DBMS Project Report

TITLE: SPORTS TOURNAMENTS MANAGEMENT SYSTEM

Collaborated on by:

1. Vikas Baliga

   PES1UG22CS695

2. Vishnu Sridhar
   PES1UG22CS702

5th Semester L section

Description about the Statement

The Sports Tournaments Management System is designed to streamline the organization, management, and tracking of sports tournaments. This system provides a platform for managing multiple aspects of tournaments, including tournament tracking, role-based access control and admin-managed tournament workflows.

User Requirement Specification in Detail

1. Purpose of the project:

The **Sports Tournaments Management System** is designed to address the challenges of organizing and managing sports tournaments by providing a comprehensive digital platform. The system enables efficient tournament tracking, ensuring that all key information such as schedules, scores, and player statistics is easily accessible and up-

todate. It also introduces a structured approach to workflows, allowing administrators to manage tournaments seamlessly from start to finish.

This system serves two primary user groups: **Admins** and **Users**, with features tailored to meet their specific needs. Admins have full control over the system, including the ability to add, edit, or remove tournaments, teams, and players. They can also manage workflows and assign roles, ensuring smooth operations. On the other hand, Users, such as players, coaches, or spectators, can view tournament details and track progress, fostering transparency and engagement. By implementing role-based access control, the system ensures that data integrity is maintained while providing relevant access to each user group.

2. Scope of the project:

The Sports Tournaments Management System is designed to provide a centralized and efficient platform for managing sports tournaments of varying sizes and complexity. The system supports core functionalities such as creating, editing, and deleting tournament details, managing player and team data, scheduling games, and tracking results. These features are tailored for Admins, who hold full control over tournament workflows, ensuring seamless organization and management.

For Registered Users, the system offers a simplified, data-driven interface with view-only access. Users can track tournament schedules, team standings, player statistics, and results, enabling transparency and engagement. By integrating role-based access control, the system ensures secure handling of data, with modification privileges exclusively for Admins and view-only access for Users. This scope makes the system ideal for enhancing efficiency, accuracy, and user experience in managing sports tournaments at any level.

3. Detailed description:

The **Sports Tournaments Management System** is built on a robust relational database structure designed to handle various aspects of sports tournaments. The project is centered around the organization and tracking of tournaments, players, teams, and games, while ensuring role-based access and data integrity.

1. **Entities and Their Relationships:**
    - **Players:** Each player is uniquely identified by a player_id. The entity stores details such as the player's name, age, and the team they are associated with. A player is linked to a team through the team_id.
    - **Teams:** Teams are uniquely identified by a team_id and include attributes like team_name, home_ground, team_captain, and associated coach_id. Each team can participate in multiple games and tournaments.
    - **Coaches:** Coaches are uniquely identified by a coach_id and are linked to the teams they coach via the team_id. The entity stores the coach's name and age.

- **Manager:** Each manager is identified by a manager_name and is associated with a specific team. The entity includes information about the manager's years with the team.
- **Merchandise (Merch):** This entity represents team merchandise, identified by merch_name, with details such as the price. Each merchandise item is linked to a specific team through team_merch.
- **Tournaments:** Tournaments are uniquely identified by a tournament_id. This entity includes attributes like tournament_name, start_date, and end_date. Teams participate in tournaments, and games are scheduled within the context of tournaments.
- **Games:** Each game is uniquely identified by a game_id and is associated with a specific tournament_id. The entity includes attributes like game_name, participating teams (team_1 and team_2), game date, and location.
- **Scores:** This entity tracks the scores of teams participating in games. It is uniquely identified by a match_id and includes attributes such as team1_score, team2_score, and the associated game_name.
- **Users:** This entity handles system users and supports role-based access. Each user is identified by a user_id and includes details like username, password, and a flag (is_admin) to distinguish between Admins and Readers.

2. **System Features and Functionalities:**
    - **Admin Access:** Admin users have complete control over managing tournaments, teams, players, games, and scores. They can add, edit, and remove records in the system. Admins also manage tournament workflows and assign roles to users.
    - **Registered Reader Access:** Regular users (Readers) can view tournament details, team standings, player statistics, game schedules, and scores. They have restricted access, ensuring the integrity of sensitive data.
    - **Game and Score Management:** The system records all game details, including participating teams, date, location, and scores, allowing tracking of tournament progress.
    - **Merchandise Integration:** The system maintains information about team merchandise, enabling integration with future e-commerce functionalities.

- **Role-Based Access Control:** Role-based access ensures that data modification is restricted to Admin users, while Registered Users have viewonly permissions.

3. **Workflow Overview:**

   - Admins create and manage tournaments by registering teams, coaches, and players. They also schedule games and input scores.
   - Users log in to view updates on tournaments, game outcomes, and team performance.
   - The system maintains referential integrity between entities, ensuring consistent and accurate data relationships.

