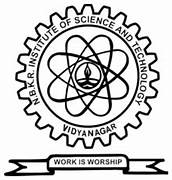
N.B.K.R INSTITUTE OF SCIENCE AND TECHNOLOGY

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that project your simple voting system being Submitted by

THALIYAKULA LIKITHA - 24KB1A05KW

THOPPANI TEJASRI - 24KB1A05LF

TUPAKULA SARIKA - 24KB1A05LN

SULLURU TEJASRI - 24KB1A05K

In partial fulfillment for the award of the degree of technology in computer science andEngineering to the N.B.K.R.Institute of Science and Tecnology, vidyanagar is a record of Bonafied work carried out under my guidance and supervision.

Padavala suneetha Dr.Ravinder Reddy

M.Tech,Ph.D

Head of the Department

Desigination

DECLARATION

I hereby declare that the dissertation entitled simple voting system submitted for the B.Tech Degree is my original work and dissertation has not formed the basic for the award Of any degree, associateship, fellowship or any other saimilar tittles.

Place:

Date:

**Acknowledgement**

I take this opportunity to express my heartfelt gratitude to all those who have helped and guided me in the successful completion of this project titled **“SIMPLE VOTING SYSTEM.”**

## First and foremost, I am deeply thankful to P Suneetha my project mentor, for their invaluable support, expert guidance, and constant encouragement throughout the development of this project. Their insightful suggestions, constructive feedback, and patient supervision played a crucial role in shaping this work to its final form.

I would also like to extend my sincere thanks to the **Head of the Department, [Dr.A Raja Sekhar Reddy]**, and all the faculty members of the **Department of [CSE]**, **[N.B.K.R Institute of Science and Technology]**, for providing the required resources, academic knowledge, and moral support during the course of the project.

A special mention goes to my peers and classmates, whose collaboration and thoughtful discussions contributed significantly to refining the project’s ideas and implementation.

I am also grateful to the various online forums, documentation, and learning platforms that served as references during the development of the system. Their resources helped in enhancing my technical understanding and troubleshooting challenges.

Finally, I owe a deep sense of appreciation to my family for their constant motivation, emotional support, and patience throughout this journey. Their belief in my capabilities was a source of great strength.

This project has not only enriched my academic learning but has also provided practical exposure and a deeper understanding of real-world system development.

Abstract

This project presents a simple Voting System designed to facilitate fair and efficient election processes using a computerized approach. The system allows registered users to cast their votes electronically in a secure and transparent manner. It ensures that each user can vote only once, maintains voter privacy, and prevents unauthorized access. Key functionalities include voter registration, candidate listing, vote casting, and result generation. The system aims to reduce manual errors, enhance the speed of counting, and promote a more accessible and reliable voting experience. This project is ideal for small-scale elections such as in schools, colleges, or local organizations.

**1. Introduction**

* **Problem Statement**: In many small-scale elections, such as those conducted in schools, colleges, clubs, or local communities, traditional voting methods like paper ballots are still commonly used. These methods are often time-consuming, error-prone, and lack transparency. Manual counting of votes can lead to delays and inconsistencies, and there's also a risk of duplicate voting or manipulation.
* There is a need for a simple, secure, and reliable electronic voting system that can streamline the voting process, ensure accuracy, and prevent duplicate votes. The system should allow voters to cast their votes easily and enable administrators to manage the election process efficiently, including viewing and tallying the results automatically.
* This project aims to develop a Simple Voting System that digitizes the voting process, minimizes human error, and increases the efficiency and integrity of small-scale elections.

**. scope:** In many small-scale elections, such as those conducted in schools, colleges, clubs, or local communities, traditional voting methods like paper ballots are still commonly used.

**.** These methods are often time-consuming, error-prone, and lack transparency. Manual counting of votes can lead to delays and inconsistencies, and there's also a risk of duplicate voting or manipulation.

**.**There is a need for a simple, secure, and reliable electronic voting system that can streamline the voting process, ensure accuracy, and prevent duplicate votes. The system should allow voters to cast their votes easily and enable administrators to manage the election process efficiently, including viewing and tallying the results automatically.

