Kuwait University

College Engineering and Petroleum

Computer Engineering Department

CpE-342: DATABASE MANAGEMENT SYSTEM

Semester: SPRING

Project Part 2: Complete Design

StudentName: Hala Saleh **Student Id:** 2211112873

StudentName: Mariam Alazmy **Student Id:** 2201117013

StudentName: Hessah Alsubaiei **Student Id:** 2181145230

Instructor Name: DR.KHALED AL-BARRAK

Teaching Assistant: Eng. Anwar Al-Ebrahim

Date: 7/4/2024

Table of Contents

Chess Players Database Management System

1. Problem Statement

In this phase, our focus shifts exclusively to chess tournaments. We aim to design database system that efficiently handles tournament organization , scheduling, participant registration, match pairings result tracking and players details. Unlike Phase 1, where we considered player management within teams, our Phase 2 objective centers on seamless tournament administration without team memberships. Our system will offer a central platform for storing and organizing important data .The **Chess Players Database Management System** seeks to simplify chess related procedures. It includes player registration, match records, player profiles, award distribution, tournament administration, and statistical data. This approach guarantees effective tournament execution and promotes an engaging chess community which benefits players and organizers

2. Database Design

2.1 User Roles and Permissions

Admins have the authority to create new tournaments, matches, and awards. Additionally, admins can modify tournament details such as adjusting start/end dates or updating rounds and update match information . They can also adjust player ratings based on performance updates. They have the power to delete tournaments, matches, and awards, as well as remove players from the system. Admins have access to all system data.

On the other hand **Players** can view their own profiles, which include personal details and statistics. They can also see information about upcoming matches and tournaments. Players have the ability to update their profiles .They are eligible to participate in tournaments and matches and competing for awards.

2.2. System Features

The chess players management system Handel several essential features. First, it maintains comprehensive player profiles, capturing unique IDs, names, email addresses, and historical performance records such as wins, losses and draws. These profiles enhance the user experience and offer valuable insights into player performance. Second, the system supports the creation and management of awards. Administrators can define award while players earn awards based on their performance. These awards recognize player achievements and contribute to the competitive spirit of the system. Lastly, the system provides customization and flexibility. Administrators can tailor tournament settings such as time controls ensuring adaptability for different tournament formats such as rapid blitz or classical.

2.3. System Requirements

1. Player Management:

1.1 Create New Player Profiles:

The system must allow Players to create detailed player profiles.
Players Details : (Unique ID,Name,Email,Rating).

1.2 Update Player Information:

Administrators must be able to modify player details:
Rating: Adjust based on performance updates
Awards: Add / remove bases on performance updates

1.3 Delete Player Profiles:

Player can delete their profiles
Data Privacy: agreement with privacy regulations when deleting personal data.

1.4 Retrieve Player Details by ID :

The system provide retrieval of player information:
Search by ID: Quickly locate player profiles using their unique ID

1.5 Player Statistics:

The system maintain players statistics:
Games Played: Total number of matches participated in.
Wins: Count of wining matches. Losses: Count of wining matches. Draws: Count of tied games

2. Tournament Management:

2.1 Create New Tournaments:

Administrators must be able to create new tournaments:
Tournament Details: Specify essential information (Unique ID ,NAME,ORGANIZER,TYPE[bullet , blitz , rapid],ROUNDS NUMBER ,LOCATION).

2.2 Specify Tournament Start and End Dates:

The system defined timeframes for each tournament.

2.3 Delete Tournaments:

Administrators must be able to delete tournaments:
Data Privacy: agreement with privacy regulations when deleting personal data

2.4 Retrieve Tournaments Details :

The system provide retrieval of Tournaments information:
Administrators and Players can Read all Tournaments that are available.

3. Match Management:

3.1 Create New Match:

Administrators must be able to create new Matches:
Matches Details: Specify essential information (Unique ID ,DATE,ROUNDS_LEVEL, FIRSTPLAYER_ID,SECONDPLAYER_ID, TOURNAMENTS_ID).

3.2 Update Match Information:

Administrators must be able to modify Matches details:
Matches Details: Update essential information (Unique ID ,DATE,ROUNDS_LEVEL, FIRSTPLAYER_ID,SECONDPLAYER_ID)
Matches Outcomes: Update Match result(RESULT_STATUS ,WIN_PLAYER_ID).

3.3 Record Match Start and End Times:

The system defined timeframes for each tournament Match.
Accuracy: Precise timing ensures accurate scheduling and fair play.

3.4 Associate Players with Each Match (Winner Loser AND DRAW):

the system must link the participating players (2 players only) with each match:
Winner: Identify the wining player.
Loser: : Identify the losing player.
Draw: Identify if the match was drawn.

3.4 Associate Tournaments with Each Match :
For every match, the system must link the Tournaments ID with matches.
3.5 Delete Match:
Administrators must be able to delete Matches.
Data Privacy: agreement with privacy regulations when deleting personal data.
3.6 Retrieve Match Details by ID:
The system provide retrieval of Match information:
Search by ID: Quickly locate Match using their unique ID

4. Awards:

4.1 Create New Awards:
Administrators must be able to create new Awards:
Awards Details: Specify essential information (Unique ID , DESCRIPTION).
4.2 Delete Awards:
Administrators must be able to delete Awards.
Data Privacy: agreement with privacy regulations when deleting personal data.
4.3 Associate Awards with Players:
The system link awards to players:
Performance Based: Recognize outstanding achievements (wins, high ratings).
Custom Awards: Allow administrators to create special awards (“Sportsmanship Award”).
.

