Soccerdata Predictor and Stat Viewer

Student Name: Aryan Joshi

Date Due: 4/07/2025

Course: Software Engineering Stage 6

GitHub URL: https://github.com/AJOSH0912/HSC-Major-Work

1. Project Requirements

1.1 Problem Statement

This project has strong real-world relevance as football is a sport watched by billions of people. This Python project will give them and the team coaches a statistical insight into the performance of the teams they support. Additionally, Premier League coaches will be able to utilise this project to analyse player performance and team performance, simplifying their job as managers.

1.2 Project definition and purpose

My software engineering solution aims to use existing football data to create a Machine Learning model to predict the outcome of future matches based on user input. Additionally, the user will be able to access team seasonal stats and player stats on the UI. There are many real-world applications of this software, such as Premier League coaching, where coaches can use this project to make statistics-based decisions for their team. Another possible application is sports analytics and media, where journalists can use predictions and data to inform pre-game discussion content, influencing the sports media industry. Additionally, this can be used by TV broadcasters to increase fan engagement and improve their broadcasting quality with relevant statistics.

1.3 Stakeholder Requirements

The stakeholders are Premier League coaches who require a reliable source of data and predictions to assist them in making decisions for their teams. The stakeholder requires an application that displays team and player stats while also providing predictions for upcoming matches.

1.4 Functional Requirements

Requirement	Explanation
Data Collection and Integration	• The system must efficiently fetch recent football, ensuring freshness and accuracy.
Match Outcome Prediction	• The ML model must identify meaningful features to make reliable predictions. • Should avoid biases in training data • Should display the probabilities of teams winning the match
Team Statistics Viewing	• Users should be able to see the seasonal statistics of any current Premier League team • Users should be able to decide which data they would like to view for their respective team
Player Statistics Viewing	• Users should be able to see the seasonal statistics of any current Premier League player • Users should be able to decide which data they would like to view for their respective players
Web Application	• Users should be able to access the

	UI flawlessly with minimal to no need for how to use it. ● The UI must display all the project's functions.
Data Security	● The system should encrypt all data to ensure that it is safe

1.5 Non-Functional Requirements

Requirement	Explanation
Performance and Efficiency	● The system should fetch data and run predictions in seconds to maintain usability
Scalability and Maintainability	● Code design allows for future expansions ● Retraining the ML model
Accessibility	● Simple, responsive UI ● Data visualisation should be clear ● and actionable
Restratibility	● It must be possible to regenerate the ML model if there is data loss or corruption

1.6 Constraints and Limitations

Time

Time is the main constraint of this project as it limits the number of features that can be added to my software project. This constraint of time ruled out possibilities like player vs player predictors and match predictions from multiple leagues.

Data Limitations

There will be limitations when it comes to data, since some sites may have limited data that is relevant to a machine learning model, and if we take the optimal data from multiple sites, there will be large amounts of merging issues when trying to assemble them into one file for the ML model.

Model Limitations

Models like random forests and logistic regression struggle to identify results with 3 or more target outcomes. This significantly reduces the model's accuracy compared to prediction 2 target outcomes, possibly creating class imbalances in one of the outcomes.

2. Project Research and Planning

2.1 Development method

The development method that is used in this project is agile development due to its straightforward structure and its potential for constant improvement during the development phase. Additionally, the agile method is flexible, allowing for the process to be easily adapted to suit any issues that may arise while coding.

The use of the agile method in this project:

Planning: Identify the problem statement, purpose, functional and non-functional requirements, and the constraints of the project.

Design: Create diagrams to plan how to format the code

Development: Develop the code based on the diagrams made in the design phase

Test: Test for potential data leakages

Deploy: Deploy the ML model to allow it to predict future outcomes.

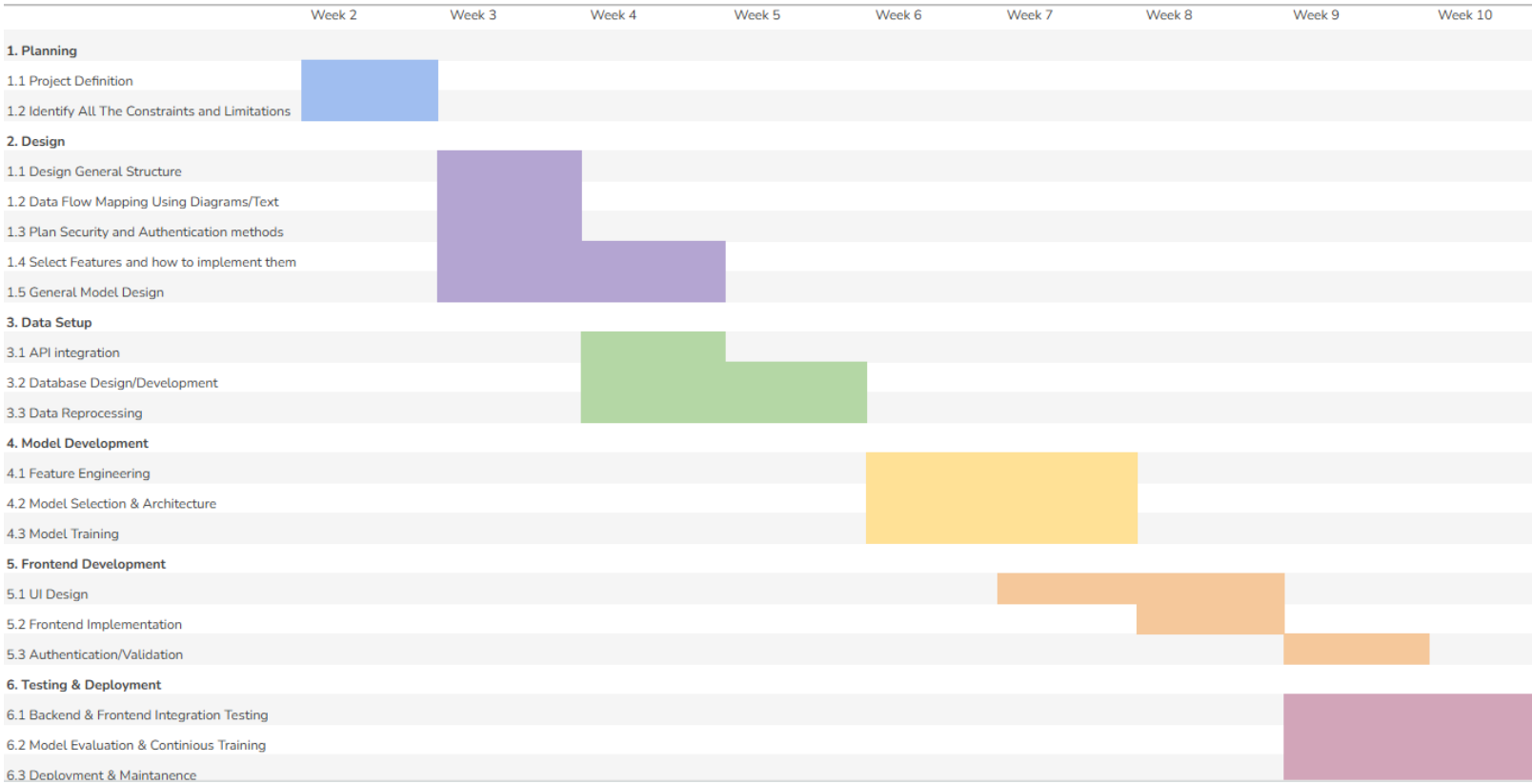
Review: Identify how well the ML model went and if there were any class imbalances.