This project aims to develop a Simple Voting System that digitizes the voting process, minimizes human error, and increases the efficiency and integrity of small-scale elections.

* **Objectives**:
* **To develop a basic electronic voting platform** that simplifies the process of casting and counting votes in small-scale elections.
* **To ensure that each voter can vote only once**, thereby maintaining vote integrity and preventing duplication.
* **To provide an easy-to-use interface** for both voters and administrators, allowing smooth navigation through the system.
* **To enable administrators to manage candidates** and view real-time or final voting results accurately and efficiently.

**2. Literature Survey / Existing System:**

**.** Electronic voting systems have been increasingly adopted to streamline the electoral process, reduce human error, and increase transparency.

**.** Various types of voting systems exist, ranging from highly secure government-grade electronic voting machines (EVMs) to basic web-based or console-based voting software used in academic and organizational settings.

**3. Software Requirement Analysis**

* **Functional Requirements**:
  + User Authentication.
  + Vote Casting.
  + Candidate Management.
  + Result Viewing.
* **Non-Functional Requirements**:
  + Usability.
  + Reliability.
  + Security.

**4. Software Design**

The design of the Simple Voting System follows a modular approach that divides the system into manageable components. This ensures ease of development, testing, and future enhancements. The design is primarily focused on a simple console application with basic input and output.

**1. System Architecture**

The Simple Voting System follows a Client-Server Architecture in a local environment. The client (voter) interacts with the system through a console-based user interface (UI), while the backend logic handles voting, result processing, and candidate management. The database or file system (local storage) stores data for the users, candidates, and votes.

* **User Interface (UI):** A console menu system where voters can choose actions such as voting, checking results, and logging out**.**
* **Backend Logic:** The core program logic that processes the user input, handles voting rules (e.g., one vote per user), and manages candidates and results.
* **Data Storage:** A simple file-based storage system where votes, candidate details, and user information are saved.

**2. System Components**

**The system is divided into the following components:**

1. **Voter Component:**
   * **Login:** The voter enters their credentials (username and password). If valid, the voter gains access to the voting menu.
   * **Vote Casting:** The voter selects a candidate from the available list and casts their vote. The system ensures that the voter can vote only once.
2. **Administrator Component:**
   * **Admin Login:** The admin logs into the system to perform administrative tasks like adding/deleting candidates and viewing results.
   * **Candidate Management:** The admin can add or remove candidates from the election list.
   * **Result Viewing**: The admin can view the current vote count for each candidate.
3. **Vote Management:**
   * The system records the votes, ensuring that each voter can vote only once. This is managed using a simple file or array-based storage mechanism**.**
4. **Data Storage:**
   * The system uses files to store voter information, election results, and candidate data. Each vote is recorded with a timestamp and associated with a particular voter to ensure no duplication.

**3. Modules and Data Flow**

**The system is divided into the following functional modules:**

1. **Authentication Module:**
   * Handles login and user validation (for both voters and admin).
   * **Data flow**: User provides credentials → System checks credentials against the stored data → Authentication is either granted or denied.
2. **Voting Module:**
   * Allows the voter to select and cast a vote for a candidate. Ensures only one vote per voter.
   * **Data flow:** Voter selects a candidate → System records vote → Updates the vote count for that candidate.
3. **Admin Module:**
   * Enables the administrator to manage the candidate list and view the election results.
   * **Data flow:** Admin adds/removes candidates → System updates candidate list → Admin views current vote counts.
4. **Storage Module:**
   * Manages storing and retrieving voter, candidate, and vote data from files.
   * **Data flow:** Data is saved to files after each vote → Data is retrieved during authentication, voting, or result viewing.

