Online Voting System

DatabaseInitializer Code

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;


public class DatabaseInitializer {

    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/";

    private static final String USERNAME = "root";

    private static final String PASSWORD = "root";

    private static final String DATABASE_NAME = "voting_system";


    public static void main(String[] args) {

        try (Connection connection = DriverManager.getConnection(JDBC_URL, USERNAME,
PASSWORD)) {

            createDatabase(connection);


        } catch (SQLException e) {

            e.printStackTrace();

        }

    }


    private static void createDatabase(Connection connection) throws SQLException {

        String createDatabaseSQL = "CREATE DATABASE IF NOT EXISTS " + DATABASE_NAME;

        try (Statement statement = connection.createStatement()) {

            statement.executeUpdate(createDatabaseSQL);

            System.out.println("Database created successfully.");

        }

    }
}
```

TableSetup Code

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;


public class TableSetup {

    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/voting_system";

    private static final String USERNAME = "root";

    private static final String PASSWORD = "root";

    private static final String DATABASE_NAME = "voting_system";


    public static void main(String[] args) {

        try (Connection connection = DriverManager.getConnection(JDBC_URL, USERNAME,
PASSWORD)) {


            createTables(connection);


        } catch (SQLException e) {

            e.printStackTrace();

        }

    }



    private static void createTables(Connection connection) throws SQLException {

        String createVotersTableSQL = "CREATE TABLE IF NOT EXISTS voters (" +

                "id INT AUTO_INCREMENT PRIMARY KEY," +

                "voter_id VARCHAR(50) UNIQUE," +
            "name VARCHAR(100)," +
            "has_voted BOOLEAN DEFAULT FALSE" +
            ")";

        String createVotesTableSQL = "CREATE TABLE IF NOT EXISTS votes (" +
            "id INT AUTO_INCREMENT PRIMARY KEY," +
            "voter_id VARCHAR(50)," +
            "party VARCHAR(50)," +
            "FOREIGN KEY (voter_id) REFERENCES voters(voter_id)" +
            ")";

        try (Statement statement = connection.createStatement()) {
            statement.executeUpdate(createVotersTableSQL);
            statement.executeUpdate(createVotesTableSQL);
            System.out.println("Tables created successfully.");
        }
    }
}
```

VotingSystem Code

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.*;

public class VotingSystem extends JFrame implements ActionListener {
    private JButton voteButton, resultsButton;
private Connection connection;
 VotingPage votingPage;

public VotingSystem() {
   setTitle("Voting System");
   setSize(300, 150);
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   setLayout(new GridLayout(2, 1));

   connectToDatabase();

   voteButton = new JButton("Vote");
   resultsButton = new JButton("Display Results");

   voteButton.addActionListener(this);
   resultsButton.addActionListener(this);

   add(voteButton);
   add(resultsButton);

   setVisible(true);
}

private void connectToDatabase() {
   try {
      String JDBC_URL = "jdbc:mysql://localhost:3306/voting_system";
      String USERNAME = "root";
      String PASSWORD = "root";
      connection = DriverManager.getConnection(JDBC_URL, USERNAME, PASSWORD);
   } catch (SQLException e) {
      e.printStackTrace();
```

```
    }
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == voteButton) {
        dispose(); // Close the current frame
        openVotingPage();
    } else if (e.getSource() == resultsButton) {
        dispose(); // Close the current frame
        openResultsPage();
    }
}

private void openVotingPage() {
    votingPage = new VotingPage(connection, this); // Pass VotingSystem instance as an argument
    votingPage.setVisible(true);
}

private void openResultsPage() {
    ResultsPage resultsPage = new ResultsPage(connection);
    resultsPage.setVisible(true);
    // After displaying results, return to the main page

    // returnToMainPage();
    // dispose();
}

public void returnToMainPage() {
    VotingSystem mainPage = new VotingSystem();
    mainPage.setVisible(true);
}
```

```
    public static void main(String[] args) {
        SwingUtilities.invokeLater(VotingSystem::new);
    }

    public void returnToVotingSystem() {
        // Reopen the voting page after vote is saved
        openVotingPage();
    }
}

// VotingPage and ResultsPage classes remain unchanged

class VotingPage extends JFrame implements ActionListener {
    private JButton party1Button, party2Button, party3Button, party4Button;
    private ImageIcon party1Icon, party2Icon, party3Icon, party4Icon;
    private Connection connection;
    private VotingSystem votingSystem;

    public VotingPage(Connection connection, VotingSystem votingSystem) {
        this.connection = connection;
        this.votingSystem = votingSystem;

        setTitle("Voting Page");
        setSize(800, 800);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridLayout(2, 2));

        party1Icon = new ImageIcon("party1_image.png");
        party2Icon = new ImageIcon("party2_image.png");
        party3Icon = new ImageIcon("party3_image.png");
    party4Icon = new ImageIcon("party4_image.png");

    party1Button = new JButton(party1Icon);
    party2Button = new JButton(party2Icon);
    party3Button = new JButton(party3Icon);
    party4Button = new JButton(party4Icon);

    party1Button.addActionListener(this);
    party2Button.addActionListener(this);
    party3Button.addActionListener(this);
    party4Button.addActionListener(this);

