Secure online voting system using BlockChain

A Project Report

Submitted in partial fulfillment of the requirements for the

Award of the Degree

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

Submitted by

J.SUDARSHAN (212W1A0545)
S.JOHITHA (212W1A0591)
G.EBINAGER RAJU (212W1A0536)
CH.VIJAY (222W5A0502)
K.VAISHNAVI (212W1A0550)

Under the Esteemed Guidance

MRS.M.Bhavya sri, M.Tech

Assistant Professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GVR&S COLLEGE OF ENGINEERING AND TECHNOLOGY

(Approved by A.I.C.T.E New Delhi, Affiliated to JNTUK, Kakinada)

Accredited by NAAC | ISO Certified Institution

Budampadu-522017, Guntur (Dt), A.P., India

APRIL-2025

GVR&S COLLEGE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, Affiliated to JNTUK, A.P, India)

Accredited by NAAC | ISO Certified Institution

Budampadu, Etukuru (P.O), Guntur (Dt) – 522017, A.P, India

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate

This is to certify that the project entitled **"SECURE ONLINE VOTING SYSTEM USING BLOCK CHAIN"** is authentic record of work done by **S.JOHITHA** (212W1A0591)**, J.SUDARSHAN** (212W1A0545)**, G.EBINAGER RAJU** (212W1A0536)**, CH.VIJAY** (222W5A0502), **K.VAISHNAVI** (212W1A0550)**,** in partial fulfillment of the requirements for the award of the Degree of **BACHELOR OF TECHNOLOGY** in **Computer Science and Engineering** under my supervision. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Sign. of Internal Guide Sign. of External Sign. Of HOD

PRINCIPAL

DECLARATION

We, the students of **G.V.R & S College of Engineering & Technology**, Guntur District, Andhra Pradesh, hereby declare that this project work titled as "**SECURE ONLINE VOTING SYSTEM USING BLOCKCHAIN"**. We, being submitted to the Department of Computer Science & Engineering of this Institute, affiliated to Jawaharlal Nehru Technological University Kakinada, Kakinada, for the award of the Degree in **Bachelor of Technology** in Computer Science and Engineering is a record of bonafide work done by us at **G.V.R & S COLLEGE OF ENGINEERING & TECHNOLOGY** and it has not been submitted to any other Institute or University for the award of any other Degree.

Name of the student	Regd.no	Signature
S.JOHITHA	212W1A0591	
J.SUDARSHAN	212W1A0545	
G.EBINAGER RAJU	212W1A0536	
CH. VIJAY	222W5A0502	
K.VAISHNAVI	212W1A0550	

Place: Guntur

Date

ACKNOWLEDGEMENT

We are very much thankful to **MRS.M.Bhavya sri,** M.Tech, Assistant Professor, Department of Computer Science and Engineering, **G.V.R & S College of Engineering & Technology**, Guntur, for her incredible support, guidance and motivation. Her deep insights helped me at various stages of our project work.

We would like to express our sincere thanks to **Mr. CH.Paparao**, Head of the Department of Computer Science and Engineering for his encouragement.

We would like to take this opportunity to express our gratitude to our Chairperson **Dr.G.Sindhura**, our Director **Dr.SK.Karemoon** and our Principal **Dr.P.Bhaskar Naidu**, for giving us this opportunity to do the project work.

We also thankful to all our faculty members for their suggestions and the moral support extend by them.

Finally yet importantly, above all, from the deep of our heart, we thank GOD, the Almighty, for granting us the wisdom, health, wealth and strength to undertake this project task and enabling us to complete it successfully.

S.JOHITHA (212W1A0591)

J.SUDARSHAN (212W1A0545)

G.EBINAGER RAJU (212W1A0536)

CHVIJAY (222W5A0502)

K.VAISHNAVI(212W1A0550)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

S.NO	SYMBOL	NAME
S.NO	SYMBOL/TERM	NAME / DESCRIPTION
1	BEV	Blockchain-enabled E-Voting
2	ABVS	Auditable Blockchain Voting System
3	AV Rule	Approval Voting Rule (used in Aqua system)
4	LWE	Learning With Errors (cryptographic assumption)
5	DFD	Data Flow Diagram
6	ER or E-R	Entity-Relationship (Diagram)
7	HTML	HyperText Markup Language
8	CSS	Cascading Style Sheets
9	UI	User Interface
10	MITM	Man-In-The-Middle (Attack)
11	SQL	Structured Query Language
12	XSS	Cross-Site Scripting
13	VPN	Virtual Private Network
14	OS	Operating System
15	DB	Database
16	UML	Unified Modeling Language (Diagram)
17	API	Application Programming Interface

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPANSION
BEV	Blockchain-enabled E-Voting
ABVS	Auditable Blockchain Voting System
AV	Approval Voting
LWE	Learning With Errors
DFD	Data Flow Diagram
ER	Entity-Relationship
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
UI	User Interface
MITM	Man-In-The-Middle
SQL	Structured Query Language
XSS	Cross-Site Scripting
VPN	Virtual Private Network
OS	Operating System
DB	Database
UML	Unified Modeling Language
API	Application Programming Interface
ID	Identity Document

ABSTRACT

The evolution of democratic systems has emphasized the critical importance of secure and fair elections, a practice deeply rooted in history since ancient Greece. Traditional voting methods, typically requiring physical presence at a polling station, face challenges such as tampering and security vulnerabilities. To address these issues, we propose a secure online voting system based on Blockchain technology, incorporating homomorphic encryption and hashing techniques to ensure the integrity and privacy of each vote. The system employs smart contracts, which activate when an election is scheduled, ensuring an efficient, tamper-resistant voting process. With an ever-growing voter base, modernizing voting infrastructure to enhance security and maintain fairness has become essential. Our approach decentralizes e-voting platform management across blockchains, leveraging a multi-winner approval voting system called Aqua on the Ethereum blockchain. This solution was rigorously tested and compared on public and private blockchains, demonstrating reliable performance at a relatively low speed while ensuring complete data privacy through homomorphic encryption.

Keywords: Python full-stack project, blockchain-based voting, homomorphic encryption, secure online voting, smart contracts, multi-winner approval voting, decentralized e-voting platform, election integrity, voter privacy

CHAPTER 1

INTRODUCTION

1. 1 OVERVIEW OF THE PROJECT

The topic of e-voting systems is still at an early stage of development. We have chosen this domain not only for its recency but also because there are not many solutions that address problems of e-voting. Nowadays, popularity grows also in the development of e-Government. However, such a system is not feasible if basic services for citizens such as elections do not become electronic. "E-voting is one of the key public sectors that can be transformed by blockchain technology". Hand by hand with e-voting come also new challenges, which need to be addressed. One of them is e.g. securing the elections, which needs to be at least as safe as the classic voting systems with ballots. That is why we have decided to create safe elections in which voters do not have to worry about someone abusing the electoral system. In recent years blockchain is often mentioned as an example of secure technology used in an online environment. Our e-voting system uses blockchain to manage all election processes. Its main advantage is that there is no need for confidence in the centralized authority that created the elections. This authority cannot affect the election results in our system. Another challenge in e-voting is the lack of transparency in the functioning of the system, leading to a lack of confidence in voters. This problem is solved by blockchain in a way of total transparency that allows everyone to see the stored data and processes such as how these data are handled. In the field of security, this technology is more suitable in every way than the classic e-voting platform without blockchain.

1.2 PROJECT ANALYSIS

Some forms of voting have been here ever since. Mostly used form all over the world are paper ballots. Electronic voting schemes are being popular only in the last decade and they are still unsolved. E-voting schemes bring problems mainly regarding security, credibility, transparency, reliability, and functionality. Estonia is the pioneer in this field and may be considered the state of the art. But there are only a few solutions using blockchain. Blockchain can deliver an answer to all of the mentioned problems and furthermore bring some advantages such as immutability and decentralization. The main problems of technologies utilizing blockchain for e-voting are their focus on only one field or lack of testing and comparison. In this paper, we present a blockchainbased e-voting platform, which can be used for any kind of voting.

It is fully utilized by blockchain and all processes can be handled within it. After the start of the voting, the platform behaves as fully independent and decentralized without possibilities to affect the voting process. The data are fully transparent, but the identity of voters is secured by homomorphic encryption. We have tested and compared our solution in three different blockchains. The results show, that both public and private blockchains can be used with only a little difference in the speed. The key novelty of our solution is a fully decentralized management of e-voting platform through blockchain, transparency of the whole process and at the same time security and privacy of the voters thanks to homomorphic encryption.

CHAPTER 2

SYSTEM ANALYSIS

2.1 Existing System

The traditional voting system relies on paper ballots or electronic voting machines, requiring voters to physically visit polling stations to cast their votes. While this method has been widely used, it presents several challenges, including:

- **Security Concerns**: There have been instances of vote tampering, ballot stuffing, and election fraud, undermining the credibility of elections.
- **Lack of Transparency**: Voters have limited visibility into how their votes are counted, making the process less transparent.
- **Voter Accessibility Issues**: Individuals with disabilities, elderly voters, and those living in remote areas face difficulties in accessing polling stations.
- **High Costs and Inefficiency**: Conducting elections involves substantial costs related to logistics, security, and manpower.
- **Slow Counting Process**: Traditional vote counting takes time, often leading to delays in announcing results.

Due to these drawbacks, there is a growing need for a more secure, transparent, and efficient voting system.

2.2 Proposed System

To overcome the limitations of the existing system, we propose a **blockchain-based online voting system** that ensures security, transparency, and efficiency in the electoral process.

Key Features of the Proposed System:

1. **Blockchain-Based Security**
    - Each vote is stored on the Ethereum blockchain using cryptographic encryption and hashing techniques.
    - Votes are immutable and tamper-proof, ensuring election integrity.
2. **Smart Contracts for Automation**
    - A smart contract is deployed on the blockchain to manage the voting process automatically.
    - Ensures that only eligible voters can cast their votes and that each voter can vote only once.

3. **Multi-Winner Approval Voting (Aqua System)**
    - The proposed system implements **Aqua**, a multi-winner approval-based voting scheme that follows the **AV rule** (Approval Voting rule).
    - Supports elections with committee sizes of up to three members.
3. **Biometric Authentication for Voter Verification**
    - Voter identity verification is performed using **face recognition** technology before allowing access to the voting system.
4. **Decentralized and Transparent Process**
    - Since all transactions (votes) are recorded on the blockchain, the system ensures transparency, preventing unauthorized modifications.
    - Voters can verify their votes without revealing their identity.
5. **User-Friendly and Accessible**
    - The voting system is web-based, making it accessible from any device with an internet connection.
    - The interface supports multiple languages, including **English and Greek**, to accommodate diverse voters.
6. **Cost-Effective and Efficient**
    - Eliminates the need for physical polling stations and reduces administrative costs.
    - Provides real-time vote counting, reducing result delays.

2.3 Module Description

The proposed system consists of three primary modules: **Admin Module, User Module, and Mechanic Module**. Each module has distinct functionalities designed to ensure secure, efficient, and transparent operations within the system.

