Mini project report on

SPORTS MANAGEMENT SYSTEM

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering UE22CS351A

– DBMS Project

Submitted by:

Abhinav V P	**PES2UG22CS018**
Abishek k	**PES2UG22CS024**

Under the guidance of

Dr. Suja C M

Assistant Professor

PES University

AUG - DEC 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

CERTIFICATE

This is to certify that the mini project entitled

Collaborative Event Planning Platform

is a bonafide work carried out by

Abhinav V P	**PES2UG22CS018**
Abishek k	**PES2UG22CS024**

In partial fulfilment for the completion of fifth semester DBMS Project (UE22CS351A) in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2024 – DEC. 2024. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

Signature

Dr. Suja C M Assistant

Professor

DECLARATION

We hereby declare that the DBMS Project entitled **University Fest Management System** has been carried out by us under the guidance of **Prof. Nivedita Kasturi, Assistant Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2024.

Abhinav V P	**PES2UG22CS018**	**<Signature>**
Abishek k	**PES2UG22CS024**	

PROJECT REPORT

Sports Management System

Abstract

This project, titled **Sports Management System**, is a webbased application designed to simplify and enhance the management of sports teams. Developed using React.js for a responsive and interactive frontend, Node.js and Express.js for backend functionality, and PostgreSQL for efficient and structured data storage, the platform serves as a comprehensive solution for coaches, players, and support staff.

The system enables team administrators to manage essential aspects of sports teams, such as player profiles, match schedules, training plans, and performance tracking. Players can access personalized training schedules, match details, and performance updates, while support staff can maintain team statistics and provide relevant notifications. A role-based dashboard provides administrators with insights into team performance, player availability, and match outcomes, fostering a data-driven approach to sports team management.

Built with scalability in mind, the application integrates modern web technologies and follows best practices for database design, ensuring high performance and reliability. With features like automated notifications, secure access control, and a user-friendly interface, the Sports Management System bridges the gap between traditional management methods and modern digital solutions.

Table of Contents

1. Introduction

The **Sports Management System** is a digital platform designed to streamline the activities of sports teams. This system caters to three primary user groups: coaches/team managers, players, and support staff. By replacing manual methods with an intuitive web-based solution, the system enhances communication, scheduling, and performance tracking within teams. Built on cutting-edge web technologies such as React.js, Node.js, and PostgreSQL, the application offers a scalable and user-friendly experience.

Key Features:

1. **Player Management**:
   - Create, update, and manage player profiles, including details such as personal information, position, and performance metrics.
   - Track player availability, health, and statistics across matches and training sessions.
2. **Match Scheduling**:

    - Schedule matches, allocate venues, and manage rosters. ○ Track and update match results and generate performance reports.

3. **Training Management**:
    - Schedule and track training sessions. ○ Develop personalized training plans and monitor progress.

4. **Notifications**:
    - Automated notifications for match schedules, training updates, and team announcements.
    - Real-time alerts for changes in schedules or team meetings.

Why Sports Management System?

Managing sports teams involves juggling multiple tasks, from scheduling and training to performance tracking and communication. Traditional methods often fall short, leading to inefficiencies like:

- Delayed notifications.
- Poorly maintained records.
- Lack of centralized data.

The Sports Management System addresses these challenges by providing a streamlined, automated approach, making team management efficient, data-driven, and collaborative.

2. Problem Definition and User Requirements

Problem Statement

Managing a sports team is often complex, involving multiple layers of planning, coordination, and communication. Inefficiencies arise due to:

- Lack of centralized systems for managing player details, match schedules, and training plans.
- Ineffective communication between players, coaches, and staff.
- Delayed notifications about schedule changes or team updates.
- Manual handling of player statistics and performance data.

User Requirements

1. **Coaches/Team Managers**:
    - Manage player profiles, schedules, and team data. ○ Access detailed statistics and generate performance reports.
2. **Players**:
    - View personalized schedules, training plans, and match details.
    - Receive notifications about team updates.
3. **Support Staff**:

    - Update match results and competition standings.
    - Manage public team profiles and announcements.