2.2 Tools and Resources

In this project, I will use several different tools as shown below:

- Soccerdata GitHub project: Soccerdata will be used to get the raw data
- Python: Python will be used as the main coding language
- Pandas: Pandas will be used to format the data
- Gradio: Gradio will be used as the UI
- Numpy: Numpy will be used for the concatenation of multiple datasets
- Visual Studio Code + GitHub Co-Pilot
- Github

2.3 Gantt Chart



2.4 Communication Plan

To ensure a solid client-developer communication, I will be regularly checking with my client (Harjas Singh), who will be able to provide feedback throughout my project and determine whether I have been effectively working towards a complete project that

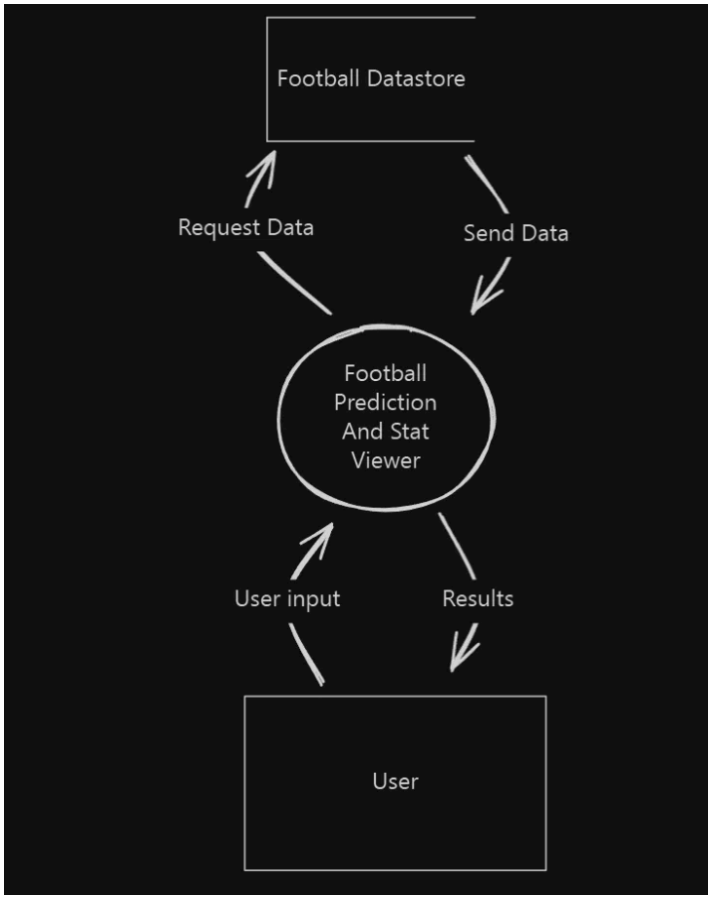
meets all his expectations as a Premier League coach. These meetings will be in person and scheduled once a week.

