Computer System Engineering Department

Sukkur Institute of Business Administration University

PROJECT REPORT

(**VotingMachine**)

SUBMITTED BY:

NAME: Abdul Rehman

CMS ID: 133-22-0031

NAME: Muhammad Faizan

CMS ID: 133-22-0034

NAME: Muhammad Zuhaib

CMS ID: 133-22-0030

NAME: Khalid Hussain

CMS ID: 133-22-0011

Certificate

It is certified that We_**Abdul Rehman,Khalid,Zuhaib&Faizan**___________ a student of **BE-III** has carried out the necessary work of **OOP** as per course of studies prevailed at the Computer System Engineering Department, Sukkur Institute of Business Administration for **FALL-2023.**

Date: _______________ Instructor's Signature

ACKNOWLEDGMENTS

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout my project work in processing sucessfully. I would like to express our deep and sincere gratitude to my project supervisor, **ENGR DR –MUMTAZ ALI** Assistant Professor at Institute of Business Administration, Sukkur for giving me the opportunity to do OOP project and providing invaluable guidance throughout this project. His dynamism, vision, sincerity and motivation have deeply inspired us. He has taught us the methodology to carry out the project and to present the project works as clearly as possible. It was a great privilege and honor to work and study under his guidance. I am extremely grateful for what he has offered me. I would also like to thank him for his friendship, empathy, and great sense of humor. I am extending my heartfelt thanks to him for their acceptance and patience during the discussion I had with him on project work and its preparation. I am extremely grateful to our parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. I am very much thankful to our family for their love, understanding. prayers and continuing support to continue this project work. Also I express my thanks to for the keen interest shown to continue this successfully.

ABSTRACT

The "Voting Machine App" project aims to provide a user-friendly and efficient platform for conducting electronic voting processes. This application employs a graphical user interface (GUI) using Java's Swing library, offering an intuitive interface for voters to cast their ballots. The system accommodates five candidates, each associated with a unique name and symbol, enhancing the visual appeal and ease of identification. Through the utilization of object-oriented programming principles, the code is organized into classes, promoting modularity and maintainability. The app ensures a secure and streamlined voting experience, incorporating features such as real-time vote tracking, display of candidate information, and a dedicated results section. Moreover, the application imposes a limit of ten voters, implementing a check for the correct voter number input before accepting votes. The "Voting Machine App" project represents a harmonious blend of user interface design, object-oriented programming, and secure voting practices, contributing to the advancement of electronic voting systems.

TABLE OF CONTENTS

Introduction

The "Voting Machine App" project introduces a modern and efficient solution to the traditional voting process, leveraging the power of technology to enhance the democratic experience. In an era where digital advancements play a pivotal role in shaping societal systems, this application serves as a testament to the integration of technology into fundamental democratic practices. The project aims to address challenges associated with traditional voting methods by providing a user-friendly graphical interface that simplifies the voting process for users while ensuring the integrity and security of the electoral system.

Built using Java's Swing library, the Voting Machine App features a visually appealing and intuitive design, allowing users to cast their votes seamlessly. The application accommodates five candidates, each distinguished by a unique name and symbol, facilitating voter recognition and engagement. Employing object-oriented programming principles, the code is structured to enhance modularity, scalability, and maintainability, reflecting a commitment to best practices in software development.

One of the notable features of this project is the incorporation of a limit on the number of voters, ensuring a controlled and monitored voting environment. The application prompts users to input their voter number, adding an additional layer of authentication before accepting votes. This security measure contributes to the overall reliability and trustworthiness of the electoral process.

In summary, the "Voting Machine App" project emerges as a significant step towards modernizing and optimizing the voting experience, embracing technology to promote efficiency, transparency, and security in the democratic process. This introduction sets the stage for a deeper exploration of the project's design, functionality, and the potential impact it can have on the evolution of electronic voting systems.

Motivation:

The motivation behind the "Voting Machine App" project stems from a recognition of the evolving landscape of democratic processes and the imperative to embrace technology for more efficient, accessible, and secure voting experiences. Traditional voting systems often face challenges such as long queues, logistical constraints, and manual errors, highlighting the need for innovative solutions to enhance the electoral process.

