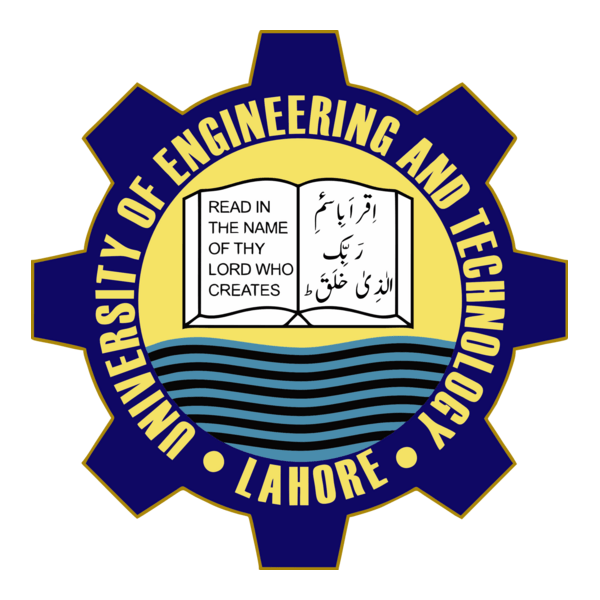
**UNIVERSITY OF ENGINERING AND TECHNOLOGY (NEW CAMPUS )**

**Prepared BY**

**Humair Ali Awan (2024-CS-634)**

**Amar Aslam(2024-CS-610)**

**Wajeeha Javad (2024-CS-601)**

**Momna Maqsood (2024-CS-612)**

**Supervised By**

**Mam Zoha Sohail**

**E-VOTING SYSTEM**

**SEMESTER PROJECT **

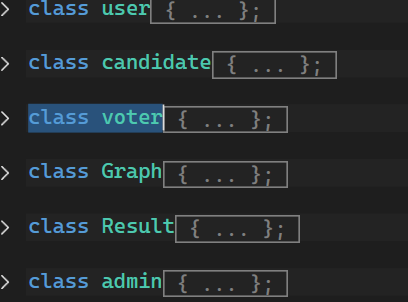
**Table of Contents**

1. **Introduction**
2. **Project Overview**
3. **System Features**
4. **Class Hierarchy & Design**
5. **Functionality & Implementation**
6. **File Handling & Data Management**
7. **User Interface & Experience**
8. **Security & Validation**
9. **Challenges & Solutions**
10. **Future Enhancements**
11. **Conclusion**

**1. Introduction**

The **E-Voting System** is a secure, efficient, and user-friendly digital platform designed to automate the election process. It ensures transparency, reduces human error, and provides real-time results. This project was developed using **C++** with **object-oriented programming (OOP)** principles, file handling for data persistence, and a terminal-based interface.

**2. Project Overview**

The system supports three main user roles:

1. **Admin** – Manages elections, approves registrations, and views results.
2. **Voter** – Casts votes in assigned areas.
3. **Candidate** – Participates in elections and checks vote counts.

Key functionalities:

* **Election creation** (National/Provincial)
* **Voter & candidate approval**
* **Secure login with password protection**
* **Real-time results with graphical representation**
* **Blocking/unblocking users after failed attempts**

**3. System Features**

**Admin Features**

* Create and manage elections.
* Add/remove voters and candidates.
* Approve pending registrations.
* Reset election results.
* Unblock users.

**Voter Features**

* View personal details.
* Cast votes (one-time).
* Check election status.

