SPORTS EVENT MANAGEMENT SYSTEM

A MINI PROJECT REPORT

SUBMITTED BY

AFRAH	**231501008**
AFRA FATHIMA	**231501007**
DHARSHINI VL	**231501038**
ABZAL BASHA	**231501005**

In Partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024-2025

BONAFIDE CERTIFICATE

Certified that this project report **"SPORTS EVENT MANAGEMENT SYSTEM**" is the bonafide work of **"AFRAH M (231501008), AFRA FATHIMA(231501007), DHARSHINI VL (231501038), ABZAL BASHA (231501005)"**who carried out the project work under my supervision.

Submitted for the Practical Examination held on ____________________

SIGNATURE

Mr. U. Kumaran,

Assistant Professor

AIML,

Rajalakshmi Engineering College,

(Autonomous),

Thandalam, Chennai - 602 105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The **Sports Event Management System** is a comprehensive web application designed to streamline the management of sports events, catering to both participants and organizers. This system aims to simplify the complexities of event organization, including participant registration, match scheduling, resource management, and feedback collection, all within a single platform. The website effectively differentiates between two main user roles—**participants** and **organizers**—and offers tailored functionalities based on these roles, ensuring efficient interaction through a user-friendly interface.

TABLE OF CONTENTS

1.1 INTRODUCTION

Introduction to the Sports Event Management System

In the modern era, sports and recreational activities have gained significant importance in promoting physical fitness, fostering teamwork, and providing a platform for individuals to showcase their talents. As the number of sports events and participants increases, so does the complexity involved in managing these events. From organizing matches and coordinating participants to tracking resources and gathering feedback, sports event management can become a daunting task. Manual management often leads to inefficiencies, miscommunication, and a lack of proper organization. Thus, there arises a pressing need for an automated solution to simplify and streamline the process of managing sports events. This project, the Sports Event Management System, addresses these challenges by providing a robust and user-friendly platform designed to automate and optimize the management of sports events.

Importance of Event Management in Sports

The management of sports events involves various stages, including participant registration, scheduling matches, arranging resources, and ensuring a smooth and well-coordinated execution of activities. Whether it is a small intra-college tournament or a large-scale national sports event, efficient management is key to ensuring success. In this context, sports event organizers must deal with numerous tasks such as collecting participant information, organizing teams, updating match schedules, notifying participants, and managing resources like equipment and venues.

Without an automated system, the event organizers often face issues such as:

- Data Overload: Managing large volumes of participant data manually can lead to errors in entry, duplication of information, and the potential for data loss.
- Miscommunication: Without a centralized system, it is challenging to keep participants updated about changes in match schedules, venues, or other important notifications.
- Time Management: Manually handling registrations, match scheduling, and team management can consume a lot of time, leading to delays in the overall event organization process.
- Resource Allocation: Managing sports-related equipment and other resources manually can result in shortages or over-provisioning, negatively affecting the event.

- Feedback Analysis: Gathering and analyzing feedback from participants can be tedious, resulting in missed opportunities to improve future events.

By adopting an automated system like the Sports Event Management System, many of these problems can be avoided, providing both participants and organizers with an efficient way to handle the events.

Purpose of the Project

The primary purpose of this project is to design and develop a web-based sports event management system that caters to both participants and organizers. The system aims to create a seamless flow of operations, eliminating the need for manual work and ensuring that all aspects of sports event management are handled efficiently. The platform offers an easy-to-navigate user interface that differentiates between two types of users—participants and organizers—and tailors its functionalities accordingly.

The participant module allows individuals or teams to register for sports events, view match schedules, access requirements for each sport, and submit feedback after participating in events. The organizer module, on the other hand, allows event organizers to manage the entire event, including adding match details, managing participants, and reviewing feedback submitted by the participants. By integrating both user roles into a single platform, the system ensures that all key operations are carried out smoothly and effectively.

Scope of the Project

The Sports Event Management System offers a wide range of features that cover almost every aspect of sports event management, including but not limited to:

1.1 User Authentication and Role-based Access:

- A secure login and signup system that ensures only authenticated users (participants or organizers) have access to the platform.
- Role-based access control ensures that users see only the features relevant to their role, thus enhancing security and user experience.

1.2. Participant Registration:

- Participants can register their team for a specific sport by providing necessary details such as team name, team leader information, and the sport for which they are registering.
- The system automatically generates a unique Team ID for each registration, ensuring proper tracking and management of teams.

1.3. Match Scheduling and Calendar Management:

- Organizers can add match details, such as date, time, and venue, to the system, which is then displayed to the participants via a calendar interface.
- The calendar feature not only shows the upcoming matches but also provides a countdown to the event, helping participants stay informed about their schedule.

1.4. Resource Management:

- Participants can view the equipment and other requirements needed for their specific sport.

- Organizers can update the requirements table, ensuring that participants are aware of the resources they need to bring or arrange for their matches.

1.5. Feedback System:

- After participating in an event, participants are encouraged to submit feedback through the system. This feedback is stored in a dedicated feedback table, allowing organizers to analyze and improve future events based on participant input.

1.6. Data Management and Automation:

- The system automatically manages data entries in various tables (e.g., participants, matches, feedback), ensuring data consistency and reducing the chances of errors.

- A key feature of the system is the automatic deletion of outdated match records from the database, which occurs after the match has been completed. This ensures that the database remains clean and free from obsolete data.

1.7. Scalability:

- The system is designed to handle multiple sports events simultaneously, making it suitable for small-scale intra-college competitions as well as larger inter-collegiate or national events.

Need for the Project

The need for a sports event management system arises from the growing scale and complexity of organizing sports events. With more participants, multiple events, and limited resources, manual management becomes inefficient and prone to errors. Moreover, with the increasing demand for transparency, data security, and real-time updates, it is crucial to have a digital system in place that can handle all aspects of sports event management.

For Participants, the need for a system is driven by the following factors:

- Ease of Registration: Participants often face long queues and paperwork when registering for events manually. A web-based system simplifies this process by allowing participants to register online from the comfort of their homes or on the go.
- Access to Real-time Updates: In manual systems, participants may miss out on important updates regarding match schedules or venue changes due to miscommunication. The digital system ensures that participants have access to real-time information through the match calendar.
- Transparency in Event Management: Participants want to be confident that the event is being managed fairly and transparently. The system offers a clear view of match schedules, requirements, and feedback, fostering trust among participants.

For Organizers, the system offers the following advantages:

- Efficiency in Handling Large Data: Organizing sports events involves managing a vast amount of data, from participant information to match schedules and feedback. The system automates the management of this data, ensuring that organizers can focus on delivering a successful event rather than being bogged down by administrative tasks.
- Simplified Scheduling and Resource Allocation: The system makes it easy for organizers to create match schedules, allocate resources, and manage participants. By automating these tasks, organizers can save time and avoid errors.
- Insights Through Feedback: Gathering participant feedback can be time-consuming in a manual system. The sports event management system simplifies this process by storing all feedback in one place, allowing organizers to quickly review and analyze it.

1.2 OBJECTIVES

The **Sports Event Management System** is designed to address various challenges faced by both participants and organizers in managing sports events. The system's key objectives focus on automating tasks, enhancing user experience, ensuring data accuracy, and improving overall event coordination. In this section, the project's objectives are discussed in detail, covering functional, technical, and performance-related goals that span across several core areas of the system.

1.2.1 To Provide a Seamless User Authentication and Role-Based Access Control

One of the primary objectives of the Sports Event Management System is to implement a secure and seamless user authentication process that ensures proper access control based on user roles (participant or organizer). The authentication process includes the following sub-objectives:

- **Account Creation & Verification**: During the signup process, users (both participants and organizers) are required to provide essential information such as their email and phone number, which are verified to prevent unauthorized access. This ensures that only valid users can create accounts on the platform.
- **Role Assignment**: The system differentiates between two user roles—participants and organizers. This role-based access control ensures that users are presented with the features and functionalities relevant to their specific role. For example, participants will have access to registration, match schedules, and feedback submission, while organizers will have access to event creation, participant management, and feedback review.
- **Login & Logout Mechanisms**: The system provides a robust login mechanism with encrypted passwords and proper session management to ensure that users can securely log in and log out without compromising data integrity.
- **Enhanced Security**: Role-based access ensures that sensitive functions (like adding matches or managing participants) are restricted to authorized organizers, while participants have limited access to their relevant data only. This ensures security, privacy, and data protection.

1.2.2. To Develop an Automated Registration System for Participants

A core functionality of the Sports Event Management System is to simplify the registration process for participants. The system aims to automate and streamline the registration process, allowing participants to register for events conveniently while ensuring the data is accurately captured and stored in the database. The key objectives related to participant registration include:

- **User-friendly Registration Form**: The registration interface should be intuitive and easy to navigate, allowing participants to input details such as Team Name, Leader Information (Name, Age, Gender, Department), and the selected sport. These inputs are mapped to the relevant fields in the database (Teams table) for future reference.
- **Auto-generation of Team ID**: Upon registration, the system automatically generates a unique Team ID for each registered team. This ensures that teams are uniquely identified, and no duplication occurs in the registration records.
- **Data Validation**: The registration system should include built-in validation rules that ensure data accuracy. For example, the form should verify that all required fields are completed, email formats are valid, and numbers (like phone numbers and age) are entered correctly.
- **Efficient Storage in the Database**: Once the participant has submitted their registration details, the system ensures that the data is securely stored in the **Teams** table of the PostgreSQL database, which will later be used for managing team-related operations such as match scheduling and feedback collection.
- **Edit & Update Feature**: Participants should be allowed to edit or update their registration details before the event commences. For example, if the team leader's information or the sport category needs to be updated, the system should accommodate these changes efficiently.

1.2.3. To Facilitate Efficient Match Scheduling and Calendar Management

Effective match scheduling is one of the most important aspects of managing sports events. The Sports Event Management System aims to simplify the process of creating, updating, and displaying match schedules for both participants and organizers. The key objectives related to match scheduling include:

- **Match Scheduling by Organizers**: Organizers can add new matches to the system by specifying the sport name, match date, time, and venue. This information is stored in the **Matches** table of the PostgreSQL database.
- **Real-time Match Calendar for Participants**: Participants should be able to view their upcoming matches via a dedicated calendar interface. The calendar should display relevant match details (sport name, date, time, and venue), fetched directly from the **Matches** table.
- **Countdown to Events**: The system should include a feature that calculates how much time is left until the next match, displayed alongside the match details. This countdown feature will help participants plan accordingly and stay informed about their event schedules.

- **Automatic Deletion of Past Matches**: Once a match has concluded, the system should automatically delete outdated match records from the **Matches** table to prevent cluttering the database with irrelevant information. This helps keep the match calendar current and accurate.

1.2.4. To Provide a Comprehensive Resource Management System

In addition to managing participants and matches, the system is designed to manage resources efficiently. Different sports require different equipment or materials, and participants need to be informed of what is required for their specific events. The objectives related to resource management include:

- **Dynamic Resource Display**: The system should dynamically fetch and display the required materials or equipment for each sport from the **Requirements** table. This ensures that participants have the necessary information about what they need to bring for their matches.
- **Efficient Update Mechanism**: Organizers should have the ability to add or update the required resources for different sports at any point during the event preparation. These updates should reflect in real-time for participants, ensuring they are always informed of the latest requirements.
- **Resource Planning and Allocation**: By ensuring that participants know what is required for each sport, the system helps prevent last-minute shortages of resources and improves overall event preparation.

1.2.5. To Implement a Participant Feedback System

Feedback is an essential component of any sports event, as it helps organizers gather insights into the event's success and identify areas for improvement. The Sports Event Management System aims to implement a comprehensive feedback system that allows participants to submit their opinions about the event, which can later be analyzed by organizers. The objectives related to feedback include:

- **Feedback Form for Participants**: After participating in a match, participants are given the opportunity to submit feedback via a simple and intuitive form. The form allows them to rate their experience and provide comments about their event experience.
- **Feedback Storage**: The system securely stores all feedback in the **Feedback** table of the PostgreSQL database, ensuring that it can be accessed by organizers for review.

- **Analysis and Insights**: Organizers can review the feedback submitted by participants to identify patterns or recurring issues. This helps organizers make data-driven decisions and improve future sports events based on participant experiences.
- **Anonymous Feedback Option**: For participants who may prefer anonymity, the system could provide the option to submit feedback without attaching their personal details, encouraging more honest and candid responses.

1.2.6. To Ensure Scalability and Flexibility for Event Management

The Sports Event Management System is designed to be scalable and flexible, capable of handling events of varying sizes and complexities. Whether it is a small intra-college tournament or a large-scale inter-collegiate event, the system must be able to accommodate different requirements. The objectives related to scalability include:

- **Multi-Sport Support**: The system should be flexible enough to handle multiple sports simultaneously. Each sport will have its own matches, participants, and resources, and the system should be able to manage all of these aspects efficiently.
- **Handling Large Volumes of Data**: As the number of participants, teams, and matches grows, the system should be able to handle large volumes of data without slowing down or becoming unresponsive. PostgreSQL is chosen as the database management system due to its ability to handle complex queries and large datasets efficiently.
- **Customization for Future Events**: The system should be built in such a way that it can easily be adapted for different types of sports events in the future. This may involve adding new sports, changing the registration fields, or modifying match scheduling rules.

1.2.7. To Deliver a User-Friendly and Responsive Interface

A key objective of the Sports Event Management System is to provide an intuitive and responsive user interface that enhances the user experience for both participants and organizers. The system's interface should be easy to navigate, providing users with clear instructions and ensuring that they can complete tasks efficiently. The objectives related to the user interface include:

- **Responsive Design**: The system's interface should be responsive, ensuring that it works well on a variety of devices, including desktops, laptops, tablets, and mobile phones. This flexibility ensures that participants and organizers can access the system from anywhere, on any device.

- **Clear Navigation**: The platform should offer clear and intuitive navigation, with participants and organizers able to easily find the features relevant to them. For example, participants should be able to quickly access the registration form, match calendar, and feedback section, while organizers should have easy access to event management tools like match creation and participant viewing.
- **Error Handling & User Support**: The system should include user-friendly error messages and validation checks that guide users through the process without confusion. Additionally, a help section or user support feature could be added to assist users with any technical difficulties they may encounter while using the platform.

1.3 MODULES

1.3.1 Authentication Module

This module handles user authentication, registration, and access control. It distinguishes between participants and organizers and ensures that only authorized users can access the system's features.

Submodules:

- **Login Module**: Allows existing users to log in using their username and password.
- **Signup Module**: Provides a form for new users to sign up by entering email, phone number, username, password, and selecting a role (participant or organizer).
- **Verification Module**: Verifies the phone number and email during signup to authenticate the user.
- **Role-Based Access Control**: Assigns specific functionalities and interfaces depending on whether the user is a participant or organizer.

1.3.2. Participant Module

This module contains all the functionalities designed for participants, including registration for events, viewing match schedules, accessing requirements, and submitting feedback.

Submodules:

- **Registration Module**:
  - Provides a form for participants to register their team by entering details such as Team Name, Leader Information, and the selected sport.
  - Automatically generates a Team ID.
  - Stores registration data in the **Teams** table.
- **Match Calendar Module**:
  - Displays upcoming matches with details such as date, time, and venue.
  - Fetches data from the **Matches** table and shows participants how much time is left for the next match.
- **Requirements Module**:
  - Displays the required materials or equipment for the sport the participant has registered for.
  - Fetches data from the **Requirements** table.

- **Feedback Module**:
    - Allows participants to submit feedback after participating in a match.
    - Stores feedback in the **Feedback** table.
    - Optional feature: Enables anonymous feedback submission.

1.3.3. Organizer Module

This module contains functionalities specifically for organizers, including adding matches, viewing participants, and reviewing feedback.

Submodules:

- **Match Management Module**:
    - Allows organizers to add new matches by providing sport name, match time, date, and venue.
    - Updates the **Matches** table with the new match data.
    - Includes automatic deletion of past matches once they are completed.
- **Participant Management Module**:
    - Displays a list of registered participants along with their details (Team ID, Team Name, Leader Name, etc.).
    - Allows organizers to delete participants if necessary, updating the **Teams** table accordingly.
- **Feedback Review Module**:
    - Displays the feedback submitted by participants.
    - Fetches and displays feedback from the **Feedback** table for analysis and review.

1.3.4. Match Scheduling & Calendar Module

This module manages the scheduling of matches and provides both participants and organizers with a calendar view of upcoming events. It pulls data from the **Matches** table and displays information based on the user's role.

Submodules:

- **Match Creation Module** (Organizer):
    - Allows organizers to add new matches to the calendar.
    - Organizers provide match details such as sport, date, time, and venue.
- **Match View Module** (Participant):

    - Displays upcoming matches for registered participants, filtering based on the sport they are participating in.
    - Includes a countdown timer showing the time remaining until the match.

1.3.5. Resource/Requirements Management Module

This module ensures that participants have access to the list of resources or equipment required for each sport. The organizer can modify these requirements based on the event's needs.

Submodules:

- **Requirements View Module** (Participant):
    - Displays required resources based on the participant's registered sport, fetched from the **Requirements** table.
- **Requirements Update Module** (Organizer):
    - Allows organizers to add or update the required resources for each sport.

1.3.6. Feedback Management Module

This module allows participants to submit feedback and provides organizers with access to view and analyze the feedback.

Submodules:

- **Feedback Submission Module** (Participant):
    - Participants can provide feedback via a simple form.
    - The feedback is stored in the **Feedback** table.
- **Feedback Review Module** (Organizer):
    - Allows organizers to view feedback submitted by participants, helping them gather insights on the event.

1.3.7. Database Management Module

This module handles all the interactions with the PostgreSQL database. It ensures that data is stored, updated, and retrieved efficiently, based on the operations performed by participants and organizers.

Submodules:

- **User Management Submodule**: Manages data in the **Users** table, including user creation, login authentication, and role assignment.
- **Team Management Submodule**: Manages data in the **Teams** table, including storing participant registrations and editing participant details.
- **Match Management Submodule**: Handles match scheduling and automatic deletion of past matches from the **Matches** table.
- **Requirements Management Submodule**: Manages data in the **Requirements** table, ensuring that the necessary resources are available and up-to-date.
- **Feedback Management Submodule**: Stores and retrieves feedback from the **Feedback** table.

1.3.8. Notification and Reminder Module

This optional module can be implemented to send notifications and reminders to participants and organizers regarding upcoming events, registration deadlines, or important updates.

Submodules:

- **Match Reminder Submodule**: Sends reminders to participants about their upcoming matches.
- **Event Update Submodule**: Notifies participants and organizers of any changes to match schedules or venue updates.

1.3.9. User Interface (UI) Module

The User Interface module is responsible for creating a user-friendly, responsive, and intuitive interface for both participants and organizers. It integrates HTML, CSS, and JavaScript to ensure an engaging user experience.

Submodules:

- **Responsive Design Submodule**: Ensures that the interface adapts to various devices, including desktops, tablets, and smartphones.
- **Error Handling Submodule**: Provides user-friendly error messages and validation checks to guide users through forms and processes.
- **Navigation Submodule**: Implements easy navigation between different pages, such as registration, match calendar, and feedback.

2. SURVEY OF TECHNOLOGIES

Survey of Technologies for the Sports Event Management System

In developing the Sports Event Management System, various technologies play a crucial role in ensuring the system is robust, user-friendly, and efficient. This section will elaborate on the technologies utilized in different layers of the system, their significance, and their contribution to achieving the project goals.

2.1. Frontend Technologies

The frontend of the Sports Event Management System is crucial for creating an intuitive user interface that allows users to interact seamlessly with the application. The following technologies are employed in the frontend development:

2.1.1 HTML (HyperText Markup Language)

- **Overview**: HTML is the standard markup language used to create the structure of web pages. It defines the layout and content of the webpage, enabling developers to format text, embed images, and create links.
- **Importance**: For this project, HTML forms the backbone of the user interface, allowing participants and organizers to access various functionalities such as login, registration, and feedback forms.

