International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 04 | April - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Toward Transparent and Modular DApps: A Web3 SaaS Prototype for Token Lifecycle Management

Priyanshu P. Wadke
Department of Computer Engineering
Datta Meghe College of Engineering
Navi Mumbai, Mumbai, India
priyanshu.wadke.comp@gmail.com

Harshal H. Gangawane
Department of Computer Engineering
Datta Meghe College of Engineering
Navi Mumbai, Mumbai, India
harshal.gangawane.comp@gmail.com

Shantanu S. Bhise
Department of Computer Engineering
Datta Meghe College of Engineering
Navi Mumbai, Mumbai, India
shantanubhise2015@gmail.com

Dr. Chandrashekhar M. Raut
Department of Computer Engineering
Datta Meghe College of Engineering
Navi Mumbai, Mumbai India
chandrashekhar.raut@dmce.ac.in

***Abstract*- The evolution of blockchain and Web3 technologies has paved the way for decentralized application platforms that enable transparent, tamper-proof transactions without relying on centralized servers. However, existing solutions such as Gitcoin and Giveth are either too complex or unsuitable for modular deployment in educational and lightweight environments. This paper proposes *MetaSuite*, a role-based, blockchain-driven Software-as-a-Service (SaaS) platform that enables users to create, transfer, and donate tokens while allowing an administrator to securely withdraw funds. Built entirely on the Ethereum blockchain using Solidity smart contracts, *MetaSuite* operates without a backend and integrates wallet-based authentication via MetaMask and Ethers.js. The platform ensures transparent fund management through on-chain event logging and role-based access controls. Performance evaluations on the Ethereum HoleskyTestnet demonstrate the system's reliability, gas-efficiency, and real-time responsiveness. By eliminating backend dependencies and emphasizing traceability, *MetaSuite* serves as a minimalistic yet scalable Web3 solution suitable for academic, experimental, and small-scale real-world deployments.**

Keywords—Blockchain, Web3, Smart Contracts, Ethereum, MetaMask, Tokenization, SaaS, Ethers.js, Decentralized Applications.

I.Introduction

Decentralized applications (DApps) are redefining how software services are designed and delivered in the Web3 era. Unlike traditional SaaS platforms that rely on centralized servers and databases, DApps leverage blockchain networks to ensure transparency, immutability, and peer-to-peer interaction. Ethereum, with its robust support for smart contracts and developer tooling, has become the de facto platform for DApp development [6].

The advent of Web3 has shifted focus toward building systems that prioritize user autonomy, modularity, and role-based access. Projects such as Gitcoin and Giveth demonstrate the potential of token-based fundraising, but their complex architectures and reliance on backend APIs limit their adoption in academic and small-scale contexts [4], [5]. Furthermore, these platforms often lack native mechanisms for user role separation, such as administrator-only fund withdrawal or audit-level logging [8].

Recent literature emphasizes the benefits of modular and reusable smart contract design through paradigms like Smart Contract as a Service (SCaaS), advocating for scalable blockchain systems that are easy to maintain and extend [3]. Yussupov et al. further argue that a serverless DApp architecture one that operates without any centralized backend — is both viable and increasingly important in decentralized ecosystems [6].

This paper presents MetaSuite, a decentralized SaaS platform designed to manage the full token lifecycle — creation, transfer, donation, and administrative fund withdrawal — in a fully frontend-driven, backend-free Web3 environment. Built using Solidity, Next.js, and Ethers.js, and secured via MetaMask wallet authentication, MetaSuite demonstrates how a simple but effective smart contract system can support secure, role-based financial interactions without compromising on decentralization, traceability, or user accessibility.

The platform was deployed and tested on the Ethereum HoleskyTestnet, and its performance was evaluated based on gas usage, functional correctness, and role-specific validations. The results indicate that MetaSuite is a viable architecture for transparent, modular, and scalable DApps tailored for educational use, experimental prototyping, and lightweight real-world deployment.

A. Motivation

While blockchain technology has matured significantly, most existing Web3 platforms are built with complex infrastructures, making them difficult to use or adapt in educational or prototype environments. Platforms like Gitcoin or Giveth offer robust features for token-based donations, but they often rely on backend systems, lack modularity, and offer no role-specific access [4], [5]. For students, educators, and early-stage developers, there is a need for a platform that offers transparent, minimal, and hands-on interaction with smart contracts and token operations.