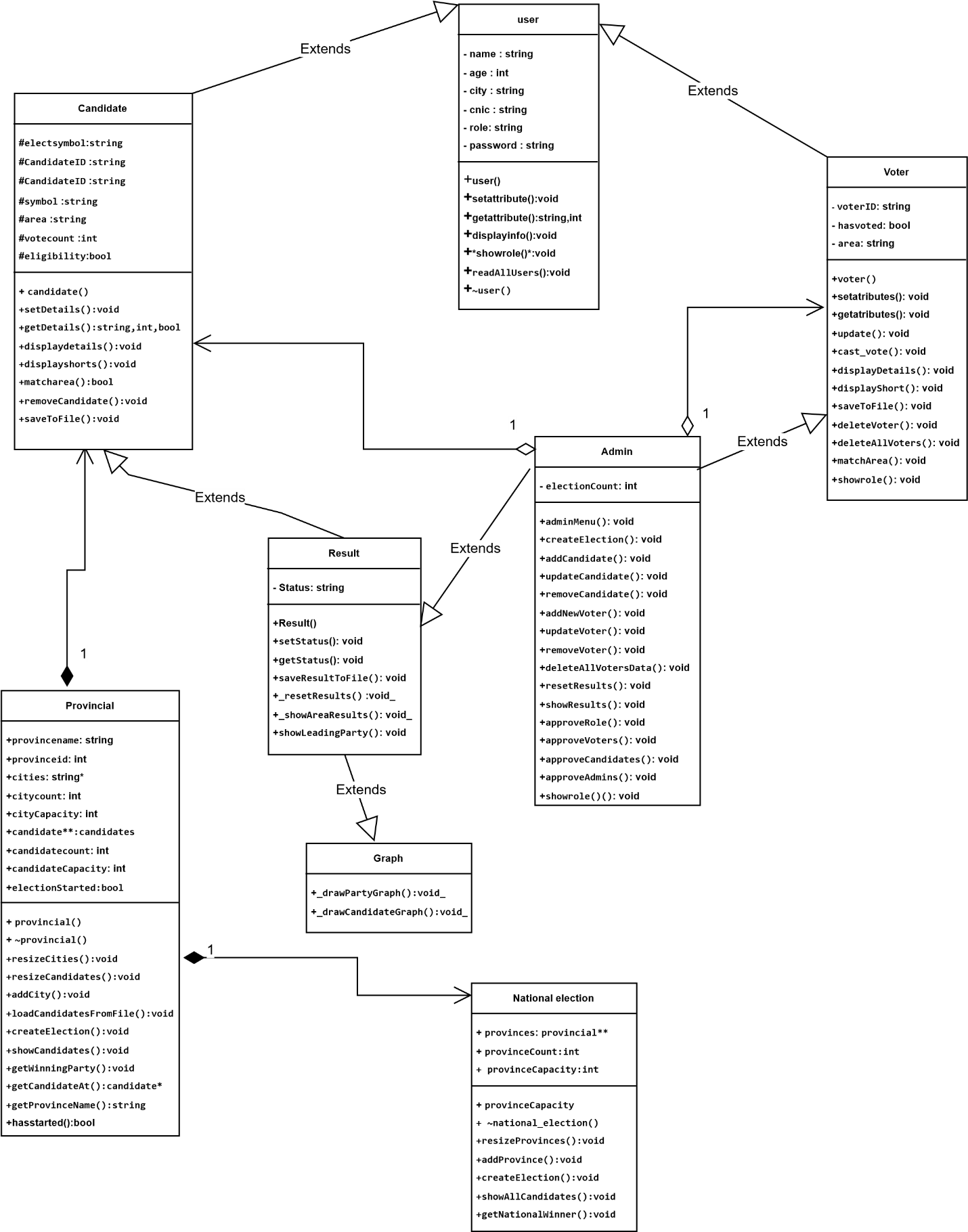
**Candidate Features**

* View vote counts.
* Check area-wise standings.
* Update profile.

**Security Features**

* **CNIC validation** (15-digit format).
* **Password protection** (3-attempt limit before blocking).
* **Role-based access control**.

