Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).

- One change we made to our original proposal for our final project was getting rid of the interest factor. We found that there was no great way to weigh how one's interests would recommend a course over a measurable factor, such as GPA, Professor weighting, or credit hours, so we decided to narrow our scope to these three. Further, we also didn't implement a functionality where one recommendation could be used as a basis for subsequent recommendations, as this seemed to be overcomplicating a more simple and useful tool. Lastly, we did not create a prospective schedule creation feature, and simply honed in on creating a course recommendation tool. We are quite pleased with this result, as our tool is more objective while still providing the user valuable input to factors that allow them to sufficiently tailor their course preferences.

Discuss what you think your application achieved or failed to achieve regarding its usefulness.

- We think one aspect our project failed to achieve was visualizations. In our vision, we intended to display all of their courses in a prospective schedule to aid the user's experience. However, we found this to be a clunky concept that was entirely different from the main functionality of the tool. While this certainly would have been a cool feature to implement, we decided to simply focus on refining the functionality and effectiveness of our recommendation tool without generating a schedule. We believe that our application achieved all of the functionalities we intended it to have. For instance, we successfully added all CRUD operations and allowed the user to have a functional way to receive course recommendations.

Discuss if you change the schema or source of the data for your application

- We ended up using the exact sources for data that we initially planned on using, being "Rate My Professor," "UIUC GPA Disparity," and "Course Catalog."

Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?

- The original design was more minimalist, however, we were missing a lot of foreign keys, such as NetId and SectionId for our Courses, which we added within our database to several functions. We also added a Prerequisite table, and an Enrollments table to help optimize our queries and simplify our overall database design. By decomposing these tables, we also needed to re-establish some of

our existing relationships, notably updating from one-to-many to one-to-one. This meant that we added a cascade to delete from the Preference and Enrollments tables in relation to Users to ensure a 1-to-1 relation. Overall, further decomposing and modifying our database design, resulted in a more optimal operation and cut some of our advanced queries in half, which is a more suitable design.

Discuss what functionalities you added or removed. Why?

- For the most part, we maintained almost all of the functionalities we planned. One thing we ended up removing was the preference for online or in-person due to inconsistencies between different data sources, as well as complications with differences between Sections and Professors. This is a feature we look to implement in the near future, however with time constraints we deemed it something we could not implement to the best of our ability, and chose to focus on refining other functionalities.

Explain how you think your advanced database programs complement your application.

- Our advanced database programs demonstrate a complete example of creating a user, with their unique preferences, which invokes the internal queries which filter the data to provide tailored recommendations. Further. In creating each user, these users are now an existing part of the database, so they can update their preferences without having to recreate themselves as users, which makes these advanced queries relevant to unlocking the ability to use other queries in our database.

Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.

- The first issue we encountered was with our CSV files, where we had a lot of inconsistencies in the way data was formatted. Because of this, we had to do a lot of manual cleaning of the data, which took a lot of time, but eventually, we had files that were able to be loaded and properly utilized. A second error we encountered was after we had our CSV files, we found our schema needed modification, so we added more tables and reworked some to include foreign keys to connect the data properly for querying. I think this is a very important part to be prepared for in database implementation, as it is one thing to prepare an initial schema, but things change as you are trying to put all of the pieces together. To develop a properly functioning database-related application, you have to be prepared to reevaluate and make major adjustments to your initial

schema to account for this. Another issue we encountered was attempting to utilize Node JS. We felt we did not have a great understanding of Node JS and how it works, and instead opted to use Flask to integrate our SQL code into the application. When creating a database app, it is very important that everything connects and runs smoothly, and likewise, it is crucial to write programs in languages that you are comfortable with, which is why we chose to use the Python Flask library instead of Node JS. This worked much better for us, but it really depends on user preference. Lastly, we struggled in general with connecting the frontend and backend in the middleware, specifically regarding being able to obtain and utilize user input. To fix this, we used YouTube, and Sofia had experience in full-stack development from her internship which was useful. Overall, it took time but we ended up figuring it all out.

Are there other things that changed comparing the final application with the original proposal?

- Again besides the removal of the interest factor, the lack of functionality for creating a schedule of potential classes, as well as the feature of subsequent recommendation of courses based on a particular recommendation, we were able to implement the exact vision we initially had for our course recommendation tool. It is our belief that while some of these ideas were ambitious and could have been interesting, these refinements were necessary for a more direct and user-friendly tool, however, these could also be seen as future work to improve/expand on, which leads into the next point.

Describe future work that you think, other than the interface, that the application can improve on

- As mentioned earlier, we ended up removing the preference for online vs in-person classes due to time constraints and technical complications in sorting out inconsistencies between tables and other errors. However, this is a feature that we definitely would like to implement in the near future, and likewise is a clear source for improvement of our application and user experience.

Describe the final division of labor and how well you managed teamwork.

- We divided our work into middleware, front-end, back-end, and GCP. As for who did what, we subdivided each of these into smaller pieces, and tackled them one by one, so we each got exposure to all of the different stages of development. Further, we tested our programs together when we had issues with debugging individually. For example, if we made a change to the GCP layout, we would

change the corresponding function in the backend, all the way up to the front end and unit test our app. This ensured that when we added new features, our app still maintained its functionality.