

Implementation Smart Contract on E-Voting System for Secure and Transparent Student Election

Hussain Abdillah Tugas Kelarno¹, Widi Widayat¹

¹ Department of Informatics Engineering, Faculty of Communication and Information, Universitas Muhammadiyah Surakarta, Surakarta, Jawa Tengah 57162, Indonesia

[Received: 6 September 2025, Revised: 1 January 2026, Accepted: 19 February 2026]
Corresponding Author: Hussain Abdillah Tugas Kelarno (email: l200214201@student.ums.ac.id)

ABSTRACT — Traditional paper-based voting system for student organization leader election has issues related to security, transparency, and trust. This research addressed these issues by implementing a blockchain on e-voting system utilizing smart contracts to ensure the security and transparency of the voting process. The system was developed using the agile software development life cycle (SDLC) methodology and was tested using black-box and system usability scale (SUS) method to evaluate its functionality and usability. Security testing was conducted through unit testing on the smart contract and block verification within the Sepolia network. The results showed that the decentralized e-voting system could prevent vote manipulation and detecting duplicate voters, as evidenced by the unit testing of the smart contract, which confirmed that recorded votes could not be manipulated and attempts to submit multiple votes were detected and rejected. Meanwhile, system transparency was demonstrated through direct verification using a block explorer, showing that the entire voting process and the smart contract code were publicly accessible and transparent. The system was successfully simulated on a small scale within a student organization, and usability testing using the SUS method was conducted with 30 respondents. The test resulted in a score of 72 points, indicating that the system was in the good category and was well accepted by users. Therefore, the decentralized approach in this e-voting system has been proven to enhance transparency and overcome the problems of security issues in the voting process.

KEYWORDS — Decentralized Application, Blockchain, Smart Contract, Web 3.

I. INTRODUCTION

Traditional paper-based election systems are still commonly used in student organizational elections across university environments. However, these systems have a number of significant weaknesses, particularly in relation to issues of security, transparency, and public trust. One of the major risks occurring on traditional election is vote manipulation, such as ballot forgery or vote inflation [1]. These challenges against in terms of voter data privacy and security. In some cases, election results have also been questioned due to the lack of publicly accessible independent verification mechanisms to ensure that every vote cast was counted correctly. All of these shortcomings have the potential to decreased participants confidence in the integrity of the election process, which can affect the legitimacy of the elected student leaders [2].

With the advancement of digital technology, e-voting systems have been introduced as an alternative solution to reduce the problems of traditional paper-based elections. However, conventional e-voting systems that rely on centralized servers still having critical challenges. These include risks of centralized data breaches, unauthorized system access, vote tampering by privileged administrators, and the lack of end-to-end transparency for voters to validate whether their vote was counted correctly. As a result, although e-voting may improve efficiency, it does not fully resolve the trust and transparency issues that remain fundamental to election credibility [3].

Along with the development of technology, new approaches that can overcome problems in both traditional and centralized e-voting systems have emerged, one of which is the implementation of blockchain technology and smart contracts. Blockchain is a decentralized technology that allows every

transaction or data recorded to be immutable and publicly verifiable [4]. Meanwhile, smart contracts are digital contracts that automatically execute themselves when predetermined conditions are met, which are very useful in automating election processes such as voting and counting results [5]. This method also allows for better vote integrity because every vote received will be permanently recorded on the blockchain without being able to be changed, and the voting process can be carried out without having to sacrifice voter privacy.

Previous research has shown the potential of blockchain technology in enhancing e-voting systems. For instance, a comparative analysis underscored blockchain's benefits in verification and cost reduction but stopped short of developing a practical, implementable system [6]. Another research progressed by developing a decentralized application on Ethereum, proving its advantages in security and data integrity, yet their system encountered significant performance issues [7]. Similar studies have been conducted focused on theoretical, while the literature review confirmed security improvements, it did not address practical aspects such as scalability, performance, or user experience [8]. More practical implementation by other researchers, resulted in a secure system on a local network, but its focus was predominantly technical, paying insufficient attention to user experience [9]. Collectively, these studies confirm blockchain's security and transparent promise but reveal a consistent research gap concerning system performance and lack of attention to the aspect of user experience.

In this study, the methodology focused on two key aspects, leveraging smart contracts to automate the voting processes in student organizational elections to enhance efficiency and minimize human error, and prioritizing user experience to ensure the system accessibility. This approach was designed to

effectively address the prevalent challenges of vote manipulation and data insecurity, thereby upholding vote integrity and voter privacy throughout the electoral process. The system was tested in small-scale simulations, providing a more realistic picture of its performance and effectiveness in real elections. The approach not only implements smart contracts in election processes but also shows the demonstration of how blockchain can be adapted into accessible applications, making secure and transparent e-voting more feasible for broader adoption.

Based on the previous studies, although blockchain-based e-voting systems have demonstrated strong potential in enhancing security, transparency, and vote integrity, most existing research primarily focuses on technical aspects such as smart contract implementation, cryptographic mechanisms, and decentralized architectures, with limited attention to system usability and user experience from the end-user perspective. Furthermore, many implementations remain theoretical or restricted to local-scale deployments, lacking comprehensive evaluation of how users interact with and perceive blockchain-based voting platforms. This reveals a clear research gap in integrating blockchain technology with a user centered design approach and standardized usability assessment. To address this gap, this research aimed to design and implement a decentralized blockchain-based e-voting system for student organizational elections emphasizing both security and usability by leveraging smart contracts to automate the voting process while providing an intuitive and accessible user interface. The novelty of this research lies in its integration of blockchain's immutability and transparency with a user-friendly interface, a combination that is rarely explored in the context of e-voting using blockchain technology. Unlike conventional e-voting systems that rely on centralized databases, this approach ensures verifiable results that can be audited by all participants while preserving voter anonymity, by using block explorer. This study is significant as it contributes a practical reference for developing usable and transparent blockchain-based e-voting systems, enhances voter trust, and provides insights for future research on secure and user-centered digital voting platforms.

II. METHODOLOGY

Blockchain technology is a decentralized distributed ledger that enables transactions or data records to be stored securely, transparently, and immutably across a peer-to-peer network. Each transaction is cryptographically linked into blocks, ensuring data integrity and preventing unauthorized modification. In the context of e-voting systems, blockchain provides a reliable mechanism for preserving vote integrity, transparency, and public verifiability without relying on a centralized authority [4]. Smart contracts further extend blockchain functionality by enabling self-executing programs that automatically enforce predefined rules, such as vote submission, validation, and tallying, thereby reducing human intervention and potential manipulation [5].

