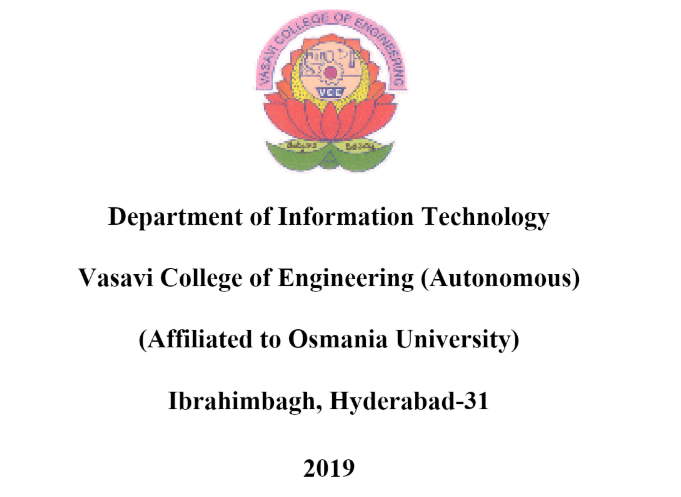
**JAVA AWT BASED- SQL CONNECTIVITY USING JDBC**

A REPORT BASED ON THE PROJECT:

AUTOMATED POLLING SYSTEM

BY:

V.SREYA <1602-18-737-092>



**BONAFIDE CERTIFICATE**

Certified that this project report titled “Online Polling System” is bonafide work of Ms. V.Sreya, who carried out the mini project work under my supervision.

Certified further that, to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion or any other candidate.

Signature of the Examiner

**B.LEELAVATHY**

Lecture

Department of Information Technology.

**ABSTRACT**

An automated polling system helps in registering and certifying voters and collecting their votes. This is a computer based vote casting system that has access to the database which checks if the voter is eligible to vote or not. The database includes 5 tables for storing the information related to the system. The five tables are:

* Voters details
* Nominees details
* Admin
* Parties that are taking part.
* City

The entity ‘voter’ is related to the entity ‘city’ through the relationship ‘from’ and to entity ‘nominee’ through ‘castes\_vote’. The entity ‘admin’ is associated to the entity ‘voter’ through the relationship ‘,manage’ whereas the entity ‘nominee’ is related to entity ‘party’ through the relation ‘belongs\_to’. The common attributes include ID and name where Id acts as the primary key for that table and as a foreign key to some other table. Domain types used are varchar2, number and date.

This system also provides a manual choice for voters to select their candidate by going to polling booth manually. It comprises of an alarm unit which will turn ON just after the completion of polling of a single person and this process will repeat every time to ensure the voters that they had casted vote successfully.

**INTRODUCTION**

REQUIREMENTS FOR AN AUTOMATED POLLING SYSTEM:

* Functional Requirements:

1. Mobility: The voter should not be restricted to cast his ballot at a single poll-site at his home precinct.

• Realistic: He shall be able to vote from any poll-site within the nation. • Unrealistic/Expensive: He shall be able to vote from any county-controlled kiosk (situated at public places such as banks, shopping malls, etc.) within the nation. (Unrealistic because of logistical and cost issues). • Infeasible: He shall be able to vote from virtually anywhere using an Internet connection. (Infeasible both for technical security issues as well as social science issues).

1. Convenience: The system shall allow the voters to cast their votes quickly, in one session, and should not require many special skills or intimidate the voter (to ensure Equality of Access to Voters).
2. User-Interface: The system shall provide an easy-to-use user-interface. Also, it shall not disadvantage any candidate while displaying the choices (e.g., by requiring the user to scroll down to see the last few choices).
3. Voter Confirmation: The voter shall be able to confirm clearly how his vote is being cast, and shall be given a chance to modify his vote before he commits it.
4. To issue Receipt or not?

• The system may issue a receipt to the voter if and only if it can be ensured that vote-coercion and vote-selling are prevented, so that he may verify his vote at any time and also contend, if necessary.

1. No Over-voting: The voter shall be prevented from choosing more than one candidate / answer.
2. Under-voting: The voter may receive a warning of not voting, but the system must not prevent under-voting.

* Security Requirements:

1. Voter Authenticity: Ensure that the voter must identify himself (with respect to the registration database) to be entitled to vote. If voting other than at his home precinct, the voter may be asked to show some legal identification document.

2. Registration: The voter registration shall be done in person only. However, the computerized registration database shall be made available to polling-booths all around the nation.

3. Voter Anonymity: Ensure that votes must not be associated with voter identity.

4. System Integrity: Ensure that the system cannot be re-configured during operation.

5. Data Integrity: Ensure that each vote is recorded as intended and cannot be tampered with in any manner, once recorded (i.e., votes should not be modified, forged or deleted without detection).

6. Secrecy / Privacy: No one should be able to determine how any individual voted.

7. Non-coercibility and No Vote-selling: Voters should not be able to prove to others how they voted (which would facilitate vote selling or coercion).

8. Reliability: Election systems should work robustly, without loss of any votes, even in the face of numerous failures, including failures of voting machines and total loss of network communication. The system shall be developed in a manner that ensures there is no malicious code or bugs.

9. Availability: Ensure that system is protected against accidental and malicious denial of service attacks. Also, setup redundant communication paths so that availability is ensured.

* Through the project:

The main goal to be achieved through this project was to provide an opportunity to display the details of various voters, admin, nominees, parties and cities taking part in the polling and to let a voter cast their vote to a nominee online.

The project also ensure that the votes that are casted are confidential and are stored in the database.

SQL particular voter, admin, nominee, party or city can be executed.

* Architecture and technology used:

SQL Plus is the most basic Oracle Database utility with a basic command-line interface, commonly used by users, administrators and programmers.

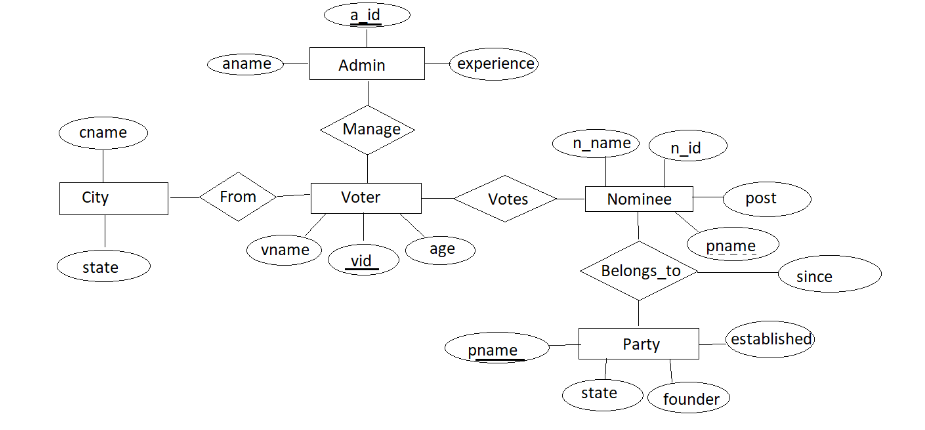
The interface of SQL Plus is used for creating the database. DDL and DML commands are implemented for operations being executed. The details of various voters, admins, nominees, parties and cities are stored in the form of tables in the database.

Eclipse is an integrated development environment(IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Erlang, JavaScripts etc.

The front end application code is written in “Java” using Eclipse. The portal for front end application is designed through Eclipse, runs and has the capacity to connect with the database which has data inserted using SQL.

* Design:

ER DIAGRAM



* DATABASE DESIGN:

CONTENT:

* Abstract
* ER Diagram
* Logical database design – DDL commands
* Enforcing primary and foreign keys.
* DML operation and outputs.

**ABSTRACT:**

An automated polling system helps in registering and certifying voters and collecting their votes. This is a computer based vote casting system that has access to the database which checks if the voter is eligible to vote or not. The database includes 5 tables for storing the information related to the system. The five table are:

* Voters details
* Nominees details
* Admin
* Parties that are taking part.
* City

The entity ‘voter’ is related to the entity ‘city’ through the relationship ‘from’ and to entity ‘nominee’ through ‘castes\_vote’. The entity ‘admin’ is associated to the entity ‘voter’ through the relationship ‘,manage’ whereas the entity ‘nominee’ is related to entity ‘party’ through the relation ‘belongs\_to’. The common attributes include ID and name where Id acts as the primary key for that table and as a foreign key to some other table. Domain types used are varchar2, number and date.

