IJARSCT

International Journal of Advanced Research in Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

ISSN: 2581-9429

Volume 5, Issue 6, April 2025

Impact Factor: 7.67

# Ethereum Transaction Web3 Decentralized App

Asst. Prof. Seema Bhuravane[1], Dr. Preeti Gupta[2], Bishal Pal[3], Rutika Mithare[4]

Assistant Professor, Department of Information Technology [1]

Associate Professor, Department of Information Technology[2]

Bachelor of Engineering in Information Technology[34]

K. C. College of Engineering and Management Studies and Research, Thane, Maharashtra, INDIA

**Abstract:** *The Ethereum Transaction Web3 Decentralized App project aims to create a decentralized application (Dapp) that leverages the Ethereum blockchain to facilitate secure and transparent transactions. By utilizing the power of Web3 technology, the Dapp will enable users to interact directly with the Ethereum network, bypassing the need for intermediaries. This decentralized approach ensures greater control over transactions, eliminates the risk of censorship, and enhances privacy. The Dapp will incorporate key features such as user authentication, transaction history tracking, and real-time updates. Users will be able to create accounts, send and receive Ether, and explore the Ethereum ecosystem.*

**Keywords:** Ethereum, Blockchain, Decentralized Application, React.js, MetaMask

## I. INTRODUCTION

This Blockchain is a technology that increases trust, transparency, and traceability of data exchanged over business networks and makes updating, hacking, or defrauding the system difficult or even impossible. Our project seeks to create a web application that integrates React JS with the blockchain and combines it with the Ethereum wallet via MetaMask for offering a secure and effective means of communication on the blockchain platform to the user. Transactions can be sent by users using the blockchain and receive UI changes as Gifs and memesbetween anonymous and identified individuals easy at times, even without an intermediary. This project aimed at integrating a React JS app into the blockchain and connecting it to an Ethereum wallet via MetaMask.

Ethereum transactions form the very existence of Web3 decentralized apps as the representation of the interaction by the user with the state of the blockchain. They are not just moves of digital cash; they are instruction signature which trigger operations on the Ethereum network. Every transaction involves vital information, such as the address of the sender and receiver, the amount of Ether being transferred, and above all, the "data" field, which bears the smart contract function calls and parameters in encoded form to be executed by the smart contract. For network security and avoiding misuse, a "gas" system is utilized, wherein people pay for the computational power needed to execute their transactions. Web3 libraries are developer-essential frameworks, streamlining the building and managing of transactions in dApps through such responsibilities as generating signatures and wallet handling. The journey of an Ethereum transaction entails its broadcast on the network, having it checked for correctness and made a part of a block through miners or validators, and its gathering of confirmations to etch it more securely onto the blockchain.

Thus, a thorough grasp of Ethereum transactions is indispensable for crafting and engaging with the decentralized applications that define the Web3 landscape. The goal was also to create a full Web3.0 application that allows users to send transactions across the blockchain. The aim of this web application is to allow users to communicate through blockchain to facilitate business and trade between unidentifiable and identifiable parties, at times without an intermediary

## II. LITERATURE SURVEY

### An Overview of Web3 technology: Infrastructure, applications, and popularity, 2024

Web3 is an internet model that is decentralized and blockchain-based, giving more power to users over the data. This study surveyed 200 Web3 projects supported by top venture capital investors and categorized them into infrastructure and applications. The study collected data from GitHub and blockchain browsers to determine their popularity. Web3 offers decentralization, security, trustless transactions, and new business models but is faced with the issues of scalability, interoperability, privacy attacks, code security, and usability. Despite these challenges, Web3 has potential innovation in finance, governance, and data ownership, ensuring transparency and empowerment of users.[1]

**The Future of Ethereum: A Technical Outlook, 2023**

This paper offers a forward-looking perspective on Ethereum's potential developments, discussing scalability solutions and the role of dApps in its future. By providing a technical outlook, the authors aim to stimulate discussion and provide a vision for the platform's trajectory. This paper's advantage is its ability to engage with the future direction of Ethereum, presenting a coherent narrative of potential advancements. However, it's important to recognize that the paper is based on speculation, and actual developments may diverge from the presented outlook.[2]

**A Comparative Study of Ethereum Transaction Fees: A Blockchain Analysis, 2022**

This study analyzes transaction fees across different Ethereum networks and time periods, utilizing blockchain analysis techniques. By examining fee fluctuations, the authors provide insights into the factors influencing transaction costs and their impact on dApp users. This research's strength lies in its ability to shed light on the economic aspects of Ethereum transactions. However, the complexity of fee determination means that the study may not capture the full range of factors that contribute to fee variations.[3]

