# ABC pvt. Ltd.'s Sports Festival Documentation

## Table of Contents

# Task - Sports Tournament Management System

**ABC private limited is about to conduct a sports festival and you are asked to design a web app which manages the dataset efficiently.**

**Sports festivals have multiple tournaments.Tournaments are identified by name, sport, number of teams/persons allowed, and event type (Single/ Team).Tournaments can have registration based on the event. Team are identified by name and number of people allowed. People are identified by their name, age, address, blood group etc.**

Conditions to be met,

1. **The above mentioned identifiers should be unique and required in order to perform a create operation**
2. **Participants should register in the app before they proceed to tournament registration. If you have already registered, proceed to the login page (You can have a static Admin data)**
3. **Only Admin can add the tournament, modify the tournament and can update the status of the team/ participants as win/lose/disqualified.**
4. **Admin should be able to view the report of data for each tournament from his home page such as number of teams/participants registered, Tournament registration start and end time and Teams disqualified**
5. **Admin should be able to,**
    a. **Cancel the tournament based on the number of participants**
    b. **Disqualify a team if they don't have the required number of team members during the event.**
6. **Participants should be allowed to register or update only during a given period**
7. **Participants will be allowed to register for multiple tournaments but not for multiple teams**
8. **Tournaments need to be held one at a time and can only be started if the required number of teams or persons registered.**
9. **The participants' home page should show the list of all available tournaments and registration link**
10. **The Schedule for the tournament should be visible for the participants from their homepage if they have already registered for the tournament.**

# Sports Tournament Major Implementations

## KEY POINTS :

1.During Login based on Role --eg : admin / user we will redirect to their respective home pages.

2.Admin will have his own previlages like he can add , update , delete tournament ,also he can view list of users ,teams and current status of tournamnets.

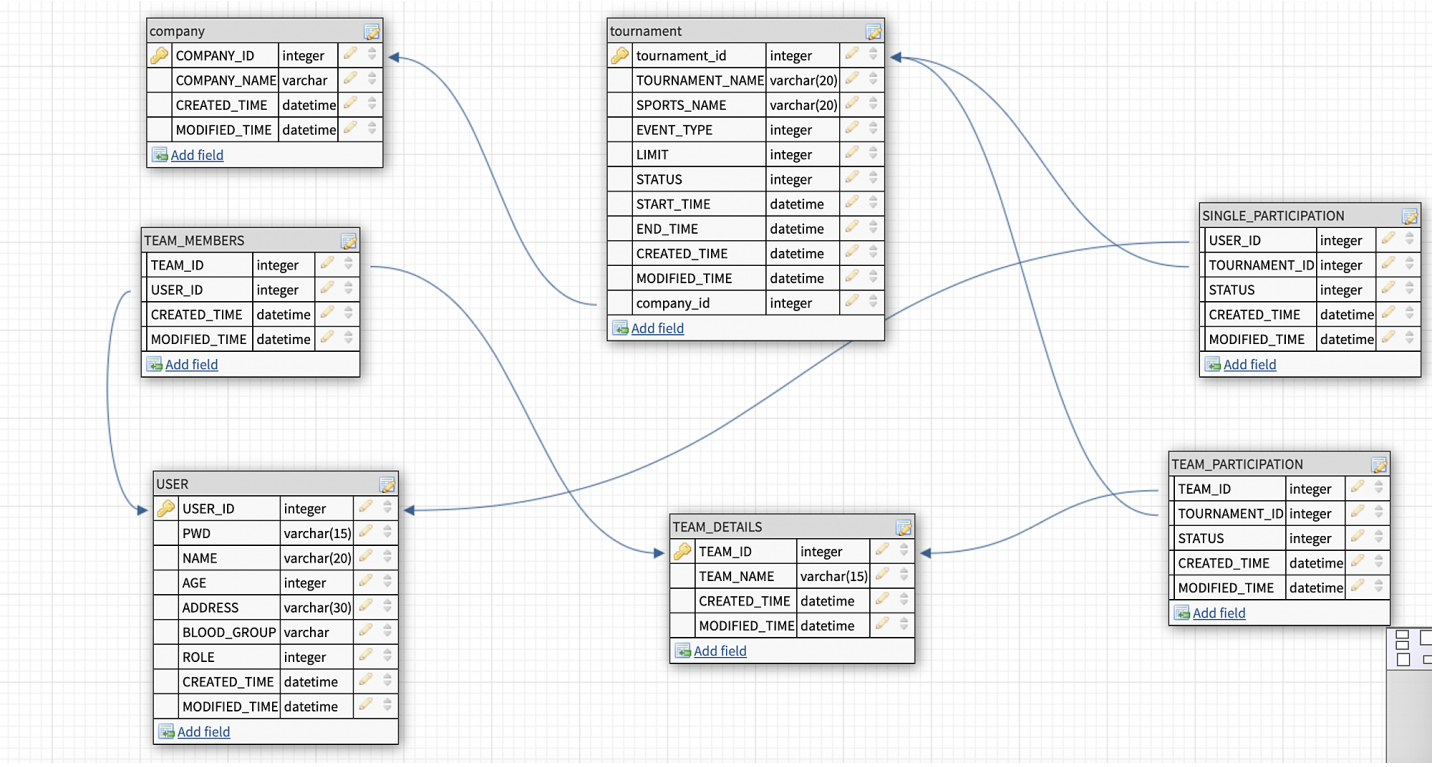
3.User can view list of tournaments and he will given access to register any of the tournaments  based on his willing, he can also join in  any team of team participation of his willing.

