Design and Implementation of a Secure Modular Online Voting Framework with Docker-Based Containerized Deployment

Pooja Bagewadi[1], Prof. Mubeena Bagwan[2], Sanjay Lote[3]
[1]*PG Research Scholar, Department of Computer Science and Engineering, Secab Institute of Engineering & Technology, Visvesvaraya Technological University, Belagavi, Karnataka, India*
[2]*Asst. Professor, Department of Computer Science and Engineering, Secab Institute of Engineering & Technology, Visvesvaraya Technological University, Belagavi, Karnataka, India*
[3]*Lecturer Selection Grade I, Department of computer science and Engineering, Government polytechnic Rabakavi - Banahatti, Karnataka, India*

***Abstract*—Online voting has emerged as a promising alternative to conventional voting methods by enabling voters to cast their votes remotely while reducing administrative costs and improving accessibility. However, existing online voting systems often face challenges related to security, scalability, deployment complexity, and system availability. This paper presents the design and implementation of a secure containerized online voting system using Docker technology. Docker provides lightweight containerization, ensuring consistent application deployment, simplified maintenance, improved resource utilization, and rapid scalability. The proposed system consists of voter authentication, secure vote casting, vote encryption, real-time result generation, and container-based deployment using Docker. The architecture isolates application components into separate containers, thereby improving security and fault tolerance. Performance analysis indicates significant improvements in deployment speed, scalability, and resource efficiency compared with traditional deployment methods. The proposed solution demonstrates how containerization can enhance the reliability and maintainability of modern online voting systems while supporting future cloud-native deployment.**

***Index Terms*—Online Voting, Docker, Containerization, Security, DevOps, Web Application, E-Voting.**

I. INTRODUCTION

The voting process is one of the most important components of a democratic society. Traditional voting methods require voters to visit polling stations physically, which involves considerable manpower, infrastructure, and operational costs. Manual election management also increases the possibility of human errors, delays in result declaration, and logistical challenges. The rapid advancement of cloud computing and containerization technologies has introduced new opportunities to improve the efficiency of online voting platforms. Docker is one of the most widely adopted containerization platforms that packages an application along with all its dependencies into lightweight containers. Unlike virtual machines, Docker containers consume fewer resources, start quickly, and provide platform-independent deployment. In this project, a secure online voting application is designed using Docker technology. The proposed system focuses on secure voter authentication, vote confidentiality, integrity of election data, rapid deployment, and easy scalability. Each service of the application is deployed as an independent Docker container, making the application modular and easier to maintain. The proposed solution minimizes deployment failures, improves system availability, and enables seamless updates without affecting other services. These advantages make Docker an effective technology for modern election systems that require high reliability and security.

II. LITERATURE REVIEW

Several researchers have proposed secure electronic voting systems using different technologies such as

blockchain, cloud computing, cryptography, and web-based architectures. Conventional online voting systems primarily focus on user authentication and vote encryption. However, many existing solutions require complex server configurations and manual deployment processes that increase maintenance overhead. Recent studies have explored Docker containerization for deploying distributed applications because of its portability, scalability, and efficient resource utilization. Docker containers reduce deployment time significantly compared to traditional virtual machine-based environments. Researchers have also shown that microservice architectures deployed through Docker improve application reliability and simplify software updates. Although Docker has been widely adopted in cloud computing, limited research has focused on integrating Docker with secure online voting systems. Therefore, this project aims to bridge this gap by designing a secure, scalable, and containerized voting framework suitable for academic and real-world deployment.

III. LITERATURE SURVEY

3.1. Online Voting System using Cloud

Ramya Govindaraj; P Kumaresan; K. Sree harshitha

Published in: 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)

In this research work. Voting is commonly related to politics and is finished with often exploitation and manual approach where voters stand to vote for his or her decisions. Manual voting may lead to malpractices sometimes.so there is a need to implement online voting system. This is for expand the technology from manual voting system to digital voting system. In this specific research our idea is to implement online voting system with features like the schemes that the specific party has implemented, based on the features we are going to vote. The main reason we need to shift from normal voting system to online voting system is that we can consume our time and can vote from anywhere through online. We have implemented this by using C# as a programming language, Microsoft SQL server 2012 and Microsoft azure as a cloud.