B. Objective

This project aims to develop a modular Web3 SaaS tool that:

- Enables token creation, transfer, and donation
- Allows admin-only access for fund withdrawals
- Uses MetaMask for wallet-based authentication [7]

- Operates without a backend, running entirely through a frontend + smart contract stack [6]
- Supports gas-efficient, auditable, and transparent operations on Ethereum [3], [6]

C. Applications

The system is suitable for:

- **Educational institutions** teaching blockchain development
- **Prototype funding platforms** for NGOs or research groups
- **Community reward systems** based on token economics
- **Transparent crowdfunding** DApps for academic or social causes [4], [10]

Its modular architecture allows easy extension into future domains such as NFTs, DAO governance, or multi-token integration.

II.Literature Survey

Blockchain platforms have gained widespread attention for their ability to support decentralized, secure, and auditable systems. In particular, smart contracts have enabled automation and trustless interactions in various domains, ranging from finance to identity management. Several studies have explored the use of smart contracts for delivering modular services, token economies, and Web3-native user experiences.

Sun et al. [3] introduced the concept of Smart Contract as a Service (SCaaS), proposing a framework where reusable and composable smart contracts can power next-generation Web3 platforms. Their emphasis on modularity directly informs the design goals of MetaSuite, which uses a clean separation of concerns between token management functions and user roles.

Yussupov et al. [6] presented a comprehensive discussion on the serverless nature of smart contracts, arguing that fully frontend-driven architectures — those without centralized backends or databases — are both feasible and desirable in DApp development. This perspective aligns with the core architecture of MetaSuite, which runs entirely in the browser via MetaMask and Ethers.js.

In the context of decentralized fundraising, platforms like Gitcoin and Giveth have popularized the model of token donations and open-source funding [4], [5]. However, these systems often require backend infrastructure, offer limited customizability, and lack fine-grained role-based access controls — a gap that MetaSuite seeks to fill.

Nguyen et al. [8] proposed PenChain, a blockchain-based service provisioning system that uses smart contracts to enforce penalties and ensure service-level agreement compliance. Their work demonstrates how smart contracts can go beyond fund transfer, serving as programmable tools for logic enforcement and access control — concepts adapted in MetaSuite through admin-only withdrawal permissions and event logging.

Other studies have addressed complementary domains: Almajed et al. [1] examined NFT-based pricing in Web3 SaaS systems; Bamakan and Far [2] built trustworthy digital twin platforms using blockchain automation; and Song [4] explored token economies in creator-centric Web3 environments. These studies highlight the expanding range of blockchain-enabled applications, but few of them present generalizable, educational platforms for token lifecycle experimentation.

In contrast, MetaSuite offers a lightweight, open-ended environment for token creation, transfer, donation, and secure fund management, aimed specifically at academic, experimental, and minimal-dependency contexts.

III.Proposed System

The proposed platform, MetaSuite, is a role-based, decentralized Web3 SaaS prototype that enables secure token lifecycle operations through Ethereum smart contracts. It is designed to be fully modular, backend-independent, and verifiable through on-chain execution and logging.

A. Architectural Component

The system architecture consists of four tightly integrated layers:

➢ *Frontend Interface (Next.js)*

This layer provides a responsive user interface that lets users interact with the smart contract. The frontend is built using *Next.js*, a React-based framework that supports client-side routing and dynamic content updates. All interactions, such as submitting token details or initiating transfers, are handled through Web3-enabled UI forms.

➢ *Wallet Integration (MetaMask)*

MetaMask serves as the user's blockchain identity provider. It allows users to sign transactions, authenticate with their Ethereum addresses, and interact securely with the DApp. The application dynamically detects the connected wallet address to identify the user role (admin or general user) and conditionally renders controls based on access rights [7].

➢ *Smart Contract Layer (Solidity)*

The core logic resides in Solidity smart contracts deployed to the Ethereum blockchain. Key functions include:

- createToken (): Deploys a new ERC-20 token with a custom name, symbol, and supply
- transferToken (): Sends tokens between wallet addresses
- donate (): Accepts token/ETH donations into the contract
- withdraw (): Allows only the admin to transfer ETH to their address

Access control is enforced using require statements, and every transaction emits an event such as TokenCreated, Transfer, DonationReceived, or FundsWithdrawn for full traceability [3], [8].

➢ *Blockchain Layer (Ethereum HoleskyTestnet)*

The HoleskyTestnet is used to simulate real blockchain deployment in a cost-effective and public setting. All transactions, contract states, and event logs are permanently stored and can be audited through explorers like Etherscan [1].

B. Logic & Execution Flow

Each function in the smart contract is designed to align with role-based access:

- Users can call functions like createToken, transferToken, and donate, all of which are permission less and wallet-bound.

