# Online Voting System

**A Project Report**

***Submitted by***

Group No : B-17


B.GEETHESH - CB.SC.U4AIE23116

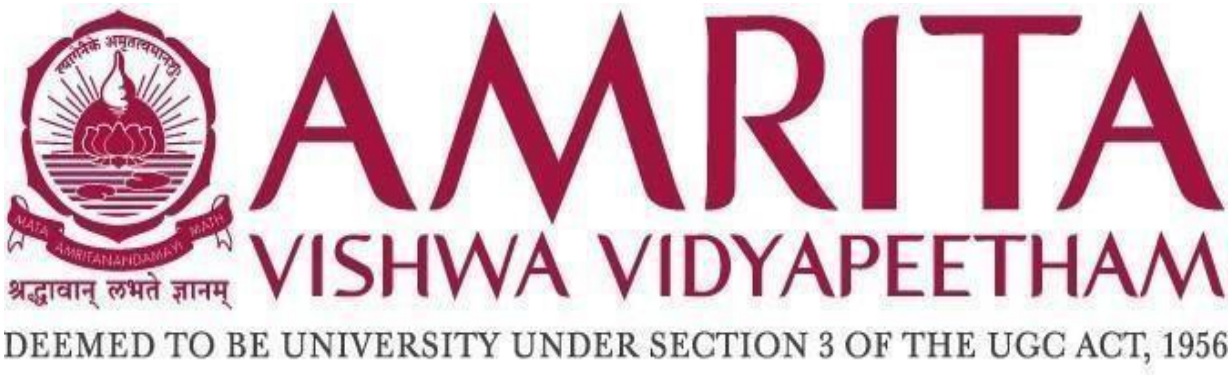
B.SAI CHARAN - CB.SC.U4AIE23117

B.VISWESH - CB.SC.U4AIE23118

K.ARUN KUMAR - CB.SC.U4AIE23137


*As a part of the subject*

22AIE101- PROBLEM SOLVING & C PROGRAMMING



# AMRITA SCHOOL OF ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112

## DECLARATION

We hereby declare that the project work entitled, “Online Voting System”

I extend our sincere and heartfelt thanks to our esteemed guide Ass.**Prof.ASWATHY P-[CEN]** mam and for her exemplary guidance, monitoring and constant encouragement throughout the course at crucial junctures and for showing usthe right way.

I would like to extend thanks to our respected Head of the department**, Dean .Dr. SOMAN K.P-[CEN]** for allowing us to use the facilities available

***Place: Amrita Vishwa Vidyapeetham, Coimbatore – 641 112***
***Date: 27-12-2023***

## ACKNOWLEDGMENT

We would like to extend our heartfelt gratitude to **Ass.Prof.ASWATHY P-[CEN]**mamfor her invaluable guidance and support throughout the development of our C program “ONLINE VOTING SYSTEM”. Her insights and mentorship have beeninstrumental in shaping this project and enhancing my understanding of the subject.

# Contents

# ABSTRACT

Voting is the one of the biggest events happen in a state or country, a large population involves in voting, and a good Voting system is necessary for an impartial election. Using C Programming Language we can develop Online Voting System, it requires basic knowledge of C like string,array,functions,etc..

The "Online Voting System" implemented in the C programming language offers a modern and secure approach to electoral processes. The project employs a structured design with functions to manage user authentication, candidate information, and voting interactions. Key features include user authentication, candidate management, and a secure file-based database system. Admin functionalities enable the addition of candidates, viewing of real-time results, and overall system management.

## Authentication :

The core functionality of the system revolves around user authentication, employing a secure mechanism that verifies user credentials before granting access to the voting interface. Registered users are stored in a structured database, encompassing essential details such as usernames, passwords, voting status, and administrative privileges.

## Candidate Management :

Administrators, designated by unique credentials, wield the capability to manage candidate information dynamically. The system accommodates the addition of new candidates, each characterized by a distinct ID, name, and position.

## Real-Time Result Display :

Administrators benefit from a real-time result display feature, offering a comprehensive overview of the ongoing election. This feature empowers administrators to make informed decisions based on voting trends, candidate performance, and overall participation.