**Security Analysis of Ethereum Smart Contracts: A Survey, 2021**

This paper delves into the security aspects of Ethereum smart contracts, conducting an in-depth analysis of common vulnerabilities and proposing mitigation techniques. By identifying potential security risks, the authors provide essential guidance for developers seeking to build secure smart contracts. This survey's advantage is its focus on a critical aspect of Ethereum development, offering practical insights into preventing exploits. However, like any survey in a rapidly evolving field, it may not cover all emerging vulnerabilities or the latest security best practices.[4]

**Building Scalable Ethereum Dapps: A Survey, 2020**

This study presents a systematic review of existing scalability solutions for Ethereum, including layer-2 protocols and sharding. By surveying the landscape of available options, the authors provided a comprehensive overview of strategies to enhance Ethereum's scalability. This survey's strength is its ability to synthesize a wide range of information, offering a valuable resource for developers and researchers. However, due to the rapid pace of innovation in the blockchain space, the survey may not encompass the most recent advancements in scalability techniques.[5]

**User Experience Design for Ethereum Dapps: A Case Study, 2019**

Focusing on the user experience (UX) of Ethereum decentralized applications (DApps), this work employed a case study methodology. By evaluating a specific dApp, the researchers identified areas for improvement and derived user-centric design principles. The advantage of this approach is its ability to provide detailed, context-specific insights into UX challenges within dApps. However, the limitation is that the findings are confined to the single dApp studied, making it difficult to generalize the conclusions to the broader dApp ecosystem.[6]

**Ethereum Transaction Processing: A Performance Evaluation, 2018**

This research paper conducted an empirical analysis to assess the performance of Ethereum transaction processing. By meticulously measuring transaction throughput, latency, and gas consumption, the authors provided a comprehensive evaluation of Ethereum's operational characteristics at that time. This study's strength lies in its practical, data-driven approach, offering valuable insights into the network's capabilities. However, its primary limitation is its temporal specificity; the findings, while accurate for 2018, may not accurately reflect the current state of Ethereum due to subsequent network upgrades and changes.[7]

## III. PROPOSED METHODOLOGY

The methodology used to develop this, includes a formalized process emphasizing usability and security. It begins with the analysis of requirements, and then the design of a scalable architecture using React.js and ethers.js. Solidity is used to write smart contracts, which are tested and deployed on a test-network. The front end gives a usable interface along with the functions required to communicate with the blockchain. A thorough test ensures stability before release, and continued monitoring and maintenance improve functionality and security after deployment.

**1. Requirement Analysis**:
The journey starts with finding out and knowing user needs, expectations, and issues. An in-depth analysis assists in specifying the functional requirements, including Ethereum transaction features, wallet support, and transaction security, as well as non-functional factors like scalability and usability.

**2. Architecture Design**:
A scalable and efficient system architecture is created based on the specifications. Technologies such as React.js are utilized for front-end development to design a responsive and user-friendly interface, while ethers.js is utilized for interacting with the Ethereum blockchain.

**3. Smart Contract Development**:
Smart contracts, coded in Solidity, form the backbone of the Decentralized Application. The contracts define the transaction logic, which is executed securely and transparently on the blockchain. Rigorous unit testing is performed to ensure the correctness and reliability of the smart contracts prior to deployment.

**4. Front-End Development**:
The front-end is developed to enable the users to readily connect their wallets (e.g., MetaMask), trigger transactions, and execute their Ethereum activity. The interface is intuitive enough to provide users with seamless experience regardless of their level of familiarity.

**5. Testing**:
The Decentralized application is thoroughly tested, including performance tests and user acceptance testing (UAT). This step makes sure that the Decentralized application performs as expected and does not contain any vulnerabilities.

**6. Mainnet Deployment**:
Once carefully tested and debugged, the DApp is deployed on the Ethereum mainnet. This is done by deploying smart contracts onto the live Ethereum blockchain, bringing the DApp to real users.

**7. Post-Launch Monitoring and Maintenance**:
After release, the Decentralized Application is under constant monitoring for performance, user complaints, and security vulnerabilities. Periodic updates and improvements are carried out on the basis of user input so that the application remains efficient, secure, and in sync with any updates to the Ethereum network.