Based on these theoretical foundations, the development of the proposed e-voting system requires an implementation approach that accommodates iterative testing, smart contract refinement, and continuous user feedback. Therefore, this system was implemented using the software development life cycle (SDLC) with the agile methodology. The agile approach was selected due to its flexibility, responsiveness to evolving requirements, and emphasis on user involvement, which

collectively supported the iterative development and validation of blockchain-based applications. Through incremental development cycles, agile facilitates timely system delivery while ensuring software quality and alignment with functional and usability requirements [10].

A. REQUIREMENTS

The purpose of this stage is to understand the main problems in traditional election systems and identify technology-based solutions that can be applied in blockchain-based e-voting systems by analyzing the requirements [11].

1) FUNCTIONAL REQUIREMENTS

The functional requirements of this blockchain-based e-voting system included several important aspects that must be met to ensure a smooth election process. First, the system must provide a secure voter registration mechanism, including identity verification before voters could participate in the election. Once registered, voters must be able to cast their votes anonymously, where the votes were recorded on the blockchain to maintain data integrity. In addition, the system must automate the vote counting through smart contracts, so that the election results could be calculated and published automatically without manual intervention. The system must display the election results after all votes were counted.

2) NONFUNCTIONAL REQUIREMENTS

The nonfunctional requirements of this system included various aspects, such as security. The security of a website is important, so it is recommended to test the security of the website using one of the frameworks or methods to avoid hacking [12]. The system must be able to protect voter data and protect votes from manipulation by other parties.

B. DESIGN SYSTEM

The design system aimed to design the entire information system architecture and creating blueprint for the system. This phase provides an overview flow of the information system to be created [13].

1) USE CASE DIAGRAM

This system had two actors: admin and user. Admin was an actor managing and organized users, also managing this system by starting and ending the voting event. Admin managed and maintained smart contracts and blockchains used to recap votes. User was an actor using the system, users could submit and see the final results of the voting. In addition, users must be verified first before voting. The use case diagram image can be seen in Figure 1.

2) ACTIVITY DIAGRAM

Figure 2 shows an activity diagram from the beginning to the voting process, starting from logging in by entering a username and password that previously registered by voter. After successfully logging in, the user was required to connect their wallet address to this system in order to use this system. After that, the user could vote, submit, and confirm their vote using MetaMask wallet, after that each submitted vote was recorded on the blockchain through a smart contract deployed on the Sepolia test network. After the election was ended, the user could see the results in the result menu.

The "Blockchain validates the vote" process on the activity diagram followed the proof of stake (PoS) consensus mechanism, which was used by the Sepolia network as part of Ethereum's transition after the merge [14]. Through PoS, transactions containing the vote were validated by the network

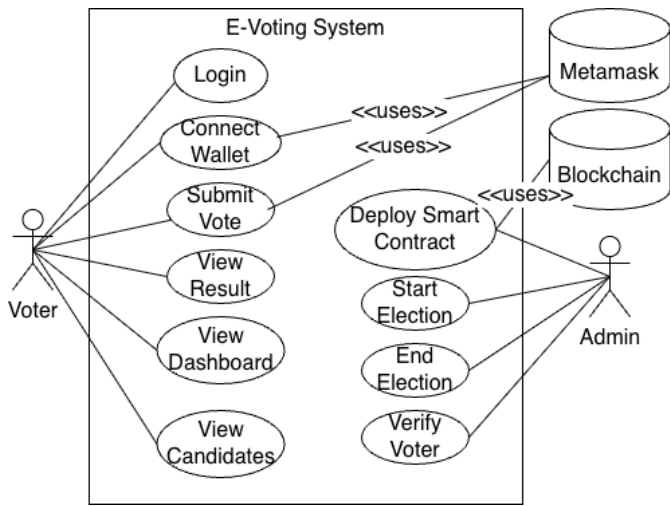


Figure 1. Use case diagram of e-voting system.

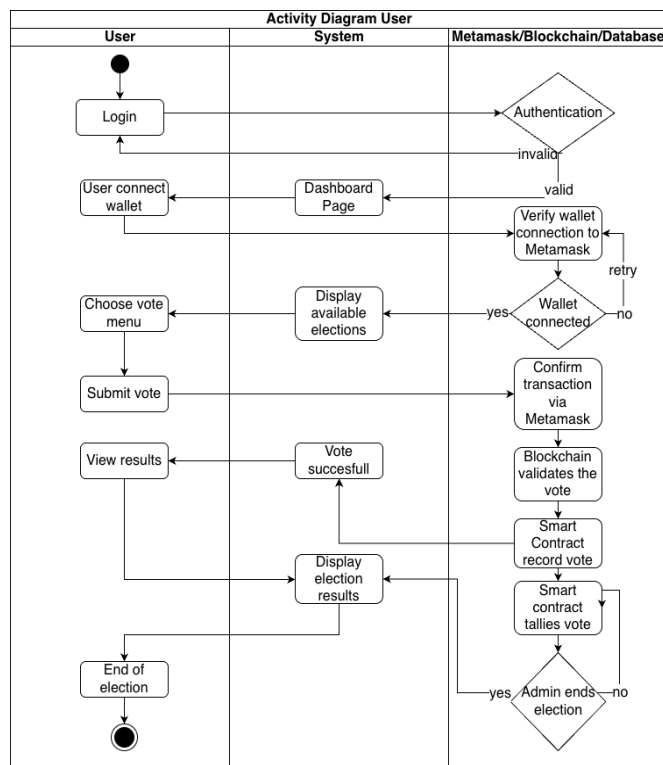


Figure 2. Activity diagram.

validators based on staking participation, ensuring the vote was confirmed securely and efficiently before being permanently recorded by the smart contract. PoS was due to its higher efficiency, lower energy consumption, and faster transaction confirmation compared to proof of work (PoW), making it more suitable for an e-voting environment requiring transparency, integrity, and near real-time validation of votes without introducing excessive computational overhead [15].

3) ENTITY RELATIONSHIP DIAGRAM

Admin managed ongoing elections by adding candidates, starting, and ending elections, while smart contract was deployed when admin started an election. The election contained several candidates. Users casted their votes through the vote entity, associating voters with elections and the selected candidates. The ERD for this e-voting system can be seen in Figure 3.