# INTRODUCTION

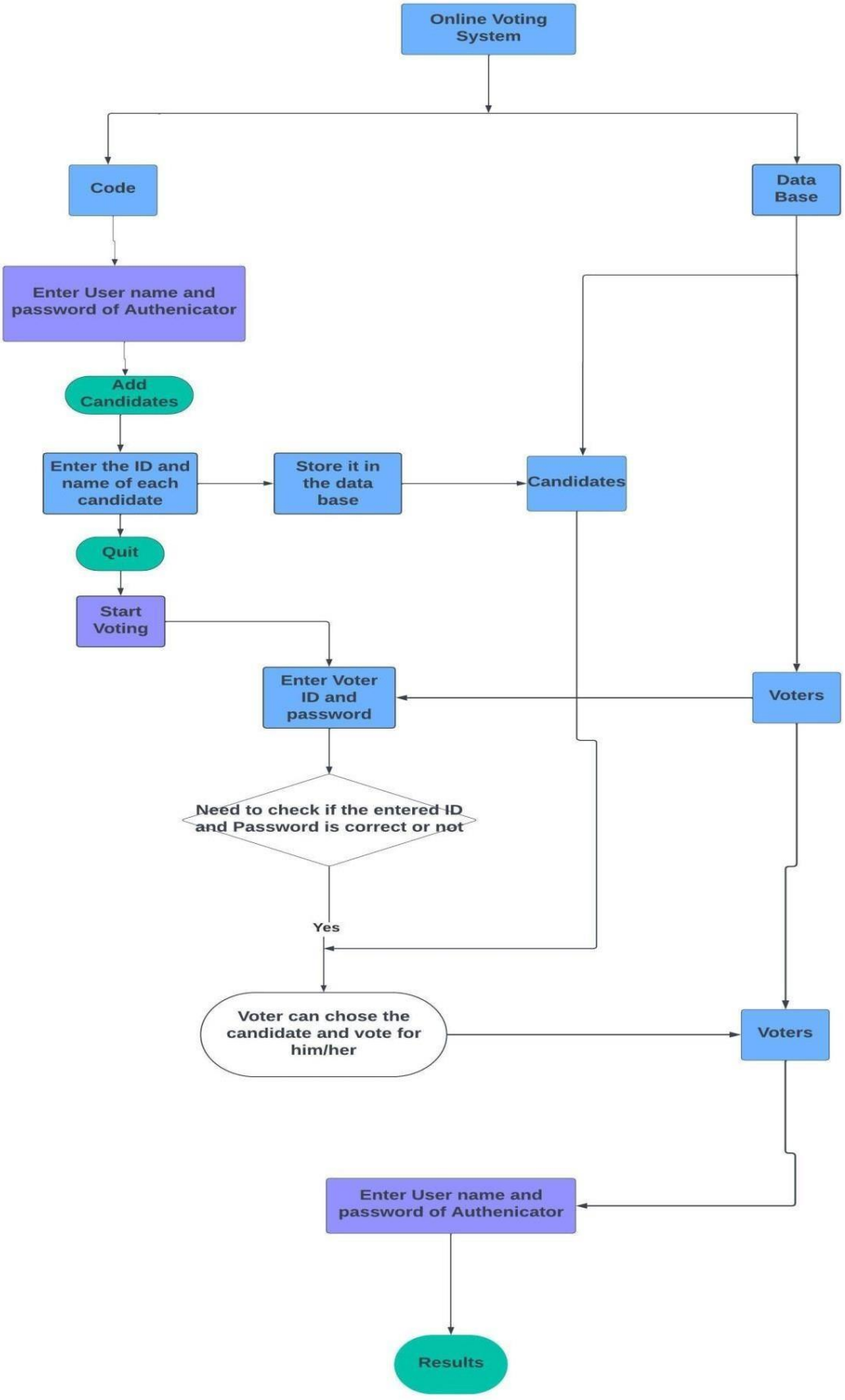
In the realm of technological advancement, the "Online Voting System" implemented in C redefines traditional election methodologies. Focused on addressing inefficiencies in paper-based systems, our project ensures a secure, transparent, and efficient online voting experience.

C's robust and efficient capabilities serve as the backbone for our system, handling critical tasks like user authentication, candidate management, and data persistence. This low-level proficiency contributes to a streamlined and reliable program, forming the basis of a secure and accessible online voting experience.

Designed with inclusivity, the project structures user and candidate databases for seamless engagement. Features like secure authentication, real-time result displays, and an interactive voting interface cater to diverse user and administrator needs, ensuring an intuitive and empowering system.

The Online Voting System addresses challenges in traditional systems by incorporating secure authentication, preventing duplicate votes, and offering real-time result displays. Administrative functionalities empower organizers to efficiently manage candidate information and monitor election progress in real-time, instilling confidence in the electoral process.

# FLOWCHART


Online Voting System
Code
Data Base
Enter User name and password of Authenicator
Add Candidates
Enter the ID and name of each candidate
Store it in the data base
Candidates
Quit
Start Voting
Enter Voter ID and password
Voters
Need to check if the entered ID and Password is correct or not
Yes
Voter can chose the candidate and vote for him/her
Voters
Enter User name and password of Authenicator
Results

# PSEUDO CODE

```
// Data Structures
struct User {
   string username[50];
   string  password[50];
   bool hasVoted;
   bool isAdmin;
};
struct Candidate {
   string id[50];
   string name[50];
   string position[50];
   int votes;
};

# User Database and Candidate Database
List<User> userDatabase;
List<Candidate> candidateDatabase;
List<hasvoted>votedusersDatabase;

 # Function to Authenticate User
function authenticateUser(username, password):
for user in userDatabase:
   if user.password == password;
      return true
    else: return false

 # Function to Authenticate Admin
function isAdmin(password):
  if Adminpassword == password;
      return true ,else:return false

# Function to Add Candidate (Password Protected))
function addCandidate(password, name, position):
   if isAdmin(password):
      candidate = AddCandidate(name, position)
      candidateDatabase.add(candidate)
      print("Candidate added successfully.")
   else: print("Invalid password. Access denied.")

 # Function to Display Voting Interface
   function authenticateUser(username, password):
    if true: UserToVote()
   hasVotedDatabase.add(user)
   elseif for user in hasvoteddatabase :
    print("your vote is already registered");
   else:("You can't vote,contact admin");
 # Function to Display Results (Password)od Pfunction displayResults(password):
   if authenticateAdmin(password):
```

```
    sortCandidatesByVotes(candidateDatabase)
    print("Election Results:")
    for candidate in candidateDatabase:
        print(“candidate.name , votes")
  else: print("Invalid password. Access denied.")
End.
```

## EXPLANATION OF CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

#define MAX_USERS 100
#define MAX_CANDIDATES 100
```

These are preprocessor directives that include necessary header files. These files provide functionalities such as input/output, memory allocation, boolean values, and string manipulation.

MAX_USERS and MAX_CANDIDATES are defined as constants, representing the maximum number of users and candidates the system can handle.