**LIST OF REQUIREMENTS:**

|  |
| --- |
| List of parties taking part in the polling. |
| Details of the nominees and voters. |
| Details of parties. |
| Cities and constituencies involved. |

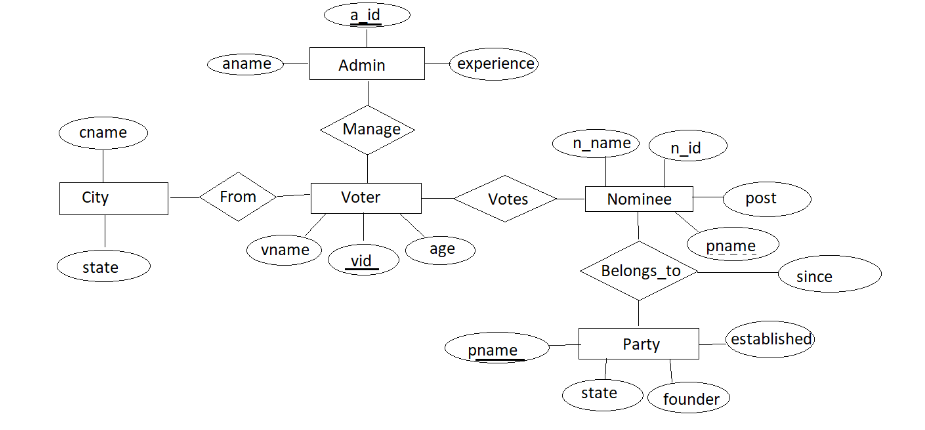
The database requires 5 table for implementation of the polling system.

The five table and their attributes are:

* Voter
* Voter id : vid- number(5)
* Voter name : vname- varchar2(20)
* Age: number(5)
* Nominee
* Nominee name : n\_name- varchar2(20)
* Nominee id : n\_id- number(5)
* Post : varchar2(20)
* Party name : p\_name- varchar2(20)
* Admin
* Admin id : a\_id- number(5)
* Admin name : a\_name- varchar2(20)
* Experience : number(5)
* Party
* Party name : p\_name- varchar2(20)
* State: varchar2(20)
* Founder : varchar2(20)
* Established : number(5)
* City
* City name : cname- varchar2(20)
* City state : cstate- varchar2(20)
* Belongs\_to
* Since :number(5)

The descriptive attribute ‘since’ in the table ‘belongs\_to’ tell about since when the nominee belongs to the party

**ER Diagram:**

****

**CONSTRAINTS APPLIED:**

The database has two constraints that are applied- The primary key constraint and foreign key constraint. The primary keys are:

* a\_id in the admin table.
* vid in the voter table.
* pname in the party table.

The attribute pname acts as foreign in the nominee table.

**DDL Commands:**

**For creating party table:**

Query: create table party(pname varchar2(20) primary key,

state varchar2(20),

founder varchar(20),

established number(5));

**For creating nominee table:**

Query: create table party(n\_name varchar2(20),

n\_id number(5),

post varchar(20),

pname varchar2(20));

**For creating voter table:**

Query: create table voter(vid number(5) primary key,

vname varchar2(20),

age number(5));

**For creating admin table:**

Query: create table admin(aid number(5) primary key,

aname varchar2(20),

experience number(5));

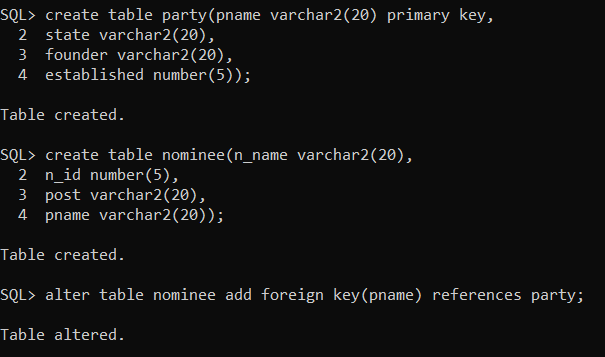
**For creating city table:**

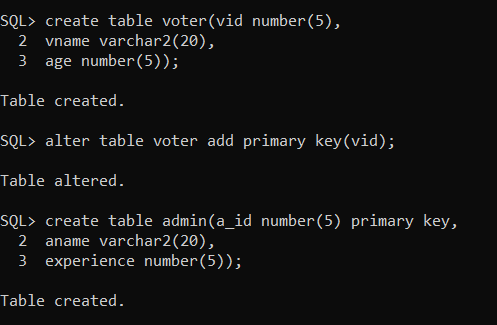
Query: create table city(cname varchar2(20),

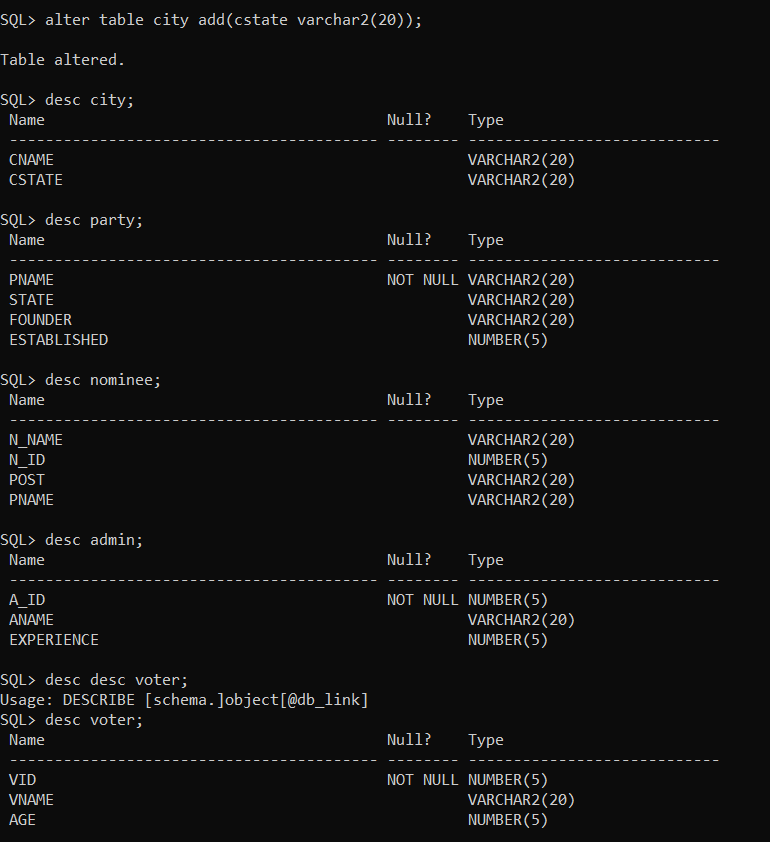
cstate varchar2(20));

**For creating belongs\_to table:**

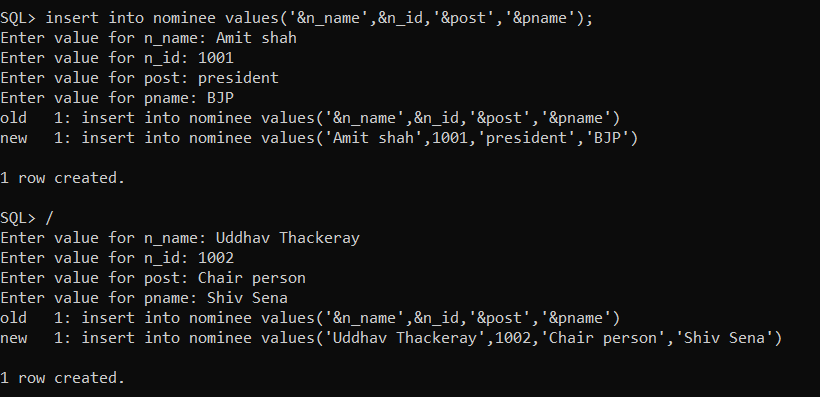
Query: create table belongs\_to(since number(5));