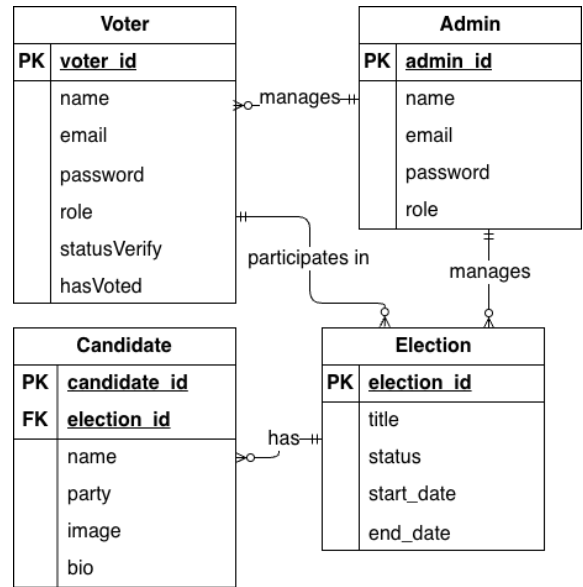


Figure 3. Entity relationship diagram of e-voting system.

4) ARCHITECTURE DIAGRAM

The architecture of this e-voting system used MetaMask for user verification and smart contract signing. Users were required to connect their web 3 wallet with the e-voting application using MetaMask. The use of digital crypto wallet such as MetaMask aimed to decentralize identity and cryptographic signing tool. MetaMask provides users with their own blockchain address and private key, enabling the e-voting process to be executed in a trustless environment without depending on the system administrator. Each vote submitted to the smart contract is cryptographically signed through MetaMask, ensuring that the vote is proven to originate from the actual voter, cannot be altered, and cannot be denied afterward. This mechanism eliminates the risk of server-side vote manipulation, and ensures transparency and auditing capability, which are essential in election systems [16].

Using the ethers.js library, the frontend application connected to the MetaMask, bridging the web application with the deployed smart contract on the blockchain. After the user selected a candidate, the voting data were sent from the web application to the blockchain, where the votes were recorded using a smart contract and validated by the Sepolia network. Each vote transaction was cryptographically signed by the user through their MetaMask wallet, ensuring both authenticity and undeniable. The architecture design for this e-voting system can be seen in Figure 4. This figure illustrates the interaction between different components, including the admin application, user application, database, blockchain/smart contract, and supporting libraries. The system was divided into two main layers, the e-voting admin application and the e-voting user application, which were developed using Next.js.

On the admin side, the system provided a main dashboard where administrators could manage users, candidates, and elections, as well as set the start and end of an election. All these administrative operations were connected to the database layer, which utilized Firebase and PostgreSQL through Prisma as the object-relational mapping (ORM), ensuring efficient and structured data management.

The smart contract implemented in Solidity and deployed on the Sepolia network served as the core of the voting process

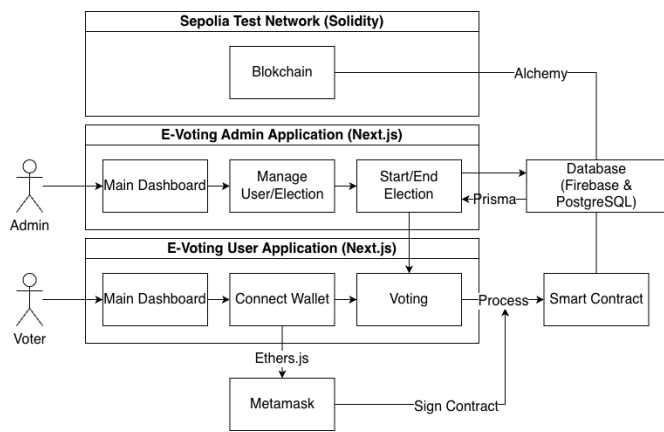


Figure 4. Architecture diagram of e-voting system.

by recording and securing votes in an immutable blockchain. Deployment of the smart contract to the blockchain network was established using Alchemy, acting as a bridge that connected smart contract developed using hardhat, with the blockchain network that stored contract, which in this case using Sepolia network. This ensured that vote transactions and the smart contract logic were publicly verifiable while maintaining voter anonymity.

Overall, this architecture ensured that administrative tasks, user interactions, and blockchain operations were seamlessly integrated, with smart contracts handling the vote processing and conventional databases supported auxiliary operations, such as users, candidates, and election data configuration. This design emphasizes security, transparency, and reliability, aligning with the main objectives of the decentralized e-voting system.

5) USER INTERFACE DESIGN

User interface design is an important aspect of software development, affecting user interaction and the initial design for building the appearance of the system. Tools such as Figma can be used for designing the user interface. User interface design focuses on creating visually appealing and user-friendly layouts, utilizing elements such as fonts, colors, icons and structures to enhance the user experience [17].

C. IMPLEMENTATION

Implementation phase began with the implementation of code and systems based on the system design that had been made. The development was carried out iteratively in several sprints, where each sprint focused on developing a specific part of the system [18]. The system implementation used Typescript programming language and Next.js framework (see Figure 4) because they are more flexible and efficient for developing modern applications that have a lot of routing, and this framework also provides a more organized and modular project structure [19]. The Blockchain technology was used to store voting data, while smart contract technology from Solidity and the Ethers.js library were utilized to interact with the Ethereum blockchain and smart contracts used Sepolia network. Wallet address management (shown in Figure 4) used third party extensions from MetaMask so that users could access decentralized application directly from their browser and made transactions on the Ethereum blockchain. PostgreSQL and Firestore database were used to store other information that was not stored on the blockchain. Hardhat was utilized for testing smart contracts environment locally. Transaction votes must be

stored on the Ethereum blockchain and not stored in a conventional database because they were sensitive information. If transaction data are stored in a regular database, they become vulnerable to hacking and data leaks [20]. Therefore, it is more important to store vote transaction data on the blockchain rather than in a conventional database.

The deployment process was divided into two parts, deploying smart contract to the Sepolia network and the front-end of the system. Smart contract deployment process was carried out using Alchemy, third party platform that helps developer deploying smart contract to live network so it can be accessible by everyone [21]. Front-end part of this deployment was containerized using Docker, this process ensured that the application ran consistently across different environments, from development to production [22]. Virtual private server (VPS) was used to host all the services such as front-end application and PostgreSQL database instance. After the front-end instance was containerized with Docker, the next phase was to deploy the docker image to the VPS. The platform that was used to deploy the front-end was Zeabur. This platform provided option to host projects using docker images, also allowing it to install any services and dependencies that required to run the instance.

