APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 1 (2025) 434-449
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

A Secure E-Voting Using Blockchain Technology

Syarafuddin Syazwan Rizauddin[1], Shamsul Kamal Ahmad Khalid[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: shamsulk@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.024

Article Info

Received: 29 July 2024
Accepted: 19 June 2025
Available online: 30 June 2025

Keywords

Keyword: E-Voting System, Blockchain Technology, Ethereum Blockchain, MFA, Smart Contracts, Zero-Knowledge Proofs,

Abstract

Voting for student leaders has been conducted regularly on campus. However, traditional voting methods that kept student records after votes in UTHM often suffer from illegitimacy including lack of transparency, security concerns of vote, and fraudulent practices. In this project, a secure e-voting system for UTHM using blockchain technology is proposed. The e-voting system will implement robust security features, including encryption, role-based access control, smart contract, and authorization mechanisms. Each role in the system will be determined by the administrator, candidate and students, and will incorporate features that ensure the transparency and integrity of the voting process. Several experiments on the system's functionality and security have been conducted with potential users. The results show that all functionalities achieve a 100% success rate. On the security sides, it achieves 78.6 % in input validation and error handling; and 100% on vote casting and counting procedures. 93% of the participants believe the proposed system is better than the existing system. The results indicated a promising result for a reliable blockchain based e-voting system.

1. Introduction

A voting system is a system used to determine how any elections are generally conducted and how the results of elections are determined. It is usually conducted through democracy government or organization to determine who will govern the area or lead the organization for a period chosen by the people itself. Meanwhile, blockchain technology is a distributed database or ledger that records data and shared among a computer network's nodes in a way that ensures security, transparency and integrity of the data. Its most notable role is in the world of cryptocurrencies, where it maintains a secure and decentralized record of transactions. However, blockchain technology is not limited to cryptocurrency uses [1]. Its primary function is to create a tamper-resistant and thrustless system for recording and verifying transactions. This characteristic ensures the security, transparency, and integrity of data, making it a valuable tool in a wide range of applications across various industries. It enables the creation of secure and transparent records that cannot be easily altered, offering enhanced integrity and trust in data management and transactions [2].

Lack of transparency is one of those problems that are being emphasized in this project. There is frequently a lack of transparency in the casting, recording, and counting of votes in traditional voting methods. This act of transparency may lead to suspicions of fraud and manipulation of votes. Besides that, security concerns of votes have been an issue since the usage of traditional voting methods being implemented. The UTHM voting system is vulnerable to cyberattacks, tampering and unauthorized access, increasing the risk of manipulation in votes. Thus, this project seeks to focus on developing a secure e-voting system in UTHM applying the concept of blockchain technology. The objectives of the project are to design a user-friendly e-voting system interface using

blockchain, to develop the proposed system using suitable development platforms and to test the usability and security of implemented system with UTHM students.

The significance of the project is the Ethereum Blockchain that is going to be applied in the UTHM e-voting system. The Ethereum blockchain is decentralized, meaning it operates on a distributed network of nodes rather than a single central authority. This decentralization ensures that no single entity has control over the entire system, enhancing security and reducing the risk of tampering or fraud. Next is the application of smart contracts that automate the execution of contract terms, reducing the need for intermediaries and manual intervention. In addition, the multi-factor authentication (MFA) in the e-voting system enhances the security with a high-level security implement to reduce the potential of getting attacked by hackers or unauthorized user. This enhances the overall integrity of the election process.

2. Literature Review

This section will discuss comparing current systems with this project more closely and provide a brief explanation of the various decision-making security measuring approaches. It will contain earlier studies on the issue and their resolution in relation to the current project.

2.1 Introduction

In this section, literature review is taking places in order to better understand current research and scholarly works about blockchain technology, electronic voting systems, and their uses in electoral processes, a literature review is conducted. This part aims to give an overview of what is known in the field right now by pointing out trends, problems, and the best ways to do things.

2.2 Voting System

2.2.1 Manual Voting

Manual voting, also known as traditional or paper-based voting, is an elections procedure in which eligible voters express their choices by marking actual paper ballots. Manual voting requires people to go to the polling station to cast their vote. Many preparations have taken place to have a successful and secure voting process. The voter must be present at the polling site to cast a ballot. These days, some people only manage to go to the polls because they are available, or it is a holiday. For those who do not have any free schedule, they had to skip their vote to fulfil their work. There are also numerous cases like the disability of a person to physically attend the polling station to cast their vote. Manual voting is great and all as it can prevent identity theft and double votes, but the disadvantages of the process also may bring harm to the vote.

2.2.2 Online Voting

Online voting, also known as electronic voting or e-voting, is the method of casting and tallying votes using electronic methods, usually conducted via the internet. Online voting requires people to cast their vote on a web-based system developed by a trusted organization involved in the voting process. This voting method uses digital technology to simplify the election process, with the goal of enhancing the convenience, accessibility, and efficiency of voting. But there are some advantages to an online voting system which make it vulnerable against cyberattacks that are larger scale and harder to detect. Online voting gives many advantages as it can reduce labor, make voting process easier to use and more productive but against cyberattacks, not much can do.

2.3 Blockchain Technology

Blockchain is a technology characterized by its desirable attributes of decentralization, autonomy, integrity, immutability, verification, fault-tolerance, anonymity, auditability, and transparency [3]. To be more specific, blockchain technology is digital ledgers that are both tamper evident and tamper resistant. In blockchain, data is kept in a distributed ledger. The blockchain technology ensures the integrity and availability of the distributed ledger, enabling members in the network to create, read, and verify transactions. Blockchain technology combines existing and tested concepts into a single solution. The application of blockchain technology does not provide a global solution, and it requires careful consideration of factors such as handling malevolent users, implementing effective controls, and acknowledging the restrictions of the implementations.

2.4 Smart Contracts

Self-executing contracts, or smart contracts, have the conditions of the contract explicitly encoded into the code. These blockchain-based contracts automatically enforce and carry out their terms when predefined criteria are satisfied. Smart contracts are initially conceptualised by scholars as intricate computer programmes composed of a sequence of codes that meticulously encapsulate the terms and conditions mutually agreed upon

by the contracting parties. Using digital platforms like Ethereum, which is known as a leading smart contract ecosystem, the activation and subsequent execution of these contracts are dependent on the satisfaction of specified conditions.