2.1.2 CSS (Cascading Style Sheets)

- **Overview**: CSS is used to style HTML elements, controlling the visual presentation of web pages, including layout, colors, fonts, and overall aesthetics.
- **Importance**: CSS enhances the user experience by ensuring that the application is visually appealing and responsive across different devices. It allows for a clean and professional look, which is essential for user engagement.

2.1.3 JavaScript (Optional)

- **Overview**: JavaScript is a programming language that enables interactive features on websites, such as form validation, dynamic content updates, and event handling.

- **Importance**: While not mandatory for basic functionalities, JavaScript can be integrated to improve user experience through features like real-time form validation, enhancing the overall responsiveness of the application.

2.2. Backend Technologies

The backend layer is responsible for processing user requests, managing business logic, and interacting with the database. The primary technology used in this layer is:

2.2.1 PHP (Hypertext Preprocessor)

- **Overview**: PHP is a widely-used open-source server-side scripting language designed for web development. It can be embedded within HTML to create dynamic web pages.
- **Importance**: In this project, PHP is used to handle user authentication, process form submissions, manage data operations (like adding matches or retrieving feedback), and facilitate communication between the frontend and the database. Its ease of use and extensive community support make it an excellent choice for this application.

2.2.3. Database Technologies

Data management is a critical aspect of any web application. The Sports Event Management System utilizes a relational database to store and manage data efficiently.

2.3. PostgreSQL

- **Overview**: PostgreSQL is a powerful, open-source relational database management system known for its robustness, scalability, and adherence to SQL standards.
- **Importance**: In this project, PostgreSQL is used to create and manage various tables, including Users, Teams, Matches, Requirements, and Feedback. Its features, such as transaction support, data integrity, and complex querying capabilities, make it suitable for handling the diverse data needs of the system. Additionally, the use of pgAdmin4, a graphical user interface for managing PostgreSQL databases, simplifies the database management process.

2.4. Additional Technologies and Tools

In addition to the core technologies mentioned above, several other tools and technologies may be integrated into the project to enhance functionality and performance.

2.4.1 Version Control System (Git)

- **Overview**: Git is a version control system that tracks changes in code, allowing developers to collaborate and manage code versions effectively.
- **Importance**: Utilizing Git for version control ensures that the development team can work collaboratively without conflicts, maintain a history of code changes, and revert to previous versions if necessary.

2.4.2 Web Hosting Services

- **Overview**: Web hosting services provide the infrastructure required to deploy web applications on the internet.
- **Importance**: A reliable hosting service is essential for making the Sports Event Management System accessible to users. It enables the backend server (running PHP and connecting to PostgreSQL) to process requests and serve the frontend to users.

2.4.3 Testing Tools

- **Overview**: Testing tools help ensure the application is functional, secure, and performs well under various conditions.
- **Importance**: Automated testing frameworks can be used to test individual components and overall functionality, ensuring that the application meets user expectations and requirements.

2.1 SOFTWARE DESCRIPTION

The Sports Event Management System is a web-based application designed to facilitate the organization and participation in sports events. It provides a platform for both participants and organizers to manage various aspects of sports events efficiently. This section provides a detailed description of the software components, their functionalities, and how they interact to deliver a seamless user experience.

2.1.1. Software Architecture

The software architecture of the Sports Event Management System is structured into three main layers:

- **Frontend Layer**: This layer is responsible for the user interface and user experience. It includes all the components that users interact with directly, such as forms, buttons, and navigation menus.
- **Backend Layer**: The backend layer manages business logic, handles data processing, and communicates with the database. It acts as the intermediary between the frontend and the database, processing user requests and returning responses.
- **Database Layer**: This layer is responsible for data storage and management. It houses all the necessary data related to users, teams, matches, requirements, and feedback.

2.1.2. Software Components

The Sports Event Management System consists of several key software components, each serving a specific function. Below is a detailed description of each component:

2.1.3 Frontend Components

- **User Interface (UI)**:
  - The UI is built using **HTML** and **CSS** to provide a visually appealing and user-friendly layout. It includes pages for login, registration, match calendars, requirements display, and feedback submission.
  - The design follows a responsive approach to ensure compatibility across different devices, such as desktops, tablets, and mobile phones.
- **Login and Signup Forms**:

    - The login form allows users to enter their credentials (username and password) for authentication. If the user is new, the signup form collects necessary information, including email, phone number, and role (participant or organizer).
- **Registration Form**:
    - Participants can register their teams by providing details such as Team ID, Leader Name, Age, Gender, Department, Sport Name, and Team Name. This information is submitted through a form and sent to the backend for processing.
- **Match Calendar**:
    - The match calendar displays upcoming matches relevant to the participant's registered sports. It provides details such as match date, time, venue, and countdown to the event.
- **Requirements Display**:
    - This component fetches and displays the necessary equipment and materials required for specific sports from the database, helping participants prepare adequately.
- **Feedback Submission Form**:
    - Participants can provide feedback regarding the events they attended. This feedback is essential for organizers to assess the quality of events and make improvements.

2.2 Backend Components

- **User Authentication**:
    - The backend handles user authentication by verifying the credentials entered in the login form against the data stored in the Users Table. It uses secure password hashing to protect user data.
- **Data Processing**:
    - Upon receiving requests from the frontend, the backend processes the data accordingly. This includes registering teams, scheduling matches, fetching match details, and submitting feedback.
- **Business Logic Implementation**:
    - The backend implements business logic to ensure the application functions correctly. For instance, it checks for duplicate usernames during signup, validates input data, and manages data integrity when updating or deleting records.
- **API Endpoints**:
    - The backend exposes various API endpoints to handle requests from the frontend. These endpoints facilitate operations like user registration, team registration, match scheduling, and feedback submission.

2.2.1 Database Components

- **Database Management System (DBMS)**:
  - The Sports Event Management System uses **PostgreSQL** as its relational database management system. PostgreSQL provides features like data integrity, concurrency control, and robust querying capabilities.
- **Data Tables**:
  - **Users Table**: Stores user information, including username, password (hashed), email, phone number, and user role (participant or organizer).
  - **Teams Table**: Contains details about registered teams, including leader information and sport name.
  - **Matches Table**: Holds information about scheduled matches, including sport name, date, time, and venue.
  - **Requirements Table**: Lists the equipment and material requirements for each sport.
  - **Feedback Table**: Stores feedback submitted by participants regarding the events

2.3. User Roles and Functionalities

The system differentiates between two primary user roles: **Participants** and **Organizers**, each with distinct functionalities.

2.3.1 Participants

- **Account Creation**: Participants can create an account through the signup process, providing necessary details for verification and future logins.
- **Team Registration**: Once logged in, participants can register their teams by filling out the registration form with relevant details.
- **View Matches**: Participants can access the match calendar to view upcoming matches, including time, date, venue, and countdown to the event.
- **Access Requirements**: Participants can view the requirements for their registered sport, helping them prepare for matches.
- **Submit Feedback**: After attending events, participants can provide feedback to the organizers, which helps in assessing the quality of the events.

2.3.2 Organizers

- **Account Management**: Organizers also create accounts through the signup process but have additional privileges.
- **Add Matches**: Organizers can schedule new matches by providing details such as sport name, date, time, and venue. This information is stored in the Matches Table.
- **View Participants**: Organizers can access a list of registered participants and their details. They can remove participants if necessary.
- **View Feedback**: Organizers can view feedback submitted by participants, allowing them to identify areas for improvement and enhance future events.

2.3.4. Security Features

The Sports Event Management System implements various security measures to protect user data and ensure safe interactions:

- **Data Encryption**: User passwords are hashed using secure algorithms to prevent unauthorized access to sensitive information.
- **Input Validation**: The system performs input validation to ensure that users provide correct and expected data formats, reducing the risk of SQL injection and other attacks.
- **Session Management**: The application manages user sessions securely, ensuring that users remain authenticated throughout their interaction with the system.

2.3.5. Technological Stack Summary

The following technologies are utilized in the Sports Event Management System:

- **Frontend**: HTML, CSS, and optional JavaScript for enhanced interactivity.
- **Backend**: PHP for server-side scripting and business logic implementation.
- **Database**: PostgreSQL for data storage and management.
- **Version Control**: Git for code management and collaboration among developers.
- **Web Hosting**: A reliable hosting service to deploy the application and make it accessible to users.

2.2 LANGUAGES

Languages Used in the Sports Event Management System

In developing the Sports Event Management System, several programming languages are employed to build a robust, efficient, and user-friendly web application. Each language serves a specific purpose, contributing to the overall functionality and performance of the system. This section will elaborate on the languages utilized in various layers of the application, highlighting their roles, advantages, and how they work together to achieve the project's objectives.

2.2.1. Frontend Languages

The frontend is the part of the application that users interact with directly. It is responsible for the presentation layer and user experience. The main languages used in the frontend development of the Sports Event Management System are:

2.2.1.1 HTML (HyperText Markup Language)

- **Overview**: HTML is the foundational markup language for creating web pages. It provides the structure and layout of the content on the website.
- **Usage in the Project**:
  - HTML is used to create various elements of the user interface, such as forms for login, signup, and feedback submission.
  - It allows the organization of content into headings, paragraphs, lists, and links, which are essential for guiding users through the application.
- **Advantages**:
  - **Simplicity**: HTML is easy to learn and use, making it accessible for developers.
  - **Standardization**: As a standardized language, HTML ensures compatibility across different web browsers, enhancing the user experience.

2.2.1.2 CSS (Cascading Style Sheets)

- **Overview**: CSS is a stylesheet language used to describe the presentation and layout of HTML elements. It controls visual aspects like colors, fonts, spacing, and overall design.
- **Usage in the Project**:
  - CSS is used to style the web pages, ensuring a visually appealing layout. This includes the arrangement of buttons, forms, and navigation elements.

    - It allows for responsive design, making the application adaptable to various screen sizes (desktops, tablets, and mobile devices).
- **Advantages**:
    - **Separation of Content and Style**: CSS enables developers to keep the content (HTML) and presentation (styles) separate, making maintenance and updates easier.
    - **Enhanced User Experience**: Well-styled interfaces lead to improved user engagement and satisfaction.

2.2.1.3 JavaScript (Optional)

- **Overview**: JavaScript is a dynamic, high-level programming language that enables interactive features on web pages. It can manipulate HTML and CSS to create a more engaging user experience.
- **Usage in the Project**:
    - While not strictly necessary for the basic functionalities of the system, JavaScript can enhance user experience by implementing features like form validation, real-time updates, and dynamic content loading.
- **Advantages**:
    - **Interactivity**: JavaScript allows for the creation of interactive elements, such as sliders, modals, and responsive navigation menus.
    - **Client-side Processing**: By processing certain tasks on the client-side, JavaScript reduces the load on the server, leading to faster response times

2.2.2. Backend Language

The backend is responsible for managing the business logic, data processing, and interaction with the database. The primary language used in the backend development of the Sports Event Management System is:

2.2.2.1 PHP (Hypertext Preprocessor)

- **Overview**: PHP is a server-side scripting language designed for web development. It can be embedded within HTML to create dynamic web pages.
- **Usage in the Project**:
    - PHP handles user authentication, processing login and signup requests, and storing user information in the database.

    - It manages data operations related to team registration, match scheduling, and feedback submission.
    - PHP interacts with the PostgreSQL database, executing SQL queries to retrieve, insert, update, and delete data as needed.
- **Advantages**:
    - **Ease of Integration**: PHP integrates easily with HTML, allowing developers to create dynamic web applications without much overhead.
    - **Strong Community Support**: Being widely used, PHP has a vast community and numerous resources, making it easier to find solutions to common issues.
    - **Extensive Libraries**: PHP provides various libraries and frameworks (such as Laravel and CodeIgniter) that facilitate rapid development and enhance functionality.

2.2.3. Database Language

The database is essential for storing and managing data in the application. The primary language used for database interactions in the Sports Event Management System is:

2.2.3.1 SQL (Structured Query Language)

- **Overview**: SQL is a standardized language used for managing and manipulating relational databases. It allows for querying, updating, and managing data within the database.
- **Usage in the Project**:
    - SQL is used to create the database schema, defining the structure of tables (Users, Teams, Matches, Requirements, Feedback) and their relationships.
    - SQL queries are executed through PHP to perform operations such as inserting new records, retrieving existing data, updating information, and deleting outdated records.
- **Advantages**:
    - **Data Manipulation**: SQL provides powerful commands for performing complex queries and data analysis.
    - **Data Integrity**: SQL allows for defining constraints and relationships between tables, ensuring data integrity and consistency.
    - 4. Scripting and Markup Languages

In addition to the main languages, the project may also utilize additional languages for specific functionalities:

2.2.1 SQL

SQL in the Sports Event Management System

SQL (Structured Query Language) is a standardized programming language specifically designed for managing and manipulating relational databases. In the context of the Sports Event Management System, SQL plays a vital role in defining the database structure, performing data operations, and ensuring data integrity and security. This section elaborates on the role of SQL within the project, its advantages, and how it facilitates various functionalities.

2.2.1.1. Overview of SQL

- **Definition**: SQL is the language used to communicate with relational databases. It enables users to create, read, update, and delete data, often abbreviated as CRUD operations.
- **Types of SQL Statements**:
  - **DDL (Data Definition Language)**: Used to define and manage database structures (e.g., CREATE, ALTER, DROP).
  - **DML (Data Manipulation Language)**: Used for inserting, updating, and deleting data (e.g., INSERT, UPDATE, DELETE).
  - **DQL (Data Query Language)**: Used for querying data from the database (e.g., SELECT).
  - **DCL (Data Control Language)**: Used to control access to data (e.g., GRANT, REVOKE).

2.2.1.2. Database Structure

The Sports Event Management System relies on a well-structured database to store and manage data effectively. The database typically consists of several tables, each serving a distinct purpose. The primary tables in the project include:

2.2.2.1 Users Table

- **Purpose**: Stores information about all users of the system, including both participants and organizers.
- **Columns**:
  - user_id: Unique identifier for each user (Primary Key).
  - username: User's chosen username.

  - password: Hashed password for authentication.
  - email: User's email address for communication.
  - phone: User's contact number.
  - role: Specifies whether the user is a participant or organizer.

2.2.2.2 Teams Table

- **Purpose**: Contains information about teams registered by participants.
- **Columns**:
  - team_id: Unique identifier for each team (Primary Key, auto-generated).
  - leader_name: Name of the team leader.
  - leader_age: Age of the team leader.
  - leader_gender: Gender of the team leader.
  - leader_dept: Department of the team leader.
  - sport_name: Name of the sport the team is registered for.
  - team_name: Official name of the team.

2.2.2.3 Matches Table

- **Purpose**: Holds information about scheduled matches.
- **Columns**:
  - match_id: Unique identifier for each match (Primary Key, auto-generated).
  - sport_name: Name of the sport for the match.
  - match_time: Scheduled time for the match.
  - match_date: Scheduled date for the match.
  - match_venue: Venue where the match will take place.

2.2.2.4 Requirements Table

- **Purpose**: Lists the equipment or materials required for different sports.
- **Columns**:
  - requirement_id: Unique identifier for each requirement (Primary Key).
  - sport_name: Name of the sport.
  - requirements: Description of the required items.

2.2.2.5 Feedback Table

- **Purpose**: Captures feedback from participants about their experience in events.

- **Columns**:
    - feedback_id: Unique identifier for each feedback entry (Primary Key).
    - sport_name: Name of the sport related to the feedback.
    - feedback: Text of the participant's feedback.

2.2.3. SQL Operations in the Project

SQL operations are crucial for maintaining the integrity and functionality of the Sports Event Management System. Here are some key operations performed using SQL in the project:

2.2.3.1 User Registration and Authentication

- **Insert User Data**: When a new user signs up, an INSERT SQL statement adds their information to the Users table.
  sql
  Copy code
  INSERT INTO Users (username, password, email, phone, role)
  VALUES ('john_doe', 'hashed_password', 'john@example.com', '1234567890', 'participant');

- **User Login**: To authenticate users, a SELECT statement retrieves user information based on the provided username and password.
  sql
  Copy code
  SELECT * FROM Users WHERE username = 'john_doe' AND password = 'hashed_password';

2.2.3.2 Team Registration

- **Insert Team Data**: Participants can register their teams by inserting their details into the Teams table.
  sql
  Copy code
  INSERT INTO Teams (leader_name, leader_age, leader_gender, leader_dept, sport_name, team_name)
  VALUES ('Jane Doe', 25, 'Female', 'Science', 'Basketball', 'Team A');

2.2.3.3 Match Scheduling

- **Add Match**: Organizers can schedule matches by inserting data into the Matches table.
  sql
  Copy code
  INSERT INTO Matches (sport_name, match_time, match_date, match_venue)
  VALUES ('Basketball', '15:00', '2024-10-10', 'Gymnasium');

- **Retrieve Matches**: Participants can view upcoming matches using a SELECT query.
  sql
  Copy code
  SELECT * FROM Matches WHERE match_date >= CURRENT_DATE ORDER BY match_date;

2.2.3.4 Requirement Management

- **View Requirements**: Participants can query the Requirements table to see what is needed for their sport.
  sql
  Copy code
  SELECT * FROM Requirements WHERE sport_name = 'Basketball';

2.2.3.5 Feedback Submission

- **Insert Feedback**: Participants can submit feedback, which is recorded in the Feedback table.
  sql
  Copy code
  INSERT INTO Feedback (sport_name, feedback)
  VALUES ('Basketball', 'Great event, well organized!');

- **Retrieve Feedback**: Organizers can view all feedback for analysis.
  sql
  Copy code
  SELECT * FROM Feedback WHERE sport_name = 'Basketball';

2.2.4. Data Integrity and Security

Maintaining data integrity and security is crucial in any application. SQL provides mechanisms to ensure this:

2.2.4.1 Constraints

- **Primary Keys**: Each table has a primary key (e.g., user_id, team_id, etc.) to uniquely identify records and prevent duplication.
- **Foreign Keys**: Relationships between tables can be established using foreign keys, which maintain referential integrity.

2.2.4.2 Data Security

- **Parameterized Queries**: To prevent SQL injection attacks, parameterized queries or prepared statements should be used, ensuring that user input is safely handled.

2.2.5. SQL Performance Optimization

To ensure efficient data handling, performance optimization techniques may be employed:

2.2.5.1 Indexing

- Creating indexes on frequently queried columns (e.g., sport_name, match_date) can significantly enhance query performance.

- Writing efficient SQL queries and analyzing their execution plans can help identify and eliminate bottlenecks.

2.2.2 PHP

2.2.2.1.Introduction to PHP

PHP is an open-source, server-side scripting language powerful enough to create dynamic web pages. It originated in 1994 by Rasmus Lerdorf, and he further expanded it into one of the leading languages in web development. It directly embeds into HTML code, which is especially good for dynamic websites and web applications where the content and the associated data need to be constantly updated or modified.

Using PHP, developers can actually merge front-end development with backend databases along with server-side processing and user interactive interfaces at runtime. The language is made to handle everything from successful form submission, session management, authentication of user identities, and many others. As such, it can be used in many kinds of web projects, for instance, content management systems, e-commerce platforms, and even event management systems like your sports event project.

The PHP script is processed on the server even before it gets presented to the user's browser for use. This means that the content seen by the user will be dynamically generated according to the inputs, making it appropriate for interactive applications.