User Requirements

Coaches/Team Managers:
- Manage team schedules, players, and training plans.
- Track match results and player performance.

Players:
- Access personal schedules and training plans.
- Receive updates and notifications.

Support Staff:
- Update competition standings and public profiles.

4. Software/Tools/Programming Languages Used

Frontend Development:

- **React.js**: For building an interactive and responsive user interface.
- **Tailwind CSS**: For custom styling and efficient UI design.

Backend Development:

- **Node.js**: For server-side processing and JavaScript runtime.

- **Express.js**: For building APIs and managing backend operations. **Database:**
- **PostgreSQL**: For storing structured data like player profiles, match schedules, and statistics.

Version Control:

- **Git/GitHub**: For tracking code changes and collaboration.

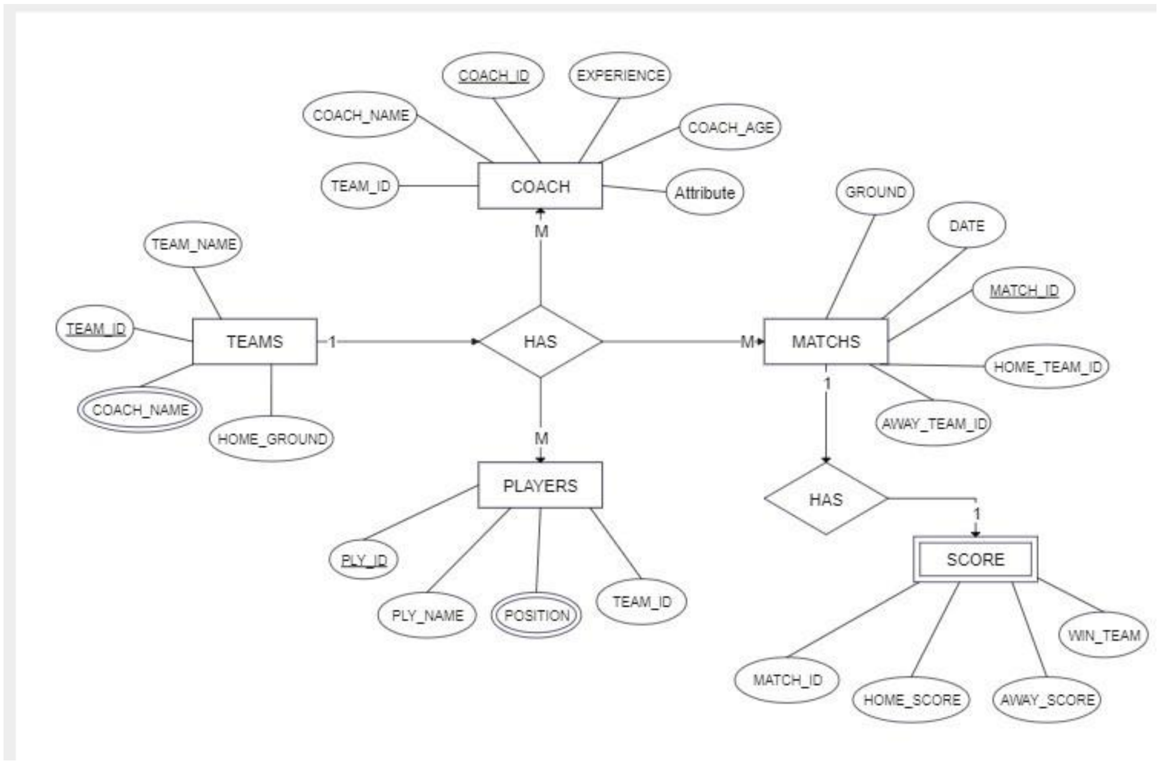
Programming Languages:

- **JavaScript**: Used for backend logic.
- **TypeScript**: For robust and scalable frontend development.

4. System Design and Diagrams

ER Model

The ER diagram represents entities such as Teams, Players, Matches, and their relationships.



Relational Mapping

Transforming the ER model into database tables with proper keys and constraints.