**4. Data Flow Diagram (DFD)**

**Here's a simple Data Flow Diagram to visualize the interactions:**

1. **Level 0 DFD:**
   * Voter logs in → Choose to vote → Vote is recorded → Result is updated.
   * Admin logs in → Manages candidates → Views vote count.
2. **Level 1 DFD:**
   * Voter provides credentials → Authentication checked → Voter selects a candidate → Vote saved in file.
   * Admin adds candidates → Candidate list updated → Admin views results**.**

**5. User Interface Design**

* **Main Menu:**

.Display options :

. 1. Log In (Voter)

* + - 1. Log In (Admin)
      2. Exit
* **Voter Menu:**
  + After login, display:
    - 1. Vote
      2. Log Out
* **Admin Menu:**
  + After login, display**:**
    - 1. **Add Candidate**
      2. Remove Candidate
      3. View Results
      4. Log Out

**6. Flowchart of Voting Process**

1. **Login Check:**
   * User enters credentials → Check validity → If valid, proceed to menu → Else, show error.
2. **Voting Flow:**
   * Voter selects candidate → Vote is recorded → Display confirmation message.
3. **Admin Flow:**
   * Admin logs in → Manage candidates → View results → Log out.

**Conclusion**

The design of the Simple Voting System ensures that all basic functionalities, such as voting, managing candidates, and viewing results, are efficiently handled. The system maintains modularity for easy maintenance and scalability.

**5. Coding**

**Example Code Outline:**

struct Candidate {

char name[50];

int votes;

};

void castVote();

void showResults();

void adminLogin();

**Function Explanation:**

* **castVote() –** Displays candidate list and records a vote.
* **showResults() –** Displays vote count per candidate**.**
* **adminLogin() –** Allows admin to control voting process.

**6. Testing**

**Black Box Testing**

* **Test Case 1:** Voting as a user → Expected: Vote is recorded.
* **Test Case 2:** Admin entering correct password → Expected: Access granted.

**White Box Testing**

* Function Loop and condition testing in vote input validation.

**7. Output Screens / Results**

**(Include console screenshots or mockups)**

* Login screen
* Voting screen
* Results screen

**8. Conclusion and Further Work**

* **Conclusion:** A simple and effective voting system for small-scale use was developed**.**
* **Further Work:** Could be extended with GUI, database integration, biometric authentication, or web access.

**9. References**

* C Programming by Dennis Ritchie
* Online tutorials (e.g., GeeksforGeeks, Tutorialspoint)
* [Your instructor’s notes or any additional materials used]

Top of Form

Bottom of Form

Code :

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Maximum number of candidates

#define MAX\_CANDIDATES 10

// Structure for voter details

typedef struct Voter {

int voterId;

char name[50];

struct Voter\* next;

} Voter;

// Structure for candidate

typedef struct {

int candidateId;

char name[50];

int votes;

Voter\* voters; // Linked list of voters who voted for this candidate

} Candidate;

// Global array of candidates

Candidate candidates[MAX\_CANDIDATES];

int candidateCount = 0;

// Function to add a new candidate

void addCandidate(int id, const char\* name) {

if (candidateCount >= MAX\_CANDIDATES) {

printf("Maximum number of candidates reached!\n");

return;

}

candidates[candidateCount].candidateId = id;

strcpy(candidates[candidateCount].name, name);

candidates[candidateCount].votes = 0;

candidates[candidateCount].voters = NULL;

candidateCount++;

printf("Candidate added successfully!\n");

}

// Function to add a voter to a candidate's list

void addVoter(int candidateId, int voterId, const char\* voterName) {

// Find the candidate

int i;

for (i = 0; i < candidateCount; i++) {

if (candidates[i].candidateId == candidateId) {

// Create new voter

Voter\* newVoter = (Voter\*)malloc(sizeof(Voter));

newVoter->voterId = voterId;

strcpy(newVoter->name, voterName);

newVoter->next = NULL;

// Add to the beginning of the linked list

newVoter->next = candidates[i].voters;

candidates[i].voters = newVoter;

candidates[i].votes++;

printf("Vote recorded successfully!\n");

return;

}

}

printf("Candidate not found!\n");

}

// Function to display all candidates and their votes

void displayCandidates() {

printf("\nCandidates List:\n");

printf("ID\tName\t\tVotes\n");

printf("--------------------------------\n");

for (int i = 0; i < candidateCount; i++) {

printf("%d\t%s\t\t%d\n",

candidates[i].candidateId,

candidates[i].name,

candidates[i].votes);