In many parts of the world, technology has become an integral part of daily life, and its incorporation into democratic practices can offer numerous benefits. The motivation for this project lies in harnessing the power of digital tools to streamline the voting process, making it more user-friendly and accommodating for a diverse electorate. By developing a graphical user interface (GUI) using Java's Swing library, the project aims to provide a modern and visually engaging platform for voters, fostering greater participation and inclusivity.

Furthermore, the decision to limit the number of voters in the application is driven by a commitment to maintaining the integrity and security of the voting process. The project's motivation is rooted in addressing concerns related to voter authentication and ensuring that each vote is cast by an eligible participant.

Ultimately, the "Voting Machine App" project is motivated by a vision of contributing to the evolution of democratic practices, embracing technology as a catalyst for positive change. By offering an innovative, secure, and user-centric voting solution, the project aims to inspire confidence in the electoral process and pave the way for more advanced and efficient voting systems in the digital age..

Methodology:

The methodology of the "Voting Machine App" code involves the use of Java's Swing library to create a graphical user interface (GUI) for an electronic voting system. The application is designed to facilitate a secure and efficient voting experience, incorporating features such as voter authentication, candidate selection, and real-time result display. Here's an overview of the key components and methods used in the code:

1. **Candidate Class:**

   - Represents a candidate with attributes such as name, symbol, and votes.
   - Provides methods to retrieve candidate information and increment vote counts.

2. **VotingMachineGUI Class:**
   - Extends `JFrame` to create the main graphical interface for the voting machine.
   - Utilizes Swing components such as buttons, labels, text fields, and text areas to design the user interface.
   - Implements action listeners for vote buttons and the result button.

3. **initializeCandidates Method:**
   - Initializes a map of candidates with unique identifiers, names, and symbols.

4. **createGUI Method:**
   - Constructs the GUI layout, incorporating components for candidate buttons, voter ID input, and result display.
   - Utilizes labels, text fields, buttons, and text areas to enhance user interaction.
   - Adds styling elements such as font sizes and bold labels for better readability.

5. **VoteButtonListener Class:**
   - Implements the `ActionListener` interface for vote buttons.
   - Validates the entered voter ID, checks its validity, and ensures that a voter has not already cast a vote.
   - Invokes the `vote` method of the selected candidate and updates the voted status.

6. **ResultButtonListener Class:**
   - Implements the `ActionListener` interface for the result button.
   - Generates and displays the election result, including the total votes for each candidate and the overall winner.
   - Uses a scrollable text area within a dialog for an organized presentation of results.

7. **Main Method (ElectionCommisionOFPakistan Class):**
   - Initiates the Swing application by creating an instance of the `VotingMachineGUI` class.
   - Invokes the `setVisible` method to display the GUI.

FUNCTIONS

The provided Java code for the "Voting Machine App" uses a variety of functions and methods to implement the functionality of the application. Here is an overview of the key functions/methods used in the code:

1. **Candidate Class:**

   - `Candidate(String name, String symbol)`: Constructor for creating a Candidate object with a name, symbol, and initial vote count.

   - `getName()`: Getter method to retrieve the name of the candidate.

   - `getSymbol()`: Getter method to retrieve the symbol of the candidate.

   - `getVotes()`: Getter method to retrieve the current vote count of the candidate.

   - `vote()`: Method to increment the vote count when a vote is cast for the candidate.

2. **VotingMachineGUI Class:**

   - `initializeCandidates()`: Method to initialize the map of candidates with their unique identifiers, names, and symbols.

   - `createGUI()`: Method to construct the graphical user interface layout, including buttons, labels, and text areas.

   - `VoteButtonListener`: Inner class implementing the `ActionListener` interface for handling vote button clicks.

   - `ResultButtonListener`: Inner class implementing the `ActionListener` interface for handling the result button click.

3. **VoteButtonListener Class:**

   - `actionPerformed(ActionEvent e)`: Implementation of the `actionPerformed` method from the `ActionListener` interface. This method is triggered when a vote button is clicked, and it handles the logic for validating the voter ID, checking its validity, ensuring the voter has not already voted, and updating the vote count.

4. **ResultButtonListener Class:**

- `actionPerformed(ActionEvent e)`: Implementation of the `actionPerformed` method from the `ActionListener` interface. This method is triggered when the result button is clicked, and it handles the logic for generating and displaying the election results.

