Criterion B: Design

Understanding Fantasy Premier League (FPL) game

During the design stage, it is important to consider FPL rules, which will be essential while optimizing team selection and choosing players in accordance with official FPL guidelines.

1) **Brief introduction to the game**

   FPL is an online fantasy football game where participants, acting as virtual team managers, create teams using players from the Premier League. Players have different price tags based on their perceived skill and performance. With a budget constraint, participants must manage wisely to build competitive teams. Points are earned based on real-life player performances, including goals, assists, and clean sheets, while negative actions lead to deductions. Managers can change their team lineup weekly for added flexibility.[1] (Scout, 2024)

2) **FPL rules and constraints to consider during team selection**

   a. Budget constraint: The total value of the squad must not exceed £100 million.
   b. Club constraint: A maximum of 3 players can be chosen from a single Premier League club.
   c. Positional constraint: The squad must be made of 15 players, consisting of 2 goalkeepers (GK), 5 defenders (DF), 5 midfielders (MF), and 3 forwards (FW)[2]. (League, 2024)

[1] https://www.premierleague.com/news/2173986
[2] https://fantasy.premierleague.com/help/rules

Preliminary user interface (UI) design

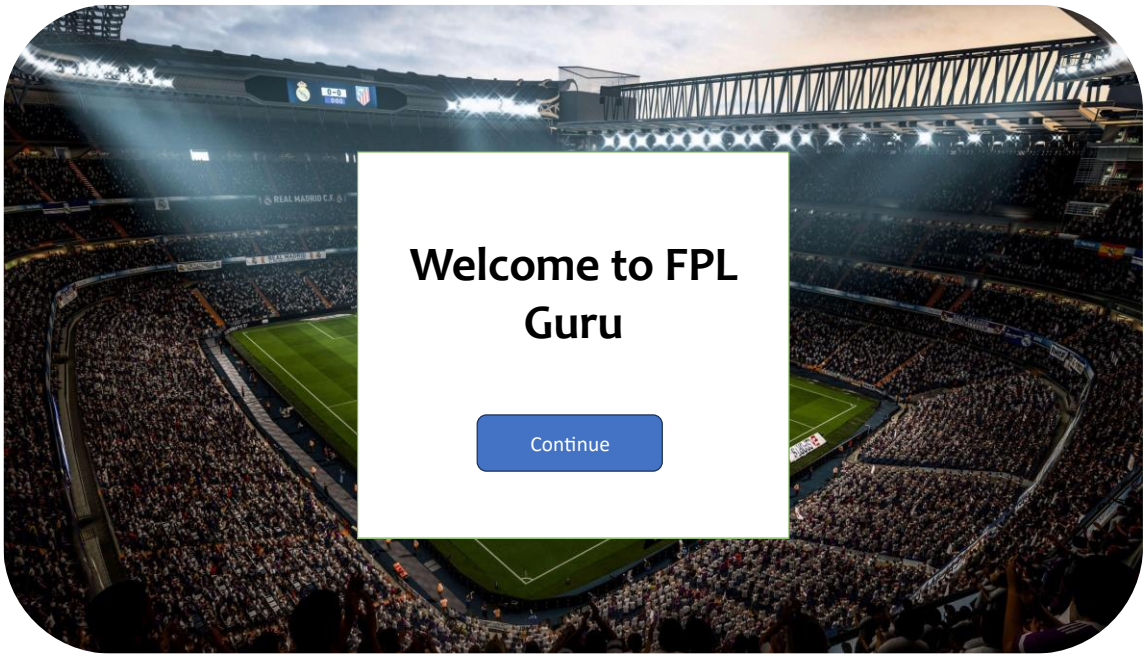


Figure 1: Home/Welcome page

Figure 2: Main window which shows the optimum team

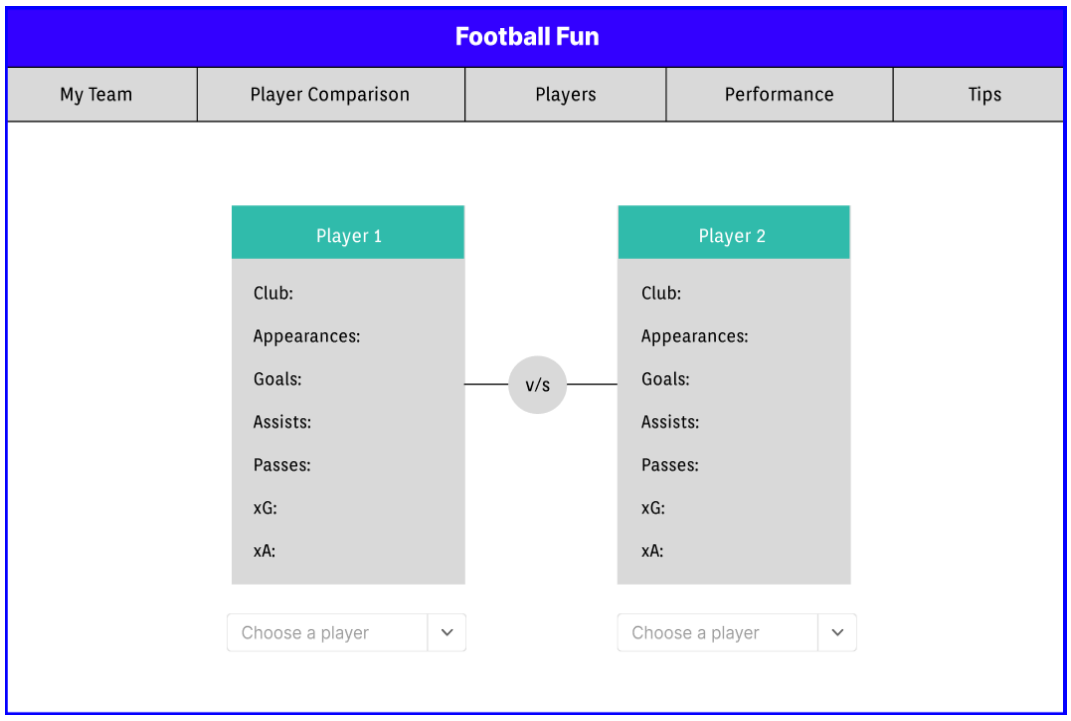


Figure 3: Comparing statistics of any two players from the database

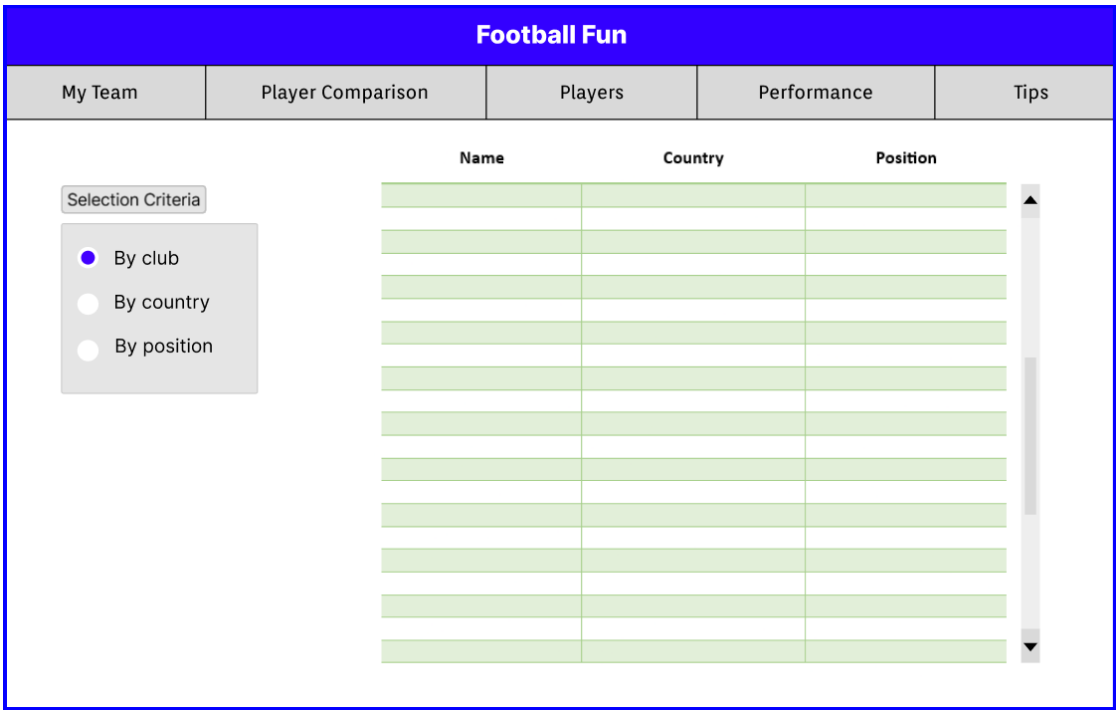


Figure 4: A list of all players in the database sorted through a selection criterion