2.4. ER Diagram Constraints Assumptions and Relational Schema

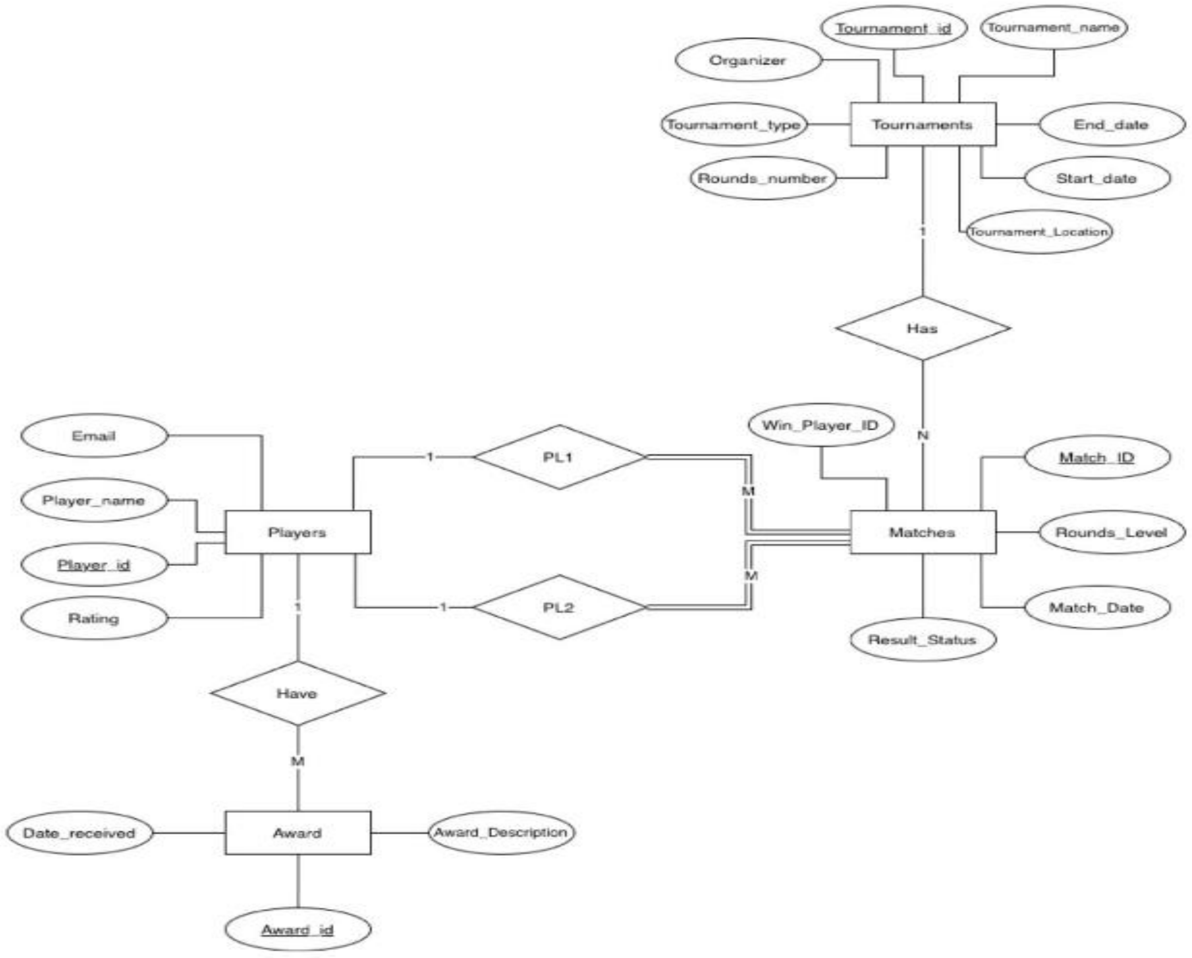


Figure 1: ER Diagram

constraints and assumptions:

-reefer to create table section for domain classifications.

PLAYERS Table:

Constraints:

Primary Key (PLAYERS_ID): Each player's ID is unique and serves as the primary identifier for a player.
NOT NULL Constraint (PLAYERS_NAME): The player's name must be provided; it cannot be left empty.
DEFAULT Constraint (RATING): If no rating is specified, the default value of 0 is assigned.

Assumptions:

Players are uniquely identified by their IDs.
Player names are essential and cannot be missing.
Ratings are optional but default to 0 if not specified.

TOURNAMENTS Table:

Constraints:

Primary Key (TOURNAMENTS_ID): Each tournament has a unique identifier.
NOT NULL Constraints (TOURNAMENTS_NAME, START_DATE, END_DATE, ORGANIZER, TOURNAMENTS_LOCATION): These attributes must have valid values; they cannot be left empty.
DEFAULT Constraint (ROUNDS_NUMBER): If the number of rounds is not specified, it defaults to 1.

Assumptions:

Tournaments are uniquely identified by their IDs.
Tournament names, start dates, end dates, organizers, and locations are essential.
The type of rounds may vary, but the default is assumed to be a single round.

MATCHES Table:

Constraints:

Primary Key (MATCHS_ID): Each match has a unique identifier.
NOT NULL Constraints (MATCHS_DATE, FIRSTPLAYER_ID, SECONDPLAYER_ID, TOURNAMENTS_ID): These attributes must be provided.
FOREIGN KEY Constraints (FIRSTPLAYER_ID, SECONDPLAYER_ID, TOURNAMENTS_ID): These establish relationships with the PLAYERS and TOURNAMENTS tables.

Assumptions:

Matches are uniquely identified by their IDs.
Matches can be played only by two players.
Match dates, player IDs, and tournament IDs are essential.
The result status and winning player ID are optional.
Matches can occur at different levels (assumed based on the ROUNDS_LEVEL attribute).

AWARDS Table:

Constraints:

Primary Key (AWARD_ID): Each award has a unique identifier.
FOREIGN KEY Constraint (WIN_PLAYER_ID): Refers to the PLAYERS table.
AWARD_DESCRIPTION: A description of the award is required and cannot be left empty (Not Null).

Assumptions:

Awards are uniquely identified by their IDs.
Awards are given to players

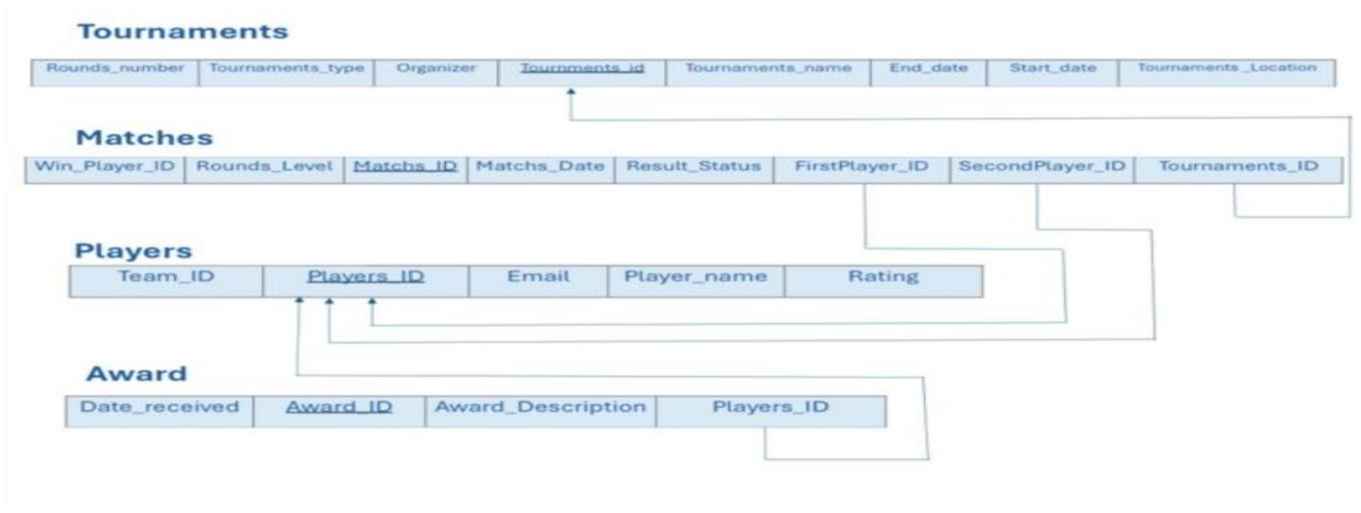
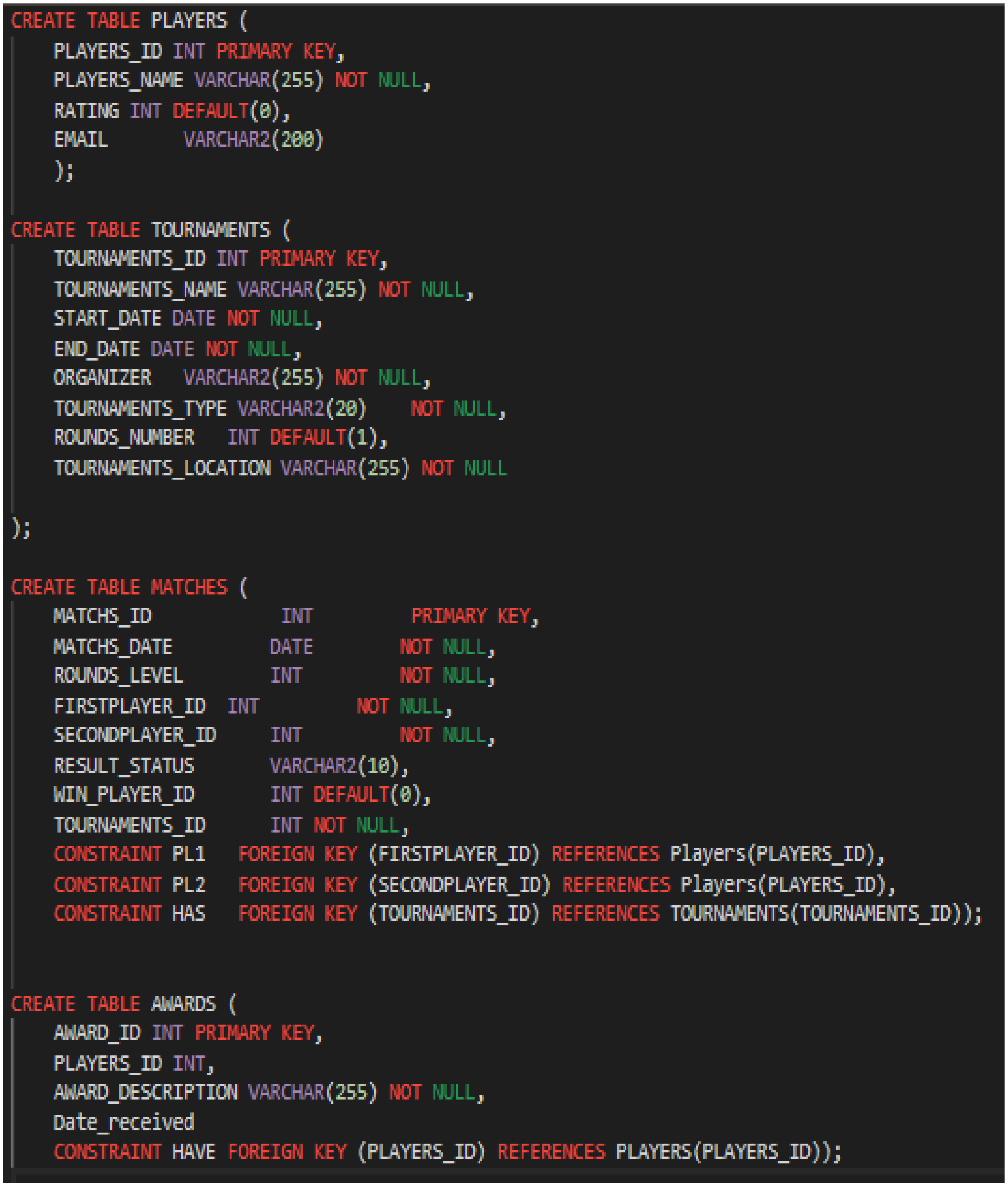


