
*Green University of Bangladesh*

*Department of Computer Science and Engineering (CSE)*
*Semester: (Fall, Year: 2023), B.Sc. in CSE (Day)*

---

# Tiger Cricket Score

---

*Course Title: Database Systems Lab*
*Course Code: CSE 210*
*Section: D12*

Students Details


| Name | **ID** |
|---|---|
| Md. Moshiur Rahman | 221902324 |


*Submission Date: 28/12/2023*
*Course Teacher's Name: Wahia Tasnim*

[For teachers use only: Don't write anything inside this box]

| **Lab Project Status** | |
|---|---|
| **Marks:** | **Signature:** |
| **Comments:** | **Date:** |

# Contents

# Chapter 1

# Introduction

This project creates a database to track player scores in Tiger cricket matches over time for research purposes. The database system was created using MySQL, SQL, and phpMyAdmin.

## 1.1 Overview

In this challenge, as a pupil, create a database to track player scores in Tiger Cricket tournaments over time for analysis. MySQL, SQL, and phpMyAdmin were used to build the database system.

## 1.2 Motivation

As a passionate member of Tiger Cricket , I want to develop a plan to digitize score-keeping which is currently a manual, unreliable process. Proper records will allow for investigation.

## 1.3 Problem Definition

### 1.3.1 Problem Statement

With paper scorecards recording scores and player profiles, analysis is nearly impossible for the coaching staff.

### 1.3.2 Complex Engineering Problem

Key challenges include creating Entity Relationship diagrams, implementing a normalized database schema in MySQL, and writing queries to handle overlapping player and match data

Table 1.1: Summary of the attributes touched by the mentioned projects

| **Name of the P Attributess** | **Explain how to address** |
|---|---|
| **P1:** Depth of knowledge required | This project required studying database concepts like normalization, integrity constraints, relationships and mappings between logical and physical schemas. Both SQL and system design skills were needed. |
| **P2:** Range of conflicting requirements | Requirements like data integrity vs query performance are often conflicting that need trade-off decisions. —- |
| **P3:** Depth of analysis required | — |
| **P4:** Familiarity of issues | Common data related issues like redundancy, inconsistency and data loss need mitigation through integrity constraints and backup strategies. |
| **P5:** Extent of stakeholder involvement and conflicting requirements | —- |
| **P6:** Interdependence | players, matches and scores data are heavily interdependent. This increases overall complexity for storage, access and updates |

Since attribute 1,2,4,5 can be touched by my project "Tiger Cricket Score" thats why my project can be defined as a complex engineering problem.

## 1.4 Design Goals

1. Centralize records digitally 2. Allow easy update of player performance data over seasons

## 1.5 Application

The players, matches and scores data are heavily interdependent. This increases overall complexity for storage, access and updates.

# Chapter 2

# Requirements

## 2.1 Software Requirement Specifications

A major element in building a system is the section of compatible software since the software in the market is experiencing in geometric progression. Selected software should be acceptable by the firm and one user as well as it should be feasible for the system. This document gives a detailed description of the software requirement specification. The study of requirement specification is focused specially on the functioning of the system. It allows the developer or analyst to understand the system, function to be carried out the performance level to be obtained and corresponding interfaces to be established. Front End : JAVA (Swings)

Back End : XAMPP server, MySQL

Operation System : Windows 10

IDE : NetBeans/ Visual Studio Code

## 2.2 Hardware Requirement Specifications

The section of hardware configuration is an important task related to the software development insufficient random-access memory may affect adversely on the speed and efficiency of the entire system. The process should be powerful to handle the entire operations. The hard disk should have sufficient capacity to store the file and application Processor : Intel PentiumT4200/ Intel Core Duo 2.0 GHz / more

RAM : Minimum 1 GB RAM capacity

Hard disk : Minimum 40 GB ROM capacity

Cache Memory : L2-1 MB (128KB - 8MB)

GPU : Intel HD Graphics

# Chapter 3

# Entity Relationship Diagram

3.1 ER Diagram

An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems. An entity is a piece of data-an object or concept about which data is stored. The cardinality or fundamental principle of one data aspect with respect to another is a critical feature. The relationship of one to the other must be precise and exact between each other in order to explain how each aspect links together. In simple words Cardinality is a way to define the relationship between two entities. The following are the notations of the ER diagram:

