Smart Voting System

Project Report

Submitted By

Name	Roll Number
Talha Khan	242210
Ahsan Khan	242206

Course: Data Structures & Algorithms

Submitted To: Prof Ammar Khan

Department of Creative Technology

Air University Islamabad

1. Introduction

Voting systems are critical information systems that require accuracy, security, and fast data processing. In large-scale elections, managing voter records, preventing duplicate voting, and calculating results efficiently are major challenges. Traditional manual or poorly structured systems can lead to errors, delays, and lack of transparency.

This project implements a Smart Voting System using core Data Structures and Algorithms (DSA) concepts. The system simulates a real-world election environment where voters are registered at specific polling stations, authenticated before voting, and allowed to cast only one vote. The system also provides live voting status, vote undo functionality, and final result calculation.

The primary objective of this project is to apply fundamental data structures such as Linked List, AVL Tree (Binary Search Tree), Stack, and Arrays to design an efficient, reliable, and structured voting system.

2. Problem Statement

An election system must manage a large number of voters while ensuring:

- Fast voter authentication
- Prevention of duplicate voting
- Accurate vote counting
- Real-time voting status
- Transparent and fair result calculation

Traditional or naive systems that rely on linear searching and unstructured data suffer from:

- Slow voter verification
- High chances of duplicate voting
- Inefficient vote management
- Poor scalability

Problem:

Design a voting system that efficiently manages voter data, ensures secure authentication, supports real-time monitoring, and calculates results accurately using appropriate data structures.

3. Proposed Solution

The proposed solution is a menu-driven Smart Voting System implemented in C++, integrating multiple data structures, each responsible for a specific task.

This multi–data structure approach ensures:

- Efficient data access
- Separation of responsibilities
- Realistic election workflow simulation
- Improved time complexity

4. Existing System

4.1 Manual Voting Systems

- Paper-based voter lists
- Manual vote counting
- Human-dependent verification

4.2 Limitations of Existing Systems

- High probability of human error
- Slow processing
- Lack of real-time monitoring
- Poor scalability
- Difficult result verification

5. System Overview

The system operates through a console-based interface. Users first select a polling station, after which all operations are restricted to that station.

Each major operation is handled by a specific data structure:

Operation	Data Structure Used
Voter storage	Linked List
Fast voter search	AVL Tree (BST)
Undo last vote	Stack
Vote counting	Arrays

Voter authentication is performed using AVL Tree traversal, votes are recorded in arrays, and undo functionality is implemented using a stack.

6. Functional Requirements

The system shall:

- Store voters for each polling station
- Authenticate voters using ID and name
- Prevent multiple voting
- Allow voters to cast votes for political parties
- Undo the last cast vote
- Display live voting status
- Calculate results and winning margins
- Restrict operations to a selected polling station

7. Non-Functional Requirements

7.1 Performance

- Voter search optimized using AVL Tree
- Vote recording in constant time

7.2 Reliability

- Ensures accurate vote counting
- Prevents duplicate voting

7.3 Scalability

- Dynamic memory allocation allows easy expansion
- Additional polling stations and voters can be added

7.4 Usability

- Simple menu-driven interface
- Clear prompts and error messages

7.5 Maintainability

- Modular code structure
- Easy to modify or extend

8. Detailed Data Structures Implementation

Purpose:

Stores voter records dynamically for each polling station.

Implementation Details:

Each node stores voter ID, name, polling station, voting status, and pointer to the next node. New voters are inserted at the head of the list.

Why Linked List?

- No fixed size limitation
- Efficient insertion
- Dynamic memory usage

8.2 AVL Tree (Binary Search Tree) – Fast Authentication

Purpose:

Provides fast searching of voters using voter ID.

Implementation Details:

- Voter ID is used as the key
- Each BST node stores a pointer to a Linked List voter node
- AVL rotations ensure the tree remains balanced

Advantages:

- Search complexity: O(log n)
- Efficient authentication
- Prevents skewed trees

8.3 Stack – Vote Undo Feature

Purpose:

Stores history of recently cast votes.