2.2.2.2. Why PHP is a Good Choice for Your Sports Event Management System

Your sport event management system will have the following features: participant registration, team management, possibility to add feedback, and scheduling matches. PHP is very well-suited for this type of project because it very easily unites server-side logic with front-end logic, which helps to create interactive web pages. Now let's get into more detail on how PHP is good for this mini-project:

2.2.2.3 Good integration of PHP with HTML and CSS:

PHP was created to work in conjunction with HTML; it makes it very easy to embed dynamic content into static web pages. In your sports event management system:

- HTML structures your front-end forms (like registration forms or feedback submission).
- CSS is focused on making the design user-friendly and pleasing to the eyes.
- PHP runs the form inputs, interacts with your database, and returns a result dynamically.

The fact that PHP is embedded means you can put your dynamic code directly into your HTML. With this, the development process will be more intuitive and faster especially for smaller projects like yours. It's more that you get to avoid complexity for it is where you keep your front-end layout and back-end logic in one place.
For example,

```
<h1>Welcome, <?php echo $_SESSION['username']; ?></h1>
```

In this line, PHP outputs dynamically the username of the logged-in participant. Such flexibility makes PHP a perfect choice for creating user-specific content within a web application.

2.2.2.4 Simple Database Integration:

PHP supports databases intensively. You can interact with databases like MySQL, PostgreSQL, SQLite, and others. Your sports event system uses PostgreSQL to store data such as participant information, team registrations, and feedback.

PHP offers a set of functions in the style of \`pg_connect()\`, \`pg_query()\`, and \`pg_query_params()\` that can be used directly to provide easy
- Insert data: for example, record participants or teams.
- Fetch data: like for viewing registered participants or upcoming matches.
- Update or delete data: such as team changes or cancellations of the event.

Unlike this, other languages are actually dependent on Object-Relational Mappers (ORMs) such as Django's ORM. This has the potential to add a layer of abstraction. With PHP you can literally write raw SQL queries, which gives one more control over their code and a quicker route to implementation.

Here is how the PHP handles the SQL query in your registration process by passing it into the database:

```
$insert_query = "INSERT INTO teams (leader_id, leader_name, sport_name) VALUES ($1, $2,
$3)";

$insert_result = pg_query_params($conn, $insert_query, array($leader_id, $leader_name,
$sport_name));
```

This simplicity is one of the greatest advantages PHP has when working on projects like your event management system, where you might need to interact directly, quickly with your database.

2.2.2.5 Built-in Session Management and User Authentication:

Session management is crucial to tracking users as they navigate through different pages of your site. PHP natively supports session handling functions `$_SESSION` that enable you to:
Log the user state: You can easily store user information like `user_id` and `username` over several pages.
Security: PHP ensures passwords are hashed securely in the database using secure hashing functions (`password_hash()` and `password_verify()`:.
Session management is an integral part of your sports event application. For example, once a user has authenticated, all of their activities-entering their team, giving feedback, and perhaps looking at match schedules-will be logged by using the session.
PHP session management is both easy and powerful, so you can store information correctly throughout your site.

For example:

```
session_start();

$_SESSION['username'] = $username;
```

It takes the logged-in user's `username` and stores it in the session, from which it can be accessed throughout other pages like `home.php`, `team_registration.php`, and `feedback.php`.

2.2.2.6 Lightweight for Small Projects:

For small projects like your sports event management system, PHP is just the thing to keep things simple and get the work done:
- Lightweight and Efficient: PHP has minimum memory and CPU usage, so it is pretty convenient to work with for small and medium-sized web applications. The type of project does not necessitate heavy frameworks or complicated setups; therefore, PHP will allow you to work more on functionality than infrastructure.
- Rapid Development: PHP is really easy to use, and it has many built-in features like session handling, form processing, and database integration. All these lead to rapid development cycles-a mini-project under time constraints demands fast turnaround.

2.2.2.7 Easy Deployment on Local Development Environments

PHP is quite straightforward to set up in local environments like XAMPP or MAMP. They come with everything you need:

Apache Server

PHP

- Database (MySQL or PostgreSQL)

These environments would try to make the development and testing process easier. With python, creating frameworks such as Django or Flask is more complex with various configurations and dependencies as well as specific hosting environments. You can easily run your web application on the local server with PHP since PHP is easy to integrate with the local server.

2.2.2.8 Large Community and Plentiful Resources

PHP is one of the widely used server-side languages in the world, driving about 78% of all websites using server-side languages. The massive community provides extensive documentation, support, and libraries. If you need any help with issues arising or anything specific related to your sports event management system, the resources available online are sheer.

2.2.2.9 Why Use PHP Over Python for This Project:

Where Python is also one of the languages used in developing a web, below is why PHP would better suit for this project:

Specific to Web Development

PHP was developed with web development in mind, while Python is a general-purpose language. It is quicker to create any form of web-based application with PHP as it natively supports server-side operations such as posting of forms or dealing with session and database interactions compared to Python, which typically calls for the use of frameworks such as Django or Flask.

2.2.2.9 Ease of Direct Integration with HTML

One of the greatest advantages of PHP is its ability to be embedded directly inside HTML code. This will particularly benefit smaller web applications, where simplicity is paramount. You can easily mix front-end logic in the form of HTML/CSS and back-end logic in the form of PHP inside one file, which reduces the complexity of the project.

The frameworks, namely Django and Flask, enforce the separation of logic and presentation for Python that brings additional overhead in case you are working on smaller projects. It will need to be adjusted as it does setting up template engines, routing, and ORM layers, which would be overkill for a project like yours.

Lower Setup Overhead

You can develop and run your web application straight away if you set up a local server environment such as XAMPP. The great thing is, you do not have to set up frameworks like in Python's web development; this low overhead makes PHP an excellent choice for small, speedy projects where quick development speed is required

Familiarity and Speed

PHP is easy to use and beginner-friendly, thereby making it a perfect choice for projects as this where you want to quickly build functionalities like:

- Team registration form submission handling

User login

Match scheduling

While Python, being very agile and having heavy-duty frameworks attached to it, can definitely do this mini-project in a more powerful and efficient way, its speed and simplicity make it a much more pragmatic choice for this mini-project, especially when you are focusing on functionality over large-scale architecture.

3. REQUIREMENT AND ANALYSIS

Requirements and Analysis of the Sports Event Management System Project

The Sports Event Management System is designed to facilitate the efficient organization and management of sports events. The project involves two types of users: participants and organizers, each with their own set of requirements. In this section, we'll break down the functional and non-functional requirements of the system and conduct an analysis to ensure that the solution is well-aligned with the project's objectives.

3.1. Requirements of the System

A. Functional Requirements

Functional requirements define the specific behavior and functions of the system that must be implemented to meet the needs of the users. For the sports event management system, the primary functional requirements are:

3.1.1. User Management (Login and Signup)

- Signup Process:
    - The system must allow new users (both participants and organizers) to sign up by providing their phone number, email address, username, and password.
    - During signup, the user should select whether they are signing up as a participant or an organizer.
- Login Process:
    - After successful signup, users should be able to log in with their username and password.
    - The system should differentiate between participants and organizers and redirect them to their respective homepages after login.

3.1.2. Participants' Features

- Registration Page:
    - Participants should be able to register their team for a sport by entering details such as the team leader’s name, phone number, gender, department, sport name, and team members' details (name, gender, department).

    - After filling out the form, participants should be able to submit the registration, and the data must be stored in the Participants table in the PostgreSQL database.
- Match Schedule Viewing:
    - Participants should be able to view the schedule of upcoming matches, including the sport name, match date, time, and venue. This information is retrieved from the Match table.
- Requirements Page:
    - Participants should be able to view the necessary requirements for each sport (such as equipment or specific rules) on match day, retrieved from the Requirements table.
- Feedback Submission:
    - Participants should be able to submit feedback by providing their team name and feedback text. The system must store this data in the Feedback table.

3.1.3. Organizers' Features

- Match Scheduling:
    - Organizers must be able to plan and schedule matches by entering details such as the sport name, match date, time, and venue. This information should be stored in the Match table.
    - The system should prevent the creation of duplicate entries for matches (i.e., no two matches should have the same sport, date, and venue).
- Participant Management:
    - Organizers should have access to view the details of all participants, which are stored in the Participants table.
    - They should also have the ability to delete or remove participants if necessary.

3.1.4. Data Management

- The system must store data in an efficient and structured way using PostgreSQL. Tables must be normalized to avoid redundancy and ensure data integrity.
- Data integrity rules (such as validation for unique team names and valid phone numbers) should be enforced

B. Non-Functional Requirements

Non-functional requirements define the attributes that describe the system's operation and usability. These include performance, scalability, security, and user experience.

3.1.1. Usability

- The system should have a simple, user-friendly interface designed using HTML and CSS. Users should be able to navigate the system with ease, and forms should be easy to fill out.
- The system must support different types of users (participants and organizers) and provide clear instructions and feedback for each action (e.g., successful registration or error messages).

3.1.2. Performance

- The system should be able to handle multiple simultaneous user requests, especially when many participants are registering or viewing match schedules.
- Database queries (especially for retrieving match schedules and participant details) must be optimized for fast access.

3.1.3. Security

- The system should securely store user credentials (username and password) using hashing algorithms for passwords to ensure that login information is not compromised.
- Access control mechanisms should be in place to ensure that participants cannot access organizer-only features, and vice versa.
- Validation and sanitization of user input (e.g., team names, phone numbers) must be implemented to prevent SQL injection attacks or data corruption.

3.1.4. Scalability

- The system should be scalable, allowing for an increasing number of participants and events as the system grows.
- The database design should be flexible enough to accommodate new features or data without requiring major restructuring.

3.1.5. Reliability

- The system must be reliable, with minimal downtime. It should be able to handle errors gracefully and recover from failures without data loss.

3.2. Analysis of the Project

A. System Architecture

The architecture of the system follows a client-server model, where the client-side (the website) interacts with the server-side (backend) to retrieve and store data in the PostgreSQL database.

- Frontend: Built using HTML and CSS, the front end consists of web pages for registration, login, match schedules, and feedback submission. The pages will be responsive and provide clear navigation between features.
- Backend: The backend handles the logic for user authentication, participant registration, match scheduling, and feedback management. It also handles database queries to store and retrieve data from the PostgreSQL database.
- Database: The database is structured into normalized tables (Participants, Matches, Requirements, Feedback), which store the core data of the system. PostgreSQL was chosen for its scalability and ability to handle complex queries efficiently.

B. Database Design

The database design is a key element in the success of this project. After careful consideration, the following tables were created to support the functionality of the system:

- Participants Table: Stores details about teams, team leaders, and members.
- Match Table: Stores scheduled matches, including sport name, date, time, and venue.
- Requirements Table: Stores the requirements for each sport, including equipment or special instructions.
- Feedback Table: Stores feedback provided by teams, linked to team names.

C. Workflow of the System

The system is divided into several key workflows to ensure smooth operation for both participants and organizers:

3.2.1. Signup/Login Process

- Users first sign up with their details and choose whether they are a participant or organizer.
- After signup, users can log in and be directed to their respective home pages based on their role.

3.2.2. Participant Workflow

- Participants can register their teams by filling in their details, which are stored in the Participants table.
- They can view the match schedule on the calendar page, which retrieves data from the Match table.
- Participants can also view game-day requirements and submit feedback, stored in the Requirements and Feedback tables, respectively.

3.2.3. Organizer Workflow

- Organizers can schedule matches by entering match details, and these are stored in the Match table.
- Organizers can also view all participants and have the option to delete participants if necessary.

D. Potential Challenges and Solutions

3.2.1. Handling Large Numbers of Participants

- Challenge: As more participants register, retrieving match schedules and participant data might slow down the system.
- Solution: Optimize SQL queries, create indexes on frequently queried columns (like sport name and match date), and paginate data in the frontend.

3.3.2. Preventing Unauthorized Access

- Challenge: Ensuring that participants do not access organizer-only features.
- Solution: Implement access control and session management to verify user roles before allowing access to specific pages.

3.3.3. Error Handling

- Challenge: Managing errors (e.g., incorrect user input, failed database connections).
- Solution: Implement comprehensive validation for user inputs and error handling mechanisms that provide user-friendly error messages while logging detailed error data for debugging.

3.1 REQUIREMENT SPECIFICATION

Requirement Specification for Sports Event Management System

A **requirement specification** document is a comprehensive outline of the functionalities, features, and constraints that a system must satisfy to fulfill the project's objectives. The **Sports Event Management System** requires a detailed breakdown of both **functional** and **non-functional requirements**, including how data is handled, the user interface design, security measures, and system performance. This document provides an exhaustive explanation of the requirement specification, focusing on user roles, the system's interaction with the database, and the performance, usability, and security standards that need to be met.

3.1.1. Introduction

The **Sports Event Management System** is designed to facilitate the organization of sports events, allowing participants to register their teams, view upcoming matches, check equipment requirements, and provide feedback. Organizers manage match schedules, oversee participant registration, and review feedback.

The system differentiates between two primary user roles: **participants** and **organizers**. Each user type has distinct functionalities, including:

- Participants can register for events, view their match schedules, check requirements, and provide feedback.
- Organizers can schedule matches, view participant lists, and monitor feedback.

This project involves a web-based system using **HTML**, **CSS**, and PHP for the frontend and backend development, with **PostgreSQL** as the database system.

3.2. Functional Requirements

Functional requirements define the specific functions the system must perform. These requirements ensure that the system achieves its goals by providing the necessary features for both participants and organizers.

3.2.1 User Authentication and Authorization

This module involves creating, logging in, and managing user accounts.

- **Login Functionality** : The system must allow users to log in using their username and password. If login credentials are incorrect, an error message should display, allowing users to retry.
- **Signup Functionality** : New users (both participants and organizers) should have a sign-up option. The system will prompt users to provide an email, phone number, and password. Users will also need to select a role—either "Participant" or "Organizer."
- **Role-Based Access Control** : The system must clearly differentiate between roles, providing different access to functionalities. Participants should have access to event registration and match

details, while organizers should be able to add and modify match schedules and view participant details.

3.2.2 Participant Registration and Management

Once a participant logs in, they can register for an event by submitting their team details.

- **Team Registration**: The registration page will allow participants to enter the following data:
  - Team ID (auto-generated serial number)
  - Leader Name
  - Leader Age, Gender, and Department
  - Sport Name
  - Team Name

  Once registered, the data should be stored in the Teams table within the PostgreSQL database.
- **View Match Calendar**: Participants should have access to a calendar page displaying the match schedule for their registered sport. The calendar must show:
  - Match Date and Time
  - Venue
  - Time remaining until the match
- **View Requirements**: Participants should be able to check a list of equipment or resources needed for their sport. These requirements are stored in the Requirements table, which is maintained by the organizers.
- **Provide Feedback**: Participants can submit feedback on matches or events via a feedback form. The feedback will be stored in the Feedback table and reviewed by organizers.

3.2.3 Organizer Match and Participant Management

Organizers are responsible for managing the matches and participants.

- **Add Match**: Organizers must be able to add new matches by entering:
  - Sport Name
  - Match Date and Time
  - Venue

  This data is stored in the **Matches** table and displayed in the match calendar for participants.
- **View Participants** : Organizers should be able to view a list of participants registered for different sports. They can also delete participants from the list if necessary.
- **View Feedback** : Organizers will have access to the feedback submitted by participants to review the quality of the event and make improvements for future matches.
- **Auto-Delete Matches** : After a match has been completed, the system will automatically remove outdated match details from the Matches table.

3.2.4 Database Management

The **PostgreSQL** database is at the heart of the system. It will store all user, team, match, requirement, and feedback data.

- **Users Table**: This table will store user credentials and roles (participant or organizer).
- **Teams Table**: This table will store team registration details, such as Team ID, Leader Name, and Sport.
- **Matches Table**: This table will store match details such as sport name, date, time, and venue.
- **Requirements Table**: This table will store the list of materials or equipment required for each sport.
- **Feedback Table**: This table will store feedback submitted by participants.

3.3. Non-Functional Requirements

Non-functional requirements specify the criteria that evaluate the system's operation, including usability, performance, security, and scalability.

3.3.1 Performance Requirements

The system must meet the following performance standards to ensure smooth operation:

- **Fast Data Retrieval**: Match details, participant data, and feedback must be retrieved from the database quickly to avoid delays for the users. The system must handle queries efficiently to provide a quick response time (preferably within 2-3 seconds).
- **Concurrency**: The system must be able to handle multiple users accessing it simultaneously. This is particularly important when there are multiple sports events happening at the same time, with organizers scheduling matches and participants viewing schedules concurrently.
- **Scalability**: As the number of users (both participants and organizers) increases, the system must scale appropriately. The database design must accommodate large volumes of data (e.g., increasing number of teams, matches, and feedback).

3.3.2 Security Requirements

Security is crucial when handling sensitive information such as login credentials and feedback.

- **Data Encryption**: Passwords and other sensitive data must be encrypted in the database using techniques like hashing (e.g., bcrypt or SHA-256).
- **Role-Based Access Control**: The system must strictly enforce role-based access control, ensuring that participants cannot access organizer features and vice versa.
- **Prevent SQL Injection**: The system must employ prepared statements and input validation to prevent SQL injection attacks. This ensures that malicious queries cannot manipulate the database.
- **Session Management**: User sessions should be managed securely, with timeouts and proper logout procedures to prevent unauthorized access after a session has ended.

3.3.3 Usability Requirements

The system must be easy to use and accessible to both technical and non-technical users.

- **User-Friendly Interface**: The interface must be intuitive and easy to navigate, ensuring that users can find the features they need without confusion. Clear instructions, form validations, and error messages must be provided throughout the system.
- **Responsive Design**: The website must be accessible on different devices, including desktops, tablets, and mobile phones. The layout should adjust dynamically based on the screen size for optimal viewing.
- **Error Handling**: The system must provide clear and user-friendly error messages when users enter invalid data (e.g., during login or registration). The error messages should guide the user on how to correct their input.

3.3.4 Reliability and Availability

The system must be reliable and available for users whenever needed.

- **Uptime**: The system should ensure high uptime, aiming for 99% availability so that users can access the platform without interruptions.
- **Data Backup**: Regular backups of the PostgreSQL database must be implemented to prevent data loss in the event of a system failure. This is critical for preserving match schedules, team registrations, and feedback data.

3.4. Use Case Scenarios

Use cases illustrate how users interact with the system. This section explains a few key scenarios for participants and organizers.

3.4.1 Use Case: Participant Registration

- **Actor**: Participant
- **Preconditions**: The participant has logged in successfully.
- **Main Success Scenario**:
  1. The participant navigates to the "Register Team" page.
  2. The participant enters their team's details, including Team Name, Leader Name, and Sport.
  3. The participant submits the form, and the system stores the team details in the **Teams** table.
  4. A confirmation message is displayed, and the participant can view their match schedule.

3.4.2 Use Case: Organizer Adds Match

- **Actor** : Organizer
- **Preconditions** : The organizer has logged in successfully.
- **Main Success Scenario** :
  1. The organizer navigates to the "Add Match" page.
  2. The organizer provides the match details, including Sport, Date, Time, and Venue.

3. The system stores the match details in the   Matches   table.
4. Participants who have registered for that sport see the new match in their calendar.

3. **4.3 Use Case: Participant Submits Feedback**

- **Actor**: Participant
- **Preconditions**: The participant has participated in a match.
- **Main Success Scenario**:

1. The participant navigates to the "Submit Feedback" page.
2. The participant enters their feedback about the match.
3. The system stores the feedback in the   Feedback   table.
4. A confirmation message is displayed, and the feedback is available for organizers to review.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

A well-defined set of hardware and software requirements is crucial for the successful development, deployment, and management of the **Sports Event Management System**. This web-based platform, designed for participants and organizers, facilitates event registration, match scheduling, resource management, and feedback collection. To ensure the project is built efficiently and functions smoothly, it’s necessary to specify the hardware and software resources that support both the development and operational environments.

3.2.1. Hardware Requirements

Hardware requirements refer to the physical resources necessary to develop, test, and run the system. These requirements vary depending on whether the system is being developed or deployed in a local or server-based environment.