    add(party1Button);
    add(party2Button);
    add(party3Button);
    add(party4Button);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == party1Button) {
        vote("Party 1");
    } else if (e.getSource() == party2Button) {
        vote("Party 2");
    } else if (e.getSource() == party3Button) {
        vote("Party 3");
    } else if (e.getSource() == party4Button) {
        vote("Party 4");
    }
}

private void vote(String party) {
        String voterId = JOptionPane.showInputDialog(this, "Enter Voter ID:");
        if (voterId != null && !voterId.isEmpty()) {
            try {
                PreparedStatement statement = connection.prepareStatement("INSERT INTO votes 
(voter_id, party) VALUES (?, ?)");
                statement.setString(1, voterId);
                statement.setString(2, party);
                int rowsAffected = statement.executeUpdate();
                if (rowsAffected > 0) {
                    JOptionPane.showMessageDialog(this, "Vote for " + party + " recorded successfully.");
                        dispose();
                    // Return to voting page
                    votingSystem.returnToMainPage();
                } else {
                    JOptionPane.showMessageDialog(this, "Failed to record vote.");
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Error occurred while recording vote.");
            }
        } else {
            JOptionPane.showMessageDialog(this, "Voter ID cannot be empty.");
        }
    }
}

class ResultsPage extends JFrame {
    private Connection connection;

    public ResultsPage(Connection connection) {
        this.connection = connection;
        setTitle("Results Page");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Display election results here
        displayResults();
    }

    private void displayResults() {
        JTextArea resultsArea = new JTextArea();
        resultsArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(resultsArea);
        add(scrollPane);

        try {
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT party, COUNT(*) AS votes FROM votes
GROUP BY party ORDER BY votes DESC LIMIT 1");
            if (resultSet.next()) {
                String winningParty = resultSet.getString("party");
                int votes = resultSet.getInt("votes");
                resultsArea.append("Winning Party: " + winningParty + ", Votes: " + votes + "\n");

                // Display image of the winning party
                ImageIcon winningIcon = null;
                switch (winningParty) {
                    case "Party 1":
                        winningIcon = new ImageIcon("party1_image.png");
                        break;
                    case "Party 2":
```

```
                winningIcon = new ImageIcon("party2_image.png");
                break;
            case "Party 3":
                winningIcon = new ImageIcon("party3_image.png");
                break;
            case "Party 4":
                winningIcon = new ImageIcon("party4_image.png");
                break;
            default:
                break;
        }

        if (winningIcon != null) {
            JLabel imageLabel = new JLabel(winningIcon);
            imageLabel.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    super.mouseClicked(e);
                    // Display additional details or perform actions on mouse click
                    JOptionPane.showMessageDialog(null, "You clicked on " + winningParty + ". ");

                }
            });
            add(imageLabel);
        }

    } else {
        resultsArea.append("No results available.");
    }
} catch (SQLException e) {
    e.printStackTrace();
```

```
        }
    }
}
```

VotingSystem_Register

```
import javax.swing.*;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.*;

import java.awt.event.*;

import java.io.File;


public class VotingSystem_Register extends JFrame implements ActionListener {

    // Components for login panel

    // JLabel loginTitleLabel, loginUsernameLabel, loginPasswordLabel;

    // JTextField loginUsernameField;

    // JPasswordField loginPasswordField;

    // JButton loginButton;


    // Components for voter information panel

    JLabel nameLabel, phoneLabel, ageLabel,  imageLabel;  //voterIdLabel,

    JTextField nameField, phoneField, ageField; //voterIdField

    JButton uploadImageButton, submitButton;


    // Card layout and container

    JPanel cardPanel;

    CardLayout cardLayout;

    ImageIcon selectedImage;


    public VotingSystem_Register() {

        // Frame setup
```

```
setTitle("Online Voting System");
setSize(400, 400);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

// Create card panel and set layout
cardPanel = new JPanel();
cardLayout = new CardLayout();
cardPanel.setLayout(cardLayout);

// Login Panel
// JPanel loginPanel = new JPanel(new GridLayout(4, 2));
// loginTitleLabel = new JLabel("Login Form");
// loginTitleLabel.setHorizontalAlignment(JLabel.CENTER);
// loginUsernameLabel = new JLabel("Username:");
// loginPasswordLabel = new JLabel("Password:");
// loginUsernameField = new JTextField();
// loginPasswordField = new JPasswordField();
// loginButton = new JButton("Login");

// loginPanel.add(loginTitleLabel);
// loginPanel.add(new JLabel(""));
// loginPanel.add(loginUsernameLabel);
// loginPanel.add(loginUsernameField);
// loginPanel.add(loginPasswordLabel);
// loginPanel.add(loginPasswordField);
// loginPanel.add(new JLabel("")); // Blank space for password field
// loginPanel.add(loginButton);

// loginButton.addActionListener(this);

