GitBounty – A Decentralized Web3 Bounty Platform

Jothsana Waikar[1], Saloni Pawar[2], Satyajit Maske[3], Shambhuraje Patil[4] Mr. Yogesh Patil[5]

[1]*Jothsana Waikar Computer Science and Engineering & Nanasaheb Mahadik College of Engineering, Sangli, India*

[2]*Saloni Pawar Computer Science and Engineering & Nanasaheb Mahadik College of Engineering, Sangli, India*

[3]*Shambhuraje Patil Computer Science and Engineering & Nanasaheb Mahadik College of Engineering, Sangli, India*

[4]*Satyajit Maske Computer Science and Engineering & Nanasaheb Mahadik College of Engineering, Sangli, India*

[5]*Mr. Yogesh Patil Computer Science and Engineering & Nanasaheb Mahadik College of Engineering, Sangli, India*

---***---

Abstract:

GitBounty is a decentralized Web3 bounty platform that connects project owners and open-source contributors by turning GitHub issues into on-chain bounties. Project owners create bounties by locking ETH in a smart contract and attaching a GitHub issue link and NFT-badge metadata. Developers submit proof (PR link). After verification, the creator approves a winner; the smart contract automatically transfers ETH and mints a unique NFT badge as verifiable recognition. The system uses Solidity smart contracts, React frontend, Node.js + Express backend, and MongoDB for off- chain metadata and indexing. Transactions run with standard gas fees (no gasless). GitBounty ensures transparent, tamper- proof reward distribution and permanent recognition of contributions.

Keywords: Bounty Marketplace, Blockchain, Smart Contracts, NFT Badge, Ethereum, MERN, GitHub Integration, Decentralized Rewards.

1. INTRODUCTION

Open-source software has become the backbone of modern technology, powering critical infrastructure, developer tools, and enterprise systems across the globe. Despite its widespread adoption and impact, the sustainability of open- source development remains a significant challenge. Most open-source projects depend heavily on voluntary contributions, where developers invest time and expertise without guaranteed compensation or formal recognition. This model often leads to delayed issue resolution, contributor burnout, and uneven participation, especially for complex or time-consuming tasks.

Bounty-based incentive models have emerged as an effective solution to encourage developer participation by attaching monetary rewards to specific issues, feature requests, or bug fixes. However, traditional bounty systems are largely manual and off-chain, relying on centralized platforms, informal agreements, or direct payments after work completion. Such approaches introduce several limitations, including lack of transparency, delayed or failed payments, disputes over no permanent record of contributor achievements.

To address these shortcomings, **GitBounty** is designed as a **decentralized, blockchain-powere d bounty distribution platform** that automates the entire lifecycle of open-source bounties using smart contracts. By leveraging Ethereum smart contracts, GitBounty ensures that bounty rewards are securely locked in an on-chain escrow at the time of bounty creation, eliminating the risk of non-payment. Once a developer

submits a valid contribution—such as a GitHub pull request or proof link—the project owner can verify the submission directly through the platform and approve or reject the claim
in a transparent manner.

Upon approval, the systemautomatically transfers the ETH reward to the developer's wallet using standard blockchain transactions, with gas fees paid by the interacting users or project owners. In addition to monetary compensation, GitBounty mints a unique **NFT badge** for the winning contributor, providing immutable, verifiable recognition of their work. These badges serve as a decentralized reputation mechanism, allowing developers to build a public, on-chain portfolio of contributions that can be showcased across platforms.

The platform combines **on-chain trust and automation** with **off-chain usability and metadata indexing**. While all critical actions—such as escrow locking, reward transfers, and badge minting—are handled by smart contracts, off-chain components manage user profiles, bounty discovery, claim tracking, and analytics. This hybrid architecture ensures scalability, usability, and efficient data retrieval without compromising the security guarantees of blockchain technology.

By integrating Ethereum smart contracts, NFT-based recognition, and a developer-friendly web interface, GitBounty bridges the gap between decentralized finance principles and real-world open-source collaboration. The platform transforms open-source contribution from an informal and trust-based process into a **structured,**
transparent, and verifiable incentive ecosystem, fostering fair compensation, accountability, and long-term sustainability in open-source development.