Figure 2 : schema

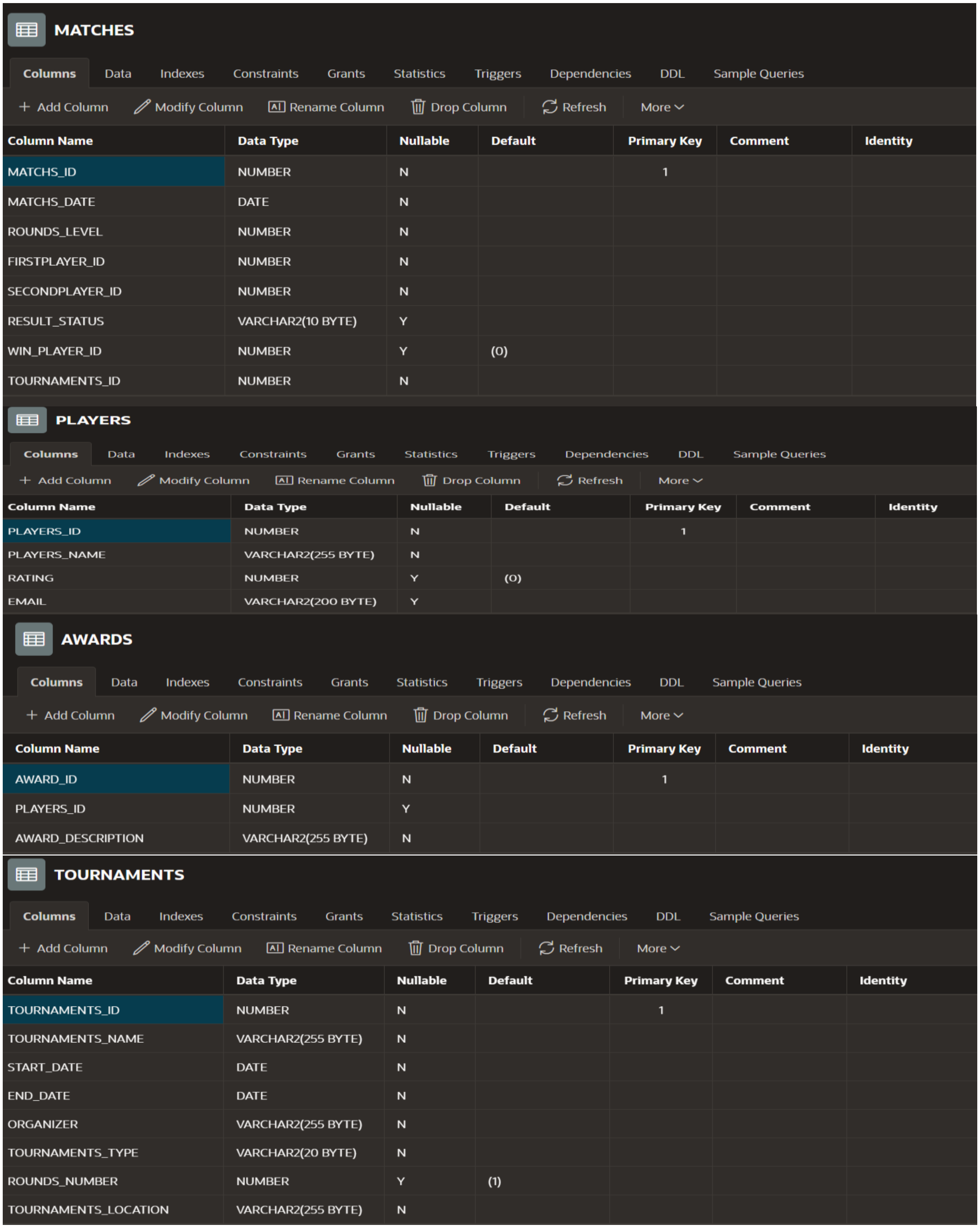
2.5. Create Tables

-Using SQL apex



```
CREATE TABLE PLAYERS (
    PLAYERS_ID INT PRIMARY KEY,
    PLAYERS_NAME VARCHAR(255) NOT NULL,
    RATING INT DEFAULT(0),
    EMAIL       VARCHAR2(200)
    );

CREATE TABLE TOURNAMENTS (
    TOURNAMENTS_ID INT PRIMARY KEY,
    TOURNAMENTS_NAME VARCHAR(255) NOT NULL,
    START_DATE DATE NOT NULL,
    END_DATE DATE NOT NULL,
    ORGANIZER   VARCHAR2(255) NOT NULL,
    TOURNAMENTS_TYPE VARCHAR2(20)    NOT NULL,
    ROUNDS_NUMBER   INT DEFAULT(1),
    TOURNAMENTS_LOCATION VARCHAR(255) NOT NULL

);

CREATE TABLE MATCHES (
    MATCHS_ID             INT         PRIMARY KEY,
    MATCHS_DATE          DATE        NOT NULL,
    ROUNDS_LEVEL         INT         NOT NULL,
    FIRSTPLAYER_ID  INT         NOT NULL,
    SECONDPLAYER_ID     INT         NOT NULL,
    RESULT_STATUS       VARCHAR2(10),
    WIN_PLAYER_ID       INT DEFAULT(0),
    TOURNAMENTS_ID      INT NOT NULL,
    CONSTRAINT PL1   FOREIGN KEY (FIRSTPLAYER_ID) REFERENCES Players(PLAYERS_ID),
    CONSTRAINT PL2   FOREIGN KEY (SECONDPLAYER_ID) REFERENCES Players(PLAYERS_ID),
    CONSTRAINT HAS   FOREIGN KEY (TOURNAMENTS_ID) REFERENCES TOURNAMENTS(TOURNAMENTS_ID));


CREATE TABLE AWARDS (
    AWARD_ID INT PRIMARY KEY,
    PLAYERS_ID INT,
    AWARD_DESCRIPTION VARCHAR(255) NOT NULL,
    Date_received
    CONSTRAINT HAVE FOREIGN KEY (PLAYERS_ID) REFERENCES PLAYERS(PLAYERS_ID));
```

Figure 3 : Tables code



MATCHES

Column Name	Data Type	Nullable	Default	Primary Key	Comment	Identity
MATCHS_ID	NUMBER	N		1		
MATCHS_DATE	DATE	N				
ROUNDS_LEVEL	NUMBER	N				
FIRSTPLAYER_ID	NUMBER	N				
SECONDPLAYER_ID	NUMBER	N				
RESULT_STATUS	VARCHAR2(10 BYTE)	Y				
WIN_PLAYER_ID	NUMBER	Y	(0)			
TOURNAMENTS_ID	NUMBER	N				

PLAYERS

Column Name	Data Type	Nullable	Default	Primary Key	Comment	Identity
PLAYERS_ID	NUMBER	N		1		
PLAYERS_NAME	VARCHAR2(255 BYTE)	N				
RATING	NUMBER	Y	(0)			
EMAIL	VARCHAR2(200 BYTE)	Y				

AWARDS

Column Name	Data Type	Nullable	Default	Primary Key	Comment	Identity
AWARD_ID	NUMBER	N		1		
PLAYERS_ID	NUMBER	Y				
AWARD_DESCRIPTION	VARCHAR2(255 BYTE)	N				

TOURNAMENTS

Column Name	Data Type	Nullable	Default	Primary Key	Comment	Identity
TOURNAMENTS_ID	NUMBER	N		1		
TOURNAMENTS_NAME	VARCHAR2(255 BYTE)	N				
START_DATE	DATE	N				
END_DATE	DATE	N				
ORGANIZER	VARCHAR2(255 BYTE)	N				
TOURNAMENTS_TYPE	VARCHAR2(20 BYTE)	N				
ROUNDS_NUMBER	NUMBER	Y	(1)			
TOURNAMENTS_LOCATION	VARCHAR2(255 BYTE)	N				

Figure 4 : Tables2

3. Software Design

1. Players Module:

Functional Components: - Players Management (CRUD operations). - player Assignments.

Specifications: - Manage players information (ID, name, email).

