**OBJECT ORIENTATED PROGRAMMING LAB**

**Project Report**

**E VOTING SYSTEM**



**Submitted by:**

**Muhammad Haris 2024-CS-613**

**Submitted to:**

**Prof. Zoha Sohail**

**Dated**: 11th May, 2025

**Department of Computer Science**

**University of Engineering and Technology Lahore, New Campus**

**E VOTING SYSTEM**

1. **Introduction and objectives:**

The **Online Voting System** is a C++ console-based application that simulates a secure and efficient digital voting platform. Built using **Object-Oriented Programming (OOP) principles**, the system enables two primary user roles—**Voters** and **Administrators**—to participate in and manage elections, respectively.

#### **Key Features Demonstrated in the Code:**

* **Role-Based Access:**
  + **Voters** can log in, view active elections, and cast votes (with duplicate voting prevention).
  + **Administrators** can create/manage elections (local, national, regional), add candidates, and view results.
* **Election Management:**
  + Supports multiple election types with customizable parameters (start/end time, region).
* Polymorphism is implicitly used via election-type handling (e.g., national elections require region counts).
* **Data Persistence:**
* Uses file handling (data/users.txt, data/elections.txt) to store user credentials and election details.
* **Security Measures:**
  + Input validation (e.g., date/time formats, CNIC for voter registration).
  + Encapsulation protects sensitive data (e.g., votes, credentials) within class methods.

1. **System work flow :**

* **Setup**: Files (users.txt, elections.txt) are initialized for data persistence.
* **Authentication**: Users log in or register via role-specific menus
* **Election Lifecycle**:
  + Admin creates → starts → ends election.
  + Voters participate during the active window.
* **Results**: Automatically computed post-election and viewable by admins.

1. **Classes used:**

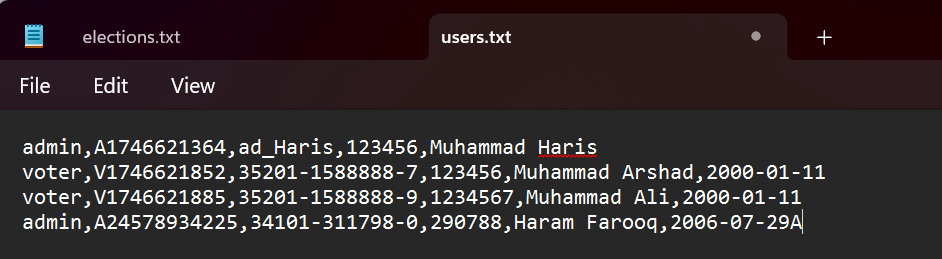
* **User class** – base class
* **Administrator class** – child class (inherited from User)- stores the admin info
* **Voter class** – child class (inherited from User)- stores voter data
* **Candidate class** – stores the voting information of the candidate
* **Election class** – stores election info and candidates
* **ElectionManager class**- manages election of different types (creating, deleting etc)
* **LocalElection class**, **Regional election class**, **National election class**.
* **Voting system class**

1. **Constraints Applied :**

* Voter can't vote if age < 18
* Voter can't vote if election doesn't exist
* Voter can't vote if election not running
* Candidate can't be added if election doesn't exist
* Candidate can't be edit if election is already running
* Two voters can't have same ID

1. **File handling:**

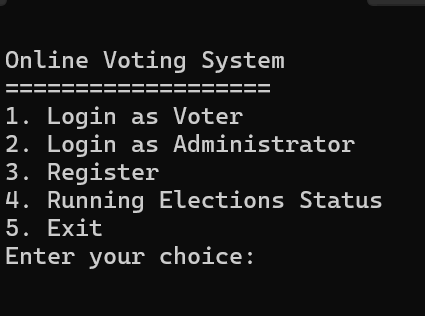
* **Election.txt –** stores the data of all type of elections (local, regional , national)
  + **Format –** new line
* **User.txt**- stores the data of all type of users (admin, voter)
  + **Format**  - CSV file



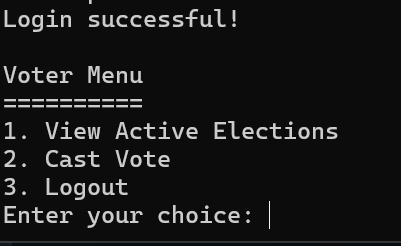
1. **User Interface (UI) menu:**

In E voting we used different menu for different users .