## IV. SYSTEM DESIGN

**Proposed Algorithm:**

**Step 1: Initialize the Environment**

- Set up the project using Hardhat for Solidity smart contract development.
- Install necessary dependencies (Ethers.js, react, dotenv, etc).
- Set up Hardhat with the needed network (for example, Ethereum testnet or mainnet).

**Step 2: Smart Contract Deployment**

- Compile and write the Solidity smart contract (Transactions.sol).
- Deploy the contract via Hardhat scripts.
- Save the deployed contract address for frontend usage.

**Step 3: React Frontend Setup**

- Initialize a React app with UI fields for transaction fields (receiver address, amount, message, GIF keyword).
- Integrate the frontend with MetaMask through ethers.js.
- Get the deployed instance of the smart contract via ethers.Contract().

**Step 4: User Transaction Flow**

- User Input Validation: Validate the wallet address format, Confirm that the amount entered is more than zero.
- Transaction Signing & Sending: Retrieve the sender's wallet address from MetaMask, Invoke the addToBlockchain(), function of the smart contract with the inputs provided, Ask the user to approve the transaction on MetaMask.

- Transaction Confirmation: Wait for the transaction to be mined and verified, Display the transaction gif along with sender's/receiver's address, amount in ETH, message of sender.

**Step 5: Retrieving Transactions**

- Call the getAllTransactions() method in the smart contract.
- Show the list of previous transactions, sender, receiver, amount, message, timestamp, and keyword.

**Step 5: Additional Features**

- Gas Optimization: Provide users an estimated gas fee before submitting a transaction.
- Transaction History UI: Show all transactions in a table layout with filters for simplicity.

**System Block Diagram**

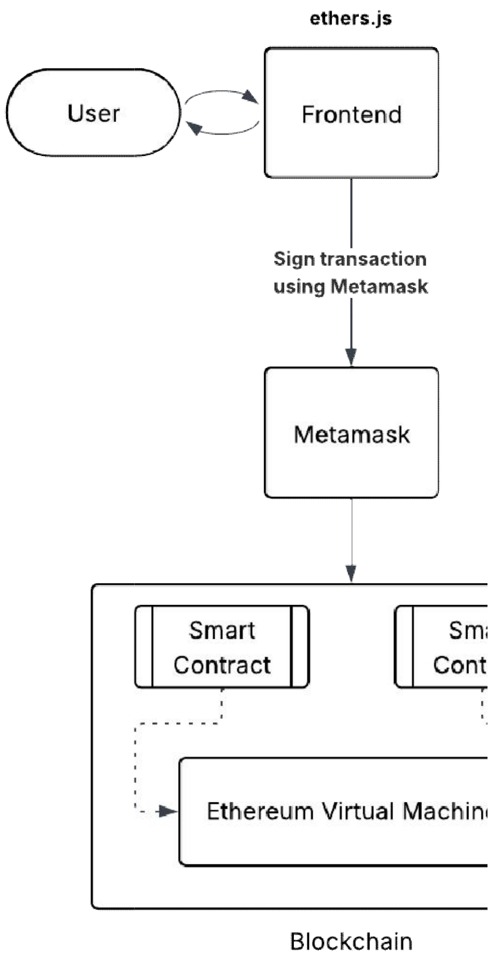
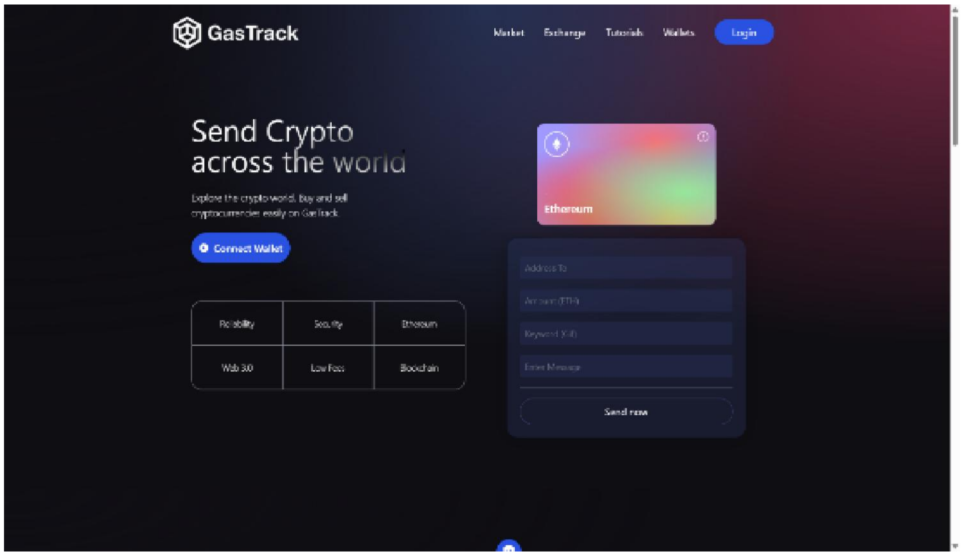