UML Diagrams

- Class Diagram: Illustrates the relationships between system components.
- Use Case Diagram: Highlights interactions between users and the system.

5. Development and Features

Key Features

1. User Management:
- Secure authentication and role-based access.
- CRUD operations for player profiles and team records.

2. Match Scheduling:
- Create, edit, and manage match schedules. -
Allocate venues and track match results.

3. Training Plans:
- Schedule training sessions and track attendance. -
Assign personalized plans to players.

4. Notifications:
- Automated alerts for matches, training, and updates.

6. Database Design and Queries

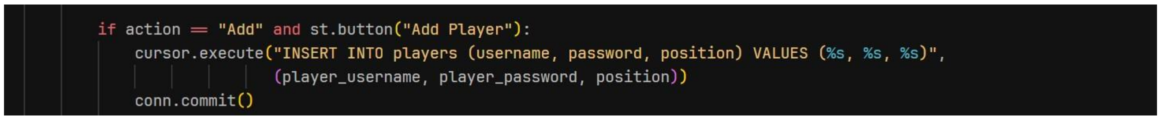
DDL Statements

Example:

```
CREATE DATABASE IF NOT EXISTS sports_team_db;

USE sports_team_db;


CREATE TABLE IF NOT EXISTS coaches (     id

INT AUTO_INCREMENT PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

password VARCHAR(100) NOT NULL

);


CREATE TABLE IF NOT EXISTS players (

id INT AUTO_INCREMENT PRIMARY KEY,
username VARCHAR(50) UNIQUE NOT
NULL,    password VARCHAR(100) NOT
NULL,
   position ENUM('Forward', 'Midfielder', 'Defender',
'Goalkeeper') NOT NULL
);

CREATE TABLE IF NOT EXISTS injuries (
id INT AUTO_INCREMENT PRIMARY KEY,
   player_id INT NOT NULL,    player_username
VARCHAR(50)   NOT   NULL,          injury_name
VARCHAR(100),
   recovery_time INT,
   FOREIGN KEY (player_id) REFERENCES players(id)
);
CREATE TABLE IF NOT EXISTS matches (    id INT
AUTO_INCREMENT PRIMARY KEY,    event_type
ENUM('Match', 'Training') NOT NULL,    opponents
VARCHAR(255) DEFAULT NULL,    event_date
DATE NOT NULL,    event_time TIME NOT NULL
);
```

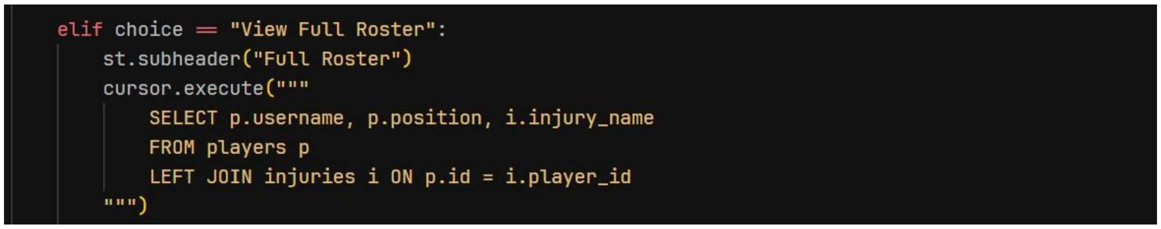
7. DML Statements(CRUD):

Add Player (Create):



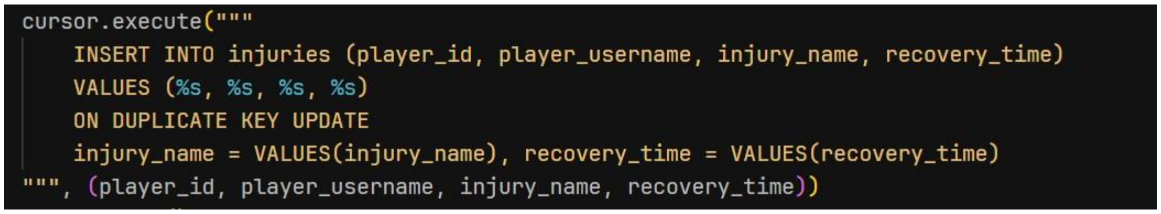
```
        if action == "Add" and st.button("Add Player"):
            cursor.execute("INSERT INTO players (username, password, position) VALUES (%s, %s, %s)",
                           (player_username, player_password, position))
            conn.commit()
```