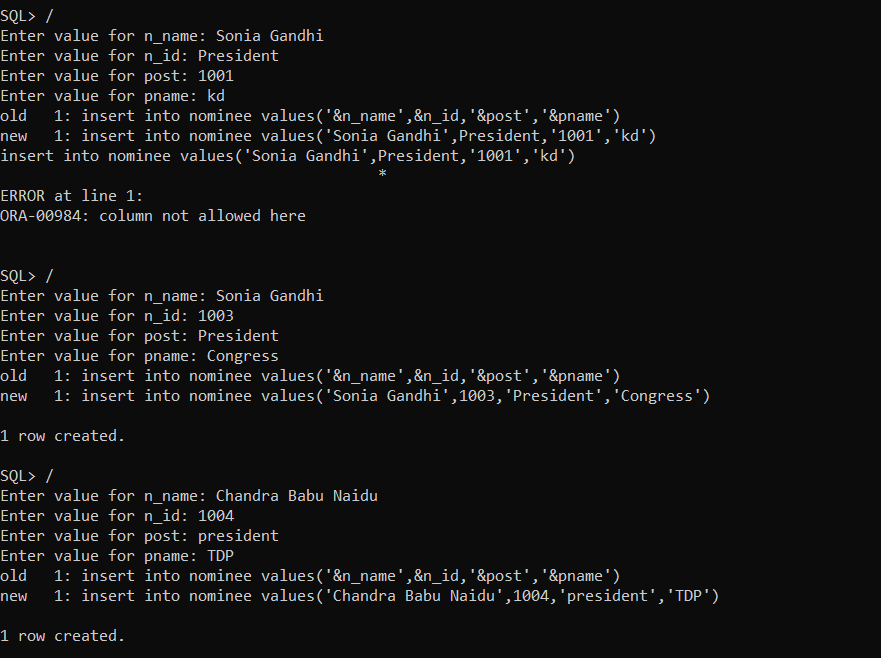
****

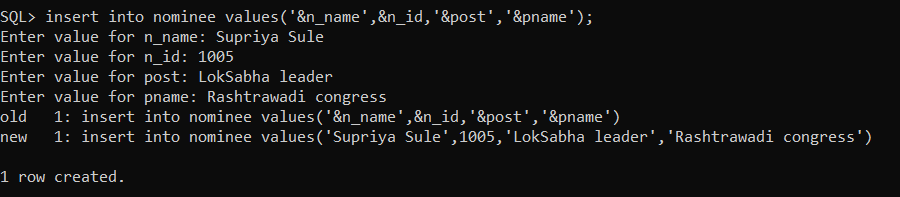
****

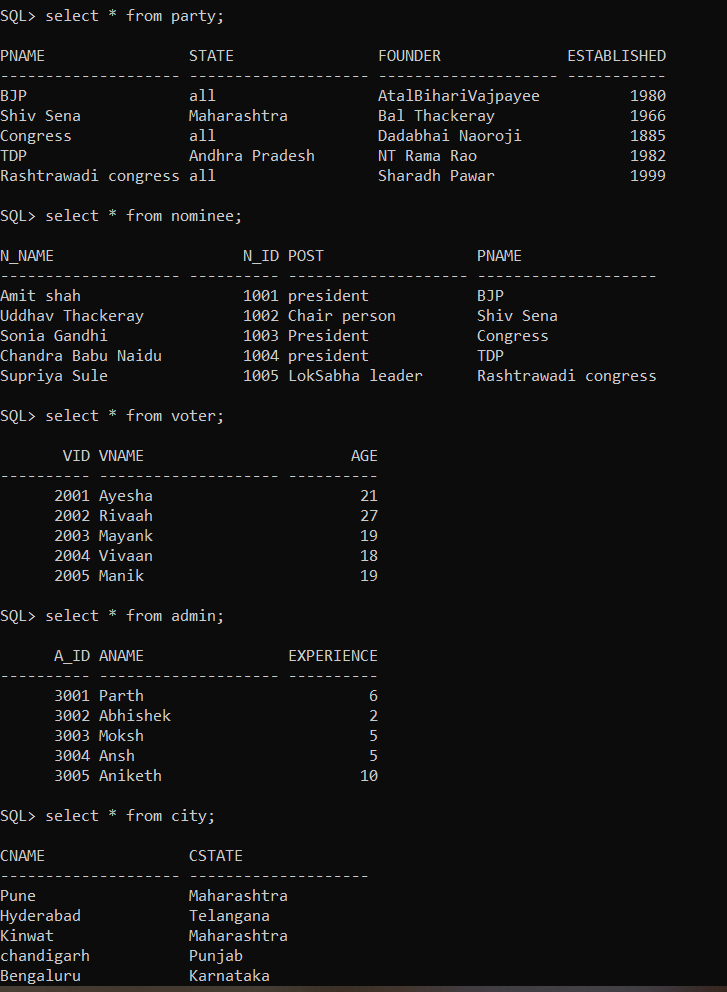
****

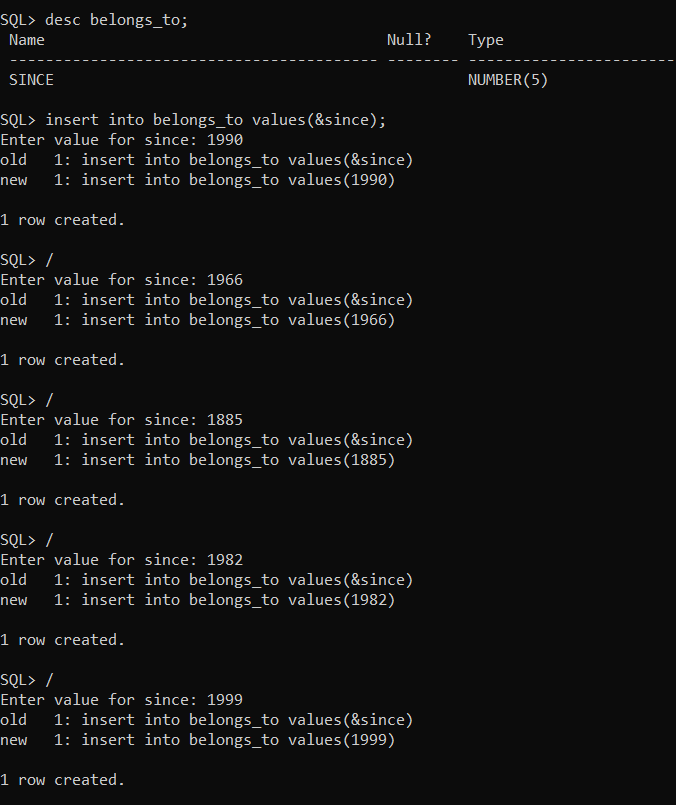
**DML Commands:**

****

****

****

****



**Implementation**

* Front end programs:

1. Insert a voter:

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class insertVoter extends Panel{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

Button insertVoterButton;

TextField vid,vname,age;

TextArea errorText;

Connection connection;

Statement statement;

public insertVoter()

{

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

}

catch(Exception e)

{

System.err.println("Unable to find and load driver");

System.exit(1);

}

connectToDB();

}

public void connectToDB()

{

try

{

connection=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","sreya","sreya");

statement=connection.createStatement();

}

catch(SQLException connectException)

{

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());

System.exit(1);

}

}

public void buildGUI()

{

insertVoterButton = new Button("Submit");

insertVoterButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

String query="INSERT INTO voter VALUES("+vid.getText()+","+"'"+vname.getText()+"',"+""+age.getText()+")";

int i=statement.executeUpdate(query);

errorText.append("\nInserted"+i+"rows.");

}

catch(SQLException insertException)

{

displaySQLErrors(insertException);

}

}

}

);

vid=new TextField(20);

vname=new TextField(20);

age=new TextField(20);

errorText=new TextArea(10,80);

errorText.setEditable(false);