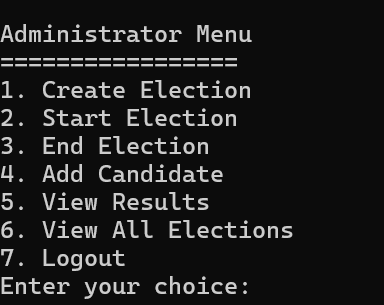
* **Starting menu:**



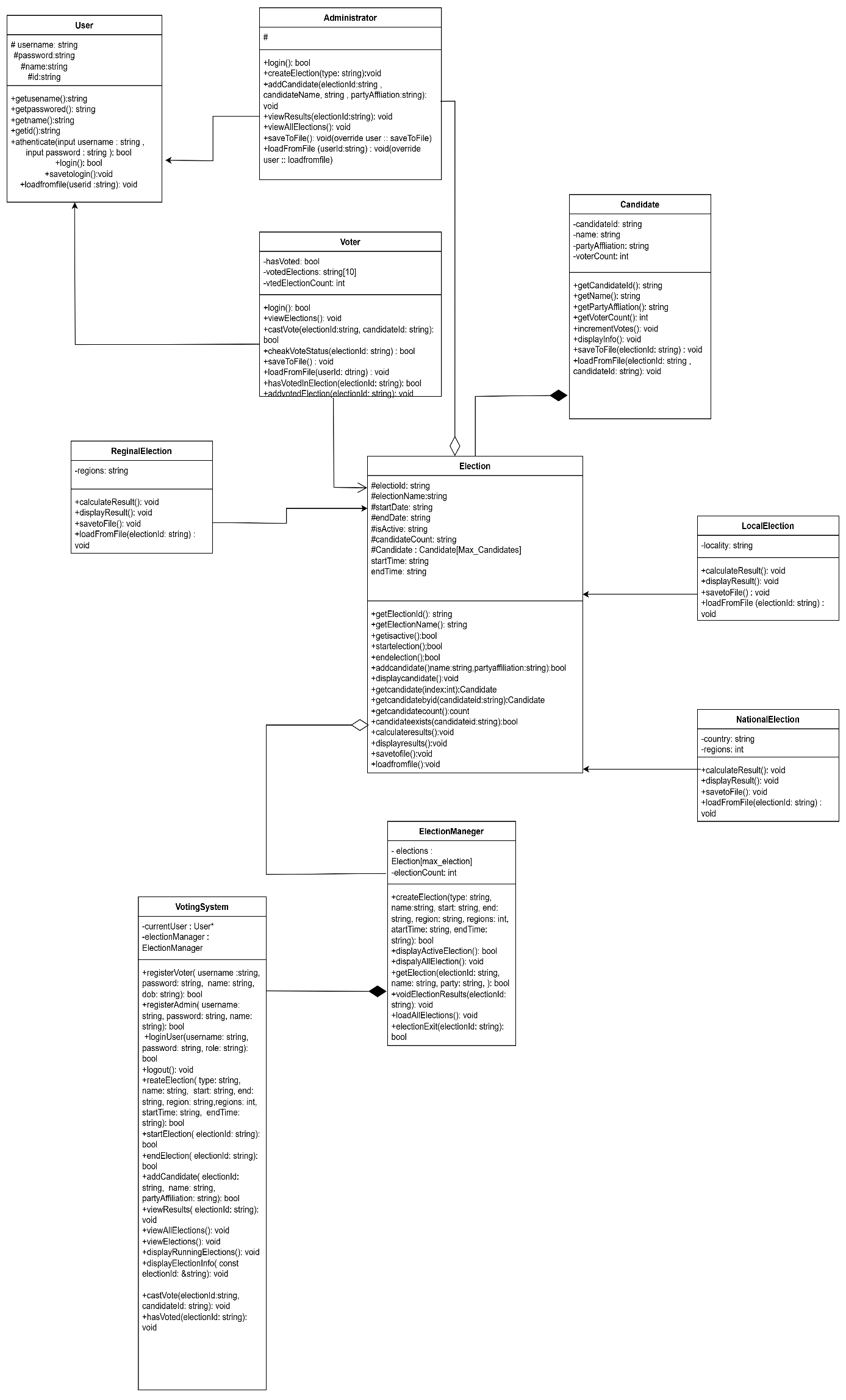
* **Voter menu:**

****

* **Admin menu:**

****

1. **Class diagram:**

****

1. **Conclusion:**

The **Online Voting System** effectively applies **OOP principles** to create a secure digital voting platform. Key features include:

* **Role-based access** for voters and admins
* **Polymorphic election handling** (local/regional/national)
* **Anti-fraud mechanisms** (duplicate vote prevention)
* **File-based persistence** for data storage

While functional, future improvements could include **database integration** for scalability and **enhanced security** like password hashing. This project demonstrates how OOP can build **real-world systems** with clear structure and extensibility.