2.5 Zero-Knowledge Proofs

Zero-knowledge proof is a cryptographic technique that convinces the verifier of the truth of a statement without revealing any significant information to the verifier [7]. A zero-knowledge proof essentially establishes the accuracy of a statement while keeping any specific information regarding the statement's matter. The idea behind "zero-knowledge" comes from the idea that the proof should show nothing about the knowledge being proven, other than the fact that it is true. Zero-knowledge proofs are highly beneficial for enhancing privacy and security in a wide range of applications, such as authentication, identity verification, and secret transactions in cryptocurrencies.

2.6 Role-Based Access Control (RBAC)

The role-based access control model provides a structured system that grants users access to resources based on their assigned functions. In an RBAC model, permissions are allocated to each agent's role based on the organization's functional description, and access rights are granted indirectly by linking a user to a certain role [8]. Meaning that RBAC is centered around roles, which are determined by distinct job tasks or responsibilities within an organization. Every role corresponds to a certain set of permissions that determine the actions or uses that a user assigned to that position can carry out. Users are later assigned to one or several roles according to their organizational tasks and responsibilities, so eliminating the complex array of individual user rights. In this framework, higher-level positions can inherit the permissions of lower-level roles, which enables easier management and minimize redundancy. An advantage of RBAC is its ability to dynamically assign users to roles based on changes in their position duties, making it easier to adapt access control to organizational changes.

2.7 System Comparison

Comparing an existing secure e-voting system without blockchain technology with one that implements blockchain technology reveals significant differences, demonstrating the exclusive benefits that blockchain offers to the field of electronic voting. The comparison with the existing system is shown in Table 1.

Table 1 *System Comparison*

Features/System	Evalato	Typeform	ElectionBuddy	Proposed System
Registration	√	√	√	√
Login	√	√	√	√
Update Profile	√	X	√	√
Access Control	√	X	X	√
Smart Contracts	X	X	X	√
Ethereum Blockchain	X	X	X	√
Zero-Knowledge Proofs	X	X	X	√
RBAC	√	√	√	√
Decentralization	X	X	X	√
Update Vote	√	√	√	√
Web-Based	√	√	√	√
Ease of use	√	X	√	√
Navigation Elements	√	√	X	√

3. Methodology

This section will explain the methodology and research design used for achieving the objectives of the study. It performs as a strategic plan for carrying out the research, outlining the selected methodologies, procedures, and approaches.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 434-449

3.1 Prototyping Model

The prototyping model is a software development methodology that prioritizes the creation of a prototype or preliminary version of the software at the beginning of the development process. It follows an iterative and progressive approach. This methodology is primarily applicable to extensive software projects and new innovations in software development [10]. Fig. 1 shows the prototyping methodology.

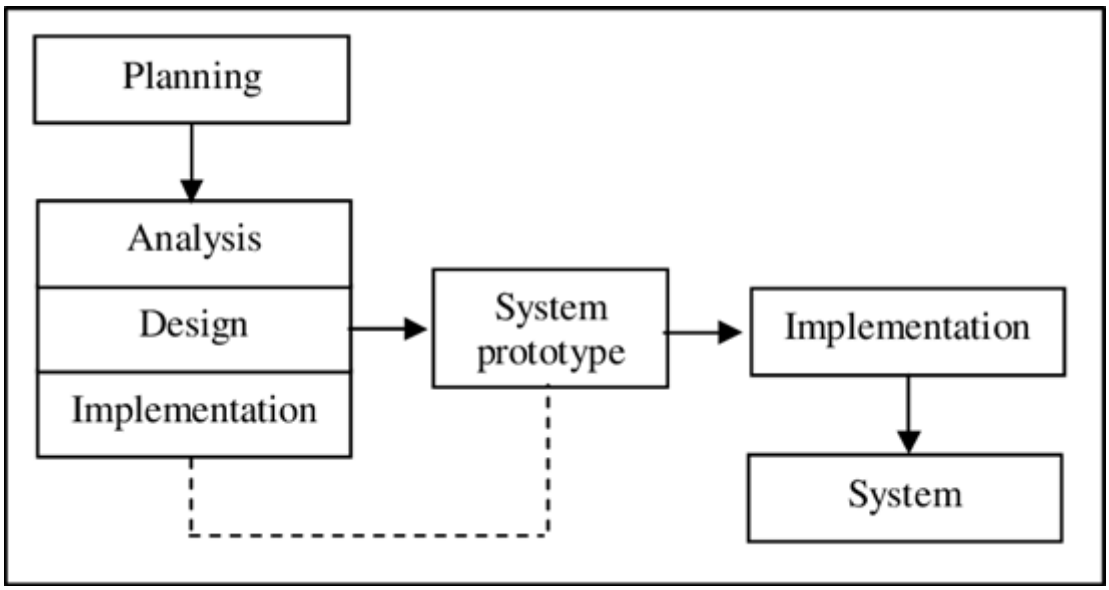

Fig.1 *Prototyping Model*

Fig.1 shows the prototyping methodology. This model consists of five phases which are planning phase, analysis phase, design phase, implementation phase and finalization phase. The iterative nature of the prototype model enables ongoing improvement and enhancement, guided by user feedback. The iterative process of prototype, review, and development is repeated until the final software product aligns with the user's expectations and requirements.

3.2 Software Requirements for System Development

Software requirements contain full specifications describing the specific functionalities, characteristics, limitations, and features that a software system needs to contain. Table 2 shows the software requirements for this project.

Table 2 *Software Requirements*

Software	Description
Visual Code Studio	For code editor
JavaScript	For programming language
Solidity	For writing smart contracts on the Ethereum platform.
JSON	For configuration and package management files
MetaMask	For Ethereum wallet
Ganache	For Ethereum Development
Draw.io	For sketch ERD and DFD diagram
Google Chrome	For web-based server provider and searching information
Microsoft Office	Documentation project
Operating System	Windows 11 Pro

3.3 Hardware Requirements for System Development

Hardware requirements are the specific features and functionalities that a computer system or device must possess to efficiently operate a software application or system. Table 3 shows the hardware requirements for this project.

Table 3 *Hardware Requirements*

Hardware	Specifications
Devices	Desktop
Processor	Intel® Core™ i5-10400F CPU
Memory (RAM)	16 GB
Storage	SSD 1 : 512 GB, SSD 2 : 1 TB
Graphics Processing Unit (GPU)	NVIDIA GeForce RTX 3060
Wi-Fi	500 Mbps

3.4 Prototyping Methodology Phase

Prototyping is an iterative development approach that involves developing an initial or incomplete version of a system to collect feedback and enhance the concept. Table 4 shows the prototype methodology phases.

