411 Final Project - UIUC Course Hub

Built by Alec Thompson, Shreya Shetty, Shreyas Udupa, Avery Plote

Changes in Project Direction

Overall, the direction of our project has remained largely the same. Our initial goal was to create a platform for students to access course reviews and GPA information for courses they wish to take, and to allow them to submit reviews for courses they have taken. We have successfully built this platform, although the front end has evolved from our initial vision. The changes in the front end were necessitated by user feedback and our own learning experiences during the project. Despite these changes, we have achieved the main goal we set out to accomplish, demonstrating our team's resilience in overcoming technical challenges.

Usefulness - Achievements & Failures

The current iteration of our application remains highly practical. Users can readily access course reviews and GPA information for all university courses and contribute their own reviews. This functionality alone makes it a valuable tool. However, there are areas that could be enhanced, such as the method of submitting course reviews, data organization on the pages, and the inclusion of course information from additional semesters.

Data Discussion - Schema & Source

Our data sources have remained consistent throughout the project. We utilized the Course Explorer API to gather information on departments, course names and numbers, sections, descriptions, and credit hours. The WadeFagan Datasets provided us with the GPA information for the courses. Additionally, we scraped the university directory page to obtain the professors' full names and emails.

ER & Table implementations - Changes & Improvements

We messed up our initial ER diagram since we included the foreign keys, but that only changed our table implementations a little. We fixed and resubmitted the ER diagram, and most of the diagram and the .ddl have remained the same. The main difference is that we added a few tables, namely Departments, which maps DepartmentCode to Department name, and Teaches, which links the many-many relationship between Professors and SectionAttributes. We also changed the implementation of some of our tables, UserFeedback being the most notable. We gave UserFeedback its primary key,

which automatically increments as the FeedbackId to resolve some non-unique issues we were having. The last significant addition was the AvgRating and NumRatings in the GeneralCourse table, linked to our triggers on the UserFeedback table. The one portion of the new design that is much more suitable for our purposes is the auto-incrementing FeedbackId since the ID can be arbitrary, and we were not sure how we would keep track of the ID before finding the auto-incrementing type. This showcases our team's effective division of labor and problem-solving skills.

Functionality - Added & Removed

Our functionality is the same as what was listed in our project report. Users can:

- Search for courses of interest and retrieve information from reviews, including difficulty, time commitment, GPA, etc.
- Sort / filter the courses they are searching for
- Login using Netid (we switched this to university email for simplicity)
- Submit reviews

And the website/database:

- Integrates review content with university data
- Generates summaries of the data upon search
- Facilitates CRUD operations
- Allows users to edit and delete their contributions

The main things we should have done were supplemental or based on presenting the data. We did not allow users to personalize their profiles, which was unnecessary for this website. We also did not aggregate the reviews into course summaries, which would have been helpful. However, it would have been challenging to aggregate the review information into a cohesive summary.

Advanced Database Programs Complement to Application

The advanced programs varied in the way we perceived their usefulness. The triggers seemed very useful for our purposes. We wanted to store and calculate the average rating for a general course without having to join a few tables together, and using a trigger on any operation on the UserFeedback table seemed the best way to accomplish that goal.

It was a struggle to figure out what we should use for our stored procedures and transactions (and really what realistically should be in a transaction/stored procedure in

an actual design). We struggled with this portion because a lot of the internet said any data access should be through a stored procedure, but we also knew we could build and test much faster with our SQL in the Javascript files. However, we found good use cases for our stored procedure and transaction (the transaction was also in a stored procedure). Our stored procedure was longer than most of the other queries we wrote in the Javascript file, so it made more sense to use it like a function call and keep it as a stored procedure. We struggled with the transaction because there was not much, and we needed it initially. Still, once we started thinking about users trying to access data with other users updating data, it made sense to make a transaction for all of our user's UserFeedback data to delete in a transaction instead of outside of one.

Finally, we implemented many of our primary and foreign keys early in the project, so the constraints portion was not pretty good. However, we did add some simple value checks for our ratings to ensure they were within range.

Team Member Perspectives - Technical Challenges

Alec

- Make sure your line endings are LF when importing data into GCP. Setting up our Database took me longer than it should have because of data issues (commas in fields, line endings messing up our data in GCP, and more). Although this was a small time sink for the project, it made a task that took a few minutes, take closer to a few hours to debug and fix. We imported our data into GCP as CSVs, and we also figured out we needed to clean our data more thoroughly and remove the headers from the CSVs in the process. Once we got the data set up, it was smooth sailing from there with GCP, but knowing the line endings were causing us significant issues would have been nice to know going into using GCP (or maybe that is just a MySQL quirk).

