Information Security

Department of Computer Science

University of Engineering and Technology, Lahore

Session-2025 **Object Oriented Programming-Lab**

Project Description: Online Voting System

No use of vector, maps or any other built-in libraries.

Objective: The Online Voting System is designed to provide a digital platform where eligible voters can cast their votes in different types of elections (e.g., local, national, or regional). The system aims to simplify the voting process, ensure transparency, and allow users to participate in elections remotely and securely. The project will use Object-Oriented Programming (OOP) principles like classes, inheritance, polymorphism, and abstraction to create a modular, flexible, and extensible solution.

System Overview:

The Online Voting System will allow registered voters to authenticate, choose candidates, and cast their votes electronically. The system will consist of various user roles (e.g., voter, administrator) that interact with the system in different ways. It will also accommodate different election types, each with its own set of rules, candidates, and processes.

Key Features:

1. **User Authentication:**
    - Users will have to log in using their credentials (username/password).
    - Only authenticated users can vote, ensuring election security.
2. **User Roles:**
    - **Voter**: A registered voter can view the list of elections, vote for a candidate, and check the status of their vote.
    - **Administrator**: The administrator will have elevated privileges, including the ability to create elections, add candidates, and manage the voting process. The admin can also view voting results and ensure the integrity of the election.
3. **Election Management:**
    - The system will support different types of elections such as **local**, **national**, and **regional** elections.
    - Each election will have its own set of candidates, voting rules, and voting methods.
    - An administrator can create and modify election details, including the type of election, the duration, and the list of candidates.
4. **Voting Process:**
    - Voters will be able to select a candidate from a list and submit their vote. Once the vote is cast, it will be recorded, and the system will prevent multiple voting by the same user for a particular election.

- The system will ensure that once the voting period ends, the results can be computed automatically based on the votes received.

5. **Polymorphism in Elections:**
   - The system will use polymorphism to handle different types of elections, as each election type (e.g., local vs. national) might have different voting procedures or rules.
   - The **Election** class will be a base class, and each specific election type (e.g., **LocalElection**, **NationalElection**) will be derived from it.
   - Each derived class can override base class methods to implement its own election-specific behavior (e.g., local elections might allow voting for more candidates compared to national elections).
6. **Candidate Management:**
   - The system will maintain a list of candidates for each election. Voters will be able to view the list of candidates and their party affiliations (if applicable).
   - Candidates can be added, modified, or removed by the administrator.
7. **Results and Statistics:**
   - After the voting period has ended, the administrator will be able to view detailed voting results.
   - The system will display results in a user-friendly manner, such as graphs or tables, showing the number of votes each candidate received.

Class Design:

1. **User Class**: A base class that will define the common attributes and methods for all users (e.g., username, password, login()).
   - **Voter Class**: Derived from the User class, this class represents a voter. It will have methods like viewElections(), castVote(), and checkVoteStatus().
   - **Administrator Class**: Also derived from the User class, the admin class will include methods such as createElection(), addCandidate(), and viewResults().
2. **Election Class**: A base class for elections with common properties and methods, including startElection(), endElection(), and addCandidate().
   - **LocalElection Class**: Derived from the Election class, this class will have election-specific logic, like the ability to handle multiple candidates in a single region.
   - **NationalElection Class**: Another derived class, tailored for national elections, potentially including rules like party affiliations and voting in multiple regions.

3. **Candidate Class**: Represents the candidates running in an election. It will have attributes like name, party affiliation, and the number of votes received.
    - Methods: incrementVotes(), getVoteCount(), getCandidateInfo().

Object-Oriented Concepts Applied:

- **Abstraction**: By using abstract classes like Election and User, the system hides the complex implementation details from the user. This ensures that each user only interacts with the methods relevant to their role (voter or admin).
- **Inheritance**: The system uses inheritance to extend the base classes (e.g., User, Election) to specialized classes like Voter, Administrator, and different election types (LocalElection, NationalElection). This allows for code reusability and clarity.
- **Polymorphism**: Polymorphism allows the system to handle different types of elections and user roles dynamically. By using virtual functions, the program can invoke specific election rules (such as vote counting) for local and national elections without needing to modify the core voting process.
- **Encapsulation**: All user data (credentials, votes, etc.) and election details (candidate lists, voting methods) are encapsulated within respective classes. Only authorized classes or methods have access to this data.

System Flow:

1. **Administrator creates an election**: The admin creates an election (e.g., national, local) and adds candidates.
2. **Voters login**: Registered voters log in to the system to access the election.
3. **Voting process**: Voters view candidates, select one, and cast their vote.
4. **End of voting period**: Once the election period ends, the system calculates the votes and generates results.
5. **Results display**: The administrator can view the final voting results and analyze the data.