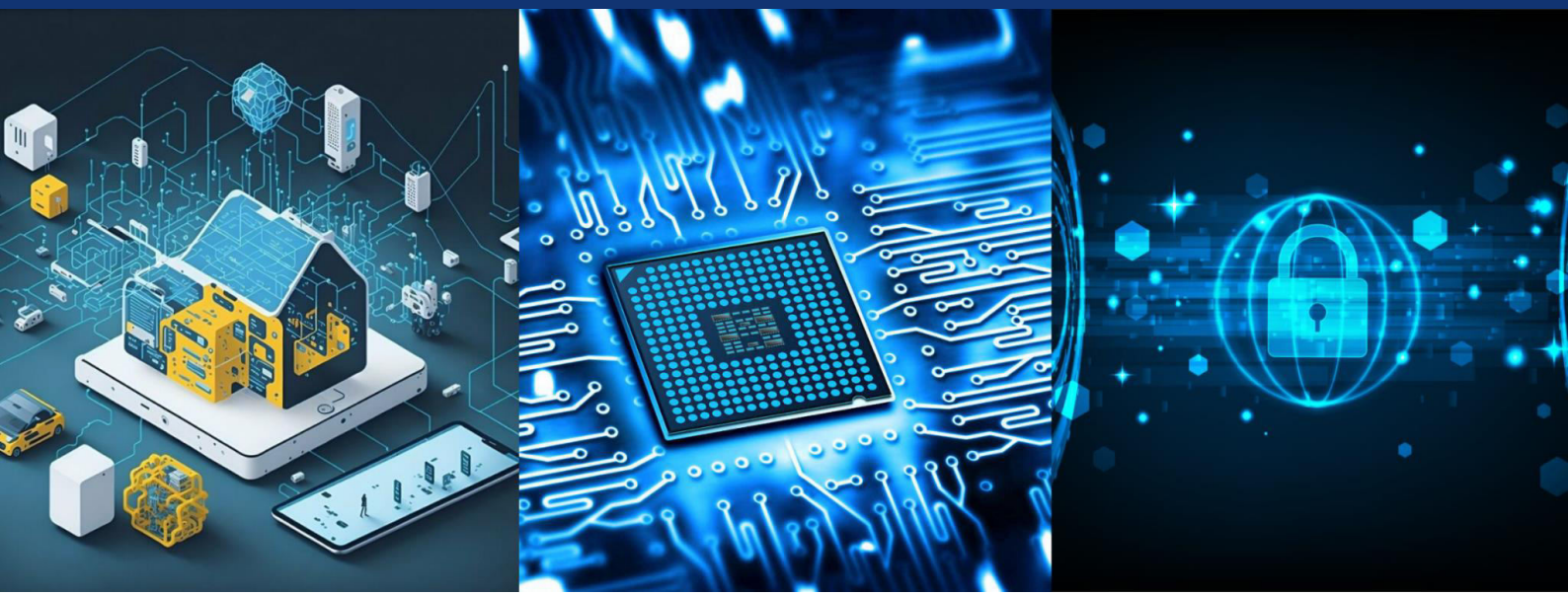
IJIRCCE

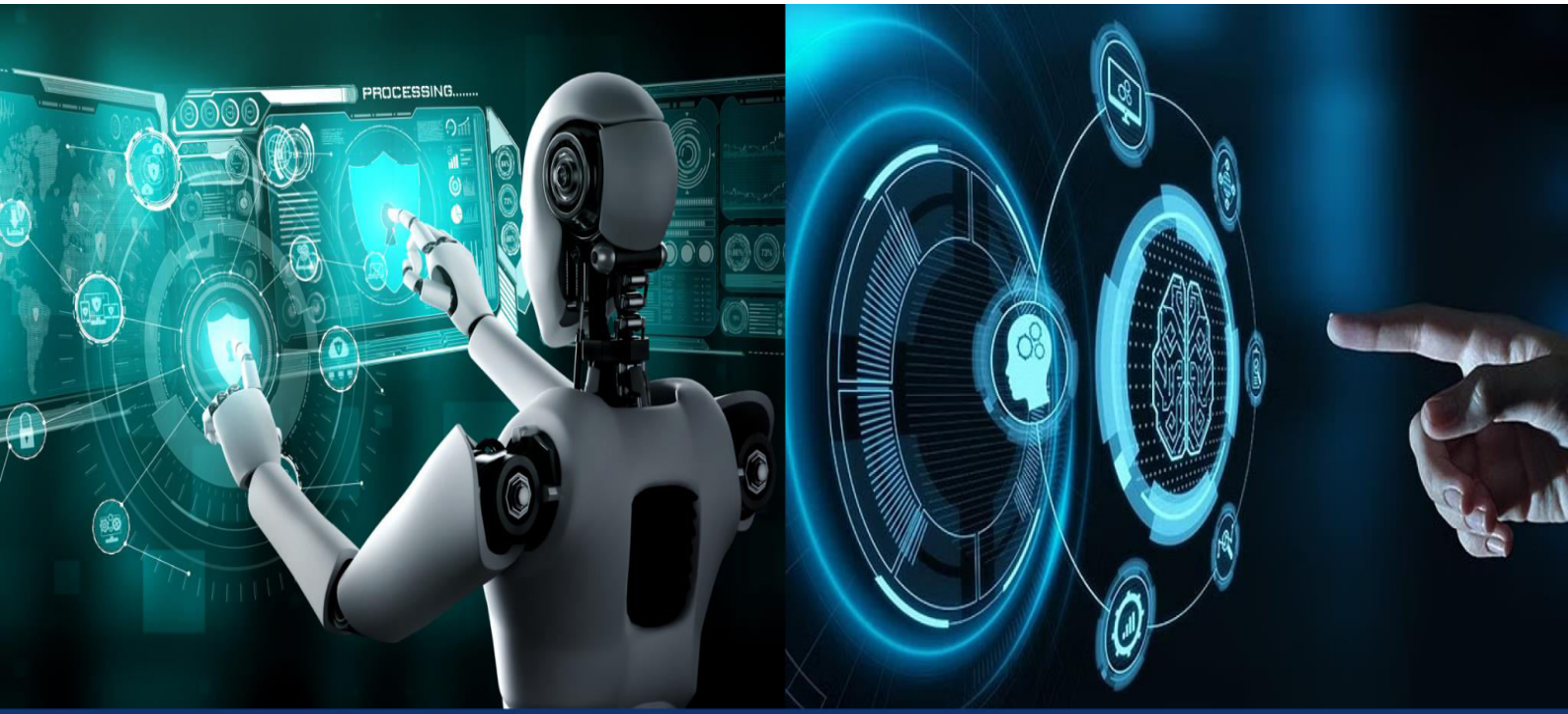
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.771

Volume 13, Issue 3, March 2025

© 2025 IJIRCCE | Volume 13, Issue 3, March 2025| DOI: 10.15680/IJIRCCE.2025.1303134

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

GitCode

Kaminee Jitendra Pachlasiya, Ajitesh Ghosh, Meet Ashish Parekh, Akash Rangbahadur Patel

Department of Computer Science and Engineering, Parul University, Vadodara, Gujarat, India

ABSTRACT: This report outlines my internship experience at Permissionless Information Technology in Bangalore, focusing on full- stack development for scalable web applications. I contributed to two major projects: GitCode, a decentralized platform enhancing collaborative coding in Web3, where I worked on frontend interfaces, API integration, and backend optimization; and Automa8x, a web-based automation solution, where I developed interactive UIs, backend logic, and seamless frontend- backend communication. Through these projects, I gained hands-on experience in full-stack development, API design, and building scalable Web2 and Web3 applications, aligning with the company's mission of using technology for innovative problem-solving

I. COMPANY OVERVIEW

Permissionless Information Technology, founded by Roshan Vaddessary in 2021 in Bangalore, focuses on building decentralized public infrastructure to address global challenges. Rejecting the profit-centric model, the company promotes inclusivity, transparency, and collective progress through blockchain-based solutions.

