ELECTION MANAGEMENT SIMULATION – (JAVA Project)

CLUSTER INNOVATION CENTRE

UNIVERSITY OF DELHI

Name of Students

MUDIT GOEL

YASH BIJALWAN

RAVI RANJAN KUMAR

May

2025

SEMESTER LONG PROJECT SUBMITTED FOR THE PAPER

Object Oriented Programming (Java)

Election Management System (Java OOP-Based Project) - Proposal

A Java-based standalone election management system that allows admins, candidates, and voters to participate in a secure, transparent, and efficient election process.

1. Features and Functionalities

A. Admin Functionalities :-

1. Manage Candidates

- Add/update candidate details (Name, Party, Election Manifesto).
- Approve/reject candidate registration.

2. Manage Voters

- Add/update voter list with unique voter IDs.
- Assign election credentials to voters

3. Election Control

- Schedule election start and end time.
- Declare the winner once the election ends.
- View real-time vote count.

B. Candidate Functionalities:-

1. Election Campaigning

- Provide an election manifesto visible to voters.
- View total registered voters.

2. Monitor Votes

- Check live vote count (but not individual voter details).
- Receive final vote count once election ends.

C. Voter Functionalities:-

1. Election Participation

- View list of candidates and their manifestos.
- Cast vote securely (one vote per voter).

2. Transparency & Information

- Get election updates (status, total votes cast).

D. Security & Data Handling:-

- Unique Voter ID & Authentication for secure login.
- One-person-one-vote validation to prevent multiple votes.
- Data Encryption for voter credentials.

Introduction:-

The **Election Management System** is a software application designed to streamline and manage the election process using Java and Java Swing. It aims to provide an efficient, secure, and user-friendly platform for managing elections, handling various roles like Admins, Voters, and Candidates. This system allows the registration of candidates, the casting of votes, vote tallying, and the declaration of results, all while ensuring secure and accurate processes. The system utilizes Object-Oriented Programming principles, multithreading, and exception handling to ensure reliability, security, and scalability.

Features of the Election Management System

1. **User Roles and Access Control**:

- **Admin**: Manages the entire election process by adding candidates, starting the election, and declaring winners. Admin can also monitor the vote count.
- **Voter**: Casts votes for candidates. A voter can only vote once, and the system ensures that no duplicate voting occurs.
- **Candidate**: View vote counts and statistics. Candidates can be added to the system and their votes tracked.

2. **Candidate Management**:

- Admin can add new candidates with specific details such as name, party affiliation, and voting location.
- Candidates can be registered with a limit on the number of words in their name.

3. **Voting Process**:

- Voters can cast votes for candidates by selecting the candidate number.
- The system ensures that a voter can vote only once using a secure method of tracking whether a voter has voted.

4. **Vote Tallying and Result Display**:

- The system keeps track of votes in real-time, updating the total votes for each candidate as votes are cast.
- Results are available at any time, and the admin can see a detailed breakdown of votes for each candidate.

5. **Multithreading Support**:

- The system supports concurrent voting by using **multithreading**. Each vote casting operation runs in a separate thread, enabling the handling of multiple votes at the same time without affecting system performance.

6. **Exception Handling**:

- Comprehensive exception handling ensures the system runs smoothly, even in cases of invalid input, duplicate voting, or incorrect operations. For example, voters are prevented from casting votes more than once, and invalid candidate IDs are handled gracefully.

7. **Dynamic User Interaction**:

- Users interact with the system through a text-based menu. Depending on their role (Admin, Voter, or Candidate), they are presented with the appropriate actions they can perform (e.g., cast vote, add candidate, check votes, etc.).

8. **Data Persistence**:

- The system can be extended to store election results and candidate information in a file or database for future reference and audit purposes.

9. **Security**:

- The system ensures that the integrity of the election is maintained. Voter authentication can be extended for added security, ensuring that only eligible voters can cast votes.

Comparison Sheet:-

The provided "Election Management System" code offers various functionalities, and a comparison with the Election Commission of India's (ECI) Election Management System can be based on several aspects:

1. Roles and Users:

ECI System: Typically, ECI's system has administrators at different levels, such as district election officers, returning officers, and booth-level officers. It also handles candidates and voters across multiple constituencies.

Provided Code: Supports three roles—Admin, Candidate, and Voter—with basic operations like adding voters/candidates, conducting campaigns, and casting votes

.

2. User Verification:

ECI System: Voter verification in the ECI system uses biometric data, Voter IDs (EPIC), and sometimes Aadhaar linkage.

Provided Code: Uses simple name and ID matching for voter verification, without biometric or external authentication integration

.

3. Candidate Information:

ECI System: Provides detailed public information about candidates, including assets, criminal records, and affidavits as per public disclosure rules.

Provided Code: Allows candidates to be added with details like name, party, symbol, assets, and criminal cases, but lacks advanced features like affidavit handling

.

4. Campaigns and Elections:

ECI System: In the ECI system, campaigns are subject to strict guidelines, with monitoring by observers. Election timings are predetermined, and results are published after formal counting.

Provided Code: Campaigns can be manually started/ended by the admin. Elections have a customizable duration with real-time vote counting and winner announcement

.

5. Security and Voting Integrity:

ECI System: Ensures voting security through Electronic Voting Machines (EVMs) with Voter Verifiable Paper Audit Trails (VVPATs), centralized control, and multi-layer encryption.

Provided Code: Has a simpler system where votes are cast through a GUI. The admin has control over starting/stopping elections, but there's no mention of encryption or audit mechanisms

.

6. Real-time Vote Counting:

ECI System: Vote counting happens after the election and is highly regulated, ensuring transparency.

Provided Code: Displays live vote counts to candidates and admins during the election. This feature may compromise secrecy, as ongoing results are not usually shown in real elections

.

7. Scalability:

ECI System: Designed for a national-scale operation involving millions of voters and thousands of candidates across hundreds of constituencies.

Provided Code: More suitable for small, localized elections due to the limited scope of data management and user roles

.

In summary, while the provided system code is a basic election management system with key features like candidate and voter management, election control, and vote counting, the ECI's system is far more robust, secure, and scalable, adhering to strict regulations to ensure transparency and accuracy in elections.

Code:-

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.util.List;
import javax.swing.*;
import javax.swing.plaf.basic.BasicButtonUI;

public class FinalElectionApp {
   // Global static fields for the system
   private static Admin admin = new Admin();
   private static Voter currentVoter;
   private static Candidate currentCandidate;
   private static volatile boolean electionRunning = false;
   private static volatile boolean campaignRunning = false;

   // For screen switching using CardLayout
   private static JFrame frame;
   private static JPanel mainPanel;
   private static CardLayout cardLayout;

   // Reusable panels to avoid duplicates
   private static JPanel candidatePanel = null;
   private static JPanel votePagePanel = null;

   // Reference to the voter panel's tabbed pane so we can disable the Vote tab
   // later
   private static JTabbedPane voterTabbedPane = null;

   public static void main(String[] args) {
      SwingUtilities.invokeLater(() -> {
         frame = new JFrame("Interactive Election Management System");
         frame.setSize(900, 750);
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

         cardLayout = new CardLayout();
         mainPanel = new JPanel(cardLayout);

         // Add panels
         mainPanel.add(createRoleSelectionPanel(), "RoleSelection");
         mainPanel.add(createAdminPanel(), "AdminPanel");
         mainPanel.add(createCandidateLoginPanel(), "CandidateLogin");
         mainPanel.add(createVoterPanel(), "VoterPanel");

         frame.add(mainPanel);
         frame.setLocationRelativeTo(null);
```

```
            frame.setVisible(true);
        });
    }

    // Helper method for interactive dialogs using HTML styling
    private static void showInteractiveDialog(Component parent, String message, String title, int
messageType) {
        String styledMessage = "<html><body style='font-family:SansSerif;font-size:18px;'>"
                + message.replaceAll("\n", "<br>") + "</body></html>";
        JOptionPane.showMessageDialog(parent, styledMessage, title, messageType);
    }

