Capstone Project

START-UP NAME: BALLIQ

Group 4:

Rodrigo Maia 20221934,

João Ferreira 20221912,

Miguel Mendes 20221904,

Tymofii Kuzmenko 20221690,

Artem Khomytskyi 20221686

Data Sources and Database Design

1. Data Sources

Users Table (Synthetic): Notebook used create_database.ipynb

Players Fantasy (Synthetic and Real): Fantasy Premier League Dataset 2024-2025

Players Stats Table (Synthetic): Notebook used create_database.ipynb

User Team Table (Synthetic and Real):

- Player_id (Real) - Fantasy Premier League Dataset 2024-2025
- User_id, starting_eleven, on_team (Synthetic) – Notebook used create_database.ipynb

Fixtures Table (Real) - EPL Fixtures List 2024-2025

PDF data (Real): Converted to pdf from https://fantasy.premierleague.com/help/rules and also PDFs created for deliveries 1 and 2.

2. Database Design

The database schema for the BallIQ chatbot/application consists of four main tables:

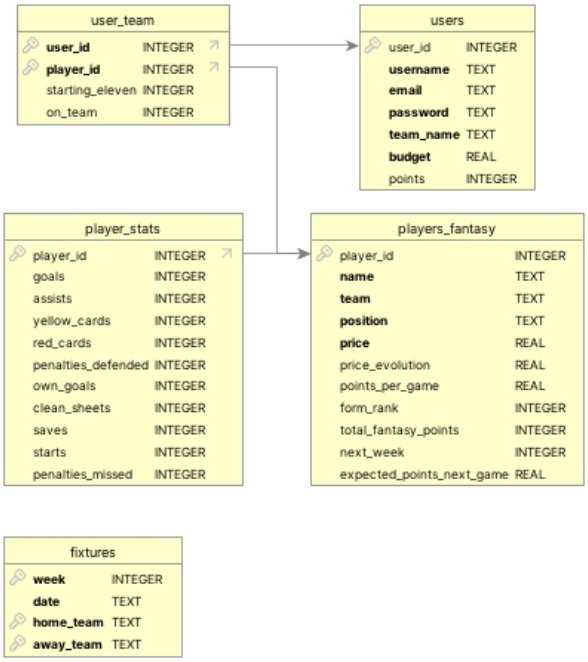
Users: Keeping user data is crucial to controlling the budget, transfers within the team and being able to contact the user, whether for marketing purposes or any other type of problem.

Players Fantasy: This table is central to the functioning of the application. All player data directly related to fantasy, from price tracking to the position and form rank, including other information. It is, therefore, essential to respond to user requests.

Players Stats: Maintaining player stats is fundamental for better monitoring of each player's points. It allows the system to answer a wider variety of questions, providing more specific information to the user.

User Team: To record the players belonging to each team and whether they are in the starting eleven. Together with the two player tables, it allows us to get a deeper understanding of how the players are performing.

Fixtures: Contains information on all 38 Premier League fixtures. It allows clearer answers and more information to the user.



Part 2

User Intentions

- Recommend Players
- Get Company Information
- Optimize Lineup
- Check Upcoming Features
- Stats and Fantasy Points Information
- Perform Transfers
- View Player Stats and Price Evolution
- Adding or Removing players in starting eleven
- View or Update Points

Chatbot Architecture Guidelines (for each intention)

1. Recommend Players

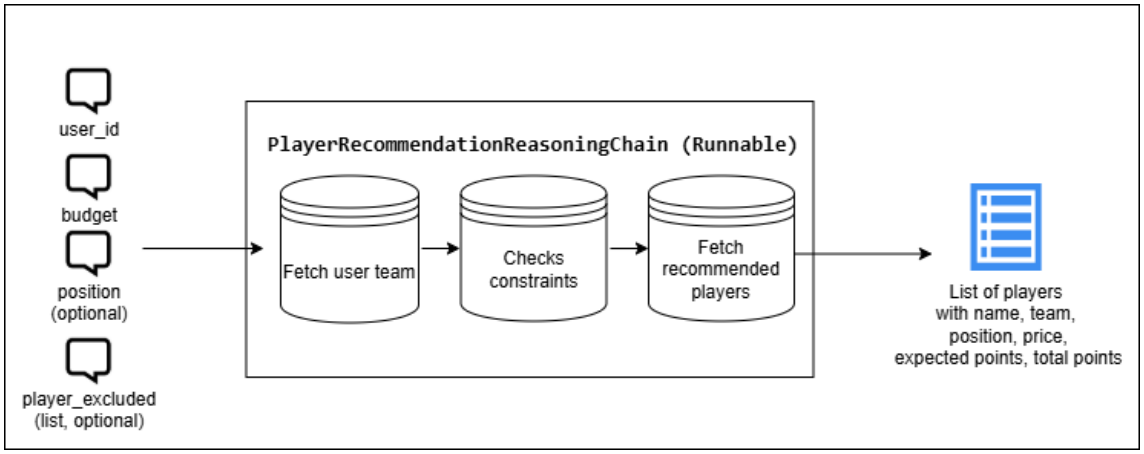
1.1. Story

“As a fantasy football player, I want to be able to have players being recommended to me based on how well they are expected to do, so that I can make informed decisions when picking my team.”

