

---

# Table Tennis Documentation

*Release 1.0*

itucsd1519

Sep 27, 2017



---

## Contents

---

<b>1</b>	<b>User Guide</b>	<b>3</b>
<b>2</b>	<b>Developer Guide</b>	<b>29</b>



**Team** itucsd1519

**Members**

- Hasan Burak Namlı
- Alican Mertan
- Fırat Bayram
- Ahmet Yılmaz

**Table Tennis Database**

Main purpose of table tennis page is keeping the record of the players, coaches, teams, tournaments, matches, stadiums, referees and technic members. Site includes match statistics, player statistics. Through this page users can reach the data about table tennis sport.

Contents:



This part explains how table tennis page works from user perspective

- Home Page is looking like this :

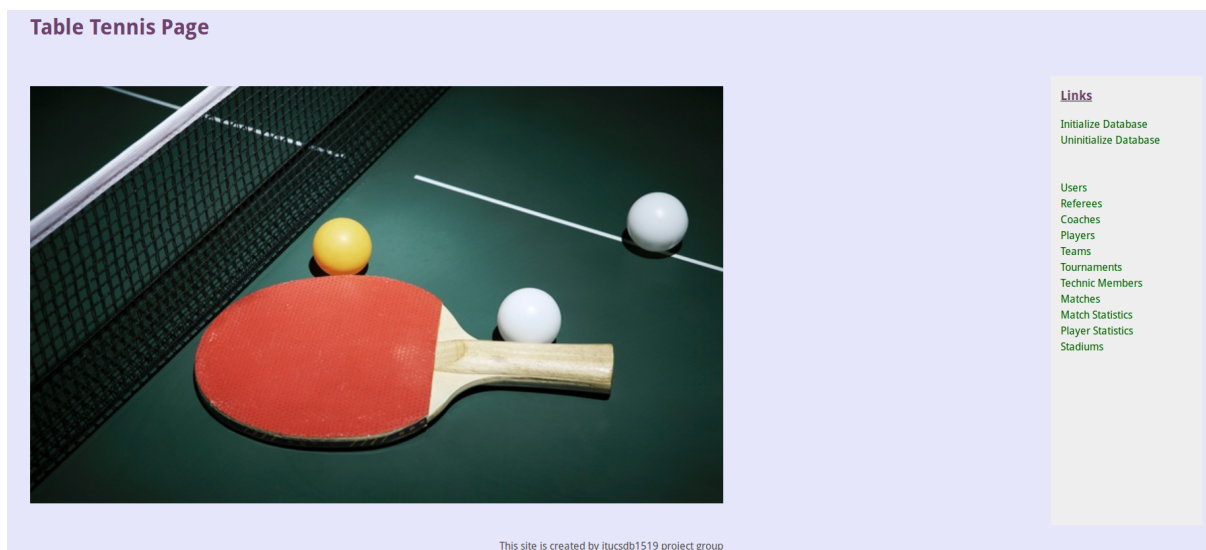


Fig. 1.1: Home Page

Initializing and uninitializing database can be implemented via 'Initialize Database' and 'Uninitialize Database' links. All of the pages can be reached via listed links

## Parts Implemented by Hasan Burak NAMLI


### Pages

1. *Teams Page*
2. *Players Page*

### 3. *Technic Members Page*

#### Teams Page

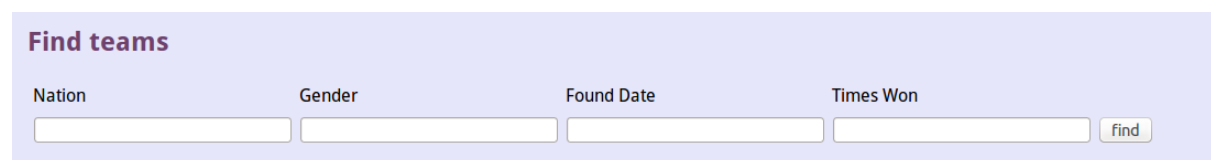
- In default teams page all teams are listed in the Teams Table Section



#	Nation	Gender	Found Date	Times Won
<input type="checkbox"/>	Turkey	Male	1920	4
<input type="checkbox"/>	England	Male	1936	3
<input type="checkbox"/>	China	Male	1906	5
<input type="checkbox"/>	Russia	Female	1943	1

Fig. 1.2: Teams Table

- Above part of the page find operation can be applied by all attributes. Found teams are listed in the teams table section. All teams showed when find operation made with all text boxes empty.

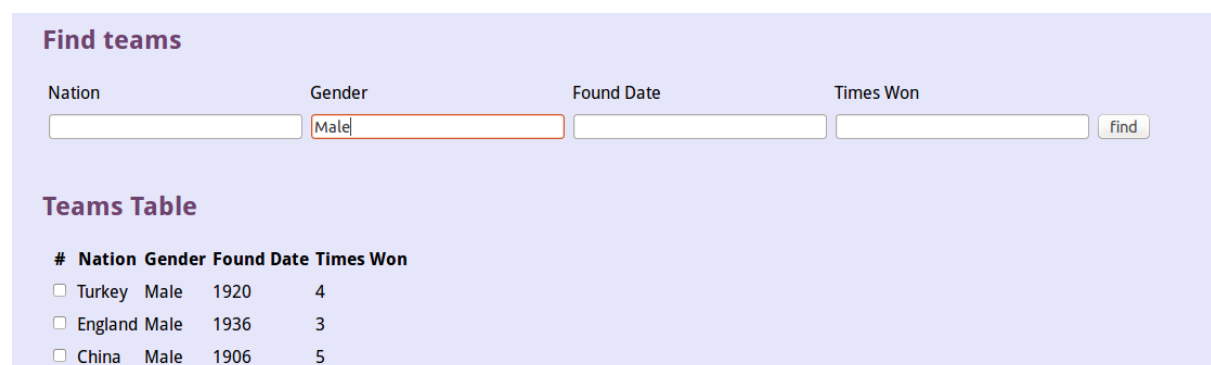


**Find teams**

Nation      Gender      Found Date      Times Won

Fig. 1.3: Find Teams Section



**Find teams**

Nation      Gender      Found Date      Times Won

**Teams Table**

#	Nation	Gender	Found Date	Times Won
<input type="checkbox"/>	Turkey	Male	1920	4
<input type="checkbox"/>	England	Male	1936	3
<input type="checkbox"/>	China	Male	1906	5

Fig. 1.4: Sample find operation

- Delete operation can be done by selecting teams to be deleted and clicking the 'click to delete selected teams' button
- Update operation can be done by selecting the team to update and entering the necessary attributes of that team and clicking the 'update' button.
- Adding a new team can be done by entering the information of the team in text boxes and clicking the 'add' button.



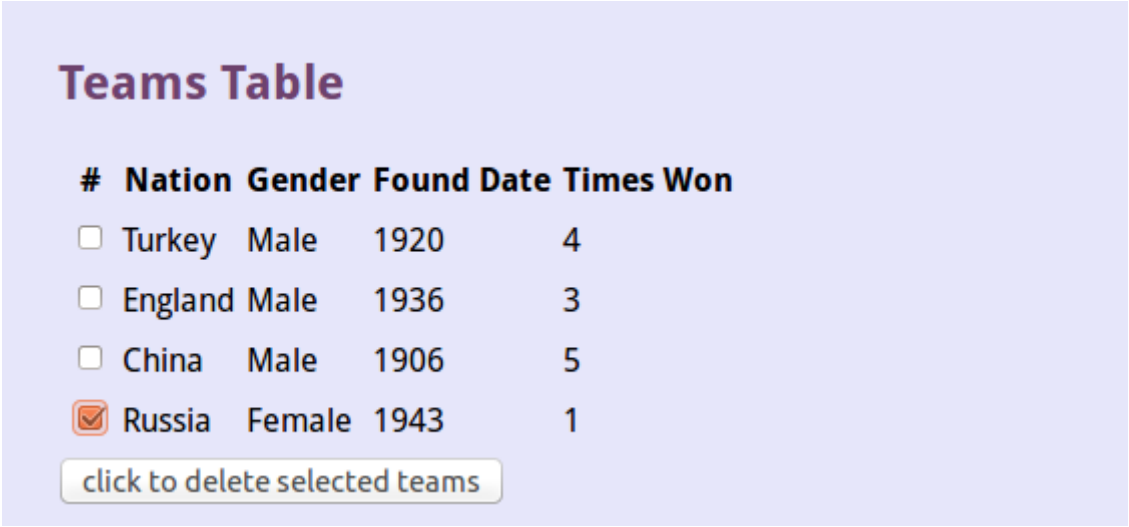


Fig. 1.5: Sample deletion

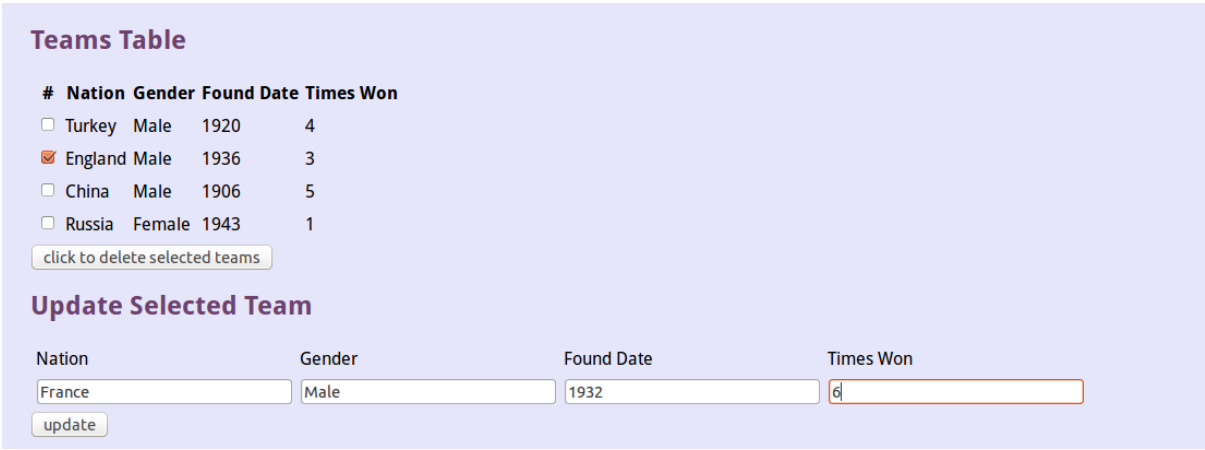


Fig. 1.6: Update Teams operation

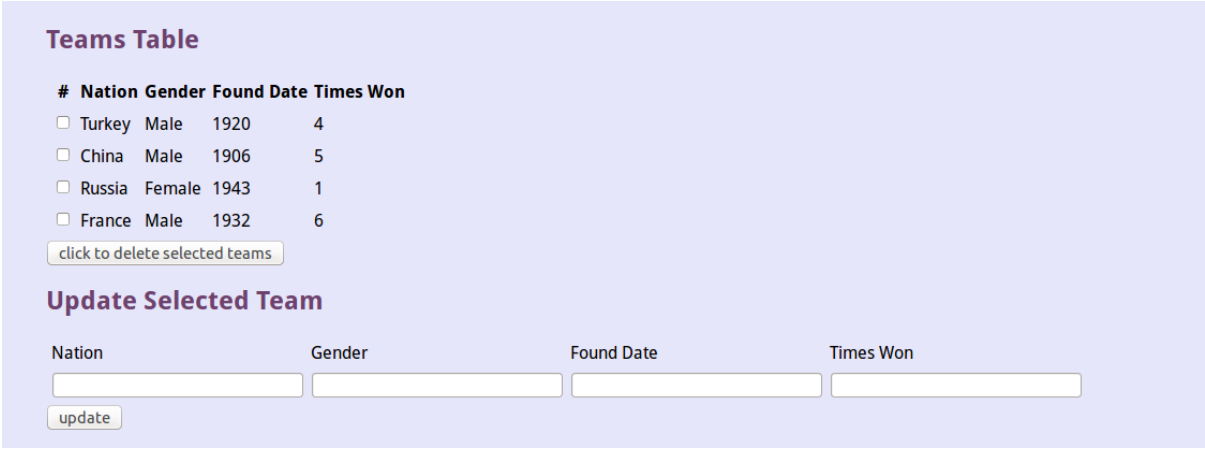


Fig. 1.7: Updated version

**Add New Team**

Nation	Gender	Found Date	Times Won
<input type="text" value="USA"/>	<input type="text" value="Female"/>	<input type="text" value="1924"/>	<input type="text" value="7"/>
<input type="button" value="add"/>			

Fig. 1.8: Adding new team operation

**Teams Table**

#	Nation	Gender	Found Date	Times Won
<input type="checkbox"/>	Turkey	Male	1920	4
<input type="checkbox"/>	England	Male	1936	3
<input type="checkbox"/>	China	Male	1906	5
<input type="checkbox"/>	Russia	Female	1943	1
<input type="checkbox"/>	USA	Female	1924	7

Fig. 1.9: New team added

## Players Page

- Like in teams page all players are listed in the Players Table Section
- Above part of the page find operation can be applied by all attributes. Found players are listed in the players table section. All players showed when find operation made with all text boxes empty.
- Delete operation can be done by selecting players to be deleted and clicking the ‘click to delete selected players’ button
- Update operation can be done by selecting the player to update and entering the necessary attributes of that player and clicking the ‘update’ button. Team attribute is selected from dropdown menu.
- Adding a new player can be done by entering the information of the player in text boxes, selecting the team from dropdown menu and clicking the ‘add’ button.