- Admins are hardcoded into the contract at deployment. Only the admin wallet can call sensitive methods like withdraw ().
- If a non-admin attempts a restricted action, the contract throws an error and reverts the transaction.

The frontend dynamically reads on-chain state using ethers.Contract.read() functions and updates the UI accordingly. When a transaction is triggered, ethers.Contract.write() is used to broadcast the action, and MetaMask prompts the user to sign it.

All funds and tokens remain on-chain at all times, and contract balances are queried in real time using Ethereum JSON-RPC methods [6].

C. Transaction Flow Explanation

The platform follows a typical transaction flow model that ensures decentralization and role-based control:

- **Wallet Connection**
  - User opens the MetaSuite frontend and connects their wallet via MetaMask.
  - The connected address determines user role (admin or general user).
- **Token Creation**
  - User fills in token details (name, symbol, supply) and submits.
  - The contract deploys the token to the blockchain; the creator receives the full supply.
- **Token Transfer**
  - Using Ethers.js, a user can transfer tokens to another wallet.
  - Transaction is signed in MetaMask and broadcast to Ethereum.
- **Donations**
  - User donates tokens or ETH to the smart contract.
  - Donation is confirmed on-chain, and event logs are emitted.
- **Activity Monitoring**
  - All users can view logs, balances, and token stats in real-time.
  - UI updates reflect blockchain state, synced through Ethers.js calls

D. Technology Stack

Layer	Tool	Purpose
Smart Contract	Solidity	On-chain logic and role enforcement [6]
Web3 Interface	MetaMask	Wallet-based auth & transaction signing [7]
Frontend	Next.js	Responsive UI for user interaction
Ethereum Library	Ethers.js	Blockchain communication from frontend [6]
Network	HoleskyTestnet	Deployment & testing environment [1]

Table 1. Technology Stack

IV.Evaluation & Result

To validate the functionality and reliability of MetaSuite, the smart contracts were deployed on the Ethereum HoleskyTestnet. The system was tested across all key modules, including token creation, transfer, donation, and admin-only withdrawal.

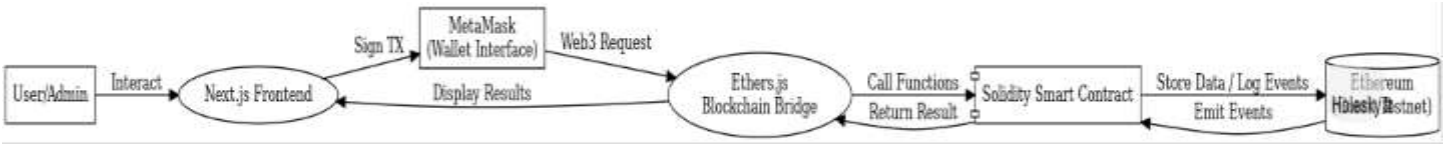

Fig. 1 System Architecture

A. Deployment and Functional Testing

The smart contract was deployed using **Remix IDE**, and all functionalities were triggered through a **Next.js frontend** connected via **MetaMask**. Each transaction was tracked through **Etherscan**, verifying:

- Successful contract execution
- Accurate gas consumption
- Proper wallet authentication
- Real-time UI state updates

Example: A token created with the symbol MTK was successfully minted and distributed to the creator's wallet. Transfers and donations executed correctly, and the admin was able to withdraw ETH.

B. Gas Efficiency

Measured average gas usage for key operations:

Operation	Avg. Gas Used
Create Token	130,000
Transfer Token	65,000
Donate Token	70,000

Table 2 Gas Efficiency

C. **Role Validation**

To ensure role-based access worked correctly, multiple tests were performed:

Action Tested	Role	Expected Outcome	Result
Create Token	User/ Admin	Token created	Success
Transfer Token	User	Transfer complete	Success
Donate Token	User	Contract balance	Success
Withdraw Funds	Admin	ETH transferred	Success
Withdraw (non-admin)	User	Access denied (revert)	Blocked

Table 3. Role Validation

D. On-Chain Logging and Transparency

Each smart contract action emitted a corresponding event (TokenCreated, Transfer, DonationReceived, FundsWithdrawn), ensuring all activity was:

- Logged publicly on **Etherscan**
- Visible in the **frontend dashboard**
- Available for third-party audits

