Agentic AI System for Football Analytics

Ahmed Maher
Dept. of Computer Science and Physics
Wilfrid Laurier University
Waterloo, Canada
mahe7984@mylaurier.ca

Emad A.Mohammed
Dept. of Computer Science and Physics
Wilfrid Laurier University
Waterloo, Canada
emohammed@wlu.ca

***Abstract*—In football, data analytics is a core element in understanding the sport to a high degree. As the field has evolved, more advanced statistics have become available, allowing for deeper interpretation of the sport. Unfortunately, accessibility and ease of use in finding and interpreting such data are limited to the general public. This paper introduces the Football Management AI, a system designed to democratize football analytics. The Football Management AI leverages an N8N workflow to orchestrate the system from end to end, Grok's Llama 3.3 Large Language Model (LLM) to interpret queries and results, and a Python backend that utilizes the SoccerData framework to extract team and player statistics across the top 5 European leagues. The system is assessed over five categories of football metric queries: league tables, team season metrics, team match metrics, player season metrics and player match metrics. The system is evaluated using over 50 queries to assess accuracy, response time, and complexity. It achieved an overall success rate of 80.77% with an average response time of 10.91 seconds. These results indicate that the Football Management AI can effectively retrieve and contextualize football statistics, offering an alternative to other analytic platforms. This work highlights the potential of LLMs for football analytics and opportunities for further applications, such as visualizations in future iterations.**

***Index Terms*—Football analytics, Large language models, Workflow automation, SoccerData, AI agents.**

I. Introduction

In football (soccer), data analysis has always been a key component in understanding the sport. Whether it is a manager using data to analyze opponents before a prominent fixture, a scout finding a player for a specific system, or simply a fan of the sport trying to learn more. With the rapid advancement of artificial intelligence and machine learning, football has witnessed a surge in the collection of event-based data and in-depth statistics. Data from events can provide insights into expected goals, possession metrics, pass accuracy, and individual player performance. However, the ability to interpret such data is often limited to analysts with closed-source systems or programming expertise. While large language models have been created to address this disparity [2]- [3], providing users with interpretations of match statistics based on key match events, the issues of accessibility and ease of use remain. For example, to find the in-depth statistics for a specific player in a match, you have to find a site that displays football data, then find the particular game, which generally only shows the in-depth information.

This paper addresses this issue by proposing an AI agent-based football analysis tool that expedites the identification of key data and provides relevant analysis based on user queries. The tool uses the N8N automation platform [10] as the workflow, integrating Grok's LLama [11], which addresses this issue by proposing an AI agent-based football analysis tool that expedites the identification of key data and provides the SoccerData framework [1]. For example, if a user prompts "Show me Eberiche Ezes stats for Arsenal against Manchester City", the application will use SoccerData [1] to retrieve the information from FBref [4].

The overall goal of this paper is to democratize football analysis through automation and advanced Python frameworks, creating an accessible, easy-to-use, and in-depth tool for any user interested in the sport. The remainder of this paper is organized as follows: Section 2 presents a literature review of related work; Section 3 outlines the methodology and approach; Section 4 presents the results and discusses the next steps; and Section 5 concludes the paper.

II. Related Works

The field of football analytics has been rapidly evolving in recent years, with a significant shift from descriptive statistics — such as possession, shots on target, and passes completed — to predictive statistics — such as expected goals (xG), expected assists (xAG), and heat maps. This transition stems from advances in machine learning and large-scale data pipelines, which, when coupled with artificial intelligence, can interpret and contextualize data to deliver advanced football analytics.

