# A Project Report on

**SIMPLE VOTING SYSTEM**

# Submitted in partial fulfillment of the requirements for the award of the Degree of

**Bachelor of Technology**

# In

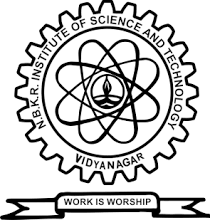
**COMPUTER SCIENCE ENGINEERING**

**By**

**M.SANDEEP- 24KB1A05BC**

**M.HARI HARA VARA PRASAD - 24KB1A05AX**

**M.DILLESHWARA RAO - 24KB1A05AK**

**Under the Esteemed Supervision of**

**SIVANRAJ SWAMY (byte xl trainer)**

**Department of computer science Engineering**

NBKR INSTITUTE OF SCIENCE AND TECHNOLOGY

**(An Autonomous Institution)**

(Approved by AICTE, Tirupati, JNTUA Accredited by NBA with ‘A’ Grade) Vidyanagar, Naidupeta road, Kota

**2024-2028**

**AKNOWLEDGEMENT**

It is with immense pleasure that we would like to express our indebted gratitude to our project guide **Mr.Sivanraj swamy (byte xl trainer),** who has guided us a lot and encouraged us in every step of the project work, her valuable moral support and guidance throughout the project helped us to a greater extent.

Our sincere thanks to my team and faculty who are supported us to complete this project and **NBKRIST**

**DECLARATION**

We hereby to declare that the project entitled “**SIMPLE VOTING SYSTEM”** is a genuine project. The work has been submitted to the **NBKR INSTITUTE OF SCIENCE AND TECHNOLOGY,** Vidyanagar, permanently affiliated to **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, ANANTHAPURAMU,** in a partial fulfillment of the **B.TECH** degree, We further declare that this project work has not been submitted in full or part for the award of any degree of this or any other educational institutions.

**By MURARISANDEEP(24KB1A05BC)**

**MUNAGALAHARIHARAVARAPRASAD(24KB1A05AX)**

**MIRYABBELLIDILLESHWARA RAO(24KB1A05AK)**

**INTRODUCTION OF SIMPLE VOTING SYSTEM :**

A simple voting system in the C programming language is a basic yet effective console-based application designed to simulate the voting process. It allows users to cast their votes for a predefined list of candidates and displays the final result based on the number of votes received by each candidate. This type of system is often developed for educational purposes to demonstrate key programming concepts such as arrays, loops, functions, and conditional statements. The system typically includes input validation to ensure only valid votes are counted, and a straightforward user interface to guide voters through the process. While it may not offer the complexity or security features of real-world electronic voting systems, a simple C-based voting program provides a clear and practical example of how logic and control structures can be used to solve real-life problems programmatically.

One of the main goals of implementing a voting system in C is to understand how data can be collected, processed, and displayed in a structured manner. The program begins by presenting a menu with a list of candidates to the user. Each voter is prompted to enter their choice, which is then recorded by incrementing a counter associated with the selected candidate. After all votes have been cast, the system tallies the results and announces the candidate with the highest number of votes as the winner. This simple mechanism demonstrates how programming logic can be used to replicate basic decision-making and tracking processes.

The application operates entirely in memory without the use of files or databases, making it ideal for educational purposes and understanding the fundamentals of data storage and manipulation in C.

# 🔑 Key Features:

* Use of structures for real-world modeling.
* Implementation of CRUD operations (Create, Read, Update, Delete).
* Interactive menu system for user input.
* Linear search and array shifting techniques for data handling.

**WORKING OF THE PROJECT :**

# ✅ Objective:

# To develop a console-based voting system application in C that demonstrates fundamental data structure concepts while simulating basic voting functionality.

