**SUMMARY REPORT**

**--TEAM INSANE**

INTRODUCTION:

The rapid evolution of technology has spurred the need for modernized election processes without any bugs. Traditional voting systems are prone to inefficiencies and inaccuracies. This project aims to address these challenges by implementing a secure and efficient online voting system using the C programming language. It provides multiple powers for the Admin, like banning some users from casting their vote, deleting some illegal votes etc.

IMPLEMENTATION:

Functions used in this project are:

1. adminPanel – void type
2. studentPanel – void type
3. authenticateAdmin – int type
4. initiateNewElection – void type
5. createCandidateFiles – void type
6. saveElectionInfoInFile – void type
7. loadElectionInfoFromFile – void type
8. deleteIllegalVote – void type
9. banID – void type
10. profileChange – void type
11. extractYear – int type
12. checkBranchCode – int type
13. extractRollNo – int type
14. isValid – int type
15. isVoted – int type
16. isBanned – int type
17. saveVote – void type
18. getWinner – int type

Global variables used in this project are:

1. numberOfCandidates – integer type
2. studentVotes – char type
3. orgpass – char type
4. orguser – char type

Structures used in this project are:

1. currentValidID – stores year (integer type), branch (char type), totalVoters (integer type)
2. candidate – stores cid (integer type), cname (char type), votes (integer type)

The most important part of the program is the main .c file, as this calls all the functions. In the header file the most important functions are the adminPanel and studentPanel.

The external libraries used are:

* <stdio.h> -- for taking inputs and giving outputs.
* <string.h> -- for using string handling functions.
* <conio.h> -- for using getch().

ADMINISTRATOR FUNCTIONALITIES

* New Elections

Inputting election details like year, branch, maximum roll number, number of candidates and their names

* Delete Illegal Votes

Admin can rectify irregularities and delete some of the illegal votes that were already casted.

* Banning Some Users

Admin has the authority to ban some users and restrict them from casting their vote.

* Result

Admin can put out the results at any time they want.

* Update Profile

Admin can update their username and password as per their convenience.

STUDENT FUNCTIONALITIES

* Casting Vote

Students can only cast their vote when they are not banned, and there precious vote will be counted only if the vote is not deleted by the admin.

DATA HANDLING

* Saving Election Information in “ElectionInfo.txt” file. This file then stores data, which include year, branch, total voters and candidate details.
* Saving Candidate details in “candidate1.txt” file. This file stores data of the candidate including the number of votes gained.
* Saving details of Banned users in “Banned.txt” file. This file stores the roll numbers of the students banned.

DATA VALIDATION

* This project does not allow everyone to enter the admin panel. The user have to get the authenticated username and password for entering the admin panel.
* Similarly, student authentication is also required. The userID for student contains year, branch and roll number as per the election.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

struct currentValidID{

int year;

char branch[6];

int totalVoters;

};

typedef struct candidate{

int cid;

char cname[20];

int votes;

}CANDIDATE;

//GLOBALS --------------------------------------------------------

struct currentValidID; //stores current Valid user ID parameters

CANDIDATE candidateArray[20]; //to store information all candidates

int numberOfCandidates; //Total number of candidates standing for election

char studentVotes[200]; //to store information of votes given by each student

char orgpass[20] = "STGY@cseb";

char orguser[20] = "Admin";

//----------------------------------------------------------------

//To extract year from userID -- For example, userID:2018btecs00064 year:2018

int extractYear(char userID[15])

{

int year=0;

char tmp;

for(int i=0;i<4;i++){

tmp=userID[i];

year=(year\*10)+(tmp-48);

}

return year;

}

int extractRollNo(char userID[15])

{

int rollno=0;

char tmp;

for(int i=9;i<14;i++){

tmp=userID[i];

rollno=(rollno\*10)+(tmp-48);

}

return rollno;

}

//Will check whether the global branch code and inputed branch code is matching or not

int checkBranchCode(char userID[15])

{

char branchCode[6];

for(int i=4;i<9;i++){

branchCode[i-4]=userID[i];

}

branchCode[5]='\0';

if(strcmp(branchCode,currentValidID.branch)==0)

return 1;

else

return 0;

}

int authenticateAdmin(){

char username[15], password[6];

printf("\nEnter username: ");

scanf("%s",username);

if((strcmp(username,orguser))!=0){

printf("\nWrong Username.....\n");

return 0;

}

else

{

printf("Enter Password: ");

int i=0;

int n=strlen(orgpass);

for(i=0;i<n;i++)

{

password[i]=getch();

printf("%c",'\*');

}

password[i]='\0';

if((strcmp(password,orgpass))!=0){

printf("\nWrong Password....\n");

return 0;