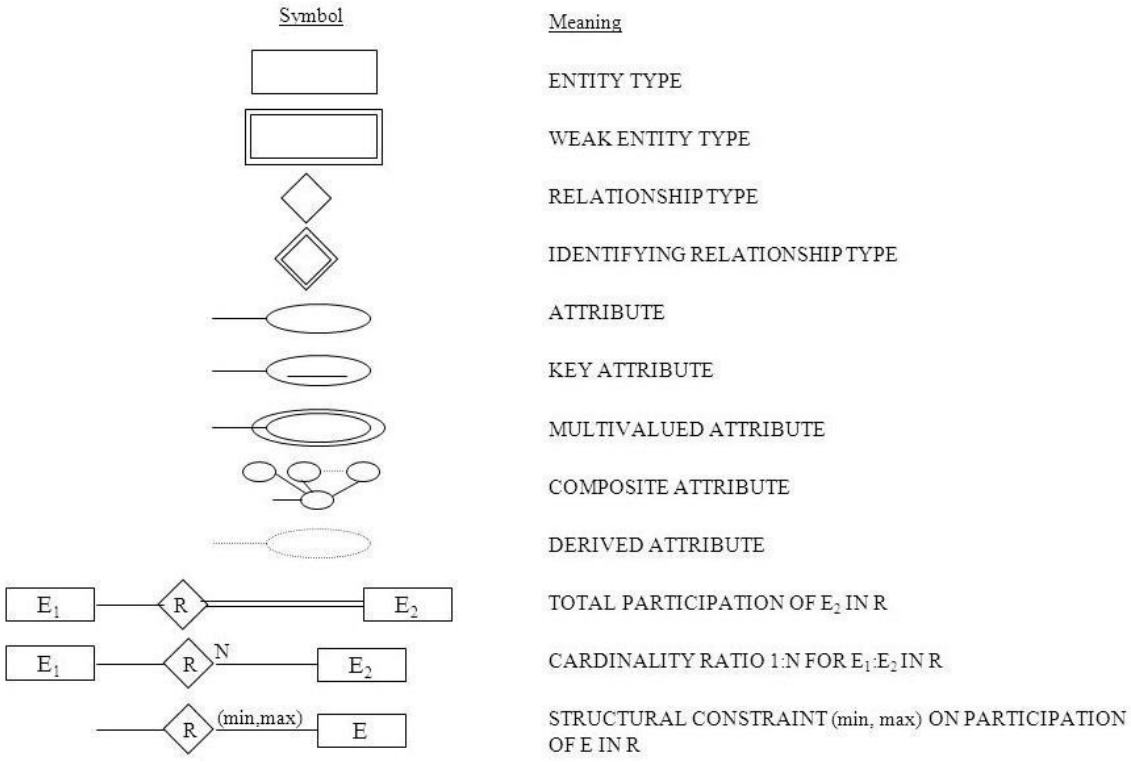


**Figure 3.1 Notation for ER Diagram**

Cardinalities can denote that an entity is optional (for example, an employee rep

could have no customers or could have many) or mandatory (for example, there must be at least one product listed in an order. The four main cardinal relationships are: • One-to-one (1: 1) - For example, each customer in a database is associated with one mailing address. • One-to-many (1: N) - For example, a single customer might place an order for multiple products. The customer is associated with multiple entities, but all those entities have a single connection back to the same customer. • Many-to-one (N: 1) – For example, many employees will have only one manager above them but one manager can have many employees below him. • Many-to-many (M: N) - For example, at a company where all call center agents work with multiple customers, each agent is associated with multiple customers, and multiple customers might also be associated with multiple agents.

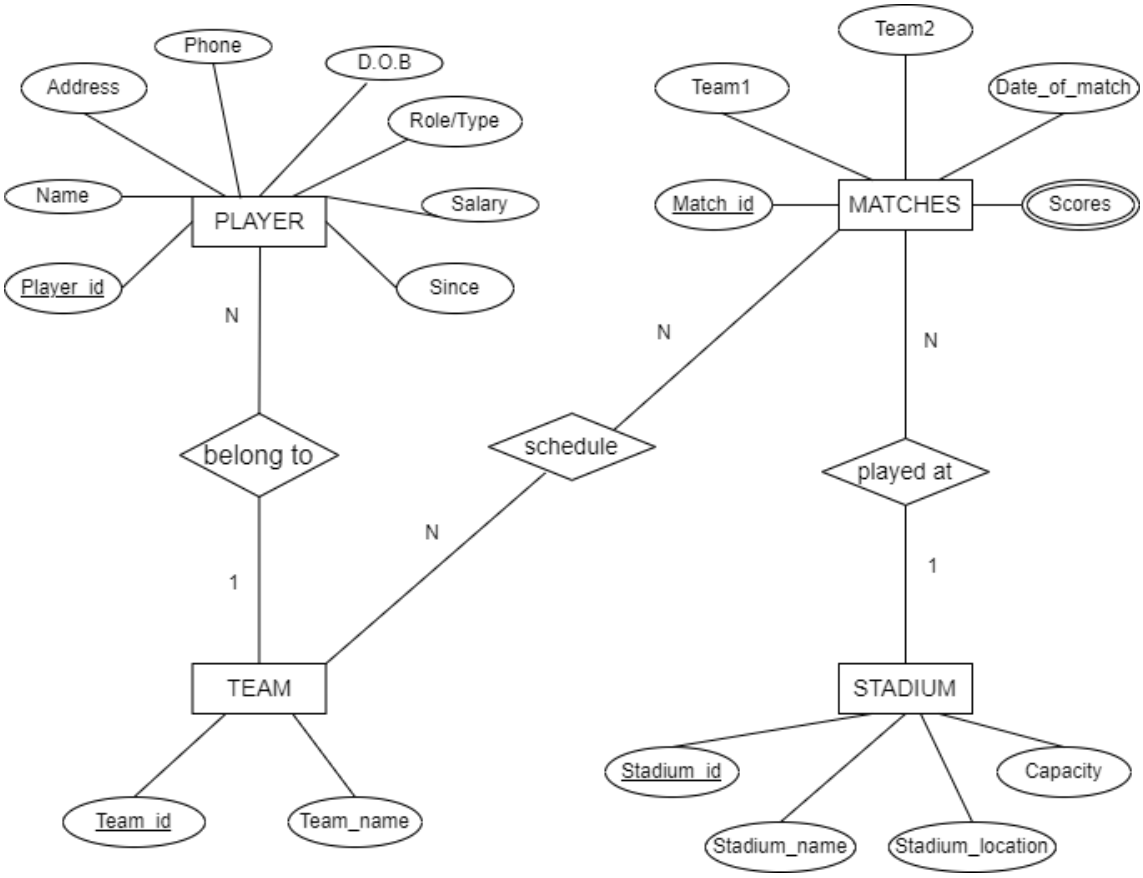


**Figure 3.2 ER-Diagram**

# Chapter 4

# Schema Diagram

## 4.1 SCHEMA DIAGRAM

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful. It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information. A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed. A database schema can be divided broadly into two categories: //break • Physical Database Schema - This schema pertains to the actual storage of data and its form of storage like files. indices, etc. It defines how the data will be stored in a secondary storage. • Logical Database Schema – This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views and integrity constraints.