2.3.1 Admin Module

The **Admin Module** is responsible for managing the system, ensuring security, and overseeing user interactions. The key functionalities include:

- **User Management**
    - Approves and manages registered users (voters, mechanics, etc.).
    - Assigns roles and permissions to users.
    - Monitors suspicious activities within the system.
- **Election Management** (if applicable)
    - Creates and schedules elections.

- Configures blockchain-based smart contracts for automated election processes.
- Manages candidate registration and vote validation.

- **System Security & Monitoring**
  - Ensures that transactions are securely recorded on the **Ethereum blockchain**.
  - Logs activities for auditing purposes.
  - Implements **access control** to prevent unauthorized actions.
- **Dashboard & Reports**
  - Provides real-time statistics on system activities.
  - Generates reports on voting results, user engagement, and security incidents.

2.3.2 User Module

The **User Module** allows individuals to interact with the system, whether as voters (in the case of an election system) or as general users.

- **User Registration & Authentication**
  - Users register with unique credentials and undergo identity verification.
  - Biometric authentication (face recognition or fingerprint) is used for secure access.
- **Voting (if applicable)**
  - Users can cast votes securely using **blockchain smart contracts**.
  - Votes are encrypted and stored immutably on the blockchain.
  - Users receive a **vote confirmation token** to verify their submission.
- **Profile & History**
  - Users can update their profiles and view past transactions or interactions.
  - Provides a **secure wallet system** (if blockchain transactions are involved).
- **Support & Helpdesk**
  - Users can report issues, seek assistance, or raise complaints.

2.3.3 Mechanic Module (if applicable)

If your system includes **service request management** (such as mechanics for vehicle repairs or technicians for maintenance), this module manages mechanics' roles and responsibilities.

- **Mechanic Registration & Authentication**
  - Mechanics sign up with verified credentials.
  - Admin approves their registration based on certifications and background checks.
- **Job Management**
  - Accept and manage assigned repair or service requests.

    - Update the status of service completion.
- **Payment & Reviews**

    - Users can rate and review mechanics based on service quality.
    - Secure payment integration for service charges.
- **Real-Time Tracking (if applicable)**

CHAPTER 3

LITERATURE SURVEY

Blockchain-enabled e-voting (BEV) could reduce voter fraud and increase voter access. Eligible voters cast a ballot anonymously using a computer or Smartphone. BEV uses an encrypted key and tamper-proof personal IDs. This article highlights some BEV implementations and the approach's potential benefits and challenges.

“Voting Process with Block-chain Technology: Auditable Block-chain Voting System,”

There are various methods and approaches to electronic voting all around the world. Each is connected with different benefits and issues. One of the most important and prevalent problems is lack of auditing capabilities and system verification methods. Blockchain technology, which recently gained a lot of attention, can provide a solution to this issue. This paper presents Auditable Blockchain Voting System (ABVS), which describes e-voting processes and components of a supervised internet voting system that is audit and verification capable. ABVS achieves this through utilization of blockchain technology and voter-verified paper audit trail.

“Bitcoin: A Peer-to-Peer Electronic Cash System,”

A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

“A Smart Contract for Boardroom Voting with Maximum Voter Privacy,”

We present the first implementation of a decentralized and self-tallying internet voting protocol with maximum voter privacy using the Blockchain. The Open Vote Network is suitable for boardroom elections and is written as a smart contract for Ethereum. Unlike previously proposed Blockchain e-voting protocols, this is the first implementation that does not rely on any trusted authority to compute the tally or to protect the voter's privacy. Instead, the Open Vote Network is a selftallying protocol, and each voter is in control of the privacy of their own vote such that it can only be breached by a full collusion involving all other voters. The execution of the protocol is enforced using the consensus mechanism that also secures the Ethereum blockchain. We tested the implementation on Ethereum's official test network to demonstrate its feasibility. Also, we provide a financial and computational breakdown of its execution cost.

"Efficient Fully Homomorphic Encryption from (Standard) LWE,"

We present a fully homomorphic encryption scheme that is based solely on the (standard) learning with errors (LWE) assumption. Applying known results on LWE, the security of our scheme is based on the worst-case hardness of "short vector problems" on arbitrary lattices. Our construction improves on previous works in two aspects: 1) we show that "somewhat homomorphic" encryption can be based on LWE, using a new re-linearization technique. In contrast, all previous schemes relied on complexity assumptions related to ideals in various rings. 2) We deviate from the "squashing paradigm" used in all previous works. We introduce a new dimension-modulus reduction technique, which shortens the ciphertexts and reduces the decryption complexity of our scheme, without introducing additional assumptions. Our scheme has very short ciphertexts and we therefore use it to construct an asymptotically efficient LWE-based single-server private information retrieval (PIR) protocol. The communication complexity of our protocol (in the public-key model) is $k \cdot \text{polylog}(k) + \log |DB|$ bits per single-bit query (here, A; is a security parameter).

and properties of zero knowledge proof systems,"In this paper we investigate some properties of zero-knowledge proofs, a notion introduced by Goldwasser, Micali, and Rackoff. We introduce and classify two definitions of zero-knowledge: auxiliary-input zero-knowledge and blackbox-simulation zero-knowledge. We explain why auxiliary-input zero-knowledge is a definition more suitable for cryptographic applications than the original [GMR1] definition. In particular, we show that any protocol solely composed of subprotocols which are auxiliary-input zero-knowledge is itself auxiliary-input zero-knowledge. We show that blackbox-simulation zero-knowledge implies auxiliary-input zero-knowledge (which in turn implies the [GMR1] definition). We argue that all known zero-knowledge proofs are in fact blackbox-simulation zero-knowledge (i.e., we proved zero-knowledge using blackbox-simulation of the verifier). As a result, all known zero-knowledge proof systems are shown to be auxiliary-input zero-knowledge and can be used for cryptographic applications such as those in [GMW2]. We demonstrate the triviality of certain classes of zero-knowledge proof systems, in the sense that only languages in BPP have zero-knowledge proofs of these classes. In particular, we show that any language having a Las Vegas zero-knowledge proof system necessarily belongs to RP. We show that randomness of both the verifier and the prover, and nontriviality of the interaction are essential properties of (nontrivial) auxiliary-input zero-knowledge proofs.

"Ethereum: A secure decentralized generalized transaction ledger,"

The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, not the least being Bitcoin. Each such project can be seen as a simple application on a decentralized, but singleton, compute resource. We can call this paradigm a transactional singleton machine with shared-state. Ethereum implements this paradigm in a generalized manner. Furthermore it provides a plurality of such resources, each with a distinct state and operating code but able to interact through a message-passing framework with others. We discuss its design, implementation issues, the opportunities it provides and the future hurdles we envisage.

3.1 SYSTEM REQUIREMENTS

HARDWARE AND SOFTWARE REQUIREMENTS

3.1.1Hardware Requirements

Hardware	Specification
Processor	INTEL CORE i5
Hard Disk	500 GB
Ram	8.0GB

3.1.2Software Requirement

Purpose	Tools & Technology
Operating system	Windows11 OS
Front End	HTML, CSS
Back End	Back End
Data Base	MYSQL.
Framework	Django 5.0.3

3.2 Software Used

3.2.1 Python

Python technology refers to the use of the Python programming language and its associated libraries, frameworks, and tools for various software development purposes. Here's a brief overview:

1. Programming Language: Python is a high-level, interpreted programming language known for its simplicity and readability. It offers dynamic typing and automatic memory management, making it easy to learn and use.
2. Versatility: Python is a versatile language used for a wide range of applications, including web development, data analysis, machine learning, artificial intelligence, scientific computing, automation, and more. Its extensive standard library and vast ecosystem of third-party packages make it suitable for diverse use cases.
3. Libraries and Frameworks: Python boasts a rich ecosystem of libraries and frameworks that simplify and accelerate development in various domains. For web development, popular frameworks like Django and Flask provide robust tools for building web applications. For data analysis and scientific computing, libraries like NumPy, pandas, and SciPy offer powerful data manipulation and analysis capabilities. For machine learning and AI, libraries like TensorFlow, PyTorch, and scikit-learn provide advanced algorithms and tools.
4. Community and Support: Python has a large and active community of developers, contributors, and enthusiasts who collaborate, share knowledge, and provide support through online forums, user groups, conferences, and documentation. This vibrant community contributes to the continuous improvement and evolution of the language and its ecosystem.
5. Ease of Learning and Prototyping: Python's simple syntax and readability make it an excellent choice for beginners learning to code. Its interpreted nature allows for rapid prototyping and experimentation, facilitating the development process and reducing time to market for projects.
6. Cross-Platform Compatibility: Python is cross-platform, meaning it runs on various operating systems, including Windows, macOS, and Linux, without requiring modifications to the code. This portability enables developers to write code once and deploy it on multiple platforms.
7. Integration and Extensibility: Python seamlessly integrates with other languages and technologies, allowing developers to leverage existing codebases, libraries, and systems. It

supports integration with C/C++, Java, and other languages via bindings and APIs, enabling interoperability and extending its capabilities.

Overall, Python technology offers a powerful and flexible platform for building a wide range of software applications, from simple scripts to complex, mission-critical systems, across diverse industries and domains. Its simplicity, versatility, and rich ecosystem make it a popular choice among developers worldwide.

3.2.2 HTML

HTML, or Hyper Text Markup Language, is the standard markup language used to create and design web pages. It's the backbone of the World Wide Web, providing the structure and content for web documents. Here's a brief overview of HTML technology:

1. Markup Language: HTML uses a markup syntax to define the structure and content of web pages. It consists of a series of elements enclosed in tags, which specify how content should be displayed in a web browser.
2. Document Structure: HTML documents are structured as a hierarchy of elements, starting with the `<html>` tag, followed by the `<head>` and `<body>` sections. The `<head>` section contains metadata such as the document title and links to external resources, while the `<body>` section contains the main content of the page.
3. Elements and Tags: HTML elements represent different types of content, such as text, images, links, forms, and multimedia. Each element is enclosed in opening and closing tags, which define its beginning and end. For example, `<p>` tags are used to create paragraphs, `<img>` tags for images, and `<a>` tags for hyperlinks.
4. Attributes: HTML elements can have attributes that provide additional information or modify their behavior. Attributes are specified within the opening tag and include properties such as `src` for the source of an image, `href` for the destination of a hyperlink, and `class` or `id` for applying CSS styles or JavaScript functionality.
5. Semantic HTML: HTML5 introduced semantic elements that convey the meaning and structure of content more clearly to both browsers and developers. Examples include `<header>`, `<nav>`, `<main>`, `<section>`, `<article>`, and `<footer>`, which describe the purpose of each section of a web page.
6. Compatibility and Standards: HTML is governed by the World Wide Web Consortium (W3C), which defines standards and specifications for web technologies. Modern web browsers support

HTML5, the latest version of HTML, ensuring compatibility across different platforms and devices.

7. Styling and Layout: While HTML provides the structure and content of web pages, styling and layout are typically handled using Cascading Style Sheets (CSS). CSS allows developers to customize the appearance of HTML elements by specifying properties such as colors, fonts, margins, padding, and layout positioning.
8. Interactivity and Behavior: HTML can be enhanced with client-side scripting languages such as JavaScript to create interactive and dynamic web experiences. JavaScript enables features like form validation, animations, event handling, and AJAX for asynchronous communication with web servers.

