Sports Stats Collection by Voice Input

LM118 – Bachelor of Engineering in Electronic and Computer Engineering

Project Final Report

SEÁN MCTIERNAN

18224385

Prof. Hussain Mahdi

Monday 21st March 2022

Abstract

This aim of this project was to create a program which could collect sports statistics by voice input. Voice input is preferrable to pen and paper or touchscreen as the user does not have to look away from the game to input events which could lead to them missing subsequent events. Voice input is also more intuitive and requires less training time than touchscreen apps where the user must learn where the buttons for each function are.

Google Speech Recognition API deals with the speech recognition while Python code parses the text received from the API, updates dataframes and displays and runs the GUI.

The final program can collect sports statistics from voice input in real-time to a high degree of accuracy and can deal with all the possible events that can occur during a game.

Acknowledgements

Firstly, to my supervisor Prof. Hussain Mahdi. Your advice, support and time were invaluable to me.

Also, to friends, family and colleagues for both technical and moral support. I couldn’t have done it without you.

Declaration

This interim report is presented in part fulfilment of the requirements for the LM118 Bachelor of Engineering in Electronic and Computer Engineering **Bachelors Project**.

It is entirely my own work and has not been submitted to any other University or Higher Education Institution or for any other academic award within the University of Limerick.

Where there has been made use of work of other people it has been fully acknowledged and referenced.

Name	**Seán McTiernan**
Signature	Seán McTiernan
Date	**21st March 2022**

Table of Contents

List of Figures

Chapter 1 **Introduction**

1.1 Motivation

The right statistics in the right hands can be the difference between winning and losing a game. These statistics are currently recorded either with pen and paper or on a mobile or tablet app. The drawback of these systems is that the user must look away from the game to input an event, during which time they may miss a subsequent event. A system in which the user can input statistics by voice input overcomes this flaw in the other systems and has the benefit of being more intuitive.

1.2 Background and Rationale

The use of statistics in sports has increased dramatically in recent years, bolstered by the advent of mobile technology [1]. Originally statistics were taken with pen and paper, but recently applications have been developed to allow statistics to be recorded with the press of a button on a mobile device [2]. This also allows the statistics to be shared with management teams in real time over Bluetooth, Wi-Fi etc. allowing them to make decisions with up to date information.

The move from pen and paper to applications meant quicker input times with a button press versus a pen stroke. However, any time when the eyes must be taken away from the field of play means an opportunity for vital moments to be missed. For this reason, voice input can make a crucial difference in reliably recording any statistic one could wish to know while constantly keeping an eye on proceedings.

There are also companies who specialise in recording match data for teams and analysts after the games who could benefit from this [3]. Voice input is much more intuitive than buttons or a keypad where you must search for the correct buttons each time. Voice input also has faster input times as it is quicker to say three words e.g., “Goal Tipperary 15” than to find and press the buttons that signify “Goal”, “Tipperary” and “15” respectively.

The user should be able to speak commands into a microphone connected to a laptop or tablet. This would then be fed into a voice recognition engine to convert the command from speech into text. Then the text will be used to create or update data. The data also must be displayed in real-time, in an easy to read format. The user should be able to speak at a natural speed and input multiple commands in quick succession. They should also be able to undo commands easily and quickly to not interfere with their work. There will be a live feed page showing the commands as they are entered that the user can look at to ensure there are no mistakes.

The quality of the data display is crucial. It must be intuitive, easy to read, and the pages must be easy to navigate.

1.3 Project Aims and Objectives

The aim of this project is to design and develop a desktop application to allow users to record sports statistics via voice input. It will use the Google Speech Recognition API to convert from audio to text, and then Python to interpret the text, record the data and display the results.

Its main objectives are to allow users to record stats in real time without looking away from the game and to make recording stats after a game quicker and easier. It must allow users to undo commands quickly if they have made a mistake so that they don't miss the next statistic to be recorded. It is also important to display the data in a clear, easy to read, intuitive way to create a seamless user experience.

1.4 System Outline

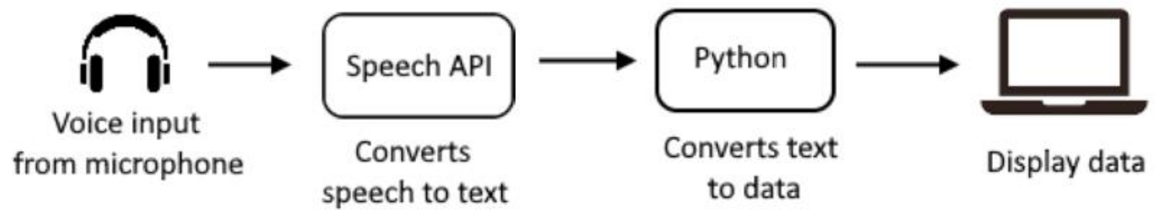


Figure 1 High-Level System Outline

The user will speak a command either in short form like "Tackle Tipperary 7", or as a full sentence e.g., "Number 7 for Tipperary made a tackle". The user can speak the key words in any order they like and surrounded by any filler words that are natural. There will also be a command to undo the previous command, make substitutions and input event with an unknown player, and buttons to change the current view and control timing events.

The Google Speech Recognition API will take in the audio from a microphone that the user is speaking into and convert it into text. The output text is then fed into Python which will interpret the text and find the key words. It will use these key words to update the relevant variables e.g., “Tackle Tipperary 7” will add 1 to the tackle count for the Tipperary team and to the personal count of number 7.

The data will be displayed in a dashboard with multiple pages. The main page to be shown during the game will show the team totals for all the stats being recorded. Key statistics will be highlighted on this page i.e., areas where your team is doing better than the opponent, and areas where the opponent is better. Another page will show individual player’s totals for each statistic and players performing above or below expectations can be highlighted. A sidebar will show the live feed of commands so the user can ensure they are correct.

After the game, data is stored to compare with future games. If player names have been input, then individual players’ performances can be compared from game to game. Each game can be exported in Excel format and screenshots of the heatmaps, and a copy of the match timeline are also saved.

1.5 Report Outline

This report is divided into the following six sections:- 3 -

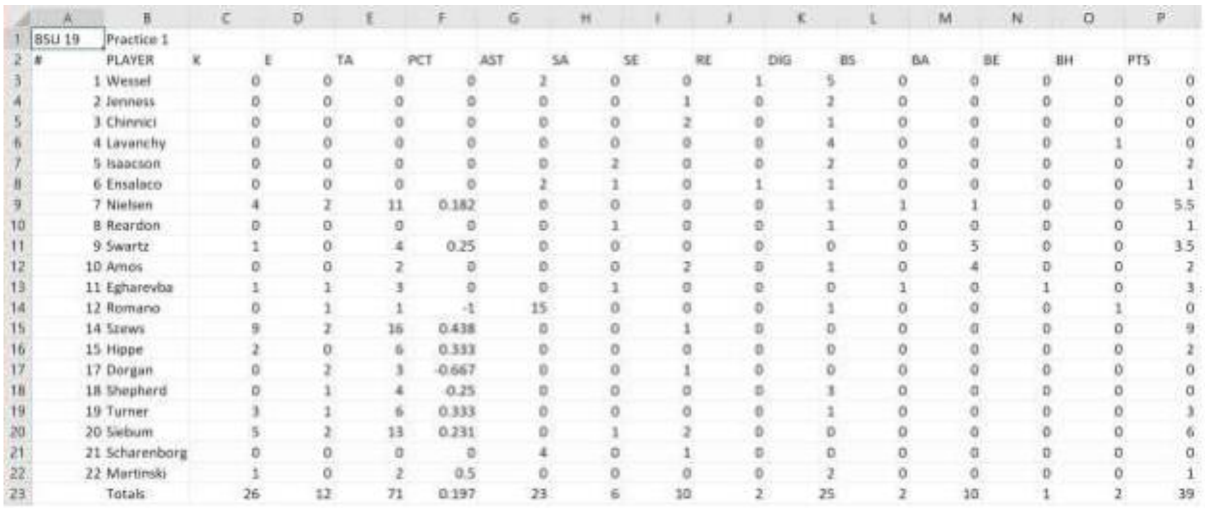
- Similar Systems: This section will discuss some similar systems that exist and analyses their strengths and weaknesses.
- Theoretical Background & Technologies: In this section, the technologies used throughout this project are discussed.
- System design: This section is a run through of the high-level system design detailing what each section of the program will do.
- System implementation: In this section, the implementation of the design is analysed in finer detail. Individual functions are explained.
- Testing: This section runs through each of the tests that were run
- Conclusion & Future Work: Final wrap-up of the project findings and the possibilities for future work.

Chapter 2 **Similar Systems**

There are very few applications in the field of voice input sports statistics, and none that are currently on the market.

2.1 PracStat

“PracStat” developed by Wessel [4] for use in volleyball has some of the same basic features as this project. It uses the Sphinx4 library for Java for speech-to-text conversion, then uses Java to process and display the data. “PracStat” takes voice input from the user, converts the audio to text and displays the data, but it lacks many features e.g., the ability to track statistics for both teams and to compare statistics from one game to another. The user interface is very basic, and the data display, as shown in Figure 1. is just a spreadsheet. It is not easy to read or navigate, and there is no option to highlight key statistics. “PracStat” is only available via flash drive from the creator.

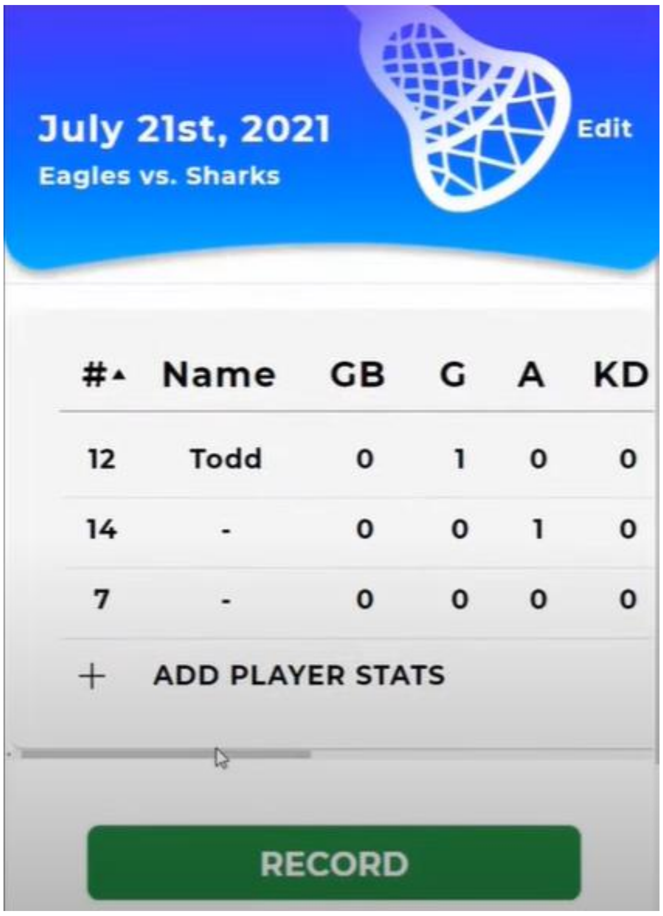


BSU 19	Practice 1														
#	PLAYER	K	E	TA	PCT	AST	SA	SE	RE	DIG	BS	BA	BE	BH	PTS
	1 Wessel	0	0	0	0	2	0	0	1	5	0	0	0	0	0
	2 Jenness	0	0	0	0	0	0	1	0	2	0	0	0	0	0
	3 Chinnici	0	0	0	0	0	0	2	0	1	0	0	0	0	0
	4 Lavanchy	0	0	0	0	0	0	0	0	4	0	0	0	1	0
	5 Isaacson	0	0	0	0	0	2	0	0	2	0	0	0	0	2
	6 Ensalaco	0	0	0	0	2	1	0	1	1	0	0	0	0	1
	7 Nielsen	4	2	11	0.182	0	0	0	0	1	1	1	0	0	5.5
	8 Reardon	0	0	0	0	0	1	0	0	1	0	0	0	0	1
	9 Swartz	1	0	4	0.25	0	0	0	0	0	0	5	0	0	3.5
	10 Amos	0	0	2	0	0	0	2	0	1	0	4	0	0	2
	11 Egharevba	1	1	3	0	0	1	0	0	0	1	0	1	0	3
	12 Romano	0	1	1	-1	15	0	0	0	1	0	0	0	1	0
	14 Szews	9	2	16	0.438	0	0	1	0	0	0	0	0	0	9
	15 Hippe	2	0	6	0.333	0	0	0	0	0	0	0	0	0	2
	17 Dorgan	0	2	3	-0.667	0	0	1	0	0	0	0	0	0	0
	18 Shepherd	0	1	4	-0.25	0	0	0	0	3	0	0	0	0	0
	19 Turner	3	1	6	0.333	0	0	0	0	1	0	0	0	0	3
	20 Siebum	5	2	13	0.231	0	1	2	0	0	0	0	0	0	6
	21 Scharenborg	0	0	0	0	4	0	1	0	0	0	0	0	0	0
	22 Martinski	1	0	2	0.5	0	0	0	0	2	0	0	0	0	1
	Totals	26	12	71	0.197	23	6	10	2	25	2	10	1	2	39

Figure 2 PracStat's data display page [4]

2.2 Baginski App

Todd Baginski developed a similar mobile app for use in lacrosse [5]. This app uses Microsoft Azure Cognitive Services for speech-to-text conversion and data processing. It has the same basic voice to statistics functionality but, again, the way in which the data is displayed is poor. The user must scroll to the side to see some of the data or down to see some of the players, and data can't be viewed and input at the same time which is vital to in-game use. Again, statistics can only be recorded for one team. This application was presented at Microsoft's Power Apps Demo Extravaganza 2021 but is not available to the public.


July 21st, 2021
Eagles vs. Sharks

#▲	Name	GB	G	A	KD
12	Todd	0	1	0	0
14	-	0	0	1	0
7	-	0	0	0	0

Figure 3 Baginski app display page [5]

Chapter 3 **Theoretical Background & Technologies**

4.1 Speech Recognition

Speech recognition is the ability of a program to convert human speech into text [6] . Speech recognition algorithms use an acoustic model, a language model and a dictionary to determine the appropriate output. The acoustic model is a mapping of audio signals to phonemes, which are the individual sounds that make up words. The dictionary is just a list of words with their pronunciations that the algorithm can match the incoming phonemes to. The language model contains phrases and how likely a certain word is to follow another word to help make sure the output is correct and makes sense.

4.2 Google Speech Recognition API

Google Speech Recognition API is a speech-to-text converter run by Google. It has a high accuracy and takes the context of the sentence into account when performing conversions so longer sentences are easier for it to understand. If it isn't sure whether a word is "call" or "goal", it will look at the rest of the sentence. If it sees "scored" elsewhere in the sentence, it will most likely go with "goal".

4.3 Python

Python is a clear and powerful object-oriented programming language, comparable to Perl, Ruby, Scheme, or Java [7]. It has strong data processing, plotting, GUI, string comparison, threading and image processing libraries. It is quick and easy to program in and to debug.

4.4 Visual Studio Code

VS Code is an editor with many useful tools such a built in debugger and terminal and version control. VS Code supports all programming languages and more including .txt, .xls and .md. It also has many useful extensions to help structure and layout your code.

Chapter 4 **System Design and Specification**

4.1 Setup File

First, the user will enter match details such as the match title, half length, team names, team nicknames and player names. They will also be able to select which statistics they wish to track from a list and will be able to add their own statistics to the list. The original idea was that this would be done in a page in the GUI, but it is better that this can be done well before the match and doing it with an excel sheet allows this. The user's settings can also be saved for future.

