# Implementation Decentralized Web Application on E-Voting System for Secure and Transparent Student Elections using Smart Contract

Hussain Abdillah Tugas Kelarno[1], Widi Widayat[2]

[1]Universitas Muhammadiyah Surakarta, Pabelan, Surakarta, Indonesia

[2]Universitas Muhammadiyah Surakarta, Pabelan, Surakarta, Indonesia

Email correspondence: l200214201@student.ums.ac.id

**Abstract**. Traditional paper-based voting system for student organization leaders election has issues related to security, transparency, and trust. This research aims to address 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 SDLC methodology and tested using Blackbox and 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 show that the decentralized e-voting system is capable of preventing vote manipulation and detecting duplicate voters, as evidenced by the unit testing of the smart contract, which confirmed that recorded votes cannot be manipulated and also attempts to submit multiple votes are 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 are 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 is in the Good category and it is 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*

## INTRODUCTION

Traditional paper-based election systems have long been used in Indonesia as the primary method for Presidential Election or Legislative Election. However, these systems have a number of significant weaknesses, particularly in relation to issues of security, data manipulation, transparency and public trust [1]. One of the major risks in traditional election is vote manipulation such as ballot forgery or vote inflation. These challenges also raise concerns regarding 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 loopholes can decrease public confidence in the integrity of the electoral process, which can affect the legitimacy of the election results themselves [2].

Along with the development of technology, new approaches have emerged that can overcome problems in traditional election systems, one of which is through the use of blockchain technology and smart contracts. Blockchain is a decentralized technology that allows every transaction or recorded data to be immutable and publicly verifiable [3]. 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 [4]. 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 by Jafar et al. [5] underscored blockchain's benefits in verification and cost reduction but stopped short of developing a practical, implementable system. Alfain et al. [6] progressed by developing a decentralized application on Ethereum, proving its advantages in security and data integrity, yet their system encountered significant performance issues. Similarly, the work by Rahi et al. [7] was purely theoretical. Although the literature review confirmed improvements in security, it did not address practical aspects such as scalability, performance, or user experience (UX). More practical implementation by Abdulah and Adnan [8] resulted in a secure system on a local network, but its focus was predominantly technical, paying insufficient attention to UX. 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.

The specific objective of this research is to develop and implement a secure, transparent, and reliable blockchain based decentralized e-voting system, using student elections as a case study. The methodology focuses on two key aspects, leveraging smart contracts to automate the voting processes to enhance efficiency and minimize human error, and prioritizing user experience (UX) to ensure the system is highly accessible. This approach is 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 will also be tested in small-scale simulations, providing a more realistic picture of its performance and effectiveness in real elections.

## METHOD

In implementing this system, the author uses the SDLC method using the Agile method. The Agile method was chosen because Agile SDLC enhances flexibility, responsiveness to changing requirements, and user involvement, leading to improved software quality and timely delivery through iterative development and collaboration among developer [9]. The Agile model consists of 6 stages which can be seen in Figure 1.

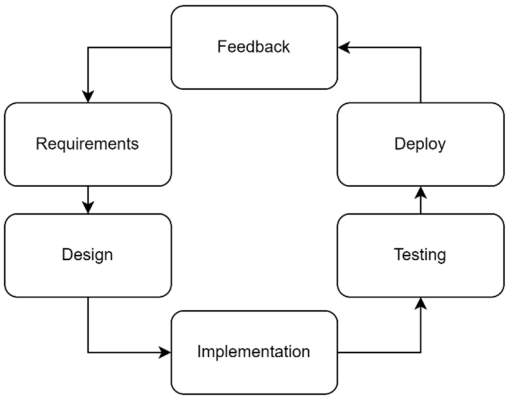


**Figure 1.** Stages of Agile Methodology

### 2.1 Requirements

In the Requirements stage, system requirements are obtained through a literature study process. Literature studies are conducted by analyzing various literature and research related to e-voting, blockchain, and smart contract. 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 [10]. After conducting a literature study, the requirements to develop this system were collected. There are functional requirements and non-functional requirements.

### 2.1.1 Functional Requirements

The functional requirements of this blockchain-based e-voting system include 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 can participate in the election. Once registered, voters must be able to cast their votes anonymously, where the votes will be 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 can be calculated and published automatically without manual intervention. The system needs to display the election results after all votes are counted.

### 2.1.2 Non-Functional Requirements

The non-functional requirements of this system include 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 [11]. The system must be able to protect voter data and protect votes from manipulation by other parties. Other aspects such as performance and user experience are also important so that users can easily access and use this system. In addition, hardware requirements such as PCs, laptops or smartphones and web browsers and internet access are needed to use this system.

## 2.2 Design

The design aims to design the entire information system architecture and creating blueprint for the system, to ensure the system is easy to use by voters. At this stage also provides an overview of how the appearance and flow of the information system to be created [12]. The system design stage consists of use case diagrams, activity diagrams and database ERD design.

### 2.2.1 Use Case Diagram

Use case diagram is a visual representation that outlines the interactions between actors and a system, showcasing the system's functionalities [13]. In this system there are two actors, admin and user. Admin is an actor who manages and organizes users, also managing this system by starting and ending the voting event. Admin is also tasked with managing and maintaining smart contracts and blockchains used to recap votes. User is an actor who uses the system as a, he can 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 the figure 2.

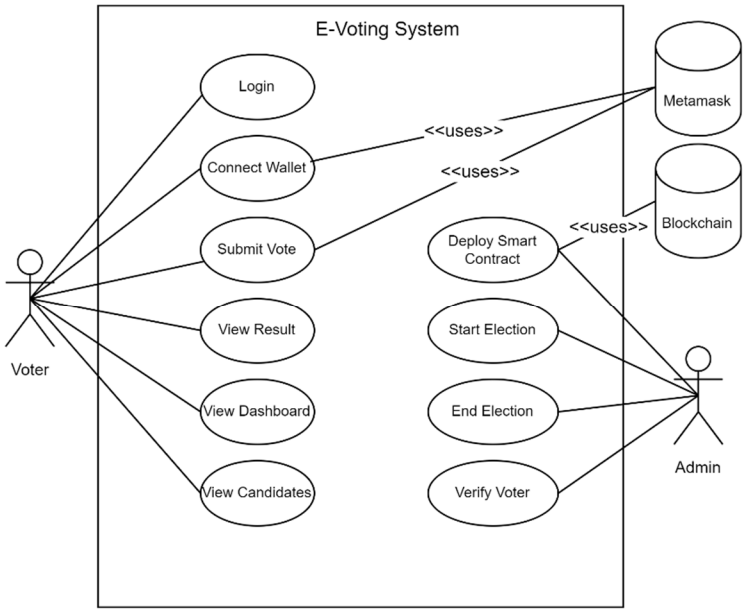


**Figure 2.** Use Case Diagram of E-Voting System

### 2.2.2 Activity Diagram

An activity diagram represents workflows of stepwise activities and actions, showcasing the dynamic aspects of a system, particularly useful in modeling processes creating design system [14]. In this research, there is a workflow carried out by the Admin and User. Figure 3 shows an activity diagram of user 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 is required to connect their wallet address to this system in order to use this system. After that, the user can vote and submit their vote and confirm it using Metamask, after that the vote will be recorded in the smart contract. After the election is ended, the user can see the results in the result menu.

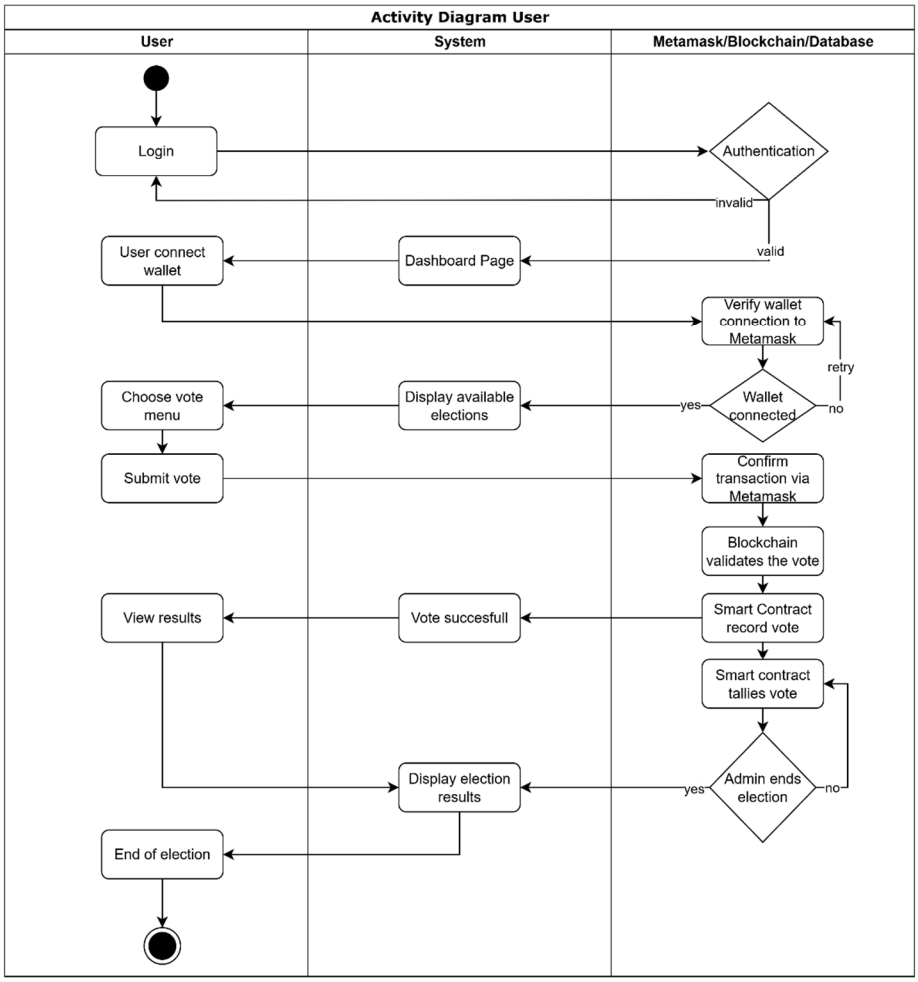


**Figure 3.** Activity Diagram User

Figure 4 shows the admin activity when managing the election. When the admin authentication process is successful, the admin can do several things such as starting the election, managing users, and publishing the election results. In this activity, the admin starts the election and the system will automatically deploy a smart contract that will be used as a vote recorder. After that, the admin can add candidates and monitor the ongoing election. After the election is complete, the admin ends the election and publishes the results of the vote that carried out by the smart contract.

