JAVA SWINGS BASED- IPL DISPLAY SYSTEM

- SQL CONNECTIVITY USING JDBC

A

Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

Ananya. M <1602-19-737-067>

Under the guidance of Ms B. Leelavathy

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

BONAFIDE CERTIFICATE

This is to certify that the project report titled "**IPL display system**" project work of Miss Ananya. M bearing Roll no:1602-19-737-067 who carried out this project under my supervision in the IV semester for the academic year 2020-2021.

Signature
Internal Examiner

Signature
External Examiner

ABSTRACT

To create an IPL display system, we will require a total of 6 table. One- for the team details, two- for the player details, three- for the sponsors, four- for the match venue details, five- for the city it represents and six- for the owner who owns the team. The basic attributes would be the name and ID of any entity besides this, descriptive attributes can also be present. Entity names can have a domain type varchar2 where the ID can be number. The relationships between various entity sets helps in creating an IPL display system. Few of these entities have primary keys which turn out to be the foreign keys in the rest.

REQUIREMENTS FOR IPL DISPLAY SYSTEM:

- Functional Requirements:
    1. User-Interface: The system shall provide an easy-to-use user-interface. Any detail about the matches or players can be fetched without any trouble.
    2. Detailed data: All the data that is available in the display system can be accessed by the users. A detailed information is provided according to the user's need.
    3. Match Status: The statistics of the matches is recorded in the database which makes it easy for the users to plan for the next match strategies.
    4. Player intake: A detailed elaboration of the players and captains of the variety of the teams are recorded in the database.
    5. Mobility: Any changes in the display system regarding the information can be made without any major changes occurring in the system.

- Security Requirements:
    1. Player Authenticity: Ensures that the player has his own unique identity and permits only if that identity is provided to access sensitive information.
    2. System Integrity: Ensure that the system cannot be re-configured during operation.
    3. Data Integrity: Ensure that each detail is recorded as intended and cannot be tampered with in any manner, once recorded.
    4. Secrecy / Privacy: No one should be able to determine any team's strategy.
    5. Reliability: IPL display system should work robustly, without loss of any details, even in the face of numerous failures. The database shall be developed in a manner that ensures there is no malicious code or bugs.

- Through the project:
  The main goal to be achieved through this project was to provide an opportunity to display the details of various players, teams, owners, sponsors, matches, and cities taking part in the IPL. The project also ensures that the information that are recorded are pretty much confidential and are provided only if user has access to this database. SQL particular player, team, owner, sponsor, match, 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 players, teams, owners, sponsors, matches 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, Java Scripts 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.

AIM AND PRIORITY OF THE PROJECT

To create a Java GUI based student assignments tracker which takes the values like: user id, password for login and stores details of all students, faculty and student assignments their marks in those and staff can assign assignments to students. These values are to be updated in the database using JDBC connectivity.

ARCHITECTURE AND TECHNOLOGY

Software used: Java Eclipse, Oracle 11g Database, Java SE version 13, SQL*Plus.

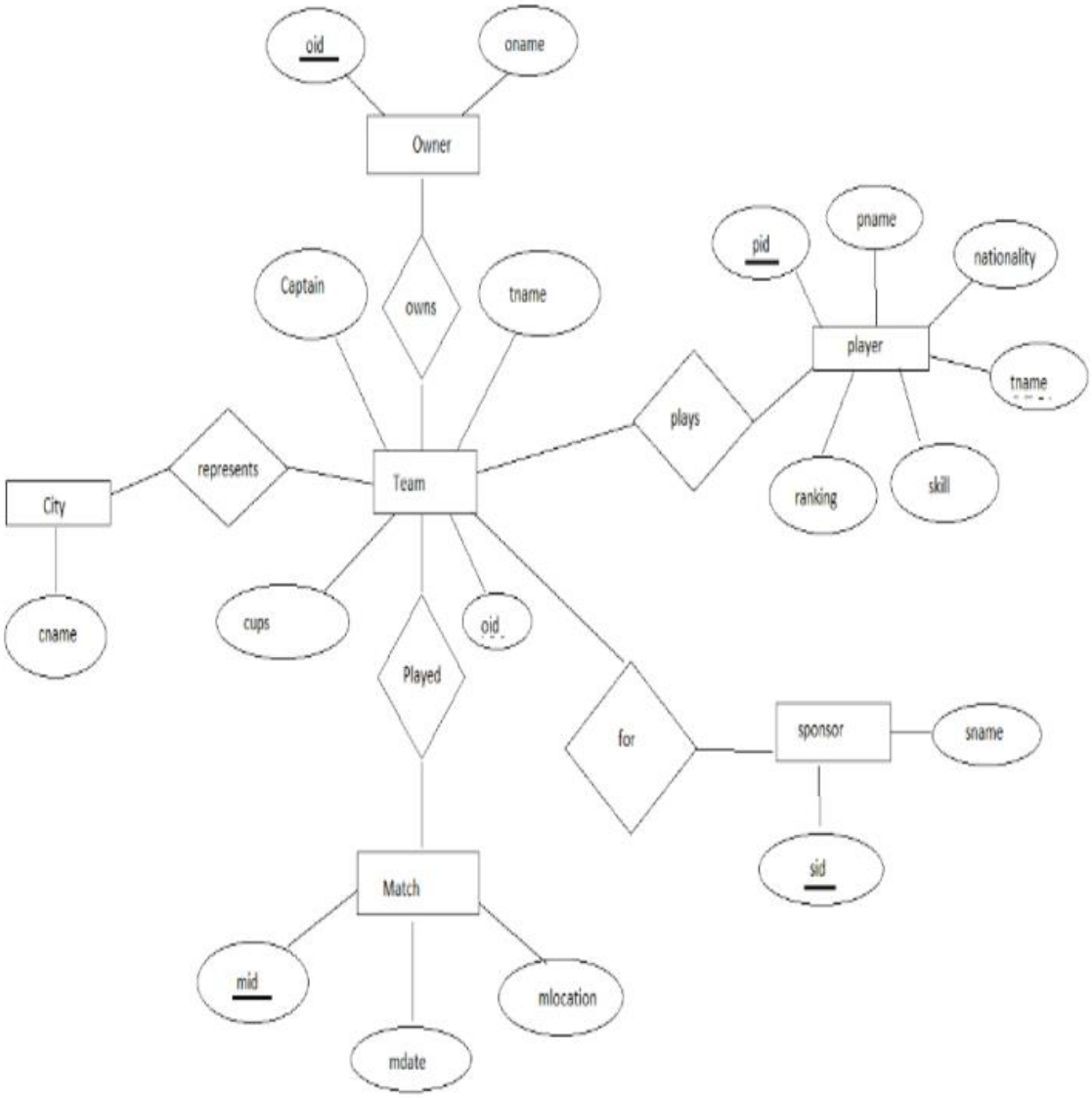
Java SWINGS: Java SWINGS is an API to develop GUI or window-based applications in java. Java SWING components are platform-independent. It is lightweight. The javax.swing package provides classes for SWING API such as JTextField, JLabel, JTextArea, JRadioButton, JCheckBox, JChoice, JList etc.

SQL:

Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's **Relational** model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

- Design:

  <u>ER diagram</u>



- DATABASE DESIGN:

  <u>CONTENT:</u>

  - Abstract
  - ER Diagram
  - Logical database design – DDL commands

- Enforcing primary and foreign keys.
- DML operation and outputs.