Panel first=new Panel();

first.setLayout(new GridLayout(6,2));

first.add(new Label("VOTER ID:"));

first.add(vid);

first.add(new Label("VOTER NAME:"));

first.add(vname);

first.add(new Label("VOTER AGE:"));

first.add(age);

first.setBounds(300,150,350,150);

Panel second=new Panel(new GridLayout(6,1));

second.add(insertVoterButton);

second.setBounds(200,350,350,150);

Panel third=new Panel();

third.add(errorText);

third.setBounds(200,500,700,600);

setLayout(null);

add(first);

add(second);

add(third);

setSize(500,600);

setVisible(true);

}

private void displaySQLErrors(SQLException e)

{

errorText.append("\nSQLException:"+e.getMessage()+"\n");

errorText.append("SQLState: "+e.getSQLState()+"\n");

errorText.append("VoterError: "+e.getErrorCode()+"\n");

}

public static void main(String[] args)

{

insertVoter voter=new insertVoter();

voter.buildGUI();

}

}

1. Delete a voter:

import java.awt.Button;

import java.awt.FlowLayout;

import java.awt.GridLayout;

import java.awt.Label;

import java.awt.List;

import java.awt.Panel;

import java.awt.TextArea;

import java.awt.TextField;

import java.awt.event.\*;

import java.sql.\*;

public class deleteVoter extends Panel

{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

Button deleteVoterButton;

List voterIDList;

TextField vid, vname,age;

TextArea errorText;

Connection connection;

Statement statement;

ResultSet rs;

public deleteVoter()

{

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

}

catch (Exception e)

{

System.err.println("Unable to find and load driver");

System.exit(1);

}

connectToDB();

}

public void connectToDB()

{

try

{

connection =DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","sreya","sreya");

statement = connection.createStatement();

}

catch (SQLException connectException)

{

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());

System.exit(1);

}

}

private void loadProvider()

{

try

{

rs = statement.executeQuery("SELECT \* FROM voter");

while (rs.next())

{

voterIDList.add(rs.getString("VID"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void buildGUI()

{

voterIDList = new List(10);

loadProvider();

add(voterIDList);

voterIDList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM voter");

while (rs.next())

{

if(rs.getString("VID").equals(voterIDList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

vid.setText(rs.getString("VID"));

vname.setText(rs.getString("VNAME"));

age.setText(rs.getString("AGE"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

deleteVoterButton = new Button("Delete voter");

deleteVoterButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM voter WHERE VID = "+ voterIDList.getSelectedItem());

errorText.append("\nDeleted " + i + " rows");

vid.setText(null);

vname.setText(null);

age.setText(null);

voterIDList.removeAll();

loadProvider();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

vid = new TextField(15);

vname = new TextField(15);

age = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5,2));

first.add(new Label("Voter ID:"));

first.add(vid);

vid.setEditable(false);

first.add(new Label("Voter Name:"));

first.add(vname);

vname.setEditable(false);

first.add(new Label("Voter Age:"));

first.add(age);

age.setEditable(false);

Panel second = new Panel(new GridLayout(5, 1));

second.add(deleteVoterButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

private void displaySQLErrors(SQLException e)

{

errorText.append("\nSQLException: " + e.getMessage() + "\n");

errorText.append("SQLState: " + e.getSQLState() + "\n");

errorText.append("VendorError: " + e.getErrorCode() + "\n");

}

public static void main(String[] args)

{

deleteVoter del = new deleteVoter();

del.buildGUI();

}

}

1. Update a voter:

import java.awt.event.\*;

import java.sql.\*;

public class updateVoter extends Panel

{

Button updateVoterButton;

List voterIDList;

TextField vidText,vnameText, vageText;

TextArea errorText;

Connection connection;

Statement statement;

ResultSet rs;

public updateVoter()

{

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

}

catch (Exception e)

{

System.err.println('Unable to find and load driver");

System.exit(1);

}

connectToDB();

}

public void connectToDB()

{

try

{

connection =DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","sreya","sreya");

statement = connection.createStatement();

}

catch (SQLException connectException)

{

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());

System.exit(1);

}

}

private void loadProvider()

{

try

{

rs = statement.executeQuery("SELECT VID FROM voter");

while (rs.next())

{

voterIDList.add(rs.getString("VID"));

}

}

catch (SQLException e)

private void loadProvider()

{

try

{

rs = statement.executeQuery("SELECT VID FROM voter");

while (rs.next())

{

voterIDList.add(rs.getString("VID"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void buildGUI()

{

voterIDList = new List(10);

loadProvider();

add(voterIDList);

voterIDList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM voter");

while (rs.next())

{

if (rs.getString("VID").equals(voterIDList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

vidText.setText(rs.getString("VID"));

vnameText.setText(rs.getString("VNAME"));

vageText.setText(rs.getString("VAGE"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

updateVoterButton = new Button(“Modify");

updateVoterButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE voter"+ "SET vname=''" + vnameText.getText() + ","+ "vage=''" + vageText.getText() +" WHERE VID ="+ voterIDList.getSelectedItem());

errorText.append("\nUpdated " + i + " rows ");

voterIDList.removeAll();

loadProvider();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

vidText = new TextField(15);

vidText.setEditable(false);

vnameText = new TextField(15);

vageText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(5, 2));

first.add(new Label("Voter ID:"));

first.add(vidText);

first.add(new Label("Voter Name:"));

first.add(vnameText);

first.add(new Label("Voter Age:"));

first.add(vageText);

Panel second = new Panel(new GridLayout(5, 1));

second.add(updateVoterButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

private void displaySQLErrors(SQLException e)

{

errorText.append("\nSQLException: " + e.getMessage() + "\n");

errorText.append("SQLState: " + e.getSQLState() + "\n");

errorText.append("VendorError: " + e.getErrorCode() + "\n");

}

public static void main(String[] args)

{

updateVoter uv = new updateVoter();

uv.buildGUI();

}

}

1. Main method

import java.awt.\*;

import java.awt.event.\*;

class OnlinePollingSystem extends Frame implements ActionListener

{

String msg = "";

Label ll,l2;

CardLayout cardLO;

insertVoter inv;

updateVoter upv;

deleteVoter delv;

insertAdmin ina;

updateAdmin upa;

deleteAdmin dela;

insertNominee inn;

updateNominee upn;

deleteNominee deln;

insertCity inc;

updateCity upc;

deleteCity delc;

insertParty inp;

updateParty upp;

deleteParty delp;

Panel home,welcome;

OnlinePollingSystem()

{

cardLO = new CardLayout();

home = new Panel();

home.setLayout(cardLO);

ll = new Label();

l2 =new Label();

ll.setAlignment(Label.CENTER);

l2.setAlignment(Label.CENTER);

ll.setText("Welcome to Online Online Polling System");

l2.setText("All @rights are reserved");

welcome = new Panel();

welcome.add(ll);

welcome.add(l2);

inv = new insertVoter();

inv.buildGUI();

upv = new updateVoter();

upv.buildGUI();

delv = new deleteVoter();

delv.buildGUI();

ina = new insertAdmin();

ina.buildGUI();

upa= new updateAdmin();

upa.buildGUI();

dela = new deleteAdmin();

dela.buildGUI();

inn = new insertNominee();

inn.buildGUI();

upn = new updateNominee();

upn.buildGUI();

deln = new deleteNominee();

deln.buildGUI();

inc = new insertCity();

inc.buildGUI();

upc = new updateCity();

upc.buildGUI();

delc = new deleteCity();

delc.buildGUI();

inp = new insertParty();

inp.buildGUI();

upp = new updateParty();

upp.buildGUI();

delp = new deleteParty();

delp.buildGUI();

//add all the panels to the home panel which has a cardlayout

home.add(welcome, "Welcome");

home.add(inv, "insertVoter");

home.add(upv, "updateVoter");

home.add(delv, "deleteVoter");

home.add(ina, "insertAdmin");

home.add(upa, "updateAdmin");

home.add(dela, "deleteAdmin");

home.add(inn, "insertNominee");

home.add(upn, "updateNominee");

home.add(deln, "deleteNominee");

home.add(inc, "insertCity");

home.add(upc, "updateCity");

home.add(delc, "deleteCity");

home.add(inp, "insertParty");

home.add(upp, "updateParty");

home.add(delp, "deleteParty");