SQL Queries:

- Retrieve all players: SELECT * FROM Players;
- Add a new players: INSERT INTO Players (ID, Name, Email)VALUES(8653902173, 'Alice','exp@gmail');
- Add a new player: INSERT INTO Players (Player_ID, Player_Name, Email) VALUES (8653902173, 'Alice','exp@gmail');
- DELETE FROM Players WHERE Player_ID = 6739286152;
- UPDATE Players SET Player_Name = 'Thomas'WHERE Player_ID = 6739286152;

Read All Players

High-Level Algorithm:

1-Execute a SELECT query to retrieve all records from the "Players" table.
2-Return the list of Players records.

Input Parameters:

- None (since it retrieves all players).

Output Parameters:

-A list of Players records (including player IDs, names, rating, team_ID and emails).

SQL Example:SELECT * FROM players;

Read Specific Players

High-Level Algorithm:

1- Execute a SELECT query to retrieve the players details based on the provided Player ID.
2- If a record with the specified Player ID exists, return the player ID and player name.
3-If no matching record is found, handle the case where the players is not found

Input Parameters:

- Player ID: The unique identifier of the players you want to retrieve.

Output Parameters:

- Success Message: "Jasmine 8846364431."
- Error Message: "Error: Player Not Found."

SQL Example: SELECT Player_ID, Player_Name FROM Players WHERE Player_ID = 6739286152

Creating a New Players

High-Level Algorithm:

1-Collect players details (ID, Name, Email).
2-Insert the player record into the “Players” table using an INSERT statement.
3-If successful, return the success message; otherwise, handle the error.

Input Parameters:

- Player ID: ID of the players being created.
- Player Name: The name of the players being created.
- Email: The Email of the players being created.

Output Parameters:

- Success Message: "ID '6789328217' successfully created."
- Error Message: "Error: Player’s ID already exists."

SQL Example:

```
INSERT INTO players (Player_ID, Player_Name, Email) VALUES (8653902173,
'Alice','exp@gmail');
```

Delete a Players

High-Level Algorithm:

1-Collect the player ID.
2-Remove the player record from the “Players” table using a DELETE statement.
3-If successful, return the success message; otherwise, handle the error.

Input Parameters:

-Player ID: The unique identifier of the players to be deleted.

Output Parameters:

-Success Message: “ID '6789328217' successfully deleted."
-Error Message: “Error: Player’s ID doesn't exist.”

SQL Example:

```
DELETE FROM Players WHERE Player_ID = 6739286152;
```

Updating a Players

High-Level Algorithm:

1-Collect the player ID and the new player name.
2-Update the player record in the "Players" table using an UPDATE statement.
3-If successful, return the success message; otherwise, handle the error.

Input Parameters:

-Player ID: The unique identifier of the players to be updated.

-New Player_Name: The updated name for the players.

Output Parameters:

- Success Message: "ID '6789328217' successfully uploaded."

-Error Message: "Error: Player's ID Not Found."

SQL Example:

```
UPDATE Players SET Player_Name = 'Thomas'WHERE Player_ID = 6739286152;
```

2. Awards Module:

Functional Components: - Awards Management (CRD operations). - Awards Assignments.

Specifications: - Manage awards information (award_ID, award_description, Date_received).

SQL Queries:

- Add a new awards: INSERT INTO Awards (award_ID, award_description, Date_received) VALUES (44287, 'category celebrates the young chess players who quickly rose through the chess ranks during the year','12\06\2023');
- Retrieve all awards: SELECT * FROM Awards;
- DELETE FROM Awards WHERE Awards_ID = 28018;

Creating a New Awards

High-Level Algorithm:

1. Collect awards details (award_ID, award_description, Date_received).
2. Insert the award record into the "Awards" table using an INSERT statement.
3. If successful, return the success message; otherwise, handle the error.

Input Parameters:

- Award ID: ID of the award being created.
- Awards Description: The description being created.
- Date received: The date of receiving the award.

Output Parameters:

- Success Message: "ID '56641' successfully created."
- Error Message: "Error: Awards's ID already exists."

SQL Example: INSERT INTO Awards (award_ID, award_description, Date_received) VALUES (44287, 'category celebrates the young chess players who quickly rose through the chess ranks during the year','12\06\2023');

Read All Awards

High-Level Algorithm:

1- 1-Execute a SELECT query to retrieve all records from the "Awards" table.
2- 2-Return the list of Awards records.

Input Parameters:

- None (since it retrieves all awards).

Output Parameters:

A list of Awards records (award_ID, award_description, Date_received).

SQL Example: SELECT * FROM awards;

Delete an awards

High-Level Algorithm:

1-Collect the award ID.
2-Remove the award record from the “Awards” table using a DELETE statement.
3-If successful, return the success message; otherwise, handle the error.

Input Parameters:

-Award ID: The unique identifier of the awards to be deleted.

Output Parameters:

-Success Message: “ID '56641' successfully deleted."
-Error Message: “Error: Award’s ID doesn't exist.”

SQL Example: DELETE FROM Awards WHERE Award_ID = 44287;

3. Tournament Module:

Functional Components: - Tournament Management (CRUD operations). – Tournaments Assignments.

Specifications: - Manage Tournaments information (tournament_ID, tournament_NAME, start_Date, end_Date, organizer, tournament_type, Rounds_number, tournament_location).

SQL Queries:

- Add a new tournaments: INSERT INTO Tournaments (tournament_ID, tournament_NAME, start_Date, end_Date, organizer, tournament_type, Rounds_number, tournament_location) VALUES (783298, ‘jjkxkcodo’, ‘, 09/01/2024’, ‘11/01/2024’, ‘Layla’, ‘swiss system’, 8, ‘Kuwait Mishref Fairgrounds’);
- Retrieve all tournaments: SELECT * FROM Tournaments;
- DELETE FROM Tournaments WHERE Tournaments _ID = 219832;

**Creating a New Tournaments **

High-Level Algorithm:

1. Collect Tournaments details (tournament_ID, tournament_NAME, start_Date, end_Date, organizer, tournament_type, Rounds_number, tournament_location).

2. Insert the tournament record into the “Tournaments” table using an INSERT statement.
3. If successful, return the success message; otherwise, handle the error.

Input Parameters:

- Tournaments ID: ID of the tournament being created.
- Tournaments name: The name of the tournament being created
- Start Date: start of the tournament date.

- End Date: end of the tournament date.
- Organizer: the name of the tournament's organizer.
- Tournaments type: how the tournament will be played.
- Rounds number: number of rounds during the tournament.
- Tournament location:

Output Parameters:

- Success Message: "ID '976531' successfully created."
- Error Message: "Error: Tournament's ID already exists."