3.User can also create new team and he will be directly joined in the respective team and tournament based on his details.

4.User can also edit his basic details from his menu bar like username,age, address and other specifications.

# DATA MODEL DIAGRAM:

# Sports Tournament DB Schema :



# JDBC(Java Database Connectivity)

## LIST OF TABLES CREATED:

```
[root=# \c root
 You are now connected to database "root" as user "aravind-pt6089".
[root=# \d
                        List of relations
 Schema |                 Name                 |   Type   |  Owner
--------+--------------------------------------+----------+----------
 public | single_participation                 | table    | postgres
 public | single_participation_single_id_seq   | sequence | postgres
 public | team_details                         | table    | postgres
 public | team_details_team_id_seq             | sequence | postgres
 public | team_members                         | table    | postgres
 public | team_members_team_memberid_seq       | sequence | postgres
 public | team_participation                   | table    | postgres
 public | team_participation_teampart_id_seq   | sequence | postgres
 public | tournament                           | table    | postgres
 public | tournament_tournament_id_seq         | sequence | postgres
 public | users                                | table    | postgres
(11 rows)
```

**users table :**

```
 user_id | pwd  |  name   | age | address | role | blood_group | created_time | modified_time
---------+------+---------+-----+---------+------+-------------+--------------+---------------
       1 | abc  | aravind |  22 | ponneri |    1 | A+          |   1663134184 |    1663134184
       2 | abcd | arjun   |  23 | chennai |    0 | A+          |   1663134184 |    1663134184
(2 rows)
```