Table 4 Prototyping Methodology Phases

Phase	Description	Outcome
Planning	Choosing supervisor for the final project and proposed project title to chosen supervisor.	A project proposal
Analysis	Analyze comprehensive evaluations, gathering data, and engaging with stakeholders to determine the project's functional and non-functional requirements. Sketch ERD and DRD of the system. Identify software and hardware to develop the system.	Entity Relationship Diagram (ERD) Data Flow Diagram (DFD)
Design	Develop system interface and database from scratch by referring to ERD and DFD as references	System Interface System Database
Implementation	Test the prototype version 1 and receive feedback from user.	Secure e-voting system for UTHM version 1 using Ethereum blockchain technology and user feedback.
System Prototype	Improvising any lacks in the system prototype.	Secure e-voting system for UTHM version 2 using Ethereum blockchain technology.
Implementation	Test the prototype version 2 and receive feedback from user.	Upgrade Secure e-voting system for UTHM using blockchain technology and user feedback.
Finalization	The developed system is success and ready to be executed.	Secure e-voting system for UTHM using blockchain technology

4. System Analysis and Design

This section covers the fundamental phases of analysis and design in the software development lifecycle. The analysis phase begins with an extensive review of requirements, including techniques such as interviews, questionnaires, and workshops to gain a full understanding of user demands and system functionalities. Then the section proceeds to the design phase, whereby the overall architectural framework is systematically developed, identifying the system's components and how they are connected. Through both phases, the section highlights the role of documentation in collecting design choices and providing guidance to the development team.

4.1 System Requirements Analysis

System requirements analysis is a systematic approach used to identify the necessary resources to meet a system's needs, as well as the specific requirements for those resources. This analysis provides a solid foundation for designing or selecting suitable resources. The process involves a methodical examination of the requirements, capabilities, and limitations of users, applying methods such as interviews, questionnaires, and workshops to gather a wide range of viewpoints from many stakeholders. The need is the fundamental system need that serves as the basis for all other requirements and design decisions.

4.1.1 Functional Requirements

Functional requirements describe the desired actions and operations of the system. The behavior can be represented as the services, tasks, or operations that the system must execute. When developing a system, it is important to differentiate between the basic features required for any system to be competitive in that field area, and the special features that set the system apart from competitors' system and other versions.

Table 5 *Functional Requirements for Admin*

Page	Functionalities
Login	Admin shall be able to login using registered account.
Admin Homepage	Admin shall be able to view, edit and delete data and information in admin dashboard.
User Management	Admin shall be able to add, modify, and deactivate user accounts, including other administrators, staff, and voters.
Create Voting Events	Admins can create new voting events, specifying details such as event name, date, time, and candidates.
Verify Student	Admins can verify or reject student applications for participation in voting events.
Modify Voting Events	Admins can edit details of existing voting events, such as changing dates, times, or candidate lists.
View Voting Results	Admins can access and review the results of voting events once the voting period has ended.
Logout	Admin shall be able to logout from the system.

Table 6 *Functional Requirements for Student*

Page	Functionalities
Registration	Student shall be able to register by providing required information, including a valid email address and personal details.
Login	Student shall be able to login using registered account.
Student Homepage	Student shall be able to view, add and delete information in user dashboard
Enroll in Voting Events	Students can enroll in upcoming voting events, ensuring they are registered to vote in those events.
View Candidate Information	Students can view detailed information about candidates, including their profiles and manifestos.
Cast Votes	Students can securely cast their votes in active voting events.
Access Voting Results	Students can view the results of elections once the voting period has ended.
Logout	User shall be able to logout from the system.

4.1.2 Non-Functional Requirements

Non-Functional Requirements, as defined by IEEE, refer to software requirements that define the manner in which the software will operate, rather than what it will accomplish. These requirements encompass aspects such as software performance, external interface, design limitations, and software quality features [11]. Non-functional requirements typically refer to features such as performance, security, dependability, usability, and scalability.

Table 7 *Non-Functional Requirements of the System*

Requirement	Description
Performance	• The system shall respond to user interactions within a maximum of 2 seconds to ensure a smooth and highly responsive voting experience. • The system shall support a minimum of 1,000 simultaneous users during peak voting periods.
Operational	• The system shall have the capacity to connect easily with external systems, including but not limited to voter registration databases or election result reporting systems. • The system shall be compatible with common web browsers
Security	• Encryption that uses industry-standard methods will be applied to all communication between the client and server. • The system shall implement strict access controls to make sure that only authorised users can retrieve sensitive election results data.
Reliability	• The system shall have an uptime of no less than 99.9% to ensure availability during the whole voting session. • The system shall include durable error-handling methods to effectively manage unexpected issues and deliver helpful error notifications to users.
Usability	• The system shall be developed with straightforward navigation and intuitive features to provide a user-friendly interface for both candidate and student. • The system shall meet accessibility requirements to assist users with disabilities.

4.2 User Requirements Analysis

User requirements analysis is a process that involves understanding, documenting, and analyzing the needs, expectations, and preferences of the end-users who will interact with the system. Showing this process in a flowchart demonstrates the organized process of actions taken to fully understand and maintain the requirements of users. Fig. 2 shows the Student Voting Process flowchart in the system.

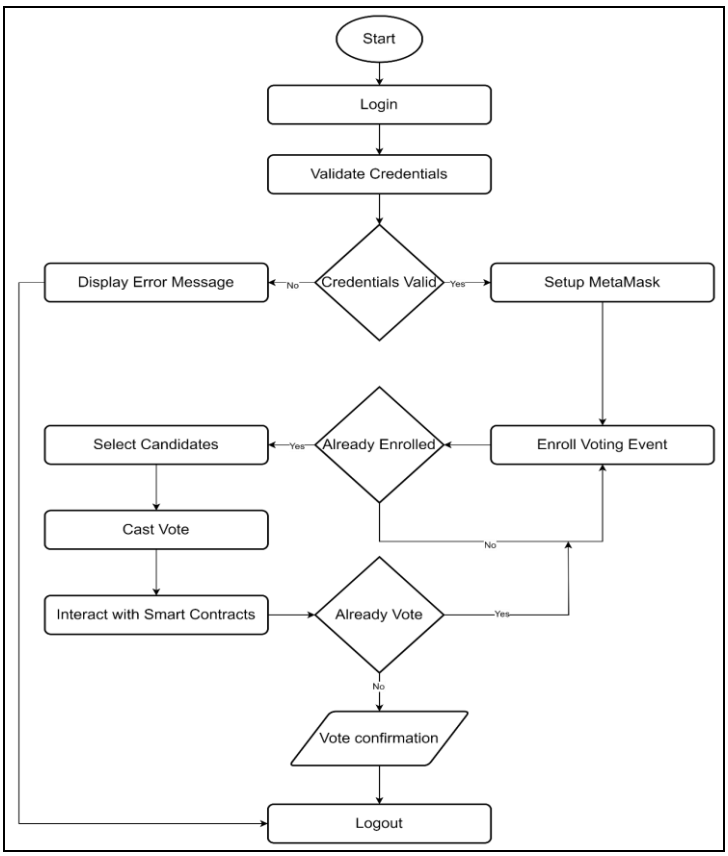

Fig. 2 *Student Voting Process Flowchart*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 434-449