Overall, HTML technology forms the foundation of web development, enabling the creation of static and dynamic web pages that can be accessed and viewed by users across the internet.

3.2.3 CSS

CSS (Cascading Style Sheets) is a fundamental technology used in web development to control the presentation and layout of web pages. It works alongside HTML (Hypertext Markup Language) to define how elements are displayed on a webpage. Here's a brief overview:

1. Styling Elements: CSS allows developers to apply styles to HTML elements, such as setting colors, fonts, margins, padding, borders, and backgrounds. This enables the customization of the appearance of text, images, buttons, and other elements on a webpage.
2. Selectors and Declarations: CSS uses selectors to target specific HTML elements and declarations to define the styles applied to those elements. Selectors can be based on element types, classes, IDs, attributes, or their relationships within the HTML structure.
3. Cascade and Specificity: CSS follows a cascading mechanism where styles can be inherited from parent elements and overridden by more specific styles or styles defined later in the document. Specificity determines which styles take precedence when multiple rules target the same element.
4. Layout Control: CSS provides powerful layout capabilities, including techniques for creating responsive designs that adapt to different screen sizes and devices. Developers can use properties like float, flexbox, grid, and positioning to control the layout of elements on the page.5. Media Queries: With CSS, developers can apply different styles based on various factors like the device's screen size, orientation, or resolution. Media queries allow for creating responsive designs by adjusting layout, typography, and other styles to optimize the user experience across

different devices.

6. Animations and Transitions: CSS supports animations and transitions, enabling developers to create interactive and visually engaging effects without relying on JavaScript. CSS animations can be used to animate properties such as size, position, color, and opacity, while transitions allow smooth changes between different states of an element.
7. Preprocessors and Postprocessors: CSS preprocessors like Sass and Less extend the capabilities of CSS by introducing features like variables, mixins, functions, and nesting, which enhance code organization and maintainability. Postprocessors like Autoprefixer automate vendor prefixing and optimize CSS code for better browser compatibility and performance.

Overall, CSS is a vital technology for web developers, allowing them to control the visual aspects of web pages and create attractive, user-friendly interfaces for a wide range of devices and screen sizes.

CHAPTER 4

IMPLEMENTATION

4.1 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

4.2 TYPES OF TESETING

4.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

4.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

4.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted. Invalid Input : identified classes of invalid input must be rejected. Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4.2.4 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

4.2.5 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

4.2.6 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a test in which the software under test is treated as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

4.2.7 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

4.2.8 TESTING RESULTS

All the test cases mentioned above passed successfully. No defects encountered.

4.3 TESTING METHODOLOGIES

The following are the Testing Methodologies:

- Unit Testing.
- Integration Testing.
- User Acceptance Testing.
- Output Testing.
- Validation Testing.

4.3.1 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually, and the module interfaces are verified for the consistency with design specification. All-important processing path are tested for the expected results. All error handling paths are also tested.

Unit testing in vehicle service management system involves testing individual units or components of the software system to ensure they function correctly in isolation. For on-road breakdown assistance, this could mean testing functions like user authentication, location tracking, service request handling, and notification systems separately to ensure each component works as expected. This helps identify and fix bugs early in the development process, improving the overall reliability and performance of the breakdown assistance system.

4.3.2 INTEGRATION TESTING

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

1) Top-Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual studs are replaced when the test proceeds downwards.

2) Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom-up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program.

The bottom-up approach tests each module individually and then each module is module is integrated with a main module and tested for functionality.

Integration testing in vehicle service management system involves testing how different components or systems within the breakdown assistance service work together. This includes testing the integration between the user interface, customer database and service provider network to ensure seamless coordination and functionality during a breakdown scenario.

4.3.3 USER ACCEPTANCE TESTING

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required.

4.3.4 OUTPUT TESTING

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

Output testing in vehicle service management system involves verifying that all the components and systems of a vehicle are functioning correctly after repair or maintenance. This ensures that the vehicle is safe to drive and will perform as expected when it's back on the road.

4.3.5 VALIDATION TESTING

Validation Checking

Validation checks are performed on the following fields.

Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it must perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testingshould be planned so that all the requirements are individually tested. A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data:

Live test data are those that are extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves. It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that would show how the system would perform for the typical processing requirement, if the live data entered is in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true system test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, makes possible the testing of all login and control paths through the program. The most effective test programs use artificial test data generated by people other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications. The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

Validation testing in vehicle service management system involves ensuring that the system accurately responds to and addresses various breakdown scenarios that drivers may encounter. This includes verifying that the system correctly identifies the location of the breakdown, dispatches assistance promptly, and communicates effectively with both the driver and the service provider.

CHAPTER 5

SYSTEM DESIGN

5.1 DATA FLOW DIAGRAM [DFD]:

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as the Structured Systems Analysis and Design Method (SSADM).

Advantages of DFD:

- It helps us to understand the functioning and the limits of a system.
- It is a graphical representation that is very easy to understand as it helps visualize contents.
- Data Flow Diagram represent detailed and well-explained diagram of system components.
- It is used as the part of system documentation file.
- Data Flow Diagrams can be understood by both technical and nontechnical person because they are very easy to understand

Disadvantages of DFD**:**

At times DFD can confuse the programmers regarding the system. Data Flow Diagram takes long time to be generated, and many times due to these reasons analysts are denied permission to work on its design.

5.2 E-R Diagram

An Entity-Relationship model (ER Model) describes the structure of a database with the help of a diagram, which is known as ENTITY RELATIONSHIP DIAGRAM (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: Entity set and Relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In term of DBMS, an entity is a table or attribute of a table in the database, so by showing relationship among the tables and their attributes, ER diagram shows the complete logical structure of a database**.**

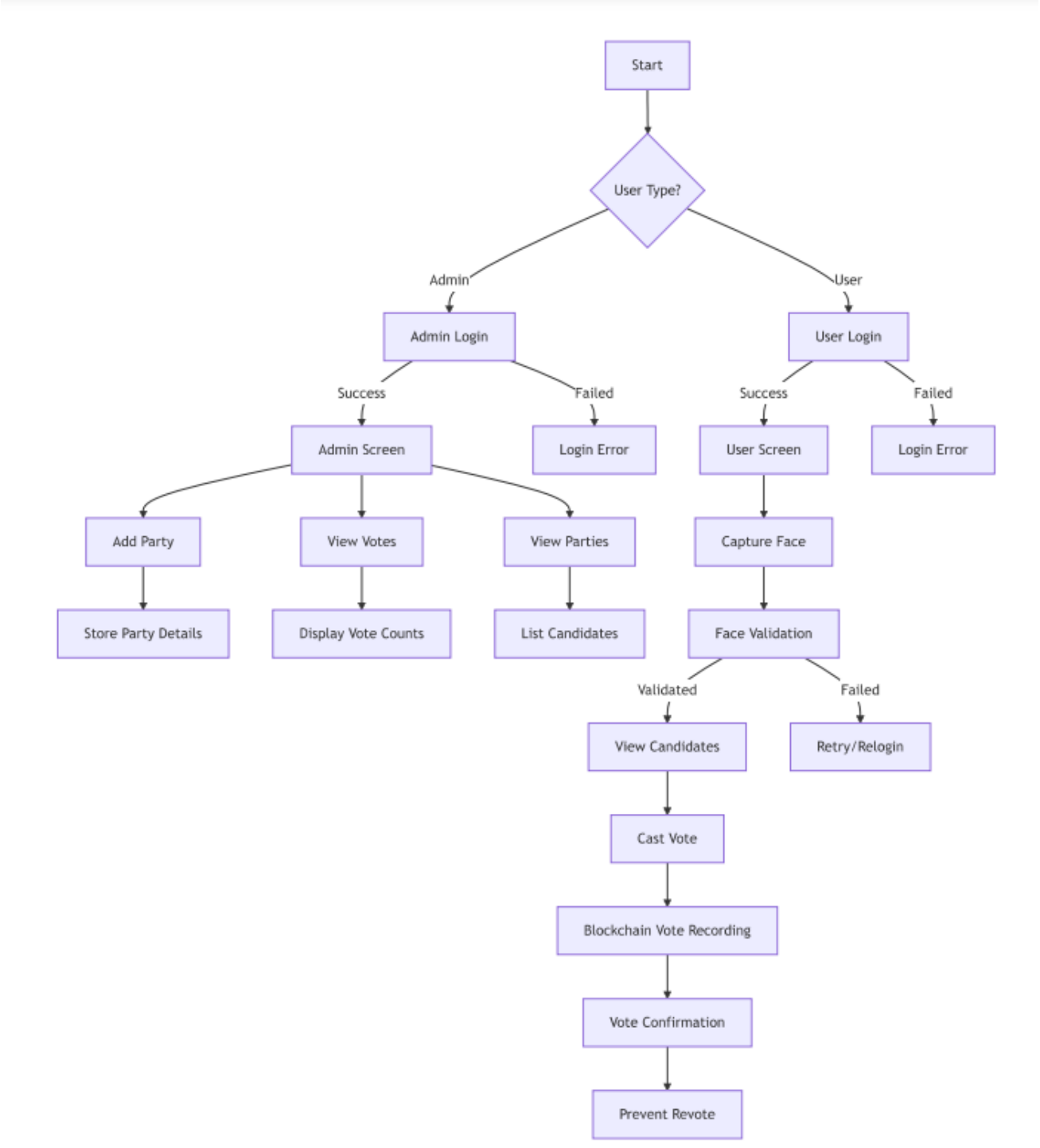
Advantages of ER model:

- Conceptually it is very simple: ER model is very simple because if we know relationship between entities and attributes, then we can easily draw an ER Diagram.
- Better visual representation: ER model is a diagrammatic representation of any logical structure of database. By seeing ER diagram, we can easily understand relationship among entities and relationship.
- Highly integrated with relational model: ER model can be easily converted into relational model by simply converting ER model into tables.
- Easy conversion to any data model: ER model can be easily converted into another data model like hierarchical data model, network model and so on.

