Check point report

Link to GitHub: https://github.com/zezhouj/si507_final.git

URLs for data:

https://www.goal.com/en-us/live-scores

https://www.goal.com/en-us/team/portugal/8gxg8f7p9299jbrz30u8bsc7g

…….

Format: HTML

how you accessed the data, and whether caching was used:

Directly scraping HTML string form the website. Utilize Python library Beautiful Soup for pulling data out of HTML files. Caching is applied for some fix information, like team members. However, the match details will need to be updated every day.

Summary of data (description of records, including important fields/attributes of each record for the purpose of and what they represent)

```
['FT', '3', 'France', '-', '1', 'Poland', 'FT', '3', 'England', '-', '0', 'Senegal', 'PEN', '1', 'Japan', '-'
, '1', 'Croatia', '(PEN 0 - 2)', '0', 'Brazil', '-', '0', 'South Korea']
```

These are shaped data extract form a HTML file, which shown the result recent matches. If the match is not played yet, then it would not have an indicator like “FT”, or “PEN” in front of the score and country.

In addition, there will be also URL extension retrieved (shown in snapshot below), for further data scraping. For example, detail of a specific match or player in a certain team.

```
['FT', '3', 'France', '-', '1', 'Poland']
/en-us/match/france-v-poland/21c4pmd61gkqhqo8uxwrk7nfu
['FT', '3', 'England', '-', '0', 'Senegal']
/en-us/match/england-v-senegal/3yowhdtoklrukfkubuqxm3gx6
['PEN', '1', 'Japan', '-', '1', 'Croatia', '(PEN 0 - 2)']
/en-us/match/japan-v-croatia/9t63lrnr7aia41uo9xyk75ld6
```

a snapshot of a chunk of your cache file:

1205_post_match.txt


```
src="https://cdn.sportfeeds.io/soccer/images/teams/30x30/uuid_e70zl10x0ayu7y10ry0wi465a.png?
application=goal.com" decoding="async" loading="lazy" width="30" height="30" alt="teams"/
>USWNT</a></li><li><a href="https://www.goal.com/en-us/uefa-womens-champions-league/
55hcphd1ccc6eai1ms77460on">UWCL</a></li><li><a href="https://www.instagram.com/indivisa/"
rel="noreferrer noopener" target="_blank">INDIVISA</a></li></ul></li></ul></li><li
data-extendable="true"><button>LIFESTYLE <i data-icon="table-down"></i></button><ul
data-level="2"><li><button>Content</button><ul data-level="3"><li><a href="https://www.goal.
com/en-us/category/culture/1/edzsmxzmp4y81pp4ozi9lrdeh">Culture News</a></li><li><a
href="https://www.goal.com/en-us/category/features-opinions/1/
rpux9itsa1271baslypn8ifq9">Feature Stories</a></li><li><a href="https://www.goal.com/en-us/
category/vice-x-goal/1/blt877d97bf017449a0">Vice x GOAL</a></li></ul></li><li><button>Guides</
button><ul data-level="3"><li><a href="https://www.goal.com/en-us/category/fifa-23/1/
bltbb26aeddc9368957">FIFA 23</a></li><li><a href="https://www.goal.com/en-us/category/
uk-shopping/1/jay1lq18ea7j1bks90cjvzor9">Buyers&#x27; Guides</a></li></ul></
li><li><button>Goal Network</button><ul data-level="3"><li><a href="https://mundialmag.com/"
rel="noreferrer noopener" target="_blank">MUNDIAL</a></li><li><a href="https://www.instagram.
com/nxgn_football/" rel="noreferrer noopener" target="_blank">NXGN</a></li><li><a
href="https://www.instagram.com/indivisa/">INDIVISA</a></li></ul></li><li><button>Follow &
amp; Subscribe</button><ul data-level="3"><li><a href="https://www.facebook.com/GOAL/"
rel="noreferrer noopener" target="_blank">Facebook</a></li><li><a href="https://twitter.com/
goal" rel="noreferrer noopener" target="_blank">Twitter</a></li><li><a href="https://www.
```


Indication of the parts of your code that implement caching

```
loaded_cache = open_cache()
if loaded_cache:
    print("cached html loaded\n")
else:
    print("cache load failed\n")
```

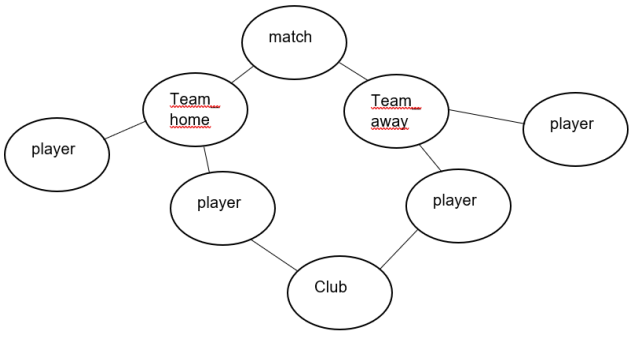
A link to a brief video showing the difference between running your data access program with and without caching

In my case, caching and noncaching difference will be available after 80% completeness of the program. Basically, I will store the fixed information in cache, like player info and team info. But for daily matches, user should refresh the program to get the new data for new matches day by day.

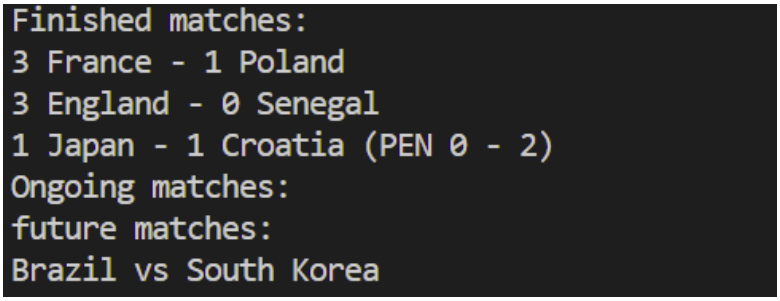
Description of the graphs or trees you plan to organize your data into

I plan to use a tree liked data structure to store all the data of matches, teams, and players. In addition, different player can be related through the same football club that they attended during their career. The below graph shows the tree liked structure planned to use.

Description of how you will plan to assemble data into those graphs or trees



Screenshots showing some progress



High-level, plain-English description of the user-facing capabilities of your project—what options does the user have for selecting and displaying data?

Firstly, this project focuses on the ongoing World Cup. And it will show the recent matches. For finished matches, it will have the final score of two teams. For ongoing matches, it will show the time in minutes also the live score. For the unhappen matches, only the names of two teams will be displayed.

Secondly, user would have options to select one of the matches interested in. Program will scrap website again based on the URL extension of the match. All the match details will be shown.

Also, user could select a team interested in. then all players in the team will be displayed. Still, user can have the information of any player in the shown list, which is age, position, etc.

In addition, user could have a special option, which is display the relation of two players. In format of a chain. E.g., player1--team(club)—match—team(club)—player2

Interactive and presentation technologies you plan to use (e.g., Flask, Plotly, command line prompts)

In current plan, command line prompts will be present. However, if time is available, I will try Flask to have a decent interface.