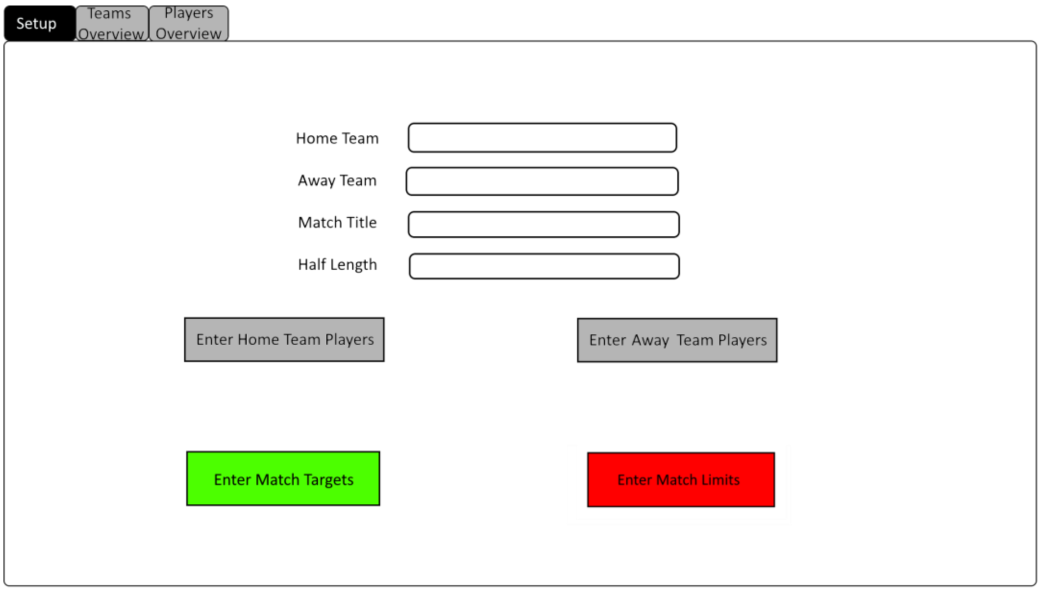
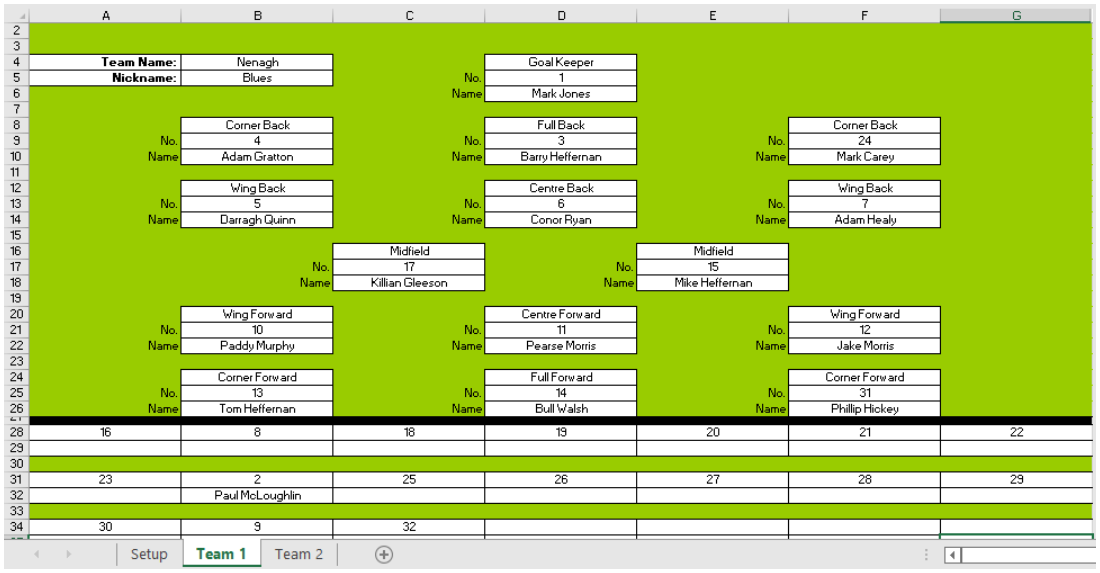


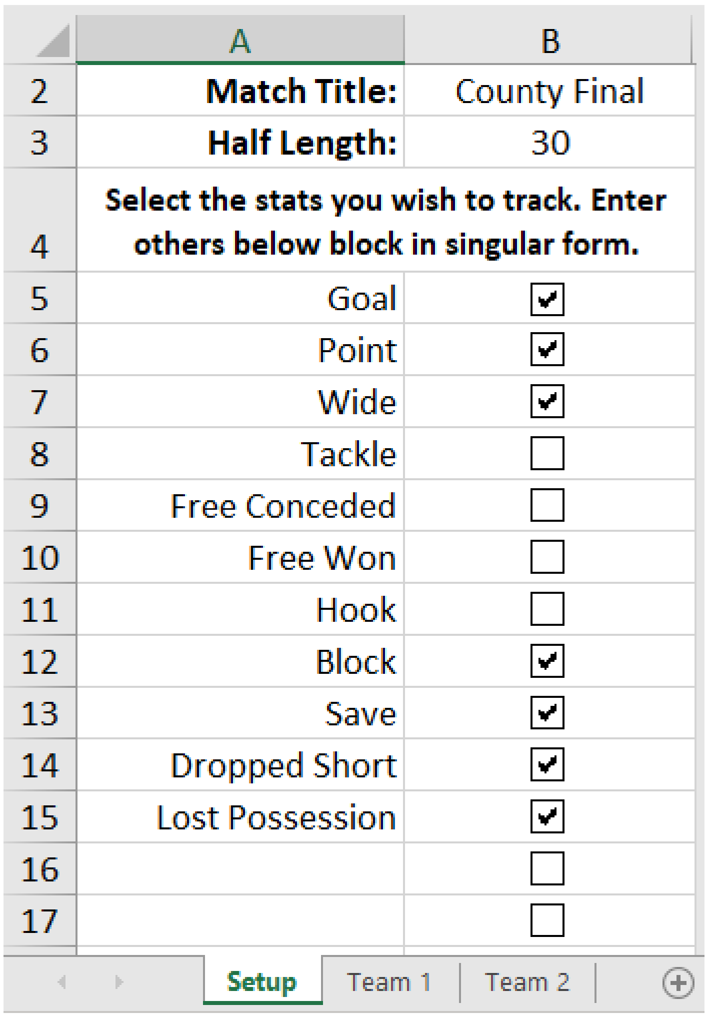
Figure 4 Setup Page Mock-up



	A	B	C	D	E	F	G
2							
3							
4	**Team Name:**	Nenagh		Goal Keeper			
5	**Nickname:**	Blues	No.	1			
6			Name	Mark Jones			
7							
8		Corner Back		Full Back		Corner Back	
9	No.	4	No.	3	No.	24	
10	Name	Adam Gratton	Name	Barry Heffernan	Name	Mark Carey	
11							
12		Wing Back		Centre Back		Wing Back	
13	No.	5	No.	6	No.	7	
14	Name	Darragh Quinn	Name	Conor Ryan	Name	Adam Healy	
15							
16			Midfield		Midfield		
17		No.	17	No.	15		
18		Name	Killian Gleeson	Name	Mike Heffernan		
19							
20		Wing Forward		Centre Forward		Wing Forward	
21	No.	10	No.	11	No.	12	
22	Name	Paddy Murphy	Name	Pearse Morris	Name	Jake Morris	
23							
24		Corner Forward		Full Forward		Corner Forward	
25	No.	13	No.	14	No.	31	
26	Name	Tom Heffernan	Name	Bull Walsh	Name	Phillip Hickey	
28	16	8	18	19	20	21	22
29							
30							
31	23	2	25	26	27	28	29
32		Paul McLoughlin					
33							
34	30	9	32				

Setup | Team 1 | Team 2

Figure 5 Team Input Page Final Version



	A	B
2	**Match Title:**	County Final
3	**Half Length:**	30
4	**Select the stats you wish to track. Enter others below block in singular form.**	
5	Goal	☑
6	Point	☑
7	Wide	☑
8	Tackle	☐
9	Free Conceded	☐
10	Free Won	☐
11	Hook	☐
12	Block	☑
13	Save	☑
14	Dropped Short	☑
15	Lost Possession	☑
16		☐
17		☐

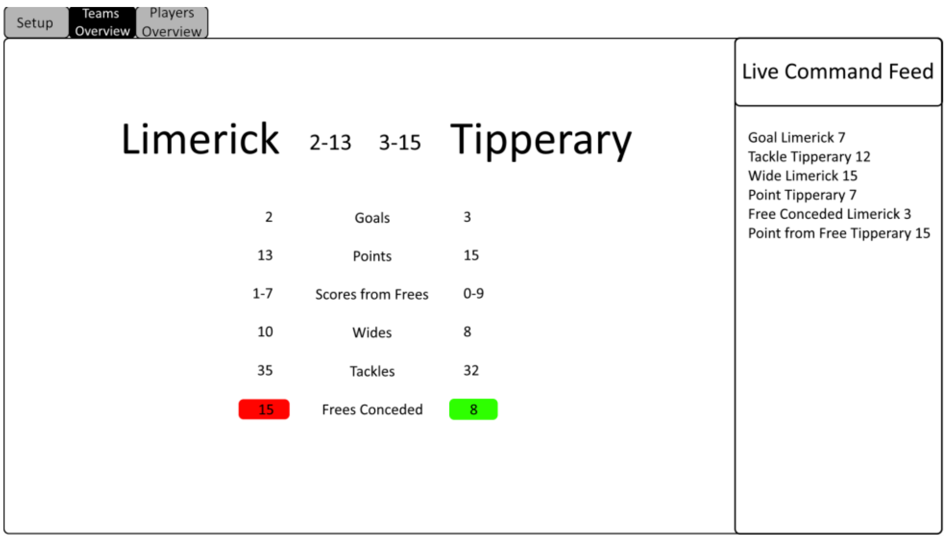
Figure 6 Setup Page Final Version

4.2 GUI

The GUI should have three pages, one for the team totals, and one for the player totals for each of the teams. These should be navigated between by buttons at the top of the screen, There will be other buttons here for exporting data and for signalling timing events like “Start Match” and “Half Time”.

4.2.1 Team Totals Page

This page shows the team totals for each statistic. Statistics in which there is a significant difference between the teams should be clearly shown.

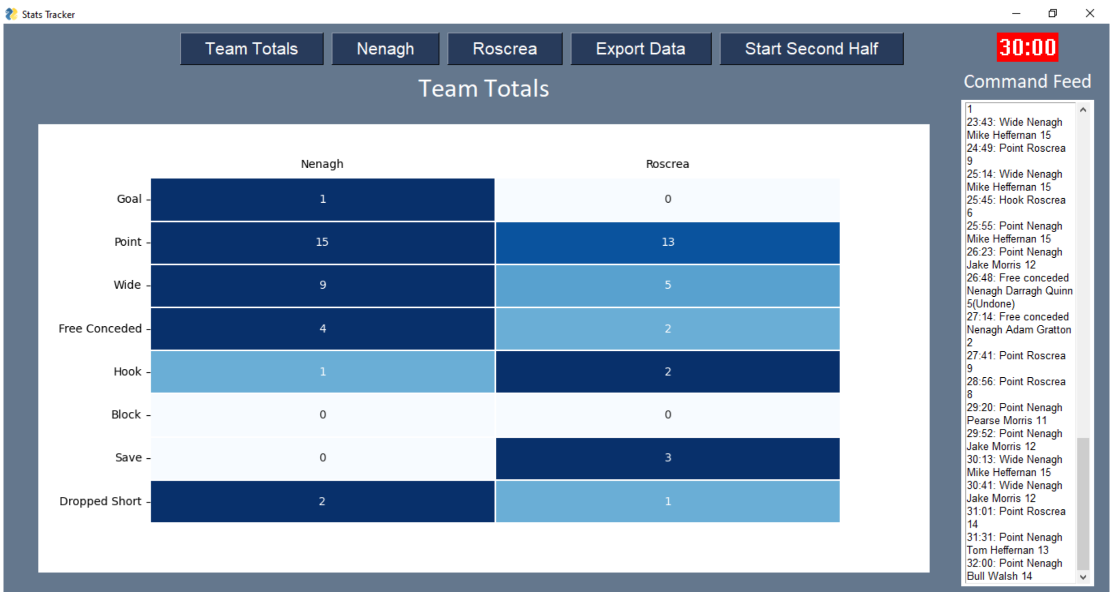


Limerick	2-13	3-15	Tipperary

2	Goals	3
13	Points	15
1-7	Scores from Frees	0-9
10	Wides	8
35	Tackles	32
🟥 15	Frees Conceded	🟩 8

Goal Limerick 7
Tackle Tipperary 12
Wide Limerick 15
Point Tipperary 7
Free Conceded Limerick 3
Point from Free Tipperary 15

Figure 7 Team Totals Page Mock-up



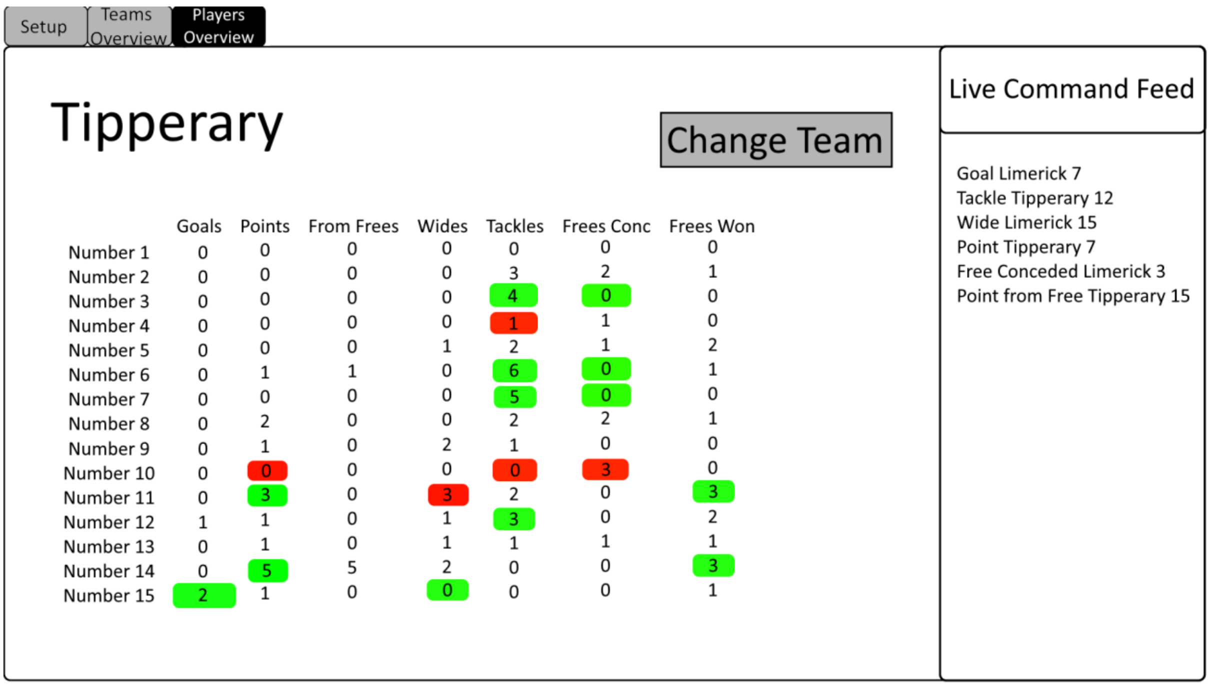
	Nenagh	Roscrea
Goal	1	0
Point	15	13
Wide	9	5
Free Conceded	4	2
Hook	1	2
Block	0	0
Save	0	3
Dropped Short	2	1

```
1
23:43: Wide Nenagh
Mike Heffernan 15
24:49: Point Roscrea
9
25:14: Wide Nenagh
Mike Heffernan 15
25:45: Hook Roscrea
6
25:55: Point Nenagh
Mike Heffernan 15
26:23: Point Nenagh
Jake Morris 12
26:48: Free conceded
Nenagh Darragh Quinn
5(Undone)
27:14: Free conceded
Nenagh Adam Gratton
2
27:41: Point Roscrea
9
28:56: Point Roscrea
8
29:20: Point Nenagh
Pearse Morris 11
29:52: Point Nenagh
Jake Morris 12
30:13: Wide Nenagh
Mike Heffernan 15
30:41: Wide Nenagh
Jake Morris 12
31:01: Point Roscrea
14
31:31: Point Nenagh
Tom Heffernan 13
32:00: Point Nenagh
Bull Walsh 14
```

Figure 8 Team Totals Page Final Version

4.2.2 Single Team Page

This page will show each individual players' totals for each statistic. Again, outliers in the data should be clearly visible.



