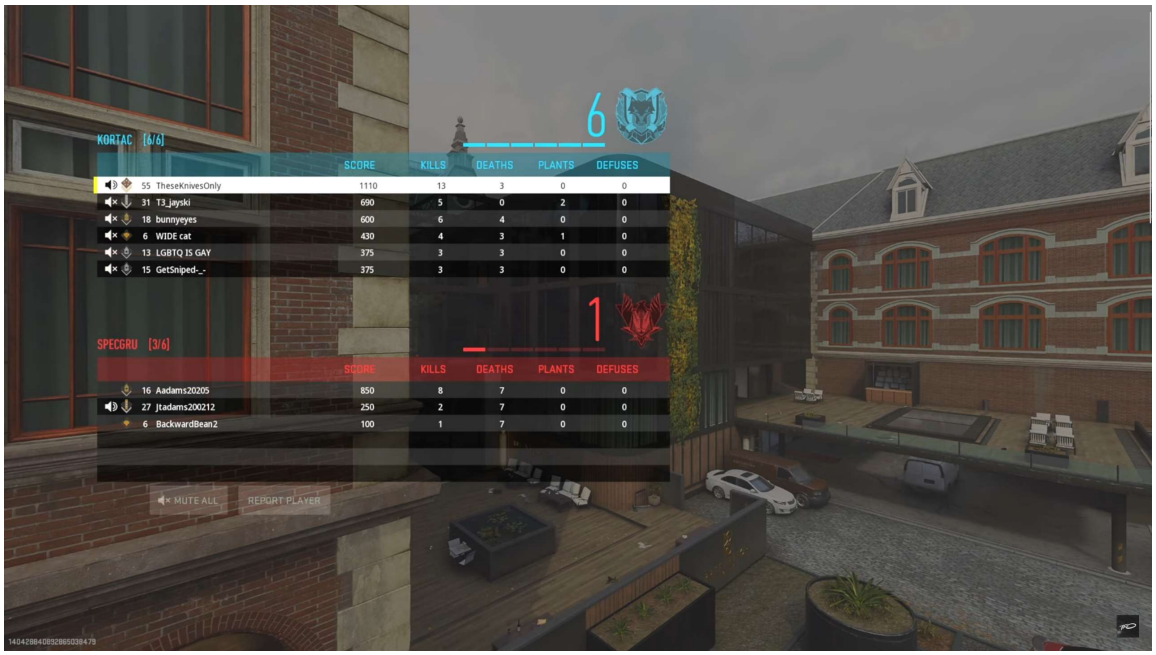
Cod Stat Tracker

What is the problem that this project aims to solve?

The new Call of Duty game doesn't have a stats page
Every game they have released for the past decade has included one. As someone who is obsessed with stats, this wasn't going to work for me, leaving me with one option.
Make my own

Step 1: Collecting Data

Unfortunately after searching online, there isn't even an API to pull data from like some past games have had. The only source for the data that I need is going to be the brief scoreboard that shows up at the end of each match. Below is an example that I will use to test the program as we go along



I decided that using optical character recognition would be the best route to get our stats from the scoreboard. After testing a few libraries I landed on EasyOCR as the best option to use.
First we need to use the PyAutoGUI library to capture a screenshot of the scoreboard.

In [ ]:
```
import pyautogui as pg

# Capture screenshot and save
screenshot = pg.screenshot()
screenshot.save('scoreboard_screenshot.png')
```

Then using the Python Imaging Library, we can crop the image down to the useful bits.
On the scoreboard above, you can see that your own row on the scoreboard will always be highlighted in white.
The way it will work is to crop the image down to just our own section, but since unfortunately I'm not at the top of the scoreboard every time, we'll have to account for that bit moving up and down on the screen from game to game.
The solution is to convert the image into a NumPy array and find the first white pixel on the vertical axis.

In [ ]:
```
from PIL import Image, ImageOps, ImageEnhance, ImageDraw
import numpy as np

#Take image and crop into a vertical row of pixels on the left of the scoreboard
imgOriginal = Image.open('scoreboard_screenshot.png')

imgFindPlayer = imgOriginal.crop((330,330,331,800))

#Convert string of pixels into an array and find the top and bottom white pixels
na = np.array(imgFindPlayer)
Y,X= np.where(np.all(na==[255,255,255],axis=2))
top, bottom = Y[0], Y[-1]

#Crop original image to just the player's row
imgNums = imgOriginal.crop((600,330+top,1450,bottom+330))
imgNums.save('scoreboard_cropped.png')
```

This will all result in an image like this.

Now we just need to do the same thing with the column headers on the scoreboard since different game modes show different stats at the end of the game. These are in the same place every game so we can just do a crop of the area we need.