2. LITERATURE REVIEW

The development of the GitBounty platform is informed by a broad review of existing research and practical implementations in the domains of blockchain technology, open-source software incentives, decentralized finance (DeFi), and digital credentialing. This literature survey examines prior work that influenced the architectural and design decisions of GitBounty, with a particular focus on transparency, trust, automation, and contributor recognition in open-source
ecosystems.

1. Blockchain-Based Bounty Platforms

“Gitcoin: Incentivizing Open Source Developme nt Using Blockchain” – K. Owocki et al. (2018)
This work introduces Gitcoin as one of the earliest platforms to apply blockchain-based incentives to open-source contributions. Gitcoin demonstrates how tokenized rewards and on-chain payments can increase participation and accountability in collaborative development. However, Gitcoin relies primarily on token-based incentives and does not provide immutable contributor recognition in the form of NFTs. GitBounty builds upon this concept by implementing ETH-based on-chain escrow combined with NFT badge
issuance, ensuring both financial compensation and permanent contribution records.

“Decentralized Bounty Systems for Software Development” – A. Peterson and M. Clark (2019)
This study evaluates decentralized bounty mechanisms that eliminate intermediaries in software reward distribution. The authors highlight the importance of escrow-based smart contracts to reduce disputes and ensure payment guarantees.
These findings directly inform GitBounty’s escrow-first design, where ETH rewards are locked in smart contracts at bounty creation time.

2. NFT-Based Digital Credentials and Contribution Badges

“POAP: Proof of Attendance Protocol” – D. Gerard (2020) POAP introduces the concept of non-fungible tokens as digital badges that represent participation or achievement. While POAP focuses primarily on event attendance, it demonstrates the social and professional value of on-chain credentials.
GitBounty extends this idea by minting NFTs that represent verified open-source contributions, transforming badges into proof-of-work artifacts rather than mere participation tokens.

“GitPOAP: Verifiable Developer Contributions Using NFTs” – GitPOAP Team (2022)
GitPOAP explores the use of NFTs to acknowledge contributions to GitHub repositories. The platform highlights how NFTs can serve as a decentralized developer résumé.
GitBounty adopts a similar philosophy but integrates NFT
minting directly into the bounty approval workflow, ensuring badges are only issued after successful task completion and reward distribution

3. Smart Contract Security and Design Patterns

“Security Analysis of Smart Contracts” – L. Atzei, M. Bartoletti, and T. Cimoli (2017)
This foundational research analyzes common vulnerabilities in Ethereum smart contracts, such as reentrancy, improper access control, and unsafe fund transfers. The study emphasizes the need for standardized security patterns.
GitBounty incorporates these lessons by using well-established design principles, including escrow isolation, role- based access, and secure ETH transfer mechanisms.

OpenZeppelin Contract Libraries (2021–2024) OpenZeppelin provides audited implementations of widely used smart contract components such as ReentrancyGuard, Ownable, and ERC standards.
GitBounty's smart contracts are inspired by these best practices to reduce attack surfaces, improve reliability, and maintain consistency with industry standards.

4. Off-Chain Indexing and User Experience in Blockchain Applications

"Designing User-Friendly Decentralized Applications" – S. Nakamoto and E. Li (2020)
This research highlights the importance of separating critical on-chain logic from non-critical off-chain data storage. It
suggests storing financial transactions on-chain while using off-chain databases for metadata, search, and analytics.
GitBounty follows this hybrid architecture by maintaining
escrow, payments, and badge minting on-chain, while indexing claims, repositories, and user profiles off-chain for efficient browsing.

"Event Indexing for Ethereum-B ased Applications" – J. Lin and P. Zhao (2021)
This study discusses the use of blockchain event logs

3.1 Conceptual Framework
combined with off-chain databases to enable real-time updates and responsive user interfaces. GitBounty leverages similar principles by indexing smart contract events to display bounty status, claim history, and reward distribution in the frontend dashboard.

5. Gas Models and Transaction Cost Considerations

"Meta-Transactions and Gas Abstraction on Ethereum" –
I. Grigg and A. Miller (2020)
This paper explores gasless transaction models and relayer- based architectures that abstract gas fees from end users. While these approaches improve usability, the study also highlights increased system complexity, reliance on third- party infrastructure, and potential centralization risks. After evaluating these trade-offs, GitBounty intentionally adopts a **standard gas payment model**, ensuring simplicity, decentralization, and compatibility with Ethereum's native transaction flow.