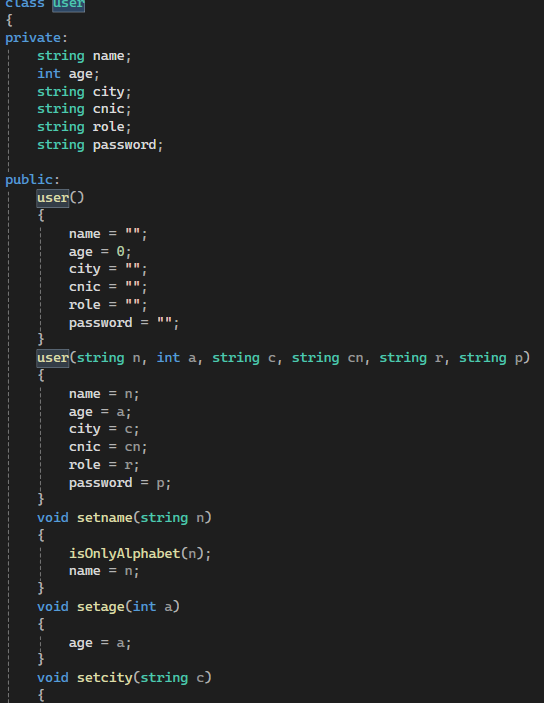
**Class Hierarchy & Design**



The system follows an **inheritance-based OOP model**:

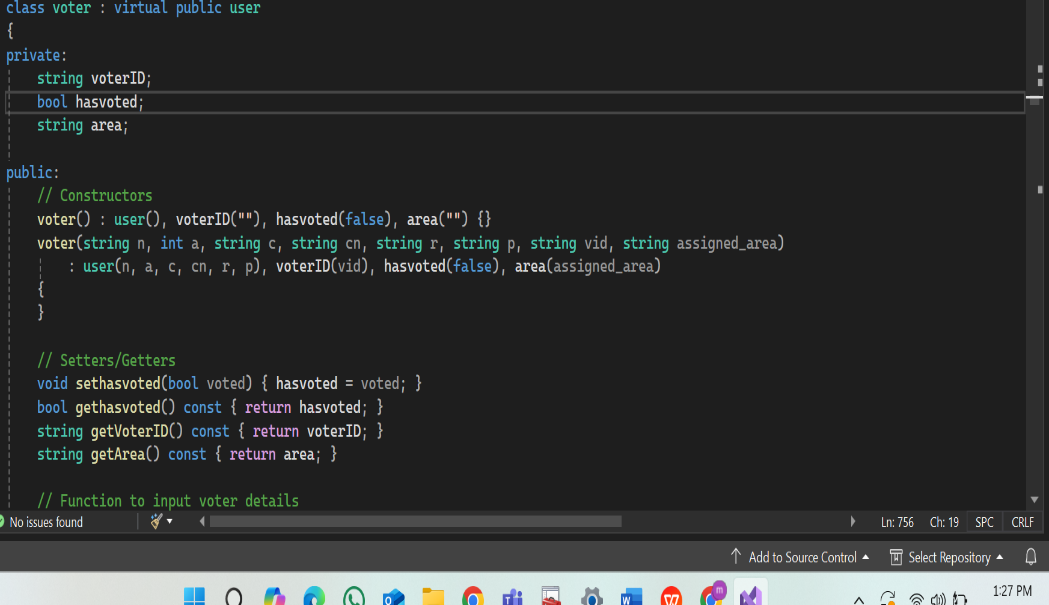
**Base Class:**

* user (Abstract class with virtual functions)
  + Stores common attributes: name, age, CNIC, role, password.

