CS 411 Final Report

There were a good amount of differences between our original project plan and what we actually ended up designing, with a lot of the changes due to the original plans being very ambitious, but hard to actually implement. We originally wanted it to be a course browsing tool like the GPA grade disparity tool that currently exists, but we wanted to also tie in the data about professors' ratings and which professors are ranked as excellent.

We were still able to do that in our final app which showed the best courses to take based on professors' ratings and GPAs, but unfortunately we weren't able to make the input with a sliding bar that would allow students to choose how much they would want to weight the importance on professor rating and weight the importance on higher GPAs. Due to this, our project wasn't as useful because although it recommended the best courses based on ratings and GPAs, it didn't have the flexibility in input for the users which would have been ideal.

Along with discussing functionalities we ended up removing from the original plan, we added some which was the running list functionality which allowed users to have the most updated information stored for their use. This allowed them to add a course to their list if it was just announced or delete a course from the list if they're not interested in taking it anymore.

In terms of the schema and data sources, they ended up being pretty similar to what we originally planned out. We planned on using Professor Wade's dataset on classes and GPAs, and we ended up using that. For professors' ratings, we weren't able to use the full RateMyProfessor dataset with all the breakdowns in ratings because they didn't have enough data for everyone, so we had to replicate missing data with a python script.

For the changes from the actual implemented design and the original ER diagram, we ended up making everything the same as the original diagram. If we were to continue our application and make it better, then we would have included the full RateMyProfessor dataset which has breakdowns of whether professors were rated in categories as easy to understand and tough grader. This dataset would have replaced the Rating relation which connected to the Instructor relation, and basically would have been a better version of the Rating relation.

The advanced functions we had were to find the "best" 400 courses and easiest CS 400 courses. We did this because a lot of people struggle to find a manageable 400 level course for advanced electives, and since this is a CS class, and a lot of CS majors/minors need to take other CS 400 level courses, we thought this would be helpful for them.

Some of the technical difficulties we had in this project related to the search, the advanced queries, and the data set. For the difficulties with the search, it was not loading in the frontend. Although the SQL for the search was processing properly in the

SQL workbench, there was a bug that wasn't allowing the results to show in the frontend. We tried to fix this bug during multiple stages, but unfortunately weren't able to figure it out, so we would suggest that the search is implemented very early in the project, maybe even before the remaining CRUD functionalities. With both of our advanced queries, we faced difficulty in reloading the page to display the correct advanced query results. When we debugged this issue, the advanced query results were being properly loaded into a variable for rendering the webpage template. The issue arose because the webpage would be reloaded once with the advanced query results and then immediately reloaded again with the original display of courses whenever the advanced query buttons were clicked. We were able to fix this issue by saving the advanced query items globally when the buttons were clicked and having a flag for when the button was clicked or not. This allowed us to properly load the correct items into the webpage template on each desired button click. Another problem we had was that we did not have enough data for the student ratings, since we wanted to use "ratemyproffesor.com" ratings. Most of them didn't have enough data for UIUC professors. We got around this by writing a simple Python script that replicated the data in a somewhat meaningful way, ensuring the application still made sense. While uploading our data to our tables, we used '.csv' formatted data. Unfortunately, the data import wizard on MySQL Workbench did not work fully. We had a total of 3000 rows but only 250 of them were going through. This was solved by using another app called "DBeaver", by using this app's import wizard.

For ways to improve this application in the future, we would want to implement all of the features that we talked about from our original plan with the ways to choose how much you value professors' rankings and GPAs. Additionally we would want to include more filters for majors and which hundred level classes you want displayed. Another way to make this application even more useful is to include more information about the class when you click on the class name, so that the student doesn't have to search up information about the class separately when the name is shown. One thing that would be good to fix on the backend would be to optimize the queries even better and have the data ordered in a way that would be most helpful based on people's most popular filters which would typically be filtered by department first and then course number. Having the data ordered in the database already by department and then course number would be ideal.

The work was divided equally with people working together on a lot of parts, but we put people in charge of specific areas so that everyone would feel a sense of responsibility to the project, although most areas were worked on by multiple people. Akshay took the lead on the backend for the application while Onat and Sahithi took a lead on the frontend and database elements.