1.2 Architecture Diagrams

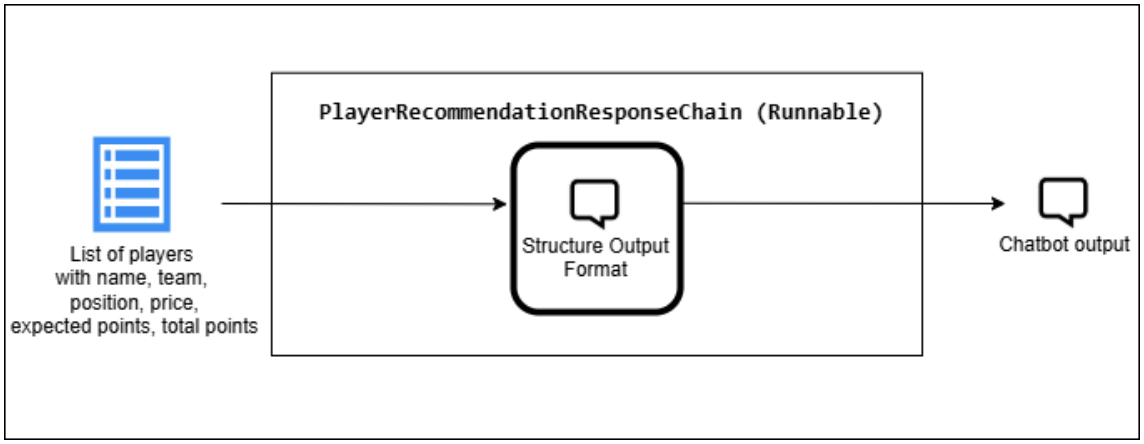
Goal: Identify the top 10 players given a budget restraint, for some position (optional) and without considering some players (optional).

Implementation: We will implement a chain that can retrieve players filtered on the price and that are not in the excluded players list, from the database and order the filtered players of the chosen position based on their expected points in the next match. At the end the chain returns a list with the top 10 players to be recommended.



Goal: Structure the top 10 list in a final output.

Implementation: We will implement a chain that receives the list as input and uses it to generate a final user output using a prompt template.



2. Get Company Information

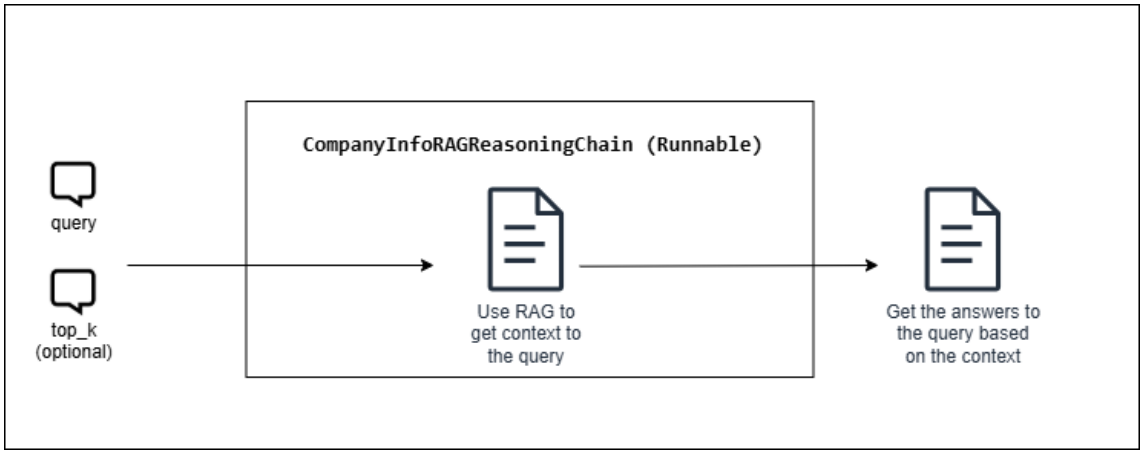
2.1. Story

“As a BallIQ user, I want to be able to access information about the company so I can learn more about them.”