## Technic Members Page

- Like in teams and players page all technic members are listed.

## Players Table

#	Name	Gender	Nation	Birth Date	Team
<input type="checkbox"/>	Hasan	Male	Turkish	1994	Turkey
<input type="checkbox"/>	Rose	Female	English	1995	England
<input type="checkbox"/>	Dimitrov	Male	Russian	1993	Russia

[click to delete selected players](#)

Fig. 1.10: Players Table

### Find players

Name	Gender	Nation	Birth Date	Team	
<input type="text"/>	<input type="text" value="Female"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="find"/>

### Players Table

#	Name	Gender	Nation	Birth Date	Team
<input type="checkbox"/>	Rose	Female	English	1995	England

[click to delete selected players](#)

Fig. 1.11: Find Players Section

## Players Table

#	Name	Gender	Nation	Birth Date	Team
<input type="checkbox"/>	Hasan	Male	Turkish	1994	Turkey
<input checked="" type="checkbox"/>	Rose	Female	English	1995	England
<input type="checkbox"/>	Dimitrov	Male	Russian	1993	Russia

[click to delete selected players](#)

Fig. 1.12: Sample deletion

## Players Table

#	Name	Gender	Nation	Birth Date	Team
<input type="checkbox"/>	Hasan	Male	Turkish	1994	Turkey
<input type="checkbox"/>	Dimitrov	Male	Russian	1993	Russia

[click to delete selected players](#)

Fig. 1.13: Deleted version

## Players Table

#	Name	Gender	Nation	Birth Date	Team
<input checked="" type="checkbox"/>	Hasan	Male	Turkish	1994	Turkey
<input type="checkbox"/>	Rose	Female	English	1995	England
<input type="checkbox"/>	Dimitrov	Male	Russian	1993	Russia

[click to delete selected players](#)

### Update Selected Player

Name	Gender	Nation	Birth Date	Team
<input type="text" value="Furkan"/>	<input type="text" value="Male"/>	<input type="text" value="Turkish"/>	<input type="text" value="1995"/>	<input type="text" value="Turkey"/>

[update](#)

Fig. 1.14: Update Player operation

## Players Table

#	Name	Gender	Nation	Birth Date	Team
<input type="checkbox"/>	Rose	Female	English	1995	England
<input type="checkbox"/>	Dimitrov	Male	Russian	1993	Russia
<input type="checkbox"/>	Furkan	Male	Turkish	1995	Turkey

[click to delete selected players](#)

Fig. 1.15: Updated version

## Add New Player

Name	Gender	Nation	Birth Date	Team
<input type="text" value="Chiao"/>	<input type="text" value="Male"/>	<input type="text" value="Chinese"/>	<input type="text" value="1992"/>	<input type="text" value="China"/>

[add](#)

Fig. 1.16: Adding new player operation

## Players Table

#	Name	Gender	Nation	Birth Date	Team
<input type="checkbox"/>	Hasan	Male	Turkish	1994	Turkey
<input type="checkbox"/>	Rose	Female	English	1995	England
<input type="checkbox"/>	Dimitrov	Male	Russian	1993	Russia
<input type="checkbox"/>	Chiao	Male	Chinese	1992	China

[click to delete selected players](#)

Fig. 1.17: New player added

#	Name	Gender	Nation	Birth Date	Coach
<input type="checkbox"/>	Veli	Male	Turkish	1978	Zehra
<input type="checkbox"/>	Ayşe	Female	Turkish	1978	Zehra
<input type="checkbox"/>	Jane	Female	English	1982	Mike

Fig. 1.18: Technic Members Table

- Above part of the page find operation can be applied by all attributes. Found technic members are listed. All technic members showed when find operation made with all text boxes empty.

### Find technic members

Name	Gender	Nation	Birth Date	Coach	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	Zehra	<input type="button" value="find"/>

#	Name	Gender	Nation	Birth Date	Coach
<input type="checkbox"/>	Veli	Male	Turkish	1978	Zehra
<input type="checkbox"/>	Ayşe	Female	Turkish	1978	Zehra

Fig. 1.19: Find Technic Members Section

- Delete operation can be done by selecting technic members to be deleted and clicking the 'delete' button
- Update operation can be done by selecting the technic member to update and entering the necessary attributes of that technic member and clicking the 'update' button. Coach attribute is selected from dropdown menu.
- Adding a new technic member can be done by entering the information of the technic member in text boxes, selecting the coach from dropdown menu and clicking the 'add' button.

#	Name	Gender	Nation	Birth Date	Coach
<input checked="" type="checkbox"/>	Veli	Male	Turkish	1978	Zehra
<input type="checkbox"/>	Ayşe	Female	Turkish	1978	Zehra
<input type="checkbox"/>	Jane	Female	English	1982	Mike

Fig. 1.20: Sample deletion

#	Name	Gender	Nation	Birth Date	Coach
<input type="checkbox"/>	Ayşe	Female	Turkish	1978	Zehra
<input type="checkbox"/>	Jane	Female	English	1982	Mike

Fig. 1.21: Deleted version

#	Name	Gender	Nation	Birth Date	Coach
<input type="checkbox"/>	Veli	Male	Turkish	1978	Zehra
<input checked="" type="checkbox"/>	Ayşe	Female	Turkish	1978	Zehra
<input type="checkbox"/>	Jane	Female	English	1982	Mike

Name	Gender	Nation	Birth Date	Coach
<input type="text" value="Fatma"/>	<input type="text" value="Female"/>	<input type="text" value="Turkish"/>	<input type="text" value="1974"/>	<input type="text" value="Zehra"/>

Fig. 1.22: Update Technic Member operation

#	Name	Gender	Nation	Birth Date	Coach
<input type="checkbox"/>	Veli	Male	Turkish	1978	Zehra
<input type="checkbox"/>	Jane	Female	English	1982	Mike
<input type="checkbox"/>	Fatma	Female	Turkish	1974	Zehra

Fig. 1.23: Updated version

**Add New Technic Member**

Name	Gender	Nation	Birth Date	Coach
<input type="text" value="Cheng"/>	<input type="text" value="Male"/>	<input type="text" value="Chinese"/>	<input type="text" value="1984"/>	<input type="text" value="Chan"/>
<input type="button" value="add"/>				

Fig. 1.24: Adding new technic member operation

#	Name	Gender	Nation	Birth Date	Coach
<input type="checkbox"/>	Veli	Male	Turkish	1978	Zehra
<input type="checkbox"/>	Ayşe	Female	Turkish	1978	Zehra
<input type="checkbox"/>	Jane	Female	English	1982	Mike
<input type="checkbox"/>	Cheng	Male	Chinese	1984	Chan

Fig. 1.25: New technic member added

## Parts Implemented by Alican Mertan

### Pages

1. *Tournaments Page*
2. *Matches Page*

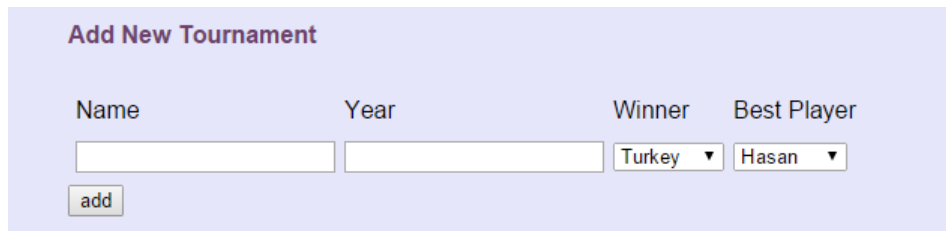
### Tournaments Page

- In tournaments page all tournaments are listed in the Tournaments Section.

Tournaments				
#	Name	Year	Winner	Best Player
<input type="checkbox"/>	World Cup 2015	Turkey	Hasan	
<input type="checkbox"/>	World Cup 2014	England	Dimitrov	
<input type="checkbox"/>	World Cup 2013	China	Rose	
<input type="checkbox"/>	World Cup 2015	Turkey	Hasan	
<input type="checkbox"/>	World Cup 2014	England	Dimitrov	
<input type="checkbox"/>	World Cup 2013	China	Rose	
<input type="button" value="delete"/>				

Fig. 1.26: Tournaments Table

- Adding a new tournament can be done by entering the information and clicking the 'add' button. Text boxes are used to get the name and the year of the tournament and dropdown lists are used to select winner team and the best player from existing ones.



The 'Add New Tournament' form is a light blue rectangular box. At the top left, the title 'Add New Tournament' is written in a dark purple font. Below the title, there are four labels: 'Name', 'Year', 'Winner', and 'Best Player'. Under 'Name' and 'Year' are empty text input boxes. Under 'Winner' is a dropdown menu showing 'Turkey' with a downward arrow. Under 'Best Player' is a dropdown menu showing 'Hasan' with a downward arrow. At the bottom left of the form is a small button labeled 'add'.

Fig. 1.27: Adding new tournament

- Delete operation can be done by selecting tournaments to be deleted and clicking the 'delete' button.



The 'Tournaments' section is a light blue rectangular box. At the top left, the title 'Tournaments' is written in a dark purple font. Below the title is a table with the following structure:

#	Name	Year	Winner	Best Player
<input type="checkbox"/>	World Cup 2015	Turkey	Hasan	
<input type="checkbox"/>	World Cup 2014	England	Dimitrov	
<input type="checkbox"/>	World Cup 2013	China	Rose	
<input type="checkbox"/>	World Cup 2015	Turkey	Hasan	
<input type="checkbox"/>	World Cup 2014	England	Dimitrov	
<input type="checkbox"/>	World Cup 2013	China	Rose	

At the bottom left of the table is a small button labeled 'delete'.

Fig. 1.28: Sample deletion

- Search operation can be done in the 'Search and Update' section by entering the information and clicking the 'find' button. If a search made with empty boxes, all the tuples will be shown.



The 'Search and Update' form is a light blue rectangular box. At the top left, the title 'Search and Update' is written in a dark purple font. Below the title, there are four labels: 'Name', 'Year', 'Winner', and 'Best Player'. Under each label is an empty text input box. At the bottom left of the form is a small button labeled 'find'.

Fig. 1.29: Search Operation

- After a search operation, update operation can be done within queried tuples. Update can be done by changing the informations and clicking the related 'update' button.

## Matches Page

- In matches page all matches are listed in the Matches Section.



Name	Year	Current Winner	New Winner	Current Best Player	New Best Player	
World Cup	2015	Turkey	Turkey ▼	Hasan	Hasan ▼	Update
World Cup	2015	Turkey	Turkey ▼	Hasan	Hasan ▼	Update

Fig. 1.30: Update Operation

**Matches**

#	Tournament	Team1	Team2	Score
<input type="checkbox"/>	World Cup	Turkey	England	5-3
<input type="checkbox"/>	World Cup	China	Russia	4-2
<input type="checkbox"/>	World Cup	China	England	2-6
<input type="checkbox"/>	World Cup	Turkey	England	5-3
<input type="checkbox"/>	World Cup	China	Russia	4-2
<input type="checkbox"/>	World Cup	China	England	2-6

Fig. 1.31: Matches Table

- Adding a new match can be done by entering the information and clicking the 'add' button. Text boxes are used to get the score of the match and dropdown lists are used to select tournament and teams from existing ones.

**Add New Match**

Tournament	Team1	Team2	Score
World Cup ▼	Turkey ▼	Turkey ▼	<input type="text"/>

Fig. 1.32: Adding new match

- Delete operation can be done by selecting matches to be deleted and clicking the 'delete' button.
- Search operation can be done in the 'Search and Update' section by entering the information and clicking the 'find' button. If a search made with empty boxes, all the tuples will be shown.
- After a search operation, update operation can be done within queried tuples. Update can be done by changing the informations and clicking the related 'update' button.

Matches

#	Tournament	Team1	Team2	Score
<input type="checkbox"/>	World Cup	Turkey	England	5-3
<input type="checkbox"/>	World Cup	China	Russia	4-2
<input type="checkbox"/>	World Cup	China	England	2-6
<input type="checkbox"/>	World Cup	Turkey	England	5-3
<input type="checkbox"/>	World Cup	China	Russia	4-2
<input type="checkbox"/>	World Cup	China	England	2-6

delete

Fig. 1.33: Sample deletion

Search and Update

Tournament	Team1	Team2	Score
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

find

Fig. 1.34: Search Operation

Current Tournament	New Tournament	Current Team1	New Team1	Current Team2	New Team2	Score
World Cup	<div>World Cup ▾</div>	China	<div>Turkey ▾</div>	Russia	<div>Turkey ▾</div>	<div>4-2</div> <div>Update</div>
World Cup	<div>World Cup ▾</div>	China	<div>Turkey ▾</div>	England	<div>Turkey ▾</div>	<div>2-6</div> <div>Update</div>

Fig. 1.35: Update Operation

## Parts Implemented by Ahmet Yilmaz

### Pages

1. *Coaches Page*
2. *Player Statistics Page*
3. *Users Page*

### Coaches Page

- Coaches listed in this page.

Delete	Name	Gender	Nationality	Birth Date	Current Team	
<input type="checkbox"/>	Zehra	female	turkish	1964	Turkey ▾	<input type="button" value="Update"/>
<input type="checkbox"/>	Mike	male	english	1954	England ▾	<input type="button" value="Update"/>
<input type="checkbox"/>	Chan	male	chinese	1962	China ▾	<input type="button" value="Update"/>
<input type="button" value="Delete"/>						

Fig. 1.36: Coaches Table

- Under the Search header in page there is textboxes for each attribute to find in database according to values entered in checkboxes. Single attribute or more than one attribute can be used for searching.

