JAVA SWING BASED –IPL DataBase –SQL CONNECTIVITY USING JDBC

A Report
Submitted in partial fulfillment of the Requirements
for the COURSE

DATABASE MANAGEMENT SYSTEMS
By

Lokesh Munagapati 1602-21-737-030
Under the guidance of Ms B. Leelavathy

Department of Information Technology
Vasavi College of Engineering (Autonomous)
(Affiliated to Osmania University)
Ibrahimbagh, Hyderabad-31
2022-2023

BONAFIDE CERTIFICATE

This is to certify that this project report titled

'IPL Database'

is a project work of **Lokesh Munagapati** bearing roll no. 1602-21-737-030 who carried out this project under my supervision in the IV semester for the academic year 2022- 2023

Signature
External Examiner

Signature
Internal Examiner

ABSTRACT

The "DML Form" project is a Java-based application that serves as a simple Database Management System (DBMS) interface. It allows users to perform basic CRUD (Create, Read, Update, Delete) operations on different database tables.

The project utilizes the Java Swing library to create a graphical user interface (GUI) with various buttons and a table for displaying data. The application connects to an Oracle database using JDBC (Java Database Connectivity) to execute SQL queries and updates.

Key Features:

1. Table View: The main screen displays a table with columns representing the different database tables and fields.
2. Button Navigation: Buttons are provided for each table, such as "Matches," "Teams," "Players," and "Coaches," allowing users to perform specific actions related to each table.
3. Data Display: Users can display all records from the database tables, such as all teams, all coaches, or all matches.
4. Refresh: A "Refresh" button allows users to clear the table and reload data from the database.

The project follows a modular approach, with separate classes for each table page (e.g., MatchesPage, TeamsPage) that handle specific actions and communicate with the main DMLForm class. The code includes exception handling for database connections and queries.

Overall, the "DML Form" project provides a user-friendly interface for managing a database, enabling users to view, edit, and manipulate data stored in different tables. It serves as a starting point for developing a more comprehensive DBMS application

Requirement Analysis

List of Tables:

-Match

-Team

-Player

-Coach

List of Attributes with their Domain Types:

Table: Match

- match_id: Number(5) (Primary Key)
- match_location: VARCHAR(20)
- Team1_id: Number(5) (Foreign Key referencing team_id in the Team table)
- Team2_id: Number(5) (Foreign Key referencing team_id in the Team table)

Table: Team

- Team_id: Number(5) (Primary Key)
- team_name: VARCHAR(20)
- coach_id: Number(5) (Foreign Key referencing coach_id in the coach table)

Table: Player

- Player_id: Number(5) (Primary Key)
- Player_name: VARCHAR(20)
- team_id: Number(5) (Foreign Key referencing team_id in the Team table)

Table: Coach

- Coach_id: Number(5) (Primary Key)
- Coach_name: VARCHAR2(20)

AIM AND PRIORITY OF THE PROJECT

The aim of the "DML Form" project is to create a user-friendly GUI application for basic database management, allowing users to perform CRUD operations on different tables. The top priority is to provide a functional and intuitive interface for viewing and manipulating data stored in the database tables, while ensuring error handling and data integrity. The project should be modular and maintainable for future enhancements and scalability.

ARCHITECTURE AND TECHNOLOGY

Software used:

Java, Oracle 11g Database, Java SE version 14, Run SQL.

Java SWING:

Java SWING is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs.

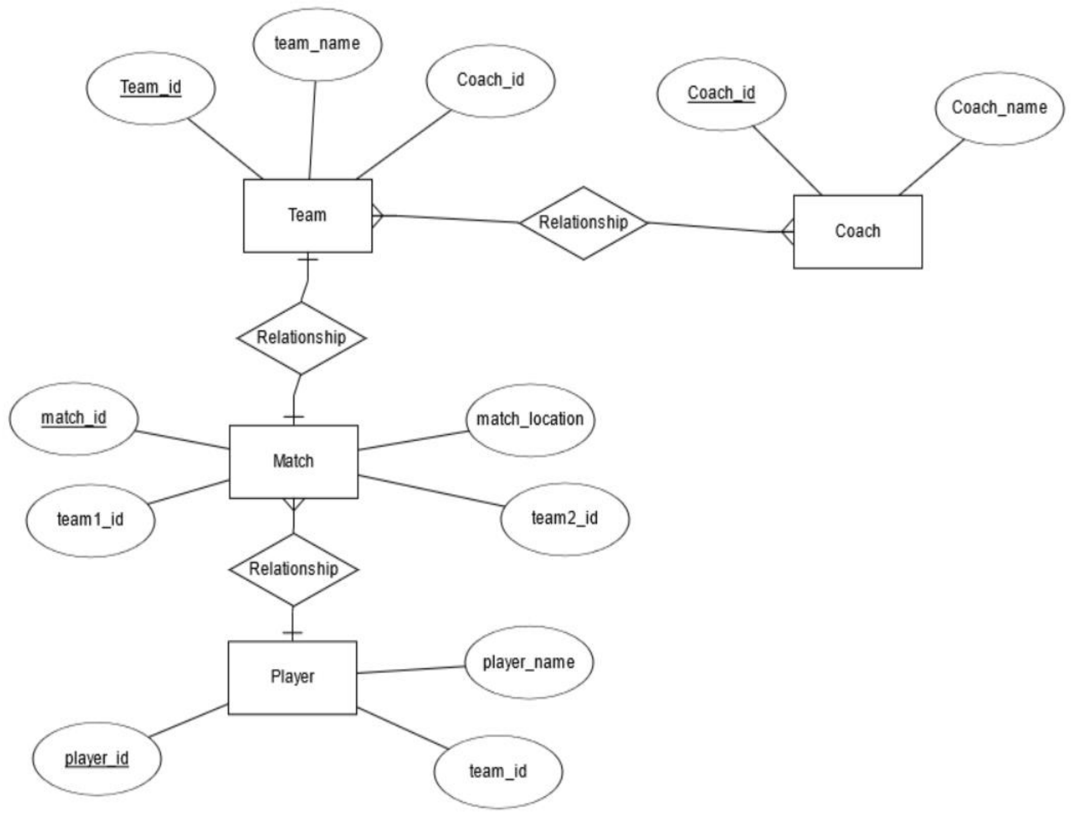
Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT. Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

SQL:

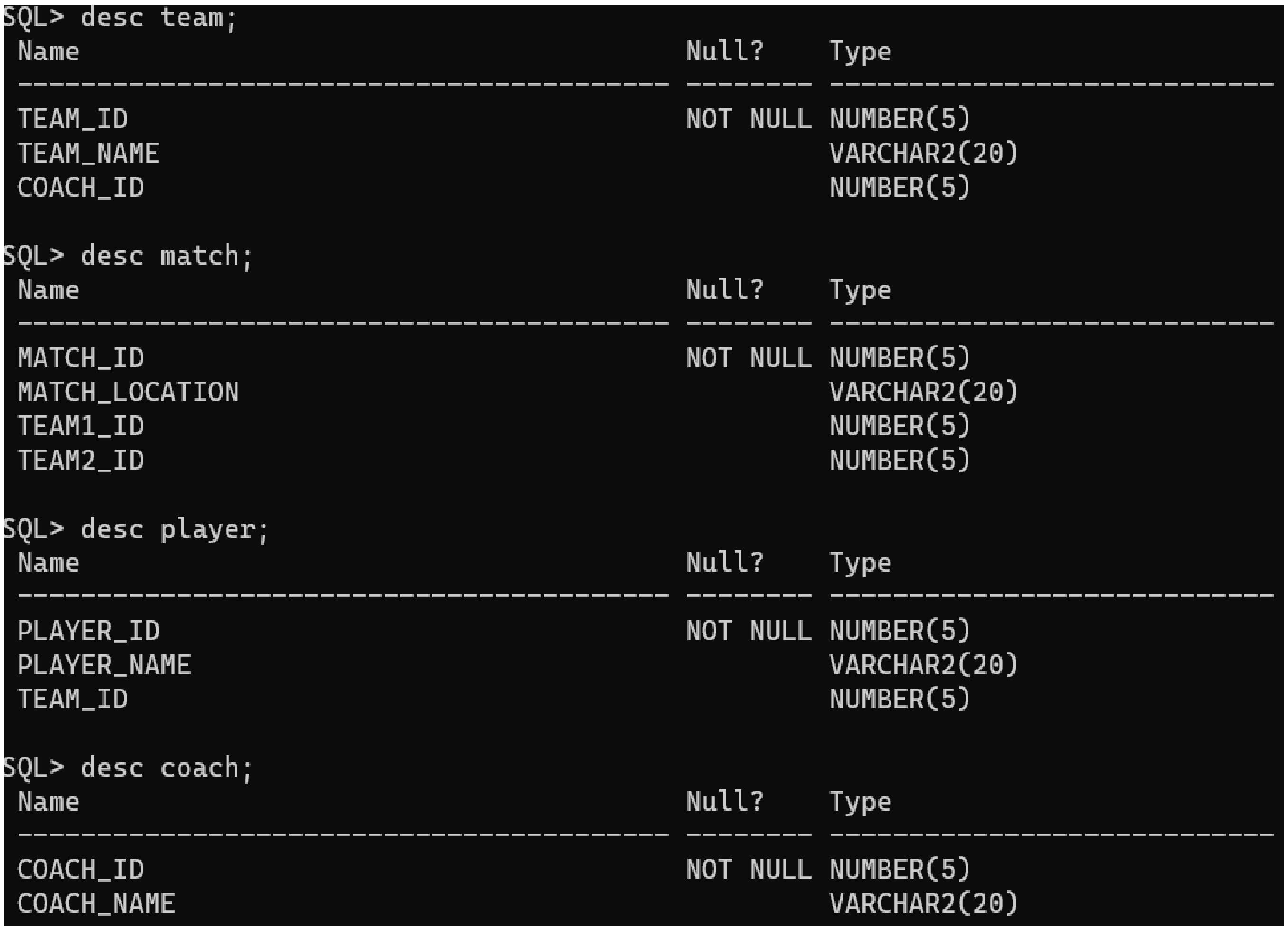
Structure Query Language(SQL) is a database query language used for storing and managing data in **Relational** DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** asthe standard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