    // ---------------------- Create Role Selection Panel ----------------------
    private static JPanel createRoleSelectionPanel() {
        GradientPanel panel = new GradientPanel();
        panel.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(15, 15, 15, 15);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JLabel headerLabel = new JLabel("Election Management Simulation", SwingConstants.CENTER);
        headerLabel.setFont(new Font("SansSerif", Font.BOLD, 40));
        headerLabel.setForeground(new Color(25, 25, 112));
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 2;
        panel.add(headerLabel, gbc);

        JLabel roleLabel = new JLabel("Select Your Role:");
        roleLabel.setFont(new Font("SansSerif", Font.BOLD, 22));
        roleLabel.setForeground(new Color(25, 25, 112));
        gbc.gridy = 1;
        gbc.gridwidth = 1;
        panel.add(roleLabel, gbc);

        String[] roles = { "Admin", "Candidate", "Voter" };
        JComboBox<String> roleBox = new JComboBox<>(roles);
        roleBox.setFont(new Font("SansSerif", Font.PLAIN, 22));
        gbc.gridx = 1;
        panel.add(roleBox, gbc);

        JButton loginButton = new JButton("Login");
        styleButton(loginButton);
        loginButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 2;
        panel.add(loginButton, gbc);
```

```
        loginButton.addActionListener(e -> {
            String selectedRole = (String) roleBox.getSelectedItem();
            if ("Admin".equals(selectedRole)) {
                if (showAdminPasswordDialog(panel)) {
                    showInteractiveDialog(panel, "Login Successful!", "Success",
JOptionPane.INFORMATION_MESSAGE);
                    cardLayout.show(mainPanel, "AdminPanel");
                }
            } else if ("Candidate".equals(selectedRole)) {
                cardLayout.show(mainPanel, "CandidateLogin");
            } else if ("Voter".equals(selectedRole)) {
                cardLayout.show(mainPanel, "VoterPanel");
            }
        });
        return panel;
    }

    // ---------------------- Custom Interactive Admin Password Dialog
    // ----------------------
    private static boolean showAdminPasswordDialog(Component parent) {
        JDialog dialog = new JDialog((Frame) null, "Admin Authentication", true);
        dialog.setSize(400, 200);
        dialog.setLocationRelativeTo(parent);

        GradientPanel content = new GradientPanel();
        content.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JLabel label = new JLabel("Enter Admin Password:");
        label.setFont(new Font("SansSerif", Font.BOLD, 20));
        label.setForeground(new Color(25, 25, 112));
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 2;
        content.add(label, gbc);

        JPasswordField pf = new JPasswordField();
        pf.setFont(new Font("SansSerif", Font.PLAIN, 20));
        gbc.gridy = 1;
        gbc.gridwidth = 2;
        content.add(pf, gbc);

        JButton okButton = new JButton("OK");
        styleButton(okButton);
        okButton.setFont(new Font("SansSerif", Font.BOLD, 18));
```

```
        gbc.gridy = 2;
        gbc.gridx = 0;
        gbc.gridwidth = 1;
        content.add(okButton, gbc);

        JButton cancelButton = new JButton("Cancel");
        styleButton(cancelButton);
        cancelButton.setFont(new Font("SansSerif", Font.BOLD, 18));
        gbc.gridx = 1;
        content.add(cancelButton, gbc);

        final boolean[] accepted = { false };
        okButton.addActionListener(ae -> {
            String password = new String(pf.getPassword());
            if ("admin123".equals(password)) {
                accepted[0] = true;
                showInteractiveDialog(dialog, "Login Successful!", "Success",
JOptionPane.INFORMATION_MESSAGE);
                dialog.dispose();
            } else {
                showInteractiveDialog(dialog, "Incorrect password! Try again.", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        });
        cancelButton.addActionListener(ae -> dialog.dispose());

        dialog.add(content);
        dialog.setVisible(true);
        return accepted[0];
    }

    // ---------------------- Create Admin Panel ----------------------
    private static JPanel createAdminPanel() {
        GradientPanel panel = new GradientPanel();
        panel.setLayout(new BorderLayout());

        JLabel titleLabel = new JLabel("Admin Panel", SwingConstants.CENTER);
        titleLabel.setFont(new Font("SansSerif", Font.BOLD, 36));
        titleLabel.setForeground(new Color(25, 25, 112));
        panel.add(titleLabel, BorderLayout.NORTH);

        JTabbedPane tabbedPane = new JTabbedPane();
        tabbedPane.setFont(new Font("SansSerif", Font.BOLD, 20));

        // For brevity, only Candidate Management tab is shown here.
        JPanel candidateTab = new GradientPanel();
        candidateTab.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
```

```
gbc.insets = new Insets(10, 10, 10, 10);
gbc.fill = GridBagConstraints.HORIZONTAL;

JLabel candidateHeader = new JLabel("Candidate Management");
candidateHeader.setFont(new Font("SansSerif", Font.BOLD, 28));
candidateHeader.setForeground(new Color(0, 102, 204));
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 2;
candidateTab.add(candidateHeader, gbc);

gbc.gridwidth = 1;
gbc.gridy = 1;
gbc.gridx = 0;
candidateTab.add(new JLabel("Name:"), gbc);
JTextField candNameField = new JTextField(15);
candNameField.setFont(new Font("SansSerif", Font.PLAIN, 20));
gbc.gridx = 1;
candidateTab.add(candNameField, gbc);

gbc.gridy = 2;
gbc.gridx = 0;
candidateTab.add(new JLabel("Place:"), gbc);
JTextField candPlaceField = new JTextField(15);
candPlaceField.setFont(new Font("SansSerif", Font.PLAIN, 20));
gbc.gridx = 1;
candidateTab.add(candPlaceField, gbc);

gbc.gridy = 3;
gbc.gridx = 0;
candidateTab.add(new JLabel("Candidate ID:"), gbc);
JTextField candIdField = new JTextField(15);
candIdField.setFont(new Font("SansSerif", Font.PLAIN, 20));
gbc.gridx = 1;
candidateTab.add(candIdField, gbc);

gbc.gridy = 4;
gbc.gridx = 0;
candidateTab.add(new JLabel("Party:"), gbc);
JTextField candPartyField = new JTextField(15);
candPartyField.setFont(new Font("SansSerif", Font.PLAIN, 20));
gbc.gridx = 1;
candidateTab.add(candPartyField, gbc);

gbc.gridy = 5;
gbc.gridx = 0;
candidateTab.add(new JLabel("Symbol:"), gbc);
JTextField candSymbolField = new JTextField(15);
```

```
        candSymbolField.setFont(new Font("SansSerif", Font.PLAIN, 20));
        gbc.gridx = 1;
        candidateTab.add(candSymbolField, gbc);

        gbc.gridy = 6;
        gbc.gridx = 0;
        candidateTab.add(new JLabel("Assets (in Rs):"), gbc);
        JTextField candAssetsField = new JTextField(15);
        candAssetsField.setFont(new Font("SansSerif", Font.PLAIN, 20));
        gbc.gridx = 1;
        candidateTab.add(candAssetsField, gbc);

        gbc.gridy = 7;
        gbc.gridx = 0;
        candidateTab.add(new JLabel("Criminal Cases (Yes/No):"), gbc);
        JTextField candCrimField = new JTextField(15);
        candCrimField.setFont(new Font("SansSerif", Font.PLAIN, 20));
        gbc.gridx = 1;
        candidateTab.add(candCrimField, gbc);

        JButton addCandButton = new JButton("Add/Update Candidate");
        styleButton(addCandButton);
        addCandButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridy = 8;
        gbc.gridx = 0;
        gbc.gridwidth = 2;
        candidateTab.add(addCandButton, gbc);
        addCandButton.addActionListener(e -> {
            String name = candNameField.getText();
            String place = candPlaceField.getText();
            String id = candIdField.getText();
            String party = candPartyField.getText();
            String symbol = candSymbolField.getText();
            String assets = candAssetsField.getText();
            String crim = candCrimField.getText();
            admin.addCandidate(name, place, id, party, symbol, assets, crim);
            showInteractiveDialog(candidateTab, "Candidate information updated interactively!",
"Success",
                    JOptionPane.INFORMATION_MESSAGE);
        });