**Search**

Name	Gender	Nationality	Birth Date	Current Team
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Find"/>	<input type="button" value="Show All"/>			

Fig. 1.37: Find Coaches Section

Name	Gender	Nationality	Birth Date	Current Team
<input type="text"/>	<input type="text" value="male"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Find"/>	<input type="button" value="Show All"/>			

Delete	Name	Gender	Nationality	Birth Date	Current Team	
<input type="checkbox"/>	Mike	male	english	1954	England ▾	<input type="button" value="Update"/>
<input type="checkbox"/>	Chan	male	chinese	1962	China ▾	<input type="button" value="Update"/>
<input type="button" value="Delete"/>						

Fig. 1.38: Sample find operation

- on the left side of table there is checkboxes for every tuple to request from HTML. Then requested tuples can be deleted by clicking on delete button below.
- Update operation can be done by simply entering new value in textbox of which attribute preferred then clicking on the Update button on the right. Team attribute selected from dropdown menu.

Delete	Name	Gender	Nationality	Birth Date	Current Team	
<input type="checkbox"/>	Mike	male	english	1954	England	<input type="button" value="Update"/>
<input checked="" type="checkbox"/>	Chan	male	chinese	1962	China	<input type="button" value="Update"/>
<input type="checkbox"/>	Zehra	female	turkish	1982	China	<input type="button" value="Update"/>
<input type="button" value="Delete"/>						

Fig. 1.39: Sample deletion

Delete	Name	Gender	Nationality	Birth Date	Current Team	
<input type="checkbox"/>	Zehra	female	turkish	1982	Turkey	<input type="button" value="Update"/>
<input type="checkbox"/>	Mike	male	english	1954	England	<input type="button" value="Update"/>
<input type="checkbox"/>	Chan	male	chinese	1962	China	<input type="button" value="Update"/>
<input type="button" value="Delete"/>						

Fig. 1.40: Update Coaches operation

- Adding a new coach can be done by entering the information of the coach in text boxes and clicking the 'add' button. Team attribute selected from dropdown menu.

## Player Statistics Page

- Player Statistics listed in the table section
- Under the Search header in page there is textboxes for each attribute to find in database according to values entered in checkboxes. Single attribute or more than one attribute can be used for searching.
- On the left side of table there is checkboxes for every tuple to request from HTML. Then requested tuples can be deleted by clicking on delete button below.
- Update operation can be done by simply entering new value in textbox of which attribute preferred then clicking on the Update button on the right. Player selected from dropdown menu.
- Adding a new player can be done by entering the information of the player in text boxes, selecting the player from dropdown menu and clicking the 'add' button.

Delete	Name	Gender	Nationality	Birth Date	Current Team	
<input type="checkbox"/>	Mike	male	english	1954	England	<input type="button" value="Update"/>
<input type="checkbox"/>	Chan	male	chinese	1962	China	<input type="button" value="Update"/>
<input type="checkbox"/>	Zehra	female	turkish	1982	China	<input type="button" value="Update"/>
<input type="button" value="Delete"/>						

Fig. 1.41: Updated version

Add New Coach

Name

Gender

Nationality

Birth Date

Current Team

Ahmet

male

turkish

1994

Turkey

Add

Fig. 1.42: Adding new coach operation

Delete	Name	Gender	Nationality	Birth Date	Current Team	
<input type="checkbox"/>	Mike	male	english	1954	England	Update
<input type="checkbox"/>	Chan	male	chinese	1962	China	Update
<input type="checkbox"/>	Zehra	female	turkish	1982	China	Update
<input type="checkbox"/>	Ahmet	male	turkish	1994	Turkey	Update
Delete						

Fig. 1.43: New coach added

Delete	Player	Matches Played	Matches Won	Win Rate	Average Score	
<input type="checkbox"/>	Hasan	2	3	2.3	1.4	Update
<input type="checkbox"/>	Rose	4	2	3.7	1.7	Update
<input type="checkbox"/>	Dimitrov	2	3	2.3	1.2	Update
Delete						

Fig. 1.44: Player Statistics Table

Player Statistics

Search

Player

Matches Played

Matches Won

Win Rate

Average Score

Hasan

Find

Show All

Delete

Player

Matches Played

Matches Won

Win Rate

Average Score

☐

Hasan

2

3

2.3

1.4

Update

Delete

Fig. 1.45: Find Player Section

Delete	Player	Matches Played	Matches Won	Win Rate	Average Score	
<input type="checkbox"/>	Hasan	2	3	2.3	1.4	Update
<input checked="" type="checkbox"/>	Rose	4	2	3.7	1.7	Update
<input type="checkbox"/>	Dimitrov	2	3	2.3	1.2	Update
Delete						

Fig. 1.46: Sample deletion

Delete	Player	Matches Played	Matches Won	Win Rate	Average Score	
<input type="checkbox"/>	Hasan ▾	<input type="text" value="2"/>	<input type="text" value="3"/>	<input type="text" value="2.3"/>	<input type="text" value="1.4"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	Dimitrov ▾	<input type="text" value="2"/>	<input type="text" value="3"/>	<input type="text" value="2.3"/>	<input type="text" value="1.2"/>	<input type="button" value="Update"/>
<input type="button" value="Delete"/>						

Fig. 1.47: Deleted version

Delete	Player	Matches Played	Matches Won	Win Rate	Average Score	
<input type="checkbox"/>	Hasan ▾	<input type="text" value="2"/>	<input type="text" value="3"/>	<input type="text" value="2.3"/>	<input type="text" value="1.4"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	Rose ▾	<input type="text" value="2"/>	<input type="text" value="3"/>	<input type="text" value="2.3"/>	<input type="text" value="1.2"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	Dimitrov ▾	<input type="text" value="2"/>	<input type="text" value="3"/>	<input type="text" value="2.3"/>	<input type="text" value="1.2"/>	<input type="button" value="Update"/>
<input type="button" value="Delete"/>						

Fig. 1.48: Update Player operation

Delete	Player	Matches Played	Matches Won	Win Rate	Average Score	
<input type="checkbox"/>	Dimitrov ▾	<input type="text" value="2"/>	<input type="text" value="3"/>	<input type="text" value="2.3"/>	<input type="text" value="1.2"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	Rose ▾	<input type="text" value="2"/>	<input type="text" value="3"/>	<input type="text" value="2.3"/>	<input type="text" value="1.4"/>	<input type="button" value="Update"/>
<input type="button" value="Delete"/>						

Fig. 1.49: Updated version

Add New Player				
Player	Matches Played	Matches Won	Win Rate	Average Score
Hasan ▾	<input type="text" value="17"/>	<input type="text" value="12"/>	<input type="text" value="0.7"/>	<input type="text" value="17.8"/>
<input type="button" value="Add"/>				

Fig. 1.50: Adding new player operation

Delete	Player	Matches Played	Matches Won	Win Rate	Average Score	
<input type="checkbox"/>	Dimitrov ▾	<input type="text" value="2"/>	<input type="text" value="3"/>	<input type="text" value="2.3"/>	<input type="text" value="1.2"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	Rose ▾	<input type="text" value="2"/>	<input type="text" value="3"/>	<input type="text" value="2.3"/>	<input type="text" value="1.4"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	Hasan ▾	<input type="text" value="17"/>	<input type="text" value="12"/>	<input type="text" value="0.7"/>	<input type="text" value="17.8"/>	<input type="button" value="Update"/>
<input type="button" value="Delete"/>						

Fig. 1.51: New player added

## Users Page

- Users are listed.

Delete	Username	Password	
<input type="checkbox"/>	hasan	hasan	Update
<input type="checkbox"/>	firat	firat	Update
<input type="checkbox"/>	ahmet	ahmet	Update
<input type="checkbox"/>	alican	alican	Update
<input type="button" value="delete"/>			

Fig. 1.52: Users Table

- Users can be found according to username and password.

### Users

#### Search

Username

hasan

Password

Delete	Username	Password	
<input type="checkbox"/>	hasan	hasan	Update
<input type="button" value="delete"/>			

Fig. 1.53: Find Users Section

- Delete operation can be done by selecting user to be deleted and clicking the ‘delete’ button

Delete	Username	Password	
<input type="checkbox"/>	hasan	hasan	Update
<input type="checkbox"/>	firat	firat	Update
<input checked="" type="checkbox"/>	ahmet	ahmet	Update
<input type="checkbox"/>	alican	alican	Update
<input type="button" value="delete"/>			

Fig. 1.54: Sample deletion

- Update operation can be done by entering new values and clicking on update button.
- Adding a new user can be done by entering the information of the user in text boxes then clicking the ‘add’ button.

Delete	Username	Password	
<input type="checkbox"/>	hasan	hasan	Update
<input type="checkbox"/>	firat	firat	Update
<input type="checkbox"/>	alican	alican	Update

delete

Fig. 1.55: Deleted version

Delete	Username	Password	
<input type="checkbox"/>	hasan	123456	Update
<input type="checkbox"/>	firat	firat	Update
<input type="checkbox"/>	alican	alican	Update

delete

Fig. 1.56: Update user operation

Delete	Username	Password	
<input type="checkbox"/>	firat	firat	Update
<input type="checkbox"/>	alican	alican	Update
<input type="checkbox"/>	hasan	123456	Update

delete

Fig. 1.57: Updated version

### Add New User

Username	Password
ali	qwe

add

Fig. 1.58: Adding new user operation

Delete	Username	Password	
<input type="checkbox"/>	firat	firat	Update
<input type="checkbox"/>	alican	alican	Update
<input type="checkbox"/>	hasan	123456	Update
<input type="checkbox"/>	ali	qwe	Update

delete

Fig. 1.59: New user added



## Parts Implemented by First Bayram

### Pages

- 1. *Referees Page*
- 2. *Match Statistics Page*
- 3. *Stadiums Page*

### Referees Page

- Referees are listed in this page.

Delete	Name	Gender	Nationality	Birth Date	Times Match	
<input type="checkbox"/>	<input type="text" value="chan"/>	<input type="text" value="male"/>	<input type="text" value="china"/>	<input type="text" value="1974"/>	<input type="text" value="4"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	<input type="text" value="kim"/>	<input type="text" value="male"/>	<input type="text" value="japan"/>	<input type="text" value="1978"/>	<input type="text" value="5"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	<input type="text" value="cho"/>	<input type="text" value="female"/>	<input type="text" value="china"/>	<input type="text" value="1976"/>	<input type="text" value="3"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	<input type="text" value="alex"/>	<input type="text" value="male"/>	<input type="text" value="brazil"/>	<input type="text" value="1982"/>	<input type="text" value="2"/>	<input type="button" value="Update"/>
<input type="button" value="delete"/>						

Fig. 1.60: Referees Table

- There is checkboxes for each attribute of referees table to find the entered value in checkboxes by searching database. One or more attributes can be searched.

**Search**

Name

Gender

Nationality

Birth Date

Times Match

Fig. 1.61: Find Referees Section

**Search**

Name

Gender

Nationality

Birth Date

Times Match

Delete	Name	Gender	Nationality	Birth Date	Times Match	
<input type="checkbox"/>	<input type="text" value="chan"/>	<input type="text" value="male"/>	<input type="text" value="china"/>	<input type="text" value="1974"/>	<input type="text" value="4"/>	<input type="button" value="Update"/>

Fig. 1.62: Sample find operation

- On the left side of each tuple a checkbox exists. With delete button below preferred tuple(referee) can be deleted.

Delete	Name	Gender	Nationality	Birth Date	Times Match	
<input type="checkbox"/>	chan	male	china	1974	4	<input type="button" value="Update"/>
<input type="checkbox"/>	kim	male	japan	1978	5	<input type="button" value="Update"/>
<input checked="" type="checkbox"/>	cho	female	china	1976	3	<input type="button" value="Update"/>
<input type="checkbox"/>	alex	male	brazil	1982	2	<input type="button" value="Update"/>
<input type="button" value="delete"/>						

Fig. 1.63: Sample deletion

- By entering new information about referee in textbox of chosen attribute and using update button on the right side update operation can be succeed.

Delete	Name	Gender	Nationality	Birth Date	Times Match	
<input type="checkbox"/>	chan	male	china	1974	4	<input type="button" value="Update"/>
<input type="checkbox"/>	kim	male	japan	1978	5	<input type="button" value="Update"/>
<input type="checkbox"/>	cho	female	china	1976	3	<input type="button" value="Update"/>
<input type="checkbox"/>	alex	male	brazil	1982	2	<input type="button" value="Update"/>
<input type="checkbox"/>	alican	male	turkish	20.01.1993	232	<input type="button" value="Update"/>
<input type="button" value="delete"/>						

Fig. 1.64: Update Referee operation

Delete	Name	Gender	Nationality	Birth Date	Times Match	
<input type="checkbox"/>	chan	male	china	1974	4	<input type="button" value="Update"/>
<input type="checkbox"/>	kim	male	japan	1978	5	<input type="button" value="Update"/>
<input type="checkbox"/>	alex	male	brazil	1982	2	<input type="button" value="Update"/>
<input type="checkbox"/>	alican	male	turkish	20.01.1993	232	<input type="button" value="Update"/>
<input type="checkbox"/>	cho	female	germany	1976	3	<input type="button" value="Update"/>
<input type="button" value="delete"/>						

Fig. 1.65: Updated Referee

- Entering the values of new referee, add operation can be done by using add button.