DESIGN

Entity Relationship Diagram

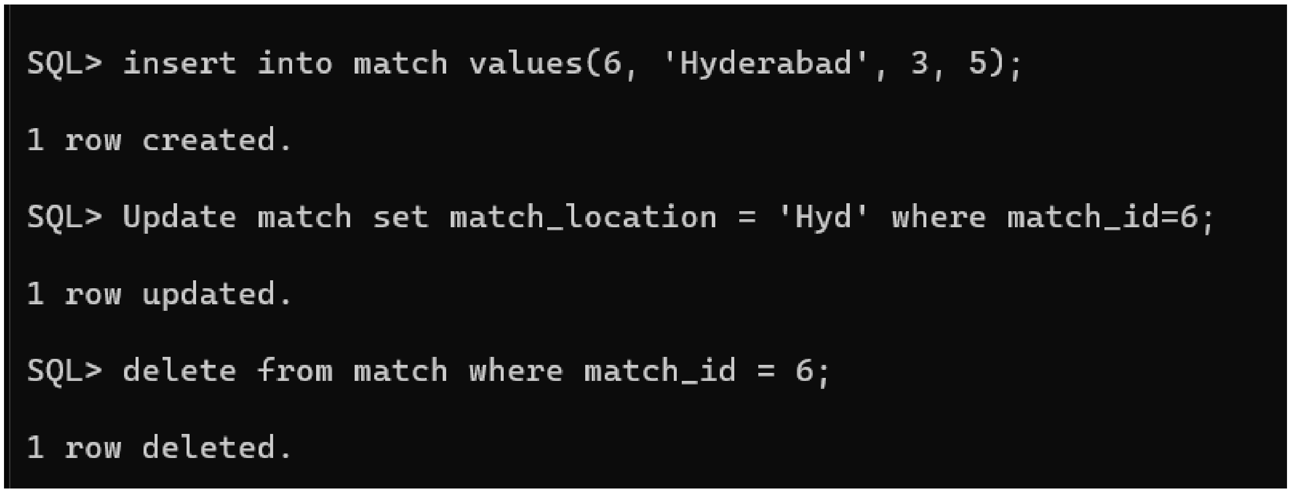


DATABASE DESIGN:



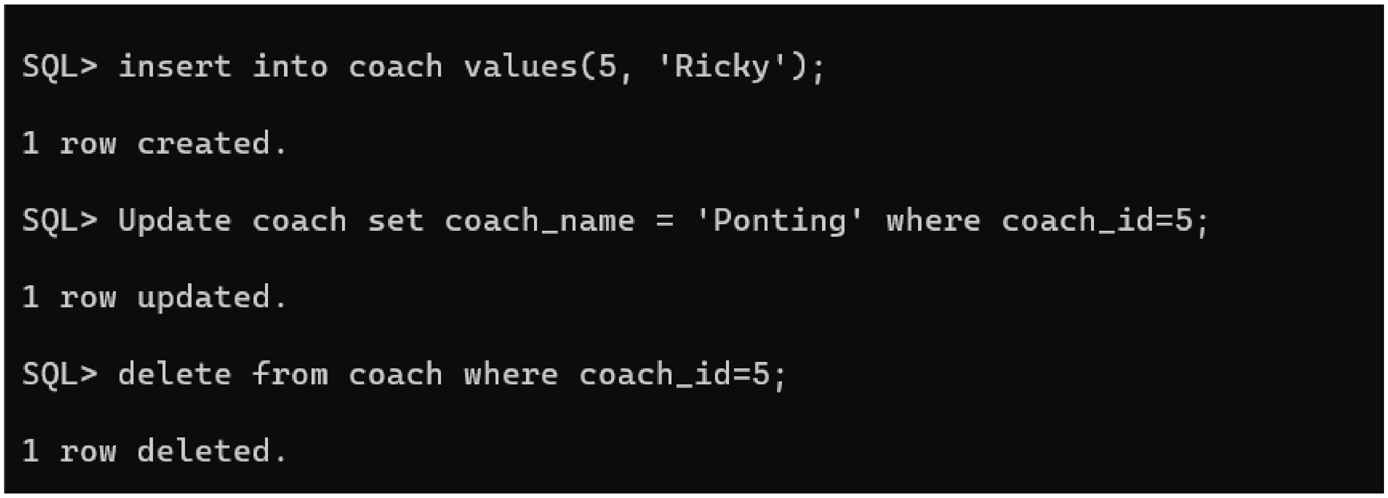
```
SQL> desc team;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TEAM_ID                                   NOT NULL NUMBER(5)
 TEAM_NAME                                          VARCHAR2(20)
 COACH_ID                                           NUMBER(5)

SQL> desc match;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MATCH_ID                                  NOT NULL NUMBER(5)
 MATCH_LOCATION                                     VARCHAR2(20)
 TEAM1_ID                                           NUMBER(5)
 TEAM2_ID                                           NUMBER(5)

SQL> desc player;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PLAYER_ID                                 NOT NULL NUMBER(5)
 PLAYER_NAME                                        VARCHAR2(20)
 TEAM_ID                                            NUMBER(5)

SQL> desc coach;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COACH_ID                                  NOT NULL NUMBER(5)
 COACH_NAME                                         VARCHAR2(20)
```

DML Operations on Match



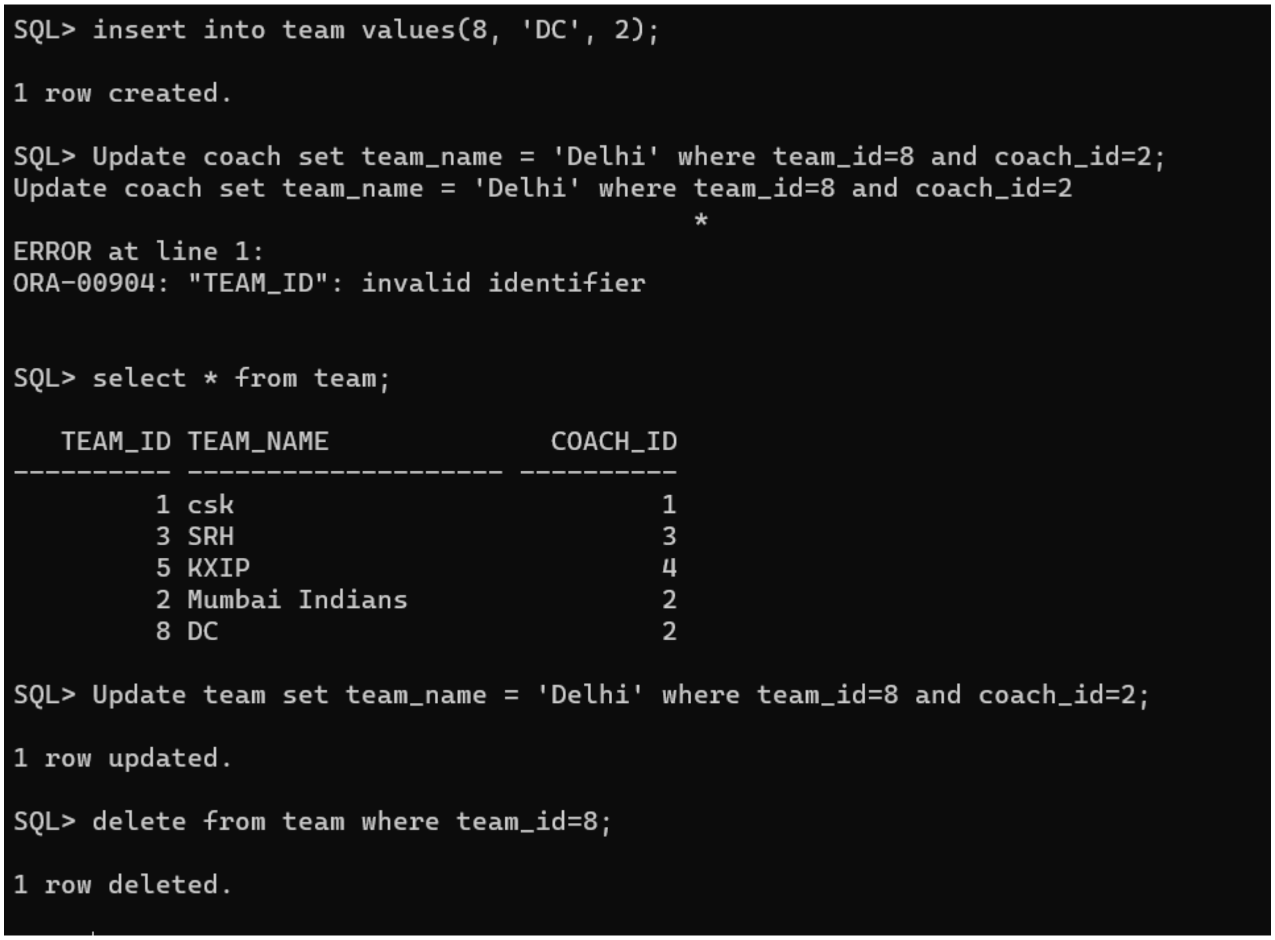
```
SQL> insert into match values(6, 'Hyderabad', 3, 5);

1 row created.

SQL> Update match set match_location = 'Hyd' where match_id=6;

1 row updated.

SQL> delete from match where match_id = 6;

1 row deleted.
```