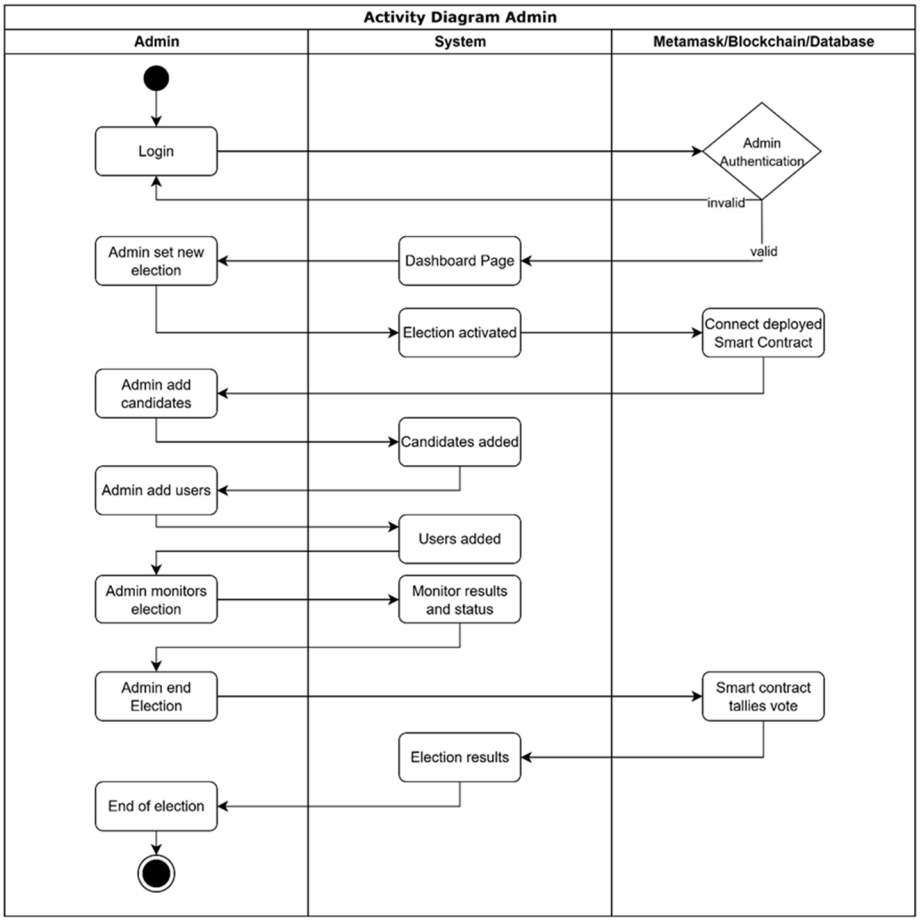


**Figure 4.** Activity Diagram Admin

### 2.2.3 Entity Relationship Diagram

Entity Relationship Diagram (ERD) is a tool in database design that visually represents data entities, attributes, and relationships, reflecting user data requirements [15]. In the e-voting system, there are several relationships. Admin manages all ongoing elections by adding candidates, starting, and ending elections, while smart contract is deployed by election when admin starts an election. Election represents an election containing several candidates. Voters cast their votes through the vote entity, which associates voters with elections and the selected candidates. The ERD for this e-voting system can be seen in Figure 5.

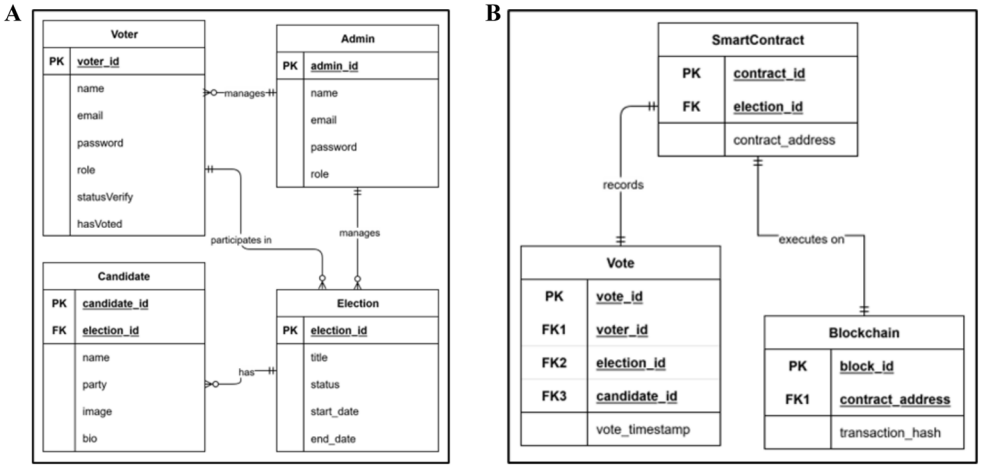


**Figure 5.** Entity Relationship Diagram. (A) Database ERD, and (B) Blockchain ERD

### 2.2.4 Architecture Diagram

Architecture design diagrams are an essential tool for visualizing the structure and components of a system. They facilitate understanding by representing the relationships and interactions between elements [16]. The architecture of this e-voting system uses Metamask for user verification and smart contract signing. Users are required to connect their web 3 wallet with the E-Voting Application using Metamask. With the ethers.js library, the frontend application can connect to the Metamask. The admin dashboard uses a Firebase and PostgreSQL database to manage users and elections by connecting to database using the ORM (Object-Relational Mapping). After the user selects a candidate, the voting data is sent from the web application to the blockchain, where the votes are recorded using a smart contract and validated by the Sepolia network. The architecture design for this e-voting system can be seen in Figure 6.

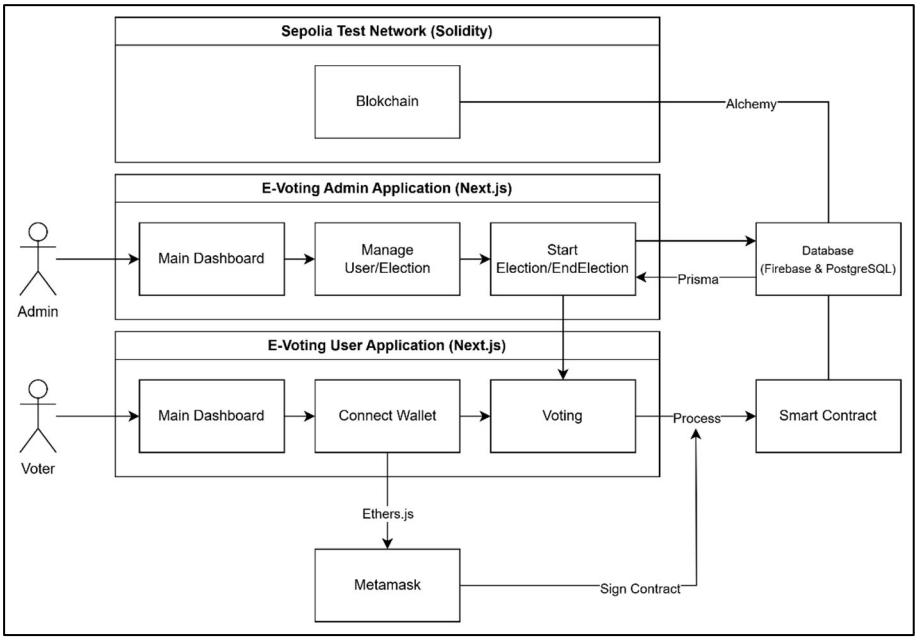


**Figure 6.** Architectural Design of E-voting System

### 2.2.5 User Interface Design

User interface (UI) 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].