ABSTRACT

To create an IPL database, we will require a total of 6 table. One- for the team details, two- for the player details, three- for the sponsors, four- for the match venue details, five- for the city it represents and six- for the owner who owns the team. The basic attributes would be the name and ID of any entity besides this, descriptive attributes can also be present. Entity names can have a domain type varchar2 where the ID can be number. The relationships between various entity sets helps in creating an IPL database. Few of these entities have primary keys which turn out to be the foreign keys in the rest. The six tables are:

- Player
- Team
- Owner
- Sponsor
- Match
- City

LIST OF REQUIREMENTS:

List of Players.
Details of players, teams and owners.
Details of sponsors.
Location of the match being played.

CONSTRAINTS APPLIED:

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

1)oid in the owner table.

2)pid in the player table.

3)mid in the match table.

4)sid in the sponsor table.

The attributes oid and tname act as foreign keys in the team table and player table respectively.

DDL Commands:

For creating player table:

Query: create table player(
pid number(5) primary key,
pname varchar2(20),
ranking number(5),
nationality varchar2(20),
skill varchar2(20),
tname varchar2(20));

For creating Team table:

Query: create table team(
captain varchar2(20),
tname varchar2(20),
cups number(5));

For creating Owner table:

Query: create table owner(
oid number(5) primary key,
oname varchar2(20));

For creating Sponsor table:

Query: create table sponsor(
sid number(5) primary key,
sname varchar2(20));

For creating Match table:

Query: create table match(
mid number(5) primary key,
mdate date,
mlocation varchar2(20));

For creating Team table:

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

```
SQL> create table owner(
  2  oid number(5) primary key,
  3  oname varchar2(20));

Table created.
```

```
SQL> conn ananya/ananya;
Connected.
SQL> create table team(
  2  captain varchar2(20),
  3  tname varchar2(20),
  4  cups number(5));

Table created.

SQL> create table city(
  2  cname varchar2(20));

Table created.

SQL> create table match(
  2  mid number(5) primary key,
  3  mdate date,
  4  mlocation varchar2(20));

Table created.

SQL> create table sponsor(
  2  sid number(5) primary key,
  3  sname varchar2(20));

Table created.

SQL> desc player;
ERROR:
ORA-04043: object player does not exist


SQL> create table player(
  2  pid number(5) primary key,
  3  pname varchar2(20),
  4  ranking number(5),
  5  nationality varchar2(20),
  6  skill varchar2(20),
  7  tname varchar2(20));

Table created.

SQL> _
```

```
SQL> desc player;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PID                                       NOT NULL NUMBER(5)
 PNAME                                              VARCHAR2(20)
 RANKING                                            NUMBER(5)
 NATIONALITY                                        VARCHAR2(20)
 SKILL                                              VARCHAR2(20)
 TNAME                                              VARCHAR2(20)

SQL> desc team;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CAPTIAN                                            VARCHAR2(20)
 TNAME                                     NOT NULL VARCHAR2(20)
 CUPS                                               NUMBER(5)
 OID                                                NUMBER(5)

SQL> desc owner;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OID                                       NOT NULL NUMBER(5)
 ONAME                                              VARCHAR2(20)

SQL> desc sponsor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SID                                       NOT NULL NUMBER(5)
 SNAME                                              VARCHAR2(20)

SQL> desc match;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MID                                       NOT NULL NUMBER(5)
 MDATE                                              DATE
 MLOCATION                                          VARCHAR2(20)

SQL> desc city;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CNAME                                              VARCHAR2(20)
```

DML Commands:

```
SQL> Spool E:\assignment.txt
SQL> desc team;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CAPTAIN                                            VARCHAR2(20)
 TNAME                                              VARCHAR2(20)
 CUPS                                               NUMBER(5)
 OID                                       NOT NULL NUMBER(5)

SQL> insert into owner values(&oid, '&oname');
Enter value for oid: 101
Enter value for oname: srk
old   1: insert into owner values(&oid, '&oname')
new   1: insert into owner values(101, 'srk')
insert into owner values(101, 'srk')
*
ERROR at line 1:
ORA-00001: unique constraint (ANANYA.SYS_C007054) violated


SQL> insert into owner values(&oid, '&oname');
Enter value for oid: 1001
Enter value for oname: srk
old   1: insert into owner values(&oid, '&oname')
new   1: insert into owner values(1001, 'srk')

1 row created.

SQL> insert into owner values(&oid, '&oname');
Enter value for oid: 1002
Enter value for oname: nita ambani
old   1: insert into owner values(&oid, '&oname')
new   1: insert into owner values(1002, 'nita ambani')

1 row created.

SQL> insert into owner values(&oid, '&oname');
Enter value for oid: 1003
Enter value for oname: vijay mallya
old   1: insert into owner values(&oid, '&oname')
new   1: insert into owner values(1003, 'vijay mallya')

1 row created.

SQL> insert into owner values(&oid, '&oname');
Enter value for oid: 1004
Enter value for oname: kalanithi maran
old   1: insert into owner values(&oid, '&oname')
new   1: insert into owner values(1004, 'kalanithi maran')
```

```
SQL> insert into match values(&mid,'&mdate','&mlacation');
Enter value for mid: 2001
Enter value for mdate: 23-apr-21
Enter value for mlacation: chennai
old   1: insert into match values(&mid,'&mdate','&mlacation')
new   1: insert into match values(2001,'23-apr-21','chennai')

1 row created.

SQL> /
Enter value for mid: 2002
Enter value for mdate: 24-apr-21
Enter value for mlacation: mumbai
old   1: insert into match values(&mid,'&mdate','&mlacation')
new   1: insert into match values(2002,'24-apr-21','mumbai')

1 row created.

SQL> /
Enter value for mid: 2003
Enter value for mdate: 28-apr-21
Enter value for mlacation: bengaluru
old   1: insert into match values(&mid,'&mdate','&mlacation')
new   1: insert into match values(2003,'28-apr-21','bengaluru')

1 row created.

SQL> 2004
SP2-0226: Invalid line number
SQL> /
Enter value for mid: 2004
Enter value for mdate: 29-apr-21
Enter value for mlacation: hyderabad
old   1: insert into match values(&mid,'&mdate','&mlacation')
new   1: insert into match values(2004,'29-apr-21','hyderabad')

1 row created.

SQL> /
Enter value for mid: 2005
Enter value for mdate: 27-apr-21
Enter value for mlacation: kolkata
old   1: insert into match values(&mid,'&mdate','&mlacation')
new   1: insert into match values(2005,'27-apr-21','kolkata')

1 row created.

SQL>
```

```
SQL> insert into sponsor values(&sid,'&sname');
Enter value for sid: 3001
Enter value for sname: nokia
old   1: insert into sponsor values(&sid,'&sname')
new   1: insert into sponsor values(3001,'nokia')

1 row created.

SQL> /
Enter value for sid: 3002
Enter value for sname: reliance
old   1: insert into sponsor values(&sid,'&sname')
new   1: insert into sponsor values(3002,'reliance')

1 row created.

SQL> /
Enter value for sid: 3003
Enter value for sname: suntv
old   1: insert into sponsor values(&sid,'&sname')
new   1: insert into sponsor values(3003,'suntv')

1 row created.

SQL> /
Enter value for sid: 3004
Enter value for sname: muthoot finance
old   1: insert into sponsor values(&sid,'&sname')
new   1: insert into sponsor values(3004,'muthoot finance')

1 row created.

SQL> 3005
SP2-0226: Invalid line number
SQL> /
Enter value for sid: 3005
Enter value for sname: britania
old   1: insert into sponsor values(&sid,'&sname')
new   1: insert into sponsor values(3005,'britania')

1 row created.

SQL> insert into city values('&cname');
Enter value for cname: mumbai
old   1: insert into city values('&cname')
new   1: insert into city values('mumbai')

1 row created.
```

```
SQL> /
Enter value for cname: chennai
old   1: insert into city values('&cname')
new   1: insert into city values('chennai')

1 row created.

SQL> /
Enter value for cname: hyderabad
old   1: insert into city values('&cname')
new   1: insert into city values('hyderabad')

1 row created.

SQL> /
Enter value for cname: kolkata
old   1: insert into city values('&cname')
new   1: insert into city values('kolkata')

1 row created.

SQL> /
Enter value for cname: bengaluru
old   1: insert into city values('&cname')
new   1: insert into city values('bengaluru')

1 row created.

SQL> select * owner;
select * owner
         *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected


SQL> select * from owner;

       OID ONAME
---------- --------------------
       101 srk
       102 nita ambani
      1001 srk
      1002 nita ambani
      1003 vijay mallya
      1004 kalanithi maran
      1005 n srinivasan

7 rows selected.
```

```
SQL> insert into team values('&captain','&tname',&cups,&oid);
Enter value for captain: dinesh karthik
Enter value for tname: kkr
Enter value for cups: 2
Enter value for oid: 1001
old   1: insert into team values('&captain','&tname',&cups,&oid)
new   1: insert into team values('dinesh karthik','kkr',2,1001)

1 row created.

SQL> /
Enter value for captain: rohit sharma
Enter value for tname: mi
Enter value for cups: 4
Enter value for oid: 1002
old   1: insert into team values('&captain','&tname',&cups,&oid)
new   1: insert into team values('rohit sharma','mi',4,1002)

1 row created.

SQL> /
Enter value for captain: virat kohli
Enter value for tname: rcb
Enter value for cups: 0
Enter value for oid: 1003
old   1: insert into team values('&captain','&tname',&cups,&oid)
new   1: insert into team values('virat kohli','rcb',0,1003)

1 row created.

SQL> kane williamson
SP2-0734: unknown command beginning "kane willi..." - rest of line ignored.
SQL> /
Enter value for captain: kane williamson
Enter value for tname: srh
Enter value for cups: 1
Enter value for oid: 1004
old   1: insert into team values('&captain','&tname',&cups,&oid)
new   1: insert into team values('kane williamson','srh',1,1004)

1 row created.
```