Fig.2 shows the student voting process throughout the entire system. It describes each phase of the e-voting system that integrates MetaMask with blockchain. User logs into the system to start the procedure, and then their credentials are validated. The system stops the process and displays an error notice if the credentials are invalid. The user is instructed to set up MetaMask, a necessary step for enabling safe interactions with the Ethereum blockchain, however, following successful confirmation. The user's enrollment in the voting event is determined at the following decision point. If not, they go ahead and sign up for the event. The system verifies that a person has voted once they are enrolled. The user is shown the candidates and given the option to vote if they haven't already. After that, this vote is safely recorded by interacting with smart contracts. The voting procedure is concluded when the user logs out after a successful vote confirmation. This systematic method highlights the creative use of blockchain technology in e-voting platform by guaranteeing the security and integrity of the voting process.

4.3 System Analysis

4.3.1 Context Diagram

Fig.3 shows the context diagram of the proposed system. There are 2 modules which contain admin and student that are revolved around UTHM E-Voting System. The administrator is in charge of managing all aspects of the electronic voting, such as organising and launching events, including the addition and removal of candidates. Additionally, they can log in and see the outcome of the voting process. Meanwhile, students engage in the system through registration, participation at voting activities, and casting of votes. They can view the results by logging in as well. The core of the system, which makes use of blockchain technology, preserves the security and integrity of the voting process and offers an open, impenetrable method for managing elections. This configuration highlights the system's dual features, where students participate in a safe and convenient way while the administrator oversees the election process.

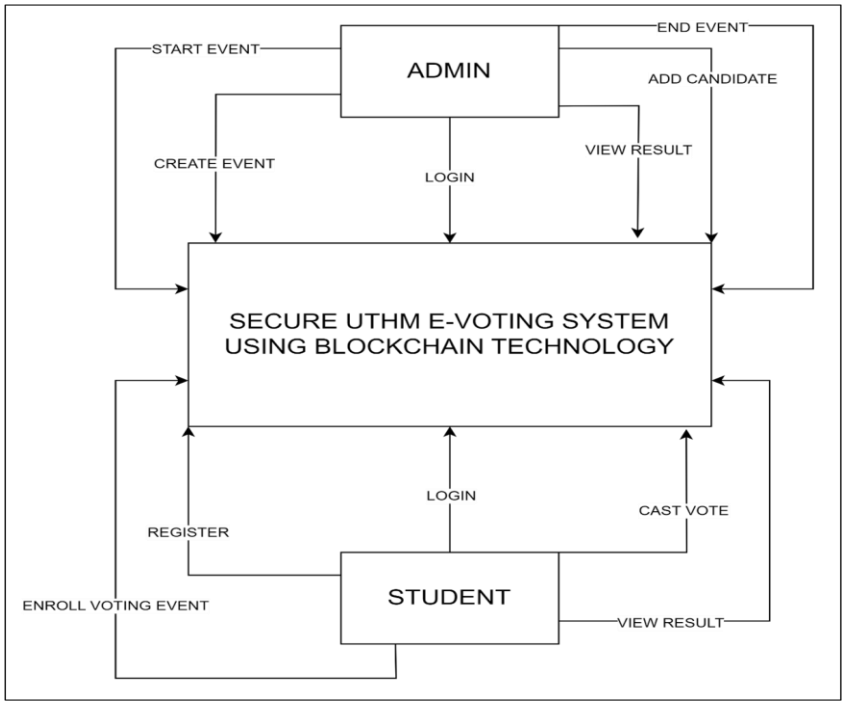

Fig. 3 *Context Diagram*

4.3.2 Entity Relation Diagram (ERD)

Fig. 4 shows the entity relation diagram of UTHM E-Voting System. The ERD represents the main entities in the system and the relationships between them. The key entities are Admin. Student, Event, and Vote. Each entity represents of what they are to other entities. Based on the Fig. 4, it is shown that the diagram represents a candidate can receive multiple votes, but each vote is cast for one candidate. Other than that, a voting event can have multiple votes, but each vote is cast in one voting event. This ERD provides a comprehensive view of the key entities, their attributes, and the relationships between them in the UTHM E-Voting System.

