# Match Analytics Dashboard

This project is a full-stack application for visualizing and analyzing data from a soccer match. The backend is built with Spring Boot and the frontend with Angular, providing a dynamic dashboard to display match details, player statistics, and event highlights.

### How to Run the Project

This application consists of two separate components: a backend API and an Angular frontend. Both need to be running for the application to function correctly.

#### 1. Start the Backend

The backend is a Spring Boot application that requires a PostgreSQL database.

* Prerequisites
* Java Development Kit (JDK) 17 or higher
* Apache Maven
* A running PostgreSQL database instance

Database Setup (Docker Optional): If you don't have a local PostgreSQL instance, you can use Docker to get one up and running quickly. Open your terminal and run the following command:

docker run --name match-db -e POSTGRES\_PASSWORD=yosra -e POSTGRES\_USER=yosra -p 5432:5432 -d postgres:latest

This command starts a PostgreSQL container and maps the default port 5432. The Spring Boot application is configured to connect to this instance by default using the credentials in application.properties.

Run the Application: Navigate to the backend project's root directory and use the following Maven command to start the application:

mvn spring-boot:run

The backend will automatically create the necessary database tables and seed them with data from match.json. The API will be available at http://localhost:8080.

#### 2. Start the Frontend

The frontend is an Angular application that will consume data from the backend.

Prerequisites:

1. Node.js and npm (or yarn)
2. Angular CLI

Run the Application: In a separate terminal, navigate to the frontend project's root directory and run the following command:

ng serve

The Angular application will be served at http://localhost:4200 and will automatically proxy API calls to the running Spring Boot backend.

### Explanation of formRating Formula

The provided code does not contain a function named formRating. Instead, the application's core analytics are handled by a playerStats computed signal within the MatchDashboardComponent.

This signal processes all events in a match to calculate the total goals and assists for each player. It does this by:

1. Initializing a map with all players and setting their goals and assists to 0.
2. Iterating through all match events.
3. Incrementing a player's goals count if the event type is 'GOAL'.
4. Checking for an assist in the event's metadata (event.meta.assistId) and, if found, incrementing the assists count for the corresponding player.

This is a simple, rule-based approach to player analytics. A more complex "form rating" could be implemented by assigning different weights to various event types (e.g., shots, tackles, passes) to create a more comprehensive performance score.