        JButton enterCandidateListButton = new JButton("Candidate List: ");
        styleButton(enterCandidateListButton);
        enterCandidateListButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridy = 9;
        gbc.gridx = 0;
        gbc.gridwidth = 2;
        candidateTab.add(enterCandidateListButton, gbc);
```

```
enterCandidateListButton.addActionListener(e -> {
   String list = admin.getCandidateList();
   showInteractiveDialog(candidateTab, "Candidate List:\n" + list, "Candidates",
         JOptionPane.INFORMATION_MESSAGE);
});

tabbedPane.addTab("Candidate Management", candidateTab);

// Other admin tabs (Voter Management, Campaign, Election Control, Results)
// Tab 2: Voter Management
JPanel voterTab = new GradientPanel();
voterTab.setLayout(new GridBagLayout());
gbc = new GridBagConstraints();
gbc.insets = new Insets(10, 10, 10, 10);
gbc.fill = GridBagConstraints.HORIZONTAL;

JLabel voterHeader = new JLabel("Voter Management");
voterHeader.setFont(new Font("SansSerif", Font.BOLD, 28));
voterHeader.setForeground(new Color(204, 0, 102));
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 2;
voterTab.add(voterHeader, gbc);

gbc.gridwidth = 1;
gbc.gridy = 1;
gbc.gridx = 0;
voterTab.add(new JLabel("Name:"), gbc);
JTextField voterNameField = new JTextField(15);
voterNameField.setFont(new Font("SansSerif", Font.PLAIN, 20));
gbc.gridx = 1;
voterTab.add(voterNameField, gbc);

gbc.gridy = 2;
gbc.gridx = 0;
voterTab.add(new JLabel("Voter ID:"), gbc);
JTextField voterIdField = new JTextField(15);
voterIdField.setFont(new Font("SansSerif", Font.PLAIN, 20));
gbc.gridx = 1;
voterTab.add(voterIdField, gbc);

JButton addVoterButton = new JButton("Add Voter");
styleButton(addVoterButton);
addVoterButton.setFont(new Font("SansSerif", Font.BOLD, 22));
gbc.gridy = 3;
gbc.gridx = 0;
gbc.gridwidth = 2;
voterTab.add(addVoterButton, gbc);
```

```
        addVoterButton.addActionListener(e -> {
            String name = voterNameField.getText();
            String id = voterIdField.getText();
            admin.addVoter(name, id);
            showInteractiveDialog(voterTab, "Voter added interactively!", "Success",
JOptionPane.INFORMATION_MESSAGE);
        });

        JButton enterVoterListButton = new JButton("Registered Voters");
        styleButton(enterVoterListButton);
        enterVoterListButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridy = 4;
        gbc.gridx = 0;
        gbc.gridwidth = 2;
        voterTab.add(enterVoterListButton, gbc);
        enterVoterListButton.addActionListener(e -> {
            String list = admin.getVoterList();
            showInteractiveDialog(voterTab, "Registered Voters:\n" + list, "Voter List",
                    JOptionPane.INFORMATION_MESSAGE);
        });

        tabbedPane.addTab("Voter Management", voterTab);

        // Tab 3: Campaign Management
        JPanel campaignTab = new GradientPanel();
        campaignTab.setLayout(new GridBagLayout());
        gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JLabel campaignHeader = new JLabel("Election Campaign Management");
        campaignHeader.setFont(new Font("SansSerif", Font.BOLD, 28));
        campaignHeader.setForeground(new Color(0, 153, 0));
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 2;
        campaignTab.add(campaignHeader, gbc);

        JButton startCampaignButton = new JButton("Start Campaign");
        styleButton(startCampaignButton);
        startCampaignButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridy = 1;
        gbc.gridx = 0;
        campaignTab.add(startCampaignButton, gbc);
        startCampaignButton.addActionListener(e -> {
            if (!campaignRunning) {
                campaignRunning = true;
                admin.startCampaign();
```

```
            showInteractiveDialog(campaignTab, "Campaign started interactively!", "Success",
                JOptionPane.INFORMATION_MESSAGE);
        } else {
            showInteractiveDialog(campaignTab, "Campaign already running!", "Info",
JOptionPane.WARNING_MESSAGE);
        }
    });

    JButton endCampaignButton = new JButton("End Campaign");
    styleButton(endCampaignButton);
    endCampaignButton.setFont(new Font("SansSerif", Font.BOLD, 22));
    gbc.gridy = 2;
    gbc.gridx = 0;
    campaignTab.add(endCampaignButton, gbc);
    endCampaignButton.addActionListener(e -> {
        if (campaignRunning) {
            campaignRunning = false;
            admin.endCampaign();
            showInteractiveDialog(campaignTab, "Campaign ended interactively!", "Success",
                JOptionPane.INFORMATION_MESSAGE);
        } else {
            showInteractiveDialog(campaignTab, "No campaign is running!", "Info",
JOptionPane.WARNING_MESSAGE);
        }
    });

    tabbedPane.addTab("Campaign Management", campaignTab);

    // Tab 4: Election Control
    JPanel electionTab = new GradientPanel();
    electionTab.setLayout(new GridBagLayout());
    gbc = new GridBagConstraints();
    gbc.insets = new Insets(10, 10, 10, 10);
    gbc.fill = GridBagConstraints.HORIZONTAL;

    JLabel electionHeader = new JLabel("Election Starting/Ending");
    electionHeader.setFont(new Font("SansSerif", Font.BOLD, 28));
    electionHeader.setForeground(new Color(204, 102, 0));
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.gridwidth = 2;
    electionTab.add(electionHeader, gbc);

    gbc.gridwidth = 1;
    gbc.gridy = 1;
    gbc.gridx = 0;
    electionTab.add(new JLabel("Duration (sec):"), gbc);
    JTextField durationField = new JTextField(10);
```

```
        durationField.setFont(new Font("SansSerif", Font.PLAIN, 20));
        gbc.gridx = 1;
        electionTab.add(durationField, gbc);

        JButton startElectionButton = new JButton("Start Election");
        styleButton(startElectionButton);
        startElectionButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridy = 2;
        gbc.gridx = 0;
        electionTab.add(startElectionButton, gbc);
        startElectionButton.addActionListener(e -> {
            try {
                int duration = Integer.parseInt(durationField.getText());
                admin.startElection(duration);
                electionRunning = true;
                showInteractiveDialog(electionTab, "Election started for " + duration + " seconds
interactively!",
                        "Success", JOptionPane.INFORMATION_MESSAGE);
            } catch (NumberFormatException nfe) {
                showInteractiveDialog(electionTab, "Invalid duration entered!", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        });

        JButton endElectionButton = new JButton("End Election");
        styleButton(endElectionButton);
        endElectionButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridx = 1;
        electionTab.add(endElectionButton, gbc);
        endElectionButton.addActionListener(e -> {
            if (electionRunning) {
                admin.endElection();
                electionRunning = false;
                showInteractiveDialog(electionTab, "Election ended interactively!", "Success",
                        JOptionPane.INFORMATION_MESSAGE);
            } else {
                showInteractiveDialog(electionTab, "Election is not running!", "Info",
JOptionPane.WARNING_MESSAGE);
            }
        });

        tabbedPane.addTab("Election Control", electionTab);

        // Tab 5: Results
        JPanel resultsTab = new GradientPanel();
        resultsTab.setLayout(new GridBagLayout());
        gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);
```

```
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JButton winnerButton = new JButton("Show Winner");
        styleButton(winnerButton);
        winnerButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridx = 0;
        gbc.gridy = 0;
        resultsTab.add(winnerButton, gbc);
        winnerButton.addActionListener(e -> {
          if(electionRunning){showInteractiveDialog(resultsTab, "Election not ended yet!..." ,
                "Info", JOptionPane.INFORMATION_MESSAGE);
          }
          else { String result = admin.getElectionWinner();
            showInteractiveDialog(resultsTab, "Election Winner:\n" + result, "Winner",
JOptionPane.INFORMATION_MESSAGE);
          }});

