The inability of my app to handle accents is one of the drawbacks I noticed. The system, which is built on American English, has trouble with non-native or non-American accents, which I acknowledge is an elaborate issue that needs an extensive amount of training. For instance, my accent wasn't correctly identified, which prevented the program from moving on to the next step, thus I couldn't test some inputs, like weekdays as dates.

Another issue I ran into was with my code's layout, particularly with regard to how I store the "yes/no" answers. At first, after changing to a new state, the response wasn't reset to null. Incorrect state transitions resulted from the program keeping the prior response in memory when it returned to a state that needed additional "yes/no" input. I manually set the answer to null upon entering the appropriate state in order to fix the issue and make sure the new input could be processed properly. I could have handled this by creating a new value in the context, but I decided against it because the input wasn't required for additional processing.

Additionally, I assumed that users might give more intricate or lengthy responses than the software was originally intended to process. A user might say, for instance, "I am meeting with Diana" in response to the query, "Who are you meeting with?" rather than just "Diana." In order to solve this, I created a function named parseUtteranceForCategory, which breaks the utterance up into separate words and verifies that each word—or word combination—is correctly recognized by comparing it to the grammar. In the example response, the function checks each word from the utterance against the grammar and identifies if any of the words match a person's name.This method guarantees that the application can process a wider range of natural user inputs while still accurately recognizing important data.

Lastly, one of the most significant limitations is that the system is only capable of "booking" meetings based on predefined entities within the grammar. In a real-world application, this approach would not be entirely feasible, particularly when interacting with someone for the first time, as the system lacks the flexibility to adapt to new or unfamiliar entities. To address this, I believe incorporating a state that can take user input, verify the correctness of the entity, and then dynamically add it to the grammar would significantly enhance the system's flexibility. This would create a more dynamic flow and, in essence, allow for an "infinite" grammar, enabling the system to handle a broader range of scenarios and interactions seamlessly.