VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi - 590018

"ONLINE VOTING SYSTEM"

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE ENGINEERING

(IOT & CYBERSECURITY WITH BLOCKCHAIN TECHNOLOGY)

BY

ALVENA DISHA MATHIAS	**4MT21IC004**
DISHA V	**4MT21IC013**
K.S CHAITANYA	**4MT21IC020**
SANJANA KUNDAR	**4MT21IC042**
SHAMYA JAIN	**4MT21IC042**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING(Iot & Cyber security with Blockchain technology)

(*Accredited by NBA*)

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution

(*A Unit of Rajalaxmi Education Trust®, Mangalore - 575001*)

Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi

Badaga Mijar, Moodabidri-574225, Karnataka

2022-23

ABSTRACT

This report examines a C program that simulates a basic electronic voting system, offering insights into its structure, functionality, strengths, weaknesses, and potential improvements. The program allows users to register, log in as voters or candidates, nominate candidates, vote, and view election results. It employs two data structures, "User" and "Candidate," along with arrays to manage user and candidate data. While the program succeeds in facilitating user interactions and basic voting processes, it falls short in several critical aspects. The strengths of the program include its user-friendly interface, effective user and candidate management, and a functional voting system. However, the program has notable weaknesses, including limited data storage capacity, security concerns regarding password handling, insufficient error handling, and scalability constraints. To enhance the program's suitability for real-world use, several improvements are recommended. These include implementing secure data storage using a database, enhancing security measures such as password hashing, implementing robust error handling, and allowing for dynamic scalability. These enhancements are essential to transform the basic electronic voting system simulation into a secure, scalable, and reliable application ready for practical use.

INTRODUCTION

Electronic voting systems have become increasingly popular as a means of conducting elections efficiently and accurately. These systems offer numerous advantages, such as faster tabulation of results, reduced risk of errors, and improved accessibility for voters. The provided C program aims to simulate a basic electronic voting system, allowing users to experience key functionalities of such a system. This report provides a comprehensive introduction to the program, outlining its purpose, structure, and key features.

Purpose of the Electronic Voting System

The primary purpose of the C program is to serve as a demonstrative model of an electronic voting system. It offers users a virtual platform where they can register as voters, candidates, cast their votes, and view election results. The program provides a simplified representation of how electronic voting systems function in practice, emphasizing essential aspects such as user registration, security, and result tabulation.

Program Structure and Organization

The program is organized into functions, each responsible for specific aspects of the electronic voting process. These functions include user registration, login, candidate registration, voting, and result display. The main function acts as a user interface, presenting users with a menu-driven system through which they can interact with the program. User and candidate data are stored in arrays within the program's memory.

Key Features of the Program

1. The program offers several key features, including:
2. User Registration: Users can register with unique usernames and passwords, indicating whether they intend to participate as voters or candidates.
3. Login System: Registered users can log in using their credentials, granting them access to the appropriate functionalities based on their role as voters or candidates.
4. Candidate Nomination: Registered users can nominate themselves as candidates by specifying their desired position.
5. Voting Mechanism: Voters can cast their ballots for candidates of their choice, with the program ensuring that each voter can only vote once.
6. Result Display: The program enables users to view the election results, showing the number of votes received by each candidate.

Strengths

1. **User-Friendly Interface**: The program's menu-driven interface simplifies user interaction, making it accessible even to individuals with limited technical knowledge.
2. **User and Candidate Management:** The program efficiently manages user and candidate data, allowing for user registration, login, and candidate nomination.
3. **Voting System**: The voting mechanism is straightforward and functional, enabling voters to cast their ballots for candidates with ease.

Objectives

User Registration: Allow individuals to register as voters by providing a username and password. These registered users are stored in the users array.

User Login: Enable registered users to log in by entering their username and password. Upon successful login, the system provides access to certain features based on whether the user is a candidate or a voter.

Candidate Nomination: Allow registered users to nominate themselves as candidates for an election. Candidates provide additional information, such as their position. Nominated candidates are stored in the candidate's array, and their status as a candidate is recorded in the isCandidate field of the corresponding user.