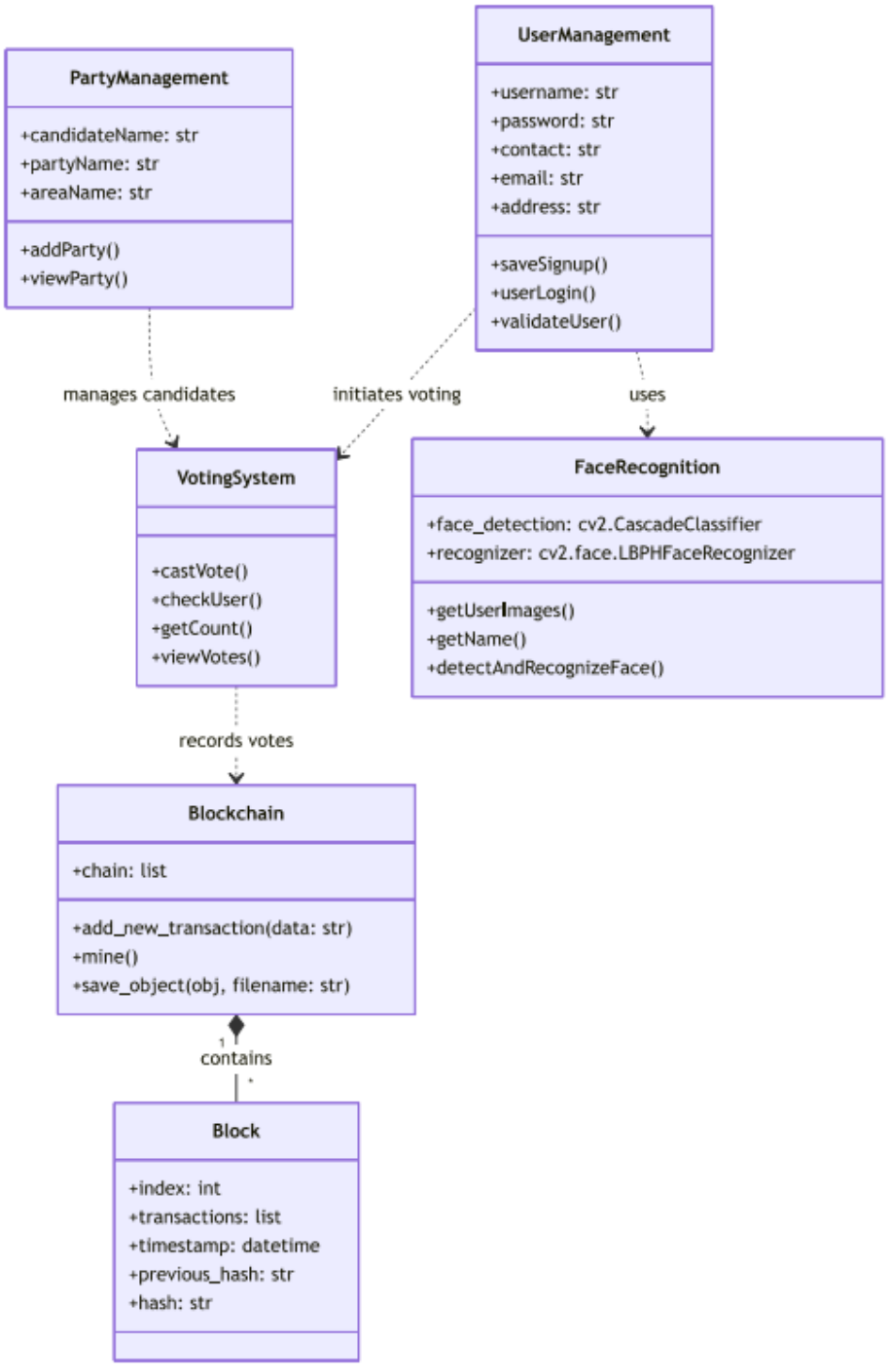
Disadvantages of ER model:

- Loss of information content: Some information be lost or hidden in ER model.
- Limited relationship representation: ER model represents limited relationship as compared to another data models like relational model etc.
- No representation of data manipulation: It is difficult to show data manipulation
- in ER model.

Flow chart diagram:



UML Diagram:


PartyManagement
+candidateName: str
+partyName: str
+areaName: str
+addParty()
+viewParty()
UserManagement
+username: str
+password: str
+contact: str
+email: str
+address: str
+saveSignup()
+userLogin()
+validateUser()
manages candidates
initiates voting
uses
VotingSystem
+castVote()
+checkUser()
+getCount()
+viewVotes()
FaceRecognition
+face_detection: cv2.CascadeClassifier
+recognizer: cv2.face.LBPHFaceRecognizer
+getUserImages()
+getName()
+detectAndRecognizeFace()
records votes
Blockchain
+chain: list
+add_new_transaction(data: str)
+mine()
+save_object(obj, filename: str)
1
contains
*
Block
+index: int
+transactions: list
+timestamp: datetime
+previous_hash: str
+hash: str

CHAPTER 6

TESTING:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the
Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS
Unit testing
Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application
.it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expectedresults.

Integration testing
Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination ofcomponents.

Functional test
Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and usermanuals.
Functional testing is centered on the following items:
Valid Input : identified classes of valid input must be accepted. InvalidInput : identified classes of invalid input must be rejected. Functions : identified functions must beexercised.
Output : identified classes of application outputs must be exercised. Systems/Procedures: interfacing systems or procedures must beinvoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test
System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing
White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black boxlevel.

Black Box Testing
Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, asablackbox.youcannot—seeintoit.Thetestprovidesinputsandrespondsto outputswithout considering how the softwareworks.

6.1 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and usermanuals.
Functional testing is centered on the following items:
Valid Input : identified classes of valid input must be accepted. InvalidInput : identified classes of invalid input must be rejected. Functions : identified functions must beexercised.

6.2 SECURITY TESTING

Security testing was conducted to ensure the system is resistant to malicious activities and vulnerabilities. The following areas were evaluated:

Data Privacy: Homomorphic encryption protected vote confidentiality, ensuring data is not decrypted during processing.

Blockchain Integrity: Tamper-proof records on the Ethereum blockchain secured against data manipulation.

Authentication & Authorization: Secure login mechanisms with role-based access control prevented unauthorized access.

Smart Contract Auditing: Checked for vulnerabilities such as reentrancy, overflow, and denial of service (DoS) within deployed contracts.

Protection Against Common Attacks: The system was tested against SQL injection, cross-site scripting (XSS), and man-in-the-middle (MITM) attacks.

Result: No major vulnerabilities were found. The platform is deemed secure for online voting operations.

6.3 PERFORMANCE TESTING

Performance testing evaluates how the system behaves under load and stress. The focus was on speed, reliability, and scalability of the application:

Load Testing: Simulated multiple users casting votes simultaneously to assess the system's throughput.

Transaction Speed: Measured the time taken to process votes on both public and private Ethereum blockchains.

Scalability: Assessed how well the system handles a growing number of voters and elections.

Smart Contract Latency: Measured delays in contract execution and result computation.

Result: The system handled moderate user loads effectively but showed slower performance on the public blockchain due to network congestion. Private blockchain setups yielded better speed and responsiveness.

6.4 USABILITY TESTING

Usability testing ensures that the system is intuitive and user-friendly. Feedback was collected from a group of test users, focusing on the following aspects:

User Interface (UI): Evaluated clarity, design consistency, and navigation ease.

Accessibility: Checked compatibility with screen readers and color contrast for visually impaired users.

Voting Flow: Measured how easily a user can complete the voting process without external assistance.

Error Handling: Tested clarity and helpfulness of error messages and prompts.

Result: Users found the platform easy to navigate and the voting process simple. Minor UI improvements were recommended to enhance accessibility and mobile responsiveness.

Chapter 7

Source code:

Index.html:

```
{% load static %}
<html>
<head>
<title>E-Voting</title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="{% static 'style.css' %}"/>
</head>
<body>
<div id="wrapper">
   <div id="header">
      <div id="logo">
         <center><font size="4" color="Black">Blockchain E-Voting Done Right: Privacy and
Transparency with Public Blockchain</font></center>
      </div>
      <div id="slogan">

      </div>
   </div>
   <div id="menu">
      <ul><center>
    <li><a href="{% url 'index' %}">Home</a></li>
    <li><a href="{% url 'Admin' %}">Admin Login</a></li>
    <li><a href="{% url 'Login' %}">User Login</a></li>
    <li><a href="{% url 'Register' %}">Register Here</a></li>

      </center></ul>
      <br class="clearfix" />
   </div>
   <div id="splash">
```

```
        <img class="pic" src="{% static 'images/vote.jpeg' %}" width="870" height="230" alt="" />
    </div>
    <br/>
    <p align="justify"><font size="3" style="font-family: Comic Sans MS">
        Blockchain E-Voting Done Right: Privacy and Transparency with Public Blockchain

</p>
</body>
</html>
```

Login.html

```
{% load static %}
<html>
<head>
<title>E-Voting</title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="{% static 'style.css' %}"/>
 <script LANGUAGE="Javascript" >
function validate(){
    var x=document.forms["f1"]["username"].value;
    var y=document.forms["f1"]["password"].value;

    if(x == null || x==""){
        window.alert("Username must be enter");
        document.f1.username.focus();
        return false;
    }
    if(y == null || y==""){
        window.alert("Password must be enter");
        document.f1.password.focus();
        return false;
    }
    return true;
```

```
}
   </script>
</head>
<body>
<div id="wrapper">
   <div id="header">
      <div id="logo">
         <center><font size="4" color="yellow">Blockchain E-Voting Done Right: Privacy and
Transparency with Public Blockchain</font></center>
      </div>
      <div id="slogan">

      </div>
   </div>
   <div id="menu">
      <ul><center>
     <li><a href="{% url 'index' %}">Home</a></li>
     <li><a href="{% url 'Admin' %}">Admin Login</a></li>
     <li><a href="{% url 'Login' %}">User Login</a></li>
     <li><a href="{% url 'Register' %}">Register Here</a></li>

      </center></ul>
<br class="clearfix" />
            </div>

   <div id="splash">
      <img class="pic" src="{% static 'images/Voter-Registration.jpg' %}" width="870" height="230"
alt="" />
   </div>
         <center>
<form name="f1" method="post" action={% url 'UserLogin' %} OnSubmit="return validate()">
   {% csrf_token %}<br/>
  <h3><b>User Login Screen</b></h3>
```

```
   <font size="" color="black"><center>{{ data }}</center></font>


                <table align="center" width="80" >
          <tr><td><font size="3" color="black">Username</b></td><td><input type="text"
name="username" style="font-family: Comic Sans MS" size="30"/></td></tr>



      <tr><td><font size="3" color="black">Password</b></td><td><input type="password"
name="password" style="font-family: Comic Sans MS" size="30"/></td></tr>

          <tr><td></td><td><input type="submit" value="Login">
          </td>
          </table>
          <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
             </div>


             </div>

   </body>
</html>
```