5. **getVoterIndex(int voterId)** (within `VoteButtonListener`):

- Helper method to retrieve the index of a voter ID in the array of valid voter IDs.

6. **Main Method (ElectionCommisionOFPakistan Class):**

- `main(String[] args)`: The main entry point of the program. It initiates the Swing application by creating an instance of the `VotingMachineGUI` class and setting it to be visible.

These functions collectively define the structure and behavior of the "Voting Machine App." The use of Swing components and event listeners enhances the interactivity and responsiveness of the graphical user interface. The object-oriented principles are evident in the design of the `Candidate` class, providing encapsulation and abstraction for candidate-related functionalities. The code overall reflects a modular and organized approach to implementing an electronic voting system.

Code for the Project

Software Requirements

1: BLUE J

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashMap;
import java.util.Map;

class Candidate {
    private String name;
    private String symbol;
    private int votes;

    public Candidate(String name, String symbol) {
        this.name = name;
        this.symbol = symbol;
        this.votes = 0;
    }

    public String getName() {
        return name;
    }

    public String getSymbol() {
        return symbol;
    }

    public int getVotes() {
        return votes;
    }

    public void vote() {
        votes++;
    }
}

class VotingMachineGUI extends JFrame {
    private Map<Integer, Candidate> candidates;
    private JTextArea votedListTextArea;
    private JTextField voterIdTextField;
    private final int[] validVoterIds = {11, 22, 33, 44, 55, 66, 77, 88, 99, 100};
    private boolean[] votedStatus;

    public VotingMachineGUI() {
        candidates = new HashMap<>();
        initializeCandidates();
        votedStatus = new boolean[validVoterIds.length];
        createGUI();
    }

    private void initializeCandidates() {
        candidates.put(1, new Candidate("Imran Khan", "Bat"));
        candidates.put(2, new Candidate("Nawaz Sharif", "Lion"));
        candidates.put(3, new Candidate("Bilawal Bhutto", "Axe"));
        candidates.put(4, new Candidate("Fazul Rehman", "Book"));
        candidates.put(5, new Candidate("Jahangir Tareen", "Eagle"));
    }

    private void createGUI() {
        setTitle("Voting Machine");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(600, 500);
        setLocationRelativeTo(null);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(8, 1));

        // Adding Election Commission Of Pakistan in Bold
        JLabel electionCommissionLabel = new JLabel("<html><b>The Election Commission Of
Pakistan</b></html>");
        electionCommissionLabel.setHorizontalAlignment(JLabel.CENTER);
        electionCommissionLabel.setFont(new Font("Arial", Font.BOLD, 20));
        panel.add(electionCommissionLabel);

        // Adding JTextField for Voter ID
        JLabel voterIdLabel = new JLabel("Voter ID:");
        voterIdLabel.setFont(new Font("Arial", Font.PLAIN, 16));
        voterIdTextField = new JTextField();
        voterIdTextField.setFont(new Font("Arial", Font.PLAIN, 16));
        panel.add(voterIdLabel);
        panel.add(voterIdTextField);

        for (Map.Entry<Integer, Candidate> entry : candidates.entrySet()) {
            Candidate candidate = entry.getValue();
            JButton button = new JButton(candidate.getName() + " (" + candidate.getSymbol() + ")");
            button.addActionListener(new VoteButtonListener(entry.getKey()));
            button.setFont(new Font("Arial", Font.PLAIN, 16));
            panel.add(button);
        }

        JButton resultButton = new JButton("Show Result");
        resultButton.addActionListener(new ResultButtonListener());
        resultButton.setFont(new Font("Arial", Font.BOLD, 16));
        panel.add(resultButton);

        votedListTextArea = new JTextArea();
        votedListTextArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(votedListTextArea);

        panel.add(scrollPane);

        add(panel);
    }

    private class VoteButtonListener implements ActionListener {
        private int candidateId;

        public VoteButtonListener(int candidateId) {
            this.candidateId = candidateId;
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            String voterIdStr = voterIdTextField.getText().trim();

            if (voterIdStr.isEmpty()) {
                JOptionPane.showMessageDialog(null, "Please enter Voter ID.", "Error",
JOptionPane.ERROR_MESSAGE);
                return;
            }

            try {
                int voterId = Integer.parseInt(voterIdStr);