```c
struct User {
    char username[50];
    char password[50];
    bool hasVoted;
    bool isAdmin;
};

struct Candidate {
    char id[50];
    char name[50];
    char position[50];
    int votes;
};
```

Struct User, which represents user data. It includes fields for username, password, a flag indicating if the user has voted (hasVoted), and a flag indicating if the user is an admin (isAdmin).

Definition of struct Candidate, which represents candidate data. It includes fields for candidate ID, name, position, and the number of votes received.

```c
// Function prototypes
int readUserDatabase(struct User *userDatabase, const char *filename);
void writeUserDatabase(const struct User *userDatabase, int count, const char *filename);
int readCandidateDatabase(struct Candidate *candidateDatabase, const char *filename);
void writeCandidateDatabase(const struct Candidate *candidateDatabase, int count, const char *filename);
bool authenticateUser(struct User *userDatabase, int userCount, char *username, char *password);
bool hasUserVoted(struct User *userDatabase, int userCount, char *username);
bool isAdmin(struct User *userDatabase, int userCount, char *adminUsername, char *adminPassword);
void addCandidates(struct Candidate *candidateDatabase, int *candidateCount, struct User *userDatabase, int userCount, char *adminUsername, char *adminPassword);
void displayVotingInterface(struct User *userDatabase, int userCount, struct Candidate *candidateDatabase, int candidateCount, char *username);
void displayResults(struct Candidate *candidateDatabase, int candidateCount, struct User *userDatabase, int userCount, char *adminUsername, char *adminPassword);
void getUserCredentials(char *username, char *password);

```

Function prototypes declare the functions used in the program. These include functions for reading and writing user and candidate databases, user authentication, admin authorization, candidate addition, and displaying the voting interface and election results.

```c
// Function to read user database from file
int readUserDatabase(struct User *userDatabase, const char *filename) {
    // ... (previous code)
    FILE *file = fopen(filename, "r");
    if (!file) {
        perror("Error opening user database file");
        return -1;
    }
    int count = 0;
    while (fscanf(file, "%s %s %d %d\n", userDatabase[count].username, userDatabase[count].password,
                  &userDatabase[count].hasVoted, &userDatabase[count].isAdmin) == 4) {
        count++;
    }
    fclose(file);
    return count;
}
```

The readUserDatabase function reads user data from a file and populates the userDatabase array. It returns the number of users read from the file or -1 in case of an error.

```c
// Function to write user database to file
void writeUserDatabase(const struct User *userDatabase, int count, const char *filename) {
    // ... (previous code)
    FILE *file = fopen(filename, "w");
    if (!file) {
        perror("Error opening user database file");
        return;
    }

    for (int i = 0; i < count; i++) {
        fprintf(file, "%s %s %d %d\n", userDatabase[i].username, userDatabase[i].password,
                userDatabase[i].hasVoted, userDatabase[i].isAdmin);
    }

    fclose(file);
}
```

The **writeUserDatabase** function writes user data to a file based on the information in the **userDatabase** array.

```
69  // Function to read candidate database from file
70  int readCandidateDatabase(struct Candidate *candidateDatabase, const char *filename) {
71      // ... (previous code)
72      FILE *file = fopen(filename, "r");
73      if (!file) {
74          perror("Error opening candidate database file");
75          return -1;
76      }
77
78      int count = 0;
79      while (fscanf(file, "%s %s %s %d\n", candidateDatabase[count].id, candidateDatabase[count].name,
80                    candidateDatabase[count].position, &candidateDatabase[count].votes) == 4) {
81          count++;
82      }
83
84      fclose(file);
85      return count;
86  }
87  // Function to write candidate database to file
88  void writeCandidateDatabase(const struct Candidate *candidateDatabase, int count, const char *filename)
89      // ... (previous code)
90      FILE *file = fopen(filename, "w");
91      if (!file) {
92          perror("Error opening candidate database file");
93          return;
94      }
95      else{
96      for (int i = 0; i < count; i++) {
97          fprintf(file, "%s %s %s %d\n", candidateDatabase[i].id, candidateDatabase[i].name,
98                  candidateDatabase[i].position, candidateDatabase[i].votes);
99      }
100     }
101
102     fclose(file);
103 }
104
```

The **readCandidateDatabase** function reads candidate data from a file and populates the **candidateDatabase** array. It returns the number of candidates read from the file or -1 in case of an error. The **writeCandidateDatabase** function writes candidate data to a file based on the information in the **candidateDatabase** array.

```c
bool authenticateUser(struct User *userDatabase, int userCount, char *username, char *password) {
    for (int i = 0; i < userCount; i++) {
        if(strcmp(userDatabase[i].username,username)==0&&strcmp(userDatabase[i].password,password)==0){
            return true;
        }
    }
    return false;
}

// Function to check if the user has already voted
bool hasUserVoted(struct User *userDatabase, int userCount, char *username) {
    for (int i = 0; i < userCount; i++) {
        if (strcmp(userDatabase[i].username, username) == 0 && userDatabase[i].hasVoted) {
            return true;
        }
    }
    return false;
}
```

The **authenticateUser** function checks if a given username and password match any user in the **userDatabase** array.The **hasUserVoted** function checks if a user with a given username has already voted based on the information in the **userDatabase** array.