2.2 Architecture Diagrams

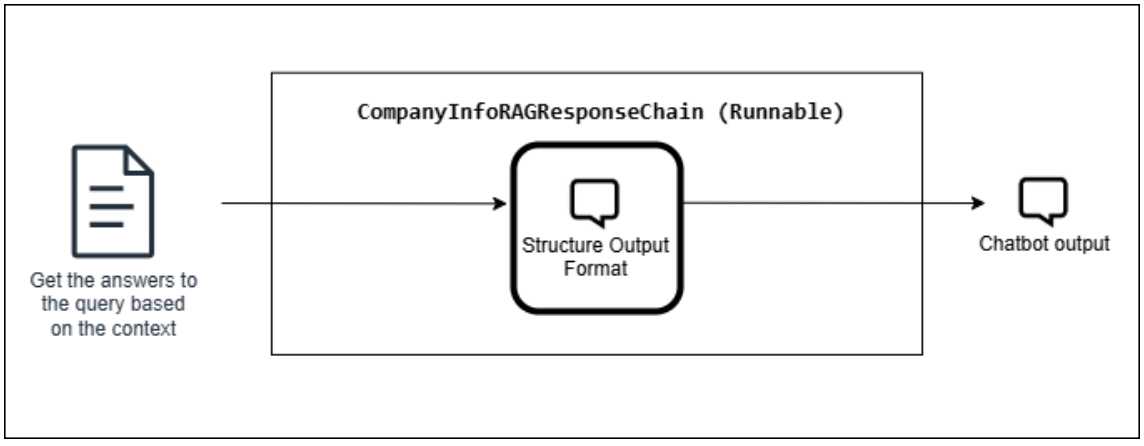
Goal: Allow users to get information about our company by using a chain to access pdfs with information about the company.

Implementation: We will implement a chain that gets the user’s query and the number of most important documents it wants to retrieve (optional), to retrieve information from PDFs using RAG. At the end the chain returns the answers to the user’s query in a string format.



Goal: Structure the answers that are in a string format in a final output.

Implementation: We will implement a chain that receives the answers in a string as input and uses it to generate a final user output using a prompt template, summarizing and giving the user the most important information for their original query.



3. Optimize Lineup

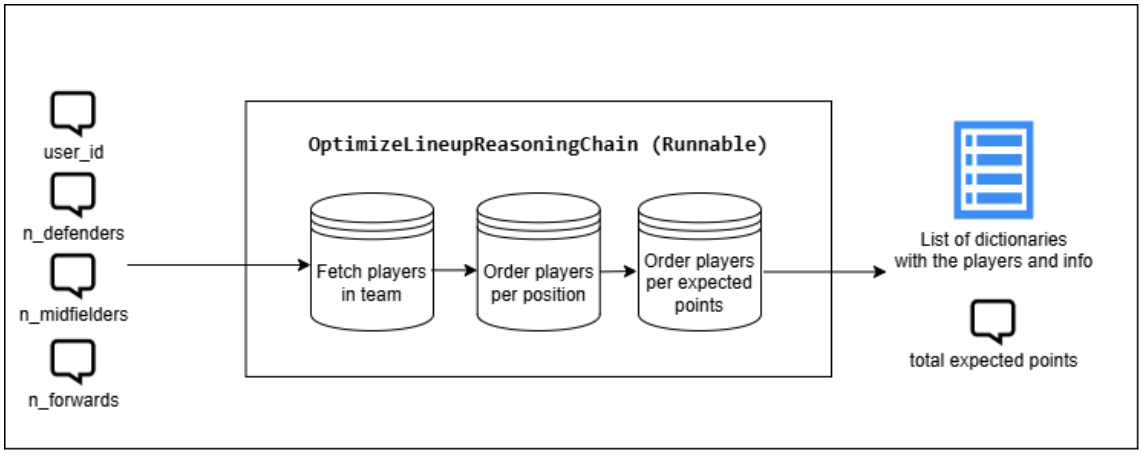
3.1. Story

“As a fantasy football player, I want a faster way to build my team and to know what is the most efficient way to use the players on my team by choosing the right ones to be on my starting eleven, so that I can outperform my friends consistently.”