Figure 1. Block Diagram

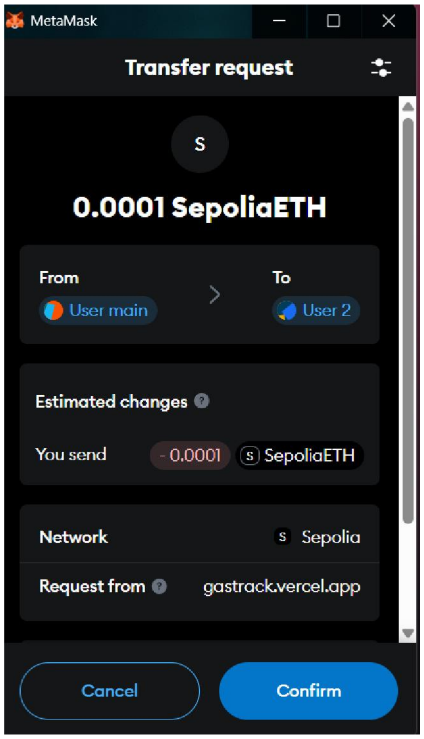
## V. IMPLEMENTATION



Send Crypto across the world

Explore the crypto world. Buy and sell cryptocurrencies easily on GasTrack

Figure 2. User Interface



Transfer request

0.0001 SepoliaETH

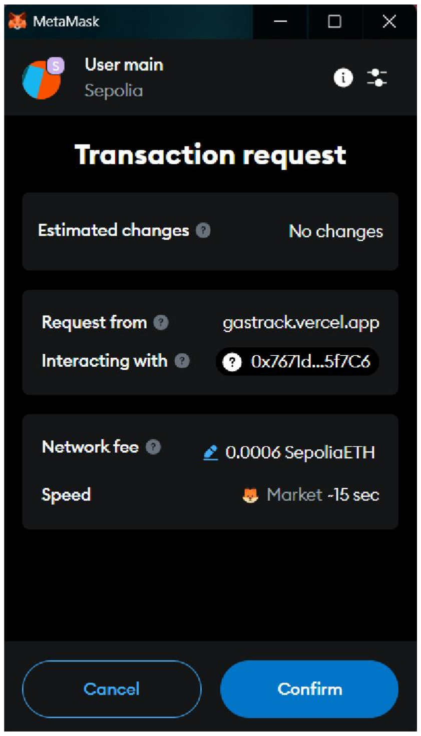
| From | To |
|---|---|
| User main | User 2 |