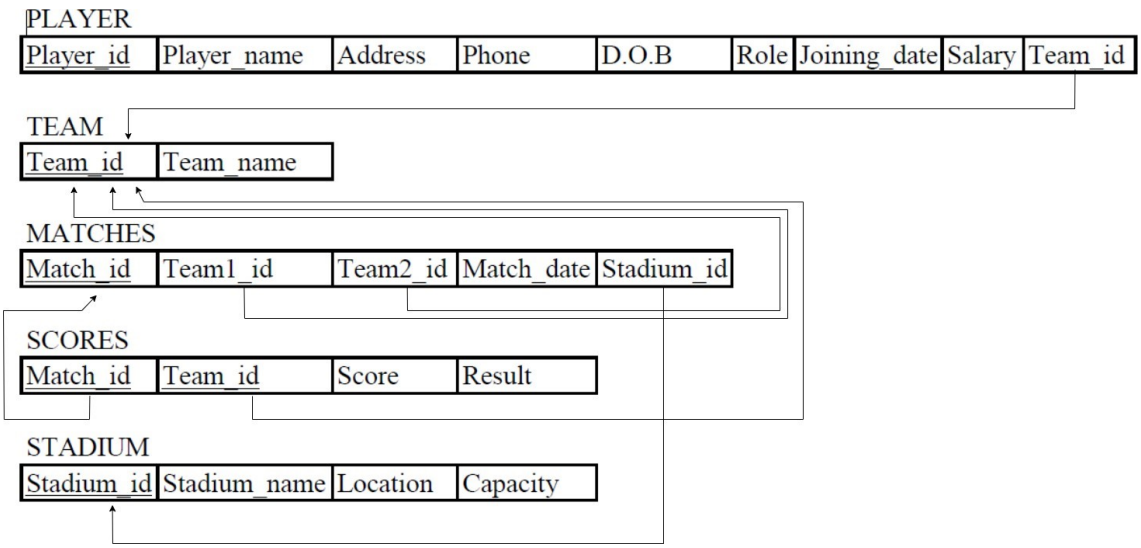


**Figure 4.1 Schema diagram**

# Chapter 5

# Implementation

### 5.0.1 Backend Implementation: Database

#### Team Table

create table team( team id int primary key, team name varchar(255) unique

| team_id | team_name |
|---|---|
| 22222 | Chennai Super King |
| 22225 | Delhi Capitals |
| 22226 | Goa Assocation |
| 22227 | Lucknow Franchise |
| 22224 | Mumbai Indians |
| 22223 | Royal Challengers |

Figure 5.1: Team

#### Player Table

create table player( player id int primary key, name varchar(255), address varchar(255) not null, phone no int, dob date, role varchar(20), since date, salary int, team id int, foreign key(team id) references team(team id) on delete set null );

| player_id | name | address | phone_no | dob | role | since | salary | team_id |
|---|---|---|---|---|---|---|---|---|
| 11111 | MS Dhoniii | Chennai | 9658326542 | 1975-01-12 | Wicket Keeper | 2010-01-06 | 10000000 | 22222 |
| 11112 | Jadeja | Chennaii | 5326457632 | 2015-01-06 | All Rounder | 2019-01-08 | 800000 | 22222 |
| 11113 | Mohin Alii | Chennai | 8657653254 | 1978-01-11 | All Ronder | 2015-01-14 | 80000000 | 22222 |
| 11114 | Virat Kholi | Bangalore | 9865732546 | 1988-01-19 | Batsman | 2003-01-14 | 170000000 | 22223 |
| 11115 | Devdutt Padikkal | Bangalore | 9865325325 | 1994-01-12 | Left-Handed Batsman | 2015-01-13 | 120000000 | 22223 |

Figure 5.2: Player Table

#### Stadium Table

:

create table stadium( stadium id int primary key, stadium name varchar(255) not null, stadium location varchar(255), capacity int

| stadium_id | stadium_name | stadium_location | capacity |
|---|---|---|---|
| 55555 | M. Chinnaswamy Stadium | Bangalore | 40000 |
| 55556 | Eden Gardens | Kolkata | 80000 |
| 55557 | Narendra Modi Stadium | Ahmedabad | 132000 |
| 55558 | Rajiv Gandhi International Cricket Stadium | Hyderabad | 55000 |
| 55559 | Wanderers Stadium | Mumbai | 33000 |
| 55560 | Green Park Stadium | Kanpur | 33000 |

Figure 5.3: Stadium table

**Matches Table**

: create table matches( match id int primary key, team1 id int not null, team2 id int not null, match date date, stadium id int, foreign key(stadium id) references stadium(stadium id) on delete set null );

| match_id | team1_id | team2_id | match_date | stadium_id |
|---|---|---|---|---|
| 33333 | 22222 | 22223 | 2022-01-03 | 55555 |
| 33334 | 22224 | 22225 | 2022-01-04 | 55556 |
| 33335 | 22225 | 22226 | 2022-01-05 | 55557 |
| 33336 | 22227 | 22226 | 2022-01-06 | 55557 |
| 33337 | 22222 | 22225 | 2022-01-07 | 55558 |
| 33338 | 22223 | 22226 | 2022-01-08 | 55559 |

Figure 5.4: Matches table

### 5.0.2 Scores Table

create table scores( match id int, team id int primary key(match id, team id), scores int, result varchar(20) not null, foreign key(match id) references matches(match id) on delete cascade, foreign key(team id) references team(team id) on delete cascade

## 5.1 Frontend Implementation

The backend and frontend are connected through object oriented programming language. The entire Front End is implemented using the Swing framework of Java. And the backend used in the project is MySQL.

| match_id | team_id | scores | result |
|---|---|---|---|
| 33333 | 22222 | 175 | Won |
| 33333 | 22223 | 170 | Lost |
| 33334 | 22224 | 165 | Lost |
| 33334 | 22225 | 166 | Won |
| 33335 | 22225 | 221 | Won |
| 33335 | 22226 | 215 | Lost |