Shreya

- One of the challenges during my contribution was to figure out how to integrate the frontend and the backend seamlessly. At first there was a struggle to ensure the data flowed smoothly between the two components. There were times when updates from the backend took longer than expected to reflect on the frontend, causing frustration and delays in development. It was a bit overwhelming trying to troubleshoot these issues while also keeping up with other project tasks.To overcome this challenge, I delved into learning more about API endpoints and frontend state management techniques.We spent hours debugging code,

tweaking configurations, and testing different solutions, collaborating with my teammates on this helped reduce the workload to a great extent. As a tip I would definitely suggest that when you are having an issue and you are stuck on it for more than two hours or so, call your teammates they most definitely will have something helpful for you!

Shreyas

- Figuring out the connections between your Frontend and Backend is extremely necessary. We were confident in our database setup but faced the most significant challenge while connecting the Database to our server on Node. JS. Furthermore, we wanted our UI to be designed in a certain way so that whatever results we wanted would appear on the same page and dynamically change based on the search criteria. This was a more significant challenge as this was my first time doing something like this. I faced numerous errors in getting this right as I had issues with the ID of the input parameters, column names of the table present in the Database, the conditions mentioned in the query, and dynamically ensuring that the results displayed were according to the data entered in the input.

Avery

- My most significant contribution to the project was collecting all course data from Course Explorer. Doing so was challenging since I sent an HTTP GET request for each XML document down the tree. So, for example, I would GET all of the departments, then GET all of the information for a specific department, then GET the info for a particular course, then GET the info for a specific section, then GET the next section, etc. Doing this takes a long time for nearly 5000 courses and 12000 sections. To get around this, I used a multithreaded approach and divided the workload so that data from multiple departments could be collected simultaneously. There may be another better way to do this, but using the multithreading approach significantly reduced the time it took to collect all course information. I recommend doing this approach for other similar scraping projects.

Other Changes

Besides the small things previously mentioned in this report, the project is the same as described in the proposal.

Future Work & Improvements

As discussed above, it would be great to improve the GUI, but setting that aside, the application could improve data aggregation and ease of user input. We wanted to aggregate the reviews into one cohesive summary for each course, which would be very useful and exciting to implement. Additionally, it would be great if users could enter their reviews without entering a CRN, but that also requires changing how the database structure is set up.

Division of Labor and Teamwork

Alec

- Helped manage collaboration and communication between members for all stages
- Wrote the ER diagram for the project
- Wrote the .ddl for the Database and set up the Database on GCP
- Did the final cleaning on all of the data before importing into the Database
- Helped run indexing analysis and report
- Wrote the transaction, stored procedure, triggers, and constraints for part 4
- Set up most of the backend-frontend communication in stage 4
- Helped with the frontend design and input fields

Shreya

- Helped with the project proposal document
- Collected all data related to the GPA history from 2019
- Cleaned the data and prepared it for processing
- Helped draft the queries required for stage 3.
- Helped with Initial indexing of the queries
- Completed GCP server setup for checkpoint 1 (Stage 4)
- Completed the checkpoint 1 frontend and backend deliverables for stage 4 on both local system and GCP
- Created the frontend for the project
- Collaborated and decided on the modules to include for our website
- Helped with the frontend and backend integration for course reviews and user feedback
- Helped with troubleshooting for the integration between frontend and backend

Shreyas

- Helped draft the queries required for stage 3.
- Assisted in troubleshooting the Relational Schema and the ER diagram
- Utilized web-scraping to get the details of the faculty at UIUC.
- Created the UI Mockup for our Project during Stage 1
- Worked on connecting the Frontend-backend-database connections in Stage 4
- Conducted troubleshooting for most of the frontend-backend integration issues
- Tried to figure out how to host the server on GCP
- Maintained Uniformity across all the web pages of the project and ensured that the links were referenced correctly

Avery

- Helped write project proposal
- Helped with ER/UML descriptions
- Wrote relational schema
- Parsed data into CSVs from Course Explorer
- Helped run indexing analysis and write indexing report
- Assisted with testing the final project
- Wrote most of this report

Overall, the division of labor was pretty good. We all had parts of the project that we worked on and could meet the deadlines we needed to. We could have had better teamwork at times, but we faced no significant team difficulties or arguments. The team got done what it needed to and now has a completed project to show for it!