                // Check if the entered voterId is valid
                int voterIndex = getVoterIndex(voterId);
                if (voterIndex != -1 && !votedStatus[voterIndex]) {
                    Candidate candidate = candidates.get(candidateId);
                    candidate.vote();
                    votedStatus[voterIndex] = true; // Mark the voter as voted
                    votedListTextArea.append(candidate.getName() + " voted.\n");
                } else if (voterIndex == -1) {
                    JOptionPane.showMessageDialog(null, "Invalid Voter ID. Please enter a valid number.", "Error", JOptionPane.ERROR_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(null, "This voter has already cast a vote.", "Error", JOptionPane.ERROR_MESSAGE);
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(null, "Invalid Voter ID. Please enter a valid number.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }

        private int getVoterIndex(int voterId) {
            for (int i = 0; i < validVoterIds.length; i++) {
                if (voterId == validVoterIds[i]) {
                    return i;
                }
            }
            return -1;
        }
    }

    private class ResultButtonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            StringBuilder result = new StringBuilder("Election Result:\n");

            Candidate winner = null;
            for (Candidate candidate : candidates.values()) {
                result.append(candidate.getName()).append(": ").append(candidate.getVotes()).append("
votes\n");
                if (winner == null || candidate.getVotes() > winner.getVotes()) {
                    winner = candidate;
                }
            }

            if (winner != null) {
                result.append("\nWinner: ").append(winner.getName()).append(" (Symbol:
").append(winner.getSymbol()).append(")");
            }

            JTextArea resultTextArea = new JTextArea(result.toString());
            resultTextArea.setFont(new Font("Arial", Font.BOLD, 16));
            resultTextArea.setEditable(false);
            JOptionPane.showMessageDialog(null, new JScrollPane(resultTextArea), "Election Result",
JOptionPane.INFORMATION_MESSAGE);
        }
    }
}