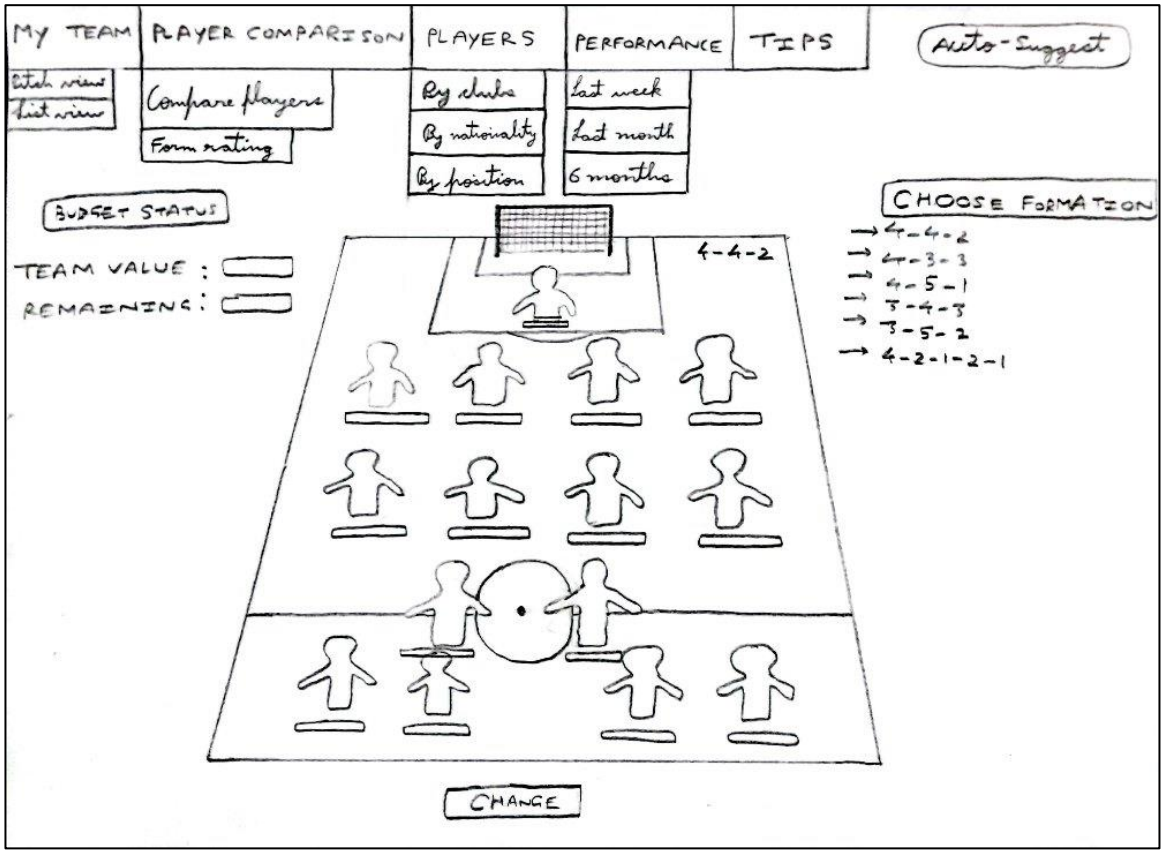
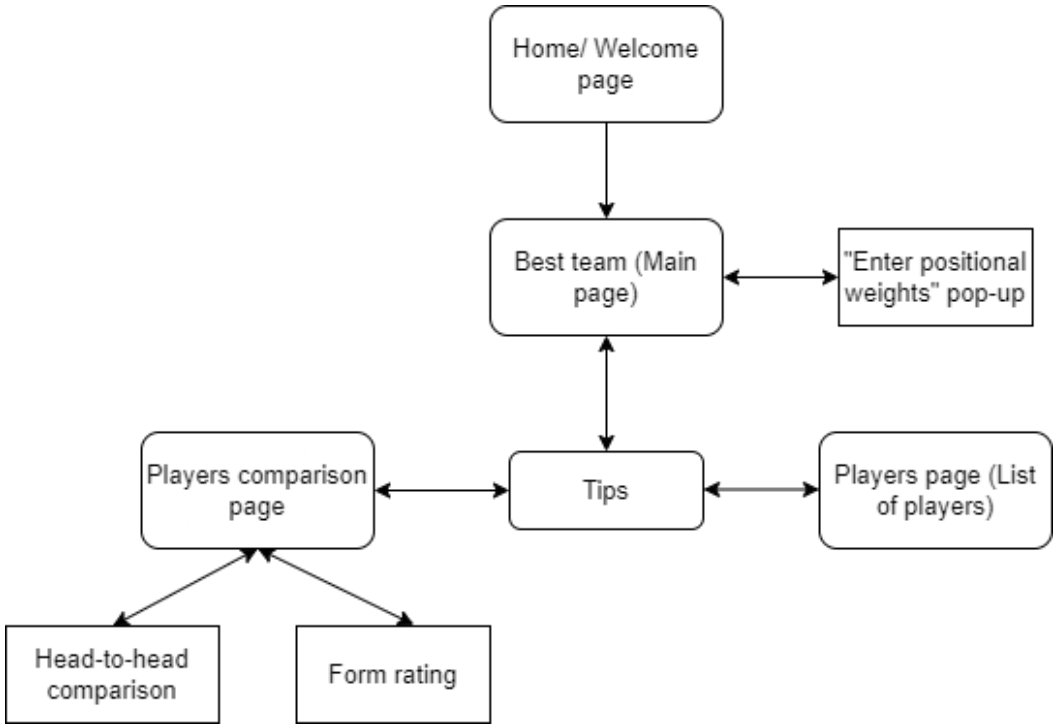