In [ ]:
```
#Crop image for column names
imgCols = imgOriginal.crop((760,330,1450,380))
imgCols.save('scoreboard_cropped_columns.png')
```

Which will look like this

Step 2: Extracting Data Using EasyOCR

After playing around with the EasyOCR parameters, I found that these settings seemed to work consistently.

In [ ]:
```
#Scan image for text. Save as list of values
```

```
playerResult = reader.readtext('scoreboard_cropped.png', detail = 0, text_threshold

#Split the string into a list of values
playerResult = playerResult[0].split()

#Turn the values all into integers
playerResult = [int(i) for i in playerResult]
print(playerResult)

#Scan the columns
columnsResult = reader.readtext('scoreboard_cropped_columns.png', detail = 0, text_

#Due to the columns not being on a solid background, a few phantom characters would
#To fix this we remove all of the leading spaces and non alphanumeric characters
columnsResult = [i.strip() for i in columnsResult if i.strip().isalpha()]
print(columnsResult)
```

```
[1110, 13, 3, 0, 0]
['SCORE', 'KILLS', 'DEATHS', 'PLANTS', 'DEFUSES']
```

Step 3: Storing the Data

Now let's make a dataframe with the data from the image.

In [ ]:
```
import pandas as pd

# Create the dataframe and assign the scanned column names as the columns
statsData = pd.DataFrame(columns=columnsResult)

# Now add in our stats
statsData.loc[len(statsData)+1] = playerResult

statsData.head()
```

Out[ ]:

	SCORE	KILLS	DEATHS	PLANTS	DEFUSES
1	1110	13	3	0	0

Now we need a place to store data over multiple games

In [ ]:
```
# This is a list of all possible columns from different game modes
statColumns = ['DATE', 'SCORE', 'KILLS', 'DEATHS', 'CONFIRMS', 'DENIES', 'TIME', 'A

# Create an empty dataframe with all possible columns
masterData = pd.DataFrame(columns = statColumns)

# Save it as a CSV file
masterData.to_csv('stats.csv')
```

Now whenever we run the program we can open open this csv file and add to it

In [ ]:
```
# Read in the CSV file
masterDate = pd.read_csv('stats.csv')
```

```
# Add data from the statData dataframe to add to our stats
masterData = pd.concat([statsData, masterData], axis=0, ignore_index=True)

# Re-save the CSV file
masterData.to_csv('stats.csv')

masterData
```

Out[ ]:

	SCORE	KILLS	DEATHS	PLANTS	DEFUSES	DATE	CONFIRMS	DENIES	TIME	ASSISTS	LATE
0	1110	13	3	0	0	NaN	NaN	NaN	NaN	NaN	N

After running the script on a few games, we end up with a dataframe like this

In [ ]:
```
masterData
```

Out[ ]:

	SCORE	KILLS	CAPTURES	DEATHS	DEFENDS	TIME	ASSISTS	RATIO	PLANTS	DEFUSES	C
0	6100.0	46.0	8.0	40.0	3.0	NaN	NaN	NaN	NaN	NaN	
1	10770.0	76.0	NaN	14.0	3.0	0.14	NaN	NaN	NaN	NaN	
2	13810.0	105.0	NaN	15.0	5.0	0.24	NaN	NaN	NaN	NaN	
3	1725.0	14.0	NaN	3.0	NaN	NaN	1.0	4.67	NaN	NaN	
4	1195.0	15.0	NaN	6.0	NaN	NaN	1.0	250.00	NaN	NaN	
5	1195.0	15.0	NaN	6.0	NaN	NaN	1.0	250.00	NaN	NaN	
6	1110.0	13.0	NaN	3.0	NaN	NaN	NaN	NaN	0.0	0.0	
7	1110.0	13.0	NaN	3.0	NaN	NaN	NaN	NaN	0.0	0.0	
8	1110.0	13.0	NaN	3.0	NaN	NaN	NaN	NaN	0.0	0.0	
9	1110.0	13.0	NaN	3.0	NaN	NaN	NaN	NaN	0.0	0.0	
10	4215.0	34.0	NaN	10.0	NaN	NaN	NaN	NaN	NaN	NaN	
11	3545.0	24.0	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	
12	3450.0	28.0	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	
13	3545.0	24.0	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	

Actually using the script

I created a .py file with all of the code then mapped a key on my keyboard to execute the script. Now at the end of a game I just press the button and my stats get updated!

Unfortunately, or thankfully rather, they added a proper stats page to the game shortly after I made this script, so I never really ended up using it.

It was still a fun project to make though.
If they hadn't added this feature in game, I had planned on adjusting the code to work on any screen resolution, because currently since the crops are being done with pixel counts and not percentages of screen real estate, it only works for a 1440p monitor.