public class ElectionCommisionOFPakistan {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            VotingMachineGUI votingMachineGUI = new VotingMachineGUI();
            votingMachineGUI.setVisible(true);
        });
    }
}
```

Algorithm

The algorithm of the "Voting Machine App" code can be outlined as follows:

1. **Initialization:**

   - Initialize a map to store candidate information (name, symbol, and votes).

   - Initialize an array to store valid voter IDs and a boolean array to track whether each voter has voted.

2. **GUI Creation:**

   - Create a graphical user interface using Java's Swing library.

   - Include components such as buttons, labels, text fields, and text areas to design the voting machine interface.

   - Set up the layout using panels and set fonts for better readability.

3. **Candidate Class:**

   - Create a `Candidate` class with attributes for name, symbol, and votes.

   - Provide methods to retrieve candidate information (name, symbol, votes) and to increment votes.

4. **Voter Authentication:**

   - Ask the user to enter their voter ID before casting a vote.

   - Validate the entered voter ID against a predefined array of valid voter IDs.

   - Ensure that a voter has not already cast a vote.

5. **Vote Casting:**

- Implement an action listener for each candidate button.

- When a vote button is clicked, verify the voter ID, check its validity, and update the vote count for the selected candidate.

- Mark the voter as having voted to prevent multiple votes.

6. **Result Display:**

- Implement an action listener for the "Show Result" button.

- Calculate and display the election results, including the total votes for each candidate and the overall winner.

- Present the results in a scrollable text area within a dialog.

7. **Main Method:**

- The main method initializes the `VotingMachineGUI` class and makes the GUI visible.

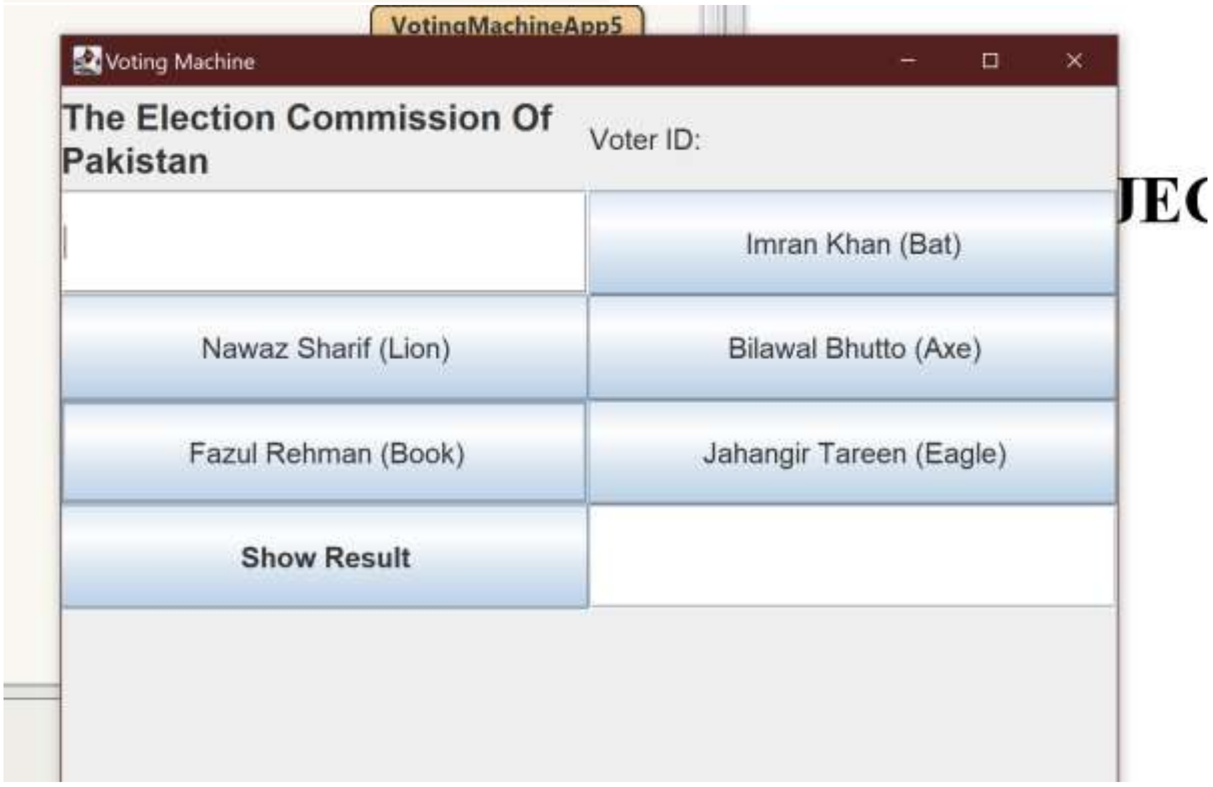
8. **Helper Methods:**

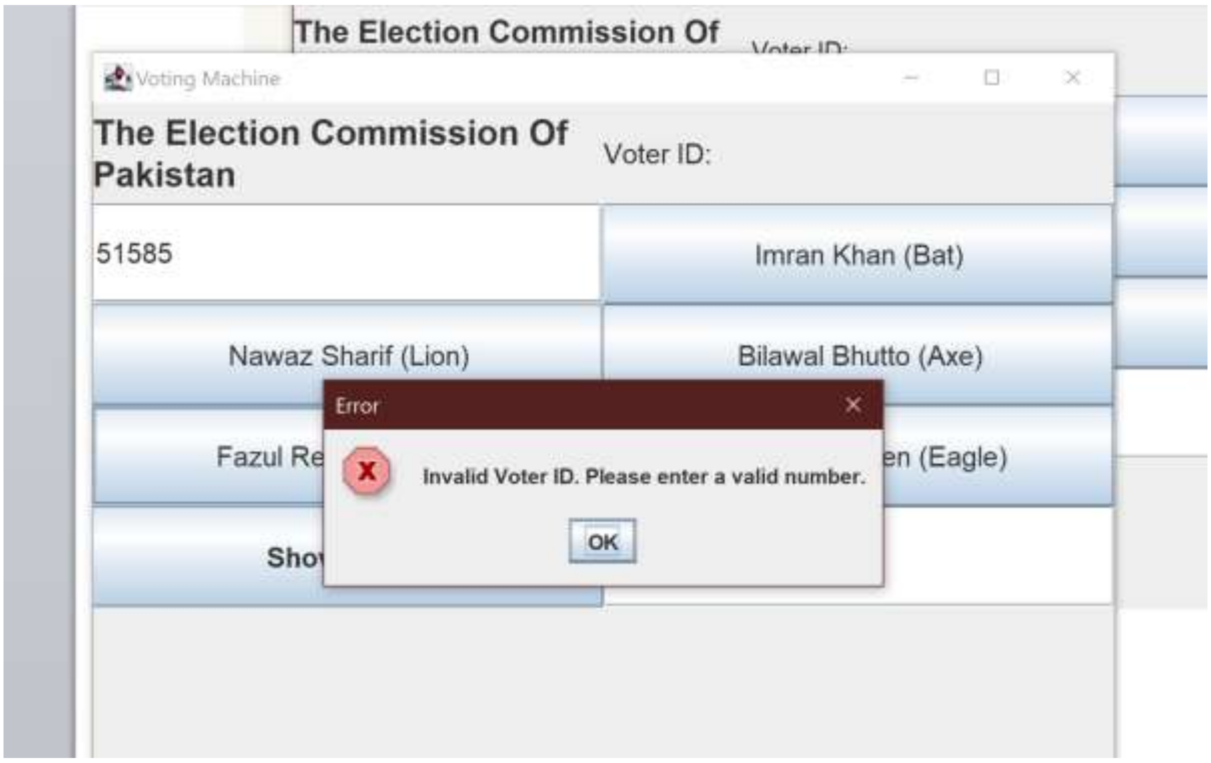