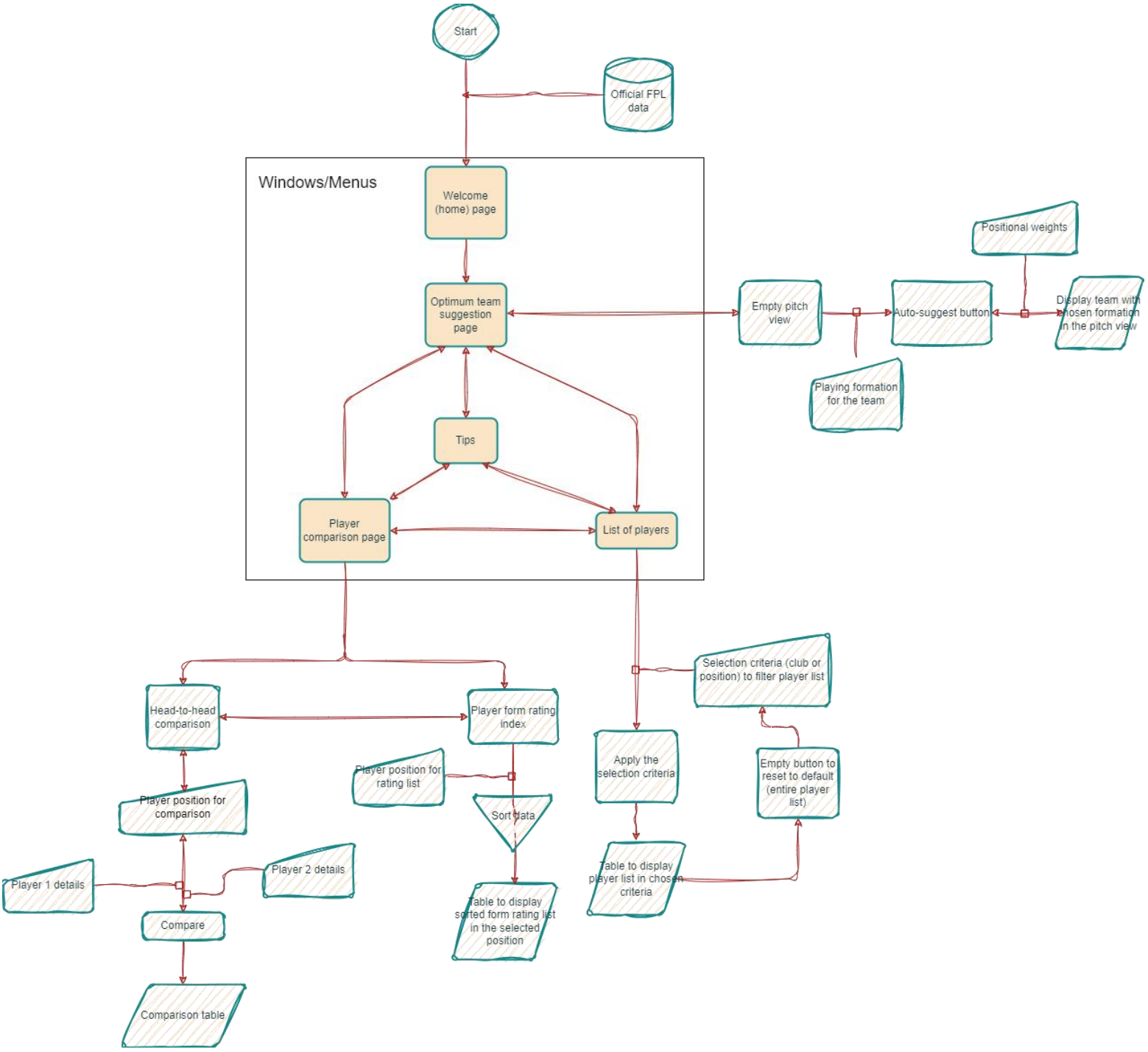