The literature reviewed demonstrates that blockchain-based incentives, NFT-based recognition, and secure smart contract design significantly enhance trust and transparency in open- source collaboration. Existing platforms validate the effectiveness of bounties and digital credentials but reveal gaps in automation, escrow security, and unified workflows. GitBounty addresses these gaps by combining secure on-chain escrow, automated ETH payouts, NFT badge issuance, and off-chain indexing within a single, cohesive platform. This literature survey provides a strong theoretical and practical foundation for GitBounty's architecture and validates its relevance as a modern solution for decentralized open-source incentivization.

3. METHODOLOGY

The GitBounty platform is developed using a structured, phased methodology grounded in decentralized systemdesign principles to ensure security, transparency, scalability, and reliability. The methodology focuses on automating the complete bounty lifecycle—from creation and escrow locking to contribution verification, reward distribution, and NFT- based recognition—using blockchain technology and modern web frameworks.

The conceptual framework of the GitBounty systemis composed of **six tightly integrated components**, each playing a critical role in ensuring trustless and automated bounty management.

Bounty Creation and Escrow Setup

Project maintainers or repository owners initiate bounties by defining task descriptions, reward amounts (in ETH),
submission deadlines, and acceptance criteria. At the time of bounty creation, the specified ETH reward is locked into an on-chain escrow smart contract. This ensures fund availability and eliminates the risk of non-payment, establishing trust between contributors and project owners.

Verification and Approval Logic

Once a claim is submitted, the repository owner or authorized reviewer evaluates the contribution. Using predefined approval rules, the owner can either approve or reject the claim. Approved claims trigger automated workflows within the smart contract, while rejected claims are recorded
transparently to ensure accountability.

Automated Reward Distribution

Upon approval of a valid claim, the smart contract automatically releases the escrowed ETH reward to the contributor's wallet. This eliminates manual payment processes, reduces delays, and ensures atomic execution — meaning approval and payment occur as a single trustless operation.

NFT Badge Minting and Recognition

Simultaneously with reward distribution, the systemmints a unique NFT badge representing the successful completion of the bounty. Each NFT contains verifiable metadata, including repository name, bounty ID, contribution type, and timestamp. These NFTs act as permanent, tamper-proof proof of contribution that can be showcased in developer portfolios.

Off-Chain Indexing and User Interface

To enhance usability, non-critical data such as bounty descriptions, user profiles, and search indexes are stored off- chain using databases. Blockchain events emitted by smart contracts are indexed and reflected in real time within the user interface, providing a seamless experience without
compromising decentralization.

This integrated framework combines **on-chain trust guarantees** with **off-chain efficiency**, enabling GitBounty to deliver a secure, transparent, and developer-friendly platform.

3.2 Methodology Workflow

The development and deployment of GitBounty follow a step- by-step workflow to ensure systematic implementation and reliable integration of all systemcomponents.

1. Requirement Analysis (Weeks 1–2)

- Identify key stakeholders, including open-source maintainers, developers, and platform administrators.
- Define functional requirements such as bounty creation, escrow locking, claim submission, approval workflows, payment automation, and NFT issuance.
- Specify non-functional requirements including security, decentralization, transaction reliability, and gas efficiency.
- Evaluate feasibility concerning Ethereum gas costs, smart contract limitations, and user adoption considerations.

2. System Design (Weeks 3–4)

- Design the high-level system architecture integrating smart contracts, frontend interfaces, and off-chain indexing services.
- Define smart contract interfaces for bounty management, escrow handling, and NFT minting.
- Create UML diagrams such as use case diagrams, sequence diagrams, and component diagrams to model interactions between users, contracts, and services.
- Design UI wireframes for dashboards, bounty listings, claim tracking, and user profiles.

3. Smart Contract Development (Weeks 5–7)

- Implement Ethereum smart contracts using Solidity to manage bounty lifecycle operations.
- Integrate secure escrow logic for ETH deposits and withdrawals.
- Apply standardized security patterns such as access control, reentrancy protection, and safe ETH transfers.
- Implement ERC-721 compliant NFT contracts for minting contributor badges.
- Perform local testing using blockchain

development tools to validate contract logic and edge cases.