3. Project Design

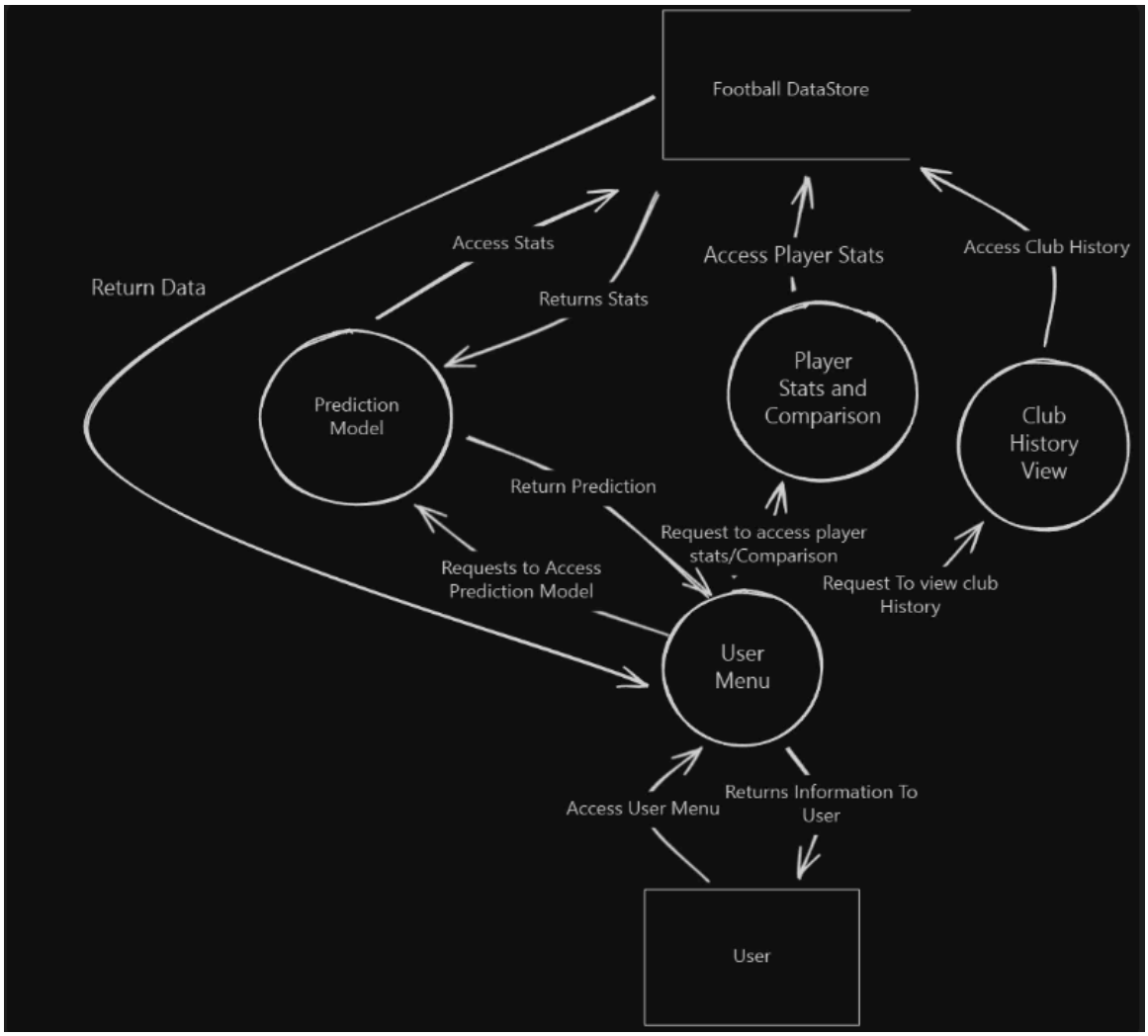
3.1 Diagrams

In the project design phase, I created several diagrams that would represent how my code would be assembled and how the data would flow. These diagrams consisted of the context diagram, level 1 DFD, level 2 DFD, structure chart, data dictionary and an IPO Chart.

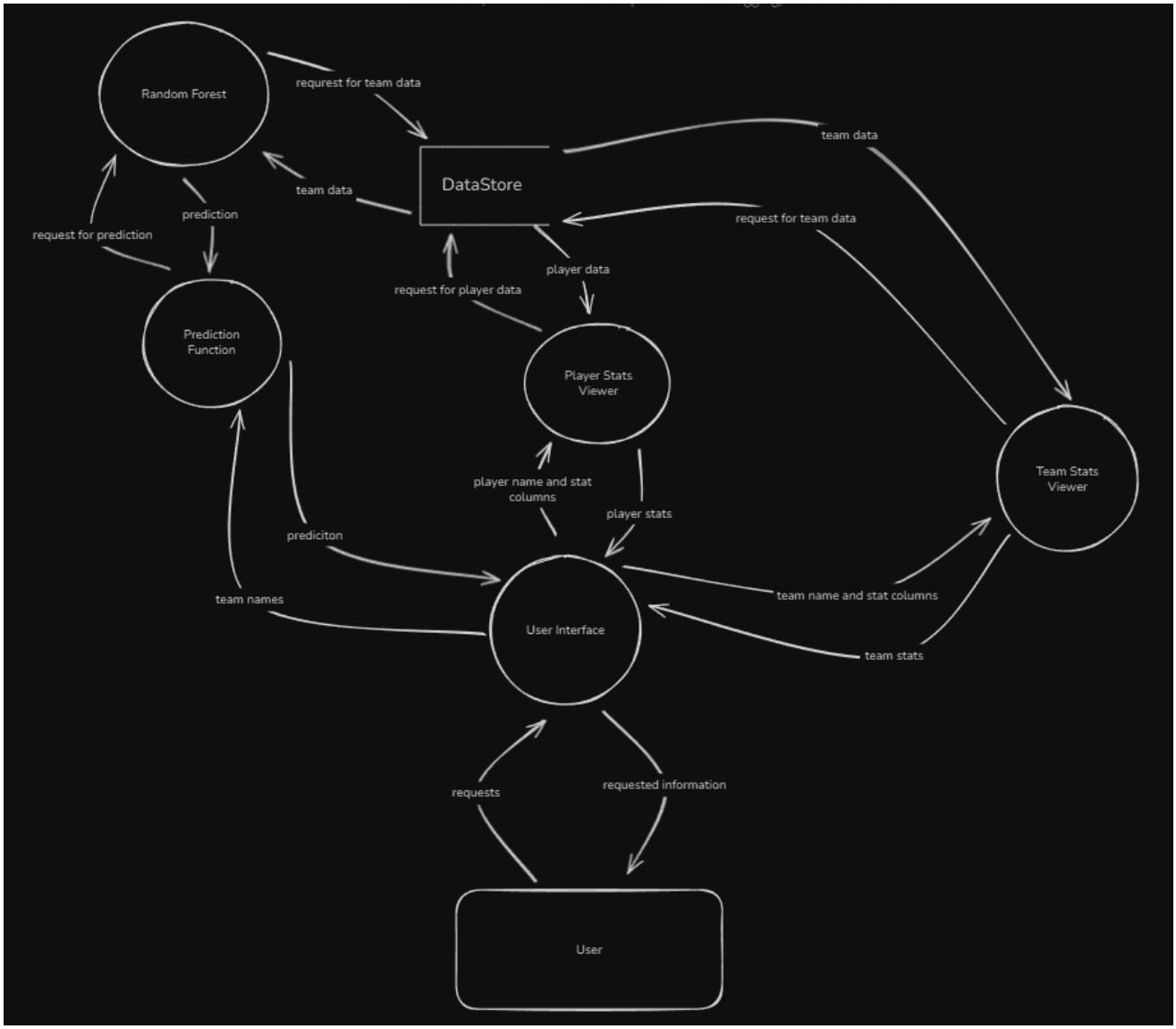
3.2 Context Diagram:



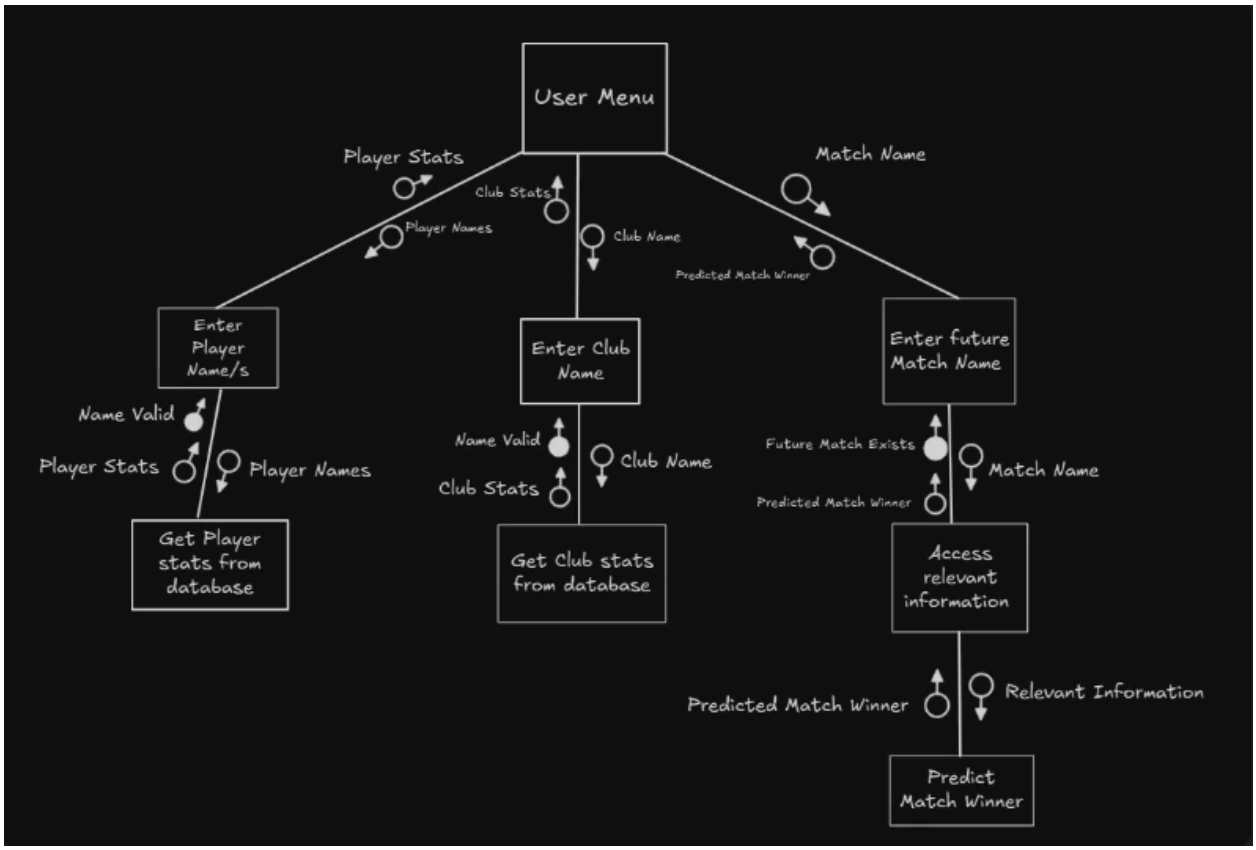
3.3 Level 1 DFD:



3.4 Level 2 DFD User Interface:



3.5 Structure Chart:



3.6 Data Dictionary:

Field Name	Data Type	Description	Example Values	Constraints
team_x	String	Home team name	"Arsenal" , "Liverpool	Must be a valid PL team
team_y	String	Away team name	"Liverpool", "Arsenal"	Must be a valid PL team
FTR	Float	Full-time Result	"1", "2", "3"	1 = Home Win, 2 = Draw, 3 = Away Win
team_1_Average_goals_scored_last5	Float	Home team's average goals scored in last 5 games	1.8 , 2.4	≥ 0
team_1_Average_HT_goals_scored_last5	Float	Home team's average half-time goals in last 5 games	0.8, 1.2	≥ 0
team_1_Average_shots_last5	Float	Home team's average shots in last 5 games	12.4, 15.6	≥ 0
team_1_Average_fouls_last5	Float	Home team's average fouls in last 5 games	10.2, 13.4	≥ 0
[Similar fields for the rest of team_1 and team_2 with last5 and last10 variations]	Float	Home/Away team statistics (previous structure)	Numbers	≥ 0