}

}

return 1;

}

void profilechange(){

if(authenticateAdmin()!=1){

printf("\nWrong Username or Password \n");

return;

}

char user\_new[20], pass\_new[20];

printf("\n\nLOGGED IN SUCCESSFULLY (Press Enter)");

getch();

int p, i;

printf("\nEnter New Username: ");

scanf("%s",user\_new);

strcpy(orguser, user\_new);

printf("\nEnter the no. of chars involved in the new password: ");

scanf("%d", &p);

printf("\nEnter New Password: ");

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

pass\_new[i]=getch();

printf("%c", '\*');

}

pass\_new[i]='\0';

strcpy(orgpass, pass\_new);

printf("\nProfile Updated Successfully");

}

void banID(){

printf("\nPreparing for Ban\n");

FILE \*fp=fopen("Banned.txt", "w");

if(fp==NULL){

printf("Error: Banned.txt not created.\n");

fclose(fp);

return;

}

printf("Just Enter last roll no to ban\nPress 0 to exit... ");

int input;

while(1){

printf("\nEnter Number: ");

scanf("%d",&input);

if(input==0)

break;

studentVotes[input-1]='$';

fprintf(fp,"%d\n",input);

}

fclose(fp);

printf("Banned Successfully\n");

}

void createCandidateFiles(){

printf("\nPreparing candidates...\n");

FILE \*fp;

char filename[20];

for(int i = 1;i <= numberOfCandidates; i++){

sprintf(filename,"candidate%d.txt",i);

fp=fopen(filename,"w");

fprintf(

fp,"0\n%s",

candidateArray[i-1].cname

);

fclose(fp);

}

printf("Prepared Candidates successfully\n");

}

void deleteIllegalVote(char userID[15])

{

FILE \*fp,\*fcp;

char filename[20];

char line[20];

int location = extractRollNo(userID);

sprintf(filename,"candidate%d.txt",candidateArray[studentVotes[location-1]-49].cid);

candidateArray[studentVotes[location-1]-49].votes--;

studentVotes[location-1]='0';

if ((fp = fopen(filename,"r")) == NULL)

{

printf("\nFile cannot be opened...Operation Failed");

return;

}

printf("\nDeleting in process...\n ");

if ((fcp = fopen("tmp.txt","w")) == NULL)

{

printf("\nFile cannot be opened...Operation Failed");

return;

}

while (!feof(fp))

{

fscanf(fp,"%s",line);

fprintf(fcp,"%s\n",line);

}

fclose(fp);

fclose(fcp);

if ((fp = fopen(filename,"w")) == NULL)

{

printf("\nFile cannot be opened...Operation Failed");

return;

}

int numFromFile;

char cnameFromFile[20];

fcp = fopen("tmp.txt","r");

fscanf(fcp,"%d",&numFromFile);

fprintf(fp,"%d",numFromFile-1);

fscanf(fcp,"%s",cnameFromFile);

fprintf(fp,"\n%s",cnameFromFile);

while(!feof(fcp)){

fscanf(fcp,"%d",&numFromFile);

if(numFromFile!=location)

fprintf(fp,"\n%d",numFromFile);

}

fclose(fp);

fclose(fcp);

remove("tmp.txt");

printf("\nVote deleted successfully\nPress any key to continue...");

getch();

}

int getWinner(){

int maxV = -1;

int winnerCid;

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

if(candidateArray[i].votes > maxV) {

winnerCid = candidateArray[i].cid;

maxV = candidateArray[i].votes;

}

else if(candidateArray[i].votes == maxV) {

return -1;

}

}

return winnerCid;

}

void initiateNewElection()

{

printf("\nNew Election Initiation:\n");

printf("\nElections for which Year: ");

scanf("%d",&currentValidID.year);

printf("Enter branch code: ");

scanf("%s",currentValidID.branch);

printf("Enter max roll no.: ");

scanf("%d",&currentValidID.totalVoters);

printf("Enter the no. of candidates: ");

scanf("%d",&numberOfCandidates);

for (int i = 0; i < currentValidID.totalVoters; i++)

{

studentVotes[i] = '0';

}

for (int i = 0;i < numberOfCandidates; i++)

{

candidateArray[i].cid=i+1;

printf("Enter name of candidate %d: ",i+1);

scanf(" %s",candidateArray[i].cname);

candidateArray[i].votes=0;

}

return;

}

void saveElectionInfoInFile(){

printf("Saving Election Info...\n");

FILE \*fp = fopen("ElectionInfo.txt", "w");

if(fp==NULL)