Estimated changes

| | | |
|---|---|---|
| You send | - 0.0001 | SepoliaETH |

| | |
|---|---|
| Network | Sepolia |
| Request from | gastrack.vercel.app |

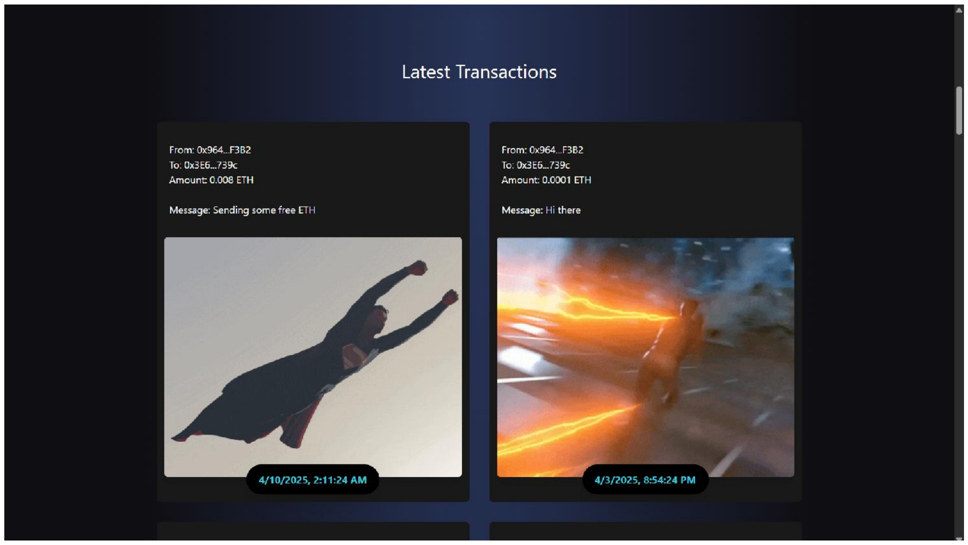
Figure 3. Transfer Request Prompt



Transaction request

| | |
|---|---|
| Estimated changes | No changes |
| Request from | gastrack.vercel.app |
| Interacting with | 0x767Id...5f7C6 |
| Network fee | 0.0006 SepoliaETH |
| Speed | Market -15 sec |

Figure 4. Transaction Confirmation Prompt



Latest Transactions

From: 0x964...F3B2
To: 0x3E6...739c
Amount: 0.008 ETH

Message: Sending some free ETH

4/10/2025, 2:11:24 AM

From: 0x964...F3B2
To: 0x3E6...739c
Amount: 0.0001 ETH

Message: Hi there

4/3/2025, 8:54:24 PM

Figure 5. Output

## VI. RESULT ANALYSIS

**1. Wallet Connection Feature:**

- **Results:** Users have the ability to connect their MetaMask wallet seamlessly to the dApp through the wallet connection feature. The success rate of connecting wallets, user interaction, and the reliability of the integration with MetaMask are monitored.
- **Analysis:** MetaMask integration was seamless, and the majority of users were able to connect successfully. The dApp was able to fetch the user's Ethereum address and have transactions signed properly. Users encountered connection issues due to browser limitations or MetaMask being locked.
- **Limitations:** Users need to have MetaMask installed, so it is less accessible for those who do not use MetaMask. Some browsers, like Safari, are incompatible.

**2. Transaction Processing Feature:**

- **Results:** The dApp allowed users to initiate Ethereum transactions worldwide. The system recorded transaction completion rates, variations in gas fees, and user experience with transaction speed.
- **Analysis:** Transactions were typically completed within Ethereum network confirmation times. Gas fees ranged based on network traffic.
- Users mostly found the transaction interface easy to use. Some users, though, were wary of gas fees.
- **Limitations**: Variable gas fees due to network congestion impact user experience.
- Lacks an integrated refund if a transaction is not completed because of a lack of gas.

**3. Smart Contract and Security:**

- **Results:** The Solidity smart contract, deployed using Hardhat, governs the transaction logic. The system analyzed contract efficiency, security vulnerabilities, and overall reliability.
- **Analysis:** The smart contract efficiently processed transactions with minimal gas consumption.
- Security audits identified no major vulnerabilities but recommended additional optimizations to reduce gas costs.
- **Limitations:** There is no upgradeability mechanism, so updates have to be redeployed.
- Gas optimization can still be enhanced for cost-effectiveness.

## VII. FUTURE SCOPE

**1. Pagination of Transaction History**

With growing transactions, loading and displaying the whole array of transaction records on the frontend side may cause performance slowdown and poor user experience. Adding paginationeither on-chain (using offset and limit parameters) or off-chain (client-side slicing with cache) would permit users to query and display data in smaller, more manageable sizes. This both enhances rendering effectiveness in the React frontend as well as minimizes memory usage and prevents unnecessary re-renders. Future versions of the DApp can use cursor-based or index-based pagination with optional filtering by date range or sender address to enable more scalable browsing of transactions.

**2. Wallet Balance Validation Before Transaction**

As of now, users can send a transaction without getting any advance feedback on whether they have enough ETH for the amount and the gas fee. To improve user friendliness and avoid unsuccessful transactions, wallet balance check can be done on the client side with Ethers.js prior to calling the send method. This would mean getting the connected user's balance through provider.getBalance(address) and checking against the input ETH value plus an approximated gas fee. Showing a prompt alert or disabling the send button where the balance is low would greatly enhance user experience and reliability.

**3. ETH Amount Validation with Debouncing**

To enhance performance and prevent unnecessary state updates or RPC requests, ETH amount input can be debounced. While the user types the amount, rather than invoking validations or calculations for each keystroke, the input can be debounced with a brief delay (e.g., 300–500ms) before performing logic like balance verification or gas estimates. This avoids excess network or computational overhead and has a smoother frontend experience. It also avoids breaking the user from premature validation feedback while typing and makes the interaction cleaner and more intuitive.