Register.hmtl:

```
{% load static %}
<html>
<head>
<title>E-Voting</title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="{% static 'style.css' %}"/>
 <script LANGUAGE="Javascript" >
function validate(){
   var x=document.forms["f1"]["username"].value;
   var y=document.forms["f1"]["password"].value;
   var c=document.forms["f1"]["contact"].value;
   var e=document.forms["f1"]["email"].value;
```

```
var a=document.forms["f1"]["address"].value;
var image=document.forms["f1"]["image"].value;
if(x == null || x==""){
    window.alert("Username must be enter");
    document.f1.username.focus();
    return false;
}
if(y == null || y==""){
    window.alert("Password must be enter");
    document.f1.password.focus();
    return false;
}
if(c == null || c==""){
    window.alert("Contact No must be enter");
    document.f1.contact.focus();
    return false;
}
if(isNaN(c)){
    window.alert("Please enter valid contact number");
    document.f1.contact.focus();
    return false;
}
if(e == null || e==""){
    window.alert("Email ID must be enter");
    document.f1.email.focus();
    return false;
}
var filter = /^([a-zA-Z0-9_\.\-])+\@(gmail+\.)+(com)+$/;
if (!filter.test(e)) {
    window.alert('enter a valid email address');
    document.f1.email.focus();
    return false;
}
if(a == null || a==""){
```

```
        window.alert("Address must be enter");
        document.f1.address.focus();
        return false;
    }

    return true;
}
</script>

</head>
<body>
<div id="wrapper">
    <div id="header">
        <div id="logo">
            <center><font size="4" color="yellow">Blockchain E-Voting Done Right: Privacy and
Transparency with Public Blockchain</font></center>
        </div>
        <div id="slogan">

        </div>
    </div>
    <div id="menu">
        <ul><center>
      <li><a href="{% url 'index' %}">Home</a></li>
      <li><a href="{% url 'Admin' %}">Admin Login</a></li>
      <li><a href="{% url 'Login' %}">User Login</a></li>
      <li><a href="{% url 'Register' %}">Register Here</a></li>

        </center></ul>
<br class="clearfix" />
            </div>

    <div id="splash">
        <img class="pic" src="{% static 'images/registerhere.jpg' %}" width="870" height="230" alt="" />
```

```
</div>
        <center>
<form name="f1" method="post" action={% url 'Signup' %} OnSubmit="return validate()">
 {% csrf_token %}<br/>
<h3><b>New User Signup Screen</b></h3>

 <font size="" color="black"><center>{{ data }}</center></font>

                <table align="center" width="80" >
       <tr><td><font size="3" color="black">Username</b></td><td><input type="text"
name="username" style="font-family: Comic Sans MS" size="30"/></td></tr>


    <tr><td><font size="3" color="black">Password</b></td><td><input type="password"
name="password" style="font-family: Comic Sans MS" size="30"/></td></tr>


    <tr><td><font size="3" color="black">Contact No</b></td><td><input type="text"
name="contact" style="font-family: Comic Sans MS" size="20"/></td></tr>

    <tr><td><font size="3" color="black">Email ID</b></td><td><input type="text"
name="email" style="font-family: Comic Sans MS" size="40"/></td></tr>

    <tr><td><font size="3" color="black">Address</b></td><td><input type="text" name="address"
style="font-family: Comic Sans MS" size="60"/></td></tr>
        <tr><td></td><td><input type="submit" value="Register">
        </td>
        </table>
        <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
          </div>

          </div>

 </body>
```

```
</html>
```

Viewparty.html

```
{% load static %}
<html>
<head>
<title>E-Voting</title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="{% static 'style.css' %}"/>
</head>
<body>
<div id="wrapper">
   <div id="header">
      <div id="logo">
         <center><font size="4" color="yellow">Blockchain E-Voting Done Right: Privacy and
Transparency with Public Blockchain</font></center>
      </div>
      <div id="slogan">

      </div>
   </div>
   <div id="menu">
      <ul><center>
    <li><a href="{% url 'AddParty' %}">Add Party Details</a></li>
    <li><a href="{% url 'ViewParty' %}">View Party Details</a></li>
    <li><a href="{% url 'ViewVotes' %}">View Votes</a></li>
    <li><a href="{% url 'index' %}">Logout</a></li>
      </center></ul>
      <br class="clearfix" />
   </div>
   <div id="splash">
      <img class="pic" src="{% static 'images/investor.png' %}" width="870" height="230" alt="" />
   </div>
   <br/>
   {{ data|safe }}
```

```
</body>
</html>
```

Casteyourvote.html

```
{% load static %}
<!doctype html>

<html lang="en">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <title>E-Voting</title>
   <style type="text/css">
      body { font-family: Helvetica, sans-serif; }
      h2, h3 { margin-top:0; }
      form { margin-top: 15px; }
      form > input { margin-right: 15px; }
      #results { float:right; margin:20px; padding:20px; border:1px solid; background:#ccc; }
   </style>
   <script type="text/javascript" src="{% static 'webcam.min.js' %}"></script>
   <script language="JavaScript">
      function take_snapshot() {
         // take snapshot and get image data
         Webcam.snap( function(data_uri) {
            // display results in page
            document.getElementById('results').innerHTML =
               '<h2>Here is your image:</h2>' +
               '<img src="'+data_uri+'"/>';
               var request = new XMLHttpRequest();
request.open("GET", "http://127.0.0.1:8000/WebCam?mytext="+data_uri);
//window.alert(data_uri)
request.onreadystatechange = function() {
if(this.readyState === 4 && this.status === 200) {
```

```
        data = this.responseText
        document.getElementById("sr").innerHTML = data;
        }
    };
    request.send();

            } );
        }

    </script>
</head>
<body>
    <div id="results">Your captured image will appear here...</div>

    <h1>E-Voting</h1>
    <h3>Demonstrates simple 320x240 capture &amp;amp; display</h3>

    <div id="my_camera"></div>

    <!-- First, include the Webcam.js JavaScript Library -->


    <!-- Configure a few settings and attach camera -->
    <script language="JavaScript">
        Webcam.set({
            width: 320,
            height: 240,
            image_format: 'jpeg',
            jpeg_quality: 90
        });
        Webcam.attach( '#my_camera' );
    </script>

    <!-- A button for taking snaps -->
```

```
   <form>
      <input type=button value="Take Snapshot" onClick="take_snapshot()">
      <div id="sr">Server Response</div>
   </form>
<form  name="f1" method="post" action="{% url 'ValidateUser' %}">
<br/>
        {% csrf_token %}<br/>
<input type="submit" value="Validate User"></input>
</form>

</body>
</html>
```