3.7 IPO Chart:

INPUT	PROCESS	OUTPUT
Data Sources:	**Data Processing & Model Training:**	**Predictions & Statistics:**
- Historical match data (CSV)	- Data cleaning and validation	- Match outcome predictions
- Team seasonal statistics	- Feature engineering	- Win/Draw/Loss probabilities
- Player performance data	- One-hot encoding of teams	- Team performance statistics
- Betting odds data	- Rolling averages calculation	- Player statistics display
User Inputs:	- Train/test data splitting	- Model accuracy metrics
- Home team selection	- Random Forest model training	- Classification reports
- Away team selection	- Model evaluation and scoring	- Error messages
- Statistical preferences	- Model serialisation	- Data visualisations
- Player search queries	**User Interface Processing:**	**Saved Files:**
Configuration:	- Input validation	- Trained model (joblib)
- Model parameters	- Data retrieval from files	- Model metadata
- File paths	- Feature preparation	- Teams list
- Random state settings	- Prediction generation	- Column specifications

3.8 Software Security Features

This project has included encryption and decryption as part of its security features to protect the data. Encryption is the only current logical security feature in the application, though once it is publicly launched and use other aspects like SQL tables, other security features like HTTPS, access controls, and input validation can be used as other security features. These features respectively provide a secure way for websites to communicate with web browsers, allow the regulation of resources and data, and ensure that no malicious data is processed in the system.

4. Project Development

4.1 Development Process

As mentioned before, I used the agile method as the overall development process, though within this process, I split up the production of the project into 4 separate sub-stages: data collection/cleaning, feature engineering, model building/training, and UI preparation.

Data Collection/Cleaning

In data collection, the main aim was to create the largest raw data files possible while also merging them in a format that is readable by the user and the ML algorithm.

Feature Engineering

During feature engineering, I added several more columns to my raw data, including rolling averages for the past 10 matches, past 5 home games, and past 5 away games for each team. This data is necessary for engineering since the model needs data to predict off and by calculating rolling averages, we can give this to the model to predict their future outcome.

Model Building

In model building, I created a simple random forest model where I hypertuned to parameters to get the best possible accuracy and recall. The hypertuning helped me identify the optimal setting for the model (Eg, 300 trees built, test data = 0.3)

UI preparation

The UI preparation consisted of creating the design of the UI and the functions that predict future match outcomes, identify team stats, and identify player stats. To create the UI, I created 3 separate tabs for all my functions and then created an overall interface that allows the users to switch between the features.

4.2 Key Features Developed

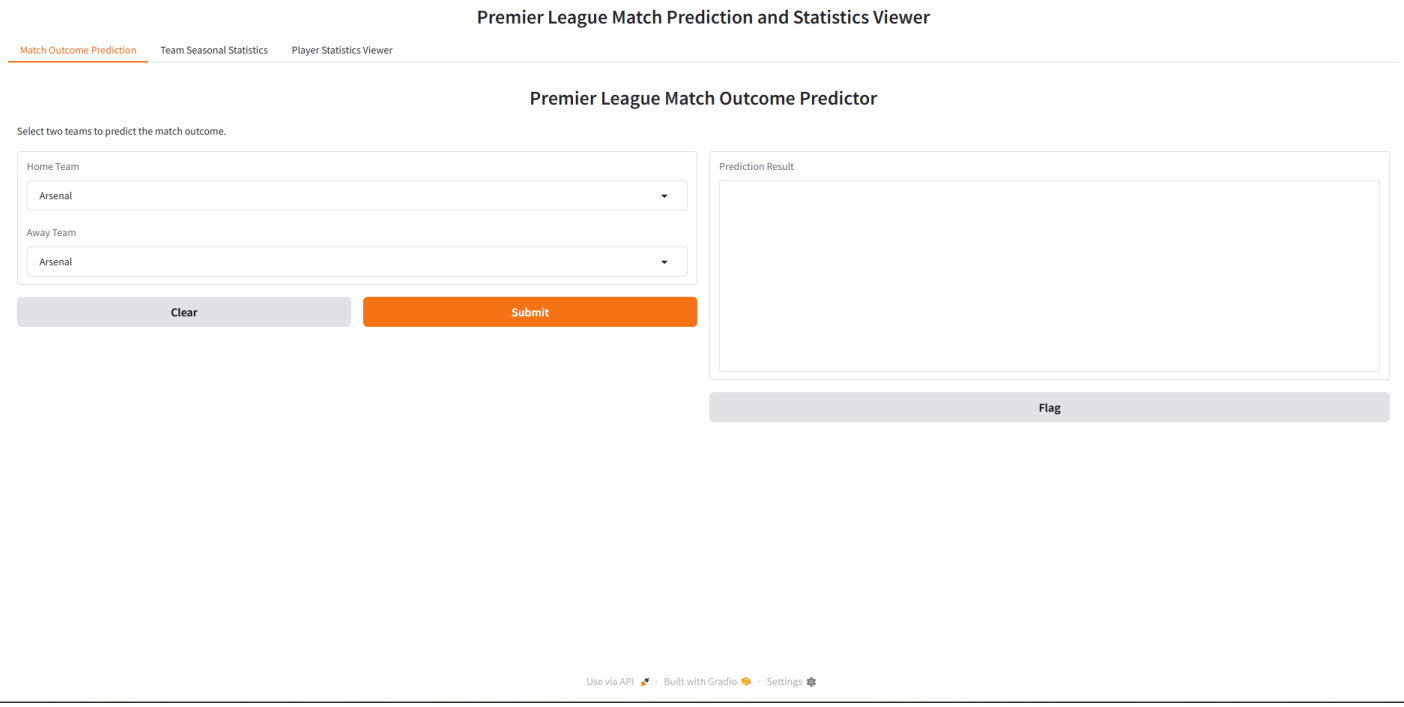
Feature	Description	Justification
Web Application	The Gradio UI is how the user interacts with the program and its functions	The Gradio UI interface is an essential feature since it is how to user can access all the other code. It