## 2.3 Implementation

Implementation phase begins with the implementation of code and systems based on the design that has been made. Development is carried out iteratively in several sprints, where each sprint focuses on developing a specific part of the system [18]. MERN stack is popular and reliable because this architecture is simple but also strong from the back-end to the front-end [19]. But in implementation, author uses the Typescript programming language and using the Next.js framework because it is 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. Using Blockchain technology to store voting data, uses smart contract technology from Solidity and the Ethers.js library to interact with the Ethereum blockchain and smart contracts using Sepolia network. For wallet address management using third party extensions from Metamask so that users can access decentralized Apps directly from their browser and make transactions on the Ethereum blockchain. As well as the PostgreSQL and Firebase database to store other information that is not stored on the blockchain. Hardhat used for testing smart contracts

environment locally. The reason why every transaction vote needs to be stored on the Ethereum blockchain and not stored in a conventional database is because every vote transaction stores sensitive information. If transaction data is stored in a regular database, it is vulnerable to hacking and data leaks [20]. Therefore, actions to protect vote transaction data are more important to store on the blockchain than to store in a database.

### 2.4 Testing

Testing stage in Agile is an important stage that is closely integrated with the development process to improve software quality and to ensure software meets business and user needs through iterative and incremental processes [21]. The testing used to test this system are using several methods, black-box testing, system usability scale (SUS), unit testing on smart contract, and verification on block explorer. Black-box testing is used to evaluate e-voting systems functionality by examining its outputs in response to various inputs, without knowledge of the internal code, ensuring user requirements are met [22]. By testing several cases such as login authentication testing, account access, and registered voters who have access and are verified who can vote using the Black-box method. To test whether this system run according to the functional requirements determined at the beginning, the entire system will be tested using the System Usability Scale (SUS) method.

This method used because it is a reliable tool for measuring usability, even with small sample sizes, providing a score to evaluate user satisfaction and experience [23]. To test the security of the system on a smart contract, it is using the unit test method. Unit testing is used to test the security of the smart contract function. Several test cases that can be tested such as whether users can double vote or not and ensuring that votes cannot be changed or modified after being recorded. This case can be tested using unit testing because the advantage that can ensure every function in the smart contract is in accordance with the specified case specifications. By using unit testing, developers can detect logical errors at the most basic level of the smart contract code [24]. Verification on block explorer is used to audit by ensuring that this system is transparent and secure by providing a verification for each voting transaction on a block explorer. This method facilitates audits without violating voter privacy. This method is conducted to prove that the implemented e-voting system is transparent and can be publicly audited in accordance with the blockchain principles, public and immutable [25].

### 2.5 Deploy

The deployment phase is carried out after the system has been tested and found to meet all the criteria. The e-voting system is launched and simulated on a small scale, such as the election of student organization, to evaluate its performance directly. The system is launched in a controlled environment, where the entire process of voting, counting votes,

and publishing results is thoroughly tested. The results of this simulation are used to identify areas of improvement and make further changes if necessary [26].

### 2.6 Feedback

Feedback phase purpose to collect all the feedback from end users. Covering system performance, security, stability, and user experience. After getting feedback from users, if there are any additional features or bug fixes, it must be done with the project backlog will be updated and must be completed in the next sprint [27]. This continuous evaluation ensures that the e-voting system can run well and effectively in helping students held election events. By implementing this stage, developers can make updates and fixes needed to improve the user experience.

## RESULT

### 3.1 Result of the System

The e-voting system has been developed according to the designed method. As a result, the system can record votes from users using their wallet addresses and the system successfully records the voting results using smart contracts deployed on the Sepolia network. The following are the results of an e-voting system using blockchain.

#### 3.1.1 Login Page

Login page is the initial display of this e-voting website, Users/Voters are required to login before using the system. Users login with their credentials. If the user does not have an account, the user can create account first on the signup page provided. The login page can be seen in Figure 6.

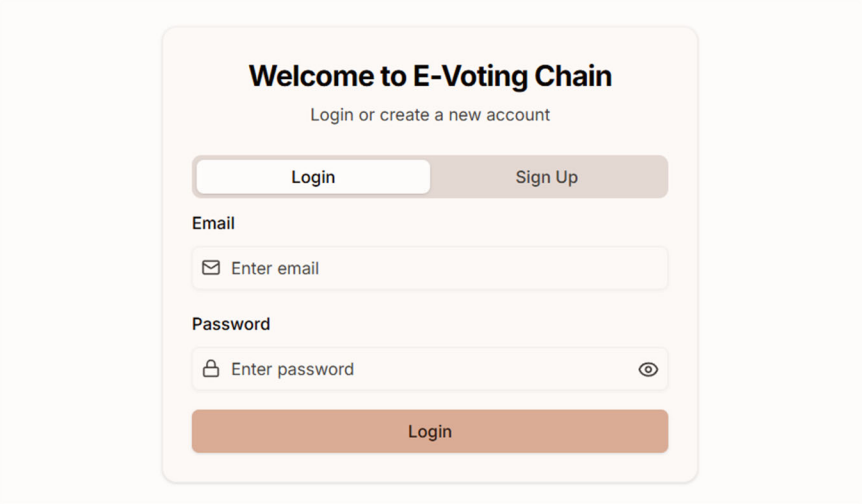


**Figure 6.** Login Page

### 3.1.2 Landing Page

After the user successfully logs into the system, this landing page is the first page that will be displayed to the user. This page displays some important information related to the election, such as the election countdown, access guide how to use the system, and access for users if they want to ask admin for a help regarding process verification or voting process. The display of the landing page is in Figure 7.

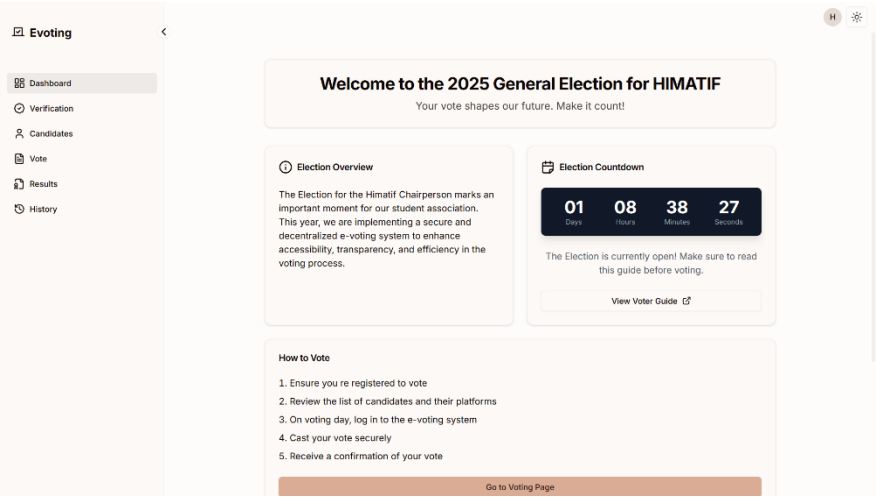


**Figure 7.** Landing Page Voter

### 3.1.3 Verification Voter Page

Users must verify their identity that they are eligible voters. On this verification page, users are required to verify by providing their student ID as their identifier. Details for user verification using student ID are shown in Figure 17.

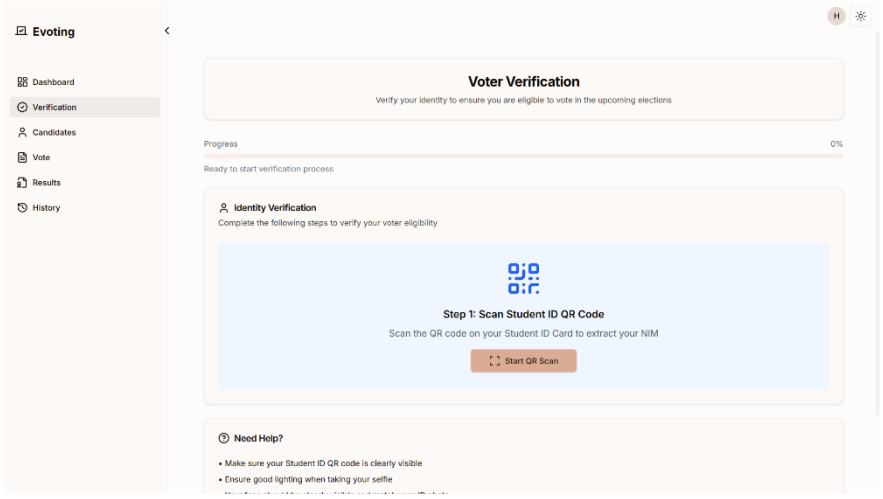


**Figure 8.** Verification Voter Page

### 3.1.4 Candidates Information Page

Before users voting for candidates, users can learn about the available candidates information and their vision and mission. The candidates information can be seen in Figure 9.

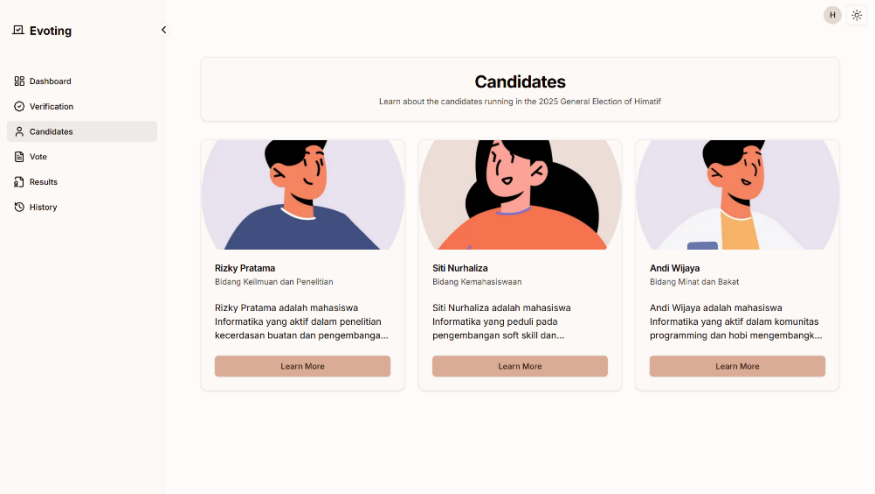


**Figure 9.** Candidates Information Page

### 3.1.5 Vote Page

Voting page displays all available candidates. Users can directly vote and confirm their choice using the Metamask wallet that user created and connected to the website. This is necessary because it is used to interact and send the vote to the smart contract. Users are required to confirm the process that appears in the Metamask wallet to send a vote, and a Sepolia token is needed which is used for network fee transactions. The process of user confirmation of voting process can be seen in Figure 10.