Views.py

```
from django.shortcuts import render
from django.template import RequestContext
from django.contrib import messages
import pymysql
from django.http import HttpResponse
from django.core.files.storage import FileSystemStorage
import os
from Blockchain import *
from Block import *
from datetime import date
import cv2
import numpy as np
import base64
import random
from datetime import datetime
from PIL import Image



global username, password, contact, email, address

blockchain = Blockchain()
```

```
if os.path.exists('blockchain_contract.txt'):
    with open('blockchain_contract.txt', 'rb') as fileinput:
        blockchain = pickle.load(fileinput)
    fileinput.close()

face_detection = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
recognizer = cv2.face.LBPHFaceRecognizer_create()

def AddParty(request):
    if request.method == 'GET':
        return render(request, 'AddParty.html', {})

def index(request):
    if request.method == 'GET':
        return render(request, 'index.html', {})

def Login(request):
    if request.method == 'GET':
        return render(request, 'Login.html', {})

def CastVote(request):
    if request.method == 'GET':
        return render(request, 'CastVote.html', {})



def Register(request):
    if request.method == 'GET':
        return render(request, 'Register.html', {})

def Admin(request):
    if request.method == 'GET':
        return render(request, 'Admin.html', {})

def WebCam(request):
```

```
    if request.method == 'GET':
        data = str(request)
        formats, imgstr = data.split(';base64,')
        imgstr = imgstr[0:(len(imgstr)-2)]
        data = base64.b64decode(imgstr)
        if os.path.exists("EVotingApp/static/photo/test.png"):
            os.remove("EVotingApp/static/photo/test.png")
        with open('EVotingApp/static/photo/test.png', 'wb') as f:
            f.write(data)
        f.close()
        context= {'data':"done"}
        return HttpResponse("Image saved")

def checkUser(name):
    flag = 0
    for i in range(len(blockchain.chain)):
        if i > 0:
            b = blockchain.chain[i]
            data = b.transactions[0]
            print(data)
            arr = data.split("#")
            if arr[0] == name:
                flag = 1
                break
    return flag

def getOutput(status):
    output = '<h3><br/>'+status+'<br/><table border=1 align=center>'
    output+='<tr><th><font size=3 color=black>Candidate Name</font></th>'
    output+='<th><font size=3 color=black>Party Name</font></th>'
    output+='<th><font size=3 color=black>Area Name</font></th>'
    output+='<th><font size=3 color=black>Image</font></th>'
    output+='<th><font size=3 color=black>Cast Vote Here</font></th></tr>'
```

```
    con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database =
'evoting',charset='utf8')
    with con:
        cur = con.cursor()
        cur.execute("select * FROM addparty")
        rows = cur.fetchall()
        for row in rows:
            cname = row[0]
            pname = str(row[1])
            area = row[2]
            image = row[3]
            output+='<tr><td><font size=3 color=black>'+cname+'</font></td>'
            output+='<td><font size=3 color=black>'+pname+'</font></td>'
            output+='<td><font size=3 color=black>'+area+'</font></td>'
            output+='<td><img src="/static/parties/'+cname+'.png" width=200 height=200></img></td>'
            output+='<td><a href="FinishVote?id='+cname+'"><font size=3 color=black>Click
Here</font></a></td></tr>'
    output+="</table><br/><br/><br/><br/><br/>"
    return output


def FinishVote(request):
    if request.method == 'GET':
        global username
        cname = request.GET.get('id', False)
        voter = ''
        today = date.today()
        data = str(username)+"#"+str(cname)+"#"+str(today)
        blockchain.add_new_transaction(data)
        hash = blockchain.mine()
        b = blockchain.chain[len(blockchain.chain)-1]
        print("Previous Hash : "+str(b.previous_hash)+" Block No : "+str(b.index)+" Current Hash :
"+str(b.hash))
```

```
        bc = "Previous Hash : "+str(b.previous_hash)+"<br/>Block No : "+str(b.index)+"<br/>Current Hash
: "+str(b.hash)
        blockchain.save_object(blockchain,'blockchain_contract.txt')
        context= {'data':'<font size=3 color=black>Your Vote Accepted<br/>'+bc}
        return render(request, 'UserScreen.html', context)

def getUserImages():
    names = []
    ids = []
    faces = []
    dataset = "EVotingApp/static/profiles"
    count = 0
    for root, dirs, directory in os.walk(dataset):
        for j in range(len(directory)):
            pilImage = Image.open(root+"/"+directory[j]).convert('L')
            imageNp = np.array(pilImage,'uint8')
            name = os.path.splitext(directory[j])[0]
            names.append(name)
            faces.append(imageNp)
            ids.append(count)
            count = count + 1
    print(str(names)+" "+str(ids))
    return names, ids, faces

def getName(predict, ids, names):
    name = "Unable to get name"
    for i in range(len(ids)):
        if ids[i] == predict:
            name = names[i]
            break
    return name

def ValidateUser(request):
    if request.method == 'POST':
```

```
    global username
    status = "unable to predict user"
    img = cv2.imread('EVotingApp/static/photo/test.png')
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    face_component = None
    faces =
face_detection.detectMultiScale(img,scaleFactor=1.1,minNeighbors=5,minSize=(30,30),flags=cv2.CASC
ADE_SCALE_IMAGE)
    status = "Unable to predict.Please retry"
    #for (x, y, w, h) in faces:
    #    face_component = gray[y:y+h, x:x+w]
    faces = sorted(faces, reverse=True,key=lambda x: (x[2] - x[0]) * (x[3] - x[1]))[0]
    (fX, fY, fW, fH) = faces
    face_component = gray[fY:fY + fH, fX:fX + fW]
    if face_component is not None:
        names, ids, faces = getUserImages()
        recognizer.train(faces, np.asarray(ids))
        predict, conf = recognizer.predict(face_component)
        print(str(predict)+" === "+str(conf))
        if(conf < 80):
            validate_user = getName(predict, ids, names)
            print(str(validate_user)+" "+str(username))
            if validate_user == username:
                status = "success"
    else:
        status = "Unable to detect face"
    if status == "success":
        flag = checkUser(username)
        if flag == 0:
            output = getOutput("User predicted as : "+username+"<br/><br/>")
            context= {'data':output}
            return render(request, 'VotePage.html', context)
        else:
            context= {'data':"You already casted your vote"}
```

```
            return render(request, 'UserScreen.html', context)
        else:
            context= {'data':status}
            return render(request, 'UserScreen.html', context)



def getCount(name):
    count = 0
    for i in range(len(blockchain.chain)):
        if i > 0:
            b = blockchain.chain[i]
            data = b.transactions[0]
            arr = data.split("#")
            if arr[1] == name:
                count = count + 1
                break
    return count

def ViewVotes(request):
    if request.method == 'GET':
        output = '<table border=1 align=center >'
        output+='<tr><th><font size=3 color=black>Candidate Name</font></th>'
        output+='<th><font size=3 color=black>Party Name</font></th>'
        output+='<th><font size=3 color=black>Area Name</font></th>'
        output+='<th><font size=3 color=black>Image</font></th>'
        output+='<th><font size=3 color=black>Vote Count</font></th>'
        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database =
'evoting',charset='utf8')
        with con:
            cur = con.cursor()
            cur.execute("select * FROM addparty")
            rows = cur.fetchall()
            for row in rows:
                cname = row[0]
```

```
            count = getCount(cname)
            pname = str(row[1])
            area = row[2]
            image = row[3]
            output+='<tr><td><font size=3 color=black>'+cname+'</font></td>'
            output+='<td><font size=3 color=black>'+pname+'</font></td>'
            output+='<td><font size=3 color=black>'+area+'</font></td>'
            output+='<td><img src="/static/parties/'+cname+'.png" width=200 height=200></img></td>'
            output+='<td><font size=3 color=black>'+str(count)+'</font></td></tr>'
        output+="</table><br/><br/><br/><br/><br/>"
        context= {'data':output}
        return render(request, 'ViewVotes.html', context)

def ViewParty(request):
    if request.method == 'GET':
        output = '<table border=1 align=center style="margin: 0 auto;">'
        output+='<tr><th><font size=3 color=black>Candidate Name</font></th>'
        output+='<th><font size=3 color=black>Party Name</font></th>'
        output+='<th><font size=3 color=black>Area Name</font></th>'
        output+='<th><font size=3 color=black>Image</font></th>'
        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database =
'evoting',charset='utf8')
        with con:
            cur = con.cursor()
            cur.execute("select * FROM addparty")
            rows = cur.fetchall()
            for row in rows:
                cname = row[0]
                pname = str(row[1])
                area = row[2]
                image = row[3]
                output+='<tr><td><font size=3 color=black>'+cname+'</font></td>'
                output+='<td><font size=3 color=black>'+pname+'</font></td>'
                output+='<td><font size=3 color=black>'+area+'</font></td>'
```

```
            output+='<td><img src="/static/parties/'+cname+'.png" width=200
height=200></img></td></tr>'
        output+="</table><br/><br/><br/><br/><br/>"
        context= {'data':output}
        return render(request, 'ViewParty.html', context)

def AddPartyAction(request):
    if request.method == 'POST':
      cname = request.POST.get('t1', False)
      pname = request.POST.get('t2', False)
      area = request.POST.get('t3', False)
      myfile = request.FILES['t4']

      fs = FileSystemStorage()
      filename = fs.save('EVotingApp/static/parties/'+cname+'.png', myfile)

      db_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root',
database = 'evoting',charset='utf8')
      db_cursor = db_connection.cursor()
      student_sql_query = "INSERT INTO addparty(candidatename,partyname,areaname,image)
VALUES('"+cname+"','"+pname+"','"+area+"','"+cname+"')"
      db_cursor.execute(student_sql_query)
      db_connection.commit()
      print(db_cursor.rowcount, "Record Inserted")
      if db_cursor.rowcount == 1:
       context= {'data':'Party Details Added'}
       return render(request, 'AddParty.html', context)
      else:
       context= {'data':'Error in adding party details'}
       return render(request, 'AddParty.html', context)



def saveSignup(request):
    if request.method == 'POST':
```

```
        global username, password, contact, email, address
        img = cv2.imread('EVotingApp/static/photo/test.png')
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        face_component = None
        faces = face_detection.detectMultiScale(gray, 1.3, 5)
        for (x, y, w, h) in faces:
            face_component = img[y:y+h, x:x+w]
        if face_component is not None:
            cv2.imwrite('EVotingApp/static/profiles/' + username + '.png', face_component)
            db_connection = pymysql.connect(host='127.0.0.1', port=3306, user='root', password='root',
                                            database='evoting', charset='utf8')
            db_cursor = db_connection.cursor()
            student_sql_query = "INSERT INTO register(username,password,contact,email,address)
VALUES('" + username + "','" + password + "','" + contact + "','" + email + "','" + address + "')"
            db_cursor.execute(student_sql_query)
            db_connection.commit()
            print(db_cursor.rowcount, "Record Inserted")
            if db_cursor.rowcount == 1:
                context = {'data': 'Signup Process Completed'}
                return render(request, 'Register.html', context)
            else:
                context = {'data': 'Unable to detect face. Please retry'}


        return render(request, 'Register.html', context)
def Signup(request):
    if request.method == 'POST':
        global username, password, contact, email, address
        username = request.POST.get('username', False)
        password = request.POST.get('password', False)
        contact = request.POST.get('contact', False)
        email = request.POST.get('email', False)
        address = request.POST.get('address', False)
        context= {'data':'Capture Your face'}
        return render(request, 'CaptureFace.html', context)
```

```
def AdminLogin(request):
    if request.method == 'POST':
        global username
        username = request.POST.get('username', False)
        password = request.POST.get('password', False)
        if username == 'admin' and password == 'admin':
            context= {'data':'Welcome '+username}
            return render(request, 'AdminScreen.html', context)
        if status == 'none':
            context= {'data':'Invalid login details'}
            return render(request, 'Admin.html', context)

def UserLogin(request):
    if request.method == 'POST':
        global username
        username = request.POST.get('username', False)
        password = request.POST.get('password', False)
        status = 'none'
        con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database =
'evoting',charset='utf8')
        with con:
            cur = con.cursor()
            cur.execute("select * FROM register")
            rows = cur.fetchall()
            for row in rows:
                if row[0] == username and row[1] == password:
                    status = 'success'
                    break
        if status == 'success':
            context= {'data':'<center><font size="3" color="black">Welcome
'+username+'<br/><br/><br/><br/>'}
            return render(request, 'UserScreen.html', context)
        else:
```

```
        context= {'data':'Invalid login details'}
        return render(request, 'Login.html', context)
```

blockchain.py

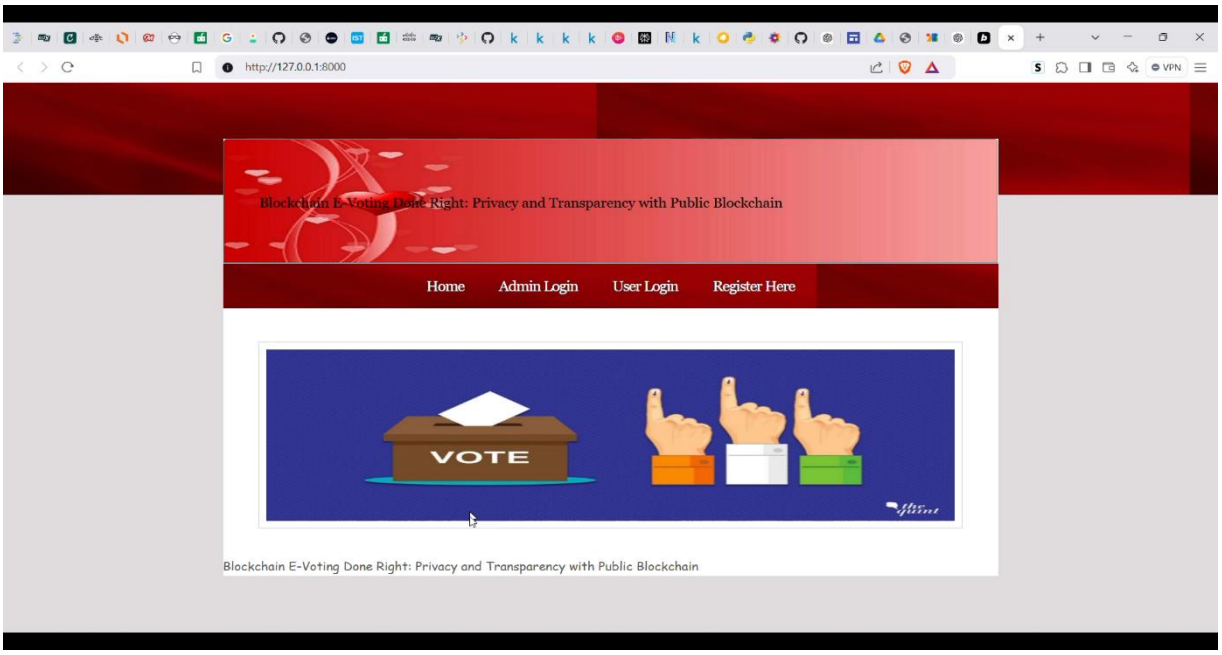
```
from hashlib import sha256
import json
import time
import pickle
from datetime import datetime
import random
import pyaes, pbkdf2, binascii, os, secrets
import base64

class Block:
    def __init__(self, index, transactions, timestamp, previous_hash):
        self.index = index
        self.transactions = transactions
        self.timestamp = timestamp
        self.previous_hash = previous_hash
        self.nonce = 0

    def compute_hash(self):
        block_string = json.dumps(self.__dict__, sort_keys=True)
        return sha256(block_string.encode()).hexdigest()

class Blockchain:
    # difficulty of our PoW algorithm
    difficulty = 2 #using difficulty 2 computation

    def __init__(self):
        self.unconfirmed_transactions = []
        self.chain = []
        self.create_genesis_block()
        self.peer = []
        self.translist = []
```

```
def create_genesis_block(self): #create genesis block
    genesis_block = Block(0, [], time.time(), "0")
    genesis_block.hash = genesis_block.compute_hash()
    self.chain.append(genesis_block)

@property
def last_block(self):
    return self.chain[-1]

def add_block(self, block, proof): #adding data to block by computing new and previous hashes
    previous_hash = self.last_block.hash

    if previous_hash != block.previous_hash:
        return False

    if not self.is_valid_proof(block, proof):
        return False

    block.hash = proof
    #print("main "+str(block.hash))
    self.chain.append(block)
    return True

def is_valid_proof(self, block, block_hash): #proof of work
    return (block_hash.startswith('0' * Blockchain.difficulty) and block_hash == block.compute_hash())

def proof_of_work(self, block): #proof of work
    block.nonce = 0

    computed_hash = block.compute_hash()
    while not computed_hash.startswith('0' * Blockchain.difficulty):
        block.nonce += 1
        computed_hash = block.compute_hash()
```