- `initializeCandidates`: Initialize the map of candidates with unique identifiers, names, and symbols.

- `getVoterIndex`: Helper method to retrieve the index of a voter ID in the array of valid voter IDs.

The code utilizes object-oriented principles, including encapsulation and abstraction, to create a modular and organized structure. Event-driven programming is employed to handle user interactions, such as button clicks, through action listeners. The algorithm ensures that the voting process is secure, with voter authentication and prevention of multiple votes. The result presentation is designed to be clear and accessible through a well-formatted dialog.

PICTURE OF PROJECT




Voting Machine

The Election Commission Of Pakistan

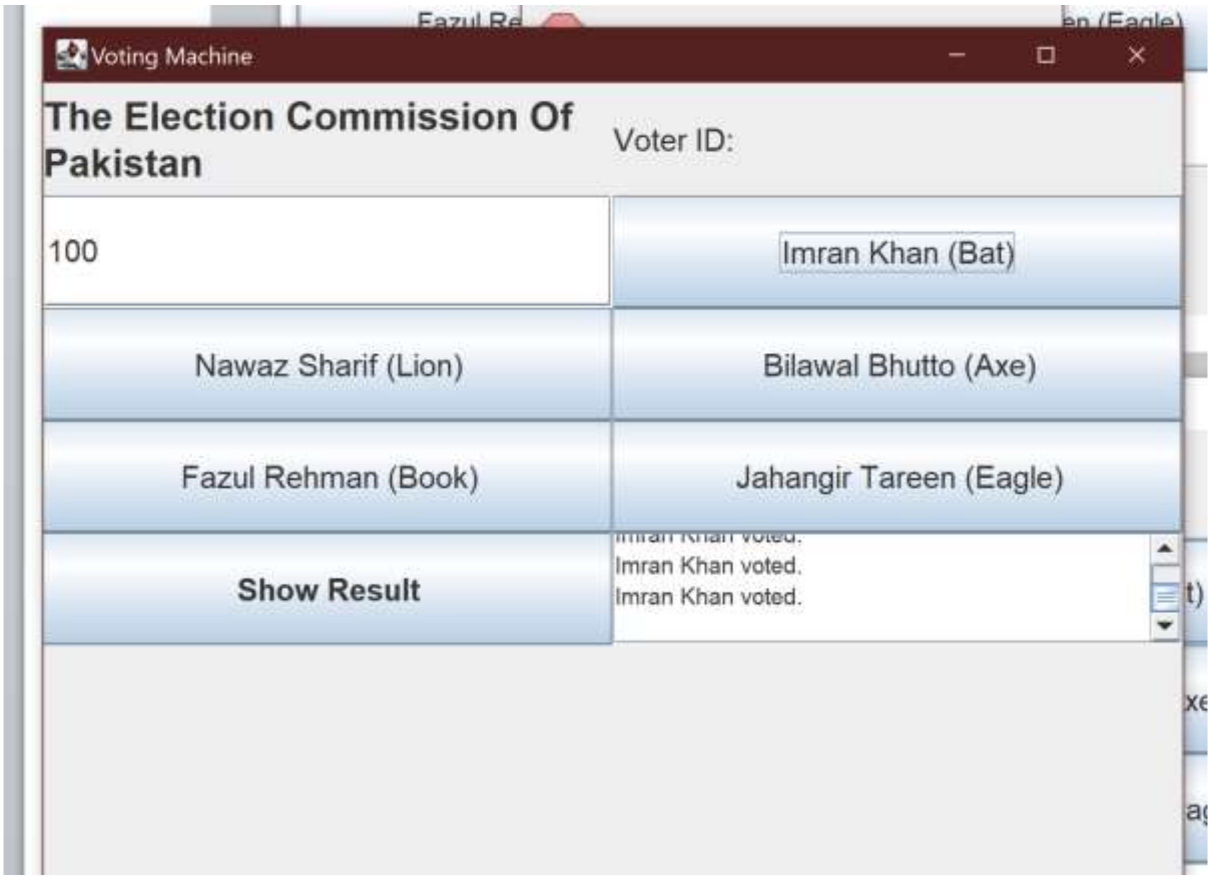
Voter ID:

51585

Error

Invalid Voter ID. Please enter a valid number.

OK

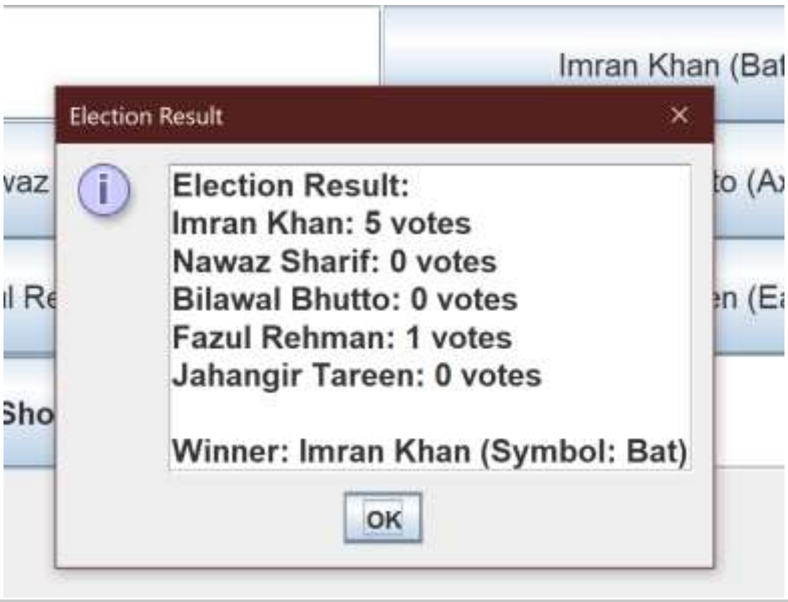

Voting Machine

The Election Commission Of Pakistan

Voter ID:

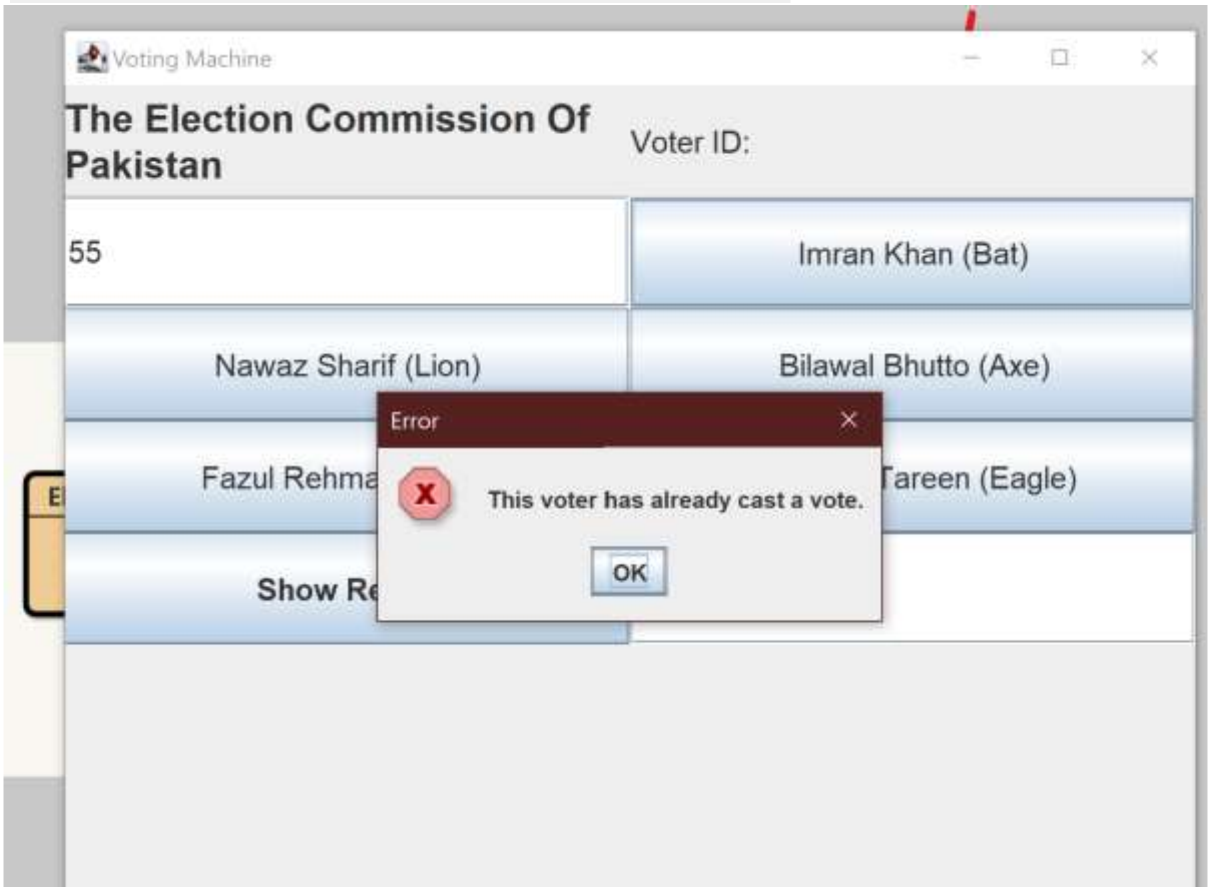
100

