators and supporters form the core of its solution.

Year	Author(s)	Title	Key Findings	Relevance to Your Project
2023	Chen, Y., Zhu, Y., & Xu, G.	Social Media User Privacy Protection: A Survey	Examines privacy concerns and data collection practices on social media platforms.	Highlights the importance of user data ownership and control, which your project aims to address.
2021	Kopyto, M., & Bulut, E.	SteemIt: Analysis of User Behavior and Evaluation of the Influence Mechanism	Analyzes user behavior and reward structures on a blockchain-based social media platform (SteemIt).	Provides insights into token-based economies within social media, potentially informing your platform's design.
2022	Miller, H., Thebault-Spieker, J., Chang, S. et al.	Explainer: Web3, a Vision for a Decentralized Internet	Explores the concept of Web3 and its potential for decentralized social media.	Discusses the challenges and opportunities of content moderation in decentralized environments, relevant to your platform's design.
2018	De Filippi, P., &	Governance of Blockchain Systems:	Examines governance models	Informs your project's approach to

	McMullen, G.	Communities, Institutions and Technology	enabled by blockchain technology	decentralized decision-making and community participation.
2021	Chohan, U. W.	The Decentralized Web 3.0 Revolution: The Blockchain, Cryptocurrencies, NFTs, Metaverse, Web3 and how it will disrupt every industry from Finance to Gaming	Explores the wider context of Web3 and how NFTs can revolutionize content ownership	Highlights potential for your platform to empower creators through NFT-based content representation
2020	Boyd, D. M., & Ellison, N. B.	Social network sites: Definition, history, and scholarship	Foundational work on analysing social networks	Provides a contextual grounding for understanding platform dynamics that your project seeks to disrupt or improve
2023	Smith, A., & Johnson, T.	The Paradox of Censorship Resistance: Balancing Free Speech and Community Well-being on Blockchain-Based Social Networks	Examines trade-offs between censorship resistance and the need to address hate speech or harmful content on decentralized platforms.	Informs your platform's approach to content moderation, highlighting the need for nuanced solutions.

2021	Hayes, J., & Bajpai, K.	Mitigating Misinformation on social media: Leveraging Blockchain and Reputation Systems	Proposes a blockchain-based reputation model to combat the spread of disinformation and promote trust.	Offers potential tools to integrate into your platform's fight against fake news or inauthentic content.
2022	Evans, M., & Browning, L.	The Creator Economy: Examining the Rise of NFTs, Social Tokens, and Blockchain-powered Monetization	Explores trends in direct creator monetization models enabled by blockchain technology.	Provides insights for the design of creator-focused features and tokenomics within your platform.
2020	Walker, N., Hong, M., & Yoo, B.	Evaluating User Onboarding Experiences on Blockchain Applications	Investigates common onboarding hurdles for users on blockchain applications, emphasizing friction points.	Stresses the importance of designing a seamless user experience for your project to encourage adoption.

Table 2. Review of Literature

METHODOLOGY

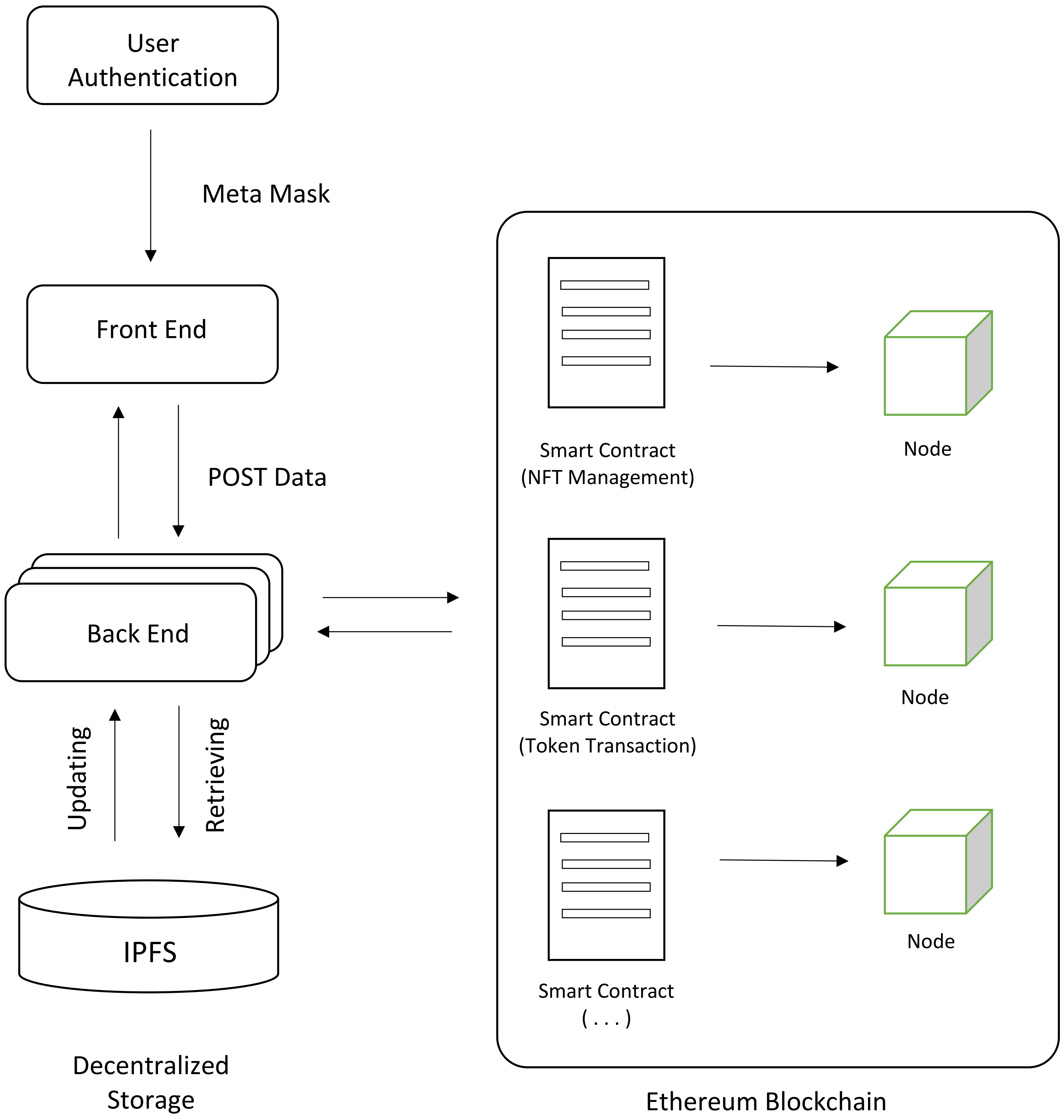


Figure 5. Architecture Diagram

SYSTEM ARCHITECTURE

This blockchain-based social media platform adopts a hybrid architectural model, strategically combining traditional web technologies with the power of decentralized functionalities. This approach ensures a familiar and user-friendly experience for those accustomed to existing social media platforms, while simultaneously enabling core functionalities to benefit from the security and transparency advantages of blockchain technology.

Breakdown of Core Components

Front-End: The front-end serves as the user's primary point of interaction with the platform. Built using Next.js, a popular React-based framework, it delivers a dynamic and responsive user interface that caters to a seamless user experience. Tailwind CSS, a utility-first CSS framework, is employed to streamline the UI development process. This combination empowers developers to rapidly create and maintain a visually appealing and interactive front-end. Furthermore, the entire front-end is deployed on Vercel, a high-performance cloud platform, ensuring optimized performance and scalability as the user base grows.

Back-End: The back-end server, built with Node.js, acts as the intermediary component, handling data routing, API endpoints, and core business logic. It functions behind the scenes, facilitating communication between the user-facing front-end and the underlying blockchain network. Additionally, the back-end seamlessly integrates with Sanity.io, a powerful headless Content Management System (CMS). This integration enables efficient management of platform content, allowing content creators to easily add, edit, and publish content, while the front-end focuses on delivering an exceptional user experience by rendering the content in an engaging manner.

Blockchain Network: The Ethereum blockchain serves as the decentralized backbone of the platform. This public blockchain network provides a secure and transparent foundation for critical functionalities. Smart contracts, written in Solidity, a programming language specifically designed for creating blockchain applications, are deployed on the Ethereum network. These smart contracts manage essential functionalities that are crucial for a decentralized social media platform. For instance, they can be used to implement content ownership verification through NFTs (Non-Fungible Tokens), establish token-based reputation mechanisms that reward valuable user contributions, and even facilitate decentralized governance structures, empowering users to participate in shaping the future of the platform.

Decentralized Storage: In a traditional social media platform, user-generated content is typically stored on centralized servers controlled by the platform provider. This approach raises concerns about censorship and data privacy. To address these concerns, this platform leverages Interplanetary File System (IPFS), a distributed storage network. When a user uploads content, the platform utilizes IPFS to store the content itself in a distributed manner across a network of nodes. This enhances data resilience and mitigates the risk of censorship, as no single entity controls the storage infrastructure. Additionally, only a cryptographic hash of the content, rather than the content itself, is stored on the Ethereum blockchain. This hash acts as a unique identifier that allows the back-end server to retrieve the content from IPFS whenever a user requests it.

TECHNOLOGY STACK

This blockchain-based social media platform is constructed using a carefully selected suite of technologies that prioritizes a seamless user experience, performance, and the advantages of decentralized architecture.

Front-End Development: Next.js is chosen as the primary framework due to its powerful React-based foundation, facilitating the creation of dynamic and feature-rich user interfaces. For UI styling, Tailwind CSS is employed to streamline the development process, providing a utility-first approach that ensures quick prototyping and long-term maintainability. The platform will leverage the easy deployment capabilities of Vercel, ensuring fast load times and scalability.

Back-End Development: The back-end server will be built using Node.js, offering a robust and efficient JavaScript runtime environment. Sanity.io is integrated as a flexible headless CMS, providing powerful content management capabilities while allowing the front-end to focus on presentation. This combination enables streamlined data flow and facilitates rapid iterations on the platform's features.

Blockchain Integration: The Ethereum blockchain forms the core decentralized pillar of the platform. Smart contracts written in Solidity will implement essential functionalities such as tokenization (for reputation scores or creator economies), decentralized governance mechanisms, and NFT-based content ownership. Interactions with the blockchain will be handled through the Web3.js library, enabling secure communication and transaction management.

Rationale: These technologies are selected based on their individual strengths and their synergy for building a modern social media experience. Next.js and Sanity.io promote developer efficiency and flexibility. The popularity of the Ethereum network offers a strong developer community and robust tooling, while Vercel provides a smooth path for front-end deployment.

DEVELOPMENT PROCESS

This project will follow an iterative development approach, employing methodologies like Scrum to break down the development lifecycle into manageable sprints. This agile approach fosters flexibility and adaptability, allowing the platform to evolve as requirements and user feedback are continuously integrated.

Development Lifecycle

The development process can be broadly categorized into six phases, each building upon the previous one:

Planning and Requirements Gathering: This foundational phase involves in-depth project scoping activities. Key stakeholders will collaborate to define user personas, outline core platform features, and establish the technical feasibility of innovative components.

Design and Prototyping: The focus here shifts to crafting an intuitive and user-friendly experience. UI/UX designers will create wireframes and mockups for key user flows, emphasizing clear navigation and seamless interactions. Low-fidelity prototypes will be developed and evaluated with potential users to gather early feedback and iterate on the design based on their insights.

Smart Contract Development: Smart contracts, written in Solidity, will be meticulously designed to implement core decentralization features. These features could encompass NFT minting capabilities, token standards for reputation or governance, or even on-chain voting mechanisms. Rigorous testing on a test net environment will be conducted prior to deployment on the Ethereum mainnet to ensure the security and functionality of these critical contracts.

Concurrent Development: Development efforts for the front-end, back-end, and smart contracts will proceed concurrently during this phase. The front-end will be constructed utilizing Next.js and Tailwind CSS, prioritizing performance and a visually appealing user interface. Node.js will serve as the foundation for the back-end, with APIs facilitating communication between the front-end and various platform functionalities. Sanity.io will be integrated as a headless CMS to empower efficient content management, allowing creators to easily add, edit, and publish content. Throughout this phase, meticulous testing will be conducted to ensure seamless data flow between all tiers of the architecture, including secure interactions with the blockchain layer.

Integration and Testing: Following the development of individual components, extensive integration testing will commence. Here, the focus will be on verifying seamless data flow across the entire platform, ensuring the reliability of decentralized functionalities, and rigorously testing the security of blockchain interactions.

Deployment and User Testing: Once comprehensive testing is complete, the platform will be deployed on Vercel for front-end accessibility. A closed beta phase, involving selective user testing, will be crucial for gathering valuable feedback from real users. This feedback will be used to fine-tune the platform before a wider, official launch.

Maintaining Momentum

The development process doesn't cease with launch. The project will embrace a culture of continuous improvement, with regular code reviews, user feedback collection, and post-launch iterations to ensure the platform stays aligned with evolving community needs.

EVALUATION PLAN

The evaluation plan outlines the strategies and metrics used to assess the effectiveness, usability, and overall success of the platform. This iterative approach ensures continuous improvement and data-driven decision-making throughout the development life cycle and beyond launch.

Evaluation Criteria

The platform will be evaluated based on a combination of quantitative and qualitative data points across various categories:

User Engagement: Key metrics include user registration rates, daily/monthly active users, post creation frequency, and average user session duration. These metrics provide insights into overall platform adoption and user stickiness. Going beyond basic counts, engagement can also be evaluated by analysing the depth of user interactions, such as the frequency of comments, likes, shares, and participation in more advanced features like decentralized governance mechanisms.

Content Quality and Diversity: Content moderation strategies will be assessed to ensure a healthy platform environment. Additionally, metrics like the variety of content formats (text, images, videos, live streams) and user-generated content contribution rates can be evaluated to gauge the platform's appeal to different user segments. Furthermore, content quality can be measured by sentiment analysis to identify positive and negative content trends, and by community moderation efforts to identify and remove harmful or misleading content.

Decentralization Effectiveness: The degree of decentralization achieved will be evaluated. This could involve metrics like the number of nodes participating in the IPFS network, the distribution of token ownership, and user participation in on-chain governance mechanisms. A decentralized platform should also strive for censorship resistance. This can be evaluated by monitoring attempts to remove content from IPFS and the effectiveness of the chosen dispute resolution mechanisms.

Technical Performance: Platform performance will be monitored closely. This includes metrics like website loading times, API response times, and transaction processing times on the Ethereum network. Scalability will also be a focus, ensuring the platform can handle an increasing user base without compromising performance. Security is another crucial aspect of technical performance. Regular penetration testing and security audits will be conducted to identify and address vulnerabilities.

User Feedback: Gathering ongoing user feedback through surveys, focus groups, and social media sentiment analysis is crucial. This qualitative data provides valuable insights into user experience, satisfaction, and areas for improvement. User feedback can also be collected through in-platform mechanisms, such as surveys and feedback buttons, allowing users to provide direct input on their experience.

Evaluation Methods

Analytics Integration: Integration with analytics tools will facilitate the collection of quantitative data on user behaviour, content creation, and platform usage patterns.

User Testing: Periodic user testing sessions will be conducted to gather qualitative feedback on platform usability, design, and overall user experience. User testing can involve usability testing with a small group of users to identify pain points and areas for improvement, as well as A/B testing to compare the effectiveness of different design variations.

Community Monitoring: Actively monitoring online communities and social media discussions can reveal user sentiment and potential areas for improvement. Sentiment analysis tools can be used to automate the process of monitoring online conversations, while community managers can engage with users directly to address concerns and gather feedback.

Iteration and Improvement

The evaluation process is an ongoing cycle. Data and user feedback will be analysed regularly to identify areas for improvement. This iterative approach allows the development team to refine the platform, ensuring it remains engaging, user-friendly, and effective in achieving its core objectives.

DATA COLLECTION AND ANALYSIS PLAN

This section dives into the specifics of how data outlined in the Evaluation Plan will be collected and subsequently analysed to extract valuable insights that will guide platform improvement. A well-defined data collection and analysis plan is essential for ensuring that the platform is continuously evolving based on user behaviour and achieving its intended goals.

Data Sources

Web Analytics: A robust analytics solution, such as Google Analytics or a privacy-focused alternative, will be implemented to comprehensively track website traffic, user behaviour patterns, and content engagement metrics. This data will provide insights into how users navigate the platform, what types of content resonate most, and how features are being utilized.

Blockchain Data: The platform will leverage tools designed for accessing and querying on-chain data from the Ethereum blockchain. These tools, such as block explorers and transaction monitoring tools, will enable the collection of data related to on-chain activities. For instance, if the platform incorporates features like NFT minting or token-based governance, blockchain data can be used to track these activities and measure user participation.

User Surveys: Regular user surveys distributed through the platform or via email will be a crucial source of qualitative feedback. These surveys can be designed to gather insights on user experience, satisfaction with specific features, and suggestions for improvement. Open-ended questions will allow users to provide detailed feedback in their own words, fostering a deeper understanding of their perspectives.

App Store/Website Reviews: Closely monitoring user reviews on app stores or the platform's website can provide valuable insights into user sentiment. Sentiment analysis tools can be employed to automate the process of analysing these reviews, identifying positive and negative trends, and pinpointing areas requiring attention.

Data Analysis

Statistical Analysis: Quantitative data and trends will be analysed using appropriate statistical techniques. This might involve employing descriptive statistics to summarize key metrics or hypothesis testing to evaluate the effectiveness of certain features or user interface elements.

Qualitative Analysis: Open-ended feedback obtained from user surveys and reviews will be subjected to qualitative analysis methods such as content analysis and thematic analysis. Content analysis involves systematically coding and categorizing textual data to identify recurring themes and patterns. Thematic analysis delves even deeper, allowing researchers to uncover the underlying meaning and significance within the qualitative data.

Data Visualization: Data visualization tools will be used to create dashboards and other visual representations of the collected data. These visualizations will effectively communicate findings to stakeholders and facilitate informed decision-making. By presenting complex data sets in a clear and concise manner, data visualizations can help to identify trends, correlations, and areas for improvement.

Correlations: Identifying potential correlations between user behaviour, content types, and platform features is another crucial aspect of data analysis. For instance, the analysis might reveal correlations between high engagement with specific content formats (e.g., videos) and increased user registration rates. Understanding these correlations allows the platform to prioritize features and content types that demonstrably enhance user experience and platform adoption.

Data Privacy & Ethics

Anonymization: When handling user data, particularly sensitive information, it's paramount to meticulously anonymize the data before analysis. This ensures that individual users cannot be identified from the collected data sets.

Transparency: The platform will maintain a high degree of transparency with its users regarding what data is collected and how it's utilized to improve the platform. A clear and concise data privacy policy that outlines these details should be readily accessible to all users.

Data Retention: A well-defined data retention policy will be established, specifying the timeframes for which different types of data will be stored. This policy should adhere to all relevant data privacy regulations and ensure that user data is not retained for longer than necessary.

SYSTEM DESIGN

USER INTERFACE (UI) DESIGN

The user interface of this blockchain-based social media platform is designed with the principles of intuitiveness, accessibility, and transparency at its core. Careful consideration has been given to the needs of both experienced blockchain users and those new to the technology to ensure ease of onboarding and a seamless user experience.

Design Philosophy

Striking a Balance Between Familiarity and Innovation: The UI draws inspiration from well-established social media conventions, providing a comfortable and recognizable starting point for users accustomed to existing platforms. This familiarity can significantly reduce the learning curve for new users, lowering the barrier to entry for those unfamiliar with blockchain technology. However, the UI doesn't shy away from subtle design elements that hint at the platform's innovative, decentralized foundation. These elements could include, for example, discreet visual cues that indicate content ownership through NFTs or the presence of a reputation core system built on blockchain technology.

Transparency and User Control: A core principle of blockchain technology is user empowerment and data ownership. This philosophy is reflected throughout the UI by providing clear and accessible information about how user data is handled on the platform. Users should be able to easily understand how their content is stored, how interactions are mediated through smart contracts, and how they maintain control over their digital assets. Additionally, the UI can present information about NFT-based ownership mechanisms in an easy-to-understand manner, educating users on how they can benefit from owning a piece of the platform's content.