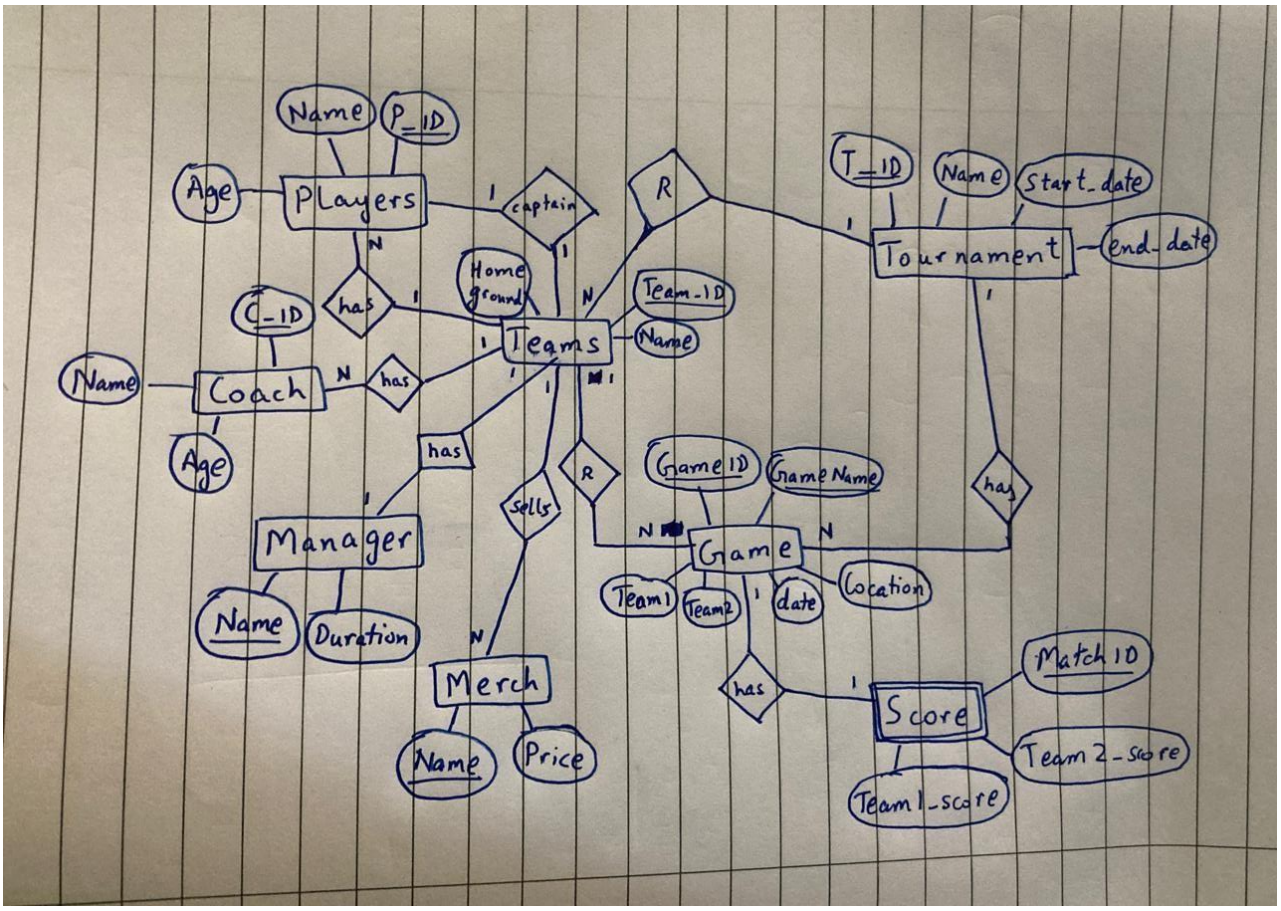
This system provides a comprehensive and user-friendly solution for managing sports tournaments, enabling organizers to maintain data accuracy, streamline operations, and enhance user engagement.