4. Backend and Off-Chain Services (Weeks 8–9)

- Develop backend services to index blockchain events and manage off-chain metadata.
- Store non-sensitive data such as bounty descriptions, repository links, and claim references in a database.
- Ensure synchronization between on-chain events and off-chain data representations.
- Implement administrative utilities for monitoring platform activity.

5. Frontend Development (Weeks 10–11)

- Build a responsive web interface using modern frontend frameworks.
- Integrate wallet connectivity for user authentication and transaction signing.
- Display real-time bounty status, claim submissions, approvals, and payment confirmations.
- Provide dashboards for maintainers and contributors to track activity and history.

6. Testing and Quality Assurance (Weeks 12–13)

- Conduct unit testing for smart contracts to ensure correct execution of escrow, approval, and payment logic.
- Perform integration testing between smart contracts, backend services, and frontend components.
- Test edge cases such as rejected claims, expired bounties, and insufficient gas scenarios.
- Validate NFT minting accuracy and metadata integrity.

7. Deployment and Documentation (Weeks 14–15)

- Deploy smart contracts to a public Ethereum-compatible network.
- Host frontend and backend services on a reliable cloud platform.
- Prepare detailed technical documentation covering smart contract APIs, systemarchitecture, and deployment steps.
- Create user guides for maintainers and contributors.

8. Maintenance and Enhancements (Ongoing)

- Monitor contract interactions and platform usage for performance and security issues.
- Apply bug fixes and UI improvements based on user feedback.
- Plan future enhancements such as multi-chain support, reputation scoring, and advanced analytics.
- Ensure compatibility with evolving Ethereum standards and tooling.

This methodology ensures that GitBounty is developed through a **secure, transparent, and scalable process** , combining blockchain automation with practical usability. By following a phased and structured approach, the platform achieves trustless bounty management, automated reward
distribution, and verifiable contributor recognition—making GitBounty a robust and future-ready solution for decentralized open-source incentivization.

4. TECHNOLOGY USED

• **Blockchain Layer**

o **Ethereum Blockchain:**
Ethereum serves as the foundational decentralized network for GitBounty. It enables trustless execution of smart contracts, ensuring that bounty funds are securely escrowed and automatically released upon successful contribution verification. Ethereum's immutability, transparency, and global accessibility make it ideal for handling financial incentives and verifiable contribution records in open-source ecosystems.

• **Smart Contract Development**

o **Solidity:**
Solidity is used to develop smart contracts that manage the complete bounty lifecycle, including bounty creation, escrow locking, claim submission, approval logic, reward
distribution, and NFT badge minting. Its compatibility with the Ethereum Virtual Machine (EVM) allows deterministic execution of contract logic, ensuring consistent and secure operations across the network.

o **OpenZeppelin Contracts:**
OpenZeppelin's audited smart contract libraries are used to implement standard security patterns and token

standards. Components such as access control mechanisms, secure ETH transfer utilities, and ERC-721 NFT implementations help minimize vulnerabilities and enforce best practices in smart contract development.

• Backend Services

o **Node.js:**
Node.js is utilized for building backend services that handle off-chain operations such as indexing blockchain events, managing metadata, and coordinating communication between the frontend and blockchain layer. Its non-blocking, event-driven architecture supports efficient handling of real- time blockchain event streams and API requests.

• Frontend Technologies

o **ReactJS:**
ReactJS is used to develop a responsive and interactive user interface for GitBounty. Its component-based architecture allows modular development of features such as bounty
listings, claim submissions, wallet interactions, and contributor dashboards. The virtual DOM ensures smooth performance even when reflecting real-time blockchain state changes.

• Wallet and Blockchain Interaction

o **Ethers.js:**
Ethers.js is employed to facilitate communication between the frontend and Ethereum smart contracts. It enables wallet connectivity, transaction signing, smart contract interaction, and event listening in a secure and lightweight manner,
ensuring reliable integration between users and the blockchain.

• Database and Off-Chain Storage

o **MongoDB:**
MongoDB is used to store off-chain data such as bounty descriptions, GitHub repository details, claim metadata, and
user profiles. Its flexible document-oriented structure supports rapid querying, indexing, and scalability while keeping non- critical data off-chain for improved performance and usability.