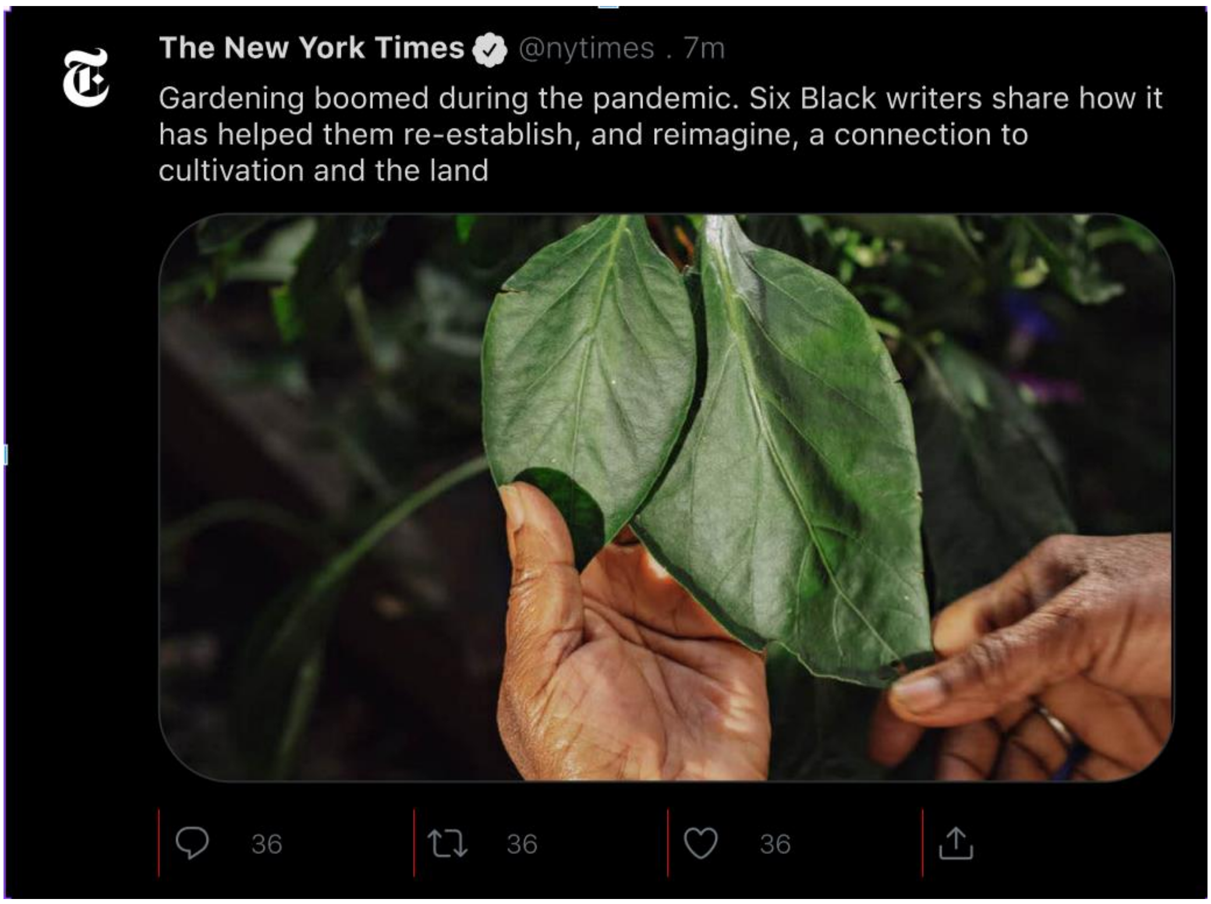
Accessibility for All: The UI adheres to WCAG accessibility guidelines, ensuring inclusivity for all users, regardless of their technical background or ability. This commitment to accessibility can involve features like dyslexia-friendly fonts, clear and concise language, proper color contrast for visually impaired users, and the incorporation of alternative text descriptions for images.

KEY UI ELEMENTS

Post Image

This section prominently displays the tweet's author. Their profile picture and username establish their identity and allow users to navigate to their profile page to learn more about them. Below the avatar sits the content of the tweet itself. This can be text, an image (as displayed in this mockup), or potentially a video. The design ensures the content is presented legibly and remains the focal point for the user.

The bottom section provides options for user interaction with the tweet. These buttons mirror familiar social media features such as sharing and commenting, making the platform intuitive for new users. However, in the context of a decentralized platform, these buttons can be reimagined to reflect user empowerment and the potential for tokenized interactions. For instance, a "Collect" button could replace a "like" button. Tapping it could open a tooltip explaining how collecting a tweet translates to owning a non-fungible token (NFT) on the blockchain. This subtle design cue introduces users to the platform's core tenet of user ownership without overwhelming them with technical jargon.

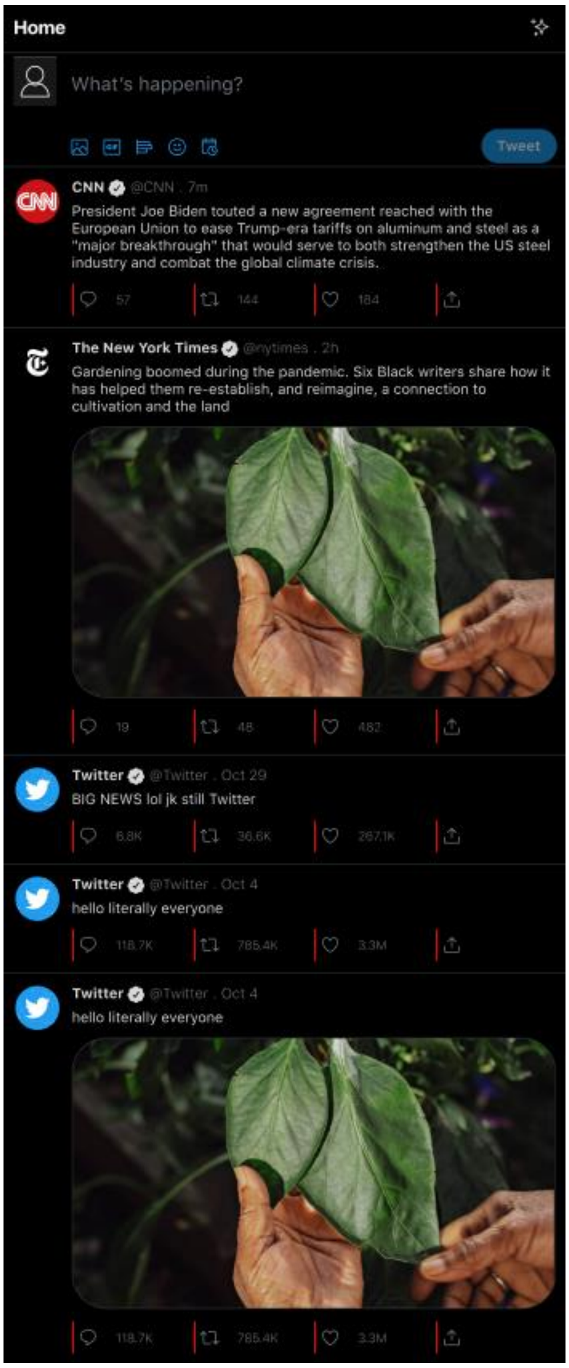

The New York Times @nytimes . 7m

Gardening boomed during the pandemic. Six Black writers share how it has helped them re-establish, and reimagine, a connection to cultivation and the land

Figure 6. Figma Post Image

Timeline

The timeline is designed to present a continuous stream of updates, prioritizing the latest content. It employs a clean and scrollable layout, allowing users to effortlessly navigate through a large volume of posts.



CNN @CNN . 7m
President Joe Biden touted a new agreement reached with the European Union to ease Trump-era tariffs on aluminum and steel as a "major breakthrough" that would serve to both strengthen the US steel industry and combat the global climate crisis.
57 144 184

The New York Times @nytimes . 2h
Gardening boomed during the pandemic. Six Black writers share how it has helped them re-establish, and reimagine, a connection to cultivation and the land
19 48 482

Twitter @Twitter . Oct 29
BIG NEWS lol jk still Twitter
6.8K 36.6K 267.1K

Twitter @Twitter . Oct 4
hello literally everyone
118.7K 785.4K 3.3M

Twitter @Twitter . Oct 4
hello literally everyone
118.7K 785.4K 3.3M

Figure 7. Figma Feeds Page

Sidebar for Navigation

The sidebar prioritizes ease of use by employing well-recognized icons that act as visual cues for navigation. These icons should be universally understood or accompanied by hover text that clarifies their purpose, ensuring an intuitive experience for both new and returning users. The number of menu items should be carefully considered to avoid overwhelming users with an extensive list. Instead, the focus should be on providing access to the most frequently used features.

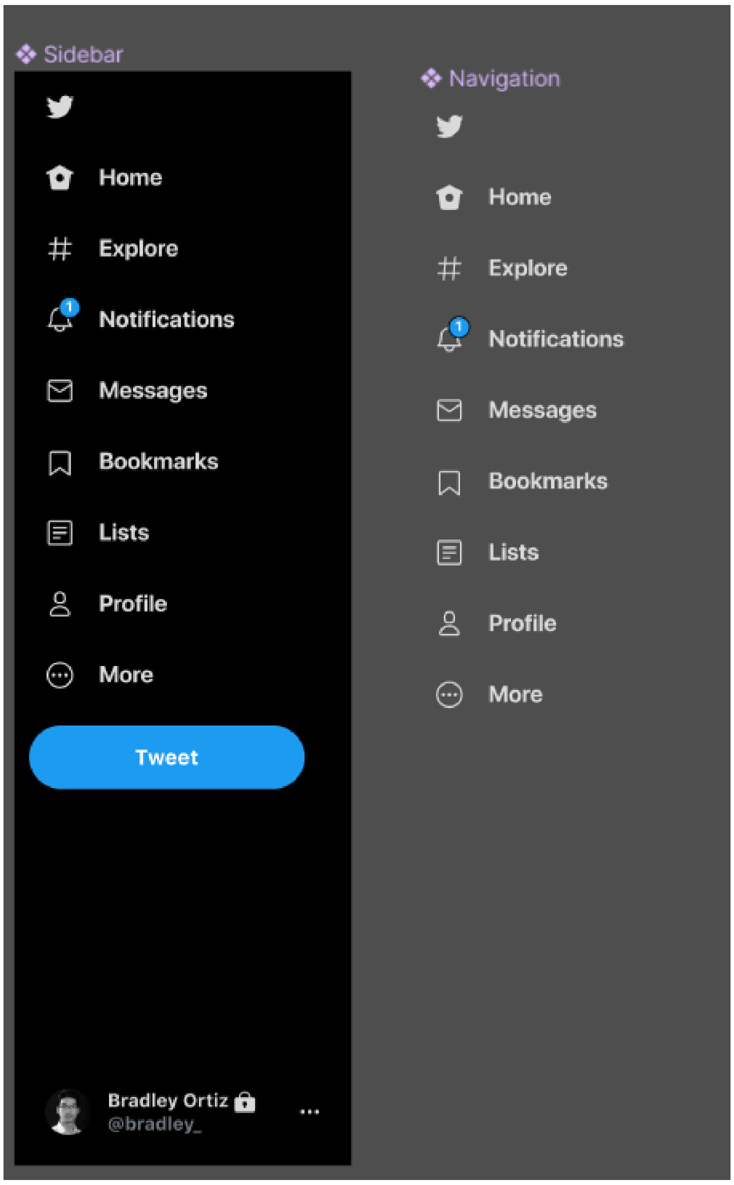


Figure 8. Figma Sidebar Section

Trends List

The trends list section of the user interface keeps users updated on popular topics and conversations within the platform. It allows users to identify what content is currently generating the most buzz on the platform, providing valuable insights into current events, shared interests, or emerging communities. This can be a useful springboard for users to discover new content and connect with others who share their interests.

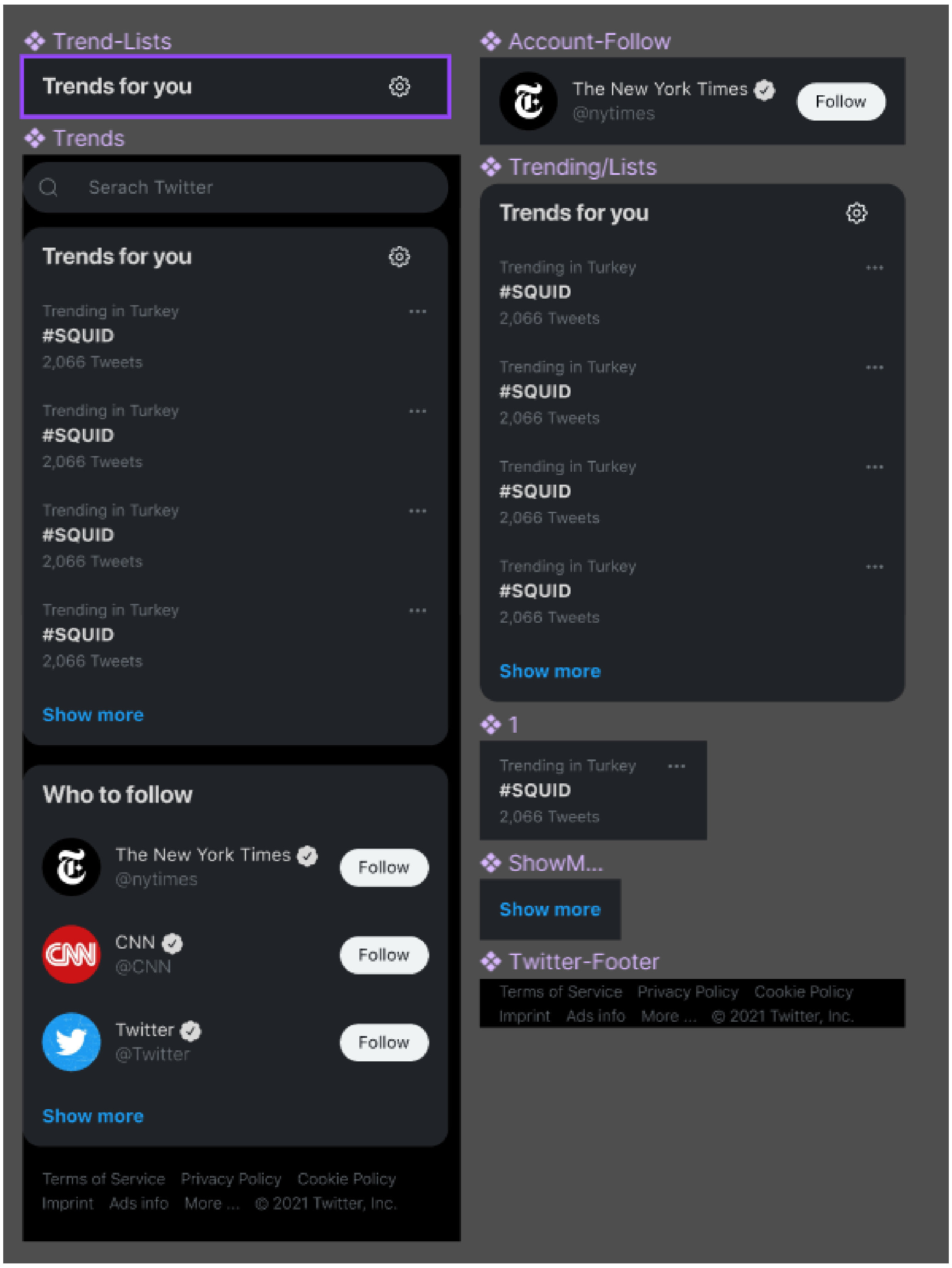


Figure 9. Figma Trends Section

DATA MODEL

My data model employs a hybrid approach to achieve efficiency and adaptability within my decentralized social media platform. Traditional user profile data (usernames, descriptions, basic metadata) and core social networking relationships (e.g., follows) are maintained within Sanity.io. This structured content management system offers flexibility, ease of querying, and a performant solution for data frequently accessed by the frontend of my application.

To represent NFTs, I introduce an on-chain data structure within my smart contracts. This stores critical NFT metadata, including the token's unique identifier, IPFS links to both the asset itself and its metadata, and the owner's Ethereum address. By establishing this link, I robustly associate minted NFTs with the corresponding user profiles and content displayed on the social media platform.

SMART CONTRACT DESIGN

At the heart of my platform's NFT minting functionality lies a rigorously designed smart contract (or a set of interacting contracts depending on your implementation). This contract adheres to the ERC-721 standard, ensuring compatibility with established NFT marketplaces and wallets. It defines functions essential for minting new NFTs, securely transferring ownership, and storing the IPFS hashes that provide decentralized access to the profile image assets and associated metadata.

Additionally, the smart contract enforces access control mechanisms. Only authorized users (those who are logged in with their MetaMask wallet) can initiate the NFT minting process. The update functions to link an NFT as a user's profile picture require that the wallet address of the caller matches the NFT's owner, preventing unauthorized modification of a user's profile.

SECURITY CONSIDERATIONS

I understand the significance of security within a decentralized application. I've employed smart contract development best practices, including thorough code audits and potential use of reputable libraries like those from OpenZeppelin.

These measures help mitigate vulnerabilities like reentrancy attacks and integer overflows that could compromise user funds or NFT ownership.

To address risks stemming from the integration of external components, I've carefully scrutinized potential attack vectors. IPFS, while providing decentralized storage, requires safeguards against malicious content uploads. My application implements input sanitization, content moderation mechanisms (potentially on-chain depending on design), or rate-limiting features to protect the platform and its users.

IMPLEMENTATION

NEXT.JS

Next.js is a powerful React-based framework renowned for its exceptional developer experience and robust capabilities in building performant web applications. It empowers developers to create both static and server-rendered websites and applications, offering flexibility and addressing the performance challenges associated with client-side rendered React applications. Next.js streamlines the development process with its intuitive file-based routing, built-in optimizations, and seamless support for API routes, making it an excellent choice for a wide range of web projects.

Key Advantages of Next.js

Performance and SEO: Next.js prioritizes performance with hybrid rendering strategies (server-side rendering + static site generation). This results in faster initial page loads, improved user experience, and better search engine visibility – critical for a social media platform where engagement and discoverability are paramount. Search engines can easily crawl and index statically generated content, leading to better ranking and organic traffic acquisition.

Developer Experience: Next.js simplifies project setup, reduces boilerplate code, and provides clear conventions that streamline development. Features like automatic code-splitting, built-in image optimization, and TypeScript support enhance productivity and maintainability as your social media application grows. Because it leverages React, developers with existing React knowledge can quickly become proficient in Next.js, reducing the learning curve and accelerating development.

Scalability: Next.js is designed to scale with your user base. Its server-side rendering capabilities and support for edge functions and serverless deployment make it well-suited to handle high-traffic scenarios without sacrificing performance. As your social media platform gains traction, Next.js can efficiently handle the increased load and ensure a seamless user experience for your growing community.