```c
// Function to authenticate admin
bool isAdmin(struct User *userDatabase, int userCount, char *adminUsername, char *adminPassword) {
    for (int i = 0; i < userCount; i++) {
        if (strcmp(userDatabase[i].username,adminUsername)==0&&strcmp(userDatabase[i].password,adminPassword)==0&&userDatabase[i].isAdmin){
            return true;
        }
    }
    return false;
}
```

The isAdmin function checks if a user with a given username and password is an admin based on the information in the userDatabase array.

```c
// Function to add candidates (Password protected)
void addCandidates(struct Candidate *candidateDatabase,int *candidateCount,struct User *userDatabase,int userCount,char *adminUsername,char *adminPassword){
    if (isAdmin(userDatabase, userCount, adminUsername, adminPassword)) {
        int numCandidates;
        printf("Enter the number of candidates: ");
        scanf("%d", &numCandidates);

        for (int i = 0; i < numCandidates; i++) {
            printf("Enter Candidate ID for candidate %d: ", i + 1);
            scanf("%s", candidateDatabase[*candidateCount].id);

            printf("Enter Candidate Name for candidate %d: ", i + 1);
            scanf(" %[^\n]s", candidateDatabase[*candidateCount].name);

            printf("Enter Candidate Position for candidate %d: ", i + 1);
            scanf(" %s", candidateDatabase[*candidateCount].position);

            candidateDatabase[*candidateCount].votes = 0;
            (*candidateCount)++;
        }

        // Write updated candidate database to file
        writeCandidateDatabase(candidateDatabase, *candidateCount, "C:\\Users\\viswe\\Downloads\\c project\\candidates.txt");

        printf("Candidates added successfully.\n");
    } else {
        printf("Invalid credentials. Only admin can add candidates.\n");
    }
}
```

The addCandidates function allows an admin to add candidates to the system. It prompts the admin for the number of candidates, collects candidate information, updates the candidate database, and writes the changes to the file.

```c
void displayVotingInterface(struct User *userDatabase, int userCount,struct Candidate*candidateDatabase, int candidateCount, char*username){
    if (hasUserVoted(userDatabase, userCount, username)) {
        printf("Your vote is already registered. You cannot vote again.\n");
    } else {
        printf("Welcome to the voting interface, %s! You can cast your vote now.\n", username);

        // Display the list of candidates
        printf("List of Candidates:\n");
        for (int i = 0; i < candidateCount; i++) {
            printf("%s. %s - %s\n", candidateDatabase[i].id, candidateDatabase[i].name, candidateDatabase[i].position);
        }

        // Get user's vote
        printf("Enter the ID of the candidate you want to vote for: ");
        char selectedCandidateID[50];
        scanf("%s", selectedCandidateID);

        // Find the selected candidate in the database
        int candidateIndex = -1;
        for (int i = 0; i < candidateCount; i++) {
            if (strcmp(candidateDatabase[i].id, selectedCandidateID) == 0) {
                candidateIndex = i;
                break;
            }
        }

        // Check if the candidate ID was found
        if (candidateIndex != -1) {
            // Update candidate's vote count
            candidateDatabase[candidateIndex].votes++;

            // Update user's voting status
            for (int j = 0; j < userCount; j++) {
                if (strcmp(userDatabase[j].username, username) == 0) {
                    userDatabase[j].hasVoted = true;
                    break;
                }
            }
            // Write updated databases to files
            writeUserDatabase(userDatabase, userCount, "C:\\Users\\viswe\\Downloads\\c project\\users.txt");
            writeCandidateDatabase(candidateDatabase, candidateCount, "C:\\Users\\viswe\\Downloads\\c project\\candidates.txt");
            printf("Your vote has been registered. Thank you!\n");
        } else {
            // If the selected candidate ID was not found
            printf("Invalid candidate ID. Vote not registered.\n");
        }
    }
}
```

userDatabase: Array of User structures containing user information.

userCount: Number of users in the user database.

candidateDatabase: Array of Candidate structures containing candidate information.

candidateCount: Number of candidates in the candidate database.

username: Username of the user accessing the voting interface.

Check if User Has Already Voted:

Use the hasUserVoted function to check if the user with the given username has already voted.

If the user has voted, print a message indicating that the vote is already registered, and exit the function.

Display Welcome Message:

If the user has not voted, display a welcome message, addressing the user by their username.

Display List of Candidates:

Iterate through the candidateDatabase array and print the ID, name, and position of each candidate.

The user will choose a candidate based on the displayed information.

Get User's Vote:

Prompt the user to enter the ID of the candidate they want to vote for.

Find Selected Candidate:

Search the candidateDatabase array to find the candidate with the entered ID.

If found, note the index (candidateIndex) of the selected candidate; otherwise, set candidateIndex to -1.

Update Candidate's Vote Count:

If candidateIndex is not -1, increment the votes attribute of the selected candidate in the candidateDatabase array.

Update User's Voting Status:

Search the userDatabase array for the user with the given username.

If found, set the hasVoted attribute of that user to true.

Write Updated Databases to Files:

Call the writeUserDatabase function to update the user database file with the modified userDatabase.

Call the writeCandidateDatabase function to update the candidate database file with the modified candidateDatabase.

Print Confirmation Message:

Print a message confirming that the user's vote has been registered.