IPL Display System

```
SQL> /
Enter value for captain: ms dhoni
Enter value for tname: csk
Enter value for cups: 3
Enter value for oid: 1005
old   1: insert into team values('&captain','&tname',&cups,&oid)
new   1: insert into team values('ms dhoni','csk',3,1005)

1 row created.

SQL> insert into player values(&pid,'&pname','&ranking','&natinality','&skill','&tname');
Enter value for pid: 4001
Enter value for pname: nitish rana
Enter value for ranking: 11
Enter value for natinality: indian
Enter value for skill: batsman
Enter value for tname: kkr
old   1: insert into player values(&pid,'&pname','&ranking','&natinality','&skill','&tname')
new   1: insert into player values(4001,'nitish rana','11','indian','batsman','kkr')

1 row created.

SQL> /
Enter value for pid: 4002
Enter value for pname: kieron pollard
Enter value for ranking: 77
Enter value for natinality: trinidadian
Enter value for skill: allrounder
Enter value for tname: mi
old   1: insert into player values(&pid,'&pname','&ranking','&natinality','&skill','&tname')
new   1: insert into player values(4002,'kieron pollard','77','trinidadian','allrounder','mi')

1 row created.

SQL> /
Enter value for pid: 4003
Enter value for pname: abd villers
Enter value for ranking: 43
Enter value for natinality: south african
Enter value for skill: batsman
Enter value for tname: rcb
old   1: insert into player values(&pid,'&pname','&ranking','&natinality','&skill','&tname')
new   1: insert into player values(4003,'abd villers','43','south african','batsman','rcb')

1 row created.
```

IPL Display System

```
SQL> /
Enter value for pid: 4004
Enter value for pname: david warner
Enter value for ranking: 7
Enter value for natinality: australian
Enter value for skill: batsman
Enter value for tname: srh
old   1: insert into player values(&pid,'&pname','&ranking','&natinality','&skill','&tname')
new   1: insert into player values(4004,'david warner','7','australian','batsman','srh')

1 row created.

SQL> /
Enter value for pid: 4005
Enter value for pname: ravindra jadeja
Enter value for ranking: 1
Enter value for natinality: indian
Enter value for skill: all rounder
Enter value for tname: csk
old   1: insert into player values(&pid,'&pname','&ranking','&natinality','&skill','&tname')
new   1: insert into player values(4005,'ravindra jadeja','1','indian','all rounder','csk')

1 row created.

SQL> select * from team;

CAPTAIN              TNAME                     CUPS        OID
-------------------- -------------------- ---------- ----------
dinesh karthik       kkr                           2       1001
rohit sharma         mi                            4       1002
virat kohli          rcb                           0       1003
kane williamson      srh                           1       1004
ms dhoni             csk                           3       1005

SQL> select * from owner;

       OID ONAME
---------- --------------------
       101 srk
       102 nita ambani
      1001 srk
      1002 nita ambani
      1003 vijay mallya
      1004 kalanithi maran
      1005 n srinivasan

7 rows selected.
```

IPL Display System

```
SQL> select * from player;

       PID PNAME                   RANKING NATIONALITY
---------- -------------------- ---------- --------------------
SKILL                TNAME
-------------------- --------------------
      4001 nitish rana                  11 indian
batsman              kkr

      4002 kieron pollard               77 trinidadian
allrounder           mi

      4003 abd villers                  43 south african
batsman              rcb


       PID PNAME                   RANKING NATIONALITY
---------- -------------------- ---------- --------------------
SKILL                TNAME
-------------------- --------------------
      4004 david warner                  7 australian
batsman              srh

      4005 ravindra jadeja               1 indian
all rounder          csk


SQL> select * from sponsor;

       SID SNAME
---------- --------------------
      3001 nokia
      3002 reliance
      3003 suntv
      3004 muthoot finance
      3005 britania
```

```
SQL> select * from match;

       MID MDATE     MLOCATION
---------- --------- --------------------
      2001 23-APR-21 chennai
      2002 24-APR-21 mumbai
      2003 28-APR-21 bengaluru
      2004 29-APR-21 hyderabad
      2005 27-APR-21 kolkata

SQL> select * from city;

CNAME
--------------------
mumbai
chennai
hyderabad
kolkata
bengaluru
```

IMPLEMENTATION

Front end programs and its connectivity

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. The connection to the database can be performed using Java programming (JDBC API) as:

```
public void connectToDB()
{
	try {
		Connection
		con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1
		521:xe","ananya","ananya");
		statement=con.createStatement();
		statement.executeUpdate("commit");
	}
	catch (SQLException connectException)
		{
		System.out.println(connectException.getMessage());
		System.out.println(connectException.getSQLState());
		System.out.println(connectException.getErrorCode());
		System.exit(1);
	}
}
```

Thus, the connection from Java to Oracle database is performed and therefore, can be used for updating tables in the database directly.