3.2 Architecture Diagrams

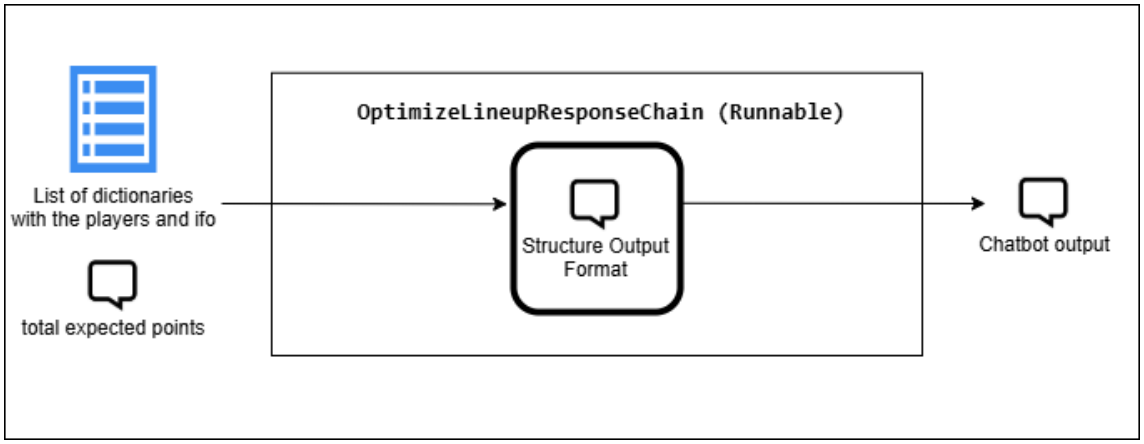
Goal: Build the best starting 11 according to the user’s players, and using a formation of their choosing by allowing them to choose the number of defenders, midfielders and forwards they want.

Implementation: We will implement a chain that can retrieve players that have the most expected points, and that fulfils constraints on the number of players per position. At the end the chain returns a list of dictionaries with the players’ names, expected points and price as well as the sum of the total expected points as an integer.



Goal: Structure the list of dictionaries containing all the previous information and the total expected points in a final output.

Implementation: We will implement a chain that receives the list of dictionaries with the players’ names, expected points and price as well as the sum of the total expected points as input and uses it to generate a final user output using a prompt template.



4. Check Upcoming Features

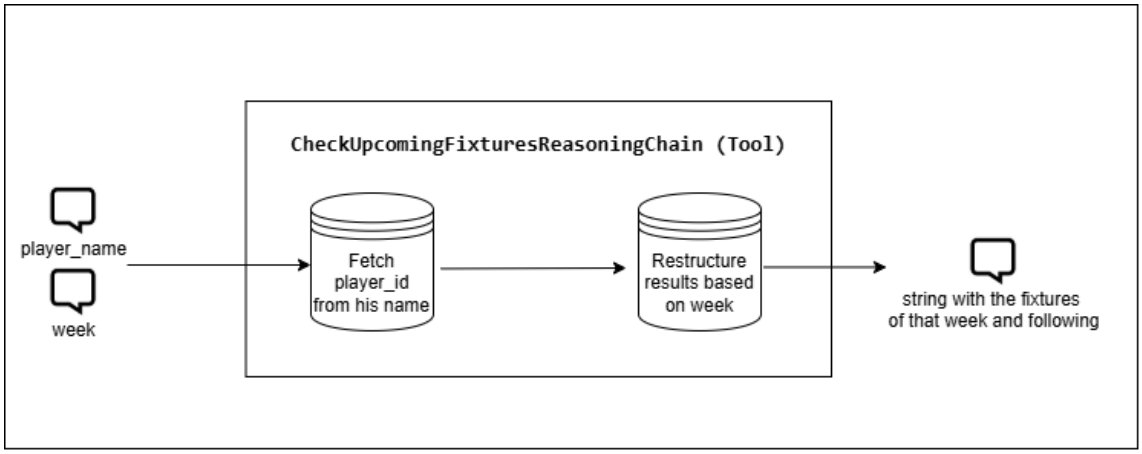
4.1. Story

"As a fantasy football player and a football fan, I want to be able to get access to what the next games of the players' in my team will be played against, so I can make decisions also based on the difficulty of the match instead of only in their expected points."

4.2 Architecture Diagrams

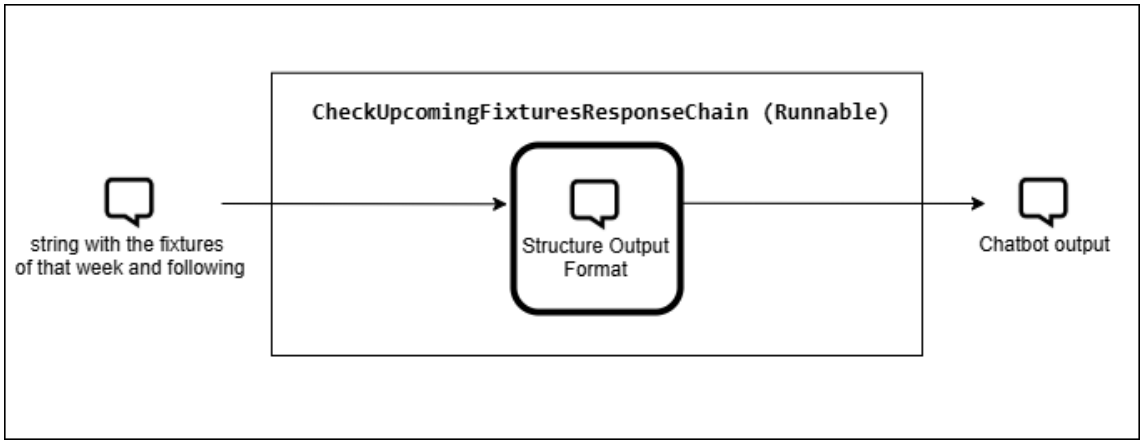
Goal: Allow users to get information on their players' next matches.