}

}

// Function to display voters for a specific candidate

void displayVoters(int candidateId) {

int i;

for (i = 0; i < candidateCount; i++) {

if (candidates[i].candidateId == candidateId) {

printf("\nVoters for %s:\n", candidates[i].name);

printf("Voter ID\tName\n");

printf("------------------------\n");

Voter\* current = candidates[i].voters;

while (current != NULL) {

printf("%d\t\t%s\n", current->voterId, current->name);

current = current->next;

}

return;

}

}

printf("Candidate not found!\n");

}

int main() {

int choice;

printf("Welcome to Election System!\n");

while (1) {

printf("\nMenu:\n");

printf("1. Add Candidate\n");

printf("2. Add Voter\n");

printf("3. Display Candidates\n");

printf("4. Display Voters for a Candidate\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: {

int id;

char name[50];

printf("Enter candidate ID: ");

scanf("%d", &id);

printf("Enter candidate name: ");

scanf("%s", name);

addCandidate(id, name);

break;

}

case 2: {

int candidateId, voterId;

char voterName[50];

printf("Enter candidate ID: ");

scanf("%d", &candidateId);

printf("Enter voter ID: ");

scanf("%d", &voterId);

printf("Enter voter name: ");

scanf("%s", voterName);

addVoter(candidateId, voterId, voterName);

break;

}

case 3:

displayCandidates();

break;

case 4: {

int candidateId;

printf("Enter candidate ID: ");

scanf("%d", &candidateId);

displayVoters(candidateId);

break;

}

case 5:

printf("Thank you for using the Election System!\n");

return 0;

default:

printf("Invalid choice! Please try again.\n");

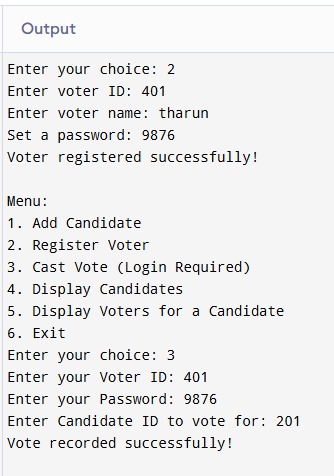
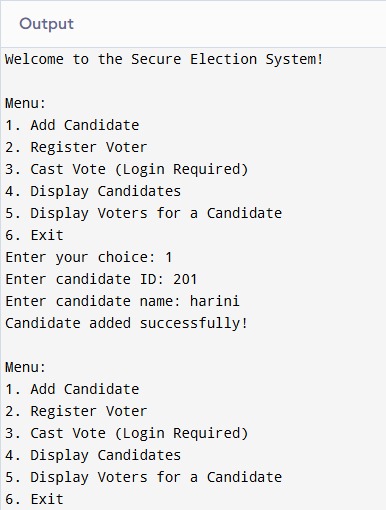
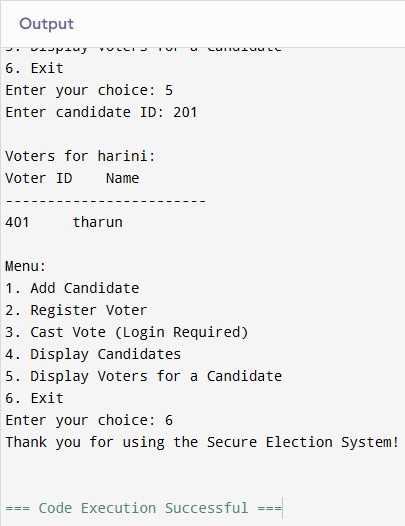
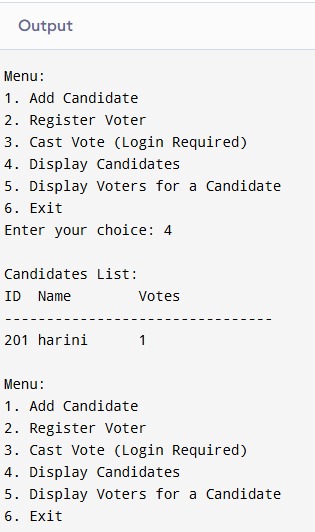
}

}

return 0;

}

Out put :



**Top of Form**

**Bottom of Form**

**Top of Form**

**Bottom of Form**