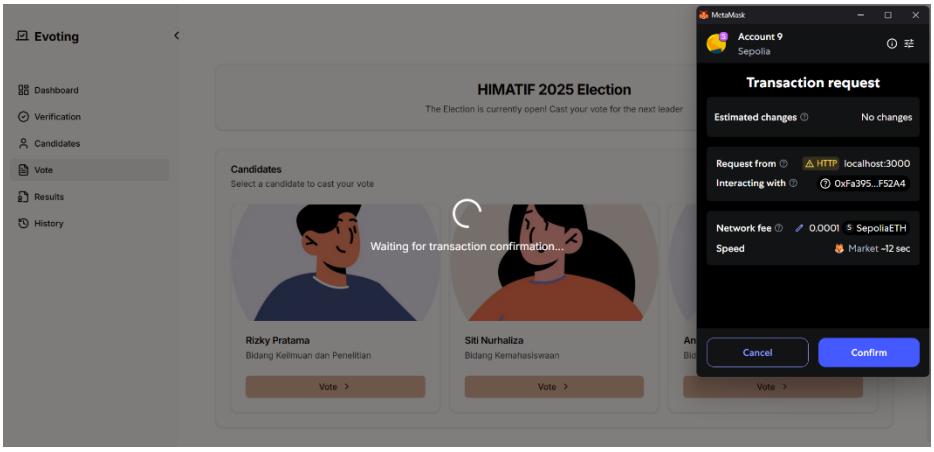


**Figure 10.** Voting Process

After confirming the transaction using Metamask, the smart contract will validate the transaction. If the process is successful, the user will get a transaction hash, displayed on the receipt that user receives on the website. The receipt that user receives can be seen in Figure 11.

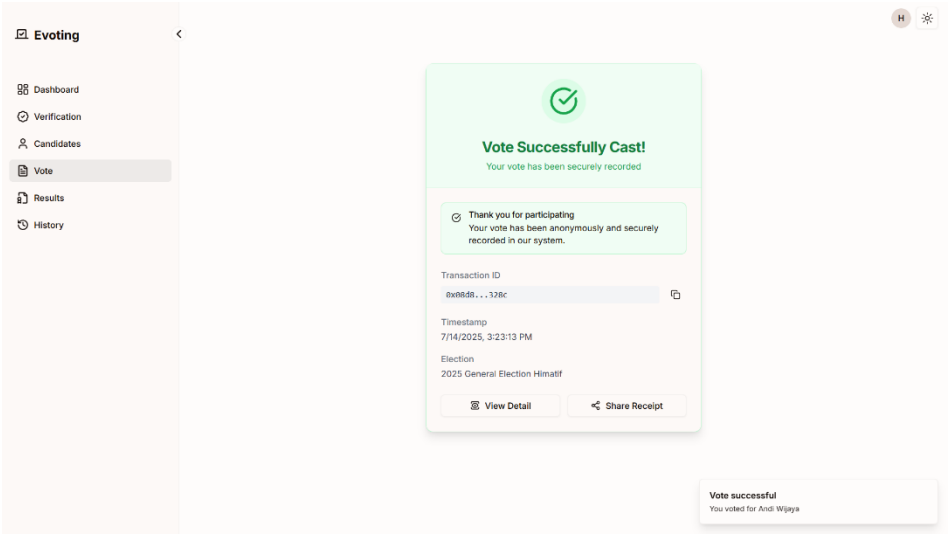


**Figure 11.** Voting Success

### 3.1.6 Election Results Page

After user successfully finish the voting process, the user can see the live results of the ongoing election. This results data is taken directly from the blockchain, so if there are any additional votes it will automatically change. Details of the election results page can be seen in Figure 12.

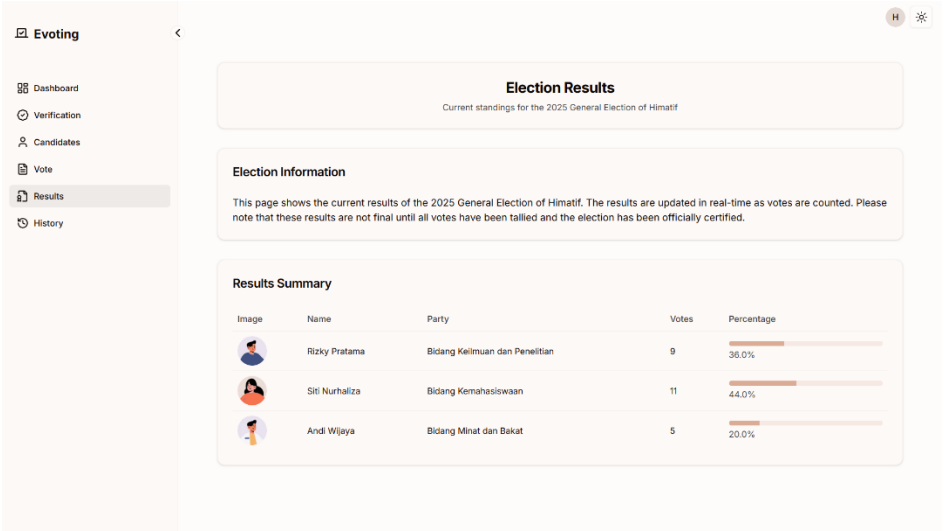


**Figure 12.** Election Results Page

### 3.1.7 Election History Page

User can see the history results of the past election. This history data is taken from the blockchain that admin use in every election, so if there are any election that already done the result will appear in this page. Details of the election history page can be seen in Figure 13.

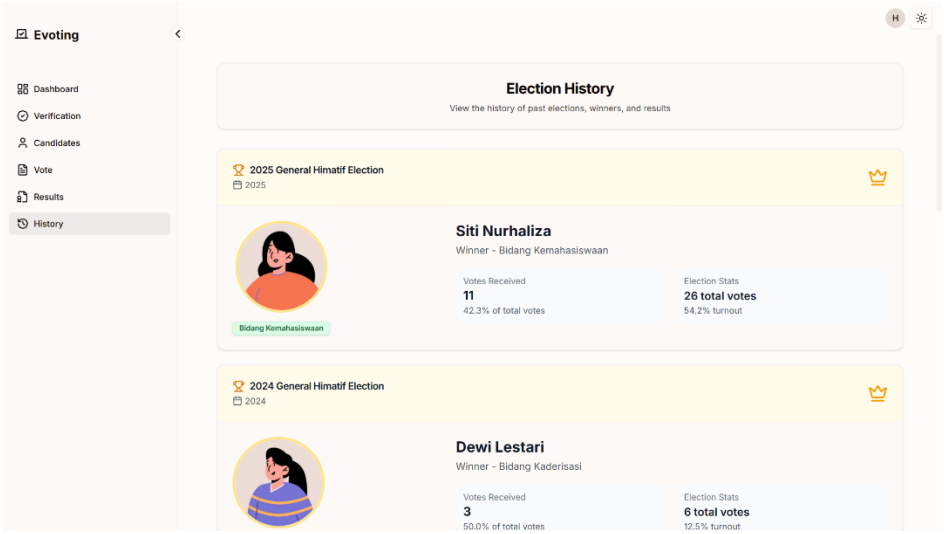


**Figure 13.** Election History Page

### 3.1.8 Admin Dashboard Page

Admin dashboard page begins with a display of some information related to the ongoing election. This page contains information about the total voters, the number of voters/users using this system. Then votes cast is the number of votes that have been recorded into the smart contract. Election status, is the status of the ongoing election, also contains the period time of the election. Then information about smart contract status and contract address that used to store voting data. Details of the admin page can be seen in Figure 14.