A. Descriptive Statistic Understanding:

Early football analysis relies heavily on understanding fundamental descriptive statistics to predict match outcomes. A study by Carlos Lago-Peñas and colleagues [5] analyzed descriptive statistics from 380 matches during the 2008-2009 Spanish league season to determine which statistics discriminate between wins, draws, and losses. The results demonstrated that total shots, shots on goal, and ball possession are among the most effective statistics to discriminate between results. For example, winning teams averaged 14.4 shots and 6.6 shots on goal, while losing teams averaged 11.9 shots and 4.2 shots on goal. Thus, higher shots and shots on goal can be interpreted as indicating that the team with more shots and shots on goal has a greater chance of winning the match, and this can be applied to ball possession and other statistics accordingly. These findings signal that descriptive data could predict match outcomes. However, descriptive metrics lack contextual depth and tactical understanding.

B. Machine learning for Predictive Statistics:

As descriptive statistics do not always paint a complete picture of a match, predictive metrics such as xG, xAG, and possession graphs aim to provide contextual depth and understanding. These metrics build on descriptive stats, such as shots and passes, to calculate the probability that a shot or pass will lead to a goal. For example, rather than looking at the number of shots to predict the likely outcome, a machine learning model can assess shot quality to determine which team was more likely to outscore its opponent. The Opta xG model [6] is widely used, drawing on nearly one million shots across 40 competitions from 2018 to 2022. The model uses several variables, such as the distance to the goal, angle to the goal, shot type, pattern of play and goalkeeper positioning, to accurately predict the probability of a goal. The Opta xAG model [7] follows a similar procedure to calculate expected assists, including the type of pass, the location where the pass is made or received, and the distance of the pass. Regarding possession interpretation, the OpenSTARLab [2] is an open-source model that uses event data sets such as Wyscout data [8] to train itself to provide accurate schemas and visualizations on a match, such as possession heat maps and Holistic Possession Utilization Score (HPUS), which showcase the effectiveness of both teams' possession in the game. While these predictive models have dramatically advanced football analytics, an issue arises with accessibility: many models are accessible only to specific organizations, or they still need development to be used in a commercial application for the average user.

C. Artificial Intelligence for Football Statistics:

The next step of football analytics looks to make both descriptive and predictive statistics more interpretable and conversational for the average user. The SoccerRAG framework [3] is a retrieval-augmented generation (RAG) system that uses a large football dataset alongside an LLM to support natural language queries and provide contextual interpretations of statistics in simple language to users, offering an accessible way to obtain in-depth analysis. This allows users to ask open-ended questions, such as "Which team scored the most goals in the 2015-2016 season?" and receive data-backed insights. While SoccerRAG [3] shows the potential for accessible, interactive football analysis, it has many limitations, including a limited dataset that prevents access to recent and advanced statistics, as well as a lack of visuals, such as graphs or heat maps. The Football Management AI addresses gaps in related work by providing an AI-driven workflow built in N8N [10], where retrieval, reasoning, and visualization are performed through a chat-based interface, streamlining soccer analytics to give interpretations of advanced statistics for coaches, researchers, and fans to explore.

III. METHODOLOGY

Figure 1 provides an overview of the Football Management AI, highlighting its components and workflow. The framework consists of four key components that are integrated and orchestrated using the automation platform [10]: (1) User Interface: A web-based chat interface that allows users to ask natural language queries and returns responses in a conversational format. (2) A large language model (LLM) to interpret user queries and determine whether tool invocation is required. (3) Natural language detection that takes the query from the LLM and calls the appropriate function based on intent. (4) The SoccerData [1] API to retrieve the requested metric.