// Voter Information Panel
```

```
 JPanel voterInfoPanel = new JPanel(new GridLayout(7, 2));
 nameLabel = new JLabel("Name:");
 phoneLabel = new JLabel("Phone:");
 ageLabel = new JLabel("Age:");
 //voterIdLabel = new JLabel("Voter ID:");
 imageLabel = new JLabel("Image:");

 nameField = new JTextField();
 phoneField = new JTextField();
 ageField = new JTextField();
// voterIdField = new JTextField();
 uploadImageButton = new JButton("Upload Image");
 submitButton = new JButton("Submit");

 voterInfoPanel.add(nameLabel);
 voterInfoPanel.add(nameField);
 voterInfoPanel.add(phoneLabel);
 voterInfoPanel.add(phoneField);
 voterInfoPanel.add(ageLabel);
 voterInfoPanel.add(ageField);
// voterInfoPanel.add(voterIdLabel);
// voterInfoPanel.add(voterIdField);
 voterInfoPanel.add(imageLabel);
 voterInfoPanel.add(uploadImageButton);
 voterInfoPanel.add(new JLabel(""));
 voterInfoPanel.add(submitButton);

 uploadImageButton.addActionListener(this);
 submitButton.addActionListener(this);

 // Add panels to card panel
```

```
    //cardPanel.add(loginPanel, "login");
    cardPanel.add(voterInfoPanel, "voterInfo");

    // Add card panel to frame
    add(cardPanel);

    // Set frame visibility
    setVisible(true);

    // Start with login panel visible
    cardLayout.show(cardPanel, "login");
  }

  // Action performed when buttons are clicked
  public void actionPerformed(ActionEvent e) {
    // if (e.getSource() == loginButton) {
    //     String username = loginUsernameField.getText();
    //     String password = new String(loginPasswordField.getPassword());

    //     if (username.isEmpty() || password.isEmpty()) {
    //         JOptionPane.showMessageDialog(this, "Please fill in all fields.");
    //     } else {
    //         // Here you would typically authenticate the user against a database
    //         // For simplicity, let's assume successful login
    //         cardLayout.show(cardPanel, "voterInfo");
    //     }
    // } else
    if (e.getSource() == uploadImageButton) {
      // Open file chooser dialog to select image
      JFileChooser fileChooser = new JFileChooser();
```

```
            FileNameExtensionFilter filter = new FileNameExtensionFilter("Image Files", "jpg", "png",
"jpeg");
            fileChooser.setFileFilter(filter);

            int returnVal = fileChooser.showOpenDialog(this);
            if (returnVal == JFileChooser.APPROVE_OPTION) {
                File file = fileChooser.getSelectedFile();
                String imageName = file.getName();
                imageLabel.setText(imageName);

                // Load selected image and display it
                ImageIcon icon = new ImageIcon(file.getAbsolutePath());
                Image image = icon.getImage().getScaledInstance(500, 500, Image.SCALE_SMOOTH);
                selectedImage = new ImageIcon(image);
                imageLabel.setIcon(selectedImage);
            }
        } else if (e.getSource() == submitButton) {
            // Process voter information and image upload
            String name = nameField.getText();
            String phone = phoneField.getText();
            String age = ageField.getText();
           // String voterId = voterIdField.getText();  + "\nVoter ID: " + voterId
            String imageName = imageLabel.getText();

            // Here you can process the data as needed
            // For now, let's just display the information
            JOptionPane.showMessageDialog(this, "Name: " + name + "\nPhone: " + phone + "\nAge: " +
age  + "\nImage: " + imageName, "Voter Information", JOptionPane.INFORMATION_MESSAGE);
            dispose();

           // VotingSystem = new VotingSystem() ;
        }
    }

    // Main method to start the application
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new VotingSystem_Register();
            }
        });
    }
}
```

SampleVoterInsertion Code

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class SampleVoterDataInsertion {
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/voting_system";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "root";

    public static void main(String[] args) {
        try (Connection connection = DriverManager.getConnection(JDBC_URL, USERNAME,
PASSWORD)) {
            // Insert sample voter IDs
            insertSampleVoters(connection);
            System.out.println("Sample voter IDs inserted successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void insertSampleVoters(Connection connection) throws SQLException {
    // Sample voter IDs to insert
    String[] sampleVoterIds = {"V1001", "V1002", "V1003", "V1004", "V1005"};

    String insertSQL = "INSERT IGNORE INTO voters (voter_id, name, has_voted) VALUES (?, ?, ?)";

    try (PreparedStatement statement = connection.prepareStatement(insertSQL)) {
        for (String voterId : sampleVoterIds) {
            statement.setString(1, voterId);
            statement.setString(2, "Sample Voter");
            statement.setBoolean(3, false); // Assuming none of them have voted initially
            statement.executeUpdate();
        }
    }
}

}
```

RunCommands (For all code in one)

