

**Castify - Online Voting Management System**

Group Members:

Areej Chaudhary 2024-CS-698

Fiza Shahid Khan 2024-CS-680

Sajid Mehmood Tariq 2024-CS-660

**VOTING MANAGEMENT SYSTEM**

OOP SEMESTER PROJECT

# 1. Introduction

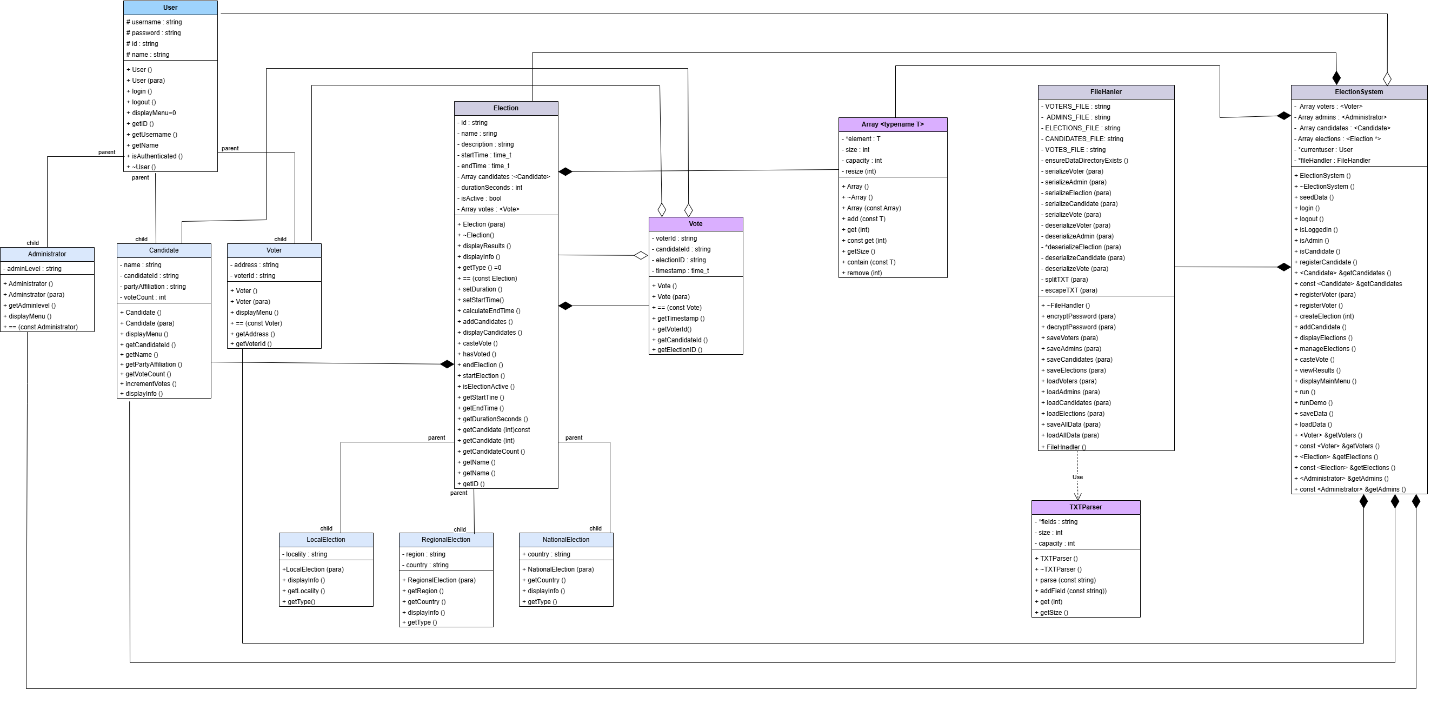
Castify is an object-oriented, file-based voting management system developed in C++. It provides a command-line interface for managing elections, candidates, voters, and votes, supporting role-based access for administrators, voters, and candidates. The system is designed to avoid STL containers, instead using a custom array template for data management, and persists all data in plain text files.

[**GitHub Repository:**](https://github.com/sajidmehmoodtariq30/OOP_Semester_Project)

# 2. UML Diagram

The following UML diagram (located in the assets folder of the repository) illustrates the main classes and their relationships:

UML Diagram



# 3. Project Structure

OOP\_Semester\_Project/

├── main.cpp

├── build.bat

├── CMakeLists.txt

├── assets/

│   ├── logo.png

│   └── oop uml.drawio.png

├── data/

│   ├── admins.txt

│   ├── candidates.txt

│   ├── elections.txt

│   ├── voters.txt

│   └── votes.txt

├── src/

│   ├── header/

│   │   ├── Admin.h, Array.h, Candidate.h, Election.h, ...