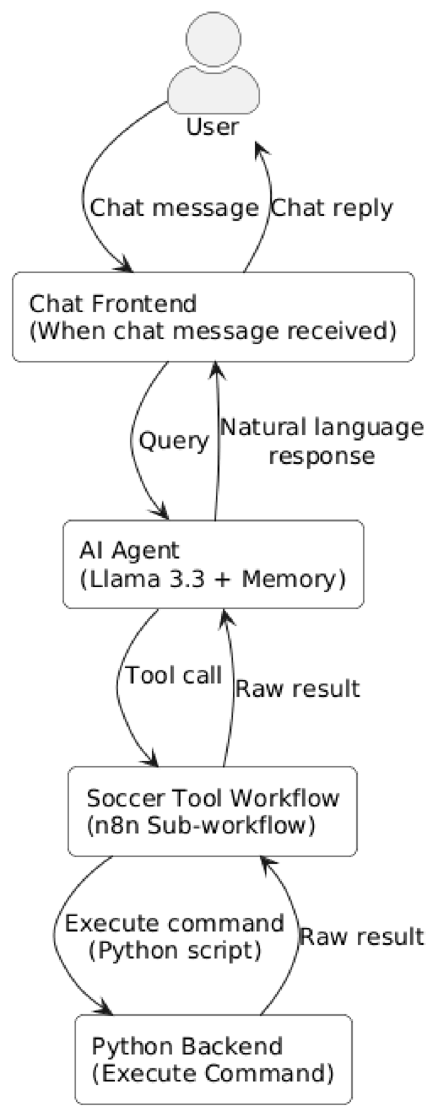

Fig. 1. Overview of Football Management AI workflow
Available at https://github.com/mramlock under research_final_cp493

A. User Interface

The system provides a web-based chat interface, implemented using React [12]. This interface is the point of access for users to the system. From this interface, the user interacts with the LLM by asking football questions. Once the user asks a question, the interface forwards the user's message to the automation platforms' webhook [10] via a built-in chat node. Once the system has an answer, it is returned to the frontend for the user to see. The frontend acts as a relay between the user and the workflow, sending and receiving messages.

B. Large Language Model

Grok's LLama 3.3 model [11] is used as the central reasoning component inside the automation platform's [10] AI Agent node. Its function is to facilitate, connect, and interpret user messages from the front-end for answering in the backend. The LLM first receives the raw user message. It determines whether the request requires invoking the Python tool or can be responded to directly with a general-knowledge answer or an error message. If the LLM decides to invoke the Python tool, it will forward the raw user query string (e.g., ”Who is first in the Premier League”) directly to the tool. After the Python tool returns the results, the LLM converts the raw result into a natural query for the user to understand. For example, the result: { “intent”: “rank”, “league”: “ENG-Premier League”, “season”: “2025/2026”, ”position”: 1, ”result”: {”position”: 1, ”team”: ”Arsenal”, } } is converted into natural language such as: “Arsenal is first place in the Premier League this season.”

C. Natural Language Detection

The third component is a section of the Python backend (nl_query_to_result() and helper functions) that detects natural language queries, identifies the query intent, and breaks the question into pieces the backend needs to retrieve the correct statistics. This component performs:

- Intent classification (e.g., table statistics, team season or match statistics, player season or match statistics, team or player comparisons).
- League and season detection using alias mapping and pattern matching.
- Team and player extraction using FBref [4] player lists, team name dictionaries, and token-based heuristics.
- Metric identification to distinguish what information to get from the intent classification. For example, retrieving the position of a tram from the table lookup.

The natural language detection converts the natural-language query into a fully structured backend request, for example:
{ ”intent”: ”rank”, ”league”: ”ENG-Premier League”, ”season”: ”2025/2026”, ”position”: 1 }

D. SoccerData API

The final component is the Python backend built using the SoccerData [1] library. This module executes the actual data extraction requested by the NL detection component. It includes five major categories:

1) The Fotmob [9] table metric uses the read_league_table() function, which returns a league table with team standings, points, match records (wins, draws, losses) and goal statistics (goals for, goals against, goal difference). The function requires two parameters: league name and season year. If no league or year is provided, the function defaults to the current Premier League season. From this point, statistics regarding a team or standings can be retrieved.
2) The FBref [4] team season stats metric uses the read_team_season_stats() function, which returns season-long statistics such as goals, assists, expected goals (xG), expected assists (xAG), average possession, etc. The function requires two parameters, league name and season year and collects data for all teams in a single call. From this point, the data can be filtered for specific teams or team comparisons.
3) The FBref [4] team match stats metric uses the read_team_match_stats() function, which returns statistics regarding all matches played for a specific team in a season, such as result (win, loss or draw), venue (home, away), score, xG, possession, etc. The function requires three parameters: the league name, the season year, and the team to find a match for. It collects data for all the games that the team has played that season in a single call. From this point, the data can be filtered to find the specific match for that team and provide the statistics.
4) The FBref [4] player season stats metric uses the read_player_season_stats() function, which returns season-long statistics such as game played, starts, goals, assists, xG, xAG, yellow cards, etc. The function requires two parameters, league name and season year and collects data for all players in the league in a single call. From this point on, the data can be filtered for specific players or player comparisons.
5) The FBref [4] player match stats metric utilizes three functions. The read_player_season_stats() (mentioned above) to confirm that the team and player exist. Then the read_schedule() function returns all fixtures in the season. The function requires two parameters, the league name and the season year. The player statistics the user is looking for can be found via the match ID. Once the match ID is retrieved, the fbref.read_player_match_stats() function is called. This function requires three parameters: the league name, the season year and the match ID and returns statistics for every player in the match, such as minutes played, goals, assists, xG, xAG, yellow cards, interceptions, etc. From this point on, the data can be filtered by the specific player to view their statistics.

   Once the query has been answered, the result is returned in a raw form, such as,
   { “intent”: “rank”, “league”: “ENG-Premier League”, “season”: “2025/2026”, ”position”: 1, ”result”: {”position”: 1, ”team”: ”Arsenal”, } } for the LLM to interpret.

E. Current Limitations

Please note that the Football Management AI currently supports queries across the top 5 leagues: Premier League, La Liga, Serie A, Bundesliga, and Ligue 1. In addition, note that previous season statistics may not be updated to include the last game of the season, meaning results are correct up to the penultimate match. For example, the Premier League 2024/2025 season contains data for 37/38 matches. Finally, note that teams or players cannot be compared across leagues. For example, Arsenal's stats cannot be compared to Barcelona's, nor can Bukayo Saka be compared to Kylian Mbappe.

IV. Results

In this section, the capabilities of the Football Management AI are tested to assess its responsiveness, accuracy, and complexity. The system is tested across five categories using the SoccerData [1] framework: Fotmob [9] table metrics, FBref [4] team-season metrics, FBref team-match metrics, FBref player-season metrics, and FBref player-match metrics. In total, fifty-two questions were tested to provide an in-depth analysis of performance. sta

A. Table Metrics (Category 1)

From testing, the system maintains an 81.8% success rate (9/11 queries) and an average response time of 4.37 seconds. This category of questions demonstrated low complexity, requiring only one API call and returning a compact dataset of 20 teams x 8 metrics.

Potential questions that utilize table metrics include:

- Show me the Premier League 2024/2025 table.
- What is Arsenal's record this season?
- Arsenal's xG this season

An example conversation is shown below:

User Query: Show me the premier league 2024/2025 table

Response Time = 3.51 seconds.

Output: The Premier League 2024/2025 table is as follows:

1) Liverpool - 84 points
2) Arsenal - 74 points
3) Manchester City - 71 points
4) Chelsea - 69 points
5) Newcastle United - 66 points
6) Aston Villa - 66 points
7) Nottingham Forest - 65 points
8) Brighton and Hove Albion - 61 points
9) AFC Bournemouth - 56 points
10) Brentford - 56 points
11) Fulham - 54 points
12) Crystal Palace - 53 points
13) Everton - 48 points
14) West Ham United - 43 points
15) Manchester United - 42 points
16) Wolverhampton Wanderers - 42 points
17) Tottenham Hotspur - 38 points
18) Leicester City - 25 points
19) Ipswich Town - 22 points
20) Southampton - 12 points