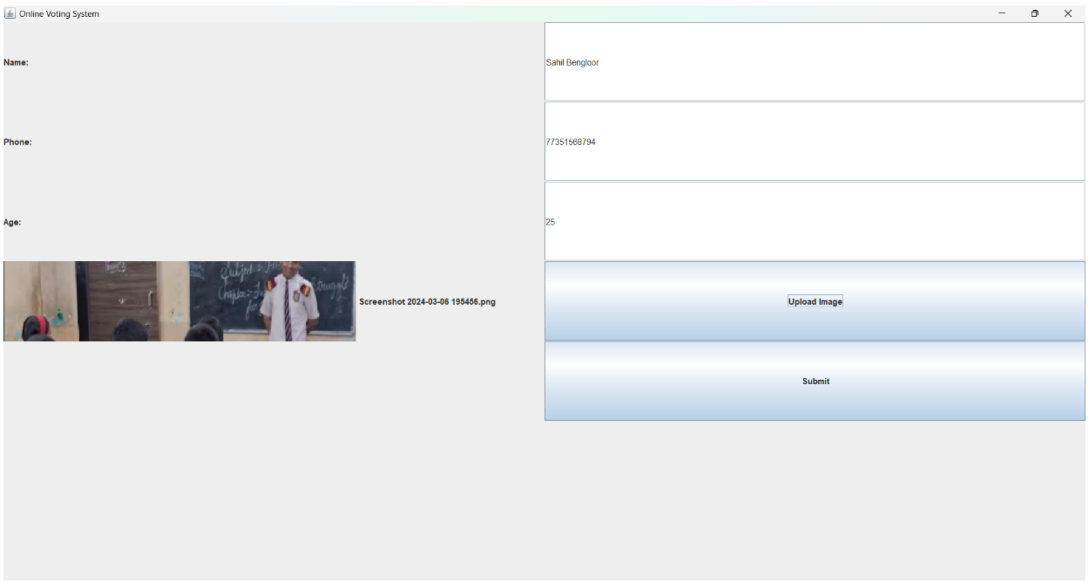
```
import java.io.IOException;
import java.util.Arrays;
import java.util.List;

public class RunCommands {
    public static void main(String[] args) {
        // Commands to be executed
        List<String> commands = Arrays.asList(
                "javac DatabaseInitializer.java SampleVoterDataInsertion.java TableSetup.java
VotingSystem_Register.java",
                "javac VotingSystem.java",
                "java DatabaseInitializer",
                "java TableSetup",