SQL Example: INSERT INTO Tournaments (tournament_ID, tournament_NAME, start_Date, end_Date, organizer, tournament_type, Rounds_number, tournament_location) VALUES (783298, 'jjkxkcodo', ', 09/01/2024', '11/01/2024', 'Layla', 'swiss system', 8, 'Kuwait Mishref Fairgrounds');

**Read All Tournaments **

High-Level Algorithm:

1-Execute a SELECT query to retrieve all records from the "Tournaments" table.
2-Return the list of Tournaments records.

Input Parameters:

- None (since it retrieves all Tournaments).

Output Parameters:

- A list of Awards records (tournament_ID, tournament_NAME, start_Date, end_Date, organizer, tournament_type, Rounds_number, tournament_location).

SQL Example: SELECT * FROM Tournaments;

**Delete a Tournaments **

High-Level Algorithm:

1- Collect the Tournaments ID.
2- Remove the Tournament record from the "Tournaments" table using a DELETE statement.
3- If successful, return the success message; otherwise, handle the error.

Input Parameters:

- Tournaments ID: The unique identifier of the Tournaments to be deleted.

Output Parameters:

-Success Message: "ID '976531' successfully deleted."
-Error Message: "Error: Tournament's ID doesn't exist."

SQL Example:

DELETE FROM Tournaments WHERE Tournaments_ID = 976531;

4. Matches Module:

Functional Components: - Matches Management (CRUD operations). – Match Assignments.

Specifications: - Manage matches information (Match_ID, Match_Date, Rounds_Level, Result_Status, Win_Player_ID).

SQL Queries:

- Retrieve all matches: SELECT * FROM Matches;
- Add a new match: INSERT INTO Matches(Match_ID, Match_Date, Rounds_Level, Result_Status, Win_Player_ID)VALUES(876530, '30/03/2024', 22, 'win', 8973625101);
- DELETE FROM Matches WHERE Match_ID = 876530;
- UPDATE Matches SET Player_Name = 'Thomas'WHERE Match_ID = 6739286152;

*Read All Matches *

High-Level Algorithm:

1- 1-Execute a SELECT query to retrieve all records from the "Matches" table.
2- 2-Return the list of Matches records.

Input Parameters:

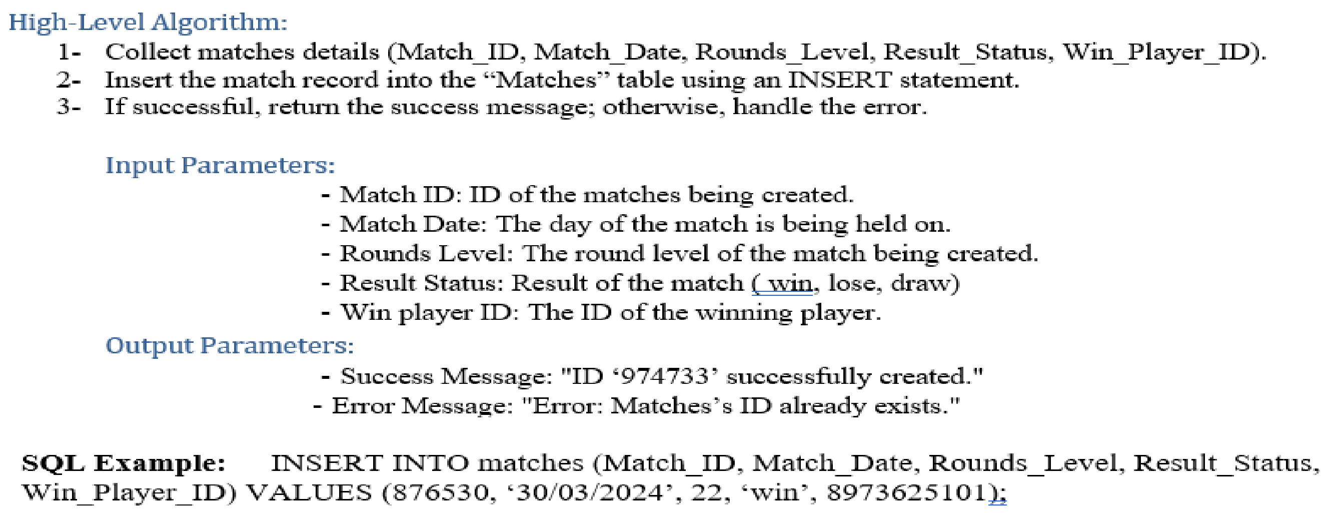
- None (since it retrieves all matches).

Output Parameters:

- A list of Matches records (Match_ID, Match_Date, Rounds_Level, Result_Status, Win_Player_ID).

SQL Example: SELECT * FROM matches;

Creating a New Matches



High-Level Algorithm:

1- Collect matches details (Match_ID, Match_Date, Rounds_Level, Result_Status, Win_Player_ID).
2- Insert the match record into the "Matches" table using an INSERT statement.
3- If successful, return the success message; otherwise, handle the error.

Input Parameters:

- Match ID: ID of the matches being created.
- Match Date: The day of the match is being held on.
- Rounds Level: The round level of the match being created.
- Result Status: Result of the match (win, lose, draw)
- Win player ID: The ID of the winning player.

Output Parameters:

- Success Message: "ID '974733' successfully created."
- Error Message: "Error: Matches's ID already exists."

SQL Example: INSERT INTO matches (Match_ID, Match_Date, Rounds_Level, Result_Status, Win_Player_ID) VALUES (876530, '30/03/2024', 22, 'win', 8973625101);

Delete a Matches

High-Level Algorithm:

1-Collect the Match ID.
2-Remove the match record from the “Matches” table using a DELETE statement.
3-If successful, return the success message; otherwise, handle the error.

Input Parameters:

-Match ID: The unique identifier of the matches to be deleted.

Output Parameters:

-Success Message: “ID ‘876530’ successfully deleted."
-Error Message: “Error: Matches’s ID doesn't exist.”

SQL Example:

```
DELETE FROM Matches WHERE Match_ID = 6739286152;
```

Updating a Matches

High-Level Algorithm:

1- Collect the match ID and the new wining player ID.
2- Update the match record in the “Matches” table using an UPDATE statement.
3- If successful, return the success message; otherwise, handle the error.

SQL Example: UPDATE Matches SET Win_Player_ID= 'Nate' WHERE Match_ID = 8877329;

Input Parameters:

-Match ID: The unique identifier of the matches to be updated.
-New win_player_ID: The updated ID number for the wining player.