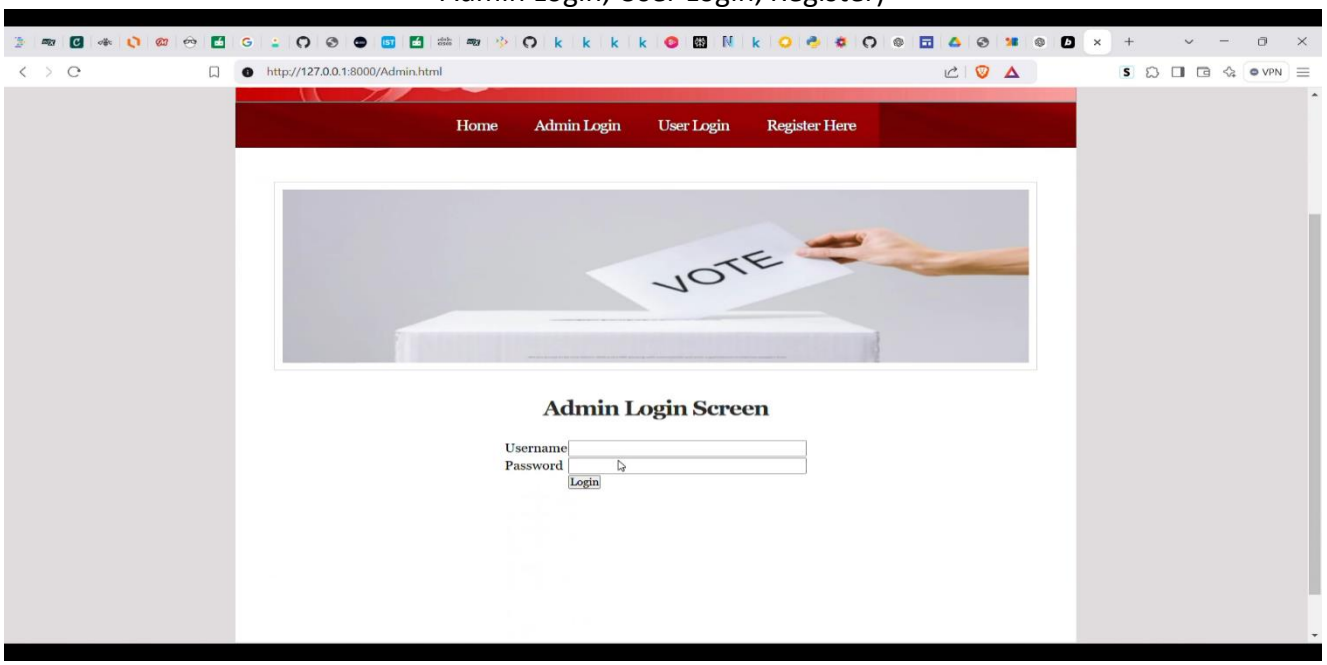
```
        return computed_hash

    def add_new_transaction(self, transaction):
        self.unconfirmed_transactions.append(transaction)

    def addPeer(self, peer_details):
        self.peer.append(peer_details)

    def addTransaction(self,trans_details): #add transaction
        self.translist.append(trans_details)

    def mine(self):#mine transaction
        if not self.unconfirmed_transactions:
            return False

        last_block = self.last_block

        new_block = Block(index=last_block.index + 1,
                          transactions=self.unconfirmed_transactions,
                          timestamp=time.time(),
                          previous_hash=last_block.hash)

        proof = self.proof_of_work(new_block)
        self.add_block(new_block, proof)

        self.unconfirmed_transactions = []
        return new_block.index

    def save_object(self,obj, filename):
        with open(filename, 'wb') as output:
            pickle.dump(obj, output, pickle.HIGHEST_PROTOCOL)
```

CHAPTER 8

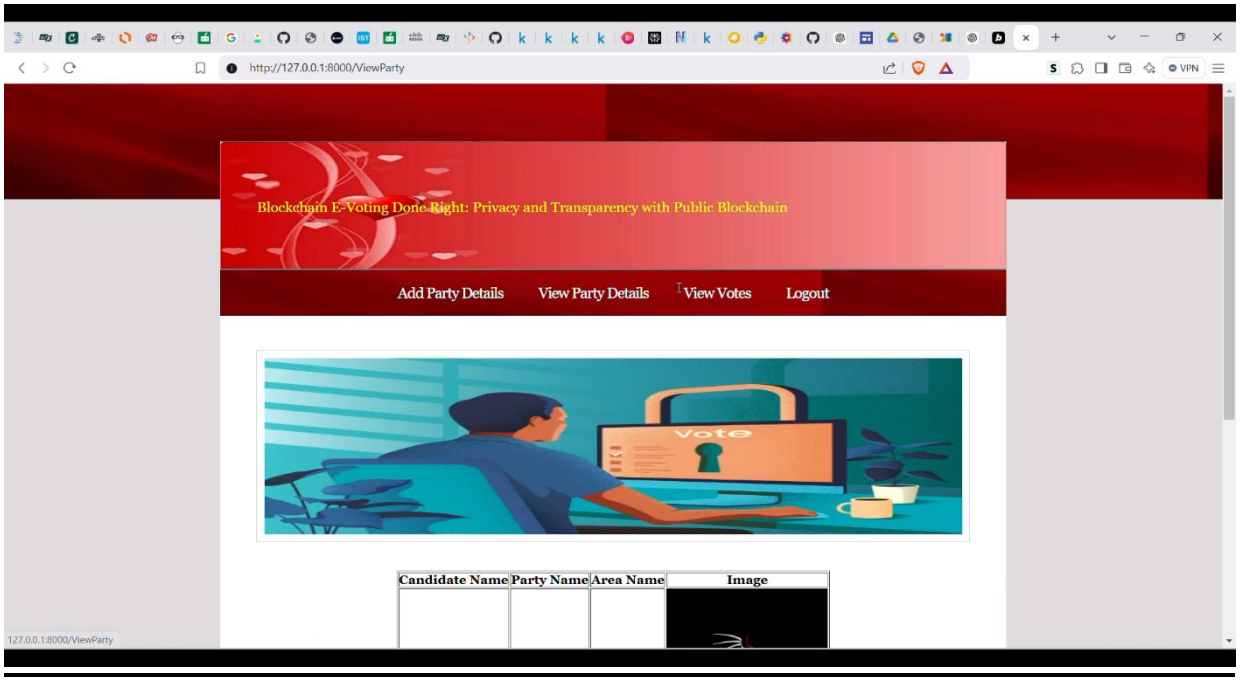
Screenshots :



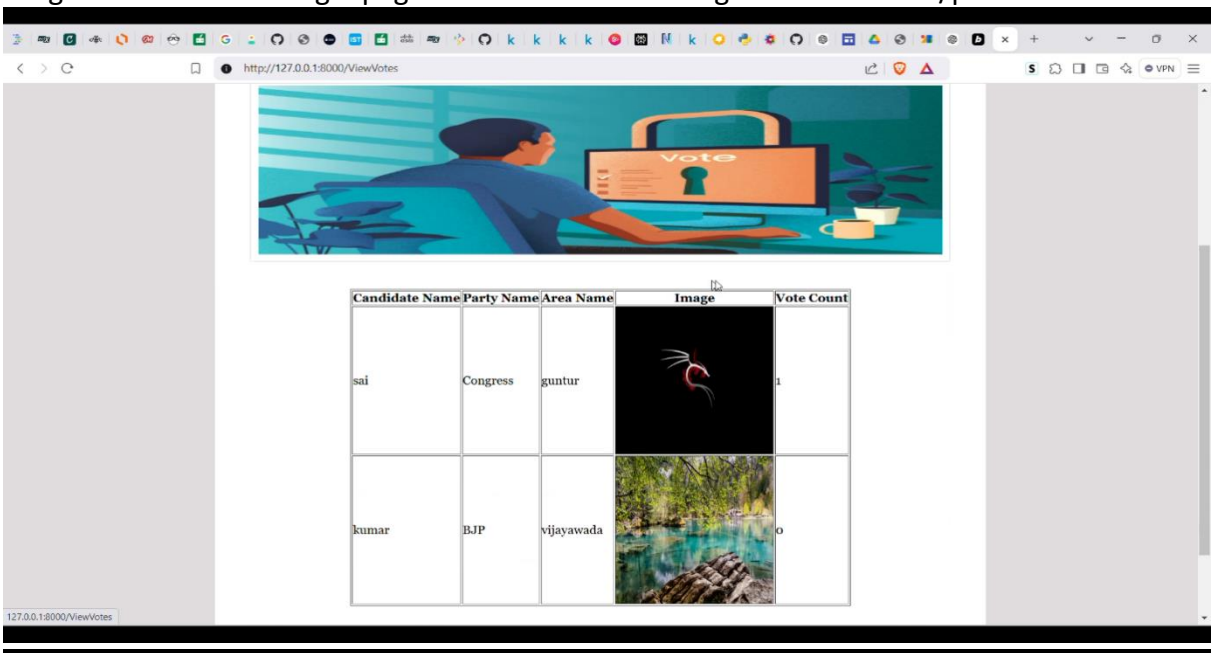
Home Page: "Blockchain E-Voting Done Right" with voting illustration and navigation menu (Home, Admin Login, User Login, Register)



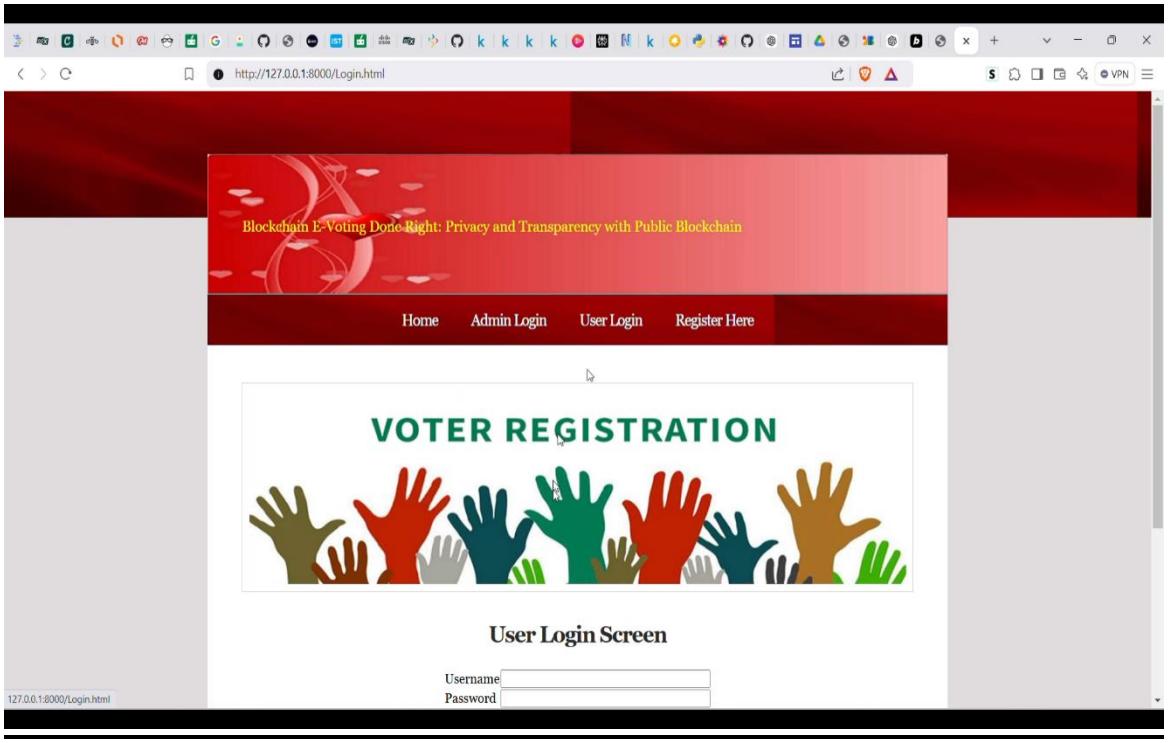
Admin Login Screen: Admin login page with a vote ballot image and username/password fields



