Team 41 (Databased) Final Project Report

- 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).

The overall direction of the project has remained the same throughout the process of developing this project, although we did make a few changes as we learned more about databases and our use case over the semester. In our proposal, we did not specify the scope of this project, but it became apparent that this was not feasible because requirements differ from degree program to degree program. Thus, we decided to change the scope to only cater to MCS and MS CS programs in the current iteration of the project.

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

The overall structure of our application stayed the same, but there were a few features that we chose not to focus on (discussed later). Thus, the application did achieve its usefulness in guiding the student in choosing and registering for classes while keeping an eye on the degree progress. The dashboard feature aims to make it easier for the student to keep track of their current enrollments by providing a summary of the breadths and advanced courses completed so far, in addition to the total credits completed. The UI for this feature was made to provide all the info at a glance. The waitlist feature was also very successful in our implementation; These two were the main features we wanted to focus on during the proposal. The student and enrollment data we used were all fabricated, showing the project is functional but only theoretically useful in its current state.

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

The schema and source of the data have stayed quite constant throughout the project. The schema is reflected in the earlier submissions, and the source of the data is the Illinois Course Explorer.

- 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?

Our initial ER diagram consisted of 8 tables. We implemented 6 of them (Courses, Enrollments, Instructors, Reviews, Students, and Waitlist) just as they were described in the diagram. The two we omitted are assistantships and degrees. We decided to omit degrees because we decided to focus on two programs with very similar requirements, so it does not make sense to create a table with just two rows. We also left out assistantships because that table only was required for a very minor functionality, which we ultimately decided to leave out. There is an argument that assistantships can be a part of students, which we had initially decided against because most rows would have this attribute as NULL.

- Discuss what functionalities you added or removed. Why?

A few functionalities we removed are related to assistantship-related adjustments in semester requirements. It would have been simple to implement, but we chose not to focus on it because it did not change the overall functionality of the system, and only added a few more queries/adjustments to the existing queries. It would still be a good feature to add to future implementations. The same can be said for recognizing other student-specific requirements, like on-campus class requirements for international students. For future work, it would be great to add these more granular features for completeness, but we chose to focus on proving the application's usefulness.

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

They are all instrumental in updating the database automatically. For example, the waitlist-enrollments ecosystem is supported by database programs that can check if a class is full to allow other students to join, if not placing them on the waitlist. It can also add the next student to the class if someone drops.

- 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.

Vibhav - Setting up Flask to query from GCP proved to be somewhat complicated. To solve this, we had to add a number of certificates that are found in the GCP console to the directory. In addition to that, we had to add the IP address of the computer running Flask to the GCP console as well. This IP changed every time we started a new computer session, so we needed to ensure that we added it every time.

Ashwath - While developing the Login and the following pages, I faced an issue with maintaining the user state, ie, the data retrieved and sent to the backend must belong to the respective student. I solved this by passing the student's NetID as part of the URL for the different pages so that the pages can reference the NetID using 'params' in ReactJS. Moreover, while creating the Reviews table by scraping reviews from Coursicle, formatting the reviews was necessary since there were missing/multiple course names and professor names.

Vishal - Keeping track of the various breadth and depth requirements while making course suggestions and writing accurate queries was quite involved. During the reviews page creation, we faced the "_MySQL server has gone away" error. We hypothesized that this could be because of us trying to fetch data every time there was a change in the text input and given the limited resources of the GCP instance, this would not be possible for a large dataset like reviews. We fixed this by catching the error and attempting reconnection multiple times, as well as, calling the query only at the click of a button.

Vinay - The Dashboard feature was quite a bit of work to handle with pure SQL. Some additional processing had to be done with Python. It primarily focuses on the recategorization of courses for students based on enrollment data and degree requirements. After fetching courses from the database, it initially categorizes them into Breadth Courses, Advanced Courses, and Other Courses. The function then meticulously handles the recategorization of courses, especially Breadth Courses, ensuring that if a student has enrolled in more breadth courses than required by their degree, the excess courses are either shifted to Advanced Courses—if they meet the advanced-level criteria (e.g., course code starting with 'CS 5') or reallocated to Other Courses. This recategorization allows us to show a summary of the enrollments to students and they can easily decide what additional courses they have to take, be it breadth or depth at a glance.

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

No, all the changes have been mentioned above.

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

Currently, the project only works for students in computer science master's programs, but with the use of NoSQL databases, it would be much easier to incorporate other degree programs and their requirements into the system. In addition, we could add basic machine learning to the project to recommend classes to match a student's interests, instead of just generically recommending classes in breadth areas not yet taken.

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

Our team managed teamwork quite well, and everyone had a part in each step of the process, both technical and non-technical. Below is how we divided the work:

Vibhav - Initial React/Flask/App setup, CourseList Pages, Development of Courses table

Ashwath - Development of the Login Page, The course enrollment page (Adding and deletion of courses), the Navigation Bar, writing Stored procedures for the application (Course suggestions), and creating the Reviews Table.

Vinay - Development of the Dashboard page that included a summary of Enrollments, and a circular progress bar to show breadths, depths, and total credits. Also implemented triggers for the course enrollment page that checks available seats before adding a student to a course. Created the data for the Students table and Enrollments (along with Vishal) table.

Vishal - Course description pages, Course review page (with instructor-based and course-based search, sorting), Course suggestion page (including queries in the backend for accurate data retrieval), scraping data for creation of reviews table, implementing a sentiment analysis model to convert reviews to ratings, creating enrollments table considering certain conditions.