                "java SampleVoterDataInsertion",
                "java VotingSystem_Register",
                "java VotingSystem"
        );

        // Execute each command
        for (String command : commands) {
            try {
                ProcessBuilder pb = new ProcessBuilder(command.split(" "));
                pb.inheritIO(); // Redirect input/output to the current process
                Process process = pb.start();
                int exitCode = process.waitFor();
                if (exitCode != 0) {
                    System.err.println("Error occurred while executing command: " + command);
                }
            } catch (IOException | InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Output:

REGISTRATION PAGE



Online Voting System

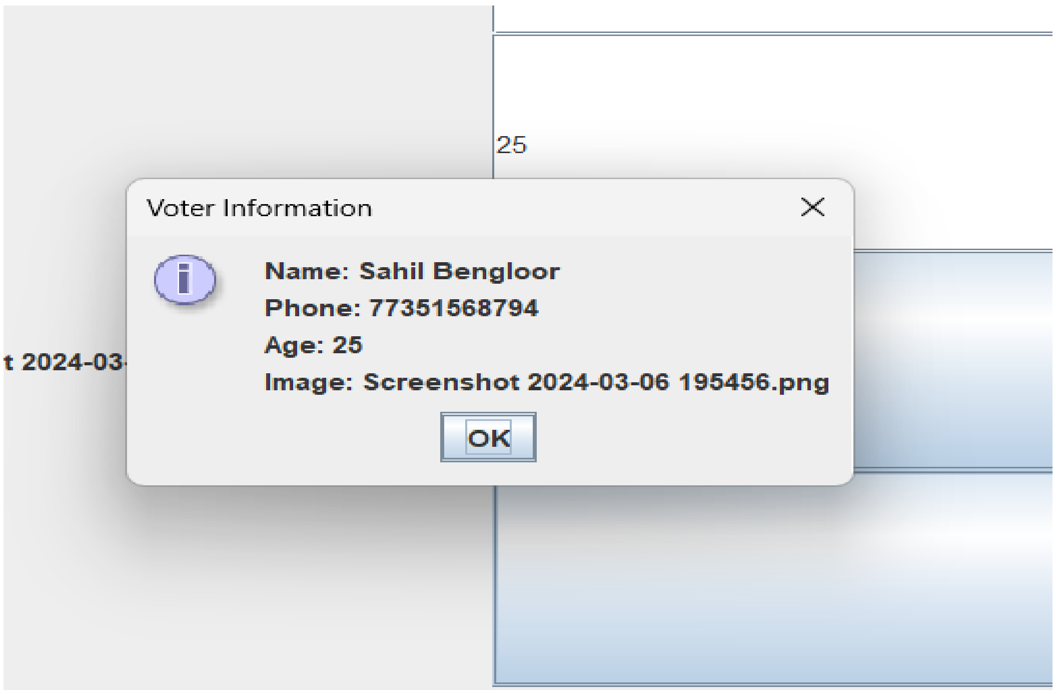
Name: Sahil Bengloor

Phone: 77351568794

Age: 25

Screenshot 2024-03-06 195456.png

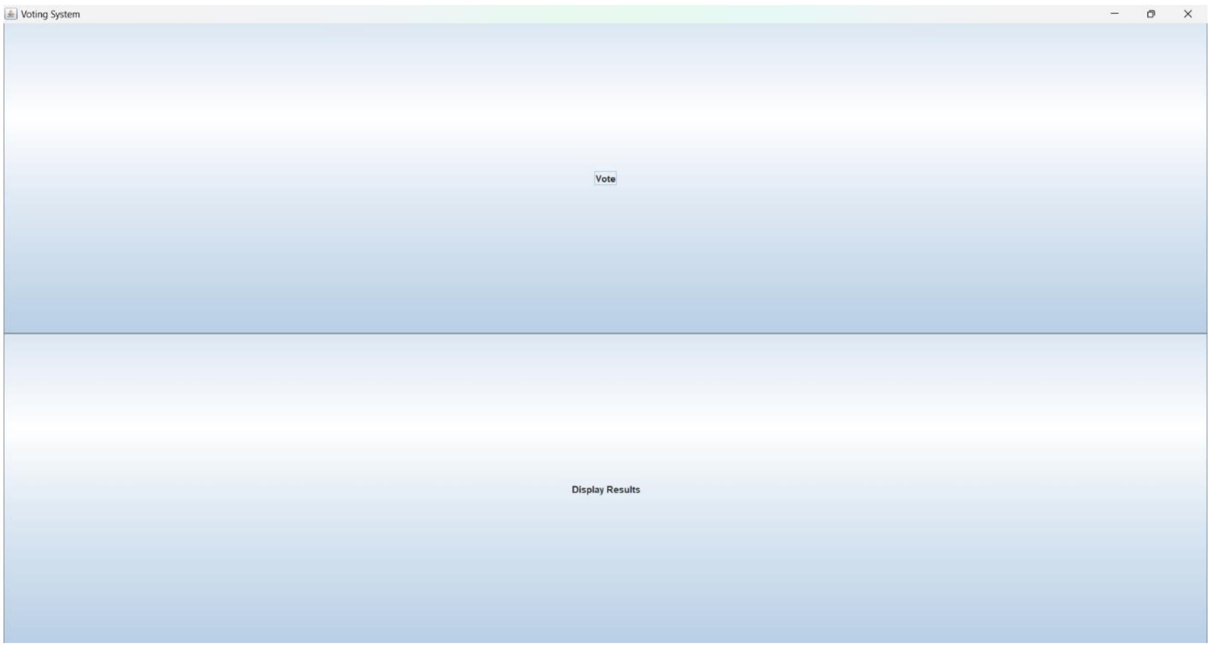
Registered Information Pop Up



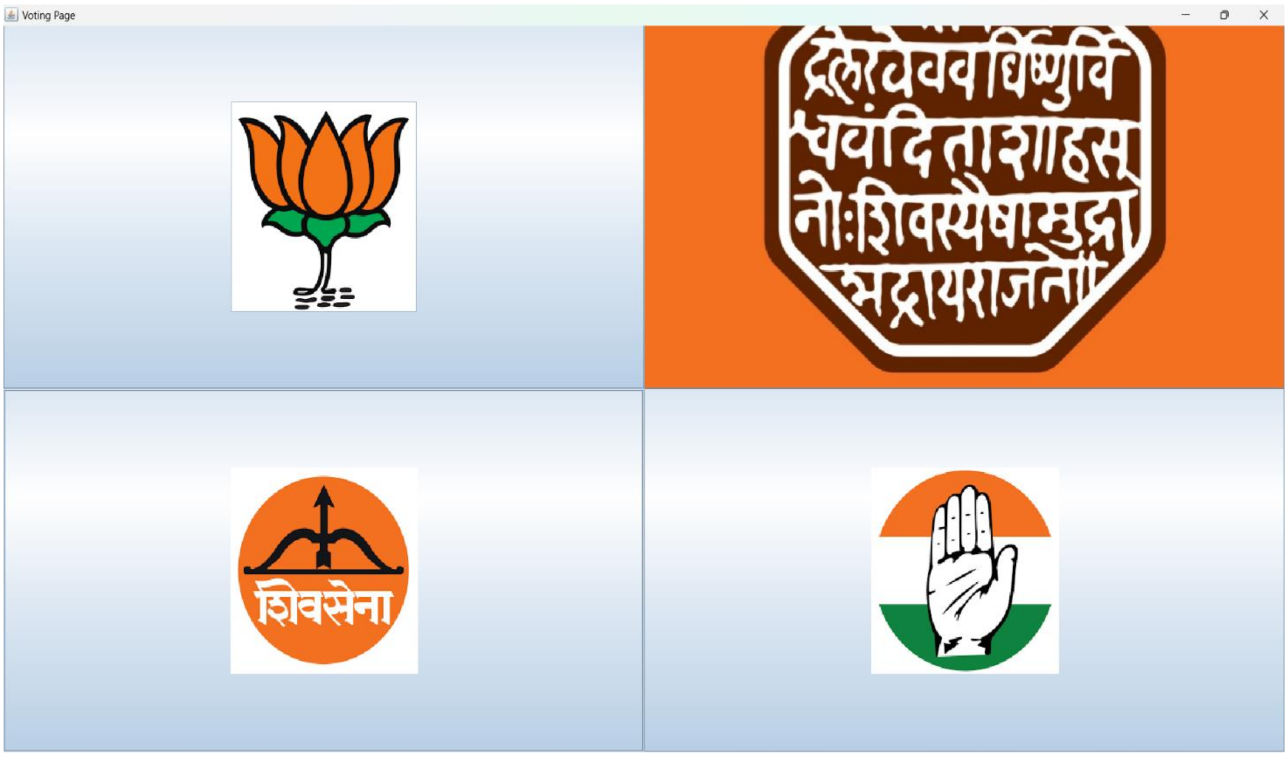
Voter Information

Name: Sahil Bengloor
Phone: 77351568794