DML Operations ON Coach



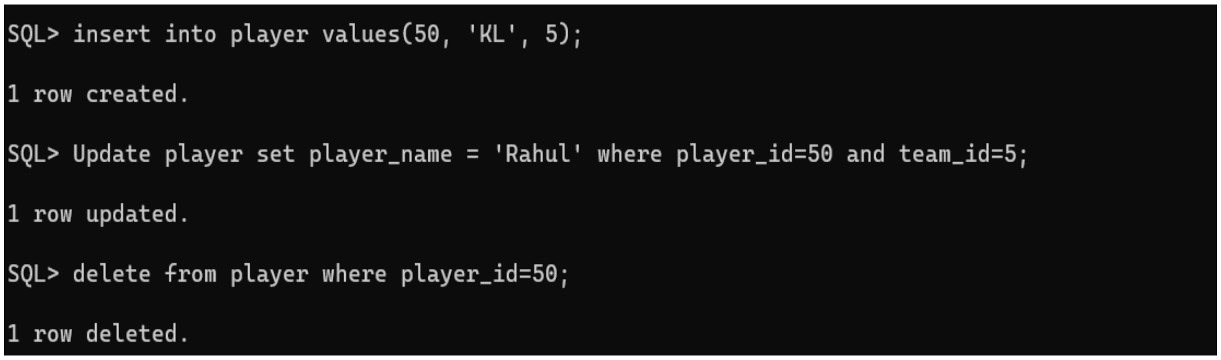
```
SQL> insert into coach values(5, 'Ricky');

1 row created.

SQL> Update coach set coach_name = 'Ponting' where coach_id=5;

1 row updated.

SQL> delete from coach where coach_id=5;

1 row deleted.
```

DML Operations on Team



```
SQL> insert into team values(8, 'DC', 2);

1 row created.

SQL> Update coach set team_name = 'Delhi' where team_id=8 and coach_id=2;
Update coach set team_name = 'Delhi' where team_id=8 and coach_id=2
                                           *
ERROR at line 1:
ORA-00904: "TEAM_ID": invalid identifier


SQL> select * from team;

   TEAM_ID TEAM_NAME              COACH_ID
---------- -------------------- ----------
         1 csk                           1
         3 SRH                           3
         5 KXIP                          4
         2 Mumbai Indians                2
         8 DC                            2

SQL> Update team set team_name = 'Delhi' where team_id=8 and coach_id=2;

1 row updated.

SQL> delete from team where team_id=8;

1 row deleted.
```

DML Operations on Player



```
SQL> insert into player values(50, 'KL', 5);

1 row created.

SQL> Update player set player_name = 'Rahul' where player_id=50 and team_id=5;

1 row updated.

SQL> delete from player where player_id=50;

1 row deleted.
```

IMPLEMENTATION

JAVA-SQL Connectivity using JDBC:

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:

DMLForm.java

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class DMLForm extends JFrame {
    private static final String DB_URL = "jdbc:oracle:thin:@localhost:1521:xe";
    private static final String USERNAME = "lokesh";
    private static final String PASSWORD = "Lokesh";
    private DefaultTableModel tableModel;
    private JTable table;

    public DMLForm() {
        setTitle("DML Form");
        setLayout(new FlowLayout());

        tableModel = new DefaultTableModel(new String[]{"Action", "Table",
"ID", "Field 1", "Field 2", "Field 3"}, 0);
        table = new JTable(tableModel);

        JButton matchesButton = createStyledButton("Matches");
        matchesButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                MatchesPage matchesPage = new MatchesPage(tableModel, table);
                matchesPage.setVisible(true);
            }
        });
        add(matchesButton);

        JButton teamsButton = createStyledButton("Teams");
        teamsButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                TeamsPage teamsPage = new TeamsPage(tableModel, table);
                teamsPage.setVisible(true);
            }
        });
        add(teamsButton);

        JButton playersButton = createStyledButton("Players");
        playersButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                PlayersPage playersPage = new PlayersPage(tableModel, table);
                playersPage.setVisible(true);
            }
        });
        add(playersButton);

        JButton coachesButton = createStyledButton("Coaches");
        coachesButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                CoachesPage coachesPage = new CoachesPage(tableModel, table);
                coachesPage.setVisible(true);
            }
        });
        add(coachesButton);

        JButton displayAllTeamsButton = createStyledButton("Display All Teams");
        displayAllTeamsButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                displayAllTeams(tableModel, table);
            }
        });
        add(displayAllTeamsButton);

        JButton displayAllCoachesButton = createStyledButton("Display All
Coaches");
        displayAllCoachesButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                displayAllCoaches(tableModel, table);
            }
        });
        add(displayAllCoachesButton);

        JButton displayAllMatchesButton = createStyledButton("Display All
Matches");
        displayAllMatchesButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                displayAllMatches(tableModel, table);
            }
        });
        add(displayAllMatchesButton);

        JButton refreshButton = createStyledButton("Refresh");
        refreshButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                refreshTable();
            }
        });
        add(refreshButton);

        JScrollPane scrollPane = new JScrollPane(table);
        scrollPane.setPreferredSize(new Dimension(600, 300));
        add(scrollPane);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack();
        setLocationRelativeTo(null); // Center the frame on the screen
    }

    private JButton createStyledButton(String text) {
        JButton button = new JButton(text);
        button.setFont(new Font("Arial", Font.BOLD, 18));
        button.setBackground(Color.WHITE);
        button.setForeground(Color.BLACK);
        button.setFocusPainted(false);
        button.setBorder(BorderFactory.createEmptyBorder(10, 20, 10, 20));
        button.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
        return button;
    }

    private void displayAllTeams(DefaultTableModel tableModel, JTable table) {
            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
                String sql = "SELECT * FROM Team";
                PreparedStatement stmt = conn.prepareStatement(sql);
                ResultSet rs = stmt.executeQuery();

                while (rs.next()) {
                    int teamId = rs.getInt("team_id");
                    String teamName = rs.getString("team_name");
                    int coachId = rs.getInt("coach_id");

                    tableModel.addRow(new Object[]{"Display", "Teams", teamId,
teamName, coachId});
                }

                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void displayAllCoaches(DefaultTableModel tableModel, JTable table)
{
            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
                String sql = "SELECT * FROM Coach";
                PreparedStatement stmt = conn.prepareStatement(sql);
                ResultSet rs = stmt.executeQuery();

                while (rs.next()) {
                    int coachId = rs.getInt("coach_id");
                    String coachName = rs.getString("coach_name");
                    tableModel.addRow(new Object[]{"Display", "Coaches", coachId,
coachName});
                }

                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void displayAllMatches(DefaultTableModel tableModel, JTable table)
{
            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
                String sql = "SELECT * FROM Match";
                PreparedStatement stmt = conn.prepareStatement(sql);
                ResultSet rs = stmt.executeQuery();

                while (rs.next()) {
                    int matchId = rs.getInt("match_id");
                    String matchLocation = rs.getString("match_location");
                    int team1Id = rs.getInt("team1_id");
                    int team2Id = rs.getInt("team2_id");

                    tableModel.addRow(new Object[]{"Display", "Matches", matchId,
matchLocation, team1Id, team2Id});
                }

                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void refreshTable() {
            tableModel.setRowCount(0); // Clear all rows in the table
        }

        public static void main(String[] args) {
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    new DMLForm().setVisible(true);
                }
            });
        }
    }

    class TeamsPage extends JFrame {
        private static final String DB_URL = "jdbc:oracle:thin:@localhost:1521:xe";
        private static final String USERNAME = "lokesh";
        private static final String PASSWORD = "Lokesh";

        private JTextField teamIdField;
        private JTextField teamNameField;
        private JTextField coachIdField;
        private DefaultTableModel tableModel;
        private JTable table;

        public TeamsPage(DefaultTableModel tableModel, JTable table) {
            this.tableModel = tableModel;
            this.table = table;
            setTitle("Teams Page");
            setSize(400, 150);