# Specific Objectives:

# Voter Management:

# Implement data structures to store and manage voter information (arrays/structures/linked lists)

# Handle voter registration with unique IDs

# Candidate Management:

# Maintain candidate list using appropriate data structures

# Track vote counts per candidate

# Core Voting Operations:

# Authenticate voters before allowing votes

# Record votes while preventing duplicate voting

# Calculate and display election results

# Data Structure Demonstration:

# Utilize arrays/structures for candidate and voter records

# Implement linked lists or queues for managing voter flow

# Use sorting algorithms to display results

# Basic Features:

# Admin interface to add candidates

# Voter interface to cast votes

# Results display showing vote percentages

# 🔄 Functionalities and Their Working:

* **1. Voter Registration & Authentication**
* **Functionality:**
* Allows new voters to register before voting.
* Ensures only registered voters can cast votes.
* **Working:**
* **Voter Registration:**
* Admin/voter enters details (Name, Age, Voter ID).
* System checks if the **Voter ID is unique**.
* If valid, the voter is added to a **voter database (array/linked list/file)**.
* **Voter Authentication (Login):**
* Voter enters their **Voter ID**.
* System searches the database to verify existence.
* Checks if the voter **has already voted** (prevent duplicate voting).
* **2. Candidate Management (Admin Functionality)**
* **Functionality:**
* Admin can **add, view, edit, or delete** candidates.
* **Working:**
* **Adding a Candidate:**
* Admin enters **Candidate ID, Name, and Party**.
* System stores this in a **candidate array/linked list/file**.
* **Viewing Candidates:**
* Displays all candidates with their details.
* **Editing/Deleting Candidates:**
* Admin selects a candidate by **ID/Name**.
* Modifies details or removes them from the list.
* **3. Casting a Vote**
* **Functionality:**
* Authenticated voters can select a candidate and cast their vote.
* **Working:**
* Voter logs in (using Voter ID).
* System displays the **list of candidates**.
* Voter selects a candidate (by entering **Candidate ID**).
* System:
* **Increments the vote count** for the chosen candidate.
* **Marks the voter as "voted"** in the database.
* Prevents the voter from voting again.
* **4. Vote Counting & Results Calculation**
* **Functionality:**
* Calculates total votes per candidate.
* Determines the winner.
* **Working:**
* **Counting Votes:**
* System reads vote counts from the candidate database.
* Uses a **loop** to sum votes for each candidate.
* **Calculating Percentages:**
* Percentage = (Candidate’s Votes / Total Votes) \* 100
* **Declaring the Winner:**
* Compares all candidate vote counts.
* The candidate with the **highest votes** is declared the winner.
* **5. Displaying Election Results**
* **Functionality:**
* Shows voting statistics in a structured format.
* **Working:**
* **Admin selects "View Results".**
* System retrieves data from the candidate database.
* Displays:
* **Candidate Name | Votes Received | Vote Percentage**
* **Total Votes Cast**
* **Winner Announcement**
* **6. Data Storage & Retrieval (File Handling)**
* **Functionality:**
* Saves voter and candidate data even after program exit.
* **Working:**
* **Saving Data:**
* Before closing, the system writes:
* Voter details → voters.txt (or .dat)
* Candidate details → candidates.txt
* **Loading Data:**
* When the program restarts, it reads from files and loads into memory (arrays/lists).
* **7. Security & Input Validation**
* **Functionality:**
* Prevents invalid inputs and unauthorized access.
* **Working:**
* **Input Validation:**
* Checks if:
* Voter ID is numeric.
* Age is ≥18 (eligible to vote).
* No blank entries.
* **Duplicate Prevention:**
* Uses **flags** (e.g., hasVoted) to block multiple votes.
* **8. Admin & Voter Roles**

| * **Feature** | * **Admin Access** | * **Voter Access** |
| --- | --- | --- |
| * Add/Edit Candidates | * ✅ Yes | * ❌ No |
| * View Candidates | * ✅ Yes | * ✅ Yes |
| * Cast a Vote | * ❌ No | * ✅ Yes |
| * View Results | * ✅ Yes | * ❌ (Optional |

**AIM :**

The aim of the simple voting system in C is to create a basic, menu-driven application that simulates the process of voting in a democratic setup. The system is intended to enable users to vote for predefined candidates, record each vote accurately, and determine the winner based on the highest number of votes received. This project aims to provide a hands-on understanding of core C programming concepts such as control structures, arrays, functions, and user input/output, while also demonstrating how logical problem-solving can be applied to build a simple, real-world application.