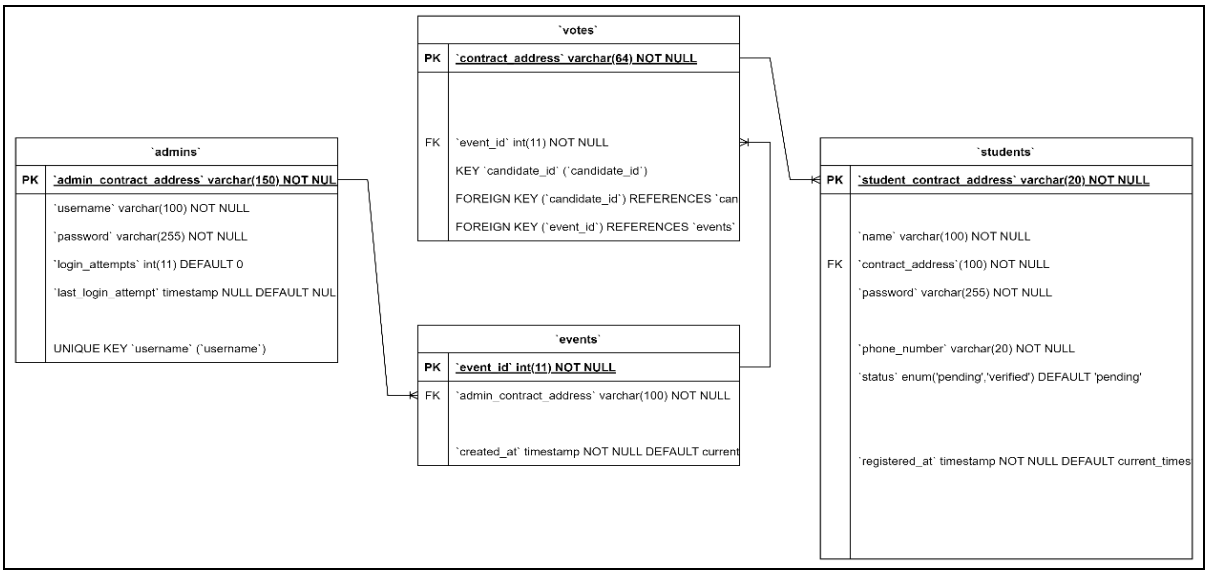

Fig. 4 *Entity Relation Diagram of UTHM E-Voting System*

4.4 System Design

In the software development life cycle (SDLC), system design is a phase where the proposed requirements are turned into an extensive construction blueprint for the real system. This phase involves making decisions regarding the structure, elements, modules, data storage, user interface, and other technical criteria. The primary goal of system design is to develop a durable and accessible solution that fulfills the required criteria, considering aspects such as performance, security, and maintainability.

4.4.1 Interface Design

Interface design, also known as user interface (UI) design, is a part of software development that focuses on the creation of user-friendly and attractive interaction between users and the system. The objective of interface design is to optimize the user experience by presenting information in a clear, easily accessible, and visually pleasing way. Wireframing serves as a means of demonstrating the fundamental framework and arrangement of the user interface ahead of including complex design components.

```
renderAdminHome = () => {
  const EMsg = (props) => {
    return <span style={{ color: "tomato" }}>{props.msg}</span>;
  };

  const AdminHome = () => {
    // Contains of Home page for the Admin
    const {
      handleSubmit,
      register,
      formState: { errors },
    } = useForm();

    const onSubmit = (data) => {
      this.registerElection(data);
    };

    return (
      <div>
        <form onSubmit={handleSubmit(onSubmit)}>
          {!this.state.elStarted & !this.state.elEnded ? (
            <div className="container-main">
              {/* about-admin */}
              <div className="about-admin">
                <h3>About Admin</h3>
                <div className="container-item center-items">
                  <div>
                    <label className="label-home">
                      Full Name{" "}
                      {errors.adminFName && <EMsg msg="*required" />}
                      <input
                        className="input-home"
                        type="text"
                        placeholder="First Name"
                        {...register("adminFName", {
                          required: true,
                        })}
```

Fig. 5 *NodeJS code for interface*

Fig. 5 shows the code segment for admin homepage interface. The code segment contains the input form function with input validation for each content. For Fig.5 (Continued) contains the code segment of what Admin should fill out to deploy the event for candidate and student.

```
            <input
              className="input-home"
              type="text"
              placeholder="Last Name"
              {...register("adminLName")}
            />
          </label>

          <label className="label-home">
            Email{" "}
            {errors.adminEmail && (
              <EMsg msg={errors.adminEmail.message} />
            )}
            <input
              className="input-home"
              placeholder="eg. you@example.com"
              name="adminEmail"
              {...register("adminEmail", {
                required: "*Required",
                pattern: {
                  value: /^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,4}$/, // email validation using
                  message: "*Invalid",
                },
              })}
            />
          </label>
```

```
          <label className="label-home">
            Job Title or Position{" "}
            {errors.adminTitle && <EMsg msg="*required" />}
            <input
              className="input-home"
              type="text"
              placeholder="eg. HR Head "
              {...register("adminTitle", {
                required: true,
              })}
            />
          </label>
        </div>
      </div>
    </div>
    {/* about-election */}
    <div className="about-election">
      <h3>About Election</h3>
      <div className="container-item center-items">
        <div>
          <label className="label-home">
            Election Title{" "}
            {errors.electionTitle && <EMsg msg="*required" />}
            <input
              className="input-home"
              type="text"
              placeholder="eg. School Election"
              {...register("electionTitle", {
                required: true,
              })}
            />
```

Fig. 5 *(Cont.)*

```
                    </label>
                    <label className="label-home">
                      Organization Name{" "}
                      {errors.organizationName && <EMsg msg="*required" />}
                      <input
                        className="input-home"
                        type="text"
                        placeholder="eg. Lifeline Academy"
                        {...register("organizationTitle", {
                          required: true,
                        })}
(property) JSX.IntrinsicElements.div:
React.DetailedHTMLProps<React.HTMLAttributes<HTMLDivElement>, HTMLDivElement>
                  </div>
                </div>
              </div>
            </div>
          ) : this.state.elStarted ? (
            <UserHome el={this.state.elDetails} />
          ) : null}
          <StartEnd
            elStarted={this.state.elStarted}
            elEnded={this.state.elEnded}
            endElFn={this.endElection}
          />
          <ElectionStatus
            elStarted={this.state.elStarted}
            elEnded={this.state.elEnded}
          />
        </form>
      </div>
    );
  };
  return <AdminHome />;
};
}
```

Fig. 5 *(Cont.)*

Fig. 6 shows the admin homepage interface that contains what has been implemented in the source code. Every input form is validated with input validation, so admin did not miss any details to create events before deploying students to vote. The form will be initiated to start a voting event.

