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).

In our Stage 1 proposal, we initially planned to include a feature for generating recommended multiple different graduation plans for students based on their target graduation year. During development, we prioritized other functionalities to ensure the project remained focused and high-quality within the given timeline. While this specific feature was not implemented, the current system is designed to allow for future integration of such capabilities. There were issues we encountered with regards to getting the correct number of prerequisites or the correct number of major requirements because of problems with scraping, so therefore creating a graduation plan wasn't a feasible solution. Regardless, the graduation plan wouldn't be accurate anyway because of the earlier data scraping issues. Instead, we have made the website a helpful guide for students to create their own academic plans, lookup course information, and identify prerequisite courses and identify the course path through the prerequisite visualizer.

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

Our application successfully achieved its intended usefulness by delivering a comprehensive solution for students to design and manage their customized academic plans. Specifically:

- **Customization and Personalization:** The app generates a personalized lesson plan tailored to individual academic objectives, prior coursework, transfer credits, AP credits.
- **Efficiency**: It provides students with the most efficient route to graduation by considering course prerequisites, graduation requirements, and opportunities for early graduation.
- **Progress Tracking:** With its user-friendly interface, the app helps students track their academic progress, ensuring they stay on course to achieve their degree goals.

Overall, our application has proven to be a valuable tool for students, aligning well with its objectives and addressing key challenges in academic planning.

3. Discuss if you changed the schema or source of the data for your application

After generating the tables for stage 3, those were the same tables that were used to retrieve from the backend to then be displayed in some format in the frontend. In the proposal, we had laid out that the data sources that we will be using are AP score conversions, major requirements, and course prerequisites. The one data source that we didn't use was transferology's transfer credits system as we believed it to be not feasible to include all transfer data sources from multiple universities without adding a significant amount of data to the program. Therefore, the users shall determine the transferability of the courses through transferology and then they can input the course information into the Add Course option. We had also added more data sources, those which will be generated by users over the course of using the application. One of them is the student data table which contains necessary information about students (this is the basis for the login system as well). We also have two data tables with plan details so that students can generate multiple plans and add several courses in each plan. However, in regards to the data source, they have been the same as the ones listed in the proposal (except for the removal of transfer credits) and the data tables adhere to the same standards that were set in the proposal.

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?

The ER diagram was followed throughout the course of the project, both in regards to the generation of tables and developing the functionality of the website. All of the entities were represented as tables, along with the relationship tables for many-to-many relationships (Planned_Course and Prerequisite). In addition, all of the attributes (including the primary and foreign keys) remained the same in the database implementation. We determined that the ER diagram was an accurate representation of how the data should be structured for further processing and therefore it was not modified. The only modifications that were made were to the actual size of the entries. Some of the tables ended up being a lot longer than expected (Requirements and Prerequisites), so those were taken into account when the final product was being developed in order for the user to have less confusion with a large amount of data. The increase in data happened because of the way that the data was scraped from the website, which is something that a future implementation of the project would fix. During the implementation of the website, we realized that the requirements table and the ap_credit tables weren't necessary in their correct form. Therefore, a more suitable design would only keep the CourseName and Score attributes in the ap_credit table and integrate the requirement and course catalog tables for more efficient database organization.

5. Discuss what functionalities you added or removed. Why?

- One of the main functionalities that we added was a prerequisite visualizer. This visualizer uses a react library but significantly builds upon the code for the user to visualize the path of courses they have to take to be able to take the course that they are

interested in. This was added later on as a tool for students to make plans with prerequisites in mind without needing to include a recommendation system
- Another feature that we added was the plan tracker. While we had a version of this indicated in the proposal, there were some changes made in regards to plan organization and course additions to the plan that made it easier for the user to access different plans and make modifications to courses within them.
- A functionality that was removed from the website that was in the proposal was the graduation recommendation system which will modify the expected graduation based on courses taken. Because of issues faced when obtaining data with major requirements and course prerequisites, we determined that it is a feature that can be added in a later implementation once a better data source is found to accomplish the task.

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

- Constraints: Constraints were handled when fetching data using a python file to setup the backend to connect to the front-end. For example, when fetching data regarding AP credits and scores, our constraint was that anything above and including a 3 should be good in a course substitution while anything below that would be a fail. We also mention to the user in the frontend that they've either passed or failed the course so choose another course to substitute
- Transactions: In the page where a student can edit anything related to their academic plans. These functions use `execute_transaction` which ensures that all database operations within the complex JOIN queries are executed as either all succeed or none, maintaining data consistency. For example, when calculating total credits, if any part of joining the Student, Academic_Plan, Planned_Course, and Course_Catalog tables fails, the entire operation is rolled back, preventing partial or incorrect credit calculations. This is particularly important because these queries touch multiple related tables, and transactions ensure that we get consistent results even if other database operations are happening simultaneously.
- Triggers: Trigger is really helpful on the same page in which transactions are also used. When a user updates anything related to their academic plan, the total number of credits get updated. This is really helpful in letting the student know how many courses they have saved in their academic plan and do something actionable with it.
- Stored Procedures: In this case, VerifyPrerequisites checks if a student has planned all the required prerequisite courses before they can take a specific course. Instead of running multiple separate queries from your application, this procedure efficiently handles all the prerequisite verification logic in one go, directly in the database, by counting the total prerequisites for a course and comparing it with the prerequisites the

student has already planned. The procedure uses variables and conditional logic (IF/ELSE) to determine if prerequisites are met, and can return both a boolean result (prerequisites_met) and a list of missing prerequisites, making it more powerful than simple queries.

