|  |  |  |
| --- | --- | --- |
| **1. User Authentication & Role Management** | Proper user authentication and role-based access control. Only authenticated users can vote. | 5 Marks |
| **2. Election Management** | Admin can create elections, add candidates, and manage election types and voting methods. | 5 Marks |
| **3. Voting Process** | Voters can cast votes, ensuring single voting per election. Voting system prevents multiple votes. | 5 Marks |
| **4. Polymorphism & Election Types** | Correct implementation of polymorphism with derived election classes (e.g., LocalElection, NationalElection). | 5 Marks |
| **5. Encapsulation & Class Design** | Proper encapsulation, clear class responsibilities, and OOP principles are applied. | 3 Marks |
| **6. Results and Statistics** | Election results are displayed clearly after the voting period ends (tables/graphs). | 3 Marks |
| **7. Code Quality & Organization** | Code is well-organized, readable, and modular with proper comments and naming conventions. | 2 Marks |
| **8. Testing and Validation** | Adequate testing and validation for the voting process and edge cases. | 2 Marks |
| **9. Documentation and demo video** | Complete documentation, explaining setup, usage, and functionalities. | 3 Marks |
| **10. OOP Concepts** | Correct application of OOP principles like inheritance, abstraction, polymorphism, and encapsulation. | 7 Marks |

**Project Rubrics**