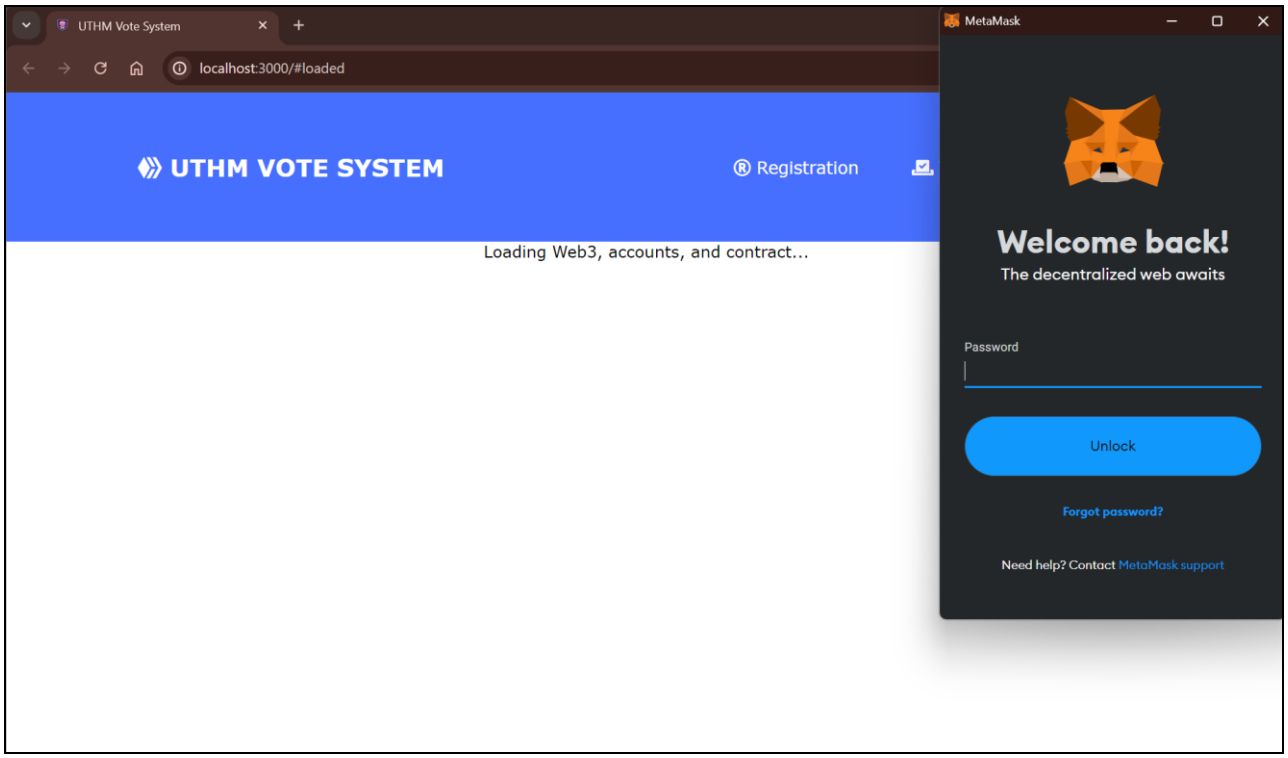

Fig. 6 *Admin Homepage Interface*

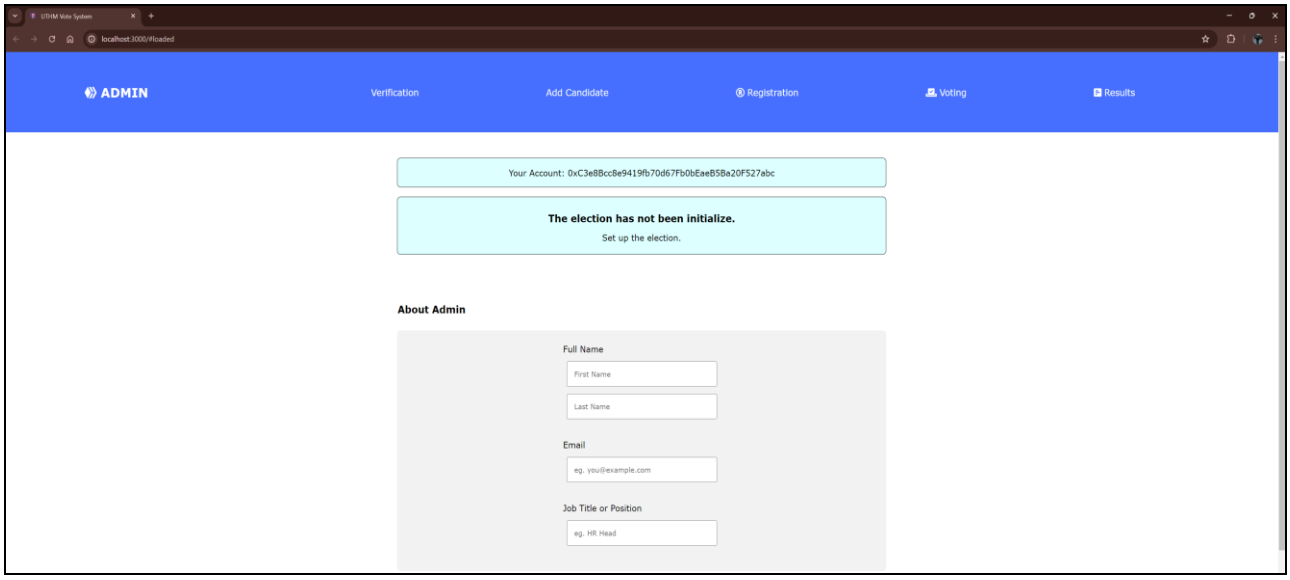

Fig. 6 *(Cont.)*

4.5 Implementation and Testing

The UTHM e-voting system using blockchain technology is built using HTML, CSS AND JavaScript (ReactJS) for the frontend, JavaScript (NodeJS and Truffle) for the backend and MetaMask and Truffle Ganache as blockchain tools as its backend.

4.5.1 Vote

For the development of the vote page for UTHM e-voting system, the following source code was used as shown in Fig. 7.

```
render() {
  if (!this.state.web3) {
    return (
      <>
        {this.state.isAdmin ? <NavbarAdmin /> : <Navbar />}
        <center>Loading Web3, accounts, and contract...</center>
      </>
    );
  }

  return (
    <>
      {this.state.isAdmin ? <NavbarAdmin /> : <Navbar />}
      <div>
        {!this.state.isElStarted && !this.state.isElEnded ? (
          <NotInit />
        ) : this.state.isElStarted && !this.state.isElEnded ? (
          <>
            {this.state.currentVoter.isRegistered ? (
              this.state.currentVoter.isVerified ? (
                this.state.currentVoter.hasVoted ? (
                  <div className="container-item success">
                    <div>
                      <strong>You've casted your vote.</strong>
                      <p />
                      <center>
                        <Link
                          to="/Results"
                          style={{
                            color: "black",
                            textDecoration: "underline",
                          }}
```

Fig. 7 *Source code for Vote page*

Fig. 7 shows the source code for the voting process. This function act as an approach of private transaction being made during voting process. Ethereum blockchain will be triggered and will be connected to MetaMask for transactions. Voter can make transactions of casting their vote after successfully connect to MetaMask. If user failed to connect to MetaMask, no vote will be counted.