Figure 5: A hand-drawn detailed image of the pitch view for best team window

System storyboard



System flowchart describing the various functionalities and features of *FPL Guru*



Algorithmic problems

1) Parsing the required data, official FPL statistics, from online sources

2) Data management: storing players by different criteria (club and position)

3) Defining a form rating index

4) Ranking players on a relative form rating index

5) Optimizing the best team per game week based on budget restrictions and club and positional constraints

1) Parsing all necessary player data from official FPL statistics database

To analyze player statistics and optimize team selection in FPL, a dynamic connection is crucial for real-time updates after each Premier League match. Utilizing the FPL API, I'll gather and store essential player data, including general information (Name, Playing position, Club, FPL ID) and player statistics (Goals, Assists, Clean sheets, Red cards, Yellow cards, etc.) in a CSV file.

The information will then be stored in a dictionary with player names as keys and a nested dictionary holding player data as values. For example,

{“player 1”: {“club”: “Arsenal”, “goals_scored”: 10, “assists”: 4, …}, “player 2”: {“club”: “Liverpool”, “goals_scored”: 14, “assists”: 1, …}, …}

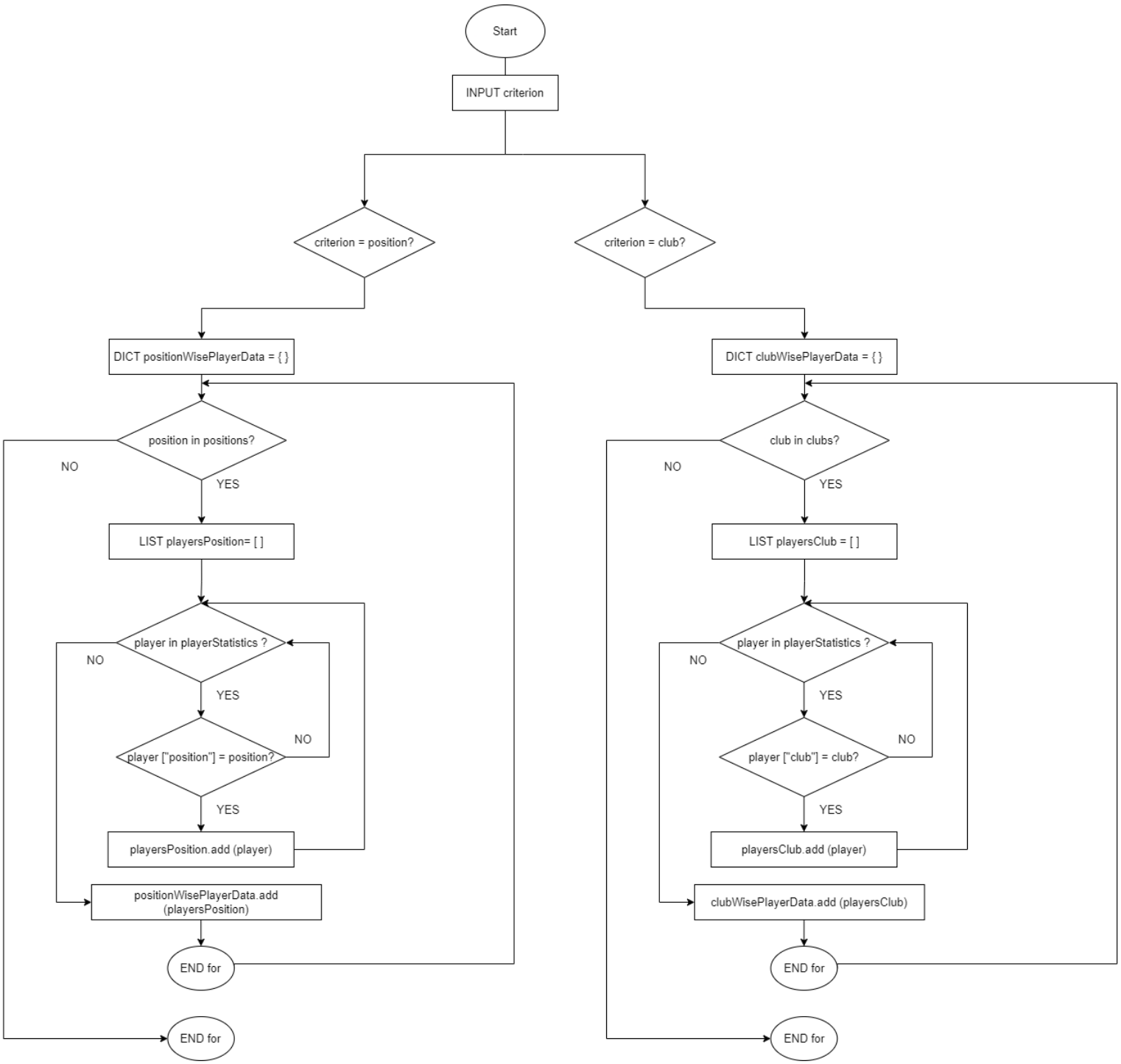
I would thus ensure that the software makes informed choices based on the latest data with each execution.