		allows the user to access match predictors, player stats and team stats, which are the main features of the program. Using gradio creates an accessible and simple visual interface for individuals, which is why it is used.
Random Forest model	The random forest is used to predict the outcome of future matches	The model is necessary in this project since it is the only method that allows us to make a prediction of future matches based on large datasets. The random forest is used since it handles large datasets quite well, and compared to other models, it is able to predict a three-way target significantly better.
Team Stats function	A function that calls the team stats based on user input	This is necessary as it uses user input to return the respective data. Without this function, the team_stats data will be useles,s as there is no way for the user to access it through the interface. Additionally, creating a function allows for quick maintenance and improvements, improving user satisfaction
Player Stats function	A function that calls the player stats based on user input	This is necessary as it uses user input to return the respective data. Without this function, the player_stats data will be useless as there is no way for the user to access it through the interface. Additionally, creating a function allows for quick maintenance and improvements, improving user satisfaction

4.3 Screenshots of Interface

The main menu allows users to access the future match prediction, team stats viewer and player stats viewer functions.

Future match prediction function:



Team Stats Viewer:

5. Testing and Evaluation

5.1 Testing Methods Used

To ensure that my project is flawless with no bugs, I used various testing methods throughout the development process. Below is a table with the testing methods used:

Testing Method	Use/Justification
User testing	User testing was used to check the frontend for any errors. Additionally, user testing allowed me to get verbal feedback on my UI, helping me improve its accessibility and function.
Unit testing	Unit testing was used to test small, separate pieces of my project so that they don't individually have errors. Additionally, it helps test the code quality and identify whether it outputs the expected behaviour. This was especially used in the feature engineering section, where each minor change to the dataset needed to be tested.
Desk Checking	Desk checking was used throughout the project regularly, as it is necessary to identify potential errors before execution. This was especially used at the end of every sprint, where I would go over and ensure that there were no errors with the code.

5.2 Test Cases and Results

Test ID	Description	Expected Result	Actual Result	Pass/Fail
TC01	Enter an Invalid Player Name	‘Invalid player selection. Please choose a current Premier League player’	‘Invalid player selection. Please choose a current Premier League player’	Pass
TC02	Enter an Invalid Team Name	Should not allow an invalid team name to be submitted	Does not allow an invalid team name to be submitted	Pass
TC03	Enter a Double Team Name	‘Please Select Two Different Teams’	‘Please Select Two Different Teams	Pass
TC04	Select No Statistic Columns	‘No valid columns selected’	‘No valid columns selected’	Pass
TC05	Test the individual player's stats function output	Player Stats in PD form	Player Stats in PD form	Pass
TC06	Test the individual team stats function output	Team stats in PD form	Team stats in PD form	Pass
TC07	Data prep, duplication, and empty value testing	No duplicate/ empty columns and rows	No duplicates, but there were rows with empty values, which were then removed	The issue was with the dataset and not the code, therefore Pass
TC08	Duplicate data testing when stacking the home and away team to find rolling	No duplicates, and the data aligns for all columns	No duplicates, and the data aligns for all columns	Pass

	averages_last10			
TC09	Home and Away's last 5 matches average testing	No duplicates and manual checks align with the actual average	No duplicates and manual checks align with the actual average	Pass
TC10	After stitching the files together, duplicate row and column testing	There are no duplicate rows or columns	There are no duplicate rows or columns	Pass

5.3 Evaluation Against Requirements

This project meets all the project requirements as it effectively meets all the functional and non-functional requirements stated before the beginning of the project. The code effectively creates predictions, views both team and player stats, fetches data, and has a highly accessible user interface while being efficient, maintainable, and accessible.

5.4 Improvements and Future Work

With a longer time frame, I would have loved to improve the model by providing it with dozens more features. Some features that could have been added were individual player statistics, team formation, and weather data. Player statistics would have allowed the model to improve its knowledge of the team's defence, midfield and attack. Additionally, implementing an advanced feature team formation could have allowed the model to identify which formations work better against others, allowing it to predict potential outlier matches between ‘strong’ and ‘weak’ teams. Furthermore, weather data could have improved the accuracy by factoring in more distinct data.

6. Client Feedback

6.1 Summary of Client Feedback

Harjas Singh (Client, Year 11)

"This is exactly the project that I have been needing. Aryan’s consistent communication and persistence have led to an application that meets all my expectations and requirements. What I liked most is how user-friendly it is. Even though there’s a machine learning model behind it, it doesn’t feel overly complicated or hard to use. The predictions are interesting to look at, especially alongside the stats. You can actually see the reasoning behind why the model predicts certain outcomes since you can see the probabilities for each team."

Jihan Juthani (Peer, Year 11)

"I was honestly impressed with how well everything came together. The predictions made sense, and I could see how this is predicting realistic possibilities. It’s also easy to follow — the way team and player stats are displayed makes it feel really usable without being too technical.”