        JButton actualResultsButton = new JButton("Actual Results");
        styleButton(actualResultsButton);
        actualResultsButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridx = 1;
        resultsTab.add(actualResultsButton, gbc);
        actualResultsButton.addActionListener(e -> {
          if(electionRunning){
            showInteractiveDialog(resultsTab, "Election not ended yet!..." ,
                  "Info", JOptionPane.INFORMATION_MESSAGE);
          }
          else{String results = admin.getVoteCount();
            showInteractiveDialog(resultsTab, "Election Results:\n" + results, "Results",
                  JOptionPane.INFORMATION_MESSAGE);
          }});

        JButton liveVotesButton = new JButton("Live Total Votes");
        styleButton(liveVotesButton);
        liveVotesButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridwidth = 2;
        resultsTab.add(liveVotesButton, gbc);
        liveVotesButton.addActionListener(e -> {
          int total = admin.getTotalVotes();
          showInteractiveDialog(resultsTab, "Total Votes Casted: " + total, "Live Votes",
                JOptionPane.INFORMATION_MESSAGE);
        });

        tabbedPane.addTab("Results", resultsTab);

        // would be added similarly.
```

```
    // Back button:
    JPanel adminBackPanel = new JPanel();
    adminBackPanel.setOpaque(false);
    JButton adminBackButton = new JButton("Back");
    styleButton(adminBackButton);
    adminBackButton.setFont(new Font("SansSerif", Font.BOLD, 22));
    adminBackPanel.add(adminBackButton);
    adminBackButton.addActionListener(e -> cardLayout.show(mainPanel, "RoleSelection"));

    panel.add(tabbedPane, BorderLayout.CENTER);
    panel.add(adminBackPanel, BorderLayout.SOUTH);
    return panel;
}

// ---------------------- Create Candidate Login Panel ----------------------
private static JPanel createCandidateLoginPanel() {
    GradientPanel panel = new GradientPanel();
    panel.setLayout(new GridBagLayout());
    GridBagConstraints gbc = new GridBagConstraints();
    gbc.insets = new Insets(15, 15, 15, 15);
    gbc.fill = GridBagConstraints.HORIZONTAL;

    JLabel titleLabel = new JLabel("Candidate Login");
    titleLabel.setFont(new Font("SansSerif", Font.BOLD, 32));
    titleLabel.setForeground(new Color(25, 25, 112));
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.gridwidth = 2;
    panel.add(titleLabel, gbc);

    gbc.gridwidth = 1;
    gbc.gridy = 1;
    gbc.gridx = 0;
    panel.add(new JLabel("Full Name:"), gbc);
    JTextField nameField = new JTextField(20);
    nameField.setFont(new Font("SansSerif", Font.PLAIN, 22));
    gbc.gridx = 1;
    panel.add(nameField, gbc);

    gbc.gridy = 2;
    gbc.gridx = 0;
    panel.add(new JLabel("Candidate ID:"), gbc);
    JTextField idField = new JTextField(20);
    idField.setFont(new Font("SansSerif", Font.PLAIN, 22));
    gbc.gridx = 1;
    panel.add(idField, gbc);

    JButton loginButton = new JButton("Login");
```

```
        styleButton(loginButton);
        loginButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridy = 3;
        gbc.gridx = 0;
        gbc.gridwidth = 2;
        panel.add(loginButton, gbc);

        loginButton.addActionListener(e -> {
            String name = nameField.getText();
            String id = idField.getText();
            Candidate candidate = admin.findCandidate(name, id);
            if (candidate != null) {
                currentCandidate = candidate;
                showInteractiveDialog(panel, "Login Successful!", "Success",
JOptionPane.INFORMATION_MESSAGE);
                nameField.setText("");
                idField.setText("");
                candidatePanel = createCandidatePanel();
                mainPanel.add(candidatePanel, "CandidatePanel");
                cardLayout.show(mainPanel, "CandidatePanel");
            } else {
                showInteractiveDialog(panel, "Invalid candidate credentials!", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        });

        JButton backButton = new JButton("Back");
        styleButton(backButton);
        gbc.gridy = 4;
        gbc.gridx = 0;
        gbc.gridwidth = 2;
        panel.add(backButton, gbc);
        backButton.addActionListener(e -> {
            nameField.setText("");
            idField.setText("");
            cardLayout.show(mainPanel, "RoleSelection");
        });

        return panel;
    }

    // ---------------------- Create Candidate Panel ----------------------
    private static JPanel createCandidatePanel() {
        GradientPanel panel = new GradientPanel();
        panel.setLayout(new BorderLayout());

        JTabbedPane candidateTabbedPane = new JTabbedPane();
        candidateTabbedPane.setFont(new Font("SansSerif", Font.BOLD, 20));
```

```
        // Tab: Profile
        JPanel profileTab = new GradientPanel();
        profileTab.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(15, 15, 15, 15);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JLabel profileLabel = new JLabel("Candidate Profile");
        profileLabel.setFont(new Font("SansSerif", Font.BOLD, 32));
        profileLabel.setForeground(new Color(25, 25, 112));
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 2;
        profileTab.add(profileLabel, gbc);

        gbc.gridwidth = 1;
        gbc.gridy = 1;
        gbc.gridx = 0;
        profileTab.add(new JLabel("Your Information:"), gbc);
        JTextArea candidateInfoArea = new JTextArea(4, 20);
        candidateInfoArea.setEditable(false);
        candidateInfoArea.setFont(new Font("SansSerif", Font.PLAIN, 22));
        candidateInfoArea.setBackground(new Color(255, 255, 255, 200));
        candidateInfoArea.setText("Name: " + currentCandidate.getName() + "\nID: " +
currentCandidate.getId() +
                "\nParty: " + currentCandidate.getParty());
        gbc.gridx = 1;
        profileTab.add(candidateInfoArea, gbc);

        gbc.gridy = 2;
        gbc.gridx = 0;
        profileTab.add(new JLabel("Your Manifesto:"), gbc);
        JTextArea manifestoArea = new JTextArea(6, 20);
        manifestoArea.setFont(new Font("SansSerif", Font.PLAIN, 22));
        manifestoArea.setLineWrap(true);
        manifestoArea.setWrapStyleWord(true);
        manifestoArea.setText(currentCandidate.getManifesto());
        gbc.gridx = 1;
        profileTab.add(new JScrollPane(manifestoArea), gbc);

        JButton updateManifestoButton = new JButton("Update Manifesto");
        styleButton(updateManifestoButton);
        updateManifestoButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridy = 3;
        gbc.gridx = 0;
        gbc.gridwidth = 2;
        profileTab.add(updateManifestoButton, gbc);
```

```
        updateManifestoButton.addActionListener(e -> {
            if (campaignRunning) {
                String newManifesto = manifestoArea.getText();
                currentCandidate.setManifesto(newManifesto);
                showInteractiveDialog(profileTab, "Manifesto updated successfully!", "Success",
                        JOptionPane.INFORMATION_MESSAGE);
            } else {
                showInteractiveDialog(profileTab, "Campaign is not running. Cannot update manifesto.",
"Error",
                        JOptionPane.ERROR_MESSAGE);
            }
        });

        JButton profileBackButton = new JButton("Back");
        styleButton(profileBackButton);
        profileBackButton.setFont(new Font("SansSerif", Font.BOLD, 22));
        gbc.gridy = 4;
        gbc.gridx = 0;
        gbc.gridwidth = 2;
        profileTab.add(profileBackButton, gbc);
        profileBackButton.addActionListener(e -> cardLayout.show(mainPanel, "CandidateLogin"));

        candidateTabbedPane.addTab("Profile", profileTab);

        // Tab: Results
        JPanel candidateResultsTab = new GradientPanel();
        candidateResultsTab.setLayout(new GridBagLayout());
        gbc = new GridBagConstraints();
        gbc.insets = new Insets(15, 15, 15, 15);
        gbc.fill = GridBagConstraints.HORIZONTAL;

