Sprint #2 Review Document

Summary of Accomplishments (Angel)

In this sprint, our team focused on improving user experience through personalized spotify authentication and the development of foundational voice command functionalities. We first enabled user login via spotify, which allowed us to retrieve and display the user’s display name, email, and playlists. By personalizing the application with spotify data, we laid the groundwork for a seamless integration that aligns with the user's music preferences. We implemented this using spotify's OAuth 2.0 protocol, which provides secure authentication and access token management. After the user successfully logged in, we retrieved their display name, email, and playlists through spotify’s Web API. Displaying this information creates an initial connection between the user and the app, enhancing engagement and preparing for future voice-driven playlist control. To ensure smooth functionality, we tested authentication with multiple accounts to confirm consistent data retrieval and display. In addition to the spotify integration, we created two sample python programs to explore microphone based voice command interactions. These programs aimed to support hands free functionality by detecting and responding to user input through voice. The first program employed a wake word, simulating how voice assistants like Alexa or Siri activate upon hearing a designated phrase. Using the SpeechRecognition library in python, we programmed the application to listen for the specified wake word, remaining idle until it was detected. Upon hearing the wake word, the program activates, signaling that it is ready to accept further commands. This setup allows for more natural and user-friendly interaction by avoiding continuous listening and focusing only on relevant user input. Testing for the wake word detection involved diverse environments with varying noise levels to measure accuracy and reliability. The second program was designed for continuous listening, immediately transcribing user input as the user spoke. To ensure the user knew when the program was actively listening, we included a text-to-speech (TTS) confirmation, which provided real-time feedback, saying, “listening” whenever the program began processing audio. This immediate feedback allowed users to understand when the program was receptive to commands and when it was idle. Using the SpeechRecognition library in tandem with a TTS library (pyttsx3), the program provided a hands-free interface with responsive feedback that simulated an interactive conversation. During testing, we assessed the program’s transcription accuracy and the effectiveness of TTS feedback. Challenges arose with managing system performance, as continuous listening and TTS feedback can be resource-intensive, so we optimized settings to balance responsiveness with efficiency. Overall, this sprint successfully achieved foundational milestones in both personalization through spotify authentication and initial explorations of voice command technology. Moving forward, we plan to build on these foundations by refining wake word detection and enhancing continuous listening efficiency, setting the stage for seamless voice based control within the app. With these developments, we are well-positioned to create a hands free experience that will allow users to control their music effortlessly.

Challenges and roadblocks (Irelyn)

Challenges that our team faced were issues with the updated app.py, calibrating the voice input, communication issues, and other technical issues mostly with compilation. The issues with app.py and compilation and communication are connected because they happened due to some communication issues and they were both problems that occurred for the front end of the project. Me and Aiden were not aware that we needed an env file, Spotify accounts, and the emails attached being sent to Angel to allow for proper testing. The lack of the emails to assign us as users resulted in some of the output beyond the login page being off when it was being tested. The lack of an env file resulted in some compilation errors that both me and Aidan ran into. The communication issues also resulted in me making an extra py and html which was deleted because it wasn't necessary. Because of this issue Angel tested the code I committed to confirm that it does display the playlist as intended. These issues have been resolved. The voice input was an issue on the back end. The input had a hard time properly picking up people's voices to get the right commands. This problem was fixed by Uriel. This is the main issue that plagued the back end.

Client/Stakeholder feedback (Uriel)

We were able to push out a small demo for the client this sprint. They liked how we have a listening feedback stream, which lets users see if the speech-to-text program is working properly. Our client gave us a description of what they wanted the user interface to look like. The key takeaway from the client is to have a user-friendly web interface with commonly used media controls and display playlists/songs. Having a box showing possible commands that can be used through voice is also desired from them. The most useful commands will be listed at the top to help users. Also, after signing in to a Spotify account, the speech-to-text program should immediately start listening to the user's voice. This part should be simple to implement soon.

The impact the feedback will have on our future sprints is positive; we now have an idea of what the interface will look like and what we should strive for with the functionality of voice controls. In the next sprint or the one after, we should be able to produce a demo that has the client's ideas implemented.