While the system successfully answered 9/11 queries, the answers varied in detail. For example, "Who is top of the Bundesliga this season?" will correctly return Bayern Munich and the number of points, but it will not mention the number of matches played or their record. While this answer is correct, it is not detailed. This is a result of the LLM. Upon testing the same query, the LLM will provide less or more detail each time the question is asked. Meaning, in one run, it may display Bayern Munich with their points and record. At the same time, in another, it may only mention that Bayern Munich is in first place, demonstrating an inconsistency in the LLM's responses. Looking at failures, queries often fail or give incorrect answers due to ambiguous wording. For example, "Who was 10th in the Serie A last season?" returned the 10th place team for the 2023/2024 season rather than the 2024/2025 season. This failure is due to the LLM misinterpreting the query, believing the last season to be 2024, and incorrectly selecting the 2023/2024 season. Another error occurred in asking, "Who won the Premier League?" This is because of the keyword "won"; in the system, when the keyword "won" is used, it looks for a match result rather than a table ranking. To avoid this error, use the keyword "top" instead.

B. Team Season Metrics (Category 2)

From testing, the system accomplished a 90% success rate (9/10 queries) with an average response time of 5.68 seconds. This category of questions demonstrated low to medium complexity, requiring only one API call and returning a dataset of 20 teams x 16 metrics.

Potential questions that utilize team season metrics include:

- Manchester City stats this season

- Barcelona stats 2023/2024 La Liga season.
- Compare Arsenal and Manchester City stats this season.

An example conversation is shown below:

User Query: Manchester city stats this season

Response Time = 4.31 seconds.

Output: Manchester City's team stats this season are as follows:

- Goals: 38
- Assists: 32
- Goal contributions: 70
- Possession: 58.3
- xG: 30.4
- xAG: 24.7
- npxG: 30.4
- per90 goals: 2.01
- per90 assists: 1.7
- per90 goal contributions: 3.46
- per90 xG: 2.03
- per90 xAG: 1.67
- per90 xG xAG: 3.7
- per90 npxG: 2.03
- per90 npxG xAG: 3.7

Results from the nine successful queries were adequate, providing sufficient detail for each question. For example, "Arsenal xG 2024/2025 season" will correctly return Arsenal's xG for the season with no additional statistics. The failed query is due to a similar issue found in category one. When asked, "Real Madrid goals last season", the LLM incorrectly interpreted last season as 2023/2024 rather than 2024/2025. In addition, despite accomplishing a higher success rate than category 1 (90% vs 81.8%), category 2 responses increased in average time by 1.31 seconds (5.68 vs 4.37 seconds). This is due to the greater complexity of the larger dataset and the additional use cases compared to category 1.

C. *Team Match Metrics (Category 3)*

From testing, the system accomplished an 80% success rate (8/10 queries) with an average response time of 11.33 seconds. This category of questions demonstrated medium complexity, requiring only one API call; however, it returns a large dataset of 38 games x 19 metrics.
Potential questions that utilize team match metrics include:

- Fulham vs Arsenal at home result.
- Arsenal vs Burnley away xG
- Bayern Munich vs Borussia Dortmund score at home this season

An example conversation is shown below:

User Query: Fulham vs Arsenal at home result

Response Time = 10.44 seconds.

Output:
Fulham lost at home to Arsenal 0-1. They had 0.4 xG, 37% possession, and used a 4-2-3-1 formation, while Arsenal used a 4-3-3 formation and had 1.8 xG.

Results from the eight successful queries were generally accurate; however, the details varied depending on the query. For example, if stats or results are requested for a match, the system will correctly provide all game information, including the result, score, xG, etc. (see example above). For queries about individual metrics, such as the score or xG, the LLM occasionally returned the requested statistic along with all other statistics, rather than just the requested value. This is a similar issue to category one, where the LLM varies in detail in its responses to questions. Regarding failed queries, a failure may occur when a team's full name is not provided; for example, "Dortmund" is not recognized as a team, whereas "Borussia Dortmund" results in the LLM not finding a match. In addition, the query "Paris Saint-Germain vs Marseille at home" returned Paris FC vs Marseille instead. This is because the Paris Saint-Germain (PSG) fixture had not been played; rather than returning nothing and noting that the game had not occurred, the system instead went to the Paris FC fixture. Furthermore, category 3 queries have the lowest success rate (80%) and the highest average response time (11.33 seconds), compared to team-based categories (categories 1-3). This is because the dataset is extremely large, requiring more time to filter through the data to find specific match fixtures.