List of Softwares/Tools/Programming Languages Used

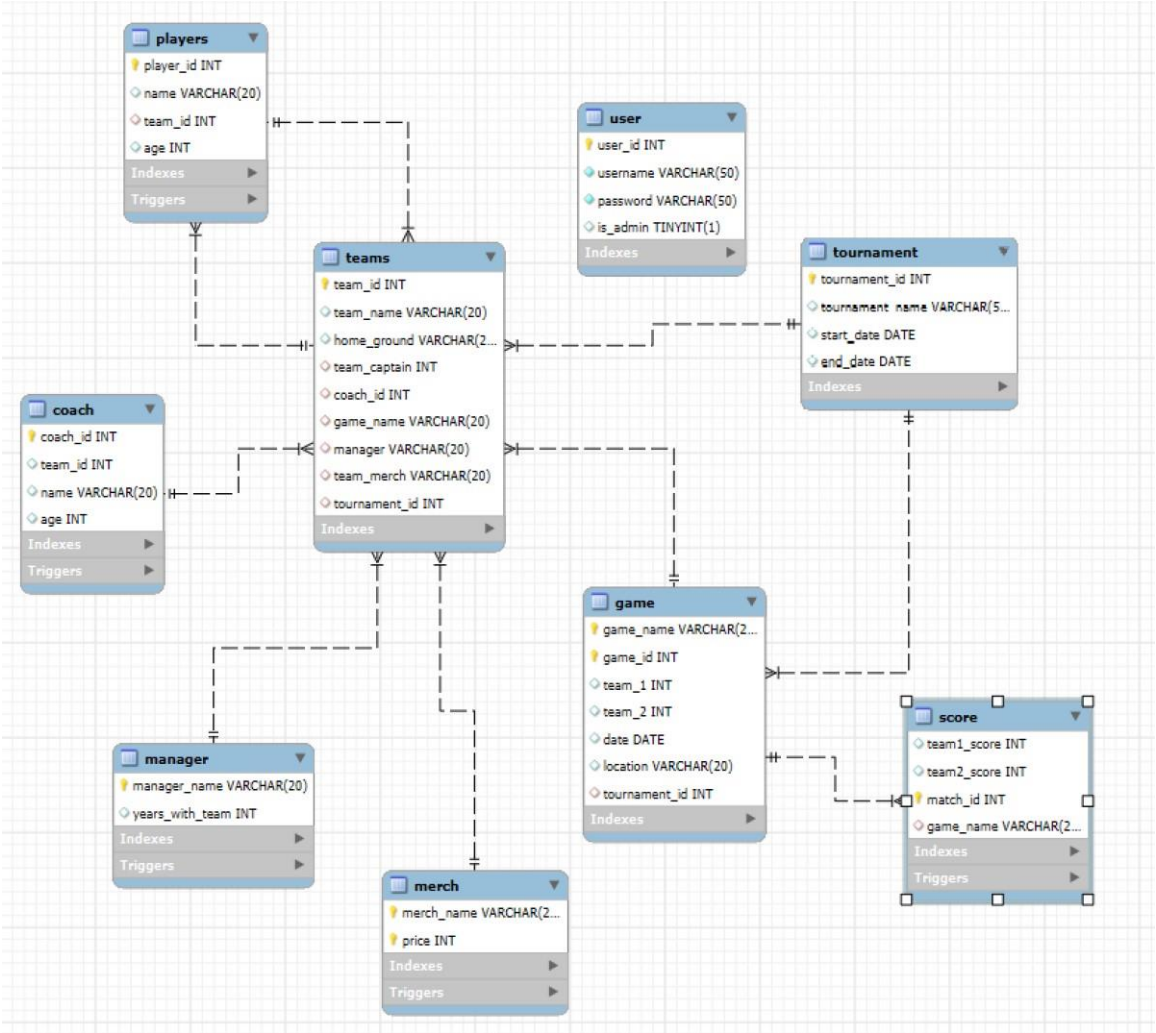
Software: MySQL (Workbench + Command Line Client)

- Programming Languages:
  - Frontend: HTML, CSS, Jinja
  - Backend: Flask (Python)

ER Diagram


Name
P_ID
Age
Players
captain
R
T_ID
Name
Start_date
Tournament
End_date
has
Home ground
Team_ID
C_ID
Teams
Name
Name
Coach
has
Age
has
Sells
R
Game ID
Game Name
has
Manager
Game
Team1
Team2
date
Location
Name
Duration
Merch
has
Score
Match ID
Team 2_score
Name
Price
Team 1_score

Relational Schema



DDL Commands

1. Coach

```
DROP TABLE IF EXISTS `coach`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `coach` (
  `coach_id` int NOT NULL AUTO_INCREMENT,
  `team_id` int DEFAULT NULL,
  `name` varchar(20) DEFAULT NULL,
  `age` int DEFAULT NULL,
  PRIMARY KEY (`coach_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `coach`
--

LOCK TABLES `coach` WRITE;
/*!40000 ALTER TABLE `coach` DISABLE KEYS */;
INSERT INTO `coach` VALUES (1,1,'Tom Coach',45),(2,2,'Anna Manager',50),(3,3,'Ella Coach',42),(4,4,'Liam Manager',38),(5,5,'Maya Manager',47),(6,1,'Gary Assistant',33)
/*!40000 ALTER TABLE `coach` ENABLE KEYS */;
UNLOCK TABLES;
```

2. Game

```
DROP TABLE IF EXISTS `game`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `game` (
  `game_name` varchar(20) NOT NULL,
  `game_id` int NOT NULL AUTO_INCREMENT,
  `team_1` int DEFAULT NULL,
  `team_2` int DEFAULT NULL,
  `date` date DEFAULT NULL,
  `location` varchar(20) DEFAULT NULL,
  `tournament_id` int DEFAULT NULL,
  PRIMARY KEY (`game_name`,`game_id`),
  KEY `fk_game_tournament` (`tournament_id`),
  CONSTRAINT `fk_game_tournament` FOREIGN KEY (`tournament_id`) REFERENCES `tournament` (`tournament_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `game`
--

LOCK TABLES `game` WRITE;
/*!40000 ALTER TABLE `game` DISABLE KEYS */;
INSERT INTO `game` VALUES ('Baseball',5,1,2,'2024-11-05','Stadium E',1),('Basketball',2,1,3,'2024-11-02','Court B',1),('Cricket',3,4,5,'2024-11-03','Ground C',2),('Hoc
/*!40000 ALTER TABLE `game` ENABLE KEYS */;
UNLOCK TABLES;
```

