**PROJECT AIM:**

The aim of the Online Voting System project is to develop a user-friendly and secure platform that enables registered voters to easily cast their votes in elections. By validating voter information, securely recording votes, and allowing users to manage their voting records, the application streamlines the voting process. Utilizing JavaFX for the interface and MySQL for data storage, the project enhances transparency and accessibility in the electoral process.

**PROJECT DESCRIPTION:**

The Online Voting System is a JavaFX application designed to facilitate online voting for registered voters. The main features and functionalities of the application include:

1. User Interface: A clean and intuitive user interface allowing users to enter their voter ID, name, select a voting date, and choose a candidate from a dropdown list.
2. Candidate Management: The application connects to a MySQL database to retrieve and display a list of candidates dynamically. This allows for easy updates and management of candidates by election officials.
3. Vote Submission: Upon entering the required information, users can submit their votes. The application checks for duplicate voter IDs or names to ensure each voter can only vote once.
4. Vote Storage: Votes are securely stored in a MySQL database, ensuring data integrity and allowing for easy retrieval for audit purposes.
5. Vote Management:

* Users can view their voting records in a separate window, which displays details like voter ID, name, date of vote, and the selected candidate.
* The application provides options to delete or change a vote, facilitating easy management of voting records.

1. Error Handling and Validation:

The application includes error handling mechanisms to alert users in case of missing information or database errors, ensuring a smooth user experience.

1. Database Interaction:

The application uses JDBC for database connectivity, allowing it to perform various operations such as inserting, deleting, and retrieving records from the database**.**

**SOURCE CODE:**

CREATE DATABASE OnlineVoting;

USE OnlineVoting;

CREATE TABLE Candidates (

candidate\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL UNIQUE

);

-- Create Votes Table

CREATE TABLE Votes (

id INT PRIMARY KEY AUTO\_INCREMENT,

voter\_id VARCHAR(20) NOT NULL,

voter\_name VARCHAR(50) NOT NULL,

vote\_date DATE NOT NULL,

candidate\_id INT,

FOREIGN KEY (candidate\_id) REFERENCES Candidates(candidate\_id),

UNIQUE (voter\_id)

);

CREATE TABLE Users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(100) NOT NULL,

has\_voted BOOLEAN DEFAULT FALSE

);

INSERT INTO Candidates (name) VALUES ('Candidate A'), ('Candidate B'), ('Candidate C');

package miniproject;

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import java.sql.\*;

import java.time.LocalDate;

public class OnlineVotingApp extends Application {

private static final String DB\_URL = "jdbc:mysql://localhost:3306/OnlineVoting";

private static final String DB\_USER = "root";

private static final String DB\_PASSWORD = "aparnar8843@\*";

private TableView<VoteRecord> voteTable;

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Online Voting System");

VBox mainPane = new VBox(10);

mainPane.setPadding(new Insets(15));

// Voter ID input

TextField voterIdField = new TextField();

voterIdField.setPromptText("Enter Voter ID");

// Voter Name input

TextField voterNameField = new TextField();

voterNameField.setPromptText("Enter Voter Name");

// Date Picker for Date of Vote

DatePicker voteDatePicker = new DatePicker();

voteDatePicker.setPromptText("Select Date of Vote");

// Candidate ComboBox

ComboBox<String> candidateComboBox = new ComboBox<>();

populateCandidates(candidateComboBox);

// Submit Button

Button submitButton = new Button("Submit Vote");

submitButton.setOnAction(e -> {

String voterId = voterIdField.getText();

String voterName = voterNameField.getText();

LocalDate voteDate = voteDatePicker.getValue();

String selectedCandidate = candidateComboBox.getSelectionModel().getSelectedItem();

if (voterId.isEmpty() || voterName.isEmpty() || voteDate == null || selectedCandidate == null) {

showAlert(Alert.AlertType.WARNING, "All fields are required.");

} else {

int candidateId = Integer.parseInt(selectedCandidate.split(":")[0]);

if (isVoterExists(voterId, voterName)) {

showAlert(Alert.AlertType.WARNING, "Voter ID or Name already exists.");

} else {

if (saveVote(voterId, voterName, voteDate, candidateId)) {

showAlert(Alert.AlertType.INFORMATION, "Vote successfully recorded!");

