CS 411 Final Project Report

Group 116 — Sruthi Kode (kode2), James Mallek (jmallek2),
Kavya Moharana (kavyam3), Samidha Sampat (ssamp2)

Changes from Original Proposal:

Our final project contains a few changes from our original proposal as we did not implement providing the user with ‘smart hints’ after incorrect guesses. We also did not implement the endless mode which was mentioned in our original proposal. This feature was supposed to allow users to continue to guess random classes after already identifying the class of the day. We made this choice in order to maintain the likeability of the game, similar to the format of the New York Times Wordle.

Application Usefulness:

Our application delivered a fun, engaging, and educational experience for current and prospective UIUC students. It offers an entertaining trivia-like game where users can continuously guess the chosen Class of the Day. The educational value comes from users gaining more information about the chosen class by receiving a detailed summary after successfully identifying the class. This feature helps students discover courses that they might not have encountered otherwise. We also are able to store information about the users such as their favorite courses, average number of guesses, and even the most frequently guessed courses by users overall. Additionally, our application is quite unique, as we haven’t come across any similar platforms focused on college courses.

We were not able to successfully implement the ‘smart hints’ feature which would access real-time data from the users and provide them with information about the course to help guess correctly. Given the vast amount of courses at the university, the lack of hints makes it more difficult to win the game and impacts overall user satisfaction. While this doesn’t affect our application’s usefulness, the feature is important for user engagement.

Project Schema and Data Sources:

We maintained the same sources of data as mentioned in our project proposal. We used the course catalog of UIUC for 2024 as well as a dataset: ‘GPAs for courses at the University of Illinois’ both found on the Data Science Discovery website.

Changes to ER Diagram and Table Implementation:

We made a few changes to our UML diagram and table implementations. We are no longer using the CollegeUnit under the Departments entity in our database tables. We found that it was adequate to simply provide the department that the course is listed under regardless of the specific college unit. Additionally, we are no longer using ‘Value’ for the Guesses entity. Using value was unnecessary as we can just directly compare the guess to the daily class. We think this final design is more suitable because it omits redundant or nonessential information without affecting the overall game.

Functionalities

Apart from the specific attributes or proposed features mentioned above, we did not have any specific functionalities that we added or removed, following the database design or implementation stages.

Advanced Database Programs

Our application contains a trigger that updates a user's favorite course (most guessed class) as it changes. The most guessed course can be seen by users in their information page which also displays other interesting user details and is where users can decide to change their email or delete their account. Without the trigger, we would have to recalculate the most guessed class every time it was checked. The application also has a stored procedure which is called for every guess. It takes a department, number, user ID, and the current date to create a new record in the guess table as well as return information about the course guessed and the correct class. This allows the backend to wrap this information in a JSON format for the frontend to display to users. The user sees the department, number, name, number of credits, and Gen Ed of the guessed course. They also see whether they were right, wrong, too high, or too low for these attributes. This means they will be more informed for their next guess.
Advanced database programs such as the trigger complement our application by improving performance and scalability by minimizing computation. The stored procedure also improves consistency and efficiency within the database logic.

Technical Weaknesses and Challenges

One technical weakness of this project was that indexing did not reduce query costs for our simple queries. We found that this was likely due to MySQL's automatic indexing, making our manual indexing redundant. Additionally, our dataset may have low-selectivity, meaning that there are many non-unique values in a single column. Low selectivity reduces the effectiveness of indexing, often leading to slower query execution. While this was not a major issue in the context of our project, it remains a minor challenge that should be considered and addressed when implementing indexing strategies.

Another technical challenge we encountered was implementing APIs to retrieve real-time course updates. However, due to time constraints it would have been difficult to fully integrate this feature. In order to implement the APIs we would have had to do it for both datasets we were using. Additionally, the APIs needed to be designed to handle high query loads efficiently while securely managing user data storage.

Integrating GCP with our web application during Stage 4 was a technical challenge as well. After completing the Stage 3 database implementation on GCP, the extra steps involved with setting up a NodeJS server on GCP with a VM instance were more complicated than we anticipated and we did not have the time to properly do so (especially after working on revisions). It would have been beneficial to become familiar with the GCP console in regards to backend/frontend development earlier on.

Finally, smart hints was another weakness we had because it limited the game's ability to provide feedback in real time after a user guessed wrong multiple times. Implementing the smart hints feature would have meant that we would have needed to do additional data analysis and utilize various methods to compare the guess that the user made to the answer and give feedback that would help the user guess the right answer. In essence, we would have had to create a feedback algorithm because we would want to help the user obtain the answer, without giving them the answer.

Future Work and Improvements:
In the future, we can implement the smart hints after incorrect guesses to improve the application. This feature provides users with helpful guidance, making each subsequent guess more informed. We could also make hints optional for users, allowing for a more challenging experience for those seeking it. These changes would help users become more familiar with courses while helping them improve on the game more each time that they play.

Division of Labor:
Our division of labor for each stage/checkpoint is listed below. We were able to balance teamwork well by delegating tasks prior to each checkpoint, discussing issues, and working together in person. Each team member contributed equally to the overall success of our project.

Stage 1:
1. Creation of doc folder, README, and TeamInfo.md - Sruthi
2. Project Proposal - Everyone

Stage 2:
1. ER/UML diagram - Everyone
2. Assumptions for each entity/relationship in model - Sruthi
3. Normalization of database - Kavya, Sruthi, Samidha
4. Relational Schema - James

Stage 3:
1. Implement 5 main tables - Everyone
2. DDL - James
3. Populate tables - Kavya
4. 4 advanced SQL queries - Sruthi, James, Samidha
5. Indexing each query - Sruthi, Kavya

Stage 4:
- ➢ Checkpoint 1 –
    1. Create a functional webpage - Everyone
    2. Connect front-end & database of application - Everyone
    3. Retrieval/CRUD capabilities - James
- ➢ Checkpoint 2 –

1. Implement transactions - Kavya, Sruthi, Samidha
2. Implement stored procedure - James
3. Implement trigger - James
4. Implement constraints - Everyone
5. Project report - Everyone