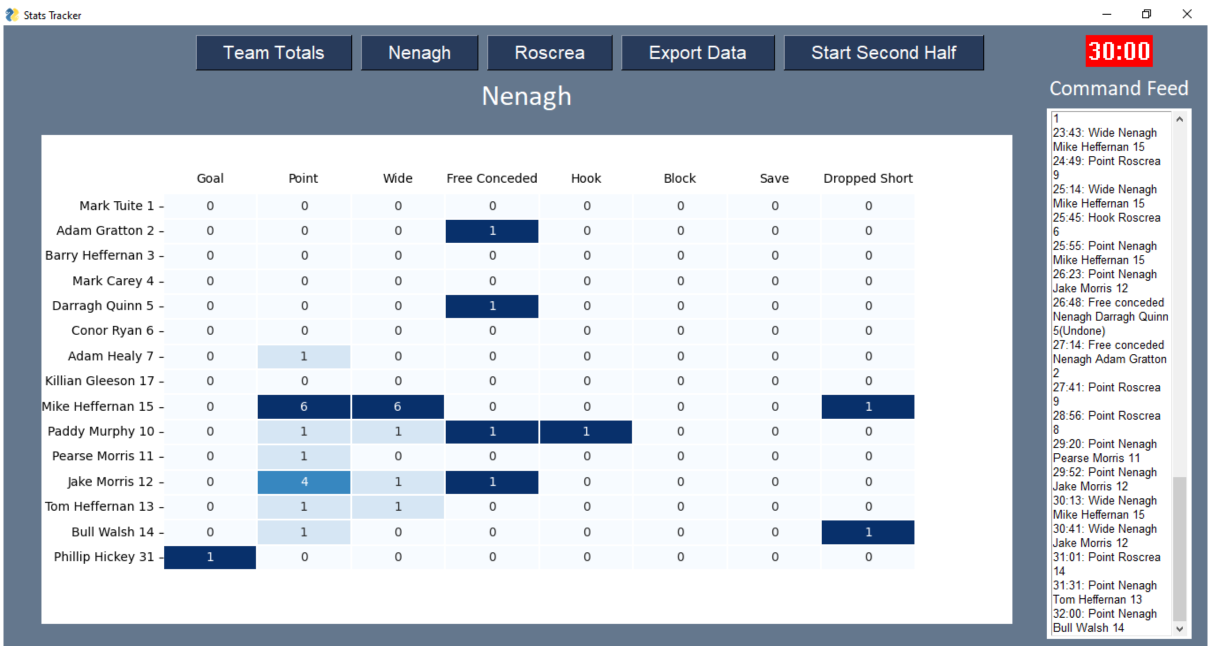
Tipperary

	Goals	Points	From Frees	Wides	Tackles	Frees Conc	Frees Won
Number 1	0	0	0	0	0	0	0
Number 2	0	0	0	0	3	2	1
Number 3	0	0	0	0	4	0	0
Number 4	0	0	0	0	1	1	0
Number 5	0	0	0	1	2	1	2
Number 6	0	1	1	0	6	0	1
Number 7	0	0	0	0	5	0	0
Number 8	0	2	0	0	2	2	1
Number 9	0	1	0	2	1	0	0
Number 10	0	0	0	0	0	3	0
Number 11	0	3	0	3	2	0	3
Number 12	1	1	0	1	3	0	2
Number 13	0	1	0	1	1	1	1
Number 14	0	5	5	2	0	0	3
Number 15	2	1	0	0	0	0	1

Live Command Feed

Goal Limerick 7
Tackle Tipperary 12
Wide Limerick 15
Point Tipperary 7
Free Conceded Limerick 3
Point from Free Tipperary 15

Figure 9 Single Team Page Mock-up



Nenagh

	Goal	Point	Wide	Free Conceded	Hook	Block	Save	Dropped Short
Mark Tuite 1 -	0	0	0	0	0	0	0	0
Adam Gratton 2 -	0	0	0	1	0	0	0	0
Barry Heffernan 3 -	0	0	0	0	0	0	0	0
Mark Carey 4 -	0	0	0	0	0	0	0	0
Darragh Quinn 5 -	0	0	0	1	0	0	0	0
Conor Ryan 6 -	0	0	0	0	0	0	0	0
Adam Healy 7 -	0	1	0	0	0	0	0	0
Killian Gleeson 17 -	0	0	0	0	0	0	0	0
Mike Heffernan 15 -	0	6	6	0	0	0	0	1
Paddy Murphy 10 -	0	1	1	1	1	0	0	0
Pearse Morris 11 -	0	1	0	0	0	0	0	0
Jake Morris 12 -	0	4	1	1	0	0	0	0
Tom Heffernan 13 -	0	1	1	0	0	0	0	0
Bull Walsh 14 -	0	1	0	0	0	0	0	1
Phillip Hickey 31 -	1	0	0	0	0	0	0	0

Command Feed

```
1
23:43: Wide Nenagh
Mike Heffernan 15
24:49: Point Roscrea
9
25:14: Wide Nenagh
Mike Heffernan 15
25:45: Hook Roscrea
6
25:55: Point Nenagh
Mike Heffernan 15
26:23: Point Nenagh
Jake Morris 12
26:48: Free conceded
Nenagh Darragh Quinn
5(Undone)
27:14: Free conceded
Nenagh Adam Gratton
2
27:41: Point Roscrea
9
28:56: Point Roscrea
8
29:20: Point Nenagh
Pearse Morris 11
29:52: Point Nenagh
Jake Morris 12
30:13: Wide Nenagh
Mike Heffernan 15
30:41: Wide Nenagh
Jake Morris 12
31:01: Point Roscrea
14
31:31: Point Nenagh
Tom Heffernan 13
32:00: Point Nenagh
Bull Walsh 14
```

Figure 10 Single Team Page Final Version

4.2.3 Command Feed

The command feed on the right side of the screen shows the list of commands already entered. Here the user can ensure that they have been understood correctly. If a command is undone, it is removed from the command feed.

4.3 Speech-to-Text

The program will begin recording audio when the spacebar is first pressed and end recording when it is pressed again. It will save this recording to a .wav file and then send that file to the Speech Recognition API to be converted to text.

4.4 Text Parsing

The incoming text will be checked against the list of event names, team names and player names to find either an exact match or a very close match.

4.5 Data Processing

The data will be held in Pandas dataframes. They will be set up in such a way that the event names, team names, and player names will be the indices required to access a certain value.

4.6 Data Visualisation

The data will be shown in Seaborn heatmaps. That is a table in which the background shading of a cell increases in colour with a higher cell value. This allows extreme values to be seen easily.

4.7 Exporting

The user should be able to save the data they collect to be used later for various purposes or to send to other members of the management team. The data should be exported as an excel file to allow the user access to the individual data points should they want them. The heatmaps will also be exported as images which can be useful to send to the rest of the management team during the game. A match timeline is also saved. Managers often want a compilation of, for example, the videos of every free the team gave away in the last game. The match timeline can be used to find these events in the match video very easily.

4.8 Possible Commands

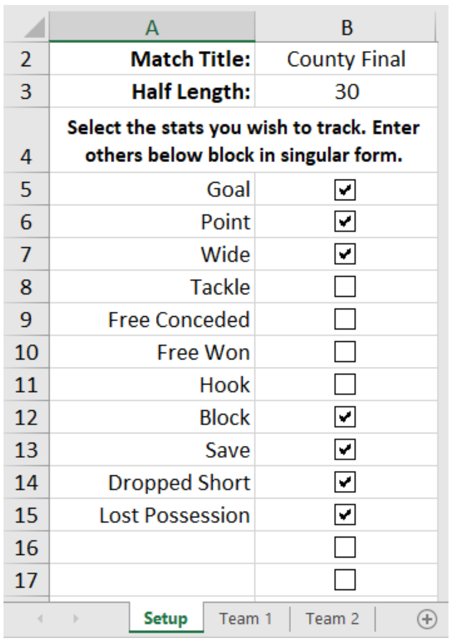
The user can choose any combination of events in the setup page, and they can also add their own to the list. This means that any event they want to track which is of the form "Player X performed action Y" can be done. Other commands which are not of this form include:

- Substitution: User must say "Sub" or "Substitution" with the team's name and two players' names.
- Undo: User must say "Undo" and the previous command is undone.
- Unknown Player: If the player who performed an action is not known, the user can simply say "Unknown" in place of the player's name and the action will not be credited to any specific player, but the team total will still be updated.
- Player Name Only: If referring to a player by their name, there is no need to say the team's name. The exception to this is if both teams have a player with the same name.
- Position: Each team's current goalkeeper can be referred to as "Goalkeeper", "Keeper" or "Goalie". The goalkeeper is the only position that would be referred to in this way.

Chapter 5 **System Implementation**

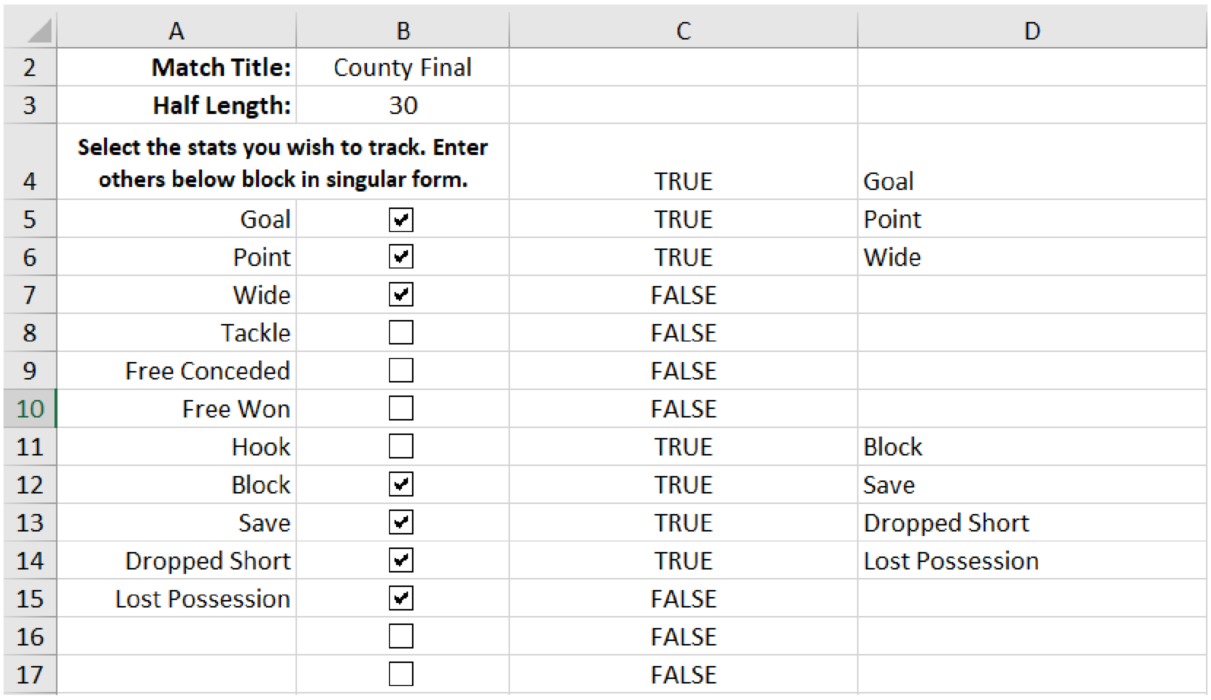
6.1 Setup File

The first thing the user does is input the match details into the "Setup" tab in the setup.xls file.



	A	B
2	**Match Title:**	County Final
3	**Half Length:**	30
4	**Select the stats you wish to track. Enter others below block in singular form.**	
5	Goal	✓
6	Point	✓
7	Wide	✓
8	Tackle	☐
9	Free Conceded	☐
10	Free Won	☐
11	Hook	☐
12	Block	✓
13	Save	✓
14	Dropped Short	✓
15	Lost Possession	✓
16		☐
17		☐

Figure 11 Setup Page User View



	A	B	C	D
2	**Match Title:**	County Final		
3	**Half Length:**	30		
4	**Select the stats you wish to track. Enter others below block in singular form.**		TRUE	Goal
5	Goal	✓	TRUE	Point
6	Point	✓	TRUE	Wide
7	Wide	✓	FALSE	
8	Tackle	☐	FALSE	
9	Free Conceded	☐	FALSE	
10	Free Won	☐	FALSE	
11	Hook	☐	TRUE	Block
12	Block	✓	TRUE	Save
13	Save	✓	TRUE	Dropped Short
14	Dropped Short	✓	TRUE	Lost Possession
15	Lost Possession	✓	FALSE	
16		☐	FALSE	
17		☐	FALSE	

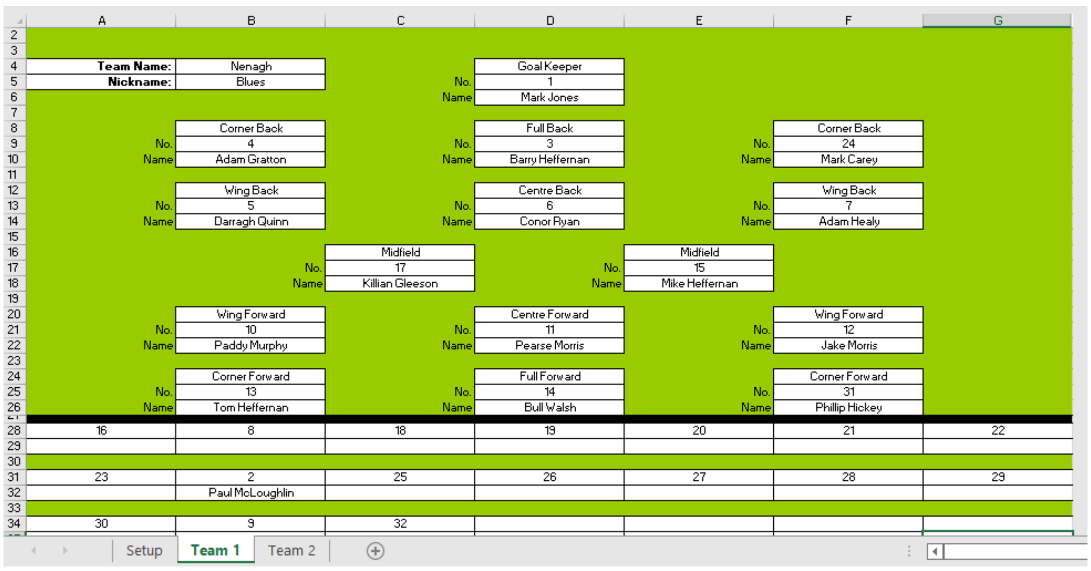
Figure 12 Setup Page Backend

Here, they enter the match title and half length, and can select which stats they wish to track for the upcoming match by ticking the checkboxes in the right column. The user can type any other statistics they wish to track that aren't in the list into the left column.

In the hidden columns, C is true if the checkbox is ticked, and D = A if C is true. Then the python file can just read the event names from the D column.

Next, the user goes to the "Team 1" tab to input the names and numbers of the players who will be starting the match. Here they also input the name of the team and the nickname if it applies, either of which can be used to refer to the team when speaking commands.

This is repeated in the "Team 2" tab and then this file gets saved and can be closed.



	A	B	C	D	E	F	G
2							
3							
4	**Team Name:**	Nenagh		Goal Keeper			
5	**Nickname:**	Blues	No.	1			
6			Name	Mark Jones			
7							
8		Corner Back		Full Back		Corner Back	
9	No.	4	No.	3	No.	24	
10	Name	Adam Gratton	Name	Barry Heffernan	Name	Mark Carey	
11							
12		Wing Back		Centre Back		Wing Back	
13	No.	5	No.	6	No.	7	
14	Name	Darragh Quinn	Name	Conor Ryan	Name	Adam Healy	
15							
16			Midfield		Midfield		
17		No.	17	No.	15		
18		Name	Killian Gleeson	Name	Mike Heffernan		
19							
20		Wing Forward		Centre Forward		Wing Forward	
21	No.	10	No.	11	No.	12	
22	Name	Paddy Murphy	Name	Pearse Morris	Name	Jake Morris	
23							
24		Corner Forward		Full Forward		Corner Forward	
25	No.	13	No.	14	No.	31	
26	Name	Tom Heffernan	Name	Bull Walsh	Name	Phillip Hickey	
28	16	8	18	19	20	21	22
29							
30							
31	23	2	25	26	27	28	29
32		Paul McLoughlin					
33							
34	30	9	32				

Figure 13 Team Input Tab

6.2 Python Files

6.2.1 Input.py

The first job is to collect the information from the setup.xls file. This is done using functions in the input.py file. The get_info() function uses Pandas to read the excel files, retrieves player names and numbers, match details and event names and sets the global variables with the read values. It also removes any NaN values from the numbers list and if a player's name is Null it is set to an empty string.

```python
def get_info():
    setup = pd.read_excel('setup.xls', sheet_name="Setup")  # Read sheet data
    team_0 = pd.read_excel('setup.xls', sheet_name="Team 1")
    team_1 = pd.read_excel('setup.xls', sheet_name="Team 2")

    globals.match_title = setup[1][0]       # Set Match Title
    globals.half_length = int(setup[1][1])  # Set Half Length
    globals.team_names[0] = team_0[1][2]    # Set Team 0 name
    globals.team_names[1] = team_1[1][2]    # Set Team 1 name

    # Retrieve player names
    team0_players, team1_players = get_players(team_0, team_1)
    # Retrieve player numbers
    team0_numbers, team1_numbers = get_numbers(team_0, team_1)

    # Remove blanks from player names
    team0_players, team1_players, team0_numbers, team1_numbers =
     remove_blanks(team0_players, team1_players, team0_numbers, team1_numbers)

    globals.event_names = get_events(setup) # Retrieve event names

    # Sets players with no name to have name ""
    team0_players, team1_players = no_name(team0_players, team1_players)

    globals.team0_players = team0_players # Set Team 0 names
    globals.team1_players = team1_players # Set Team 1 names

    # Set Team 0 numbers
    globals.team0_numbers = [int(number) for number in team0_numbers]
    # Set Team 1 numbers
    globals.team1_numbers = [int(number) for number in team1_numbers]
```

The get_numbers() function has the list of cells where the player numbers will be and collects and returns those values. Get_players() works in the same way.

```
def get_numbers(team_0, team_1):
    """Retrieve the player numbers from the cells in the excel sheet"""
    team_0 = [team_0[3][3], team_0[1][7], team_0[3][7] …………
    team_1 = [team_1[3][3], team_1[1][7], team_1[3][7] …………

    return team_0, team_1
```

Get_events() returns the list of values in the D columns of the setup sheet which are the event names.

```
def get_events(setup):
    """Retrieve the event names from the cells in the excel sheet"""
    return [event for event in setup[3].values if type(event) == str]
```