## Stadiums Page

- Stadiums are listed in this section
- There is checkboxes for each attribute of stadiums table to find the entered value in checkboxes by searching database. One or more attributes can be searched.
- On the left side of each tuple a checkbox exists. With delete button below preferred tuple(stadium) can be deleted.
- By entering new information about stadium in textbox of chosen attribute and using update button on the right side update operation can be succeed. Country information can be selected from dropdown menu.

**Add New Referee**

Name	Gender	Nationality	Birth Date	Times Match
<input type="text" value="alican"/>	<input type="text" value="male"/>	<input type="text" value="turkish"/>	<input type="text" value="20.01.1993"/>	<input type="text" value="232"/>

Fig. 1.66: Adding new referee operation

Delete	Name	Gender	Nationality	Birth Date	Times Match	
<input type="checkbox"/>	<input type="text" value="chan"/>	<input type="text" value="male"/>	<input type="text" value="china"/>	<input type="text" value="1974"/>	<input type="text" value="4"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	<input type="text" value="kim"/>	<input type="text" value="male"/>	<input type="text" value="japan"/>	<input type="text" value="1978"/>	<input type="text" value="5"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	<input type="text" value="cho"/>	<input type="text" value="female"/>	<input type="text" value="china"/>	<input type="text" value="1976"/>	<input type="text" value="3"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	<input type="text" value="alex"/>	<input type="text" value="male"/>	<input type="text" value="brazil"/>	<input type="text" value="1982"/>	<input type="text" value="2"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	<input type="text" value="alican"/>	<input type="text" value="male"/>	<input type="text" value="turkish"/>	<input type="text" value="20.01.1993"/>	<input type="text" value="232"/>	<input type="button" value="Update"/>

Fig. 1.67: New referee added

Delete	Name	Capacity	City	Country	
<input type="checkbox"/>	<input type="text" value="Marmara"/>	<input type="text" value="20000"/>	<input type="text" value="Istanbul"/>	<input type="text" value="Turkey"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	<input type="text" value="Wembley"/>	<input type="text" value="30000"/>	<input type="text" value="London"/>	<input type="text" value="England"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	<input type="text" value="Moscow"/>	<input type="text" value="25000"/>	<input type="text" value="St.Petersburg"/>	<input type="text" value="Russia"/>	<input type="button" value="Update"/>

Fig. 1.68: Stadiums Table

**Search**

Name	Capacity	City	Country
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Fig. 1.69: Find Stadium

**Search**

Name	Capacity	City	Country
<input type="text" value="m"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Delete	Name	Capacity	City	Country	
<input type="checkbox"/>	<input type="text" value="Marmara"/>	<input type="text" value="20000"/>	<input type="text" value="Istanbul"/>	<input type="text" value="Turkey"/>	<input type="button" value="Update"/>

Fig. 1.70: Find Stadium

Delete	Name	Capacity	City	Country	
<input checked="" type="checkbox"/>	Marmara	20000	Istanbul	Turkey ▼	Update
<input type="checkbox"/>	Wembley	30000	London	England ▼	Update
<input type="checkbox"/>	Moscow	25000	St.Petersburg	Russia ▼	Update
Delete					

Fig. 1.71: Sample deletion

Delete	Name	Capacity	City	Country	
<input type="checkbox"/>	Wembley	30000	London	England ▼	Update
<input type="checkbox"/>	Moscow	25000	St.Petersburg	Russia ▼	Update
Delete					

Fig. 1.72: Deleted Version

Delete	Name	Capacity	City	Country	
<input type="checkbox"/>	Wembley	30000	London	England ▼	Update
<input type="checkbox"/>	Moscow	25000	St.Petersburg	Turkey England China Russia	Update
Delete					

Fig. 1.73: Update Stadium

Delete	Name	Capacity	City	Country	
<input type="checkbox"/>	Moscow	25000	St.Petersburg	Russia ▼	Update
<input type="checkbox"/>	Wembley	30000	London	England ▼	Update
Delete					

Fig. 1.74: Updated Stadium

- Entering the values of new stadium, add operation can be done by using add button. Country information can be selected from dropdown menu.

**Add New Stadium**

Name	Capacity	City	Country
<input type="text" value="Istanbul"/>	<input type="text" value="40000"/>	<input type="text" value="Istanbul"/>	<input type="text" value="Turkey"/>
<input type="button" value="Add"/>			

Fig. 1.75: Add Stadium

Delete	Name	Capacity	City	Country	
<input type="checkbox"/>	<input type="text" value="Moscow"/>	<input type="text" value="25000"/>	<input type="text" value="St.Petersburg"/>	<input type="text" value="Russia"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	<input type="text" value="Wembley"/>	<input type="text" value="30000"/>	<input type="text" value="London"/>	<input type="text" value="England"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	<input type="text" value="Istanbul"/>	<input type="text" value="40000"/>	<input type="text" value="Istanbul"/>	<input type="text" value="Turkey"/>	<input type="button" value="Update"/>
<input type="button" value="Delete"/>					

Fig. 1.76: New stadium added

## Match Statistics Page

- Match Statistics are listed.

Delete	Home Team	Current Home Team	Away Team	Current Away Team	Referee	Current Referee	Match Date	Score	
<input type="checkbox"/>	Turkey	<input type="text" value="Turkey"/>	England	<input type="text" value="Turkey"/>	chan	<input type="text" value="chan"/>	<input type="text" value="12-10-2014"/>	<input type="text" value="4-3"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	China	<input type="text" value="Turkey"/>	England	<input type="text" value="Turkey"/>	kim	<input type="text" value="chan"/>	<input type="text" value="21-03-2015"/>	<input type="text" value="7-1"/>	<input type="button" value="Update"/>
<input type="checkbox"/>	Russia	<input type="text" value="Turkey"/>	Turkey	<input type="text" value="Turkey"/>	kim	<input type="text" value="chan"/>	<input type="text" value="07-06-2014"/>	<input type="text" value="6-2"/>	<input type="button" value="Update"/>
<input type="button" value="Delete"/>									

Fig. 1.77: Match Statistics Table

- There is checkboxes for each attribute of match statistics table to find the entered value in checkboxes by searching in database. One or more attributes can be searched.
- On the left side of each tuple a checkbox exists. With delete button below preferred tuple(match statistics) can be deleted.
- By entering new information about match statistics in textbox of chosen attribute and using update button on the right side update operation can be succeed.Current Home Team Current Away Team and Current Referee can be selected from dropdown menu.

**Search**

Home Team	Away Team	Match Place	Score	Referee
<input type="text" value="Russia"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Delete	Home Team	Current Home Team	Away Team	Current Away Team	Referee	Current Referee	Match Date	Score
<input type="checkbox"/>	Russia	<input type="text" value="Turkey"/>	Turkey	<input type="text" value="Turkey"/>	kim	<input type="text" value="chan"/>	<input type="text" value="07-06-2014"/>	<input type="text" value="6-2"/> <input type="button" value="Update"/>

Fig. 1.78: Find Match Statistics

Delete	Home Team	Current Home Team	Away Team	Current Away Team	Referee	Current Referee	Match Date	Score
<input type="checkbox"/>	Turkey	<input type="text" value="Turkey"/>	England	<input type="text" value="Turkey"/>	chan	<input type="text" value="chan"/>	<input type="text" value="12-10-2014"/>	<input type="text" value="4-3"/> <input type="button" value="Update"/>
<input type="checkbox"/>	China	<input type="text" value="Turkey"/>	England	<input type="text" value="Turkey"/>	kim	<input type="text" value="chan"/>	<input type="text" value="21-03-2015"/>	<input type="text" value="7-1"/> <input type="button" value="Update"/>
<input checked="" type="checkbox"/>	Russia	<input type="text" value="Turkey"/>	Turkey	<input type="text" value="Turkey"/>	kim	<input type="text" value="chan"/>	<input type="text" value="07-06-2014"/>	<input type="text" value="6-2"/> <input type="button" value="Update"/>

Fig. 1.79: Sample deletion

Delete	Home Team	Current Home Team	Away Team	Current Away Team	Referee	Current Referee	Match Date	Score
<input type="checkbox"/>	Turkey	<input type="text" value="Turkey"/>	England	<input type="text" value="Turkey"/>	chan	<input type="text" value="chan"/>	<input type="text" value="12-10-2014"/>	<input type="text" value="4-3"/> <input type="button" value="Update"/>
<input type="checkbox"/>	China	<input type="text" value="Turkey"/>	England	<input type="text" value="Turkey"/>	kim	<input type="text" value="chan"/>	<input type="text" value="21-03-2015"/>	<input type="text" value="7-1"/> <input type="button" value="Update"/>

Fig. 1.80: Deleted version

Delete	Home Team	Current Home Team	Away Team	Current Away Team	Referee	Current Referee	Match Date	Score
<input type="checkbox"/>	Turkey	<input type="text" value="Turkey"/>	England	<input type="text" value="Turkey"/>	chan	<input type="text" value="chan"/>	<input type="text" value="12-10-2014"/>	<input type="text" value="4-3"/> <input type="button" value="Update"/>
<input type="checkbox"/>	China	<input type="text" value="Turkey"/>	England	<input type="text" value="Turkey"/>	kim	<input type="text" value="chan"/>	<input type="text" value="21-03-2015"/>	<input type="text" value="7-1"/> <input type="button" value="Update"/>
<input type="checkbox"/>	China	<input type="text" value="Turkey"/>	England	<input type="text" value="Turkey"/>	cho	<input type="text" value="chan"/>	<input type="text" value="12-12-2012"/>	<input type="text" value="5-3"/> <input type="button" value="Update"/>

**Add New Match**

chan  
kim  
cho  
alex

Fig. 1.81: Update match statistics

Delete	Home Team	Current Home Team	Away Team	Current Away Team	Referee	Current Referee	Match Date	Score
<input type="checkbox"/>	Turkey	<input type="text" value="Turkey"/>	England	<input type="text" value="Turkey"/>	chan	<input type="text" value="chan"/>	<input type="text" value="12-10-2014"/>	<input type="text" value="4-3"/> <input type="button" value="Update"/>
<input type="checkbox"/>	China	<input type="text" value="Turkey"/>	England	<input type="text" value="Turkey"/>	kim	<input type="text" value="chan"/>	<input type="text" value="21-03-2015"/>	<input type="text" value="7-1"/> <input type="button" value="Update"/>
<input type="checkbox"/>	Turkey	<input type="text" value="Turkey"/>	Turkey	<input type="text" value="Turkey"/>	alex	<input type="text" value="chan"/>	<input type="text" value="12-12-2012"/>	<input type="text" value="5-3"/> <input type="button" value="Update"/>

Fig. 1.82: Updated match statistics

- Entering the values of new match statistics, add operation can be done by using add button. Home Team and Away Team can be selected from dropdown menu.

**Add New Match**

Home Team	Away Team	Match Date	Score	Referee
China ▼	England ▼	12-12-2012	5-3	cho ▼
<input type="button" value="Add"/>				

Fig. 1.83: Add match statistics

Delete	Home Team	Current Home Team	Away Team	Current Away Team	Referee	Current Referee	Match Date	Score	
<input type="checkbox"/>	Turkey	Turkey ▼	England	Turkey ▼	chan	chan ▼	12-10-2014	4-3	<input type="button" value="Update"/>
<input type="checkbox"/>	China	Turkey ▼	England	Turkey ▼	kim	chan ▼	21-03-2015	7-1	<input type="button" value="Update"/>
<input type="checkbox"/>	China	Turkey ▼	England	Turkey ▼	cho	chan ▼	12-12-2012	5-3	<input type="button" value="Update"/>
<input type="button" value="Delete"/>									

Fig. 1.84: New match statistics added





### Database Design

- E/R Diagram of table tennis database:

### Code

#### Parts Implemented by Hasan Burak NAMLI

##### Database Design

##### 1 Tables

##### 1.1 Teams Table

- Teams table keeping record of the teams data

Name	Type	Not Null	Primary K.
ID	INTEGER	0	1
NATION	VARCHAR	0	0
GENDER	VARCHAR	0	0
FOUNDDATE	VARCHAR	0	0
TIMESWON	VARCHAR	0	0

- *nation* keeps the record of nation of the given team.
- *gender* keeps the record of gender of the given team.
- *founddate* keeps the record of the found date of the given team.
- *timeswon* keeps the record of how many times team has won the game.