Implementation

- Front end programs:

1) Insert a Player:

```
package DBMS;

import java.awt.Button;
import java.awt.Panel;
import java.awt.TextArea;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.*;
import java.sql.*;
public class insertPlayer extends Panel{
	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	Button insertPlayerButton;
	TextField pid,pname,ranking,nationality,skill,tname;
	TextArea errorText;
	Connection connection;
	Statement statement;
	public insertPlayer()
	{
		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:xe","ananya","ananya");
                        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()
        {
                insertPlayerButton = new Button("InsertPlayer");
                insertPlayerButton.addActionListener(new ActionListener()
                {
                        public void actionPerformed(ActionEvent e)
                        {
                                try
                                {
                                        Statement statement =
connection.createStatement();
                                        String query="INSERT INTO player VALUES(" +
pid.getText() + ", " + "'" + pname.getText() + "','"+"'" + ranking.getText() + ","+"'" +
nationality.getText() + ","+"'" + skill.getText() + ","+"'" + tname.getText() + ")";
                                        int i=statement.executeUpdate(query);
                                        errorText.append("\nInserted"+i+"rows.");
                                }
                                catch(SQLException insertException)
                                {
                                        displaySQLErrors(insertException);
                                }
                        }
                }
                                );
                pid=new TextField(20);
                pname=new TextField(20);
```

```
            ranking=new TextField(20);
            nationality=new TextField(20);
            skill=new TextField(20);
            tname=new TextField(20);
            errorText=new TextArea(10,80);
            errorText.setEditable(false);
            Panel first=new Panel();
            first.setLayout(new GridLayout(5,2));
            first.add(new Label("PLAYER ID:"));
            first.add(pid);
            first.add(new Label("PLAYER NAME:"));
            first.add(pname);
            first.add(new Label("RANKING:"));
            first.add(ranking);
            first.add(new Label("NATIONALITY:"));
            first.add(nationality);
            first.add(new Label("SKILL:"));
            first.add(skill);
            first.add(new Label("TEAM NAME:"));
            first.add(tname);
            first.setBounds(125,90,200,100);
            Panel second=new Panel(new GridLayout(4,1));
            second.add(insertPlayerButton);
            second.setBounds(125,220,150,100);
            Panel third=new Panel();
            third.add(errorText);
            third.setBounds(125,320,300,200);
            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)
        {
                insertPlayer player=new insertPlayer();
                player.buildGUI();
        }
}
```

2. Update a Player:

```
package DBMS;


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 updatePlayer extends Panel

{

        /**

         *

         */
```

```
private static final long serialVersionUID = 1L;
Button updatePlayerButton;
List PlayerIDList;
TextField pid,pname,ranking,nationality,skill,tname;
TextArea errorText;
Connection connection;
Statement statement;
ResultSet rs;

public updatePlayer()
{
    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:xe","ananya"
,"ananya");

                        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 PID FROM PLAYER");
                while (rs.next())
                {
                    PlayerIDList.add(rs.getString("PID"));
                }
            }

            catch (SQLException e)
            {
                displaySQLErrors(e);
            }
        }

        public void buildGUI()
        {
            PlayerIDList = new List(10);
            loadProvider();
            add(PlayerIDList);
            PlayerIDList.addItemListener(new ItemListener()
            {
                public void itemStateChanged(ItemEvent e)
                {
```

```
                    try
                    {
                        rs = statement.executeQuery("SELECT * FROM
player");


                        while (rs.next())
                        {
                            if
(rs.getString("PID").equals(PlayerIDList.getSelectedItem()))
                            break;
                        }
                        if (!rs.isAfterLast())
                        {
                            pid.setText(rs.getString("PID"));
                            pname.setText(rs.getString("PNAME"));
                            ranking.setText(rs.getString("RANKING"));

    nationality.setText(rs.getString("NATIONALITY"));
                            skill.setText(rs.getString("SKILL"));
                            tname.setText(rs.getString("TNAME"));
                        }
                    }
                    catch (SQLException selectException)
                    {
```

```
                                displaySQLErrors(selectException);
                        }
                }
        });
        updatePlayerButton = new Button("Update Player");
        updatePlayerButton.addActionListener(new ActionListener()
        {
                public void actionPerformed(ActionEvent e)
                {
                        try
                        {
                                Statement statement =
connection.createStatement();
                                int i = statement.executeUpdate("UPDATE
Player "
                                        + "SET pname='" +
pname.getText() + "', "
                                        + "ranking=" + ranking.getText() +
", "
                                        + "nationality='" +
nationality.getText() + "', "
                                        + "skill='" + skill.getText() + "', "
                                        + "tname='" + tname.getText() +
                                        " WHERE pid = "+
PlayerIDList.getSelectedItem());
```

```
                    errorText.append("\nUpdated " + i + " rows ");
                    PlayerIDList.removeAll();
                    loadProvider();
                }
                catch (SQLException insertException)
                {
                    displaySQLErrors(insertException);
                }
            }
        });
        pid = new TextField(15);
        pid.setEditable(false);
        pname = new TextField(15);
        ranking = new TextField(15);
        nationality = new TextField(15);
        skill = new TextField(15);
        tname = 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("Player ID:"));
```

```
first.add(pid);
first.add(new Label("Player Name:"));
first.add(pname);
first.add(new Label("Ranking:"));
first.add(ranking);
first.add(new Label("Nationality:"));
first.add(nationality);
first.add(new Label("skill:"));
first.add(skill);
first.add(new Label("Team name:"));
first.add(tname);
Panel second = new Panel(new GridLayout(5, 1));
second.add(updatePlayerButton);

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)
        {
            updatePlayer up = new updatePlayer();


            up.buildGUI();
        }
}
```