Products and Scope:

- GitCode: A decentralized platform supporting open-source sustainability with automated funding, transparent contribution rewards, and a bounty-based incentive system using blockchain and Chainlink.
- Developer Empowerment: Tools that facilitate collaboration and ensure fair compensation without reliance on traditional funding models.

Workforce:

Permissionless operates with a team of 35-50 employees, including co-founders, management, technical staff, and developers. As a tech-driven company, all operations are conducted digitally without physical plant infrastructure.

Overview of Departments and Processes

Company Structure and Departments: Permissionless Information Technology has a streamlined organizational hierarchy, including:

- Founding Members: Provide strategic direction for Web2 and Web3 innovations.
- Senior Management: Oversee operations and decision-making
- Team Leaders: Manage development teams in Web and Blockchain departments.
- Project Experts: Offer technical guidance
- Generalist Resources (Web3): Specialize in blockchain technologies, smart contract development, and decentralized infrastructure.
- Technical Resources: Ensure robust IT infrastructure, network management, and cybersecurity.

Technical Tools and Infrastructure:

- Software Tools: Trello, Jira, Google Workspace, Microsoft Teams, and Chainlink for blockchain applications.
- Hardware: High-performance laptops with Apple M3 chips, Dell PowerEdge servers, and NVIDIA GPUs for AI tasks.
- Cloud and Security: AWS, Filecoin for storage, and Cisco for network management.

Process Workflow for GitCode Platform:

1. Planning and Research: Define objectives, features, and blockchain technologies for decentralized funding and governance.
2. Design and Architecture: Develop front-end and back-end architectures for blockchain integration, ensuring

© 2025 IJIRCCE | Volume 13, Issue 3, March 2025| DOI: 10.15680/IJIRCCE.2025.1303134

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

seamless user experiences.

3. Development and Integration: Implement smart contracts, integrate crypto wallets, and establish Web2-Web3 communication.
4. Testing and Optimization: Conduct security audits, performance testing, and user experience enhancements.
5. Deployment and Maintenance: Deploy the platform on decentralized hosting and monitor for continuous updates and user feedback.

This comprehensive structure and workflow enable Permissionless to develop innovative, scalable Web2 and Web3 solutions.

II. INTERNSHIP SUMMARY AND MANAGEMENT

Internship Summary During my internship at GitCode, I contributed to developing a decentralized funding platform for open-source projects, focusing on Web3 infrastructure and full-stack web development. My responsibilities included smart contract development for automated funding distribution, blockchain- based authentication, and contributor tracking. Additionally, I built user-friendly dashboards, integrated blockchain APIs, and ensured seamless interaction between Web2 and Web3 components.

2.1 Purpose The purpose was to gain hands-on experience in blockchain development, focusing on smart contract automation, decentralized governance, and secure Web3 integrations. I also aimed to enhance my full-stack development skills to create an intuitive and scalable user experience.

2.2 Objectives

- Develop smart contracts for automated funding distribution.
- Build and optimize GitCode's web platform with blockchain integration.
- Implement secure blockchain authentication.
- Create APIs and UI components for tracking contributions and automated payments.
- Ensure performance scalability and security.

2.3 Scope Capabilities:

- Automated fund distribution
- Smart contract-based payouts
- Blockchain-powered contributor tracking
- Decentralized authentication
- User-friendly web platform

Limitations:

- Dependency on external funding sources
- Transaction costs from gas fees
- Need for human oversight in governance decisions

2.4 Technologies and Literature Review

Technologies:

- Blockchain: Ethereum, Polygon
- Smart Contracts: Solidity
- Web3 Libraries: Web3.js, Ethers.js
- Decentralized Storage: IPFS
- Full-Stack Development: React.js, Next.js, Tailwind CSS, Node.js, Express.js
- Databases: PostgreSQL, Firebase
- Security: OpenZeppelin, MythX
- Hosting: AWS, Vercel

© 2025 IJIRCCE | Volume 13, Issue 3, March 2025| DOI: 10.15680/IJIRCCE.2025.1303134

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

III. LITERATURE REVIEW

- Blockchain Basics by Daniel Drescher for blockchain fundamentals.
- Ethereum Whitepaper by Vitalik Buterin for smart contract development.
- Building Scalable Web Apps with Next.js by Vercel Developers for front-end optimization.
- The Road to Web3 by Antonio Garcia for Web3 application integration.