3. Manager

```
DROP TABLE IF EXISTS `manager`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `manager` (
  `manager_name` varchar(20) NOT NULL,
  `years_with_team` int DEFAULT NULL,
  PRIMARY KEY (`manager_name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `manager`
--

LOCK TABLES `manager` WRITE;
/*!40000 ALTER TABLE `manager` DISABLE KEYS */;
INSERT INTO `manager` VALUES ('Anna Manager',5),('Ella Coach',8),('Gary Assistant',2),('Liam Manager',6),('Linda Assistant',4),('Maya Manager',7),('Nina Assistant',1),
/*!40000 ALTER TABLE `manager` ENABLE KEYS */;
UNLOCK TABLES;
```

4. Merch

```
DROP TABLE IF EXISTS `merch`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `merch` (
  `merch_name` varchar(20) NOT NULL,
  `price` int NOT NULL,
  PRIMARY KEY (`merch_name`,`price`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `merch`
--

LOCK TABLES `merch` WRITE;
/*!40000 ALTER TABLE `merch` DISABLE KEYS */;
INSERT INTO `merch` VALUES ('Bag H',25),('Cap B',20),('Hoodie E',30),('Jersey A',50),('Keychain F',5),('Mug D',10),('Poster G',7),('Scarf C',15),('Socks I',8),('Wristb
/*!40000 ALTER TABLE `merch` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `players`
```

5. Players

```
DROP TABLE IF EXISTS `players`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `players` (
  `player_id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(20) DEFAULT NULL,
  `team_id` int DEFAULT NULL,
  `age` int DEFAULT NULL,
  PRIMARY KEY (`player_id`),
  KEY `team_id` (`team_id`),
  CONSTRAINT `players_ibfk_1` FOREIGN KEY (`team_id`) REFERENCES `teams` (`team_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `players`
--

LOCK TABLES `players` WRITE;
/*!40000 ALTER TABLE `players` DISABLE KEYS */;
INSERT INTO `players` VALUES (1,'Alice Smith',1,25),(2,'Bob Johnson',1,28),(3,'Charlie Lee',2,22),(4,'Diana Wang',2,26),(5,'Evan Thomas',3,24),(6,'Fiona Green',3,27),(
/*!40000 ALTER TABLE `players` ENABLE KEYS */;
UNLOCK TABLES;
```

6. Score

```
DROP TABLE IF EXISTS `score`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `score` (
  `team1_score` int DEFAULT NULL,
  `team2_score` int DEFAULT NULL,
  `match_id` int NOT NULL,
  `game_name` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`match_id`),
  KEY `game_name` (`game_name`),
  CONSTRAINT `score_ibfk_1` FOREIGN KEY (`game_name`) REFERENCES `game` (`game_name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `score`
--

LOCK TABLES `score` WRITE;
/*!40000 ALTER TABLE `score` DISABLE KEYS */;
INSERT INTO `score` VALUES (3,1,1,'Soccer'),(90,85,2,'Basketball'),(200,150,3,'Cricket'),(2,3,4,'Hockey'),(5,4,5,'Baseball');
/*!40000 ALTER TABLE `score` ENABLE KEYS */;
UNLOCK TABLES;
```

7. Teams

```
DROP TABLE IF EXISTS `teams`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `teams` (
  `team_id` int NOT NULL AUTO_INCREMENT,
  `team_name` varchar(20) DEFAULT NULL,
  `home_ground` varchar(20) DEFAULT NULL,
  `team_captain` int DEFAULT NULL,
  `coach_id` int DEFAULT NULL,
  `game_name` varchar(20) DEFAULT NULL,
  `manager` varchar(20) DEFAULT NULL,
  `team_merch` varchar(20) DEFAULT NULL,
  `tournament_id` int DEFAULT NULL,
  PRIMARY KEY (`team_id`),
  KEY `coach_id` (`coach_id`),
  KEY `team_captain` (`team_captain`),
  KEY `next_game` (`game_name`),
  KEY `manager` (`manager`),
  KEY `team_merch` (`team_merch`),
  KEY `fk_teams_tournament` (`tournament_id`),
  CONSTRAINT `fk_teams_tournament` FOREIGN KEY (`tournament_id`) REFERENCES `tournament` (`tournament_id`),
  CONSTRAINT `teams_ibfk_1` FOREIGN KEY (`coach_id`) REFERENCES `coach` (`coach_id`),
  -- CONSTRAINT `teams_ibfk_2` FOREIGN KEY (`team_captain`) REFERENCES `players` (`player_id`),
  CONSTRAINT `teams_ibfk_3` FOREIGN KEY (`game_name`) REFERENCES `game` (`game_name`),
  CONSTRAINT `teams_ibfk_4` FOREIGN KEY (`manager`) REFERENCES `manager` (`manager_name`),
  CONSTRAINT `teams_ibfk_5` FOREIGN KEY (`team_merch`) REFERENCES `merch` (`merch_name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
LOCK TABLES `teams` WRITE;
/*!40000 ALTER TABLE `teams` DISABLE KEYS */;
INSERT INTO `teams` VALUES (1,'Warriors','Stadium A',NULL,1,'Soccer','Tom Coach','Jersey A',NULL),(2,'Knights','Stadium B',NULL,2,'Basketball','Anna Manager','Cap B',NU
/*!40000 ALTER TABLE `teams` ENABLE KEYS */;
UNLOCK TABLES;
```