2) Sorting players by different selection criteria (position and club)

Input	1. *playerStatistics*: Players' data (input from FPL website using API) 2. *criterion*: Filtering criterion which can be either position or club (user input) 3. A list of possibilities for the filtering criterion (FPL specifics): ○ *positions:* For position, different playing positions: "FW", "GK", "MF", and "DF" ○ *clubs*: For club, different clubs: "Manchester City", "Liverpool", "Chelsea", etc.
Process	Sorting players based on the selected filtering criterion
Output	positionWisePlayerData/clubWisePlayerData: A dictionary containing the different criteria as the keys and a list of players who fulfill the criteria as values. For e.g., if the selected filtering criterion is position, the output would be of the form: {"FW": [A list of all forwards], "MF": [A list of all midfielders], "GK": [A list of all goalkeepers], "DF": [A list of all defenders]}

Context: Due to the club and positional constraints in FPL, Mr. X, requires the program to sort players based on positions and clubs to help him make decisions while choosing his team.

Explanation: The algorithm creates dictionaries, *positionWisePlayerData* and *clubWisePlayerData*, enabling filtering by position or club criteria. It generates lists (*playersPosition* or *playersClub*) based on specified criteria like "FW" for position or "Manchester City" for club. During the player data loop, players meeting the criteria have their information added to the respective list, which is then appended to *positionWisePlayerData* or *clubWisePlayerData* accordingly.


Start
INPUT criterion
criterion = position?
criterion = club?
DICT positionWisePlayerData = { }
DICT clubWisePlayerData = { }
position in positions?
NO
YES
club in clubs?
NO
YES
LIST playersPosition= [ ]
LIST playersClub = [ ]
player in playerStatistics ?
NO
YES
player in playerStatistics ?
NO
YES
player ["position"] = position?
NO
YES
player ["club"] = club?
NO
YES
playersPosition.add (player)
playersClub.add (player)
positionWisePlayerData.add (playersPosition)
clubWisePlayerData.add (playersClub)
END for
END for
END for
END for

3) Defining a form rating index

During my interviews with him, Mr. X explained his preferences and accordingly suggested a form rating index (Appendix 3), specific to different playing positions, which is as follows:

Position	Form rating calculation method	Exact weights given to each factor	
		Factor	**Weight (%)**
Goalkeeper (GK)	Calculated by combining the player's saves, clean sheets, penalties saved, goals conceded, total points, and percentage selection.	Saves	10
		Clean sheets	70
		Goals conceded	-20
		Penalties saved	10
		Total points	10
		Percentage selection	20
Defender (DF)	Determined based on a combination of clean sheets, starts, goals, assists, total points, and percentage selection.	Starts	10
		Clean sheets	30
		Goals	20
		Assists	10
		Total points	10
		Percentage selection	20
Forward (FW)	Computed by considering goals, assists, expected goals (xG), total points, and percentage selection.	Goals	55
		Assists	10
		Expected goals (xG)	5
		Total points	10
		Percentage selection	20
Midfielder (MF)	Derived from a combination of assists, goals, expected assists (xA), total points, and percentage selection.	Assists	25
		Goals	40
		Expected assists (xA)	5
		Total points	10
		Percentage selection	20