3. Delete a Player:

```
package DBMS;


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 deletePlayer extends Panel

{

        /**

         *

         */

        private static final long serialVersionUID = 1L;

        Button deletePlayerButton;

        List PlayerIDList;

        TextField pid,pname,ranking,nationality,skill,tname;

        TextArea errorText;

        Connection connection;

        Statement statement;

        ResultSet rs;


        public deletePlayer()
```

```
        {

                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:xe","ananya"
,"ananya");

                        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 player");
                while (rs.next())
                {
                    PlayerIDList.add(rs.getString("PID"));
                }
            }
            catch (SQLException e)
            {
```

```
                displaySQLErrors(e);
            }
        }

        public void buildGUI()
        {
            PlayerIDList = new List(10);
            loadProvider();
            add(PlayerIDList);



        PlayerIDList.addItemListener(new ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                try

                {
                    rs = statement.executeQuery("SELECT * FROM
player");

                    while (rs.next())
                    {
```

```
if(rs.getString("PID").equals(PlayerIDList.getSelectedItem()))
                    break;
            }
            if (!rs.isAfterLast())
            {
                    pid.setText(rs.getString("PID"));
                    pname.setText(rs.getString("PNAME"));
                    ranking.setText(rs.getString("RANKING"));

nationality.setText(rs.getString("NATIONALITY"));
                    skill.setText(rs.getString("SKILL"));
                    tname.setText(rs.getString("TNAME"));
            }
        }
        catch (SQLException selectException)
        {
            displaySQLErrors(selectException);
        }
    }
});
deletePlayerButton = new Button("Delete Player");
deletePlayerButton.addActionListener(new ActionListener()
```

```
        {
                public void actionPerformed(ActionEvent e)
                {
                        try
                        {
                                Statement statement = connection.createStatement();
                                int i = statement.executeUpdate("DELETE FROM
player WHERE PID = "+ PlayerIDList.getSelectedItem());
                                errorText.append("\nDeleted " + i + " rows");
                                pid.setText(null);
                                pname.setText(null);
                                ranking.setText(null);
                                nationality.setText(null);
                                skill.setText(null);
                                tname.setText(null);
                                PlayerIDList.removeAll();
                                loadProvider();
                        }
                        catch (SQLException insertException)
                        {
                                displaySQLErrors(insertException);
                        }
                }
        });
```

```
pid = new TextField(15);
pname = new TextField(15);
ranking = new TextField(15);
nationality = new TextField(15);
skill = new TextField(15);
tname = 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("Player ID:"));
first.add(pid);
pid.setEditable(false);
first.add(new Label("Player Name:"));
first.add(pname);
pname.setEditable(false);
first.add(new Label("Ranking:"));
first.add(ranking);
ranking.setEditable(false);
first.add(new Label("Nationality:"));
first.add(nationality);
```

```
        nationality.setEditable(false);
        first.add(new Label("Skill:"));
        first.add(skill);
        skill.setEditable(false);
        first.add(new Label("Team Name:"));
        first.add(tname);
        tname.setEditable(false);
        Panel second = new Panel(new GridLayout(5, 1));
        second.add(deletePlayerButton);

        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)
        {
                deletePlayer del = new deletePlayer();
                del.buildGUI();
        }
}
```

Main Program:

```
package DBMS;

import java.awt.*;
import java.awt.event.*;

public class IplDisplaySystem extends Frame implements ActionListener
{
        /**
        *
```

```
 */
private static final long serialVersionUID = 1L;
String msg = "";
  Label ll,l2;
  CardLayout cardLO;
  insertCity inc;
  updateCity upc;
  deleteCity delc;
  insertMatch inm;
  updateMatch upm;
  deleteMatch delm;
  insertOwner ino;
  updateOwner upo;
  deleteOwner delo;
  insertPlayer inp;
  updatePlayer upp;
  deletePlayer delp;
  insertSponsor ins;
  updateSponsor ups;
  deleteSponsor dels;
  insertTeam inte;
  updateTeam upt;
  deleteTeam delt;
```

```
Panel home,welcome;


IplDisplaySystem()
{
        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 IPL Display System");
        l2.setText("All @rights are reserved");
        welcome = new Panel();
        welcome.add(ll);
        welcome.add(l2);
        inc = new insertCity();
        inc.buildGUI();
        upc = new updateCity();
        upc.buildGUI();
        delc = new deleteCity();
        delc.buildGUI();
        inm = new insertMatch();
```

```
inm.buildGUI();
upm = new updateMatch();
upm.buildGUI();
delm = new deleteMatch();
delm.buildGUI();
ino = new insertOwner();
ino.buildGUI();
upo = new updateOwner();
upo.buildGUI();
delo = new deleteOwner();
delo.buildGUI();
inp = new insertPlayer();
inp.buildGUI();
upp = new updatePlayer();
upp.buildGUI();
delp = new deletePlayer();
delp.buildGUI();
ins = new insertSponsor();
ins.buildGUI();
ups = new updateSponsor();
ups.buildGUI();
dels = new deleteSponsor();
dels.buildGUI();
```

```
inte = new insertTeam();
inte.buildGUI();
upt = new updateTeam();
upt.buildGUI();
delt = new deleteTeam();
delt.buildGUI();
//add all the panels to the home panel which has a
cardlayout
home.add    (welcome, "Welcome");
home.add    (inc, "insertCity");
home.add    (upc, "updateCity");
home.add    (delc, "deleteCity");
home.add    (inm, "insertMatch");
home.add    (upm, "updateMatch");
home.add    (delm, "deleteMatch");
home.add    (ino, "insertOwner");
home.add    (upo, "updateOwner");
home.add    (delo, "deleteOwner");
home.add    (inp, "insertPlayer");
home.add    (upp, "updatePlayer");
home.add    (delp, "deletePlayer");
home.add    (ins, "insertSponsor");
home.add    (ups, "updateSponsor");
home.add    (dels, "deleteSponsor");
```

```
home.add    (inte, "insertTeam");
home.add    (upt, "updateTeam");
home.add    (delt, "deleteTeam");

// 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 city = new Menu("City");
MenuItem item1, item2, item3;
city.add(item1 = new MenuItem("Insert City"));
city.add(item2 = new MenuItem("View City"));
city.add(item3 = new MenuItem("Delete City"));
mbar.add(city);

Menu team = new Menu("Team");
MenuItem item4, item5, item6;
team.add(item4 = new MenuItem("Insert Team"));
team.add(item5 = new MenuItem("View Team"));
```

```
team.add(item6 = new MenuItem("Delete Team"));
mbar.add(team);

Menu match = new Menu("Match");
MenuItem item7, item8, item9;
match.add(item7 = new MenuItem("Insert Match"));
match.add(item8 = new MenuItem("View Match"));
match.add(item9 = new MenuItem("Delete Match"));
mbar.add(match);

Menu owner = new Menu("Owner");
MenuItem item10, item11, item12;
owner.add(item10 = new MenuItem("Insert Owner"));
owner.add(item11 = new MenuItem("View Owner"));
owner.add(item12 = new MenuItem("Delete Owner"));
mbar.add(owner);

Menu player = new Menu("Player");
MenuItem item13, item14, item15;
player.add(item13 = new MenuItem("Insert Player"));
player.add(item14 = new MenuItem("View Player"));
player.add(item15 = new MenuItem("Delete Player"));
mbar.add(player);
```

```
Menu sponsor = new Menu("Sponsor");
MenuItem item16, item17, item18;
sponsor.add(item16 = new MenuItem("Insert Sponsor"));
sponsor.add(item17 = new MenuItem("View Sponsor"));
sponsor.add(item18 = new MenuItem("Delete Sponsor"));
mbar.add(sponsor);


// 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);
        item16.addActionListener(this);
        item17.addActionListener(this);
        item18.addActionListener(this);

        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent we)
            {
                System.exit(0);
            }
        });

        //Frame properties
        setTitle("IplDisplaySystem");
        Color clr = new Color(50, 150, 100);
        setBackground(clr);
        setFont(new Font("SansSerif", Font.CENTER_BASELINE, 18));
        setLayout(null);
        setSize(900, 1000);
        setVisible(true);

    }
```

```
public void actionPerformed(ActionEvent ae)
{
        String arg = ae.getActionCommand();
        if(arg.equals("Insert City"))
        {
                cardLO.show(home, "insertCity");
        }

        else if(arg.equals("View City"))
        {
                cardLO.show(home, "updateCity");
        }

        else if(arg.equals("Delete City"))
        {
                cardLO.show(home, "deleteCity");
        }

        else if(arg.equals("Insert Team"))
        {
                cardLO.show(home, "insertTeam");
        }
```