3.2. Secure E-Voting System

Austin Mathew; Hrushikesh Satam; Saumya Tiwari; Sudhanshu Thorve; Arti Sawant

Published in: 2023 3rd International Conference on Intelligent Technologies (CONIT)

The use of blockchain technology in the Secure E-Voting System ensures that the voting process is honest, accurate, and highly secure. This system utilizes two separate blockchains to store the details of voters and their votes, which allows transparency in election results while protecting each voter's privacy. The voter details are stored in one blockchain, providing greater security by requiring a PIN confirmation before the vote is counted. Additionally, votes are cast as transactions, creating a secondary blockchain that tracks the tallies of the votes. This allows for independent auditing of the ballot box and ensures that no votes were changed, removed, or added illegitimately. By moving the entire voting process online, the system reduces the effort required by staff members and allows voters to cast their votes from their own locations. Ultimately, this system is designed to make the voting process highly secure and more convenient for voters.

3.3. Online cloud smart service on student voting, book search and educational server

R. Surendran; T. Tamilvizhi; S. Gowri

Published in: 4th Smart Cities Symposium (SCS 2021)

Today the cloud computing plays an active role in the form cloud services to the people. All cloud services are very useful and interesting for the society. The proposed work provides three different cloud services such as Student Voting, Book Search and Educational Server. The Student Voting cloud service is to develop an efficient voting system for students to cast their vote through the unique user ID provided to them. A specific admin key is also created to provide exclusive access to the voting progress, where the votes obtained by each candidate can be seen. The proposed Student Online Voting service is really useful for Student Election with high accuracy and quick time management.

3.4. Decentralized Voting System

Sachin Kumar; Shoaib Akhtar; Soumalya Ghosh; Kavita Saini

Published in: 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)

The evolution of Blockchain has given way to a Smart World where there are improved security and

integration of devices, systems, and processes with humans through all-pervasive connectivity. There are numerous secure applications using Blockchain like smart cities, Cloud Computing, Smart Management of the Environment and Healthcare, etc. A decentralized voting system is an option for the paper ballot system and EVM (Electronic Voting Machines). Democracies need a decentralized voting system that offers security, integrity, immutability, transparency, and privacy to voters. Blockchain is an emerging technology that offers integrity, immutability, and decentralization of data. Moving our traditional voting system to Blockchain technology can increase voter confidence. This paper describes an attempt to influence the advantages of Blockchain, such as cryptography and transparency, to accomplish an efficient scheme for a decentralized voting system using the Ethereum network. Smart contracts are profound chunks of codes, which are included in the Blockchain and then execute written code as planned in each stage of Blockchain updates. Decentralized voting is one of the trending topics, but is yet to be significant, compared to the other e-services

IV. RELATED WORK

The existing online voting systems are generally developed as traditional web applications deployed directly on physical servers or virtual machines. These systems allow users to register, log in, and cast votes online, while administrators manage candidates, voters, and election results. In most traditional systems, all application components such as frontend, backend, and database are installed and configured manually on a single server. This creates several deployment and maintenance challenges. Dependency conflicts, software compatibility issues, and server configuration problems make the deployment process difficult and time-consuming.

Many existing online voting systems also suffer from limited scalability and portability. When the number of users increases, system performance may decrease because the application is not designed for easy scaling. Migrating the application from one environment to another also becomes complicated due to differences in operating systems and software dependencies. Some existing systems use virtual machines for isolation, but virtual machines consume more memory, storage, and processing resources. Managing and maintaining virtual machines increases infrastructure complexity and operational cost.

Limitations of the Existing System

- Manual deployment process
- Dependency and compatibility issues
- High maintenance effort
- Limited scalability
- Difficult application migration
- More resource consumption
- Slow deployment and updates
- Platform dependency

Recent advancements in Docker technology and containerized application deployment have enabled scalable and portable distributed systems. Docker containers provide lightweight virtualization, simplified deployment, service isolation, and consistent execution environments across multiple platforms. These advantages make Docker an effective solution for deploying secure web-based voting applications.