D. *Player Season Metrics (Category 4)*

From testing, the system accomplished a 81.8% success rate (9/11 queries) with an average response time of 10.528 seconds. This category of questions demonstrated medium-high complexity, requiring only one API call; however, it returns an extremely large dataset of 500+ players * 37 metrics.

- Bukayo Saka's stats this season.
- Bukayo Saka overperforming this season?
- Compare Erling Haaland and Mohammed Salah this season.

An example conversation is shown below:

TABLE I
PERFORMANCE ANALYSIS ACROSS QUERY CATEGORIES

Query Type	**Avg Time (sec)**	**Min Time (sec)**	**Max Time (sec)**	**Success Rate (%)**	**Complexity**
FotMob Table (1)	4.37	2.69	7.71	81.81	Low
FBref Team Season (2)	5.68	2.71	11.71	90.00	Low-Medium
FBref Team Match (3)	11.33	7.87	20.90	80.00	Medium
FBref Player Season (4)	10.53	4.99	26.371	81.81	Medium-High
FBref Player Match (5)	22.66	11.02	35.16	70	High
Overall	**10.91**	**–**	**–**	**80.77**	**–**

User Query: Compare Erling Haaland and Mohammed Salah this season.

Response Time = 4.99 seconds.

Output: Erling Haaland leads Mohammed Salah in terms of goals scored this season, with 14 goals to Salah's 4. Haaland also has more assists and expected goals, but Salah has more expected assists. The player comparisons are as follows:

- Goals: Erling Haaland (14) better than Mohammed Salah (4)
- Assists: Mohammed Salah (2) better than Erling Haaland (1)
- Goals + Assists: Erling Haaland (15) better than Mohammed Salah (6)
- Expected Goals (xG): Erling Haaland (11.1) better than Mohammed Salah (3.6)
- Expected Assists (xAG): Mohammed Salah (1.8) better than Erling Haaland (1.2)

Overall, Erling Haaland has been performing better this season, both in goals scored and expected goals.

Of the nine successful queries, the responses demonstrated contextually appropriate detail levels. For example, "Bukayo Saka's goals this season" will correctly return his goals for that season or occasionally his goals and expected goals. As expected, goals are a related stat and fit the context of the question. Regarding the failed queries, one failed due to a frequent LLM inconsistency. Once again, the query asked for last season's stats, but the LLM misinterpreted it as the 2023/2024 season rather than the 2024/2025 season, resulting in the wrong season's stats. The other failure is due to ambiguous player names. For example, "What are Gabriel's stats this season in the Premier League?" As there are numerous Gabriels in the Premier League, such as Gabriel Magalhaes, Gabriel Martinelli, Gabriel Jesus, etc. The system may retrieve incorrect players' statistics; therefore, it is preferred to mention a player's full name. Although more complex, category 4 queries achieve a higher success rate (81.8% vs 80%) and a lower average response time (10.53 vs 11.33 seconds) than category 3, which is less complex (medium-high vs medium complexity). However, category 4 has a higher average response time compared to categories 1 and 2, as expected.

E. Player Match Metrics (Category 5)

From testing, the system accomplished a 70% success rate (7/10 queries) with an average response time of 22.66 seconds. This category of questions demonstrated high complexity, requiring only three API calls. One to find the player and team, second to find the match ID, and third to gather the match stats. Due to three API calls, this category yields the largest dataset: (500+ players * 37 metrics) + (340-380 matches * 18 metrics) + (22-32 players *33 metrics).

Potential questions that utilize team match metrics include:

- Eberechi Eze stats in Arsenal vs Manchester City at home.
- Kylian Mbappe stats in the Real Madrid vs Barcelona game at home.
- Declan Rice's goals in the Arsenal vs Burnley away game this season.