        JButton resultsButton = new JButton("Results");
        styleButton(resultsButton);
        resultsButton.setFont(new Font("SansSerif", Font.BOLD, 28));
        gbc.gridx = 0;
        gbc.gridy = 0;
        candidateResultsTab.add(resultsButton, gbc);
        resultsButton.addActionListener(e -> {
            if(electionRunning) showInteractiveDialog(candidateResultsTab, "Election not ended yet!..." ,
                    "Info", JOptionPane.INFORMATION_MESSAGE);
            else {String results = admin.getVoteCount();
                showInteractiveDialog(candidateResultsTab, "Election Results:\n" + results, "Results",
                        JOptionPane.INFORMATION_MESSAGE);
            }});

        JButton winnerButton = new JButton("Show Winner");
        styleButton(winnerButton);
        winnerButton.setFont(new Font("SansSerif", Font.BOLD, 28));
```

```
    gbc.gridx = 0;
    gbc.gridy = 1;
    candidateResultsTab.add(winnerButton, gbc);
    winnerButton.addActionListener(e -> {
        if(electionRunning) {
            showInteractiveDialog(candidateResultsTab, "Election not ended yet!..." ,
                    "Info", JOptionPane.INFORMATION_MESSAGE);
        }
        else{String result = admin.getElectionWinner();
            showInteractiveDialog(candidateResultsTab, "Election Winner:\n" + result, "Winner",
                    JOptionPane.INFORMATION_MESSAGE);
        }});

    JButton liveVotesButton = new JButton("Live Total Votes");
    styleButton(liveVotesButton);
    liveVotesButton.setFont(new Font("SansSerif", Font.BOLD, 28));
    gbc.gridy = 2;
    candidateResultsTab.add(liveVotesButton, gbc);
    liveVotesButton.addActionListener(e -> {
        int total = admin.getTotalVotes();
        showInteractiveDialog(candidateResultsTab, "Total Votes Casted: " + total, "Live Votes",
                JOptionPane.INFORMATION_MESSAGE);
    });

    candidateTabbedPane.addTab("Results", candidateResultsTab);

    panel.add(candidateTabbedPane, BorderLayout.CENTER);
    return panel;
}

// ---------------------- Create Voter Panel with Interactive Tabs
// ----------------------
private static JPanel createVoterPanel() {
    GradientPanel panel = new GradientPanel();
    panel.setLayout(new BorderLayout());

    JLabel titleLabel = new JLabel("Voter Portal", SwingConstants.CENTER);
    titleLabel.setFont(new Font("SansSerif", Font.BOLD, 32));
    titleLabel.setForeground(new Color(25, 25, 112));
    panel.add(titleLabel, BorderLayout.NORTH);

    // Use static voterTabbedPane to modify it later
    voterTabbedPane = new JTabbedPane();
    voterTabbedPane.setFont(new Font("SansSerif", Font.BOLD, 20));

    // Tab: Voter Login (light green background)
    JPanel loginTab = new JPanel(new GridBagLayout());
    loginTab.setBackground(new Color(200, 255, 200));
```

```
GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(15, 15, 15, 15);
gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.gridx = 0;
gbc.gridy = 0;
loginTab.add(new JLabel("Enter Voter Name:"), gbc);
JTextField voterNameField = new JTextField(20);
voterNameField.setFont(new Font("SansSerif", Font.PLAIN, 22));
gbc.gridx = 1;
loginTab.add(voterNameField, gbc);

gbc.gridx = 0;
gbc.gridy = 1;
loginTab.add(new JLabel("Enter Voter ID:"), gbc);
JTextField voterIdField = new JTextField(20);
voterIdField.setFont(new Font("SansSerif", Font.PLAIN, 22));
gbc.gridx = 1;
loginTab.add(voterIdField, gbc);

JButton loginButton = new JButton("Login");
styleButton(loginButton);
loginButton.setFont(new Font("SansSerif", Font.BOLD, 22));
gbc.gridx = 0;
gbc.gridy = 2;
gbc.gridwidth = 2;
loginTab.add(loginButton, gbc);

JLabel loginStatusLabel = new JLabel("", SwingConstants.CENTER);
loginStatusLabel.setFont(new Font("SansSerif", Font.BOLD, 22));
gbc.gridy = 3;
loginTab.add(loginStatusLabel, gbc);

loginButton.addActionListener(e -> {
   String name = voterNameField.getText();
   String id = voterIdField.getText();
   Voter voter = admin.findVoter(name, id);
   if (voter != null) {
      if (voter.hasVoted()) {
         loginStatusLabel.setText("You have already voted.");
      } else {
         currentVoter = voter;
         loginStatusLabel.setText("Login Successful! Proceed to vote.");
         voterTabbedPane.setEnabledAt(1, true);
         voterTabbedPane.setSelectedIndex(1);
      }
   } else {
      loginStatusLabel.setText("Invalid voter credentials.");
```

```
        }
    });

    JButton loginBackButton = new JButton("Back");
    styleButton(loginBackButton);
    gbc.gridy = 4;
    loginTab.add(loginBackButton, gbc);
    loginBackButton.addActionListener(e -> {
        voterNameField.setText("");
        voterIdField.setText("");
        if (voterTabbedPane != null) {
            voterTabbedPane.setEnabledAt(1, false);
        }
        cardLayout.show(mainPanel, "RoleSelection");
    });

    voterTabbedPane.addTab("Login", loginTab);

    // Tab: Vote (light orange background) – navigates to Vote Page
    JPanel voteTab = new JPanel(new GridBagLayout());
    voteTab.setBackground(new Color(255, 230, 200));
    gbc = new GridBagConstraints();
    gbc.insets = new Insets(15, 15, 15, 15);
    gbc.fill = GridBagConstraints.HORIZONTAL;

    JButton voteNowButton = new JButton("VOTE NOW");
    styleButton(voteNowButton);
    voteNowButton.setFont(new Font("SansSerif", Font.BOLD, 28));
    gbc.gridx = 0;
    gbc.gridy = 0;
    voteTab.add(voteNowButton, gbc);
    voteNowButton.addActionListener(e -> {
        if (currentVoter == null) {
            showInteractiveDialog(voteTab, "Please login first.", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }
        if (currentVoter.hasVoted()) {
            showInteractiveDialog(voteTab, "You have already voted.", "Info",
JOptionPane.WARNING_MESSAGE);
            return;
        }
        if (!electionRunning) {
            showInteractiveDialog(voteTab, "Voting has not yet started.", "Info",
JOptionPane.INFORMATION_MESSAGE);
            return;
        }
        if (votePagePanel != null) {
```

```
        mainPanel.remove(votePagePanel);
    }
    votePagePanel = createVotePagePanel();
    mainPanel.add(votePagePanel, "VotePage");
    cardLayout.show(mainPanel, "VotePage");
});

gbc.gridy = 1;
JButton viewManifestoButton = new JButton("View Manifestos");
styleButton(viewManifestoButton);
viewManifestoButton.setFont(new Font("SansSerif", Font.BOLD, 22));
voteTab.add(viewManifestoButton, gbc);
viewManifestoButton.addActionListener(e -> {
    String manifestos = admin.getCandidateManifestos();
    showInteractiveDialog(voteTab, "Candidate Manifestos:\n" + manifestos, "Manifestos",
            JOptionPane.INFORMATION_MESSAGE);
});

gbc.gridy = 2;
JButton viewCandidateDetailsButton = new JButton("View Candidate Details");
styleButton(viewCandidateDetailsButton);
viewCandidateDetailsButton.setFont(new Font("SansSerif", Font.BOLD, 22));
voteTab.add(viewCandidateDetailsButton, gbc);
viewCandidateDetailsButton.addActionListener(e -> {
    String details = admin.getCandidateDetails();
    showInteractiveDialog(voteTab, "Candidate Details:\n" + details, "Details",
            JOptionPane.INFORMATION_MESSAGE);
});

voterTabbedPane.addTab("Vote", voteTab);
voterTabbedPane.setEnabledAt(1, false);

// Tab: Results
JPanel resultsTab = new JPanel(new GridBagLayout());
resultsTab.setBackground(new Color(220, 220, 255));
gbc = new GridBagConstraints();
gbc.insets = new Insets(15, 15, 15, 15);
gbc.fill = GridBagConstraints.HORIZONTAL;