The blockchain infrastructure used in this study was based on the Ethereum Sepolia test network, which inherently operates as a distributed blockchain environment maintained by multiple validator nodes. Rather than implementing a custom blockchain network, this research focuses on the application layer by deploying and interacting with smart contracts on an existing public blockchain. Communication between the web-based e-voting application and the blockchain network is facilitated through Alchemy as a node provider, enabling reliable access to the decentralized network. As a result, the distributed nature, consensus mechanism, and data replication are handled by the Ethereum network itself, while this study concentrates on the secure implementation of voting logic, transaction verification, and transparency through smart contracts.

D. TESTING METHOD

Several testing methods were used in this study, including black-box testing, system usability scale (SUS) method, and blockchain-specific testing through unit testing of the smart contracts, as well as verification using a block explorer. Black-box testing was used to evaluate the e-voting system's functionality by examining its outputs in response to various inputs, without knowledge of the internal code, ensuring user requirements are met [23]. The black-box method was used to test several cases, including login authentication, account access, and verified registered voters who can vote. To test whether this system ran according to the functional requirements determined at the beginning, the entire system was tested using the SUS method. The SUS method was employed since it is a reliable tool for measuring usability, even with small sample sizes, providing a score to evaluate user satisfaction and experience [24]. Meanwhile, the unit testing method was used to test the security of the system on a smart contract. Several test cases were conducted, including verifying whether users could perform double voting and ensuring that votes could not be changed or modified after being recorded. This case was tested using unit testing to ensure that every function in the smart contract complied with the specified case specifications. By using unit testing, developers can detect

logical errors at the most basic level of the smart contract code [25]. Verification on block explorer was used to audit, ensuring that this system was transparent and secure by providing a verification for each voting transaction on a block explorer. This method facilitates audits without violating voter privacy. It proves that the implemented e-voting system is transparent and can be publicly audited in accordance with the blockchain principles, public, and immutable [26]. The blockchain-specific testing was conducted through unit testing of the smart contract and direct verification on the blockchain block explorer. The unit testing was applied to validate the correctness and security of smart contract functions, including vote immutability and prevention of double voting, allowing logical errors to be detected at the code level. In addition, verification on the Sepolia block explorer was performed to ensure transparency and auditability by allowing public inspection of voting transactions and verified smart contract code without revealing voter identities, in accordance with blockchain principles of immutability and openness.

III. RESULTS AND DISCUSSION

The e-voting system was developed according to the designed method. As a result, the system could record votes from users using their wallet addresses and the system successfully recorded the voting results using smart contracts deployed on the Sepolia network.

A. RESULTS OF THE SYSTEM

This e-voting system was successfully developed and deployed for the internal use within the campus organization. Students participated as testers by accessing the website and simulating the voting using the system. Prior to the voting, students were required to create an account and wallet so that they could vote using the system. The systems also provided page for verification to restrict unverified students from voting and abusing the system. Student must verify their identity, ensuring that they were eligible voters. On the verification page, users were required to complete the verification process by providing their student ID and undergoing face recognition process to confirm that they were the original holder of the student ID. Figure 5 shows the details the student eligibility verification process.

On the voting process, users could directly vote and confirm their choice using the MetaMask wallet that they had created and connected to the website. MetaMask was necessary because it was used to interact and send the vote to the smart contract. Users were required to confirm the process in their MetaMask wallets to send a vote, and a Sepolia token was used for network fees. Figure 6 shows an example of the e-voting system display during the voting process, as the user chooses a candidate.

After confirming the transaction using MetaMask, the smart contract validated the transaction. If the process was successful, the user got a transaction hash, displayed on the receipt that user received on the website. The system performed as expected, and the smart contract-based e-voting simulation was successfully executed. All vote data were recorded and stored on the blockchain and publicly and transparently accessible to everyone. User were able to view the current election results through the results menu. The details of the results can be seen on Figure 7.

The admin role in this system was responsible for managing the ongoing election and candidate selection. They were

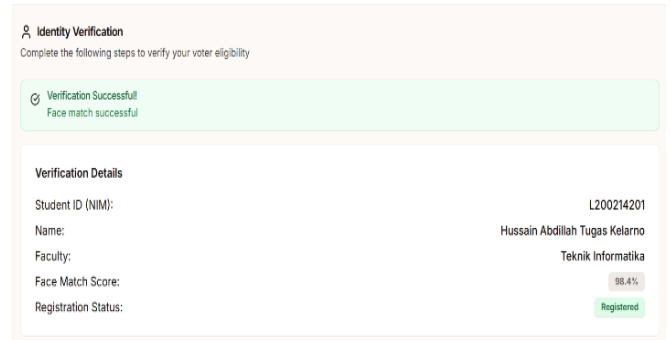


Figure 5. Student eligibility verification process.

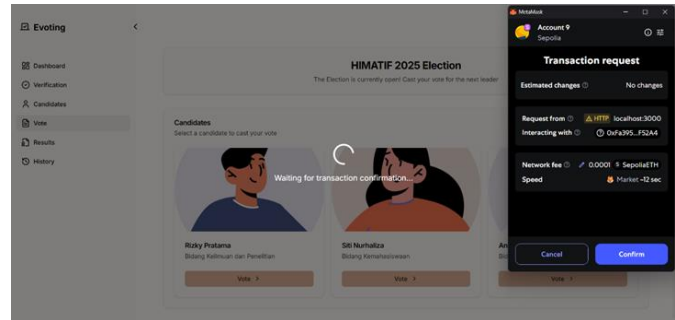


Figure 6. User voting candidate selection process.

Results Summary

Image	Name	Party	Votes	Percentage
	Rizky Pratama	Bidang Keilmuan dan Penelitian	9	33.3%
	Siti Nurhaliza	Bidang Kemahasiswaan	12	44.4%
	Andi Wijaya	Bidang Minat dan Bakat	6	22.2%

Figure 7. Election results.

responsible for setting the start and end dates of the election. The status of the election was automatically set to active if the current date fell within the start and end time period. If current date fell outside the specified period, the election was automatically set to inactive, and the user could not vote. The admin was also responsible for managing all the candidates, such as adding, editing candidates, and deleting candidates. This also applied to the user management page, the admin was able to add user accounts by adding email and password directly, thereby users could directly log in using the credentials provided by the admin. In addition, the admin was liable for deploying a smart contract to record user vote data.

The system interface ran as expected and in accordance with the requirements functions, allowing it to monitor the election and manage all registered users. As shown in Figure 8, the admin dashboard monitors the election's status and conditions in real time.