E. Results

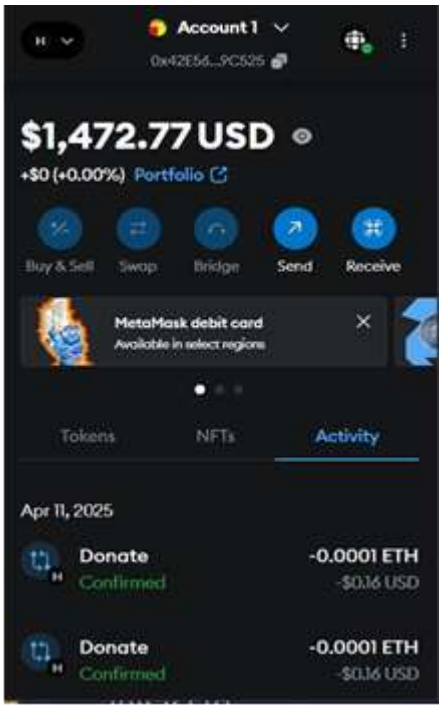

Fig. 2 MetaMask Account (Admin)

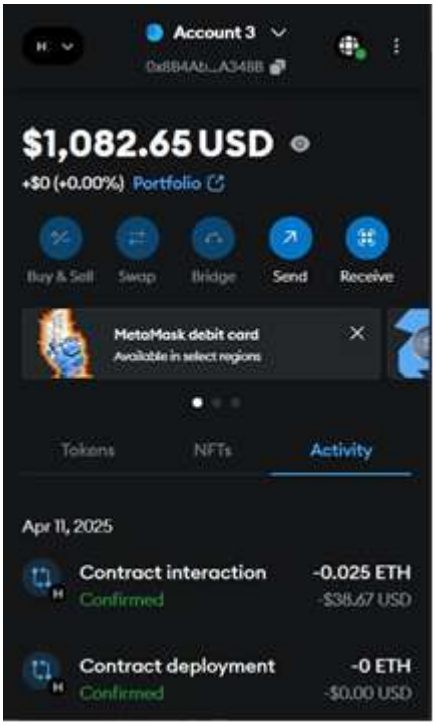

Fig. 3 MetaMask Account (User)

Fig. 4 Home Page Screen 1

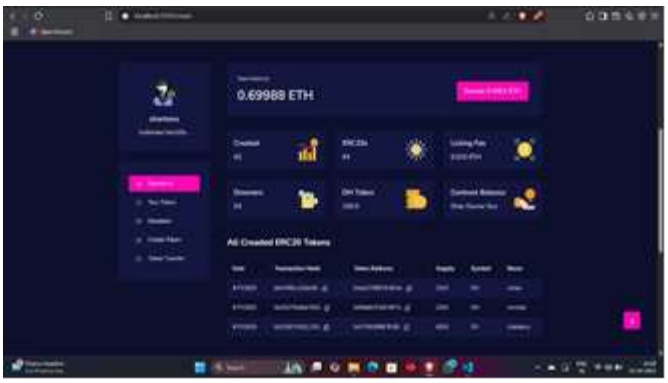

Fig. 5 Dash Board with Transaction Logs

F. Performance Summary

MetaSuite was found to be:

- Accurate in role validation
- Efficient in gas use
- Modular and easy to extend
- Transparent in logging and fund flow

The platform meets its core goal: providing a transparent, backend-free, token management system for educational and experimental use

V.Conclusion and Future Work

This paper proposed MetaSuite, a role-based, modular, and fully decentralized Web3 SaaS platform for managing token lifecycles on the Ethereum blockchain. Developed using Solidity, MetaMask, Ethers.js, and Next.js, the system eliminates backend dependencies while enabling secure and transparent token interactions through smart contracts. The platform specifically addresses challenges in existing tools such as Gitcoin and Giveth by offering lightweight, backend-free architecture and fine-grained role control [4], [5].

A. Summary of Findings

The platform was successfully deployed and validated on the Ethereum HoleskyTestnet, where it demonstrated:

- Accurate **role validation logic** through admin-only withdrawals
- Efficient execution of **token creation, transfer, and donation** operations
- Consistent **on-chain event logging** enabling auditability and traceability
- Minimal gas costs, confirming the system's suitability for low-resource contexts

The entire application stack — consisting of Solidity, MetaMask, Ethers.js, and Next.js — worked in sync to deliver a secure and scalable DApp that performs all core operations without relying on a centralized server.

The system also ensured that each transaction emitted corresponding events, which were auditable via Etherscan, supporting the platform's commitment to transparency. Compared to other platforms like Gitcoin, which rely on staking, governance tokens, oracles, and API layers [4], *MetaSuite* remains minimalistic, yet functionally rich.

