#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

// ANSI color codes

#define RESET       "\033[0m"

#define RED         "\033[1;31m"

#define GREEN       "\033[1;32m"

#define YELLOW      "\033[1;33m"

#define CYAN        "\033[1;36m"

// Constants

#define MAX\_CANDIDATES 5

#define MAX\_VOTERS 1000

#define PASSWORD "official@1234"

// Candidate structure

typedef struct {

    char username[50];

    char name[50];

    int candidate\_no;

    char party;

    int votes;

    int registered;

} Candidate;

// Global arrays for each party

Candidate partyA[MAX\_CANDIDATES] = {0};

Candidate partyB[MAX\_CANDIDATES] = {0};

Candidate partyC[MAX\_CANDIDATES] = {0};

Candidate partyD[MAX\_CANDIDATES] = {0};

Candidate partyE[MAX\_CANDIDATES] = {0};

// Array to track voters' NICs

char registeredNICs[MAX\_VOTERS][20];

int totalNICs = 0;

// Function declarations

void vote();

void registerCandidate();

void viewOfficials();

int isValidNIC(char \*nic);

int main() {

    int choice;

    while (1) {

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

        printf(YELLOW "      🎉 Welcome to the Election Voting System 🎉   \n" RESET);

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

        printf(GREEN "1️⃣  Vote\n" RESET);

        printf(GREEN "2️⃣  Candidate Register\n" RESET);

        printf(GREEN "3️⃣  Officials View\n" RESET);

        printf(RED   "4️⃣  Exit\n" RESET);

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

        printf("👉 Enter your choice (1-4): ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                vote();

                break;

            case 2:

                registerCandidate();

                break;

            case 3:

                viewOfficials();

                break;

            case 4:

                printf(YELLOW "👋 Exiting... Thank you for participating!\n" RESET);

                return 0;

            default:

                printf(RED "❌ Invalid choice. Please try again.\n" RESET);

        }

    }

    return 0;

}

// Officials View

void viewOfficials() {

    char password[50];

    printf(CYAN "\n------ 🛡️ OFFICIAL VIEW 🛡️ ------\n" RESET);

    printf("🔐 Enter your password: ");

    scanf("%s", password);

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

        printf(RED "❌ Incorrect password. Access denied.\n" RESET);

        return;

    }

    printf(YELLOW "\n------ 🗳️ MATARA DISTRICT VOTE COUNTS 🗳️ ------\n\n" RESET);

    Candidate \*parties[] = {partyA, partyB, partyC, partyD, partyE};

    char partyLabels[] = {'A', 'B', 'C', 'D', 'E'};

    for (int p = 0; p < 5; p++) {

        printf(CYAN "  Party %c   Candidate\_No  Votes\n\n" RESET, partyLabels[p]);

        for (int i = 0; i < MAX\_CANDIDATES; i++) {

            if (parties[p][i].registered) {

                printf(GREEN "%d️⃣  %s \t%02d         %02d\n" RESET, i + 1,

                    parties[p][i].name,

                    parties[p][i].candidate\_no,

                    parties[p][i].votes);

            } else {

                printf("%d️⃣  [Empty] \t00         00\n", i + 1);

            }

        }

        printf("\n");

    }

}

// Voting

void vote() {

    int age;

    char nic[20];

    int candidate\_no;

    int found = 0;

    printf(CYAN "\n------ 🗳️ VOTING SECTION 🗳️ ------\n\n" RESET);

    printf("🎂 Enter your age (only eligible if 18 or above): ");

    scanf("%d", &age);

    if (age < 18) {

        printf(RED "❌ You are not eligible to vote.\n" RESET);

        return;

    }

    printf("🆔 Enter your 12-digit NIC no: ");

    scanf("%s", nic);

    if (!isValidNIC(nic)) {

        printf(RED "❌ Invalid NIC. NIC must be exactly 12 digits and numeric only.\n" RESET);

        return;

    }

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

        if (strcmp(registeredNICs[i], nic) == 0) {

            printf(RED "🚫 You have already voted. Duplicate voting is not allowed.\n" RESET);

            return;

        }

    }

    printf(YELLOW "\n------- 📄 BALLOT PAPER 📄 -------\n\n" RESET);