No_name turns NaN player name values into empty strings. It iterates through the players lists and checks if the value is NaN. A TypeError is thrown if the value is a string so that can be ignored.

```
def no_name(team0_players, team1_players):
    """If a player has a number but no name, set name to """""
    for i, _ in enumerate(team0_players):
        with contextlib.suppress(TypeError):
            if math.isnan(team0_players[i]):
                team0_players[i] = ""
    for i, _ in enumerate(team1_players):
        with contextlib.suppress(TypeError):
            if math.isnan(team1_players[i]):
                team1_players[i] = ""
    return team0_players, team1_players
```

Remove_blanks() remove items from the players and numbers lists if both corresponding entries are Null. It checks if the name is empty or NaN and if the number is NaN and pops the entries from the lists if both are true.

```python
def remove_blanks(team0_players, team1_players, team0_numbers, team1_numbers):
    """Remove blanks values from the player names and numbers"""
    for i in reversed(range(len(team0_players))):
        if team0_players[i] in ['', 'nan'] and math.isnan(team0_numbers[i]):
            team0_numbers.pop(i)
            team0_players.pop(i)
        if team1_players[i] in ['', 'nan'] and math.isnan(team1_numbers[i]):
            team1_numbers.pop(i)
            team1_players.pop(i)
    return team0_players, team1_players, team0_numbers, team1_numbers
```

6.2.2 Audio.py

The first job of this file is to remove old voice recordings. The delete_log_files() function searches the “audio_files” folder for files and deletes them if they exist.

```python
def delete_log_files(): # delete all audio files
    files = [f for f in listdir('audio_files') if isfile(join('audio_files',
                                                               f))]
    for file in files:
        remove(f'audio_files/{file}')
```

The audio_recording() function in this file runs constantly in its own thread to record the users voice inputs. When the spacebar is pressed to start a new recording, the GUI can be in another process so the recorder must take the previous few seconds of audio from before it gets the recording flag set, and it continues recording until the spacebar is pressed again. It then resets the recording flags, saves the audio to a file in the “audio_files” folder and starts a thread to process that file. It then goes back to waiting for the spacebar to be pressed signalling a new recording.

```python
def audio_recording():
    frames = []
    i=0
    while True:
        stream = pyaudio.PyAudio().open(format=pyaudio.paInt16, channels=2,
                                   rate=44100, input=True, frames_per_buffer=3024)
        while not globals.start_recording:
            data = stream.read(chunk*50)
            frames = [data]
        while not globals.end_recording: # when recording flag set
            data = stream.read(chunk)
            frames.append(data)
        globals.end_recording = False # reset flags
        globals.start_recording = False

        stream.close()

        ## Save file
        file = f'audio_files/phrase{i}.wav'

        wf = wave.open(file, 'wb')
        wf.setnchannels(channels)
        wf.setsampwidth(audio.get_sample_size(pyaudio.paInt16))
        wf.setframerate(rate)
        wf.writeframes(b''.join(frames))
        wf.close()

        z = threading.Thread(target=process_audio, args=(file,), daemon=True)
        z.start()
        i += 1
```

The process_audio() function is run from audio_recording() in a separate thread so that recordings can be continuous. It uses the speech_recognition library to do speech to text conversion on the audio files. This library returns its top five guesses as to what was said, and these are all passed to the parse() function in parse.py. This function returns the event, team and player which are then passed on the other functions depending on if the event is a substitution, an undo or something else. The parse() function returns None for each value if it can't understand the text from the recognizer and the system plays a sound to notify the user of this.

```python
def process_audio(path): # pass audio file to recognizer, pass text to parser,
pass commands to data updater
    r = sr.Recognizer()
    audio_file = sr.AudioFile(path)

    with audio_file as source:
        r.adjust_for_ambient_noise(source)
        audio = r.record(source)

    text = r.recognize_google(audio, show_all=True)

    event, team, player = parse(text)

    if "Undo" in [event, team, player]:
        dp.undo_command()
        return

    if event == "Sub":
        dp.sub_command(team, player)
        return

    if None in [event, team, player]:
        winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS)
        remove(path)
        return

    dp.update_data(event, team, player)
```

6.2.3 Parse.py

This file contains functions to read the event, team and player from the text from the speech recognizer.

First it splits the predictions into a list. It removes filler words like "a", "by", "scored" etc. It then checks if the incoming text contains the undo command, and if so, performs the necessary action to the command feed and returns "Undo" which will be used to run an undo_command() function. It then checks if the event is a substitution and if so, returns the necessary details for a substitution. Next it searches for a team name. If none is found, it searches for a player name without a team name which is a valid input. If both checks are failed, None is returned. Then it checks for an event name, returning None if not found. Last check is for a player name if it hasn't already been found. Last steps are to write the command to the command feed, add it to the command list and return event, team, player.

```
def parse(text):
    try: # Split predictions into a list
        text = [phrase['transcript'] for phrase in text['alternative']]
    except:
        return None, None, None

    text = clean_input(text) # remove filler words

    if check_undo(text): # Check if undo command entered
        write(undo=True)
        return "Undo", "Undo", "Undo"

    if check_sub(text):
        return sub(text)

    team, text = check_team(text)
    if team is None:
        player, team, text = player_check_without_team(text)
    if team is None:
        return None, None, None

    event, text = check_event(text)
    if event is None:
        return None, None, None

    if 'player' not in locals():
        player, text = check_player(text, team)
    if player is None:
        return None, None, None

    write(f'{event.capitalize()} {team.capitalize()} {player}')

    globals.commands.append([event, team, player])

    return event, team, player
```

The check_undo runs fuzzy() on “Undo” and “Undone”. Fuzzy() compares the strings and returns true if there is a greater than 60% match to any of the input text. Check_sub() works the same way.

```
def check_undo(text): # This function checks if the event is "Undo"
    words = ['undo', 'undone']
    if fuzzy(words, text, True) in words:
        return True
```

The check_team() function searches the input text for the team names and nicknames. It uses the fuzzy() function from before and also exact_check() which first checks through the text to see if there is an exact match. If the team is found, fuzzy_remove() removes the name and very similar words from the input text. If the team was found by nickname, return the team's proper name, otherwise just return the team's name. Check_event() works the same as this but simpler as it doesn't have to worry about nicknames.

```
def check_team(text):
    team = exact_check(globals.team_names, text)
    if team is None:
        team = fuzzy(globals.team_names.copy(), text, strict=True)
        if team is None:
            team = exact_check(globals.team_nicknames, text)
            if team is None:
                team = fuzzy(globals.team_nicknames.copy(), text, strict=True)
                if team is None:
                    return None, text

    text = fuzzy_remove(text, team) # Remove team name from input
    if team in globals.team_nicknames:
        team = globals.team_names[globals.team_nicknames.index(team)]
    return team, text
```

Check_player() works in the same way just with some small extras. The player is searched for by their name and by their number in both integer and string form. It must call combine_name_number() to combine the said name with its number or vice versa before returning the player. It must also check if the player is referred to as "Goalkeeper" or some variant of it. This is the only position that might be referred to in such a way.

```
if check_keeper(text):
        player = combine_name_number(team, "1")
        text = fuzzy_remove(text, ["keeper", "goalie", "goalkeeper"])
        return player, text
```

The user can also say that the player who performed an action is "unknown" so this must be checked for as well.

```
if unknown(text):
        return ("Unknown", text)
```

The write() function writes each command and the current match time to the command log text file. It also removes commands when from this file when they are undone. This file is read to create the command feed and also serves as a timeline after the match.

```python
def write(command="", undo=False): # Write command to log
    if not undo:
        with open(f"text_files/{globals.match_title}_command_log.txt","a+") as
                                                                          f:
            with contextlib.suppress(TypeError):
                if globals.commands == []:
                    f.write(f'{globals.window["-TIME-"].DisplayText}:
                                                              {command}\n')
                else:
                    f.write(f'\n{globals.window["-TIME-"].DisplayText}:
                                                              {command}\n')
    else:
        with open(f"text_files/{globals.match_title}_command_log.txt", 'r+')
                                                                       as f:
            lines = f.readlines()
            f.seek(0)
            f.truncate()
            f.writelines(lines[:-2])
```

6.2.4 Fuzzy.py

This file contains functions to compare strings and find both exact matches and close matches.

Fuzzy() is first called with a list of choices e.g., event names, team names, the input text, and how strict the comparator should be. First it cleans the choices, removes empty strings and duplicates. Next it creates list of pairs and triples that are beside each other in the input text. These are used if comparing with a choice that is two or three words long. Then it splits the input into a list of single words. It then checks each of the choices against either the single words list or the pairs or triples. The results arrays contain a count of each time a choice had a comparison ratio over 80, 60 or 49 %. If there is a single choice with the most over 80% comparisons it is returned. 60% comparisons are used if there is no clear winner at 80% and then 49% if none at 60. If there is still no clear winner, None is returned.

```
def fuzzy(choices, text, strict=False):
    choices = clean_choices(choices)

    pairs, triples = create_pairs_triples(text)

    input_text = format_input(text)

    results_80, results_60, results_49 = perform_fuzzy(choices, input_text,
                                                       strict, pairs, triples)

    try:
        if results_80.count(max(results_80)) == 1:
            return choices[results_80.index(max(results_80))]
        elif results_60.count(max(results_60)) == 1:
            return choices[results_60.index(max(results_60))]
        elif results_49.count(max(results_49)) == 1:
            return choices[results_49.index(max(results_49))]
        else:
            return None
    except ValueError:
        return None
```

Create_pairs_triples() creates the pairs and triples lists by call word_groups and then flattens the nested lists that are returned.

```
def create_pairs_triples(text):
    pairs = [word_groups(item, 2) for item in text]
    pairs = [item for sublist in pairs for item in sublist]
    triples = [word_groups(item, 3) for item in text]
    triples = [item for sublist in triples for item in sublist]
    return pairs, triples
```

Word_groups() splits the given words into combinations of length “number”. It ignores any combinations that contain empty strings or strings that only contain spaces.

```python
def word_groups(words, number):
    groups = []
    for j in range(len(words.split())-(number-1)):
        group = ""
        for i in range(number):
            if words.split()[j + i] != " ":
                group += words.split()[j+i]
                if i < number-1 and group != '':
                    group += " "
            else:
                break
        if group not in ['', ' ', '  ']:
            groups.append(group)
    return groups
```

Perform_fuzzy() finds the ratios between each of the choices and the input text and it returns the arrays holding how many times each choice got above a certain score.

```python
def perform_fuzzy(choices, input_text, strict, pairs, triples):
    results_80 = [0]*len(choices)
    results_60 = [0]*len(choices)
    results_49 = [0]*len(choices)
    for i, choice in enumerate(choices):
        if len(str(choice).split(" ")) == 1: # If single word choice
            words = input_text
        elif len(str(choice).split(" ")) == 2: # If 2 word choice
            words = pairs
        elif len(str(choice).split(" ")) == 3: # If 3 word choice
            words = triples
        for word in words: # Iterate through input words/pairs/triples
            # Get comparison score
            ratio = fuzz.ratio(word.lower(), str(choice).lower())
            if ratio > 80: # If > 80%, add 1 to 80% list
                results_80[i] += 1
            if not strict: # If strict, only scores above 80% are counted
                if ratio > 60: # If > 60%, add 1 to 60% list
                    results_60[i] += 1
                elif ratio > 49: # If > 49%, add 1 to 49% list
                    results_49[i] += 1
    return results_80, results_60, results_49
```

Fuzzy_remove() is used to remove words from the input text that closely resemble the input "result". It will iterate through the list of either single words, pairs or triples depending on the length of "result". If the comparison ratio is greater than 60%, that words or combination is removed from the input text. If "result" is a list, fuzzy_remove is run recursively over each item in the list.

```
def fuzzy_remove(text, result):
    if type(result) == list: # Recursion
        for item in result:
            text = fuzzy_remove(text, item)
        return text
    result = str(result)

    if len(result.split(" ")) == 1:
        for i, phrase in enumerate(text):
            words = phrase.split(" ")
            for word in words:
                if fuzz.ratio(word.lower(), result.lower()) > 60:
                    text[i] = phrase.replace(f" {word}", '')
                    text[i] = text[i].replace(word, '')

    elif len(result.split(" ")) == 2:
        for i, phrase in enumerate(text):
            pairs = word_groups(phrase, 2)
            for pair in pairs:
                if fuzz.ratio(pair.lower(), result.lower()) > 60:
                    text[i] = text[i].replace(f" {pair}", "")
                    text[i] = text[i].replace(pair, "")
    # Repeat for triples
    return text
```

Exact_check() runs through each choice in the list of choices and returns it if it is in the input text.

```
def exact_check(choices, text):
    choices = list(filter(None, choices)) # Remove empty strings
    # Loop through choices and phrases
    for phrase, choice in itertools.product(text, choices):
        if type(choice) != int:
            if choice.lower() in [word.lower() for word in word_groups(phrase,
                                                       len(choice.split()))]:
                return choice
        elif str(choice) in phrase.split():
            return choice
    return None
```

6.2.5 Data_processing.py

This file has functions that deal with the creation and updating of dataframes as well as plotting the data in heatmap tables.

Pandas is a library that is used for creating dataframes. It takes an array as an input, so arrays of zeros are created using NumPy and used to create dataframes for the team totals, team 1 players and team 2 players. The columns names on the team totals dataframe are the team names and the row indices are the event names. For the individual team dataframes, these are flipped to create a better shape for display, so the column names are the event names, and the row indices are a combination of the players' names and their numbers.

```python
def create_dataframe():
    ## Create arrays of zeroes
    teams_data = np.zeros((len(globals.event_names), 2))
    teams_data = np.array(teams_data,dtype=int)

    team0_data =
              np.zeros((len(globals.team0_numbers),len(globals.event_names)))
    team0_data = np.array(team0_data,dtype=int)

    team1_data =
              np.zeros((len(globals.team1_numbers),len(globals.event_names)))
    team1_data = np.array(team1_data,dtype=int)

    ## Turn arrays into dataframes
    globals.df = pd.DataFrame(teams_data, index = globals.event_names,
                                               columns=globals.team_names)

    globals.df0 = pd.DataFrame(team0_data, index=[f'{globals.team0_players[i]}
      {globals.team0_numbers[i]}' for i in range(len(globals.team0_numbers))],
                                              columns=globals.event_names)

    globals.df1 = pd.DataFrame(team1_data, index=[f'{globals.team1_players[i]}
      {globals.team1_numbers[i]}' for i in range(len(globals.team1_numbers))],
                                              columns=globals.event_names)
```

The update_data() function updates the dataframes given event, team and player. If player is unknown, it doesn't update anything in the individual team dataframe and only updates the team total. Otherwise, it also updates the individual player's total. Event, team and player can be used directly as indices due to their formatting earlier.

```python
def update_data(event, team, player):
    if player != "Unknown":
        if team == globals.team_names[0]: # Update team dataframe
            globals.df0[event][player] += 1
        elif team == globals.team_names[1]:
            globals.df1[event][player] += 1

    globals.df[team][event] += 1 # Update totals dataframe
```

If the event is "Sub", the sub_command() function will be run. It gets the list of players, swaps the position of the players being changed, and runs reindex() on the dataframe.

```python
def sub_command(team, players):
    if team == globals.team_names[0]:
        index_list = [f'{globals.team0_players[i]} {globals.team0_numbers[i]}'
                                  for i in range(len(globals.team0_numbers))]

        a, b = index_list.index(players[0]), index_list.index(players[1])

        index_list[b], index_list[a] = index_list[a], index_list[b]

        globals.df0 = globals.df0.reindex(index_list, axis="index")
        globals.team0_players[b], globals.team0_players[a] =
                              globals.team0_players[a], globals.team0_players[b]
        globals.team0_numbers[b], globals.team0_numbers[a] =
                              globals.team0_numbers[a], globals.team0_numbers[b]
```

If the event is “Undo”, the undo_command() function will be run. First it collects the last command from the list of commands and removes it from the list. If it was a substitution, the sub_command() function can just be run again to reverse the change. If the player was known, their individual total is reduced. The team total is also reduced.

```python
def undo_command():
    event, team, player = globals.commands[len(globals.commands)-1]
    globals.commands.pop()

    if event == "Sub":
        sub_command(team, player)
        return
    elif player == "Unknown":
        pass
    elif team == globals.team_names[0]: # subtract from dataframe
        globals.df0[event][player] -= 1
    elif team == globals.team_names[1]: # subtract from dataframe
        globals.df1[event][player] -= 1

    globals.df[team][event] -= 1 # subtract from dataframe
```

The plot_data() function creates heatmap tables using the Seaborn library and Matplotlib. It checks which is the current view (Team Totals, Team 1, Team 2), and plots the heatmap based on the relevant dataframe. The data in each row is scaled to between 0 and 1 so that the heatmap can be run on each row individually. This scaling causes zeros to become NaNs so these must be replaced afterwards. The ticks along each axis are turned off and the y-axis ticks are rotated to be the right way up.

```python
def plot_data():
    fig = plt.figure(figsize=(11, 9))

    if globals.view== globals.views[0]:
        dataframe = globals.df
        scaled_dataframe = dataframe.div(dataframe.max(axis=1), axis=0)
        scaled_dataframe.fillna(0, inplace=True)
        sb.heatmap(scaled_dataframe, cmap="Blues", robust=True,
                   linewidth=0.3,cbar=False, annot=dataframe)
      # Other views hidden for simplicity, in Appendix if wanted
    plt.tick_params(axis='both', which='major', labelbottom = False,
                                bottom=False, top = False, labeltop=True)
    plt.yticks(rotation=0)
    return fig
```

6.2.6 Drawing.py

This file contains functions to draw, delete and redraw the main figure, update the command feed and run the on-screen clock.

