Notes2Notes Project Proposal by Divya Viswanathan

Project Description: Notes2Notes is a digital journal where you can type out your thoughts and emotions for each day (as you would when writing in a traditional journal), and the app will analyze your writing with the songs in your Spotify to generate a playlist of songs that best reflect your emotions/thoughts using song lyrics.

Competitive Analysis: There are various playlist generators that are online such as MagicPlaylist and Spotalike that can cleverly generate a playlist based on different user inputs. MagicPlaylist and Spotalike both generate a playlist similar to a song or artist the user enters. There are also playlist generators integrated within websites like playlists.net that generate playlists based on mood of the user. There is some sentiment analysis there, which is what I hope to accomplish in the future with some basic natural language processing. My playlist generator is more interactive for the user, who can use it to enter their thoughts like they would in a traditional journal or diary. From there, my application analyzes the words in the journal entry and specifically analyzes lyrics of songs that the user already listens to. This generates a playlist of familiar songs that are relevant to the thoughts they had that day.

Structural Plan: A folder that contains 3 Python files (one with the music/spotify parsing, one with functions to filter through journal data, and the main application that has the graphics and imports the other file to run the application) and a folder with journal entries (or yet to be journal entries) saved in them. The music setup methods are stored under the MusicSetup object and the graphics use modes to have different screens and views in the app.

Algorithmic Plan: Getting spotify info and matching songs to lyrics involves a few algorithmic steps. First, I have to create a set of all the songs in all of the users playlists and saved songs (so all songs are unique). Next, I filter out the non-vocal songs from the current song set using a specific spotify API endpoint called ‘instrumentalness’. Now, the song set contains all the options to lyric scrape. Now, I have to filter the journal entries themselves. I filter through the journal entry, only extracting a list of the relevant words from the entry (for now not including conjunctions, prepositions, or words under 3 letters). The song set is likely very large to scrape lyrics for every song, so I filter that down to a list with just songs that have titles that match with any of the relevant words. Then I scrape lyrics of all the songs possible and store the song with its lyrics in a dictionary. Then I count the occurrences of each relevant word in each song’s lyrics, store that in a dictionary, and total the counts for each song to get a score for each song. Songs are ranked from highest to lowest non-zero score and stored in an ordered list. Score calculation can also be changed to a different formula (such as a percentage of words in the song or other scoring methods). Then a playlist is created using those songs with the order of the playlist reflecting the ranking order. The default description of the playlist is the keywords (successful relevant word matches in any/all song lyrics) used to generate the playlist itself.

Timeline Plan:

TP1: Complete basic graphics and journal saving/loading system

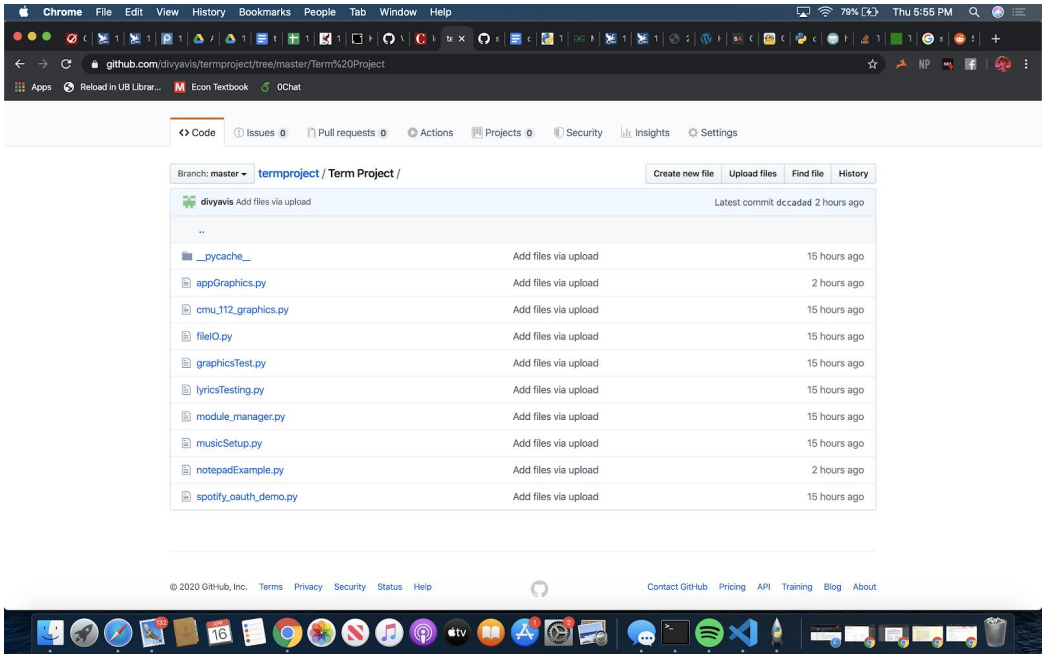
CP5: Be able to generate playlists from a few journal entries (integrate musicSetup and appGraphics)

TP2: reach MVP (ideally a couple days before TP2): input journal entries and save, parse through spotify playlist, solid algorithm for matching song lyrics to journal entries, create playlist based off of matches, customize playlists (daily, monthly, etc.), make UI more aesthetically pleasing

CP7: integrate extra features, sync with google calendar/have planner option, make matched song alarm based off of google calendar/planner data, see if i can use machine learning/nlp to match song lyrics and journal entries

TP3: complete final project with nice UI and integrated extra features

Version Control Plan: I am using a private GitHub repository to back up my code.



Module List: spotipy (external library that calls Spotify API to parse through Spotify data), BeautifulSoup (webscraping lyrics), lyricsgenius (external library that calls Genius API to parse through Genius song data)

TP2 Update:

- Design follows plan from previous submission
- Was able to figure out how to handle some bugs with lyrics scraping (now more accurate lyric-gathering method with BeautifulSoup) as well as other bugs found during the integration of the front-end with back-end
    - Handles requests better without crashes
- Fixed cursor issue with individual approach to text entry, but not sure where to implement text entry method with cursor (which is a canvas implementation instead of current text widget)
- Considerably spruced up UI
- Things to still work on (extra features and things to clean up):
    - After testing and learning some or all of these features will be implemented by next week:
        - Add a reset user button
        - Add a logout button
        - Add a get out button after playing songs
        - Add a loading screen when creating playlists
        - Indicate on calendar screen when someone has made an entry/playlist
        - Sync with google calendar
        - Get data on public spotify-generated playlists
        - Learn more NLP techniques to spruce up word filtering
            - Perhaps utilize nltk module
        - Extra user preferences if i can implement it with the api