**4. Minimizing Gas Fees Through Contract & UX Optimization**

Gas fees are an essential bottleneck in Ethereum applications. To work around this, various measures can be taken. On the smart contract front, improving storage operations (e.g., removing unnecessary pushes, minimizing unwanted state modifications) and reducing costly operations can reduce deployment and execution costs. Further, utilizing Layer 2 tools like Arbitrum, Optimism, or Polygon can significantly minimize transaction costs without compromising Ethereum-level security. On the frontend, pre-estimating gas with Ethers.js and displaying it to the user helps them time their transactions during periods of low network congestion. Advanced options could include batching transactions or adopting meta-transactions to subsidize gas for users through relayers.

## VIII. CONCLUSION

In summary, the Ethereum Transaction Web3 Decentralized App developed through React, Hardhat, Ethers.js, and Solidity is an efficacious and interactive solution for the execution of decentralized peer-to-peer transactions. With the feature that enables users to enter the wallet address of the receiver, transaction value, custom message, and GIF keyword, the application elevates not only the usability of basic ETH transfer but also personalization and user interaction. The smart contract underlying it handles transaction information in an efficient manner using event logging and structured storage, while the React frontend provides a user-friendly interface. The application of up-to-date Web3 development tools and best practices renders this DApp a scalable and sustainable solution with a lot of potential for future upgrades like cross-chain compatibility, NFT support, and identity verification. Overall, the project succeeds in showcasing the potential of blockchain-based applications to provide secure, transparent, and interactive financial transactions on the Ethereum network.

## IX. FUTURE SCOPE

To improve the Ethereum transaction Web3 dApp, a few essential enhancements can be made. Pagination will be implemented to control and present transaction history effectively, so the user experience is smoother even with a high volume of data. Next.js will be used to boost SEO by supporting server-side rendering (SSR) and static site generation (SSG), allowing the app to be better discoverable by search engines and with faster loads. Wallet balance verification will be added to ensure that the user has enough ETH before sending a transaction. This will be done with debouncing so that the balance is verified only after the user finishes typing the ETH value, minimizing unnecessary API calls and enhancing performance. Moreover, gas charges will be reduced through the smart contract code being optimized, transactions being batched, and implementing Layer 2 solutions such as Polygon to eliminate network congestion fees, while maintaining user transactions as smooth and cost-efficient as possible.

## REFERENCES

**[1].** Dannen, Chris. "Introducing Ethereum and Solidity: Foundation of Cryptocurrency and Blockchain Programming for Beginners." Apress, 2017. Page 69.

**[2].** Swan, Melanie. "BLOCKCHAIN: Blueprint for a New Economy." O'Reilly Media, 2015. Page 16.

**[3].** Blockchain Quick Reference- Brenn Hill,Samanyu Chopra and Paul Valencourt Page-198.

**[4].** Copes, Flavio. "The React Beginner's Handbook: A Comprehensive Guide to Modern Web Development." Self-published, 2020. Page 4.

**[5].** Uriawan, Wisnu, et al. "Trust Lend: Using Borrower Trustworthiness for Lending on Ethereum." 19th International Conference on Security and Cryptography. SCITEPRESS-Science and Technology Publications, 2022.

**[6].** Huang, Yuxin, et al. "Blockchain Applications and Performance Evaluation in Ethereum." Journal of Physics: Conference Series, Vol. 1748, 2021, 042016.

**[7].** Sholeh, Moch, et al. "Designing an Ethereum-based Blockchain for Tuition Payment System using Smart Contract Service." Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi) 6.2 (2022): 275-280.

**[8].** Dika, Ardit," Ethereum Smart Contracts: Security",2017 Norwegian University of Science and Technology.

**[9].** Thakur, Namrata & Shinde, Dr. (2021). Ethereum Blockchain based Smart Contract for Secured Transactions between Founders/Entrepreneurs and Contributors under Start-up Projects. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 01-08.10.32628/CSEIT2174140.

**[10].** Mohanta, Bhabendu & Panda, Soumyashree & Jena, Debasish. (2018). An Overview of Smart Contract and Use Cases in Blockchain Technology. 10.1109/ICCCNT.2018.8494045.

**[11].** Chatterjee, Rishav & Chatterjee, Rajdeep. (2017). An Overview of the Emerging Technology: Blockchain.10.1109/CINE.2017.33.

**[12].** Knezevic, dusko ,"Impact of Blockchain Technology Platform in Changing",2018. Montenegrin Journal of Economics. Vol. 14, pp. 109-120.