

Smørås Fotball Technical Documentation

Generated: April 07, 2025

Table of Contents

1. Project Overview
2. Project Structure
3. Key Technologies
4. Database Schema
5. Key Features
6. Technical Implementation Details
7. Code Execution Flow

1. Project Overview

The Smørås Fotball application is a comprehensive team management platform designed for youth football teams. It provides tools for managing teams, players, matches, and formations, as well as detailed statistics and reporting. The application is built using the Django web framework and employs a PostgreSQL database for data persistence.

2. Project Structure

The application follows the standard Django project structure with some customizations:

Directory/File	Description
smorasfootball/	Main Django project directory
smorasfootball/smorasfootball/	Core project settings and configuration
smorasfootball/teammanager/	Primary application module
smorasfootball/teammanager/models.py	Database models
smorasfootball/teammanager/views_*.py	View functions organized by feature
smorasfootball/teammanager/urls.py	URL routing configuration
smorasfootball/templates/	HTML templates
smorasfootball/static/	Static assets (CSS, JS, images)

3. Key Technologies

Technology	Version	Purpose
Django	5.1.x	Web framework
PostgreSQL	Latest	Database
Bootstrap	5.x	Frontend UI framework
JavaScript	ES6+	Client-side interactivity
jQuery	Latest	DOM manipulation
Chart.js	Latest	Data visualization

4. Database Schema

The application uses the following main models:

Model	Description	Key Relationships
Team	Football team	Players, Matches
Player	Individual player	Team, MatchAppearance
Match	Football match	Teams, MatchAppearances
MatchAppearance	Player's appearance in a match	Player, Match
Formation	Team formation template	Positions
Position	Player position in a formation	Formation
User	Application user	UserProfile
UserProfile	Extended user information	User

Model Relationships Example:

A Team has many Players (one-to-many relationship). A Match involves two Teams (many-to-many through a foreign key). A Player appears in Matches through MatchAppearance (many-to-many with additional data).

5. Key Features

Feature	Description
Team Management	Create and edit teams, assign players to teams
Player Management	Track player information, skills, and statistics
Match Management	Schedule matches, record results, and track statistics
Formation Builder	Create and edit team formations with different team sizes (5er, 7er, 9er, 11er)
Match Session	Real-time match tracking with player substitutions and timer
Statistics Dashboard	Visual display of player and team statistics
Player Matrix	Visualization showing which players have played together
Multi-language Support	Support for English and Norwegian languages
User Authentication	Role-based access control (Admin, Coach, Player)

6. Technical Implementation Details

Match Timer Implementation

The match timer is implemented using JavaScript for client-side counting and AJAX for synchronization with the server. The timer state is persisted in the database to allow resuming matches after page reloads.

Client-side Timer Code Example:

```
// Match Timer JavaScript
let matchTimer; let elapsedSeconds = 0; let
elapsedSecondsPreviousPeriods = 0; let isRunning = false;
function startTimer() { isRunning = true; matchTimer =
setInterval(function() { elapsedSeconds++; updateTimerDisplay(); // Sync
with server every 5 seconds if (elapsedSeconds % 5 === 0) { syncTimerWithServer(); } }, 1000);
} function stopTimer() { isRunning = false; clearInterval(matchTimer); syncTimerWithServer();
}
```

Authentication System

The application uses Django's built-in authentication system with customizations for role-based access control. Users can have Admin, Coach, or Player roles, each with different permissions.

7. Code Execution Flow

Match Session Flow

1. User creates a new match session or opens an existing one
2. Server loads match data and player assignments
3. Client initializes the match view with pitch visualization
4. User starts the match timer
5. Client-side JavaScript updates the timer and player statuses
6. Periodic AJAX requests sync data with the server
7. User makes substitutions by dragging players between active and bench areas
8. Server records player participation times
9. User ends the match, and final statistics are saved