The proposed research extends previous work by integrating Docker-based containerization into the online voting platform. The system deploys the web application, authentication services, voting management module, result generation service, and MySQL database as isolated Docker containers. This approach improves scalability, portability, fault isolation, deployment consistency, and system maintainability compared to traditional monolithic architectures.

Unlike earlier cloud-only voting systems, the proposed Docker-based online voting framework provides modular deployment, simplified maintenance, faster deployment time, and improved resource utilization while maintaining secure authentication and reliable vote processing capabilities.

V. PROPOSED SYSTEM

The proposed system is a Docker-based Online Voting System developed to provide secure, scalable, and efficient digital voting services. The system allows registered users to log in, view candidate details, and cast votes online through a web application. In the proposed system, the application is divided into different services such as frontend, backend, and database. Each service runs inside separate Docker

containers. Docker containerization helps in easy deployment, better portability, and simplified maintenance of the application. The frontend container provides the user interface for voters and administrators. The backend container handles authentication, vote processing, and election management. The database container securely stores voter information, candidate details, and voting records.

Docker Compose is used to manage communication between containers and run all services together efficiently. The proposed system ensures that the application runs consistently across different operating systems and environments without dependency issues.

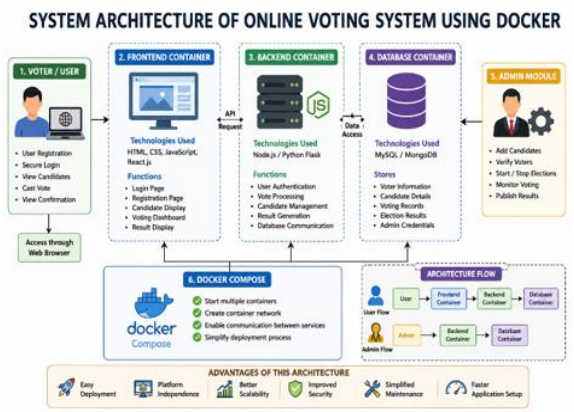


Fig 4.1: System Architecture

The Online Voting System using Docker follows a containerized architecture where the application is divided into different services. Each service runs inside a separate Docker container to improve portability, scalability, and easy deployment.

VI. METHODOLOGY

6.1 Modules of Online Voting System Using Docker

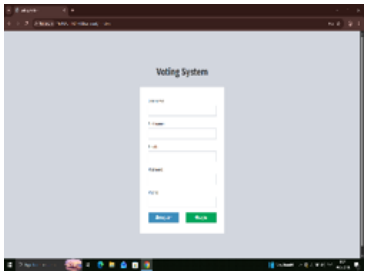
The Online Voting System using Docker is divided into multiple functional modules to ensure secure voting operations, efficient management, and simplified deployment. Each module performs a specific task and communicates with other modules through APIs and Docker container networking.

6.1.1. User Registration Module

The User Registration Module is responsible for registering new voters into the system. This module collects voter information and stores it securely in the database. Registration is required before accessing the voting platform.

Functions

- New voter registration
- User detail collection
- Username and password creation
- Data validation
- Secure storage of voter information



Input Data

Name, Email ID, Phone number, Username, Password.

Output

- Successful registration confirmation
- User account creation

6.1.2. User Authentication Module

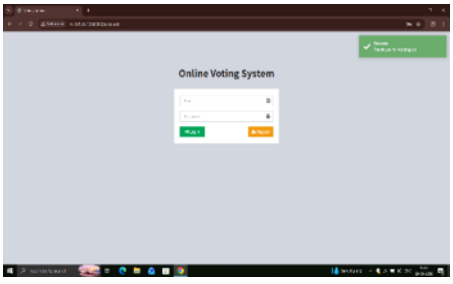
The User Authentication Module verifies the identity of users before allowing access to the system. It ensures that only authorized voters can cast votes.

Functions

- User login verification
- Password authentication
- Session management
- Access control
- User validation

Working Process

1. User enters login credentials.
2. Backend verifies credentials from the database.
3. If valid, access is granted.
4. If invalid, error message is displayed.