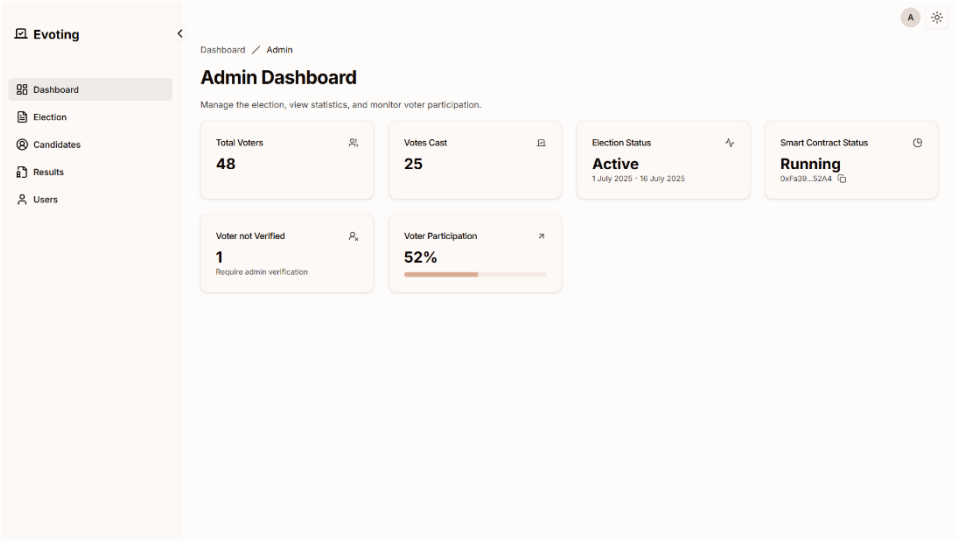


**Figure 14.** Admin Page

### 3.1.9 Election Management Page

To manage the ongoing election, the admin can set the start and end dates of the election. The status will be automatically active if the today's date is in the start and end

time period. If today's date is outside the specified period, the election will be automatically inactive and the user cannot make a voting. Details of the election management page are in Figure 15.

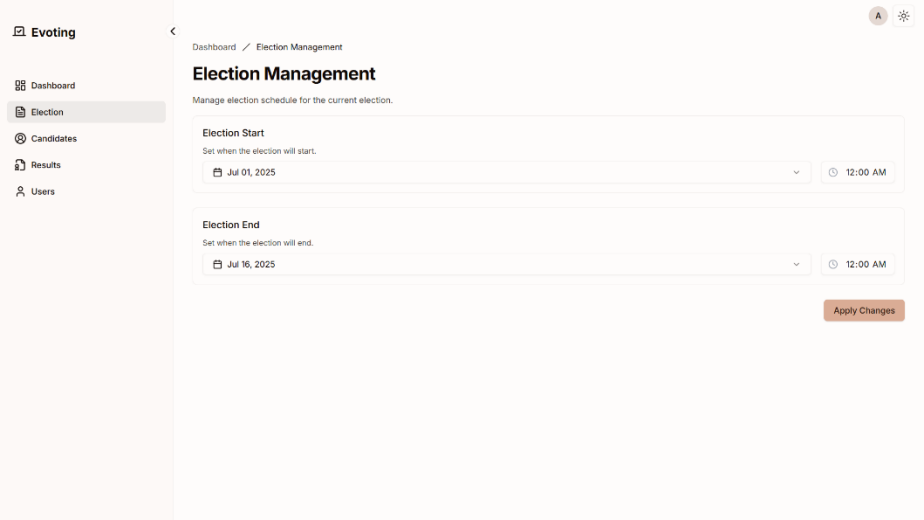


**Figure 15.** Election Management Page

### 3.1.10 Candidates Management Page

Candidates management page is used to manage candidates, there are several features such as add candidates, edit candidates, and delete candidates. To add candidates, the admin can press the add candidates button, and pop-up form will appear. The admin can fill in information related to the name, party, image and description of each candidate. The appearance of the candidates page can be seen in Figure 16 and the appearance of the add candidate form is in Figure 17.

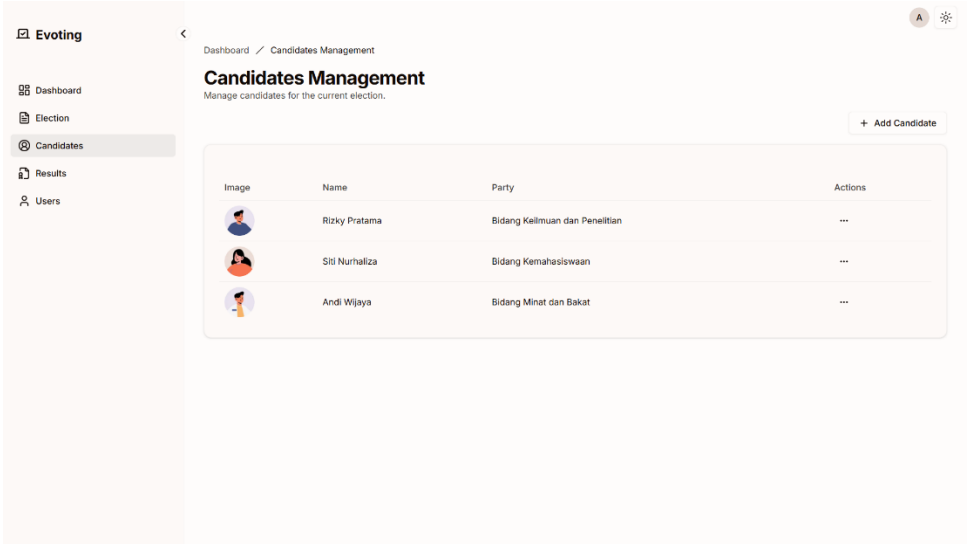


**Figure 16.** Candidates Management Page

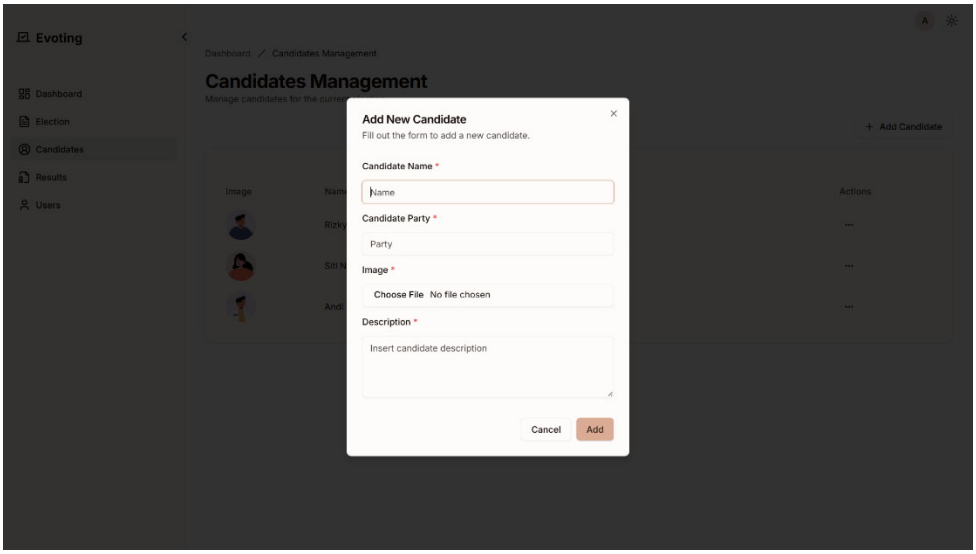


**Figure 17.** Add Candidates Form

### 3.1.11 Users Management Page

On the user management page, the admin can perform several activities. Such as adding user accounts by adding email and password directly. So that users can directly log in using the credentials provided by the admin. Then another feature is to verify user. The admin can change the user status, where users who have verified their accounts can perform the voting process. While those who have not verified, cannot make the voting process. The details of user management can be seen in Figure 18 and form to add users can be seen in Figure 19.