**Sql statement that initialize the teams table:**

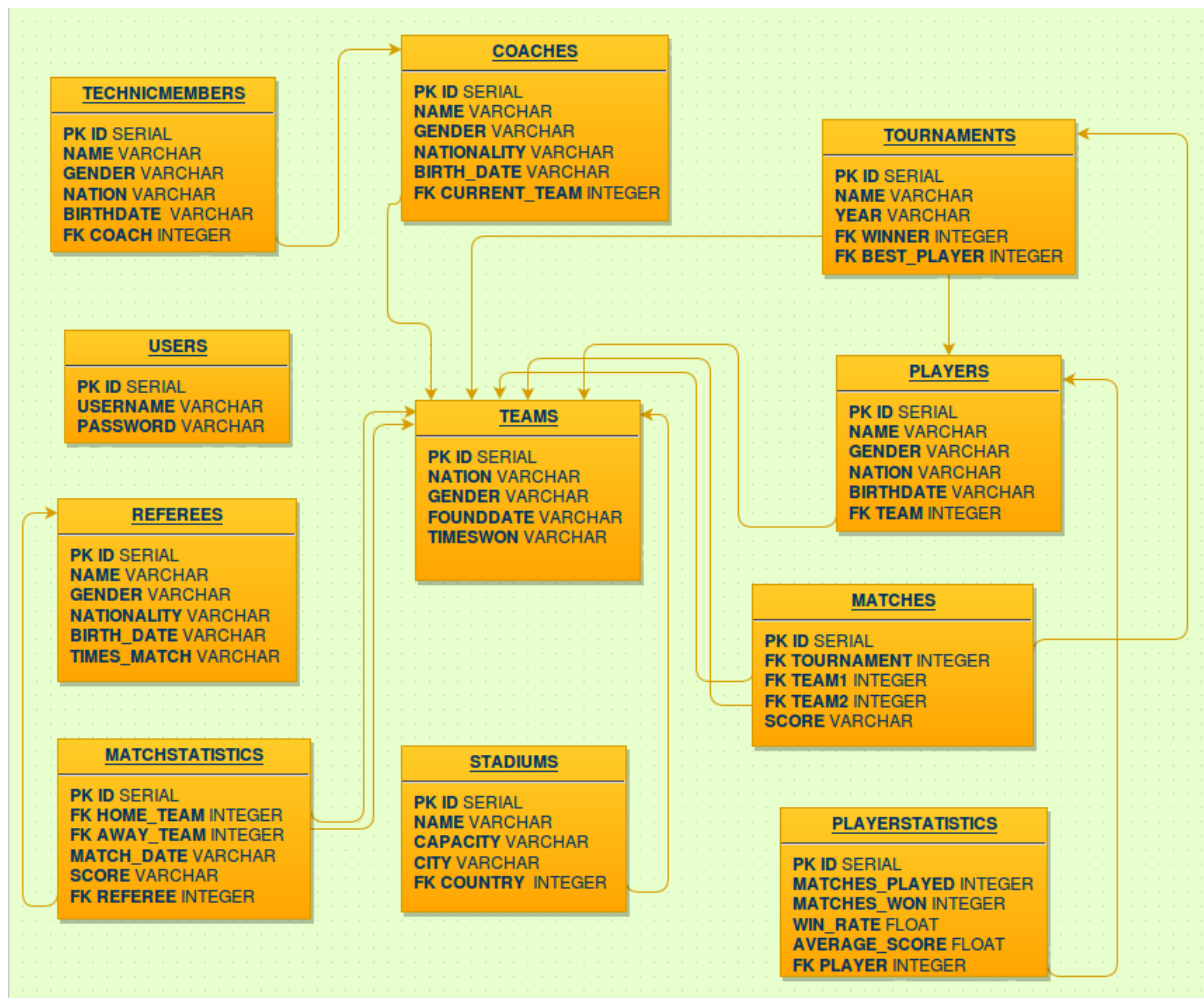


Fig. 2.1: E/R Diagram

```
CREATE TABLE TEAMS (
  ID SERIAL PRIMARY KEY,
  NATION VARCHAR(45),
  GENDER VARCHAR(6),
  FOUNDDATE VARCHAR(20),
  TIMESWON VARCHAR(10)
)
```

## 1.2 Players Table

- Players table keeping record of the players data

Name	Type	Not Null	Primary K.
ID	INTEGER	0	1
NAME	VARCHAR	0	0
GENDER	VARCHAR	0	0
NATION	VARCHAR	0	0
BIRTHDATE	VARCHAR	0	0
TEAM	INTEGER	0	0

- *name* keeps the record of name of the given player.
- *gender* keeps the record of gender of the given player.
- *nation* keeps the record of nation of the given player.
- *birthdate* keeps the record of birth date of the given player.
- *team* references to *teams* table and on delete and update operations it cascades the operation

Sql statement that initialize the players table:

```
CREATE TABLE PLAYERS (
  ID SERIAL PRIMARY KEY,
  NAME VARCHAR(45),
  GENDER VARCHAR(6),
  NATION VARCHAR(45),
  BIRTHDATE VARCHAR(10),
  TEAM INTEGER REFERENCES TEAMS ON DELETE CASCADE ON UPDATE CASCADE
)
```

## 1.3 Technic Members Table

- Technic members table keeping record of the technic members data

Name	Type	Not Null	Primary K.
ID	INTEGER	0	1
NAME	VARCHAR	0	0
GENDER	VARCHAR	0	0
NATION	VARCHAR	0	0
BIRTHDATE	VARCHAR	0	0
COACH	INTEGER	0	0

- *name* keeps the record of name of the given technic member.
- *gender* keeps the record of gender of the given technic member.
- *nation* keeps the record of nation of the given technic member.
- *birthdate* keeps the record of birth date of the given technic member.
- *coach* references to *coach* table and on delete and update operations it cascades the operation

Sql statement that initialize the technic members table:

```
CREATE TABLE TECHNICMEMBERS (  
    ID SERIAL PRIMARY KEY,  
    NAME VARCHAR(45),  
    GENDER VARCHAR(6),  
    NATION VARCHAR(45),  
    BIRTHDATE VARCHAR(10),  
    COACH INTEGER REFERENCES COACHES ON DELETE CASCADE ON UPDATE_  
→ CASCADE  
)
```

## Code

### 1 MVC and team, player,tm classes

MVC pattern tried to use in the implementation of teams, players and technic members tables in web application. For all tables classes implemented. Instances of those classes keep the data of one tuple. Objects are implemented via html sending parameters and objects are sending to the functions of store classes and table classes' attributes implement the database tables via store classes' functions.

#### 1.1 class team

- Teams table class :

```
class team:  
    def __init__(self, nation, gender, foundDate, timesWon):  
        self.nation = nation  
        self.gender = gender  
        self.foundDate = foundDate  
        self.timesWon = timesWon
```

#### 1.2 class player

- Players table class :

```
class player:  
    def __init__(self, name, gender, nation, birthDate, team):  
        self.name = name  
        self.gender = gender  
        self.nation = nation  
        self.birthDate = birthDate  
        self.team = team
```

#### 1.3 class tm

- Technic members table class :

```
class tm:  
    def __init__(self, name, gender, nation, birthDate, coach):  
        self.name = name  
        self.gender = gender  
        self.nation = nation  
        self.birthDate = birthDate  
        self.coach = coach
```

## 2 Store classes

### 2.1 store.py

Store classes is implemented in store.py file. In store classes database is handling via some functions. Beginning of store.py is like this:

```
import psycopg2 as dbapi2

from technicmember import tm
from player import player
from team import team

from config import app
```

It imports psycopg2 editor as a dbapi2 for using as database api. Classes tm, player and team also imported. From config.py file it imports app object. In config.py file app object implemented in this way:

```
from flask import Flask

app = Flask(__name__)

app.debug = True
```

### 2.2 class StoreTeam

- class StoreTeam init function and createTable function is implemented like this:

```
class StoreTeam:
    def __init__(self, dbSettings):
        self.dsn = dbSettings

    def createTable(self, dsn):
        try:
            connection = dbapi2.connect(dsn)
            cursor = connection.cursor()
            statement = """ CREATE TABLE TEAMS (
                ID SERIAL PRIMARY KEY,
                NATION VARCHAR(45),
                GENDER VARCHAR(6),
                FOUNDDATE VARCHAR(20),
                TIMESWON VARCHAR(10)
            ) """
            cursor.execute(statement)
            connection.commit()
            cursor.close()
        except dbapi2.DatabaseError:
            connection.rollback()
        finally:
            connection.close()
```

createTable() function makes the connection with database via dbapi2 database api. cursor variable created as a cursor of connection and statement variable keeps the statement of SQL for creating table in database. After cursor execution and connection committing try, except and finally block handles the exceptions. If any error occurs connection rollback else connection closes.

all functions which needs to handle some operations on database uses the with .. as context manager of psycopg2

- addTeam() function of class StoreTeam:

```
def addTeam(self, team, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            cursor.execute("INSERT INTO TEAMS (NATION, GENDER, FOUNDDATE,
↪ TIMESWON) VALUES(%s, %s, %s, %s)", (team.nation, team.gender, team.
↪ foundDate, team.timesWon))
```

This function gets a team object from teams.py file html-side function. It adds the team object as a tuple into the database. It executes the SQL statement into the database.

- deleteTeam() function of class StoreTeam:

```
def deleteTeam(self, id, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """ DELETE FROM TEAMS WHERE ID = {}""".format(id)
            cursor.execute(query)
```

This function gets the id of the tuple to be deleted. It deletes the tuple from the database.

- updateTeam() function of class StoreTeam:

```
def updateTeam(self, team, id, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """UPDATE TEAMS SET NATION = '{}', GENDER = '{}',
↪ FOUNDDATE = '{}', TIMESWON = '{}' WHERE ID = {} """.format(team.
↪ nation, team.gender, team.foundDate, team.timesWon, id)
            cursor.execute(query)
```

This function gets the id of the tuple to be updated. It reaches the tuple with its' id and update the tuple with the team object which it gets.

- getAllTeams() function of class StoreTeam:

```
def getAllTeams(self, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """ SELECT * FROM TEAMS """
            cursor.execute(query)
            teams = cursor.fetchall()
            return teams
```

This function select all teams and return all teams as an array.

- selectTeams() function of class StoreTeam:

```
def selectTeams(self, team, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """SELECT * FROM TEAMS WHERE(NATION LIKE '{}%' )
↪ AND (GENDER LIKE '{}%' ) AND (FOUNDDATE LIKE '{}%' ) AND (TIMESWON
↪ LIKE '{}%' )""".format(team.nation, team.gender, team.foundDate,
↪ team.timesWon)
            cursor.execute(query)
            teams = cursor.fetchall()
            return teams
```

This function select teams with a specific search. It returns the team table tuples which it found as an array.

- createInitTeams() function of class StoreTeam:

```
def createInitTeams(self, dsn):
    app.storeT = StoreTeam(app.config['dsn'])
```

```

newTeam = team('Turkey', 'Male', '1920', '4')
app.storeT.addTeam(newTeam, dsn)
newTeam2 = team('England', 'Male', '1936', '3')
app.storeT.addTeam(newTeam2, dsn)
newTeam3 = team('China', 'Male', '1906', '5')
app.storeT.addTeam(newTeam3, dsn)
newTeam4 = team('Russia', 'Female', '1943', '1')
app.storeT.addTeam(newTeam4, dsn)

```

This function creates initial elements when database has initialized. It uses add function to create initial tuples.

## 2.2 class StoreP

- class StoreP init function and createTable() function:

```

def __init__(self, dbSettings):
    self.dsn = dbSettings

def createTable(self, dsn):
    try:
        connection = dbapi2.connect(dsn)
        cursor = connection.cursor()
        statement = """ CREATE TABLE PLAYERS (
            ID SERIAL PRIMARY KEY,
            NAME VARCHAR(45),
            GENDER VARCHAR(6),
            NATION VARCHAR(45),
            BIRTHDATE VARCHAR(10),
            TEAM INTEGER REFERENCES TEAMS ON DELETE CASCADE ON UPDATE CASCADE
        ) """
        cursor.execute(statement)
        connection.commit()
        cursor.close()
    except dbapi2.DatabaseError:
        connection.rollback()
    finally:
        connection.close()

```

createTable() function works like StoreTeam class' createTable() function.

Also in StoreP functions with .. as context manager of psycopg2 has used.

- addPlayer() function of class StoreP:

```

def addPlayer(self, player, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            cursor.execute("INSERT INTO PLAYERS (NAME, GENDER, NATION, ↵
↵BIRTHDATE, TEAM) VALUES(%s, %s, %s, %s, %s)", (player.name, player.
↵gender, player.nation, player.birthDate, player.team))

```

This function works as same as addTeam function of StoreTeam.

- deletePlayer() function of class StoreP:

```

def deletePlayer(self, id, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """ DELETE FROM PLAYERS WHERE ID = {}""".format(id)
            cursor.execute(query)

```

This function works as same as deleteTeam function of StoreTeam.

- updatePlayer() function of class StoreP:

```
def updatePlayer(self, player, id, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """UPDATE PLAYERS SET NAME = '{}', GENDER = '{}',
↪NATION = '{}', BIRTHDATE = '{}', TEAM = '{}' WHERE ID = {} """
            ↪format(player.name, player.gender, player.nation, player.birthDate,
            ↪player.team, id)
            cursor.execute(query)
```

This function also works as same as updateTeam function of StoreTeam.

- getAllPlayers() function of class StoreP:

```
def getAllPlayers (self, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """ SELECT PLAYERS.ID, PLAYERS.NAME, PLAYERS.GENDER,
↪PLAYERS.NATION, PLAYERS.BIRTHDATE, TEAMS.NATION FROM PLAYERS INNER
↪JOIN TEAMS ON TEAMS.ID = PLAYERS.TEAM """
            cursor.execute(query)
            players = cursor.fetchall()
            return players
```

This function also works as same as getAllTeams() function of StoreTeam.

- selectPlayers() function of class StoreP:

```
def selectPlayers(self, player, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """SELECT PLAYERS.ID, PLAYERS.NAME, PLAYERS.GENDER,
↪PLAYERS.NATION, PLAYERS.BIRTHDATE, TEAMS.NATION
            FROM PLAYERS INNER JOIN TEAMS ON TEAMS.ID = PLAYERS.TEAM
            WHERE(PLAYERS.NAME LIKE '{}' ) AND (PLAYERS.GENDER LIKE '{}'
↪%' ) AND
            (PLAYERS.NATION LIKE '{}' ) AND (PLAYERS.BIRTHDATE LIKE '{}'
↪' ) AND
            (TEAMS.NATION LIKE '{}' ) """
            ↪format(player.name, player.
            ↪gender, player.nation, player.birthDate, player.team)
            cursor.execute(query)
            players = cursor.fetchall()
            return players
```

This function also select players with a specific search. The SQL statement joins the teams and players tables and searches what to search in joined tables. After that it returns the players table tuples which it found as an array.