{

printf("\nError in file creation\n");

fclose(fp);

return;

}

fprintf(

fp,"%d\n%s\n%d\n%d",

currentValidID.year,

currentValidID.branch,

currentValidID.totalVoters,

numberOfCandidates

);

fclose(fp);

printf("Saved Successfully : )");

}

void loadElectionInfoFromFile()

{

FILE \*f1,\*f2,\*f3;

f1=fopen("ElectionInfo.txt","r");

if(f1==NULL)

printf("Not Exist");

fscanf(f1,"%d",&currentValidID.year);

fseek(f1,2,SEEK\_CUR);

fscanf(f1,"%s",currentValidID.branch);

fseek(f1,2,SEEK\_CUR);

fscanf(f1,"%d",&currentValidID.totalVoters);

fseek(f1,2,SEEK\_CUR);

fscanf(f1,"%d",&numberOfCandidates);

fclose(f1);

//load candidates info and student votes

for (int i = 0; i < currentValidID.totalVoters; i++)

{

studentVotes[i] = '0';

}

for(int i=1;i<=numberOfCandidates;i++)

{

int location;

char filename[20];

sprintf(filename,"candidate%d.txt",i);

f2=fopen(filename,"r+");

candidateArray[i-1].cid=i;

fscanf(f2,"%d",&candidateArray[i-1].votes);

fscanf(f2,"%s",candidateArray[i-1].cname);

while(!feof(f2)){

fscanf(f2,"%d",&location);

studentVotes[location-1] = i+48;

}

fclose(f2);

}

//load banned votes

int location;

f3=fopen("banned.txt","r+");

while(!feof(f3)){

fscanf(f3,"%d",&location);

studentVotes[location-1] = '$';

}

fclose(f3);

}

void adminPanel()

{

while(1){

if(authenticateAdmin()!=1){

break;

}

printf("\n\nLOGGED IN SUCCESSFULLY (Press Enter)");

getch();

while(1)

{

char inputID[15];

int input;char banInp;

int WinnerCid, totalVotedNow=0;

printf("\n1.New Election\n2.Continue Previous Election\n3.Delete Illegal Vote\n4.Ban User IDs\n5.Result\n6.Update Profile\n7.Logout\nOption: ");

scanf("%d",&input);

switch(input)

{

case 1:

initiateNewElection();

saveElectionInfoInFile();

createCandidateFiles();

break;

case 2:

loadElectionInfoFromFile();

break;

case 3:

printf("\nEnter user ID to delete its vote: ");

scanf("%s",inputID);

deleteIllegalVote(inputID);

break;

case 4:

printf("Do you want to ban particular ID's?\nPress 1 if yes or any other key to continue...");

scanf(" %c",&banInp);

if(banInp=='1'){

banID();

}

break;

case 5:

WinnerCid = getWinner();

if(WinnerCid != -1){

printf("\nWinner is %s with %d votes\n",candidateArray[WinnerCid-1].cname,candidateArray[WinnerCid-1].votes);

}

else{

printf("\nIts A TIE");

}

printf("\nFull Result\n");

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

totalVotedNow+=candidateArray[i].votes;

printf("%d. %s -> %d votes\n",candidateArray[i].cid,candidateArray[i].cname,candidateArray[i].votes);

}

printf("\nVoting Percentage: %.3f %%\n\n",(float)(totalVotedNow\*100)/currentValidID.totalVoters);

break;

case 6:

profilechange();

case 7:

return;

default:

printf("Invalid Option");

getch();

}

}

}

};

int isValid(char userID[15])

{

if(strlen(userID)!=14)

return 0;

int inputedYear=extractYear(userID);

int inputedRollNo = extractRollNo(userID);

if(inputedYear!=currentValidID.year || checkBranchCode(userID)!=1 || inputedRollNo>currentValidID.totalVoters)

return 0;

return 1;

}

int isVoted(char userID[15])

{

int location=extractRollNo(userID);

if(studentVotes[location-1]=='0')

return 0;

else

return 1;

}

int isBanned(char userID[15]){

int location=extractRollNo(userID);

if(studentVotes[location-1]=='$')

return 1;

else

return 0;

}

void saveVote(char userID[15],char voteInput)

{

char filename[20];

sprintf(filename,"candidate%d.txt",voteInput-48);

FILE \*fp = fopen(filename,"r+");

int location=extractRollNo(userID);

studentVotes[location-1]=voteInput;

candidateArray[voteInput-49].votes++;

fseek(fp, 0, SEEK\_SET);

fprintf(fp,"%d\n",candidateArray[voteInput-49].votes);

fseek(fp, 0, SEEK\_END);

fprintf(fp,"\n%d",location);

fclose(fp);