Figure 5.5: Scores Table

**Source Code - Login.java**

```
private void loginActionPerformed(java.awt.event.ActionEvent evt) {

  if (username.getText().equals("admin")
      && password.getText().equals("pass")) {
    setVisible(false);
    new Home().setVisible(true);

  } else {
    JOptionPane.showMessageDialog(null,
      "Incorrect Username or Password");
  }

}

private void exitActionPerformed(java.awt.event.ActionEvent evt) {

  int a = JOptionPane.showConfirmDialog(null,
    "Do you want to close Application",
    "Select", JOptionPane.YES_NO_OPTION);

  if (a == 0)
    System.exit(0);

}

private void LogoutActionPerformed(java.awt.event.ActionEvent evt) {

  int a = JOptionPane.showConfirmDialog(null,
    "Do you want to Logout",
    "Select", JOptionPane.YES_NO_OPTION);
```

```
  if (a == 0) {
    setVisible(false);
    new Login().setVisible(true);
  }
}


\\section{Source Code - {ConnectionProvider.java}}

\begin{verbatim}


\begin{verbatim}
package CricketManagementSystem;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ConnectionProvider {

public static Connection getConnection() {


try {
  Class.forName("com.mysql.cj.jdbc.Driver");

  Connection con = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/cricket_database",
    "root",
    ""
  );

  return con;

} catch (Exception e) {
  System.out.println("Error: " + e);
}

return null;

}

}

private void saveActionPerformed(ActionEvent evt) {
```

```
int playerId = Integer.parseInt(player_id.getText());

String playerName = player_name.getText();

String addr = address.getText();

BigInteger phon = new BigInteger(phone.getText());

SimpleDateFormat dFormat = new SimpleDateFormat("yyyy-MM-dd");

String birthDate = dFormat.format(dob.getDate());

String role = role.getText();

String joiningDate = dFormat.format(joining_date.getDate());

String sal = salary.getText();

String teamId = team_id.getText();

try {

Connection con = ConnectionProvider.getConnection();

Statement st = con.createStatement();

st.executeUpdate("INSERT INTO player VALUES('" + playerId + "', '" +
 playerName + "', '" + addr + "', '" + phon + "', '" +
birthDate + "', '" + role + "', '" + joiningDate + "', '" +
sal + "', '" + teamId + "')");

JOptionPane.showMessageDialog(this, "Successfully Saved");

setVisible(false);

new AddUpdateDeletePlayer().setVisible(true);

} catch (SQLException ex) {

Logger.getLogger(AddUpdateDeletePlayer.class.getName()).log(Level.SEVERE, null, ex

}

}

\\subsection{Show Palyer Details}
```

```
PreparedStatement pst;
ResultSet rs;
DefaultTableModel d;
public void
userLoad ()
{
  int count;
  try
  {
    Connection con = ConnectionProvider.getConnection ();
    pst = con.prepareStatement ("select * from player");
    rs = pst.executeQuery ();
    ResultSetMetaData rsd = rs.getMetaData ();
    count = rsd.getColumnCount ();
    d = (DefaultTableModel) tblPlayer.getModel ();
    d.setRowCount (0);
    while (rs.next ())
      {
Vector v2 = new Vector ();
for (int i = 1; i <= count; i++)
  {
    v2.add (rs.getString ("player_id"));
    v2.add (rs.getString ("name"));
    v2.add (rs.getString ("address"));
    v2.add (rs.getString ("phone_no"));
    v2.add (rs.getString ("dob"));
    v2.add (rs.getString ("role"));
    v2.add (rs.getString ("since"));
    v2.add (rs.getString ("salary"));
    v2.add (rs.getString ("team_id"));
  }
d.addRow (v2);
      }
  } catch (SQLException ex)
  {
    Logger.getLogger (ShowPlayerDetails.class.getName ()).log (Level.SEVERE,
    null, ex);
  }
}
```

# Chapter 6

# Snapshots

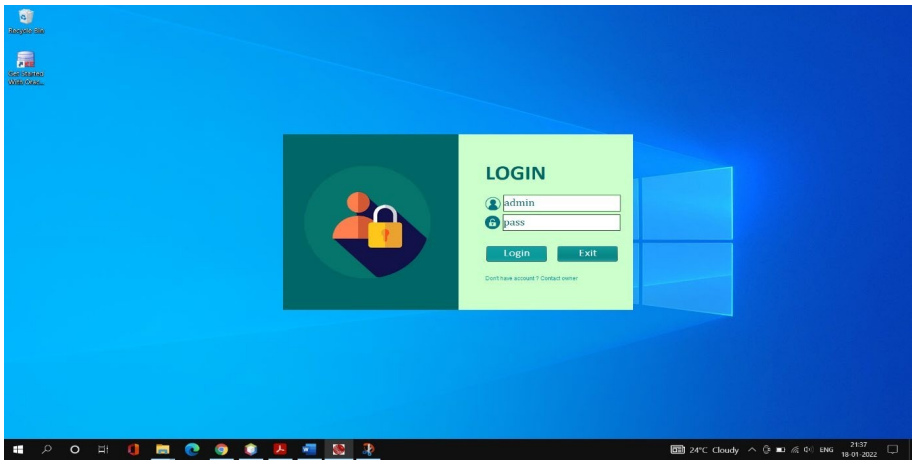


Figure 6.1: Admin Login Panel
It shows admin login page which gives authentication to enter into the Admin page.


Figure 6.2: Home page
In the above Figure 6.2, shows which allows to add, update and delete of players, teams, matches, score, stadiums and about page.


Figure 6.3: Home page

In the above Figure 6.2, shows which allows to add, update and delete of players, teams, matches, score, stadiums and about page.