```
void displayResults(struct Candidate *candidateDatabase, int candidateCount, struct User *userDatabase, int userCount, char *adminUsername, char *adminPassword) {
    if (isAdmin(userDatabase, userCount, adminUsername, adminPassword)) {
        // Calculate total votes cast
        int totalVotes = 0;
        for (int i = 0; i < candidateCount; i++) {
            totalVotes += candidateDatabase[i].votes;
        }
```

The function takes as input the candidate and user databases, along with admin credentials.

It checks if the user with provided credentials is an admin by calling the isAdmin function.

If the user is an admin, it calculates the total votes cast by summing up the votes for each candidate.

```
// Display the election results
printf("Election Results:\n");
int maxVotes = 0;  // To keep track of the maximum number of votes
int winnerIndex = -1;  // To store the index of the candidate with the maximum votes

for (int i = 0; i < candidateCount; i++) {
    double percentage = (double)candidateDatabase[i].votes / totalVotes * 100;
    printf("%s. %s - %s - Votes: %d - Percentage: %.2f%%\n", candidateDatabase[i].id, candidateDatabase[i].name, candidateDatabase[i].position, candidateDatabase[i].votes, percentage);

    // Determine the winner
    if (candidateDatabase[i].votes > maxVotes) {
        maxVotes = candidateDatabase[i].votes;
        winnerIndex = i;
    }
}
```

It then initializes variables to keep track of the maximum votes and the index of the winning candidate. It iterates through each candidate, calculates the percentage of votes they received, and prints their details.

While iterating, it identifies the candidate with the maximum votes and updates the **winnerIndex** accordingly.

```
    // Display the winner
    if (winnerIndex != -1) {
        printf("\nWinner:\n");
        printf("%s. %s - %s - Votes: %d\n", candidateDatabase[winnerIndex].id, candidateDatabase[winnerIndex].name, candidateDatabase[winnerIndex].position, candidateDatabase[winnerIndex].votes);
    } else {
        printf("\nNo winner determined.\n");
    }

} else {
    printf("Invalid credentials. Only admin can view results.\n");
}
```

After the iteration, it checks if a winner was determined and prints the winner's details if found, otherwise, it indicates that no winner is determined.
If the user is not an admin, it prints a message indicating that only admins can view the results.

```c
void getUserCredentials(char *username, char *password) {
    printf("Enter your username: ");
    scanf("%s", username);

    printf("Enter your password: ");
    scanf("%s", password);
}


void runVotingSystem();

int main() {
    runVotingSystem();
    return 0;
}
```

This function prompts the user to enter their username and password and reads the input using **scanf**.

```c
void runVotingSystem() {
    // Paths to database files
    const char userDatabasePath[] = "C:\\Users\\viswe\\Downloads\\c project\\users.txt";
    const char candidateDatabasePath[] = "C:\\Users\\viswe\\Downloads\\c project\\candidates.txt";

    while (1) { // Start an infinite loop
        // Read user and candidate databases from files
        struct User userDatabase[MAX_USERS];
        int userCount = readUserDatabase(userDatabase, userDatabasePath);

        struct Candidate candidateDatabase[MAX_CANDIDATES];
        int candidateCount = readCandidateDatabase(candidateDatabase, candidateDatabasePath);

        // Get user credentials
        char username[50];
        char password[50];
        getUserCredentials(username, password);

        // Authenticate user
        if (isAdmin(userDatabase, userCount, username, password)) {
            // Admin authentication successful
            printf("Admin authentication successful. You may proceed.\n");
```

The function begins by defining the paths to the user and candidate database files. Initiates an infinite loop to keep the voting system running until the user chooses to quit. Reads user and candidate databases from the respective files and stores the count in userCount and candidateCount.

Prompts the user for their credentials using the **getUserCredentials** function. Authenticates the user, checks if they are an admin using the **isAdmin** function.

If the user is an admin, prints a success message.

```
int choice;
do {
    // Display admin options
    printf("Choose an option:\n");
    printf("1. Add Candidates\n");
    printf("2. View Results\n");
    printf("3. Quit\n");

    scanf("%d", &choice);

    switch (choice) {
        case 1:
            // Add candidates
            addCandidates(candidateDatabase, &candidateCount, userDatabase, userCount, username, password);
            // Write updated candidate database to file
            writeCandidateDatabase(candidateDatabase, candidateCount, candidateDatabasePath);
            // Break out of the *switch to prompt admin credentials again
            break;
        case 2:
            // View results
            displayResults(candidateDatabase, candidateCount, userDatabase, userCount, username, password);
            break;
        case 3:
            // Quit the loop
            printf("Quitting...\n");
            exit(0); // Exit the program
            break;
        default:
            printf("Invalid choice. Try again.\n");
    }
} while (choice != 3); // Loop until the user chooses to quit
else {
  // Regular user authentication
  int userIndex = -1;
  for (int i = 0; i < userCount; i++) {
      if (strcmp(userDatabase[i].username, username) == 0 && strcmp(userDatabase[i].password, password) == 0) {
          userIndex = i;
          break;
      }
  }
```

Displays admin options in a loop until the user chooses to quit.

Calls the addCandidates function to add candidates and updates the candidate database file.

Calls the **displayResults** function to show election results.

Exits the program if the user chooses to quit.

If the user is not an admin, the program moves to regular user authentication.

Checks if the entered user credentials match any user in the database.