- createInitPlayers() function of class StoreP:

```
def createInitPlayers(self,dsn):
    app.store = StoreP(app.config['dsn'])

    newPlayer = player('Hasan', 'Male', 'Turkish', '1994', 1)
    app.store.addPlayer(newPlayer, dsn)
    newPlayer2 = player('Rose', 'Female', 'English', '1995', 2)
    app.store.addPlayer(newPlayer2, dsn)
    newPlayer3 = player('Dimitrov', 'Male', 'Russian', '1993', 4)
    app.store.addPlayer(newPlayer3, dsn)
```

This function creates initial elements when database has initialized. It uses add function to create initial tuples.



## 2.3 class StoreTM

- class StoreTM init function and createTable() function:

```
def __init__(self, dbSettings):
    self.dsn = dbSettings

def createTable(self, dsn):
    try:
        connection = dbapi2.connect(dsn)
        cursor = connection.cursor()
        statement = """ CREATE TABLE TECHNICMEMBERS (
            ID SERIAL PRIMARY KEY,
            NAME VARCHAR(45),
            GENDER VARCHAR(6),
            NATION VARCHAR(45),
            BIRTHDATE VARCHAR(10),
            COACH INTEGER REFERENCES COACHES ON DELETE CASCADE ON UPDATE_
↪CASCADE
        ) """
        cursor.execute(statement)
        connection.commit()
        cursor.close()
    except dbapi2.DatabaseError:
        connection.rollback()
    finally:
        connection.close()
```

createTable() function works like StoreTeam class' createTable() function.

Also in StoreTM functions with .. as context manager of psycopg2 has used.

- addTm() function of class StoreTM:

```
def addTm(self, tm, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            cursor.execute("INSERT INTO TECHNICMEMBERS (NAME, GENDER,
↪NATION, BIRTHDATE, COACH) VALUES(%s, %s, %s, %s, %s)", (tm.name, tm.
↪gender, tm.nation, tm.birthDate, tm.coach))
```

This function works as same as addTeam function of StoreTeam.

- deleteTm() function of class StoreTM:

```
def deleteTm(self, id, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """ DELETE FROM TECHNICMEMBERS WHERE ID = {} """
↪format(id)
            cursor.execute(query)
```

This function works as same as deleteTeam function of StoreTeam.

- updateTm() function of class StoreTM:

```
def updateTm(self, tm, id, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """UPDATE TECHNICMEMBERS SET NAME= '{}', GENDER = '{}
↪', NATION = '{}', BIRTHDATE = '{}', COACH = '{} WHERE ID = {} """
↪format(tm.name, tm.gender, tm.nation, tm.birthDate, tm.coach, id)
            cursor.execute(query)
```

This function also works as same as updateTeam function of StoreTeam.

- getAllTms() function of class StoreTM:

```
def getAllTms (self, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """ SELECT TECHNICMEMBERS.ID, TECHNICMEMBERS.NAME,
↪TECHNICMEMBERS.GENDER, TECHNICMEMBERS.NATION, TECHNICMEMBERS.
↪BIRTHDATE, COACHES.NAME FROM TECHNICMEMBERS INNER JOIN COACHES ON
↪COACHES.ID = TECHNICMEMBERS.COACH """
            cursor.execute(query)
            tms = cursor.fetchall()
            return tms
```

This function also works as same as getAllTeams() function of StoreTeam.

- selectTms() function of class StoreTM:

```
def selectTms(self, tm, dsn):
    with dbapi2.connect(dsn) as connection:
        with connection.cursor() as cursor:
            query = """SELECT TECHNICMEMBERS.ID, TECHNICMEMBERS.NAME,
↪TECHNICMEMBERS.GENDER, TECHNICMEMBERS.NATION, TECHNICMEMBERS.
↪BIRTHDATE, COACHES.NAME
            FROM TECHNICMEMBERS INNER JOIN COACHES ON COACHES.ID =
↪TECHNICMEMBERS.COACH
            WHERE (TECHNICMEMBERS.NAME LIKE '{ }%' ) AND (TECHNICMEMBERS.
↪GENDER LIKE '{ }%' ) AND
            (TECHNICMEMBERS.NATION LIKE '{ }%' ) AND (TECHNICMEMBERS.
↪BIRTHDATE LIKE '{ }%' ) AND
            (COACHES.NAME LIKE '{ }%' ) """ .format(tm.name, tm.gender, tm.
↪nation, tm.birthDate, tm.coach)
            cursor.execute(query)
            tms = cursor.fetchall()
            return tms
```

This function also select players with a specific search. The SQL statement joins the technicmembers and coaches tables and searches what to search in joined tables. After that it returns the technicmembers table tuples which it found as an array.

- createInitTMs() function of class StoreTM:

```
def createInitTMs(self, dsn):
    app.storeT = StoreTM(app.config['dsn'])

    newTm = tm('Veli', 'Male', 'Turkish', '1978', 1)
    app.storeT.addTm(newTm, dsn)
    newTm = tm('Ayşe', 'Female', 'Turkish', '1978', 1)
    app.storeT.addTm(newTm, dsn)
    newTm = tm('Jane', 'Female', 'English', '1982', 2)
    app.storeT.addTm(newTm, dsn)
```

This function creates initial elements when database has initialized. It uses add function to create initial tuples.

## 3 HTML handling .pys

### 3.1 teams.py

- Import part of the teams.py file

```

from flask import redirect
from flask import render_template
from flask import request
from flask import url_for

from config import app
from team import team
from store import StoreTeam
import psycopg2 as dbapi2

```

redirect, render\_template, request and url\_for features of Flask web framework have used. Like in store classes app object of Flask has imported from config. team class has imported. StoreTeam class has imported from store file. psycopg2 has imported as a dbapi2 for use as a database api.

- teams function for rendering teams.html file

```

@app.route('/teams', methods = ['GET', 'POST'])
def teams():

    dsn = app.config['dsn']

    app.store = StoreTeam(dsn)

    if request.method == 'GET':
        allTeams = app.store.getAllTeams(dsn)

```

This part of teams function renders the url '/teams' with teams.html file. Uses 'GET' and 'POST' methods. Uses dsn as a database settings which is implementing in config.py whether ElephantSQL or Vagrant database system. app.store variable assigned to a StoreTeam object. All StoreTeam object methods can be used via app.store variable. If request.method equals to 'GET' method getAllTeams function of StoreTeam object has invoked and gets all teams as an array and passed them to html file. After all operations allTeams variable assigned to an array which holds all the teams data. At the end of the function allTeams variable send to html file for listing.

- delete method

```

elif 'delete' in request.form:
    ids = request.form.getlist('teams')
    for id in ids:
        app.store.deleteTeam(id, dsn)
    allTeams = app.store.getAllTeams(dsn)

```

If request method from user is delete this part of teams function has invoked. It gets the clicked checkbox ids and passed that ids one by one to the deleteTeam() function of StoreTeam class. After that tuples get deleted.

- add method

```

elif 'add' in request.form:
    nation = request.form['nationToAdd']
    gender = request.form['genderToAdd']
    foundDate = request.form['foundDateToAdd']
    timesWon = request.form['timesWonToAdd']
    newTeam = team(nation, gender, foundDate, timesWon)
    app.store.addTeam(newTeam, dsn)
    allTeams = app.store.getAllTeams(dsn)

```

If request method from user is add this part of teams function has invoked. Datas in the add textboxes has passed as attributes of a team object and addTeam function of StoreTeam class added the tuple using team object attributes as a tuple attributes.

- update method

```
elif 'update' in request.form:
    ids = request.form.getlist('teams')
    id = ids[0]
    nation = request.form['nationToUpdate']
    gender = request.form['genderToUpdate']
    foundDate = request.form['foundDateToUpdate']
    timesWon = request.form['timesWonToUpdate']
    updatedTeam = team(nation, gender, foundDate, timesWon)
    app.store.updateTeam(updatedTeam, id, dsn)
    allTeams = app.store.getAllTeams(dsn)
```

If request method from user is update this part of teams function has invoked. Datas in update textboxes has passed as a attributes of a team object and id of the tuple that needs to be update. After that updateTeam function StoreTeam class updated the necessary tuple using team object attributes as a tuple attributes.

- find method

```
elif 'find' in request.form:
    nation = request.form['nationToFind']
    gender = request.form['genderToFind']
    foundDate = request.form['foundDateToFind']
    timesWon = request.form['timesWonToFind']
    findTeam = team(nation, gender, foundDate, timesWon)
    allTeams = app.store.selectTeams(findTeam, dsn)

return render_template('teams.html', teams = allTeams)
```

Finally if request method from user is find this part of teams function has invoked. Datas in find textboxes has passed as a attributes of a team object and selectTeams function of StoreTeam find the searched datas with SQL operation in the database and returns an array of teams that found.

At the end of function teams.html file rendered with this python code and allTeams attribute passed to html file to be listed.

## 3.2 players.py

- Import part of the players.py file

```
from flask import redirect
from flask import render_template
from flask import request
from flask import url_for

from config import app
from player import player
from store import StoreP
from store import StoreTeam
import psycpg2 as dbapi2
```

Like in teams.py file necessary imports for Flask web framework use has implemented. player class has imported. Both StoreP and StoreTeam classes has imported because of the team foreign key in the players table. After all operations allPlayers variable assigned to an array which holds allPlayers data as well. At the end of the function allPlayers variable send to html file for listing and also allTeams variable too.

- players function for rendering players.html file

```
@app.route('/players', methods = ['GET', 'POST'])
def players():

    dsn = app.config['dsn']
```

```

app.store = StoreP(dsn)

app.storeT = StoreTeam(dsn)
allTeams = app.storeT.getAllTeams(dsn)

if request.method == 'GET':
    allPlayers = app.store.getAllPlayers(dsn)

```

This part of players function works as same as teams function in teams.py file. With a little difference it has another app.storeT variable for using StoreTeam class for getting all teams for foreign key of players table team attribute.

- delete method

```

elif 'delete' in request.form:
    ids = request.form.getlist('players')
    for id in ids:
        app.store.deletePlayer(id, dsn)
    allPlayers = app.store.getAllPlayers(dsn)

```

This part of players function works as same as teams function of teams.py file.

- add method

```

elif 'add' in request.form:
    name = request.form['nameToAdd']
    gender = request.form['genderToAdd']
    nation = request.form['nationToAdd']
    birthDate = request.form['birthDateToAdd']
    team = request.form['teamToAdd']
    newPlayer = player(name, gender, nation, birthDate, team)
    app.store.addPlayer(newPlayer, dsn)
    allPlayers = app.store.getAllPlayers(dsn)

```

This part of players function also works as same as teams function of teams.py file.

- update method

```

elif 'update' in request.form:
    ids = request.form.getlist('players')
    id = ids[0]
    name = request.form['nameToUpdate']
    gender = request.form['genderToUpdate']
    nation = request.form['nationToUpdate']
    birthDate = request.form['birthDateToUpdate']
    team = request.form['teamToUpdate']
    updatedPlayer = player(name, gender, nation, birthDate, team)
    app.store.updatePlayer(updatedPlayer, id, dsn)
    allPlayers = app.store.getAllPlayers(dsn)

```

Also this part of players function works as same as teams function in teams.py

- find method

```

elif 'find' in request.form:
    name = request.form['nameToFind']
    gender = request.form['genderToFind']
    nation = request.form['nationToFind']
    birthDate = request.form['birthDateToFind']
    team = request.form['teamToFind']
    findPlayer = player(name, gender, nation, birthDate, team)
    allPlayers = app.store.selectPlayers(findPlayer, dsn)

```

```
return render_template('players.html', players = allPlayers, teams =  
↪allTeams )
```

Final part is also same as teams function in teams.py file.

At the end of function players.html file rendered with this python code and allPlayers and allTeams attribute passed to html file to be listed.

### 3.3 technicmembers.py

- Import part of the technicmembers.py file

```
from flask import redirect  
from flask import render_template  
from flask import request  
from flask import url_for  
  
from config import app  
from technicmember import tm  
from store import StoreTM  
import psycopg2 as dbapi2  
  
import coaches
```

Like in teams.py file necessary imports for Flask web framework use has implemented. tm class has imported. StoreTM class has imported and also coaches has imported because of the coach foreign key in the technicmembers table. After all operations allTms variable assigned to an array which holds allTms data as well. At the end of the function allTms variable send to html file for listing and also allCoaches variable too.

- technicmembers function for rendering technicMembers.html file

```
@app.route('/technicMembers', methods = ['GET', 'POST'])  
def technicMembers():  
  
    dsn = app.config['dsn']  
  
    app.store = StoreTM(dsn)  
  
    allCoaches = coaches.get_coaches()  
  
    if request.method == 'GET':  
        allTms = app.store.getAllTms(dsn)
```

This part of technicmembers function works as same as teams function in teams.py file. With a little difference it has allCoaches variable for using getting all coaches from coaches file with coaches.get\_coaches function for foreign key of technicmembers table coach attribute.

- delete method

```
elif 'delete' in request.form:  
    ids = request.form.getlist('tms')  
    for id in ids:  
        app.store.deleteTm(id, dsn)  
    allTms = app.store.getAllTms(dsn)
```

This part of technicmembers function works as same as teams function of teams.py file.

- add method

```
elif 'add' in request.form:  
    name = request.form['nameToAdd']
```

```

gender = request.form['genderToAdd']
nation = request.form['nationToAdd']
birthDate = request.form['birthDateToAdd']
coach = request.form['coachToAdd']
newTm = tm(name, gender, nation, birthDate, coach)
app.store.addTm(newTm, dsn)
allTms = app.store.getAllTms(dsn)

```

This part of technicmembers function also works as same as teams function of teams.py file.