3.2.1.1 Development Machine

A **development machine** is the hardware used by developers to code, test, and debug the system. The machine must have adequate processing power, memory, and storage to handle multiple software development tools and libraries simultaneously. Since the project involves database operations, web server configuration, and real-time testing, the development machine needs moderate to high performance. Below are the recommended specifications:

- **Processor**: Intel Core i5 (Quad-core or higher) or AMD Ryzen 5 or equivalent. These processors offer adequate speed for running multiple applications, compiling code, and managing database queries.
- **RAM**: Minimum of 8 GB of RAM, with 16 GB recommended for smoother performance, especially if working with large datasets and running multiple services simultaneously (such as a local server and database management tool).
- **Storage**: At least 100 GB of available hard drive space is required. An **SSD (Solid State Drive)** is recommended for faster data access, especially when running virtual servers or local databases.
- **Graphics**: Basic integrated graphics is sufficient, as the project is not graphics-intensive.
- **Network**: High-speed Ethernet or wireless connection to access online resources, libraries, and APIs.
- **Monitor**: Resolution of 1366x768 or higher, with a dual-monitor setup recommended for productivity when working with both the front-end and back-end simultaneously.

- **Operating System**: The system can be developed on **Windows (Windows 10 or higher)**, **macOS (10.14 or higher)**, or **Linux (Ubuntu 20.04 or higher)** depending on developer preference and familiarity.

3.2.1.2 Web Server (For Deployment)

Once the system is ready for deployment, it must be hosted on a web server that can handle multiple requests from users. This server will manage tasks such as handling user logins, storing team data, managing match schedules, and processing feedback submissions. Below are the server specifications for hosting the web application:

- **Processor**: Intel Xeon or AMD EPYC with Quad-core or higher, capable of handling multiple user requests in real time.
- **RAM**: Minimum 8 GB, with 16 GB recommended. Adequate RAM ensures that the server can handle concurrent connections, caching, and database operations efficiently.
- **Storage**: A minimum of 500 GB of storage, preferably **SSD**, is recommended for storing application files, user data, match schedules, and feedback submissions.
- **Network Interface**: A high-speed network interface with a minimum of 1 Gbps connectivity is essential for smooth communication between the server and clients, especially when dealing with multiple users.
- **Operating System**: A Linux-based server OS, such as **Ubuntu Server (20.04 LTS)** or **CentOS**, is highly recommended due to its performance and compatibility with PHP and PostgreSQL.

3.2.1.3 Database Server

If the system is designed to separate the database management from the web application server, the **database server** needs to be optimized to handle data storage and retrieval operations efficiently. This is especially important if the number of participants and organizers grows over time. Below are the recommended specifications for a dedicated database server:

- **Processor**: Intel Xeon or AMD EPYC with Quad-core or higher, optimized for handling database queries and large volumes of data.
- **RAM**: A minimum of 16 GB of RAM, with 32 GB recommended for managing complex queries, data storage, and retrieval from large tables (Teams, Matches, Requirements, Feedback).

- **Storage**: SSD with at least 1 TB of storage capacity. SSDs offer faster read/write speeds, which are essential for database operations.
- **Network Interface**: High-speed network connection to ensure smooth data transfer between the database and the web server.
- **Operating System**: A Linux-based OS such as **Ubuntu Server** or **CentOS** is recommended for optimal compatibility with PostgreSQL, the database system used for this project.

3.2.2. Software Requirements

The software requirements for this project encompass the tools, libraries, programming languages, and frameworks required to build and maintain the Sports Event Management System. These requirements are divided into **development tools**, **frontend technologies**, **backend technologies**, and **database management systems**.

3.2.2.1 Development Environment

The development environment includes the operating systems, text editors, and tools that developers will use to write and test code.

- **Operating System**: The system can be developed on a variety of platforms, depending on the developer's preference:
    - **Windows 10 or higher**
    - **macOS 10.14 or higher**
    - **Linux distributions** such as **Ubuntu 20.04** or higher for a lightweight and customizable environment.
- **Integrated Development Environment (IDE)**:
    - **Visual Studio Code**: A highly customizable, open-source IDE with a rich ecosystem of extensions for PHP, PostgreSQL, and HTML/CSS.
    - **PHPStorm**: A powerful commercial IDE tailored for PHP development with advanced features such as debugging, code completion, and integration with version control systems (like Git). This is particularly useful for larger projects.
    - **Sublime Text** or **Atom**: Lightweight text editors for developers who prefer simplicity and fast performance.
- **Web Browsers**: For testing and debugging the web application, the following browsers are recommended:
    - **Google Chrome** (with developer tools for performance testing)
    - **Mozilla Firefox** (for cross-browser compatibility testing)

    - **Microsoft Edge** (to ensure the system functions well across multiple platforms)

3.2.2.2 Backend Software

The backend of the Sports Event Management System is responsible for handling the server-side logic, including managing user accounts, storing data, and responding to user requests.

- **Programming Language**:
    - **PHP (7.4 or higher)**: PHP is the primary server-side scripting language used in this project. It handles the core logic of the application, such as validating login credentials, managing session states, interacting with the database, and processing form data.
- **Database**:
    - **PostgreSQL (12 or higher)**: PostgreSQL is a powerful, open-source object-relational database system that is used to store and manage all the application's data, including user accounts, teams, match schedules, feedback, and requirements.
- **Database Management Tool**:
    - **pgAdmin 4**: This is a graphical tool for managing PostgreSQL databases. It allows developers to execute queries, create and modify tables, and manage data stored in the database.
- **Web Server**:
    - **Apache HTTP Server**: This is an open-source web server used to serve the PHP scripts and HTML pages to users. It handles requests from the client side and communicates with the server-side logic.
    - **XAMPP**: For local development, XAMPP (which includes Apache, MySQL, PHP, and Perl) is a popular package that simplifies the setup of a local development environment.

3.2.2.3 Frontend Software

The frontend technologies dictate the user interface and user experience of the Sports Event Management System.

- **HTML5**: HTML is used for structuring the content of the web pages, including the login form, registration form, match schedule, feedback form, and other UI components.

- **CSS3**: CSS is used for styling the web pages. It ensures that the website is visually appealing and that the layout is responsive, adjusting to different screen sizes (e.g., desktops, tablets, and smartphones).
- **JavaScript**: JavaScript is used to add interactivity to the website. For example, it handles client-side form validation, provides dynamic updates (such as showing how much time is left before a match), and enhances the user experience by creating a more responsive and dynamic UI.

3.2.2.4 Frameworks and Libraries

- **Bootstrap 5** (optional): Bootstrap is a popular front-end framework for building responsive, mobile-first websites. It offers pre-designed components such as buttons, forms, and grids, which can speed up the development of the system's user interface.
- **jQuery** (optional): jQuery is a fast, small, and feature-rich JavaScript library that simplifies HTML document manipulation, event handling, and Ajax interactions.

3.2.2.5 Security Software

Security is a key concern for any web application, especially those handling sensitive user data (like usernames, passwords, and contact information).

- **HTTPS**: The system will require a Secure Sockets Layer (SSL) certificate to ensure secure communication between the client and server. **Let’s Encrypt** can be used to provide free SSL certificates.
- **Encryption Libraries**: For password storage, PHP’s built-in functions like **password_hash()** and **password_verify()** will be used to securely hash and store user passwords in the database.

3.2.2.6 Testing and Debugging Tools

Testing and debugging tools ensure that the system works as expected and helps identify and fix bugs.

- **PHPUnit**: PHPUnit is a unit testing framework for PHP. It allows developers to write and execute tests to ensure the backend logic works as expected.
- **Postman**: Postman is a popular API testing tool that allows developers to test and monitor the system’s interaction with the server through HTTP requests, especially for form submissions or AJAX calls.

- **XDebug**: XDebug is a PHP debugging tool that offers a way to step through code, monitor variables, and troubleshoot issues during development.

3.2.2.7 Version Control

Version control systems are essential for managing code changes, especially in collaborative environments.

- **Git**: Git is a distributed version control system that allows developers to track code changes, revert to previous states, and collaborate with other team members.
- **GitHub** or **GitLab**: These platforms provide cloud-based hosting for Git repositories, enabling team collaboration, project management, and code reviews.

3.2.3. Deployment Environment

The deployment environment consists of the software and tools necessary to host the application in a production environment where real users can access the system.

- **Hosting Platform**: Cloud providers like **AWS (Amazon Web Services)**, **DigitalOcean**, or **Heroku** can host the application. The hosting platform must support **PHP** and **PostgreSQL**.
- **Linux-based Web Server**: A **Linux-based server** is preferred for deployment due to its compatibility with PHP and PostgreSQL and its cost-effectiveness. The server can be managed using tools like **SSH** and control panels like **cPanel** for easy deployment and maintenance.
- **DNS Configuration**: The deployment environment will require a domain name (e.g., www.sportseventmanager.com) to make the system accessible via the internet.

3.2.4. Additional Tools

For data analysis and report generation, the following tools may be used:

- **Microsoft Excel** or **Google Sheets**: These tools can be used to store and analyze exported data from the system. For example, reports on participants, match results, or feedback can be exported in CSV format and analyzed in these tools.
- **Backup Tools**: Regular backups are essential to ensure that data is not lost. Tools like **pg_dump** or **pgBackRest** can be used to create backups of the PostgreSQL database and restore it in case of failure.

3.3 ARCHITECTURE DIAGRAM

To create a simplified architecture diagram, we focus on three key layers: **Frontend**, **Backend**, and **Database**, each with specific roles and interactions. Let's walk through the core elements and interactions between these components in detail.

3.3.1. Frontend Layer

The **Frontend Layer** represents the part of the system that users interact with directly. This layer is responsible for displaying the user interface and sending requests to the backend. It consists of various web pages and elements built using **HTML**, **CSS**, and optionally, some JavaScript to enhance interactivity.

Key Features:

- **Login and Signup Forms**: Users, either participants or organizers, enter their credentials to access the system. A login page accepts usernames and passwords, while a signup page gathers user details like email, phone number, and role selection (participant or organizer).
- **Event Registration Form**: Participants can fill in the details to register their team for a particular sport event. They provide information such as leader’s name, team name, and sport of choice.
- **Match Calendar**: This section displays a schedule of upcoming matches for registered sports. The calendar pulls data from the backend and database and shows match details like date, time, and venue.
- **Requirements Display**: Participants can view a list of items or equipment required for a specific sport. This is populated based on the sport the participant has registered for.
- **Feedback Form**: Participants can submit feedback about the event or sport. This feedback is sent to the backend for processing.

The **Frontend Layer** sends data (such as login information, registration details, feedback) to the **Backend** through HTTP requests and receives responses that display relevant content, like match details or confirmation messages.

3.3.2 Backend Layer

The **Backend Layer** serves as the brain of the system, handling business logic, user authentication, and interaction with the database. This layer is primarily developed using **PHP**, a server-side programming language.

Key Responsibilities:

- **User Authentication**: Upon receiving login credentials, the backend verifies them against the database, granting access to participants or organizers based on valid credentials.
- **Registration Handling**: For participants, when they submit their registration form, the backend processes the input, validates it, and stores the information in the appropriate database tables.
- **Match Scheduling and Management**: Organizers can use the backend to schedule new matches by inputting details like match time, date, and venue. These details are stored in the **Matches Table** in the database.
- **Requirements and Feedback Processing**: The backend fetches sport-specific requirements from the **Requirements Table** for participants to view. Similarly, when participants submit feedback, it processes and stores the feedback in the **Feedback Table**.
- **Dynamic Data Display**: For features like the **Match Calendar**, the backend retrieves match information from the **Matches Table** in the database and sends it to the frontend for display. It ensures that users only see relevant data, such as upcoming matches.

The backend layer acts as a mediator between the frontend and the database, ensuring that user actions are properly processed and that the system functions as intended.

3.3.3 Database Layer

The **Database Layer** is where all the application data is stored and managed. This layer is built on **PostgreSQL**, a robust relational database management system.

Core Tables in the Database:

- **Users Table**: This stores information about all registered users, including usernames, passwords (hashed for security), emails, phone numbers, and roles (either participant or organizer). When a user logs in, the backend authenticates them by checking this table.

- **Teams Table**: Contains the details of each registered team, including the team leader's name, age, gender, department, sport name, and team name. Each team is linked to a participant who has completed the registration form.
- **Matches Table**: Stores match details such as the sport name, match time, date, and venue. When an organizer adds a new match, this table is updated. The backend uses this data to display match information in the frontend calendar.
- **Requirements Table**: Holds data on the specific requirements for each sport, such as equipment, clothing, or other materials. This table is queried by the backend to show participants what they need to prepare for their sport.
- **Feedback Table**: Contains feedback submitted by participants about the events or sports. Organizers can view this data to gain insights into participants' experiences and make improvements.

The **Database Layer** stores all critical data securely and provides it to the backend upon request. It ensures data consistency and integrity throughout the system

Elaboration on the Interaction Between Layers

The architecture of the **Sports Event Management System** operates by facilitating smooth communication between the **Frontend**, **Backend**, and **Database** layers.

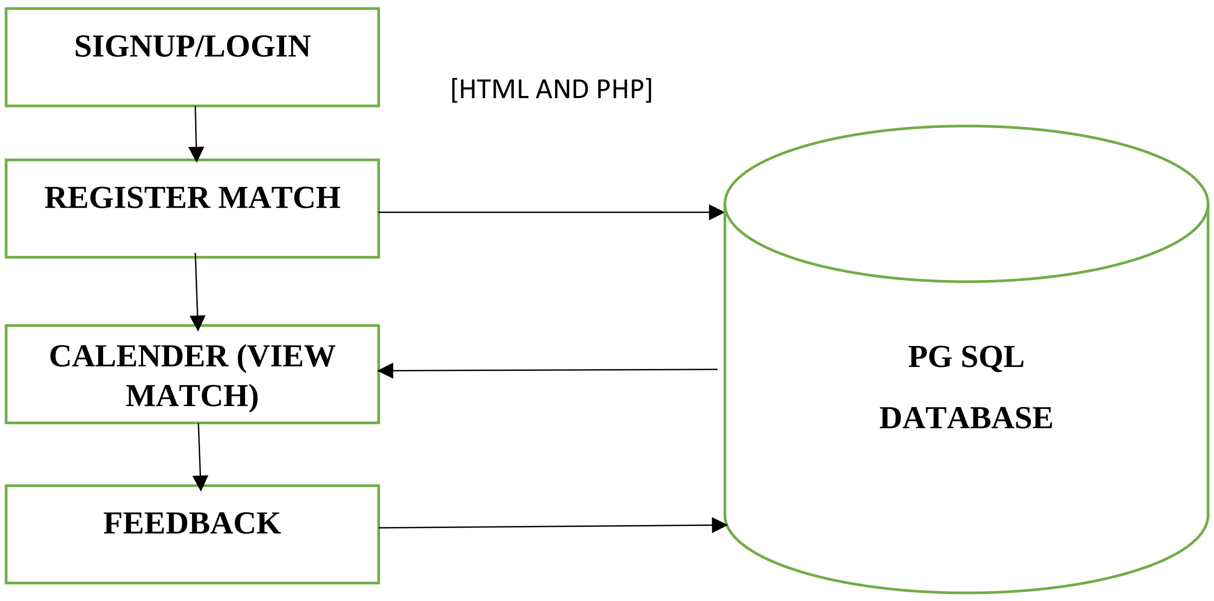
- **Frontend-to-Backend Communication**: When a user interacts with the system (e.g., logs in, registers a team, or submits feedback), the frontend sends the user inputs to the backend through HTTP requests. This data is usually sent using forms (login, registration) or buttons (feedback submission). Once the backend processes the request, it returns a response to the frontend, which updates the interface accordingly.
- **Backend-to-Database Communication**: The backend acts as an intermediary between the frontend and the database. When it receives a request (e.g., to register a team or fetch match details), it queries the database for relevant information. After processing the data, the backend either updates the database or sends the data back to the frontend for display. For example, when the backend retrieves upcoming match details from the **Matches Table**, it sends this data back to the frontend, where the **Match Calendar** is updated.
- **Frontend Display and Updates**: Once the backend has processed the user's request, it sends a response back to the frontend. For example, when a participant submits their registration, the backend verifies and stores the data, then responds with a success message. This message is displayed on the frontend, giving users confirmation of their action.

Example Use Case: Registration Process

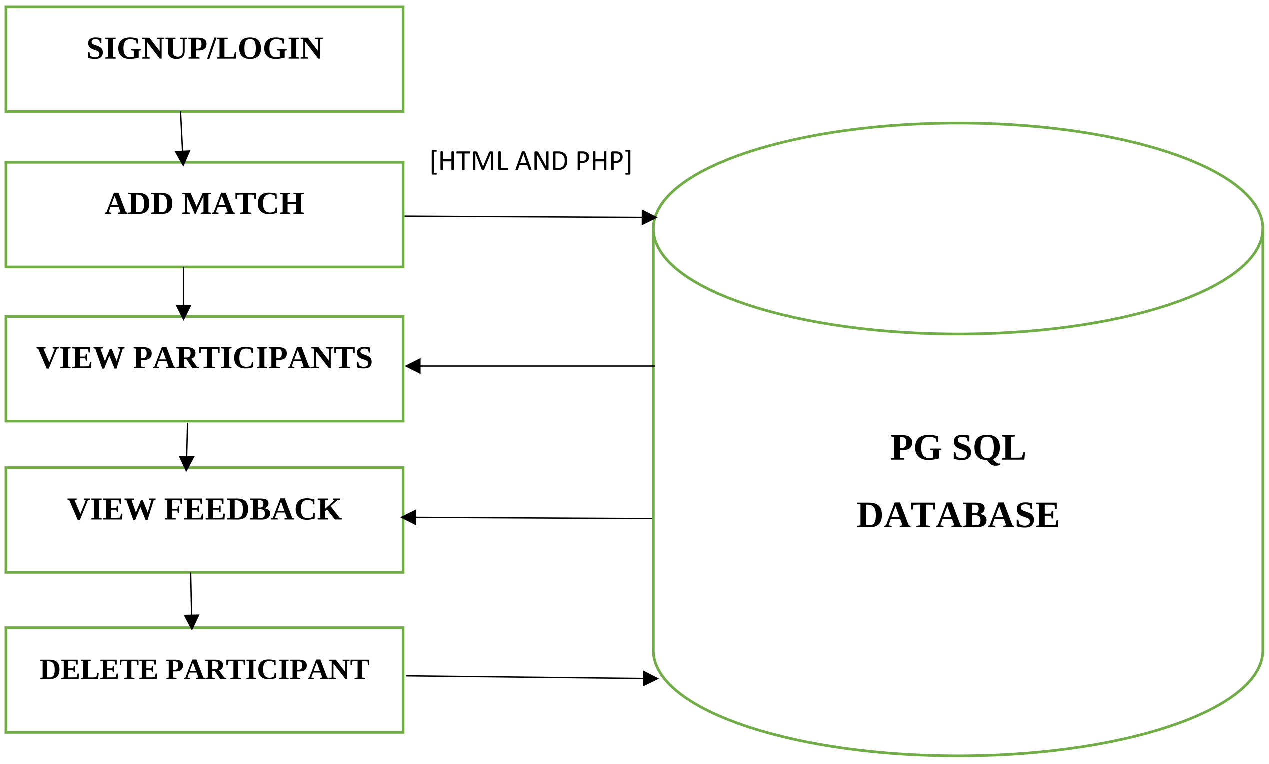
Let’s explore how a participant might register for an event using this architecture:

1. **Frontend Interaction**: The participant fills out a registration form in the frontend, entering details like team name, sport, and leader information. When they click “Register,” the frontend sends an HTTP POST request to the backend with this data.
2. **Backend Processing**: The backend receives the form data, validates it (ensuring fields are filled correctly), and then updates the **Teams Table** in the database with the new team information.
3. **Database Storage**: The backend inserts a new row into the **Teams Table**, storing the participant’s registration data.
4. **Frontend Response**: The backend sends a response back to the frontend confirming that the registration was successful. The frontend displays a message to the user, confirming that their team has been registered for the event.