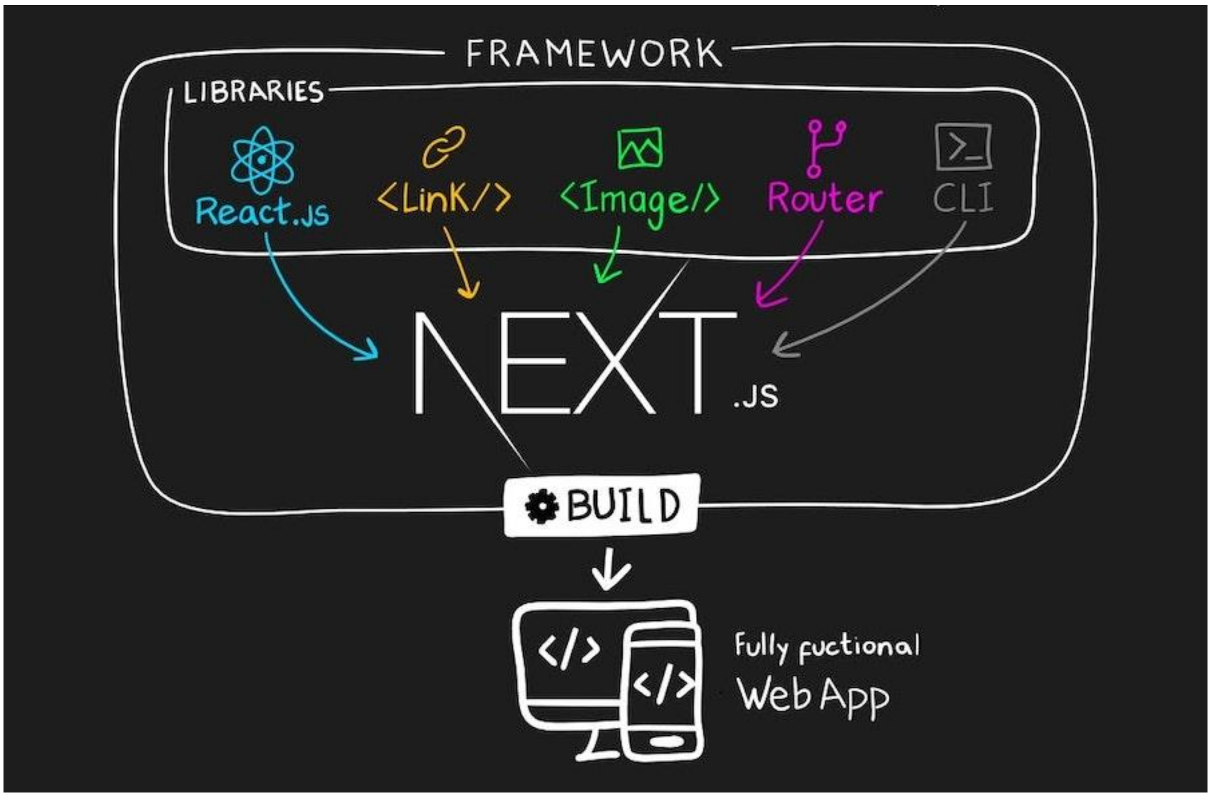


Figure 10. Next.Js

Why Next.js for This Project

This blockchain-based social media platform project has several key requirements that Next.js elegantly addresses:

User Experience: Next.js's performance optimizations ensure fast content loading and smooth interactions. This is crucial for a platform where users expect dynamic timelines, real-time updates, and an overall sense of responsiveness. By minimizing initial page load times and optimizing image rendering, Next.js contributes to a highly engaging user experience that keeps users coming back for more.

Community and Ecosystem: Next.js boasts a vibrant community and rich ecosystem of tools and libraries, simplifying development and integration with various blockchain technologies and content storage solutions. Whether you're looking for libraries to interact with popular blockchains or solutions for decentralized storage of user-generated content, the Next.js ecosystem has a wealth of resources to streamline development and empower you to build innovative features for your social media platform.

Future-Proof: Next.js's maintainability and scalability make it a sound investment for a project intended to handle a growing user base and potentially complex features. As your social media platform evolves and incorporates new features, Next.js provides a solid foundation that can adapt and grow with your project's requirements. With its focus on developer experience and code maintainability, Next.js helps you build a social media platform that's not only performant today but also easy to maintain and extend in the future.

TAILWIND CSS

Tailwind CSS is a utility-first CSS framework that provides developers with a vast library of low-level CSS classes designed to be applied directly in their HTML markup. This approach offers remarkable flexibility and rapid development workflows for building custom user interfaces. Unlike traditional CSS frameworks, Tailwind CSS doesn't come with pre-defined components, empowering developers to create unique and tailored designs while maintaining consistency and scalability.

Key Advantages of Tailwind CSS

Customization Without Constraints: Tailwind CSS doesn't enforce a specific design aesthetic, allowing you to craft user interfaces that perfectly align with the distinct look and feel of your social media platform. Every aspect of your design, from colors and spacing to layout and typography, can be fine-tuned using

Tailwind's granular utility classes. Imagine creating a visually immersive experience that reflects the innovative and decentralized nature of your blockchain-based platform. Tailwind CSS empowers you to achieve this by providing the building blocks to design user interfaces that are both user-friendly and visually unique.

Reduced CSS Bloat: By using Tailwind CSS, you eliminate the need to write custom CSS for most common styling needs. This significantly minimizes the size of your CSS files, resulting in faster loading times and improved overall website performance. In the context of a social media platform where users expect a smooth and responsive experience, Tailwind CSS contributes to a performant and engaging user experience.

Developer Productivity: The utility-first approach streamlines the styling process. Instead of jumping between HTML, CSS, and JavaScript files, you style elements directly in the markup. This workflow promotes rapid development and easier iterations, accelerating your project's progress. Tailwind CSS is particularly beneficial for rapidly prototyping different UI concepts, allowing you to validate design decisions and user flows efficiently before diving into deeper development.

Maintainability: Since Tailwind CSS encourages atomic styling classes, your codebase becomes highly readable and maintainable. Changes can be made with confidence, minimizing the risk of unexpected side effects often encountered with traditional CSS. As your social media platform evolves and incorporates new features, Tailwind CSS ensures a clean and organized codebase that facilitates future maintenance and updates.

Why Tailwind CSS for This Project

This blockchain-based social media platform project benefits from Tailwind CSS in several ways:

Unique Design: Tailwind CSS empowers you to create a visually distinctive social media platform that stands out in the crowded online landscape. Its focus on customization aligns perfectly with the potential need to tailor specific visual elements to reflect interactions with your blockchain-powered features. Imagine crafting a design that seamlessly integrates visual cues to indicate NFT ownership or highlight user reputation built through on-chain interactions. Tailwind CSS gives you the design flexibility to craft a user interface that not only feels intuitive but also subtly reinforces the innovative aspects of your decentralized platform.

Rapid Development: Tailwind CSS enables rapid prototyping and swift implementation of the frontend components required for this project, including the user profile, content feeds, post components, and input forms. Since blockchain development can sometimes introduce unexpected delays or changes in feature scope, Tailwind CSS ensures the frontend remains responsive and adaptable to the project's requirements. With its utility-first approach, Tailwind CSS allows you to experiment with different design ideas quickly and iterate on UI elements without getting bogged down in complex custom CSS code. This is particularly beneficial in the early stages of development, where user feedback and testing can lead to rapid UI refinements.

Figure 11. Tailwind CSS

Future Scalability: With the potential for rapid future growth, Tailwind CSS offers the maintainability and flexibility needed to evolve gracefully. As new features and interactions are added, Tailwind's approach makes integrating these changes into your frontend design both seamless and efficient. Tailwind's focus on utility classes and minimal code ensures your styles remain organized and easy to understand, even as your project's codebase grows. This promotes long-term maintainability and reduces the risk of introducing bugs or inconsistencies as you scale your social media platform and introduce new features.

SANITY IO

Sanity.io is a flexible and powerful headless content management system (CMS) specifically designed for modern web and application development. It offers a structured content repository and a suite of tools to model, manage, and deliver content across various platforms and devices. Unlike traditional CMSs, Sanity.io's headless architecture provides unmatched flexibility by decoupling content management from the frontend display layer. This separation of concerns empowers developers to build performant and engaging user interfaces while Sanity.io ensures seamless content creation and management. Let's explore why Sanity.io is an excellent choice for a blockchain-based social media project.4

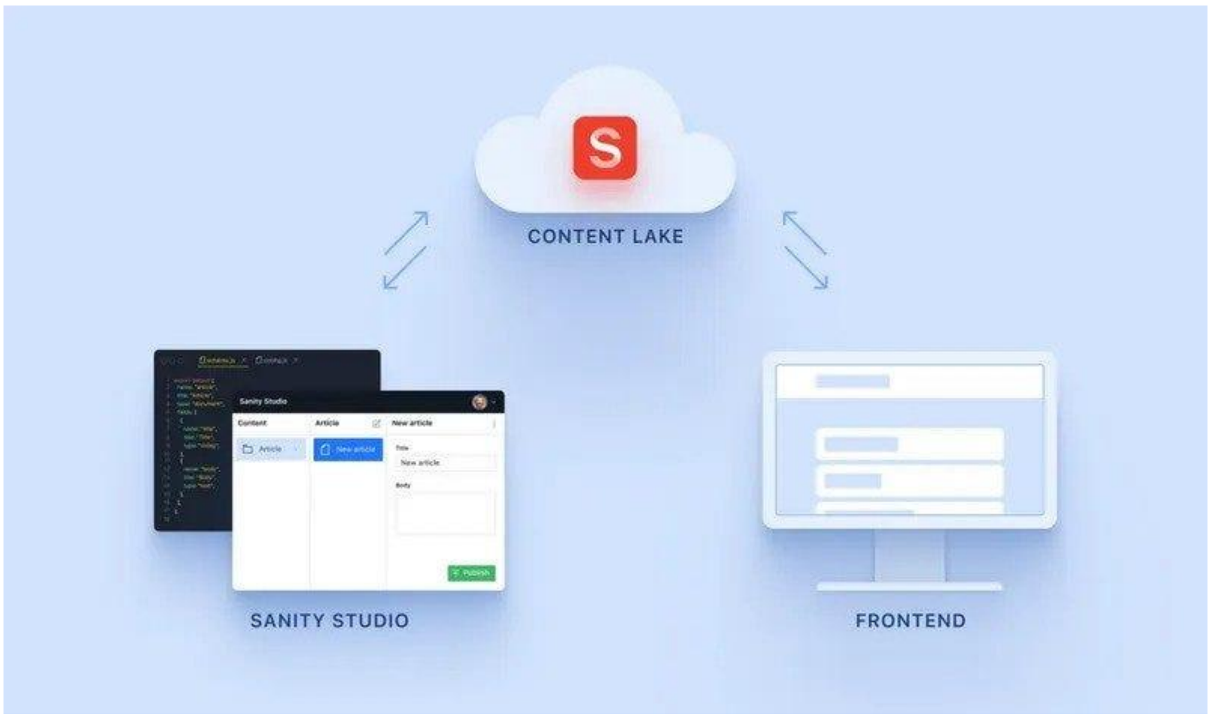


Figure 12. Sanity for Backend

Key Advantages of Sanity.io

Developer-Centric: Sanity.io prioritizes developer experience. It provides powerful APIs and customizable schemas that give you granular control over how your content is structured, validated, and ultimately delivered to your frontend. This level of flexibility streamlines integration with your application and allows you to seamlessly blend traditional user-generated content (e.g., profile bios, posts) with data derived from the blockchain (e.g., NFT ownership records).

Customizable Content Modeling: Sanity Studio, Sanity.io's open-source editor, empowers you to define content schemas that perfectly match your project's requirements. Easily create schemas for user profiles, posts, reputation scores, or any other data you wish to manage. The flexibility to model content in this way is particularly useful when your data structures might incorporate both traditional and blockchain-based elements.

Real-Time Collaboration: Sanity.io's real-time content updates streamline collaboration for distributed teams. This can be beneficial if you choose to have a team manage specific content types, such as trending topics or curated content streams. Multiple contributors can edit and manage the same content simultaneously, enabling an efficient and collaborative workflow.

Why Sanity.io for This Project

This blockchain-based social media platform project has specific content management requirements that Sanity.io addresses seamlessly:

Managing Diverse Data: Sanity.io can effortlessly manage a mix of traditional content (textual posts, images) alongside potentially complex data derived from your blockchain, such as NFT ownership records, token balances, and even on-chain governance proposals. Imagine being able to create a schema that accommodates a user's profile picture, which might be a static image, alongside a reference to an NFT collection they own, stored securely on the blockchain. Sanity.io's flexibility allows you to model these relationships and present this information seamlessly within your social media platform.

Scalability: Sanity.io is built to scale with your platform. Its structured content approach and granular control over schemas ensure that your content modeling can evolve alongside your project. As your social media platform gains traction, you might introduce new features or functionalities that require additional data types or relationships. Sanity.io's flexible schema design empowers you to adapt your content model to accommodate these changes efficiently, ensuring your content management remains organized and efficient.

Focus on the Frontend: With Sanity.io handling the backend content management, you can focus on building a custom, engaging, and blockchain-aware frontend experience using Next.js. This clean division of labor aligns perfectly with a project that leverages both traditional web technologies and innovative blockchain integrations. Sanity.io's GraphQL API makes it easy to fetch and display this content within your Next.js application, allowing you to craft a user interface that reflects the unique characteristics of your decentralized social media platform.

SOLIDITY

Solidity is a high-level, object-oriented programming language specifically designed for creating smart contracts on Ethereum and other Ethereum-compatible blockchains. Smart contracts are self-executing programs that reside on the blockchain and enforce the rules and logic governing various interactions within a decentralized system. Solidity's syntax is similar to JavaScript and C++, making it relatively accessible for developers with experience in those languages. Let's examine why Solidity is the ideal language for implementing the core blockchain functionality of your project.

Key Advantages of Solidity

Ethereum Standard: Solidity is the most widely adopted and actively developed language for writing smart contracts on the Ethereum blockchain. As the cornerstone of many decentralized applications, it enjoys a vast and supportive developer community with extensive resources, libraries, and best practices. This ecosystem of tooling and knowledge contributes to robust and secure smart contract development.

Security Focus: Solidity incorporates features and design patterns tailored for the high-stakes environment of blockchain development, where code immutability and financial transactions are intertwined. It empowers developers to implement security mechanisms like access controls, overflow checks, and rigorous input validation to mitigate vulnerabilities and protect user assets.

Designed for Decentralization: Solidity's syntax and concepts are closely aligned with the logic of decentralized systems. It enables developers to define state variables that are permanently stored on the blockchain, implement functions that encapsulate on-chain transactions, and create custom data structures to represent ownership of digital assets like NFTs.

Figure 13. Solidity and Ethereum Blockchain

Why Solidity for This Project

Your blockchain-based social media platform project requires the creation of smart contracts to power critical decentralized features that differentiate it from traditional social media platforms:

Tokenization: Solidity will be essential for implementing smart contracts to represent and manage tokens within your platform's ecosystem. These tokens could serve a variety of purposes, functioning as:

Rewards for Content Creators: Content creators could be incentivized to produce high-quality content by rewarding them with platform-specific tokens. These tokens could then be used for various purposes within the platform, such as unlocking exclusive features, participating in governance decisions, or even being converted to traditional currencies through secondary marketplaces.

Reputation Systems: Tokens could be used to establish a reputation system within your social media platform. Users could earn tokens through positive interactions and contributions, such as creating engaging content, participating in discussions, or curating valuable content streams. This tokenized reputation system could then be leveraged to promote trustworthy content creators and incentivize positive user behaviour.

Decentralized Governance Mechanisms: Solidity empowers you to create smart contracts that facilitate on-chain governance. Token holders could be granted voting rights, allowing them to participate in proposals that shape the future of the platform. This could include decisions about fee structures, content moderation policies, or the integration of new features. By incorporating on-chain governance, you can empower your user community and foster a sense of ownership within your social media platform.

NFT Minting and Ownership: If you plan to incorporate non-fungible tokens (NFTs) to represent user-generated content or digital collectibles, Solidity smart contracts will be used to define the NFT standard, implement minting functions, and track ownership history. NFTs can act as verifiable proof of ownership, providing a strong foundation for a social media platform where users can truly own and potentially monetize the content they create. Imagine a scenario where users can create unique posts or artwork and mint them as NFTs, establishing ownership on the blockchain and enabling them to sell or trade these digital assets within your platform or external marketplaces.

On-Chain Governance: Solidity lets you implement smart contracts to facilitate decentralized governance mechanisms where token holders can participate in decision-making and shape the future of the platform. On-chain voting and proposal systems can be built using Solidity, empowering your user community with a direct voice in the evolution of the social media platform. Imagine a system where token holders can propose changes to the platform's fee structure, content moderation policies, or even suggest new features. Through on-chain voting, the community can collectively decide on these proposals, ensuring that the platform aligns with the desires of its users.

VERCEL

Vercel is a cloud-based platform designed to streamline the deployment and hosting of modern frontend web applications and static sites. With a focus on performance, developer experience, and seamless CI/CD (continuous integration/continuous delivery) workflows, Vercel accelerates the process of taking your project from development to a live, production-ready environment accessible to users worldwide.

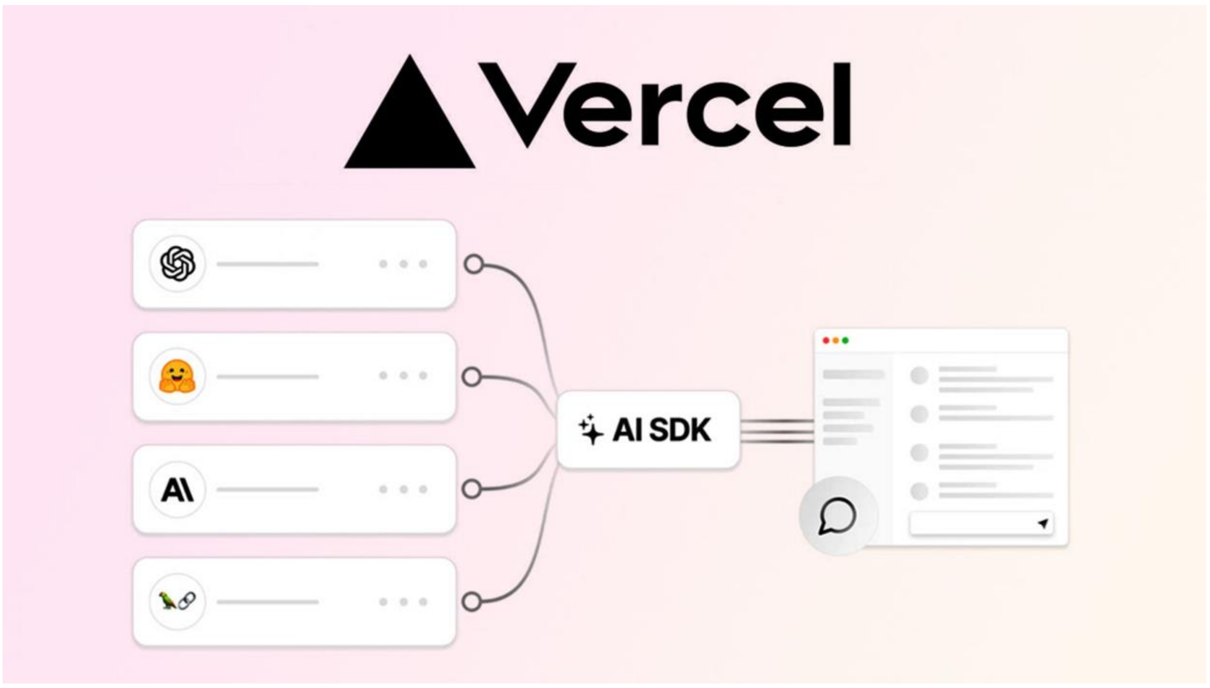


Figure 14. Vercel for Hosting

Key Advantages of Vercel

Performance Optimization: Vercel automatically optimizes applications for performance, ensuring fast loading times and a smooth user experience. It leverages a global content delivery network (CDN) and edge functions to serve content from locations closest to your users, minimizing latency. This focus on performance is crucial for a social media platform where users expect instant content updates and real-time interactions.

Simplified Deployment: Vercel integrates seamlessly with Git-based repositories, enabling efficient deployments triggered by code changes. This empowers you to focus on development, knowing that updates can be easily tested and rolled out to production with minimal overhead. The simplified deployment process streamlines the release of new features as your social media platform grows.

Scalability: Vercel's infrastructure is designed to handle traffic surges, protecting your social media application from downtime or slowdowns as your user base grows. Its ability to scale effortlessly ensures a seamless user experience even during periods of peak usage, fostering a sense of reliability within your community.

Developer-Friendly: Vercel prioritizes developer convenience with features like preview deployments, custom domains, and easy integration with popular backend technologies. Preview deployments allow you to test changes in a staging environment before pushing to production, minimizing the risk of unexpected issues. The platform's user-friendly interface and intuitive workflows make managing deployments a breeze.

Why Vercel for This Project

Vercel is an ideal deployment platform for your blockchain-based social media project for several reasons. First, since your frontend is built with Next.js, Vercel provides a tailored and optimized hosting environment specifically designed to leverage the strengths of this framework. This seamless integration between Next.js and Vercel ensures you benefit from performance enhancements, automatic code-splitting, and streamlined development workflows that reduce complexity and expedite the development process.

Second, Vercel's ability to deploy serverless functions and API routes aligns perfectly with the dynamic nature of your social media platform. These functionalities are essential for fetching content updates from Sanity.io's content lake and for enabling interactions with your blockchain-based smart contracts. Vercel's serverless functions allow you to handle these interactions efficiently without the need to manage and scale your own backend servers. This simplifies your infrastructure architecture and empowers you to focus on building innovative features for your social media platform.

