Successes of the Project

Project Structure: The separation of concerns between the backend and frontend proved to be highly effective. Our utilization of the backend for OpenAI API interactions and the frontend for displaying the results created a smooth, efficient workflow.

Effective Use of Chat Completions API: By employing careful prompting techniques, the project successfully generated quizzes that were not only somehow relevant but also consistent in format and content. Specifying requirements such as topics and desired JSON format in the prompts led to a consistent output from the GPT model.

Interactive Frontend UI: The use of React for frontend development resulted in an interactive and user-friendly interface. This aspect of the project significantly enhanced the overall user experience. Rather than just talking to the GPT, there are some levels of masking above that, which make the quizzes more engaging. The users can get the response from their quiz selection in real time, and they can also modify the levels of details of the feedback.

Innovative Result Page Generation: A notable feature was the generation of a detailed results page upon quiz completion. This HTML report included a unique 'confidence' indicator, providing valuable feedback to students on their performance. Moreover, this report could be used as a basis for generating new, tailored quizzes, adding a layer of personalization to the learning experience. This report also includes a hash, which can not be modified by the user (if they do, it can not be used to generate a new quiz). The instructors can use this performance report to check the ability of the students.

Challenges and Lessons Learned

Inconsistencies with the Assistant API: While the Assistant API was generally effective, it exhibited occasional inconsistencies in generating reproducible output. The success rate of approximately 9 out of 10 indicates room for improvement, particularly in the speed and reliability of question generation. This highlights the need for further optimization, or alternative solutions to ensure a more consistent and efficient response. As of right now there are not many parameters in the Assistant API we can modify to make it more reproducible. We also found some difficulty with prompting the Assistant API with GPT-3.5 Turbo (the results were generally inconsistent), and eventually switched to GPT-4.

For both versions of GPT, we attempted to use the function calling API to enforce a JSON schema, but found it still somewhat unreliable, as GPT would occasionally insert assignment statements. Also, we found the additional tokens that function-calling incurred was too heavy of a cost. We decided to utilize chat completions for the reminder of our demo, simply because we believe the Assistant API as of currently is too unreliable and costly for production.

Future Improvements: We would like to improve our interface to include confidence flags for each question, which was included for our proposal. We could encode this information into our stats download, which would reflect the user's confidence in the report. This could help with gauging the student's own understanding of their performance, a metric that is not usually included with online tests.

We also hope to utilize the Assistant API in the future, when it becomes more stable. The ability to cite the files we uploaded, including the course materials, makes Assistant especially powerful, and we believe that when the parameters for temperature are included, Assistant will be more reliable in generating information. We also believe the function calling ability, while in reality just a wrapper around the basic functionality of GPT, could still be quite useful when explored. However, as of currently, our future improvements to the project would largely be based around improving our Chat Completion prompts.