CS 411 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).

- Our project is similar to what we had in our original proposal. Initially, we wanted to make a project that could intelligently find courses that students may find interesting based on professor ratings, average GPAs, and interests. We wanted to create something that could suggest courses to students based on certain preferences like minimum GPA, rating, term offered, and others. We also wanted users to be able to indicate no interest, some interest, and a lot of interest. Finally, we wanted a search functionality to allow users to look for courses by their GPA, rating, and content, enhanced by machine learning.
- Overall, we mainly stuck to the original plan, with a couple of changes made after starting to code. Firstly, we opted to go with a “yes” or “no” instead of “no interest, some interest, and a lot of interest”. Furthermore, we thought it would be better to not have too much information displayed to the user, so the main page only has the course name, course code, professor, average GPA, and whether or not the course was diverse or not. We chose to add diversity because it allowed us to have more advanced queries, and because we greatly value diversity and believe that it is something that is important to take into consideration. Finally, while we did use machine learning and NLP to suggest courses for users, we did not use machine learning in our actual search component. Rather, we instead allowed users to search for keywords across the course descriptions of all the classes stored in the database.

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

- We think that our application achieved what we intended it to regarding its usefulness. It’s an application that us students can see ourselves using in the future because it suggests courses for students to take based on their previous preferences. Furthermore, it has a simple UI that is easy to navigate and use.

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

- For the most part, we kept our schema and original source of data, however, we ended up not completely using all of the data in our tables. This was because we did not want to overwhelm users with too much information, so we instead picked what data us as UIUC students take most into consideration when picking courses. Finally, while our original proposal did not take into consideration the diversity of the classes, we later added that as a dataset because we think that it is something that is important to consider.

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?

- Originally, we had separate tables for ProfessorData and ProfessorAwards, which was only connected together via the GPA data table via ProfessorName. We quickly realized

that this was inefficient, so we extracted the fields that encapsulate a Professor out of both tables to form its own table, ProfessorInfo, and made a new table, Awards, that was a weak entity, using ProfessorName to uniquely identify itself. This choice was definitely more suitable, as it made scaling our project easier. We can add information about professors and awards independently now, as opposed to our old design, which required both fields to be entered at once.

- Additionally, our original design had GPA data as its own table, but we decided to move it to a relation between CourseInfo and ProfessorInfo as GPA data was uniquely identified by its CourseCode and ProfessorName, to follow the traditional DB design patterns shown in class. This design is definitely more suitable, as it establishes a clear relationship between Courses and Professors now, specifically: “teaches”.

Discuss what functionalities you added or removed. Why?

- While implementing the app, we added a course search feature to search for specific courses and the ability for users to delete their accounts. This is to comply with the CRUD requirements for stage 4 of this project. In addition, we added a feature that allows users to delete their account, as well as a feature that shows when a user is logged in. We decided to do this so that it is easier for users to see when they’re logged in.

Explain how you think your advanced database programs complement your application

- We used our advanced database programs to add specific information about each course recommendation. For example, some of our advanced queries like Course_AvgGPA would find the average GPA of the course taught amongst all professors who have taught that course. A student could then use that information to gauge how lenient/harsh a course is graded.

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.

- **Max:** The main technical challenge I worked on was figuring out how to connect our Google Cloud Storage database backend to our application. This involved heavy research into different python libraries to send sql queries. Additionally, I had to figure out how to link the Google Application Credentials to our backend, so we could freely query the database. Sqlalchemy was the library we used to connect and interact with our database. For other groups solving a similar challenge, this library has great documentation to explain how to link everything together.

- **Michelle**: A technical challenge we encountered was when it came to scraping data. Because we used data from a variety of sources, when it came time to insert data into the tables we realized that there were a lot of inconsistencies across data. One of our datasets had names as first name, middle name, and then last name whereas another only had last name, first initial. We ended up having to do a lot of data processing before

we could actually insert into the tables and we also had to do things such as removing cross-listed courses (ECE/CS 374 for example). Finally, another issue we ran into was that the CSV files did not have quotations around the strings, so when we uploaded to Google Cloud they could not recognize our strings. For other teams who plan on using datasets from a bunch of different sources, we'd recommend that they look through the data for inconsistencies and make sure to budget time to clean up data and ensure everything is consistent before uploading and using it.

- **Nathan**: One technical challenge we found while working on actually implementing the backend of the project was running into version issues of certain libraries that prevented us from actually compiling the code or running into runtime errors. Since we used FastText along with a bunch of other libraries that would process word vectors needed to give course recommendations, installing each library individually would lead to incompatible code within the group. For example, I ran into errors with the Scipy library because I installed a newer version of it. To prevent version incompatibility, we ran "pip freeze" to get the list of all of the libraries and their versions to make a requirements.txt file, which we could share with each other to limit the possibility of version incompatibility across our devices.

- **Saashin**: A technical challenge that we faced while working on stage 3 was figuring out how to actually view our data tables in Google Cloud. Initially, we had thought that there would simply be a part of the dashboard that showed every table in our database along with the contents. However, after a good amount of exploring and research, we realized that we needed to query the database in order to view the contents of tables. We also faced an issue uploading data to our tables, as it would fail without reason. Again, through research, we learned that this was because we needed to upload data in the order of dependencies so that foreign keys could properly link. There was a particular video from the Google Cloud YouTube that was especially helpful for these issues, and I referred to it several times. My advice to other teams would be to refer to this video and similar resources in order to acquaint themselves with the Google Cloud system and how to interact with their database.

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

- We did not go through with the functionality for users to insert their custom preferences, which was proposed all the way back in stage 1 of our project. Once we had implemented the ability to automatically recommend users courses via FastText, it did not make sense for users to manually insert all of their custom preferences, as that would make our project more like a Course Catalog than a Course Recommender.

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

- Other than the interface, another thing that the application can improve on is factoring students' graduation year and major into the recommendations. Right now, our

application asks users for their name, graduation year, and major for purposes of logging in, but it would be really cool if we could also use that information to recommend classes. For example, this could be something like recommending more STEM courses for engineering students and more 300-400 level courses for upperclassmen.

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

- Max: NLP, backend, connecting sql to google cloud, sql queries
- Michelle: data scraping/cleaning, frontend graphics, sql queries
- Nathan: data scraping/cleaning, sql queries
- Saashin: frontend, backend, connecting sql to google cloud, sql queries

Overall, we managed teamwork well. A majority of the work we did was all together in person (up until the very end). We progressed consistently and did not defer tasks to the last minute. When not meeting in person, we also checked in with one other frequently over Discord.