- update method

```

elif 'update' in request.form:
    ids = request.form.getlist('tms')
    id = ids[0]
    name = request.form['nameToUpdate']
    gender = request.form['genderToUpdate']
    nation = request.form['nationToUpdate']
    birthDate = request.form['birthDateToUpdate']
    coach = request.form['coachToUpdate']
    newTm = tm(name, gender, nation, birthDate, coach)
    app.store.updateTm(newTm, id, dsn)
    allTms = app.store.getAllTms(dsn)

```

Also this part of technicmembers function works as same as teams function in teams.py

- find method

```

elif 'find' in request.form:
    name = request.form['nameToFind']
    gender = request.form['genderToFind']
    nation = request.form['nationToFind']
    birthDate = request.form['birthDateToFind']
    coach = request.form['coachToFind']
    findTm = tm(name, gender, nation, birthDate, coach)
    allTms = app.store.selectTms(findTm, dsn)

return render_template('technicMembers.html', tms = allTms, coaches = ↵
↵allCoaches )

```

Final part is also same as teams function in teams.py file.

At the end of function technicMembers.html file rendered with this python code and allTms and allCoaches attribute passed to html file to be listed.

## Parts Implemented by Alican MERTAN

### Database Design

#### 1 Tables

##### 1.1 Tournaments Table

- Tournaments table keeping records of the tournaments data

Name	Type
ID	INTEGER
NAME	VARCHAR
YEAR	VARCHAR
WINNER	INTEGER
BEST_PLAYER	INTEGER

- *name* keeps the record of name of the given tournament.
- *year* keeps the record of year of the given tournament.
- *winner* keeps the record of the winner of the given tournament. It references the *teams* table.
- *best\_player* keeps the record of the best player of the given tournament. It references the *players* table.

Sql statement that initialize the tournaments table:

```
CREATE TABLE TOURNAMENTS (  
  ID SERIAL PRIMARY KEY,  
  NAME VARCHAR(45),  
  YEAR VARCHAR(4),  
  WINNER INTEGER REFERENCES TEAMS ON DELETE CASCADE ON UPDATE CASCADE,  
  BEST_PLAYER INTEGER REFERENCES PLAYERS ON DELETE CASCADE ON UPDATE   
→CASCADE  
)
```

## 1.2 Matches Table

- Matches table keeping records of the matches data

Name	Type
ID	INTEGER
TOURNAMENT	INTEGER
TEAM1	INTEGER
TEAM2	INTEGER
SCORE	VARCHAR

- *tournament* keeps the record of name of the tournament for given match. It references the *tournaments* table.
- *team1* keeps the record of name of the team for given match. It references the *teams* table.
- *team2* keeps the record of name of the team for given match. It references the *teams* table.
- *score* keeps the record of the score of the given match.

Sql statement that initialize the matches table:

```
CREATE TABLE MATCHES (  
  ID SERIAL PRIMARY KEY,  
  TOURNAMENT INTEGER REFERENCES TOURNAMENTS ON DELETE CASCADE ON UPDATE   
→CASCADE,  
  TEAM1 INTEGER REFERENCES TEAMS ON DELETE CASCADE ON UPDATE CASCADE,  
  TEAM2 INTEGER REFERENCES TEAMS ON DELETE CASCADE ON UPDATE CASCADE,  
  SCORE VARCHAR(3)  
)
```

## Code

### 1 Functions



## 1.1 creating tables

create\_table functions used in order to create tables.

- create\_table function in tournaments.py:

```
def create_table():
    try:
        cursor = create_connection()
        statement = """ CREATE TABLE TOURNAMENTS(
            ID SERIAL PRIMARY KEY,
            NAME VARCHAR(45),
            YEAR VARCHAR(4),
            WINNER INTEGER REFERENCES TEAMS ON DELETE CASCADE ON UPDATE_
↪CASCADE,
            BEST_PLAYER INTEGER REFERENCES PLAYERS ON DELETE CASCADE ON_
↪UPDATE CASCADE
        ) """
        cursor.execute(statement)
        cursor.connection.commit()
        close_connection(cursor)
    except psycopg2.DatabaseError:
        cursor.connection.rollback()
    finally:
        cursor.connection.close()
```

- create\_table function in matches.py:

```
def create_table():
    try:
        cursor = create_connection()
        statement = """ CREATE TABLE MATCHES(
            ID SERIAL PRIMARY KEY,
            TOURNAMENT INTEGER REFERENCES TOURNAMENTS ON DELETE CASCADE ON_
↪UPDATE CASCADE,
            TEAM1 INTEGER REFERENCES TEAMS ON DELETE CASCADE ON UPDATE_
↪CASCADE,
            TEAM2 INTEGER REFERENCES TEAMS ON DELETE CASCADE ON UPDATE_
↪CASCADE,
            SCORE VARCHAR(3)
        ) """
        cursor.execute(statement)
        cursor.connection.commit()
        close_connection(cursor)
    except psycopg2.DatabaseError:
        cursor.connection.rollback()
    finally:
        cursor.connection.close()
```

## 1.2 initilazing database

create\_init functions used in order to initilaze database with some tuples.

- create\_init\_tournaments function in tournaments.py:

```
def create_init_tournaments():
    add_new_tournament('World Cup', '2015', 1, 1)
    add_new_tournament('World Cup', '2014', 2, 3)
    add_new_tournament('World Cup', '2013', 3, 2)
```

- create\_init\_matches function in matches.py:

```
def create_init_matches():  
    add_new_match(1, 1, 2, '5-3')  
    add_new_match(1, 3, 4, '4-2')  
    add_new_match(1, 3, 2, '2-6')
```

### 1.3 adding new tuples

add\_new functions used in order to add new tuples to a table. Function gets attribute values as a parameter.

- add\_new\_tournament function in tournaments.py:

```
def add_new_tournament(name, year, winner, best_player):  
    cursor = create_connection()  
  
    cursor.execute("INSERT INTO tournaments (name, year, winner, best_  
→player) VALUES (%s, %s, %s, %s)", (name, year, winner, best_player))  
    cursor.connection.commit()  
  
    close_connection(cursor)  
  
    return True
```

- add\_new\_match function in matches.py:

```
def add_new_match(tournament, team1, team2, score):  
    cursor = create_connection()  
  
    cursor.execute("INSERT INTO matches (tournament, team1, team2,   
→score) VALUES (%s, %s, %s, %s)", (tournament, team1, team2, score))  
    cursor.connection.commit()  
  
    close_connection(cursor)  
  
    return True
```

### 1.4 deleting tuples

delete functions used in order to delete tuples. Function takes primary key value as a parameter.

- delete\_tournament function in tournaments.py:

```
def delete_tournament(id):  
    cursor = create_connection()  
    statement = "DELETE FROM TOURNAMENTS WHERE ID={}".format(id)  
    cursor.execute(statement)  
    cursor.connection.commit()  
  
    close_connection(cursor)
```

- delete\_match function in matches.py:

```
def delete_match(id):  
    cursor = create_connection()  
    statement = "DELETE FROM MATCHES WHERE ID={}".format(id)  
    cursor.execute(statement)  
    cursor.connection.commit()  
  
    close_connection(cursor)
```

## 1.5 updating tuples

**update functions used in order to update selected tuples. Function takes primary key as a parameter to find selected tuple and takes attributes values as a paramater to update tuple.**

- update\_tournament function in tournaments.py:

```
def update_tournament(id, nameUpdate, yearUpdate, winnerUpdate, best_playerUpdate):
    cursor = create_connection()
    statement = """UPDATE TOURNAMENTS SET NAME = '{}', YEAR = '{}', WINNER = '{}',
↪BEST_PLAYER = {} WHERE ID={} """.format(nameUpdate, yearUpdate, winnerUpdate,
↪best_playerUpdate, id)
    cursor.execute(statement)
    cursor.connection.commit()

    close_connection(cursor)
```

- update\_match function in matches.py:

```
def update_match(id, tournamentUpdate, team1Update, team2Update, scoreUpdate):
    cursor = create_connection()
    statement = """UPDATE MATCHES SET TOURNAMENT = '{}', TEAM1 = '{}', TEAM2 = '{}',
↪SCORE = '{}' WHERE ID={} """.format(tournamentUpdate, team1Update, team2Update,
↪scoreUpdate, id)
    cursor.execute(statement)
    cursor.connection.commit()

    close_connection(cursor)
```

## 1.6 finding tuples

**findInJointTables functions used in order to query tuples. Function takes attribute values as a parameter and returns tuples as an array. If an empty search made, all the tuples will be returned.**

- findInJointTables function in tournaments.py:

```
def findInJointTables(nameFind, yearFind, winnerFind, best_playerFind):
    statement= """ SELECT TOURNAMENTS.ID, TOURNAMENTS.NAME, YEAR, TEAMS.NATION ,
↪PLAYERS.NAME FROM TOURNAMENTS INNER JOIN PLAYERS ON PLAYERS.ID=TOURNAMENTS.BEST_
↪PLAYER INNER JOIN TEAMS ON TEAMS.ID=TOURNAMENTS.WINNER WHERE (TOURNAMENTS.NAME
↪LIKE '{}%' ) AND (YEAR LIKE '{}%' ) AND (TEAMS.NATION LIKE '{}%' ) AND
↪(PLAYERS.NAME LIKE '{}%' )""".format(nameFind, yearFind, winnerFind, best_
↪playerFind)

    cursor = create_connection()
    cursor.execute(statement)
    tournaments = cursor.fetchall()
    cursor.connection.commit()

    close_connection(cursor)

    return tournaments
```

- findInJointTables function in matches.py:

```
def findInJointTables(tournamentFind, team1Find, team2Find, scoreFind):
    statement= """ SELECT MATCHES.ID, TOURNAMENTS.NAME, t1.NATION, t2.NATION, MATCHES.
↪SCORE FROM MATCHES INNER JOIN TOURNAMENTS ON TOURNAMENTS.ID=MATCHES.TOURNAMENT_
↪INNER JOIN TEAMS t1 ON t1.ID=MATCHES.TEAM1 INNER JOIN TEAMS t2 ON t2.ID=MATCHES.
↪TEAM2 WHERE (TOURNAMENTS.NAME LIKE '{}%' ) AND (t1.NATION LIKE '{}%' ) AND (t2.
↪NATION LIKE '{}%' ) AND (MATCHES.SCORE LIKE '{}%' )""".format(tournamentFind,
↪team1Find, team2Find, scoreFind)
```

```
cursor = create_connection()
cursor.execute(statement)
matches = cursor.fetchall()
cursor.connection.commit()

close_connection(cursor)

return matches
```

## 1.7 fetching all tuples

showJointTables functions used in order to fetch all the tuples. Function returns tuples as an array.

- showJointTables function in tournaments.py:

```
def showJointTables():
    cursor = create_connection()
    statement= """ SELECT TOURNAMENTS.ID, TOURNAMENTS.NAME, YEAR, TEAMS.NATION ,
↳PLAYERS.NAME FROM TOURNAMENTS INNER JOIN PLAYERS ON PLAYERS.ID=TOURNAMENTS.BEST_
↳PLAYER INNER JOIN TEAMS ON TEAMS.ID=TOURNAMENTS.WINNER """
    cursor.execute(statement)
    tournaments = cursor.fetchall()
    cursor.connection.commit()

    close_connection(cursor)
    return tournaments
```

- showJointTables function in matches.py:

```
def showJointTables():
    cursor = create_connection()
    statement= """ SELECT MATCHES.ID, TOURNAMENTS.NAME, t1.NATION, t2.NATION, MATCHES.
↳SCORE FROM MATCHES INNER JOIN TOURNAMENTS ON TOURNAMENTS.ID=MATCHES.TOURNAMENT_
↳INNER JOIN TEAMS t1 ON t1.ID = MATCHES.TEAM1 INNER JOIN TEAMS t2 ON t2.
↳ID=MATCHES.TEAM2 """
    cursor.execute(statement)
    matches = cursor.fetchall()
    cursor.connection.commit()

    close_connection(cursor)
    return matches
```

## 2 HTML handling

tournaments() and matches() functions used in order to handle HTML related works.

- tournaments function in tournaments.py:

```
@app.route("/tournaments/", methods=['GET', 'POST'])
def tournaments():

    dsn = app.config['dsn']

    app.storeT = StoreTeam(dsn)
    allTeams = app.storeT.getAllTeams(dsn)

    app.store = StoreP(dsn)
    allPlayers = app.store.getAllPlayers(dsn)
```

```

if request.method == 'GET':

    all_tournaments = showJointTables()
    queriedTournaments = findInJointTables('?', '?', '?', '?')

```

- matches function in matches.py:

```

@app.route("/matches", methods=['GET', 'POST'])
def matches():

    allTournaments = tournaments.get_tournaments()

    dsn = app.config['dsn']

    app.storeT = StoreTeam(dsn)
    allTeams = app.storeT.getAllTeams(dsn)

    if request.method == 'GET':
        all_matches = showJointTables()
        queriedMatches = findInJointTables('?', '?', '?', '?')

```

## 2.1 add block

In the add block, add\_new functions called with the parameters from HTML.

- add block in tournaments.py:

```

elif 'add' in request.form:
    # -----
    name = request.form['name']
    year = request.form['year']
    winner = request.form['winner']
    best_player = request.form['best_player']
    # -----

    add_new_tournament(name, year, winner, best_player) # save to db

    all_tournaments = showJointTables()
    queriedTournaments = findInJointTables('?', '?', '?', '?')