3.1 Development Approach and Planning Adopting an Agile methodology allowed flexibility for iterative development and continuous testing.

The project was divided into two primary phases over six months:

Phase 1: Smart Contract Development and Web3 Integration (3 Months)

- Developed and deployed smart contracts.
- Conducted security audits and optimized gas fees.
- Integrated blockchain APIs and wallet connections.

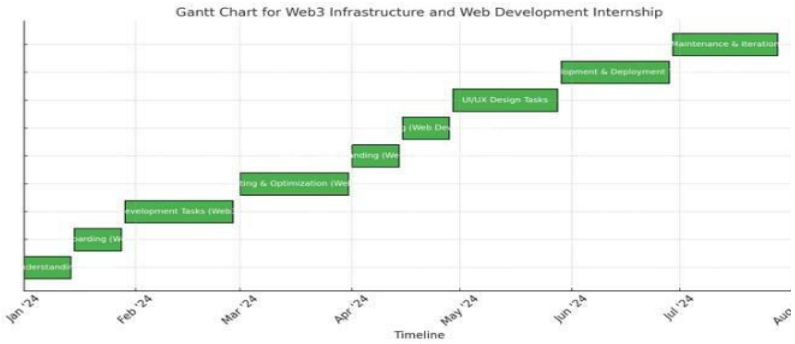
Phase 2: Full-Stack Web Development (3 Months)

- Designed an interactive dashboard with React.js.
- Developed API services using Node.js and Express.js.
- Deployed on Vercel and AWS for scalability.

3.2 Roles and Responsibilities

- Developed Solidity smart contracts with security measures.
- Integrated Web3.js/Ethers.js for blockchain interactions.
- Built responsive dashboards using React.js and Tailwind CSS.
- Developed and managed APIs with Node.js.
- Conducted smart contract audits and ensured security.

3.3 Gantt Chart A Gantt chart was used to visualize the timeline, ensuring effective tracking of each development phase, testing, and deployment.



IV. ANALYSIS

4.1 Existing Web3 Infrastructure and Development Challenges

Decentralized funding platforms like GitCode face challenges in Web3 integration, scalability, security, and user experience.

1. Complex Integration

- Smart contract-based funding requires accurate implementation for transparency.
- Cross-chain interoperability across Ethereum, Polygon, and Solana remains difficult.
- Secure decentralized identity (DID) and wallet authentication are necessary.

2. Scalability & Performance

Ethereum's high gas fees and slow transaction speeds hinder real-time payouts.

- Layer 2 scaling solutions like Optimism, Arbitrum, or zk-rollups are needed.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- Efficient backend management for tracking funding pools is crucial.

3. Security Vulnerabilities
- Smart contracts require audits to prevent exploits (e.g., reentrancy attacks).
- Wallet integration risks include phishing and key mismanagement.

4. User Experience
- Non-technical users struggle with Web3 wallets and transactions.
- Improved onboarding and intuitive interfaces are necessary.
- Real-time feedback on transaction status builds user trust.

4.2 Problems and Weaknesses in Web3 Infrastructure
- Integration Complexity: Handling gas fees, cross- chain fund transfers, and interoperability.
- Scalability Issues: Layer 2 solutions are essential to mitigate Ethereum's limitations.
- Security Risks: Potential vulnerabilities like front- running and DDoS attacks.
- User Experience Challenges: Simplified wallet management and user-friendly interfaces are necessary.
- Cost Constraints: High gas fees and node maintenance.
- Lack of Standardization: Inconsistent APIs and wallet implementations.