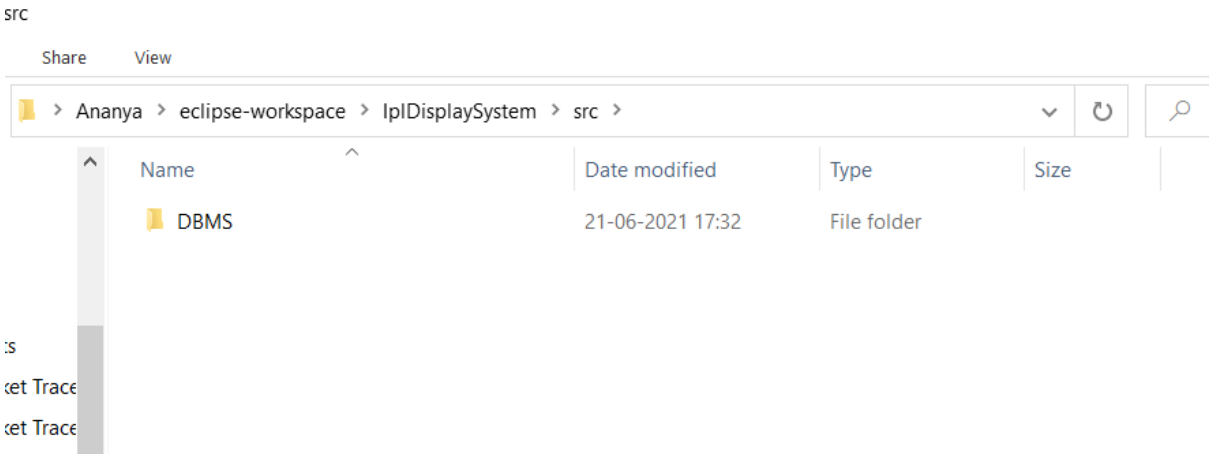
```
else if(arg.equals("View Team"))
{
        cardLO.show(home, "updateTeam");
}

else if(arg.equals("Delete Team"))
{
        cardLO.show(home, "deleteTeam");
}

else if(arg.equals("Insert Match"))
{
        cardLO.show(home, "insertTeam");
}

else if(arg.equals("View ,Match"))
{
        cardLO.show(home, "updatMatch");
}

else if(arg.equals("Delete Match"))
{
```

```
            cardLO.show(home, "deleteMatch");
        }

        else if(arg.equals("Insert Owner"))
        {
            cardLO.show(home, "insertOwner");
        }

        else if(arg.equals("View Owner"))
        {
            cardLO. show(home, "updateOwner");
        }

        else if(arg.equals("Delete Owner"))
        {
            cardLO.show(home, "deleteOwner");
        }

        else if(arg.equals("Insert Player"))
        {
            cardLO.show(home, "insertPlayer");
        }
```

```
else if(arg.equals("View Player"))
{
        cardLO. show(home, "updatePlayer");
}

else if(arg.equals("Delete Player"))
{
        cardLO.show(home, "deletePlayer");
}

else if(arg.equals("Insert Sponsor"))
{
        cardLO.show(home, "insertSponsor");
}

else if(arg.equals("View Sponsor"))
{
        cardLO. show(home, "updateSponsor");
}

else
{
        cardLO.show(home, "deleteSponsor");
```

```
            }
        }

        public static void main(String ... args)
        {
            new IplDisplaySystem();
        }
}
```

FOLDER STRUCTURE



IPL Display System

IplDisplaySystem

Share View

Ananya > eclipse-workspace > IplDisplaySystem >

Name	Date modified	Type	Size
.settings	19-06-2021 22:00	File folder	
bin	22-06-2021 11:13	File folder	
src	21-06-2021 16:29	File folder	
xtend-gen	21-06-2021 16:29	File folder	
.classpath	21-06-2021 16:29	CLASSPATH File	1 KB
.gitattributes	24-06-2021 00:02	GITATTRIBUTES File	1 KB
.project	21-06-2021 16:29	PROJECT File	1 KB

DBMS

Share View

Ananya > eclipse-workspace > IplDisplaySystem > src > DBMS

Search DB

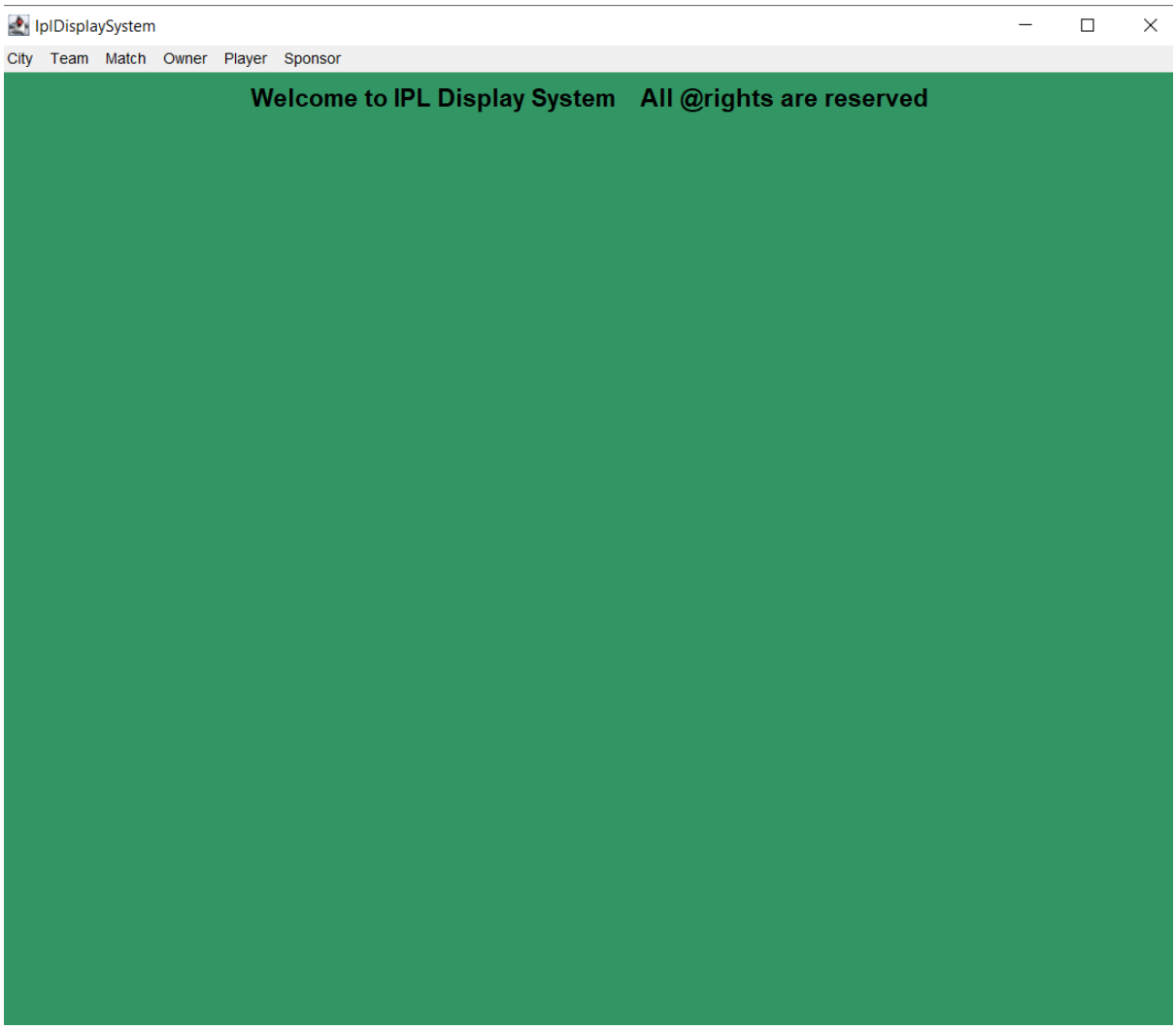
Name	Date modified	Type	Size
deleteCity	23-06-2021 16:44	JAVA File	4 KB
deleteMatch	23-06-2021 16:43	JAVA File	4 KB
deleteOwner	22-06-2021 20:49	JAVA File	4 KB
deletePlayer	22-06-2021 20:49	JAVA File	5 KB
deleteSponsor	22-06-2021 20:49	JAVA File	4 KB
deleteTeam	22-06-2021 20:49	JAVA File	4 KB
insertCity	23-06-2021 15:28	JAVA File	3 KB
insertMatch	23-06-2021 15:10	JAVA File	3 KB
insertOwner	23-06-2021 15:10	JAVA File	3 KB
insertPlayer	23-06-2021 20:15	JAVA File	4 KB
insertSponsor	23-06-2021 15:10	JAVA File	3 KB
insertTeam	22-06-2021 20:49	JAVA File	3 KB
IplDisplaySystem	23-06-2021 20:31	JAVA File	8 KB
updateCity	23-06-2021 11:47	JAVA File	4 KB
updateMatch	23-06-2021 11:47	JAVA File	4 KB
updateOwner	23-06-2021 11:47	JAVA File	4 KB
updatePlayer	23-06-2021 11:47	JAVA File	5 KB
updateSponsor	23-06-2021 12:34	JAVA File	4 KB
updateTeam	23-06-2021 11:55	JAVA File	5 KB