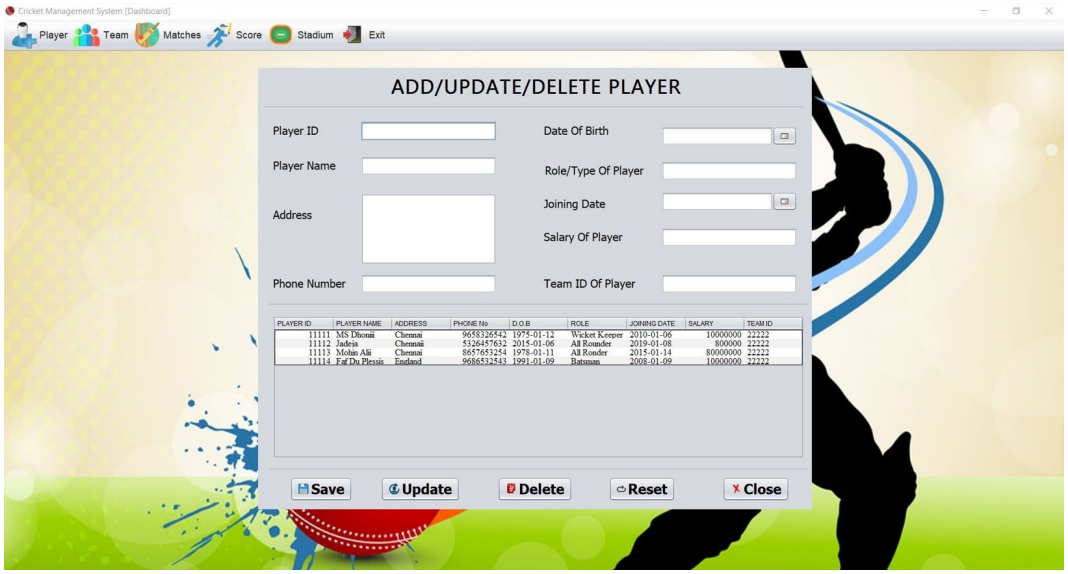


Figure 6.4: Add/ Update/ Delete player

It show player window which allows admin to add, update and delete the Player details

Add/ Update/ Delete Team

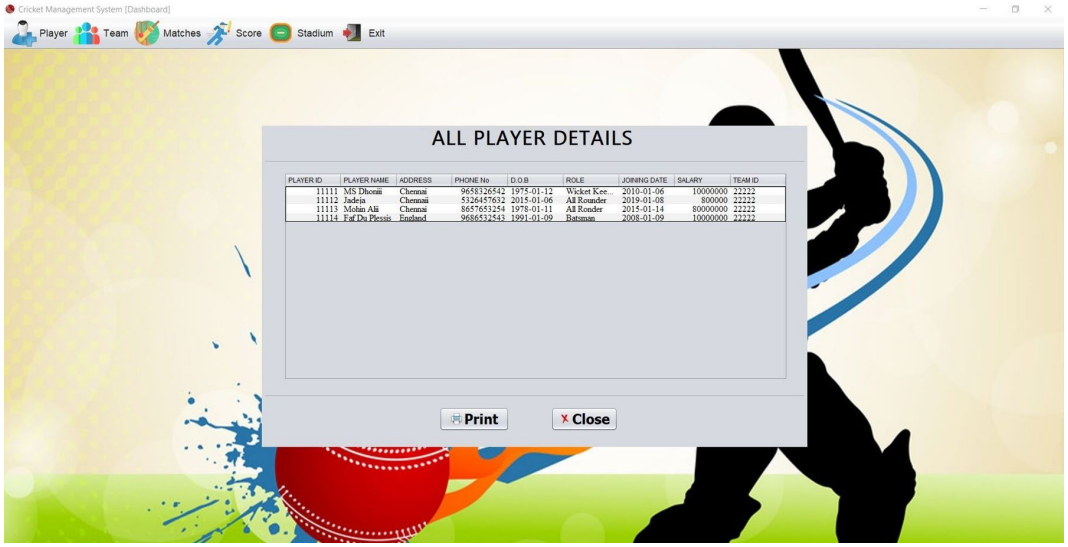


Figure 6.5: Player details

The above shows player details window which allows admin to print the details of player from the connected printer or just can save as pdf document