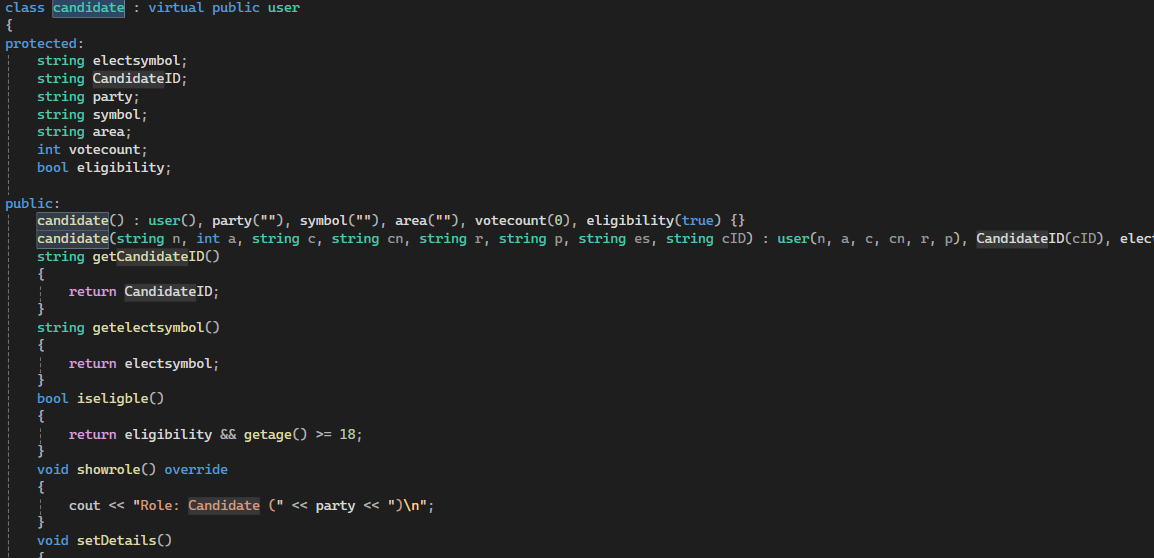


**Derived Classes:**

1. voter – Inherits from user.
   * Additional attributes: voterID, hasVoted, area.



1. candidate – Inherits from user.
   * Additional attributes: party, symbol, voteCount

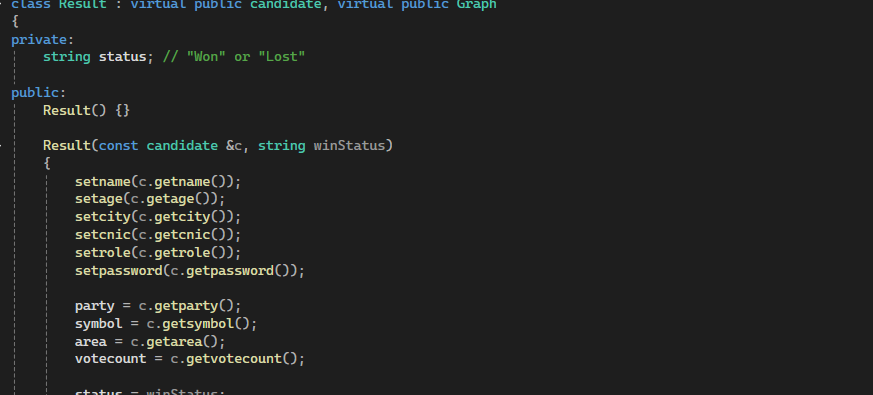
.

1. admin – Inherits from voter and Result (multiple inheritance).
   * Manages elections and user approvals.

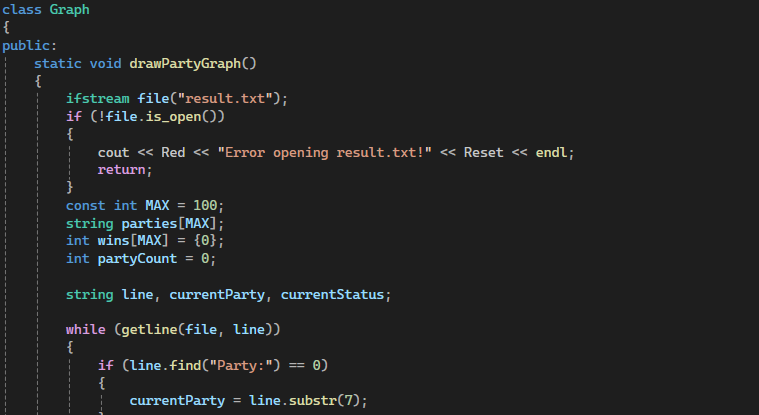


**Supporting Classes:**

* Result – Handles election results and vote counting.



* Graph – Displays results as bar graphs.



* provincial**&**national\_election – Manage election logistics.