JButton winnerButton = new JButton("Show Winner");
styleButton(winnerButton);
winnerButton.setFont(new Font("SansSerif", Font.BOLD, 28));
gbc.gridx = 0;
gbc.gridy = 0;
resultsTab.add(winnerButton, gbc);
winnerButton.addActionListener(e -> {
    if(electionRunning) showInteractiveDialog(resultsTab, "Election not ended yet!..." ,
            "Info", JOptionPane.INFORMATION_MESSAGE);
```

```
        else{
            String result = admin.getElectionWinner();
            showInteractiveDialog(resultsTab, "Election Winner:\n" + result, "Winner",
JOptionPane.INFORMATION_MESSAGE);
        }});

    JButton resultsButton = new JButton("Show Election Results");
    styleButton(resultsButton);
    resultsButton.setFont(new Font("SansSerif", Font.BOLD, 28));
    gbc.gridx = 0;
    gbc.gridy = 1;
    resultsTab.add(resultsButton, gbc);
    resultsButton.addActionListener(e -> {
      if (electionRunning) {
        showInteractiveDialog(resultsTab, "Election not ended yet!..." ,
            "Info", JOptionPane.INFORMATION_MESSAGE);
      } else {
        showInteractiveDialog(resultsTab, "Final Election Results:\n" + admin.getVoteCount(),
"Results",
            JOptionPane.INFORMATION_MESSAGE);
      }
    });

    voterTabbedPane.addTab("Results", resultsTab);

    panel.add(voterTabbedPane, BorderLayout.CENTER);
    return panel;
  }

  // ---------------------- Create Vote Page Panel ----------------------
  private static JPanel createVotePagePanel() {
    GradientPanel votePage = new GradientPanel();
    votePage.setLayout(new BorderLayout());

    JLabel voteHeader = new JLabel("Vote Now", SwingConstants.CENTER);
    voteHeader.setFont(new Font("SansSerif", Font.BOLD, 36));
    voteHeader.setForeground(new Color(25, 25, 112));
    votePage.add(voteHeader, BorderLayout.NORTH);

    // Get candidate summary using "|" as delimiter
    String[] candidateSummary = admin.getCandidateSummary();
    JList<String> candidateList = new JList<>(candidateSummary);
    candidateList.setFont(new Font("SansSerif", Font.PLAIN, 24));
    candidateList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    candidateList.setFixedCellHeight(40);
    candidateList.setFixedCellWidth(300);
    JScrollPane listScrollPane = new JScrollPane(candidateList);
    votePage.add(listScrollPane, BorderLayout.CENTER);
```

```
        JPanel bottomPanel = new JPanel();
        bottomPanel.setOpaque(false);
        JButton castVoteButton = new JButton("Cast Vote");
        styleButton(castVoteButton);
        castVoteButton.setFont(new Font("SansSerif", Font.BOLD, 26));
        bottomPanel.add(castVoteButton);

        JButton votePageBackButton = new JButton("Back");
        styleButton(votePageBackButton);
        votePageBackButton.setFont(new Font("SansSerif", Font.BOLD, 26));
        bottomPanel.add(votePageBackButton);

        castVoteButton.addActionListener(e -> {
            if (currentVoter == null) {
                showInteractiveDialog(votePage, "Please login first.", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }
            if (currentVoter.hasVoted()) {
                showInteractiveDialog(votePage, "You have already voted.", "Info",
JOptionPane.WARNING_MESSAGE);
                return;
            }
            if (!electionRunning) {
                showInteractiveDialog(votePage, "Voting has not yet started.", "Info",
JOptionPane.INFORMATION_MESSAGE);
                return;
            }
            String selected = candidateList.getSelectedValue();
            if (selected == null || selected.isEmpty()) {
                showInteractiveDialog(votePage, "Please select a candidate.", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }
            // Use "|" as the delimiter and trim spaces to extract the candidate's name.
            String candidateName = selected.split("\\|")[0].trim();
            if (!currentVoter.hasVoted()) {
                admin.castVote(currentVoter, candidateName);
                showInteractiveDialog(votePage, "Vote cast successfully!", "Success",
JOptionPane.INFORMATION_MESSAGE);
            }
        });

        votePageBackButton.addActionListener(e -> {
            cardLayout.show(mainPanel, "VoterPanel");
            if (voterTabbedPane != null) {
                voterTabbedPane.setEnabledAt(1, false);
```

```
        }
    });

    votePage.add(bottomPanel, BorderLayout.SOUTH);
    return votePage;
}

// ---------------------- Helper Methods for Styling ----------------------
private static void styleButton(JButton button) {
    Color normalColor = new Color(70, 130, 180); // steel blue
    Color hoverColor = new Color(100, 149, 237); // cornflower blue
    button.setBackground(normalColor);
    button.setForeground(Color.WHITE);
    button.setFont(new Font("SansSerif", Font.BOLD, 16));
    button.setFocusPainted(false);
    button.setUI(new BasicButtonUI());
    button.addMouseListener(new MouseAdapter() {
        public void mouseEntered(MouseEvent e) {
            button.setBackground(hoverColor);
        }

        public void mouseExited(MouseEvent e) {
            button.setBackground(normalColor);
        }
    });
}

private static void styleLabel(JLabel label) {
    label.setFont(new Font("SansSerif", Font.BOLD, 24));
    label.setForeground(new Color(25, 25, 112));
}

// ---------------------- Custom Gradient Panel ----------------------
static class GradientPanel extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;
        int width = getWidth();
        int height = getHeight();
        Color startColor = new Color(173, 216, 230); // light blue
        Color endColor = new Color(224, 255, 255); // light cyan
        GradientPaint gp = new GradientPaint(0, 0, startColor, width, height, endColor);
        g2d.setPaint(gp);
        g2d.fillRect(0, 0, width, height);
    }
}
```

```
// ----------------------- Supporting Classes -----------------------
static class Admin {
    private List<Candidate> candidates;
    private List<Voter> voters;

    public Admin() {
        this.candidates = FileHandler.readCandidates();
        this.voters = FileHandler.readVoters();
    }

    public int getTotalVotes() {
        int total = 0;
        for (Candidate candidate : candidates) {
            total += candidate.getVotes();
        }
        return total;
    }

    public void addCandidate(String name, String place, String id, String party, String symbol, String
assets,
                             String criminalCases) {
        Candidate candidate = new Candidate(name, place, id, party, symbol, assets, criminalCases);
        boolean exists = false;
        for (int i = 0; i < candidates.size(); i++) {
            Candidate c = candidates.get(i);
            if (c.getId().equals(id)) {
                candidates.set(i, candidate);
                exists = true;
                break;
            }
        }
        if (!exists) {
            candidates.add(candidate);
        }
        FileHandler.writeCandidates(candidates);
    }

    public void addVoter(String name, String id) {
        Voter voter = new Voter(name, id);
        voters.add(voter);
        FileHandler.writeVoters(voters);
    }

    public void startCampaign() {
        System.out.println("Campaign started.");
    }

    public void endCampaign() {
```

```
        System.out.println("Campaign ended.");
    }

    public void startElection(int durationInSeconds) {
        VoteCounter voteCounter = new VoteCounter(candidates);
        voteCounter.start();
        ElectionTimer timer = new ElectionTimer(durationInSeconds, this);
        timer.start();
    }

    public void endElection() {
        Candidate winner =
candidates.stream().max(Comparator.comparingInt(Candidate::getVotes)).orElse(null);
        if (winner != null) {
            JOptionPane.showMessageDialog(null, "Election Ended. Winner: " + winner.getName());
        } else {
            JOptionPane.showMessageDialog(null, "Election Ended. No winner.");
        }
        FileHandler.writeCandidates(candidates);
    }

    public String getVoterList() {
        StringBuilder sb = new StringBuilder();
        for (Voter v : voters) {
            sb.append(v.getName()).append(" (ID: ").append(v.getId()).append(")\n");
        }
        return sb.toString();
    }

    public String getVoteCount() {
        StringBuilder sb = new StringBuilder();
        for (Candidate c : candidates) {
            sb.append(c.getName()).append(" : ").append(c.getVotes()).append(" votes\n");
        }
        return sb.toString();
    }

    public String getCandidateList() {
        StringBuilder sb = new StringBuilder();
        for (Candidate c : candidates) {
            sb.append(c.getName()).append("\n");
        }
        return sb.toString();
    }