Age: 25
Image: Screenshot 2024-03-06 195456.png

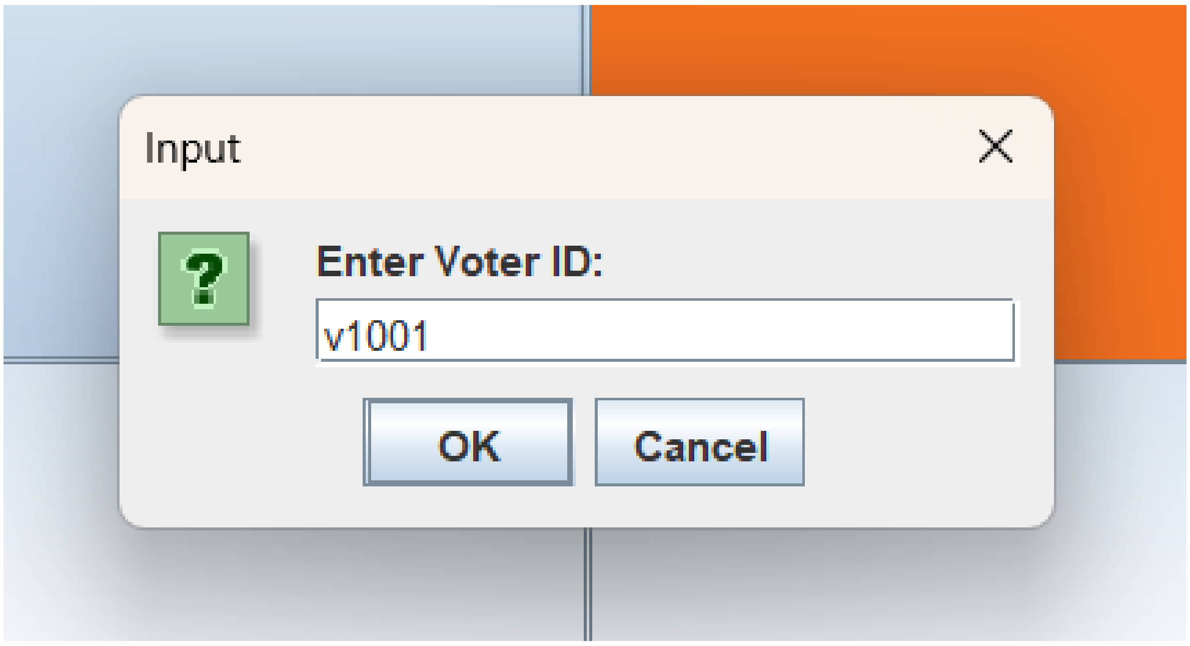
Vote or Result



Candidates

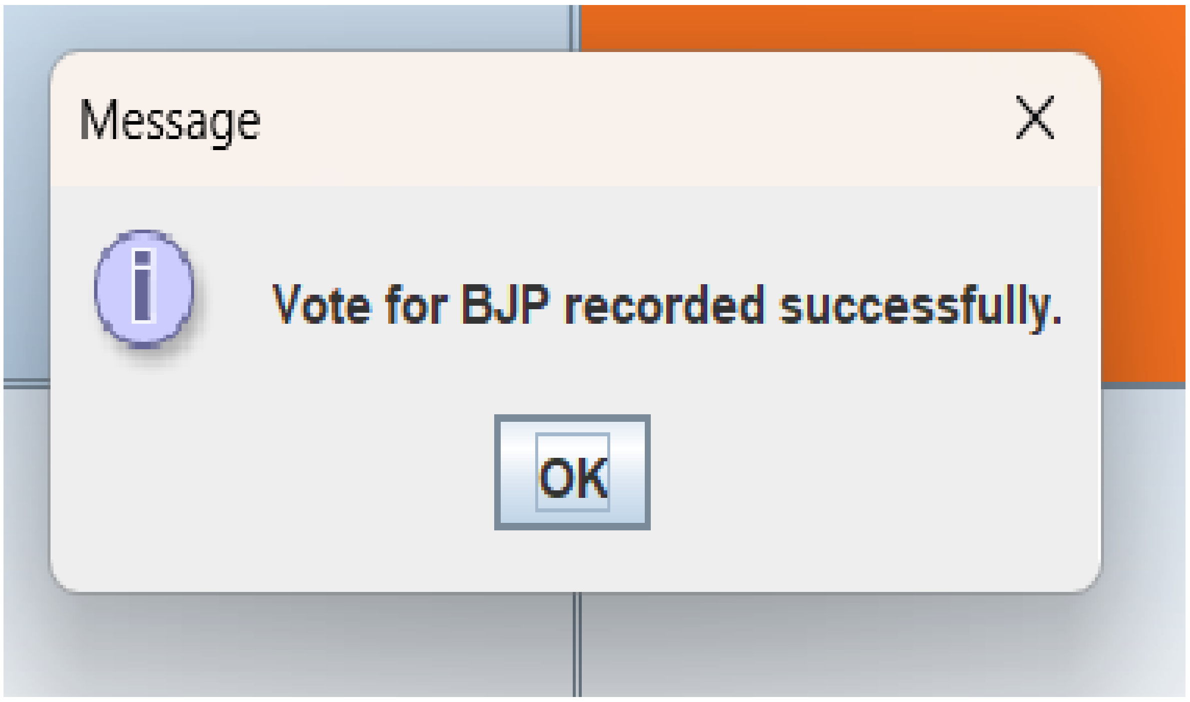


Enter Voter Id


Input

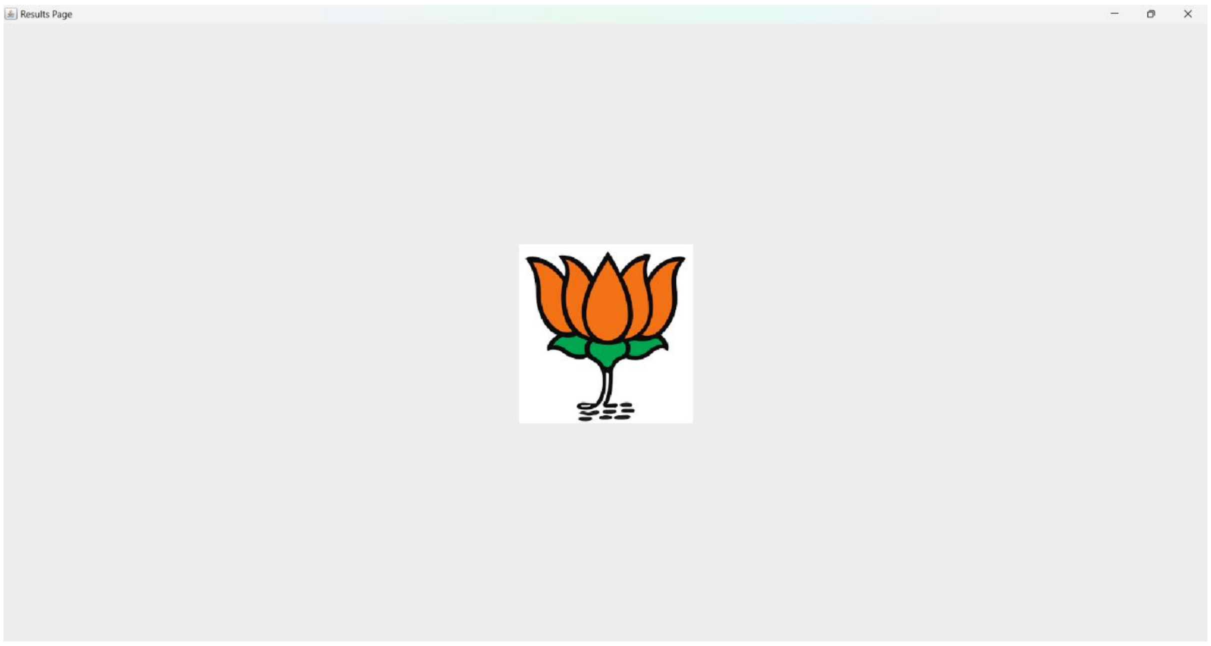
Enter Voter ID:

v1001

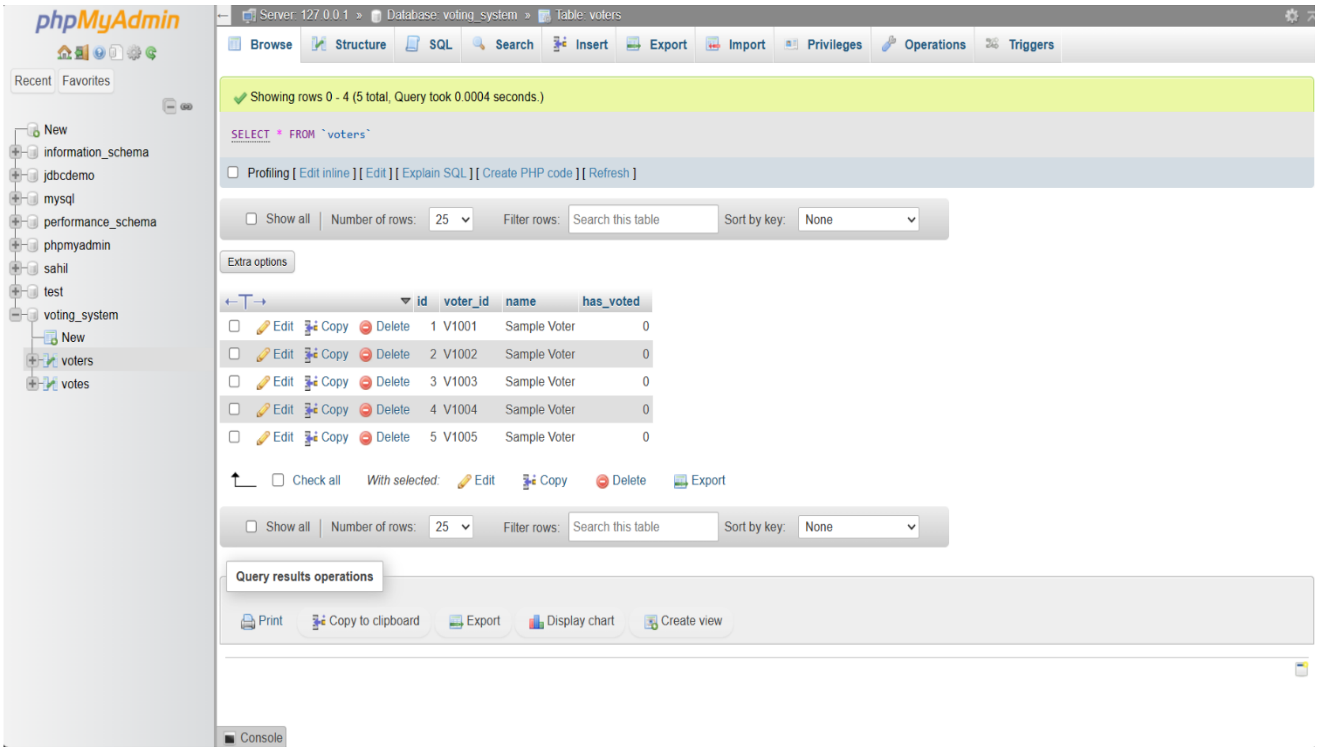

Message

Vote for BJP recorded successfully.

Result



Database of voter who can vote



Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)

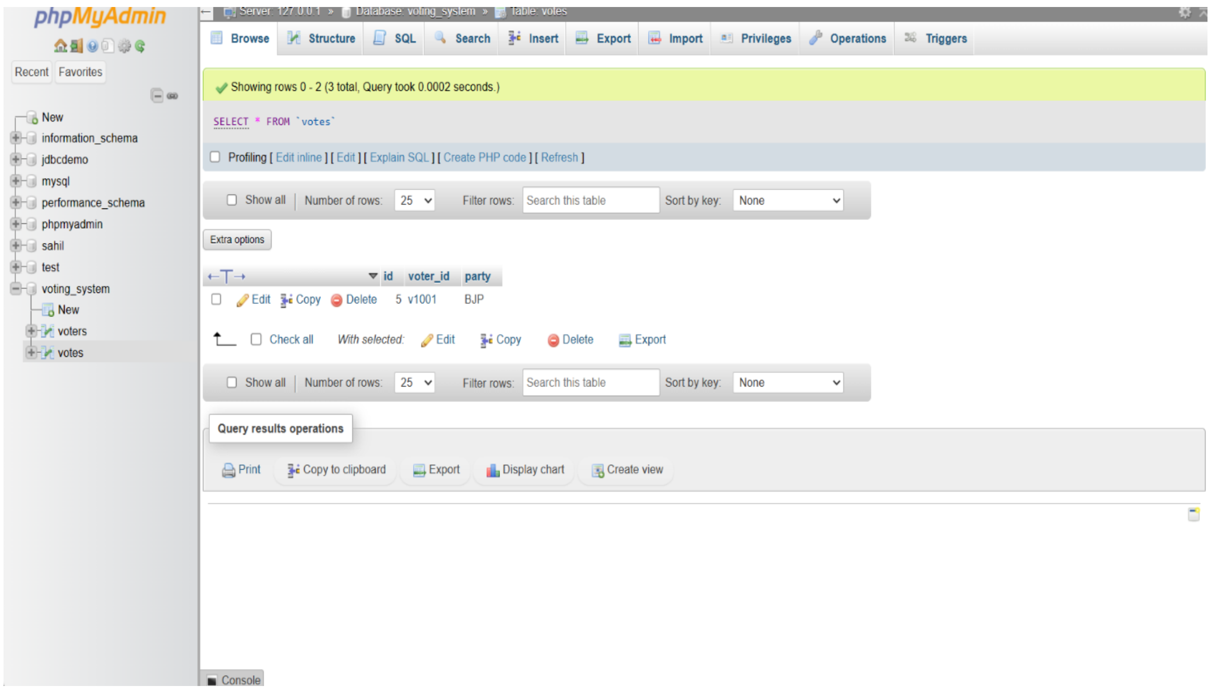
```
SELECT * FROM `voters`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

	id	voter_id	name	has_voted
Edit Copy Delete	1	V1001	Sample Voter	0
Edit Copy Delete	2	V1002	Sample Voter	0
Edit Copy Delete	3	V1003	Sample Voter	0
Edit Copy Delete	4	V1004	Sample Voter	0
Edit Copy Delete	5	V1005	Sample Voter	0

Database where Result stored



Server: 127.0.0.1 » Database: voting_system » Table: votes

Showing rows 0 - 2 (3 total, Query took 0.0002 seconds.)

```
SELECT * FROM `votes`
```

Number of rows: 25 Filter rows: Search this table Sort by key: None

id	voter_id	party
5	v1001	BJP