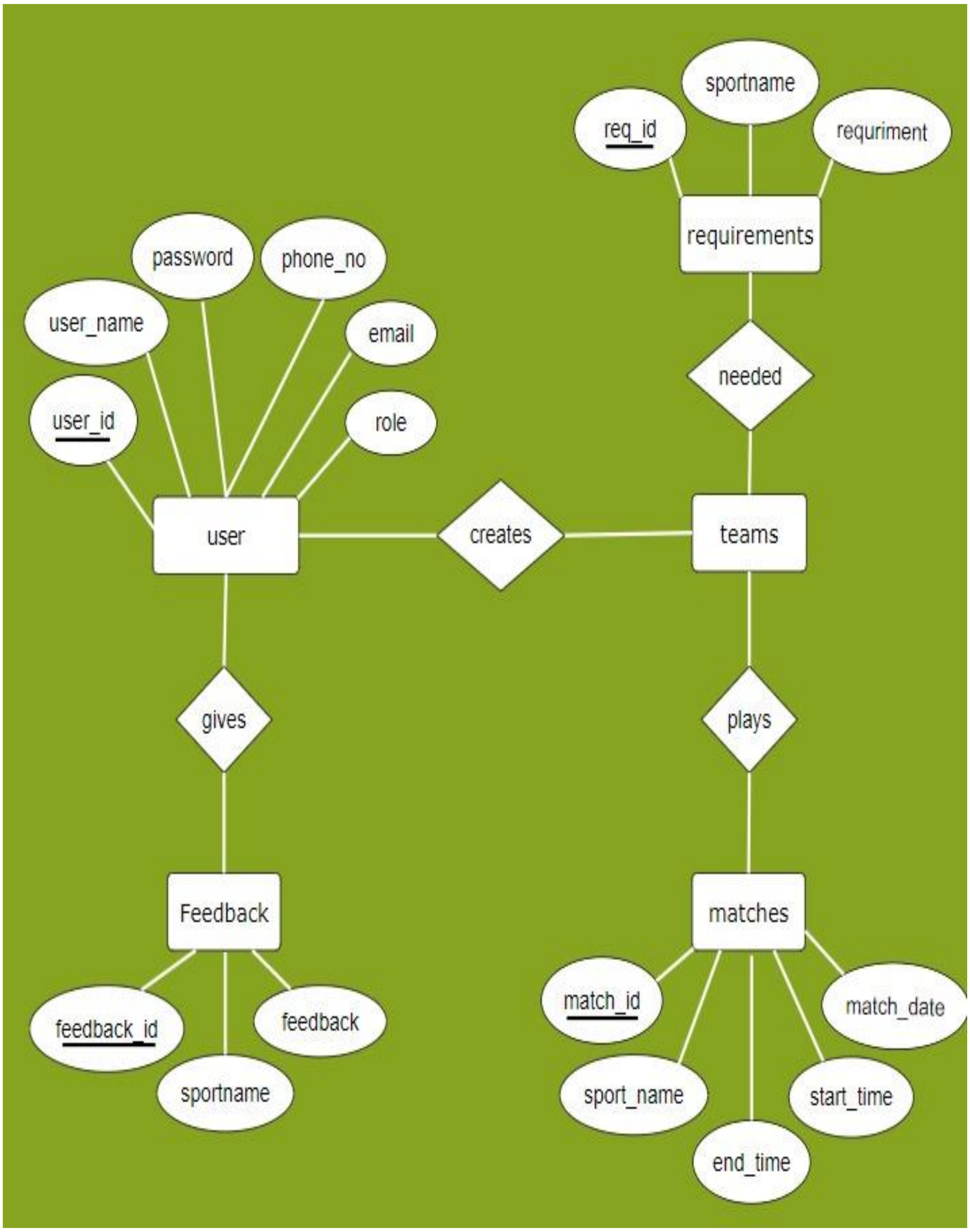
Participants flow:



ORGANIZER FLOW:



3.4 ER DIAGRAM



3.5 NORMALIZATION

Normalization in the Sports Event Management System

Normalization is a crucial concept in database design, used to organize data to minimize redundancy and avoid undesirable characteristics like insertion, update, or deletion anomalies. The primary goal of normalization is to split larger tables into smaller, related tables, ensuring efficient storage and retrieval of data. For your **sports event management system**, normalization helps ensure the database is structured for optimal performance, accuracy, and data integrity.

Why Normalization is Necessary for this Project

In the **sports event management system**, you are dealing with various entities such as **Participants**, **Matches**, **Requirements**, and **Feedback**. If the data for these entities were all stored in one large table, it would lead to several issues:

- **Redundancy**: Repetitive data (e.g., storing the same team information across different events).
- **Update anomalies**: Updating a team's details would require changing data in multiple rows, increasing the chance of errors.
- **Insertion anomalies**: Without proper normalization, it may be difficult to insert data (e.g., adding a participant before a match is scheduled).
- **Deletion anomalies**: Deleting a match might remove important participant or feedback data.

To avoid these issues, the database is broken into related tables, and the relationships between them are carefully managed. Here’s a detailed explanation of how normalization applies to each of the tables in the sports event management system:

3.5.1. First Normal Form (1NF)

1NF eliminates repeating groups by ensuring that each table contains atomic (indivisible) values and each record is unique.

In this project:

- **Participants Table**: Initially, if we store all the details of a team (team leader, team members, sport name) in one row, it would lead to redundancy and repeating groups for team members. To satisfy **1NF**, we ensure that each participant’s details are stored as a separate record, and the team leader’s information is atomic (no repeated team members in one field).

Example: Instead of storing the details of team members in a single field like this:

css

```
Team_Name | Team_Leader | Sport | Team_Members

-----------|-------------|-------|-------------

Team A     | Alice       | Soccer| John, David, Emma
```

We break it down to have separate records for each team member:

Css

```
Team_Name | Team_Leader | Sport | Member_Name | Member_Department | Member_Gender

-----------|-------------|-------|-------------|-------------------|-------------

Team A     | Alice       | Soccer| John        | Engineering       | Male

Team A     | Alice       | Soccer| David       | Science           | Male

Team A     | Alice       | Soccer| Emma        | Engineering       | Female
```

3.5.2. Second Normal Form (2NF)

2NF removes partial dependencies by ensuring that all non-key attributes are fully dependent on the entire primary key.

In this project, consider the **Participants** table, where the primary key is the combination of **Team_Name** and **Sport_Name** (composite key). If we include other fields like team leader’s phone number, these fields should depend entirely on both the team and sport.

However, if a team leader’s phone number is stored redundantly for each participant in the team, this violates **2NF**. To solve this, we can separate the team leader’s details into a **Team Leader** table, which will have a one-to-many relationship with the **Participants** table. This avoids storing repeated data about the team leader for every team member.

Example of moving to 2NF:

Before normalization:

css

```
Team_Name | Sport_Name | Leader_Name | Leader_Phone | Member_Name

----------|------------|-------------|--------------|-------------

Team A    | Soccer     | Alice       | 1234567890   | John

Team A    | Soccer     | Alice       | 1234567890   | David

Team A    | Soccer     | Alice       | 1234567890   | Emma
```

After 2NF:

- Create a separate **Team Leader** table:

css

```
Team_Name | Leader_Name | Leader_Phone

----------|-------------|--------------

Team A    | Alice       | 1234567890
```

- Modify **Participants** table to remove redundant data:

css

```
Team_Name | Sport_Name | Member_Name | Member_Department | Member_Gender

----------|------------|-------------|-------------------|-------------

Team A    | Soccer     | John        | Engineering       | Male

Team A    | Soccer     | David       | Science           | Male
Team A    | Soccer     | Emma        | Engineering       | Female
```

3.5.3. Third Normal Form (3NF)

3NF eliminates transitive dependencies by ensuring that no non-primary-key attribute depends on another non-primary-key attribute.

In the **Match** table of this project, suppose you store not only the **Sport_Name** and **Match_Venue**, but also the **Venue_Location** and **Venue_Manager** details. Here, **Venue_Location** and **Venue_Manager** depend on the **Match_Venue** rather than directly on the **Sport_Name** or **Match_Date**, creating a transitive dependency.

To satisfy **3NF**, we create a separate **Venue** table to store venue details and remove those attributes from the **Match** table.

Example of moving to 3NF:

Before 3NF:

yaml

```
Sport_Name | Match_Date | Match_Time | Match_Venue | Venue_Location | Venue_Manager
-----------|------------|------------|-------------|----------------|--------------
Soccer     | 2024-10-05 | 10:00 AM   | Stadium 1   | Main Campus    | Mr. Smith
Basketball | 2024-10-06 | 02:00 PM   | Gym 2       | North Campus   | Ms. Johnson
```

After 3NF:

- Create a **Venue** table:

css

```
Venue_ID | Venue_Name | Venue_Location | Venue_Manager
---------|------------|----------------|---------------
1        | Stadium 1  | Main Campus    | Mr. Smith
```

```
2       | Gym 2      | North Campus   | Ms. Johnson
```

- Modify the **Match** table to reference the **Venue**:

yaml

```
Sport_Name | Match_Date | Match_Time | Venue_ID

-----------|------------|------------|---------

Soccer     | 2024-10-05 | 10:00 AM   | 1

Basketball | 2024-10-06 | 02:00 PM   | 2
```

Normalization Applied Across All Tables

Participants Table

- **Normalization:** By separating **Team_Leader** details from the **Participants** table, you reduce redundancy and ensure that each participant's details are stored uniquely.

Match Table

- **Normalization:** By separating **Venue** information into a different table, the **Match** table avoids storing repeated data about the same venue across multiple matches, thus ensuring efficient data retrieval and updates.

Requirements Table

- This table stores the necessary items (e.g., equipment) required for each sport on match day. Normalization here avoids duplicating information by keeping sport names unique, with each sport having its own list of requirements.

Feedback Table

- The **Feedback** table is relatively simple, containing **Team_Name** and **Feedback**. Since each feedback is linked to a unique team, the table is already in **3NF** because there is no transitive dependency.

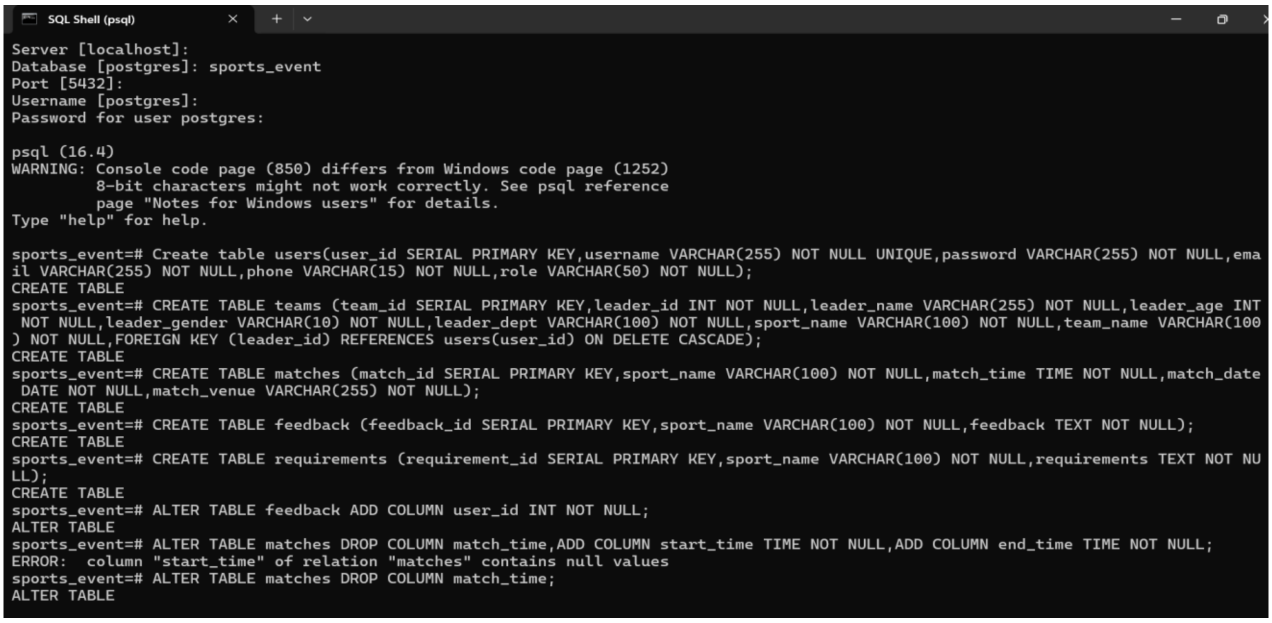
Advantages of Normalization in the Project

1. **Data Integrity:** By breaking down the data into smaller, related tables, normalization ensures that all relationships between entities are maintained accurately.
2. **Reduced Redundancy:** The structure prevents the duplication of data, such as the team leader’s phone number being stored multiple times for the same team.
3. **Easier Maintenance:** Changes in one part of the data (e.g., updating venue details) do not require multiple updates across the entire database, making the system easier to maintain.
4. **Efficient Queries:** With well-normalized tables, the system can efficiently retrieve and update data without unnecessary complexity or performance overhead.
5. **Better Scalability:** As the system grows, normalized tables ensure that the database can handle an increasing number of participants, matches, and feedback without performance issues.

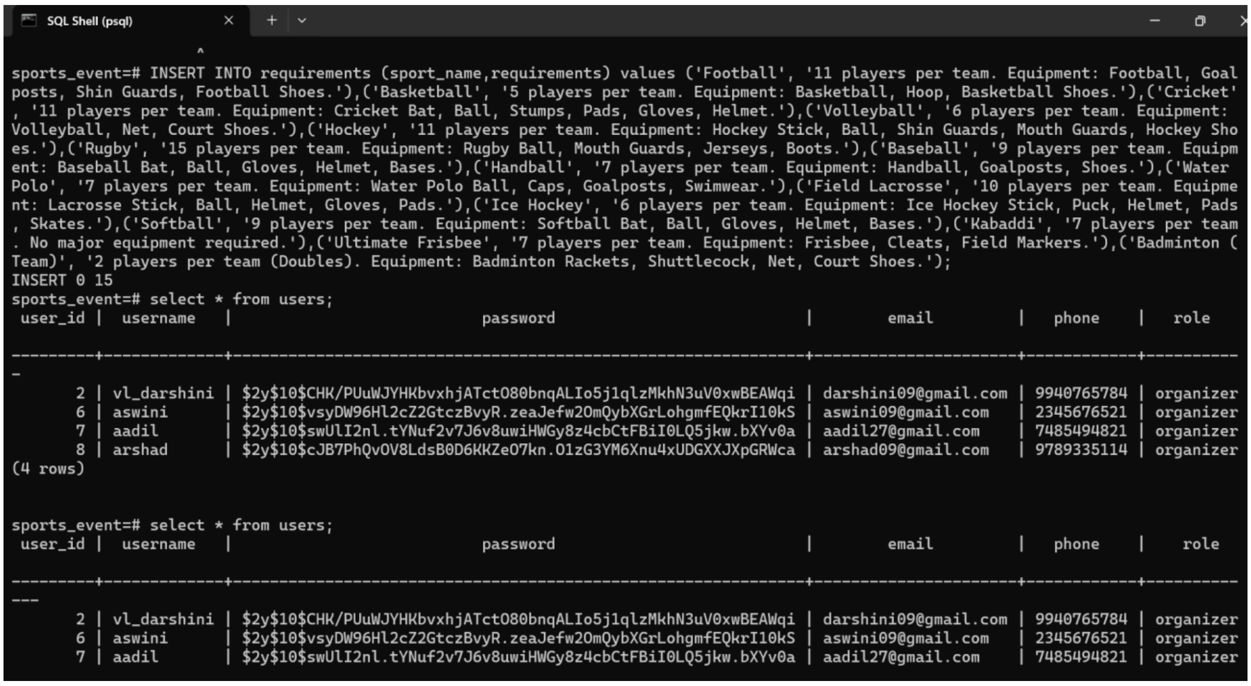
.

3.5.4.PROGRAM SOURCE CODE

POSTGRE SQL CODE:



```
Server [localhost]:
Database [postgres]: sports_event
Port [5432]:
Username [postgres]:
Password for user postgres:

psql (16.4)
WARNING: Console code page (850) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

sports_event=# Create table users(user_id SERIAL PRIMARY KEY,username VARCHAR(255) NOT NULL UNIQUE,password VARCHAR(255) NOT NULL,ema
il VARCHAR(255) NOT NULL,phone VARCHAR(15) NOT NULL,role VARCHAR(50) NOT NULL);
CREATE TABLE
sports_event=# CREATE TABLE teams (team_id SERIAL PRIMARY KEY,leader_id INT NOT NULL,leader_name VARCHAR(255) NOT NULL,leader_age INT
 NOT NULL,leader_gender VARCHAR(10) NOT NULL,leader_dept VARCHAR(100) NOT NULL,sport_name VARCHAR(100) NOT NULL,team_name VARCHAR(100
) NOT NULL,FOREIGN KEY (leader_id) REFERENCES users(user_id) ON DELETE CASCADE);
CREATE TABLE
sports_event=# CREATE TABLE matches (match_id SERIAL PRIMARY KEY,sport_name VARCHAR(100) NOT NULL,match_time TIME NOT NULL,match_date
 DATE NOT NULL,match_venue VARCHAR(255) NOT NULL);
CREATE TABLE
sports_event=# CREATE TABLE feedback (feedback_id SERIAL PRIMARY KEY,sport_name VARCHAR(100) NOT NULL,feedback TEXT NOT NULL);
CREATE TABLE
sports_event=# CREATE TABLE requirements (requirement_id SERIAL PRIMARY KEY,sport_name VARCHAR(100) NOT NULL,requirements TEXT NOT NU
LL);
CREATE TABLE
sports_event=# ALTER TABLE feedback ADD COLUMN user_id INT NOT NULL;
ALTER TABLE
sports_event=# ALTER TABLE matches DROP COLUMN match_time,ADD COLUMN start_time TIME NOT NULL,ADD COLUMN end_time TIME NOT NULL;
ERROR:  column "start_time" of relation "matches" contains null values
sports_event=# ALTER TABLE matches DROP COLUMN match_time;
ALTER TABLE
```



```
                   ^
sports_event=# INSERT INTO requirements (sport_name,requirements) values ('Football', '11 players per team. Equipment: Football, Goal
posts, Shin Guards, Football Shoes.'),('Basketball', '5 players per team. Equipment: Basketball, Hoop, Basketball Shoes.'),('Cricket'
, '11 players per team. Equipment: Cricket Bat, Ball, Stumps, Pads, Gloves, Helmet.'),('Volleyball', '6 players per team. Equipment:
Volleyball, Net, Court Shoes.'),('Hockey', '11 players per team. Equipment: Hockey Stick, Ball, Shin Guards, Mouth Guards, Hockey Sho
es.'),('Rugby', '15 players per team. Equipment: Rugby Ball, Mouth Guards, Jerseys, Boots.'),('Baseball', '9 players per team. Equipm
ent: Baseball Bat, Ball, Gloves, Helmet, Bases.'),('Handball', '7 players per team. Equipment: Handball, Goalposts, Shoes.'),('Water
Polo', '7 players per team. Equipment: Water Polo Ball, Caps, Goalposts, Swimwear.'),('Field Lacrosse', '10 players per team. Equipme
nt: Lacrosse Stick, Ball, Helmet, Gloves, Pads.'),('Ice Hockey', '6 players per team. Equipment: Ice Hockey Stick, Puck, Helmet, Pads
, Skates.'),('Softball', '9 players per team. Equipment: Softball Bat, Ball, Gloves, Helmet, Bases.'),('Kabaddi', '7 players per team
. No major equipment required.'),('Ultimate Frisbee', '7 players per team. Equipment: Frisbee, Cleats, Field Markers.'),('Badminton (
Team)', '2 players per team (Doubles). Equipment: Badminton Rackets, Shuttlecock, Net, Court Shoes.');
INSERT 0 15
sports_event=# select * from users;
 user_id |  username   |                           password                           |        email         |   phone    |   role

---------+-------------+--------------------------------------------------------------+----------------------+------------+----------
-
       2 | vl_darshini | $2y$10$CHK/PUuWJYHKbvxhjATctO80bnqALIo5j1qlzMkhN3uV0xwBEAWqi | darshini09@gmail.com | 9940765784 | organizer
       6 | aswini      | $2y$10$vsyDW96Hl2cZ2GtczBvyR.zeaJefw2OmQybXGrLohgmfEQkrI10kS | aswini09@gmail.com   | 2345676521 | organizer
       7 | aadil       | $2y$10$swUlI2nl.tYNuf2v7J6v8uwiHWGy8z4cbCtFBiI0LQ5jkw.bXYv0a | aadil27@gmail.com    | 7485494821 | organizer
       8 | arshad      | $2y$10$cJB7PhQvOV8LdsB0D6KKZeO7kn.O1zG3YM6Xnu4xUDGXXJXpGRWca | arshad09@gmail.com   | 9789335114 | organizer
(4 rows)


sports_event=# select * from users;
 user_id |  username   |                           password                           |        email         |   phone    |    role

---------+-------------+--------------------------------------------------------------+----------------------+------------+----------
---
       2 | vl_darshini | $2y$10$CHK/PUuWJYHKbvxhjATctO80bnqALIo5j1qlzMkhN3uV0xwBEAWqi | darshini09@gmail.com | 9940765784 | organizer
       6 | aswini      | $2y$10$vsyDW96Hl2cZ2GtczBvyR.zeaJefw2OmQybXGrLohgmfEQkrI10kS | aswini09@gmail.com   | 2345676521 | organizer
       7 | aadil       | $2y$10$swUlI2nl.tYNuf2v7J6v8uwiHWGy8z4cbCtFBiI0LQ5jkw.bXYv0a | aadil27@gmail.com    | 7485494821 | organizer
```