Voting: Registered voters can cast their votes for one of the nominated candidates. The system displays a list of available candidates, and the voter selects their choice by entering the candidate's number. The code ensures that voters can only vote once and that candidates cannot vote.

Displaying Election Results: The system can display the results of the election by showing the number of votes received by each candidate and their respective positions.

Limitations on Users and Candidates: The code sets limits on the maximum number of users (MAX_USERS) and candidates (MAX_CANDIDATES) that can be registered. If these limits are reached, the system prevents further registration.

Menu-Driven Interface: The code provides a menu-driven interface in the main function, allowing users to select from various options, such as registration, login, nomination, voting, displaying results, or exiting the application.

User and Candidate Data Management: The code uses structures (User and Candidate) to organize and manage user and candidate data, including their usernames, passwords, and relevant information.

Technologies Used

C Programming Language: The entire program is written in the C programming language.

Standard Input/Output (stdio.h) Library: This library is used for input and output operations. It provides functions like printf and scanf for reading from and writing to the console.

Standard Library (stdlib.h): The stdlib.h library is used for standard functions like malloc and fee, although these functions are not present in this code.

String Manipulation (string.h) Library: The string.h library is used for string manipulation functions, such as strcpy and strcmp, which are used to manage user and candidate information.

The code defines two structures - User and Candidate - to represent user and **Structures:** candidate data. These structures store information such as usernames, passwords, positions, and vote counts.

Arrays: Arrays are used to store multiple instances of user and candidate data. For example, the users and candidates' arrays are used to store user and candidate information.

Control Structures: The code utilizes control structures like if statements, for loops, and a do-while loop in the main function to manage user interactions, registration, login, candidate nomination, voting, and result display.

User Input: The program uses scanf to accept user input from the console. Users provide usernames and passwords for registration and login. They also choose options by entering numbers corresponding to menu choices.

Output: The program uses printf to display messages and results to the console. For example, it prints registration success messages, login status, candidate nomination success, and election results.

Constants: The code defines constants such as MAX_USERS and MAX_CANDIDATES to set limits on the number of users and candidates that can be registered.

Variables: Variables like userCount and candidateCount are used to keep track of the number of registered users and candidates.

Menu System: The program provides a menu system in the main function, allowing users to choose various actions like registration, login, nomination, voting, and result display.

SYSTEM ARCHITECTURE

The provided C code implements a basic voting system with a simple architecture. The system's architecture can be described as follows:

User Interface:

- The user interacts with the system through a command-line interface (CLI).
- The main function presents a menu-driven interface to the user, allowing them to select various options such as registration, login, nomination, voting, displaying results, or exiting the application.

Data Structures:

- The code defines two main data structures: User and Candidate, implemented as C structs.
- User stores user information, including usernames, passwords, and status (candidate or voter).
- Candidate stores candidate information, including usernames, passwords, positions, and vote counts.

Data Storage:

- Arrays of User and Candidate are used to store multiple instances of user and candidate data.
- users is an array for storing registered users.
- candidates is an array for storing nominated candidates.
- userCount and candidateCount keep track of the number of registered users and candidates.

Functions:

- The code is organized into functions that encapsulate specific functionality:
- registerUser(): Allows users to register and stores user data in the users array.
- login(): Allows users to log in by verifying their credentials and returns a pointer to the logged-in user.
- registerCandidate(): Allows users to nominate themselves as candidates and stores candidate data in the candidates array.
- vote(): Allows voters to cast their votes for nominated candidates and updates vote counts.
- displayResults(): Displays the results of the election, showing candidate names, positions, and vote counts.
- main(): The central function that manages the program's flow, user interactions, and menu options.

Flow Control:

- The main() function utilizes control structures such as switch statements and loops (specifically a do-while loop) to control the program's flow.
- The menu-driven interface guides users through the available options based on their selections.

Input and Output:

- The code uses standard input/output functions (scanf and printf) for user input and displaying messages and results.
- Users input data such as usernames, passwords, and candidate choices, while the program outputs messages confirming actions and displaying results.