            JPanel panel = new JPanel(new FlowLayout());
```

```
panel.add(new JLabel("Team ID:"));
teamIdField = new JTextField(10);
panel.add(teamIdField);

panel.add(new JLabel("Team Name:"));
teamNameField = new JTextField(10);
panel.add(teamNameField);

panel.add(new JLabel("Coach ID:"));
coachIdField = new JTextField(10);
panel.add(coachIdField);

JButton okButton = new JButton("OK");
okButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int teamId = Integer.parseInt(teamIdField.getText());
        fetchTeamDetails(teamId);
    }
});
panel.add(okButton);

JButton insertButton = new JButton("Insert");
insertButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        insertTeam();
    }
});
panel.add(insertButton);

JButton modifyButton = new JButton("Modify");
modifyButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        modifyTeam();
    }
```

```
          });
          panel.add(modifyButton);

          JButton deleteButton = new JButton("Delete");
          deleteButton.addActionListener(new ActionListener() {
             public void actionPerformed(ActionEvent e) {
                deleteTeam();
             }
          });
          panel.add(deleteButton);
          setLocationRelativeTo(null);

          add(panel);
          setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
       }

       private void fetchTeamDetails(int teamId) {
          try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
             String sql = "SELECT * FROM Team WHERE team_id = ?";
             PreparedStatement stmt = conn.prepareStatement(sql);
             stmt.setInt(1, teamId);
             ResultSet rs = stmt.executeQuery();

             if (rs.next()) {
                String teamName = rs.getString("team_name");
                int coachId = rs.getInt("coach_id");

                teamIdField.setText(String.valueOf(teamId));
                teamNameField.setText(teamName);
                coachIdField.setText(String.valueOf(coachId));
             } else {
                JOptionPane.showMessageDialog(this, "Team not found!");
                clearFields();
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void insertTeam() {
            int teamId = Integer.parseInt(teamIdField.getText());
            String teamName = teamNameField.getText();
            int coachId = Integer.parseInt(coachIdField.getText());

            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
                String sql = "INSERT INTO Team (team_id, team_name, coach_id)
VALUES (?, ?, ?)";
                PreparedStatement stmt = conn.prepareStatement(sql);
                stmt.setInt(1, teamId);
                stmt.setString(2, teamName);
                stmt.setInt(3, coachId);
                stmt.executeUpdate();

                JOptionPane.showMessageDialog(this, "Team inserted successfully!");

                // Update the table
                tableModel.addRow(new Object[]{"Insert", "Teams", teamId, teamName,
coachId});
                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));

                clearFields();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
```

```
    private void modifyTeam() {
        int teamId = Integer.parseInt(teamIdField.getText());
        String teamName = teamNameField.getText();
        int coachId = Integer.parseInt(coachIdField.getText());

        try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
            String sql = "UPDATE Team SET team_name = ?, coach_id = ? WHERE
team_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, teamName);
            stmt.setInt(2, coachId);
            stmt.setInt(3, teamId);
            stmt.executeUpdate();

            JOptionPane.showMessageDialog(this, "Team modified successfully!");

            // Update the table
            tableModel.addRow(new Object[]{"Modify", "Teams", teamId,
teamName, coachId});
            table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));

            clearFields();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    private void deleteTeam() {
        int teamId = Integer.parseInt(teamIdField.getText());
```

```
            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
                String sql = "DELETE FROM Team WHERE team_id = ?";
                PreparedStatement stmt = conn.prepareStatement(sql);
                stmt.setInt(1, teamId);
                stmt.executeUpdate();

                JOptionPane.showMessageDialog(this, "Team deleted successfully!");

                // Update the table
                tableModel.addRow(new Object[]{"Delete", "Teams", teamId, "", ""});
                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));

                clearFields();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void clearFields() {
            teamIdField.setText("");
            teamNameField.setText("");
            coachIdField.setText("");
        }
    }

    // Implement the other pages (MatchesPage, PlayersPage, CoachesPage) in a
similar manner

    class MatchesPage extends JFrame {
        private static final String DB_URL = "jdbc:oracle:thin:@localhost:1521:xe";
        private static final String USERNAME = "lokesh";
        private static final String PASSWORD = "Lokesh";
        private JTextField matchIdField;
        private JTextField locationField;
        private JTextField team1IdField;
        private JTextField team2IdField;
        private DefaultTableModel tableModel;
        private JTable table;

        public MatchesPage(DefaultTableModel tableModel, JTable table) {
            this.tableModel = tableModel;
            this.table = table;
            setTitle("Matches Page");
            setSize(400, 200);

            JPanel panel = new JPanel(new FlowLayout());

            panel.add(new JLabel("Match ID:"));
            matchIdField = new JTextField(10);
            panel.add(matchIdField);

            panel.add(new JLabel("Location:"));
            locationField = new JTextField(10);
            panel.add(locationField);

            panel.add(new JLabel("Team 1 ID:"));
            team1IdField = new JTextField(10);
            panel.add(team1IdField);

            panel.add(new JLabel("Team 2 ID:"));
            team2IdField = new JTextField(10);
            panel.add(team2IdField);

            JButton okButton = new JButton("OK");
            okButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    int matchId = Integer.parseInt(matchIdField.getText());
                    fetchMatchDetails(matchId);
                }
            });
            panel.add(okButton);

            JButton insertButton = new JButton("Insert");
            insertButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    insertMatch();
                }
            });
            panel.add(insertButton);

            JButton modifyButton = new JButton("Modify");
            modifyButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    modifyMatch();
                }
            });
            panel.add(modifyButton);

            JButton deleteButton = new JButton("Delete");
            deleteButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    deleteMatch();
                }
            });
            panel.add(deleteButton);
            setLocationRelativeTo(null);

            add(panel);
            setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
    }

    private void fetchMatchDetails(int matchId) {
        try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
            String sql = "SELECT * FROM Match WHERE match_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, matchId);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                String matchLocation = rs.getString("match_location");
                int team1Id = rs.getInt("team1_id");
                int team2Id = rs.getInt("team2_id");

                locationField.setText(matchLocation);
                team1IdField.setText(String.valueOf(team1Id));
                team2IdField.setText(String.valueOf(team2Id));
            } else {
                JOptionPane.showMessageDialog(this, "Match not found!");
                clearFields();
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }


    private void insertMatch() {
        int matchId = Integer.parseInt(matchIdField.getText());
        String location = locationField.getText();
        int team1Id = Integer.parseInt(team1IdField.getText());
        int team2Id = Integer.parseInt(team2IdField.getText());
```

```
            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
                String sql = "INSERT INTO Match (match_id, match_location, team1_id,
team2_id) VALUES (?, ?, ?, ?)";
                PreparedStatement stmt = conn.prepareStatement(sql);
                stmt.setInt(1, matchId);
                stmt.setString(2, location);
                stmt.setInt(3, team1Id);
                stmt.setInt(4, team2Id);
                stmt.executeUpdate();

                JOptionPane.showMessageDialog(this, "Match inserted successfully!");

                // Update the table
                tableModel.addRow(new Object[]{"Insert", "Matches", matchId, location,
team1Id, team2Id});
                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));

                clearFields();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void modifyMatch() {
            int matchId = Integer.parseInt(matchIdField.getText());
            String location = locationField.getText();
            int team1Id = Integer.parseInt(team1IdField.getText());
            int team2Id = Integer.parseInt(team2IdField.getText());

            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
```

```
                String sql = "UPDATE Match SET match_location = ?, team1_id = ?,
team2_id = ? WHERE match_id = ?";
                PreparedStatement stmt = conn.prepareStatement(sql);
                stmt.setString(1, location);
                stmt.setInt(2, team1Id);
                stmt.setInt(3, team2Id);
                stmt.setInt(4, matchId);
                stmt.executeUpdate();

                JOptionPane.showMessageDialog(this, "Match modified successfully!");

                // Update the table
                tableModel.addRow(new Object[]{"Modify", "Matches", matchId,
location, team1Id, team2Id});
                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));

                clearFields();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void deleteMatch() {
            int matchId = Integer.parseInt(matchIdField.getText());

            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
                String sql = "DELETE FROM Match WHERE match_id = ?";
                PreparedStatement stmt = conn.prepareStatement(sql);
                stmt.setInt(1, matchId);
                stmt.executeUpdate();

                JOptionPane.showMessageDialog(this, "Match deleted successfully!");
                // Update the table
                tableModel.addRow(new Object[]{"Delete", "Matches", matchId, "", "",
"");
                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));

                clearFields();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void clearFields() {
            matchIdField.setText("");
            locationField.setText("");
            team1IdField.setText("");
            team2IdField.setText("");
        }
    }

    class PlayersPage extends JFrame {
        private static final String DB_URL = "jdbc:oracle:thin:@localhost:1521:xe";
        private static final String USERNAME = "lokesh";
        private static final String PASSWORD = "Lokesh";

        private JTextField playerIdField;
        private JTextField playerNameField;
        private JTextField teamIdField;
        private DefaultTableModel tableModel;
        private JTable table;

        public PlayersPage(DefaultTableModel tableModel, JTable table) {
            this.tableModel = tableModel;
            this.table = table;
            setTitle("Players Page");
            setSize(400, 150);