HTML+PHP CODE:

db_connection.php

```
<?php
$conn = pg_connect("host=localhost dbname=sports_event user=postgres password=Afrah#27");
if (!$conn) {
    die("Connection failed: " . pg_last_error());
}
?>
```

login.php

```
<?php
session_start();
require 'db_connection.php';
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = $_POST['username'];
    $password = $_POST['password'];
    $query = "SELECT * FROM users WHERE username = $1";
    $result = pg_query_params($conn, $query, array($username));
    if ($result && pg_num_rows($result) > 0) {
        $user = pg_fetch_assoc($result);
        if (password_verify($password, $user['password'])) {
            $_SESSION['username'] = $user['username'];
            $_SESSION['role'] = $user['role'];
            if ($user['role'] === 'participant') {
                header('Location: home.php');
            } else {
                header('Location: organizer_home.php');
            }
        } else {
            echo "Invalid credentials!";
        }
    } else {
        echo "User not found!";
    }
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Login</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="form-container">
        <form action="login.php" method="POST">
            <h2>Login</h2>
            <input type="text" name="username" placeholder="Username" required>
            <input type="password" name="password" placeholder="Password" required>
            <button type="submit">Login</button>
            <p><a href="signup.php">Sign Up</a></p>
        </form>
    </div>
</body>
```

```
</html>
```

signup.php

```
<?php

require 'db_connection.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {

    $username = $_POST['username'];

    $password = password_hash($_POST['password'], PASSWORD_BCRYPT);

    $email = $_POST['email'];

    $phone = $_POST['phone'];

    $role = $_POST['role'];

    $query = "INSERT INTO users (username, password, email, phone, role) VALUES ($1, $2, $3,
$4, $5)";

    $result = pg_query_params($conn, $query, array($username, $password, $email, $phone,
$role));

    if ($result) {

        header('Location: login.php');

    } else {

        echo "Error: " . pg_last_error($conn);

    }

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Sign Up</title>

    <link rel="stylesheet" href="styles.css">

</head>
```

```
<body>
    <div class="form-container">
        <form action="signup.php" method="POST">
            <h2>Sign Up</h2>
            <input type="text" name="username" placeholder="Username" required>
            <input type="password" name="password" placeholder="Password" required>
            <input type="email" name="email" placeholder="Email" required>
            <input type="text" name="phone" placeholder="Phone" required>
            <select name="role" required>
                <option value="participant">Participant</option>
                <option value="organizer">Organizer</option>
            </select>
            <button type="submit">Sign Up</button>
        </form>
    </div>
</body>
</html>
```

home.php

```
<?php
session_start();
require 'db_connection.php';
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = $_SESSION['username'];
$user_query = "SELECT user_id FROM users WHERE username = $1";
$user_result = pg_query_params($conn, $user_query, array($username));
if ($user_row = pg_fetch_assoc($user_result)) {
        $leader_id = $user_row['user_id'];

        $leader_name = $_POST['leader_name'];

        $leader_age = $_POST['leader_age'];

        $leader_gender = $_POST['leader_gender'];

        $leader_dept = $_POST['leader_dept'];

        $sport_name = $_POST['sport_name'];

        $team_name = $_POST['team_name'];

 $insert_query = "INSERT INTO teams (leader_id, leader_name, leader_age, leader_gender,
leader_dept, sport_name, team_name) VALUES ($1, $2, $3, $4, $5, $6, $7)";

        $insert_result = pg_query_params($conn, $insert_query, array($leader_id, $leader_name,
$leader_age, $leader_gender, $leader_dept, $sport_name, $team_name));

if ($insert_result) {

            echo "<p style='color: green;'>Team registered successfully!</p>";

        } else {

            echo "<p style='color: red;'>Error registering team: " . pg_last_error($conn) . "</p>";

        }

    } else {

        echo "<p style='color: red;'>Error retrieving user ID.</p>";

    }

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="styles.css">
    <title>Team Registration</title>
</head>
<body>
   <div class="login-button">
        <a href="login.php">Login</a>
    </div>
    <h2>Register here!</h2>
    <div class="nav-buttons">
        <a href="calendar.php">Go to Calendars</a>
        <a href="requirements.php">Go to Requirements</a>
        <a href="feedback.php">Go to Feedback</a>
    </div>
    <form action="home.php" method="POST">
        <label for="leader_name">Leader Name:</label>
        <input type="text" name="leader_name" required>
        <label for="leader_age">Leader Age:</label>
        <input type="number" name="leader_age" required>
        <label for="leader_gender">Leader Gender:</label>
        <select name="leader_gender" required>
            <option value="Male">Male</option>
            <option value="Female">Female</option>
            <option value="Other">Other</option>
        </select>
        <label for="leader_dept">Leader Department:</label>
        <input type="text" name="leader_dept" required>
        <label for="sport_name">Sport Name:</label>
        <select name="sport_name" required>
            <option value="Cricket">Cricket</option>
            <option value="Football">Football</option>
            <option value="Basketball">Basketball</option>
            <option value="Hockey">Hockey</option>
            <option value="Volleyball">Volleyball</option>
            <option value="Rugby">Rugby</option>
            <option value="Badminton(Team)">Badminton(Team)</option>
            <option value="Baseball">Baseball</option>
            <option value="Handball">Handball</option>
            <option value="Waterpolo">Wterpolo</option>
            <option value="Field Lacrosse">Field Lacrosse</option>
            <option value="Ice Hockey">Ice Hockey</option>
            <option value="Softball">Softball</option>
            <option value="Kabaddi">Kabaddi</option>
            <option value="Ultimate Frisbee">Ultimate Frisbee</option>
        </select>
        <label for="team_name">Team Name:</label>
        <input type="text" name="team_name" required>
        <button type="submit">Register Team</button>
    </form>
</body>
</html>
```

calendars.php

```
<?php
session_start();
```

```
require 'db_connection.php';

$username = $_SESSION['username'];

$query_user = "SELECT user_id FROM users WHERE username = $1";

$result_user = pg_query_params($conn, $query_user, array($username));

if ($result_user && pg_num_rows($result_user) > 0) {

    $user = pg_fetch_assoc($result_user);

    $user_id = $user['user_id'];

    $query_team = "SELECT sport_name FROM teams WHERE leader_id = $1";

    $result_team = pg_query_params($conn, $query_team, array($user_id));

    if ($result_team && pg_num_rows($result_team) > 0) {

        $team = pg_fetch_assoc($result_team);

        $sport_name = $team['sport_name'];

        $query_matches = "

            SELECT *

            FROM matches

            WHERE sport_name = $1

            AND match_date >= CURRENT_DATE

            ORDER BY match_date ASC, start_time ASC";

        $matches_result = pg_query_params($conn, $query_matches, array($sport_name));

    }

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <title>Calendar</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h2>Upcoming Matches for <?= htmlspecialchars($sport_name) ?></h2>
    <div class="nav-buttons">
        <a href="home.php">Go to Home</a>
        <a href="feedback.php">Go to Feedback</a>
        <a href="requirements.php">Go to Requirements</a>
    </div>
    <div class="matches-container">
        <?php if (isset($matches_result) && pg_num_rows($matches_result) > 0): ?>
            <?php while ($match = pg_fetch_assoc($matches_result)): ?>
                <div class="match-box">
                    <p><strong>Date:</strong> <?= htmlspecialchars($match['match_date']) ?></p>
                    <p><strong>Time:</strong> <?= htmlspecialchars($match['start_time']) ?> to <?=
htmlspecialchars($match['end_time']) ?></p>
                    <p><strong>Venue:</strong> <?= htmlspecialchars($match['match_venue']) ?></p>
                    <p><strong>Days Left:</strong>
                        <?= date_diff(new DateTime(), new DateTime($match['match_date']))->format('%a
days left') ?>
                    </p>
                </div>
            <?php endwhile; ?>
        <?php else: ?>
            <p>No matches scheduled at the moment.</p>
        <?php endif; ?>
    </div>
```

```
</body>

</html>
```

requirements.php

```
<?php

session_start();

require 'db_connection.php';

$username = $_SESSION['username'] ?? null;

$sport_name = null;

$requirements_result = null;

if ($username) {

    $query_user = "SELECT user_id FROM users WHERE username = $1";

    $result_user = pg_query_params($conn, $query_user, array($username));

    if ($result_user && pg_num_rows($result_user) > 0) {

        $user = pg_fetch_assoc($result_user);

        $user_id = $user['user_id'];

        $query_team = "SELECT sport_name FROM teams WHERE leader_id = $1";

        $result_team = pg_query_params($conn, $query_team, array($user_id));

        if ($result_team && pg_num_rows($result_team) > 0) {

            $team = pg_fetch_assoc($result_team);

            $sport_name = $team['sport_name'];

            $query_requirements = "SELECT * FROM requirements WHERE sport_name = $1";

            $requirements_result = pg_query_params($conn, $query_requirements,
array($sport_name));

        }

    }

}

?>
```

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Requirements</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <div class="nav-buttons">

        <a href="home.php">Go to Home</a>

        <a href="calendar.php">Go to Calendars</a>

        <a href="feedback.php">Go to Feedback</a>

    </div>


    <?php if ($sport_name): ?>

        <h2>Requirements for <?= htmlspecialchars($sport_name) ?></h2>

        <ul>

            <?php if ($requirements_result && pg_num_rows($requirements_result) > 0): ?>

                <?php while ($requirement = pg_fetch_assoc($requirements_result)): ?>

                    <li><?= htmlspecialchars($requirement['requirements']) ?></li>

                <?php endwhile; ?>

            <?php else: ?>

                <li>No requirements found for this sport.</li>

            <?php endif; ?>

        </ul>

    <?php else: ?>
```

```
        <h2>No sport associated with your team.</h2>
    <?php endif; ?>


</body>
</html>
```

feedback.php

```
<?php
session_start();
require 'db_connection.php';
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = $_SESSION['username'];
    $user_query = "SELECT user_id FROM users WHERE username = $1";
    $user_result = pg_query_params($conn, $user_query, array($username));
    if ($user_row = pg_fetch_assoc($user_result)) {
        $user_id = $user_row['user_id'];
        $sport_name = $_POST['sport_name'];
        $feedback = $_POST['feedback'];
  $insert_query = "INSERT INTO feedback (user_id, sport_name, feedback) VALUES ($1, $2,
$3)";
        $insert_result = pg_query_params($conn, $insert_query, array($user_id, $sport_name,
$feedback));
        if ($insert_result) {
            echo "<p style='color: green;'>Feedback submitted successfully!</p>";
        } else {
            echo "<p style='color: red;'>Error submitting feedback: " . pg_last_error($conn) . "</p>";
        }
    } else {
        echo "<p style='color: red;'>Error retrieving user ID.</p>";
    }
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Feedback</title>
</head>
<body>
    <div class="nav-buttons">
        <a href="home.php">Go to Home</a>
        <a href="calendar.php">Go to Calendars</a>
        <a href="requirements.php">Go to Requirements</a>
    </div>
    <h2>Submit Your Feedback</h2>
    <form action="feedback.php" method="POST">
        <label for="sport_name">Sport Name:</label>
        <input type="text" name="sport_name" required>
        <label for="feedback">Your Feedback:</label>
        <textarea name="feedback" required></textareaz
        <button type="submit">Submit Feedback</button>
    </form>
```

```
</body>
</html>
```

organizers_home.php

```
<?php
session_start();
if (!isset($_SESSION['username']) || $_SESSION['role'] !== 'organizer') {
    header('Location: login.php');
    exit();
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Organizer Home</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="login-button">
        <a href="login.php">Login</a>
    </div>
    <h2>Welcome, <?= $_SESSION['username'] ?></h2>
    <div class="nav-buttons">
        <a href="add_match.php">Add Match</a>
        <a href="view_participants.php">View Participants</a>
        <a href="view_feedback.php">View Feedback</a>
        <a href="delete_participant.php">delete participant</a>

    </div>

</body>

</html>
```

add_match.php

```
<?php

session_start();

require 'db_connection.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {

    $sport_name = $_POST['sport_name'];

    $match_date = $_POST['match_date'];

    $start_time = $_POST['start_time'];

    $end_time = $_POST['end_time'];

    $match_venue = $_POST['match_venue'];

    $query = "INSERT INTO matches (sport_name, match_date, start_time, end_time,
match_venue) VALUES ($1, $2, $3, $4, $5)";

    $result = pg_query_params($conn, $query, array($sport_name, $match_date, $start_time,
$end_time, $match_venue));



    if ($result) {

        echo "Match added successfully!";

    } else {

        echo "Error: " . pg_last_error($conn);

    }

}

?>

<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <title>Add Match</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="nav-buttons delete-page-nav">
        <a href="organizer_home.php">Back to Home</a>
    </div>
    <form action="add_match.php" method="POST">
        <h2>Add Match</h2>
        <input type="text" name="sport_name" placeholder="Sport Name" required>
        <input type="date" name="match_date" required>
        <input type="time" name="start_time" required>
        <input type="time" name="end_time" required>
        <input type="text" name="match_venue" placeholder="Venue" required>
        <button type="submit">Add Match</button>
    </form>
</body>
</html>
```

view_participants.php

```
<?php
session_start();
require 'db_connection.php';


$query = "SELECT * FROM users WHERE role = 'participant'";
```

```
$participants_result = pg_query($conn, $query);
?>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>View Participants</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h2>Participants List</h2>
    <div class="nav-buttons delete-page-nav">
        <a href="organizer_home.php">Back to Home</a>
    </div>


    <div class="feedback-container">
        <?php while ($participant = pg_fetch_assoc($participants_result)): ?>
            <div class="feedback-item">
                <strong><?= htmlspecialchars($participant['username']) ?></strong><br>
                Email: <?= htmlspecialchars($participant['email']) ?><br>
                Phone: <?= htmlspecialchars($participant['phone']) ?>
            </div>
             <?php endwhile; ?>
    </div>
```

```
  </body>

</html>
```

view_feedback.php

```
<?php

session_start();

require 'db_connection.php';

$query = "SELECT feedback.feedback, users.username FROM feedback JOIN users ON
feedback.user_id = users.user_id";

$feedback_result = pg_query($conn, $query);

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>View Feedback</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <h2>Feedback</h2>

    <div class="nav-buttons delete-page-nav">

        <a href="organizer_home.php">Back to Home</a>

    </div>

    <div class="feedback-container">

        <?php while ($feedback = pg_fetch_assoc($feedback_result)): ?>

            <div class="feedback-item">
```

```
                <strong><?= htmlspecialchars($feedback['username']) ?>:</strong> <?=
htmlspecialchars($feedback['feedback']) ?>

            </div>

        <?php endwhile; ?>

    </div>

   </body>

</html>
```

match_cleanup.php

```
<?php

session_start();

require 'db_connection.php';

if (!isset($_SESSION['username']) || $_SESSION['role'] !== 'organizer') {

    header('Location: login.php');

    exit();

}

$current_date = date('Y-m-d');

$current_time = date('H:i:s');

$query = "DELETE FROM matches WHERE match_date < $1 OR (match_date = $1 AND
end_time < $2)";

$result = pg_query_params($conn, $query, array($current_date, $current_time));

if ($result) {

    echo "Expired matches deleted successfully!";

} else {

    echo "Error cleaning up matches: " . pg_last_error($conn);