```
    if (userIndex != -1) {
        // User found in the database
        if (hasUserVoted(userDatabase, userCount, username)) {
            // User has already voted
            printf("Your vote is already registered. You cannot vote again.\n");
        } else {
            // User has not voted yet
            displayVotingInterface(userDatabase, userCount, candidateDatabase, candidateCount, username);

            // Update user's voting status
            userDatabase[userIndex].hasVoted = true;

            // Write updated user database to file
            writeUserDatabase(userDatabase, userCount, userDatabasePath);
        }
    } else {
        // User not found, contact admin
        printf("User not found. Please contact the admin for assistance.\n");
    }
}
```

If the user is found and has not voted, it displays the voting interface.

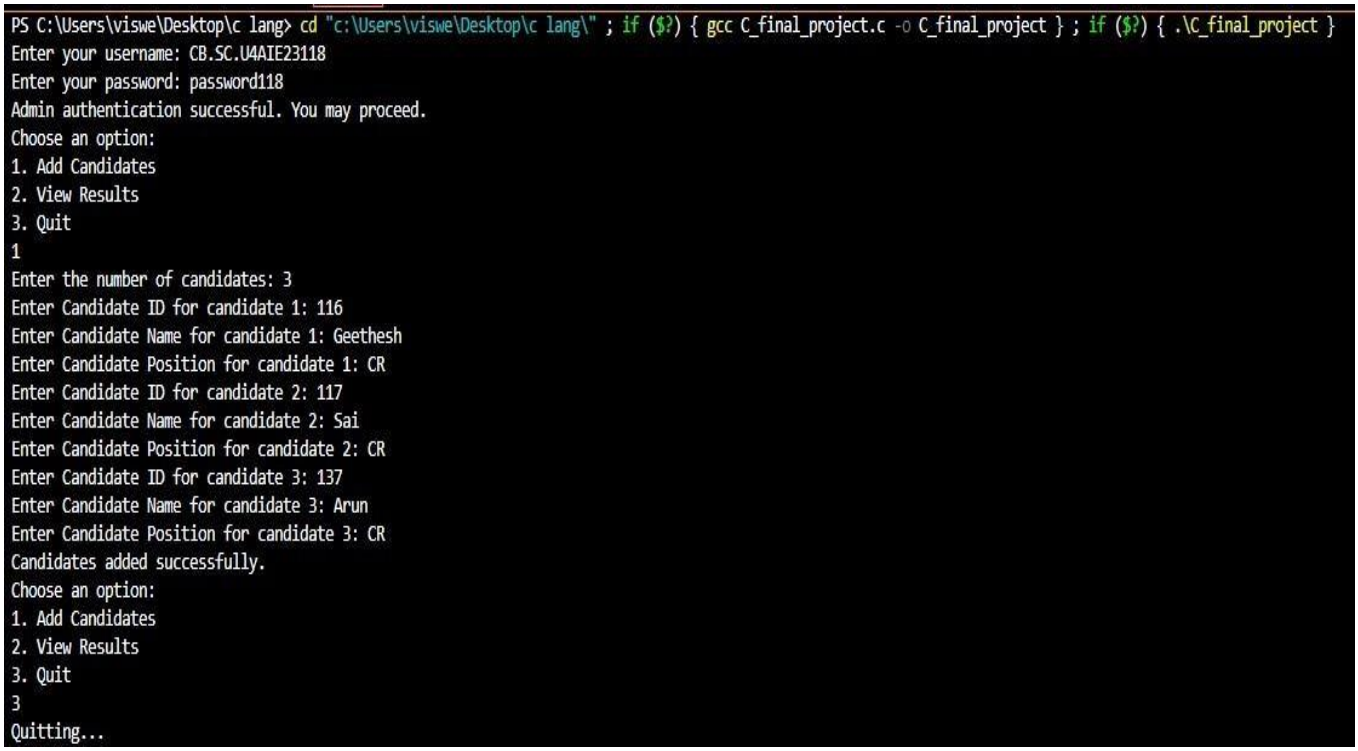
Updates the user's voting status and writes the updated user database to the file.

If the user is not found, it suggests contacting the admin for assistance.

Calls the **runVotingSystem** function to start the voting system.

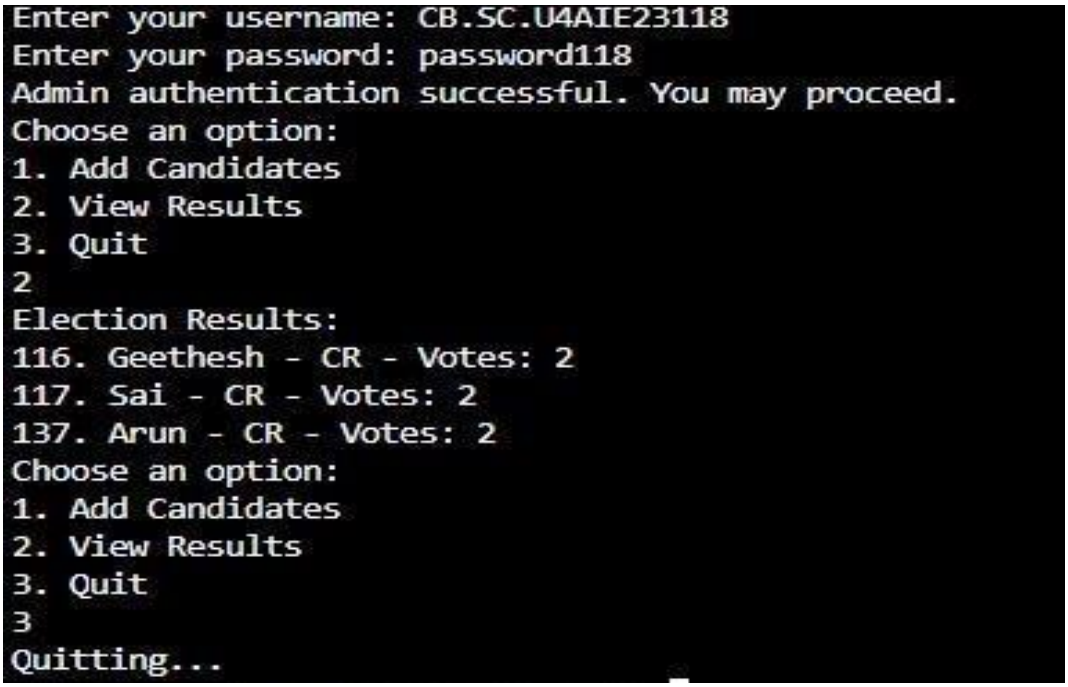
# RESULTS

Terminal Output:



```
PS C:\Users\viswe\Desktop\c lang> cd "c:\Users\viswe\Desktop\c lang\" ; if ($?) { gcc C_final_proj
Enter your username: CB.SC.U4AIE23101
Enter your password: password101
Welcome to the voting interface, CB.SC.U4AIE23101! You can cast your vote now.
List of Candidates:
116. Geethesh - CR
117. Sai - CR
137. Arun - CR
Enter the ID of the candidate you want to vote for: 116
Your vote has been registered. Thank you!
Enter your username: CB.SC.U4AIE23102
Enter your password: password102
Welcome to the voting interface, CB.SC.U4AIE23102! You can cast your vote now.
List of Candidates:
116. Geethesh - CR
117. Sai - CR
137. Arun - CR
Enter the ID of the candidate you want to vote for: 117
Your vote has been registered. Thank you!
Enter your username: CB.SC.U4AIE23103
Enter your password: password103
Welcome to the voting interface, CB.SC.U4AIE23103! You can cast your vote now.
List of Candidates:
116. Geethesh - CR
117. Sai - CR
137. Arun - CR
Enter the ID of the candidate you want to vote for: 116
Your vote has been registered. Thank you!
```

```
Enter your username: CB.SC.U4AIE23104
Enter your password: password104
Welcome to the voting interface, CB.SC.U4AIE23104! You can cast your vote now.
List of Candidates:
116. Geethesh - CR
117. Sai - CR
137. Arun - CR
Enter the ID of the candidate you want to vote for: 137
Your vote has been registered. Thank you!
Enter your username: CB.SC.U4AIE23101
Enter your password: password101
Your vote is already registered. You cannot vote again.
Enter your username: CB.SC.U4AIE23105
Enter your password: Password105
User not found. Please contact the admin for assistance.
Enter your username: CB.SC.U4AIE23105
Enter your password: password105
Welcome to the voting interface, CB.SC.U4AIE23105! You can cast your vote now.
List of Candidates:
116. Geethesh - CR
117. Sai - CR
137. Arun - CR
Enter the ID of the candidate you want to vote for: 117
Your vote has been registered. Thank you!
Enter your username: CB.SC.U4AIE23001
Enter your password: password001
User not found. Please contact the admin for assistance.
Enter your username: CB.SC.U4AIE23111
Enter your password: password111
Welcome to the voting interface, CB.SC.U4AIE23111! You can cast your vote now.
List of Candidates:
116. Geethesh - CR
117. Sai - CR
137. Arun - CR
Enter the ID of the candidate you want to vote for: 137
Your vote has been registered. Thank you!
```


Enter your username: CB.SC.U4AIE23118
Enter your password: password118
Admin authentication successful. You may proceed.
Choose an option:
1. Add Candidates
2. View Results
3. Quit
2
Election Results:
116. Geethesh - CR - Votes: 2
117. Sai - CR - Votes: 2
137. Arun - CR - Votes: 2
Choose an option:
1. Add Candidates
2. View Results
3. Quit
3
Quitting...

## Explanation of Result:

```
PS C:\Users\viswe\Desktop\c lang> cd "c:\Users\viswe\Desktop\c lang\" ; if ($?) { gcc C
Enter your username: CB.SC.U4AIE23118
Enter your password: password118
Admin authentication successful. You may proceed.
```

Here, the program asks for admin credentials. The entered username is **CB.SC.U4AIE23118**, and the password is **password118**. The authentication is successful, and the admin is allowed to proceed.