📌 **Objectives:**

* **To understand and apply structures in C for managing real-world entities (like bank accounts).**
* **To implement menu-driven programming for interactive user operations.**
* **To perform basic data management using static arrays and procedural programming.**
* **To develop hands-on skills in fileless data handling, user input validation, and control structures.**

🔄 **Processes**

# Initialization Phase

# Load existing voter and candidate data from files (if any)

# Set up data structures for storage

# Admin Operations

# Add/remove candidates

# View registered voters

# Generate election results

# Voter Operations

# Register new voters

# Authenticate and login

# View candidate list

# Cast vote

# Result Processing

# Calculate vote counts

# Determine winner

# Display statistics

# Data Persistence

# Save all data to files before exit

# 2.2 Technical Process

# Data Structure Selection

# Arrays/Structures for fixed-size data

# Linked lists for dynamic data

# File handling for persistent storage

# Algorithm Implementation

# Searching algorithms for voter authentication

# Sorting algorithms for result display

# Input validation routines

# User Interface Design

# Menu-driven console interface

# Clear display formatting

# Error handling mechanisms

# Program :

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#define MAX\_CANDIDATES 3**

**#define NAME\_LEN 50**

**// Structure for a Voter (Node in linked list)**

**typedef struct Voter {**

**int voterID;**

**struct Voter\* next;**

**} Voter;**

**// Structure for Candidate**

**typedef struct Candidate {**

**char name[NAME\_LEN];**

**int voteCount;**

**Voter\* voterList; // Head of linked list**

**} Candidate;**

**// Global array of candidates**

**Candidate candidates[MAX\_CANDIDATES] = {**

**{"Alice", 0, NULL},**

**{"Bob", 0, NULL},**

**{"Charlie", 0, NULL}**

**};**

**// Function to create a new voter node**

**Voter\* createVoter(int id) {**

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

**newVoter->voterID = id;**

**newVoter->next = NULL;**

**return newVoter;**

**}**

**// Function to cast a vote**

**void castVote() {**

**int voterID, choice;**

**printf("Enter your Voter ID: ");**

**scanf("%d", &voterID);**

**printf("Choose a candidate to vote:\n");**

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

**printf("%d. %s\n", i + 1, candidates[i].name);**

**}**

**printf("Enter your choice: ");**

**scanf("%d", &choice);**

**if (choice < 1 || choice > MAX\_CANDIDATES) {**

**printf("Invalid choice. Vote not counted.\n");**

**return;**

**}**

**// Add voter to candidate's linked list**

**Candidate\* selected = &candidates[choice - 1];**

**Voter\* newVoter = createVoter(voterID);**

**newVoter->next = selected->voterList;**

**selected->voterList = newVoter;**

**selected->voteCount++;**

**printf("Vote cast successfully for %s!\n", selected->name);**

**}**

**// Function to display results**

**void displayResults() {**

**printf("\n--- Voting Results ---\n");**

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

**printf("Candidate: %s | Votes: %d\n", candidates[i].name, candidates[i].voteCount);**

**printf("Voters who voted for %s: ", candidates[i].name);**

**Voter\* curr = candidates[i].voterList;**

**while (curr != NULL) {**

**printf("%d ", curr->voterID);**

**curr = curr->next;**

**}**

**printf("\n");**

**}**

**}**

**int main() {**

**int option;**

**while (1) {**

**printf("\n--- Simple Voting System ---\n");**

**printf("1. Cast Vote\n");**

**printf("2. View Results\n");**

**printf("3. Exit\n");**

**printf("Choose an option: ");**

**scanf("%d", &option);**

**switch (option) {**

**case 1:**

**castVote();**

**break;**

**case 2:**

**displayResults();**

**break;**

**case 3:**

**printf("Exiting program.\n");**

**return 0;**

**default:**

**printf("Invalid option.\n");**

**}**

**}**

**return 0;**

**}**

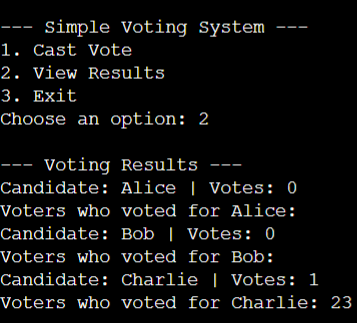