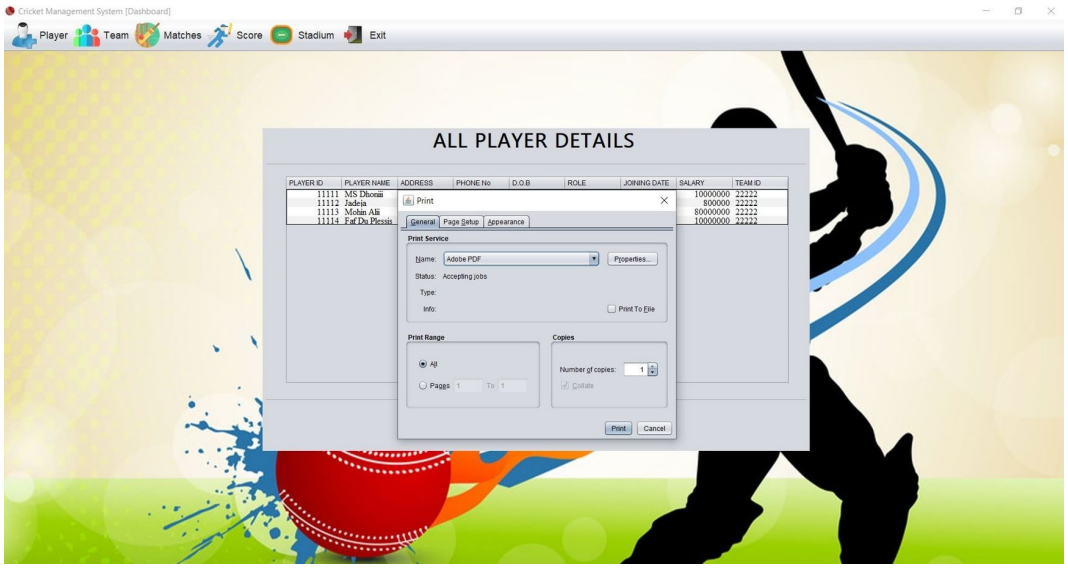


Figure 6.6: Print player details

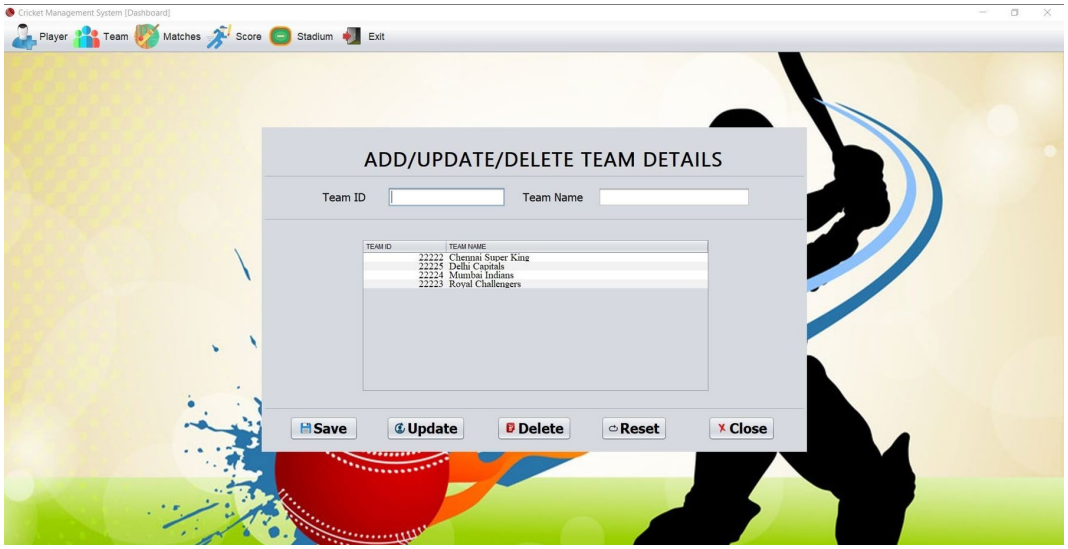


Figure 6.7: Add/ Update/ Delete Team
It show team window which allows admin to add, update and delete the team details

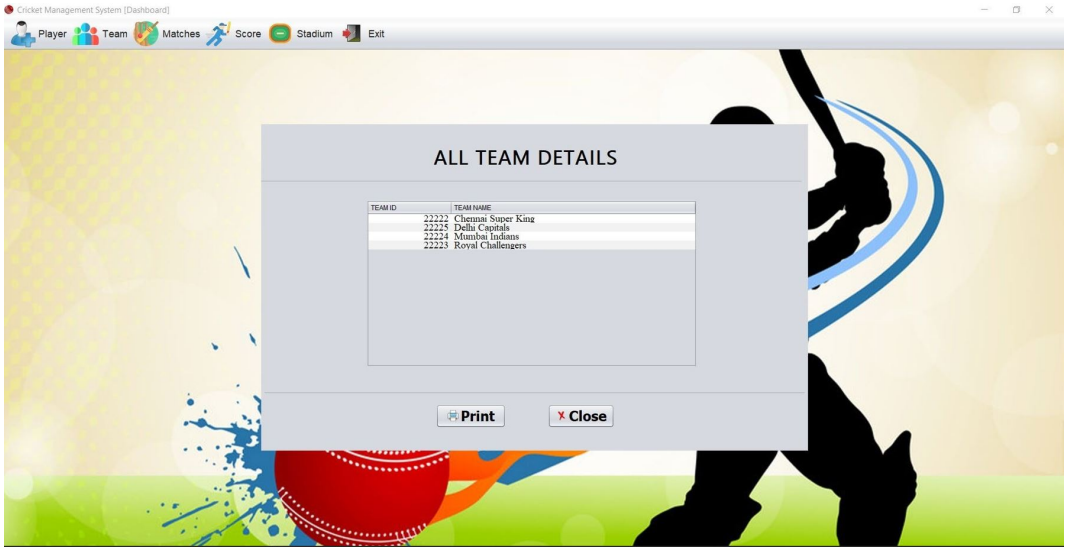


Figure 6.8: Team Details
It shows team details window which allows admin to print the details of team from the connected printer or just can save as pdf document

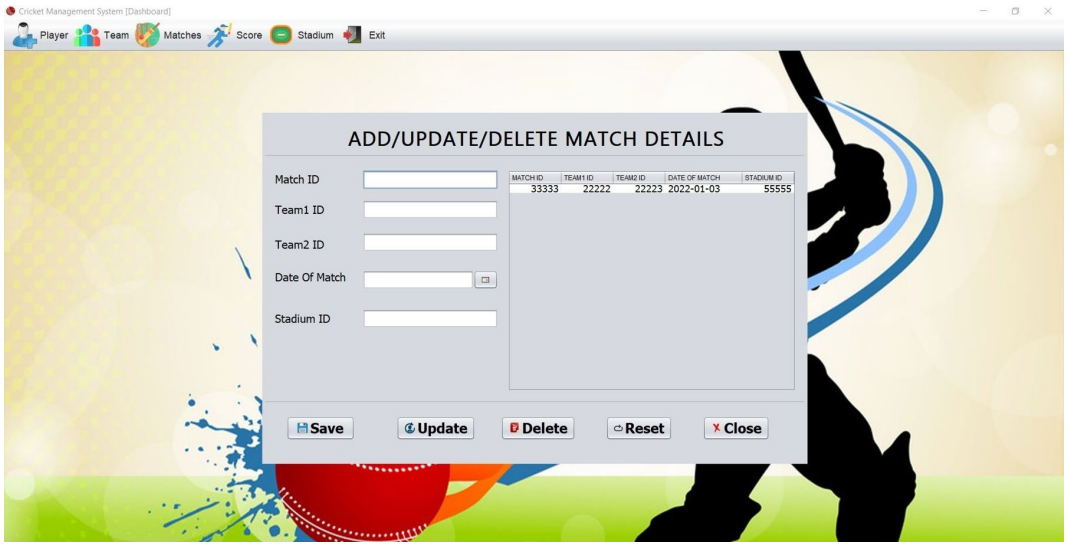


Figure 6.9: Add/ Update/ Delete match

It allows user to schedules the matches between the teams and also can update or delete a scheduled match document