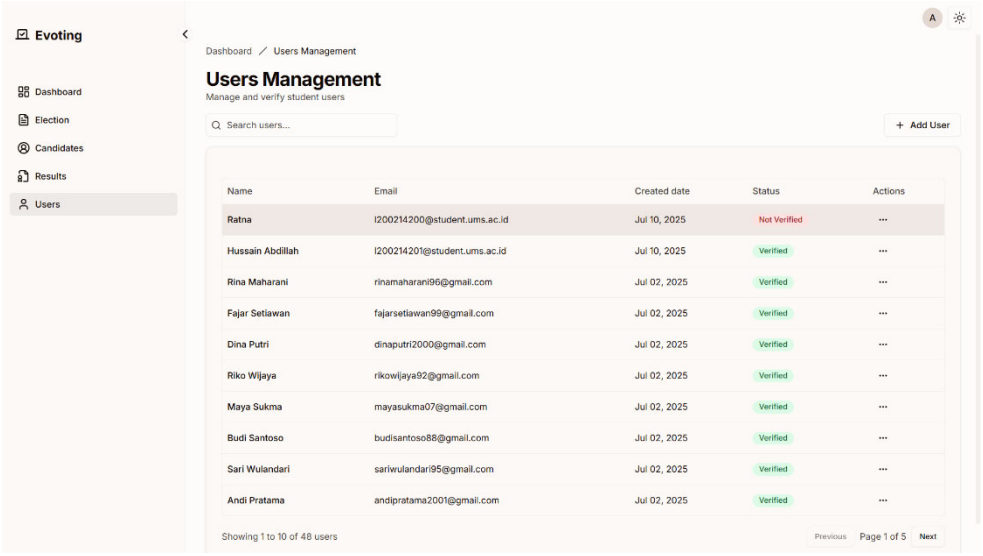


**Figure 18.** Users Management Page

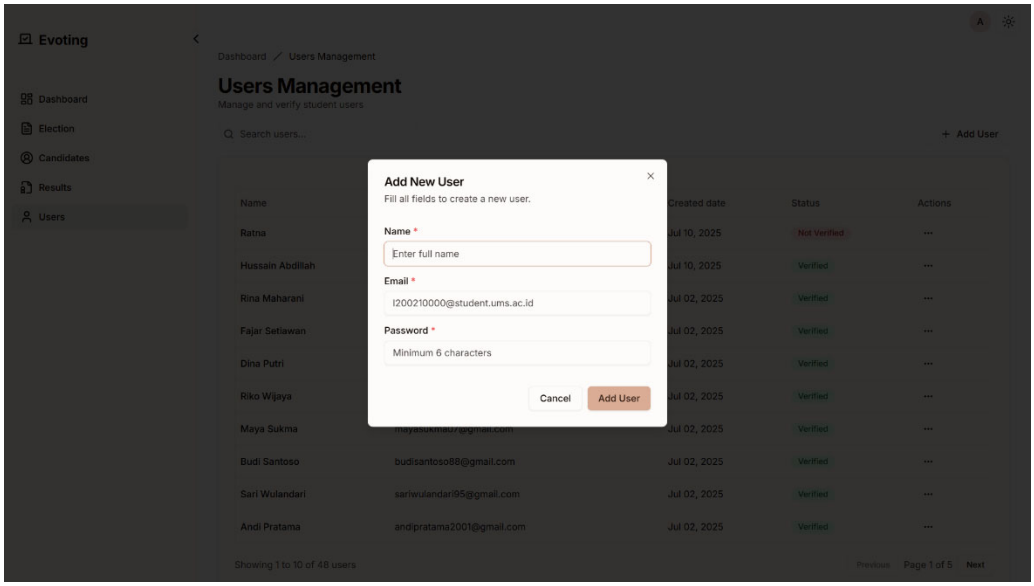


**Figure 19.** Add Users Form

## 3.2 Testing and Evaluate System

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. This section presents a detailed results of the system's performance and outcomes.

### 3.2.1 Black Box Testing

To test the system functionality and to make sure the system is running properly, black box testing method is used. Of the 22 test cases performed, all test cases ran according to expected results. Below are the results of Black box testing on this e-voting system. Details of the results can be seen in table 1.

**Table 1.** Black Box Testing

| Test Case | Input | Expected Results | Status |
|---|---|---|---|
| Login user | Enter a valid user email and password | Enter the user page | Valid |
| Login admin | Enter a valid admin email and password | Successfully enter to the admin dashboard page | Valid |
| Login with invalid credentials | Enter invalid email and password | Displaying error message wrong credential | Valid |

| | | | |
|---|---|---|---|
| User sign up | Enter valid name, email and password | Sign up successful and user created | Valid |
| User sign up same data | Inputting user data that has been created previously | Display error message user already exists | Valid |
| Password not match | Enter a different password when confirming the password | Display the message password does not match | Valid |
| User Voting | The user presses the vote button and confirms the transaction via Metamask | Voting is successfully recorded in the smart contract, and displays a success receipt | Valid |
| User voting without wallet | User presses the vote button but not connected his wallet to the website | Displaying message voting failed please try again | Valid |
| User voting but cancel the transaction | User presses vote button but cancels the transaction in Metamask | Displaying message voting failed has rejected | Valid |
| User already vote, but wants to vote again | Users who have voted accessing again the voting | Display message you have already voted | Valid |
| User not verified | Unverified users access the voting page | Display message your account is not verified and cannot continue for voting | Valid |
| The election has not started / has already ended | User accesses the vote page when the election is finished / has not started | Display message that the election has not started yet / the election is finished | Valid |
| Manage election | Admin sets the election schedule by entering the start and end dates and then pressing apply button. | The system successfully saved the schedule and display message that the | Valid |

| | | | |
|---|---|---|---|
| | | schedule had been successfully updated | |
| Adding new candidate | Admin inputs data to add candidates such as name, party, image, and description. | The system successfully added the new candidate. | Valid |
| Adding incomplete data candidate | Admin did not complete fill the data in the add candidate form | Displaying message all fields are required | Valid |
| Edit data candidate | Admin edits the form data and presses the save button | The system successfully saved the edited candidates data. | Valid |
| Delete data candidate | Admin presses the delete button and confirms in the dialog | The system successfully deleted candidate data | Valid |
| Vote results | Admin enters the results menu | The system successfully displays the voting results from the smart contract. | Valid |
| Add new user | Admin adds and inputs new user data such as name, email, and password | User account successfully created | Valid |
| Adding incomplete data user | Admin did not complete the add user form data | Display message all fields are required | Valid |
| User verification | Admin presses the verify / unverify user button | The system successfully saved the new user status. | Valid |
| User / admin perform logout | User / admin presses the logout button | Session successfully logged out and | Valid |

| | | | |
|---|---|---|---|
| | | returned to the login page | |

### 3.2.2 System Usability Scale Testing

System usability testing is done to assess usability of the system by testing it directly by users and admin. Usability test is done by giving 10 questions and then measured using a Likert scale. Testing was conducted by taking respondents from students who participated in student organization activities around universities, with a total of 33 respondents. A total of 28 students tested it as users and 5 students tested it as admin. The score is categorized into five levels of usability based on the following score ranges. Scores of 80.3 and above are 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 results of the SUS testing can be seen in Figure 20.

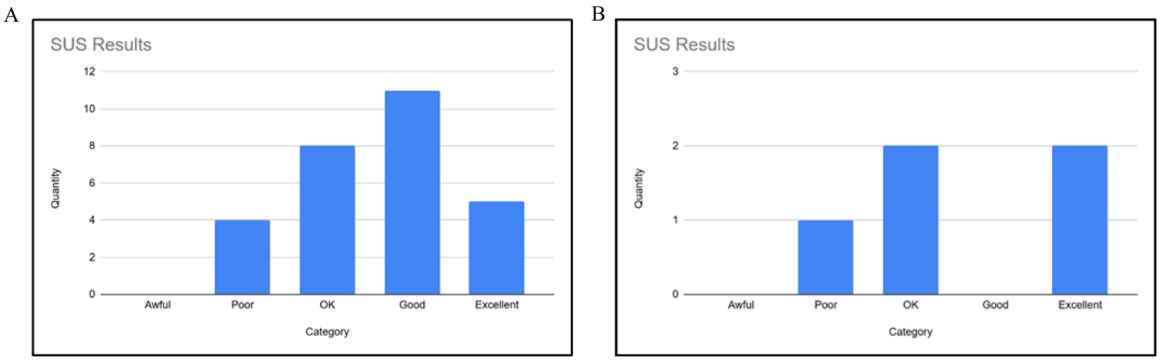


**Figure 20.** SUS Testing Results. (A) User Results, and (B) Admin Results

Based on the results of the SUS testing conducted, for each test carried out on the user and admin roles, the results shown were almost similar. User respondent show that the average score obtained is 71, and for admin respondent the average score obtained is 74. Each scores indicates that the system is at a Good category according to the score interpretation displayed in Figure 21.

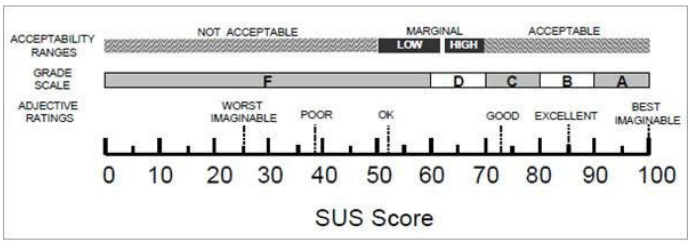


**Figure 21.** SUS Score Interpretation

### 3.2.3 Smart Contract Unit Testing

To test the security, functionality and properties of the smart contract, the testing system is carried out using the unit testing method [24]. Some cases tested from the smart contract that listed in the following test scenarios in Table 2.

**Table 2.** Unit Test Scenario

| Test Case | Input | Expected Results | Status |
|---|---|---|---|
| User successfully voted to smart contract | User vote to candidateId = 1 using vote () function | Votes for candidate with id=1 is 1 | Valid |
| User cannot vote more than one. | Calling vote () function twice by 1 account | Second transaction failed with error | Valid |
| Accumulate the corresponding vote counts from multiple voters | Multiple accounts vote for a candidate, and call the getVotes () function. | The number of votes corresponds to the number of accounts that voted | Valid |
| Checking whether the user has voted | Checking user wallet address | True if user have voted, False if user have not | Valid |
| User who has voted, overriding and modifying votes. | Overriding vote () with different choice of candidate | The previously vote remains and does not change. | Valid |
| Checking whether the admin has the ability to manipulate the voting results directly | Admin overrides vote () directly in smart contract | Error, it can't be done, because there is no function to change vote in the smart contract. | Valid |

Unit testing is carried out by using code script that running in the hardhat project environment, containing the solidity smart contract code and the unit test script code. The

following are detailed results from the solidity smart contract unit testing by running it through the script code which can be seen in Figure 22.

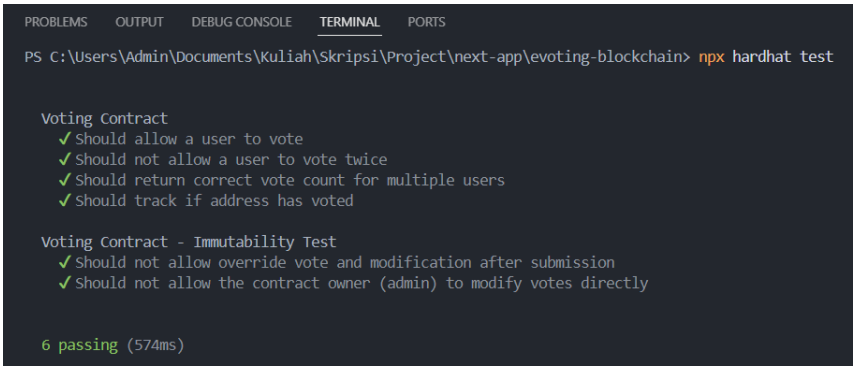


**Figure 22.** Unit Testing Code Results

The test results show that all test scenarios from the unit testing were successful and in accordance with the expectations. With a result of 6 passing with a test duration of 574 milliseconds.