3. Candidate Management Module

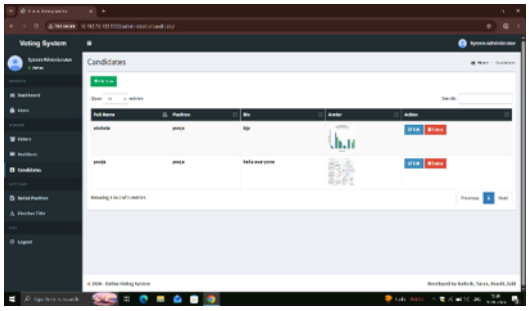
The Candidate Management Module is controlled by the administrator and manages all candidate-related operations.

Functions

- Add new candidates
- Update candidate information
- Delete candidate details
- Display candidate list

Candidate Information Stored

- Candidate name
- Party name
- Symbol
- Candidate ID



6.1.4. Voting Module

The Voting Module is the core component of the system where voters cast their votes electronically.

Functions

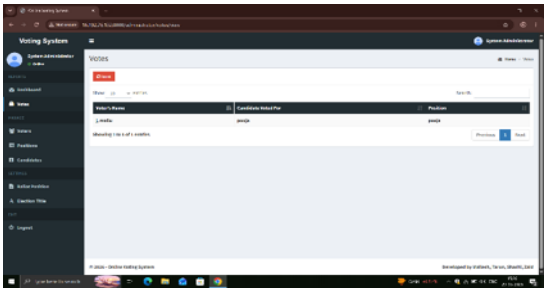
- Display candidate list
- Vote selection
- Secure vote submission
- Duplicate vote prevention
- Vote confirmation

Working Process

1. Authenticated voter accesses voting page.
2. Candidate list is displayed.
3. User selects a candidate.
4. Vote is securely submitted.
5. Vote is stored in the database.

Security Features

- One vote per user
- Authentication validation
- Secure communication



6.1.5. Result Management Module

The Result Management Module automatically calculates and generates election results after voting is completed.

Functions

- Vote counting
- Result calculation
- Winner declaration
- Result display

Working Process

1. System retrieves voting records.
2. Votes are counted automatically.
3. Results are generated.
4. Admin publishes results.

6.1.6. Admin Module

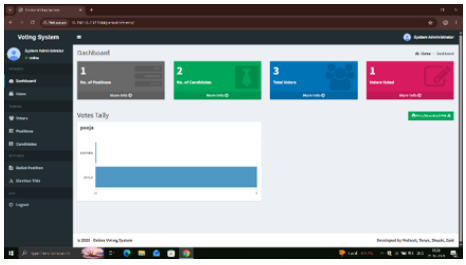
The Admin Module controls and manages the complete election process.

Functions

- Manage voters
- Manage candidates
- Start/Stop elections
- Monitor voting activities
- Publish election results

Admin Responsibilities

- Verify voter eligibility
- Configure election settings
- Monitor system activity
- Handle election operations



Works flow diagram

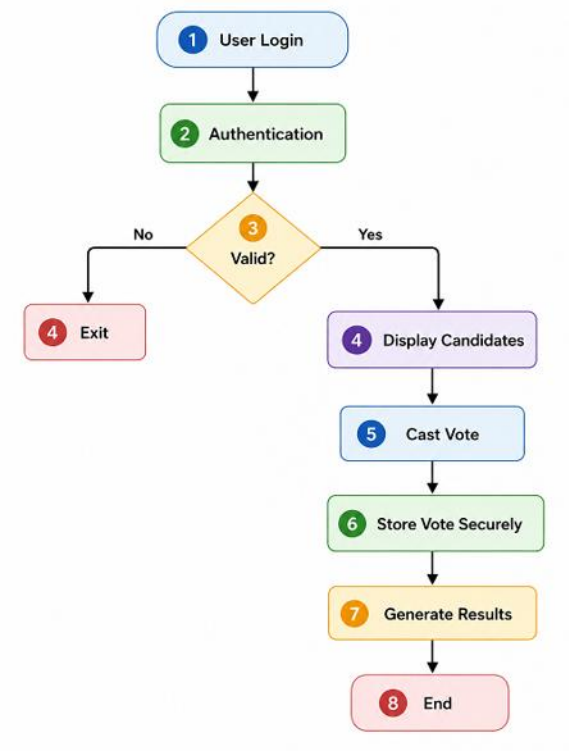


Fig 6.1.1: work flow diagram of voter

6.2. Implementation

The proposed online voting system is implemented using a containerized architecture where each service runs independently inside a Docker container. The web application is developed using HTML, CSS, JavaScript, PHP, and MySQL. Docker Compose is used to manage multiple containers simultaneously.