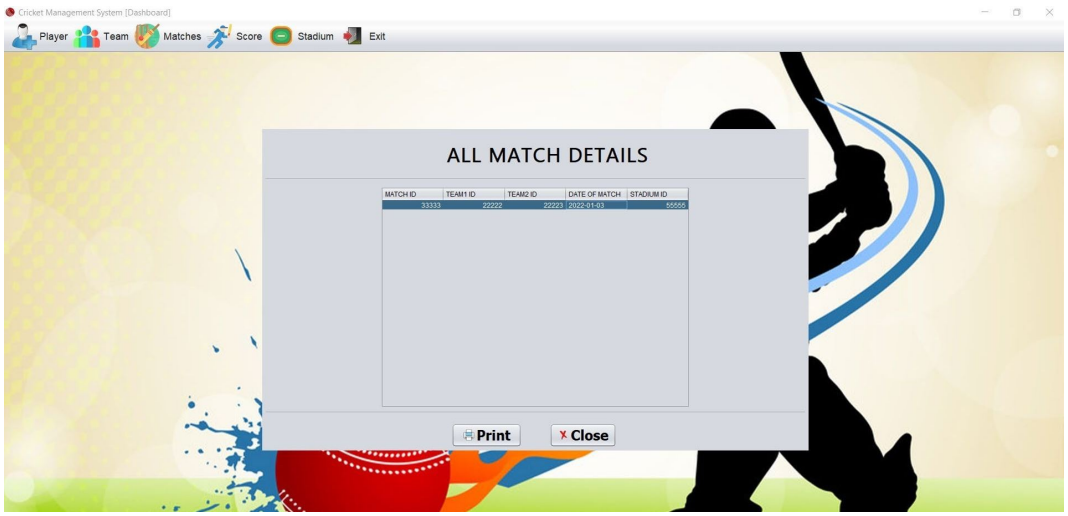


Figure 6.10: Match details

, It shows match details window which allows admin to print the details of match from the connected printer or just can save as pdf document.document

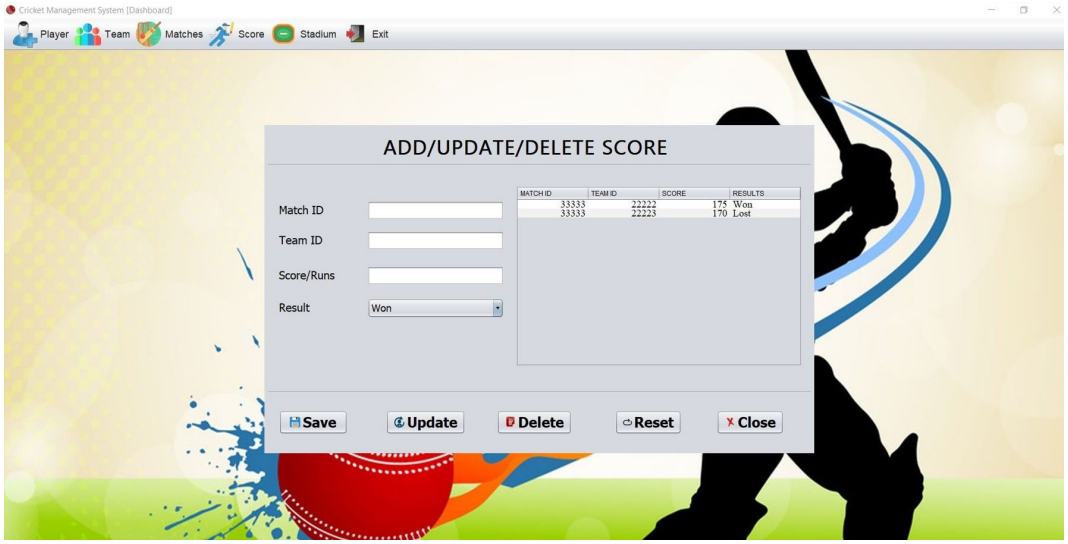


Figure 6.11: Add/ Update/ Delete score

It shows score window which allows admin to add score of a particular team and can also update or delete the score.

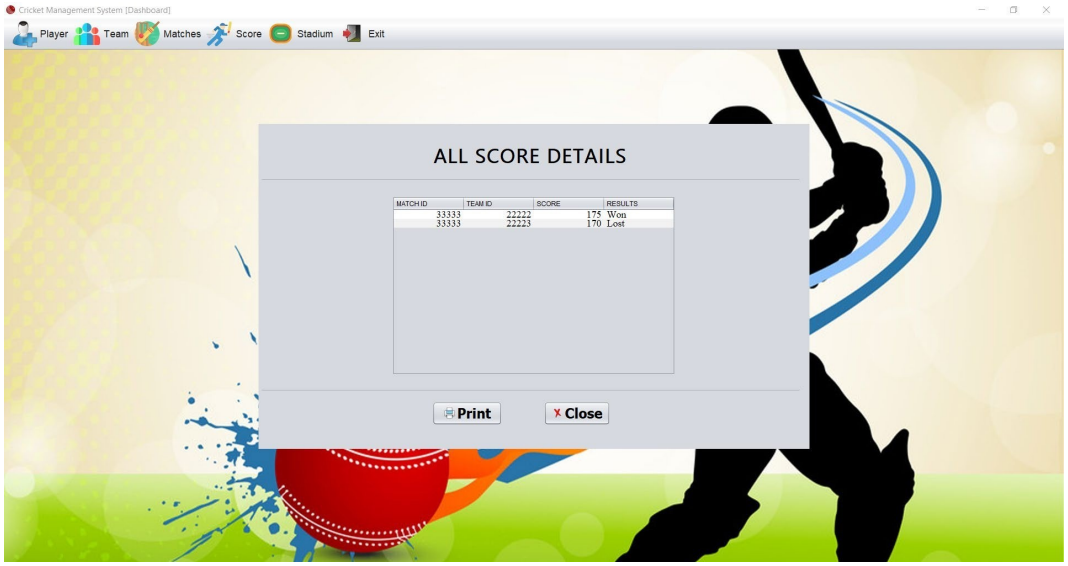


Figure 6.12: Score details

It shows score details window which allows admin to print the details of score from the connected printer or just can save as pdf document