4.3 Requirements for an Improved Web3 Infrastructure
- Smart Contracts: Secure, gas-optimized, and interoperable across chains.
- Wallet Integration: Easy wallet connection using MetaMask or WalletConnect.
- Scalability: Implement Layer 2 solutions for faster transactions.
- Security: Automated audits using tools like MythX and Slither.
- Testing: Use Test-Driven Development (TDD) for reliable deployment.

4.4 System Feasibility
- Technology: Solidity, Hardhat, Ethereum, Polygon, and Solana.
- Front-End: React.js and Next.js.
- Back-End: Node.js and Express.js.
- Storage: Decentralized storage using IPFS or Arweave.
- Integration: APIs for compatibility with GitHub and GitLab.

4.5 Activity/Process in the New System
- Develop and audit smart contracts.
- Deploy on Ethereum and Layer 2 networks.
- Enable wallet authentication and signing.
- Design intuitive UI/UX for contributors.
- Conduct automated contract testing.

4.6 Features of the New System
- Decentralized Funding: Transparent fund allocation via smart contracts.
- Scalable Architecture: Layer 2 solutions for low- cost payments.
- Enhanced UX: Seamless wallet integration and transaction tracking.

4.7 Main Modules
- Smart Contract Development: Secure funding logic.
- UI/UX: Responsive design.
- Blockchain Integration: Cross-chain support.
- Security & Testing: Automated audits and penetration testing.

4.8 Selection of Hardware/Software/Algorithms/Methodology
- Hardware: High-performance GPUs and cloud- based infrastructure.
- Software: Solidity, Hardhat, React.js, Node.js.

© 2025 IJIRCCE | Volume 13, Issue 3, March 2025| DOI: 10.15680/IJIRCCE.2025.1303134

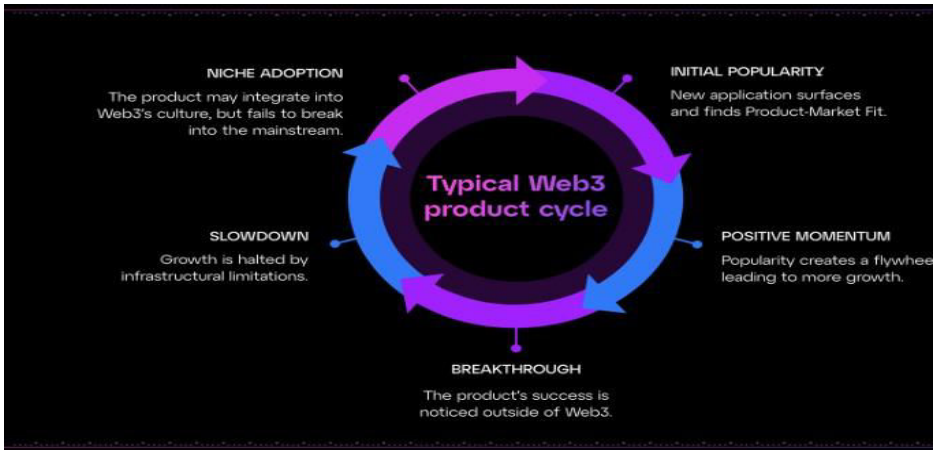
www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

- Algorithms: SHA-256, Layer 2 Rollups.

Methodology: Agile, Test-Driven Development



V. SYSTEM DESIGN

5.1 System Design & Methodology

Our Web3-based full-stack application follows a modular and scalable architecture, integrating decentralized storage, blockchain, and smart contracts.

System Architecture:

- Modular Design: Independent components for blockchain integration, decentralized storage (e.g., IPFS), and web interfaces.
- Client-Server Model: Front-end communicates with the blockchain and back-end for smart contract execution and data retrieval.
- Cloud and Decentralized Infrastructure: Combines traditional cloud services and blockchain networks like Ethereum or Polygon.
- Scalability: Supports high transaction volumes using Layer 2 solutions.