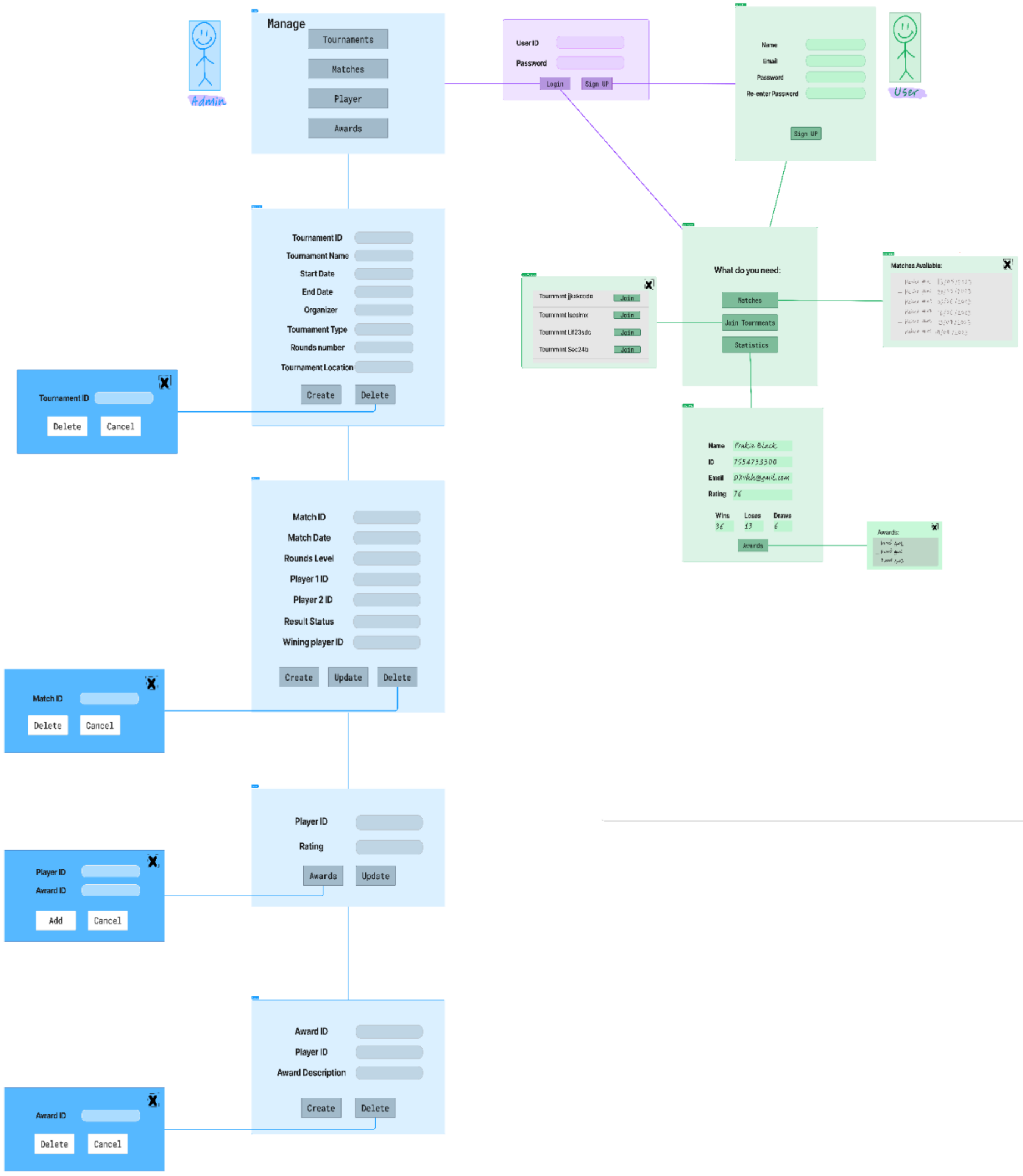
Output Parameters:

- Success Message: “ID ‘8877329’ successfully uploaded."
-Error Message: “Error: Matches’s ID Not Found.”

SQL Example:

```
UPDATE Matches SET Win_Player_ID = 'Nate' WHERE Match_ID = 8877329;
```

4. User Interface (UI)



Name	Frakie Black
ID	7554733300
Email	DXvkh@gmail.com
Rating	76

Wins	Loses	Draws
36	13	6

Figure 5 :User Interface diagram

1. **Login / Sign Up Page**:

There are two types of users: Player and administrators(admin).

Administrators can directly log in without signing up.

If a player is already registered, they can log in by providing their ID and password on the login page.

On the other hand If a player doesn't have an ID and password, they need to register by signing up.

2. **Player Registration (Sign Up):**

Fields for registration :

Name: The player provides their name.

Email: The player enters their email address.

Password: The player sets a password.

Re-enter Password: The player confirms the password by entering it again.

After completing these Fields, their account will be created.

If any errors occur during registration (invalid email format, password mismatch etc) appropriate error messages will be displayed.

For successful registration a confirmation message will appear and they will be redirected to player panel.

3. **Admin Panel**:

After successful login, administrators access the admin panel.

The panel is titled Manage and contains four menus:

Tournaments: This menu allows administrators to manage tournaments.

Fields for creating a new tournament as shows in figure 5 .

An option to delete a tournament: A pop-up message appears when the delete button is pressed.

The user must input the tournament ID to be deleted.

If the ID doesn't exist, an error message displays.

If the ID exists, the tournament is deleted, and a success message appears.

Matches: This menu allows administrators to manage matches.

Fields for creating or updating a match is shown in figure 5 .

To delete a match: The admin must provide the match ID

A success or error message appears accordingly.

Awards: : This menu allows administrators to manage Awards.

Fields for creating or updating a Award is shown in figure 5.

To delete a Award: The admin must provide the Award ID

If the award ID exists, the award will be removed, and a success message will be displayed. Otherwise, an error message will appear .

Players:

When updating a player's rating, the system will prompt the admin to provide the following details:

Player ID
New rating

After inputting these details, the player's rating will be updated, and a success message will appear.
To add an award to a player, the user will need to provide:
Player ID
Award ID
The system will associate the award with the specified player, and a success message will be displayed
Otherwise, an error message will appear .

4. **Player Panel**:

After successful login, Players can access the player panel.

The panel is titled **What do you need** and contains Three menus:

Matches :
In this panel, players can view their matches, including both past matches and upcoming ones.
The information displayed for each match could include:
Match ID
Date
Opponent (first player , second player)
Result (if available)
Players can easily keep track of their performance.

Join Tournament:
In this panel, players can explore available tournaments to join.
The system should display a list of tournaments that players can participate in.
Each tournament entry could include:
Tournament ID
Name
Start date
End date
Organizer
Tournament type
Number of rounds
Players can select a tournament to join.