│   └── source/

│       ├── admin.cpp, candidate.cpp, election.cpp, ...

├── build/ (output .exe)

├── docs/

│   └── OOP project 2025.docx, ...

└── README.md

# 4. Core Features

## 4.1 User Management

Roles: Administrator, Voter, Candidate.

Authentication: Users log in with a username and password. Passwords are stored in encrypted form using a Caesar cipher.

Role-based Menus: Each user type sees a different menu with actions relevant to their role.

## 4.2 Election Management

Types: Local, National, Regional elections, each with specific attributes (e.g., locality, country, region).

Creation: Administrators can create elections, specifying type, duration, and other details.

Activation: Elections can be started and ended by administrators. Elections automatically end after their set duration.

## 4.3 Candidate Management

Registration: Administrators can register candidates for elections.

Party Restrictions: For national elections, only one candidate per party is allowed.

Profile Viewing: Candidates can view their profiles and election results.

## 4.4 Voting System

Voting: Authenticated voters can cast a vote in active elections.

Duplicate Prevention: Each voter can vote only once per election.

Vote Counting: Votes are counted and associated with candidates.

## 4.5 Results Processing

Real-Time Results: Results can be viewed at any time, showing vote counts and percentages for each candidate.

Reporting: Results are displayed in a user-friendly format.

## 4.6 Data Persistence

File-Based Storage: All data (users, elections, candidates, votes) is stored in plain text files in the data directory.

Serialization/Deserialization: Custom logic for reading and writing objects to files, including handling of special characters and encryption for passwords.

Automatic Loading/Saving: Data is loaded at startup and saved on shutdown or after significant changes.

# 5. Implementation Details

## 5.1 Object-Oriented Design

Inheritance: User is an abstract base class, with Administrator, Voter, and Candidate as derived classes.

Polymorphism: Menus and actions are handled via virtual functions, allowing role-specific behavior.

Encapsulation: All data members are private or protected, accessed via getters/setters.

## 5.2 Custom Array Template

The project uses a custom Array<T> template (Array.h) for dynamic arrays, replacing STL containers.

Provides methods for adding, accessing, removing, and checking elements.

## 5.3 File Handling

FileHandler (FileHandler.h, FileHandler.cpp) manages all file I/O.

Includes methods for serializing and deserializing each entity (voters, admins, candidates, elections, votes).

Uses a simple CSV-like format with custom escaping for commas and quotes.

## 5.4 Security

Password Encryption: Passwords are encrypted using a Caesar cipher before being saved to disk.

Role-Based Access Control: Only administrators can perform sensitive actions like registering voters/candidates or managing elections.

## 5.5 User Interface

CLI: All interaction is via the command line, with clear prompts and input validation.

Masked Password Input: Passwords are entered with masking for privacy.

# 6. Data Files

* data/admins.txt: Stores administrator credentials and info.
* data/voters.txt: Stores voter credentials and info.
* data/candidates.txt: Stores candidate info, including election association.
* data/elections.txt: Stores election metadata.
* data/votes.txt: Stores votes cast, linking voters, candidates, and elections.

# 7. Example Workflow

Startup: System loads data from files or seeds default data if files are missing.

Login: User selects role and logs in.

Admin Actions: Create elections, register voters/candidates, manage elections, view results.

Voter Actions: Cast vote in active elections, view results.

Candidate Actions: View profile, view election results.

Shutdown: Data is saved back to files.

# 8. Strengths and Limitations

Strengths

Clear OOP structure with extensibility for new roles or election types.

No STL dependency, meeting assignment constraints.

Simple, readable file format for data persistence.

Role-based security and input validation.

Limitations

Password encryption is weak (Caesar cipher, not secure for real-world use).

No concurrency or multi-user support (single-user CLI).

No graphical interface (CLI only).

Limited error handling for file corruption or malformed data.

# 9. Conclusion

Castify is a well-structured, educational voting management system that demonstrates core C++ OOP principles, custom data structures, and file-based persistence. It is suitable for academic purposes and as a foundation for more advanced systems.