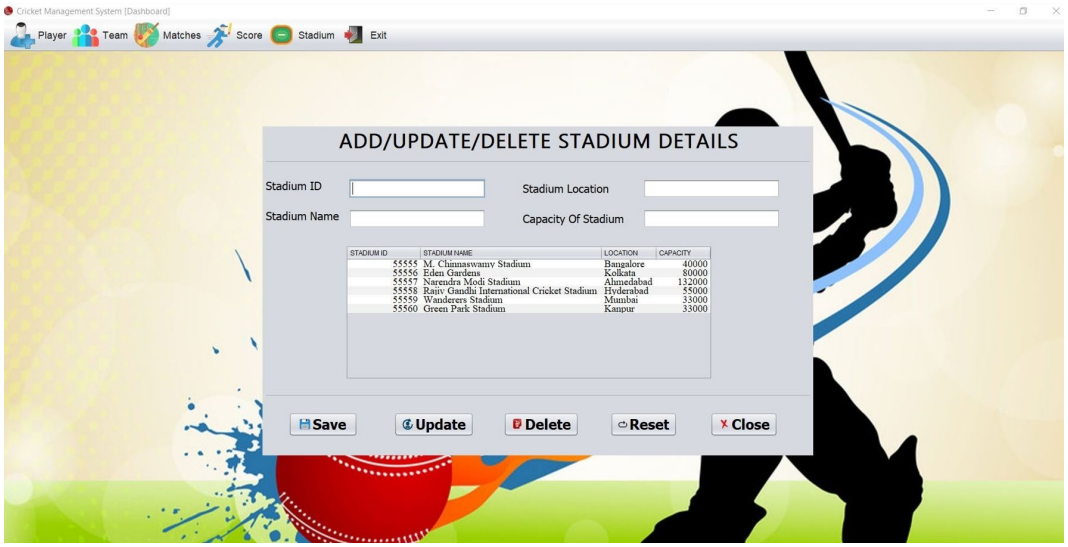


Figure 6.13: Add/ Update/ Delete Stadium details

the above shows stadium window which allows admin to add a new stadium and can also update or delete the stadium details.

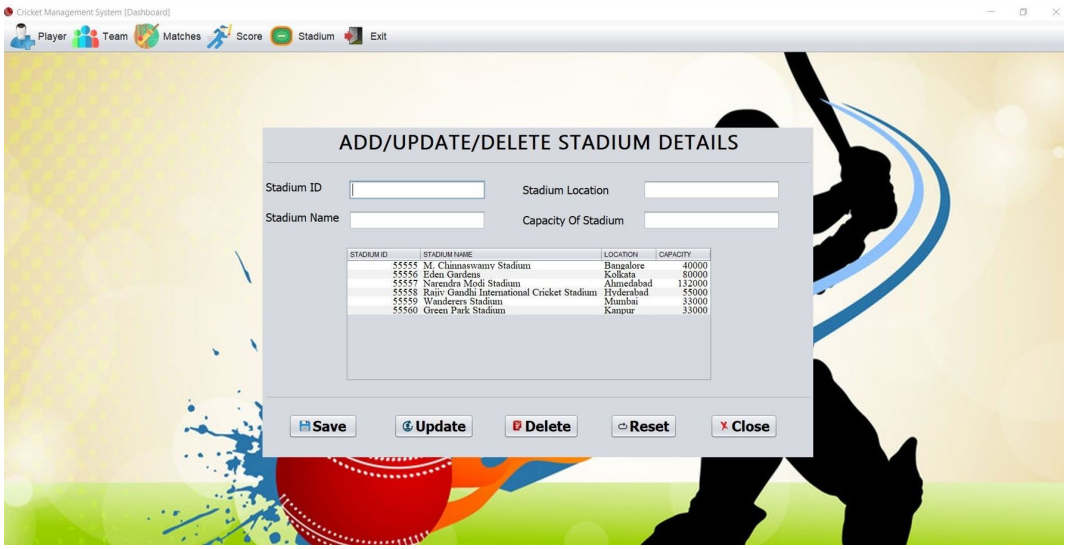


Figure 6.14: Stadium details

It shows stadium details window which allows admin to print the details of stadium from the connected printer or just can save as pdf document


Figure 6.15: Exit menu

The above figure show the exit menu which has got Logout, Exit Application and About Us.

# Chapter 7

# Conclusion

The project, developed using JAVA and MySQL is based on the requirement specification of the user and the analysis of the existing system, with flexibility for future enhancement. The expanded functionality of today's software requires an appropriate approach towards software development. This Cricket database management software is designed for people who want to manage various particulars can be known by recording them in the database. Various records and particulars about match got increased rapidly. Thereby the numbers of matches and there is going to be increased day-by-day. And hence there is a lot of strain on the person who are watching a tournament like IPL or BPL T20)to know about future matches and also to see the records done by various players and getting datils in fingertips. Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user friendly and more GUI oriented.

# Chapter 8

# References

[1] Silberschatz Korth and Sudharshan, “Database System Concepts”, 6th Edition McGraw Hill, 2013.

[2] Data Modeling Using the Entity–Relationship (ER) FUNDAMENTALS OF Database Systems SEVENTH EDITION - (Model 59)