```

- add block in matches.py:

```

elif 'add' in request.form:
    # -----
    tournament = request.form['tournament']
    team1 = request.form['team1']
    team2 = request.form['team2']
    score = request.form['score']
    # -----

    add_new_match(tournament, team1, team2, score) # save to db

    all_matches = showJointTables() # get all matches
    queriedMatches = findInJointTables('?', '?', '?', '?')

```

### 2.2 delete block

In the delete block, delete functions called with the parameters from HTML.

- delete block in tournaments.py:

```
elif 'delete' in request.form:
    ids = request.form.getlist('tournaments_to_delete')
    for id in ids:
        delete_tournament(id)
    all_tournaments = showJointTables()
    queriedTournaments = findInJointTables('?', '?', '?', '?')
```

- delete block in matches.py:

```
elif 'delete' in request.form:
    ids = request.form.getlist('matches_to_delete')
    for id in ids:
        delete_match(id)
    all_matches = showJointTables()
    queriedMatches = findInJointTables('?', '?', '?', '?')
```

### 2.3 find block

In the find block, findInJointTables functions called with the parameters from HTML.

- find block in tournaments.py:

```
elif 'find' in request.form:
    nameFind = request.form['nameFind']
    yearFind = request.form['yearFind']
    winnerFind = request.form['winnerFind']
    best_playerFind = request.form['best_playerFind']

    all_tournaments = showJointTables()
    queriedTournaments = findInJointTables(nameFind, yearFind, winnerFind, best_
↪playerFind)
```

- find block in matches.py:

```
elif 'find' in request.form:
    tournamentFind = request.form['tournamentFind']
    team1Find = request.form['team1Find']
    team2Find = request.form['team2Find']
    scoreFind = request.form['scoreFind']

    all_matches = showJointTables()
    queriedMatches = findInJointTables(tournamentFind, team1Find, team2Find, ↪
↪scoreFind)
```

### 2.4 update block

In the update block, update functions called with the parameters from HTML.

- update block in tournaments.py:

```
elif 'update' in request.form:
    ids = request.form.getlist('update')
    for id in ids:
        nameUpdate = request.form['nameUpdate'+id]
```

```

        yearUpdate = request.form['yearUpdate'+id]
        winnerUpdate = request.form['winnerUpdate'+id]
        best_playerUpdate = request.form['best_playerUpdate'+id]
        update_tournament(id, nameUpdate, yearUpdate, winnerUpdate, best_
↪playerUpdate)

    all_tournaments = showJointTables()
    queriedTournaments = findInJointTables('?', '?', '?', '?')

```

- update block in matches.py:

```

elif 'update' in request.form:
    ids = request.form.getlist('update')
    for id in ids:
        tournamentUpdate = request.form['tournamentUpdate'+id]
        team1Update = request.form['team1Update'+id]
        team2Update = request.form['team2Update'+id]
        scoreUpdate = request.form['scoreUpdate'+id]
        update_match(id, tournamentUpdate, team1Update, team2Update, scoreUpdate)

```

## Parts Implemented by Ahmet Yilmaz

### Database Design

#### 1 Tables

##### 1.1 Coaches Table

- Coaches table keeping record of the coaches data

Name	Type	Not Null	Primary K.
ID	INTEGER	0	1
NAME	VARCHAR	0	0
GENDER	VARCHAR	0	0
NATIONALITY	VARCHAR	0	0
BIRTH_DATE	VARCHAR	0	0
CURRENT_TEAM	INTEGER	0	0

- *name* keeps the record of name of the given coach.
- *gender* keeps the record of gender of the given coach.
- *nationality* keeps the record of the nationality of the given coach.
- *birth\_date* keeps the record of birth date of the given coach.
- *current\_team* keeps the record of current team of the given coach which refers to teams table.

Sql statement that initialize the coaches table:

```

CREATE TABLE COACHES (
    ID SERIAL PRIMARY KEY,
    NAME VARCHAR(45),
    GENDER VARCHAR(6),
    NATIONALITY VARCHAR(45),
    BIRTH_DATE VARCHAR(10),
    CURRENT_TEAM INTEGER REFERENCES TEAMS ON DELETE CASCADE ON UPDATE_
↪CASCADE
)

```

## 1.2 Player Statistics Table

- Player Statistics table keeping record of the statistics data of players.

Name	Type	Not Null	Primary K.
ID	INTEGER	0	1
PLAYER	INTEGER	0	0
MATCHES_PLAYED	VARCHAR	0	0
MATCHES_WON	VARCHAR	0	0
WIN_RATE	VARCHAR	0	0
AVERAGE_SCORE	VARCHAR	0	0

- *player* keeps the record of name of the given player refers to players table on delete and on update cascades
- *matches\_played* keeps the count of played matches of given player.
- *matches\_won* keeps the count of won matches of given player.
- *win\_rate* keeps the record of win rate of the given player.
- *average\_score* keeps the record of average score of the given player according to matches he/she played.

**Sql statement that initialize the playerstatistics table:**

```
CREATE TABLE PLAYERSTATISTICS (  
  ID SERIAL PRIMARY KEY,  
  matches_played VARCHAR(10),  
  matches_won VARCHAR(10),  
  win_rate VARCHAR(10),  
  average_score VARCHAR(10),  
  player INTEGER REFERENCES players ON DELETE CASCADE ON UPDATE CASCADE  
)
```

## 1.3 Users Table

- Users table keeping record of the users data

Name	Type	Not Null	Primary K.
ID	INTEGER	0	1
USERNAME	VARCHAR	0	0
PASSWORD	VARCHAR	0	0

- *username* keeps the record of name of the given user.
- *password* keeps the record of gender of the given user.

**Sql statement that initialize the technic members table:**

```
CREATE TABLE USERS (  
  ID SERIAL PRIMARY KEY,  
  USERNAME VARCHAR(45),  
  PASSWORD VARCHAR(6)  
)
```

## Code

### 1 Python Flask Extension Parts

#### coaches.py

- Import part of the coaches.py file



```

from flask import request
from flask import render_template
from config import app
from config import create_connection, close_connection
import psycopg2
import teams
from store import StoreTeam

```

render\_template and request features of Flask web framework have used. app object of Flask has imported from config. team class and StoreTeam class has imported. Since “current\_team” attribute refers to teams table psycopg2 has imported as a dbapi2 for use as a database api.

- route function of coaches for rendering coaches.html file

```

@app.route("/coaches/", methods=['GET', 'POST'])
def coaches():
    dsn = app.config['dsn']

    app.store = StoreTeam(dsn)
    all_teams = app.store.getAllTeams(dsn)

    if request.method == 'GET':
        all_coaches = join_tables()

```

This part of coaches function renders the url ‘/teams’ with coaches.html file. Uses ‘GET’ and ‘POST’ methods. Uses dsn as a database settings which is implementing in config.py whether ElephantSQL or Vagrant database system. To get all teams for joint tables first getAllTeams function used and teams assigned to all\_teams variable. after that join\_tables function used to join teams and coaches tables to get meaningful data and assigned to all\_coaches variable.

- delete method

```

elif 'delete' in request.form:
    ids = request.form.getlist('coaches_to_delete')
    for id in ids:
        delete_coach(id)

    all_coaches = join_tables()

```

If request method is delete which means if delete button has clicked on html page. Code requests checked box information and assign this to a list called ids. Then each element in ids list sent to delete\_coach function which deletes tuple from table. Then all\_coaches variable renewed with remaining tuples.

- add method

```

elif 'add' in request.form:
    # -----
    name = request.form['name']
    gender = request.form['gender']
    nationality = request.form['nationality']
    birth_date = request.form['birth_date']
    current_team = request.form['current_team']
    # -----

    add_new_coach(name, gender, nationality, birth_date, current_team) # ↵
    ↪ save to db

    all_coaches = join_tables()

```

If request method is add which means add button has clicked on html page. Code requests the values entered in textboxes. These values sent to add\_new\_coach function which adds the tuple with according values to the table. New all tuples fetched to print to the screen

- update method

```
elif 'update' in request.form:
    ids = request.form.getlist('update')
    for id in ids:
        name_update = request.form['name_update'+id]
        gender_update = request.form['gender_update'+id]
        nationality_update = request.form['nationality_update'+id]
        birth_date_update = request.form['birth_date_update'+id]
        current_team_update = request.form['current_team_update'+id]

        update_coach(id, name_update, gender_update, nationality_update,
        ↪ birth_date_update, current_team_update)

    all_coaches = join_tables()
```

If request method is update which means if update button clicked on html page. Code requests the values entered in textboxes and gets the ids with checkboxes. for each id tuples changed with new values. New all tuples fetched to print to the screen.

- find method

```
elif 'find' in request.form:
    para_1 = request.form['name_find']
    para_2 = request.form['gender_find']
    para_3 = request.form['nationality_find']
    para_4 = request.form['birth_date_find']
    para_5 = request.form['current_team_find']

    all_coaches = find_coach(para_1, para_2, para_3, para_4, para_5)
```

Else if request method is find which means find button clicked on html page. Code requests the values from textboxes for attributes. values sent to find\_coach function to select according tuples from table and print.

- showall method

```
elif 'showall' in request.form:

    all_coaches = join_tables()
```

Else if request method is showall which means Show All button has clicked on html page. Coaches table and teams table joint to show all coaches tables tuples with according curen\_team value.

```
return render_template("coaches.html", coaches=all_coaches, teams_
    ↪ select=all_teams)
```

Route function returns render\_template function which gets all coaches and send them to html page to print to the screen.

## playerstatistics.py

Same things did as did in coaches.py But playerstatistics table has a different foreign key which refers to players table

## users.py

Same things did as did in coaches.py

## 2 Python PostgreSql Parts

### coaches.py

- get\_coaches function

```
def get_coaches():
    cursor = create_connection()

    cursor.execute("SELECT * FROM coaches;")
    coaches = cursor.fetchall()

    close_connection(cursor)

    return coaches
```

This function selects all tuples from table without condition.

- create\_init\_coaches function

```
def create_init_coaches():

    add_new_coach('Zehra', 'female', 'turkish', '1964', 1)
    add_new_coach('Mike', 'male', 'english', '1954', 2)
    add_new_coach('Chan', 'male', 'chinese', '1962', 3)
```

This function adds 3 initial tuple when initialize database function called using add\_new\_coach function.

- update\_coach function

```
def update_coach(id, name_update, gender_update, nationality_update,
    ↪ birth_date_update, current_team_update):
    cursor = create_connection()
    statement = """UPDATE COACHES SET NAME = '{}', GENDER = '{}',
    ↪ NATIONALITY = '{}', BIRTH_DATE = '{}', CURRENT_TEAM = '{}' WHERE ID_
    ↪ = {}""".format(name_update, gender_update, nationality_update, birth_
    ↪ date_update, current_team_update, id)
    cursor.execute(statement)
    cursor.connection.commit()

    close_connection(cursor)
```

This function updates tuples with values coming from html page.

- find\_coach function

```
def find_coach(name_find, gender_find, nationality_find, birth_date_
    ↪ find, current_team_find):
    statement= """ SELECT COACHES.ID, COACHES.NAME, COACHES.GENDER,
    ↪ NATIONALITY, BIRTH_DATE, TEAMS.NATION FROM COACHES INNER JOIN TEAMS_
    ↪ ON TEAMS.ID=COACHES.CURRENT_TEAM WHERE (COACHES.NAME LIKE '{}%' )
    ↪ AND (COACHES.GENDER LIKE '{}%' ) AND (NATIONALITY LIKE '{}%' ) AND
    ↪ (BIRTH_DATE LIKE '{}%' ) AND (TEAMS.NATION LIKE '{}%' )"""
    ↪ format(name_find, gender_find, nationality_find, birth_date_find,
    ↪ current_team_find)

    cursor = create_connection()
    cursor.execute(statement)
    coaches = cursor.fetchall()
    cursor.connection.commit()

    close_connection(cursor)
```

```
return coaches
```

This function finds tuples with values coming from html page.

- add\_new\_coach function

```
def add_new_coach(name, gender, nationality, birth_date, current_team):
    cursor = create_connection()

    cursor.execute("INSERT INTO coaches (name, gender, nationality,
↪birth_date, current_team) VALUES (%s, %s, %s, %s, %s)", (name,
↪gender, nationality, birth_date, current_team))
    cursor.connection.commit()

    close_connection(cursor)

    return True
```

This function adds a new tuple to table with given values from html page.

- delete\_coach function

```
def delete_coach(id):
    cursor = create_connection()
    statement = "DELETE FROM COACHES WHERE ID={}".format(id)
    cursor.execute(statement)
    cursor.connection.commit()

    close_connection(cursor)
```

This function deletes the tuples from table which selected with checkboxes from html page.

- join\_tables function

```
def join_tables():
    cursor = create_connection()
    statement= "SELECT COACHES.ID, COACHES.NAME, COACHES.GENDER,
↪NATIONALITY, BIRTH_DATE, TEAMS.NATION FROM COACHES INNER JOIN TEAMS_
↪ON TEAMS.ID=COACHES.CURRENT_TEAM"
    cursor.execute(statement)
    coaches = cursor.fetchall()
    cursor.connection.commit()

    close_connection(cursor)
    return coaches
```

This function joins coaches and teams table for current\_team value of coaches table can be printed with the vale it refers.

### playerstatistics.py

Same things did as did in coaches.py But playerstatistics table has a different foreign key which refers to players table

### users.py

Same things did as did in coaches.py

## Parts Implemented by Firat Bayram