Team Management Web Application Documentation

1. Introduction and Objectives

The Team Management Web Application is a Flask-based web platform designed to manage sports teams, players, and matches. Built with Python, Flask, SQLAlchemy, Flask-WTF, and Bootstrap, the application provides a secure, modular, and user-friendly interface for authenticated users to create, edit, view, and delete teams, manage team rosters, and track match details. The primary goal is to offer a scalable and maintainable system for sports team management, emphasizing data integrity and user experience.

Objectives:

- Allow authenticated users to manage their teams and associated data securely.

- Implement a robust database structure to store team, player, and match information.

- Ensure secure operations with user authentication and CSRF protection.

- Provide a responsive UI using Bootstrap for accessibility across devices.

- Organize the codebase using Flask blueprints and OOP principles for maintainability.

2. Description of Features

The application includes the following features:

- User Authentication:

- Users must log in to access team management features, ensuring data privacy and security.

- Supported by a separate authentication blueprint (login, logout, registration).

- Team Management:

- Create Team: Users can create a team by specifying a name, coach, founded year, and an optional logo (JPG or PNG).

- Edit Team: Update team details and replace the logo if needed.

- Delete Team: Remove a team and its associated players (with cascade deletion).

- View Team: Display team details, including a list of players and matches.

- Player Management:

- Add players to a team, with each player linked to a specific team via a foreign key.

- Players are automatically deleted when their team is deleted (cascade).

- Match Management:

- View matches involving a team, with each match linking two teams (team1 and team2).

- Dashboard:

- A centralized view listing all teams owned by the logged-in user.

- Provides quick access to create, edit, delete, and view team actions.

- Security:

- CSRF protection for all form submissions using Flask-WTF.

- Secure file uploads with validation for logo images (JPG/PNG only).

- Sanitized filenames using `secure\_filename` to prevent path traversal attacks.

3. ER Diagram of the Database

The database consists of three main tables: Team, Player, and Match. Below is a textual representation of the Entity-Relationship (ER) diagram, describing the tables and their relationships.

Table: Team

- id: Integer, Primary Key

- name: String, Not Null

- coach: String

- founded\_year: Integer

- user\_id: Integer, Not Null

- logo: String

- Relationships:

- One-to-Many with Player (via player.team\_id)

- Many-to-Many with Match (via match.team1\_id and match.team2\_id)

Table: Player

- id: Integer, Primary Key

- name: String, Not Null

- team\_id: Integer, Foreign Key (references Team.id, ondelete=CASCADE), Not Null

- Relationships:

- Many-to-One with Team (via team\_id)

Table: Match

- id: Integer, Primary Key

- team1\_id: Integer, Foreign Key (references Team.id, ondelete=CASCADE), Not Null

- team2\_id: Integer, Foreign Key (references Team.id, ondelete=CASCADE), Not Null

- Relationships:

- Many-to-One with Team (via team1\_id)

- Many-to-One with Team (via team2\_id)

Relationships

- Team -> Player: A team can have multiple players (one-to-many). When a team is deleted, its players are deleted (CASCADE).

- Team -> Match: A team can participate in multiple matches as either team1 or team2 (many-to-many via team1\_id and team2\_id). When a team is deleted, its matches are deleted (CASCADE).

Textual ER DiagramИзображение выглядит как снимок экрана, текст, 3D-моделирование, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как диаграмма, снимок экрана, План

Контент, сгенерированный ИИ, может содержать ошибки.

4. Explanation of Code Structure

The application is structured using Object-Oriented Programming (OOP) principles and Flask blueprints to ensure modularity, reusability, and maintainability.

OOP Principles

- Encapsulation:

- Models (Team, Player, Match) encapsulate data and relationships using SQLAlchemy ORM. For example, the Team model defines attributes (name, coach, logo) and a relationship to Player (players).

- Forms (TeamForm, DeleteTeamForm) encapsulate form validation logic using Flask-WTF.

- Modularity:

- Routes are grouped into blueprints (team, auth, player) to separate concerns.

- Utility functions like `allowed\_file` encapsulate file validation logic for reuse.

- Reusability:

- Common functionality, such as authentication checks (@login\_required) and file upload handling, is reused across routes.

- Templates inherit from a base template (base.html) for consistent layouts.

Flask Blueprints

The application uses Flask blueprints to organize routes and views:

- team Blueprint:

- Handles team-related operations: /dashboard, /create, /edit, /delete, /view.

- Includes routes for rendering the dashboard, creating/editing teams, deleting teams, and viewing team details.

- auth Blueprint:

- Manages user authentication (login, logout, registration).