```
Imran Khan voted.
Imran Khan voted.
```



```
Election Result:
Imran Khan: 5 votes
Nawaz Sharif: 0 votes
Bilawal Bhutto: 0 votes
Fazul Rehman: 1 votes
Jahangir Tareen: 0 votes

Winner: Imran Khan (Symbol: Bat)
```



```
This voter has already cast a vote.
```

Recommendations & Conclusion

Recommendations:

1. **Security Enhancements:**

   - Implement additional security features to ensure the integrity of the voting process. This may include encryption of voter IDs and votes to prevent tampering.

2. **User Authentication:**

   - Introduce a more robust voter authentication mechanism, such as the use of biometrics or voter cards, to enhance the reliability of the system.

3. **Logging and Auditing:**

   - Implement a logging mechanism to record all voting activities. This can be valuable for auditing and ensuring transparency in the electoral process.

4. **User Experience Improvements:**

   - Enhance the user interface for better user experience. Consider incorporating visual cues and feedback to guide users through the voting process more intuitively.

5. **Error Handling:**

   - Strengthen error handling to provide more informative messages to users in case of invalid inputs or unexpected issues during the voting process.

6. **Scalability:**

- Evaluate the code for scalability to handle a larger number of candidates and voters. Consider optimizing data structures and algorithms if the scale of the election increases.

Conclusion:

The "Voting Machine App" project provides a foundational framework for an electronic voting system, leveraging Java's Swing library to create an interactive and user-friendly interface. The implementation incorporates fundamental features such as candidate representation, voter authentication, and result presentation.

The project successfully demonstrates the integration of object-oriented programming principles, ensuring modularity, maintainability, and readability of the code. The GUI design, coupled with event-driven programming, enhances the overall user experience and engagement.

While the current implementation serves as a functional prototype, the project could benefit from additional security measures, user authentication enhancements, and improvements to user experience. These recommendations aim to refine the system's robustness, security, and usability for potential deployment in real-world electoral scenarios.

In conclusion, the "Voting Machine App" project lays the groundwork for a digital voting system and can serve as a starting point for further development and refinement based on specific requirements and considerations. The project encapsulates the core principles of electronic voting and provides a platform for future enhancements and advancements in the field of digital democracy.

.