An example conversation is shown below:

User Query: Eberechi Eze stats in Arsenal vs Manchester City at home.

Response Time = 16.06 seconds.

Output: Eberechi Eze played 45 minutes in the Arsenal vs Manchester City match at home, with 1 assist, 1 shot, and 1 shot on target. He also had 30 touches, 1 tackle, and 5 progressive passes. His xG was 0.1, and xAG was 0.2.

From the seven successful queries, the responses demonstrated contextually appropriate detail levels. For example, "Bukayo Saka's stats in Arsenal vs Burnley away this season " will correctly return his key stats in that match. However, several issues were discovered in the failed queries. Similar to category 4, not using a player's full name may result in incorrect player stats or no match stats if the player cannot be found. Additionally, failures may occur in ambiguous fixtures. For example, "Eberechi Eze stats vs Manchester City at home.", will not provide an answer as the first team is not mentioned, where both are required. To work, the query

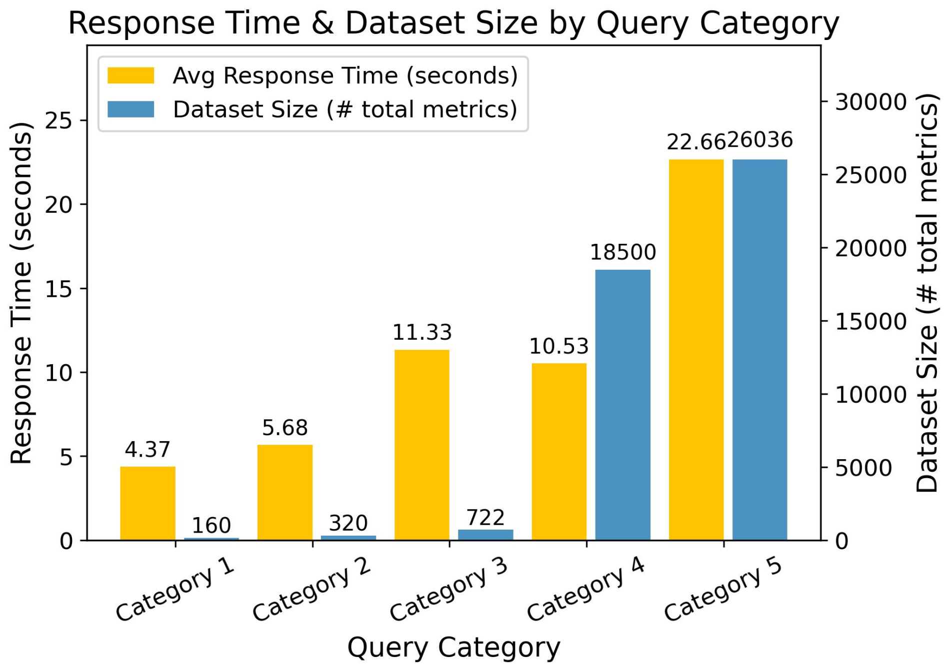

Fig. 2. Response Time & Dataset Size by Query Category

must be "Eberechi Eze stats in Arsenal vs Manchester City at home". In addition, a failure may occur when a team's full name is not mentioned. The query "William Saliba stats in Arsenal vs Leeds at home this season" failed because the system could not find the team Leeds, as it is recognized as Leeds United in the dataset. While the system can handle alternate names for some larger clubs, it is generally preferred to use the team's full name rather than a nickname. As this category is the most complex, due to its dataset size and multiple API calls, it performs as expected, with the lowest success rate of 70% and the highest average response time of 22.66 seconds, compared to the second-highest of 11.33 seconds in category 3.