Constants:

Constants like MAX_USERS and MAX_CANDIDATES are defined to set limits on the number of users and candidates that can be registered in the system.

Error Handling:

The code includes basic error handling, such as checking if the maximum user or candidate limits have been reached and displaying appropriate error messages.

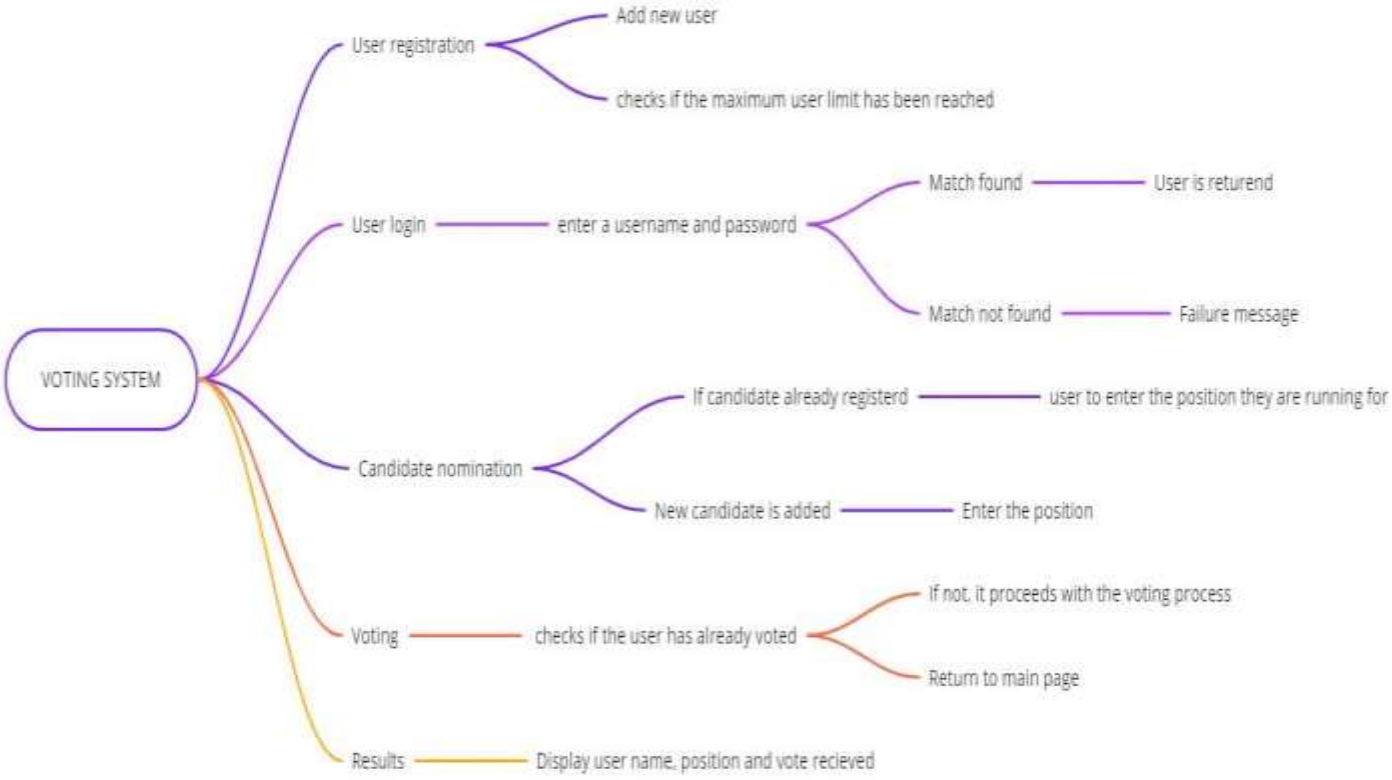
Data Flow:

Data flows through the system as users register, log in, nominate candidates, cast votes, and view election results. The data is stored in arrays and manipulated through the functions.