Comm_feed() updates the on-screen command feed with the commands in the command log text file.

```
def comm_feed(): # Add command from log to on screen feed
    with open(f"text_files/{globals.match_title}_command_log.txt", "r+") as
                                                                       text:
        globals.window['-FEED-'].update(text.read())
```

Draw_figure() takes the canvas(area in the GUI where the figure is plotted) and the heatmap as input and draws the figure on the canvas.

```
def draw_figure(canvas, figure): # Draw figure on canvas
        figure_canvas_agg = FigureCanvasTkAgg(figure, canvas)
        figure_canvas_agg.draw()
        figure_canvas_agg.get_tk_widget().pack(side='left', fill='both',
                                                                   expand=1)
        globals.fig_agg = figure_canvas_agg
```

Delete_fig_agg() clears the canvas ready for a new figure to be plotted.

```
def delete_fig_agg(): # Delete figure from canvas
    globals.fig_agg.get_tk_widget().forget()
    plt.close('all')
```

Redraw_figure() clears the canvas, creates the figure and then draws the figure on the canvas.

```
def redraw_figure():
    if globals.fig_agg is not None: # Delete current figure
        delete_fig_agg()
    figure = dp.plot_data() # Plot new figure
    if figure is None:
        return
    canvas_elem = globals.window['-CANVAS-'].TKCanvas # Find canvas
    draw_figure(canvas_elem, figure) # Draw figure on canvas
```

The run_clock() function is constantly running in its own thread. It runs every 0.8 seconds to save resources as there is no need for it to tun any faster. During the first half the time is simply current time subtract start time. During half time the clock is frozen on the time that the second half will start from. During the second half, the time is current time subtract the time at which the second half started and add on the half length. Once the match is over, the clock stops.

```python
def run_clock(): # Constantly running function for clock
    while not globals.half_time_begin: # Check for half time
        time.sleep(0.8) # Sleep so as to not run unnecessary loops

        ## Find time
        now = datetime.now()
        minutes = ((now.hour - globals.start_hour) * 60) + (now.minute -
                                                         globals.start_minute)
        seconds = now.second - globals.start_second
        if seconds < 0:
                minutes -= 1
                seconds = 60 + seconds
        globals.window['-TIME-'].update(f'{minutes:02d}:{seconds:02d}')

    while not globals.half_time_end: # During half time
        globals.window['-TIME-'].update(f'{globals.half_length}:00')
        time.sleep(2) # Sleep while waiting to start second half

    while not globals.full_time: # During second half
        time.sleep(0.8) # Sleep to prevent unnecessary looping

        ## Find time
        now = datetime.now()
        minutes = ((now.hour - globals.second_half_hour) * 60) + (now.minute -
                            globals.second_half_minute) + globals.half_length
        seconds = now.second - globals.second_half_second
        if seconds < 0:
                minutes -= 1
                seconds = 60 + seconds
        globals.window['-TIME-'].update(f'{minutes:02d}:{seconds:02d}')
```

6.2.7 Gui.py

This file contains the methods to create the GUI window, run it, and collect events from it.

The gui_setup() function creates the GUI window. First It runs get_w_h() which reads the screen height and width by creating a blank window.

```python
def get_w_h():
    blank = sg.Window("",layout=[], alpha_channel=0) # Create a blank window
    w, h = blank.get_screen_size() # Use blank window to get screen h and w
    blank.close() # Close blank window
    return w, h
```

Gui_setup() then defines all the GUI elements and their positions using PySimpleGUI.

Gui() reads events from the GUI window.

It runs constantly in the main thread. First it draws the initial figure on the canvas, then it begins listening for events. Each event is checked against a match-case statement.

```python
def gui():
    # Draw figure on canvas
    drawing.redraw_figure()

    while True:
        event, _ = globals.window.read(timeout=3000) # Read window for events
                                              and timeout after 3 seconds
        match event:
```

The first case is spacebar i.e., the recording stop/start button. If the program is not currently recording, the start_recording flag is set to True and the recording notifier in the GUI is shown. If the program is currently recording, end_recording flag is set to True and the recording notifier is hidden.

```python
            case " ": # Recording button
                if not globals.start_recording:
                    globals.start_recording = True
                    globals.window['-REC-'].update(visible=True)
                elif not globals.end_recording:
                    globals.end_recording = True
                    globals.window['-REC-'].update(visible=False)
```

The next two cases are event timeout which happens after 3 seconds, and window closed. Timeout just passes to the common commands at the end of the match-case. If the window has been closed, the log files are deleted and then the program exits.

```python
            case '__TIMEOUT__': # Window read timeout after 3 seconds
                pass
            case sg.WIN_CLOSED:
                files = [f for f in listdir('text_files') if
                                                isfile(join('text_files', f))]
                for file in files:
                    remove(f'text_files/{file}')
                exit()
```

The next three cases are for changing views. They come from the buttons at the top of the page. They change the current view so that when the heatmap is updated it will show the new view. It also changes the page title.

```python
            case '-TOTALS-': # Team totals button, change view to team totals
                title = "Team Totals"
                globals.window['-TITLE-'].update(title)
                globals.view = globals.views[0]
            case '-TEAM0-': # Team 0 button, change view to team 0
                title = globals.team_names[0]
                globals.window['-TITLE-'].update(title)
                globals.view = globals.views[1]
            case '-TEAM1-': # Team 1 button, change view to team 1
                title = globals.team_names[1]
                globals.window['-TITLE-'].update(title)
                globals.view = globals.views[2]
```

The next case is from the timing button at the top of the screen. It changes the label of the button to the next time event, sets the necessary flags based on the current time event, changes the colour of the clock and starts the clock thread if the event is “Start Match”.

```
            case '-TIMING-': # Timing button, updates latest timing action
                if not globals.start_match: # Start Match
                    globals.start_match = True
                    globals.window['-TIME-'].Widget
                                                .configure(background='green')
                    globals.window['-TIMING-'].update("Half Time")
                    now = datetime.now()
                    globals.start_hour = now.hour
                    globals.start_minute = now.minute
                    globals.start_second = now.second
                    d = threading.Thread(target=drawing.run_clock,
                                                                  daemon=True)
                    d.start()
                elif not globals.half_time_begin: # Half Time
                    globals.window['-TIME-'].Widget
                                                  .configure(background='red')
                    globals.window['-TIMING-'].update("Start Second Half")
                    globals.half_time_begin = True
```

The last events are export and Null. Export starts a thread running the export_data() function. The empty case collects any other keyboard events and ignores them.

```
            case '-EXPORT-': # Export Data button
                c = threading.Thread(target=export.export_data, daemon=True)
                c.start()
            case _:
                continue
```

The last task in the loop is to redraw the figure and the command feed.

```
        drawing.redraw_figure() # Redraw figure on canvas
        b = threading.Thread(target=drawing.comm_feed, daemon=True)
        b.start()
```

6.2.8 Export.py

This file contains functions to export the data that was collected.

Export_data() first creates a folder to store all the export files. It then copies the command log into this folder. Next it saves each of the heatmaps to the folder. Finally, it converts the dataframes to an excel sheet and saves that in the folder as well.

```python
def export_data(): # Export dataframes to excel
    today = date.today().strftime("%d-%m-%Y")

    path = f'{os.getcwd()}/Match Reports/{globals.match_title}
             {globals.team_names[0]} vs {globals.team_names[1]} {today}'

    with contextlib.suppress(FileExistsError):
        os.mkdir(path) # Create folder to hold exported data and command log
    file = f'{path}/{globals.match_title} {globals.team_names[0]} vs
                       {globals.team_names[1]} {today}.xlsx'

    # Copy command log to export folder
    shutil.copyfile(f"text_files/{globals.match_title}_command_log.txt",
                     f"{path}/{globals.match_title}_command_log.txt")

    save_heatmaps(path)

    try:
        with pd.ExcelWriter(file) as writer: # Write excel sheets
            globals.df.to_excel(writer, sheet_name='Team Totals')
            globals.df0.to_excel(writer, sheet_name=globals.team_names[0])
            globals.df1.to_excel(writer, sheet_name=globals.team_names[1])
    except PermissionError:
        sg.popup("Close File and Try Again")
        return

    #Popup to confirm export
    sg.popup_no_buttons("\nData Exported\n",
       keep_on_top=True,auto_close=True,icon=None, auto_close_duration=2,
         no_titlebar=True)
```

Chapter 6 **Testing**

6.1 Commands

Command	Expected Result	Pass/Fail
Goal Tipperary 7	Add to team total for Tipperary and player total for number 7	Pass
Goal for Tipperary scored by number 7	Add to team total for Tipperary and player total for number 7	Pass
Goal Tipperary Ronan Maher(No. 7)	Add to team total for Tipperary and player total for Ronan Maher(No. 7)	Pass
Goal for Tipperary scored by Ronan Maher(No. 7)	Add to team total for Tipperary and player total for Ronan Maher(No. 7)	Pass
Goal Tipperary Unknown	Add to team total for Tipperary and not to any player total	Pass
Goal for Tipperary scored by unknown	Add to team total for Tipperary and not to any player total	Pass
Save Tipperary Goalkeeper	Add to team total for Tipperary and player total for the goalkeeper	Pass
Save made by the Tipperary Goalkeeper	Add to team total for Tipperary and player total for the goalkeeper	Pass
Undo (Goal Tipperary 7)	Subtract from team total for Tipperary and player total for number 7	Pass
Undo (Goal Tipperary Ronan Maher(No. 7))	Subtract from team total for Tipperary and player total for Ronan Maher(No. 7)	Pass
Undo (Goal Tipperary Unknown)	Subtract from team total for Tipperary and not from any player total	Pass
Sub Tipperary 7 for 23	Swap players 7 and 23 in the table	Pass
Sub Tipperary Ronan Maher(7) for Jake Morris(23)	Swap players Ronan Maher(7) and Jake Morris(23) in the table	Pass
Undo (Sub Tipperary 7 for 23)	Swap players 7 and 23 in the table	Pass

6.2 Timing

Event	Expected Result	Pass/Fail
Click Button “Start Match”	Clock changes to green, clock starts ticking	Pass
Click Button “Half Time”	Clock changes to red, clock stops ticking, time changes to time at beginning of second half	Pass
Click Button “Start Second Half”	Clock changes to green, clock starts ticking from half time	Pass
Click Button “Full Time”	Clock changes to red, clock starts ticking, buttons disabled	Pass
Click Button when disabled	Nothing	Pass

Chapter 7 **Conclusions & Future Work**

7.1 Conclusions

This project has proven that it is possible to collect sports statistics by voice input in a quick and accurate way. The benefits of such a system over its competitors have already been discussed in the introduction.

This project was one in which there was no clear end point. One could always add new features and capabilities, but I believe the program in its current state is fit for use. As such, this is only an early version of this program and there is large scope for future work.

7.2 Future Work

Possible future features could include:

- Support for player nicknames
- Track Yellow and Red Cards separate to the heatmap table. There is no need for an entire column just to show red and yellow cards. It could instead be done with icons beside the player's name.
- The ability to add player names during a game if the user didn't know them before the game.
- If a certain command cannot be understood by the program, the user may type it in.
- Showing how many minutes each player has played so far to distinguish between substitutes and players who have played the full match.

This project only dealt with the data collection side of sports statistics, it didn't enter into the realm of data analysis, which could be a topic for future work. The ability to study the data and return useful, meaningful, actionable analysis is difficult to do without also watching the match in question but with enough data hopefully it could be done. The system could also compare matches to find trends and again return actionable analysis.

7.3 Timeframe

This project was completed in time with some sections taking less time than anticipated. As a result, “Additional Features” was given more time than planned at the end of the project.

The project Gantt Chart is included in Appendix A.

References

[1] L. Steinberg, “CHANGING THE GAME: The Rise of Sports Analytics,” [Online]. Available: https://www.forbes.com/sites/leighsteinberg/2015/08/18/changing-the-game-the-rise-of-sports-analytics/?sh=46e69fe24c1f.

[2] R. Villiers, “Sports Stats Recorder,” [Online]. Available: https://play.google.com/store/apps/details?id=com.milfordgaa.sportsstar&hl=en_IE&gl=US.

[3] “StatsPerform,” StatsPerform, [Online]. Available: https://www.statsperform.com.

[4] A. Wessel, “PracStat: A voice-activated statistic tracking software for the Ball State men’s volleyball team,” Ball State University, 5 2019. [Online]. Available: https://core.ac.uk/download/pdf/231877586.pdf. [Accessed 18 10 2021].

[5] T. Baginski, “Sports Statistics Tracker - Speech to Text to Dataverse!,” Microsoft, 06 07 2021. [Online]. Available: https://powerusers.microsoft.com/t5/Demo-Extravaganza-2021/Sports-Statistics-Tracker-Speech-to-Text-to-Dataverse/cns-p/945380. [Accessed 18 10 2021].

[6] IBM, “What is speech recognition?,” [Online]. Available: https://www.ibm.com/cloud/learn/speech-recognition. [Accessed 22 October 2021].

[7] P. S. Foundation, “Python Beginner's Guide,” [Online]. Available: https://wiki.python.org/moin/BeginnersGuide/Overview.

[8] P. S. Foundation, “pocketsphinx 0.1.15,” [Online]. Available: https://pypi.org/project/pocketsphinx/.

Appendix A: Project Gantt chart

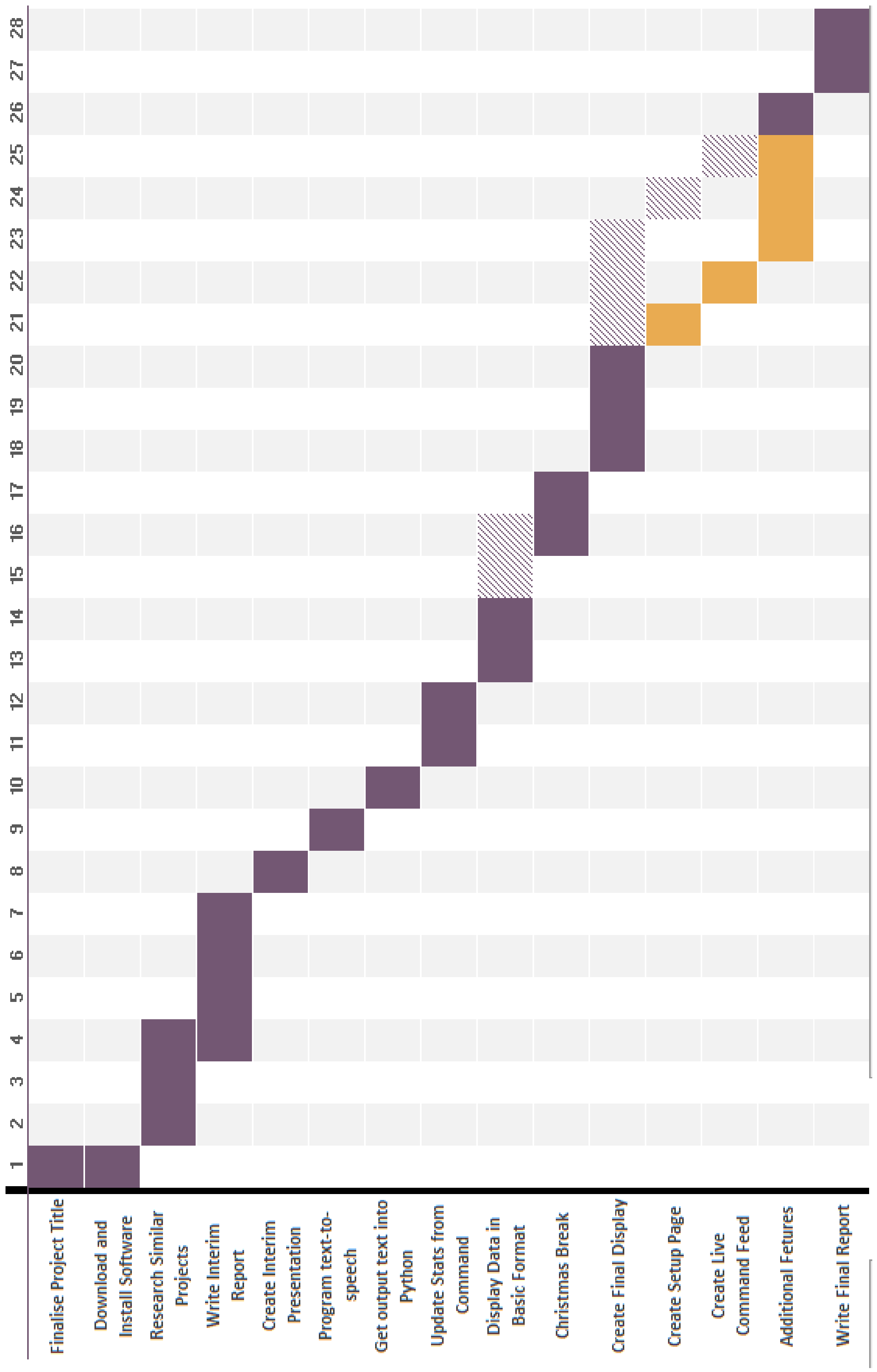


Figure 14 Gantt Chart

Purple, both solid and striped indicates the planned time. Solid purple and yellow are the actual time taken, with yellow being time before the original planned start time.