```
PS C:\Users\viswe\Desktop\c lang> cd "c:\Users\viswe\Desktop\c lang\" ; if ($?) { gcc C
Enter your username: CB.SC.U4AIE23118
Enter your password: password118
Admin authentication successful. You may proceed.
Choose an option:
1. Add Candidates
2. View Results
3. Quit
1
Enter the number of candidates: 3
Enter Candidate ID for candidate 1: 116
Enter Candidate Name for candidate 1: Geethesh
Enter Candidate Position for candidate 1: CR
Enter Candidate ID for candidate 2: 117
Enter Candidate Name for candidate 2: Sai
Enter Candidate Position for candidate 2: CR
Enter Candidate ID for candidate 3: 137
Enter Candidate Name for candidate 3: Arun
Enter Candidate Position for candidate 3: CR
Candidates added successfully.
Choose an option:
1. Add Candidates
2. View Results
3. Quit
3
```

The admin chooses option 1 to add candidates. Three candidates are added with IDs 116, 117, and 137, along with their names and positions (CR). The information is then stored in the candidate's database.

```
PS C:\Users\viswe\Desktop\c lang> cd "c:\Users\viswe\Desktop\c lang\" ; if ($?) { gcc C_final_pro
Enter your username: CB.SC.U4AIE23101
Enter your password: password101
Welcome to the voting interface, CB.SC.U4AIE23101! You can cast your vote now.
List of Candidates:
116. Geethesh - CR
117. Sai - CR
137. Arun - CR
Enter the ID of the candidate you want to vote for: 116
Your vote has been registered. Thank you!
```

A regular user with the username **CB.SC.U4AIE23101** logs in and casts a vote for candidate 116 (Geethesh). The user is informed that the vote has been registered.

```
Enter your username: CB.SC.U4AIE23101
Enter your password: password101
Your vote is already registered. You cannot vote again.
```

The same user attempts to vote again, but the system recognizes that the user has already voted and informs them that they cannot vote again.

```
Enter your username: CB.SC.U4AIE23105
Enter your password: Password105
User not found. Please contact the admin for assistance.
Enter your username: CB.SC.U4AIE23105
Enter your password: password105
Welcome to the voting interface, CB.SC.U4AIE23105! You can cast your vote now.
List of Candidates:
116. Geethesh - CR
117. Sai - CR
137. Arun - CR
Enter the ID of the candidate you want to vote for: 117
Your vote has been registered. Thank you!
```

An attempt is made to log in with an incorrect password (Password105) for a user (**CB.SC.U4AIE23105**). The system informs that the user is not found and suggests contacting the admin for assistance.Then the same user log in with correct password(password105) and casts a vote for candidate 117 (Sai).

```
Enter your username: CB.SC.U4AIE23118
Enter your password: password118
Admin authentication successful. You may proceed .
Choose an option:
1. Add Candidates
2. View Results
3. Quit
2
Election Results:
116. Geethesh - CR - Votes: 2 - Percentage: 33.33%
117. Sai - CR - Votes: 2 - Percentage: 33.33%
137. Arun - CR - Votes: 2 - Percentage: 33.33%

No winner determined.
```

The authenticating admin then views the updated election results, showing that candidates 116,117,137 now has 2 votes and each having voting percentage of 33.33%. None of them was the winner so ‘No Winner Determined’. The admin chooses option 3 to quit the program, program exits.

```
Enter your username: CB.SC.U4AIE23118
Enter your password: password118
Admin authentication successful. You may proceed .
Choose an option:
1. Add Candidates
2. View Results
3. Quit
2
Election Results:
116. Geethesh - CR - Votes: 2 - Percentage: 28.57%
117. Sai - CR - Votes: 2 - Percentage: 28.57%
137. Arun - CR - Votes: 3 - Percentage: 42.86%

Winners:
137. Arun - CR - Votes: 3
Choose an option:
1. Add Candidates
2. View Results
3. Quit
3
Quitting...
```

The authenticating admin then views the updated election results, showing that candidates 116,117 now has 2 votes each having voting percentage of 28.57%, 137 now has 3 votes with voting percentage of 42.86%. The winner is Arun, admin chooses option 3 to quit the program, program exits.

This output demonstrates the functionality of the voting system, including admin authentication, adding candidates, user authentication, voting, and viewing election results.

## CONCLUSION

The provided C code implements a simple online voting system with user and admin

functionalities. The program employs modular functions and structures to manage user and candidate databases, read and write data to files, and authenticate users based on their roles. The key features include:

User Authentication: The system authenticates users as either regular voters or administrators based on provided credentials.

Admin Options: Administrators have the authority to add candidates, view election results, and quit the program. Admin authentication is required to access these options.

Election Results Display: The program calculates and displays the election results, showing the percentage of votes each candidate received and declaring the winner.

Voting Interface: Regular users can vote if they have not done so already. The system prevents users from voting multiple times.

Modularity: The code is well-organized into functions, promoting code readability, reusability, and ease of maintenance.

File Handling: User and candidate data are read from and written to files, ensuring persistence of data between program runs.

Infinite Loop: The program runs in an infinite loop, allowing users to perform multiple actions until they choose to exit.

Command-Line Interface: Interaction with the program occurs through the command line, with prompts for credentials and display of voting options.

While the code provides a foundational structure for an online voting system, it is important to note that security considerations (e.g., encryption of passwords) and additional features (e.g., handling ties in the election) would be necessary for a complete and secure implementation. Additionally, error handling and input validation could be enhanced for robustness. Overall, the code serves as a starting point for building a more comprehensive and secure online voting system.

**REFERENCES:**

**https://www.geeksforgeeks.org/online-voting-system-in-c/**

**https://www.studytonight.com/c-projects/election-system-project-using-c-language**