6.2 Personal Reflection

I was extremely happy with this project, as I was able to develop an application that I was genuinely interested in and was able to play around with machine learning and data engineering using Python. I learnt significantly about machine learning and how it uses data to make predictions, while also discovering the vast possibilities and functions of feature engineering. Overall, I enjoyed creating this soccer data project, and it has been educational and rewarding towards my future in software.

7. Appendices

7.1 Full Gantt chart

The full Gantt chart can be accessed here: Software Major Project

7.2 Test Logs

The full test logs can be found here: Test Logs

7.3 Diagrams

The diagrams in PNG form are available here: Diagrams

7.4 Exemplar Code Snippets:

Testing and training of the model:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
model = RandomForestClassifier(n_estimators=300, random_state=42) # Initialises the Random
model.fit(X_train, Y_train) # Fits the model to the training data


Y_pred = model.predict(X_test) # Makes predictions on the test data
accuracy = accuracy_score(Y_test, Y_pred) # Calculates the accuracy of the model
print(f"Model accuracy: {accuracy:.2f}") # Prints the accuracy of the model
print(classification_report(Y_test, Y_pred, target_names=['HomeWin', 'Draw', 'AwayWin'])) #

joblib.dump(model, 'joblib/rf_model.joblib') # Saves the trained model to a file for use in
joblib.dump(X.columns, 'joblib/model_columns.joblib')
joblib.dump(sorted(set(data['team_x']).union(set(data['team_y']))), 'joblib/teams.joblib')
```

Merging of raw data from Soccerdata:

```
merge_df = union_all([df_2004,df_2005,df_2006, df_2007,df_2008, df_2009, df_2010,
                      df_2011,df_2012,df_2013,df_2014,df_2015,df_2016,df_2017,df_2018,
                      df_2019,df_2020,df_2021,df_2022,df_2023,df_2024])

merge_df.insert(0, 'match_id', range(1, len(merge_df) + 1)) #Adds match_id for later uses

merge_df['FTR'] = merge_df['FTR'].map({'H': 1, 'D': 2, 'A': 3}) #Maps the FTR column to numerical values

merge_df.dropna(inplace=True)  # Drops any rows with empty values in the merged data

merge_df.to_csv('data/merged_data2.csv',index=False)

print("Data Merged Successfully!")
```

Selection of relevant column names for player_stats:

```python
dfs = [df1, df2, df3, df4, df5, df6]
for i, d in enumerate(dfs):
    d.columns = [' '.join(col).strip() if isinstance(col, tuple) else col for col in d.columns] #Combines both rows of the hea

# Select essential columns from each DataFrame
df1_selected = df1[['player', 'team', 'Playing Time MP', 'Playing Time Starts', 'Playing Time Min', 'Performance Gls',
                    'Performance Ast', 'Performance PK', 'Performance CrdY', 'Performance CrdR']]

# Passing stats
df2_selected = df2[['player', 'team', 'Total Cmp', 'Total Cmp%', 'Short Cmp', 'Short Cmp%',
                    'Medium Cmp', 'Medium Cmp%', 'Long Cmp', 'Long Cmp%']]

# Shooting stats
df3_selected = df3[['player', 'team', 'Standard Sh', 'Standard SoT', 'Standard SoT%',
                    'Standard SoT/90', 'Standard G/Sh', 'Standard FK']]      You, yesterday • Uncommitted changes

# Defense stats
df4_selected = df4[['player', 'team', 'Tackles Tkl', 'Tackles TklW', 'Int', 'Blocks Sh', 'Clr']]

# Possession stats
df5_selected = df5[['player', 'team', 'Touches Touches', 'Touches Def 3rd', 'Touches Mid 3rd' , 'Touches Att 3rd',
                    'Carries Carries', 'Take-Ons Succ', 'Take-Ons Succ%']]

# Keeper stats (for goalkeepers only)
df6_selected = df6[['player', 'team', 'Performance Saves', 'Performance Save%', 'Penalty Kicks PKsv', 'Penalty Kicks Save%']]

# Merge all selected dataframes
df_combined = df1_selected.merge(df2_selected, on=['player', 'team'], how='outer', suffixes=('', '_pass'))
df_combined = df_combined.merge(df3_selected, on=['player', 'team'], how='outer')
df_combined = df_combined.merge(df4_selected, on=['player', 'team'], how='outer')
df_combined = df_combined.merge(df5_selected, on=['player', 'team'], how='outer')
df_combined = df_combined.merge(df6_selected, on=['player', 'team'], how='outer')

# Remove duplicate assists column
df_combined = df_combined.drop(columns=['assists_pass'], errors='ignore')
```

Feature engineering the past 10 rolling averages for the model data:

```
avg_last10_columns = [
    'Average_goals_scored_last10', 'Average_HT_goals_scored_last10',
    'Average_shots_last10', 'Average_shots_on_target_last10', 'Average_fouls_last10',
    'Average_corners_last10', 'Average_yellow_cards_last10', 'Average_red_cards_last10',
    'Average_B365_odds_last10', 'Average_B365_draw_odds_last10', 'Average_BW_odds_last10',
    'Average_BW_draw_odds_last10', 'Average_WH_odds_last10', 'Average_WH_draw_odds_last10',
    'Average_FullTimeResult_last10',
    'Average_goals_conceded_last10', 'Average_HT_goals_conceded_last10',
    'Average_shots_conceded_last10', 'Average_shots_on_target_conceded_last10',
    'Average_fouls_conceded_last10', 'Average_corners_conceded_last10',
    'Average_yellow_cards_conceded_last10', 'Average_red_cards_conceded_last10',
    'Average_B365_odds_against_last10', 'Average_BW_odds_against_last10',
    'Average_WH_odds_against_last10']

team_stats = team_stats.sort_values(['team', 'date']) # Sorts the team stats by team and date to ensure the last 10 matches a
#team_stats.to_csv('data/team_stats.csv', index=False) # Saves the sorted team stats to a CSV file