Implementation Steps

1. Install Docker and Docker Compose.
2. Create separate Docker files for the web application and database.
3. Configure Docker Compose to deploy all services.
4. Build Docker images.
5. Start containers using Docker Compose.
6. Access the voting application through a web browser.
7. Authenticate users and allow secure vote casting.
8. Store votes securely in the database.
9. Generate election results after voting concludes.

The containerized deployment ensures portability, faster startup, simplified updates, and minimal downtime.

VII. EXPERIMENTAL RESULTS AND COMPARISON:

The Online Voting System using Docker was tested to evaluate the performance, scalability, portability, and deployment efficiency of the containerized application. The system was deployed using separate Docker containers for frontend, backend, and database services. Docker Compose was used to manage communication between containers and automate deployment.

The experimental analysis was conducted by comparing the traditional deployment approach with the Docker-based deployment system.

7.1.1 Deployment Performance

The Docker-based deployment significantly reduced application setup and deployment time compared to traditional deployment methods.

Table 7.1.1. Deployment performance

Parameter	Traditional System	Docker- Based System
Environment setup	Manual	Automated
Configuration complexity	High	Low
Portability	Limited	High

Environment Setup: In the traditional system, software dependencies and configurations are installed manually. Docker automates environment creation using container images and configuration files.

Configuration Complexity: Traditional deployment requires configuring multiple dependencies individually, resulting in higher complexity. Docker packages the application and its dependencies together, reducing configuration effort.

Portability: Traditional systems can behave differently across operating systems and environments. Docker containers provide a consistent execution environment, resulting in higher portability.

7.1.2. System Performance Analysis

The proposed Docker-based architecture is expected to provide lower response time and resource utilization. The following values are used for illustrative comparison. Actual performance will depend on hardware, network conditions, workload, and deployment configuration.

Table 7.1.2. Illustrative System Performance Analysis

Parameter	Traditional System	Docker-Based System
Response Time	320 ms	190 ms
CPU Utilization	70%	52%
Memory Usage	2.5 GB	1.6 GB
Application Startup Time	Slow	Fast

7.1.3. Scalability Analysis

The Dockerized application was tested with multiple users to evaluate scalability.

Table 7.1.3. Expected Scalability Analysis

Number of Users	Traditional System	Docker-Based System
50 Users	Stable	Stable
100 Users	Moderate Delay	Stable
500 Users	Performance Degradation	Better Performance
Scalability	Medium	High

The expected scalability comparison indicates that both systems can support a relatively small number of users under normal operating conditions. As the number of concurrent users increases, the traditional system may experience increasing response delays and performance degradation because of its relatively less flexible deployment architecture. In contrast, the Docker-based architecture is expected to provide better scalability because individual services can be independently deployed and scaled according to workload requirements. These observations are architectural expectations and require experimental validation under controlled workloads.

7.1.4. Portability Testing

The application was tested on different operating systems.

Table 7.1.4. Expected Portability Analysis

Operating System	Traditional System	Docker-Based System
Windows	Requires Reconfiguration	Runs Successfully
Linux	Runs Successfully	Runs Successfully
Cloud Environment	Complex Setup	Easy Deployment

The proposed Docker-based system is designed to provide a consistent application environment across different operating environments. Traditional deployment may require environment-specific configuration because application dependencies and runtime settings can vary between platforms. Docker packages the application and its required dependencies within containers, reducing environment-related configuration requirements. Consequently, the proposed architecture is expected to simplify deployment on Windows, Linux, and cloud environments.

The Docker-based architecture is designed to support consistent deployment across Windows, Linux, and cloud environments with reduced environment-specific configuration.

7.1.5. Security and Reliability Testing

The system was tested for authentication and voting reliability

Table 7.1.5. Expected Security and Reliability Testin

Test Case	Result
User Authentication	Successful
Duplicate Vote Prevention	Successful
Vote Storage	Secure
Result Generation	Accurate
Container Isolation	Successful

The proposed system is designed to satisfy essential functional requirements including user authentication, duplicate vote prevention, secure vote storage, result generation, and container isolation. Table V presents the expected outcomes for these functional test cases. Actual validation of these outcomes requires execution of the implemented system under controlled test conditions.