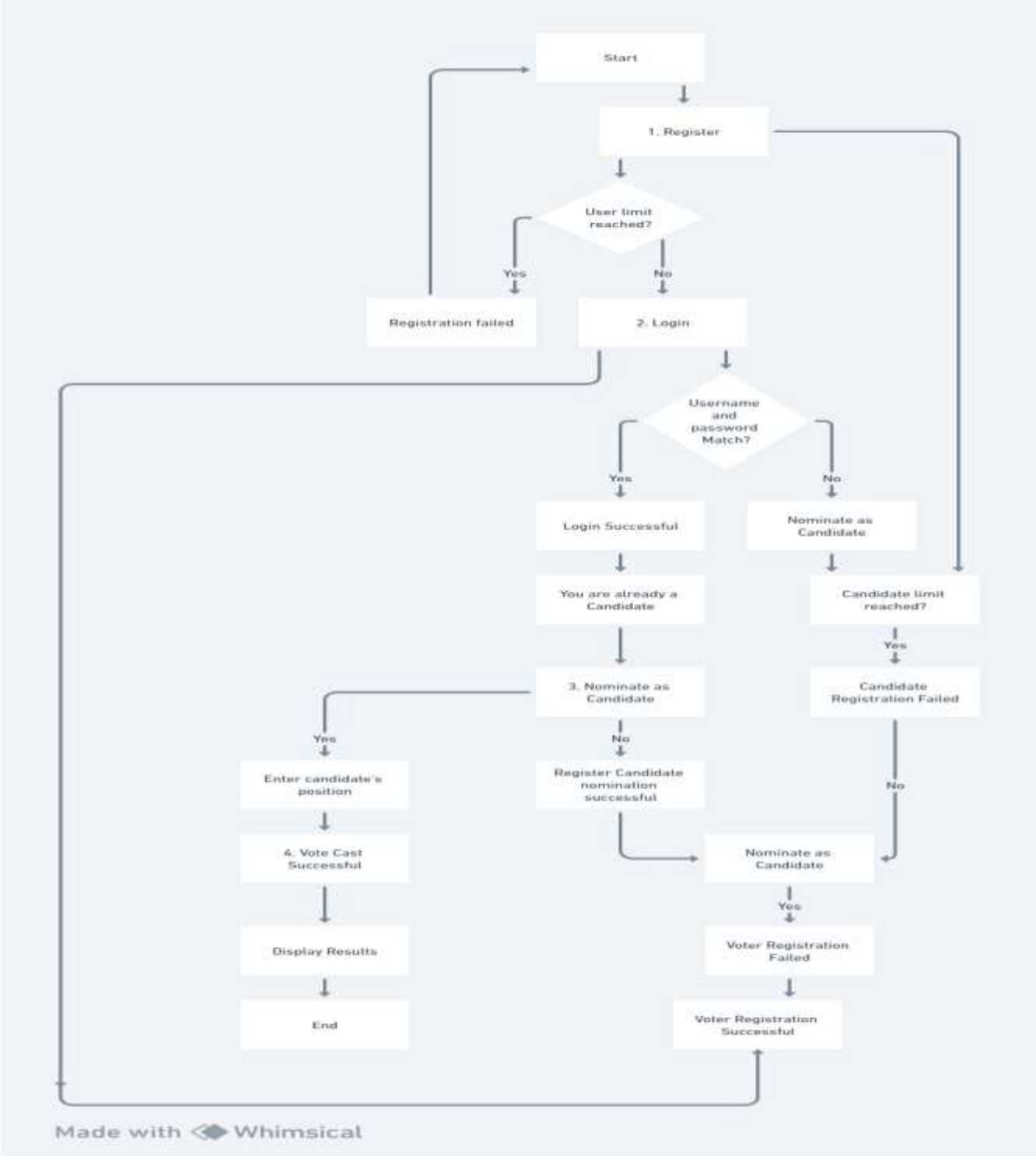
User Authentication and Authorization:

The code handles user authentication during login and authorizes users to perform certain actions based on their status as candidates or voters.