    // Returns candidate summary using a pipe delimiter.
    public String[] getCandidateSummary() {
        String[] summary = new String[candidates.size()];
```

```
    for (int i = 0; i < candidates.size(); i++) {
        Candidate c = candidates.get(i);
        summary[i] = c.getName() + " | " + c.getParty() + " | " + c.getSymbol();
    }
    return summary;
}

public String getCandidateManifestos() {
    StringBuilder sb = new StringBuilder();
    for (Candidate c : candidates) {
        sb.append(c.getName()).append(":\n").append(c.getManifesto()).append("\n\n");
    }
    return sb.toString();
}

public String getCandidateDetails() {
    StringBuilder sb = new StringBuilder();
    for (Candidate c : candidates) {
        sb.append("Name: ").append(c.getName())
                .append(", Place: ").append(c.getPlace())
                .append(", Party: ").append(c.getParty())
                .append(", Assets: ").append(c.getAssets())
                .append(", Criminal Cases: ").append(c.getCriminalCases())
                .append("\n");
    }
    return sb.toString();
}

public void castVote(Voter voter, String candidateName) {
    if (voter.hasVoted())
        return;
    for (Candidate c : candidates) {
        if (c.getName().equals(candidateName)) {
            c.incrementVote();
            voter.setVoted();
            FileHandler.writeCandidates(candidates);
            FileHandler.writeVoters(voters);
            break;
        }
    }
}

public Voter findVoter(String name, String id) {
    for (Voter v : voters) {
        if (v.getName().equals(name) && v.getId().equals(id)) {
            return v;
        }
    }
```

```
        return null;
    }

    public Candidate findCandidate(String name, String id) {
        for (Candidate c : candidates) {
            if (c.getName().equals(name) && c.getId().equals(id)) {
                return c;
            }
        }
        return null;
    }

    public String getElectionWinner() {
        if (candidates.isEmpty())
            return "No candidates available.";

        // Check if all candidates have 0 votes
        boolean votingNotStarted = candidates.stream().allMatch(c -> c.getVotes() == 0);
        if (votingNotStarted) {
            return "Voting has not yet started.";
        }

        candidates.sort((c1, c2) -> c2.getVotes() - c1.getVotes());
        Candidate winner = candidates.get(0);
        String winnerName = winner.getName();
        int winnerVotes = winner.getVotes();
        int runnerUpVotes = (candidates.size() > 1) ? candidates.get(1).getVotes() : 0;
        int voteDifference = winnerVotes - runnerUpVotes;
        return "Winner: " + winnerName + "\nVotes: " + winnerVotes + "\nVote Difference: " +
voteDifference;
    }
  }

  static class Candidate {
    private String name;
    private String place;
    private String id;
    private String party;
    private String symbol;
    private String assets;
    private String criminalCases;
    private String manifesto;
    private int votes;

    public Candidate(String name, String place, String id, String party, String symbol, String assets,
            String criminalCases) {
        this.name = name;
        this.place = place;
```

```
    this.id = id;
    this.party = party;
    this.symbol = symbol;
    this.assets = assets;
    this.criminalCases = criminalCases;
    this.manifesto = "No manifesto provided.";
    this.votes = 0;
}

public String getName() {
    return name;
}

public String getPlace() {
    return place;
}

public String getId() {
    return id;
}

public String getParty() {
    return party;
}

public String getSymbol() {
    return symbol;
}

public String getAssets() {
    return assets;
}

public String getCriminalCases() {
    return criminalCases;
}

public String getManifesto() {
    return manifesto;
}

public int getVotes() {
    return votes;
}

public void setManifesto(String manifesto) {
    this.manifesto = manifesto;
}
```

```
    public void incrementVote() {
      votes++;
    }

    public void setVotes(int votes) {
      this.votes = votes;
    }

    @Override
    public String toString() {
      return name + "," + place + "," + id + "," + party + "," + symbol + "," + assets + "," +
criminalCases + ","
          + manifesto + "," + votes;
    }
  }

  static class Voter {
    private String name;
    private String id;
    private boolean hasVoted;

    public Voter(String name, String id) {
      this.name = name;
      this.id = id;
      this.hasVoted = false;
    }

    public String getName() {
      return name;
    }

    public String getId() {
      return id;
    }

    public boolean hasVoted() {
      return hasVoted;
    }

    public void setVoted() {
      this.hasVoted = true;
    }

    @Override
    public String toString() {
      return name + "," + id + "," + hasVoted;
    }
```

```
    }

    static class FileHandler {
        // Using lowercase file names for consistency
        private static final String CANDIDATE_FILE = "candidate.txt";
        private static final String VOTER_FILE = "voter.txt";

        public static List<Candidate> readCandidates() {
            List<Candidate> candidates = new ArrayList<>();
            try (BufferedReader br = new BufferedReader(new FileReader(CANDIDATE_FILE))) {
                String line;
                while ((line = br.readLine()) != null) {
                    String[] parts = line.split(",");
                    if (parts.length < 9)
                        continue;
                    Candidate candidate = new Candidate(parts[0], parts[1], parts[2], parts[3], parts[4],
parts[5],
                            parts[6]);
                    candidate.setManifesto(parts[7]);
                    try {
                        int votes = Integer.parseInt(parts[8]);
                        candidate.setVotes(votes);
                    } catch (NumberFormatException nfe) {
                    }
                    candidates.add(candidate);
                }
            } catch (IOException e) {
                System.err.println("Error reading candidate file: " + e.getMessage());
            }
            return candidates;
        }

        public static List<Voter> readVoters() {
            List<Voter> voters = new ArrayList<>();
            try (BufferedReader br = new BufferedReader(new FileReader(VOTER_FILE))) {
                String line;
                while ((line = br.readLine()) != null) {
                    String[] parts = line.split(",");
                    if (parts.length < 3)
                        continue;
                    Voter voter = new Voter(parts[0], parts[1]);
                    if (parts[2].equals("true"))
                        voter.setVoted();
                    voters.add(voter);
                }
            } catch (IOException e) {
                System.err.println("Error reading voter file: " + e.getMessage());
            }
```

```
        return voters;
    }

    public static void writeCandidates(List<Candidate> candidates) {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(CANDIDATE_FILE))) {
            for (Candidate c : candidates) {
                bw.write(c.toString());
                bw.newLine();
            }
        } catch (IOException e) {
            System.err.println("Error writing candidate file: " + e.getMessage());
        }
    }

    public static void writeVoters(List<Voter> voters) {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(VOTER_FILE))) {
            for (Voter v : voters) {
                bw.write(v.toString());
                bw.newLine();
            }
        } catch (IOException e) {
            System.err.println("Error writing voter file: " + e.getMessage());
        }
    }
}

static class VoteCounter extends Thread {
    private final List<Candidate> candidates;

    public VoteCounter(List<Candidate> candidates) {
        this.candidates = candidates;
    }

    @Override
    public void run() {
        while (electionRunning) {
            System.out.println("Updating Vote Count...");
            for (Candidate c : candidates) {
                System.out.println(c.getName() + " - " + c.getVotes() + " votes");
            }
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                System.err.println("VoteCounter interrupted!");
                break;
            }
        }
    }
```

```
    }

    static class ElectionTimer extends Thread {
        private final int duration;
        private final Admin admin;

        public ElectionTimer(int duration, Admin admin) {
            this.duration = duration;
            this.admin = admin;
        }

        @Override
        public void run() {
            try {
                Thread.sleep(duration * 1000);
                admin.endElection();
                electionRunning = false;
            } catch (InterruptedException e) {
                System.err.println("ElectionTimer interrupted!");
            }
        }