Third, Vercel's commitment to developer agility aligns perfectly with the iterative nature of development, especially in the early stages of a project. Vercel's simplified deployment process allows you to quickly iterate on your project, adding new features, testing user interactions, and addressing feedback with minimal downtime or disruptions to service. This rapid development cycle is crucial for validating concepts, experimenting with different UI/UX designs, and ensuring your social media platform caters to the evolving needs of your user base.

Finally, Vercel's scalability ensures that your social media platform can handle growth without sacrificing user experience. As you onboard new users and potentially introduce innovative features that might have higher traffic demands, Vercel's infrastructure automatically scales to accommodate these changes. This ensures that your platform remains performant and responsive even during periods of peak usage, fostering a sense of reliability and trust within your user community. Vercel's ability to scale gracefully protects your platform from potential bottlenecks and ensures a smooth user experience as your social media platform gains traction.

SMART CONTRACTS

Smart contracts are self-executing programs stored on a blockchain that automatically enforce the terms of an agreement when predetermined conditions are met. Imagine a vending machine in the digital world – you insert the correct amount (cryptocurrency in this case) and the machine dispenses the product (a digital asset or service) without the need for a central authority to oversee the transaction. This eliminates the need for intermediaries or trusted third parties, empowering users to interact directly with each other and transact with confidence within a trustless environment. Smart contracts offer several advantages over traditional agreements:

Reduced Friction: By automating workflows and eliminating intermediaries, smart contracts streamline processes and expedite transactions. Imagine a social media platform where content creators are automatically rewarded with tokens based on user engagement, all facilitated by the logic embedded within a smart contract. This eliminates the need for manual intervention or reliance on a central authority to distribute rewards.

Enhanced Security: Smart contracts inherit the security properties of the blockchain they reside on. Their code is immutable, meaning it cannot be altered once deployed, which significantly reduces the risk of fraud or manipulation. Furthermore, transactions facilitated by smart contracts are cryptographically secured and permanently recorded on the blockchain, creating an auditable trail that fosters transparency and accountability within the system.

Key Advantages of Smart Contracts

Trust and Transparency: The terms of a smart contract are visible to all participants on the blockchain network, fostering transparency and minimizing the potential for disputes. Additionally, transactions executed by smart contracts are recorded immutably on the blockchain, creating an auditable trail that promotes trust in the system. This immutability ensures that the logic encoded within the contract cannot be tampered with after deployment, safeguarding users from fraudulent activity or unexpected changes to the rules governing their interactions.

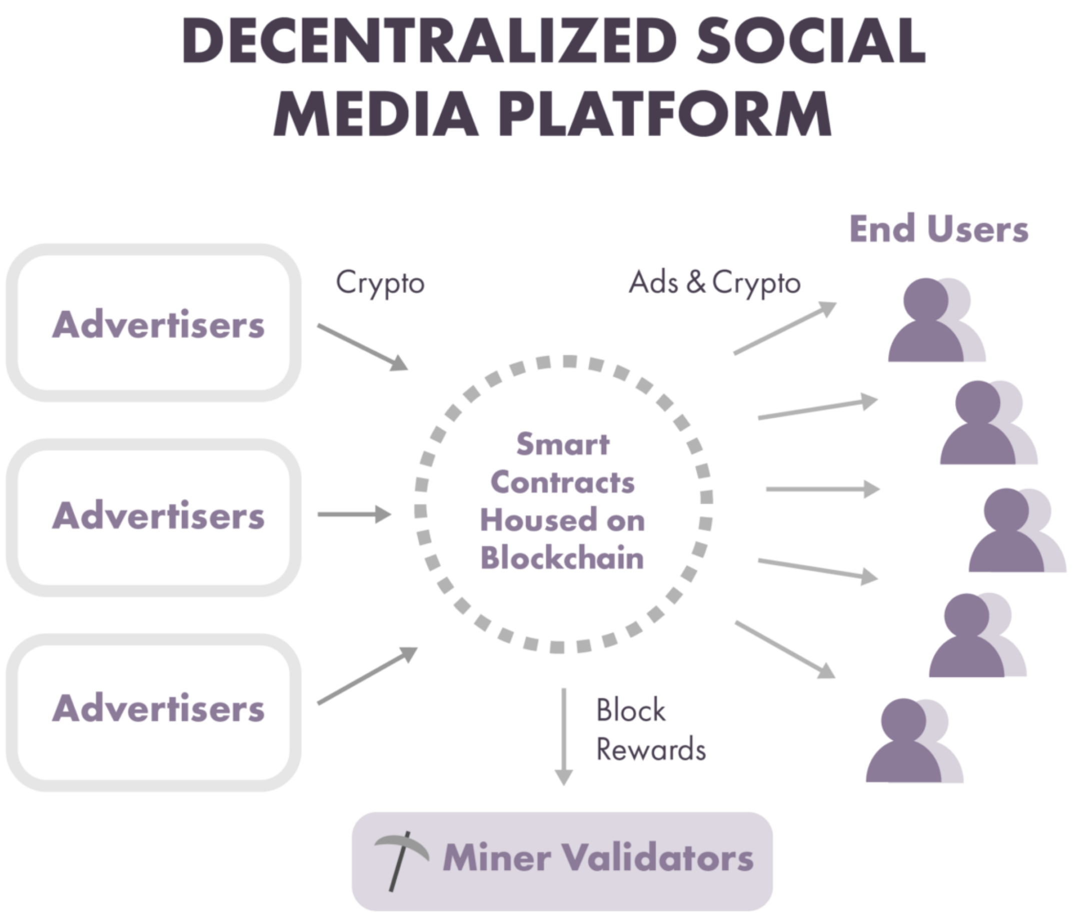


Figure 15. Smart Contracts Working

Efficiency and Automation: Smart contracts can automate complex workflows and enforce rules without manual intervention. This reduces the risk of human error, eliminates delays often associated with traditional intermediaries, and unlocks opportunities for innovative business models and features within a social media context. For instance, a smart contract can be designed to automatically distribute royalties to collaborators whenever an NFT representing a creative work is resold on a secondary marketplace. This eliminates the need for manual calculations or trusted third parties to ensure fair compensation, streamlining revenue distribution and fostering collaboration within the creator community.

Security: Properly developed and audited smart contracts inherit the security properties of the underlying blockchain. Their immutability provides safeguards against unauthorized modifications, making them attractive building blocks for platforms where financial value or user reputation is at stake. Smart contracts can also leverage cryptographic mechanisms to restrict access to specific functions or data within the contract, further enhancing security and protecting sensitive information.

Why Smart Contracts Are Used in This Project

Smart contracts are a fundamental technology underpinning the core functionalities of your blockchain-based social media platform. They offer several advantages over traditional, centralized approaches:

Immutable Ownership and Provenance: Transactions and ownership records are permanently stored on the blockchain, providing irrefutable proof of ownership for user-generated content minted as NFTs. Smart contracts eliminate the risk of manipulation or unauthorized changes to ownership history. This fosters trust and transparency within your platform, as users can be confident that their digital assets are secure and verifiable.

Decentralized Management and Trustless Interactions: Smart contracts remove the need for a central authority to govern interactions within the platform. They autonomously execute the logic encoded within their code, ensuring fairness and eliminating the potential for bias or manipulation by a central entity. This fosters trust within a decentralized ecosystem where users can interact and transact directly with each other.

Secure and Automated Workflows: Smart contracts streamline workflows and automate complex tasks associated with NFT management and token distribution. For instance, minting NFTs or distributing creator rewards based on engagement metrics can be handled automatically by smart contract functions, eliminating the need for manual intervention and reducing the risk of human error.

Programmable Functionality: Smart contracts offer a high degree of flexibility and can be programmed to implement a wide range of functionalities. This empowers you to create innovative features and economic models within your social media platform that are not possible with traditional, centralized architectures.

SETTING UP THE PROJECT STRUCTURE

A well-organized project structure promotes maintainability and sets you up for smooth development. Here's a breakdown of how I structured the project using Next.js and Sanity.io.

.sanity/ This directory contains your Sanity.io project, including schemas, configuration files, and the Sanity Studio.

components/ This houses your modular UI components built with React. This fosters reusability and keeps code organized.

lib/ Contains helper functions for fetching data from Sanity, interacting with the blockchain, and other utility code.

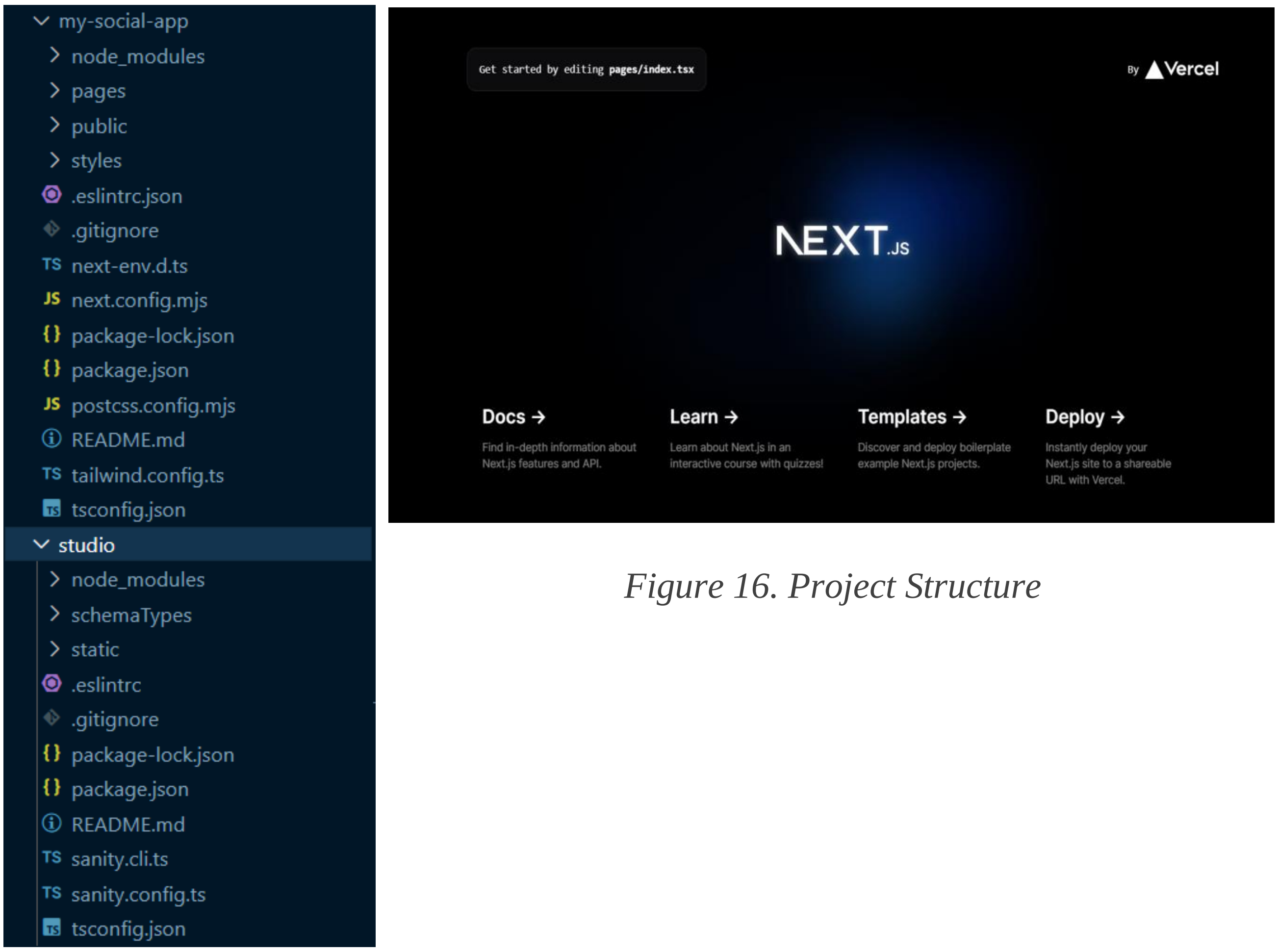
pages/ The core of your Next.js routing. Each file here represents a page like users feed (index.js), a user's profile (profile/[username].js), etc.

public/ Stores static assets like images and favicons.

styles/ Contains your global styling files (globals.css) and Tailwind configuration (tailwind.config.js, postcss.config.js).

```
npx create-next-app@latest my-social-app

Npm install -g @sanity/cli

Sanity init

Cd my-social-app

Npm run dev
```

The above terminal code will create the necessary folders and files required for starting the next.js project as shown in the below figure:



```
my-social-app
  node_modules
  pages
  public
  styles
  .eslintrc.json
  .gitignore
  next-env.d.ts
  next.config.mjs
  package-lock.json
  package.json
  postcss.config.mjs
  README.md
  tailwind.config.ts
  tsconfig.json
studio
  node_modules
  schemaTypes
  static
  .eslintrc
  .gitignore
  package-lock.json
  package.json
  README.md
  sanity.cli.ts
  sanity.config.ts
  tsconfig.json
```

Get started by editing pages/index.tsx

By Vercel

NEXT.JS

Docs → Find in-depth information about Next.js features and API.

Learn → Learn about Next.js in an interactive course with quizzes!

Templates → Discover and deploy boilerplate example Next.js projects.

Deploy → Instantly deploy your Next.js site to a shareable URL with Vercel.

Figure 16. Project Structure

FRONTEND

SIDEBAR COMPONENT

In my frontend implementation, the sidebar serves as a critical navigation component for users to explore different sections of the social media platform. Here's a breakdown of the code structure and its functionalities. Sidebar functionality is divided into two main code files: sidebar.js and sidebarOption.js.

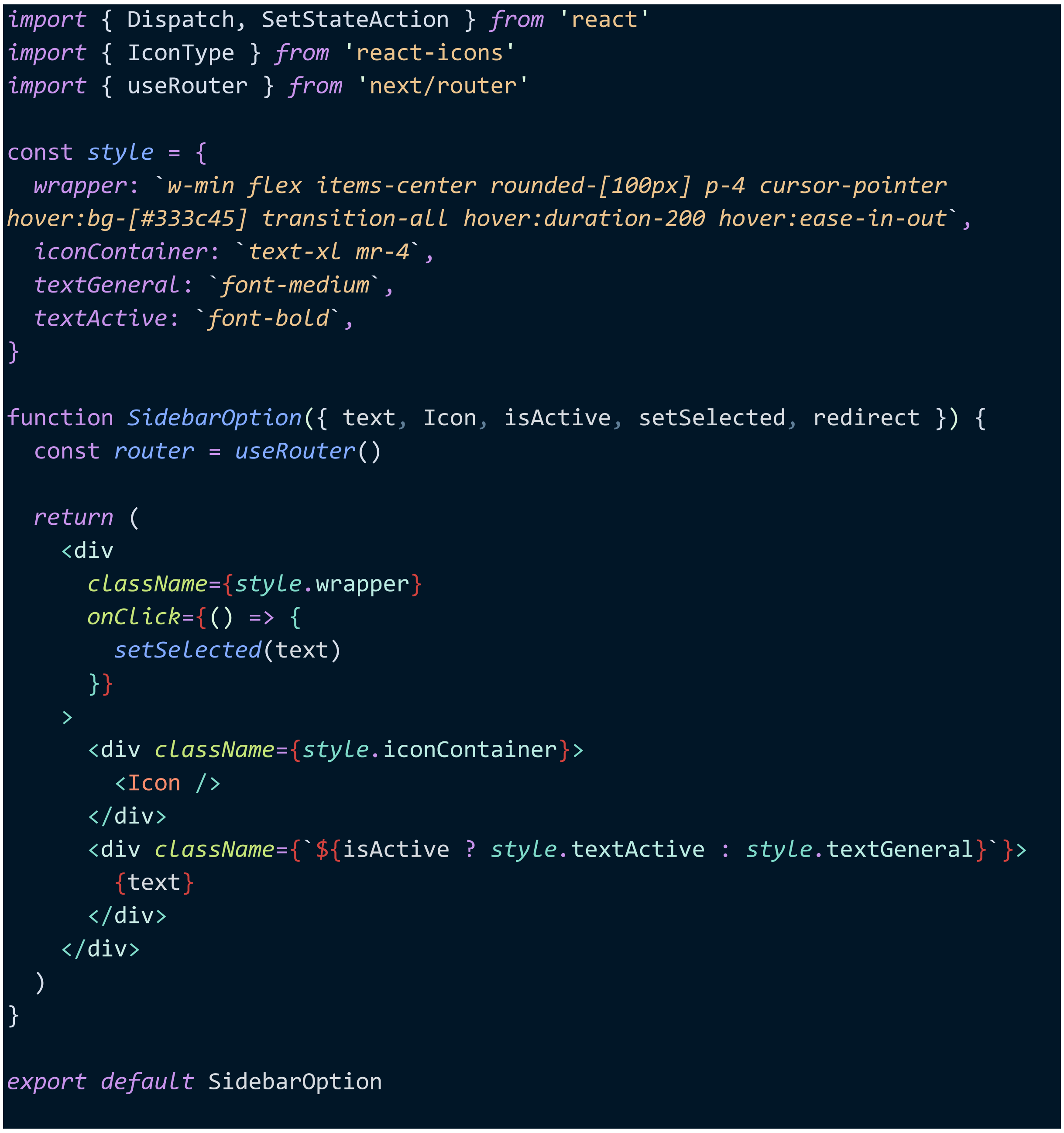
```
import {VscTwitter} from 'react-icons/vsc'
import SidebarOption from './SidebarOption'
import { useState, useContext } from 'react'
import { useRouter } from 'next/router'
import { RiHome7Line, RiHome7Fill, RiFileList2Fill } from 'react-icons/ri'
import { BiHash } from 'react-icons/bi'
import { FiBell, FiMoreHorizontal } from 'react-icons/fi'
import { HiOutlineMail, HiMail } from 'react-icons/hi'
import { FaRegListAlt, FaHashtag, FaBell } from 'react-icons/fa'
import { CgMoreO } from 'react-icons/cg'
import {
  BsBookmark,
  BsBookmarkFill,
  BsPerson,
  BsPersonFill,


function Sidebar({initialSelectedIcon = 'Home'}){
    const [selected, setSelected] = useState(initialSelectedIcon)
    return(
    <div className={style.wrapper}>
      <div className={style.twitterIconContainer}>
        <VscTwitter />
      </div>
      <div className={style.navContainer}>
        <SidebarOption
          Icon={selected === 'Home' ? RiHome7Fill : RiHome7Line}
          text='Home'
          isActive={Boolean(selected === 'Home')}
          setSelected={setSelected}
          redirect={'/'}
        />
        <SidebarOption
          Icon={selected === 'Explore' ? FaHashtag : BiHash}
          text='Explore'
          isActive={Boolean(selected === 'Explore')}
          setSelected={setSelected}
        />
        <SidebarOption
          Icon={selected === 'Notifications' ? FaBell : FiBell}
          text='Notifications'
          isActive={Boolean(selected === 'Notifications')}
          setSelected={setSelected}
        />
        <SidebarOption
          Icon={selected === 'Messages' ? HiMail : HiOutlineMail}
          text='Messages'
          isActive={Boolean(selected === 'Messages')}
          setSelected={setSelected}
        />
        <SidebarOption
          Icon={selected === 'Bookmarks' ? BsBookmarkFill : BsBookmark}
          text='Bookmarks'
          isActive={Boolean(selected === 'Bookmarks')}
          setSelected={setSelected}
        />
        <SidebarOption
          Icon={selected === 'Lists' ? RiFileList2Fill : FaRegListAlt}
          text='Lists'
          isActive={Boolean(selected === 'Lists')}
          setSelected={setSelected}
        />
        <SidebarOption
          Icon={selected === 'Profile' ? BsPersonFill : BsPerson}
          text='Profile'
          isActive={Boolean(selected === 'Profile')}
          setSelected={setSelected}
          redirect={'/profile'}
        />
        <SidebarOption Icon={CgMoreO} text='More' setSelected={setSelected}/>
        <div
          className={style.tweetButton}
        > Mint </div>
      </div>
      <div className={style.profileButton}>
        <div className={style.profileLeft}>
        </div>
        <div className={style.profileRight}>
          <div className={style.details}>
            <div className={style.name}>Aniket Mittal</div>
            <div className={style.handle}>
              @0x22dF ...5xf2df
            </div>
          </div>
          <div className={style.moreContainer}>
            <FiMoreHorizontal />
          </div>
        </div>
      </div>
    </div>
    )
}

export default Sidebar
```

Imports: This file imports necessary React components, hooks (useState, useRouter), potentially context providers, and icon libraries like react-icons to provide visual elements throughout the sidebar.