} else {

showAlert(Alert.AlertType.ERROR, "Failed to record vote.");

}

}

}

});

// View Details Button

Button viewDetailsButton = new Button("View Vote Details");

viewDetailsButton.setOnAction(e -> showVoteDetailsWindow());

// Add elements to the main layout

mainPane.getChildren().addAll(

new Label("Enter Voter ID"), voterIdField,

new Label("Enter Voter Name"), voterNameField,

new Label("Date of Vote"), voteDatePicker,

new Label("Select Candidate"), candidateComboBox,

submitButton, viewDetailsButton

);

primaryStage.setScene(new Scene(mainPane, 600, 400));

primaryStage.show();

}

// Check if voter ID or voter name already exists

private boolean isVoterExists(String voterId, String voterName) {

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD)) {

String sql = "SELECT COUNT(\*) FROM Votes WHERE voter\_id = ? OR voter\_name = ?";

PreparedStatement statement = conn.prepareStatement(sql);

statement.setString(1, voterId);

statement.setString(2, voterName);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

return resultSet.getInt(1) > 0;

}

} catch (SQLException e) {

showAlert(Alert.AlertType.ERROR, "Error checking voter existence: " + e.getMessage());

}

return false;

}

// Show vote details in a separate window

private void showVoteDetailsWindow() {

Stage detailsStage = new Stage();

detailsStage.setTitle("Vote Details");

voteTable = new TableView<>();

initializeTable();

loadVotesIntoTable();

VBox detailsPane = new VBox(10);

detailsPane.setPadding(new Insets(15));

// Delete Button

Button deleteButton = new Button("Delete Selected Vote");

deleteButton.setOnAction(e -> deleteSelectedVote());

// Change Button

Button changeButton = new Button("Change Selected Vote");

changeButton.setOnAction(e -> changeSelectedVote());

detailsPane.getChildren().addAll(voteTable, deleteButton, changeButton);

detailsStage.setScene(new Scene(detailsPane, 600, 400));

detailsStage.show();

}

// Populate ComboBox with candidates from the database

private void populateCandidates(ComboBox<String> candidateComboBox) {

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD)) {

String sql = "SELECT \* FROM Candidates";

PreparedStatement statement = conn.prepareStatement(sql);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

int id = resultSet.getInt("candidate\_id");

String name = resultSet.getString("name");

candidateComboBox.getItems().add(id + ": " + name); // Display "id: name" in ComboBox

}

} catch (SQLException e) {

showAlert(Alert.AlertType.ERROR, "Error loading candidates: " + e.getMessage());

}

}

// Save vote to database

private boolean saveVote(String voterId, String voterName, LocalDate voteDate, int candidateId) {

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD)) {

String insertVoteSQL = "INSERT INTO Votes (voter\_id, voter\_name, vote\_date, candidate\_id) VALUES (?, ?, ?, ?)";

PreparedStatement statement = conn.prepareStatement(insertVoteSQL);

statement.setString(1, voterId);

statement.setString(2, voterName);

statement.setDate(3, Date.valueOf(voteDate));

statement.setInt(4, candidateId);

statement.executeUpdate();

return true;

} catch (SQLException e) {

showAlert(Alert.AlertType.ERROR, "Error saving vote: " + e.getMessage());

return false;

}

}

// Initialize table columns

private void initializeTable() {

TableColumn<VoteRecord, String> voterIdColumn = new TableColumn<>("Voter ID");

voterIdColumn.setCellValueFactory(new PropertyValueFactory<>("voterId"));

TableColumn<VoteRecord, String> voterNameColumn = new TableColumn<>("Voter Name");

voterNameColumn.setCellValueFactory(new PropertyValueFactory<>("voterName"));

TableColumn<VoteRecord, LocalDate> voteDateColumn = new TableColumn<>("Date of Vote");

voteDateColumn.setCellValueFactory(new PropertyValueFactory<>("voteDate"));

TableColumn<VoteRecord, String> candidateNameColumn = new TableColumn<>("Candidate");

candidateNameColumn.setCellValueFactory(new PropertyValueFactory<>("candidateName"));

voteTable.getColumns().addAll(voterIdColumn, voterNameColumn, voteDateColumn, candidateNameColumn);