            JPanel panel = new JPanel(new FlowLayout());

            panel.add(new JLabel("Player ID:"));
            playerIdField = new JTextField(10);
            panel.add(playerIdField);

            panel.add(new JLabel("Player Name:"));
            playerNameField = new JTextField(10);
            panel.add(playerNameField);

            panel.add(new JLabel("Team ID:"));
            teamIdField = new JTextField(10);
            panel.add(teamIdField);

            JButton okButton = new JButton("OK");
            okButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                   int playerId = Integer.parseInt(playerIdField.getText());
                   fetchPlayerDetails(playerId);
                }
            });
            panel.add(okButton);

            JButton insertButton = new JButton("Insert");
            insertButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                   insertPlayer();
                }
            });
            panel.add(insertButton);
```

```
        JButton modifyButton = new JButton("Modify");
        modifyButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                modifyPlayer();
            }
        });
        panel.add(modifyButton);

        JButton deleteButton = new JButton("Delete");
        deleteButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                deletePlayer();
            }
        });
        panel.add(deleteButton);
        setLocationRelativeTo(null);

        add(panel);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }

    private void fetchPlayerDetails(int playerId) {
        try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
            String sql = "SELECT * FROM Player WHERE player_id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, playerId);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                String playerName = rs.getString("player_name");
                int teamId = rs.getInt("team_id");
                playerIdField.setText(String.valueOf(playerId));
                playerNameField.setText(playerName);
                teamIdField.setText(String.valueOf(teamId));
            } else {
                JOptionPane.showMessageDialog(this, "Player not found!");
                clearFields();
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    private void insertPlayer() {
        int playerId = Integer.parseInt(playerIdField.getText());
        String playerName = playerNameField.getText();
        int teamId = Integer.parseInt(teamIdField.getText());

        try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
            String sql = "INSERT INTO Player (player_id, player_name, team_id)
VALUES (?, ?, ?)";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, playerId);
            stmt.setString(2, playerName);
            stmt.setInt(3, teamId);
            stmt.executeUpdate();

            JOptionPane.showMessageDialog(this, "Player inserted successfully!");

            // Update the table
            tableModel.addRow(new Object[]{"Insert", "Players", playerId,
playerName, teamId});
            table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));
```

```
              clearFields();
      } catch (SQLException ex) {
              ex.printStackTrace();
            }
        }

        private void modifyPlayer() {
            int playerId = Integer.parseInt(playerIdField.getText());
            String playerName = playerNameField.getText();
            int teamId = Integer.parseInt(teamIdField.getText());

            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
              String sql = "UPDATE Player SET player_name = ?, team_id = ?
WHERE player_id = ?";
              PreparedStatement stmt = conn.prepareStatement(sql);
              stmt.setString(1, playerName);
              stmt.setInt(2, teamId);
              stmt.setInt(3, playerId);
              stmt.executeUpdate();

              JOptionPane.showMessageDialog(this, "Player modified successfully!");

              // Update the table
              tableModel.addRow(new Object[]{"Modify", "Players", playerId,
playerName, teamId});
              table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));

              clearFields();
            } catch (SQLException ex) {
              ex.printStackTrace();
            }
        }

        private void deletePlayer() {
            int playerId = Integer.parseInt(playerIdField.getText());

            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
                String sql = "DELETE FROM Player WHERE player_id = ?";
                PreparedStatement stmt = conn.prepareStatement(sql);
                stmt.setInt(1, playerId);
                stmt.executeUpdate();

                JOptionPane.showMessageDialog(this, "Player deleted successfully!");

                // Update the table
                tableModel.addRow(new Object[]{"Delete", "Players", playerId, "", ""});
                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));

                clearFields();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void clearFields() {
            playerIdField.setText("");
            playerNameField.setText("");
            teamIdField.setText("");
        }
    }

    class CoachesPage extends JFrame {
        private static final String DB_URL = "jdbc:oracle:thin:@localhost:1521:xe";
        private static final String USERNAME = "lokesh";
        private static final String PASSWORD = "Lokesh";

        private JTextField coachIdField;
        private JTextField coachNameField;
        private DefaultTableModel tableModel;
        private JTable table;

        public CoachesPage(DefaultTableModel tableModel, JTable table) {
            this.tableModel = tableModel;
            this.table = table;
            setTitle("Coaches Page");
            setSize(400, 150);

            JPanel panel = new JPanel(new FlowLayout());

            panel.add(new JLabel("Coach ID:"));
            coachIdField = new JTextField(10);
            panel.add(coachIdField);

            panel.add(new JLabel("Coach Name:"));
            coachNameField = new JTextField(10);
            panel.add(coachNameField);

            JButton okButton = new JButton("OK");
            okButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    int coachId = Integer.parseInt(coachIdField.getText());
                    fetchCoachDetails(coachId);
                }
            });
            panel.add(okButton);

            JButton insertButton = new JButton("Insert");
```

```
          insertButton.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                  insertCoach();
              }
          });
          panel.add(insertButton);

          JButton modifyButton = new JButton("Modify");
          modifyButton.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                  modifyCoach();
              }
          });
          panel.add(modifyButton);

          JButton deleteButton = new JButton("Delete");
          deleteButton.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                  deleteCoach();
              }
          });
          panel.add(deleteButton);
          setLocationRelativeTo(null);

          add(panel);
          setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      }

      private void fetchCoachDetails(int coachId) {
          try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
              String sql = "SELECT * FROM Coach WHERE coach_id = ?";
              PreparedStatement stmt = conn.prepareStatement(sql);
              stmt.setInt(1, coachId);
```

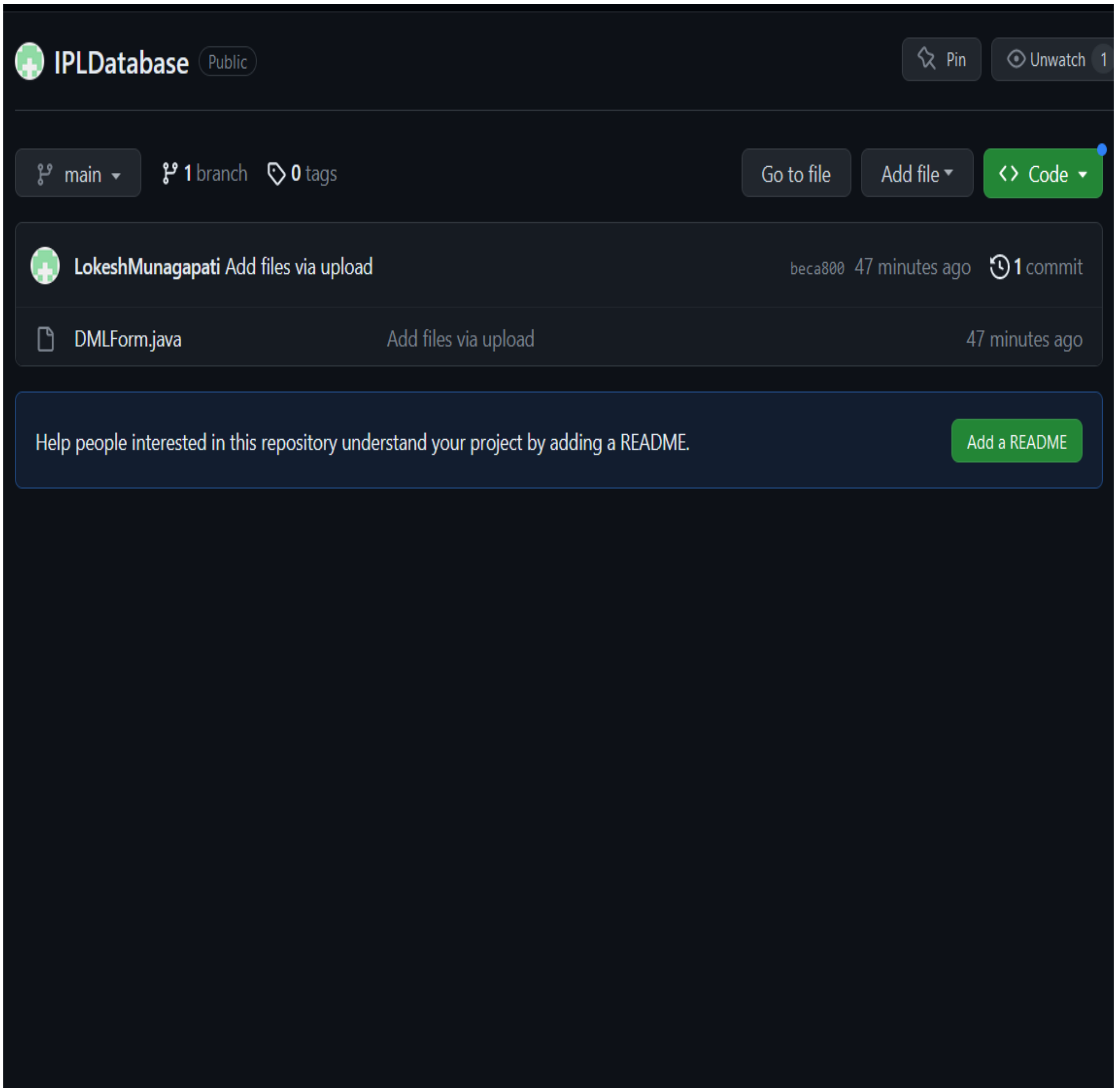
```
                ResultSet rs = stmt.executeQuery();

                if (rs.next()) {
                    String coachName = rs.getString("coach_name");

                    coachIdField.setText(String.valueOf(coachId));
                    coachNameField.setText(coachName);
                } else {
                    JOptionPane.showMessageDialog(this, "Coach not found!");
                    clearFields();
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void insertCoach() {
            int coachId = Integer.parseInt(coachIdField.getText());
            String coachName = coachNameField.getText();