View All Players(Read):



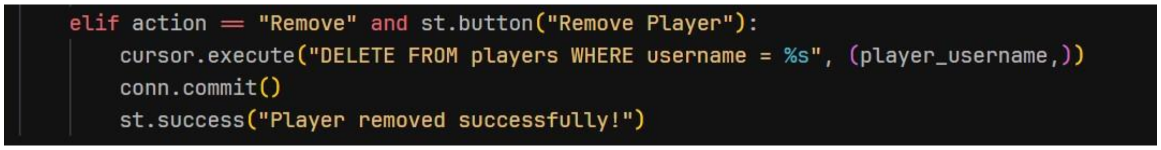
```
    elif choice == "View Full Roster":
        st.subheader("Full Roster")
        cursor.execute("""
            SELECT p.username, p.position, i.injury_name
            FROM players p
            LEFT JOIN injuries i ON p.id = i.player_id
        """)
```

Update Injury Status(Update):



```
cursor.execute("""
    INSERT INTO injuries (player_id, player_username, injury_name, recovery_time)
    VALUES (%s, %s, %s, %s)
    ON DUPLICATE KEY UPDATE
    injury_name = VALUES(injury_name), recovery_time = VALUES(recovery_time)
""", (player_id, player_username, injury_name, recovery_time))
```

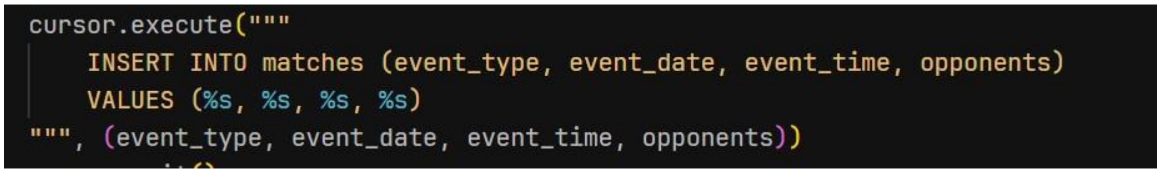
Remove Player(Delete):



```
    elif action == "Remove" and st.button("Remove Player"):
        cursor.execute("DELETE FROM players WHERE username = %s", (player_username,))
        conn.commit()
        st.success("Player removed successfully!")
```

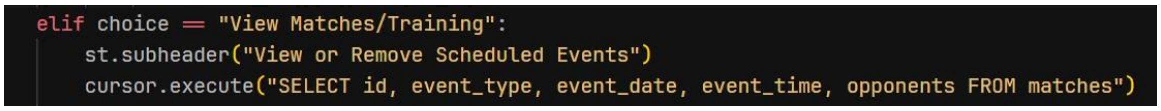
Queries:

Scheduling Matches –



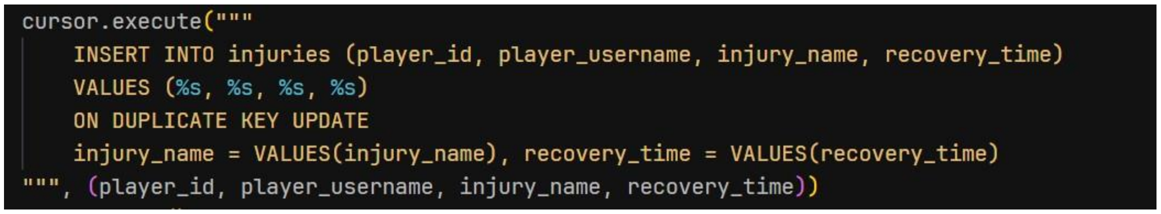
```
cursor.execute("""
    INSERT INTO matches (event_type, event_date, event_time, opponents)
    VALUES (%s, %s, %s, %s)
""", (event_type, event_date, event_time, opponents))
```

