Database Code

```
CREATE TABLE Player(

-- Common to all users

username char(50) NOT NULL,
nationality char(50) NOT NULL,
name_ char(50) NOT NULL,
surname char(50) NOT NULL,
password_ char(50) NOT NULL,
PRIMARY KEY(username),

date_of_birth date NOT NULL,
elo_rating int NOT NULL,
fide_ID int NOT NULL,
CHECK(elo_rating > 1000)
);

CREATE TABLE Title( -- finished

title_ID int NOT NULL,
title_name char(50) NOT NULL,
player_name char(50) NOT NULL,

PRIMARY KEY(title_ID),
FOREIGN KEY(player_name) REFERENCES Player(username)
);

CREATE TABLE Coach( -- finished
```

```
-- Common to all users

username char(50) NOT NULL,
nationality char(50) NOT NULL,
name_ char(50) NOT NULL,
surname char(50) NOT NULL,
password_ char(50) NOT NULL,
PRIMARY KEY(username),

certificate_name char(50) NOT NULL,
speciality_name char(50) NOT NULL
);

CREATE TABLE Certification1( -- finished

certificate_ID int NOT NULL,
certificate_name char(50) NOT NULL,
owner_name char(50) NOT NULL,

PRIMARY KEY(certificate_ID),
Foreign Key (owner_name) REFERENCES Coach(username)
);

CREATE TABLE Speciality( -- finished

speciality_ID int NOT NULL,
speciality_name char(50) NOT NULL,
owner_name char(50) NOT NULL,

PRIMARY KEY(speciality_ID),
```

```
Foreign Key (owner_name) REFERENCES Coach(username)
);

CREATE TABLE Arbiter( -- finished

-- Common to all users

username char(50) NOT NULL,
nationality char(50) NOT NULL,
name_ char(50) NOT NULL,
surname char(50) NOT NULL,
password_ char(50) NOT NULL,
PRIMARY KEY(username),

certificate_name char(50) NOT NULL,
experience_elevel int NOT NULL
);

CREATE TABLE Certification2( -- finished

certificate_ID int NOT NULL,
certificate_name char(50) NOT NULL,
owner_name char(50) NOT NULL,

PRIMARY KEY(certificate_ID),
Foreign Key (owner_name) REFERENCES Arbiter(username)
);

CREATE TABLE Sponsor( -- finished
```

```
sponsor_ID int NOT NULL,
sponsor_name char(50) NOT NULL,
PRIMARY KEY(sponsor_ID)
);

CREATE TABLE Tournament( -- finished

tournament_ID int NOT NULL,
tournament_name char(50) NOT NULL,
start_date_ date NOT NULL,
end_date_ date NOT NULL,
formal char(10) NOT NULL,
chief_arbiter char(50) NOT NULL, -- The chief arbiter
hall_ID int NOT NULL, -- Every tournament must have at least one hal

PRIMARY KEY(tournament_ID),
FOREIGN KEY(chief_arbiter) REFERENCES Arbiter(username)
);

CREATE TABLE Team( -- finished

team_ID int NOT NULL,
team_name char(50) NOT NULL,
contract_start date NOT NULL,
contract_finish date NOT NULL,
sponsor_ID int NOT NULL, -- A team may not have a sponsor
coach_username char(50) NOT NULL,
tournament_ID int NOT NULL,

PRIMARY KEY(team_ID),
```

```
FOREIGN KEY(coach_username) REFERENCES Coach(username),
FOREIGN KEY(sponsor_ID) REFERENCES Sponsor(sponsor_ID),
FOREIGN KEY(tournament_ID) REFERENCES Tournament(tournament_ID)
);

CREATE TABLE Player_Team( -- finsihed

team_player_ID int NOT NULL,
team_ID int NOT NULL,
player_name char(50) NOT NULL,

PRIMARY KEY(team_player_ID),
FOREIGN KEY(team_ID) REFERENCES Team(team_ID),
FOREIGN KEY(player_name) REFERENCES Player(username)
);

CREATE TABLE Hall( -- finished

hall_ID int NOT NULL,
hall_country char(50) NOT NULL,
hall_name char(50) NOT NULL,
tournament_ID int, -- hall may not host a tournament

PRIMARY KEY(hall_ID),
FOREIGN KEY(tournament_ID) REFERENCES Tournament(tournament_ID)
);

CREATE TABLE Table_( -- finished

table_ID int NOT NULL,
```

```
hall_ID int NOT NULL,

PRIMARY KEY(table_ID),
FOREIGN KEY(hall_ID) REFERENCES Hall(hall_ID)
);

CREATE TABLE Match_( -- finished

match_ID int NOT NULL,
tournament_ID int NOT NULL,
hall_ID int NOT NULL,
table_ID int NOT NULL,
white_player_team int NOT NULL,
white_player char(50) NOT NULL,
black_player_team int NOT NULL,
black_player char(50) NOT NULL,
result char(10) NOT NULL,
time_slot int NOT NULL,
time1 int NOT NULL, -- how to calculate time1/2 from time_slot. We w
time2 int NOT NULL,
date_ date NOT NULL,
assigned_arbiter_username char(50) NOT NULL,
rating int NOT NULL,

PRIMARY KEY(match_ID),
UNIQUE(hall_ID,table_ID,time1,date_), -- to ensure no collision occu
UNIQUE(hall_ID,table_ID,time2,date_),
CHECK(rating >= 0 AND rating <= 10),
FOREIGN KEY(white_player_team) REFERENCES Team(team_ID),
FOREIGN KEY(black_player_team) REFERENCES Team(team_ID),
```

```
FOREIGN KEY(white_player) REFERENCES Player(username),
FOREIGN KEY(black_player) REFERENCES Player(username),
FOREIGN KEY(assigned_arbiter_username) REFERENCES Arbiter(username),
FOREIGN KEY(tournament_ID) REFERENCES Tournament(tournament_ID),
FOREIGN KEY(hall_ID) REFERENCES Hall(hall_ID),
FOREIGN KEY(table_ID) REFERENCES Table_(table_ID)
);
```

Discussion Part

We have successfully implemented all the necessary tables and functionalities, including the ELO rating system.