            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
                String sql = "INSERT INTO Coach (coach_id, coach_name) VALUES (?,
?)";
                PreparedStatement stmt = conn.prepareStatement(sql);
                stmt.setInt(1, coachId);
                stmt.setString(2, coachName);
                stmt.executeUpdate();

                JOptionPane.showMessageDialog(this, "Coach inserted successfully!");

                // Update the table
                tableModel.addRow(new Object[]{"Insert", "Coaches", coachId,
coachName});
```

```
                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));

                clearFields();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void modifyCoach() {
            int coachId = Integer.parseInt(coachIdField.getText());
            String coachName = coachNameField.getText();

            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
                String sql = "UPDATE Coach SET coach_name = ? WHERE coach_id =
?";
                PreparedStatement stmt = conn.prepareStatement(sql);
                stmt.setString(1, coachName);
                stmt.setInt(2, coachId);
                stmt.executeUpdate();

                JOptionPane.showMessageDialog(this, "Coach modified successfully!");

                // Update the table
                tableModel.addRow(new Object[]{"Modify", "Coaches", coachId,
coachName});
                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));

                clearFields();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void deleteCoach() {
            int coachId = Integer.parseInt(coachIdField.getText());

            try (Connection conn = DriverManager.getConnection(DB_URL,
USERNAME, PASSWORD)) {
                String sql = "DELETE FROM Coach WHERE coach_id = ?";
                PreparedStatement stmt = conn.prepareStatement(sql);
                stmt.setInt(1, coachId);
                stmt.executeUpdate();

                JOptionPane.showMessageDialog(this, "Coach deleted successfully!");

                // Update the table
                tableModel.addRow(new Object[]{"Delete", "Coaches", coachId, ""});
                table.scrollRectToVisible(table.getCellRect(table.getRowCount() - 1, 0,
true));