Statistics:
In this panel, players can view their detailed statistics:
Player Details:
Name
ID
Email
Rating
Total wins   Total losses   Total draws
Awards Button:
Clicking this button will redirect players to a page displaying all the awards they have received and their description

5. Implementation Plan

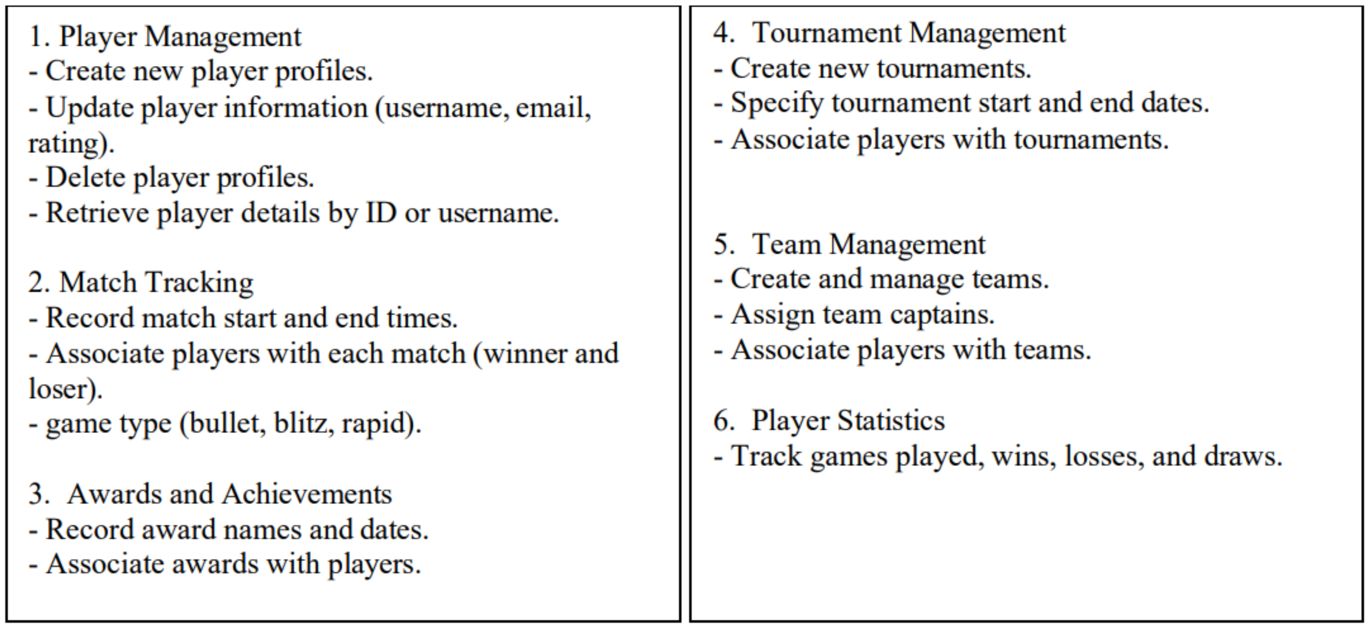
5.1. System Overview

Objective: Develop application for managing chess players in joining tournaments and assigning player to matches in tournament and assign awards to players .
Stakeholders: Players, administrators

5.2. Hardware and Software Selection

Hardware:
DESKTOP-G2LBPMO , Window11 64-bit operating system, x64-based processor
Software:
DBMS: Oracle APEX
Programming languages: PL/SQL

5.3. Basic Program Modules



1. Player Management
- Create new player profiles.
- Update player information (username, email, rating).
- Delete player profiles.
- Retrieve player details by ID or username.

2. Match Tracking
- Record match start and end times.
- Associate players with each match (winner and loser).
- game type (bullet, blitz, rapid).

3. Awards and Achievements
- Record award names and dates.
- Associate awards with players.

4. Tournament Management
- Create new tournaments.
- Specify tournament start and end dates.
- Associate players with tournaments.

5. Team Management
- Create and manage teams.
- Assign team captains.
- Associate players with teams.

6. Player Statistics
- Track games played, wins, losses, and draws.

5.4. User Interface (UI)

1. **Login / Sign Up**:

Existing players can log in with their ID and password.
New players can sign up by providing their name, email, and password [Registration] .
Admin can log in with their ID and password.

2. **Main Menus : for admin (“Manage”)**

Tournaments: allows administrators to create/delete tournaments
Matches: allows administrators to create/delete Matches
Awards: allows administrators to create/delete Awards
Players: allows administrators to updating a player’s rating and associate awards to players

for players (“What Do You Need?”):

Matches: View past and upcoming matches.
Join Tournament: Explore available tournaments to join
Statistics: Access player details and awards.

For more detail please refer to User Interface section.

5.5 Responsibilities and Timeline (April 7 - May 5)

Week 1 (April 7 - April 13)

- Team Meeting:
  - Set up Oracle Apex.
  - Review Feedback
  - Change the ERD based on feedback.
  - Review Timeline Make changes if needed.

Week 2 (April 14 - April 20)

- Hala Saleh , Mariam Alazmy :
  - Implement Login for both admin and player
  - Implement player registration system

- Hessah Alsubaiei :
  - Implement the Admin menu .
  - Implement Player menu

- Team Meeting:
  - Review work
  - Get team feedback
  - Make adjustments as needed

Week 3 (April 21 - April 27) Match

- Mariam Alazmy :
  - Implement Tournaments panel For the Admin menu
  - Implement Match panel For the Admin menu
  - Test Implementation
  - Get team feedback

- Hessah Alsubaiei :
  - Implement Awards panel For the Admin menu
  - Implement Players panel For the Admin menu
  - Test Implementation
  - Get team feedback

- Team Meeting:
  - Review work
  - Get team feedback
  - Get Instructors feedback
  - Make adjustments as needed

Week 4 (April 28 - May 5)

- Hala Saleh :
  - Implement Match panel For the Players menu
  - Implement Tournaments panel For the Players menu
  - Implement Player Statis panel For the Players menu
- Team Meeting:
  - Review work
  - Get team feedback
  - Make adjustments as needed
  - Get Instructors feedback
- Team Meeting:
  - Review the entire system.
  - prepare for presentation week

5.6.Testing And Deployment:

-Test SQL procedures locally Using Oracle Apex.
-Using a sample dataset for testing.
-Verify UI components within Oracle Apex.
-Deploying the system on Oracle Apex..