Mind Map:



Flowchart:


Start
1. Register
User limit reached?
Yes
No
Registration failed
2. Login
Username and password Match?
Yes
No
Login Successful
Nominate as Candidate
You are already a Candidate
Candidate limit reached?
Yes
3. Nominate as Candidate
Candidate Registration Failed
Yes
No
Enter candidate's position
Register Candidate nomination successful
No
4. Vote Cast Successful
Nominate as Candidate
Yes
Display Results
Voter Registration Failed
End
Voter Registration Successful
Made with Whimsical

Project Modules

User Registration Module:

- Functionality: Allows individuals to register as voters.
- Components:
    - RegisterUser() function to capture user registration data (username and password).
    - Data storage for registered users in the users array.

User Authentication Module:

- Functionality: Verifies user credentials during login.
- Components:
    - login() function to validate username and password.
    - Access control based on user type (candidate or voter).

Candidate Nomination Module:

- Functionality: Enables registered users to nominate themselves as candidates.
- Components:
    - RegisterCandidate() function for candidate registration.
    - Data storage for nominated candidates in the candidates' array.
    - Updates the user's status as a candidate.

Voting Module:

- Functionality: Allows voters to cast their votes for nominated candidates.
- Components:
    - vote() function to facilitate the voting process.
    - Ensures voters can only vote once.
    - Updates candidate vote counts.

Election Results Module:

- Functionality: Displays the results of the election, showing candidate names, positions, and vote counts.
- Components:
    - displayResults() function to present election results to the user.

Menu Interface Module:

- Functionality: Provides a menu-driven interface for users to interact with the system.
- Components:
    - The main() function that presents the main menu and handles user choices.
    - Control flow to various modules based on user selections.
    - Error handling for invalid inputs and system limits.

Data Structures Module:

- Functionality: Defines the data structures (structs) for users and candidates.
- Components:
    - User and Candidate structures to store user and candidate data.

Data Storage Module:

- Functionality: Manages data storage and retrieval.
- Components:
    - Arrays (users and candidates) to store user and candidate data.
    - Counters (userCount and candidateCount) to track the number of users and candidates.

Error Handling Module:

- Functionality: Provides error messages and handles exceptional cases.
- Components:
    - Error messages displayed to users for cases like registration limits reached.

Constants and Configuration Module:

- Functionality: Stores and manages system constants and configuration settings.
- Components:
    - Constants like MAX_USERS and MAX_CANDIDATES to define system limits.

Design and Implementation

Step 1: Define Requirements

Gather and document the requirements for your voting system. Determine what the system needs to achieve and who the users are (voters, candidates, administrators).

Step 2: Data Model

Design the data model for your system. Determine what data needs to be stored, such as user accounts, candidate information, and vote records. Create appropriate data structures or database tables for each.

Step 3: User Registration and Authentication

Implement user registration functionality where users can create accounts with unique usernames and secure passwords.

Implement user authentication to verify the identity of users when they log in.

Step 4: Candidate Nomination

Allow registered users to nominate themselves as candidates. Implement a process to collect candidate information such as their name, position, and any other relevant details.

Validate candidate eligibility, such as checking if a user is not already a candidate.

Step 5: Voting

Allow registered users to cast their votes for candidates. Ensure that users can only vote once.

Implement the logic to tally and record votes for each candidate.

Step 6: Election Results

Develop a mechanism to calculate and display election results. You may want to sort candidates by the number of votes they received to show the winner first.

Create a user-friendly interface to present the results to voters and administrators.

Step 7: Security

Ensure the security of the system. Use secure password storage methods (e.g., bcrypt), sanitize user inputs, and protect against common web vulnerabilities like SQL injection and cross-site scripting (XSS).