    }
}
```

OOPs Concept in the Code:-

Below is a walkthrough of the core object-oriented principles illustrated in *FinalElectionApp.java*, with focused code snippets from your project. All citations refer to the uploaded file .

---

1. Classes & Objects

* *Definition*: A class is a blueprint; an object is an instance.

* *In Your Code*: You define multiple classes (FinalElectionApp, Admin, Candidate, Voter, FileHandler, VoteCounter, ElectionTimer), and create objects of them.

* *Example*:

```
java
// Creating the Admin singleton and later Candidate/Voter instances
private static Admin admin = new Admin();
private static Voter currentVoter;
private static Candidate currentCandidate;
…
Candidate candidate = new Candidate(name, place, id, party, symbol, assets, criminalCases);
Voter voter = new Voter(name, id);
```

---

2. Encapsulation

* *Definition*: Hiding internal state and requiring all interaction through methods.

* *In Your Code*: Use of private fields with public getters/setters in Candidate and Voter.

* *Example*:

```
java
public class Candidate {
    private String name;
    private String manifesto;
    private int votes;
    // …
    public String getName() { return name; }
    public String getManifesto() { return manifesto; }
    public void setManifesto(String manifesto) { this.manifesto = manifesto; }
    public void incrementVote() { votes++; }
}
```

---

3. Abstraction

* *Definition*: Exposing only relevant operations, hiding complex implementation.

* *In Your Code*: Admin offers high-level methods (addCandidate, castVote, getElectionWinner), while FileHandler encapsulates file I/O details.

* **Example (Admin abstraction):

```
java
public void castVote(Voter voter, String candidateName) {
    if (voter.hasVoted()) return;
    for (Candidate c : candidates) {
        if (c.getName().equals(candidateName)) {
            c.incrementVote();
```

```
            voter.setVoted();
            FileHandler.writeCandidates(candidates);
            FileHandler.writeVoters(voters);
            break;
        }
    }
}
```

---

4. Inheritance

* *Definition*: A class can inherit fields and methods from another.
* *In Your Code*: Both VoteCounter and ElectionTimer extend Thread, inheriting its behavior and overriding run().
* *Example*:

```
java
static class VoteCounter extends Thread {
    private final List<Candidate> candidates;
    @Override
    public void run() {
        while (electionRunning) {
            // periodically print vote counts…
        }
    }
}
```

```
static class ElectionTimer extends Thread {
    private final int duration;
    @Override
    public void run() {
        Thread.sleep(duration * 1000);
        admin.endElection();
        electionRunning = false;
    }
}
```

---

5. Polymorphism

* *Definition*: The same operation behaving differently on different classes (-overloading and -overriding).

1. *Method Overriding*: run() in both subclasses of Thread.
2. *Interface Polymorphism*: Swing event listeners (addActionListener) accept any ActionListener.

* *Example (Overriding)*:

```
java
@Override
public void run() {
    // in VoteCounter: print updates
}
@Override
public void run() {
```

```
    // in ElectionTimer: sleep then end election
}
```

---

6. Constructor Overloading

* *Definition*: Multiple constructors with different parameter lists.

* *In Your Code*: While most classes have single constructors, the GradientPanel subclasses JPanel and relies on the default no-arg constructor of its superclass. You also see multiple constructors in Swing (new JFrame(title), new JDialog(owner, title, modal)).

* *Example*:

```
java
// Different Swing constructors
frame = new JFrame("Interactive Election Management System");
JDialog dialog = new JDialog((Frame) null, "Admin Authentication", true);
```

---

7. Static Members

* *Definition*: Belong to the class rather than any instance.

* *In Your Code*: Many GUI components and state flags are declared static so they’re shared across the app.

* *Example*:

```
java
private static boolean electionRunning = false;
private static JFrame frame;
private static CardLayout cardLayout;
…
private static void styleButton(JButton button) { … }
```

---

8. Nested & Inner Classes

* *Definition*: Classes defined within another class.

* *In Your Code*: All supporting classes (Admin, Candidate, Voter, FileHandler, VoteCounter, ElectionTimer) and the custom GradientPanel are declared as static nested classes inside FinalElectionApp.

* *Example*:

```
java
static class GradientPanel extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        // custom gradient background…
    }
}
```

---

9. Interfaces

* *Definition*: A contract that classes can implement.
* *In Your Code*: You rely on Swing’s ActionListener interface via lambdas for button clicks.
* *Example*:

```
java
loginButton.addActionListener(e -> {
    // handle login…
});
```

---

10. Exception Handling (Bonus)

* *Definition*: Managing runtime errors via try–catch, throw, throws, and finally.
* *In Your Code*: Parsing durations and file I/O.
* *Example*:

```
java
try {
    int duration = Integer.parseInt(durationField.getText());
    admin.startElection(duration);
} catch (NumberFormatException nfe) {
    showInteractiveDialog(panel, "Invalid duration entered!", "Error",
JOptionPane.ERROR_MESSAGE);
```

}

Each of these points ties the textbook OOP concepts back to concrete code in *FinalElectionApp.java*, demonstrating how your project leverages Java’s object-oriented design.

GUI Components Explanation:-

Your Java Swing project is an interactive Election Management System with separate portals for Admin, Candidate, and Voter. Below is a detailed explanation of all major Java Swing GUI concepts used in your project, organized by category and component:

1. JFrame and JPanel – Base Containers

* JFrame frame: The main application window.

* JPanel mainPanel: Holds all different "screens" (login, admin panel, voter panel, etc.).

* CardLayout cardLayout: Allows switching between different screens in the main panel (like pages in an app).

2. Layout Managers

* BorderLayout: Used in panels like AdminPanel and CandidatePanel to divide the panel into North, Center, South, etc.

* GridBagLayout: Used for complex form layouts, providing flexibility in component placement using GridBagConstraints.

* CardLayout: Used for navigation between panels (pages) like "RoleSelection", "AdminPanel", etc.

3. Components (JComponents)

* JLabel: Display static text (e.g., "Election Management Simulation", form labels).

* JTextField: For user input (e.g., voter name, candidate ID).

* JPasswordField: For admin password entry.

* JTextArea: Used for multi-line text like manifestos and result displays.

* JButton: Action triggers (e.g., Login, Add Candidate, Cast Vote).

* JComboBox: Drop-down list for role selection.

* JTabbedPane: Tabbed panels used in Admin, Voter, and Candidate interfaces (e.g., "Candidate Management", "Results").

* JList: Used in the vote page to list candidates.

* JScrollPane: Enables scrolling for large content like JTextArea or JList.

4. Event Handling

* ActionListener: Used to handle button clicks with .addActionListener(e -> {...}).

* MouseAdapter: Used in styleButton() to change button color on hover (mouseEntered, mouseExited).

* Component: Parent passed into showInteractiveDialog() to position modal dialogs relative to the current UI.

5. Dialog Boxes

* JOptionPane: Displays messages (info, warnings, errors) using showMessageDialog().

* JDialog: Used for custom dialogs (e.g., Admin Password Authentication Dialog).

6. Custom GUI Enhancements

* GradientPanel: A custom JPanel subclass with overridden paintComponent to draw a gradient background for enhanced aesthetics.

* styleButton(): A helper function to standardize button appearance, colors, fonts, and hover behavior.

7. Data-Driven Interaction

* Text fields are used to collect user input and interact with the backend classes (Admin, Candidate, Voter).

* User input is validated and dynamic updates are shown using dialog boxes.

8. Conditional Navigation and Behavior

* CardLayout and boolean flags (electionRunning, campaignRunning) control panel transitions and UI restrictions.

* Voter tab "Vote" is disabled until successful login and election start.

* Conditional dialogs ensure voters can’t vote multiple times or outside the election window.

9. Threading and Timers (UI-Integrated)

* ElectionTimer: Ends election automatically after specified seconds.

* VoteCounter: Continuously updates vote count during the election (though its output is console-based).

10. File I/O (Backend Interaction)

While not part of Swing, classes like FileHandler support saving and reading candidates and voters, reflecting changes in the GUI.

Limitations:-

Data is not encrypted in my project because voters and candidate data is stored in candidate.txt and voter.txt file. But Its encryption can be secured by storing data in SQL file using DBMS concepts.