View Matches –



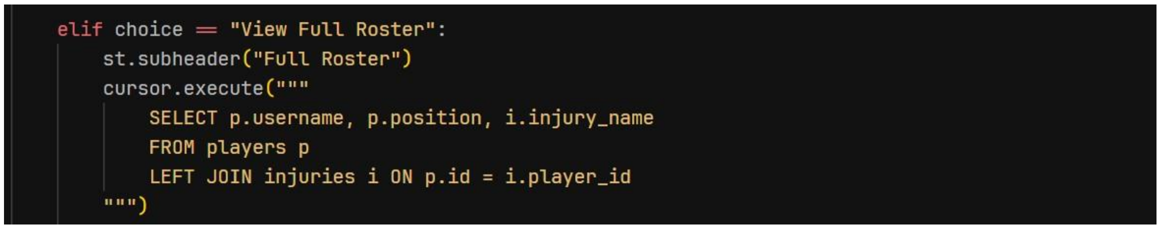
```
elif choice == "View Matches/Training":
    st.subheader("View or Remove Scheduled Events")
    cursor.execute("SELECT id, event_type, event_date, event_time, opponents FROM matches")
```

Adding injuries –



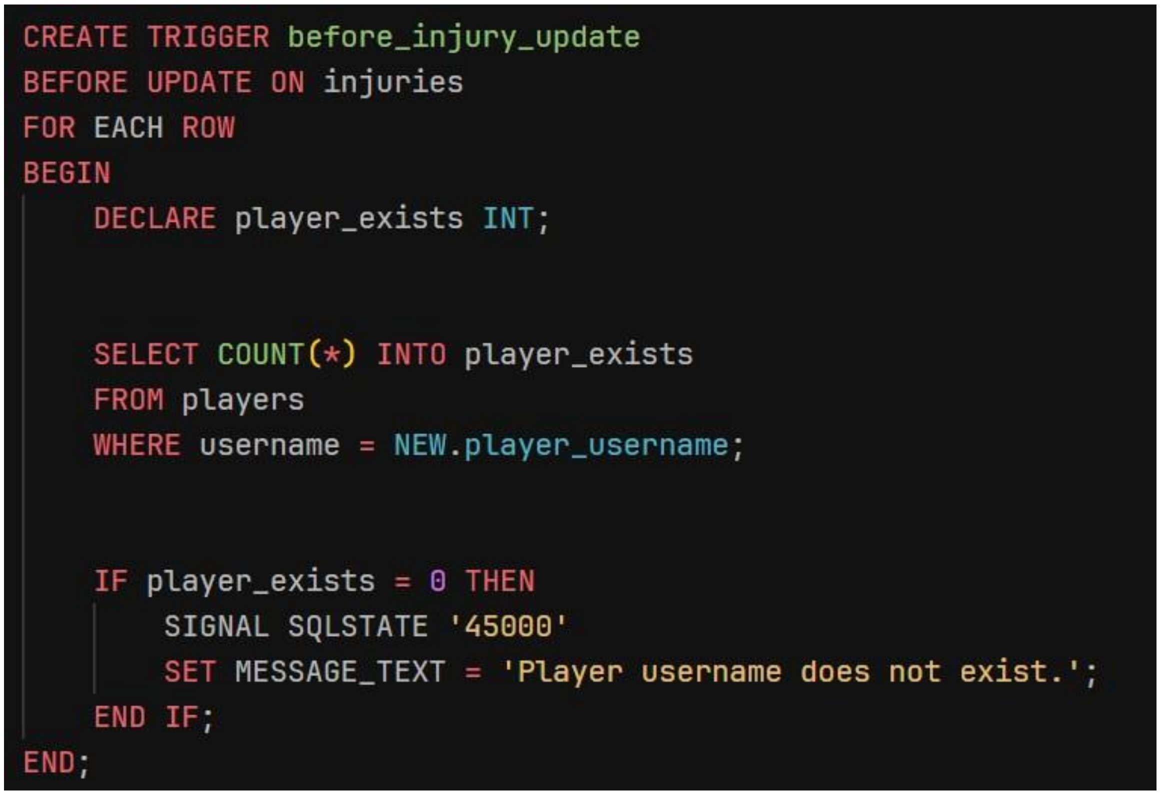
```python
cursor.execute("""
    INSERT INTO injuries (player_id, player_username, injury_name, recovery_time)
    VALUES (%s, %s, %s, %s)
    ON DUPLICATE KEY UPDATE
    injury_name = VALUES(injury_name), recovery_time = VALUES(recovery_time)
""", (player_id, player_username, injury_name, recovery_time))
```