                clearFields();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        private void clearFields() {
            coachIdField.setText("");
            coachNameField.setText("");
        }
    }
```

GitHub Links and Folder Structure

GitHub link: https://github.com/LokeshMunagapati/IPLDatabase

Folder Structure:



LokeshMunagapati Add files via upload beca800 47 minutes ago 1 commit

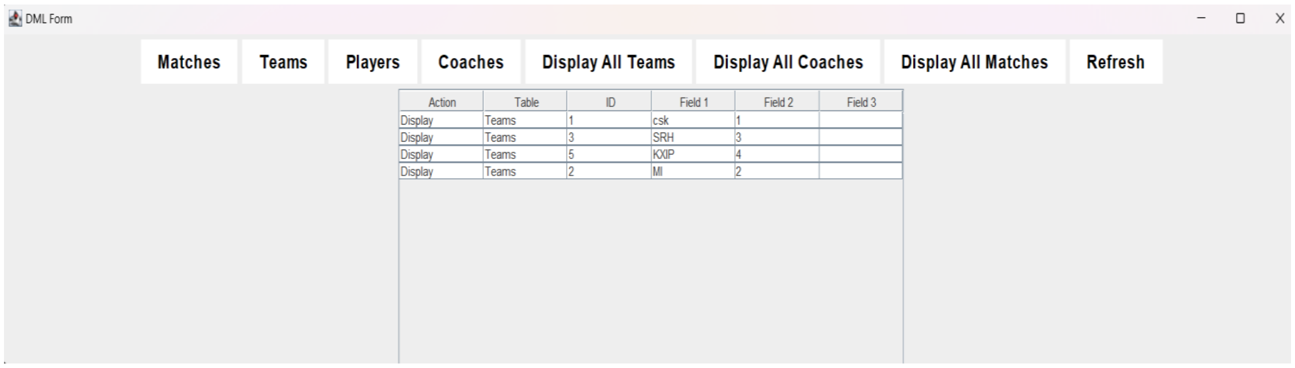
DMLForm.java Add files via upload 47 minutes ago

Help people interested in this repository understand your project by adding a README.

1602-21-737-030
M. Lokesh

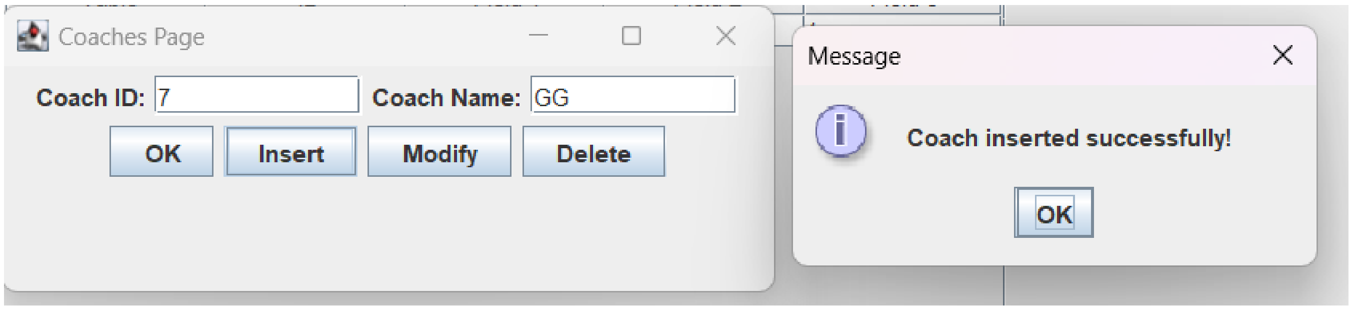
TESTING

Main Interface:



Action	Table	ID	Field 1	Field 2	Field 3
Display	Teams	1	csk	1	
Display	Teams	3	SRH	3	
Display	Teams	5	KXIP	4	
Display	Teams	2	MI	2	

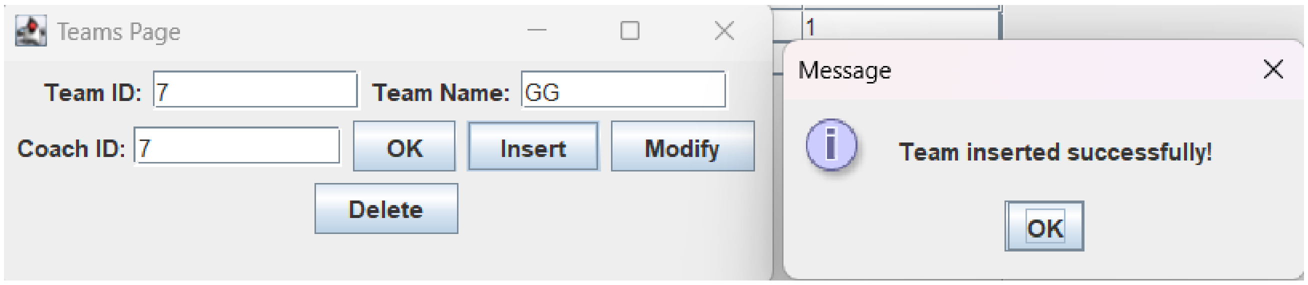
Insert operations : (Team, Match, Player, Coach)


Coaches Page

Coach ID: 7 Coach Name: GG

Message

Coach inserted successfully!

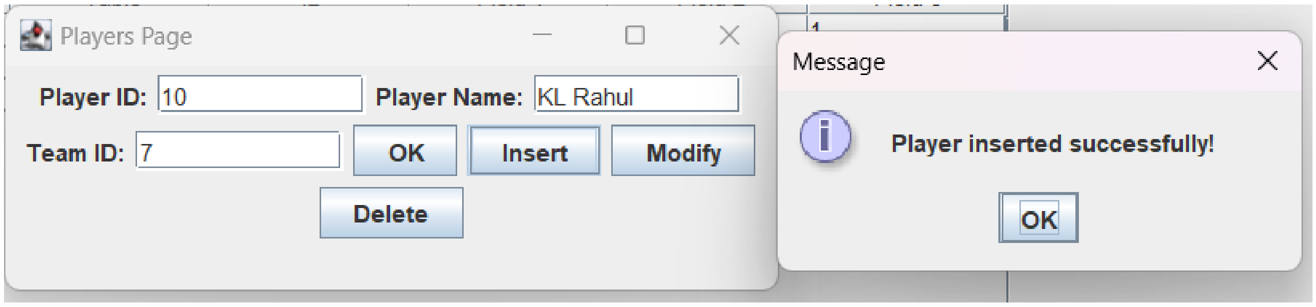

Teams Page

Team ID: 7 Team Name: GG

Coach ID: 7

Message

Team inserted successfully!

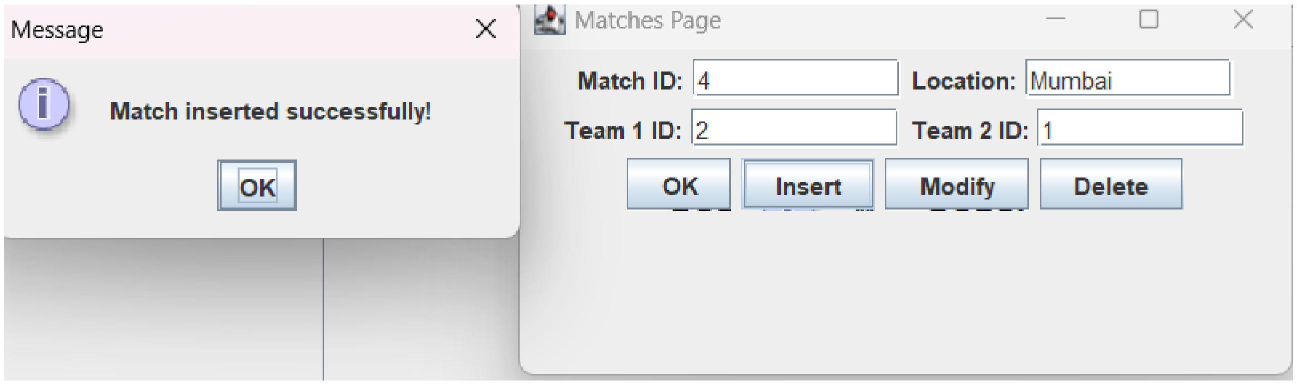

Players Page

Player ID: 10 Player Name: KL Rahul

Team ID: 7

Message

Player inserted successfully!


Message

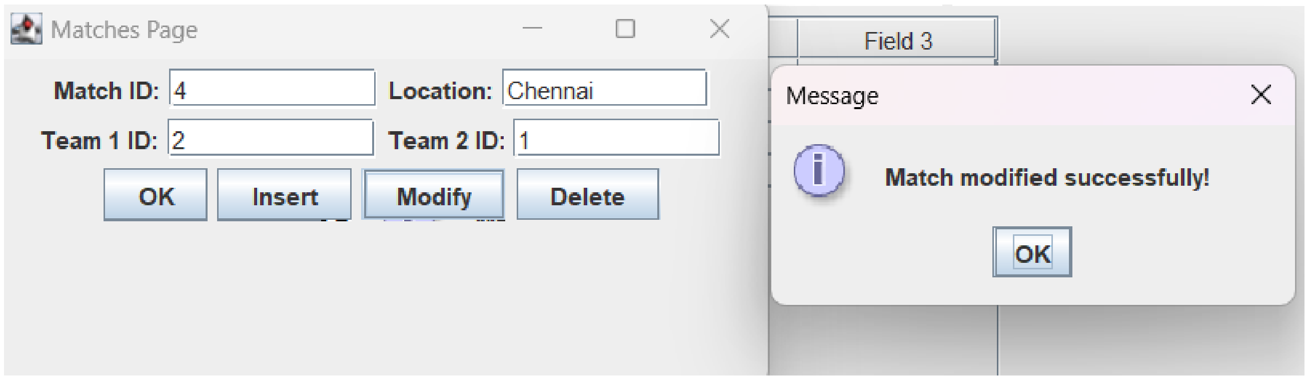
Match inserted successfully!

Matches Page

Match ID: 4 Location: Mumbai

Team 1 ID: 2 Team 2 ID: 1

Modify:

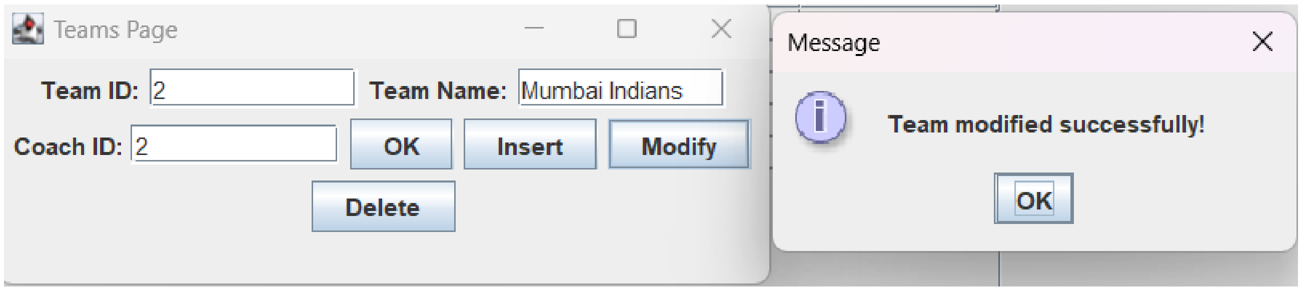

Matches Page

Match ID: 4 Location: Chennai

Team 1 ID: 2 Team 2 ID: 1

Message

Match modified successfully!

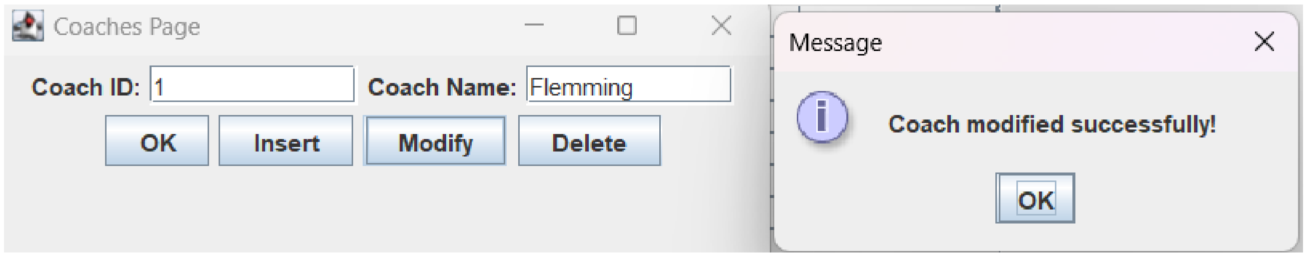

Teams Page

Team ID: 2 Team Name: Mumbai Indians

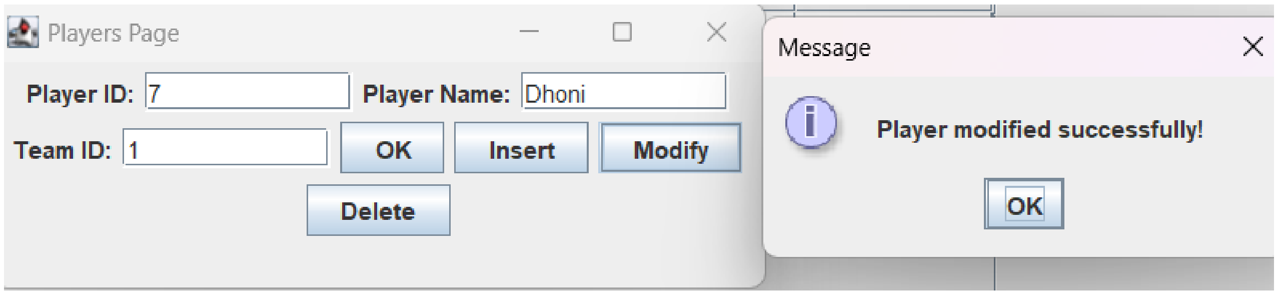
Coach ID: 2

Message

Team modified successfully!

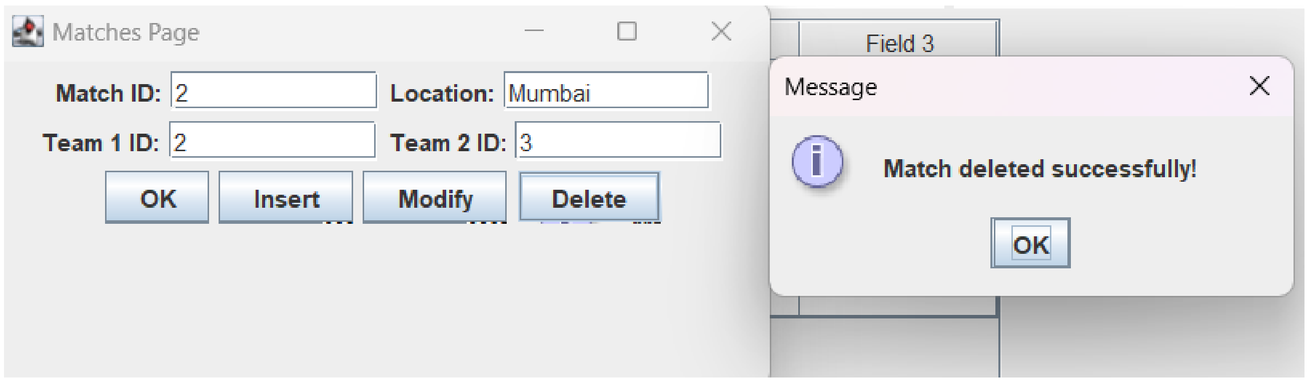


Coach modified successfully!



Player modified successfully!

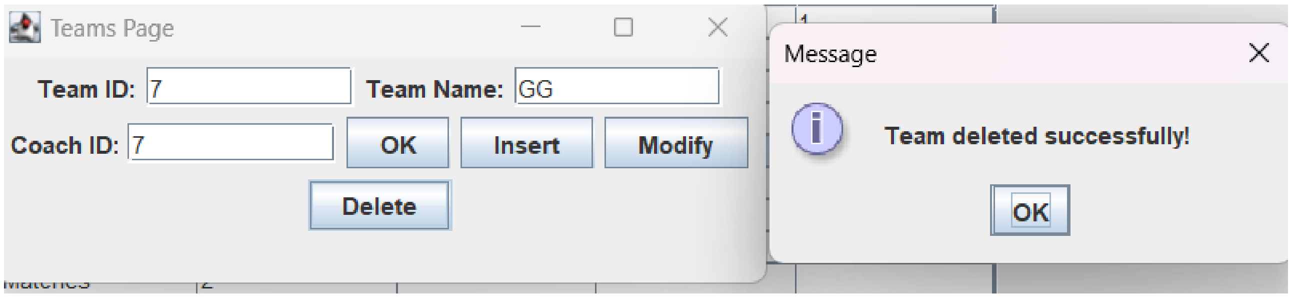
Delete:


Matches Page

Match ID: 2
Location: Mumbai
Team 1 ID: 2
Team 2 ID: 3

Message

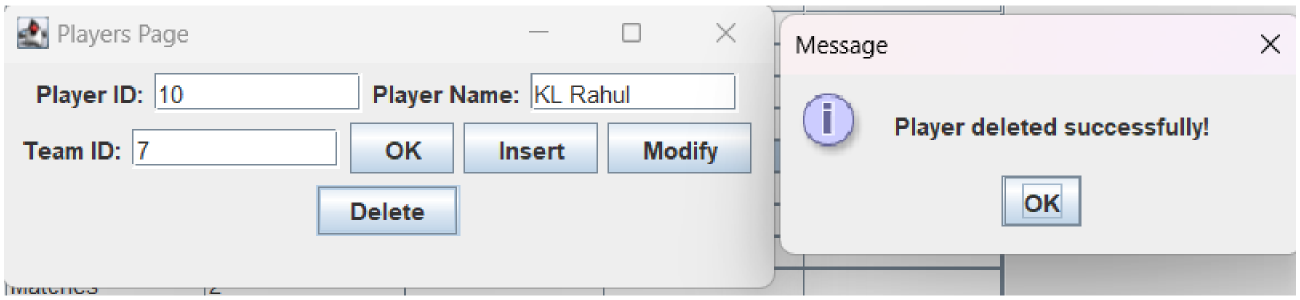
Match deleted successfully!


Teams Page

Team ID: 7
Team Name: GG
Coach ID: 7

Message

Team deleted successfully!

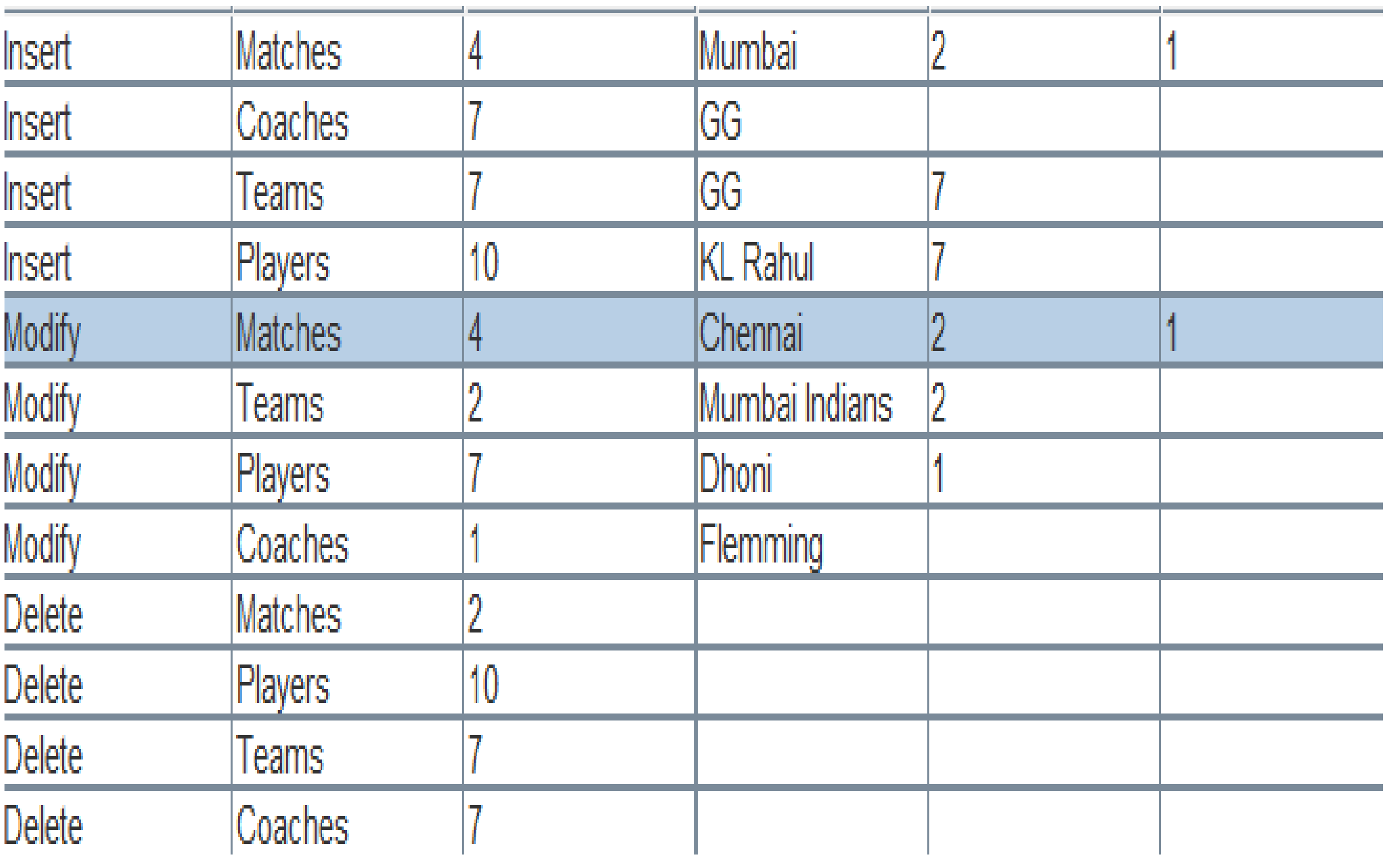

Players Page

Player ID: 10
Player Name: KL Rahul
Team ID: 7

Message

Player deleted successfully!

Final Display:



Insert	Matches	4	Mumbai	2	1