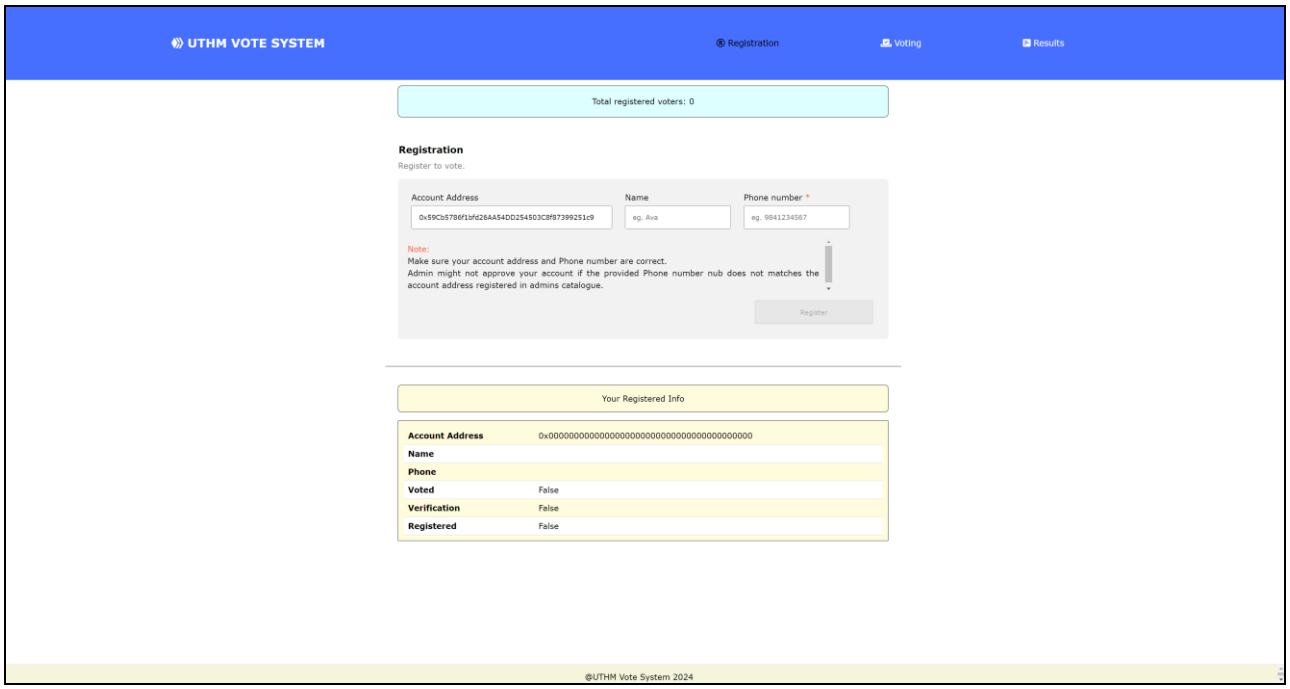

Fig. 8 *Student vote page*

Fig. 8 shows the student vote page. Student will have to input their contract address to show the validation to vote for the event. It is to make sure that one student can only vote once for an event. Then, they will proceed on choosing an event to vote. The list of candidates for the chosen event will be shown for voter to vote their cast for candidates.

4.5.2 Ethereum Blockchain

For the development of Ethereum Blockchain implementation in the UTHM e-voting system, the following code was used as shown in Fig. 9.

```
try {
  // Get network provider and web3 instance.
  const web3 = await getWeb3();

  // Use web3 to get the user's accounts.
  const accounts = await web3.eth.getAccounts();

  // Get the contract instance.
  const networkId = await web3.eth.net.getId();
  const deployedNetwork = Election.networks[networkId];
  const instance = new web3.eth.Contract(
    Election.abi,
    deployedNetwork && deployedNetwork.address
  );

  // Set web3, accounts, and contract to the state, and then proceed with an
  // example of interacting with the contract's methods.
  this.setState({
    web3: web3,
    ElectionInstance: instance,
    account: accounts[0],
  });
```

Fig. 9 *Code segment for Ethereum Blockchain*

Fig. 9 shows the code segment for Ethereum Blockchain used in the system to implement smart contracts, which can automate and securely handle the voting process. The code checks for the presence of an Ethereum provider (like MetaMask) and requests account access. The code interacts with a smart contract on the Ethereum blockchain by specifying a contract address and ABI (Application Binary Interface). Web3.js is a JavaScript library that provides an interface for interacting with the Ethereum blockchain. The contract address

uniquely identifies the deployed smart contract on the Ethereum blockchain, while the ABI defines the interface of the smart contract, including its functions, arguments, and return types. With this, Web3.js can create JavaScript objects representing the smart contract's functions, allowing the script to call those functions and interact with the contract's state.

4.6 Testing

Testing will revolve around system functional testing, security test case result and user acceptance testing. Testing is to be done to ensure that the system satisfies all user criteria and ready to deploy. The testing is meant to assess whether it is ready to be use widely and meet standard requirements.

4.6.1 System Functional Testing

System functional testing is meant to test that every module produced in the project is working perfectly and no interruption when used. Table 8 shows a list of test case results of the system. Based on the result, all cases have been executed successfully by users.

Table 8 *List of test cases*

No.	Test Cases	Description	Status
Test Cases Login (TEST_01)			
1.	TEST_01_01	Admin and Student enter correct username and password	PASS
2.	TEST_01_02	Admin and Student enter invalid username and password	PASS
3.	TEST_01_03	Admin and Student enter contract address	PASS
4.	TEST_01_04	All input validation was successfully use	PASS
Test Cases Registration (TEST_02)			
1.	TEST_02_01	New Student register contract address, username and phone number.	PASS
2.	TEST_02_02	All input validation was successfully use	PASS
Test Cases Admin Module (TEST_03)			
1.	TEST_03_01	Admin can verify and reject student	PASS
2.	TEST_03_02	Admin can view, create and end vote events	PASS
3.	TEST_03_03	Admin can view total number of students	PASS
4.	TEST_03_04	Admin can view total number of candidates	PASS
5.	TEST_03_05	Admin can view result vote event	PASS
Test Cases Student Module (TEST_04)			
1.	TEST_04_01	Student can view total number of candidates	PASS
2.	TEST_04_02	Student can view candidate`s details	PASS
3.	TEST_04_03	Student can view result vote event	PASS
4.	TEST_04_04	Student can cast vote	PASS
5.	TEST_04_05	Student can update their personal details	PASS
Test Cases Log Out Module (TEST_05)			
1.	TEST_05_01	Admin and Student can log out from their account	PASS

4.6.2 Security Test Case Result

Security test case results are based on detailed findings from various security tests performed. The security test case results indicate that the application is secure against common web vulnerabilities, including SQL injection, XSS, CSRF, and improper input validation. The application follows the best practices for password storage and network security.