Implementation Details:

- Each vote is pushed onto the stack
- Undo operation pops the last vote
- Implements LIFO (Last In First Out) principle

Advantages:

- Easy undo functionality
- Accurate rollback of last vote

8.4 Arrays – Vote Counting

Purpose:

Stores vote counts for each party at each polling station.

Implementation Details:

- 2D array used for stations and parties
- Fast access and update

Advantages:

- Constant-time access
- Simple implementation

9. Algorithmic Flow

1) Initialize voter data
2) Select polling station
3) Display main menu
4) Authenticate voter
5) Cast or undo vote
6) Update vote counts
7) Display live status or results
8) Repeat until exit

10. Error Handling and Validation

- Prevents voting by unregistered users
- Blocks already voted voters
- Handles invalid menu inputs
- Prevents undo when no vote exists

11. Advantages of the System

- Fast and efficient voter authentication
- Real-world election simulation
- Prevents duplicate voting
- Demonstrates multiple DSA concepts

- Easy to extend and improve

12. Limitations

- No file handling (data not persistent)
- Console-based interface only
- Single-user system
- Limited security features

13. Conclusion

The Smart Voting System successfully demonstrates how core data structures can be combined to solve a real-world problem efficiently. By integrating Linked Lists, AVL Trees, Stacks, and Arrays, the system ensures fast authentication, secure voting, real-time monitoring, and accurate result calculation.

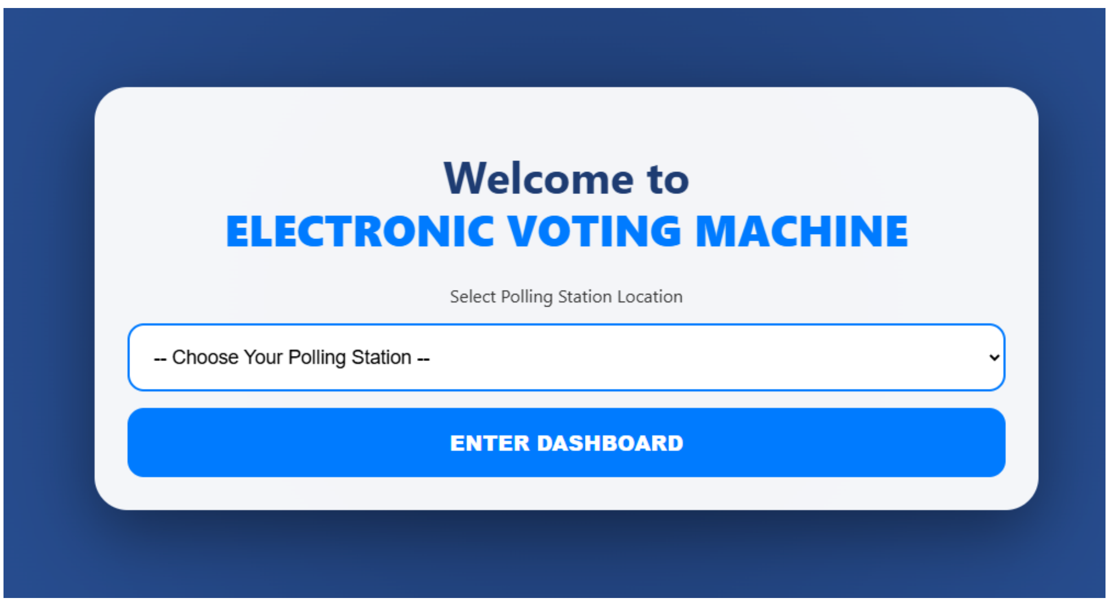
This project strengthens both theoretical understanding and practical implementation of Data Structures and Algorithms.

