1. **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).**
    - We didn't implement a recommendation function since this functionality didn't really match up with what we had in our databases
    - Other than that, we pretty much have what we outlined in our proposal
2. **Discuss what you think your application achieved or failed to achieve regarding its usefulness.**
    - We achieved a combined view of course/section data along with ratings
    - We didn't manage to achieve crowdsourced data yet, as we haven't launched the product to the public yet
        i. So we had to create some mock data to test our application
    - We believe we were able to create something that is course explorer-esque, meaning we displayed all the necessary information and more information such as GPA and ratings, and we gave users the ability to login/logout as well as add favorites and ratings.
3. **Discuss if you changed the schema or source of the data for your application**
    - We mainly stuck with the schema specified in our raw data
    - We didn't change or add new data sources from what we originally specified in the beginning
    - We did also add some mock data into the favorites, ratings, and users table to simulate people actually using our application
    - However, we did make some minor changes, especially the addition of more primary keys/foreign keys in the Section table.
        i. This helped us out in identifying specific sections (it was more complicated than we originally expected)
4. **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?**
    - We had to change some of the foreign keys/primary keys in the 'sections' table because we later realized that we needed more attributes to help identify the different sections and courses (for filtering, etc).
    - Otherwise we didn't differ from the original design too much.
    - We definitely believe the new design is better because it allows us to single out different sections and courses much more easily.
5. **Discuss what functionalities you added or removed. Why?**
    - We didn't implement a recommendation function since this functionality didn't really match up with what we had in our databases

- We've added new filtering choices to help users look at different classes based on ratings and favorites (with the help of stored procedures, etc), gives users an easier time finding classes

6. **Explain how you think your advanced database programs complement your application.**
    - With our advanced queries, we were able to better filter the information users may want to see.
    - Our triggers, transactions and stored procedures also helped us streamline more complicated queries.
        i. For example a trigger so that when a user wants to add Ratings, we ensure that there are no duplicates with ease.
        ii. Stored procedures can help lower network traffic (since we just need to pass the stored procedure name to the DB)
    - By having well designed constraints, it made identifying different sections and courses much easier. The constraints also made our development process more modular and organized.
7. **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.**
    - Brad:
        i. Our team used flask (python) instead of Node (javaScript), so we couldn't just follow the tutorial given from the course to connect to the GCP. So we had to go the extra mile to figure our way though. Thankfully there were plenty of examples and python is obviously a well documented language.
    - Michael:
        i. I thought that actually turning an initial idea into a full-fledged database design can be a bit tricky. Because you have to have a really clear idea on what your application does and what it can do. We managed to overcome it by outlining all the details of the application and having a clear understanding of our data and the constraints that come with it.
    - Peter:
        i. I actually had some issues setting up and using Git in the beginning. There were some authentication issues since we are accessing the school's github organization. Although I was able to get the issue fixed by re-authenticating and following some tutorials on youtube.

- Benjamin:
    i. I thought that setting up the GCP and inserting all the data (and mock data) was surprisingly difficult. Perhaps there were slight issues with the initial design of the database in which it didn't perfectly fit the data we got.

8. **Are there other things that changed comparing the final application with the original proposal?**
    - We didn't change too much from what we originally envisioned, although we definitely tweaked with some of the original UI/UX design and some slight functionality modifications (filters and some search features).
9. **Describe future work that you think, other than the interface, that the application can improve on**
    - We can definitely add more advanced searches and filtering
    - We can also add recommendations based on past searches
    - We can maybe also add more visualizations and graphs on GPA or Ratings or Favorites to help users find trends and patterns
10. **Describe the final division of labor and how well you managed teamwork.**
    - **Brad**: Mainly worked on the backend/server side also helped out with frontend and SQL
    - **Michael**: Mainly worked on the backend/server side. Helped a lot when it came to setting up GCP.
    - **Peter**: Mainly worked on frontend.
    - **Benjamin**: Helped out with frontend, SQL, and documentation.
    - We believe that we were able to work together well. Everyone understood their roles and helped each other out when needed. Everyone was responsive and collaborated without any major issues. Brad and Michael had more Backend and flask experience so they were tasked to handle that. Peter was well-versed in frontend so he was responsible for that. Benjamin helped out on the frontend and documentation, because he was less familiar with flask.