Table 9 *List of security test case result*

No.	Security Test Cases	Pass	Fail
1.	All input fields were properly sanitized and used prepared statements when login.	P	0
2.	All user inputs escaped and filtered correctly. No XSS vulnerabilities were identified.	P	0
3.	Validation messages will show up if user does not fill all fields provided in module,	P	0
4.	Password in the textbox need to be hidden.	P	0
5.	Error messages were user-friendly and did not expose sensitive information.	P	0
6.	The system will automatically logout after 45 minutes of inactive.	P	0
7.	Access controls were properly enforced. Unauthorized access attempts were blocked.	P	0
8.	Smart contract interactions were secure, with no vulnerabilities found in transaction handling.	P	0
9.	Only necessary services were exposed, and appropriate network security measures were in place.	P	0

4.6.3 User Acceptance Test

The user acceptance test ensures that the application meets the standard requirements and is ready for deployment. 14 students have been asked to test the system and given their responses on UTHM e-voting system.

Table 10 *User Acceptance test results*

No.	Description	PASS/person	FAIL/person
1.	Can user login to the UTHM Vote System?	14	0
2.	Can user sign-up to the UTHM Vote System?	14	0
3.	Did user experience any input validation, error handling and login attempt when sign-up or login?	11	3
4.	Can user get into user homepage after login or sign-up?	14	0
5.	Can user view candidate list in their account?	13	1
6.	Can user cast a vote?	14	0
7.	Can user see the result vote displayed at the end of current event?	14	0
8.	Can user logout from the UTHM Vote System?	14	0
9.	What is user impression on UTHM Vote System?	13	1
10.	Do user enjoy using the UTHM Vote System?	13	1
11.	Did user think the UTHM Vote System is better than existing system?	13	1

5. Conclusion

In conclusion, this project has successfully designed a web-based UTHM e-voting system using Blockchain Technology by incorporating an intuitive interface designed with the end-user in mind. The user experience (UX) design principles ensure that the interface is easy to navigate for all users, including students with varying levels of technical proficiency. The system has also been successfully developed and utilizes the Ethereum blockchain for its decentralized ledger, ensuring secure and transparent recording of votes. While integrating with HTML, CSS and ReactJS as the frontend technology, expend the usage of the backend technology with solidity, NodeJS, Web3.js, and tools usage, MetaMask and Truffle Ganache. Based on the testing, several advantages can be derived from testing experience which involves Ethereum Blockchain that ensures voting records are immutable, providing a secured web-based system and offers decentralized nature of blockchain, reducing the risk of single points of failure and attacks and enhancing the overall security of the system. Also, the use of cryptographic techniques and smart contracts ensures that only eligible votes are counted, and multiple voting is prevented. Voters can verify that their vote has been correctly recorded and counted without compromising privacy. Disadvantages also found when deploying the system which Student and Candidate need to understand how to use blockchain technology and tools like MetaMask, which can be a learning curve. Ethereum Blockchain voting requires reliable internet access, which can be a problem in remote or underdeveloped areas in which dependence on technology that having technical failures or outages could

disrupt the voting process. Recommendations have been made which to provide extensive training and educational resources for Student and Candidate to ensure they understand how to use the system securely and provide support for student and candidate who may not have access to necessary technology or the internet. This could include setting up secure voting kiosks in accessible locations.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design****: S. S. Rizauddin, S. K. Ahmad Khalid;* ***data collection****: S. S. Rizauddin, S. K. Ahmad Khalid;* ***analysis and interpretation of results****: S. S. Rizauddin, S. K. Ahmad Khalid;* ***draft manuscript preparation****: S. S. Rizauddin, S. K. Ahmad Khalid. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] A. Hayes, "Blockchain Facts: What is it, how it works, and how it can be used," *Investopedia*, May 18, 2024. https://www.investopedia.com/terms/b/blockchain.asp

[2] A. Feger, "Blockchain Technology explained: Benefits & applications," *EMARKETER*, Mar. 14, 2024. https://www.insiderintelligence.com/insights/blockchain-technology-applications-use-cases/

[3] H. Guo and X. Yu, "A survey on blockchain technology and its security," *Blockchain. Research and Applications*, vol. 3, no. 2, p. 100067, Jun. 2022, doi: 10.1016/j.bcra.2022.100067.

[4] D. J. Yaga, P. M. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," *Blockchain. Research and Applications*, Oct. 2018, doi: 10.6028/nist.ir.8202.

[5] F. Sinigaglia, R. Carbone, G. Costa, and N. Zannone, "A survey on multi-factor authentication for online banking in the wild," *Computers & Security*, vol. 95, p. 101745, Aug. 2020, doi: 10.1016/j.cose.2020.101745.

[6] S. Das, B. Wang, Z. Tingle, and L. J. Camp, "Evaluating User Perception of Multi-Factor Authentication: A Systematic review," *arXiv.org*, Aug. 16, 2019. https://arxiv.org/abs/1908.05901

[7] T. Feng, P. Yang, C. Liu, J. Fang, and R. Ma, "Blockchain data Privacy Protection and Sharing scheme based on Zero-Knowledge Proof," *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–11, Feb. 2022, doi: 10.1155/2022/1040662.

[8] R. Xu, Y. Chen, E. Blasch, and G. Chen, "Exploration of blockchain-enabled decentralized capability-based access control strategy for space situation awareness," *Optical Engineering*, vol. 58, no. 04, p. 1, Feb. 2019, doi: 10.1117/1.oe.58.4.041609.

[9] D. Di Francesco Maesa, P. Mori, and L. Ricci, "A blockchain based approach for the definition of auditable Access Control systems," *Computers & Security*, vol. 84, pp. 93–119, Jul. 2019, doi: 10.1016/j.cose.2019.03.016.

[10] S. SoobiaEtAl, "Analysis of software development methodologies," *International Journal of Computing and Digital System/International Journal of Computing and Digital Systems*, vol. 8, no. 5, pp. 445–460, Jan. 2019, doi: 10.12785/ijcds/080502.

[11] A. Matoussi and R. Laleau, "A Survey of Non-Functional Requirements in Software Development Process," 2008. https://hal.science/hal-01224656/

[12] Bjeladinović, "A proposal of architecture for integration and uniform use of hybrid SQL/NoSQL database components.," *Journal of Systems and Software*, Oct. 2020, [Online]. Available: https://doi.org/10.1016/j.jss.2020.11063