B. TESTING AND SYSTEM EVALUATION

The evaluation and testing of the system confirmed its operational success, demonstrating the capability to accurately capture votes linked to user wallet addresses. Furthermore, the deployed smart contract on the Sepolia network functioned as intended, successfully and immutably recording the final voting results. The methodology and evaluation process are systematically presented through a series of functional, performance, and usability tests, where each testing scenario and its corresponding results are clearly explained in detail. In addition to conventional system testing, blockchain-specific

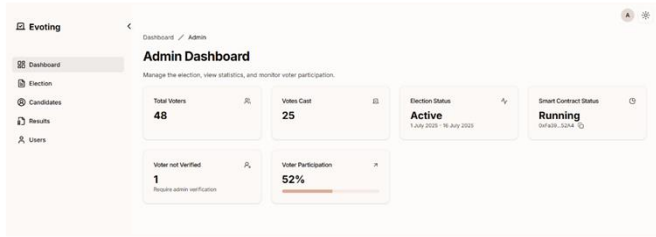


Figure 8. Admin dashboard display.

evaluations were conducted through unit testing of the smart contract using the Hardhat framework with the Chai testing library to verify critical properties such as vote immutability, single-vote enforcement, and access control. Moreover, direct verification was performed using the Sepolia block explorer to publicly inspect transaction records and the deployed smart contract code, ensuring transparency, correctness, and auditability of the voting process. This structured evaluation approach ensured that the system’s reliability, correctness, usability, and blockchain integrity were thoroughly assessed and transparently reported.

1) BLACK-BOX TESTING

E-voting system functionality running properly, the test conducted using black-box testing method. All test cases ran according to the expected results. Based on the black-box test, the authentication test case showed valid results for user and admin login attempts: they were successfully redirected to their respective dashboards, while invalid credentials returned an appropriate error message. The sign-up process also ran as expected, where valid new user data could be registered and duplicate accounts or mismatched passwords displayed proper error notifications. This result indicates that the system’s authentication and account creation mechanisms are reliable in preventing unauthorized access and ensuring data integrity. In the voting process, several scenarios were tested to ensure vote integrity. A user connected to their wallet and confirmed the transaction via MetaMask, successfully recorded their vote on the smart contract, and received a success receipt. Conversely, when users attempted to vote without connecting their wallet, cancelled the transaction in MetaMask, or attempted to vote more than once, the system correctly displayed error messages and prevented the action to vote. Similarly, unverified users and users accessing the system outside the election period were restricted from voting, ensuring that only eligible voters could participate at the correct time. These results validate that the system effectively mitigates risks of double voting, unauthorized access, and improper timing of participation. Table I presents the results of black-box testing on this e-voting system.

For the election management case, the admin was able to set and update schedules, add new candidates with complete information, and edit or delete candidate data. Incomplete data entry triggered the system to display error messages, which enforced data consistency. Additionally, the system allowed the admin to add and manage user accounts, including verifying or unverified users, with results accurately saved in the database. The vote results feature was validated by displaying results directly from the deployed smart contract, ensuring transparency and preventing manipulation. Since the results were sourced from blockchain data, neither admin nor users could alter them, which enhances trust in the integrity of the election outcome. The logout process also functioned correctly,

TABLE I
BLACK-BOX TEST RESULTS

Test case	Input	Expected Results	Results
Login user	Valid email and password	Show user page	Valid
User Voting	Vote button and confirm transactions	Voting successfully	Valid
User verification	Unverified users accessing page	Your account is not verified	Valid
Election	Admin sets election schedule	Schedule saved, and successfully updated	Valid
Adding new data	Admin inputs data	Successfully added additional data	Valid
Vote results	Enter results menu	Displays voting results from smart contract	Valid

returning both user and admin sessions to the login page after termination.

Overall, the black-box testing confirms that this e-voting system fulfills its intended functionalities while addressing critical issues such as security, transparency, and reliability. Each test case resulted in valid outcomes, demonstrating that the implementation of blockchain and smart contracts provides an effective safeguard against common problems in traditional e-voting systems, such as double voting, data manipulation, or unauthorized access.

2) SUS TESTING

The SUS testing is done by giving 10 questions and measured using a Likert scale [27]. Testing was conducted among students who participated in university student organization activities, resulting in 33 respondents. A total of 28 students participated as users and 5 students participated as admins. The score was categorized into five levels of usability based on the following score ranges: scores of 80.3 and above were classified as excellent, scores between 68 and 80.2 as good, scores from 51 to 67.9 as OK, scores between 38 and 50.9 as poor, and scores below 38 as awful [28]. The calculation of the SUS score in this research followed a structured process.

$$Qx - 1. \tag{1}$$

Equation (1) calculated, for each odd-numbered question, the respondent’s score using the formula, where Qx represents the score for the odd question.

$$5 - Qn. \tag{2}$$

Meanwhile, for each even-numbered question, the score was transformed using (2), where Qn is the score of the even question. As a result, each item produced a value ranging from 0 to 4. These values were then summed across all ten questions, yielding a maximum possible total of 40 points per respondent.

$$\bar{x} = \frac{\sum x}{n}. \tag{3}$$

To convert into a standardized SUS score on a scale of 0–100, the total was multiplied by 2.5. The final SUS result for

the system was obtained by calculating the average score across all respondents using (3), where \bar{x} is the mean of the SUS score, $\sum x$ is the total of all respondent score, and n is the number of respondents.

Based on the SUS testing results, the tests conducted on the user and admin roles (Figure 9) showed almost identical results. User respondent showed that the average score obtained was 71, and for admin respondent the average score obtained was 74. Each scores indicates that the system is at a good category.

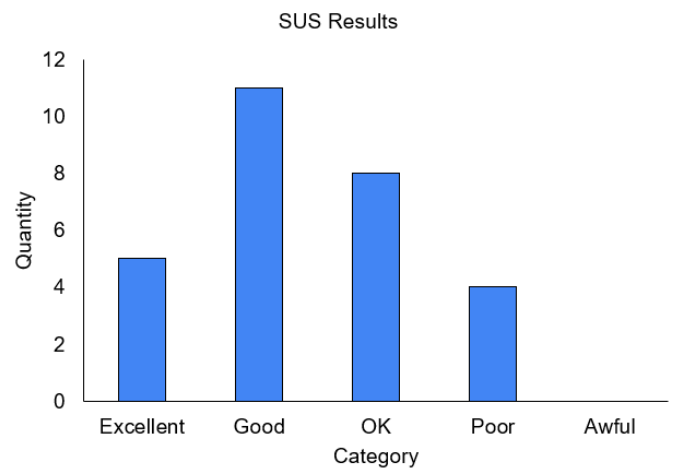
3) SMART CONTRACT UNIT TESTING

Unit testing was carried out by using code script running in the Hardhat project environment, containing the Solidity smart contract code and the unit test script code. Several scenarios were designed to validate the functionality and security of the smart contract used in the e-voting system. The first test verified that a user could successfully cast a vote to the smart contract. When a user voted for a candidate with an assigned wallet address, the vote was correctly recorded, and the candidate's vote count increased as expected. Another test case was conducted to ensure that one account could not vote more than once. When the vote function was called twice by the same account, the second transaction failed with an error, thereby confirming the one-vote-per-user rule. Another scenario tested the accumulation of votes from multiple voters. When multiple different accounts cast votes for the same candidate, the get votes function accurately displayed the total number of votes, matching it with the number of accounts that had participated. To verify whether a user had already voted, the system checked the wallet address associated with the smart contract. The result returned "true" if the user had previously voted and "false" if they had not, which confirmed the correctness of the vote-tracking mechanism.