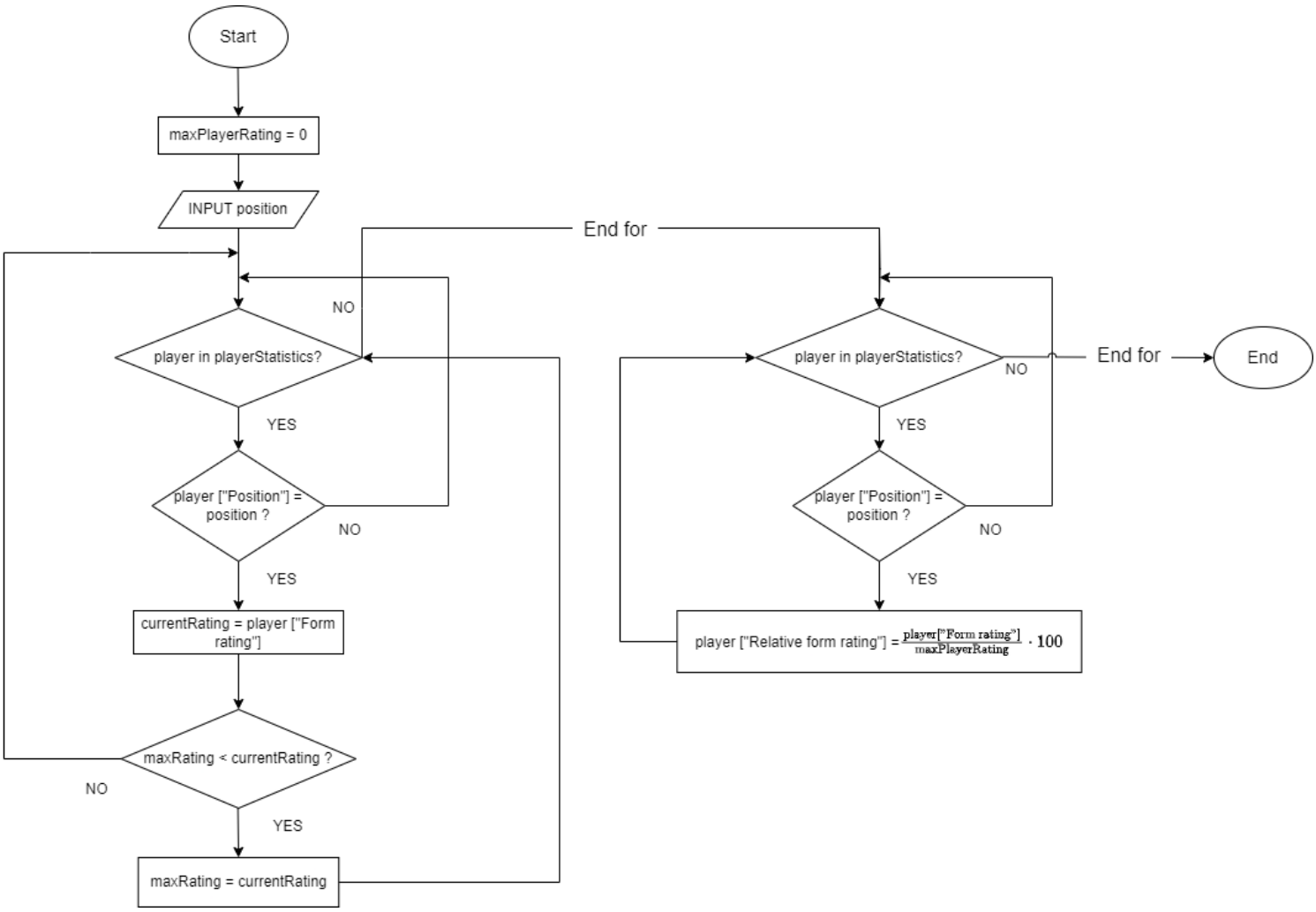
Exact details of the different stats, required for better understanding of the weighting method, can be found in the data dictionaries included in the documentation of criterion C.

4) Creating a relative form rating index for the players based on other players

Input	1. *position*: Playing position 2. *playerStatistics*: Players' data (input from FPL website using API) 3. *currentRating*: Player's form rating calculated as defined above in Table
Process	Assigning relative form rating to each player calculated by comparing to other players' form rating
Output	*Relative form rating:* Relative form rating is added to each player's individual information

Context: To form the optimal team, identifying the top players in their respective positions is crucial. Mr. X tasked me with developing a rating index that assigns a 100.0 rating to the best player in each position and then rates other players relative to this benchmark.

Explanation: The algorithm initializes a variable, maxPlayerRating, to 0.0. It iterates through all players, identifying the maximum form rating for the targeted position and assigning it to maxPlayerRating. Subsequently, in the next loop, each player's relative rating is calculated as a percentage of the maximum rating (100.0).