}

?>
```

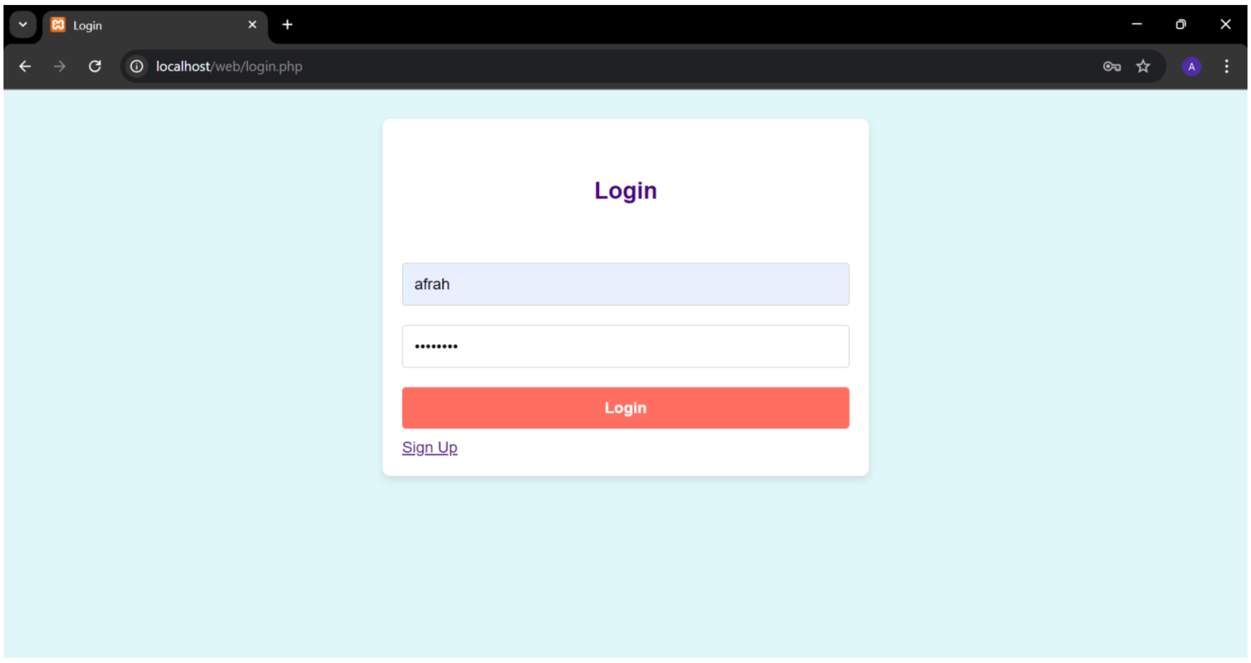
styles.css

```
*{
   margin: 0;
   padding: 0;
   box-sizing: border-box;
   font-family: Arial, sans-serif;
}
body {
   background-color: #e0f7fa;
   color: #2c3e50;
   font-size: 16px;
   display: flex;
   flex-direction: column;
   align-items: center;
   margin: 0;
   padding: 0;
}

h1, h2 {
   text-align: center;
   margin-top: 40px;
   color: #4b0082;
   font-size: x-large;
   font-weight: 1200;
   margin-bottom: 50px;
}
```

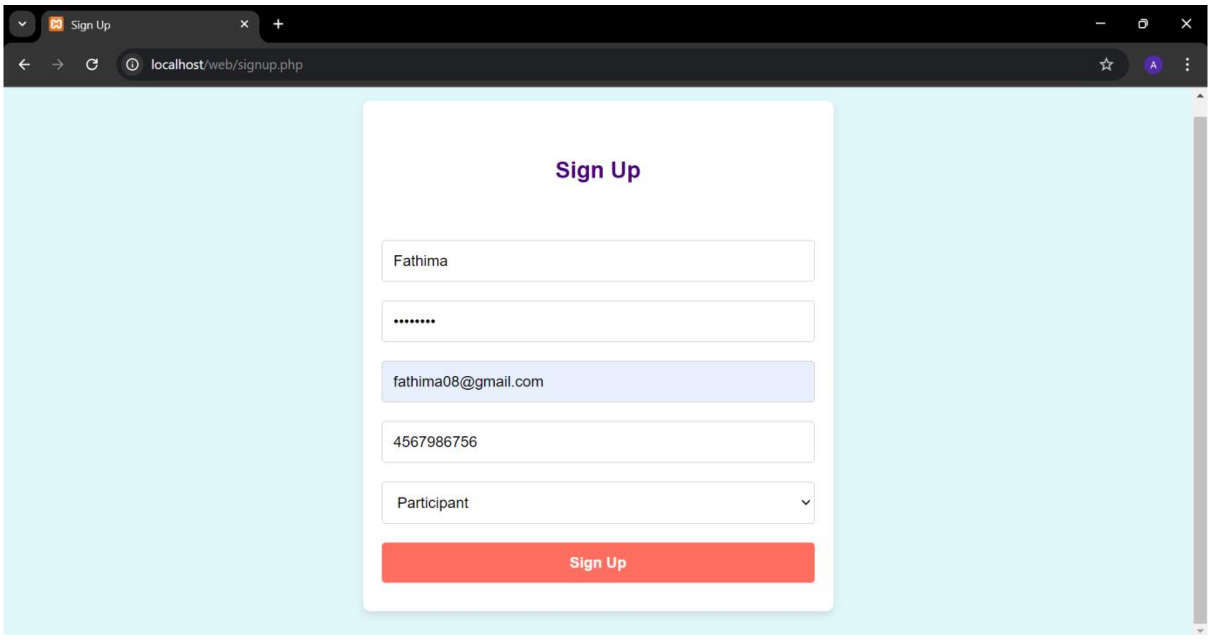
```
.nav-buttons {
    text-align: center;
    margin-top: 20px;
}
.nav-buttons a {
    display: inline-block;
    padding: 10px 20px;
    margin: 0 10px;
    background-color: #ff6f61;
    color: white;
    text-decoration: none;
    font-weight: bold;
    border-radius: 4px;
    transition: background-color 0.3s ease;
}
.nav-buttons a:hover {
    background-color: #ffa07a;
}
.delete-page-nav {
    display: flex;
    justify-content: center;
    margin-top: 20px;
}
.delete-page-nav a {
    padding: 10px 20px;
    background-color: #ff6f61;
    color: white;
    text-decoration: none;
    font-weight: bold;
    border-radius: 4px;
    transition: background-color 0.3s ease;
}
.delete-page-nav a:hover {
    background-color: #ffa07a;
}
form {
    max-width: 500px;
    margin: 30px auto;
    padding: 20px;
    background-color: #fff;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    border-radius: 8px;
}
input, select, textarea, button {
    width: 100%;
    padding: 12px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 4px;
    font-size: 16px;
}
button {
    background-color: #ff6f61;

    color: white;

    cursor: pointer;

    font-weight: bold;

    border: none;

    transition: background-color 0.3s ease;

}

button:hover {

    background-color: #ffa07a;

}

.feedback-container {

    display: flex;

    flex-direction: column;

    align-items: center;

    margin-top: 40px;

}

.feedback-item {

    background-color: #fff;

    padding: 15px;

    border-radius: 8px;

    margin: 10px 0;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

    width: 80%;

    max-width: 600px;

}

.participants-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-top: 40px;
}
.participant-item {
    background-color: #fff;
    padding: 15px;
    border-radius: 8px;
    margin: 10px 0;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    width: 80%;
    max-width: 600px;
}
.matches-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-top: 20px;
    width: 100%;
}
.match-box {
    background-color: #fff;
    padding: 20px;
    margin: 15px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    width: 80%;
    max-width: 600px;
    text-align: center;
    transition: transform 0.3s ease;
}
.match-box:hover {
    transform: scale(1.05);
}
.match-box p {
    margin: 10px 0;
}
.login-button {
    position: fixed;
    top: 10px;
    right: 10px;
    z-index: 1000;
}
.login-button a {
    display: inline-block;
    padding: 10px 20px;
    background-color: #007bff;
    color: white;
    text-decoration: none;
    border-radius: 5px;
    font-weight: bold;
    transition: background-color 0.3s ease;
}
.login-button a:hover {
    background-color: #0056b3;
}
@media (max-width: 768px) {
    form, table {
        width: 90%;
        margin: auto;
    }
    button, input, select, textarea {
        font-size: 14px;
    }
    .nav-buttons a {
        margin: 5px 0;
        display: block;
    }
    .feedback-item {
        width: 90%;
    }
    .participant-item {
        width: 90%;
    }
    .match-box {
        width: 90%;
    }
```