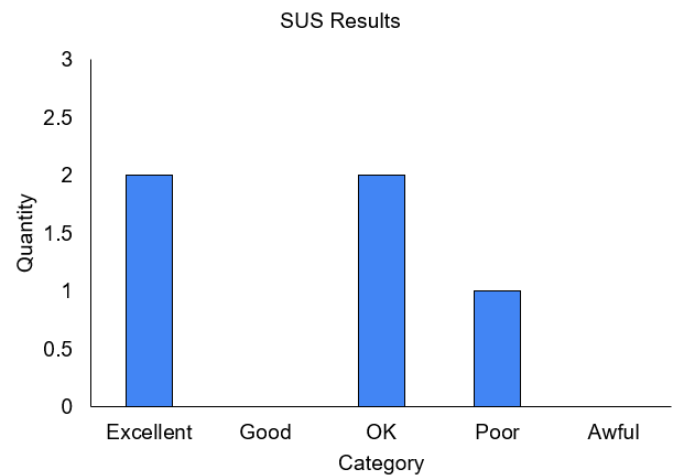
Further test scenario addressed vote immutability. When a user who had already voted attempted to override their choice by calling the vote function again with a different candidate ID, the original vote data remained unchanged, thereby preventing vote modification. Lastly, the system was tested to determine whether the administrator could directly manipulate voting results. When an attempt was made to override votes within the smart contract, the transaction returned an error since no function was available to modify the recorded votes. This confirmed that even administrators were unable to alter the stored vote results, ensuring fairness and integrity in the voting process. Figure 10 presents the detailed results from the Solidity smart contract unit testing by running it through the script code and Hardhat project environment. The test results showed that all unit testing scenarios were successful and in accordance with expectations, with 6 passing and a test duration of 574 ms.

4) VERIFICATION ON BLOCK EXPLORER

Verification in the block explorer is a testing method that directly verifies the smart contract deployed through it. This test aimed to demonstrate that all voting transactions in the e-voting system were well audited and publicly accessible. Voting transactions performed by users could be publicly accessible, ensuring their validity while maintaining anonymity, as the smart contract did not store any personal identity information but only the wallet address known to each user. The deployed smart contract code was also publicly available, allowing users to verify its functions and ensure that no manipulation attempts were made by the administrator or



(a)



(b)

Figure 9. Results of SUS testing, (a) respondent from user, (b) respondent from admin.

```

Voting Contract
✓ Should allow a user to vote
✓ Should not allow a user to vote twice
✓ Should return correct vote count for multiple users
✓ Should track if address has voted

Voting Contract - Immutability Test
✓ Should not allow override vote and modification after submission
✓ Should not allow the contract owner (admin) to modify votes directly

6 passing (574ms)
    
```

Figure 10. Unit testing code results.

organizer. The verification test was conducted by first accessing the Sepolia block explorer website and entering the transaction hash into the search bar. Once entered, the transaction details could be examined, followed by checking the transaction history recorded on the smart contract address used to store votes. Then, verified source code of the smart contract could be reviewed to confirm its authenticity and integrity in handling the voting process.

The results showed that all voting transactions recorded by the smart contract. Although users could see the transaction history, they could not know the candidate that other users selected due to the anonymity. They could only view the wallet address that sent the transaction without knowing the identity of the person behind the address. They also could view detailed

information such as transaction hash, status, timestamp, wallet address, smart contract address and network fee. Users might be able to check the Solidity contract code deployed to the smart contract to find out any manipulation attempt by the admin/organizer regarding the vote collection/counting process. All codes could be accessed publicly and verified by the system from the Sepolia block explorer. Detailed information from the smart contract can be seen in Figure 11.

With the results of the testing and verification, it is proven that the implemented e-voting system has transparency and it is in accordance with the blockchain principles, being open and auditable by all users but also anonymous. This also supporting the security and trust in e-voting principles.

C. DISCUSSION

Compared to traditional centralized e-voting systems that rely on server-side databases and administrator, this research introduces a decentralized architecture. Core election rules, such as vote integrity, and single-vote enforcement are governed directly by immutable smart contract logic. This eliminates reliance on trusted intermediaries and reduces the risk of data manipulation or administrative abuse. Furthermore, although previous studies have implemented blockchain-based e-voting systems, many of them are limited to local or experimental implementations and primarily focus on the technical feasibility of blockchain integration, with minimal attention to user interface design and user experience. In contrast, this research combines protocol-level security and transparency with a user-centered web-based interface, ensuring that the system is not only secure and auditable but also accessible and usable for non-technical users. These findings emphasis on decentralized security mechanisms and improved usability represents the main novelty of this study and contributes to enhancing security, transparency, and trust in practical e-voting deployments. Building upon these findings, the following subsections further discuss the key results of this research.

1) FUNCTIONALITY AND RELIABILITY

The success of the black-box testing demonstrates that the e-voting system functions correctly from the end user's perspective. Each key interaction from wallet integration to casting and viewing votes performed as intended. This suggests that the system meets the functional requirements necessary for an operational voting process.

2) USABILITY EVALUATION

The SUS scores from user respondents averaged 71, while those from admin respondents averaged 74. These results suggest that the system has a good level of usability and is acceptable to both users and administrators. However, qualitative feedback indicated that while some users appreciated the security and innovation of the system, others felt unfamiliar with using wallets such as MetaMask, highlighting a learning curve in adopting blockchain-based applications. This suggests that educational materials or user guidance may be necessary for broader adoption.