- Provides the @login\_required decorator to restrict access to team operations.

- player Blueprint:

- Manages player creation and association with teams (e.g., /player/create/<team\_id>).

Example (team blueprint excerpt):

@team\_bp.route('/dashboard')

@login\_required

def dashboard():

teams = Team.query.filter\_by(user\_id=session['user\_id']).all()

return render\_template('team/dashboard.html', teams=teams)

Directory Structure

project/

├── app.py # Main Flask app configuration

├── models.py # SQLAlchemy models (Team, Player, Match)

├── forms.py # Flask-WTF forms (TeamForm, DeleteTeamForm)

├── templates/ # Jinja2 templates

│ ├── base.html # Base template with Bootstrap

│ ├── team/ # Team-related templates

│ │ ├── dashboard.html

│ │ ├── create.html

│ │ ├── edit.html

│ │ ├── view.html

├── static/ # Static assets

│ ├── uploads/ # Team logo uploads

│ ├── css/ # Custom CSS

│ ├── js/ # Custom JavaScript

├── blueprints/ # Blueprint modules

│ ├── \_\_init\_\_.py

│ ├── auth.py # Authentication routes

│ ├── team.py # Team routes

│ ├── player.py # Player routes

Key Components

- app.py: Initializes the Flask app, configures SQLAlchemy, Flask-WTF, and CSRF protection.

- models.py: Defines Team, Player, and Match models with relationships and cascade deletion.

- forms.py: Defines TeamForm for team creation/editing and DeleteTeamForm for deletion.

- templates: Use Jinja2 with Bootstrap for responsive UI, inheriting from base.html.

- static/uploads: Stores team logo images securely.

5. Screenshots of Key Pages

Below are descriptions of key pages (replace with actual screenshots in your submission):

- Team Dashboard (Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.):

- Displays a grid of team cards for the logged-in user.

- Each card shows the team name, coach, founded year, player count, and logo (or a placeholder).

- Includes buttons for adding players, editing, deleting, and viewing team details.

- Example: A card shows "Team Awesome, Coach: John Doe, Founded: 2000, Players: 5" with Edit/Delete buttons.

- Create Team Page (Изображение выглядит как текст, снимок экрана, линия, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.):

- Shows a form with fields for name, coach, founded year, and logo upload.

- Includes a "Create" button and validation error messages (e.g., for invalid file types).

- Example: A form with "Name: Team Awesome" and a file input for logo upload.

- View Team Page (Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.):

- Displays team details (name, coach, logo) and lists associated players and matches.

- Example: Shows "Team Awesome" with a table of players (e.g., "John Smith") and matches (e.g., "vs Team B").

6. Challenges Faced and Solutions

Several challenges were encountered during development, with solutions implemented as follows:

- Challenge: CSRF Token Missing Error

- Issue: The delete route (/team/delete/<id>) returned a "Bad Request: The CSRF token is missing" error because the form lacked a CSRF token.

- Solution: Added `<input type="hidden" name="csrf\_token" value="{{ csrf\_token() }}">` to the delete form in dashboard.html, ensuring Flask-WTF validated the POST request.

- Challenge: IntegrityError on Team Deletion

- Issue: Deleting a team caused an IntegrityError (NOT NULL constraint failed: player.team\_id) because player.team\_id was set to NULL, violating the NOT NULL constraint.

- Solution: Updated the Player model to include `ondelete='CASCADE'` on the team\_id foreign key, ensuring players are deleted when their team is deleted. The database schema was recreated to apply this change.

- Challenge: Secure File Uploads

- Issue: Ensuring only valid image files (JPG/PNG) were uploaded for team logos to prevent security risks.

- Solution: Implemented the `allowed\_file` function to validate file extensions and used `secure\_filename` to sanitize filenames, preventing path traversal attacks.

- Challenge: Database Schema Updates

- Issue: SQLite does not support modifying foreign keys, complicating the addition of `ondelete='CASCADE'`.

- Solution: Recreated the database with the updated schema during development. In production, Flask-Migrate/Alembic would be used to manage schema changes.

7. Conclusion

The Team Management Web Application successfully meets its objectives, providing a secure, modular, and user-friendly platform for managing teams, players, and matches. The use of Flask blueprints, SQLAlchemy, Flask-WTF, and Bootstrap ensures a maintainable and responsive application. Challenges such as CSRF errors and database constraints were resolved through careful configuration and testing.

- Source Code: Uploaded to GitHub/GitLab (https://github.com/imerone/football\_management.git).

- Hosting: The application is hosted on localhost or a free-tier cloud service (actual local host).