14. Future Work

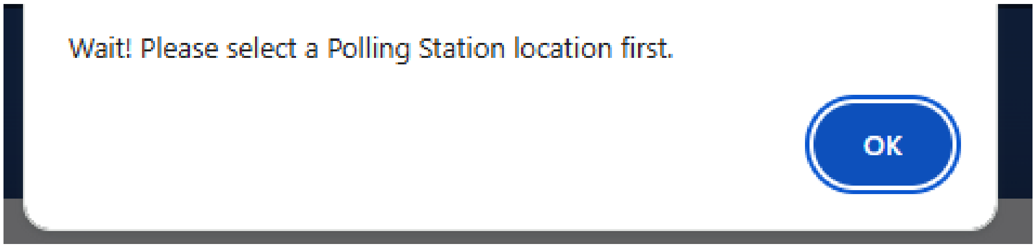
- File handling for permanent data storage
- GUI-based interface
- Online voting simulation
- Advanced security mechanisms
- Multi-user and network-based system

Interface-UI

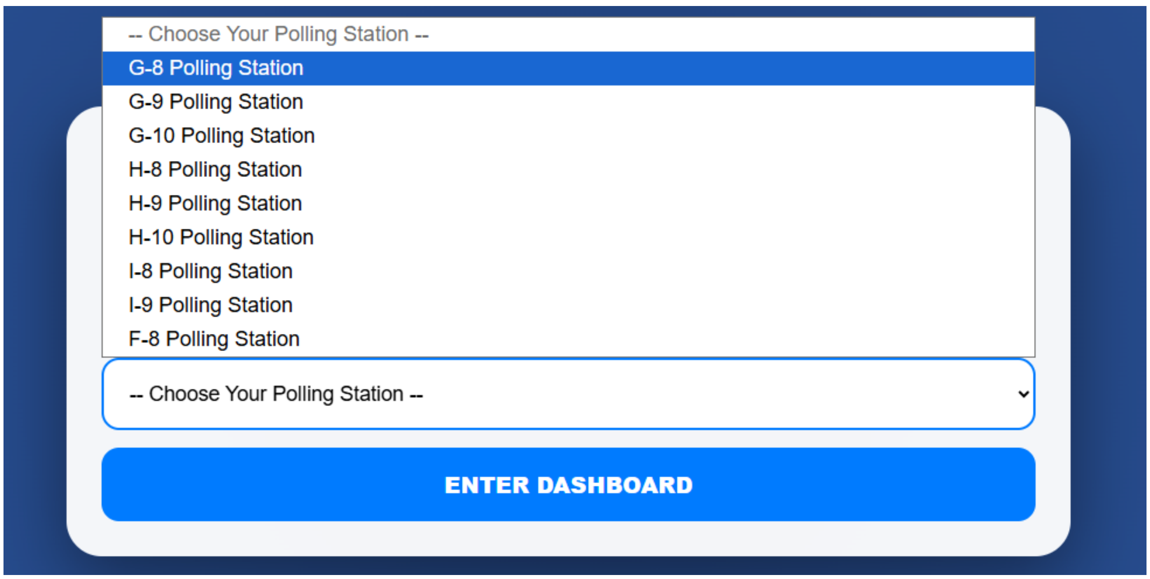
Welcome To Electronic Voting Machine



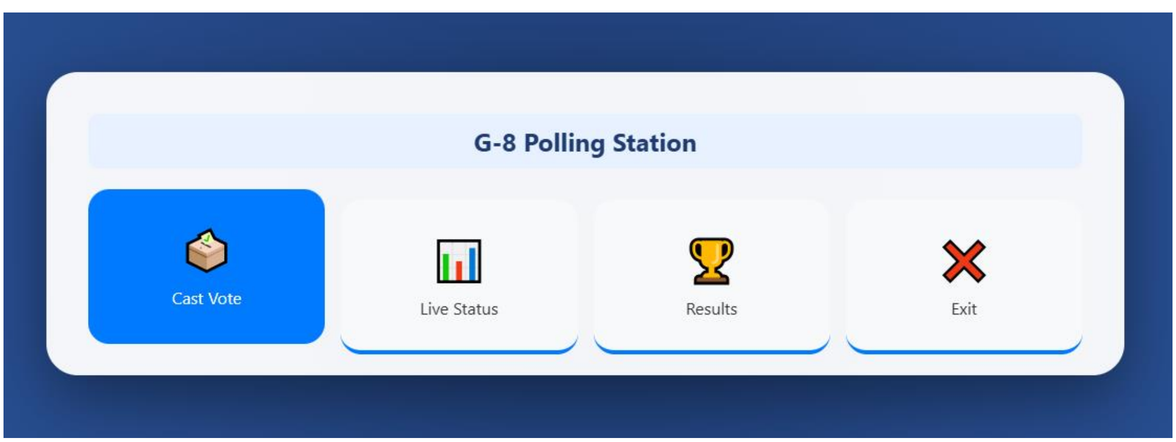
Must Select Your Polling Station before proceeding:



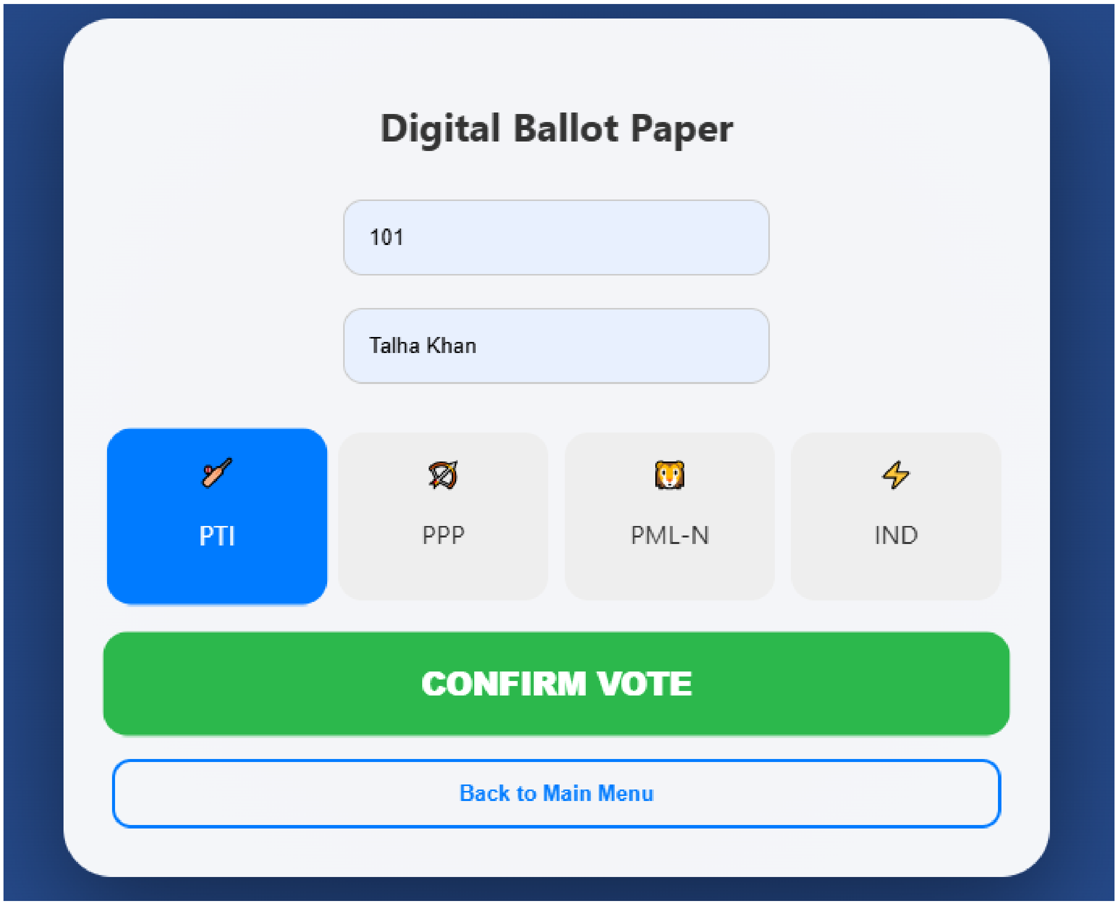
Select your Polling Station



Main Menu-(Cast Vote)