Appendix B: Source Code

a. Main.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#----------------------------------------------------------------------------
# Author   : Seán McTiernan
# ---------------------------------------------------------------------------
""" This file contains the main function calls for the program. It performs
    all of the setup actions and then calls the gui() main function."""
# ---------------------------------------------------------------------------

# Imports
import threading                # Method to create threads

import data_processing as dp    # Data processing functions
from gui import gui, gui_setup  # Functions to setup and run the GUI
import audio                    # Functions to record and process audio
import input                    # Functions to retrieve information
                                                               from setup.xls
import globals                  # Global variables
# ---------------------------------------------------------------------------


if __name__ == "__main__":

    audio.delete_log_files() # Delete all audio files from audio_files folder

    input.get_info() # Load info from excel file

    gui_setup() # Create GUI Window

    # Create command log
    with open(f"text_files/{globals.match_title}_command_log.txt", "w+") as f:
        f.write("")

    # Start constant audio recording thread
    x = threading.Thread(target=audio.audio_recording, daemon=True)
    x.start()

    # Create dataframes
    dp.create_dataframe()

    # Run main function
    gui()
```

b. Audio.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#----------------------------------------------------------------------------
# Author  : Seán McTiernan
# ---------------------------------------------------------------------------
""" This file contains functions to record and process audio and to delete log
files."""
# ---------------------------------------------------------------------------

# Imports
import wave                     # Functions to parse wave files
import speech_recognition as sr # Function for Google Speech Recognition API
import pyaudio
import threading                # Method to create threads
from os import listdir, remove  # Functions to find and delete files
from os.path import isfile, join# Functions to check files and combine strings
import winsound                 # Function to play system sounds

from parse import parse
import data_processing as dp    # Data processing functions

import globals                  # Global variables
# ---------------------------------------------------------------------------

chunk=3024
channels=2
rate=44100
audio=pyaudio.PyAudio()

def delete_log_files(): # delete all audio files
    files = [f for f in listdir('audio_files') if isfile(join('audio_files',
f))]
    for file in files:
        remove(f'audio_files/{file}')


def process_audio(path):
    r = sr.Recognizer()
    audio_file = sr.AudioFile(path)

    with audio_file as source:
        r.adjust_for_ambient_noise(source)
        audio = r.record(source)

    text = r.recognize_google(audio, show_all=True)

    event, team, player = parse(text)
    if "Undo" in [event, team, player]:
        dp.undo_command()
        return

    if event == "Sub":
        dp.sub_command(team, player)
        return

    if None in [event, team, player]:
        winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS)
        remove(path)
        return

    dp.update_data(event, team, player)

def audio_recording(): # Constantly record audio, save to file when flag set
    frames = []
    i=0
    while True:
        stream = pyaudio.PyAudio().open(format=pyaudio.paInt16, channels=2,
rate=44100, input=True, frames_per_buffer=3024)
        while not globals.start_recording:
            data = stream.read(chunk*50)
            frames = [data]
        while not globals.end_recording: # when recording flag set
            data = stream.read(chunk)
            frames.append(data)
        globals.end_recording = False # reset flags
        globals.start_recording = False

        stream.close()

        ## Save file
        file = f'audio_files/phrase{i}.wav'
        wf = wave.open(file, 'wb')
        wf.setnchannels(channels)
        wf.setsampwidth(audio.get_sample_size(pyaudio.paInt16))
        wf.setframerate(rate)
        wf.writeframes(b''.join(frames))
        wf.close()

        z = threading.Thread(target=process_audio, args=(file,), daemon=True)
        z.start()
        i += 1
```

c. Data_processing.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#----------------------------------------------------------------------------
# Author  : Seán McTiernan
# ---------------------------------------------------------------------------
""" This file contains functions to create and update the dataframes and plot
the heatmaps."""
# ---------------------------------------------------------------------------

# Imports
import numpy as np                # Functions to create arrays
import pandas as pd               # Functions to create dataframes
import matplotlib.pyplot as plt # Functions to plot figures
import seaborn as sb              # Functions to create heatmap tables

import globals                    # Global variables
# ---------------------------------------------------------------------------

def create_dataframe():
    ## Create arrays of zeroes
    teams_data = np.zeros((len(globals.event_names), 2))
    teams_data = np.array(teams_data,dtype=int)

    team0_data =
np.zeros((len(globals.team0_numbers),len(globals.event_names)))
    team0_data = np.array(team0_data,dtype=int)

    team1_data =
np.zeros((len(globals.team1_numbers),len(globals.event_names)))
    team1_data = np.array(team1_data,dtype=int)

    ## Turn arrays into dataframes
    globals.df = pd.DataFrame(teams_data, index = globals.event_names,
      columns=globals.team_names)

    globals.df0 = pd.DataFrame(team0_data, index=[f'{globals.team0_players[i]}
      {globals.team0_numbers[i]}' for i in range(len(globals.team0_numbers))],
      columns=globals.event_names)

    globals.df1 = pd.DataFrame(team1_data, index=[f'{globals.team1_players[i]}
      {globals.team1_numbers[i]}' for i in range(len(globals.team1_numbers))],
      columns=globals.event_names)
```

```python
# Update function
def update_data(event, team, player):
    if player != "Unknown":
        if team == globals.team_names[0]: # Update team dataframe
            globals.df0[event][player] += 1
        elif team == globals.team_names[1]:
            globals.df1[event][player] += 1

    globals.df[team][event] += 1 # Update totals dataframe

def plot_data():

    fig = plt.figure(figsize=(11, 9)) # Create a figure

    if globals.view== globals.views[0]: # Team Totals
        dataframe = globals.df
        # Scale data in each row to between 0 and 1
        scaled_dataframe = dataframe.div(dataframe.max(axis=1), axis=0)
        # Scaling turns 0s to NaNs so replace those with 0s
        scaled_dataframe.fillna(0, inplace=True)
        # Put scaled data into heatmap with non-scaled data as annotations
        sb.heatmap(scaled_dataframe, cmap="Blues", robust=True,
                   linewidth=0.3,cbar=False, annot=dataframe)

    elif globals.view== globals.views[1]: # Team 1
        dataframe = globals.df0
        # Scale data in each column to between 0 and 1
        scaled_dataframe = (dataframe -
        dataframe.min(axis=0))/(dataframe.max(axis=0) - dataframe.min(axis=0))
        # Scaling turns 0s to NaNs so replace those with 0s
        scaled_dataframe.fillna(0, inplace=True)
        # Put scaled data into heatmap with non-scaled data as annotations
        sb.heatmap(scaled_dataframe[:15], cmap="Blues", robust=True,
                   linewidth=0.3,cbar=False, annot=dataframe[:15])

    elif globals.view== globals.views[2]: # Team 2
        dataframe = globals.df1
        # Scale data in each column to between 0 and 1
        scaled_dataframe = (dataframe -
        dataframe.min(axis=0))/(dataframe.max(axis=0) - dataframe.min(axis=0))
        # Scaling turns 0s to NaNs so replace those with 0s
        scaled_dataframe.fillna(0, inplace=True)
        # Put scaled data into heatmap with non-scaled data as annotations
        sb.heatmap(scaled_dataframe[:15], cmap="Blues", robust=True,
                   linewidth=0.3,cbar=False, annot=dataframe[:15])


    plt.tick_params(axis='both', which='major',
                    labelbottom = False, bottom=False, top = False,
labeltop=True) # Set tick labels parameters
    plt.yticks(rotation=0) # Rotate labels
    return fig

def undo_command():
    # Get last command
    event, team, player = globals.commands[len(globals.commands)-1]
    globals.commands.pop()

    if event == "Sub":
        sub_command(team, player)
        return
    elif player == "Unknown":
        pass
    elif team == globals.team_names[0]: # subtract from dataframe
        globals.df0[event][player] -= 1
    elif team == globals.team_names[1]: # subtract from dataframe
        globals.df1[event][player] -= 1

    globals.df[team][event] -= 1 # subtract from dataframe


def sub_command(team, players):
    if team == globals.team_names[0]:
        index_list = [f'{globals.team0_players[i]} {globals.team0_numbers[i]}'
for i in range(len(globals.team0_numbers))]
        a, b = index_list.index(players[0]), index_list.index(players[1])
        index_list[b], index_list[a] = index_list[a], index_list[b]
        globals.team0_players[b], globals.team0_players[a] =
globals.team0_players[a], globals.team0_players[b]
        globals.team0_numbers[b], globals.team0_numbers[a] =
globals.team0_numbers[a], globals.team0_numbers[b]
        globals.df0 = globals.df0.reindex(index_list, axis="index")
    elif team == globals.team_names[1]:
        index_list = [f'{globals.team1_players[i]} {globals.team1_numbers[i]}'
for i in range(len(globals.team1_numbers))]
        a, b = index_list.index(players[0]), index_list.index(players[1])
        index_list[b], index_list[a] = index_list[a], index_list[b]
        globals.team1_players[b], globals.team1_players[a] =
globals.team1_players[a], globals.team1_players[b]
        globals.team1_numbers[b], globals.team1_numbers[a] =
globals.team1_numbers[a], globals.team1_numbers[b]
        globals.df1 = globals.df1.reindex(index_list, axis="index")
```

d. Drawing.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#----------------------------------------------------------------------------
# Author  : Seán McTiernan
# ---------------------------------------------------------------------------
""" This file contains functions to draw, redraw and delete figures on the
canvas. Also, to run the clock and update the command feed."""
# ---------------------------------------------------------------------------

# Imports
import time                             # Function to perform sleeps
from datetime import datetime           # Functions to find the current time
import matplotlib.pyplot as plt         # Functions to plot figures
# Class to combine figure and canvas
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import data_processing as dp            # Contains data processing functions

import globals                           # Global variables
# ---------------------------------------------------------------------------

def draw_figure(canvas, figure): # Draw figure on canvas
        figure_canvas_agg = FigureCanvasTkAgg(figure, canvas)
        figure_canvas_agg.draw()
        figure_canvas_agg.get_tk_widget().pack(side='left', fill='both',
expand=1)
        globals.fig_agg = figure_canvas_agg

def delete_fig_agg(): # Delete figure from canvas
    globals.fig_agg.get_tk_widget().forget()
    plt.close('all')

def redraw_figure():
    if globals.fig_agg is not None: # Delete current figure
        delete_fig_agg()
    figure = dp.plot_data() # Plot new figure
    if figure is None:
        return
    canvas_elem = globals.window['-CANVAS-'].TKCanvas # Find canvas
    draw_figure(canvas_elem, figure) # Draw figure on canvas

def run_clock(): # Constantly running function for clock
    while not globals.half_time_begin: # Check for half time
        time.sleep(0.8) # Sleep to not run unnecessary loops
        ## Find time
        now = datetime.now()
        minutes = ((now.hour - globals.start_hour) * 60) + (now.minute –
                        globals.start_minute)
        seconds = now.second - globals.start_second
        if seconds < 0:
                minutes -= 1
                seconds = 60 + seconds
        # Update clock with time string
        globals.window['-TIME-'].update(f'{minutes:02d}:{seconds:02d}')

    while not globals.half_time_end: # During half time
        # Set clock to start of second half time
        globals.window['-TIME-'].update(f'{globals.half_length}:00')
        time.sleep(2) # Sleep while waiting to start second half

    while not globals.full_time: # During second half
        time.sleep(0.8) # Sleep to prevent unnecessary looping

        ## Find time
        now = datetime.now()
        minutes = ((now.hour - globals.second_half_hour) * 60) + (now.minute –
                   globals.second_half_minute) + globals.half_length
        seconds = now.second - globals.second_half_second
        if seconds < 0:
                minutes -= 1
                seconds = 60 + seconds
        # Print time string to clock
        globals.window['-TIME-'].update(f'{minutes:02d}:{seconds:02d}')


def comm_feed(): # Add command from log to on screen feed
    with open(f"text_files/{globals.match_title}_command_log.txt", "r+") as
                                                                      text:
        globals.window['-FEED-'].update(text.read())
```

e. Parse.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#----------------------------------------------------------------------------
# Author  : Seán McTiernan
# ---------------------------------------------------------------------------
""" This file contains functions to find the event, team and player in the
input text."""
# ---------------------------------------------------------------------------

# Imports
import contextlib                               # Function to supress errors
import itertools                                # Functions to combine loops
# Functions to check and remove close and exact string matches
from fuzzy import fuzzy, fuzzy_remove, exact_check
# Class to combine figure and canvas
from number2word import number2word as n2w      # Function to convert numbers
to words
from word2number import word2number as w2n      # Function to convert words to
numbers

import globals                                  # Global variables
# ---------------------------------------------------------------------------

def clean_input(text): # This function removes common filler words that are
not used to determine the event and will only slow down and confuse the parser
    mapping = [ ('/', ' '), ('-', ' '), (' a ', ' '), (' an ', ' '), (' by ',
' '), (' for ', ' '), (' scored ', ' '), ('subtle', 'sub'), (' number ', ' '),
('that\'s', ' '), ('there\'s', ' '), ('that', ' '), ('there', ' '),
('go4kora', ''), ('pt4', 'point'), ('three conceded', 'free conceded'), ('3
conceded', 'free conceded'), ('3 considered', 'free conceded'), ('three
considered', 'free conceded'), ('point four', 'point'), ('point 4', 'point'),
('goal 4', 'goal'), ('goal four', 'goal') ]
    for (i, _), (k, v) in itertools.product(enumerate(text), mapping):
        text[i] = text[i].replace(k, v)

    return text

def check_undo(text): # This function checks if the event is "Undo"
    words = ['undo', 'undone']
    if fuzzy(words, text, True) in words:
        return True

def check_sub(text):
    words = ['sub', 'substitution']
    if fuzzy(words, text, True) in words:
        return True
```

```python
def unknown(text):
    words = ['unknown', 'Unknown']
    if fuzzy(words, text, True) in words:
        return True

def write(command="", undo=False): # Write command to log
    if not undo:
        with open(f"text_files/{globals.match_title}_command_log.txt","a+") as 
f:
            with contextlib.suppress(TypeError):
                if globals.commands == []:
                    f.write(f'{globals.window["-TIME-"].DisplayText}:
{command}\n')
                else:
                    f.write(f'\n{globals.window["-TIME-"].DisplayText}:
{command}\n')
    else:
        with open(f"text_files/{globals.match_title}_command_log.txt", 'r+') 
as f:
            lines = f.readlines()
            f.seek(0)
            f.truncate()
            f.writelines(lines[:-2])

def check_team(text):
    team = exact_check(globals.team_names, text)
    if team is None:
        team = fuzzy(globals.team_names.copy(), text, strict=True) # Find team
        if team is None:
            team = exact_check(globals.team_nicknames, text)
            if team is None:
                team = fuzzy(globals.team_nicknames.copy(), text, strict=True)
                if team is None:
                    return None, text

    text = fuzzy_remove(text, team) # Remove team name from input
    if team in globals.team_nicknames:
        team = globals.team_names[globals.team_nicknames.index(team)]
    return team, text

def check_event(text):
    event = exact_check(globals.event_names, text)
    if event is None:
        event = fuzzy(globals.event_names.copy(), text) # Find event
    if event is None:
        return None, text
    text = fuzzy_remove(text, event)
    return event, text
```

```
def check_player(text, team):
    if check_keeper(text):
        player = combine_name_number(team, "1")
        text = fuzzy_remove(text, ["keeper", "goalie", "goalkeeper"])
        return player, text

    if unknown(text):
        return ("Unknown", text)
    if team == globals.team_names[0]:
        player = exact_check(globals.team0_players+[str(number) for number in 
globals.team0_numbers]+n2w(globals.team0_numbers), text)
        if player is None:
            player = fuzzy(globals.team0_players.copy()+[str(number) for 
number in globals.team0_numbers]+n2w(globals.team0_numbers), text)
    elif team == globals.team_names[1]:
        player = exact_check(globals.team1_players+[str(number) for number in 
globals.team1_numbers]+n2w(globals.team1_numbers), text)
        if player is None:
            player = fuzzy(globals.team1_players.copy()+[str(number) for 
number in globals.team1_numbers]+n2w(globals.team1_numbers), text)
    if player is None:
        return (player, text)
    text = fuzzy_remove(text, player)
    if w2n(player) is not None:
        text = fuzzy_remove(text, w2n(player))
    if n2w(player) is not None:
        text = fuzzy_remove(text, n2w(player))

    player = combine_name_number(team, player)

    return (player, text)

def check_keeper(text):
    words = ['keeper', 'goalie', 'goalkeeper']
    if fuzzy(words, text, True) in words:
        return True

def player_check_without_team(text):
    player = exact_check(globals.team0_players, text)
    team = None
    if player is None:
        player = exact_check(globals.team1_players, text)
    else:
        team = globals.team_names[0]
        return combine_name_number(team, player), team, fuzzy_remove(text, 
player)
```