1602-19-737-067
Ananya. M

TESTING

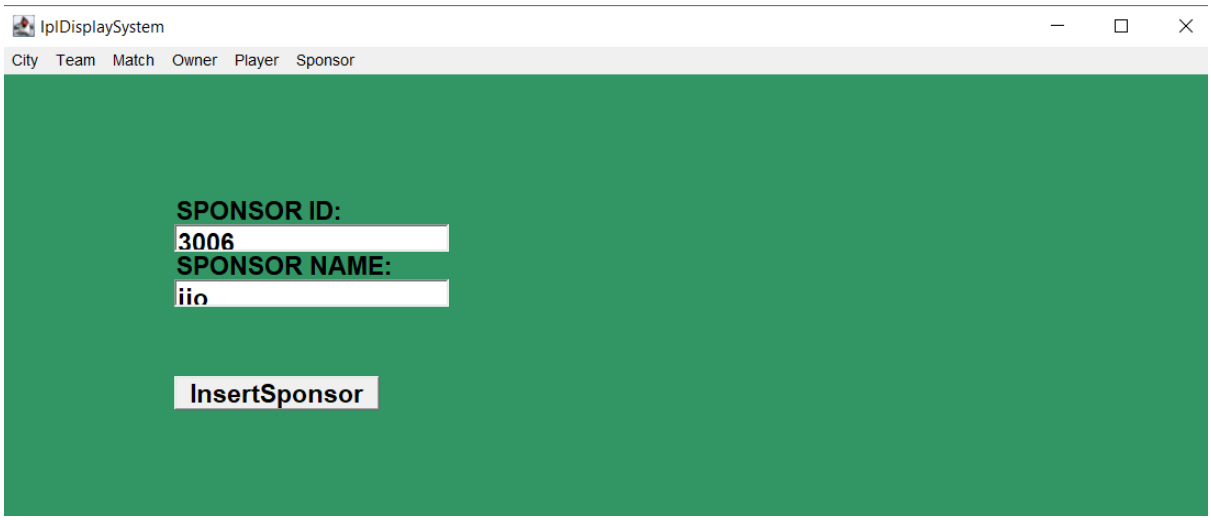
The program runs for execution of three basic operations of insertion, update and delete on 6 different table. Along with this, it also has an output column which gives the information about how many rows have been edited. Errors, syntactical or exceptional will be shown if occurred.

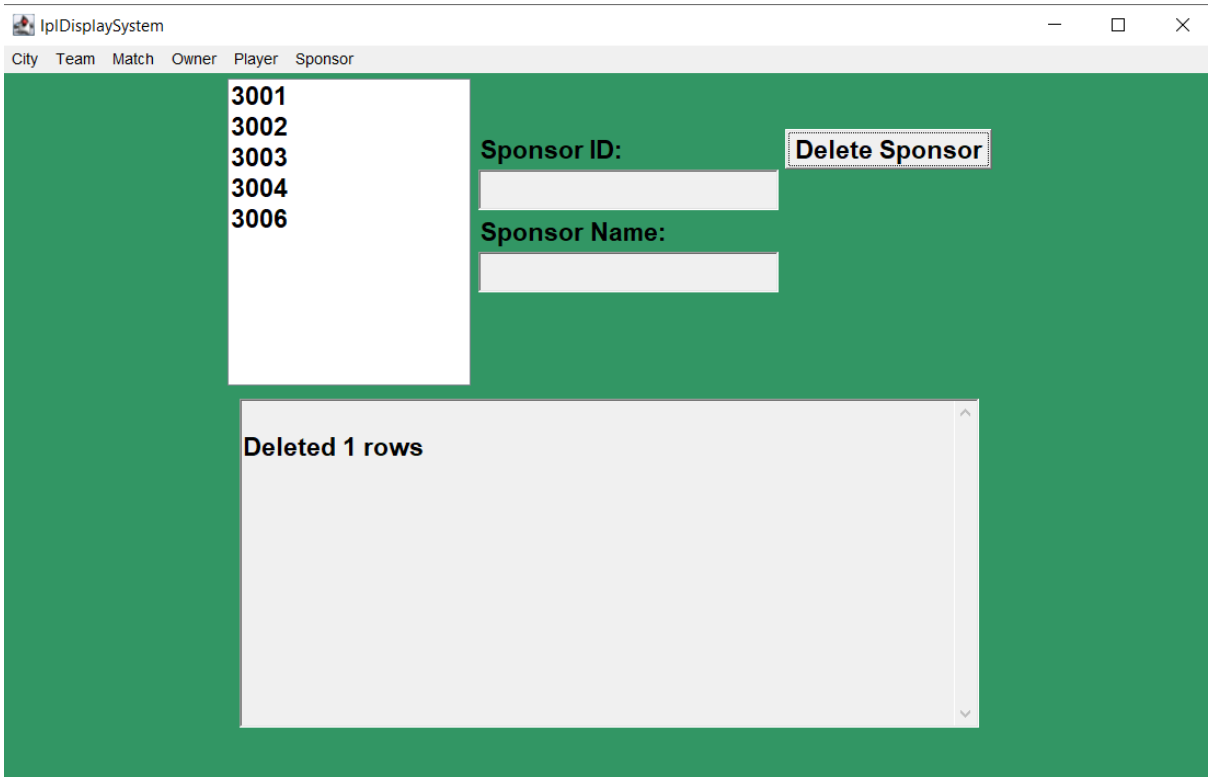
OUTPUT SCREENSHOTS:



IPL Display System

```
SQL> select * from sponsor;

       SID SNAME
---------- --------------------
      3001 nokia
      3002 reliance
      3003 suntv
      3004 muthoot finance
      3005 britania
```