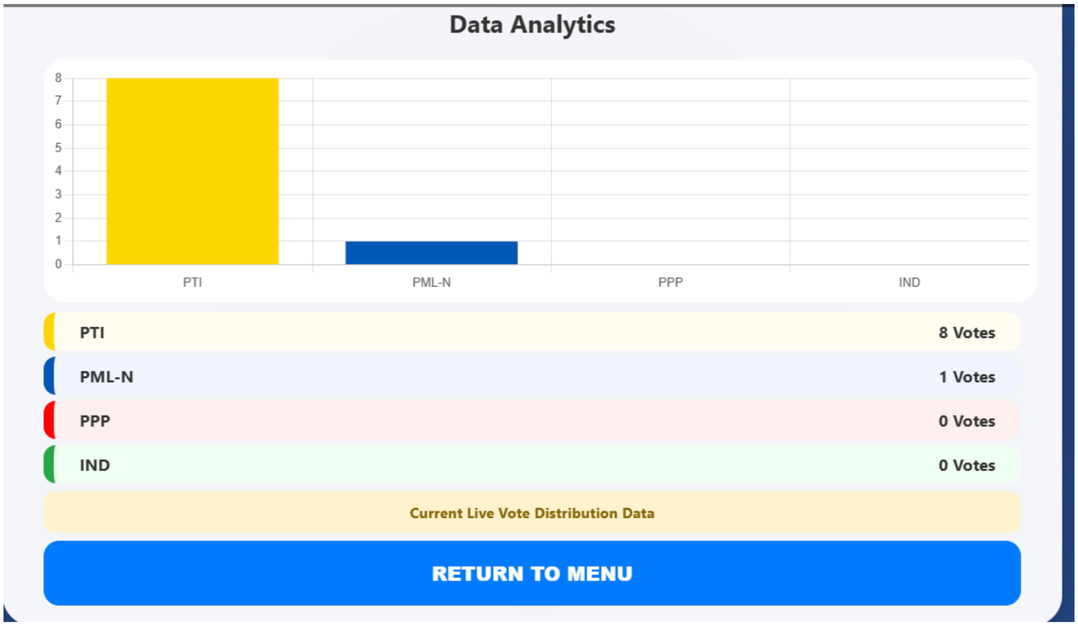


Cast Vote

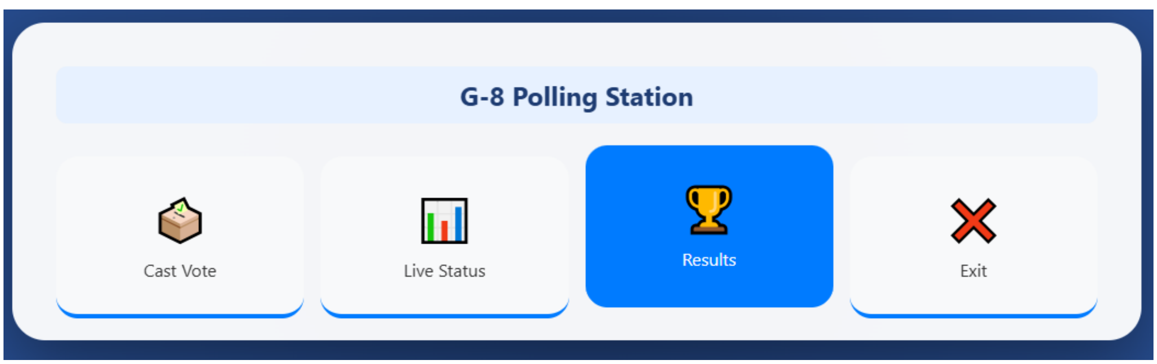


Main Menu-(Live Updates)