Methodology:

- Agile Development: Iterative improvements with continuous feedback.
- CI/CD Pipelines: Automated testing and deployment.

Data-Driven Development: Efficient data management between UI, APIs, and blockchain.

Training & Evaluation:

- Smart contract testing for performance, security, and gas fee optimization.
- Regular security audits and penetration testing.
- User feedback analysis for continuous improvement.

5.2 Database & Process Design

Database Design: A hybrid architecture using relational databases for traditional data and blockchain for decentralized logging.

- Users Table: Stores profiles, authentication data, and wallet addresses.
- Transactions Table: Logs blockchain interactions and token transfers.
- Smart Contracts Table: Tracks contract deployment and status.
- Feedback Table: Captures user experience data.

Data Structure:

- JSON-based data for blockchain communication.
- Indexed databases for transaction retrieval.
- Off-chain encrypted storage for sensitive data.

© 2025 IJIRCCE | Volume 13, Issue 3, March 2025| DOI: 10.15680/IJIRCCE.2025.1303134

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

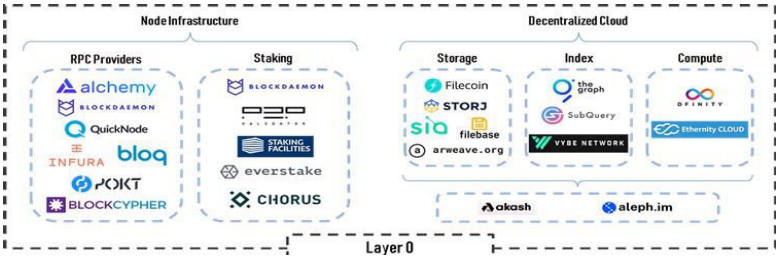
(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Process Design:

1. User Interaction: Users initiate transactions through the UI.
2. Blockchain Transaction: Transactions are validated and recorded.
3. Smart Contract Execution: Contracts execute autonomously.
4. Output & Feedback: Results displayed on the UI, with user feedback collected.
5. Monitoring & Optimization: Performance data informs updates.

Structure Design:

- Blockchain Integration: Manages blockchain interactions.
- Web Application: Handles UI and API interactions.
- Security & Compliance: Provides encryption and audits.
- Feedback & Optimization: Monitors user input for improvements.



5.3 Input / Output and Interface Design Input:

- User-submitted requests for transactions or API interactions.
- Guided templates for token transfers and staking.

Output:

- Transaction status, smart contract details, and funding updates.
- Results presented in a transparent, user-friendly format

Interface:

- Web dashboard for interacting with Git contributions, funding, and governance.
- Real-time feedback for transparent contribution evaluation.

5.4 State Transition Diagram System states include:

- Idle: Waiting for user input.
- Processing: Executing blockchain tasks.
- Generating Output: Displaying results.
- Completed: Process finished.

State	Action / Transition	Next State	Description
Idle	System awaits user input.	Processing	The Web3 system is idle, waiting for a user request (e.g., transaction, data query).
Processing	User submits a request.	Generating Output	The system processes the request, interacting with the blockchain or decentralized network.
Generating Output	Network processes and validates the request.	Completed	The system generates the required output (e.g., transaction confirmation, data retrieval).
Completed	Output is ready for review or feedback.	Feedback Received	The user views the result and can optionally provide feedback.
Feedback Received	User provides feedback on the output quality.	Idle	The system may use feedback to improve future iterations of the Web3 application.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

REFERENCES

1. Ethereum Documentation: https://ethereum.org/en/developers/doc s
2. Polygon Documentation: https://docs.polygon.technology
3. IPFS Documentation: https://docs.ipfs.io
4. Hardhat Documentation: https://hardhat.org/docs
5. Solidity Documentation: https://docs.soliditylang.org
6. GitHub or GitLab Repositories (for any referenced code)
7. Audit Tools like MythX or Slither
8. CI/CD Tools (e.g., GitHub Actions, Jenkins)
9. User Feedback & Reports:
10. User surveys or feedback data sources if applicable

.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details