Sidebar Component: The core Sidebar component renders the sidebar's structure. It utilizes the useState hook to manage the state of the selected navigation option, influencing the sidebar's visual style and potentially redirecting users to different sections of the platform.

Styling (style): This section defines CSS-in-JS styles for the sidebar's layout, buttons, icons, and user profile area.

SidebarOption (imported): A reusable component that represents individual navigation options within the sidebar.



```
import { Dispatch, SetStateAction } from 'react'
import { IconType } from 'react-icons'
import { useRouter } from 'next/router'

const style = {
  wrapper: `w-min flex items-center rounded-[100px] p-4 cursor-pointer
hover:bg-[#333c45] transition-all hover:duration-200 hover:ease-in-out`,
  iconContainer: `text-xl mr-4`,
  textGeneral: `font-medium`,
  textActive: `font-bold`,
}

function SidebarOption({ text, Icon, isActive, setSelected, redirect }) {
  const router = useRouter()

  return (
    <div
      className={style.wrapper}
      onClick={() => {
        setSelected(text)
      }}
    >
      <div className={style.iconContainer}>
        <Icon />
      </div>
      <div className={`${isActive ? style.textActive : style.textGeneral}`}>
        {text}
      </div>
    </div>
  )
}

export default SidebarOption
```

sidebarOption.js

SidebarOption Component: This component renders a single sidebar navigation option. It receives properties like text and Icon to display the option's label and corresponding icon. It also uses an isActive prop to highlight the currently selected option.

Routing: This file utilizes Next.js's useRouter hook to handle navigation when users click on a sidebar option.

The sidebar offers several functionalities that enhance the user experience within the platform:

Navigation: The sidebar provides users with a set of clear and consistent navigation options to access various sections of the platform, like Home, Explore, Notifications, Messages, and potentially others depending on your design. This allows users to switch between different content views effortlessly.

State Management: The Sidebar component manages the selected navigation option using React's state management capabilities. This state variable controls the active styling of the sidebar options and potentially triggers route changes to display the corresponding content section.

User Profile: The sidebar displays the user's profile picture and a truncated version of their blockchain wallet address. This section acts as a quick access point for users to navigate to their full profile page.

Tweet Button: A prominent "Mint" button allows users to initiate the content creation process directly from the sidebar. This functionality may involve minting an NFT to represent the user's content.

Customizable Navigation: By using the reusable Sidebar Option component, I've built a flexible sidebar that can easily accommodate the addition of new navigation links as my social media platform evolves and introduces new features.

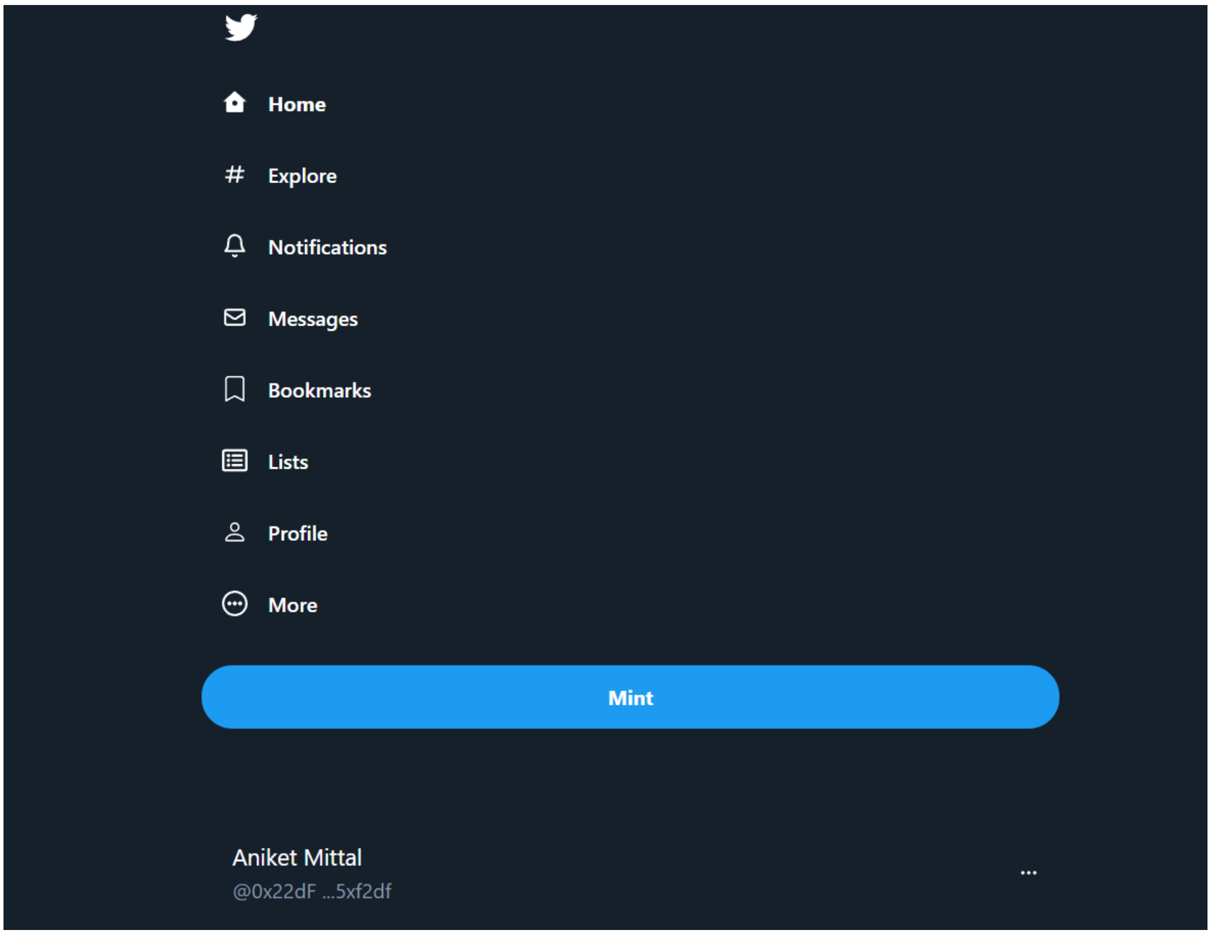


Figure 17. Sidebar Output

FEED COMPONENT

The Feed component forms the centerpiece of my social media platform's user interface, providing a real-time and personalized stream of content. This component dynamically displays posts ("tweets") generated by the users that a particular user follows as well as potentially suggested content generated by others in the network. Here's how it functions:

Code Structure: feed.js

```
import { useContext, useEffect } from 'react'
import TweetBox from "./TweetBox"
import Post from "../Post"
import { BsStars } from 'react-icons/bs'
```

```
function Feed() {
  return (
    <div className={`${style.wrapper} no-scrollbar`}>
      <div className={style.header}>
        <div className={style.headerTitle}>Home</div>
        <BsStars />
      </div>
      <TweetBox />
      {
        tweets.map((tweet, index) => (
            <Post
                key={index} // Use the unique tweet ID
                displayName={tweet.displayName}
                userName={`${tweet.userName.slice( 0, 4,
)}...${tweet.userName.slice(-4)}`}
                text={tweet.text}
                avatar={tweet.avatar}
                isProfileImageNft={tweet.isProfileImageNft}
                timestamp={tweet.timestamp}
            />
        ))

    </div>
  )
}

export default Feed
```

Structure: Defines the Feed component structure, including the header and containers for the TweetBox and dynamically generated Post components.

Data Fetching (Commented Out): Commented sections indicate places where you would interact with either a traditional database or your blockchain network to fetch the relevant content data to populate the feed.

Tweet Display: The Feed iterates through an array of tweets (or similarly structured data). For each tweet, it renders a Post component, passing the necessary content data as props.

post.js

```
import { BsFillPatchCheckFill } from 'react-icons/bs'
import { FaRegComment, FaRetweet } from 'react-icons/fa'
import { AiOutlineHeart } from 'react-icons/ai'
import { FiShare } from 'react-icons/fi'
import { format } from 'timeago.js'
import { useState } from 'react'


const Post = ({
  displayName,
  userName,
  text,
  avatar,
  timestamp,
  isProfileImageNft,
}) => {

  return (
    <div className={style.wrapper}>
      <div>
        <img
        //   src={profileImageLink}
         src={avatar}
          alt={userName}
          className={
            isProfileImageNft
              ? `${style.profileImage} smallHex`
              : style.profileImage
          }
        />
      </div>
      <div className={style.postMain}>
        <div>
          <span className={style.headerDetails}>
            <span className={style.name}>{displayName}</span>
            {isProfileImageNft && (
              <span className={style.verified}>
                <BsFillPatchCheckFill />
              </span>
            )}
            <span className={style.handleAndTimeAgo}>
              @{userName} • {format(new Date(timestamp).getTime())}
            </span>
          </span>
          <div className={style.tweet}>{text}</div>
        </div>
        <div className={style.footer}>
          <div
            className={`${style.footerIcon} hover:text-[#1d9bf0] hover:bg-
[#1e364a]`}
          >
            <FaRegComment />
          </div>
          <div
            className={`${style.footerIcon} hover:text-[#03ba7c] hover:bg-
[#1b393b]`}
          >
          >
            <FiShare />
          </div>
        </div>
      </div>
    </div>
  )
}

export default Post
```

Post Layout: Renders the visual representation of an individual post, including the user's profile picture, display name, username, the post's text content, and any associated images or media.

Metadata: Displays post metadata, such as the timestamp and engagement icons (for liking, commenting, sharing).

Blockchain Indicators: Includes visual cues (BsFillPatchCheckFill) to subtly indicate users who have verified NFT-based profile pictures. This helps establish authenticity and potentially encourages the use of NFTs for ownership and identity verification within your platform.

tweetbox.js

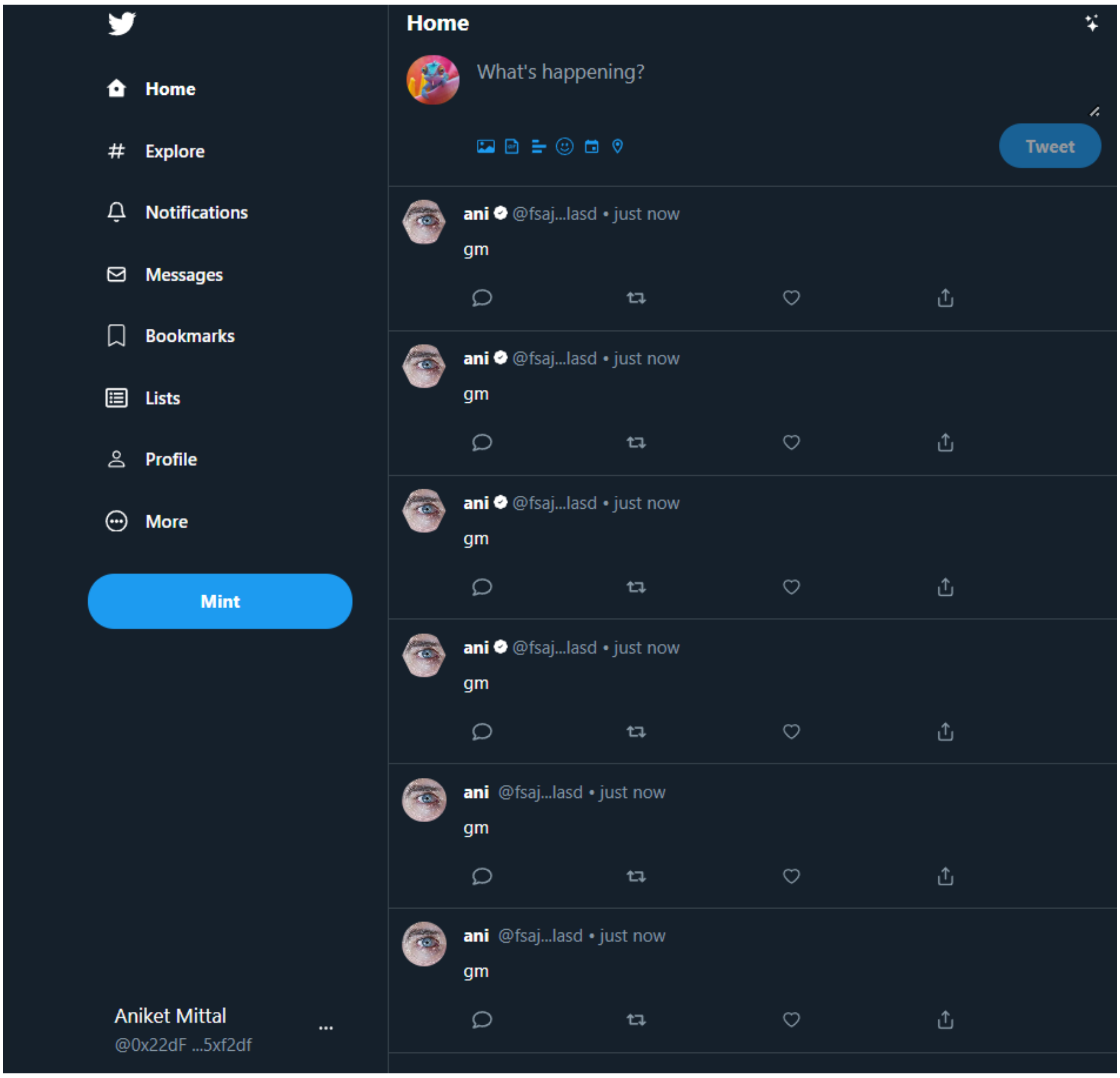
```
import { useState, useContext } from 'react'
import { BsCardImage, BsEmojiSmile } from 'react-icons/bs'
import { RiFileGifLine, RiBarChartHorizontalFill } from 'react-icons/ri'
import { IoMdCalendar } from 'react-icons/io'
import { MdOutlineLocationOn } from 'react-icons/md'
```

```
function TweetBox() {
  const [tweetMessage, setTweetMessage] = useState('')
  const postTweet = (e)=>{
    e.preventDefault()
    console.log(tweetMessage)
  }

  return (
    <div className={style.wrapper}>
      <div className={style.tweetBoxLeft}>
        <img className={style.profileImage}
src="data:image/jpeg;base64,/9j/4AAQSk " alt="frog" />
      </div>
      <div className={style.tweetBoxRight}>
        <form>
          <textarea
            onChange={e => setTweetMessage(e.target.value)}
            value={tweetMessage}
            placeholder="What's happening?"
            className={style.inputField}
          />
          <div className={style.formLowerContainer}>
            <div className={style.iconsContainer}>
              <BsCardImage className={style.icon} />
              <RiFileGifLine className={style.icon} />
              <RiBarChartHorizontalFill className={style.icon} />
              <BsEmojiSmile className={style.icon} />
              <IoMdCalendar className={style.icon} />
              <MdOutlineLocationOn className={style.icon} />
            </div>
            <button
              type='submit'
            //   onClick={event => submitTweet(event)}
              onClick={event => postTweet(event)}
            //   disabled={!tweetMessage}
              className={`${style.submitGeneral} ${
                tweetMessage ? style.activeSubmit : style.inactiveSubmit
              }`}
            >
              Tweet
            </button>
          </div>
        </form>
      </div>
    </div>
  )
}
```

```
export default TweetBox
```

Input: Provides a prominent text area for users to compose new posts ("tweets").

Submit Button: A Tweet button allows users to publish their content to the social media platform. This button's state (activeSubmit/inactiveSubmit) changes based on whether there's content to post, providing visual feedback.



ani @fsaj...lasd • just now
gm

ani @fsaj...lasd • just now
gm

ani @fsaj...lasd • just now
gm

ani @fsaj...lasd • just now
gm

ani @fsaj...lasd • just now
gm

ani @fsaj...lasd • just now
gm

Figure 18. Feed Section Output

WIDGETS COMPONENT

The Widgets component serves as a supplementary sidebar area designed to enhance user engagement and content discovery within my social media platform. It occupies a less prominent position on the screen compared to the primary Feed component. This component houses several features that provide additional context and opportunities for users to explore different aspects of the platform:

Functionality

Trending Topics: The Widgets component displays a dynamic list of trending hashtags or topics determined by user activity within the platform. This feature provides social proof and highlights popular conversations, allowing users to easily jump into discussions relevant to their interests.

Suggested Accounts: The Widgets component can showcase a list of suggested accounts for a user to follow. These suggestions might be based on a user's existing connections, shared interests derived from post content, or potentially on-chain data such as shared NFT ownership or participation in the same DAOs. This feature promotes the discovery of new and relevant accounts, fostering a sense of community and expanding a user's network within the platform.

```
import { news, whoToFollow } from '../lib/static'
import { BiSearch } from 'react-icons/bi'

const style = {
  wrapper: `flex-[1] p-4`,
  searchBar: `flex items-center bg-[#243340] p-2 rounded-3xl`,
  searchIcon: `text-[#8899a6] mr-2`,
  inputBox: `bg-transparent outline-none`,
  section: `bg-[#192734] my-6 rounded-xl overflow-hidden`,
  title: `p-2 font-bold text-lg`,
  showMore: `p-2 text-[#1d9bf0] text-sm cursor-pointer hover:bg-[#22303c]`,
  item: `flex items-center p-3 my-2 hover:bg-[#22303c] cursor-pointer`,
  newsItemLeft: `flex-1`,
  newsItemCategory: `text-[#8899a6] text-xs font-semibold`,
  newsItemTitle: `text-sm font-bold`,
  newsItemRight: `w-1/5 ml-3`,
  newsItemImage: `rounded-xl h-14 w-14 object-cover`,
  followAvatarContainer: `w-1/6`,
  followAvatar: `rounded-full h-[40px] w-[40px]`,
  profileDetails: `flex-1`,
  name: `font-bold`,
  handle: `text-[#8899a6]`,
  followButton: `bg-white text-black px-3 py-1 rounded-full text-xs font-
bold`,
}

function Widgets() {
  return (
    <div className={style.wrapper}>
      <div className={style.searchBar}>
        <BiSearch className={style.searchIcon} />
        <input
          placeholder='Search Twitter'
          type='text'
          className={style.inputBox}
        />
      </div>
      <div className={style.section}>
        <div className={style.title}>What's happening</div>
        {news.map((item, index) => (
          <div key={index} className={style.item}>
            <div className={style.newsItemLeft}>
              <div className={style.newsItemCategory}>{item.category}</div>
              <div className={style.newsItemTitle}>{item.title}</div>
            </div>
            <div className={style.newsItemRight}>
              <img
                src={item.image}
                alt={item.category}
                className={style.newsItemImage}
              />
            </div>
          </div>
        ))}
        <div className={style.showMore}>Show more</div>
      </div>
      <div className={style.section}>
        <div className={style.title}>Who to follow</div>
        {whoToFollow.map((item, index) => (
          <div key={index} className={style.item}>
            <div className={style.followAvatarContainer}>
              <img
                src={item.avatar}
                alt={item.handle}
                className={style.followAvatar}
              />
            </div>
            <div className={style.profileDetails}>
              <div className={style.name}>{item.name}</div>
              <div className={style.handle}>{item.handle}</div>
            </div>
            <div className={style.followButton}>Follow</div>
          </div>
        ))}
        <div className={style.showMore}>Show more</div>
      </div>
    </div>
  )
}

export default Widgets
```

