Approach:

For this, I used the Django framework and Python as these are the resources I currently have because I do not want to spend time setting up a new environment using my work laptop as I am not at home at the moment.

To make it short, I made a Django application here that fetches 100 board games from the website “boardgamegeek.com”, putting them in a list that will be used as data for the endpoints that I have made for outputting the similarity scores between users in that website (the similarity being ratings on games). You can read more in the assumptions.txt that I have also provided in this repository.

My application has these 3 main core files: fetch_games.py, utils.py, and views.py.

I separated the logic as follows:

fetch_games.py is the start of the data gathering. This is the method that gets the board games. The method I created there will be used by the utils.py but for checking/debugging purposes, it can also be run to output the games it fetches (in a terminal).

utils.py is the brain of my endpoints. This is where most of the logic is abstracted. There’s a few hardcoded stuff that might be a problem if boardgamegeek changed its web templates like how I fetch the rating columns (class_='collection_bggrating') and the rows (tr[id^='row_']) . This is also where we use numpy’s cosine similarity logic:

$$\text{similarity} = \frac{A \cdot B}{||A|| \cdot ||B||}$$

views.py is what I used in dealing with the outputs. These return Django responses (similar to json responses) to display a somewhat human readable result. I also have 7 test users that exist on the website itself to produce the outputs I have attached to my README in the github.

All in all, I don’t think I needed to make a frontend design that takes user inputs since I think the endpoints are sufficient enough to demonstrate my knowledge on coding. I do believe in myself and I hope you guys do too!