### 3.2.4 Verification on Block Explorer

Verification on block explorer is testing method that directly verifying the smart contract that has been deployed through the sepolia block explorer. The results show that all voting transactions carried out in the e-voting system are well audited and can be accessed publicly. Users can see detailed information such as transaction hash, status, timestamp, wallet address, smart contract address and network fee. The following are details of the information if the user finds the transaction hash obtained after successfully voting, the details can be seen in Figure 23.

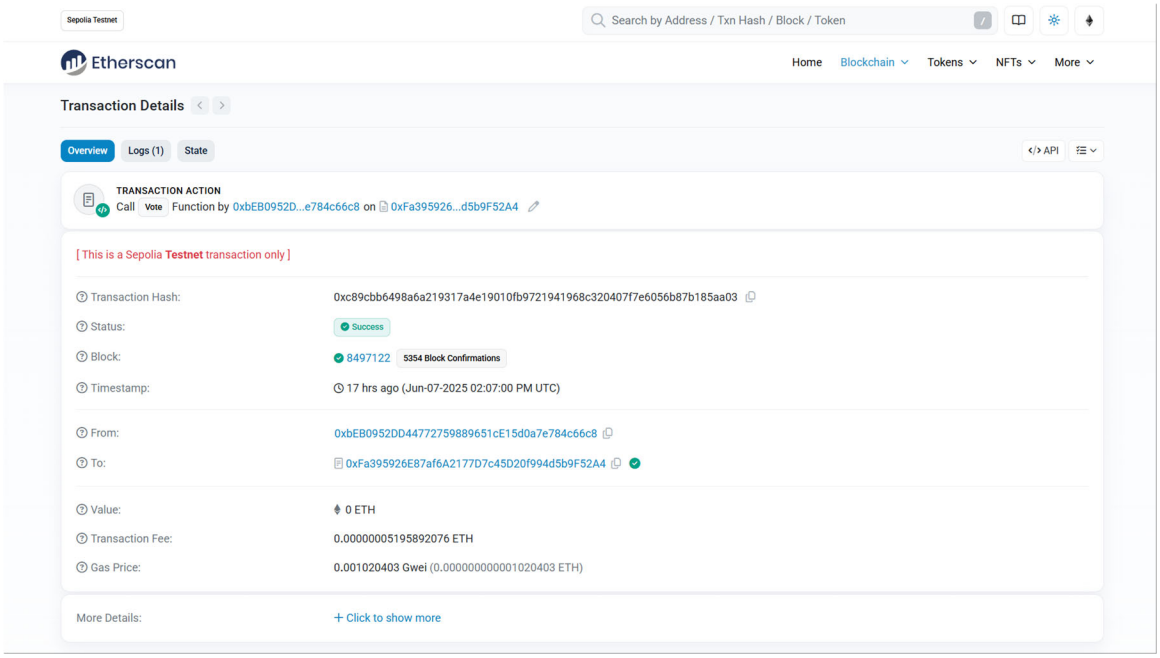


**Figure 23.** Detail from Block Explorer

All voting transactions recorded by the smart contract, although users can see the transaction history, users cannot know who and which candidate that other users choose because the anonymity. Where users can only see the wallet address that sent the transaction without knowing who is behind the address. Detailed information from the smart contract history can be seen in Figure 24.

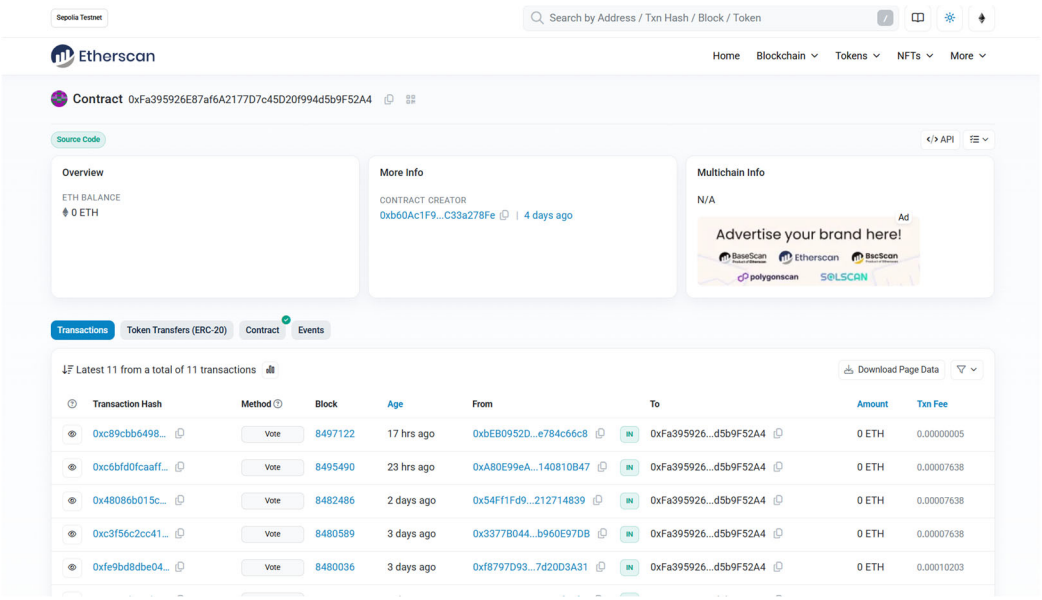


**Figure 24.** History Detail from Block Explorer

The solidity contract code deployed to the smart contract can be checked to find out if there is any manipulation attempt by the admin/organizer regarding the vote collection/counting process. All codes can be accessed publicly and verified by the system from the sepolia block explorer. Details of the smart contract code that deployed to the network can be seen in Figure 25.

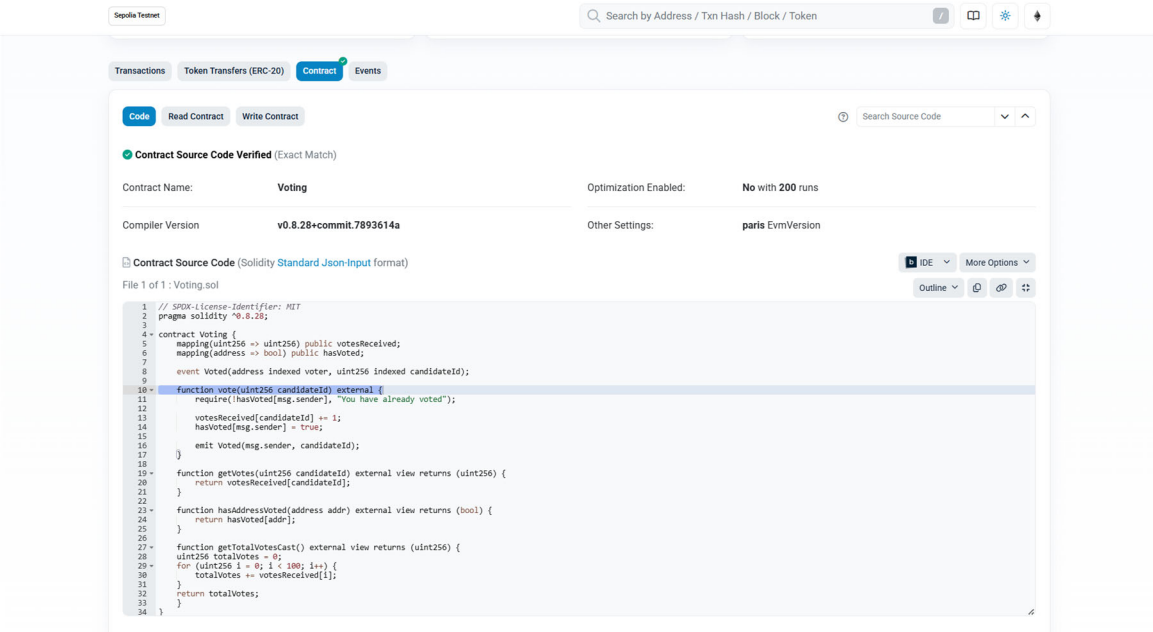


**Figure 25.** Smart Contract Source Code

## 4. DISCUSSION

This chapter interprets the findings presented in the previous chapter to evaluate how the developed blockchain e-voting system achieves its objectives of enhancing security, transparency, and trust. The results are discussed in the context of existing literature and the fundamental principles of blockchain technology

### 4.1 Functionality and Reliability

The success of the black box testing demonstrates that the e-voting system functions correctly from an end-user 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 [22].

### 4.2 Usability Evaluation

The SUS score from user respondent show that the average score obtained is 71, and for admin respondent the average score obtained is 74 suggests that the system is a good

level of usability and can be accepted by users and admin. 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.

### 4.3 Transparency and Trust

The contract verification via Sepolia block explorer contributes significantly to the transparency of the system. Users and auditors can inspect the logic behind the voting process, increasing trust in the system's integrity. This addresses one of the core objectives of the study providing an auditable and tamper-proof election mechanism [3]. The unit testing results confirm 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. This can be concluded that the smart contract is proven to be secure and immutable, because voters who have made transactions/votes cannot add or even manipulate data that has been confirmed by the smart contract. This finding supports the argument previous research by Rahi et al. [7] that the immutability of data is a key for secure digital voting.

