****Title: Digital Voting System Project****

## *Introduction :*

The Digital Voting System is a secure and efficient solution designed to manage voting processes electronically. This project aims to ensure transparency, accuracy, and user-friendliness by utilizing robust programming concepts and structured coding in C.

## *Objectives :*

* Develop a simple yet secure voting system using C programming language.
* Ensure accurate vote counting and result display.
* Provide a user-friendly interface for voters.

## *Features :*

* **User-Friendly Interface:** Easy navigation for voting.
* **Vote Management:** Users can cast votes securely.
* **Result Display:** Displays real-time voting results.
* **Error Handling:** Ensures proper input validation.

## *Tools and Technologies :*

* **Programming Language:** C
* **Compiler:** GCC (Code::Blocks, Dev-C++, etc.)
* **Operating System:** Windows / Linux

## *Code Implementation :*

### **> Sample Code for Digital Voting System in C**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**// Structure for Voter using Linked List**

**typedef struct Voter {**

**int id;**

**char name[50];**

**int hasVoted;**

**struct Voter\* next;**

**} Voter;**

**// Structure for Candidate using Binary Tree**

**typedef struct Candidate {**

**int id;**

**char name[50];**

**int votes;**

**struct Candidate\* left, \*right;**

**} Candidate;**

**// Queue Node for Voting Process**

**typedef struct QueueNode {**

**int voterId;**

**struct QueueNode\* next;**

**} QueueNode;**

**typedef struct Queue {**

**QueueNode \*front, \*rear;**

**} Queue;**

**Voter\* voterList = NULL;**

**Candidate\* candidateRoot = NULL;**

**Queue votingQueue = {NULL, NULL};**

**// Function to register voters**

**void registerVoter(int id, char name[]) {**

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

**newVoter->id = id;**

**strcpy(newVoter->name, name);**

**newVoter->hasVoted = 0;**

**newVoter->next = voterList;**

**voterList = newVoter;**

**}**

**// Function to add candidates using Binary Search Tree**

**Candidate\* addCandidate(Candidate\* root, int id, char name[]) {**

**if (root == NULL) {**

**Candidate\* newCandidate = (Candidate\*)malloc(sizeof(Candidate));**

**newCandidate->id = id;**

**strcpy(newCandidate->name, name);**

**newCandidate->votes = 0;**

**newCandidate->left = newCandidate->right = NULL;**

**return newCandidate;**

**}**

**if (id < root->id)**

**root->left = addCandidate(root->left, id, name);**

**else**

**root->right = addCandidate(root->right, id, name);**

**return root;**

**}**

**// Function to enqueue a voter**

**void enqueue(int voterId) {**

**QueueNode\* newNode = (QueueNode\*)malloc(sizeof(QueueNode));**

**newNode->voterId = voterId;**

**newNode->next = NULL;**

**if (votingQueue.rear == NULL) {**

**votingQueue.front = votingQueue.rear = newNode;**

**return;**

**}**

**votingQueue.rear->next = newNode;**

**votingQueue.rear = newNode;**

**}**

**// Function to process voting**

**void castVote(int voterId, int candidateId) {**

**Voter\* v = voterList;**

**while (v && v->id != voterId) v = v->next;**

**if (!v || v->hasVoted) {**

**printf("Invalid voter or already voted!\n");**

**return;**

**}**

**Candidate\* c = candidateRoot;**

**while (c && c->id != candidateId)**

**c = (candidateId < c->id) ? c->left : c->right;**

**if (!c) {**

**printf("Invalid candidate!\n");**

**return;**

**}**

**c->votes++;**

**v->hasVoted = 1;**

**printf("Vote cast successfully!\n");**

**}**

**// Function to display election results**

**void displayResults(Candidate\* root) {**

**if (root == NULL) return;**

**displayResults(root->left);**

**printf("%s: %d votes\n", root->name, root->votes);**

**displayResults(root->right);**

**}**

**int main() {**

**// Sample Data**

**voterList = NULL;**

**candidateRoot = addCandidate(candidateRoot, 1, "Alice");**

**candidateRoot = addCandidate(candidateRoot, 2, "Bob");**

**registerVoter(101, "John");**

**registerVoter(102, "Emma");**

**enqueue(101);**

**enqueue(102);**

**castVote(101, 1);**

**castVote(102, 2);**

**printf("Election Results:\n");**

**displayResults(candidateRoot);**

**return 0;**

**}**

## *Explanation :*

* **The system uses a **structure** to define candidates and their vote counts.**
* **The castVote() function updates the candidate's vote count.**
* **The displayResults() function shows the current voting status.**
* **The main() function controls the flow of the program through a menu-driven interface.**

## *Future Enhancements :*

* **Implement user authentication for secure access.**
* **Add data encryption to protect voter information.**
* **Enable cloud storage to handle larger voter databases.**

## *Output :*

## *Conclusion :*

**The Digital Voting System project showcases how efficient and secure voting can be achieved using C programming. This project can be expanded with additional features to improve security and functionality.**

## Thank You!