F. Overall Performance

In Table 1, the system's performance across all five categories and the overall system is shown, including the maximum, minimum, and average response times, as well as the success rate and query complexity. Overall, the system achieves an average response time of 10.91 seconds and succeeds on 80.77% of queries. The system's timing and accuracy were, for the most part, as expected, demonstrating that the more complex the query, the longer the response time and the lower the success rate, as shown in Figure 2. While the system more often than not provides contextually correct answers, a minor issue emerged: the LLM occasionally provides too little or too much information for some queries. Although this issue does not affect The validity of responses and improvements to LLM responses would enhance the user experience. Analysis of the system failures reveals a consistent pattern across all query categories:

- Ambiguous teams or names: occasionally, if the entire team name or the player's full name is not provided, the LLM may return no results or the wrong team or player.
- Keyword placement leading to the incorrect system intent: due to the way the system determines the purpose of a query, whether it be to find a fixture result, a player's stats, etc, specific keyword placements may lead to incorrect queries. For example, "Kylian Mbappe stats for the Real Madrid vs Barcelona game at home" will incorrectly return the fixture stats rather than the player stats because of the "for" keyword. However, Kylian Mbappe's stats in the Real Madrid vs Barcelona game at home" correctly return the players' stats because of the "in" keyword.

G. Next Steps

The current prototype of the Football Management AI demonstrates the potential of the application in the future. Going forward, several features, such as heat map visualizations, charts, shot maps, and more, would be beneficial. In addition, incorporating more advanced stat interpretations, such as club strengths and weaknesses, and player profile matching to see what players a club would benefit from signing. The Football Management AI also faces several limitations that need to be addressed. First, the limited number of leagues and the lack of cross-league comparisons hinder the user's experience. In addition, the LLM faces the weakness of ambiguity and occasionally poor interpretation of user questions. Exploring other LLMs may provide system improvements in that regard, boosting system performance. Finally, ensuring ethical data retrieval is a crucial step to be maintained in future iterations.

Conclusion

Throughout this paper, the Football Management AI is introduced to streamline football analysis utilizing an LLM alongside the SoccerData API [1] to retrieve and interpret football statistics for clubs and players. Through system evaluation, the Football Management AI can analyze and answer a variety of football questions, including team and player stats across a season or in specific matches. The Football Management AI provides all those interested in Football with a convenient way to access and interpret advanced statistics. This early version of the Football Management AI demonstrates a bright future in which AI can be a catalyst for deep understanding of the sport.

References

[1] SoccerData, "SoccerData: A Python interface for football data," Available: https://soccerdata.readthedocs.io/.

[2] C. Yeung et al., "OpenSTAR: Open Approach for Spatio-Temporal Agent Data Analysis in Soccer arXiv:2502.02785, 2025.

[3] A. Theo Strand et al., "SoccerRAG: Multimodal Soccer Information Retrieval via Natural Queries for soccer analytics," arXiv:2406.01273, 2024.

[4] FBref, "FBref Football Statistics," Available: https://fbref.com/.

[5] C. Lago-Peñas et al., "Game-related statistics that discriminated winning, drawing and losing teams from the Spanish soccer league," Journal of Sports Science and Medicine, vol. 9, pp. 288–293, 2010.

[6] J. Whitmore, "What is expected goals (xG)?," Opta Analyst, Aug. 8, 2023. Available: https://theanalyst.com/articles/what-is-expected-goals-xg/

[7] J. Whitmore, "What are expected assists (xA)?," Opta Analyst, Mar. 24, 2021 Available: https://theanalyst.com/articles/what-are-expected-assists-xa/.

[8] L. Pappalardo et al., "A public data set of spatio-temporal match events in soccer competitions," Scientific Data, vol. 6, no. 1, pp. 1–15, 2019.

[9] FotMob, "Football scores and statistics," Available: https://www.fotmob.com/.

[10] N8N, "n8n Automation Platform Documentation," Available: https://docs.n8n.io/.

[11] Meta, "Llama 3.3 Model Card," Available: https://github.com/meta-llama/llama-models/.

[12] React, "React: A JavaScript library for building user interfaces," Available: https://react.dev/.

V. Appendix

Source code for the workflow and backend is available at: https://github.com/mramlock under research_final_cp493

Source code for front end is available at: https://github.com/mramlock/soccer-ai-dashboard