}

// Load votes from database into table

private void loadVotesIntoTable() {

voteTable.getItems().clear();

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD)) {

String sql = "SELECT voter\_id, voter\_name, vote\_date, Candidates.name AS candidate\_name " +

"FROM Votes " +

"JOIN Candidates ON Votes.candidate\_id = Candidates.candidate\_id";

PreparedStatement statement = conn.prepareStatement(sql);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

String voterId = resultSet.getString("voter\_id");

String voterName = resultSet.getString("voter\_name");

LocalDate voteDate = resultSet.getDate("vote\_date").toLocalDate();

String candidateName = resultSet.getString("candidate\_name");

voteTable.getItems().add(new VoteRecord(voterId, voterName, voteDate, candidateName));

}

} catch (SQLException e) {

showAlert(Alert.AlertType.ERROR, "Error loading votes: " + e.getMessage());

}

}

// Delete selected vote

private void deleteSelectedVote() {

VoteRecord selectedVote = voteTable.getSelectionModel().getSelectedItem();

if (selectedVote == null) {

showAlert(Alert.AlertType.WARNING, "No vote selected for deletion.");

return;

}

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD)) {

String deleteSQL = "DELETE FROM Votes WHERE voter\_id = ?";

PreparedStatement statement = conn.prepareStatement(deleteSQL);

statement.setString(1, selectedVote.getVoterId());

statement.executeUpdate();

showAlert(Alert.AlertType.INFORMATION, "Vote deleted successfully.");

loadVotesIntoTable(); // Refresh the table

} catch (SQLException e) {

showAlert(Alert.AlertType.ERROR, "Error deleting vote: " + e.getMessage());

}

}

// Change selected vote

private void changeSelectedVote() {

VoteRecord selectedVote = voteTable.getSelectionModel().getSelectedItem();

if (selectedVote == null) {

showAlert(Alert.AlertType.WARNING, "No vote selected for change.");

return;

}

// Here you can implement logic to change the vote (e.g., open a new window to edit details)

// For example, you could create a new dialog or window similar to the main voting interface

// Populate fields with the selectedVote data and allow the user to make changes

}

// Show alert for validation or error messages

private void showAlert(Alert.AlertType type, String message) {

Alert alert = new Alert(type);

alert.setContentText(message);

alert.showAndWait();

}

public static void main(String[] args) {

launch(args);

}

// Data class for table records

public static class VoteRecord {

private final String voterId;

private final String voterName;

private final LocalDate voteDate;

private final String candidateName;

public VoteRecord(String voterId, String voterName, LocalDate voteDate, String candidateName) {

this.voterId = voterId;

this.voterName = voterName;

this.voteDate = voteDate;

this.candidateName = candidateName;

}

public String getVoterId() { return voterId; }

public String getVoterName() { return voterName; }

public LocalDate getVoteDate() { return voteDate; }

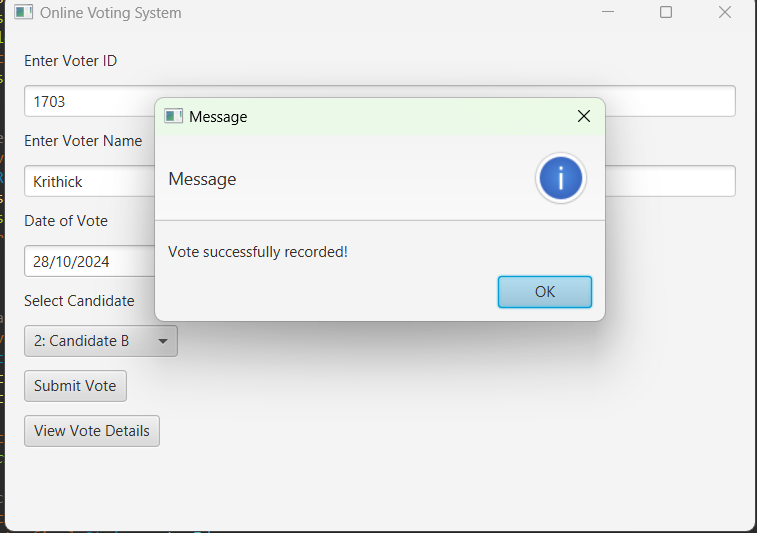
public String getCandidateName() { return candidateName; }