• NFT and Metadata Management

o **ERC-721 Standard:**
NFT badges are implemented using the ERC-721 token standard to ensure uniqueness, interoperability, and
marketplace compatibility. Each badge represents a verified
contribution and includes immutable metadata such as bounty ID, repository reference, and completion timestamp.

o **IPFS (InterPlanetary File System):**
IPFS is used to store NFT metadata and badge-related assets in a decentralized manner. This ensures long-term availability, tamper resistance, and independence from centralized servers.

• Development and Testing Tools

o **Hardhat:**
Hardhat is used as the primary blockchain development environment for compiling, testing, and deploying smart contracts. It provides advanced debugging tools, local blockchain simulation, and automated testing capabilities to ensure contract reliability.

o **Git and GitHub:**
GitHub serves as the version control and collaboration platform for managing source code, tracking issues, and linking pull requests directly to bounty claims within GitBounty.

5. APPLICATIONS

The GitBounty platform enables multiple real-world applications within open-source software development and collaborative programming ecosystems. By combining blockchain-based escrow, smart contract automation, and NFT-based recognition, GitBounty addresses long-standing challenges related to trust, incentive management, and contributor visibility.

• Decentralized Bounty Management for Open-Source Projects:
GitBounty allows repository owners to create ETH-

based bounties for specific GitHub issues or feature requests. Once a bounty is created, the reward amount is securely locked in an on-chain escrow smart contract, ensuring that funds cannot be withdrawn arbitrarily. This guarantees transparency and builds trust between project maintainers and contributors.

• **Secure and Trustless Reward Distribution:**
Through smart contract automation, GitBounty eliminates the need for manual or off-chain payments. When a project owner approves a developer's contribution, the platform automatically transfers the locked ETH reward to the winning contributor's wallet. This prevents disputes related to delayed or non-payment and ensures fair compensation based on predefined contract logic.

• **Verifiable Contributor Recognition (NFT Badges):**
Each successful bounty completion results in the minting of a unique ERC-721 NFT badge for the developer. These badges serve as immutable proof of contribution, containing verifiable metadata such as bounty ID, repository reference, and completion timestamp. Developers can display these NFTs as part of their professional portfolio, increasing
visibility and credibility within the developer community.

• **Transparent Claim Submission and Review Workflow:** Developers can submit claims by linking completed GitHub pull requests to active bounties. Repository owners can review submitted proofs on-chain and either approve or reject claims based on predefined criteria. This structured workflow ensures accountability and reduces ambiguity in contribution validation.

• **Enhanced Trust in Open-Source Collaboration:**
By storing critical operations—bounty creation, escrow locking, approval, payment, and badge minting—on the blockchain, GitBounty creates a transparent and tamper-proof system. All transactions and decisions are publicly verifiable, significantly reducing trust issues commonly associated with informal open-source incentive models.

• **Off-Chain Metadata Indexing and Searchability:**
While financial operations remain on-chain, GitBounty stores descriptive data such as issue details, repository information, and claim metadata off-chain using a database. This hybrid approach improves performance and enables advanced search, filtering, and browsing of bounties without compromising blockchain security.

• **Standard Gas-Based Economic Model:**
GitBounty operates using the standard Ethereum gas fee model, where users and repository owners pay transaction fees for contract interactions. This approach simplifies
deployment, avoids the operational complexity of meta- transactions, and ensures predictable execution aligned with Ethereum network standards.

• **Developer Portfolio and Career Advance ment:**
NFT badges earned through GitBounty act as permanent, verifiable records of real-world open-source contributions.

These badges can be showcased on resumes, personal websites, and blockchain explorers, helping developers demonstrate technical expertise and active participation in meaningful projects.

• **Scalable Incentive Mechanism for Organizations:**
Organizations and startups can use GitBounty to outsource development tasks, bug fixes, or feature implementations in a secure and measurable manner. The platform scales efficiently to support multiple concurrent bounties across repositories, teams, and communities.

• **Reduced Administrative Overhead:**
By automating bounty handling through smart contracts, GitBounty minimizes manual intervention, dispute resolution, and administrative coordination. Project maintainers can focus more on development and code quality rather than payment logistics and contributor management.

.6. RESULTS AND DISCUSSION

The GitBounty project successfully demonstrates a secure and transparent systemfor managing open-source bounties using blockchain technology. By implementing smart contracts with on-chain escrow, the platform ensures that rewards are locked at the time of bounty creation and automatically transferred to contributors upon successful claim approval, eliminating trust issues and payment disputes.