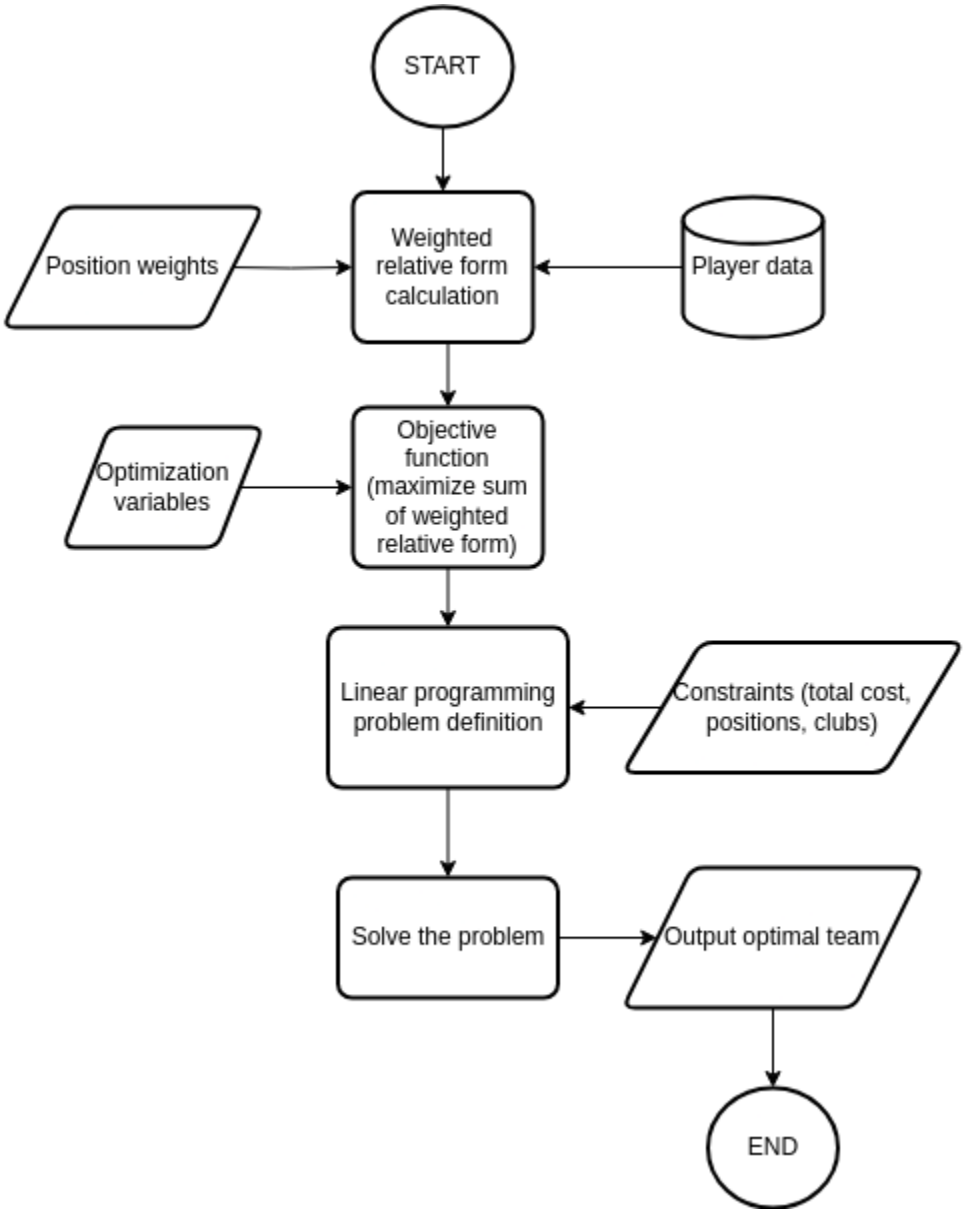

Start
maxPlayerRating = 0
INPUT position
player in playerStatistics?
NO
YES
player ["Position"] = position ?
NO
YES
currentRating = player ["Form rating"]
maxRating < currentRating ?
NO
YES
maxRating = currentRating
End for
player in playerStatistics?
NO
End for
End
YES
player ["Position"] = position ?
NO
YES
player ["Relative form rating"] = player["Form rating"] / maxPlayerRating · 100

5) Optimizing team based on constraints

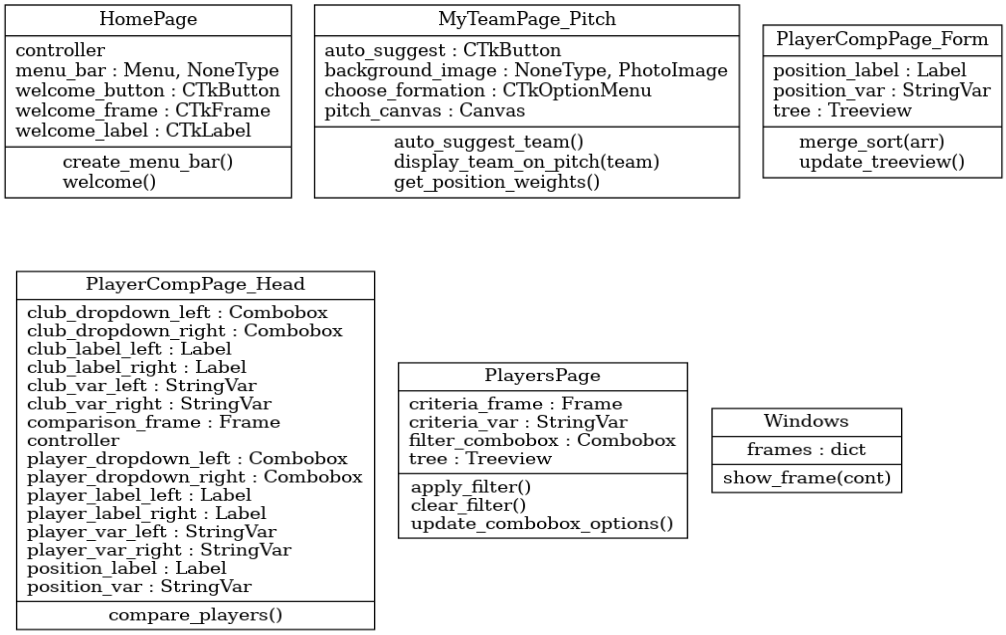
Input	1. Position weights: Positional weights (user input) 2. Player data: Official FPL statistics of players (input from FPL website using API) 3. Constraints (total cost, positions, clubs): Financial, club, and positional constraints (input according to in-game rules of FPL) 4. Relative form rating (calculated based on user input and players' data)
Process	Choosing a 11-player starting lineup using convex linear optimization[3] *(Skinner, 2023)* based on inputted constraints
Output	Optimal team: The optimum team is selected and starting lineup is shown on the pitch view menu.

It is important to note here that the exact mathematical definition and explanation of the convex linear programming approach used to optimize the team is beyond the scope of this IA. Hence, the flowchart below represents a high-level abstraction of the algorithm's implementation.