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.

Harshita Thota - Initially, our project encountered a technical challenge with the database path in DB_PATH, which was originally set as os.path.join(BASE_DIR, '../../doc/database.db'). This caused issues during deployment because the relative path depended on the file structure and varied across environments. To resolve this, we updated DB_PATH to use an absolute path with os.path.join(BASE_DIR, 'data', 'database.db'), ensuring a consistent directory structure. This change allowed the application to reliably locate the database file, regardless of the working directory or environment. Future teams should avoid using relative paths for critical resources and opt for absolute paths tied to a stable project directory.

Karthik Bagavathy - One of the issues that our project encountered was properly developing the prerequisites table and prerequisite path visualizer. One technical issue was that the keyword search would take a very long amount of time given the fact that there are a lot of entries in the prerequisites table. Therefore, the fix that I have implemented uses a query search optimization in the backend so that, as the user types in the course, the querying will be done and the prerequisites will be returned. Another issue that was encountered was the prerequisite path visualizer not displaying in an user friendly manner. This was a limitation of the library that we were using and the fix was to incorporate styling to color code the prerequisites with different colors based on the course level. This, along with modifications to the size of vertices and edges made the visualization a lot more user friendly.

Kundan Mergu - In our development of the AP Credit functionality, we initially faced several key challenges. First, we needed to properly map AP courses and scores to their equivalent university courses, which we solved by creating an endpoint that queries the AP_Credits table to find the corresponding CourseID. Second, we encountered issues with popup handling where they were closing automatically, which we fixed by removing the auto-close logic and implementing proper user-controlled closing mechanisms. Additionally, we needed to handle different scenarios for AP scores - showing a congratulatory message with course selection for scores 3 and above, and showing an error message for scores 1-2. Finally, we integrated new functionality to track students' academic progress by adding features to view total credits and requirements fulfilled, which provides students with a better understanding of their academic standing.

Vashishth Goswami - I faced issues using sqlite for stage 3 indexing which followed to stage 4 as well. And most of our implementation was in sqlite and we figured that we had to change the database to mysql for collision resistance and after changing it to mysql it worked perfectly while facing issues while doing the transition.

8. Are there other things that changed comparing the final application with the original proposal?
   The changes made were regarding data processing, feature development (prerequisite visualizer vs recommendation system), and UI development. All of these were discussed in other sections. However, in regards to the UI aspect of the website, we modified our plan from the original proposal. Instead of having one page with a large visualizer and course recommendation system, we developed multiple different pages each with their own unique functionality. This was done for two reasons. First, it makes the process of developing these pages much easier and allows for smoother work split among team members. Second, the features were distinct enough that we determined it will be a better option to split them into multiple pages with a navigation component as it will be easier for the users to use a specific aspect of the project without getting overwhelmed seeing all of the features at once.

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

Future work on the application could focus on expanding the functionality to provide students with more elective options rather than limiting them to a single recommended course. This enhancement would allow users to view a range of electives that fit their academic plan, providing greater flexibility and personalization. Additionally, incorporating features such as filtering electives by interest areas, difficulty level, or scheduling preferences could further improve the user experience. The app could also include predictive analytics to suggest electives based on trends or career outcomes in addition to academic goals. These improvements would make the app even more valuable for students designing their paths to graduation.

10. Describe the final division of labor and how well you managed teamwork.
    This was the final division of labor that we had written in the proposal:
    1. Kundan Mergu: Frontend Development, UI/UX Design, Data Visualization
    2. Vashishth Goswami: Backend Development, Data Integration, API Requests

3. Karthik Bagavathy: Data Collection, Data Cleaning, Data Analysis
4. Harshita Thota: Project Management, Documentation, Testing, Product Design

There were a few changes that were made overtime in regards to this. Since we had a significant portion of stage 3 to complete (for the regrade) during the deadline for stage 4 checkpoint 1, Karthik completed a larger portion of the data visualization aspect of the project and ensured that the database is connected with correct CRUD operations along with search features development and data management while Kundan completed more of the documentation and the query creation to finish the rest of the requirements in stage 3. Vashisth worked on developing the advanced programs for the website's functionality along with completing the documentation for some of the earlier stages. Harshita worked on core functionalities in the frontend and contributed to the backend code to ensure that the data collection from the database was done efficiently and accurately. Overall, all 4 of the team members worked on significant portions of the project and we were effectively able to modify responsibilities for individual team members as the situation demanded.