Soccer Management Application Documentation

Business Problem to be Solved

The Soccer Management Application addresses the challenge of efficiently managing soccer teams and players for sports managers and administrators. It enables tracking players, teams, positions, and traits while streamlining operations such as adding, deleting, and viewing detailed records. The application ensures organized management, reduces manual effort, and increases data accuracy.

Executive Summary

The Soccer Management Application provides a web-based interface for managing soccer teams and players. Users can add new players, view team rosters, manage traits, and delete records seamlessly. Built using Flask as the backend and MySQL for data storage, the solution integrates essential functionalities such as team management, player trait assignments, and record updates, enhancing sports data management.

Technical Specifications

Technology Stack:

- **Backend**: Flask (Python-based micro web framework)
- **Database**: MySQL
- **Frontend**: HTML templates rendered using Jinja2
- **Libraries**: `flask_mysqldb` for MySQL connectivity

Configuration:

- MySQL setup with database `SoccerManager`.

Application configuration for database credentials:
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = '7804'

- app.config['MYSQL_DB'] = 'SoccerManager'

Main Functionalities:

- **View Players**: Displays detailed information about players sorted by team and name.
- **View Teams**: Lists all soccer teams with related information.
- **Add Player**: Adds a new player to the database with team and position assignments.
- **Manage Players**: Allows deleting a player and their related traits.
- **View Team Roster**: Displays players for a specific team, organized by position.
- **View Player Traits**: Shows traits associated with a player.

Source File Descriptions

`app.py`:

- Contains the Flask application logic.
- Defines all routes for managing players, teams, and traits.
- Connects to the MySQL database and performs SQL queries.

Templates Folder:

- `main_menu.html`: The application home page.
- `view_players.html`: Displays the list of players in a table.
- `teams.html`: Lists all teams and their details.
- `add_player.html`: Form page for adding a new player.
- `view_team.html`: Shows detailed team information including players and tactics.
- `view_player_traits.html`: Displays traits for a specific player.
- `manage_players.html`: Interface to delete players and search for records.
- `success.html`: Confirmation page after successful player deletion.

Database Schema:

- **Tables**:
  - `Players`: Player details (PlayerID, Name, Age, Height, etc.).
  - `Teams`: Team details (TeamID, Name, Coach, Colors, etc.).
  - `Positions`: Positions (PositionID, PositionName).
  - `PlayerTraits`: Available traits for players (TraitID, Name).
  - `AcquiredTraits`: Many-to-one relationship between players and traits.
  - `AcquiredTeamTactics`: Many-to-one relationship between teams and tactics.

- **Relationships**:
  - `Players` → Foreign keys for `PositionID` and `TeamID`.
  - `AcquiredTraits` → Links `Players` to `PlayerTraits`.
  - `AcquiredTeamTactics` → Links `Teams` to team tactics.

Database Utilization

- All data, including player stats, team information, and traits, are stored in MySQL tables.
- SQL queries fetch, insert, or delete data based on user actions:
  - `SELECT` statements are used for viewing data.
  - `INSERT` handles adding players and their traits.
  - `DELETE` allows managing players.

How to Run the Application

Setup:

1. Install MySQL and create the `SoccerManager` database.
2. Load the provided SQL schema to set up the tables.

Install Dependencies:

pip install flask flask-mysqldb

Run the Application:

1. Start the Flask server:
   python app.py
2. Access the application at `http://localhost:5000`.

Navigate:

- Use the home page to access options like "View Players," "View Teams," and "Add Player."
- Use forms to add or delete records seamlessly.

Issues and Blockers

MySQL Connector for Unity:

- Faced significant challenges integrating MySQL with Unity due to compatibility issues with multiple plugins.
- Attempted various solutions across both macOS and Windows environments, including testing different MySQL libraries and third-party connectors, but encountered persistent errors with database connectivity.
- Ultimately resolved the issue after extensive troubleshooting and testing by identifying a plugin compatible with the Unity version and ensuring proper configuration of MySQL credentials.

Database Configuration:

- After successfully integrating the MySQL connector, verified database setup and resolved minor credential mismatches.

Edge Cases:

- Addressed issues where player or team data was missing from the database by implementing error handling and user-friendly fallback messages.

Team Contributions

- **Your glorious developer Zack**

Conclusion

The Soccer Management Application successfully resolves the challenges of managing teams and players. By providing streamlined data management functionalities using a robust backend (Flask) and a relational database (MySQL), the application ensures accuracy, efficiency, and scalability for soccer management operations.