}

void studentPanel()

{

char userID[15];

char voteInput;

while(1)

{

printf("\n\n To exit press 0");

printf("\n Enter user ID:");

scanf("%s",userID);

if(strcmp(userID, "0")==0)

return;

if(isValid(userID)!=1)

{

printf("\n Invalid User ID(Press Enter)");

getch();

continue;

}

if(isBanned(userID)!=0)

{

printf("\nThis User ID is currently banned...\nContact Admin for the reason...(Press Enter to continue)");

getch();

continue;

}

if(isVoted(userID)!=0)

{

printf("\nYour PRN entered is already voted\n Contact Admin for furthur query");

getch();

continue;

}

printf("\n\nCandidates for election:");

for (int i = 0; i < numberOfCandidates; i++)

{

printf("\n %d. %s",i+1,candidateArray[i].cname);

}

printf("\n\nYour Vote(Enter Number):");

voteInput=getch();

printf("\*");

if(voteInput-48 < 1 || voteInput-48 > numberOfCandidates)

{

printf("\nInvalid Vote\nTry Again...");

getch();

continue;

}

saveVote(userID,voteInput);

printf("\n\nThanks for your precious vote(Press Enter)");

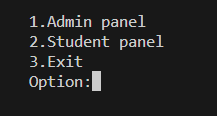
getch();

}

};

TEST CASES:

Test Case 1

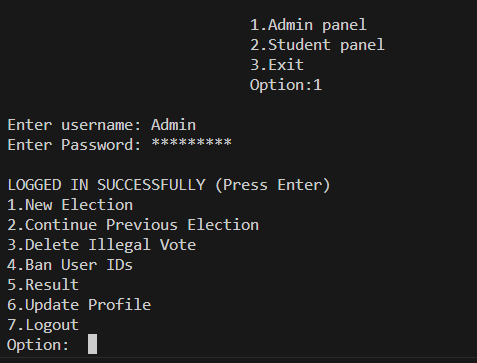


Input = 1

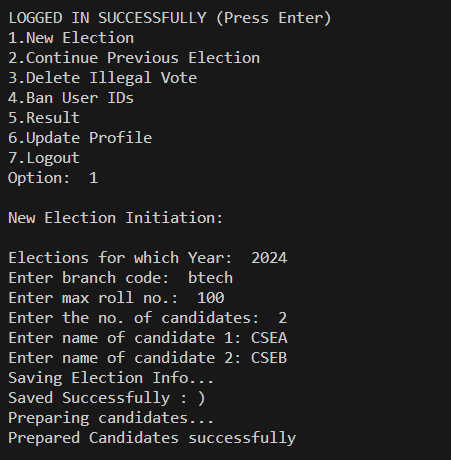
Username: Admin

Password: STGY@cseb

Output =



Input = 1



Year = 2024

Branch Code = btech

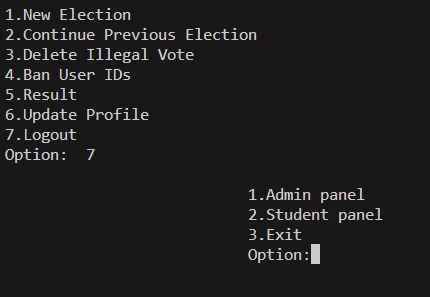
Max roll no. = 100

No. of candidates = 2

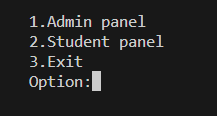
Candidate 1 = CSEA

Candidate 2 = CSEB

Input = 7

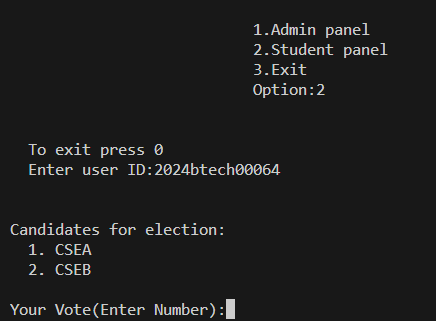


Test Case 2



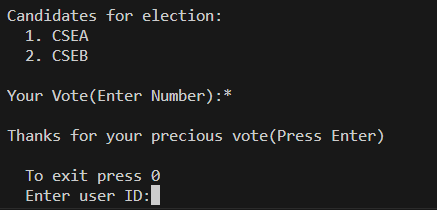
Input = 2

UserID = 2024btech00064 (year + branch code + roll no.)



Input = 1

Your vote will be casted to CSEA….



CONCLUSION:

This program successfully achieves its objective of a fair election. Future enhancements can be done in the password section.