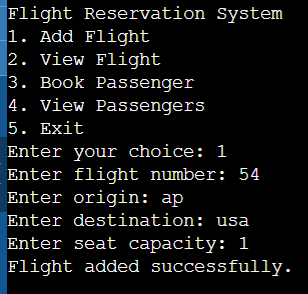
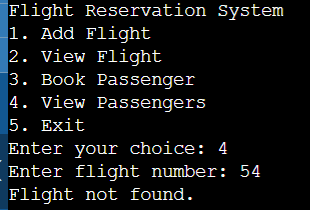
# Work flow :

📌

**Workflow of a Simple Voting System**

1. **System Initialization**
   * **Admin sets up the voting event.**
   * **Candidates and voter information are registered in the system.**
2. **Voter Authentication**
   * **Voter logs into the system using a unique identifier (e.g., voter ID, email, password).**
   * **System verifies the voter's credentials to prevent unauthorized access.**
3. **Ballot Display**
   * **Authenticated voter is presented with a list of candidates or options.**
   * **Instructions for casting a vote are clearly shown.**
4. **Vote Casting**
   * **Voter selects their preferred candidate or option.**
   * **Voter confirms their selection before submission.**
5. **Vote Recording**
   * **The system securely stores the vote in a database or file.**
   * **Duplicate voting is prevented by marking the voter as "voted".**
6. **Vote Counting**
   * **After the voting period ends, the system automatically tallies the votes.**
   * **Results are calculated based on the number of votes each candidate received.**
7. **Result Display**
   * **Final results are displayed to authorized users or the public.**
   * **Optionally, a report can be generated for record-keeping.**
8. **System Termination**
   * **Election is marked as complete.**
   * **Data is archived or exported for future reference.**

**OUTPUT IMAGES :**

****

**A screen shot of a computer

AI-generated content may be incorrect.**

**A screen shot of a computer

AI-generated content may be incorrect.**

**ADVANTAGES OF SIMPLE BANKING SYSTEM :**

 **Easy to Understand**: It's simple for voters to know how to vote and how votes are counted.

 **Quick to Count**: Results can be counted and announced quickly, often on the same day.

 **Low Cost**: It requires minimal infrastructure compared to more complex systems.

 **Strong Majority Governments**: Often leads to clear winners, which can create stable, decisive governments.

 **Discourages Extremism**: Tends to favor moderate, broadly appealing candidates over fringe or extreme options.

 **Direct Accountability**: Each representative is directly accountable to a specific geographical area or constituency.

 **Reduces Coalition Governments**: Because it favors larger parties, it often avoids the complications of multi-party coalitions.

# ✅ Real-Time Uses:

**1. National Elections**

* **United States**: Used for most elections, including Congressional and local elections (not for the presidential Electoral College system).
* **United Kingdom**: Used for electing Members of Parliament (MPs) in the House of Commons.
* **India**: Used for electing members to the Lok Sabha (lower house of Parliament).
* **Canada**: Used in federal and many provincial elections.
* **Nigeria**: Used in presidential and legislative elections.

**2. Local Government Elections**

* Many countries (like Kenya, Bangladesh, and the U.S.) use simple voting in mayoral, council, and local government elections.

**3. School and University Elections**

* Student councils and university boards often use a simple voting system for elections because it's easy to implement and understand.

**4. Organizational Voting**

* Clubs, societies, unions, and small businesses often use it for board member elections or policy decisions.

**5. Referendums and Ballot Questions**

* Some referendums (e.g., Brexit vote in the UK) use a simple majority to determine outcomes.

# Conclusion :

* The simple voting system is a straightforward and efficient method for making democratic decisions. Its ease of use, quick results, and low cost make it especially practical for large-scale elections and local decision-making. While it may not always reflect the full diversity of voter preferences, it provides clear outcomes and promotes strong, accountable leadership. Overall, it remains a widely used and effective voting method in many parts of the world.