8. Tournament

```
DROP TABLE IF EXISTS `tournament`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `tournament` (
  `tournament_id` int NOT NULL AUTO_INCREMENT,
  `tournament_name` varchar(50) DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `end_date` date DEFAULT NULL,
  PRIMARY KEY (`tournament_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `tournament`
--

LOCK TABLES `tournament` WRITE;
/*!40000 ALTER TABLE `tournament` DISABLE KEYS */;
INSERT INTO `tournament` VALUES (1,'Champions League','2024-01-10','2024-02-10'),(2,'Premier Cup','2024-03-15','2024-04-15'),(3,'Global Invitational','2024-05-20','2024
/*!40000 ALTER TABLE `tournament` ENABLE KEYS */;
UNLOCK TABLES;
```

9. User

```
DROP TABLE IF EXISTS `user`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `user` (
  `user_id` int NOT NULL AUTO_INCREMENT,
  `username` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `is_admin` tinyint(1) DEFAULT '0',
  PRIMARY KEY (`user_id`),
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `user`
--

LOCK TABLES `user` WRITE;
/*!40000 ALTER TABLE `user` DISABLE KEYS */;
INSERT INTO `user` VALUES (1,'alice','password123',0),(2,'bob','password456',1),(3,'charlie','password789',0),(4,'doug','password147',0),(5,'ezekial','password258',0);
/*!40000 ALTER TABLE `user` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

CRUD Operation Screenshots

CREATE

```
values = [team_name, home_ground, team_captain, coach_id, game_name, manager, team_merch, tournament_id]
column_names = ["team_name", "home_ground", "team_captain", "coach_id", "game_name", "manager", "team_merch", "tournament_id"]

if 'add' in request.form:

    query = f"INSERT INTO teams ({', '.join(column_names)}) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
    cur.execute(query, values)

    flash('Team added successfully', 'success')
```

READ

```
db = get_db()
cur=db.cursor()

cur.execute('''
SELECT
    t.team_id, t.team_name, g.game_name, t.tournament_id
FROM
    teams t
JOIN
    players p ON t.team_id = p.team_id
WHERE
    t.team_id IN (SELECT team_id FROM players
                  GROUP BY team_id HAVING MAX(age) < 20);
''')
eligible=cur.fetchall()
cur.close()
```

UPDATE

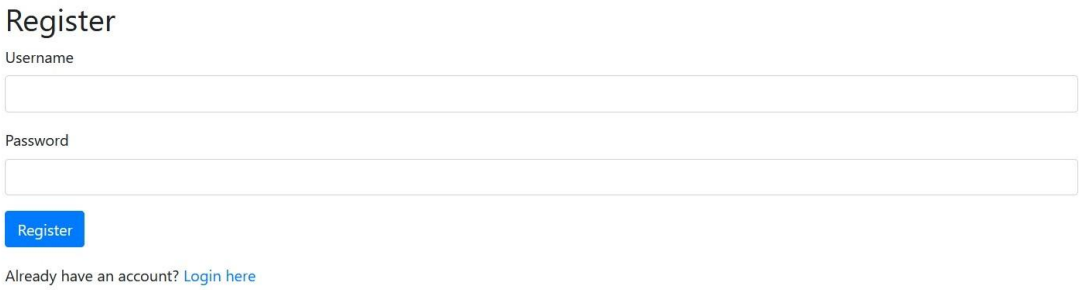
```
if 'edit' in request.form and team_id:
    # Prepare the update query
    query = """
        UPDATE teams
        SET team_name = %s, home_ground = %s, team_captain = %s,
            coach_id = %s, game_name = %s, manager = %s,
            team_merch = %s, tournament_id = %s
        WHERE team_id = %s
    """
    values = [team_name, home_ground, team_captain, coach_id, game_name, manager, team_merch, tournament_id, team_id]

    # Execute the update query
    cur.execute(query, values)
    db.commit()
    flash('Team updated successfully', 'success')
```

DELETE