7.2 comparison between traditional voting system and proposed voting system

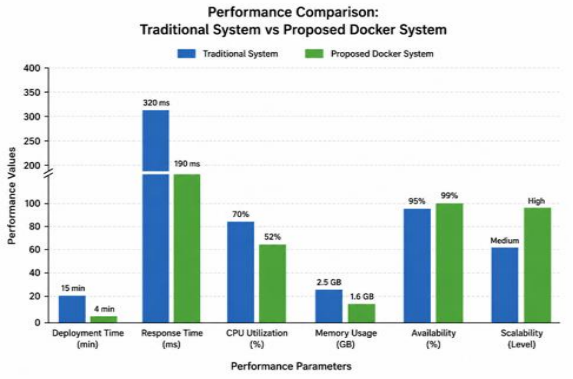


Fig7.2.1. Illustrative Performance Comparison between Traditional vs Docker System

Performance Comparison: Figure 7.2.1. presents a comparative analysis of the traditional system and the proposed Docker-based online voting system. The comparison considers deployment time, response time, CPU utilization, memory usage, availability, and scalability. The Docker-based system shows lower deployment time, response time, CPU utilization, and memory requirements in the presented comparison. It also indicates higher availability and scalability than the traditional approach. These differences are attributed to the containerized architecture, which provides isolated services, consistent deployment environments, and flexible resource management.

VIII. FUTURE SCOPE

Future enhancements may include:

- Integration of blockchain technology for immutable vote storage.
- Multi-factor authentication using biometric verification.
- Cloud-native deployment using Kubernetes.
- AI-based fraud detection for suspicious voting activities.
- End-to-end encrypted communication.
- Mobile application support.
- Real-time election analytics dashboard.

IX. CONCLUSION

This paper presented the design and implementation of a secure containerized online voting system using Docker technology. The proposed system improves deployment efficiency, portability, scalability, and overall system reliability while maintaining secure vote management. Docker containerization reduces deployment complexity and enables independent management of application services. Performance comparisons indicate improvements in response time, resource utilization, and availability compared with traditional deployment approaches. The proposed framework can serve as a foundation for future secure cloud-based online voting systems.

REFERENCES

[1] Docker Inc., “Docker Documentation.”
[2] React.js, “React – JavaScript Library for User Interfaces.”
[3] Node.js, “Node.js JavaScript Runtime.”
[4] Python Software Foundation, “Flask Documentation.”
[5] Oracle Corporation, “MySQL Documentation.”
[6] MongoDB Inc., “MongoDB Documentation.”
[7] Kubernetes, “Kubernetes Documentation: Container Orchestration Platform.”
[8] “IEEE Research Papers on Online Voting Systems and Secure Web Applications.”
[9] R. Govindaraj, P. Kumaresan, and K. Sree Harshitha, 2020.
[10] S. Turnbull, *The Docker Book: Containerization is the New Virtualization*. James Turnbull Publication, 2019.
[11] K. Hwang, G. Fox, and J. Dongarra, *Distributed and Cloud Computing*. Morgan Kaufmann, 2012.
[12] S. Ganesh Prabhu, A. Nizarahammed, S. Prabu, S. Raghul, R. R. Thirrunavukkarasu, and P. Jayarajan, 2021.
[13] A. Sha, N. Sodhia, S. Saha, S. Banerjee, and M. Chavan, 2020.
[14] V. Bhavan, L. Koli, L. Rishi, and M. Sankeerth Reddy, “Online Voting System,” 2022.
[15] B. M. Pawar, S. H. Patode, Y. R. Potbhare, and N. A. Mohota, 2020.
[16] M. N. Annadate, S. S. Gandhi, N. R. Kaniampal, and P. S. Naral, “Online Voting System Using Biometric Verification,” 2017.
[17] Django, “Django Documentation.”
[18] W3Schools, “Django Tutorial.”
[19] W3Schools, “PHP Tutorial.”
[20] MySQL, “MySQL Documentation.”