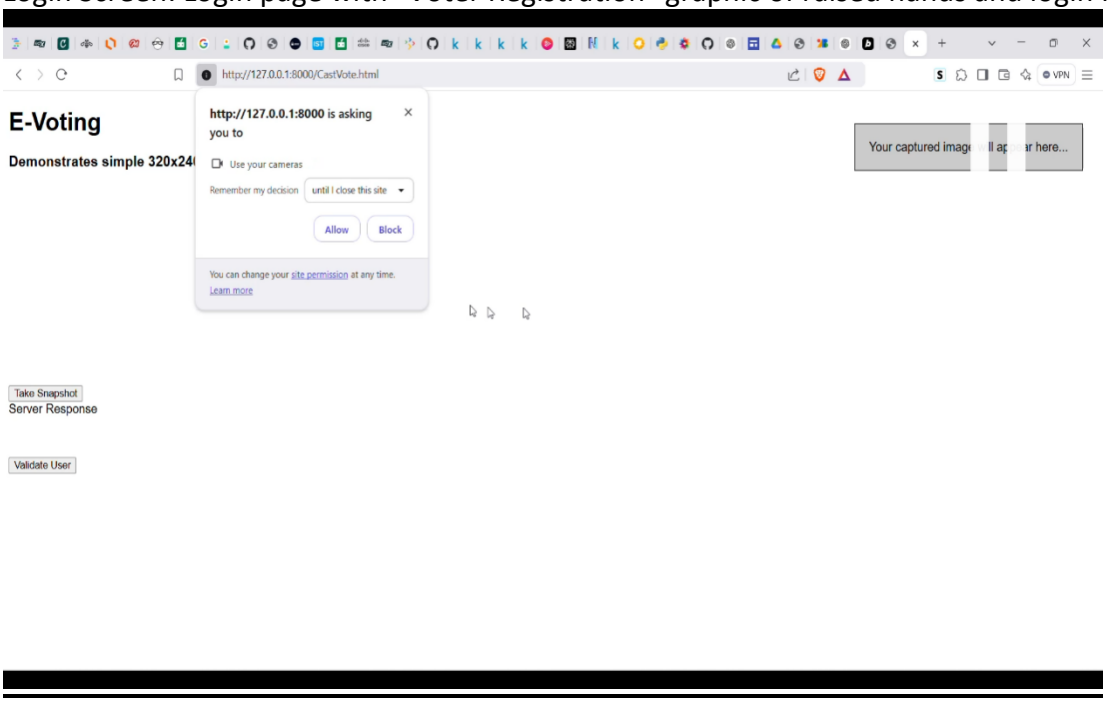
Admin Login Screen: Admin login page with a vote ballot image and username/password fields



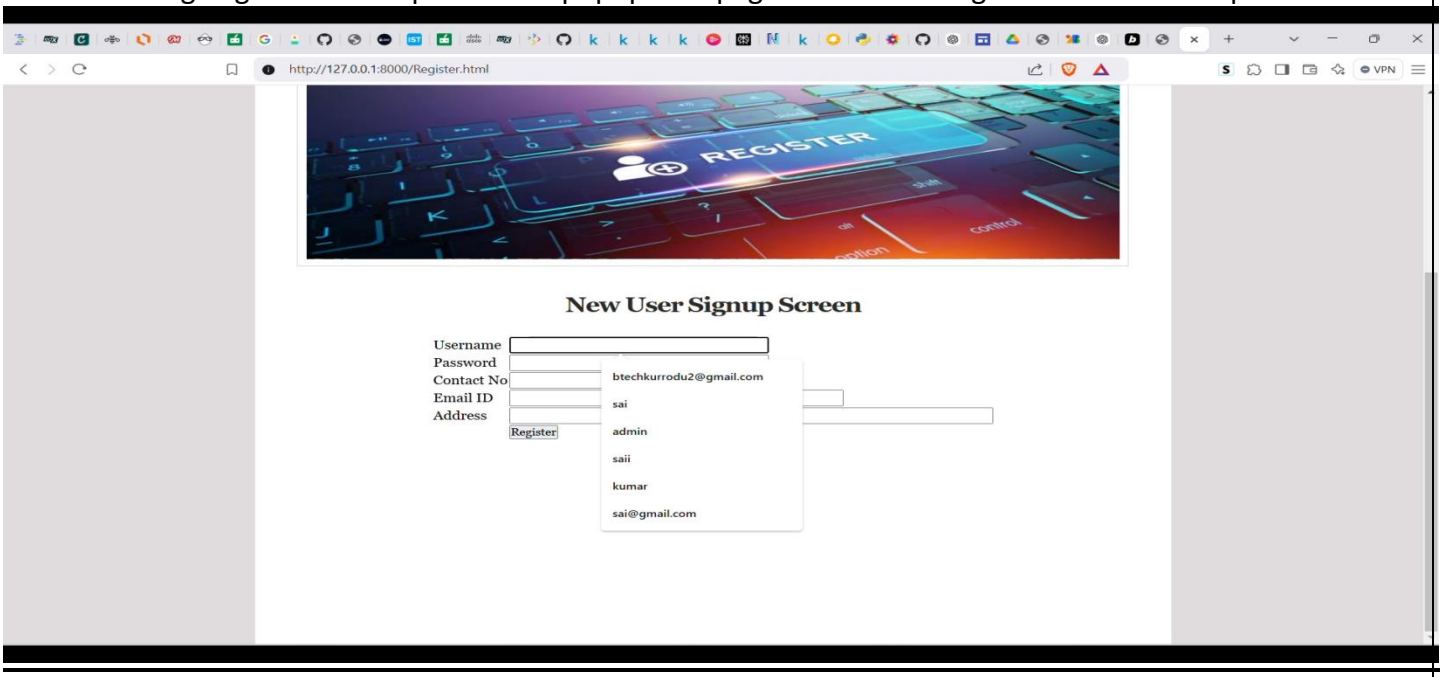
View Party Page: Similar to View Votes page but with admin navigation (Add Party Details, View Party Details, View Votes, Logout)



User Login Screen: Login page with "Voter Registration" graphic of raised hands and login fields



E-Voting Page: A camera permission popup on a page titled "E-Voting Demonstrates simple"



User Registration Screen: A registration form with a colorful hands graphic, allowing new users to sign up with username, password, contact number, and email

CHAPTER 9

CONCLUSION

The proposed system effectively enhances security, efficiency, and transparency by integrating **blockchain, AI, and federated learning** technologies. The **Admin Module** ensures seamless system management, **User Module** provides a secure and user-friendly experience, and the **Mechanic Module** streamlines service requests.

By leveraging **Ethereum blockchain, smart contracts, and AI-driven analytics**, the system guarantees **data integrity, fraud prevention, and real-time monitoring**. Additionally, differential privacy and federated learning enhance data security in sensitive environments.

This system ultimately provides a **robust, scalable, and secure** solution that improves user trust, simplifies operations, and ensures accountability. Future enhancements can incorporate **IoT-based automation, advanced AI-driven threat detection, and decentralized identity management** for further system improvement

CHAPTER 10

BIBLIOGRAPHY

[1] A. M. AL-MADANI, A. T. GAIKWAD,V. MAHALE and Z. A. T. AHMED. Decentralized Evoting system based on Smart Contract by using Blockchain Technology. In Proceedings of the 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC), Aurangabad, India, pp. 176-180, 2020.

[2] S. T. ALVI, M. N. UDDIN and L.ISLAM. Digital Voting: A Blockchain-based E-Voting System using Biohash and Smart Contract. In Third International Conference on Smart Systems andInventive Technology (ICSSIT), Tirunelveli, India, pp. 228-233, 2020.

[3] A. M. ANTONOPOULOS and G. WOOD. Mastering ethereum: building smart contracts and dapps. O'reilly Media, 2018.

[4] H. AZIZ, M. BRILL, V. CONITZER, E. ELKIND, R. FREEMAN and T. WALSH. Justified representation in approval-based committee voting. Social Choice and Welfare, 48(2), 461– 485, 2017.

[5] P. FALISZEWSKI, P. SKOWRON, A. M . SLINKO, and N. TALMON. Multiwinner voting: A new challenge for social choice theory. In U. Endriss (Ed.), Trends in computational social choice, AI access, pp. 27–47, 2017.

[6] F. FUSCO, M. ILARIA LUNESU, F. EROS PANI and A. PINNA. Crypto-voting, a blockchain based e-voting system. In Proccedings of the 10th Int. Conf. on Knowledge Management and Information Sharing, vol 3, pp 223-227, 2018.

[7] F. Þ. HJÁLMARSSON, G. K. HREIÐARSSON, M. HAMDAQA and G. HJÁLMTÝSSON. Blockchain-Based E-Voting System. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, pp. 983-986, 2018.

[8] A. INDAPWAR, M. CHANDAK and A. JAIN. E-voting system using Blockchain technology. Int. J. of Advanced Trends in Computer Science and Engineering, 9, No.3, pp. 2020.

[9] D. M. KILGOUR. Approval balloting for multi-winner elections. In J. F. Laslier & M. R. Sanver (Eds.), Handbook on approval voting (pp. 105–124). Springer, 2010.

[10] M. LACKNER and P. SKOWRON. Multi-winner voting with approval preferences. Springer Nature, 2023.

CHAPTER 11

FUTURE SCOPE

The secure online voting system using blockchain has strong potential to improve how elections are conducted. It can be expanded to connect with national digital ID systems, making voter verification more reliable. This will help prevent fraud and impersonation.

To make the system more inclusive, it can support additional regional languages. Features like screen readers and voice assistance can help elderly and differently-abled users.

Artificial intelligence can be added to detect cyber threats and suspicious activity in real time. This would increase the system's overall security.

A mobile app version of the voting system can be developed. This would make voting easier and more accessible, especially in remote areas.

To handle large-scale elections, the system can be optimized with advanced blockchain technologies like Layer 2 solutions or sharding. These improvements will help manage millions of votes quickly and efficiently.

Different blockchains can be connected to allow separate components—like voter registration and vote counting—to run smoothly across networks.

Voters will gain more confidence if they can verify their own vote without revealing their identity. This can be achieved using cryptographic techniques like zero-knowledge proofs.

In the future, the system must follow government rules and election laws. Working with election commissions will help create a legal framework for using blockchain in voting.

As quantum computers become more powerful, the system should use quantum-resistant encryption to stay secure in the long term.

Lastly, election officials can use blockchain analytics tools to track voting patterns and system activity. This will help monitor elections in real time while keeping voter data private.