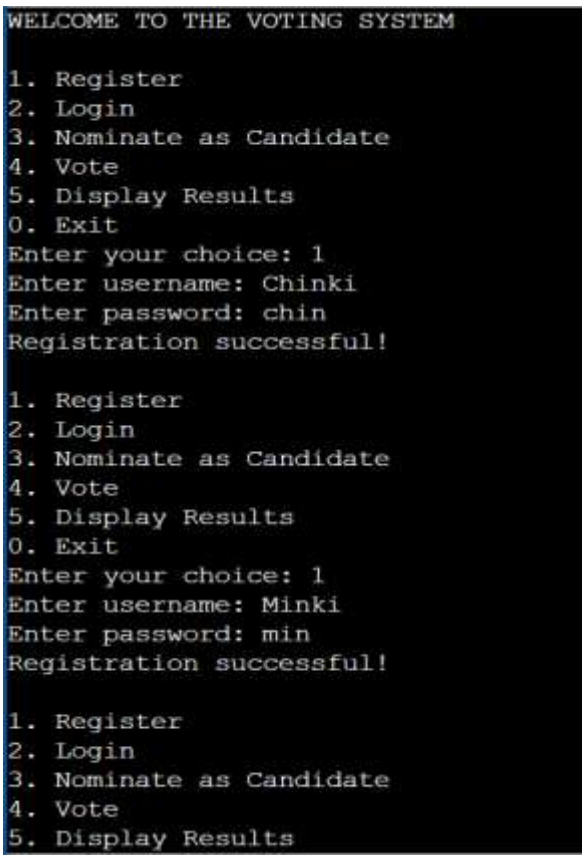
Implement access control mechanisms to restrict certain actions to authorized users only.

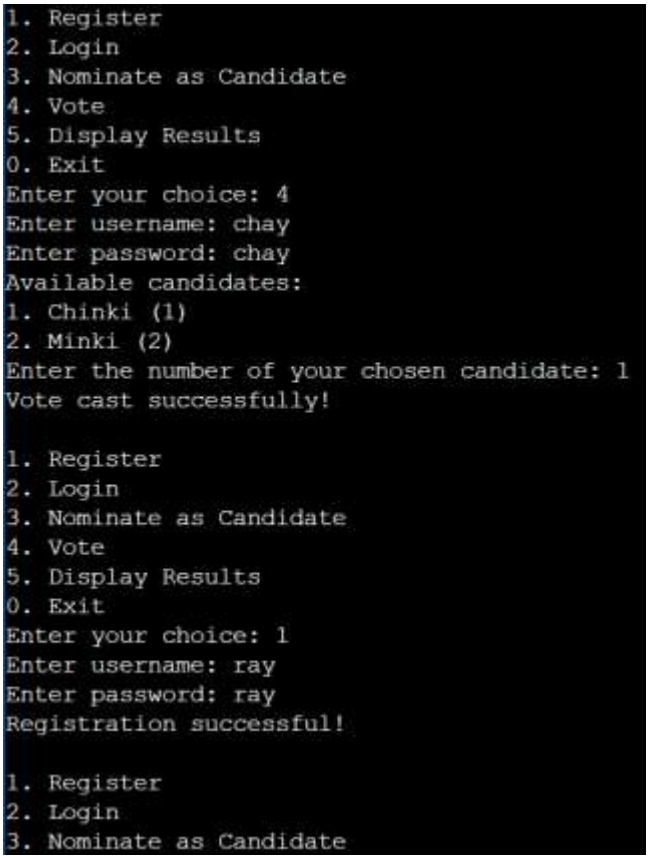
Step 8: User Interface

Design a user-friendly interface for voters to register, log in, vote, and view election results.

Design an administrative interface for managing candidate nominations and overseeing the election process.

Step 9: Testing





```
Enter your choice: 3
Enter username: Chinki
Enter password: chin
Enter candidate's position: 1
Candidate nomination successful!

1. Register
2. Login
3. Nominate as Candidate
4. Vote
5. Display Results
0. Exit
Enter your choice: 3
Enter username: Minki
Enter password: min
Enter candidate's position: 2
Candidate nomination successful!

1. Register
2. Login
3. Nominate as Candidate
4. Vote
5. Display Results
0. Exit
Enter your choice: 1
Enter username: chay
Enter password: chay
Registration successful!
```

```
4. Vote
5. Display Results
0. Exit
Enter your choice: 4
Enter username: ray
Enter password: ray
Available candidates:
1. Chinki (1)
2. Minki (2)
Enter the number of your chosen candidate: 2
Vote cast successfully!

1. Register
2. Login
3. Nominate as Candidate
4. Vote
5. Display Results
0. Exit
Enter your choice: 1
Enter username: bay
Enter password: bay
Registration successful!

1. Register
2. Login
3. Nominate as Candidate
4. Vote
5. Display Results
0. Exit
```

```
Enter your choice: 4
Enter username: bay
Enter password: bay
Available candidates:
1. Chinki (1)
2. Minki (2)
Enter the number of your chosen candidate: 1
Vote cast successfully!

1. Register
2. Login
3. Nominate as Candidate
4. Vote
5. Display Results
0. Exit
Enter your choice: 5
Election Results:
+----------------------+------------+
| Candidate Name       | Votes      |
+----------------------+------------+
| Chinki               | 2          |
| Minki                | 1          |
+----------------------+------------+
```