3) TRANSPARENCY AND TRUST

Contract verification via the Sepolia block explorer significantly enhances the system's transparency. Users and auditors can inspect the logic behind the voting process, increasing trust in the system's integrity. This addresses one of the study's core objectives by providing an auditable, tamper-proof election mechanism [4]. The unit testing results confirm

Transaction Hash	Method	Block	Age	From
0x832bb0e2220...	Vote	8923998	29 days ago	0x6c490A56...cB501809F
0x08d89581cb...	Vote	8761125	51 days ago	0x1cb21BfE...c5e3F68aa
0xebc093f470c...	Vote	8754736	52 days ago	0x62672c4E...c75249d58
0x7d68b65a11...	Vote	8680753	63 days ago	0x05e15Aa5...31805d398
0x84c1dca56b...	Vote	8675544	63 days ago	0x896A9389...f8a008e0D
0xbd077ce5ca...	Vote	8675533	63 days ago	0xE81e5c14...4Cf6f803D
0xe3ee95df920...	Vote	8675465	63 days ago	0xF4243608...AB9E065da

Figure 11. Transaction history from block explorer.

that the smart contract logic is robust and performs critical checks to preserve voting integrity. These include preventing duplicate votes and restricting administrative functions to authorized addresses. It can be concluded that the smart contract is secure and immutable, because voters who have made transactions/votes cannot add or manipulate data that has been confirmed by the smart contract. This finding supports the argument of previous research that the immutability of data is a key to secure digital voting [8].

4) CONTRIBUTION TO VOTING SECURITY

By implementing blockchain and smart contract technology, the system offers several security enhancements over a traditional web-based e-voting system. Data immutability, decentralized verification, and wallet-based identity reduce the risk of tampering, fraud, or abuse of central control. Based on the testing and verification results, it is proven that the implemented e-voting system provides transparency and is in accordance with blockchain principles, being open and auditable by all users, while also being anonymous [26]. This also supports the security and trust principles of e-voting.

5) IMPLICATIONS, LIMITATIONS, AND FUTURE RESEARCH

The findings demonstrate that blockchain technology is an effective solution for building secure and transparent e-voting systems. However, this study has limitations. It was deployed on an Ethereum Sepolia test network, so issues of transaction costs (gas fees) and scalability under high load were not fully addressed.

Future research should focus on layer-2 scaling solutions to reduce gas fees and improve transaction throughput. An alternative deployment approach is to implement the e-voting system on a private or permissioned blockchain network that does not require gas fees. By operating in a private environment, the system can eliminate transaction costs and reduce reliance on public validators, making it more suitable for institutional or organizational elections [29]. In this context, the consensus mechanism can be shifted from PoS to more efficient models such as proof of authority (PoA) or practical Byzantine fault tolerance (PBFT). These consensus mechanisms do not rely on token-based incentives, offer faster transaction finality, and are well-suited for trusted or semi-trusted environments. Platforms such as Hyperledger Besu can be considered, as they support permissioned networks and PoA or PBFT-based consensus while preserving core blockchain principles such as immutability, distributed validation, and audibility. This

approach enables the e-voting system to operate without gas fees while maintaining security, transparency, and data integrity [30]. Additionally, usability and trust cases should be conducted with a more diverse demographic to ensure the system is accessible and trustworthy for general citizens.

IV. CONCLUSION

The results of the e-voting website showed that the developed system was able to overcome the problems of security, transparency, and trust in the process of student organization leaders' election through the implementation of blockchain technology and smart contracts. This system was built using the Agile SDLC methodology and tested through black-box testing, SUS, smart contract unit testing, and verification through a block explorer on the Sepolia network. The results of the black-box test showed that all functionality ran as expected without errors. The results of the SUS testing were conducted on two types of respondents, users and admins. Results from user respondents show an average score of 71. Admin respondents show results an average score of 74. Which each score was included in the good category and was acceptable by both users and admin. The smart contract test carried out using unit testing proved to be secure, immutable and could not be manipulated by users or admins, and the voting process has transparent and anonymous which could be verified by the public through a block explorer. Thus, the decentralized e-voting system developed in this research proved that it can enhance security, transparency, and trust in the voting process.

CONFLICTS OF INTEREST

The authors of this research declares that this work is free from any conflicts of interest.

AUTHORS' CONTRIBUTIONS

Methodology and system implementation, Hussain Abdillah Tugas Kelarno; writing—original draft preparation, Hussain Abdillah Tugas Kelarno; writing—reviewing and editing, Widi Widayat; guidance and supervision, Widi Widayat.

ACKNOWLEDGMENT

I would like to express my sincere appreciation to my supervisor, Mr. Widi Widayat, S.Kom., M.Eng., for his invaluable guidance, patience, and insightful feedback throughout this research process. As well as all the lecturers and staff of the Informatics Engineering Universitas Muhammadiyah Surakarta, for their invaluable guidance and support so that this research can be completed successfully.

REFERENCES

[1] S. Wulandari, Hendrawansyah, and N. Hariyani, "Sistem informasi e-voting pemilihan ketua badan eksekutif mahasiswa (BEM) berbasis web pada kampus STMIK Amika Soppeng," *J. Minfo Polgan*, vol. 14, no. 1, pp. 490–500, May 2025, doi: 10.33395/JMP.V14I1.14761.

[2] M.A. Adhani *et al.*, "Pemilihan ketua dan wakil ketua badan eksekutif mahasiswa Universitas Lambung Mangkurat (BEM ULM) berbasis elektronik," *KLIK (Kumpul. J. Ilmu Komput.)*, vol. 8, no. 3, pp. 291–304, Oct. 2021, doi: 10.20527/klik.v8i3.404.

[3] J.F. Potalangi, D.P. Kartikasari, and N.H. Shaffan, "Implementasi jaringan permissioned blockchain pada sistem e-voting pemilwa untuk menjamin autentikasi pemilih dan integritas data," *J. Pengemb. Teknol. Inf. Ilmu Komput.*, vol. 9, no. 4, pp. 1–14, Apr. 2025.

[4] Y. Chen and C. Bellavitis, "Blockchain disruption and decentralized finance: The rise of decentralized business models," *J. Bus. Ventur. Insights*, vol. 13, pp. 1–11, Jun. 2020, doi: 10.1016/J.JBVI.2019.E00151.

[5] S. Khan *et al.*, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-Peer Netw. Appl.*, vol. 14, pp. 2901–2925, Sep. 2021, doi: 10.1007/s12083-021-01127-0.

[6] U. Jafar, M.J.A. Aziz, and Z. Shukur, "Blockchain for electronic voting system—Review and open research challenges," *Sensors*, vol. 21, no. 17, pp. 1–22, Sep. 2021, doi: 10.3390/s21175874.

[7] Z.W. Alfain, H. Setiawan, and I.K.S. Buana, "Analysis of centralized vs decentralized electronic voting," in *2022 IEEE 8th Inf. Technol. Int. Semin. (ITIS)*, 2022, pp. 173–177, doi: 10.1109/ITIS57155.2022.10010100.

