AI Flashcard Generator Project

Flashcards are a great tool for learning and memorizing concepts, but creating them from scratch is a tedious and time-consuming process. Our project aims to alleviate that stress by leveraging the gpt-4o mini model to generate flashcards from class notes automatically. At the click of a button, students can create flashcards to study from. Users can input their notes either by pasting directly into the textbox or uploading a file, and the website processes the content using the OpenAI API to create a deck of flashcards. By automating the tedious task of creating flashcards, our app allows students to focus on what matters most: studying.

Learnings

While using ChatGPT to build our application, we began to get a better grasp on phrasing prompts effectively to generate functional and relevant code. We started off by asking ChatGPT to build us a web app that calls the OpenAI API to generate flashcards from lecture notes. While we got the basic functionality working after some error handling, it was too basic so we decided it would be best to start again from scratch. This time, we got tunnel vision when attempting to implement some technical features, which led to ineffective prompting. We wanted to add a toggle which allowed users to switch the default orientation of the flashcards to show the definitions first. Instead of directly stating our problem and how we wanted the solution to look, we asked ChatGPT how to have our backend interact with the HTML element, and slightly alter the API call if the element is toggled on. While this worked, the default orientation of the flashcards would be fixed from the moment the flashcards were generated, and not dynamic as we had hoped. After sending ChatGPT many more messages, it understood the gist of the feature we were trying to implement, and it presented us with a much more elegant solution. It turns out that we did not need to change the API call since a simple function in the Javascript would be able to change the orientation of the deck depending on if the HTML element is checked. This solution was better than ours since it allows the user to dynamically change if the term/definition appeared first, even after the deck is generated. We learned that asking ChatGPT for a solution that we think will solve the original problem without enough context leads to a suboptimal solution overall. If instead we ask ChatGPT for a solution to the original problem while also providing our proposed approach, it would be able to generate a much better solution. Ultimately, we realized that this lesson can be extrapolated to beyond AI, and that it is important to communicate your overall goal when describing your problem to other engineers to avoid limiting potential solutions.

We learned how to handle different input formats to make our app more convenient and broaden its use cases. Our app was initially only capable of processing text input, but we wanted the option to insert a pdf file as well. Text is directly inputted into the ChatGPT prompt, while the PDF input requires the text to be extracted first, then passed into the prompt. Using the PyPDF2 pdf to text extraction library, we iterated different versions of an extraction function along with some minor prompt engineering to build a robust backend service for PDF inputs as well as text.

In the beginning, we started with a static prompt, which we refined to get concise and accurate responses. We wanted to keep responses as short as possible while still retaining the most important information since it would mean that we could generate more flashcards given the same quota of output tokens. Eventually as we added more features for the user, we needed a way to dynamically change the API call based on the specified settings. We realized we could build a complex prompt by adding variables which reflect the specifications, which would all be passed into the API call together. Additionally, we tinkered with the model's parameters along with the prompts to get ideal responses for our goal of generating flashcards. We noticed that lowering the temperature made the model's outputs more consistent and predictable, which was ideal for creating accurate flashcards. On the other hand, higher temperature values resulted in more creative but less predictable results, and if high enough, the model would occasionally hallucinate, which is too unreliable for students studying rigid course content. We ended up keeping this parameter close to the default value to satisfy our app's educational goals.

Successes and Failures

We believe the main success of this project was the UI, and that is what ultimately differentiates it from using ChatGPT directly. At first, the user would have to click the text precisely in order to flip the flashcard. Later, we were able to update the HTML so that the user could click anywhere in the flashcard box to toggle between the term and definition. We also added active feedback when a user hovers over a flashcard—it glows to make the UI even more polished and interactive. Additionally, we added an open-ended text box for user input, where they can add personal customization preferences to the flashcard. This allows the student to make the flashcards suitable for their specific studying needs and styles. An example of an input the user could put is "explain the definition in NBA terms." ChatGPT is great at drawing analogies between seemingly unrelated fields, allowing students to form stronger understandings about new material by leveraging deep understandings in topics they are already knowledgeable on.

We attempted to add image generation to the flashcards using the DALL-E API, where it would use the text content on each flashcard to create an image to add to the flashcard. However, we ultimately decided to abandon this feature since we could not consistently generate images, let alone keep them confined to a specific boundary on the flashcard. We ultimately decided that the image generated would likely not add much value to the flashcards and be more distracting than helpful to the student, so it was not worth investing more time and effort into this feature.

Key Takeaways

From this project, we learned the importance of effective prompt engineering, clear communication with LLMs, and adapting to modern development processes using AI. We were able to improve the functionality of our app by leveraging dynamic inputs, ensuring it could adapt seamlessly to various user preferences. While we couldn't implement all of our ideas, we gained insight into the importance of balancing creativity and practicality when designing a product to serve other users. Ultimately, this project experience strengthened our understanding of software development in the changing AI driven landscape.