FEATURES AND FUNCTIONALITY

1.registerUser() Function:

•This function allows a user to register by providing a username and password.

•It checks if the maximum user limit (MAX_USERS) has been reached and displays a message if registration is not possible.

•It creates a new user object, initializes their properties, and adds them to the users array.

•After successful registration, it increments the userCount variable.

•It provides a confirmation message upon successful registration.

2.login() Function:

•This function handles user login by prompting for a username and password.

•It iterates through the users array to find a match for the provided credentials.

•If a match is found, it returns a pointer to the logged-in user; otherwise, it displays a login failure message.

•Depending on the user type (voter or candidate), it provides an appropriate login message.

3.**registerCandidate(User *user) Function:**

•This function allows a registered user (voter) to nominate themselves as a candidate.

•It checks if the maximum candidate limit (MAX_CANDIDATES) has been reached and displays a message if nomination is not possible.

•It creates a new candidate object, copying the username and password from the user who is nominating.

•It prompts for the candidate's position and initializes the number of votes received to 0.

•The new candidate is added to the candidates array, and candidateCount is incremented.

•The isCandidate flag of the nominating user is set to 1 to indicate their candidacy.

4.vote(User *voter) Function:

•This function allows a voter to cast their vote for a candidate.

•It checks if the voter has already voted and prevents multiple votes.

•It verifies if there are candidates available for voting.

•It displays the list of available candidates, showing their usernames and positions.

•The user selects a candidate by entering a number corresponding to their choice.

•The program records the vote by incrementing the candidate's vote count and setting the hasVoted flag for the voter.

•It provides a confirmation message upon successful voting.

5.displayResults() Function:

•This function displays the election results, showing the usernames of candidates, their positions, and the number of votes received.

•It iterates through the candidates array and prints the relevant information for each candidate.

6.main() Function:

•The main() function acts as the program's control center, providing a menu-driven interface.

•It repeatedly displays a menu of options, allowing the user to select one of the following actions: registration, login, candidate nomination, voting, display of results, or exit.

•Depending on the user's choice, it calls the corresponding functions.

•The loop continues until the user chooses to exit (by entering 0).

FUTURE ENHANCEMENT

- **Database Integration:**

Implement a database system to store user and candidate information securely. Using a database offers better data management, scalability, and data integrity.

- **Password Security:**

Store passwords securely by hashing and salting them. This is essential for protecting user data from security breaches.

- **User Authentication:**

Implement a more secure and sophisticated user authentication system, possibly using encryption and token-based authentication, to enhance user security.

- **Error Handling and Validation:**

Implement robust error handling and input validation to prevent common security vulnerabilities such as SQL injection. Validate user inputs for registration and login to ensure they meet certain criteria (e.g., password complexity).

- **User Roles and Permissions:**

Implement a role-based access control system to manage different user roles, such as voters and candidates. Define permissions for each role to restrict access to specific functionalities.

CONCLUSION

In conclusion, the provided voting system is a basic implementation that can serve as a starting point for further development and refinement. It offers essential features for user registration, candidate nomination, voting, and displaying election results. However, it is crucial to recognize that a production-ready voting system requires significant enhancements and considerations to meet the standards of security, usability, and legal compliance expected in real-world elections.

REFERENCES

www.pewresearch.org

www.brennancenter.org

electionlab.mit.edu