Implementation: We will implement a chain with a player's name and a week can retrieve the player id and team so that it can access that player's future fixtures from the defined week on. At the end the chain returns a string with the player's future fixtures as well as the dates.



Goal: Structure the string containing the player's fixtures in a final output.

Implementation: We will implement a chain that receives the string with the player's next fixtures as input and uses them to generate a final user output using a prompt template.



5. Stats and Fantasy Points Information

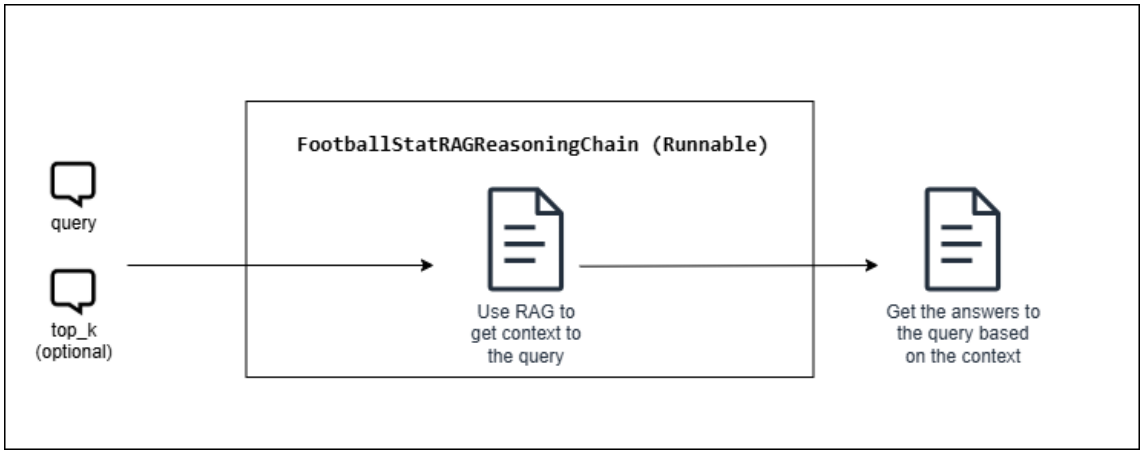
5.1. Story

“As a new fantasy football player and a football fan, sometimes I get confused on what some stats mean and specially confused on how players earn fantasy points, so I would want to be able to easily get the information I need to clarify my doubts.”

5.2 Architecture Diagrams

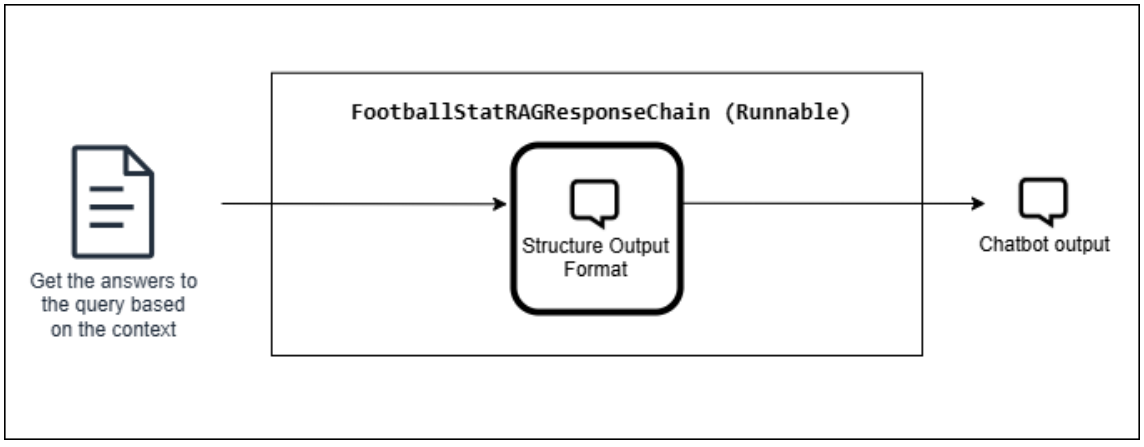
Goal: Allow users to clarify doubts regarding stats and fantasy points attribution.