```
if 'delete' in request.form and team_id:
    # Execute deletion of the team with the given team_id
    cur.execute('DELETE FROM teams WHERE team_id = %s', (team_id,))
    db.commit()
    flash('Team deleted successfully', 'success')
else:
    flash('Please enter a valid Team ID', 'warning')
```

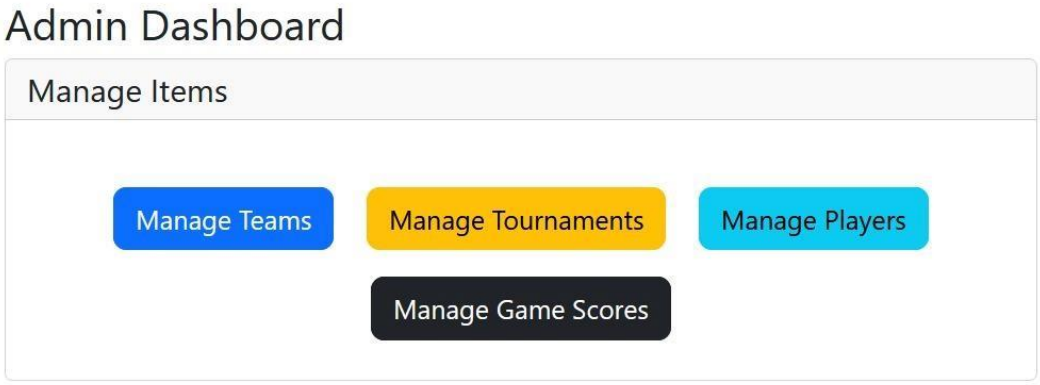
List of Functionalities/Features of the Application and Its Associated Screenshots Using Front-End

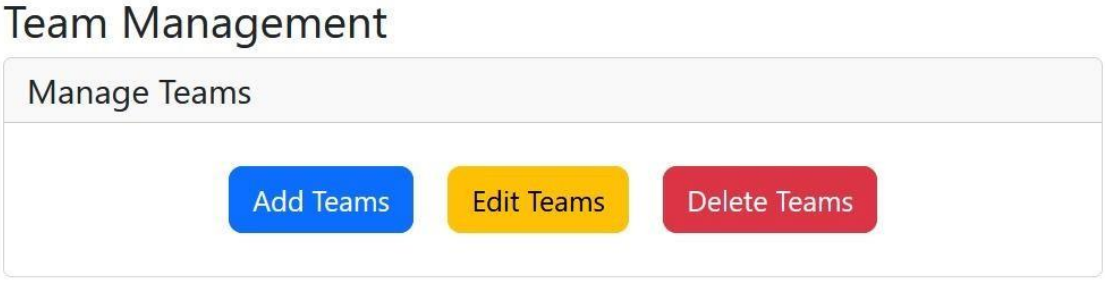
1. Login checks for registered credentials and maps to the role (admin/user) of the username. There is also an option to register for new users

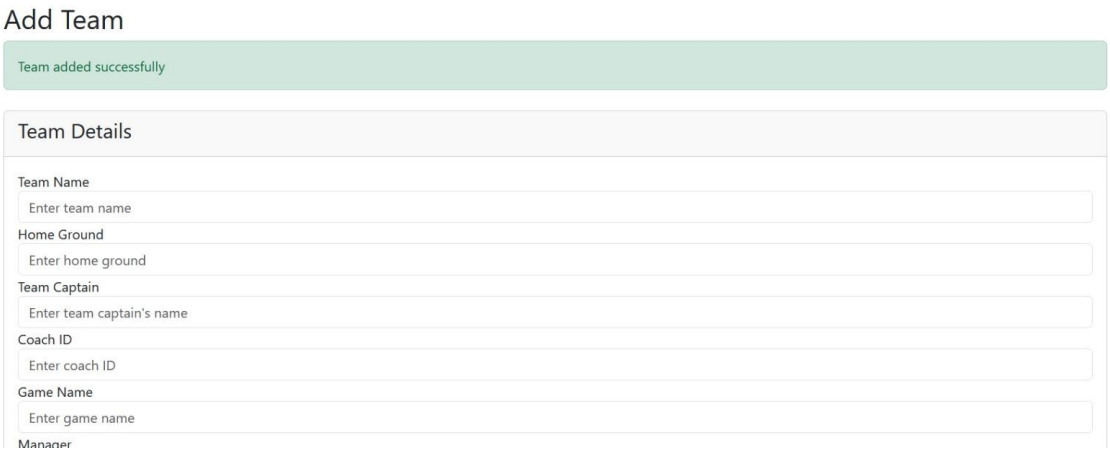


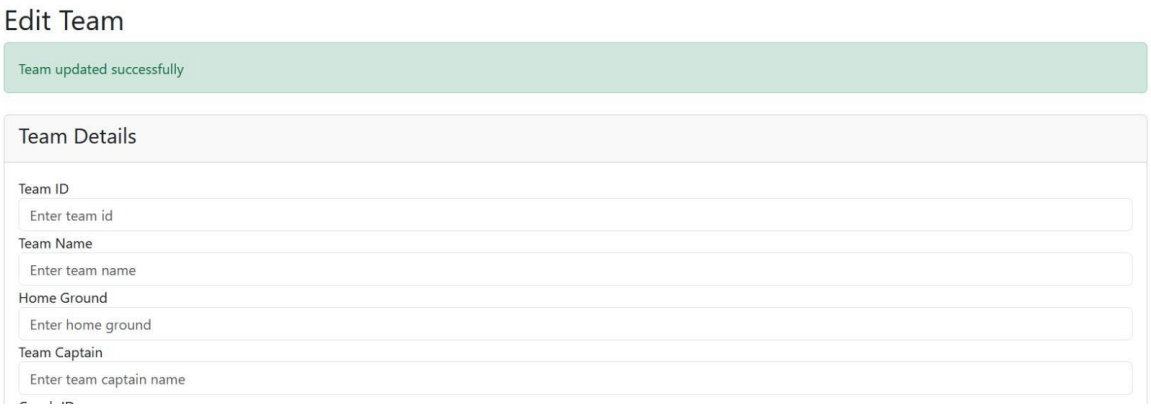
2. Admins can add, update, and delete

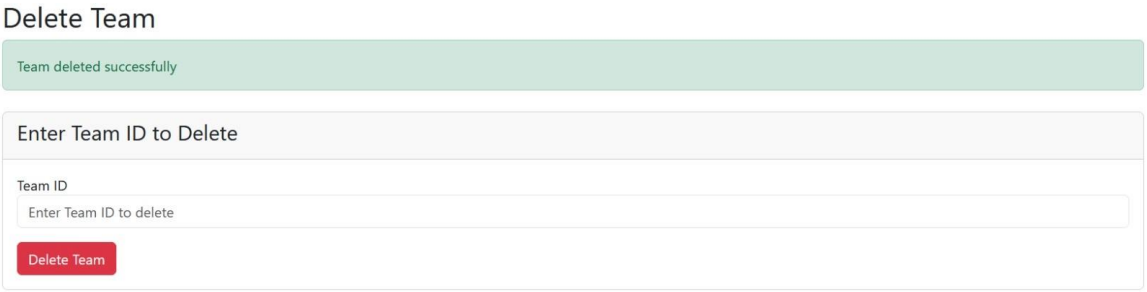
    a. Tournaments
    b. Teams
    c. Players
    d. Game scores


Admin Dashboard
Manage Items
Manage Teams
Manage Tournaments
Manage Players
Manage Game Scores

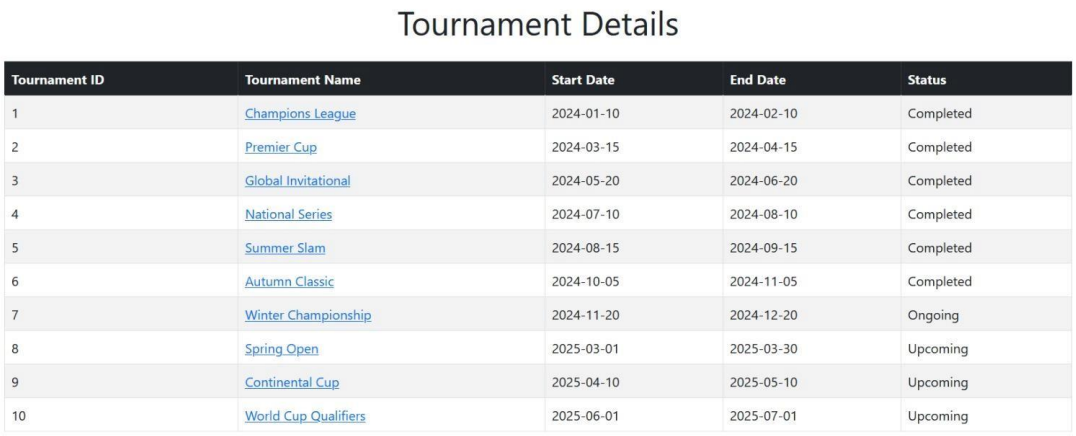

Team Management
Manage Teams
Add Teams
Edit Teams
Delete Teams


Add Team
Team added successfully
Team Details
Team Name
Enter team name
Home Ground
Enter home ground
Team Captain
Enter team captain's name
Coach ID
Enter coach ID
Game Name
Enter game name