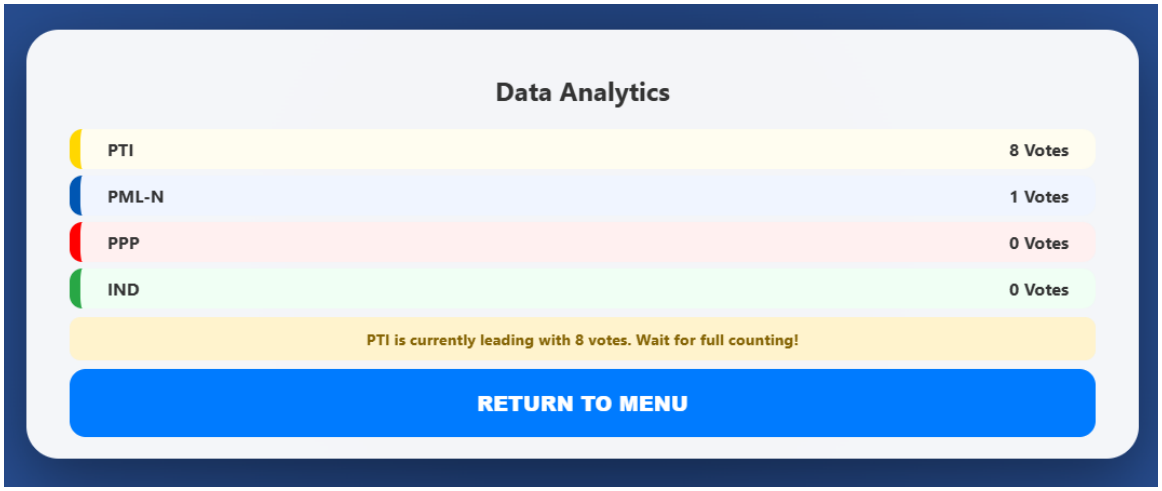
Graph for easy visualization



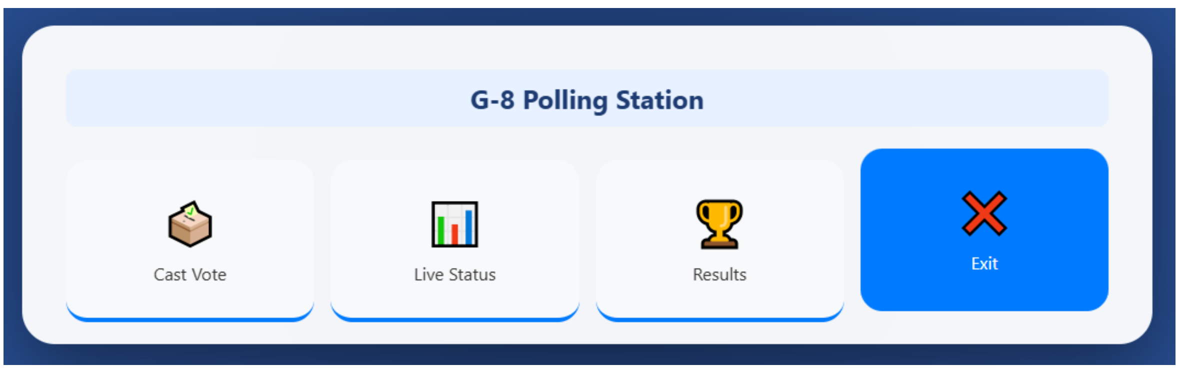
Main Menu-(Results)



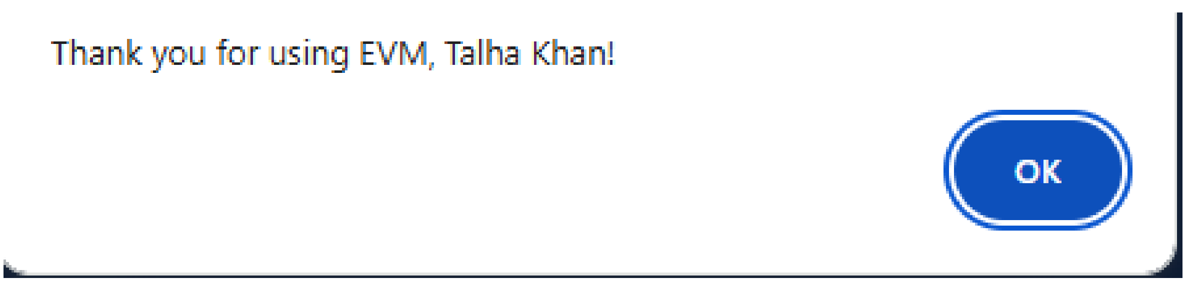
Result



Exit From Program:



End Message



Thank You!!!