Implementation: We will implement a chain that gets the user’s query and the number of most important documents it wants to retrieve (optional), to retrieve information from PDFs using RAG. At the end the chain returns the answers to the user’s query in a string format.



Goal: Structure the answers that are in a string format in a final output.

Implementation: We will implement a chain that receives the answers in a string as input and uses it to generate a final user output using a prompt template, summarizing and giving the user the most important information for their original query.



6. Perform Transfers

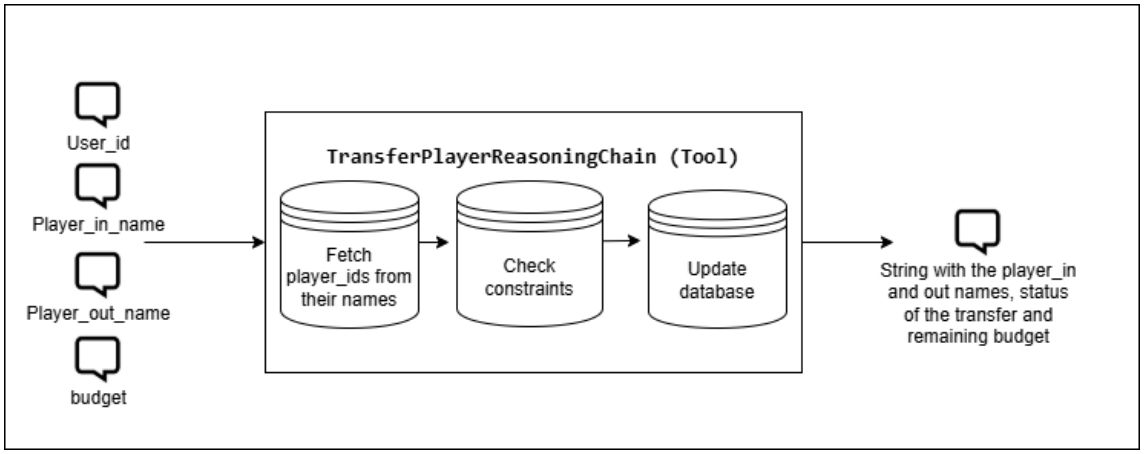
6.1. Story

"As a fantasy football player, I want to be able to store my team and make changes to it, so that it can be the same as the one on my EPL fantasy app even after doing transfers there and therefore getting more significant insights about my team."

6.2 Architecture Diagrams

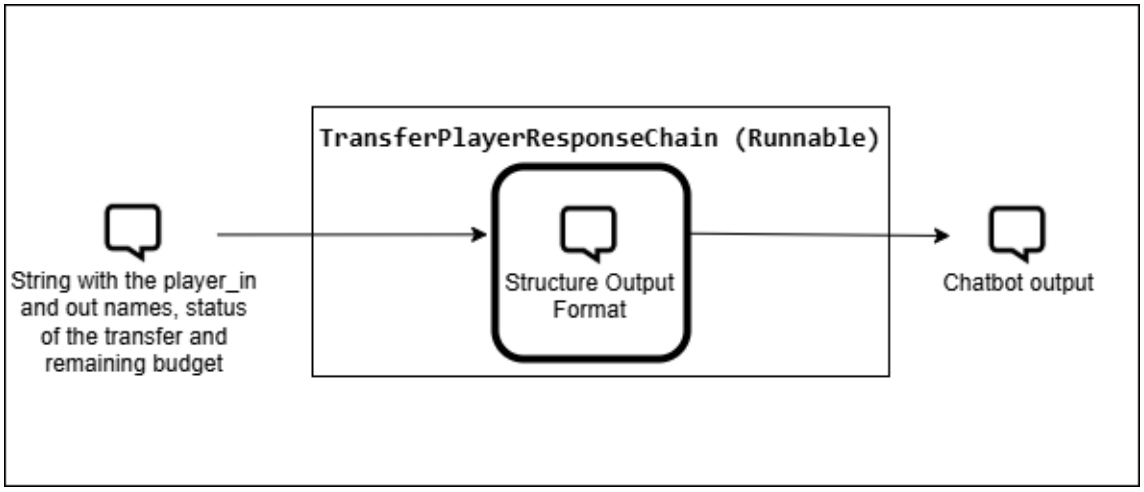
Goal: Enable users to make transfers on their teams, replacing one player by another one or simply adding a new player to their team.

Implementation: We will implement a chain that can retrieve the user's team and then go and remove a player from that team to allow for another one to replace that one, or if only given the name of one player it will add him to the team. In both cases the transaction is completed only if the available budget is not exceeded. At the end the chain returns a string with players' names, the remaining budget and the transfer status. This chain would be able to perform CRUD operations.