Manager





3. Users can view the Tournament data. If they click on a tournament's name, they get redirected to view the details of the games and teams participating in the tournament. Clicking on the team further redirects the viewer to all the players.



Team Details

Team ID	Team Name	Game Name	Tournament ID
18	Sprinters	Soccer	1
17	Sprinters	Soccer	1
15	ABC	Soccer	1
10	Walkers	Soccer	1
1	Warriors	Soccer	1

Check for Eligible Teams for raffle. Requirements: Under 20

Game score Details

Game ID	Game Name	Team-1	Team-2	Team 1 Score	Team 2 Score
1	Soccer	None	None	3	1
1	Soccer	None	None	3	1
1	Soccer	None	None	3	1
1	Soccer	None	None	3	1
1	Soccer	None	None	3	1

Player Details

Player ID	Name	Team ID	Age
1	Bobby Jackson	2	24
2	Bob Johnson	1	28
3	Charlie Lee	2	22
4	Diana Wang	2	26
5	Evan Thomas	3	24
6	Fiona Green	3	27
7	George Harris	4	29
8	Hannah Brown	4	23
9	Ian White	5	30
10	Jack Black	5	21

Triggers, Procedures/Functions, Nested Query, Join, Aggregate Queries

The following is a complex nested join query used to join team, game and tournament tables in order to get a resultant table of all the teams participating in a tournament, which is the filtered with a specific tournament_id to show the teams and games under each tournament.