#Calclates the rolling average for the last 10 matches for each team by taking all the columns 4th and onward and
# calculating the rolling average for each team with a window of 10 matches, shifting by 1 to avoid including the current mat
rolling_average_last10 = (

team_stats
    .groupby('team')
    .apply(lambda group: group.iloc[:, 4:].shift(1).rolling(window=10, min_periods=1).mean())
    .reset_index(level=0, drop=True)

)


rolling_average_last10.columns = avg_last10_columns # Renames the columns to the average last 10 matches columns
rolling_average_last10.drop(columns=['Average_FullTimeResult_last10'], inplace=True) # Drops the FullTimeResult column as it
rolling_average_last10 = pd.concat([team_stats[['team', 'date', 'match_id' , 'home_away']], rolling_average_last10], axis=1)
rolling_average_last10.to_csv('data/rolling_average_last10.csv', index=False) # Saves the rolling average last 10 matches to
```

Combining all the separate tabs in the interface:

```
UI = gr.TabbedInterface(
    [prediction_interface, team_stats_interface, player_stats_interface],
    tab_names=["Match Outcome Prediction", "Team Seasonal Statistics" , "Player Statistics Viewer"],
    title="Premier League Match Prediction and Statistics Viewer",
)

UI.launch()
```

Functions of prediction and Stat Viewer:

```python
def predict(home_team, away_team): ···

def view_team_stats(team, selected_columns):      You, 23 hours ago • Completed Project minus saf

available_stat_columns = [col for col in team_stats.columns if col != 'team']

def view_player_stats(player_name, selected_columns): ···

available_player_columns = [col for col in player_stats.columns if col not in ['player', 'team']]
```

Merging of the feature-engineered columns/data:

```python
rolling_average_last5_away.columns = avg_last5_away_columns # Renames the columns to the average last 5 matches columns
rolling_average_last5_away.drop(columns=['Average_FullTimeResult_last5'], inplace=True) # Drops the FullTimeResult column as it is not needed for the rollin
rolling_average_last5_away = pd.concat([away_matches[['team', 'date', 'match_id']], rolling_average_last5_away], axis=1) # Combines the team and date column
#rolling_average_last5_away.to_csv('data/rolling_average_last5_away.csv', index=False) # Saves the rolling average last 5 away matches to a CSV file

# Renames the columns to include team_1_ for the away team for when we merge later
rolling_average_last5_home.columns = rolling_average_last5_home.columns[:3].tolist() + ['team_1_'+str(col) for col in rolling_average_last5_home.columns[3:]

# Renames the columns to include team_2_ for the away team for when we merge later
rolling_average_last5_away.columns = rolling_average_last5_away.columns[:3].tolist() + ['team_2_'+str(col) for col in rolling_average_last5_away.columns[3:]

# Merges the last 5 home and away matches dataframes on team, date and match_id
Last5_Home_Away = pd.merge( rolling_average_last5_home, rolling_average_last5_away, on=['match_id' , 'date'], how='left')

# Last5_Home_Away.to_csv('data/Last5_Home_Away.csv', index=False) # Saves the last 5 home and away matches to a CSV file

# Creates a copy of the rolling average last 10 matches for home teams so that we can put them side to side again with pd merge
home_team = rolling_average_last10[rolling_average_last10['home_away'] == 'H'].copy()
away_team = rolling_average_last10[rolling_average_last10['home_away'] == 'A'].copy()

# Drops the team and home_away columns from the home and away teams dataframes so that we can merge them later without doubling up
home_team = home_team.drop(['team', 'home_away'], axis=1)  # Keep only match_id, date, and stats
away_team = away_team.drop(['team', 'home_away'], axis=1)  # Keep only match_id, date, and stats

home_team.columns = home_team.columns[:3].tolist() + ['team_1_'+str(col) for col in home_team.columns[3:]] # Renames the columns to include team_1_ for the
away_team.columns = away_team.columns[:3].tolist() + ['team_2_'+str(col) for col in away_team.columns[3:]] # Renames the columns to include team_2_ for the

last10_stats = pd.merge(home_team, away_team, on=['match_id', 'date'], how='left') # Merges the last 10 home and away matches dataframes on team, date and m

last10_stats = last10_stats.dropna().reset_index(drop=True) # Drops any rows with empty values
Last5_Home_Away = Last5_Home_Away.dropna().reset_index(drop=True) # Drops any rows with empty values

# Merges the last 5 home and away matches with the last 10 home and away matches on team, date and match_id giving us our almost finished dataset
Combined_data = pd.merge(Last5_Home_Away, last10_stats, on=['match_id', 'date'], how='left')
Combined_data.to_csv('data/Combined_data.csv', index=False) # Saves the final data to a CSV file

# Merges the combined data with the original merged data to get the FullTimeResult column since it is our target variable
Final_data = pd.merge(Combined_data, merged_data2[['date', 'match_id', 'FTR']], on=['match_id', 'date'], how='left') # Merges the final data with the team s
```

Prediction Interface Tab:

```
prediction_interface = gr.Interface(
    fn=predict,  # Change to predict_simple if main function does
    inputs=[
        gr.Dropdown(teams, label="Home Team"),
        gr.Dropdown(teams, label=" Away Team")
    ],
    outputs=gr.Textbox(label="Prediction Result", lines=12),
    title="Premier League Match Outcome Predictor",
    description="Select two teams to predict the match outcome.",
    theme=gr.themes.Soft()
)
```