[8] P. Rahi *et al.*, "Use of blockchain technology in electronic voting systems: An overview from computer security," in *2023 Int. Conf. Commun. Secur. Artif. Intell. (ICCSAI)*, 2023, pp. 1–5, doi: 10.1109/ICCSAI59793.2023.10420927.

[9] W.A.B.W. Abdulah and S.F.S. Adnan, "Blockchain based electronic voting system design with smart contracts," in *2023 IEEE Symp. Comput. Inform. (ISCI)*, 2023, pp. 98–103, doi: 10.1109/ISCI58771.2023.10391913.

[10] S.G. Tetteh, "Empirical study of agile software development methodologies: A comparative analysis," *Asian J. Res. Comput. Sci.*, vol. 17, no. 5, pp. 30–42, Feb. 2024, doi: 10.9734/ajrcos/2024/v17i5436.

[11] K. Asad and M. Muqem, "A critical analysis of requirement management in agile development," in *Adv. Data Inf. Sci.*, 2022, pp. 79–93, doi: 10.1007/978-981-19-5292-0_8.

[12] D. Priyawati, S. Rokhmah, and I.C. Utomo, "Website vulnerability testing and analysis of internet management information system using OWASP," *Int. J. Comput. Inf. Syst. (IJCIS)*, vol. 03, no. 03, pp. 143–147, Sep. 2022.

[13] Wisnumurti, Y. Trimarsiah, and S.T. Faulina, "Penerapan agile development methodology pada sistem informasi penjualan ecer dan grosir Toko Kinanti Martapura," *JUTIM (J. Tek. Inform. Musirawan)*, vol. 7, no. 2, pp. 109–120, Dec. 2022, doi: 10.32767/JUTIM.V7I2.1727.

[14] N. Kandwal, K.L. Agarwal, and D. Upadhyay, "Ethereum's merge: A comprehensive analysis of per-block energy consumption, greenhouse gas emissions and block size," in *2025 2nd Int. Conf. Comput. Intell. Commun. Technol. Netw. (CICTN)*, 2025, pp. 644–650, doi: 10.1109/CICTN64563.2025.10932395.

[15] R. Menaka *et al.*, "Proof of stake based voting system using block chain," in *2024 13th Int. Conf. Syst. Model. Adv. Res. Trends (SMART)*, 2024, pp. 495–498, doi: 10.1109/SMART63812.2024.10882186.

[16] S. Tanwar, N. Gupta, P. Kumar, and Y.-C. Hu, "Implementation of blockchain-based e-voting system," *Multimed. Tools Appl.*, vol. 83, pp. 1449–1480, Jan. 2024, doi: 10.1007/S11042-023-15401-1.

[17] Hita, Djonni, Culita, and R. Yunis, "Pemanfaatan figma dalam perancangan user interface e-commerce," *Nusant. J. Pengabd. Kpd. Masy.*, vol. 4, no. 3, pp. 104–111, Aug. 2024, doi: 10.55606/nusantara.v4i3.3047.

[18] M. Nakash, "Agile software development: The experience of working in sprints," in *Proc. InSITE 2024: Informing Sci. Inf. Technol. Educ. Conf.*, 2024, pp. 1–7, doi: 10.28945/5252.

[19] D. Gunawan *et al.*, "Implementasi MERN stack pada pengembangan sistem penerimaan peserta didik baru," *SWABUMI*, vol. 11, no. 2, p. 102–110, Sep. 2023, doi: 10.31294/swabumi.v11i2.15965.

[20] D. Gunawan *et al.*, "Preserving individual privacy from inference attack in transaction data publishing," in *2023 8th Int. Conf. Inform. Comput. (ICIC)*, 2023, pp. 1–6, doi: 10.1109/ICIC60109.2023.10381942.

[21] J. Minango, M. Zambrano, and C. Minaya, "Exploring the use of blockchain for academic certificates: Development, testing, and deployment," in *Innov. Res. – Smart Technol. Syst.*, 2024, pp. 123–137, doi: 10.1007/978-3-031-63434-5_10.

[22] M. Sunny *et al.*, "Deploy—Web hosting using docker container," in *Adv. Comput. Netw. Commun.*, 2021, pp. 335–345, doi: 10.1007/978-981-33-6977-1_26.

[23] F.C. Hudi and C.M. Karyanti, "Pengujian black box testing pada sistem informasi assesment berbasis web di bidang pariwisata," *J. Ilm. Komputasi*, vol. 22, no. 4, pp. 553–560, Dec. 2023, doi: 10.32409/jikstik.22.4.3490.

[24] I. Akbar, "Penerapan sistem usability scale dalam pengukuran kebergunaan website SMKN 13 Bandung," *INTERNAL (Inf. Syst. J.)*, vol. 7, no. 1, pp. 1–7, Jun. 2024, doi: 10.32627/internal.v7i1.865.

[25] D.P. Bauer, "Unit tests for smart contracts," in *Getting Started with Ethereum*, Berkeley, CA, USA: Apress, 2022, pp. 49–53.

[26] S. Tanwar, *Blockchain Technology*. Singapore, Singapore: Springer 2022.

- [27] W. Cheah *et al.*, "Mobile technology in medicine: Development and validation of an adapted system usability scale (SUS) questionnaire and modified technology acceptance model (TAM) to evaluate user experience and acceptability of a mobile application in MRI safety screening," *Indian J. Radiol. Imaging*, vol. 33, no. 1, pp. 36–45, Jan. 2023, doi: 10.1055/S-0042-1758198.
- [28] B.D. Rahmawati, A.W.A. Wibowo, and S.N. Fitrianingrum, "System usability scale (SUS) as an analysis method for official website," *Telematika : Jurnal Informatika dan Teknologi Informasi*, vol. 21, no. 2, pp. 173–180, Jun. 2024, doi: 10.31315/TELEMATIKA.V21I2.12918.
- [29] C.N. Samuel, S. Glock, F. Verdier, and P. Guitton-Ouhamou, "Choice of ethereum clients for private blockchain: Assessment from proof of authority perspective," in *2021 IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, 2021, pp. 1–5, doi: 10.1109/ICBC51069.2021.9461085.
- [30] C. Fan, C. Lin, H. Khazaei, and P. Musilek, "Performance analysis of hyperledger besu in private blockchain," in *2022 IEEE Int. Conf. Decentralized Appl. Infrastruct. (DAPPS)*, 2022, pp. 64–73, doi: 10.1109/DAPPS55202.2022.00016.