Goal: Structure the string with players names, the remaining budget and the transfer status in a final output.

Implementation: We will implement a chain that receives string with players names, the remaining budget and the transfer status and use it to generate a final user output using a prompt template.



7. View Player Stats and Price Evolution

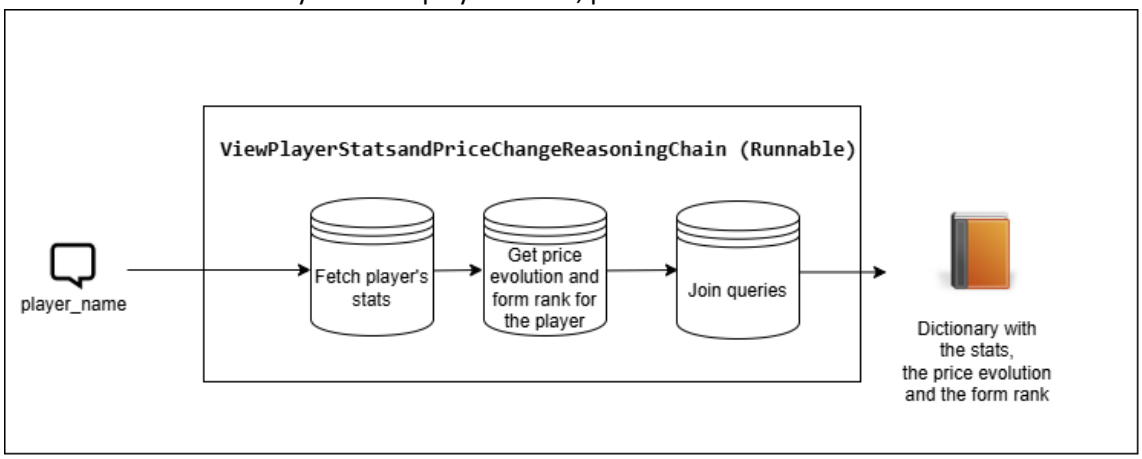
7.1. Story

"As a fantasy football player and a football fan, I want to be able to have easy access to the stats and how the fantasy price of any player is evolving. In order to make more money out of my players and also be able to understand how they have been performing this season."

7.2 Architecture Diagrams

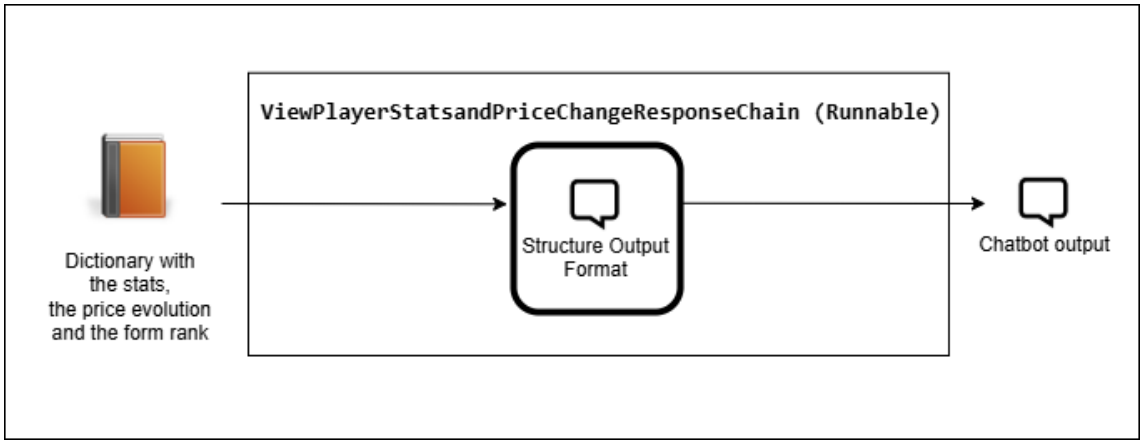
Goal: Enable users to see whether the price of a player has gone up or down compared to the week before, to see the player's stats and see his form rank.

Implementation: We will implement a chain that fetches the player's stats, then fetches his price evolution and form rank, by the end joins the results from both queries. At the end the chain returns a dictionary with the player's stats, price evolution and form rank.



Goal: Structure the dictionary with the player's stats, price evolution and form rank in a final output.

Implementation: We will implement a chain that receives the dictionary with the player's stats, price evolution and form rank and uses it to generate a final user output using a prompt template.