```
  .home-button {
      display: flex;
      justify-content: center;
      margin-top: 20px;
      color: #ff6f61;
  }
  .home-button a {
      text-decoration: none;
      padding: 10px 20px;
      background-color: #007bff;
      color: white;
      border-radius: 5px;
      font-size: 18px;
  }
}
```

LOGIN PAGE:



SIGNUP PAGE:



Sign Up

Fathima

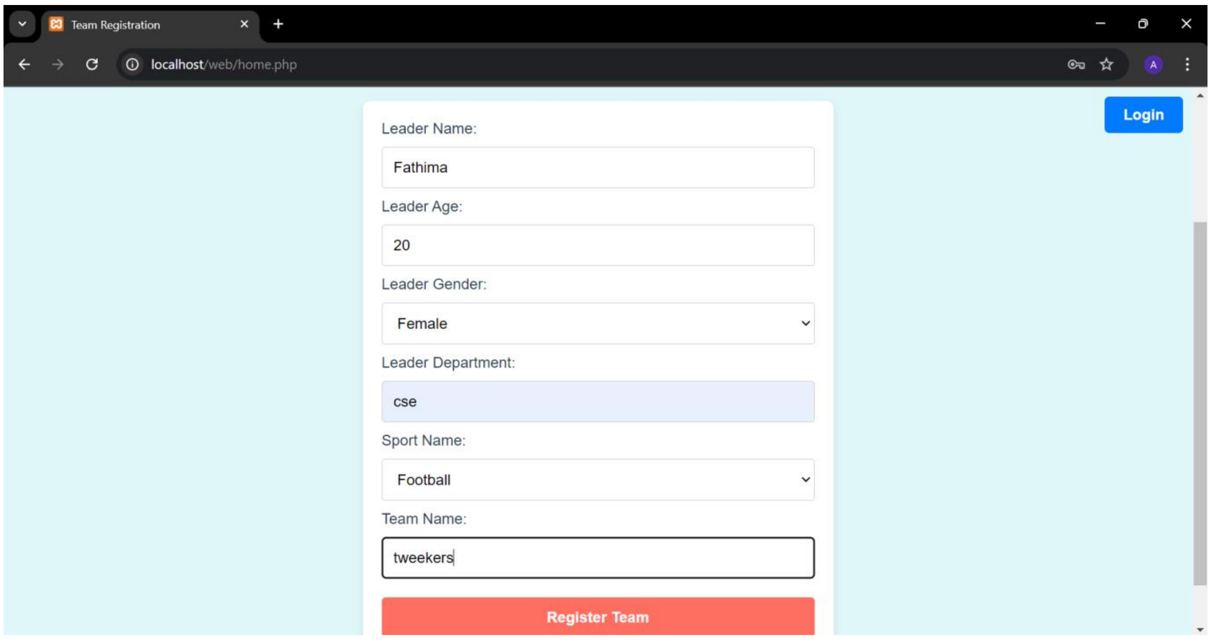
••••••••

fathima08@gmail.com

4567986756

Participant

HOME PAGE



Leader Name:

Fathima

Leader Age:

20

Leader Gender:

Female

Leader Department:

cse

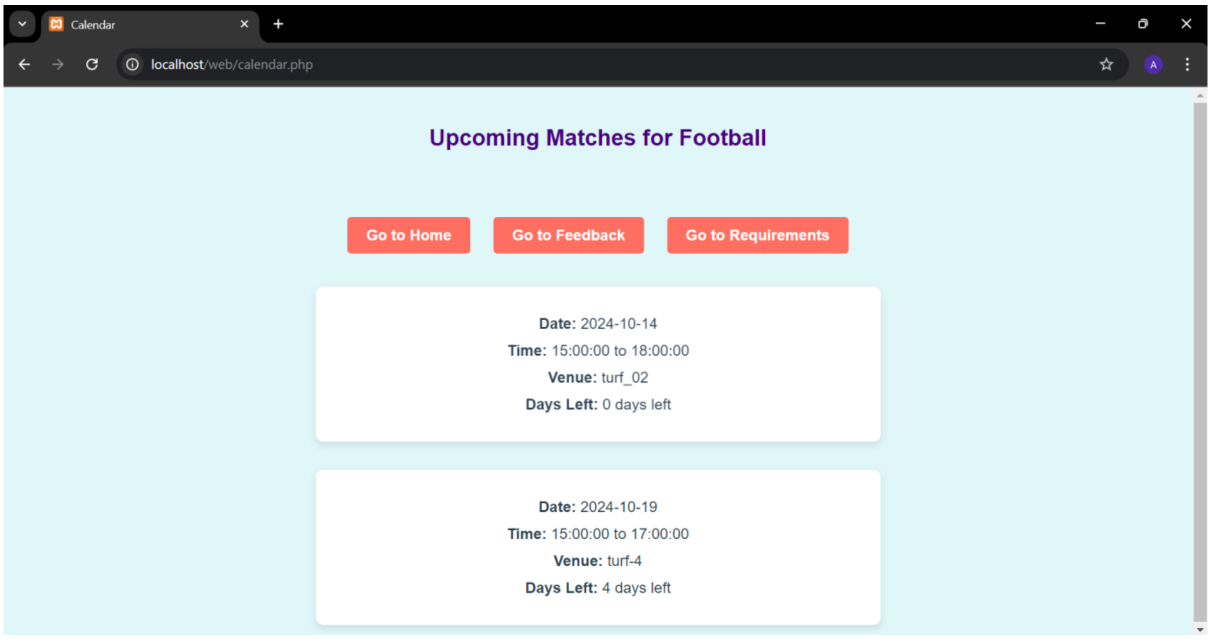
Sport Name:

Football

Team Name:

tweekers

CALENDARS PAGE

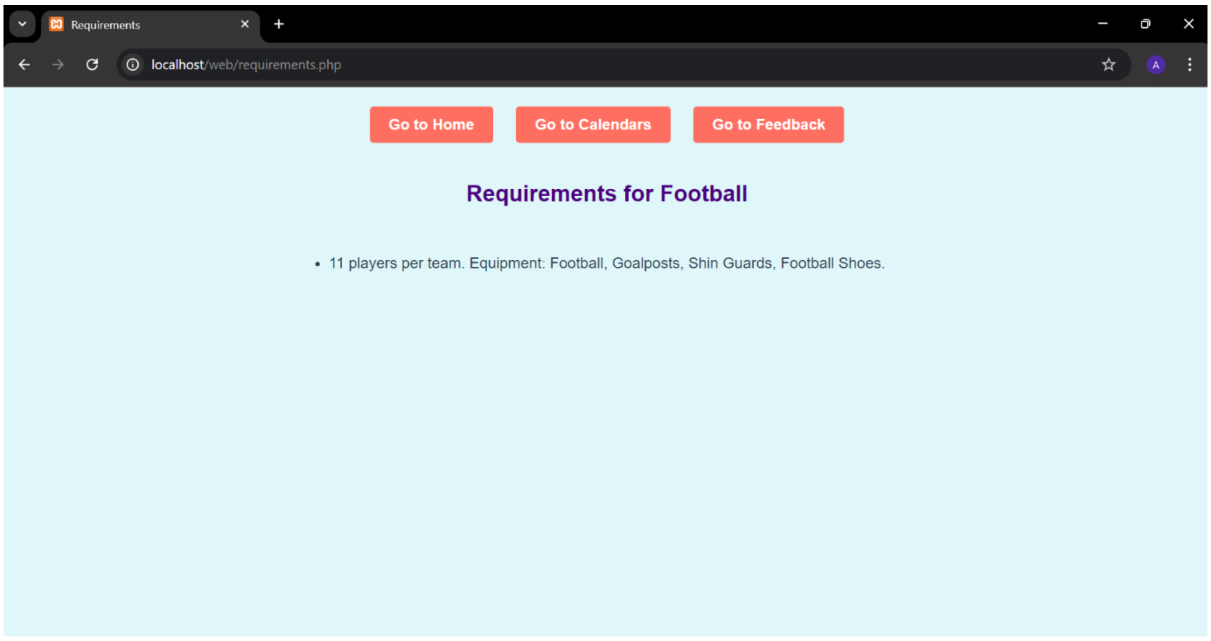


Upcoming Matches for Football

Date: 2024-10-14
Time: 15:00:00 to 18:00:00
Venue: turf_02
Days Left: 0 days left

Date: 2024-10-19
Time: 15:00:00 to 17:00:00
Venue: turf-4
Days Left: 4 days left

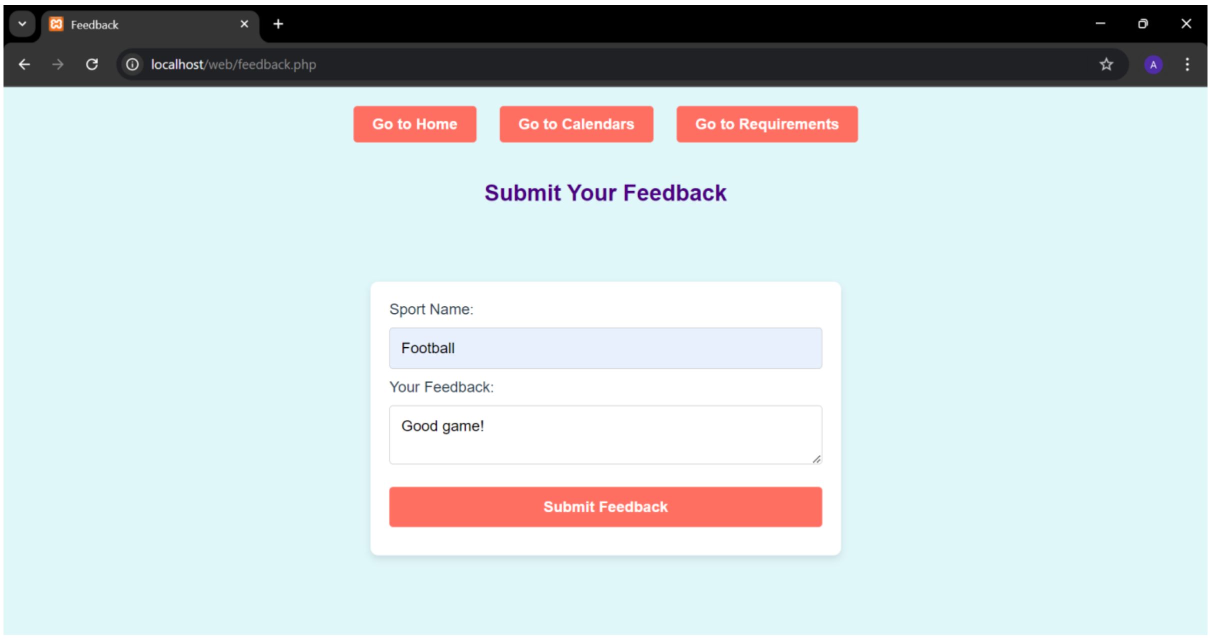
REQUIREMENTS PAGE



Requirements for Football

- 11 players per team. Equipment: Football, Goalposts, Shin Guards, Football Shoes.

FEEDBACK PAGE:

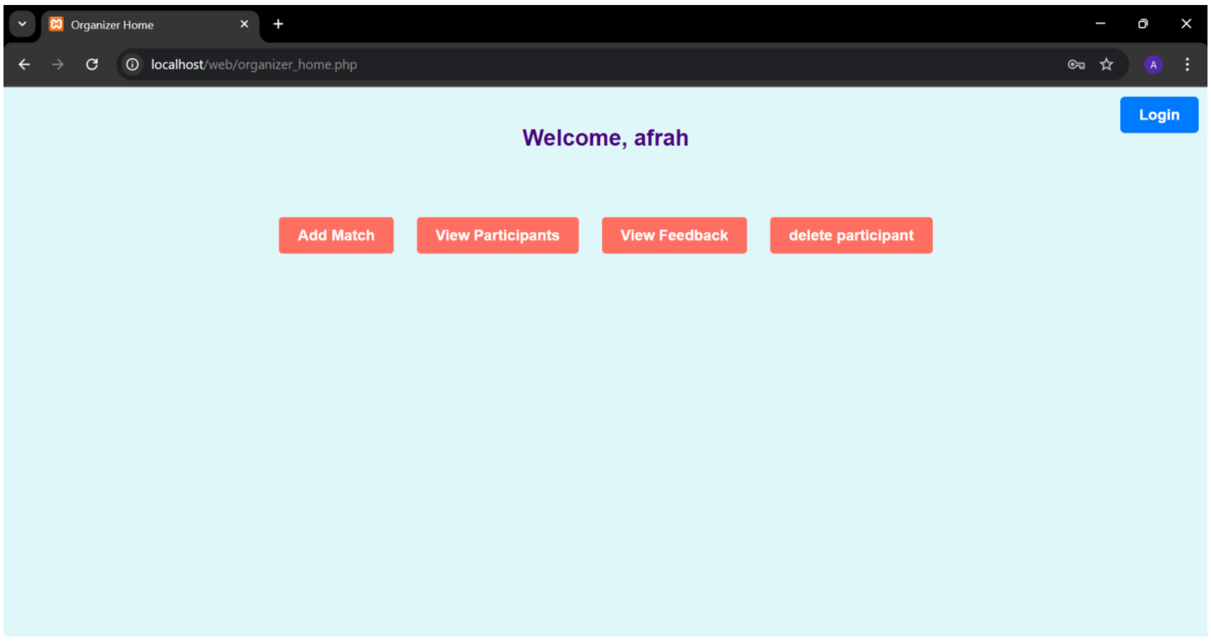


Submit Your Feedback

Sport Name:
Football

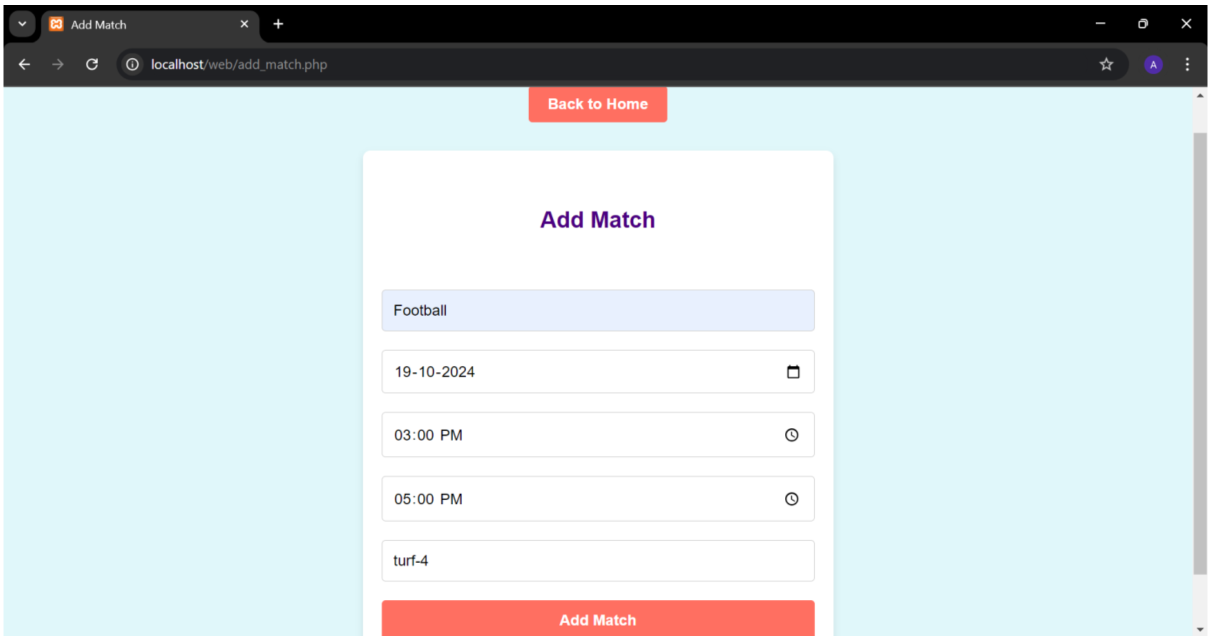
Your Feedback:
Good game!

ORGANIZERS_HOME PAGE:



Welcome, afrah

ADD_MATCH PAGE:



Add Match

Football

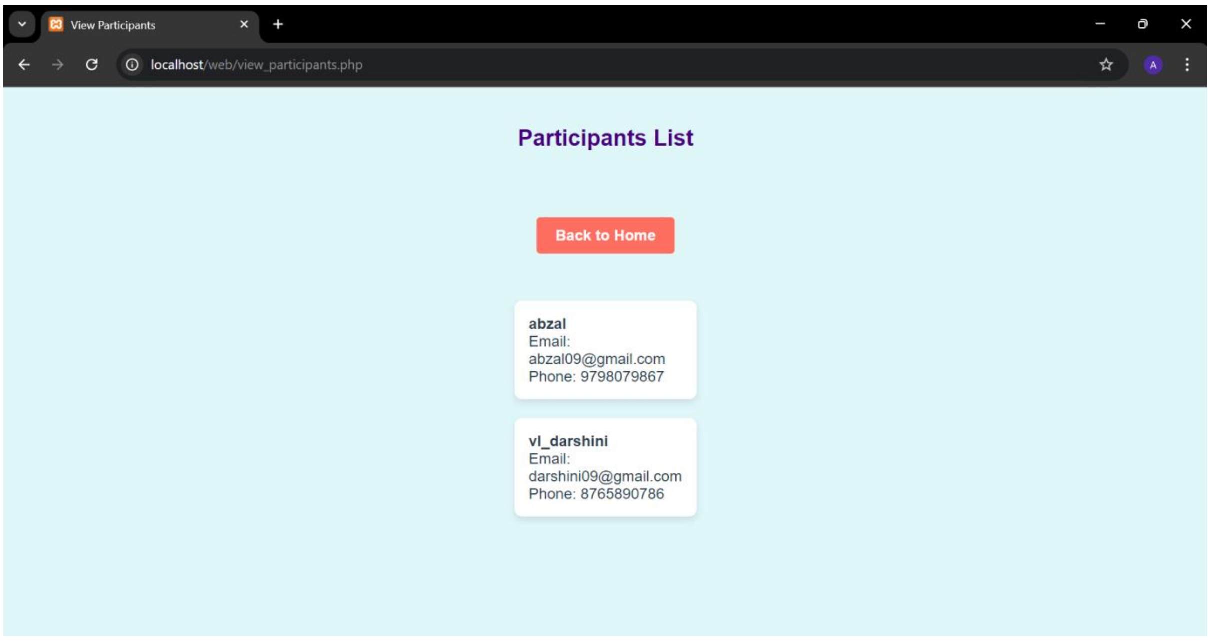
19-10-2024

03:00 PM

05:00 PM

turf-4

VIEW PARTICIPANTS PAGE:

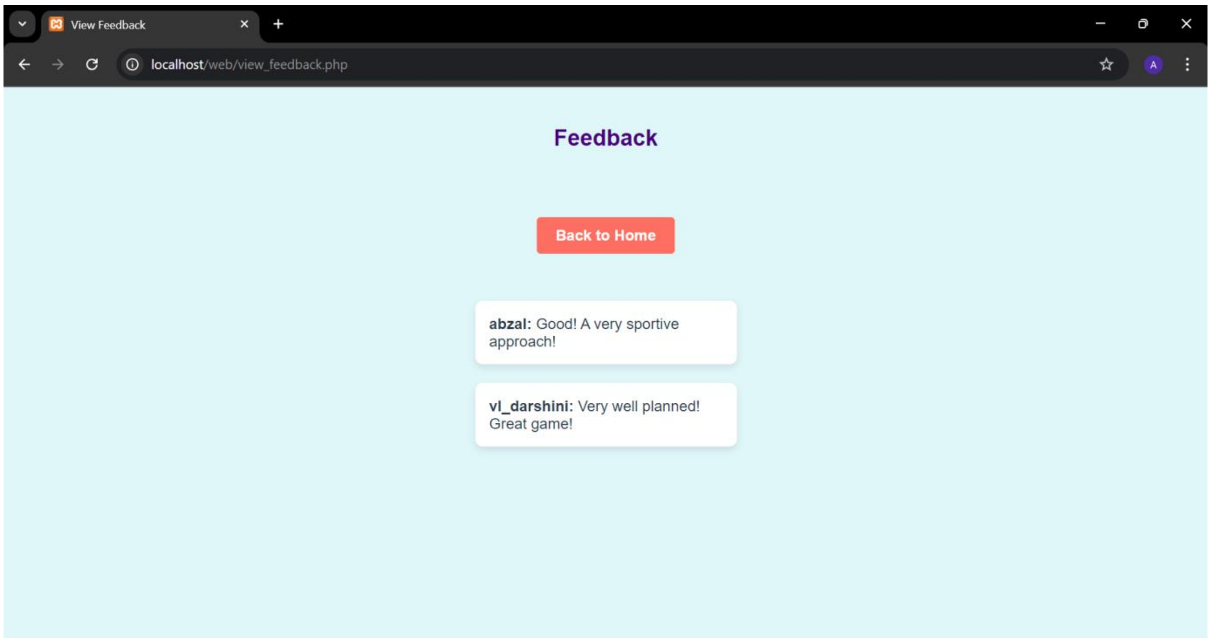


Participants List

abzal
Email:
abzal09@gmail.com
Phone: 9798079867

vl_darshini
Email:
darshini09@gmail.com
Phone: 8765890786

VIEW FEEDBACK PAGE:

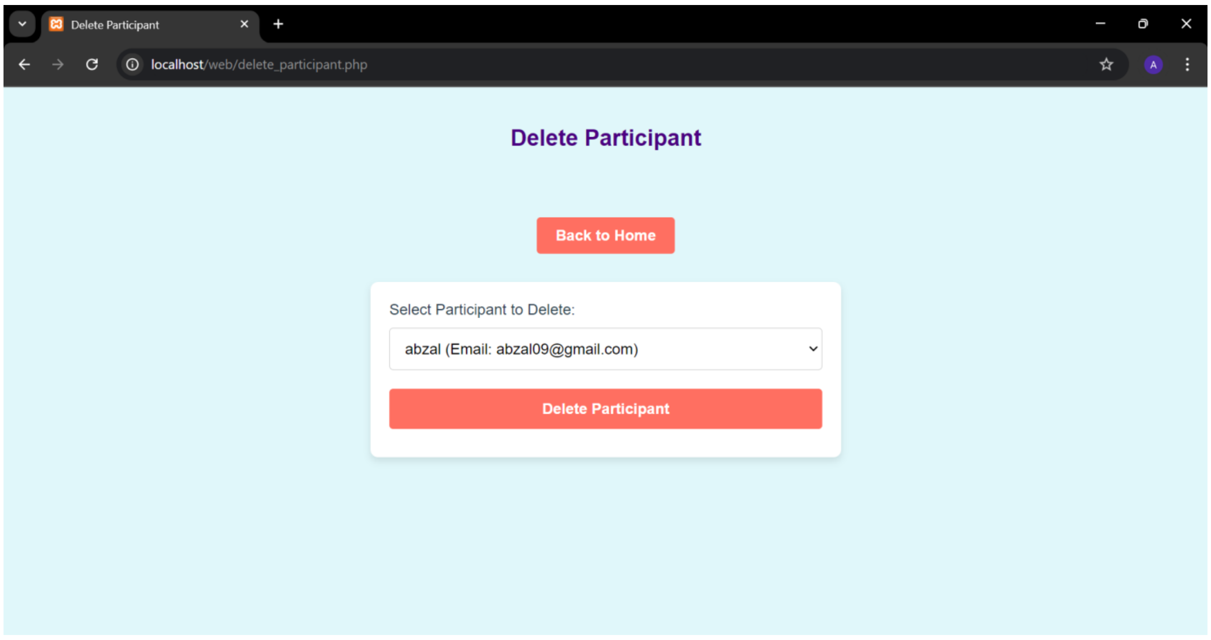


Feedback

abzal: Good! A very sportive approach!

vl_darshini: Very well planned! Great game!

DELETE PARTICIPANTS PAGE



Delete Participant

Select Participant to Delete:

abzal (Email: abzal09@gmail.com)

5.RESULTS AND DISCUSSION

Results and Discussion of the Sports Event Management System Project

The results and discussion section is where you reflect on the outcomes of your sports event management system and analyze its effectiveness, functionality, and potential improvements. This project involves multiple features such as user registration, event management, participant feedback, and more, so discussing these aspects thoroughly is essential.

5.1. Project Objectives Revisited

The primary objectives of the project were:

- To create a web-based system that manages sports events and participants efficiently.
- To provide a platform for participants to register teams, view match details, and provide feedback.
- To allow organizers to manage matches, plan events, and oversee participant data.

The key features of the system include:

- Login/Signup for participants and organizers.
- Participants registration with detailed team information.
- Match Management, allowing organizers to schedule matches with details about the venue, time, and date.
- Requirements Management for each sport, where necessary equipment or arrangements are listed.
- Feedback collection from participants about the event.

Now let’s break down the results achieved in the system:

5.2. Results

5.2.1. User Authentication (Login/Signup)

- Participants: The system successfully implements a secure login/signup mechanism where participants can sign up using their phone number, email, and password. Once signed in, they are directed to a dashboard with relevant options such as registration and viewing match details.
- Organizers: A separate login for organizers is implemented, granting them access to event management options. This segregation ensures that participants do not have access to organizer features, maintaining system security.

Discussion:

- The login/signup feature was designed with validation in place to ensure that no invalid or duplicate entries are accepted. Email and phone numbers are verified for uniqueness. The hashed password storage also adds an additional layer of security.

5.2.2. Participants Registration

Participants are required to provide team leader details and register a team, along with details of other team members, which are stored in the Participants table in PostgreSQL. The form captures:

- Team leader name, gender, department, and contact number.
- Sport name.
- Team name (unique for each team).
- Details of each team member (name, gender, department).

Discussion:

- The registration process was tested for performance, and data was successfully entered into the database. By normalizing the Participants table, data redundancy was minimized.
- The First Normal Form (1NF) and Second Normal Form (2NF) were implemented, where team members' details are stored in separate records, ensuring each piece of data is atomic.
- Challenges faced included validation of unique team names and managing cases where the same participant might be in multiple teams.

5.2.3. Match Management

Organizers can schedule matches, specifying the sport, date, time, and venue. Once matches are scheduled, the information is automatically updated in the Match table and displayed on the participants' calendar page. This prevents double booking of matches or venues.

Discussion:

- Match Duplication: The system checks for duplicates and prevents organizers from scheduling overlapping matches. This was implemented using unique constraints in the database, ensuring that no two matches for the same sport are scheduled at the same time and venue.
- The calendar display was successfully implemented using HTML and CSS to show upcoming matches, ensuring that participants can easily access this information.
- A Potential Improvement could be adding an edit feature for organizers to update match details dynamically and notify participants of changes.

5.2.4. Requirements Display

For each match, the necessary requirements (e.g., equipment like balls, nets) are fetched from the Requirements table and displayed on the participants' side, ensuring they are aware of what is needed on the day of the event.

Discussion:

- Requirements management performed well in the system. The system was able to dynamically fetch and display requirements for each sport based on the Requirements table, which was normalized to prevent data redundancy.

- An issue encountered was ensuring that all sports had their requirements listed properly. During testing, sports without requirements in the database were flagged, helping organizers to complete the data.

5.2.5. Feedback Collection

Participants are prompted to provide feedback on their experience after the event. The Feedback table stores the unique team name and corresponding feedback. This data helps organizers improve the future events.

Discussion:

- The feedback feature allowed participants to submit their responses, which were then displayed on the organizer's dashboard. This feature helps organizers to track user satisfaction.
- During testing, multiple feedback entries were allowed per team. One Potential Improvement could be limiting the feedback to one entry per team or adding a feedback update feature.

5.3. Performance Analysis

5.3.1. Database Performance

The system uses PostgreSQL as its database management system, and through normalization, redundancy was minimized. The database handled concurrent reads and writes efficiently, particularly when multiple participants were registering or when organizers were scheduling events.

- Advantages:
  - The normalized database structure reduced unnecessary duplication of data, ensuring better storage efficiency.
  - Query performance was optimal as the system was able to fetch match details and participant information quickly.
- Challenges:
  - While the system performed well with a limited dataset, scaling up the system with a larger number of participants and events could require optimization, particularly in terms of query performance.

5.3.2. Frontend Performance

The frontend was designed using HTML and CSS, ensuring a responsive and user-friendly interface. It was simple but functional, allowing users to access the necessary features easily.

- Advantages:
  - The navigation between participant and organizer sections was seamless.
  - The registration forms and match calendar were intuitive and easy to use.
- Challenges:

- More interactive features such as real-time notifications for participants or match reminders could improve the overall experience.

5.4. Key Takeaways

1. User Experience: The project effectively meets the user experience goals for both participants and organizers, providing intuitive navigation, secure login, and a smooth registration process.
2. Data Integrity: By applying normalization techniques, the database maintains high data integrity and reduces redundancy, ensuring efficient data management.
3. Security: User authentication and data protection measures (such as password hashing) ensure that the system is secure from unauthorized access.
4. Future Enhancements: There is room for further improvements:
   - Adding a feature for organizers to notify participants about match updates.
   - Implementing feedback analytics for organizers to assess the quality of events.
   - Improving frontend design for a more modern user experience.
5. Scalability: As the system grows with more events and participants, optimization of database queries and the introduction of more efficient scheduling algorithms may be needed.

6. CONCLUSION

Conclusion of the Sports Event Management System Project

The sports event management system was designed and developed to streamline the organization and management of sports events by providing separate functionalities for participants and organizers. By leveraging HTML and CSS for the frontend, and PostgreSQL for the backend, the system achieved the following:

1. User-Friendly Interface: The system offered an intuitive and simple interface for both participants and organizers. Participants could easily register their teams, view match schedules, and submit feedback, while organizers could efficiently manage events and participant details.
2. Effective Database Design: The use of normalization techniques ensured that the database was well-structured, reducing redundancy and improving data integrity. With the application of First, Second, and Third Normal Forms, the system ensured efficient data storage and retrieval.
3. Efficient Event and Match Management: The organizer features allowed for smooth scheduling of events, including setting dates, times, and venues for matches. The system also avoided duplication of entries and ensured that participants were well-informed about the matches.
4. Feedback Mechanism: The feedback system allowed participants to provide their opinions, helping organizers to assess event quality and make improvements in the future.
5. Scalability and Security: The system was built to handle multiple events and participants, with proper user authentication and data validation in place to ensure secure access and data management.

Future Scope:

- Real-time Notifications: Implementing notifications to alert participants about match updates or reminders.
- Enhanced Feedback Analysis: Using analytical tools to provide organizers with insights into participant feedback.
- Advanced Scheduling: Introducing more dynamic scheduling options for organizers to handle overlapping or complex events.

7. REFERENCES

7.1. PostgreSQL Official Documentation

This is the official documentation for PostgreSQL, providing detailed insights into SQL queries, database design, and management.

- Link: https://www.postgresql.org/docs/

7.2. W3Schools: HTML & CSS

W3Schools provides comprehensive tutorials and examples for HTML and CSS, which were used in building the front-end of the project.

- HTML Reference: https://www.w3schools.com/html/
- CSS Reference: https://www.w3schools.com/css/

7.3. PHP Official Documenta

tion

If you are using PHP for the backend integration with PostgreSQL, this official guide will help you understand the language better.

- Link: https://www.php.net/docs.php

7.4. Python-PostgreSQL Integration (Psycopg2)

For projects that use Python as the integration layer with PostgreSQL, Psycopg2 is a popular library. This documentation explains how to connect Python with PostgreSQL.

- Link: https://www.psycopg.org/docs/

7.5. Normalization in DBMS - GeeksforGeeks

This article explains normalization in DBMS, including the different normal forms which were applied to your project.

- Link: https://www.geeksforgeeks.org/normalization-in-dbms/

7.6. Bootstrap for Frontend Design

Bootstrap is a CSS framework that can be used to create responsive and modern front-end designs. This can be helpful if you want to upgrade your current design.

- Link: https://getbootstrap.com/docs/

7.7. MDN Web Docs (Mozilla)

MDN provides an in-depth guide to web technologies, including HTML, CSS, and JavaScript. It's a great resource for understanding the front-end components of the project.

- HTML & CSS Documentation: https://developer.mozilla.org/en-US/docs/Web

7.8. OpenClassrooms: Database Design

A step-by-step guide to database design, normalization, and SQL queries, which is valuable for understanding database creation for your project.

- Link: https://openclassrooms.com/en/courses/4614926-design-a-relational-database

7.9. Stack Overflow

For any programming or database-related issues you may face, Stack Overflow is a reliable source of community-driven solutions.

- Link: https://stackoverflow.com/

7.10. GitHub (For Code Hosting and Collaboration)

GitHub is a popular platform to host, collaborate, and share code repositories. You can use it to manage your project's code.

- Link: https://github.com/

These references will give you access to the documentation and resources required for successful development, deployment, and understanding of the concepts used in your sports event management system.