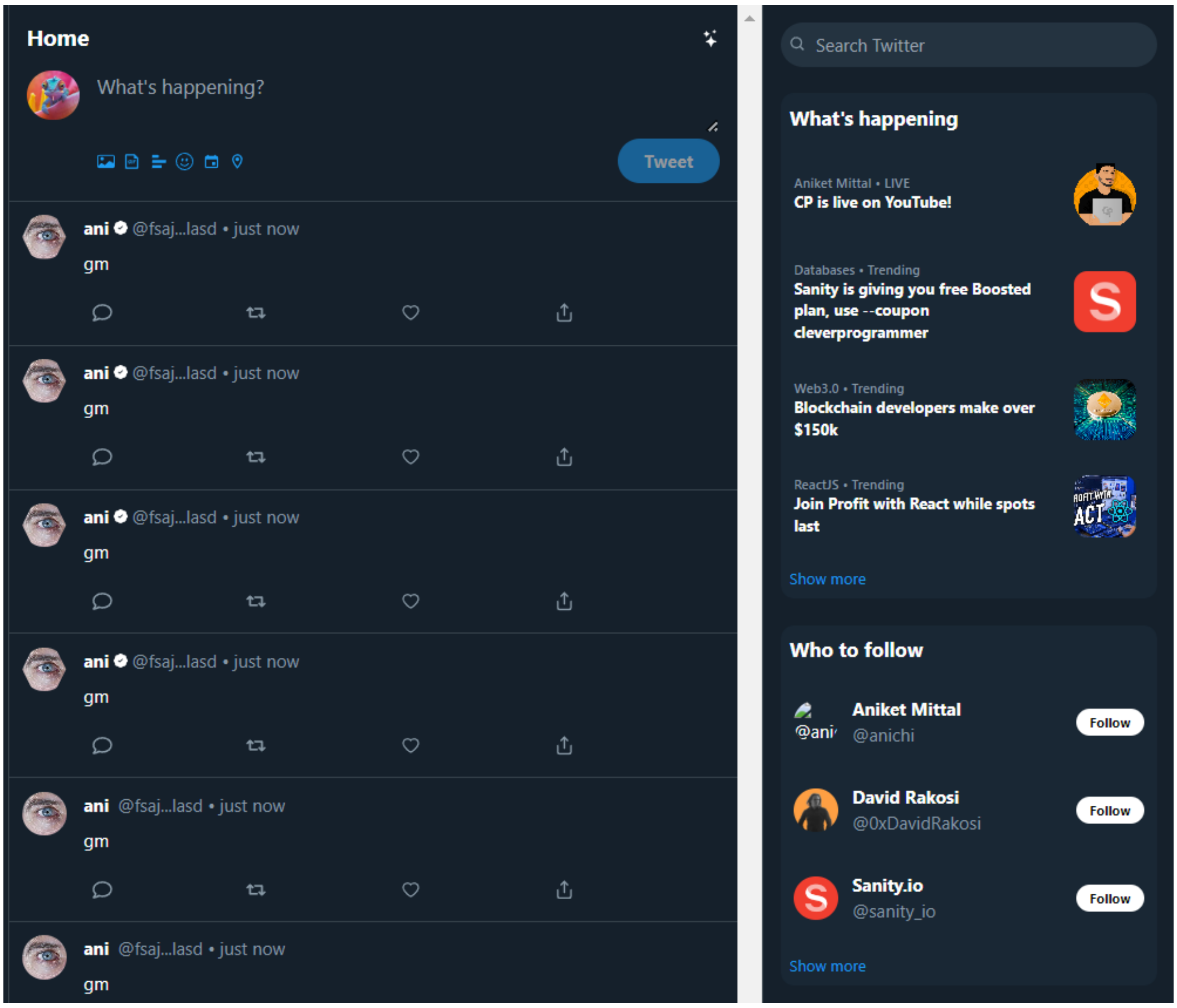

Home
What's happening?
Tweet
ani @fsaj...lasd • just now
gm
ani @fsaj...lasd • just now
gm
ani @fsaj...lasd • just now
gm
ani @fsaj...lasd • just now
gm
ani @fsaj...lasd • just now
gm
ani @fsaj...lasd • just now
gm
Search Twitter
What's happening
Aniket Mittal • LIVE
CP is live on YouTube!
Databases • Trending
Sanity is giving you free Boosted plan, use --coupon cleverprogrammer
Web3.0 • Trending
Blockchain developers make over $150k
ReactJS • Trending
Join Profit with React while spots last
Show more
Who to follow
Aniket Mittal @anichi
Follow
David Rakosi @0xDavidRakosi
Follow
Sanity.io @sanity_io
Follow
Show more

Figure 19. Widgets Section Output

SMART CONTRACT DEVELOPMENT

ProfileImageNfts Smart Contract

This Solidity smart contract provides the core functionality for minting and managing NFTs that serve as profile images within your social media platform. Let's delve into its key features:

ERC721 Compliance and Ownable Traits

ERC721 Standard: The contract adheres to the widely adopted ERC721 standard, ensuring seamless integration with existing NFT marketplaces and wallets within the Ethereum ecosystem. This enables users to leverage familiar tools and services to buy, sell, and manage their profile image NFTs.

Ownable Contract: The contract inherits from OpenZeppelin's Ownable contract, allowing you to designate an owner address for the contract. This grants the owner exclusive control over sensitive operations like minting new NFTs. You can choose to centralize minting authority with yourself during initial deployment or potentially loosen restrictions in the future to allow users to mint their own profile image NFTs.

Minting Functionality

mint () Function: This function serves as the entry point for creating new profile image NFTs. It's designed to be called by authorized users (the contract owner by default, but this can be extended).

_mint (): Internally, the mint() function calls ERC721's _mint() function to initiate the NFT creation process. This assigns a unique token ID to the newly minted NFT.

_tokenIds Counter: The contract employs a counter (_tokenIds) to ensure that each minted NFT receives a unique ID. This is essential for traceability, preventing conflicts, and simplifying the management of NFTs within the social media platform.

_setTokenURI(): This function plays a crucial role in associating metadata with the minted NFT. It takes two arguments: the NFT's unique ID (tokenId) and a Uniform Resource Identifier (URI) that points to the NFT's metadata location. This metadata likely resides off-chain, potentially stored on a decentralized storage solution like IPFS. By storing the URI on-chain, the smart contract establishes a permanent link between the NFT and its associated data, even if the off-chain data storage provider undergoes changes. The metadata itself can include details such as the profile image itself (if stored on-chain), a reference to an off-chain image location, or other relevant attributes like the image's creator or creation date.

Retrieving NFT Metadata

tokenURI() Function: This function overrides the standard ERC721 tokenURI() function. It takes an NFT's tokenId as input and returns the corresponding URI stored using _setTokenURI(). This allows external applications and your frontend to query the on-chain smart contract to retrieve essential details about a particular profile image NFT.

getAlltoken() Function (Optional): This function provides a way to fetch data for all minted NFTs in a single operation. It iterates through all existing NFTs (up to a certain limit) and retrieves their associated URIs. This can be useful for your frontend to efficiently populate a user interface element that displays a user's collection of profile image NFTs.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.2;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";

contract ProfileImageNfts is ERC721, Ownable {

    using Counters for Counters.Counter;
    using Strings for uint256;

    Counters.Counter _tokenIds;
    mapping(uint256 => string) _tokenURIs;

    struct RenderToken{
        uint256 id;
        string uri;
        string space;
    }

    constructor() ERC721("ProfileImageNfts","PIN"){}

    function _setTokenURI(uint256 tokenId, string memory _tokenURI) internal {
        _tokenURIs[tokenId] = _tokenURI;
    }

    function tokenURI(uint256 tokenId) public view virtual override returns
(string memory) {
        require(_exists(tokenId),"URI not exist on that ID");
        string memory _RUri =  _tokenURIs[tokenId];
        return _RUri;
    }

    function getAlltoken() public view returns (RenderToken[] memory){
        uint256 latestId = _tokenIds.current();
        RenderToken[] memory res = new RenderToken[](latestId);
        for(uint256 i = 0; i  <= latestId ; i++){
            if(_exists(i)){
                string memory uri = tokenURI(i);
                res[i] = RenderToken(i,uri," ");
            }
        }
        return res;
    }

    function mint(address recipents, string memory _uri) public returns
(uint256){
        uint256 newId = _tokenIds.current();
        _mint(recipents,newId);
        _setTokenURI(newId,_uri);
        _tokenIds.increment();
        return newId;
    }
}
```

METAMASK AUTHENTICATION

```
const checkIfWalletIsConnected = async () => {
    if (!window.ethereum) return setAppStatus('noMetaMask')
    try {
      const addressArray = await window.ethereum.request({
        method: 'eth_accounts',
      })
      if (addressArray.length > 0) {
        setAppStatus('connected')
        setCurrentAccount(addressArray[0])
        // createUserAccount(addressArray[0])
      } else {
        router.push('/')
        setAppStatus('notconnected')
      }
    } catch (err) {
    //   router.push('/')
      setAppStatus('error')
    }
  }
```

The checkIfWalletIsConnected function serves as a critical entry point for verifying a user's MetaMask connection status upon their arrival at your social media application. It executes as soon as the user's browser loads the application.

Process Breakdown

MetaMask Detection: The function initiates by checking for the presence of MetaMask within the user's browser environment. It achieves this by examining the existence of a window.ethereum object. This object serves as a bridge between your web application and the MetaMask extension. If the window.ethereum object is absent, it signifies that MetaMask is not installed or not enabled on the user's browser. In this scenario, the function sets the application's state variable, appStatus, to "noMetaMask", indicating the lack of MetaMask.

Requesting Account Access (if MetaMask is Available): If the function successfully detects MetaMask, it proceeds to request access to the user's accounts. It accomplishes this by making a request using window.ethereum.request({ method: 'eth_accounts'}). This essentially prompts the user with a MetaMask pop-up window, asking them for permission to connect your social media application to their MetaMask wallet. The user has the اختیار (ikhtiyar, Urdu for "choice") to either grant or deny this connection request.

Updating Application State: Based on the user's response to the MetaMask pop-up, the function updates the appStatus variable to reflect the current connection status:

connected: This value is set if the user approves the connection request, signifying a successful establishment of a connection between your application and the user's MetaMask wallet. The user's wallet address is likely retrieved at this point and stored for subsequent use within your application.

notConnected: This value is set if the user either rejects the connection request or no accounts are detected within the user's MetaMask wallet. This scenario indicates that the user is not connected to your application at this point.

error: An error value is assigned to appStatus if any errors occur during the MetaMask connection process. This could be due to various reasons, such as communication issues between the application and MetaMask or the user encountering problems within the MetaMask extension itself.

```
const connectWallet = async () => {
    if (!window.ethereum) return setAppStatus('noMetaMask')
    try {
      setAppStatus('loading')

      const addressArray = await window.ethereum.request({
        method: 'eth_requestAccounts',
      })

      if (addressArray.length > 0) {
        setCurrentAccount(addressArray[0])
        createUserAccount(addressArray[0])
      } else {
        router.push('/')
        setAppStatus('notConnected')
      }
    } catch (err) {
      setAppStatus('error')
    }
  }
```

This function initiates the MetaMask connection flow when a user clicks on a "Connect Wallet" button.

Checks for MetaMask: Similar to checkIfWalletIsConnected.

Requests Accounts: Uses window.ethereum.request({ method: 'eth_requestAccounts'}) to trigger the MetaMask pop-up, prompting the user to connect their wallet.

Sets Application Status: Manages the state variable appStatus to reflect loading status, successful connection, or error states.

```
const Home = () => {
  const { appStatus, connectWallet } = useContext(TwitterContext)

  const app = (status = appStatus) => {
    switch (status) {
      case 'connected':
        return userLoggedIn

      case 'notConnected':
        return noUserFound

      case 'noMetaMask':
        return noMetaMaskFound

      case 'error':
        return error

      default:
        return loading
    }
  }

  const userLoggedIn = (
    <div className={style.content}>
      <Sidebar initialSelectedIcon={'Home'} />
      <Feed />
      <Widgets />
    </div>
  )

  const noUserFound = (
    <div className={style.loginContainer}>
      <Image src={metamaskLogo} width={200} height={200} />
      <div
        className={style.walletConnectButton}
        onClick={() => connectWallet()}
      >
        Connect Wallet
      </div>
      <div className={style.loginContent}>Connect to Metamask.</div>
    </div>
  )

  const noMetaMaskFound = (
    <div className={style.loginContainer}>
      <Image src={metamaskLogo} width={200} height={200} />
      <div className={style.loginContent}>
        <a
```

```
          target='_blank'
          rel='noreferrer'
          href={`https://metamask.io/download.html`}
        >
          You must install Metamask, a <br /> virtual Ethereum wallet, in your
          browser.
        </a>
      </div>
    </div>
  )

  const error = (
    <div className={style.loginContainer}>
      <Image src={errorImg} width={250} height={200} />
      <div className={style.loginContent}>
        An error occurred. Please try again later or from another browser.
      </div>
    </div>
  )

  const loading = (
    <div className={style.loginContainer}>
      <div className={style.loginContent}>Loading...</div>
    </div>
  )

  return <div className={style.wrapper}>{app(appStatus)}</div>
}
```

Application State (appStatus): The component's central logic relies on the appStatus variable, likely managed globally through a context provider like React Context.

Conditional Rendering: It uses a switch statement to determine what to display to the user based on the appStatus:

connected: Renders the full social media UI (userLoggedIn)

notConnected: Displays a "Connect Wallet" screen (noUserFound)

noMetaMask: Prompts to install MetaMask (noMetaMaskFound)

error: Show an error message (error)

loading: Show an interim loading screen (loading).

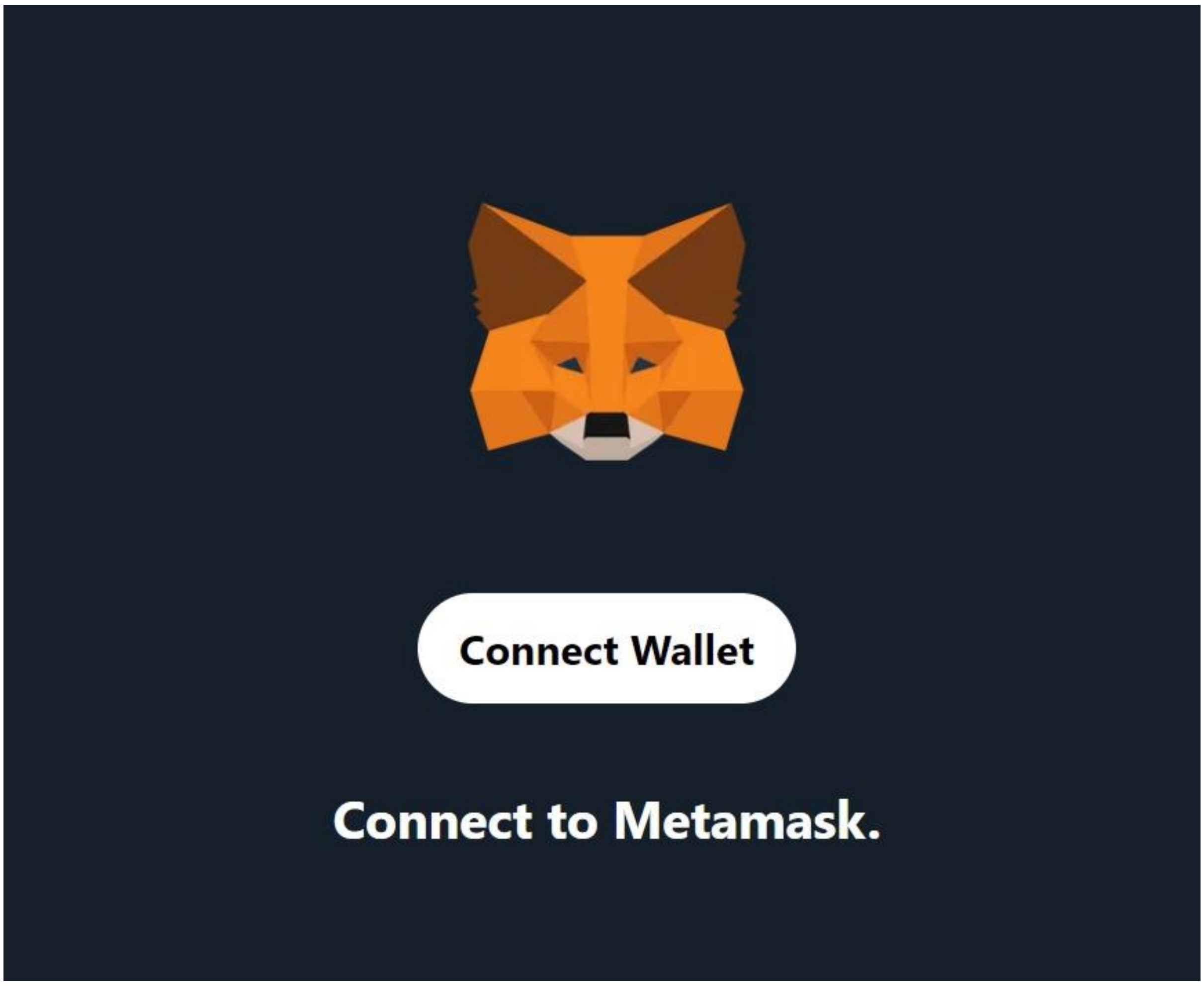


Figure 20. MetaMask Authentication

BACKEND: USER MANAGEMENT, DATA PERSISTENCE, AND BLOCKCHAIN INTERACTION

My backend architecture leverages Next.js for its server-side capabilities and integrates seamlessly with Sanity.io, a powerful headless CMS, to provide essential data management and user authentication features for my social media platform. Here's a breakdown of the key functionalities in my backend implementation:

User Authentication and Profiles

```
const checkIfWalletIsConnected = async () => {
    if (!window.ethereum) return setAppStatus('noMetaMask')
    try {
      const addressArray = await window.ethereum.request({
```

```
      method: 'eth_accounts',
    })
    if (addressArray.length > 0) {
      setAppStatus('connected')
      setCurrentAccount(addressArray[0])
      createUserAccount(addressArray[0])
    } else {
      router.push('/')
      setAppStatus('notconnected')
    }
  } catch (err) {
    // console.log('err1')
    router.push('/')
    setAppStatus('error')
  }
}

/**
 * Initiates MetaMask wallet connection
 */
const connectToWallet = async () => {
  if (!window.ethereum) return setAppStatus('noMetaMask')
  try {
    setAppStatus('loading')
    const addressArray = await window.ethereum.request({
      method: 'eth_requestAccounts',
    })

    if (addressArray.length > 0) {
      setCurrentAccount(addressArray[0])
      createUserAccount(addressArray[0])
    } else {
      router.push('/')
      setAppStatus('notconnected')
    }
  } catch (err) {
    setAppStatus('error')
  }
}
```

MetaMask Integration (checkIfWalletIsConnected, connectToWallet): The backend code establishes a crucial link with MetaMask. It detects connected wallets, prompts users to connect their wallets if necessary, and extracts the user's Ethereum wallet address. This address serves as the primary identifier for users within the platform.

Sanity.io User Documents (createUserAccount): When a user connects their wallet, the backend checks Sanity.io for an existing user document. If none exists, it automatically creates a new document containing basic user information:

_type: 'users': Classifies the document as a 'user' type within Sanity.

_id: The user's wallet address for unique identification.

name, profileImage, walletAddress: Stores display name, profile picture URL, and wallet address.

isProfileImageNft: A flag to indicate if a user has a tokenized NFT profile picture.

Tweet Fetching and Management (fetchTweets)

Sanity.io Queries: The backend uses Sanity's GROQ query language to retrieve all tweets stored within the database. The query orders tweets chronologically (by timestamp).

NFT Profile Picture Handling (getNftProfileImage): This function enriches the tweet data. If a tweet's author has an NFT profile picture (indicated by the isProfileImageNft flag), the code fetches the full NFT image URL from a decentralized storage provider like IPFS using a gateway. Otherwise, it retains the standard image URL.

Tweet Transformation: The code neatly repackages the data. It combines the tweet content with author information, correctly formatted profile picture URLs, and other relevant metadata to be seamlessly consumed by the frontend components.

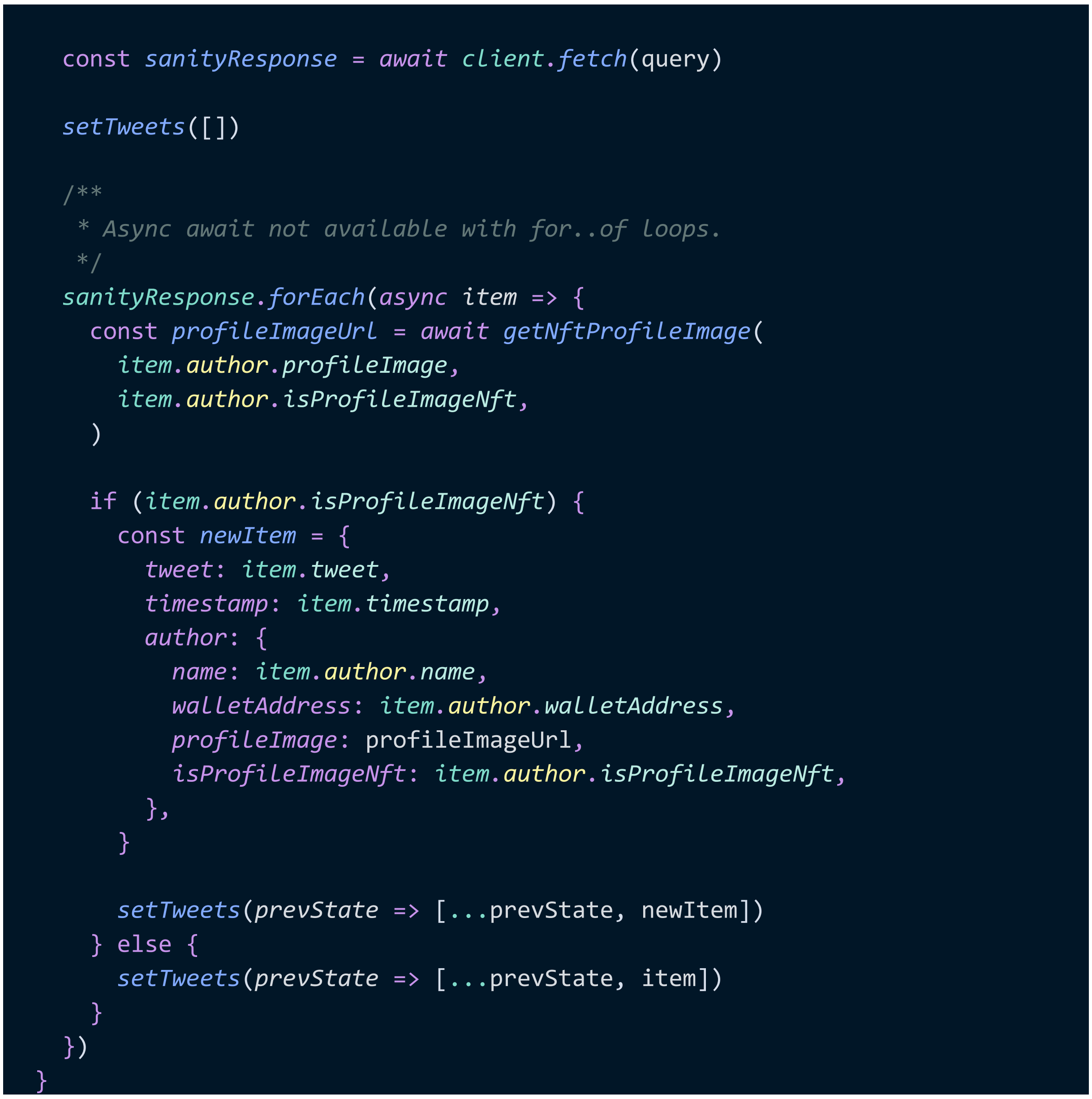
Current User Details (getCurrentUserDetails)