1602-19-737-067
Ananya. M

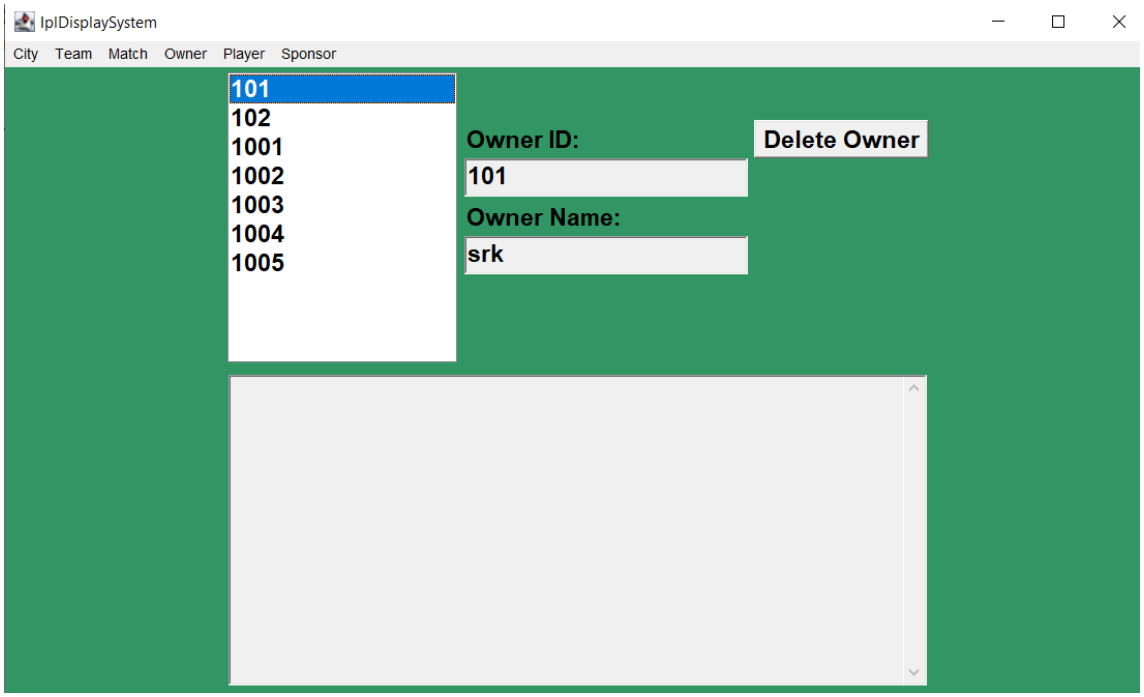
Deleted sponsor id 3005:

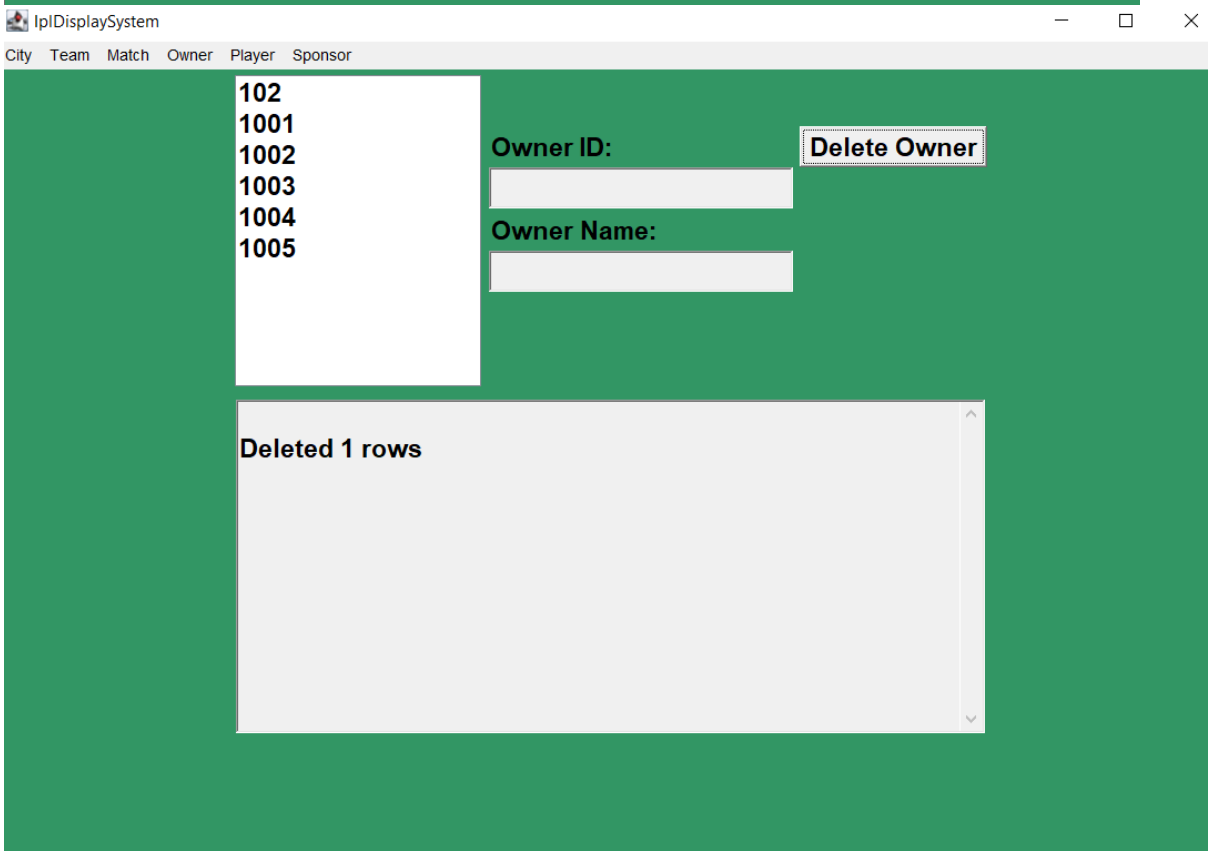
```
SQL> select * from sponsor;

       SID SNAME
---------- --------------------
      3001 nokia
      3002 reliance
      3003 suntv
      3004 muthoot finance
      3006 jio
```

Owner table:

```
SQL> select * from owner;

       OID ONAME
---------- --------------------
       101 srk
       102 nita ambani
      1001 srk
      1002 nita ambani
      1003 vijay mallya
      1004 kalanithi maran
      1005 n srinivasan
```

IPL Display System





1602-19-737-067
Ananya. M

```
SQL> select * from owner;

       OID ONAME
---------- --------------------
       102 nita ambani
      1001 srk
      1002 nita ambani
      1003 vijay mallya
      1004 kalanithi maran
      1005 n srinivasan

6 rows selected.
```

DISCUSSIONS

The application “IPL Display System” helps to find the details about the players, teams and various attributes in involved in the ipl. A user can check the details he/she requires but access to any confidential details is provided only if the user has his/her specific identity which needs to be recorded in the database.

The details of the teams after every match is played is recorded in the database and can be accessed whenever required.

The data entered in the database is highly secure and can only be altered by the authorized person through their own identity. The data entered is stored into the database immediately to avoid loss or tampering of data.

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

REFERENCES

1. https://github.com/Ananyaaa-m/IPLDispalySystem
2. https://en.wikipedia.org/wiki/List_of_Indian_Premier_League_seasons_and_results