8. Adding or Removing players in starting eleven

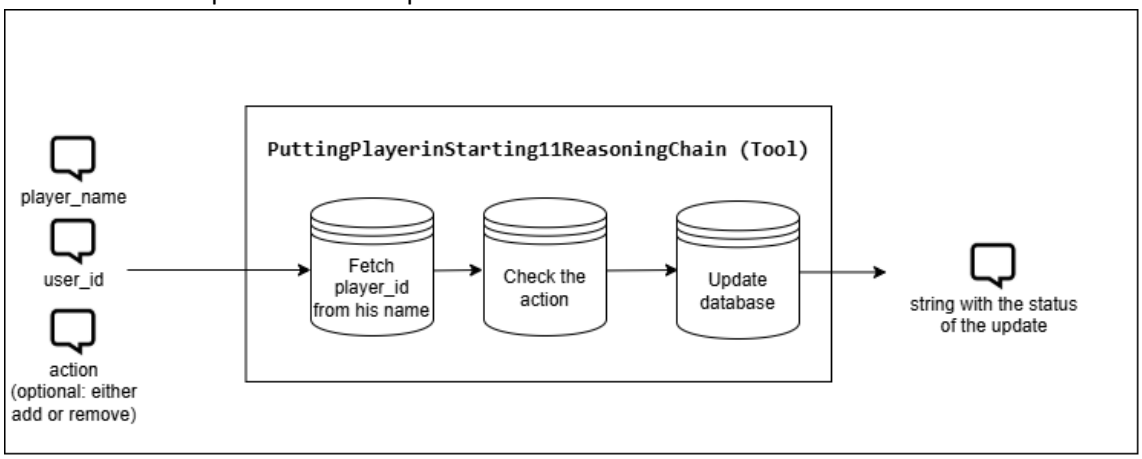
8.1. Story

“As a fantasy football player, I want to be able to alter my starting eleven so that it can be the same as my EPL even after the changes I make there.”

8.2 Architecture Diagrams

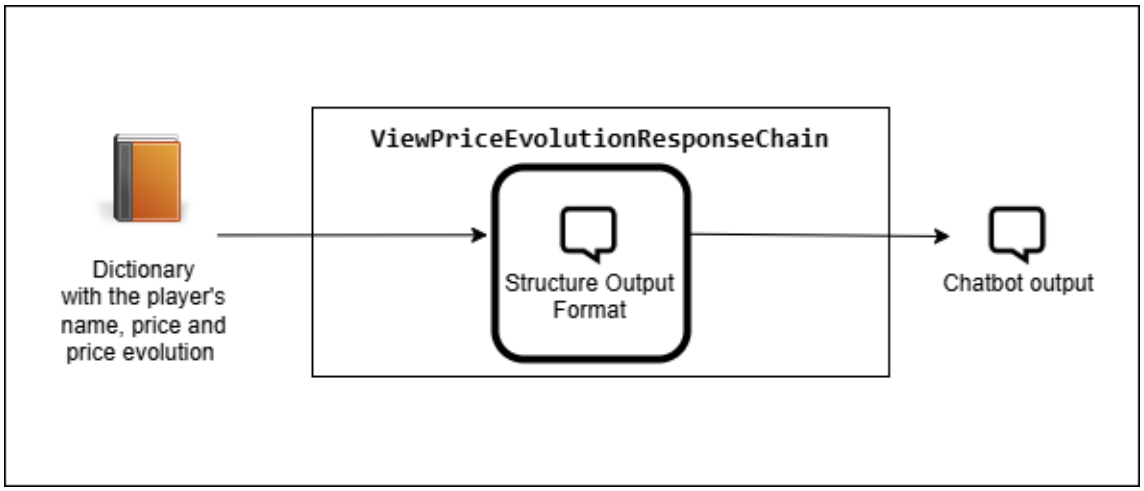
Goal: Enable users to alter which players are in their starting eleven (adding players and removing also).

Implementation: We will implement a chain that gets a player id from their name, then check what the user wants to perform (optional, default is add). Then the update is made on the user team if possible. At the end the chains return a string with the status of the update. This chain would be able to perform CRUD operations.



Goal: Structure the string with the update status in a final output.

Implementation: We will implement a chain that receives string with the update status and uses it to generate a final user output using a prompt template.



9. View or Update Points

9.1. Story

“As a fantasy football player, I want a way to be able to keep track of my points and to update them if needed, so that they remain the same as the ones in the EPL fantasy app and I can have access to them more easily and faster.”

9.2 Architecture Diagrams

Goal: Create an Agent that can deal with updating or just viewing the user’s points

Implementation: We will implement an Agent that will decide whether the user wants to update their points to a new value or wants to see how many points they have, then the user will proceed by satisfying the user’s need. If the user wants to update, then the agent will enable a tool to go and update the value of the user’s points in the database, outputting a string with the status of this operation. If the user just wants to view their points, the agent will enable a tool to go and fetch that value, outputting a string with the user’s points. Then the output of the tool used by the agent will be used to generate a final user output using a prompt template.