    Candidate \*parties[] = {partyA, partyB, partyC, partyD, partyE};

    char partyLabels[] = {'A', 'B', 'C', 'D', 'E'};

    for (int p = 0; p < 5; p++) {

        printf(CYAN "  Party %c   Candidate No\n" RESET, partyLabels[p]);

        for (int i = 0; i < MAX\_CANDIDATES; i++) {

            if (parties[p][i].registered) {

                printf("   %d️⃣  %s \t%02d\n", i + 1,

                    parties[p][i].name,

                    parties[p][i].candidate\_no);

            } else {

                printf("   %d️⃣  [Slot Empty] \t00\n", i + 1);

            }

        }

        printf("\n");

    }

    printf("🗳️ Choose your candidate using their candidate number (2-digit): ");

    scanf("%d", &candidate\_no);

    for (int p = 0; p < 5 && !found; p++) {

        for (int i = 0; i < MAX\_CANDIDATES; i++) {

            if (parties[p][i].registered && parties[p][i].candidate\_no == candidate\_no) {

                parties[p][i].votes++;

                found = 1;

                strcpy(registeredNICs[totalNICs++], nic);

                break;

            }

        }

    }

    if (found)

        printf(GREEN "✅ Your vote has been accepted. Thank you for voting!\n" RESET);

    else

        printf(RED "❌ Invalid candidate number. Vote not counted.\n" RESET);

}

// Candidate Registration

void registerCandidate() {

    Candidate newCandidate;

    printf(CYAN "\n------ 🧾 CANDIDATE REGISTRATION ------\n\n" RESET);

    printf("👤 Enter your username: ");

    scanf("%s", newCandidate.username);

    printf("🧍 Enter your name: ");

    scanf("%s", newCandidate.name);

    printf("🔢 Enter your candidate number: ");

    scanf("%d", &newCandidate.candidate\_no);

    // Check for duplicate candidate number

    Candidate \*allParties[] = {partyA, partyB, partyC, partyD, partyE};

    for (int p = 0; p < 5; p++) {

        for (int i = 0; i < MAX\_CANDIDATES; i++) {

            if (allParties[p][i].registered && allParties[p][i].candidate\_no == newCandidate.candidate\_no) {

                printf(RED "❌ Candidate number already taken. Registration failed.\n" RESET);

                return;

            }

        }

    }

    printf("🏳️ Choose your party (A, B, C, D, E): ");

    scanf(" %c", &newCandidate.party);

    newCandidate.votes = 0;

    newCandidate.registered = 1;

    Candidate \*partyList = NULL;

    switch (toupper(newCandidate.party)) {

        case 'A': partyList = partyA; break;

        case 'B': partyList = partyB; break;

        case 'C': partyList = partyC; break;

        case 'D': partyList = partyD; break;

        case 'E': partyList = partyE; break;

        default:

            printf(RED "❌ Invalid party selection.\n" RESET);

            return;

    }

    for (int i = 0; i < MAX\_CANDIDATES; i++) {

        if (!partyList[i].registered) {

            partyList[i] = newCandidate;

            printf(GREEN "\n✅ You are successfully registered as candidate number %02d for Party %c.\n" RESET,

                   newCandidate.candidate\_no, toupper(newCandidate.party));

            return;

        }

    }

    printf(RED "\n❌ Registration failed. Max candidates reached for Party %c.\n" RESET, toupper(newCandidate.party));

}

// Utility function to validate NIC

int isValidNIC(char \*nic) {

    if (strlen(nic) != 12) {

        return 0;

    }

    for (int i = 0; i < 12; i++) {

        if (!isdigit(nic[i])) {

            return 0;

        }

    }

    return 1;

}

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

# Generate SSH key (just press Enter when it asks for a file path)

ssh-keygen -t ed25519 -C "you@example.com"

Then add the SSH key to your GitHub account:

# Copy the key to clipboard

cat ~/.ssh/id\_ed25519.pub

Go to GitHub → Settings → SSH and GPG keys → New SSH key → Paste it

Then test the connection:

ssh -T git@github.com

🔐 3. Personal Access Token for HTTPS (If Not Using SSH)

If you’re using HTTPS, GitHub no longer supports password login. You must use a token:

# When you push, Git will ask for a username and password:

# Use your GitHub username

# Use a personal access token as the password (not your GitHub password)

Go to https://github.com/settings/tokens

⚙️ 4. Initialize a New Git Repo

git init

git add .

git commit -m "Initial commit"

🌐 5. Link to GitHub Repository

git remote add origin <https://github.com/yourusername/your-repo.git>

or

git remote add origin git@github.com:yourusername/your-repo.git

Then push:

git push -u origin main