B. Implications of the Study

From an academic and developer perspective, this study shows that:

- Full-stack decentralization is possible with a minimal architecture
- Smart contract-based role enforcement can replace traditional access control systems
- Educational and lightweight deployments can benefit from modular, frontend-driven blockchain apps
- Emerging developers can leverage this architecture as a teaching, prototyping, or demo platform

This work proves that backend-free, role-based DApps can be implemented with a limited toolchain and still achieve enterprise-grade transparency and role separation. By enforcing admin-only functions at the smart contract level and maintaining a fully public activity log, MetaSuite helps mitigate centralized vulnerabilities common in donation and tokenized platforms.

Beyond the technical contribution, this research has social implications — particularly in how on-chain governance and transparency can restore trust in fundraising, grant allocation, and peer-to-peer exchanges. Additionally, MetaSuite's simplicity and modularity offer strong pedagogical value, making it suitable for academic programs, hands-on workshops, and blockchain development curricula**.**

C. Future Research Directions

Several promising areas have emerged from this study that deserve further investigation:

- **Governance Models**: Integration of DAO structures with token-holder voting and multi-admin layers.
- **NFT Support**: Extension of smart contracts to support ERC-721/1155 tokens for NFT-based utility.
- **Cross-Chain Compatibility**: Deployment across multiple EVM-compatible chains to explore interoperability and gas optimization.
- **Scalability**: Migration to Layer-2 platforms (e.g., Arbitrum, Polygon) to enhance throughput and reduce costs.
- **Security Audits**: Applying formal verification tools to evaluate contract robustness under attack vectors.

D. Conclusion

This study contributes to the blockchain development landscape by demonstrating a transparent, accessible, and functionally complete Web3 SaaS platform designed with real-world use cases and academic learning in mind. It confirms that modular, frontend-driven DApps can meet core functional and security requirements without backend logic — thus advancing the goals of decentralization, accessibility, and scalability in blockchain research and practice.

REFERENCES

[1] R. Almajed, A. Abualkishik, and A. Ibrahim, "Forecasting NFT Prices on Web3 Blockchain Using Machine Learning to Provide SAAS NFT Collectors," Fusion: Practice and Applications, 2023. [Online].

→ Cited in Future Scope – NFT pricing & SaaS relevance — pp. 4–5

[2] M. Bamakan and S. B. Far, "Distributed and Trustworthy Digital Twin Platform Based on Blockchain and Web3 Technologies," Cyber Security and Applications, Elsevier, vol. 3, 2024.

→ Cited in architecture modeling and smart contract trust — pp. 3–5

[3] J. Sun, J. Yan, and K. Z. Zhang, "Smart Contract as a Service: Architecture, Applications, and Research Issues," IEEE Internet Computing, vol. 25, no. 2, pp. 66–73, Mar. 2021.

→ Cited in SCaaS model & modular design relevance — pp. 67–70

[4] Y. Song, "The European Creator Economy's Web3.0 Business Model," Decentralized Economy Journal, vol. 2, no. 1, pp. 23–35, 2022.

→ Cited for Gitcoin/Giveth-like platforms and token use — pp. 28–31

[5] S. Sarkar, "Centralized Intermediation in a Decentralized Economy: Risks and Opportunities," arXiv preprint arXiv:2302.09561, 2023.

→ Cited for limitations of existing Web3 ecosystems — pp. 2–4

[6] D. Yussupov et al., "On the Serverless Nature of Blockchains and the Tiny Data Challenge in Smart Contracts," arXiv preprint arXiv:2007.16029, 2020.

→ Cited in architecture design & backend-free implementation — pp. 3–6

[7] M. Ahmad, "Integration of IoT Devices via DApps Using Solidity and MetaMask," Academia.edu, 2017.

→ Cited for wallet-based smart contract interaction — pp. 5–7

[8] N. Nguyen, H. Ngo, and M. Vuong, "PenChain: Penalty-Aware Smart Contract Service Provisioning," IEEE Access, vol. 11, pp. 54677–54690, 2023.

→ Cited in access control and admin-only logic — pp. 54682–54685

[9] Y. Cheng, "Web 3.0: Concept, Content and Context," in Digital Transformation with Blockchain Technology, Springer, 2024, pp. 115–134.

→ Cited in Web3 architecture and evolution — pp. 120–124

[10] J. Kölbel, "Next Generation Business Ecosystems Using Decentralized Tokens and Smart Contracts," CORE: Blockchain Innovation Series, vol. 3, no. 1, 2023. → Cited in modularity, token design, and DAO outlook — pp. 18–21