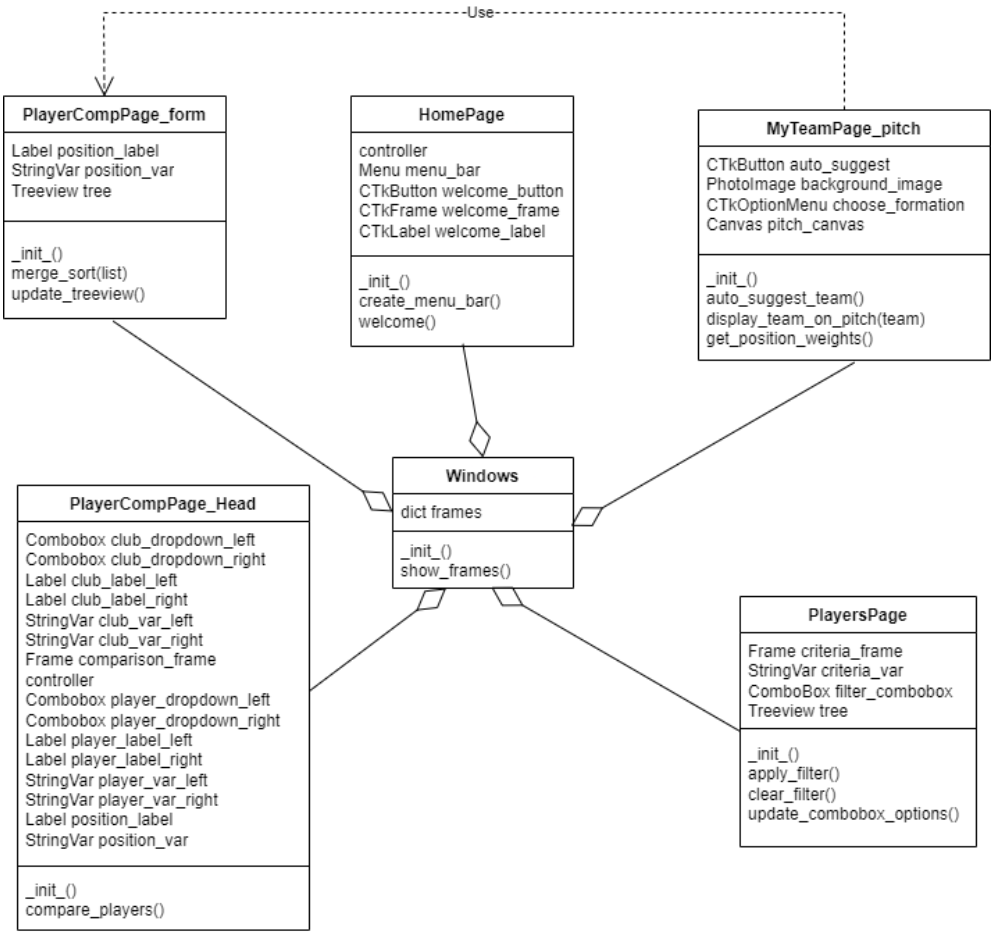
[3] https://medium.com/@dylanskinner65/convex-linear-optimization-with-cvxpy-5fa1024254ff


START
Position weights
Weighted relative form calculation
Player data
Optimization variables
Objective function (maximize sum of weighted relative form)
Linear programming problem definition
Constraints (total cost, positions, clubs)
Solve the problem
Output optimal team
END

Single classes' representations



UML class diagram



Discussion of significant data structures

Data structure	Description and requirement
Pandas DataFrame (df)	▪ Data structuring o A df will be used to organize and structure the data obtained from FPL API which is in JSON format. It will also allow for a tabular representation of all the player information, as opposed to an in-line textual form, which would be more convenient for me, the developer. ▪ Integration of multiple data sources o A df will be needed to merge and join information from different API endpoints. ▪ Efficient data export o To easily use the extracted FPL data, it will be stored in structured CSV files. These files can be conveniently accessed through a DataFrame.
Dictionary	▪ I will use a dictionary to store player information since it will allow for a standard key-value pair system in which each player will have the same keys, such as “club name”, “player name”, “goals scored”, etc., but different values for the keys. ▪ A dictionary will also be needed to store the information about different playing formations, wherein the positions remain same, however the number of players in each position differs. For e.g., the formation “4-4-2” can be represented using the following dictionary: {“FW”: 2, “MF”: 4, “DF”: 4, “GK”:1}. ▪ As seen above in the algorithmic design, dictionaries would be required to store the sorted information about players filtered according to their playing positions and clubs.

Test plan

Success criteria	Test description	Input data	Expected outcome
1	User can filter the list of players according to the playing position.	Select a playing position out of "FW," "DF," "GK, "and "MF."	A filtered table containing a list of all players in the chosen position appears.
1	User can filter the list of players according to the players' clubs.	Select a club from the drop-down menu.	A filtered table containing a list of all players from the chosen club appears.
2	User can compare statistics of two players playing in the same position.	Select a playing position out of "FW," "DF," "GK, "and "MF." Choose two players playing in that position (test for players from the same clubs as well as different clubs).	A comparison table containing the head-to-head statistics of the two players is shown.
3	User can view a form rating table which ranks players, based on statistics, against other players playing in the same position.	Select a playing position out of "FW," "DF," "GK, "and "MF."	A relative form rating table ranks the players playing in the chosen position in a descending order. It also shows their current costs in FPL.
4, 5	The optimal team for the gameweek is suggested according to the selected playing formation and positional weights and shown through the GUI interface.	a. Select a playing formation from the drop-down menu. Enter positional weights, such as "2,3,1,4."	The best possible team of 11 players for the selected formation and positional weights is computed and represented on the pitch view in the window.
		b. Select a playing formation from the drop-down menu. Enter erroneous positional weights which don't add up to 10, such as "2,3,1,3." (Error handling)	An error message box appears stating that the positional weights don't add up to 10.

		c. Select a playing formation from the drop-down menu. Enter erroneous positional weights with string values such as “2,eded,1,wedw. (Error handling)	An error message box appears stating that positional weights must be integer values.
6	User is given tips to play FPL.	Click on the “Tips” menu.	A pop menu appears with a tip/suggestion for in-game strategies.

Word count: 530

Works Cited

League, P. (2024, February 1). *Fantasy Premier League, Official Fantasy Football Game of the Premier League - Rules*. Retrieved from Fantasy Premier League: https://fantasy.premierleague.com/help/rules

Scout, T. (2024, January 15). *FPL basics: What is FPL and how to play*. Retrieved from Fantasy Premier League: https://www.premierleague.com/news/2173986

Skinner, D. (2023, October 20). *Convex Linear Optimization with CVXPY*. Retrieved from Medium: https://medium.com/@dylanskinner65/convex-linear-optimization-with-cvxpy-5fa1024254ff