// add home panel to main frame

add(home);

// create menu bar and add it to frame

MenuBar mbar = new MenuBar();

setMenuBar(mbar);

// create the menu items and add it to Menu

Menu voter = new Menu("Voter");

MenuItem item1, item2, item3;

voter.add(item1 = new MenuItem("Insert Voter"));

voter.add(item2 = new MenuItem("View Voter"));

voter.add(item3 = new MenuItem("Delete Voter"));

mbar.add(voter);

Menu nominee = new Menu("Nominee");

MenuItem item4, item5, item6;

nominee.add(item4 = new MenuItem("Insert Nominee"));

nominee.add(item5 = new MenuItem("View Nominee"));

nominee.add(item6 = new MenuItem("Delete Nominee"));

mbar.add(nominee);

Menu admin = new Menu("Admin");

MenuItem item7, item8, item9;

admin.add(item7 = new MenuItem("Insert Admin"));

admin.add(item8 = new MenuItem("View Admin"));

admin.add(item9 = new MenuItem("Delete Admin"));

mbar.add(admin);

Menu party = new Menu("Party");

MenuItem item10, item11, item12;

party.add(item10 = new MenuItem("Insert Party"));

party.add(item11 = new MenuItem("View Party"));

party.add(item12 = new MenuItem("Delete Party"));

mbar.add(party);

Menu city = new Menu("City");

MenuItem item13, item14, item15;

city.add(item13 = new MenuItem("Insert City"));

city.add(item14 = new MenuItem("View City"));

city.add(item15 = new MenuItem("Delete City"));

mbar.add(city);

// register listeners

item1.addActionListener(this);

item2.addActionListener(this);

item3.addActionListener(this);

item4.addActionListener(this);

item5.addActionListener(this);

item6.addActionListener(this);

item7.addActionListener(this);

item8.addActionListener(this);

item9.addActionListener(this);

item10.addActionListener(this);

item11.addActionListener(this);

item12.addActionListener(this);

item13.addActionListener(this);

item14.addActionListener(this);

item15.addActionListener(this);

addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent we)

{

System.exit(0);

}

});

//Frame properties

setTitle("Online Polling System");

Color clr = new Color(50, 150, 100);

setBackground(clr);

setFont(new Font("SansSerif", Font.CENTER\_BASELINE, 18));

setSize(900, 1000);

setVisible(true);

}

public void actionPerformed(ActionEvent ae)

{

String arg = ae.getActionCommand();

if(arg.equals("Insert Voter"))

{

cardLO.show(home, "insertVoter");

}

else if(arg.equals("View Voter"))

{

cardLO.show(home, "updateVoter");

}

else if(arg.equals("Delete Voter"))

{

cardLO.show(home, "deleteVoter");

}

else if(arg.equals("Insert Nominee"))

{

cardLO.show(home, "insertNominee");

}

else if(arg.equals("View Nominee"))

{

cardLO.show(home, "updateNominee");

}

else if(arg.equals("Delete Nominee"))

{

cardLO.show(home, "deleteNominee");

}

else if(arg.equals("Insert Admin"))

{

cardLO.show(home, "insertAdmin");

}

else if(arg.equals("View Admin"))

{

cardLO.show(home, "updateAdmin");

}

else if(arg.equals("Delete Admin"))

{

cardLO.show(home, "deleteAdmin");

}

else if(arg.equals("Insert Party"))

{

cardLO.show(home, "insertParty");

}

else if(arg.equals("View Party"))

{

cardLO.show(home, "updateParty");

}

else if(arg.equals("Delete Party"))

{

cardLO.show(home, "deleteParty");

}

else if(arg.equals("Insert City"))

{

cardLO.show(home, "insertCity");

}

else if(arg.equals("View City"))

{

cardLO.show(home, "updateCity");

}

else

{

cardLO.show(home, "deleteCity");

}

}

public static void main(String ... args)

{

new OnlinePollingSystem();

}

}

Connectivity with the Database:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

Block of code for JAVA- SQL connectivity with JDBC:

public void connectToDB()

{

try

{

connection=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","sreya","sreya");

statement=connection.createStatement();

}

catch(SQLException connectException)

{

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());

System.exit(1);

}

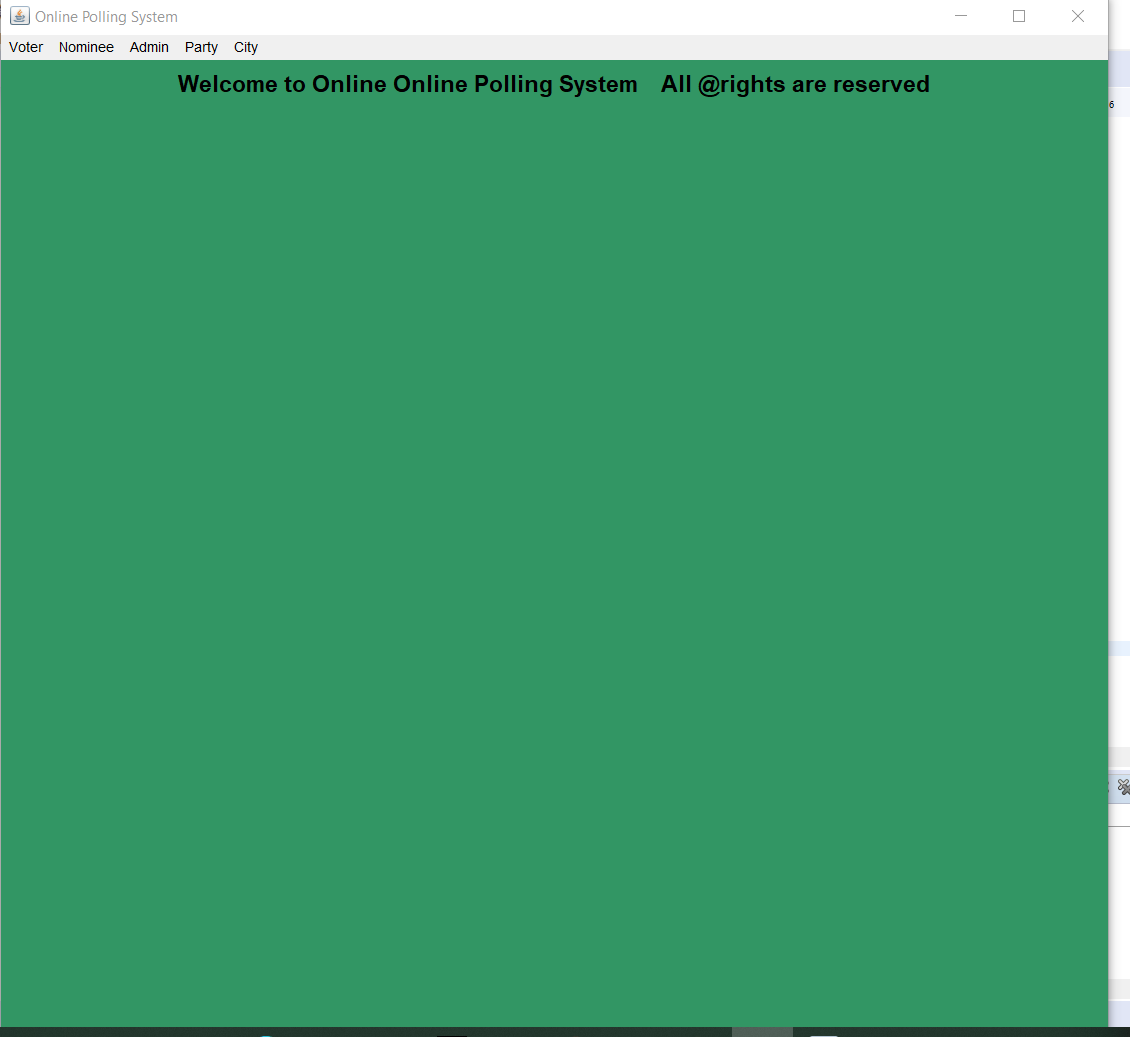
}

GITHUB LINK:

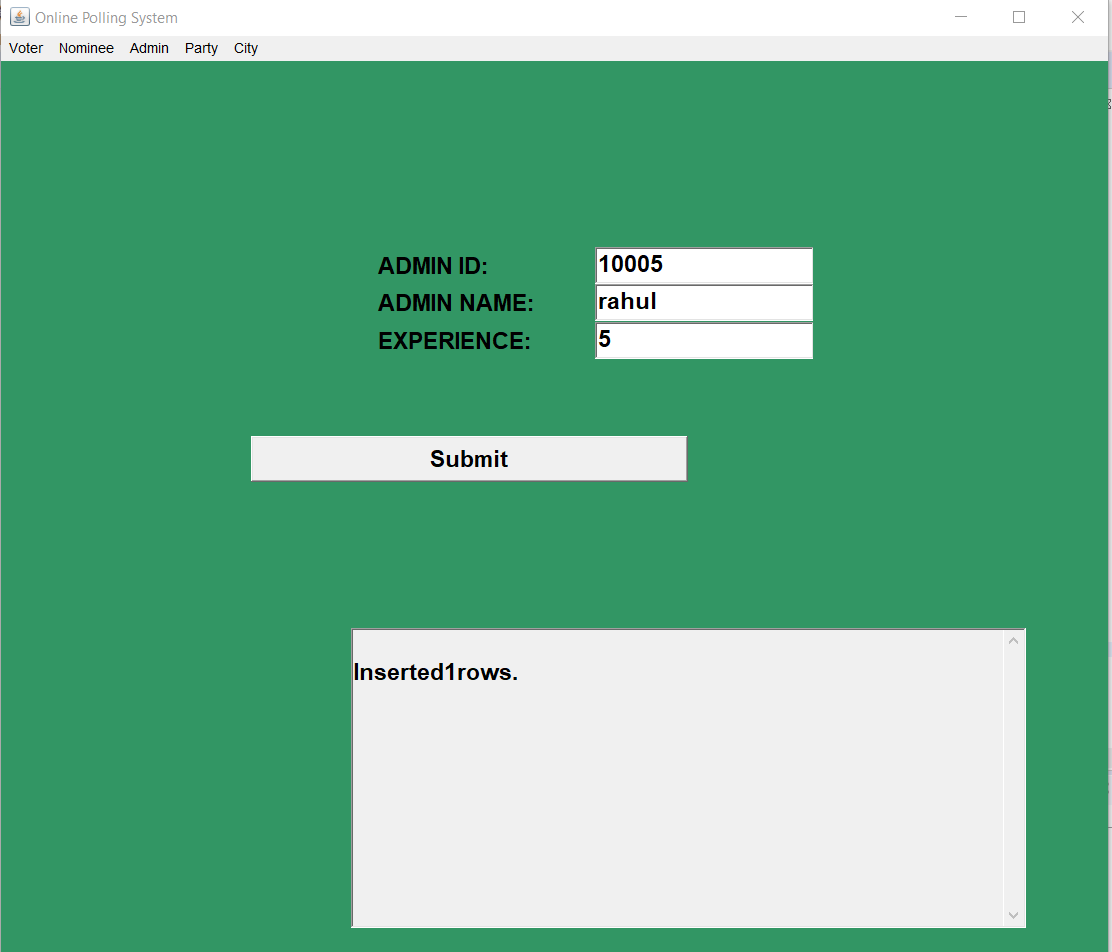
https://github.com/sreya177/dbms-assignment-2.git

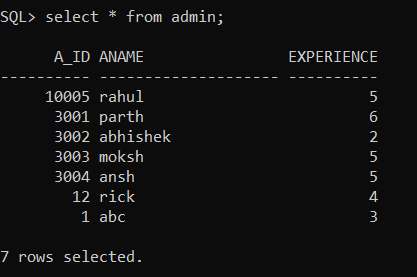
**TESTING**

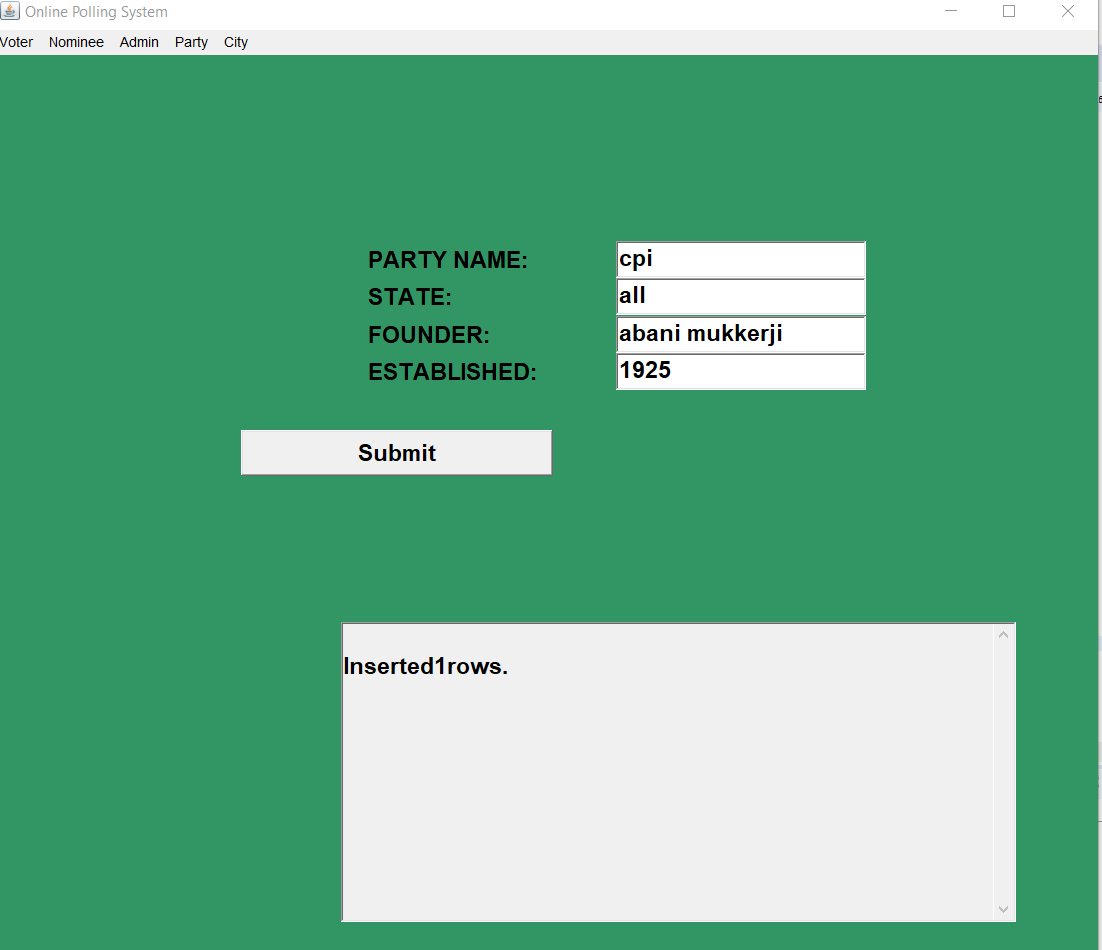
The program runs for execution of three basic operations of insertion, update and delete on 5 different table. Along with this, it also has a output column which gives the information about how many rows have been edited. Errors, syntactical or exceptional will be shown if occurred.