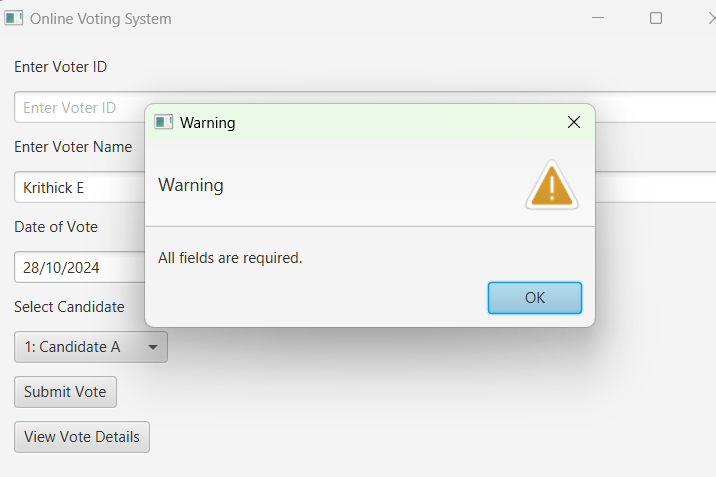
}

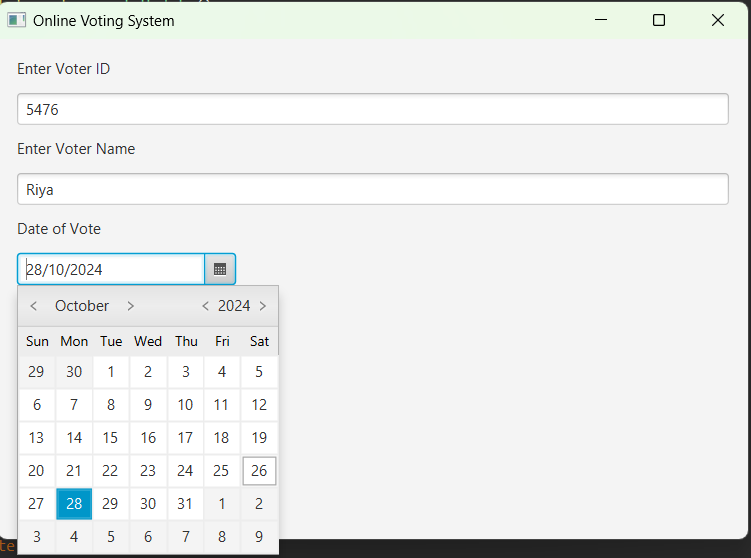
}

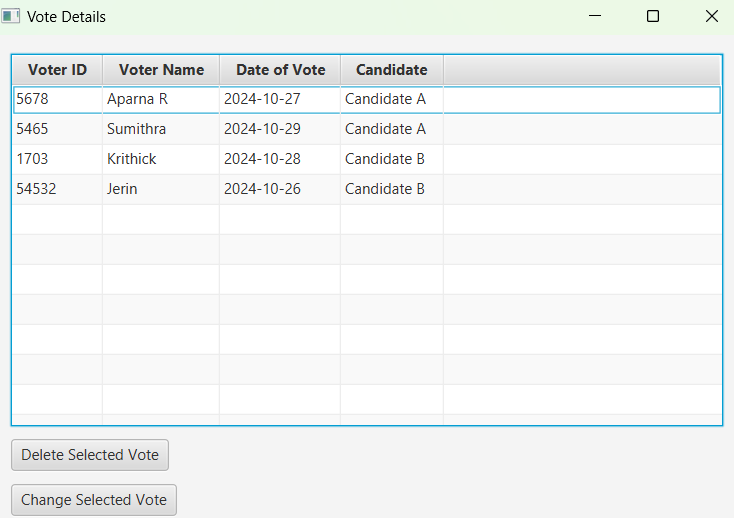
**OUTPUT:**

## 

****

****

****

****

**RESULT:**

The Online Voting System effectively enables registered voters to cast their votes through a user-friendly and secure platform.