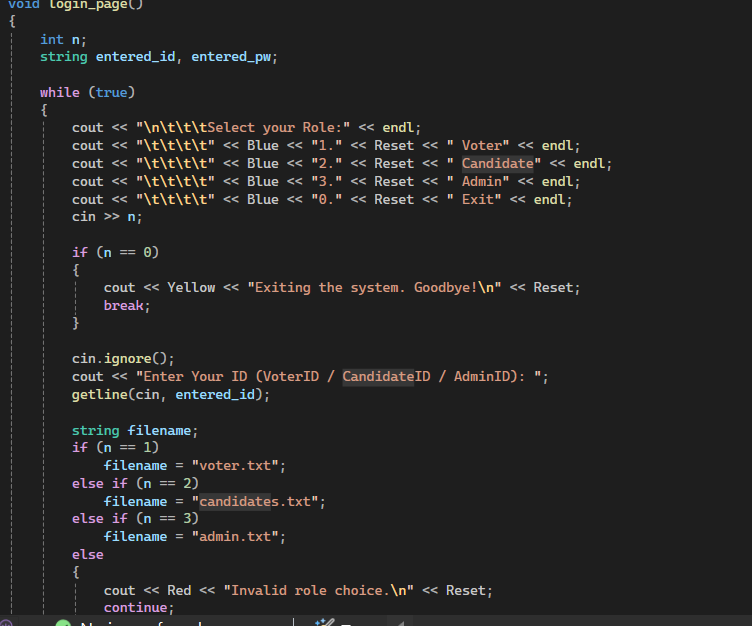
**5. Functionality & Implementation**

**Key Algorithms**

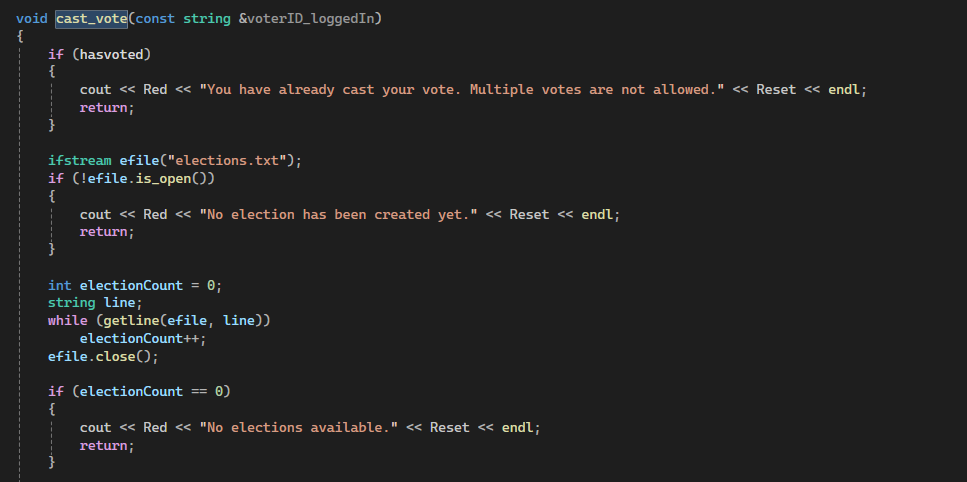
* **File Handling** – Data stored in .txt files (voter.txt, candidate.txt, etc.).
* **Sorting** – Manual selection sort for candidate rankings.
* **Time Management** – ctime library for election deadlines.

**Critical Functions**

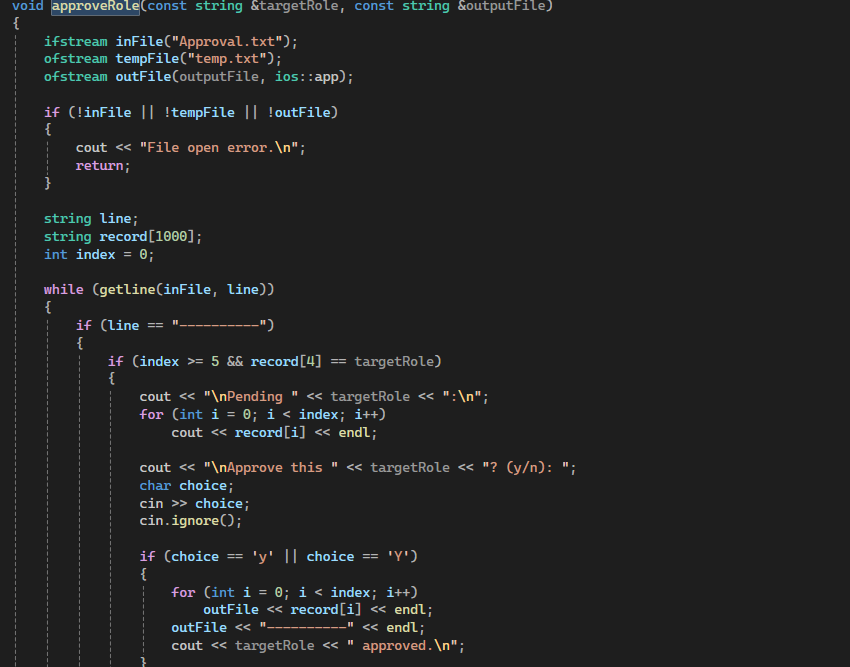
1. login\_page() – Authenticates users with role-based access.