```
    if player is None:
        player = fuzzy(globals.team0_players.copy(), text)
    else:
        team = globals.team_names[1]
        return combine_name_number(team, player), team, fuzzy_remove(text,
player)

    if player is None:
        player = fuzzy(globals.team1_players.copy(), text)
    else:
        team = globals.team_names[0]
        return combine_name_number(team, player), team, fuzzy_remove(text,
player)

    if player is not None:
        team = globals.team_names[1]

    return combine_name_number(team, player), team, fuzzy_remove(text, player)

def sub(text):
    text = fuzzy_remove(text, ["sub", "substitution"])
    team, text = check_team(text)
    player_1, text = check_player(text, team)
    player_2, text = check_player(text, team)
    if None in [team, player_1, player_2]:
        return None, None, None
    write(f'Sub {team.capitalize()} {player_1} for {player_2}') # Write to
command log
    globals.commands.append(["Sub", team, [player_1, player_2]])
    return "Sub", team, [player_1, player_2]

def parse(text):
    try: # Split predictions into a list
        text = [phrase['transcript'] for phrase in text['alternative']]

    except:
        return None, None, None

    text = clean_input(text) # remove filler words

    if check_undo(text): # Check if undo command entered
        write(undo=True)
        return "Undo", "Undo", "Undo"

    if check_sub(text):
        return sub(text)
    team, text = check_team(text)
    if team is None:
        player, team, text = player_check_without_team(text)
    if team is None:
        return None, None, None

    event, text = check_event(text)
    if event is None:
        return None, None, None

    if 'player' not in locals():
        player, text = check_player(text, team)
    if player is None:
        return None, None, None

    # Write to command log
    write(f'{event.capitalize()} {team.capitalize()} {player}')

    # Add command to list of commands
    globals.commands.append([event, team, player])

    return event, team, player

def combine_name_number(team, player):
    if w2n(player) is not None:
        player = w2n(player)
    if team == globals.team_names[0]:
        if player in globals.team0_players: # If player found by name
            index = globals.team0_players.index(player)
        elif int(player) in globals.team0_numbers: # If player found by number
            index = globals.team0_numbers.index(int(player))
        player = f'{globals.team0_players[index]}
{globals.team0_numbers[index]}' # Combine player name and number

    elif team == globals.team_names[1]:
        if player in globals.team1_players: # If player found by name
            index = globals.team1_players.index(player)
        elif int(player) in globals.team1_numbers: # If player found by number
            index = globals.team1_numbers.index(int(player))
        player = f'{globals.team1_players[index]}
{globals.team1_numbers[index]}' # Combine player name and number
    return player
```

f. Fuzzy.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#----------------------------------------------------------------------------
# Author  : Seán McTiernan
# ---------------------------------------------------------------------------
""" This file contains functions to compare strings that are not an exact
match."""
# ---------------------------------------------------------------------------

# Imports
from thefuzz import fuzz   # Function to find comparison ratio between strings
import itertools           # Functions to combine loops

# ---------------------------------------------------------------------------

def fuzzy(choices, text, strict=False):
    choices = clean_choices(choices) # Removes empty strings and sorts the
list

    pairs, triples = create_pairs_triples(text) # Create pairs and triples of
words

    input_text = format_input(text) # Split input into single words and remove
duplicates

    results_80, results_60, results_49 = perform_fuzzy(choices, input_text,
strict, pairs, triples)

    try:
        if results_80.count(max(results_80)) == 1: # If single most >80%
accuracies, return that value
            return choices[results_80.index(max(results_80))]
        elif results_60.count(max(results_60)) == 1: # If single most >60%
accuracies, return that value
            return choices[results_60.index(max(results_60))]
        elif results_49.count(max(results_49)) == 1: # If single most >50%
accuracies, return that value
            return choices[results_49.index(max(results_49))]
        else:
            return None
    except ValueError:
        return None
```

```
def perform_fuzzy(choices, input_text, strict, pairs, triples):
    results_80 = [0]*len(choices) # Hold number of times each choice had an
accuracy of over 80%
    results_60 = [0]*len(choices) # Hold number of times each choice had an
accuracy of over 60%
    results_49 = [0]*len(choices) # Hold number of times each choice had an
accuracy of over 50%
    for i, choice in enumerate(choices): # Iterate through
events/teams/players
        if len(str(choice).split(" ")) == 1: # If single word choice
            words = input_text
        elif len(str(choice).split(" ")) == 2: # If 2 word choice
            words = pairs
        elif len(str(choice).split(" ")) == 3: # If 3 word choice
            words = triples
        for word in words: # Iterate through input words/pairs/triples
            ratio = fuzz.ratio(word.lower(), str(choice).lower()) # Get
comparison score
            if ratio > 80: # If > 80%, add 1 to 80% list
                results_80[i] += 1
            if not strict: # If strict, only scores above 80% are counted
                if ratio > 60: # If > 60%, add 1 to 60% list
                    results_60[i] += 1
                elif ratio > 49: # If > 49%, add 1 to 49% list
                    results_49[i] += 1
    return results_80, results_60, results_49

def format_input(text):
    input_text=[] # Hold list of words input
    for phrase in text: # Split incoming phrases into list of words
        input_text.extend(iter(phrase.split()))
    return input_text

def clean_choices(choices):
    choices = [choice for choice in choices.copy() if choice != '']
    choices = list(filter(None, choices))
    try:
        choices.sort() # sort list of events/team names/players
    except TypeError:
        print(choices)
    return choices

def create_pairs_triples(text):
    pairs = [word_groups(item, 2) for item in text]
    pairs = [item for sublist in pairs for item in sublist]
    triples = [word_groups(item, 3) for item in text]
    triples = [item for sublist in triples for item in sublist]
    return pairs,triples
```

```python
def word_groups(words, number): # Creates a list of pairs, triples etc. of
words
    groups = []
    for j in range(len(words.split())-(number-1)):
        group = ""
        for i in range(number):
            if words.split()[j + i] != " ":
                group += words.split()[j+i]
                if i < number-1 and group != '':
                    group += " "
                else:
                    break
        if group not in ['', ' ', '  ']:
            groups.append(group)
    return groups

def fuzzy_remove(text, result):
    if type(result) == list: # Recursion
        for item in result:
            text = fuzzy_remove(text, item)
        return text
    result = str(result)

    if len(result.split(" ")) == 1:
        for i, phrase in enumerate(text):
            words = phrase.split(" ")
            for word in words:
                if fuzz.ratio(word.lower(), result.lower()) > 60:
                    text[i] = phrase.replace(f" {word}", '')
                    text[i] = text[i].replace(word, '')
    elif len(result.split(" ")) == 2:
        for i, phrase in enumerate(text):
            pairs = word_groups(phrase, 2)
            for pair in pairs:
                if fuzz.ratio(pair.lower(), result.lower()) > 60:
                    text[i] = text[i].replace(f" {pair}", "")
                    text[i] = text[i].replace(pair, "")
    elif len(result.split(" ")) == 3:
        for i, phrase in enumerate(text):
            triples = word_groups(phrase, 3)
            for triple in triples:
                if fuzz.ratio(triple.lower(), result.lower()) > 60:
                    text[i] = text[i].replace(f" {triple}", "")
                    text[i] = text[i].replace(triple,)

    return text
```

```python
def exact_check(choices, text):
    choices = list(filter(None, choices)) # remove empty strings
    for phrase, choice in itertools.product(text, choices): # Loop through
choices and phrases
        if type(choice) != int:
            if choice.lower() in [word.lower() for word in word_groups(phrase,
len(choice.split()))]:
                return choice
        elif str(choice) in phrase.split():
            return choice
    return None
```

g. Word2number.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#-----------------------------------------------------------------------------
# Author  : Seán McTiernan
# ----------------------------------------------------------------------------
""" This file contains a function to convert numbers from word to int."""
# ----------------------------------------------------------------------------
def word2number(text): # Takes string input and returns int
    numwords = {
        "one" : 1,
        "two" : 2,
        "three" : 3,
        "four" : 4,
        "five" : 5,
        "six" : 6,
        "seven" : 7,
        "eight" : 8,
        "nine" : 9,
        "ten" : 10,
        "eleven" : 11,
        "twelve" : 12,
        "thirteen" : 13,
        "fourteen" : 14,
        "fifteen" : 15,
        "sixteen" : 16,
        "seventeen" : 17,
        "eighteen" : 18,
        "nineteen" : 19,
        "twenty" : 20,
        "twenty one" : 21,
        "twenty two" : 22,
        "twenty three" : 23,
        "twenty four" : 24,
        "twenty five" : 25,
        "twenty six" : 26,
        "twenty seven" : 27,
        "twenty eight" : 28,
        "twenty nine" : 29,
        "thirty" : 30,
        "thirty one" : 31,
        "thirty two" : 32
    }
    try:
        return numwords[text]
    except KeyError: # String not in list of numbers
        return None
```

h. Number2word.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#------------------------------------------------------------------------------
# Author   : Seán McTiernan
# ----------------------------------------------------------------------------
""" This file contains a function to convert numbers in int format into word
format."""
# ----------------------------------------------------------------------------

def number2word(number): # Takes int or list of ints input and returns string
or list of strings
    if type(number) == list: # if list
        return [number2word(num) for num in number]

    try:
        number = int(number)
    except ValueError:
        pass

    if type(number) != int: # if not int
        return None

    number = int(number)

    numwords = {
        1: 'one',
        2: 'two',
        3: 'three',
        4: 'four',
        5: 'five',
        6: 'six',
        7: 'seven',
        8: 'eight',
        9: 'nine',
        10: 'ten',
        11: 'eleven',
        12: 'twelve',
        13: 'thirteen',
        14: 'fourteen',
        15: 'fifteen',
        16: 'sixteen',
        17: 'seventeen',
        18: 'eighteen',
        19: 'nineteen',
        20: 'twenty',
        21: 'twenty one',
        22: 'twenty two',
        23: 'twenty three',
        24: 'twenty four',
        25: 'twenty five',
        26: 'twenty six',
        27: 'twenty seven',
        28: 'twenty eight',
        29: 'twenty nine',
        30: 'thirty',
        31: 'thirty one',
        32: 'thirty two'}

    try:
        return numwords[number]
    except KeyError: # Number not in list
        return None
```

i. Export.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#----------------------------------------------------------------------------
# Author   : Seán McTiernan
# ---------------------------------------------------------------------------
""" This file contains functions to export the match data to a local
folder."""
# ---------------------------------------------------------------------------

# Imports
import seaborn as sb                    # Functions to create heatmaps
import matplotlib.pyplot as plt         # Functions to plot figures
import os                               # Functions to control the OS
import shutil                           # Functions to copy files
from datetime import date               # Function to get today's date
import PySimpleGUI as sg                # Functions to run the GUI
import contextlib                       # Functions to supress errors
import pandas as pd                     # Functions to create dataframes and read
excel sheets
from PIL import Image                   # Functions to alter images

from drawing import redraw_figure       # Function to redraw the figure on the
canvas

import globals                          # Globals variables

# ---------------------------------------------------------------------------

def save_heatmaps(path):
    dataframe = globals.df0
    scaled_dataframe = (dataframe -
dataframe.min(axis=0))/(dataframe.max(axis=0) - dataframe.min(axis=0)) # scale
data in each column to between 0 and 1
    scaled_dataframe.fillna(0, inplace=True) # Scaling turns 0s to NaNs so
replace those with 0s
    heatmap = sb.heatmap(scaled_dataframe[:15], cmap="Blues", robust=True,
                    linewidth=0.3,cbar=False, annot=dataframe[:15]) # Put
scaled data into heatmap with non-scaled data as annotations
    plt.tick_params(axis='both', which='major',
                    labelbottom = False, bottom=False, top = False,
labeltop=True) # Set tick labels parameters
    plt.yticks(rotation=0) # Rotate labels
    heatmap = heatmap.get_figure()
    heatmap.savefig(f'{path}/{globals.team_names[0]}.png', dpi=400)

    dataframe = globals.df1
    scaled_dataframe = (dataframe -
dataframe.min(axis=0))/(dataframe.max(axis=0) - dataframe.min(axis=0)) # scale
data in each column to between 0 and 1
    scaled_dataframe.fillna(0, inplace=True) # Scaling turns 0s to NaNs so
replace those with 0s
    heatmap = sb.heatmap(scaled_dataframe[:15], cmap="Blues", robust=True,
                    linewidth=0.3,cbar=False, annot=dataframe[:15]) # Put
scaled data into heatmap with non-scaled data as annotations
    plt.tick_params(axis='both', which='major',
                    labelbottom = False, bottom=False, top = False,
labeltop=True) # Set tick labels parameters
    plt.yticks(rotation=0) # Rotate labels
    heatmap = heatmap.get_figure()
    heatmap.savefig(f'{path}/{globals.team_names[1]}.png', dpi=400)

    dataframe = globals.df
    scaled_dataframe = dataframe.div(dataframe.max(axis=1), axis=0) # scale
data in each row to between 0 and 1
    scaled_dataframe.fillna(0, inplace=True) # Scaling turns 0s to NaNs so
replace those with 0s
    heatmap = sb.heatmap(scaled_dataframe, cmap="Blues", robust=True,
               linewidth=0.3,cbar=False, annot=dataframe) # Put scaled data
into heatmap with non-scaled data as annotations
    plt.tick_params(axis='both', which='major',
                    labelbottom = False, bottom=False, top = False,
labeltop=True) # Set tick labels parameters
    plt.yticks(rotation=0) # Rotate labels
    heatmap = heatmap.get_figure()
    heatmap.savefig(f'{path}/Team Totals.png', dpi=400)

    redraw_figure()
    crop_image(f'{path}/Team Totals.png')


def export_data(): # Export dataframes to excel
    today = date.today().strftime("%d-%m-%Y")

    path = f'{os.getcwd()}/Match Reports/{globals.match_title}
{globals.team_names[0]} vs {globals.team_names[1]} {today}'

    with contextlib.suppress(FileExistsError):
        os.mkdir(path) # Create folder to hold exported data and command log
    file = f'{path}/{globals.match_title} {globals.team_names[0]} vs
{globals.team_names[1]} {today}.xlsx'

    # Copy command log to export folder
    shutil.copyfile(f"text_files/{globals.match_title}_command_log.txt",
f"{path}/{globals.match_title}_command_log.txt")
```

```
    save_heatmaps(path)

    try:
        with pd.ExcelWriter(file) as writer: # Write excel sheets
            globals.df.to_excel(writer, sheet_name='Team Totals')
            globals.df0.to_excel(writer, sheet_name=globals.team_names[0])
            globals.df1.to_excel(writer, sheet_name=globals.team_names[1])
    except PermissionError:
        sg.popup("Close File and Try Again")
        return

    #Popup to confirm export
    sg.popup_no_buttons("\nData Exported\n",
keep_on_top=True,auto_close=True,icon=None, auto_close_duration=2,
no_titlebar=True)

def crop_image(path):
    # Opens an image in RGB mode
    im = Image.open(path)