```
SELECT team_id, team_name, game.game_name, teams.tournament_id, game_id,
       team_1, team_2, team1_score, team2_score
FROM ((teams
        JOIN game ON teams.game_name = game.game_name)
        JOIN score ON game.game_name = score.game_name)
        JOIN tournament ON tournament.tournament_id = teams.tournament_id
WHERE teams.tournament_id = %s
''', (tournament_id,))  # Dynamically filter by tournament_id
```

Triggers – Used to ensure that admins cannot accidentally enter wrong information for details like age.

```
DELIMITER $$

CREATE TRIGGER check_age_before_insert_update
BEFORE INSERT ON players
FOR EACH ROW
BEGIN
    IF NEW.age > 100 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Age cannot be greater than 100';
    END IF;
END$$

DELIMITER ;
```

Procedures – Used to check validity of tournament dates

```
DELIMITER $$

CREATE PROCEDURE validate_tournament_dates(
    IN p_tournament_name VARCHAR(255),
    IN p_start_date DATE,
    IN p_end_date DATE
)
BEGIN
    -- Validate that the end date is not before the start date
    IF p_end_date < p_start_date THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'End date must be on or after start date';
    END IF;

    -- Insert the tournament record if validation passes
    INSERT INTO tournaments (tournament_name, start_date, end_date)
    VALUES (p_tournament_name, p_start_date, p_end_date);

END $$

DELIMITER ;
```

Functions – Assigning status to a tournament based on its start and end dates

```
DELIMITER $$
CREATE FUNCTION get_tournament_status(starting_date DATE, end_date DATE)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    DECLARE status VARCHAR(20);
    DECLARE today DATE;

    -- Get today's date
    SET today = CURDATE();

    -- Compare today's date with the start and end dates to determine the status
    IF today < starting_date THEN
        SET status = 'Upcoming'; -- Tournament hasn't started yet
    ELSEIF today > end_date THEN
        SET status = 'Completed'; -- Tournament has ended
    ELSE
        SET status = 'Ongoing'; -- Tournament is currently ongoing
    END IF;

    RETURN status;
END $$

DELIMITER ;

DELIMITER $$
```

Nested Query, Join, Aggregate queries – Used to check if teams satisfy the criteria of under20 for specific tournaments/prizes

```
SELECT
    t.team_id, t.team_name, t.game_name, t.tournament_id
FROM
    teams t
JOIN
    players p ON t.team_id = p.team_id
WHERE
    t.team_id IN (
        SELECT
            team_id
        FROM
            players
        GROUP BY
            team_id
        HAVING
            MAX(age) < 20
    );
''')
```

Code Snippets for Invoking the Procedures/Functions/Trigger

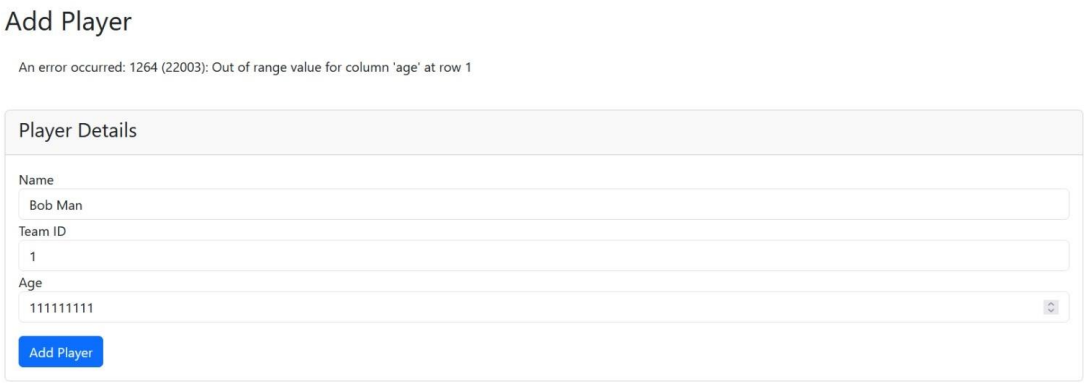
Function

```
# Fetch updated tournament details
cur.execute('''
    SELECT tournament_id, tournament_name, start_date, end_date,
        get_tournament_status(start_date, end_date) AS status
    FROM tournament
''')
tournaments = cur.fetchall()

cur.close()
```

Procedure

```
@user_bp.route('/tournaments', methods=['GET', 'POST'])
def Tournaments_view():
    db = get_db()
    cur = db.cursor()

    if request.method == 'POST':
        tournament_name = request.form.get('tournament_name')
        start_date = request.form.get('start_date')
        end_date = request.form.get('end_date')

        # Call the stored procedure to validate dates and insert the tournament
        try:
            # Execute the stored procedure with the provided inputs
            cur.callproc('validate_tournament_dates', [tournament_name, start_date, end_date])

            # Commit the transaction (if the procedure doesn't raise an error)
            db.commit()

            flash('Tournament added successfully', 'success')
```

Trigger



GitHub Repo Link

Provide the link to the GitHub repository containing the project files.

https://github.com/VikasBaliga/Sports_Tournament_Management_System.git