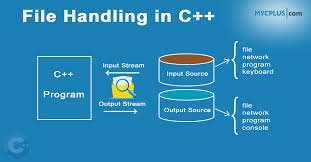
1. cast\_vote() – Ensures one vote per voter.



1. approveRole() – Processes pending registrations.



1. unblockUser() – Restores blocked accounts.
2. **File Handling & Data Management**



* **Text files** store user data with structured formats (e.g., ---------- separators).
* **Example:**

plaintext

Copy

Download

Ali

25

42201-1234567-8

Lahore

Voter

password123

V001

Punjab

0

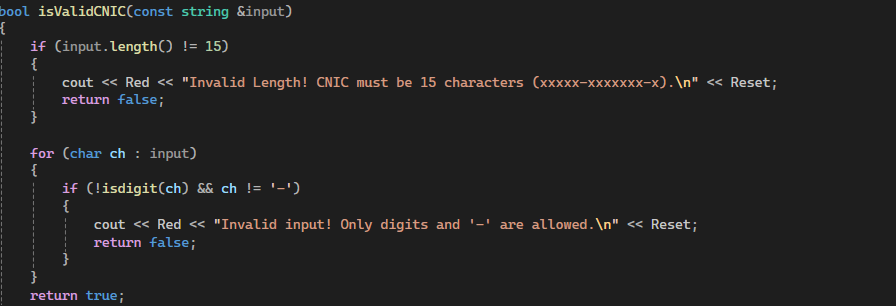
----------

**7. User Interface & Experience**

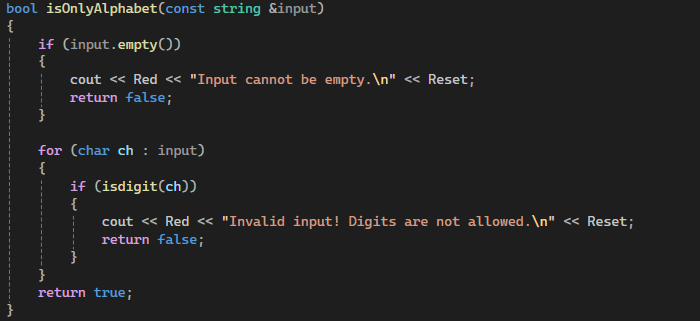
* **Color-coded menus** (red for errors, green for success).
* **Interactive prompts** for navigation.
* **Graphical vote displays** using ASCII art.

**8. Security & Validation**

* **CNIC Format Validation**: Enforces XXXXX-XXXXXXX-X pattern with digit checks



* **Password Security**: Blocks accounts after 3 failed login attempts
* **Age Restriction**: Requires minimum age of 18 for voters/candidates
* **Alphabet-Only Validation**: Rejects numeric characters in name/city fields.



**9. Challenges & Solutions**

| **Challenge** | **Solution** |
| --- | --- |
| Multiple inheritance complexity | Used virtual inheritance for admin. |
| File corruption risks | Temp files for safe updates. |
| Real-time result updates | Frequent file reads/writes. |

**10. Future Enhancements**

1. **Database Integration** (SQLite for scalability).
2. **GUI** (Qt or SFML for better UX).
3. **Blockchain** for tamper-proof voting.
4. **Mobile App** for remote access.

**11. Conclusion**

This project demonstrates a functional e-voting system with **secure authentication**, **efficient data handling**, and **real-time result tracking**. It adheres to OOP principles and can be expanded for larger-scale deployments.

THANK YOU