    # Size of the image in pixels
    width, height = im.size

    # Setting the points for cropped image
    left = 0
    top = 0
    right = width
    bottom = height - 140

    # Cropped image of above dimension
    im1 = im.crop((left, top, right, bottom))

    im1.save(path)
```

j. Import.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#----------------------------------------------------------------------------
# Author  : Seán McTiernan
# ---------------------------------------------------------------------------
""" This file contains functions for loading the information from the
setup.xls
    file into the relevant globals variables."""
# ---------------------------------------------------------------------------
# Imports
import contextlib       # Contains method to supress errors by type
import pandas as pd     # Used to read the excel sheets
import math             # Contains functions to test if variable is NaN

import globals
# ---------------------------------------------------------------------------
def get_info():
    """ Main function of the file. Reads the excel sheet, removes blank names
and NaN numbers and
    applies the values to the global variables"""

    setup = pd.read_excel('setup.xls', sheet_name="Setup")  # Read sheet data
    team_0 = pd.read_excel('setup.xls', sheet_name="Team 1")
    team_1 = pd.read_excel('setup.xls', sheet_name="Team 2")

    globals.match_title = setup[1][0]       # Set Match Title
    globals.half_length = int(setup[1][1])  # Set Half Length
    globals.team_names[0] = team_0[1][2]    # Set Team 0 name
    globals.team_names[1] = team_1[1][2]    # Set Team 1 name

    team0_players, team1_players = get_players(team_0, team_1) # Retrieve
player names

    team0_numbers, team1_numbers = get_numbers(team_0, team_1) # Retrieve
player numbers

    team0_players, team1_players, team0_numbers, team1_numbers =
remove_blanks(team0_players, team1_players, team0_numbers, team1_numbers)

    globals.event_names = get_events(setup) # Retrieve event names

    team0_players, team1_players = no_name(team0_players, team1_players) #
Sets players with no name to have name ""

    globals.team0_players = team0_players # Set Team 0 names
    globals.team1_players = team1_players # Set Team 1 names
    globals.team0_numbers = [int(number) for number in team0_numbers] # Set
Team 0 numbers
    globals.team1_numbers = [int(number) for number in team1_numbers] # Set
Team 1 numbers

def no_name(team0_players, team1_players):
    """If a player has a number but no name, set name to """""
    for i, _ in enumerate(team0_players):
        with contextlib.suppress(TypeError):
            if math.isnan(team0_players[i]):
                team0_players[i] = ""
    for i, _ in enumerate(team1_players):
        with contextlib.suppress(TypeError):
            if math.isnan(team1_players[i]):
                team1_players[i] = ""
    return team0_players, team1_players

def remove_blanks(team0_players, team1_players, team0_numbers, team1_numbers):
    """Remove blanks values from the player names and numbers"""
    for i in reversed(range(len(team0_players))):
        if team0_players[i] in ['', 'nan'] and math.isnan(team0_numbers[i]):
            team0_numbers.pop(i)
            team0_players.pop(i)
        if team1_players[i] in ['', 'nan'] and math.isnan(team1_numbers[i]):
            team1_numbers.pop(i)
            team1_players.pop(i)
    return team0_players, team1_players, team0_numbers, team1_numbers


def get_players(team_0, team_1):
    """Retrieve the player names from the cells in the excel sheet"""
    try:
        team_0 = [team_0[3][4], team_0[1][8], team_0[3][8], team_0[5][8],
team_0[1][12], team_0[3][12], team_0[5][12], team_0[2][16], team_0[4][16],
team_0[1][20], team_0[3][20], team_0[5][20], team_0[1][24], team_0[3][24],
team_0[5][24], team_0[0][27], team_0[1][27], team_0[2][27], team_0[3][27],
team_0[4][27], team_0[5][27], team_0[6][27], team_0[0][30], team_0[1][30],
team_0[2][30], team_0[3][30], team_0[4][30], team_0[5][30], team_0[6][30],
team_0[0][33], team_0[1][33], team_0[2][33], team_0[3][33], team_0[4][33],
team_0[5][33], team_0[6][33]]
    except KeyError: # If no names in last row of sheet, row is not imported
and KeyError occurs
        team_0 = [team_0[3][4], team_0[1][8], team_0[3][8], team_0[5][8],
team_0[1][12], team_0[3][12], team_0[5][12], team_0[2][16], team_0[4][16],
team_0[1][20], team_0[3][20], team_0[5][20], team_0[1][24], team_0[3][24],
team_0[5][24], team_0[0][27], team_0[1][27], team_0[2][27], team_0[3][27],
team_0[4][27], team_0[5][27], team_0[6][27], team_0[0][30], team_0[1][30],
team_0[2][30], team_0[3][30], team_0[4][30], team_0[5][30], team_0[6][30], '',
'', '', '', '', '', '']
```

```
    try:
        team_1 = [team_1[3][4], team_1[1][8], team_1[3][8], team_1[5][8],
team_1[1][12], team_1[3][12], team_1[5][12], team_1[2][16], team_1[4][16],
team_1[1][20], team_1[3][20], team_1[5][20], team_1[1][24], team_1[3][24],
team_1[5][24], team_1[0][27], team_1[1][27], team_1[2][27], team_1[3][27],
team_1[4][27], team_1[5][27], team_1[6][27], team_1[0][30], team_1[1][30],
team_1[2][30], team_1[3][30], team_1[4][30], team_1[5][30], team_1[6][30],
team_1[0][33], team_1[1][33], team_1[2][33], team_1[3][33], team_1[4][33],
team_1[5][33], team_1[6][33]]
    except KeyError: # If no names in last row of sheet, row is not imported
and KeyError occurs
        team_1 = [team_1[3][4], team_1[1][8], team_1[3][8], team_1[5][8],
team_1[1][12], team_1[3][12], team_1[5][12], team_1[2][16], team_1[4][16],
team_1[1][20], team_1[3][20], team_1[5][20], team_1[1][24], team_1[3][24],
team_1[5][24], team_1[0][27], team_1[1][27], team_1[2][27], team_1[3][27],
team_1[4][27], team_1[5][27], team_1[6][27], team_1[0][30], team_1[1][30],
team_1[2][30], team_1[3][30], team_1[4][30], team_1[5][30], team_1[6][30], '',
'', '', '', '', '', '']

    return team_0, team_1

def get_numbers(team_0, team_1):
    """Retrieve the player numbers from the cells in the excel sheet"""
    team_0 = [team_0[3][3], team_0[1][7], team_0[3][7], team_0[5][7],
team_0[1][11], team_0[3][11], team_0[5][11], team_0[2][15], team_0[4][15],
team_0[1][19], team_0[3][19], team_0[5][19], team_0[1][23], team_0[3][23],
team_0[5][23], team_0[0][26], team_0[1][26], team_0[2][26], team_0[3][26],
team_0[4][26], team_0[5][26], team_0[6][26], team_0[0][29], team_0[1][29],
team_0[2][29], team_0[3][29], team_0[4][29], team_0[5][29], team_0[6][29],
team_0[0][32], team_0[1][32], team_0[2][32], team_0[3][32], team_0[4][32],
team_0[5][32], team_0[6][32]]
    team_1 = [team_1[3][3], team_1[1][7], team_1[3][7], team_1[5][7],
team_1[1][11], team_1[3][11], team_1[5][11], team_1[2][15], team_1[4][15],
team_1[1][19], team_1[3][19], team_1[5][19], team_1[1][23], team_1[3][23],
team_1[5][23], team_1[0][26], team_1[1][26], team_1[2][26], team_1[3][26],
team_1[4][26], team_1[5][26], team_1[6][26], team_1[0][29], team_1[1][29],
team_1[2][29], team_1[3][29], team_1[4][29], team_1[5][29], team_1[6][29],
team_1[0][32], team_1[1][32], team_1[2][32], team_1[3][32], team_1[4][32],
team_1[5][32], team_1[6][32]]

    return team_0, team_1

def get_events(setup):
    """Retrieve the event names from the cells in the excel sheet"""
    return [event for event in setup[3].values if type(event) == str]
```

k. Gui.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#----------------------------------------------------------------------------
# Author  : Seán McTiernan
# ---------------------------------------------------------------------------
""" This file contains functions to create and run the GUI window."""
# ---------------------------------------------------------------------------
# Imports
import threading                # Contains method to create threads
import PySimpleGUI as sg        # Functions to run the GUI
from datetime import datetime   # Functions to get current time
from os import listdir, remove  # Functions to find and delete files
from os.path import isfile, join# Functions to check files and combine strings
into a path

import drawing                  # Functions to draw the figure on the GUI
canvas
import export                   # Functions to export the match data

import globals                  # Contains global variables
# ---------------------------------------------------------------------------
def gui_setup():
    w, h = get_w_h()
    image_col = [[sg.Image(r'images/recording.png', key='-REC-')]] # Load
recording notification image

    # Define the window's contents i.e., layout
    data_col = [ # Column with navigation buttons and canvas
        [   sg.Push(),
            sg.Column(image_col, element_justification='c'),
            sg.Push(),
            sg.Button('Team Totals',enable_events=True, key='-TOTALS-',
font='Helvetica 16', expand_x=True),
            sg.Button(globals.team_names[0],enable_events=True, key='-TEAM0-',
font='Helvetica 16', expand_x=True),
            sg.Button(globals.team_names[1],enable_events=True, key='-TEAM1-',
font='Helvetica 16', expand_x=True),
            sg.Button('Export Data', enable_events=True, key='-EXPORT-',
font='Helvetica 16', expand_x=True),
            sg.Button('Start Match', enable_events=True, key='-TIMING-',
font='Helvetica 16', expand_x=True),
            sg.Push()],
        [sg.Push(), sg.Text("", font=('calibri',25), justification='center',
auto_size_text=True, key='-TITLE-'),sg.Push()],
        [sg.Push(), sg.Canvas(key='-CANVAS-', pad=(20,20)), sg.Push()]
    ]
    com_feed = [
        [sg.Multiline("", key="-FEED-", text_color='black',
background_color='white', size=(190, h-45), autoscroll=True)]
    ]
    com_feed_col = [ # Column with command feed and clock
        [sg.Text('00:00', key='-TIME-', font='System 18',
background_color='red', expand_y=True)],
        [sg.Text("Command Feed", font=('calibri',19), justification='center',
auto_size_text=True)],
        [sg.Frame("",com_feed, size=(200,h-35), background_color='white',
title_color='black', border_width=0, title_location='n')]]

    data_layout = [ # combine columns
        [sg.Push(),sg.Column(data_col, element_justification='c'),
sg.Column(com_feed_col, element_justification='c'),sg.Push()]
        ]

    # Create window
    globals.window = sg.Window('Stats Tracker', data_layout, resizable = True,
finalize=True, return_keyboard_events=True)
    globals.window.Maximize()

    # Set window title
    title = "Team Totals"
    globals.window['-TITLE-'].update(title)

    globals.window['-REC-'].update(visible=False)

def get_w_h():
    blank = sg.Window("",layout=[], alpha_channel=0) # Create a blank window
    w, h = blank.get_screen_size() # Use blank window to get screen height and
width
    blank.close() # Close blank window
    return w, h
```

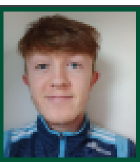
```
def gui():
    # Draw figure on canvas
    drawing.redraw_figure()

    while True:
        event, _ = globals.window.read(timeout=3000) # Read window for events
and timeout after 3 seconds
        match event:
            case " ": # Recording button
                if not globals.start_recording:
                    globals.start_recording = True
                    globals.window['-REC-'].update(visible=True)
                elif not globals.end_recording:
                    globals.end_recording = True
                    globals.window['-REC-'].update(visible=False)
            case '__TIMEOUT__': # Window read timeout after 3 seconds
                pass
            case sg.WIN_CLOSED:
                files = [f for f in listdir('text_files') if
isfile(join('text_files', f))]
                for file in files:
                    remove(f'text_files/{file}')
                exit()
            case '-TOTALS-': # Team totals button, change view to team totals
                title = "Team Totals"
                globals.window['-TITLE-'].update(title)
                globals.view = globals.views[0]
            case '-TEAM0-': # Team 0 button, change view to team 0
                title = globals.team_names[0]
                globals.window['-TITLE-'].update(title)
                globals.view = globals.views[1]
            case '-TEAM1-': # Team 1 button, change view to team 1
                title = globals.team_names[1]
                globals.window['-TITLE-'].update(title)
                globals.view = globals.views[2]
            case '-TIMING-': # Timing button, updates latest timing action
                if not globals.start_match: # Start Match
                    globals.start_match = True
                    globals.window['-TIME-
'].Widget.configure(background='green')
                    globals.window['-TIMING-'].update("Half Time")
                    now = datetime.now()
                    globals.start_hour = now.hour
                    globals.start_minute = now.minute
                    globals.start_second = now.second
                    d = threading.Thread(target=drawing.run_clock,
daemon=True)
                    d.start()
                elif not globals.half_time_begin: # Half Time
                    globals.window['-TIME-
'].Widget.configure(background='red')
                    globals.window['-TIMING-'].update("Start Second Half")
                    globals.half_time_begin = True
                elif not globals.half_time_end: # Start Second Half
                    globals.window['-TIME-
'].Widget.configure(background='green')
                    globals.window['-TIMING-'].update("Full Time")
                    globals.half_time_end = True
                    now = datetime.now()
                    globals.second_half_hour = now.hour
                    globals.second_half_minute = now.minute
                    globals.second_half_second = now.second
                else: # Full Time
                    globals.full_time = True
                    globals.window['-TIME-
'].Widget.configure(background='red')
                    globals.window['-TIMING-'].update(disabled=True)
            case '-EXPORT-': # Export Data button
                c = threading.Thread(target=export.export_data, daemon=True)
                c.start()
            case _:
                continue

        drawing.redraw_figure() # Redraw figure on canvas
        b = threading.Thread(target=drawing.comm_feed, daemon=True) # New
thread to update command feed
        b.start()
```

I globals.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#----------------------------------------------------------------------------
# Author  : Seán McTiernan
# ---------------------------------------------------------------------------
""" This file contains the globals variables needed by the program.
    They contain sample values so that the program can be used without
setup.xls"""
# ---------------------------------------------------------------------------

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]
team0_players = [""]*32
team1_players = [""]*32

# Lists of player numbers
team0_numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]
team1_numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]

# List of event names
event_names = ["goal", "point", "wide", "tackle", "block", "free conceded",
"save"]

# List of team names
team_names = ["A", "B"]
team_nicknames = ["Blues", "Reds"]

# Match Title
match_title = "Munster Final"

# List of views for canvas
views = ["Team Totals", team_names[0], team_names[1]]

# Set default view
view = views[0]

# Holds latest command for undo function
commands = []

# GUI window so that it can be accessed from all files and threads
window = None
```

```python
# Recording flags shared between threads
start_recording = False
end_recording = False

# Figure on canvas
fig_agg = None

# Dataframes
df, df0, df1 = None, None, None

# half Length
half_length = 30

# Time of match start
start_hour = 0
start_minute = 0
start_second = 0

# Time of second half start
second_half_hour = 0
second_half_minute = 0
second_half_second = 0

# Flags for time events
start_match = False
half_time_begin = False
half_time_end = False
full_time = False
```

Appendix C: Poster



Sports Stats Collection by Voice Input

Department of Electronic and Computer Engineering

Seán McTiernan

Electronic and Computer Engineering

Introduction

The right statistics in the right hands can be the difference between winning and losing a game. These statistics are currently recorded either with pen and paper or on a mobile or tablet app. The drawback of these systems is that the user must look away from the game in order to input an event, during which time they may miss a subsequent event.

A system in which the user can input statistics by voice input overcomes this flaw in the other systems and also has the benefit of being more intuitive.

Aim

The aim of this project is to create a program to allow users to record stats in real time without looking away from the game.

The program must handle substitutions, timing and when the user does not know which player completed a certain action.

It must allow users to undo commands quickly if they have made a mistake so that they don't miss the next statistic to be recorded.

It is also important to display the data in a clear, easy to read, intuitive way to create a seamless user experience.

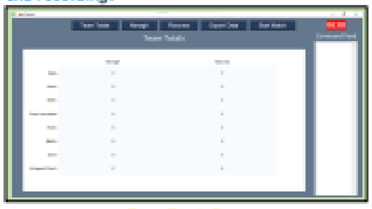
Method

This program is written in Python. It uses the Google Speech Recognition API for speech recognition, NumPy and Pandas for data operations and PySimpleGUI to make the GUI. It also uses Matplotlib and Seaborn to create the data displays.

Players Input Page

On the above screen, the user inputs player names, numbers and positions, along with team names and which statistics they want to track.

The program then starts with the below screen showing team totals. The user presses the spacebar to begin recording, speaks a command, and presses the spacebar again to end recording.



Team Totals Page

The recording is then passed to the speech recognition API which returns it's top five guesses as to what was said.

A function then searches each of theses guesses for event names, team names and player names and numbers.

The team totals dataframe and the single team dataframe for the team in question are updated.

Then the data display in the GUI window is updated to show the new values.

The command is added to the command feed on the side of the screen where the user can check that the command was understood correctly.

The user can navigate to other pages showing the individual player totals for each team.

The user can also export the data to an excel file to save for future analysis and can export a timeline of the game.

Results

The program can accurately record statistics from voice input. It works well with short commands like "Goal Cork 7", and works even better with longer sentences like "Number 7 for Cork scored a goal". This is because Google Speech Recognition uses the context of the sentence when predicting words so is more likely to predict "goal" if it comes after "scored".

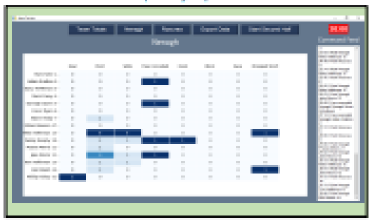
The program tracks the game clock and can handle half time, full time etc. and puts the time on every command in the command feed.

It can also handle substitutions, moving players off the display when they are taken off and vice versa.

If the player who performed an action is unknown, the user must simply say "Unknown" in place of the player name and the event will only count towards the team total and not to any player's total.

By saying "Undo", the user can wipe the previous command from the display if they have made a mistake. If player names have been input they can be used in place of the player's number, which can also still be used. If the user uses the player's name in a command, they need not specify the team, so long as there aren't two players with the same name.

Teams can be referred to by a nickname if their name is too long or unwieldy. The data in the displays is shown in a heatmap format which allows the user to quickly spot the outliers.



Player Totals Page with sample data

The program sounds a warning when a command has not been understood, prompting the user to repeat it. The audible warning means the user doesn't have to look away from the game to know what has happened.

There is also the option to send the data to a manager/coach etc. when exporting.

Team Totals Page with sample data

Conclusion and personal reflection

This program is able to collect sports statistics from voice input in real-time to a high degree of accuracy and can deal with all of the possible events that occur during a game.

I believe it could be very useful to teams who are currently using pen and paper or mobile application systems.

There are very few similar programs which is surprising given the potential for such a system to improve on previous ones.

Future work could involve performing detailed analysis on the data to provide insights for the management team and the ability to compare the current match with previous matches in real time.

The key skills developed over the course of this project were technical skills and creativity.

Acknowledgements

I would like to thank my supervisor, Dr. Hussain Mahdi, for his time and guidance throughout the year.

Also, thanks to friends and family for their advice and support.

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

Figure 15 Poster