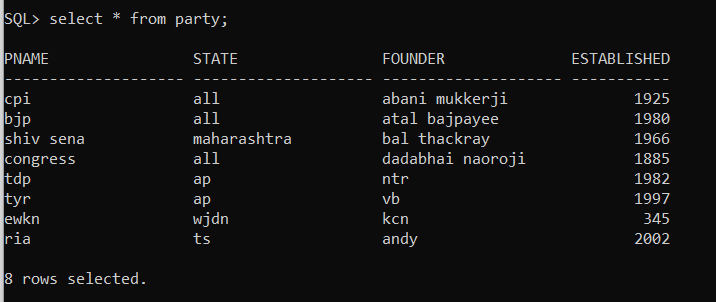
HOME PAGE:

INSERT:

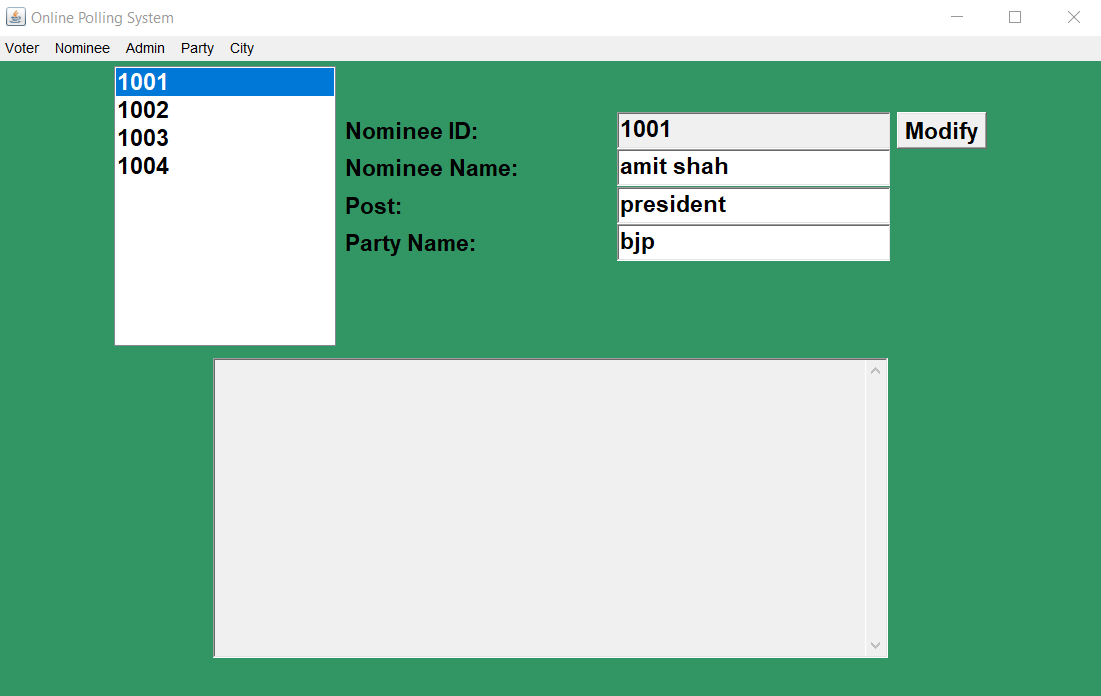


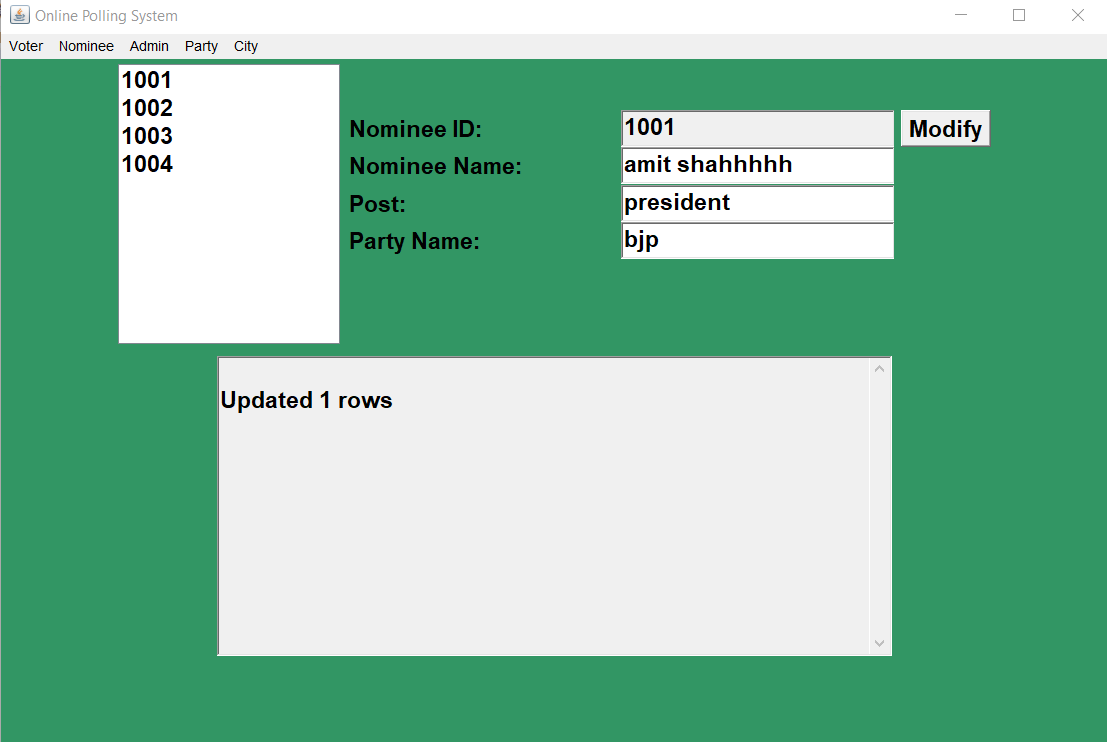


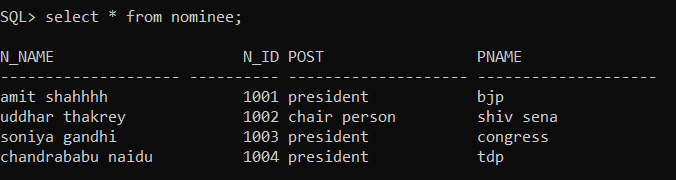


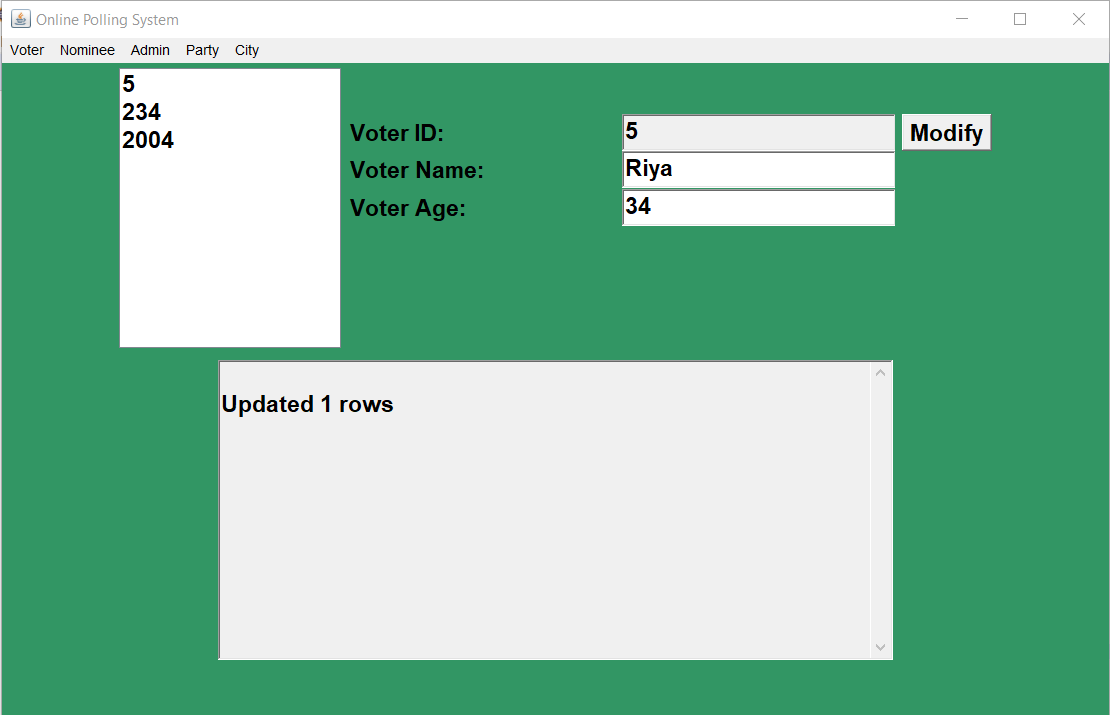


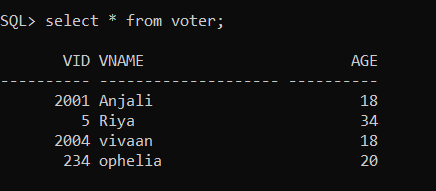
UPDATE:



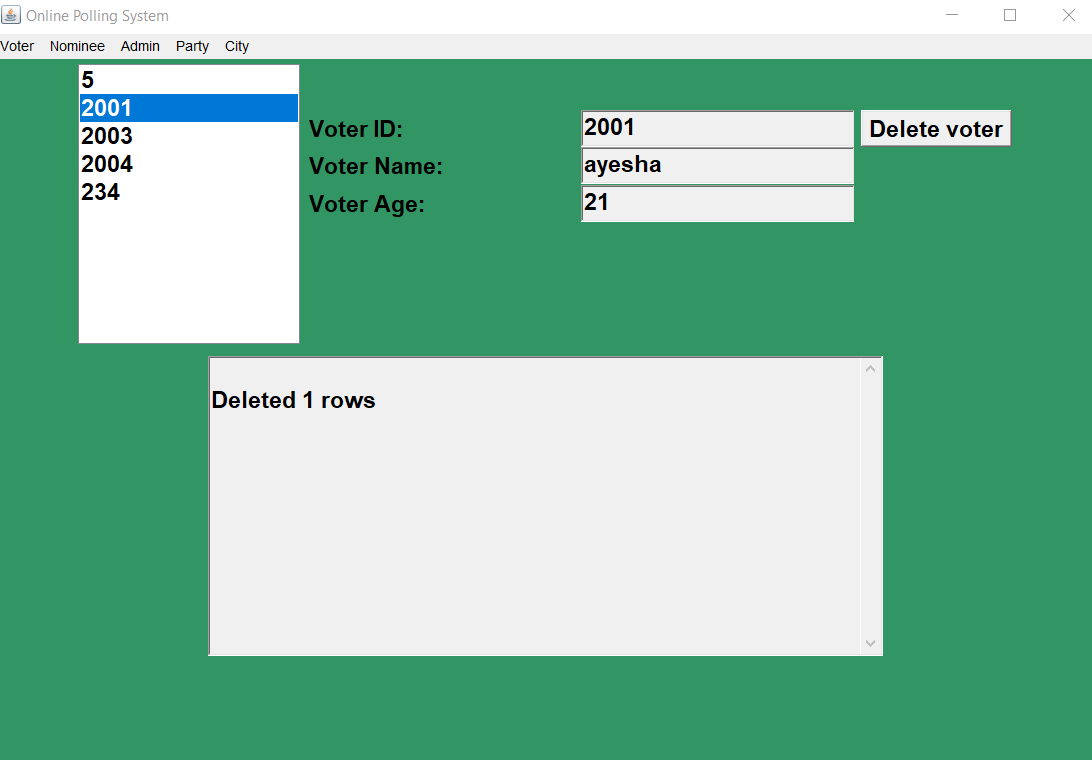


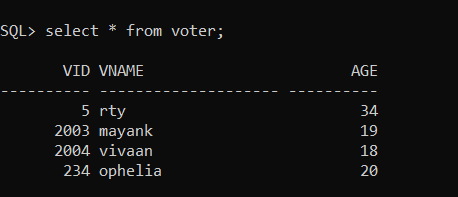


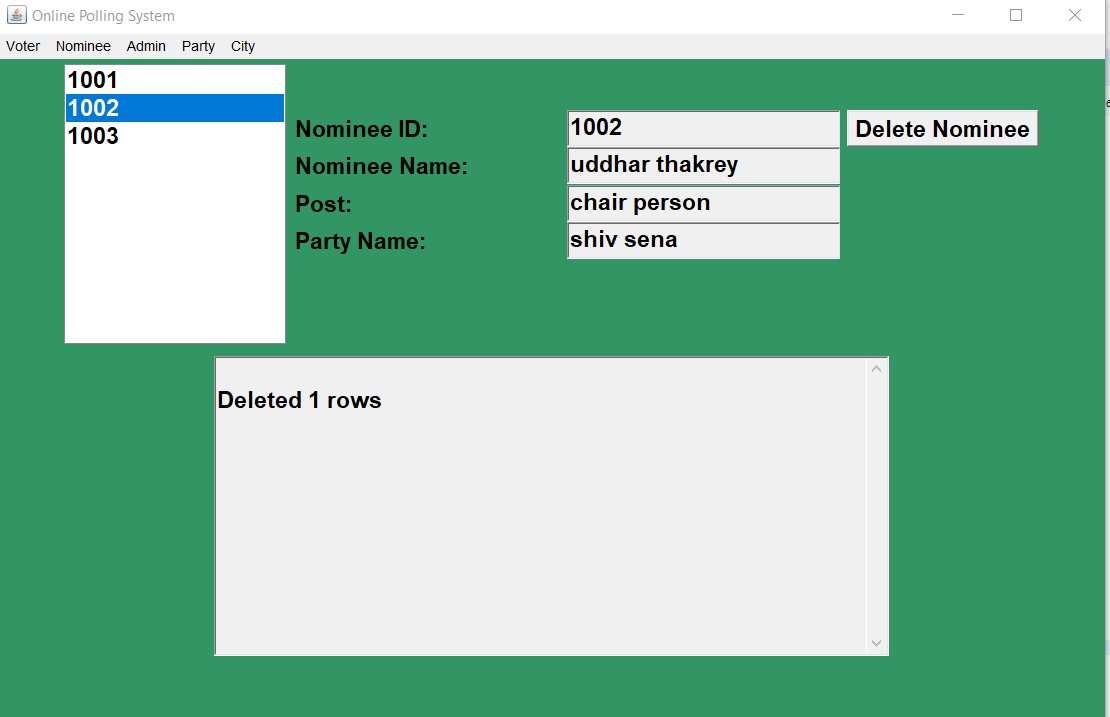


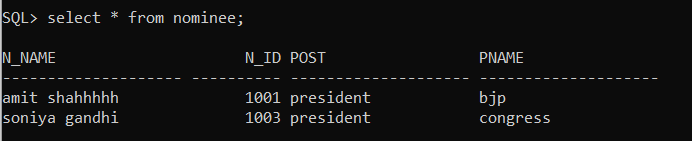


DELETE:









**RESULTS**

The DML commands, Insert, update and delete for one of the tables in given below:

For voters table: (in java, as per the application)

Insert: INSERT INTO voter VALUES(" +vidText.getText() + "," + "'" + vnameText.getText() + "',"+"'"+vageText.getText()+")";

Update: "UPDATE voter"+ "SET vname=''" + vnameText.getText() + ","+ "vage=''" + vageText.getText() +" WHERE VID ="+ voterIDList.getSelectedItem();

Delete: DELETE FROM voter WHERE VID = "+ providerIDList.getSelectedItem();

**DISCUSSIONS**

The application “Automated polling system” helps to caste votes to a party’s nominee as per a voter’s wish, online. Since the practise in person to person, on individual basis. A voter can cast a vote only once he/she enters his/her voter ID. This would help in removing multiple time voting.

The volunteers or admins can give their names as well through the same application. The voting details, timing cities they are taking place; everything is provided at a just click away.

The votes casted by the voter and highly secure and maintained anonymously. The data entered is stored into the database immediately to avoid loss or tampering or data.

The update choice is only provided to corresponding city coordinator who is given the access to the database.

The future works associated with the project can be imposing biometric face authentication to get an access to the voting system. Linking voter card with voters phone number so that he/she can be provided with a one time password through SMS at the time of voting. This OTP can be used as an alternative for face auth.

**REFERENCES**

1. <https://www.ijitee.org/wp-content/uploads/papers/v8i11/J98880881019.pdf>
2. <https://en.wikipedia.org/wiki/Electronic_voting>
3. https://github.com/sreya177/dbms-assignment-1.git