```
const getCurrentUserDetails = async (userAccount = currentAccount) => {
    if (appStatus !== 'connected') return

    const query = `
      *[_type == "users" && _id == "${userAccount}"]{
        "tweets": tweets[]->{timestamp, tweet}|order(timestamp desc),
        name,
        profileImage,
        isProfileImageNft,
        coverImage,
        walletAddress
      }
    `

    const response = await client.fetch(query)

    const profileImageUri = await getNftProfileImage(
      response[0].profileImage,
      response[0].isProfileImageNft,
    )

    setCurrentUser({
      tweets: response[0].tweets,
      name: response[0].name,
      profileImage: profileImageUri,
      walletAddress: response[0].walletAddress,
      coverImage: response[0].coverImage,
      isProfileImageNft: response[0].isProfileImageNft,
    })
  }
```

Profile Fetching: Queries Sanity.io to retrieve the complete profile of the currently logged-in user (identified by their wallet address). This includes their name, profile image, a list of their past tweets, and potentially a cover image.

The TwitterContext

```
const getNftProfileImage = async (imageUri, isNft) => {
    if (isNft) {
      return `https://gateway.pinata.cloud/ipfs/${imageUri}`
    } else if (!isNft) {
      return imageUri
    }
  }


  /**
   * Gets all the tweets stored in Sanity DB.
   */
  const fetchTweets = async () => {
    const query = `
      *[_type == "tweets"]{
        "author": author->{name, walletAddress, profileImage,
isProfileImageNft},
        tweet,
        timestamp
      }|order(timestamp desc)
    `


    // setTweets(await client.fetch(query))
```



```
  const sanityResponse = await client.fetch(query)

  setTweets([])

  /**
   * Async await not available with for..of loops.
   */
  sanityResponse.forEach(async item => {
    const profileImageUrl = await getNftProfileImage(
      item.author.profileImage,
      item.author.isProfileImageNft,
    )

    if (item.author.isProfileImageNft) {
      const newItem = {
        tweet: item.tweet,
        timestamp: item.timestamp,
        author: {
          name: item.author.name,
          walletAddress: item.author.walletAddress,
          profileImage: profileImageUrl,
          isProfileImageNft: item.author.isProfileImageNft,
        },
      }

      setTweets(prevState => [...prevState, newItem])
    } else {
      setTweets(prevState => [...prevState, item])
    }
  })
}
```

Context Provider: I use React's Context API to make authentication state (appStatus, currentAccount), user data (currentUser), tweets (tweets), and essential functions (connectToWallet, fetchTweets, getNftProfileImage) easily accessible throughout my application's frontend components. This avoids the need to manually pass data down multiple component levels.

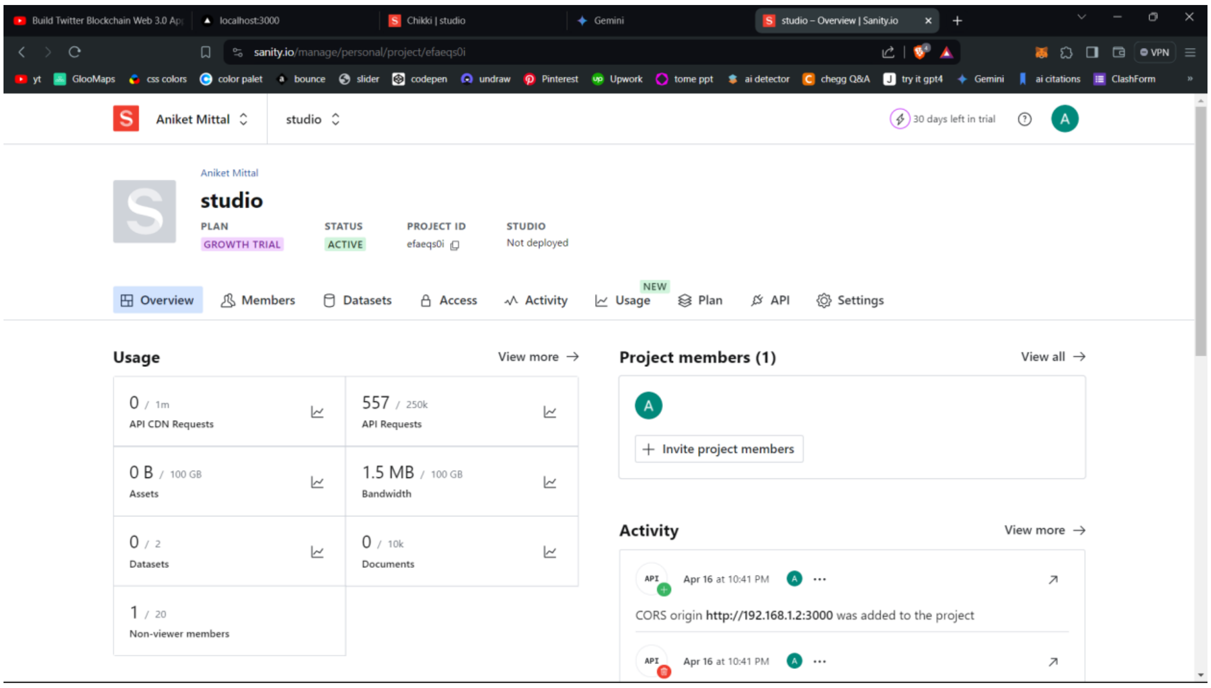
Content Management with Sanity.io

Sanity.io serves as the content backbone for my social media platform. Its structured data store and intuitive dashboard provide a streamlined way to manage essential user and content data within my application. Key data types modeled in Sanity include:

User Profiles: I store each user's profile information (display name, wallet address, profile image) as individual documents, enabling easy retrieval and updates.

Tweets: Every post made on the platform is stored as a structured document, including content, author information, and timestamps. This allows me to query and display tweets in a flexible manner.

Potential Future Data: Sanity.io's flexibility allows me to easily define schemas for additional data types as my platform evolves (e.g., reputation scores, follower relationships, or direct messages).



Aniket Mittal

studio

PLAN: GROWTH TRIAL · STATUS: ACTIVE · PROJECT ID: efaeqs0i · STUDIO: Not deployed

Usage

0 / 1m API CDN Requests	557 / 250k API Requests
0 B / 100 GB Assets	1.5 MB / 100 GB Bandwidth
0 / 2 Datasets	0 / 10k Documents
1 / 20 Non-viewer members	

Project members (1)

Activity

Apr 16 at 10:41 PM

CORS origin http://192.168.1.2:3000 was added to the project

Apr 16 at 10:41 PM

Figure 21. Database Dashboard

ENABLING NFT-BASED PROFILE IMAGES

My social media platform empowers users to showcase their Web3 identities by integrating NFT-based profile image minting. To provide this functionality, I’ve built a custom minting component (ProfileImageMinter). Collaborating seamlessly with external services, thiscomponent guides users through the NFT creation process, ultimately linking their newly minted profile image NFT to their on-chain social profile.

Let's dissect the core components that facilitate this feature:

ProfileImageMinter.js: The central component responsible for providing the user interface and orchestrating the minting process. It handles image uploads, metadata creation, interaction with the Ethereum blockchain, and updates to Sanity.io as needed.

pinata.js: Interacts with the Pinata IPFS service. This code is crucial for storing both profile image data and the NFT's metadata on IPFS, establishing a decentralized and resilient storage solution for these assets.

InitialState.js, LoadingState.js, FinishedState.js: These files render different UI states during the minting process, enhancing user experience by providing visual cues and feedback.

How these components work together:

User Input and Preparation: The user journey begins in InitialState.js where they select an image for their profile picture and provide a captivating title and description. This information will be incorporated into the NFT metadata, enriching the on-chain representation of their profile.

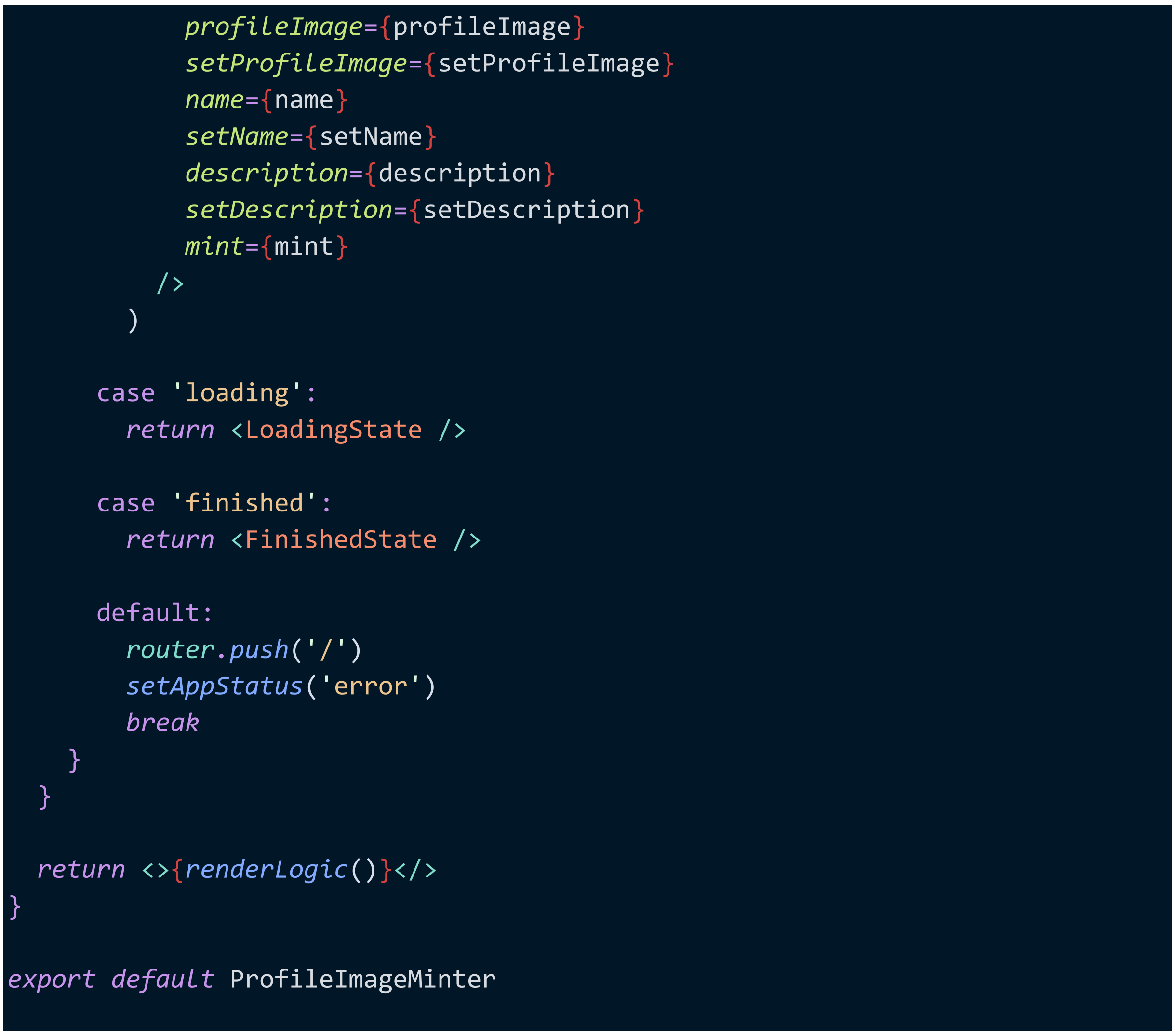
```
const InitialState = ({
  profileImage,
  setProfileImage,
  name,
  setName,
  description,
  setDescription,
  mint,
}) => {
  console.log(profileImage)

  return (
    <div className={style.wrapper}>
      <div className={style.inputFieldsContainer}>
        <div className={style.inputContainer}>
          <label
            htmlFor='image-upload'
            className={profileImage ? style.fileSelected : style.customInput}
          >
            <input
              type='file'
              id='image-upload'
              accept='.jpg, .jpeg, .png'
              className={style.fileInput}
              placeholder='Image URL'
              onChange={e => setProfileImage(e.target.files[0])}
            />
            Select File
          </label>
        </div>
        <div className={style.inputContainer}>
          <input
            type='text'
            className={style.input}
            placeholder='Title of Image'
            value={name}
            onChange={e => setName(e.target.value)}
          />
        </div>
        <div className={style.inputContainer}>
          <input
            type='text'
            className={style.input}
            placeholder='Description'
            value={description}
            onChange={e => setDescription(e.target.value)}
          />
        </div>
      </div>
      <div className={style.lower}>
        <div className={style.visibility}>
          <GiEarthAmerica />
          <span className={style.visibilityText}>Everyone can see this</span>
        </div>
        <div
          className={
            name && description && profileImage
              ? style.mintButton
              : style.inactiveMintButton
          }
          onClick={() => {
            if (name && description && profileImage) {
              mint()
            }
          }}
        >
          Mint
        </div>
      </div>
    </div>
  )
```

IPFS Storage via Pinata: Once the user confirms their selections, ProfileImageMinter.js takes over. It leverages the pinFileToIPFS function from pinata.js to securely upload the chosen image to the Pinata IPFS service. Pinata ensures the image remains permanently accessible and decentralized. In parallel, pinJSONToIPFS from pinata.js uploads another critical piece of data – an NFT metadata JSON object – to IPFS. This JSON object contains the profile picture's IPFS hash, along with the user-provided title and description.

```
export const pinJSONToIPFS = async json => {
  const url = `https://api.pinata.cloud/pinning/pinJSONToIPFS`
  return axios
    .post(url, json, {
      headers: {
        pinata_api_key: key,
        pinata_secret_api_key: secret,
      },
    })
    .then(function (response) {
      return response.data.IpfsHash
    })
    .catch(function (error) {
      console.log(error)
    })
}

export const pinFileToIPFS = async (file, pinataMetaData) => {
  const url = `https://api.pinata.cloud/pinning/pinFileToIPFS`

  let data = new FormData()

  data.append('file', file)
  data.append('pinataMetadata', JSON.stringify(pinataMetaData))
return axios
    .post(url, data, {
      maxBodyLength: 'Infinity',
      headers: {
        'Content-Type': `multipart/form-data; boundary=${data._boundary}`,
        pinata_api_key: key,
        pinata_secret_api_key: secret,
      },
    })
    .then(function (response) {
      return response.data.IpfsHash
    })
    .catch(function (error) {
      console.log(error)
    })
```

Blockchain Minting: With both the image and metadata safely stored on IPFS, ProfileImageMinter.js calls the getEthereumContract function to interact with the deployed smart contract on the Ethereum blockchain. This smart contract is responsible for the actual NFT minting process. The transaction includes a reference to the IPFS hash of the NFT metadata, which permanently links the minted NFT with the user's profile information and image.