# JAVA PROJECT ARCHITECTURE:

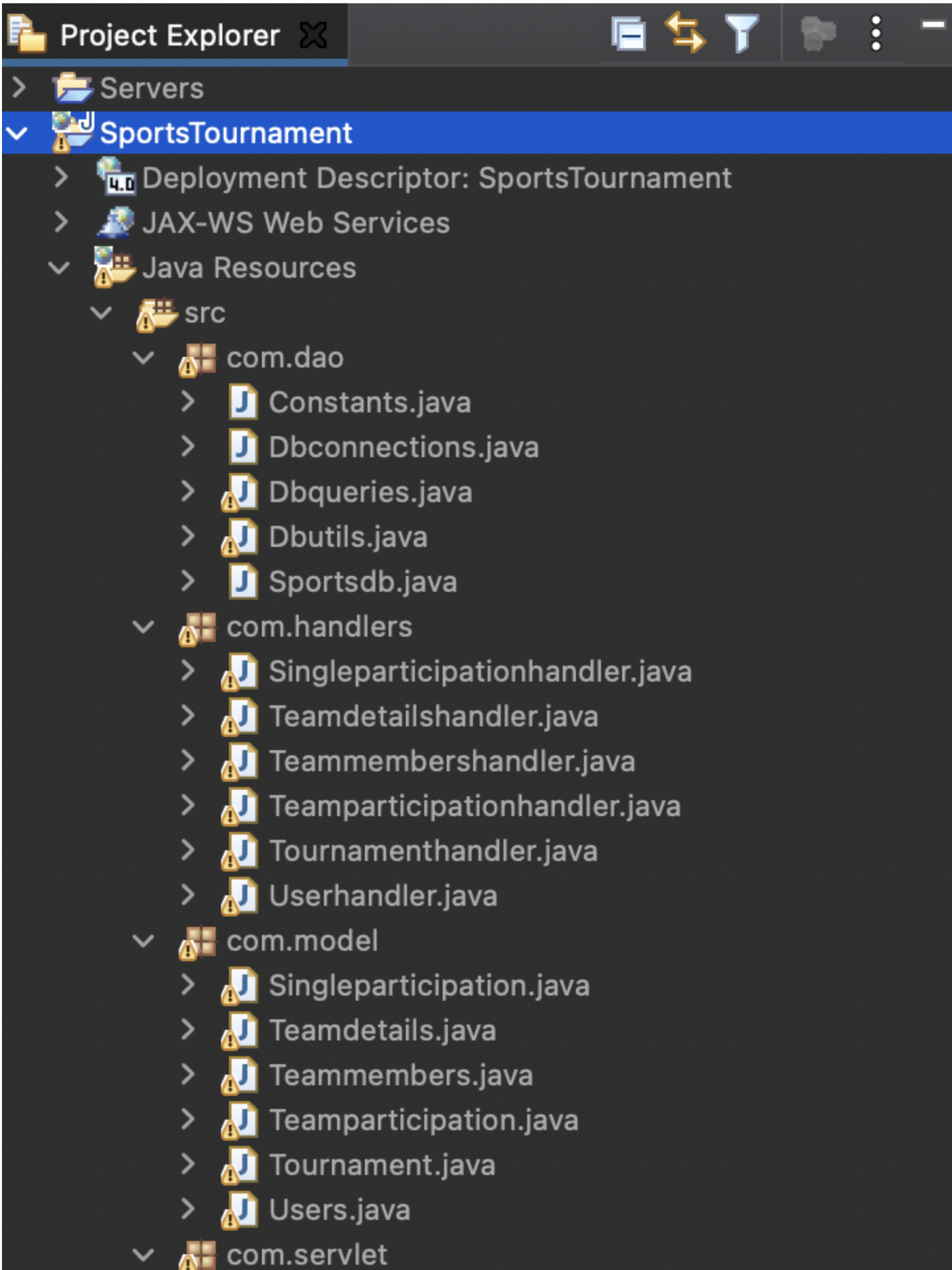
## KEY POINTS:

1.Created  Handlers Package to call respective methods in the handler  based on the URI called.

2.Created com.dao package to ensure all DB related queries are handled in single package.

3.Dbutils is used to handle all DB related functions in a single class.

4.Constants is used to store default table values in Enum function.


Project Explorer
Servers
SportsTournament
Deployment Descriptor: SportsTournament
JAX-WS Web Services
Java Resources
src
com.dao
Constants.java
Dbconnections.java
Dbqueries.java
Dbutils.java
Sportsdb.java
com.handlers
Singleparticipationhandler.java
Teamdetailshandler.java
Teammembershandler.java
Teamparticipationhandler.java
Tournamenthandler.java
Userhandler.java
com.model
Singleparticipation.java
Teamdetails.java
Teammembers.java
Teamparticipation.java
Tournament.java
Users.java
com.servlet

# Packages:

**com.dao :**

Dbconnections.java
Dbqueries.java
Dbutils.java
Constants.java

It contains the methods to establish a connection with Postgres Database using JDBC Drivers and do further Queries on database.

**com.model:**

Tournament.java
Singleparticipants.java
Teamparticipants.java
Teamdetails.java
Teammembers.java
USER.java

It contains constructors and getter methods for Respective Model,which is used while setting Json input array to bean class of respective model.

**com.servlet:**

Corsfilter.java
MainServlet.java

1.Corsfilter is established to enable access to our URI from different domains , so that data can be transfered to other domain portals.
2.MainServlet is used to handle all GET , PUT ,POST , and DELETE operations and call respective handler and invoke respective methods.

## VARIOUS METHODS USED :

## DBconnections :

```
1 1. public static Connection startconnection()
  throws Exception {
2 2. public  static void closeconnection (Connection
  c) throws Exception{
```

### startconnection() :

start connection method is used create a connection with DB and return connection as paramater and this method can be used from main class .

**`closeconnection (Connection conn) :`**

Used to close connection with DB and this method can be used from main class ..

## Dbqueries :

## Common Dynamic Methods for CURD Operations :

### Method : INSERTINTO

```
1 public static void insertDatas(LinkedHashMap<String,
  Object> dataMap, String tableName) throws Exception {
```

1.Used to insert values into tables .
2.Used to pass column values as array objects .
3.Used to pass columns as String of columns.