Viewing players –



```python
    elif choice == "View Full Roster":
        st.subheader("Full Roster")
        cursor.execute("""
            SELECT p.username, p.position, i.injury_name
            FROM players p
            LEFT JOIN injuries i ON p.id = i.player_id
        """)
```

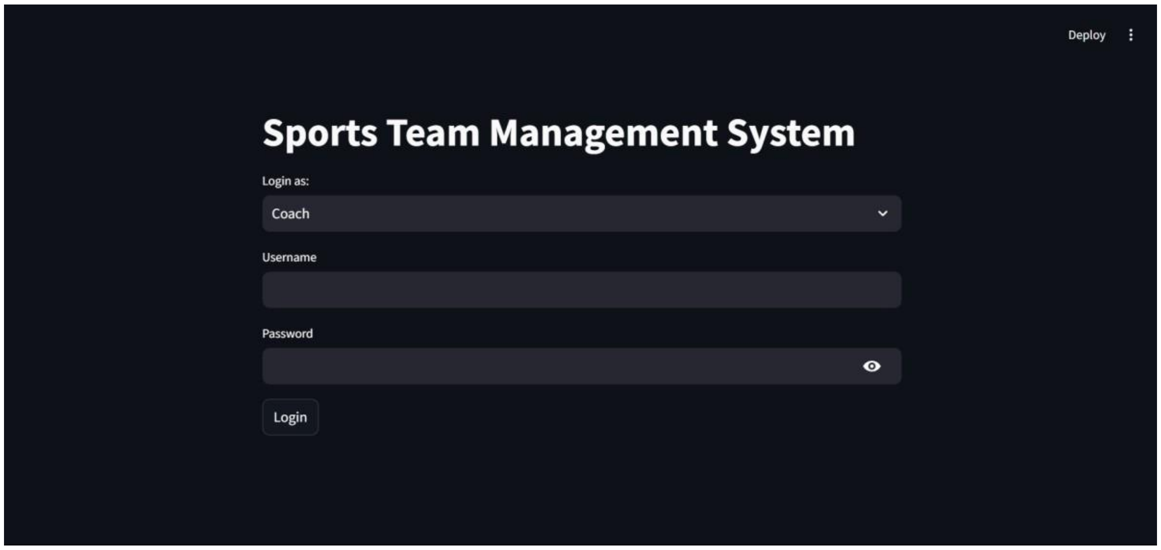
Trigger for automatic data & time update after adding injury –



```sql
CREATE TRIGGER before_injury_update
BEFORE UPDATE ON injuries
FOR EACH ROW
BEGIN
    DECLARE player_exists INT;


    SELECT COUNT(*) INTO player_exists
    FROM players
    WHERE username = NEW.player_username;


    IF player_exists = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Player username does not exist.';
    END IF;
END;
```

Screenshots

- Login Page



Sports Team Management System

Login as:

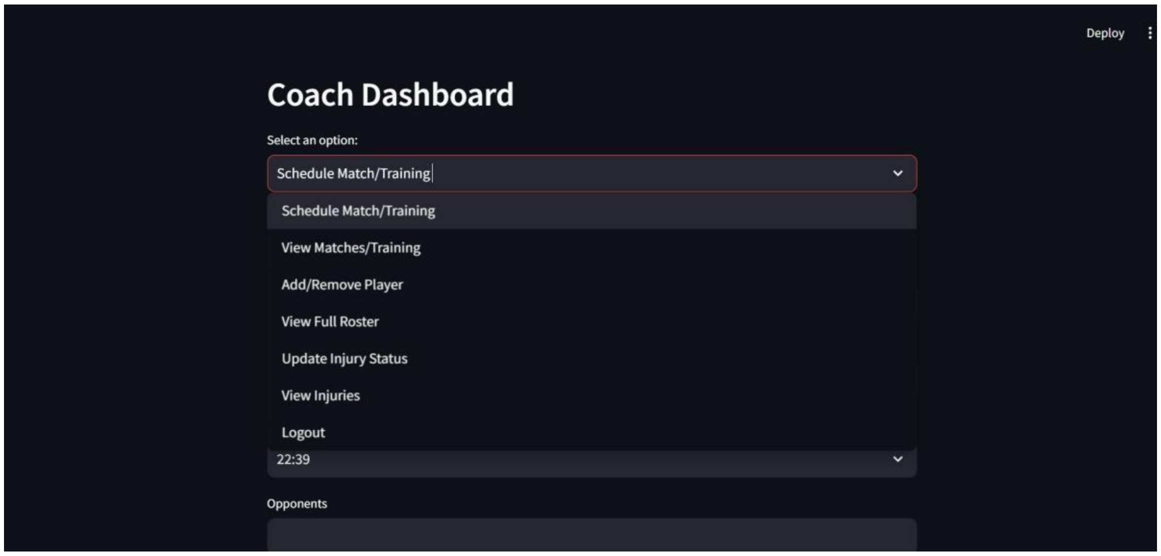
Coach

Username

Password

Login

- Dashboard



Coach Dashboard

Select an option:

Schedule Match/Training

Schedule Match/Training

View Matches/Training

Add/Remove Player

View Full Roster

Update Injury Status

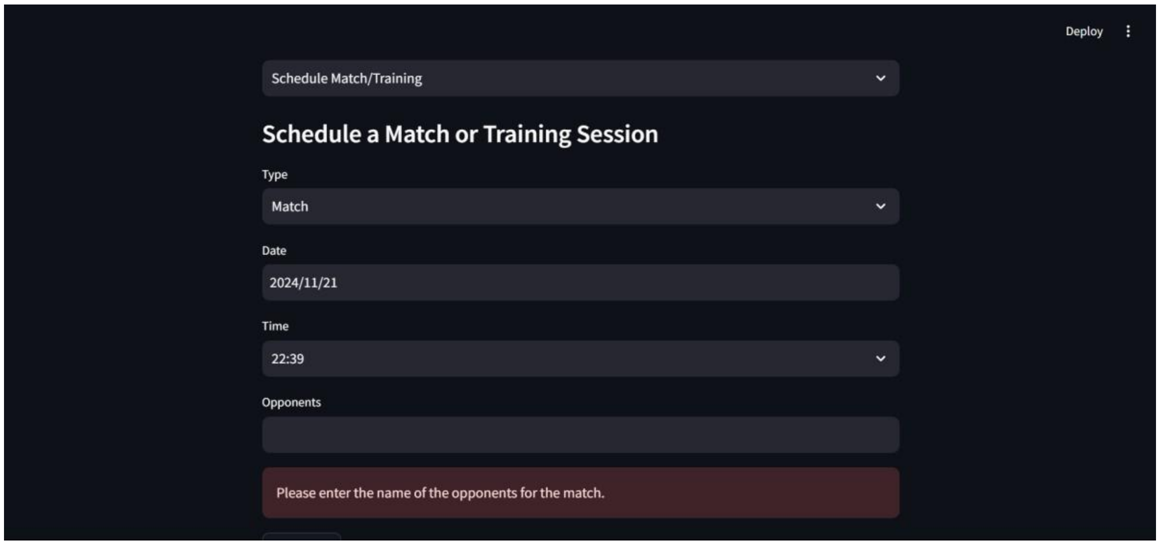
View Injuries

Logout

22:39

Opponents

- Match Schedule



Schedule a Match or Training Session

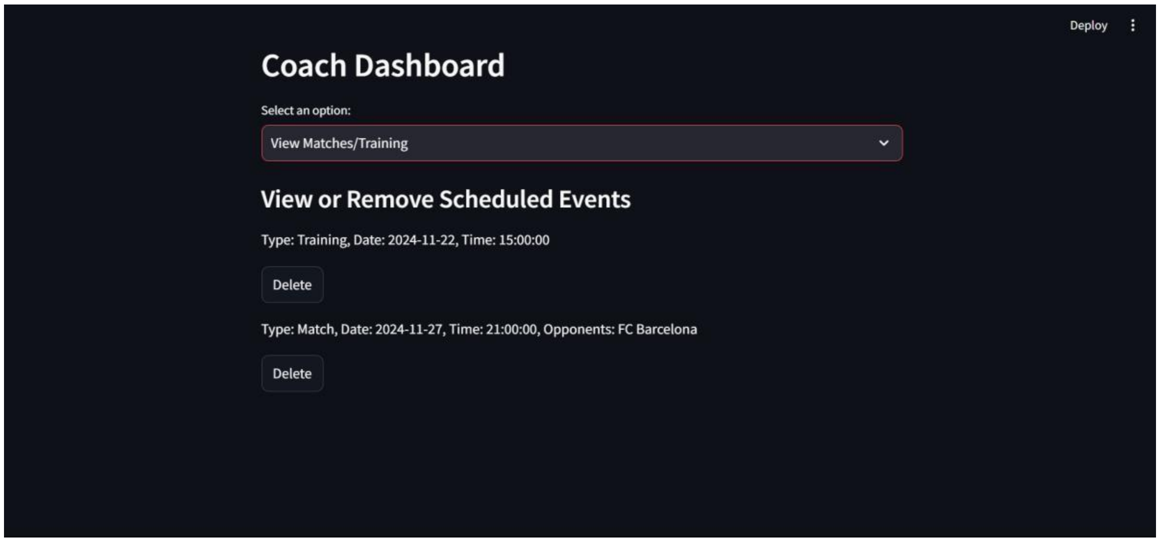
Type: Match

Date: 2024/11/21

Time: 22:39

Opponents:

Please enter the name of the opponents for the match.



Coach Dashboard

Select an option: View Matches/Training

View or Remove Scheduled Events

Type: Training, Date: 2024-11-22, Time: 15:00:00

Delete

Type: Match, Date: 2024-11-27, Time: 21:00:00, Opponents: FC Barcelona

Delete

8. Future Scope

1. Mobile Integration: Develop mobile applications for better accessibility.
2. IoT Integration: Enable real-time player performance tracking.
3. AI Analytics: Use AI to predict injuries and optimize team strategies.
4. Blockchain: Incorporate blockchain for secure contract management.

9. References

- PostgreSQL Official Documentation
- React.js and Node.js Developer Guides
- GitHub Repositories for Version Control

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Definitions

1. **Database**: An organized collection of data stored and accessed electronically.
2. **SQL (Structured Query Language)**: A programming language used for managing and manipulating relational databases.
3. **Trigger**: A database object that is automatically executed or fired when certain events occur.
4. **Function**: A reusable block of code in SQL or a programming language that performs a specific task and returns a result.
5. **API (Application Programming Interface)**: A set of protocols and tools that allows different software applications to communicate with each other.

Acronyms

1. **DDL** - Data Definition Language
2. **DML** - Data Manipulation Language
3. **API** - Application Programming Interface
4. **MVC** - Model-View-Controller

5. **ORM -** Object-Relational Mapping
6. **REST -** Representational State Transfer
7. **CRUD -** Create, Read, Update, Delete

Abbreviations

1. **ID** – Identifier
2. **DB** – Database
3. **UI -** User Interface
4. **JSON -** JavaScript Object Notation
5. **URL -** Uniform Resource Locator