### 4.4 Contribution to E-voting Security

By implementing blockchain and smart contract technology, the system offers several security enhancements over traditional web-based e-voting systems. Data immutability, decentralized verification, and wallet-based identity reduce the risk of tampering, fraud, or central control abuse. 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 [25]. This also supporting the security and trust in e-voting principles.

### 4.5 Implications, Limitations, and Future Research

The findings demonstrate that blockchain technology is effective solution for building secure and transparent e-voting systems. However, this study has limitations. It was deployed on a 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. Additionally, usability and trust case should be conducted with a more diverse demographic to ensure the system is accessible and trustworthy for general citizens.

## 5. 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 Blackbox testing, System Usability Scale (SUS), smart contract unit testing, and verification through a block explorer on the Sepolia network. The results of the Blackbox 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.

## 6. ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Allah SWT for blessings and grace in completing this thesis. I am incredibly grateful 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 program, for their invaluable guidance and support. My sincerest thanks also go to my beloved family and friends for their unwavering support, endless encouragement, and constant motivation. This accomplishment would not have been possible without all of you.

## REFERENCES

[1] Zamhasari, “Dampak Pemilihan Kepala Daerah (Pilkada) Terhadap Demokrasi: Tinjauan Kelebihan dan Kekurangan Pilkada Serentak di Indonesia Tahun 2024,” *Jurnal Pendidikan Dasar dan Sosial Humaniora* , vol. 3, no. 10, pp. 873–880, Aug. 2024, Accessed: Oct. 04, 2024. [Online]. Available: https://www.bajangjournal.com/index.php/JPDSH/article/view/8470

[2] D. Rahmani, “Tantangan dan Harmoni Antara Keadilan dan Kepastian Dalam Pemilu Serentak di Indonesia,” *JURNAL SULTAN: Riset Hukum Tata Negara*, vol. 2, no. 1, pp. 54–60, Oct. 2023, doi: 10.35905/SULTANHTN.V2I1.5620.

[3] Y. Chen and C. Bellavitis, "Blockchain disruption and decentralized finance: The rise of decentralized business models," *Journal of Business Venturing Insights*, vol. 13, p. e00151, Jun. 2020, doi: 10.1016/J.JBVI.2019.E00151.

[4] S. Khan, F. Loukil, C. Guegan, E. Benkhelifa, and A. Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer Peer Netw Appl*, vol. 14, no. 5, pp. 2901–2925, Sep. 2021, doi: 10.1007/s12083-021-01127-0.

[5] U. Jafar, M. Aziz, and Z. Shukur, "Blockchain for Electronic Voting System—Review and Open Research Challenges," *Sensors*, vol. 21, no. 17, 2021, doi: 10.3390/s21175874.

[6] Z. Alfain, H. Setiawan, and I. Buana, "Analysis of Centralized vs Decentralized Electronic Voting," in *2022 IEEE 8th Information Technology International Seminar (ITIS)*, Oct. 2022, pp. 173–177. doi: 10.1109/ITIS57155.2022.10010100.

[7] P. Rahi, A. Singh, P. Badoni, I. Singh, and R. Khan, "Use of Blockchain Technology in Electronic Voting Systems: An Overview from Computer Security," in *2023 International Conference on Communication, Security and Artificial Intelligence (ICCSAI)*, 2023, pp. 1–5. doi: 10.1109/ICCSAI59793.2023.10420927.

[8] W. Abdulah and S. Adnan, "Blockchain based Electronic Voting System Design with Smart Contracts," in *2023 IEEE Symposium on Computers & Informatics (ISCI)*, 2023, pp. 98–103. doi: 10.1109/ISCI58771.2023.10391913.

[9] S. Tetteh, "Empirical Study of Agile Software Development Methodologies: A Comparative Analysis," *Asian Journal of Research in Computer Science*, vol. 17, no. 5, pp. 30–42, Feb. 2024, doi: 10.9734/ajrcos/2024/v17i5436.

[10] K. Asad and M. Muqeem, "A Critical Analysis of Requirement Management in Agile Development," *Lecture Notes in Networks and Systems*, vol. 522, pp. 79–93, 2023, doi: 10.1007/978-981-19-5292-0_8.

[11] D. Priyawati, S. Rokhmah, and I. Utomo, "Website Vulnerability Testing and Analysis of Internet Management Information System Using OWASP," *International Journal of Computer and Information System (IJCIS)*, vol. 06, no. 03, pp. 143–147, 2022, [Online]. Available: https://ijcis.net/index.php/ijcis/index

[12] Wisnumurti, Y. Trimarsiah, and S. Faulina, "Penerapan Agile Development Methodology pada Sistem Informasi Penjualan Ecer dan Grosir Toko Kinanti Martapura," *JUTIM (Jurnal Teknik Informatika Musirawas)*, vol. 7, no. 2, pp. 109–120, Dec. 2022, doi: 10.32767/JUTIM.V7I2.1727.

[13] B. Hnatkowska and P. Zabawa, "A reusability-oriented use-case model specification language," *Annals of Computer Science and Information Systems*, vol. 35, pp. 567–576, 2023, doi: 10.15439/2023F5469.

[14] X. Jiang, H. Li, and Y. Hou, "Research on the transformation from the activity diagram to the sequence diagram considering the element mapping," *https://doi.org/10.1117/12.3011517*, vol. 12941, pp. 951–959, Dec. 2023, doi: 10.1117/12.3011517.

[15] S. Pulungan, R. Febrianti, T. Lestari, N. Gurning, and N. Fitriana, "Analisis Teknik Entity-Relationship Diagram Dalam Perancangan Database," *Jurnal Ekonomi Manajemen dan Bisnis (JEMB)*, vol. 1, no. 2, pp. 143–147, Feb. 2023, doi: 10.47233/jemb.v1i2.533.

[16] M. Rana and O. Saleh, "High assurance software architecture and design," in *System Assurances: Modeling and Management*, Academic Press, 2022, pp. 271–285. doi: 10.1016/B978-0-323-90240-3.00015-1.

[17] Hita, Djoni, Culita, and Roni Yunis, "Pemanfaatan Figma dalam Perancangan User Interface E-Commerce," *NUSANTARA Jurnal Pengabdian Kepada Masyarakat*, vol. 4, no. 3, pp. 104–111, Jul. 2024, doi: 10.55606/nusantara.v4i3.3047.

[18] M. Nakash, "Agile Software Development: The Experience of Working in Sprints," in *InSITE 2024: Informing Science + IT Education Conferences*, Informing Science Institute, Jul. 2024, pp. 002-undefined. doi: 10.28945/5252.

[19] D. Gunawan *et al.*, "Implementasi MERN Stack pada Pengembangan Sistem Penerimaan Peserta Didik Baru," *JURNAL SWABUMI*, vol. 11, no. 2, p. 2023, 2023.

[20] D. Gunawan, D. Priyawati, Y. Nugroho, F. Irsyadi, I. Andreansyah, and S. Islam, "Preserving Individual Privacy from Inference Attack in Transaction Data Publishing," in *2023 Eighth International Conference on Informatics and Computing (ICIC)*, 2023, pp. 1–6. doi: 10.1109/ICIC60109.2023.10381942.

[21] P. Leloudas, "Testing in Agile Environment," in *Introduction to Software Testing*, Apress, Berkeley, CA, 2023, pp. 173–187. doi: 10.1007/978-1-4842-9514-4_8.

[22] F. Hudi and C. Karyanti, "The Pengujian Black Box Testing pada Sistem Informasi Assesment Berbasis WEB di Bidang Pariwisata," *Jurnal Ilmiah Komputasi*, vol. 22, no. 4, pp. 553–560, Jan. 2024, doi: 10.32409/jikstik.22.4.3490.

[23] I. Akbar, "Penerapan System Usability Scale dalam Pengukuran Kebergunaan Website SMKN 13 Bandung," *INTERNAL (Information System Journal)*, vol. 7, no. 1, pp. 1–7, Jul. 2024, doi: 10.32627/internal.v7i1.865.

[24] D. Bauer, "Unit Tests for Smart Contracts," *Getting Started with Ethereum*, pp. 49–53, 2022, doi: 10.1007/978-1-4842-8045-4_4.

[25] S. Tanwar, *Blockchain Technology*. in Studies in Autonomic, Data-driven and Industrial Computing. Singapore: Springer Nature Singapore, 2022. doi: 10.1007/978-981-19-1488-1.

[26] A. Putra and H. Kabetta, "Implementation of DevSecOps by Integrating Static and Dynamic Security Testing in CI/CD Pipelines," in *ICOSNIKOM 2022 - 2022 IEEE International Conference of Computer Science and Information Technology: Boundary Free: Preparing Indonesia for Metaverse Society*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICOSNIKOM56551.2022.10034883.

[27] J. Tripp and V. Sambamurthy, "How Agile Feedback Practice Use Impacts Software Quality," *Journal of Computer Information Systems*, vol. 65, no. 3, pp. 314–330, Dec. 2023, doi: 10.1080/08874417.2023.2286557.

[28] S. Fitrianingrum, A. Wahyu, A. Wibowo, and B. Rahmawati, "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.