Insert	Coaches	7	GG		
Insert	Teams	7	GG	7	
Insert	Players	10	KL Rahul	7	
Modify	Matches	4	Chennai	2	1
Modify	Teams	2	Mumbai Indians	2	
Modify	Players	7	Dhoni	1	
Modify	Coaches	1	Flemming		
Delete	Matches	2			
Delete	Players	10			
Delete	Teams	7			
Delete	Coaches	7			

RESULTS

I have successfully completed the mini-project ***“IPL Database”***.

Future Work: Here are some potential areas for future work and enhancements to consider for the "DML Form" project:

1. User Authentication: Implement a login system to ensure secure access to the application and database.
2. Validation and Error Handling: Enhance the application by adding data validation checks and error handling mechanisms to ensure data integrity and improve the user experience.
3. Search and Filter Functionality: Implement search and filtering options to allow users to search for specific records based on certain criteria.
4. Sorting and Pagination: Enable sorting of table columns and implement pagination for large datasets to improve performance and user experience.
5. Reports and Data Analysis: Add features to generate reports and perform data analysis, such as aggregations, statistics, and visualizations.
6. Database Migration: Implement database migration scripts or tools to handle database schema changes and versioning.
7. User Interface Enhancements: Improve the visual design of the GUI, enhance the user interface components, and make it more intuitive and user-friendly.
8. Support for Additional Database Systems: Extend the application to support other database systems, such as MySQL, PostgreSQL, or SQL Server, by modifying the database connection code accordingly.
9. Multithreading: Implement multithreading to handle concurrent database operations and improve the application's performance and responsiveness.
10. Error Logging and Monitoring: Integrate logging frameworks to track and log errors, exceptions, and application events for debugging and monitoring purposes.

By focusing on these areas, the "DML Form" project can be expanded into a more comprehensive and robust DBMS application, offering additional features and improved functionality for managing and analysing database data

REFERENCES

- https://docs.oracle.com/javase/7/docs/api/
- https://www.javatpoint.com/java-swing
- https://stackoverflow.com/

Summary of Annotations