Initially, I knew I was really interested in making a Flask application. Whenever I do more time-intensive coding projects, I really love having something to show for them. My favorite part of any project is sharing it and showing off my hard work. Thus, a Flask application seemed like the best option for me because it would allow me to show off anything I wanted. I am also working on a research project at Mudd right now which has involved a lot of web development and I have learned a lot over the course of the past year, so I wanted to incorporate that into my final project as well. Granted, most of the work we have done is in React (and thus javascript), the html/css skills were still helpful in working on this project! Once I had decided I wanted to make a Flask application, I needed to figure out what I wanted it to actually do. Another aspect of my studies/career that is very important to/enjoyable for me is data science. While this term is almost incomprehensibly broad, what I mean to say is that I really love working with data, and I especially enjoyed the work we did with data for our prediction models in class. After scouring through countless datasets, I landed on the Netflix and Disney+ library datasets on Kaggle. I decided then that a Flask application which can recommend Netflix movies and shows would be what I would work on for my project.

While the idea sounds relatively simple, I soon found that this was a much larger task than I thought it would be. I decided to tackle the front-end first, and utilized css templates and some basic html formatting to get a basic page that I could fit the back-end to. Once I got the front end working in the most basic and tedious way possible, I worked on making it more efficient, “variable-ize” as Prof Dodds might say. Satisfied with that, I moved on to the back-end. Initially I worked on everything for the backend in a completely separate set of python files so that I could make sure everything was working on its own before trying to put it all together. From what we learned in class and some of my own experience in other data science classes/projects, I decided to use the TF-IDF (Term Frequency-Inverse Document Frequency) score to vectorize the movies/shows, which is the frequency of a word occurring in a document, down-weighted by the number of documents in which it occurs. This score is really useful in analysis of this type of data because it reduces the importance of words that occur frequently in plot overviews and therefore, their significance in computing the final similarity score. Since the cosine similarity is independent of the magnitude of the vectors it is comparing, I used it as the metric for the similarity score. In my initial recommendation system (get_reccomendations(title, cosine_sim = cosine_sim) in the code) I calculated the pairwise similarity scores (cosine similarity) of all movies/shows with the movie/show that is taken as the input for the function, sorted the movies/shows based on that similarity score, and return the top 9 most similar movies/shows. The reason I chose to only return 9 movies/shows was purely for aesthetic reasons, as the 3x3 grid of movies looked very nice and symmetrical on the front-end. To make the recommendations a little better (get_recommendations_new(title, cosine_sim = cosine_sim) in the code), I decided to specify which features were the most important and only recommend based on those because I had a hard time believing that anyone just really wanted 140 minute movies as opposed to more similarity in director, cast, plot, and so on. The system worked really well, and I was eager to add a bit more functionality to it (as I was only really making a few sklearn calls). To make the front-end even nicer/more interactive, I decided to find a way to incorporate the posters/cover images. At first, I tried to do this while iterating the back-end in, but found that far too difficult so instead I worked on that in a separate set of files as well. After a long long time I was able to write a script that downloaded images from the IMDb api and then

displayed them accordingly for each recommendation on the front-end. At this point (after a lot of debugging), the system was capable of taking a user input, finding the 9 most similar movies/shows on Netflix, and displaying these recommendations with their titles, durations, and posters. While I was satisfied with this functionality, I decided something like a Google “I’m Feeling Lucky” button might be a fun feature to add. This feature pulled 9 random samples from the data set and then displayed them in the same way that the recommendations are displayed.

While I am incredibly satisfied with the functionality of the current iteration of my Flask application, I recognize that there is always room for improvement. Initially, I was debating between using Disney+ and Netflix (and even came across a data set for Hulu), and I think it would be cool to be able to search multiple streaming services at the same time. I also ran into an issue for some of the movie/show posters where some didn't have posters or they were not available on IMDb, so I would definitely look for a solution to this problem if I had more time.

In retrospect, I think I probably should have saved most of the front-end work for last, or at least built it up as I increased the functionality of the back-end as I had a lot of trouble integrating the two while also fixing bugs. Granted, I utilized test files a lot and I found that to be a very easy way to test and fix code that was buried deep in a less accessible file. Overall, I am pretty happy with how everything went and turned out!