The use of gas-based transactions provides reliable execution and compatibility with standard Ethereum practices, avoiding the complexity of gasless models. Additionally, NFT-based badges offer permanent and verifiable recognition of contributors' work, adding long-term professional value beyond monetary rewards.

Overall, GitBounty improves trust, efficiency, and motivation within open-source ecosystems by combining automated payments, transparent verification, and digital recognition.

7. CONCLUSION

GitBounty represents a significant advancement in the way open-source contributions are incentivized and managed by introducing a fully blockchain-based bounty system. The platform addresses long-standing challenges in traditional bounty handling, such as lack of trust, delayed or missed payments, and absence of verifiable contributor recognition.
By leveraging smart contracts, GitBounty ensures transparency, fairness, and automation throughout the entire bounty lifecycle.

Through on-chain escrow and standard gas-based Ethereum transactions, GitBounty guarantees that funds are securely locked at bounty creation and released only after successful verification of contributions. This removes reliance on manual coordination and trust between project owners and contributors. Additionally, the minting of NFT badges provides permanent, tamper-proof recognition of contributors' work, enhancing professional visibility and credibility within the developer community.

The integration of a user-friendly web interface with off-chain metadata indexing enables easy discovery, management, and tracking of bounties and contributions. This combination of blockchain reliability and traditional web usability makes GitBounty both technically robust and practically accessible. The platform promotes accountability, motivates quality contributions, and strengthens the overall open-source
ecosystem.

Potential future enhancements for GitBounty include:

• **Multi-chain Support:** Expanding beyond Ethereum to support other blockchain networks for reduced costs and wider adoption.
• **Advance d Reputation System:** Introducing contributor reputation scores based on completed bounties, NFT history, and community feedback.
• **DAO-Based Governance:** Enabling community-driven decision-making for bounty verification and dispute
resolution.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to Prof. P.R. Bhosale for her invaluable guidance and support throughout the development of the Student Marks Evaluation Monitoring System. Her expertise and encouragement were instrumental in the successful completion of this project.

REFERENCES

1. **Buterin, V.**: *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. Ethereum Whitepaper, 2014.
2. **Wood, G.**: *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Ethereum Yellow Paper, 2015.
3. **OpenZeppelin Team**: *OpenZeppelin Contracts: Secure Smart Contract Development Library*. OpenZeppelin Documentation, 2022.
4. **Gitcoin**: *Gitcoin: Funding Open Source with Blockchain Incentives*. Gitcoin Platform Whitepaper, 2019.
5. **BountySource**: *Crowdsourced Software Development through Bounty Programs*. BountySource Technical Overview, 2018.
6. **POAP Team**: *Proof of Attendance Protocol (POAP): NFT-Based Digital Badges*. POAP Documentation, 2021.
7. **GitPOAP Contributors** : *GitPOAP: On-Chain Recognition for Open-Source Contributions*. GitPOAP Whitepaper, 2022.
8. **Szabo, N.**: *Smart Contracts: Building Blocks for Digital Markets*. Extropy Journal, 1997.
9. **Antonopoulos, A. M., & Wood, G.**: *Mastering Ethereum: Building Smart Contracts and DApps*. O'Reilly Media, 2018.
10. **Al-Bassam, M.**: *Blockchain Applications in Open Source Ecosystems*. IEEE Blockchain Conference Proceedings, 2020.

11. **Morabito, V.**: *Blockchain Technology and Its Applications in Incentive Systems*. Springer International Publishing, 2017.
12. **The Graph Foundation**: *Indexing Blockchain Data for Decentralized Applications*. The Graph Documentation, 2021.

Online Resources:

• **Ethereum Documentation**
https://ethereum.org/en/developers/docs/

• **Solidity Documentation (Language Reference)**
https://docs.soliditylang.org/

• **OpenZeppelin Smart Contract Library**
https://docs.openzeppelin.com/contracts/

• **React Documentation**
https://react.dev/

• **Ethers.js Documentation**
https://docs.ethers.org/

• **Hardhat Documentation**
https://hardhat.org/getting-started/

Tools and Platforms:

• GitHub : https://github.com

• Hardhat : https://hardhat.org

• MetaMask : https://metamask.io