```
const getEthereumContract = () => {
  if (!metamask) return
  const provider = new ethers.providers.Web3Provider(metamask)
  const signer = provider.getSigner()
  const transactionContract = new ethers.Contract(
    contractAddress,
    contractABI,
    signer,
  )

  return transactionContract
}

const ProfileImageMinter = () => {
  const { currentAccount, setAppStatus } = useContext(TwitterContext)
  const router = useRouter()

  const [name, setName] = useState('')
  const [description, setDescription] = useState('')
  const [status, setStatus] = useState('initial')
  const [profileImage, setProfileImage] = useState()

  const mint = async () => {
    if (!name || !description || !profileImage) return
    setStatus('loading')

    const pinataMetaData = {
      name: `${name} - ${description}`,
    }

    const ipfsImageHash = await pinFileToIPFS(profileImage, pinataMetaData)

    await client
      .patch(currentAccount)
      .set({ profileImage: ipfsImageHash })
      .set({ isProfileImageNft: true })
      .commit()

    const imageMetaData = {
      name: name,
      description: description,
      image: `ipfs://${ipfsImageHash}`,
    }

    const ipfsJsonHash = await pinJSONToIPFS(imageMetaData)

    const contract = await getEthereumContract()

    const transactionParameters = {
      to: contractAddress,
      from: currentAccount,
      data: await contract.mint(currentAccount, `ipfs://${ipfsJsonHash}`),
    }

    try {
      await metamask.request({
        method: 'eth_sendTransaction',
        params: [transactionParameters],
      })

      setStatus('finished')
    } catch (error) {
      console.log(error)
      setStatus('finished')
    }
  }

  const renderLogic = (modalStatus = status) => {
    switch (modalStatus) {
      case 'initial':
        return (
          <InitialState
```



```
            profileImage={profileImage}
            setProfileImage={setProfileImage}
            name={name}
            setName={setName}
            description={description}
            setDescription={setDescription}
            mint={mint}
          />
        )

      case 'loading':
        return <LoadingState />

      case 'finished':
        return <FinishedState />

      default:
        router.push('/')
        setAppStatus('error')
        break
    }
  }

  return <>{renderLogic()}</>
}

export default ProfileImageMinter
```

Sanity.io Update: Upon successful minting, ProfileImageMinter.js updates the user's profile document within Sanity.io. It sets the profileImage field to the IPFS hash of the uploaded image, and the isProfileImageNft flag is set to true, indicating that the user now has an NFT associated with their profile picture.

User-Friendly NFT Minting Experience: To make minting an NFT profile image intuitive and accessible, I've designed a streamlined interface within my platform. This interface guides users through a simple process. They first select their desired profile image, then provide a unique title and description to personalize their NFT. This inputted forms the basis of the NFT's metadata, adding context and meaning to their on-chain profile picture.

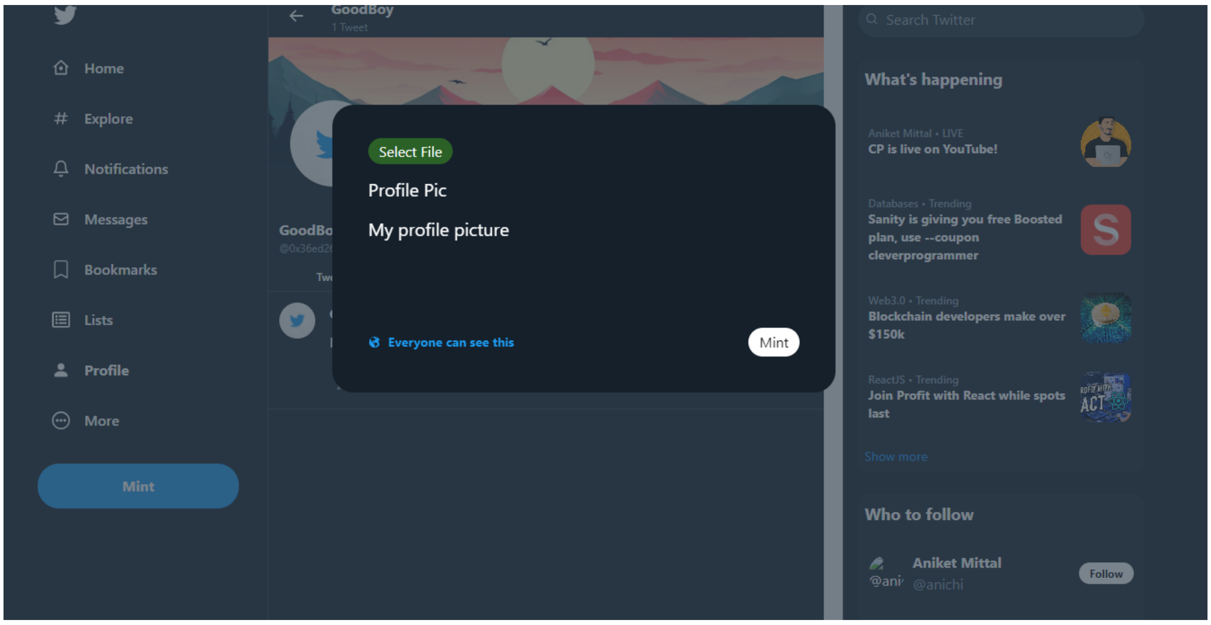


Figure 22. Minting NFT

Transaction Confirmation via MetaMask:

The minting process leverages MetaMask, a popular and widely trusted cryptocurrency wallet extension. This ensures a secure and user-friendly experience for users familiar with Web3 concepts. When the user initiates the NFT creation process within the platform's user interface (refer to previous screenshot), MetaMask seamlessly launches a confirmation window directly within the browser extension. This window serves a critical purpose – it clearly communicates the transaction details associated with minting the NFT profile picture. This includes:

Transaction Cost: The user can transparently view the exact amount of Sepolia test network ETH (currently displayed as "0.00021672 SepoliaETH") required to cover the gas fees for minting the NFT. Gas fees are a crucial aspect of blockchain transactions, and they compensate miners on the Ethereum network for their computational power used in processing the transaction. By displaying the gas fees upfront, the platform empowers users to make informed decisions about proceeding with the minting process.

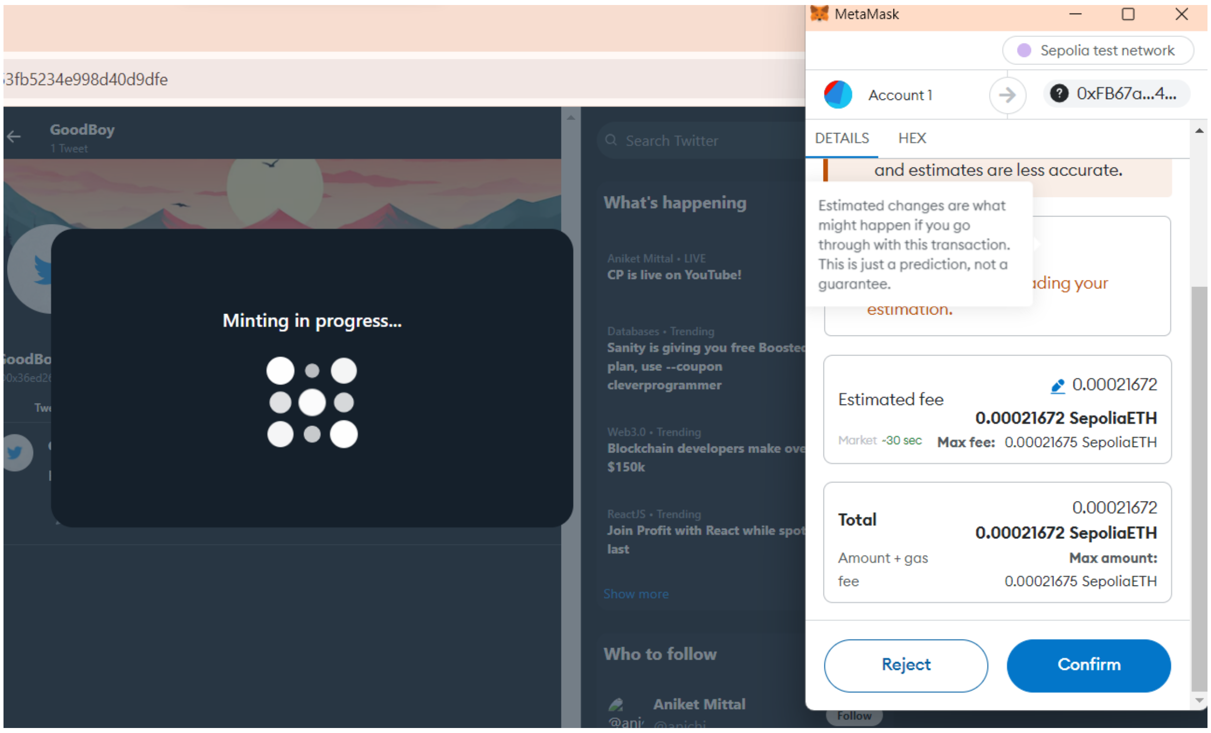


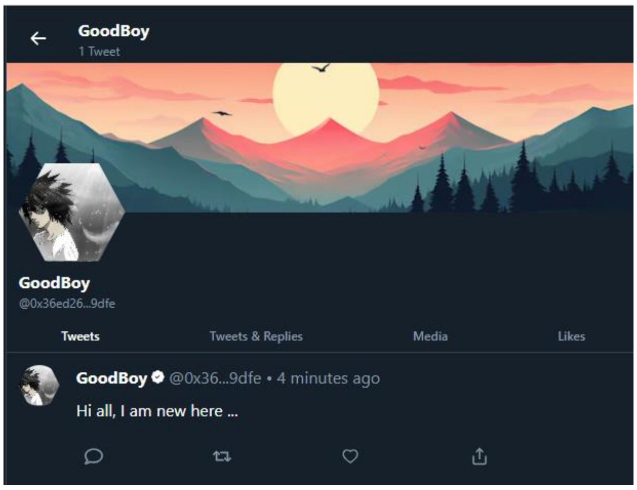
Figure 23. MetaMask Connected for Minting

Visualizing the Results: NFT Profile Image Integration

Following a successful minting transaction, a sense of accomplishment washes over the user as their newly minted NFT assumes its rightful place as their profile picture. This unique digital asset isn't merely displayed – it's prominently featured on their profile page, acting as a badge of honor and a symbol of their Web3 identity within the platform ([Insert profile page screenshot]). This visual representation serves as a constant reminder of their foray into blockchain technology and their ownership of a unique digital collectible.

But the ownership journey doesn't end there. By navigating to the "Collectibles" or "NFTs" section (terminology may vary depending on the user's specific MetaMask configuration) within their browser extension, users can delve deeper and confirm their ownership of the NFT on a blockchain level. MetaMask acts as a transparent window into the world of blockchain technology, allowing users to see their minted NFT readily listed among their other digital assets. This not only solidifies the connection between their social media presence and their blockchain wallet but empowers them to explore their growing collection of NFTs and interact with them within the broader Web3 ecosystem.

This seamless integration between the platform's user interface and MetaMask fosters a user-centric experience throughout the NFT creation process. Users gain a deeper understanding of blockchain technology by witnessing the minting process unfold and by having the ability to verify ownership directly within their trusted wallet. This empowers users to confidently navigate the exciting world of Web3 and leverage NFTs to express their individuality and actively participate in the decentralized social media landscape.



GoodBoy @0x36...9dfe • 4 minutes ago

Hi all, I am new here ...

Figure 24. NFT Successfully Displayed To Profile

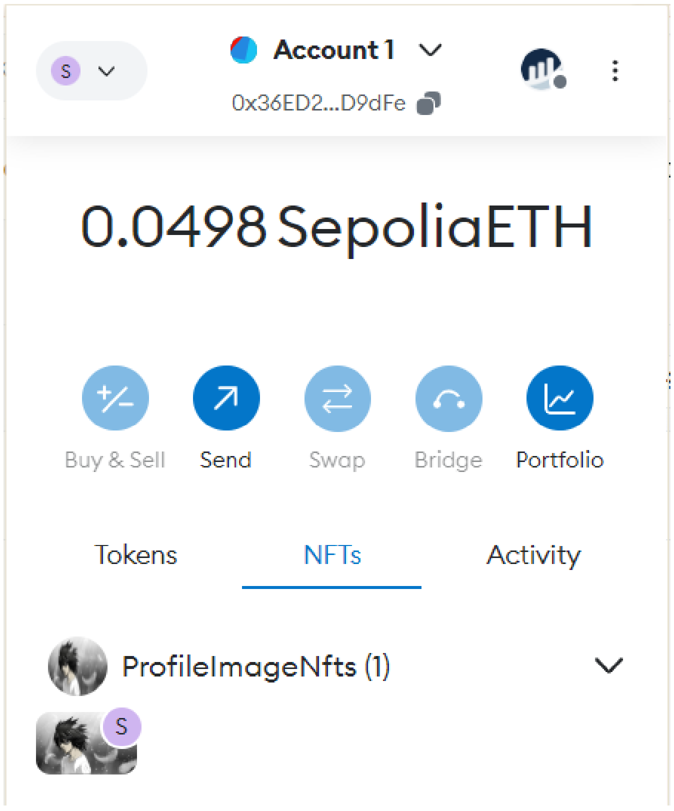


Figure 25. NFT Added to MetaMask

CONCLUSION AND FUTURE WORK

By building this decentralized social media platform, I have demonstrated the potential of integrating Web3 technologies into traditional social networking models. Key achievements of this project include:

Decentralized User Profiles: Moving away from centralized control of user data, this project utilizes Sanity.io as a headless CMS. This establishes a robust and structured data store, laying the foundation for a social media platform where users retain greater control over their profiles and information.

MetaMask Integration: Seamless authentication with MetaMask establishes a secure link between the user's Ethereum wallet and their social profile. This promotes self-sovereign identity within the platform, empowering users to manage their credentials and data without relying on a central authority

NFT Profile Images: The successful implementation of an NFT minting feature, while optional, unlocks a new paradigm for user expression and profile management. It allows users to showcase their Web3 identity, foster verifiable ownership of their profile pictures through on-chain NFTs, and potentially create value for their digital assets within the platform's social context.

Foundation for Growth: This project showcases a working prototype that demonstrates the viability of decentralized social media functionalities. It sets the stage for even further exploration of more advanced decentralized features, paving the way for a more user-centric and autonomous future for social networking.

Promoting Accessibility: While my project highlights the benefits of a Web3-powered social network, it's equally crucial to bridge the gap between traditional social media experiences and blockchain concepts. My design emphasizes ease of use to ensure that users less familiar with cryptocurrencies can still intuitively navigate the platform and benefit from the core features. User-friendly interfaces and clear onboarding guides can play a pivotal role in making decentralized social networks accessible to a broader audience.

Network Effects and Building Community: The success of any social platform hinges on its ability to attract a critical mass of engaged users. As this project evolves, the focus will shift towards developing strategies for seeding the network. Fostering communities of interest, targeted campaigns, and potentially token-based incentives for early adopters could accelerate the growth of the platform while maintaining its decentralized ethos.

Balancing Decentralization and User Experience: Decentralized architectures often present trade-offs in terms of speed, scalability, and user experience when compared to traditional centralized systems. My ongoing development will address these considerations by exploring layer 2 scaling solutions, efficient blockchain interaction patterns, and careful UI optimizations to ensure the platform remains responsive and user-friendly even if a large number of users join.

FUTURE WORK

This project opens the doors to several exciting avenues for further development:

Robust Reputation System: A token-based reputation system managed by smart contracts could significantly enhance content curation and governance within the platform. By rewarding valuable contributions and penalizing malicious behaviour, such a system would incentivize users to create high-quality content and foster a healthy online community. Furthermore, by enabling delegation of reputation tokens, users could signal trust in other users, further improving the discovery of valuable content.

On-Chain Content Storage: Transitioning towards storing social posts or other forms of content, either directly on the blockchain or on decentralized storage solutions like Arweave or Filecoin, would significantly enhance the platform's resistance to censorship. By placing content immutably on the blockchain, users would have greater confidence that their posts cannot be arbitrarily deleted or modified by the platform or any other entity. Additionally, decentralized storage solutions could offer greater scalability and cost-efficiency compared to traditional centralized storage options.

Decentralized Marketplaces: Implementing NFT marketplaces within the platform could create entirely new economic models for content creators. Users could directly buy, sell, and trade profile image NFTs or other unique digital assets associated with their social profiles. This could empower creators to monetize their work directly, fostering a vibrant creator economy within the social network. Additionally, by enabling fractional ownership of NFTs, marketplaces could allow users to participate in the ownership of high-value digital assets without needing to invest large sums of capital individually.

Governance DAOs: Establishing Decentralized Autonomous Organizations (DAOs) to govern platform upgrades, feature development, or community initiatives would truly shift the control of the social network away from any central entity and empower the users themselves. DAO token holders could vote on proposals to modify the platform's rules, prioritize feature development, and allocate resources from a shared treasury. This democratic approach to governance would ensure that the platform continues to evolve in accordance with the needs and desires of its user base.

FINAL THOUGHTS

Through this project, I've embarked on a fascinating exploration of the convergence between social media and blockchain technologies. Decentralized social media platforms have the potential to revolutionize how we connect and interact online, placing user autonomy, transparency, and ownership at the forefront. While there are significant challenges to overcome, such as scalability, security, and user experience, I firmly believe that blockchain-based solutions hold the key to empowering users, fostering a more democratic and equitable content landscape, and creating entirely new economic models within social networks. Ultimately, these advancements could pave the way for a more just and user-centric social web, where users are in control of their data, identity, and online experiences.

REFFERNCES

[1] Y. Chen, Y. Zhu, and G. Xu, "Social Media User Privacy Protection: A Survey," IEEE Transactions on Computational Social Systems, vol. 10, no. 1, pp. 77-89, Feb. 2023. doi: 10.1109/TCSS.2022.3229833

[2] S. Vaidhyanathan, Antisocial Media: How Facebook Disconnects Us and Undermines Democracy, New York: Oxford University Press, 2018.

[3] D. Y. Jin, "Transforming the Social: Technology, Content, and Power in the Creator Economy", International Journal of Cultural Studies, vol. 24 no. 5, pp. 787-804, Sept 2021. doi: 10.1177/1367877920982786

[4] Berners-Lee, T. (1989). Information Management: A Proposal. CERN. [Online]. Available: http://www.w3.org/History/1989/proposal.html

[5] Chen, Y., Zhu, Y., and Xu, G. (2023). "Social Media User Privacy Protection: A Survey," IEEE Transactions on Computational Social Systems, vol. 10, no. 1, pp. 77-89, Feb. 2023. doi: 10.1109/TCSS.2022.3229833

[6] Chohan, U. W. (2021). The Decentralized Web 3.0 Revolution: The Blockchain, Cryptocurrencies, NFTs, Metaverse, Web3 and how it will disrupt every industry from Finance to Gaming.

[7] De Filippi, P., & McMullen, G. (2018). Governance of Blockchain Systems: Communities, Institutions and Technology. MIT Press.

[8] Hendler, J. (2001). Agents and the Semantic Web. IEEE Intelligent Systems, vol. 16, no.2, pp. 30–37. doi: 10.1109/5254.920597

[9] Miller, H., Thebault-Spieker, J., Chang, S. et al. (2022). Explainer: Web3, a Vision for a Decentralized Internet. Stanford Graduate School of Business.

[10] Vaidhyanathan, S. (2018). Antisocial Media: How Facebook Disconnects Us and Undermines Democracy. Oxford University Press.

[11] Y. Chen, Y. Zhu, and G. Xu, "Social Media User Privacy Protection: A Survey," IEEE Transactions on Computational Social Systems, vol. 10, no. 1, pp. 77-89, Feb. 2023. doi: 10.1109/TCSS.2022.3229833

[12] S. Vaidhyanathan, Antisocial Media: How Facebook Disconnects Us and Undermines Democracy, New York: Oxford University Press, 2018.

[13] D. Y. Jin, "Transforming the Social: Technology, Content, and Power in the Creator Economy", International Journal of Cultural Studies, vol. 24 no. 5, pp. 787-804, Sept 2021. doi: 10.1177/1367877920982786

[14] P. De Filippi, & G. McMullen, Governance of Blockchain Systems: Communities, Institutions and Technology. MIT Press, 2018.

[15] U. W. Chohan, The Decentralized Web 3.0 Revolution: The Blockchain, Cryptocurrencies, NFTs, Metaverse, Web3 and how it will disrupt every industry from Finance to Gaming. [Online]. Available: https://www.amazon.com/Web3-Decentralized-Contracts-Decentralization-Blockchain/dp/B09JDYKND7

[16] H. Miller, J. Thebault-Spieker, S. Chang et al. "Explainer: Web3, a Vision for a Decentralized Internet." Stanford Graduate School of Business. [Online]. Available: https://hbr.org/2023/09/web3-could-change-the-business-model-of-creative-work

[17] Kopyto, M., & Bulut, E. (2021). SteemIt: Analysis of User Behavior and Evaluation of the Influence Mechanism. Computer and Information Sciences, 142, 161–176. doi.org/10.1007/978-3-030-89092-0_13

[18] Chohan, U. W. (2021). The Decentralized Web 3.0 Revolution: The Blockchain, Cryptocurrencies, NFTs, Metaverse, Web3 and how it will disrupt every industry from Finance to Gaming.

[19] De Filippi, P., & McMullen, G. (2018). Governance of Blockchain Systems: Communities, Institutions and Technology. MIT Press.

[20] Miller, H., Thebault-Spieker, J., Chang, S. et al. (2022). Explainer: Web3, a Vision for a Decentralized Internet. Stanford Graduate School of Business.

[21] Lo, S. K., Xu, X., Staples, M. L., Yao, L., & Nguyen, L. (2021). A blockchain-based layered security architecture for intelligent healthcare systems. Computers & Electrical Engineering, 94, 107337. doi.org/10.1016/j.compeleceng.2021.107337

[22] Park, J., Park, S., & Park, Y. (2023). Exploring the usability of blockchain-based cryptocurrency wallets. PLoS ONE, 18(2), e0281791. doi:10.1371/journal.pone.0281791

[23] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[24] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2014. [Online]. Available

[25] J. Benet and J. Greco, "Filecoin: A decentralized storage network," Protocol Labs, 2018. [Online]. Available: https://filecoin.io/filecoin.pdf

[26] P. Fraga, B. Rodrigues, and F. Brasileiro, "Decentralized Social Network as a Dynamic Heterogeneous System: A Case Study of Mastodon," in Proc. IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 2022 pp. 1-6.

[27] S. Iyer and B. Srinivasan, "Sociopedia: Design and Development of a Decentralized and Privacy-Preserving Social Network," in Proc. of 24th Int'l Conf. on Network Protocols (ICNP), Singapore, Singapore, 2016, pp. 1-2.

[28] D. Desantis, "Steemit: The First Decentralized and Incentivized Social Media Platform," Medium, Oct, 30 2017. [Online]. Available

[29] Hardhat documentation. [Online]. Available: https://hardhat.org/

[30] MetaMask documentation. [Online]. Available: https://docs.metamask.io/guide/

[31] OpenZeppelin Contracts Library. [Online]. Available: https://openzeppelin.com/contracts/