### Method : UPDATETABLE

```
1  public static void updateTable(String tablename,
  HashMap<String, Object> map, HashMap<String,
  Object> map1) throws Exception {
```

1.This method is used to generate update query  and return updated query as string.
2.I used 2 HashMaps : one HashMap for (column,value) pair as  (key,value) and (id,value) as (key,value).

**Method  :  DELETE**

```
1   public static void delete(HashMap<String, Object> map, String segments) throws Exception {
```

1.This method is used to generate delete  query  and return delete query as string.
2.HashMap is used to get column  as key and column value as a value .

## Method : Select

```
1   public static ResultSet fetchAll(String table_name) throws Exception {
```

1.Used to select all data from table  just by using table name.

# Back End - API  EndPoints :

## HTTP Methods :

1.The doPost method handle the login/signup and adding tournament, users and team event.

2.The doGet method handles the  team event ,single participation list,tournament  homepage data and, reports API call.

3.The doPut method handles adding/removing team for a tournament, updating teams.

4. The doDelete method handles tournament, users and team event delete function for Admin.

# List of Back End - API EndPoints:

| | |
|---|---|
| **GET api/login** | login for existing users |
| **GET api/tournament** | Returns all tournaments |
| **GET api/tournament/id** | Returns particular tournament |
| **POST api/tournament** | Creates new tournament |
| **PUT api/tournament/id** | Update existing tournament details |
| **DELETE api/tournament/id** | Deletes particular tournament |
| **GET api/user/id** | Returns particular user |
| **GET api/user** | Returns all user details |
| **PUT api/user/id** | Update existing user details |
| **POST api/user** | Creates new user |
| **DELETE api/user/id** | Deletes particular user |
| **GET api/singleparticipation** | Returns all for the Single Participations |
| **GET api/singleparticipation/id** | Returns particular Single Participation |
| **POST api/singleparticipation** | Register for a Single Participation |

| | |
|---|---|
| **PUT api/singleparticipation/id** | Update status of a Single participation. |
| **DELETE api/singleparticipation/id** | Deletes particular Single Participation |
| **GET api/teamparticipation** | Returns all registrations existing for the Team participation |
| **GET api/teamparticipation/id** | Returns particular registrations existing for the Team participation |
| **POST api/teamparticipation** | Register for a Team participation |
| **PUT api/teamparticipation/id** | Update status of a Team participation |
| **DELETE api/teamparticipation/id** | Deletes particular Team participation |
| **GET api/teammembers** | Returns all for the Team members |
| **GET api/teammembers/id** | Returns particular registrations existing for the Team members |
| **POST api/teammembers** | Register for a Team members |
| **PUT api/teammembers/id** | Update status of a Team members |
| **DELETE api/teammembers/id** | Deletes particular Team members |

| | |
|---|---|
| **GET tournament/id/singleparticipation/id** | show the particular single participation under particular tournament id |
| **GET tournament/id/singleparticipation** | show all single participations under particular tournament id |
| **GET tournament/id/teamparticipation/id** | show the particular team participation under particular tournament id |
| **GET tournament/id/teamparticipation** | show all team participations under particular tournament id |

## Points :

1.Inner joins where used in certains GET operations based on use cases required like to team participation of respective tournament and to get single participations of respective tournaments.

# EMBER.JS :

Ember.js is an open-source JavaScript web framework that utilizes a component-service pattern. It allows developers to create scalable single-page web applications by incorporating common idioms, best practices, and patterns from other single-page-app ecosystem patterns into the framework.

**For this Sports Tournament Management System, Frontend is created using Ember framework.**

## Routes used are :

1.login
2.signup
3.admin
4.user

Admin : To handle particularly admin specific usecases :

## Sub Routes :

1.admin/users
2.admin/tournament
3.admin/teamdetails
4.admin/addtournament

5.admin/edittourrnament

User: To handle particularly User specific usecases :

## Sub Routes :

1.user/tournament
2.user/edituser

## Router.js file configurations:

```
this.route('login');
 this.route('signup');
 this.route('admin', function() {
   this.route('users');
   this.route('addtournament');
   this.route('tournament');
   this.route('edittournament');
   this.route('teamdetails');
 });
  this.route('user', function(){
    this.route('edituser');
    this.route('tournament');
  });
});
```