GMR's Algo-Machine

An algorithm based playlist builder

Graham Ritter

Problem Statement:
Users on Spotify continually search for new music, often turning to Spotify's algorithm generated playlists. Users listen to these playlists through at a greater rate than Spotify produces them. Therefore, users continually need new music content to generate the soundtrack to their life.

Impact:
Empowering users with more catered music will lead to greater Spotify usage as well as higher user utility.

User Needs:
All regular users of Spotify are affected by the lack of new music suggestions. Users need individualized recommendations based on their listening habits beyond the two generated for all users (Discover Weekly and Release Radar). With greater control of algorithm produced playlists, users can find more new music.

Current State Processes:
The major pain point in the current listening flow is when users want to find new music, they have limited options within Spotify.
[Appendix A]

Potential Solutions:
- Source song recommendations from 3rd party services
  Pros: curated playlists, potential higher quality of recommendations, number of potential sources with APIs (8tracks, Bandcamp, Last.fm)
  Cons: reliant on another service, not able to modify their service, not based on user listening preferences, solution would be more playlist transcription as opposed to generation
- Use Spotify's API to feed seeds for recommendations
  Pros: based on user inputs, based on user listening habits/playlists, only one other major stakeholder in the generation process
  Cons: Recommendations based on playlists yield high randomness, potential overload on Spotify's API with too many seed/recommendation calls, high computing power requirements with scale
- Curated Playlists, human generated playlists for different categories
  Pros: high quality of playlists
  Cons: lack of automation in generation, not user based, not different than Spotify's current offerings, need to pay someone to create playlists

Proposed Solution:
Use user's current listening habits to feed seeds for recommendation to Spotify's API. This solution allows for users to have control over the types of playlists they can create using the playlist generator. Users may use their playlists and top tracks to seed for new recommendations, resulting in a playlist of new music. Users access this service through a web app, authorizing the application to access user information, resulting in playlists directly to their library.

Future State Processes:

The playlist generation portion has been automated. The process for finding new music has increased in complexity but only due to the greater range of options. [Appendix B]

System Objectives:
The system should create a playlist based on user input, allowing choices from playlists and top tracks. It should be accessed through a web application, through which a user will authenticate with Spotify.

Functionality:
Accessed via flask app, run in browser.
- homepage, login, user input

Authentication
- Uses Spotipy's authentication function

Retrieve user information
- Playlists
- Top tracks

Present to user for selection
- Take user input

Add selected seed tracks to array
Create playlist
- Seed selected tracks with Spotify's recommendation feature
- Name playlist

Information Requirements:
Inputs:
- Spotify provided inputs include playlists, short-term, medium-term, and long-term top tracks, and other Spotify generated playlists
- User inputs based on desired seed songs and algorithm options
- User authentication and cache creation

Outputs:
- Playlist created in Spotify
    - Manage number of playlists created
- Saved authentication token

APPENDIX C: Dataflow Diagram

Interface Requirements:
Users will access via a flask web application based on the web-app-starter-flask from the course repository [APPENDIX D]. User input will be taken via buttons and checkboxes for seed selection and algorithm options [APPENDIX E: prototype login page].

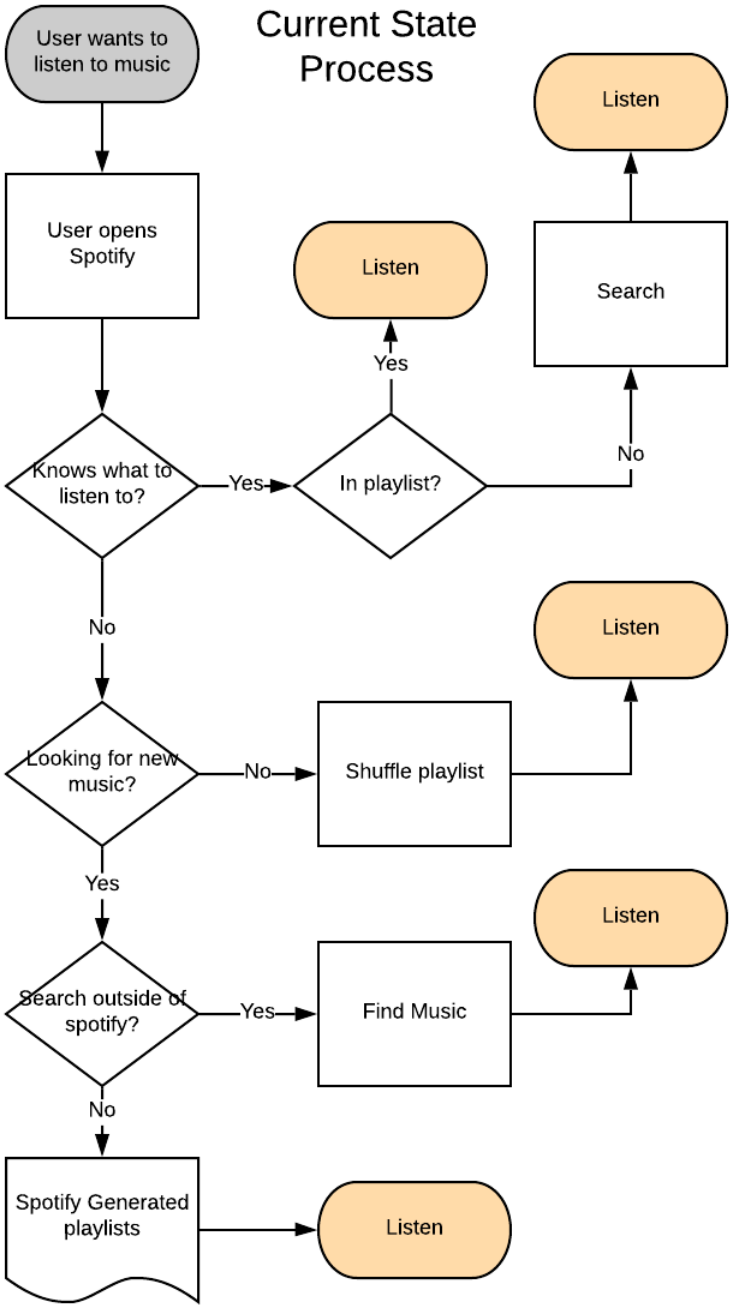
Technology Requirements:
APIs: This project will rely primarily on the Spotify API, the backbone of the project
Python Packages:

- Spotipy: used to interface with the Spotify API,
    Must install with command
    pip install git+https://github.com/plamere/spotipy.git –upgrade
- Flask: used to create the web application, necessary for user interface

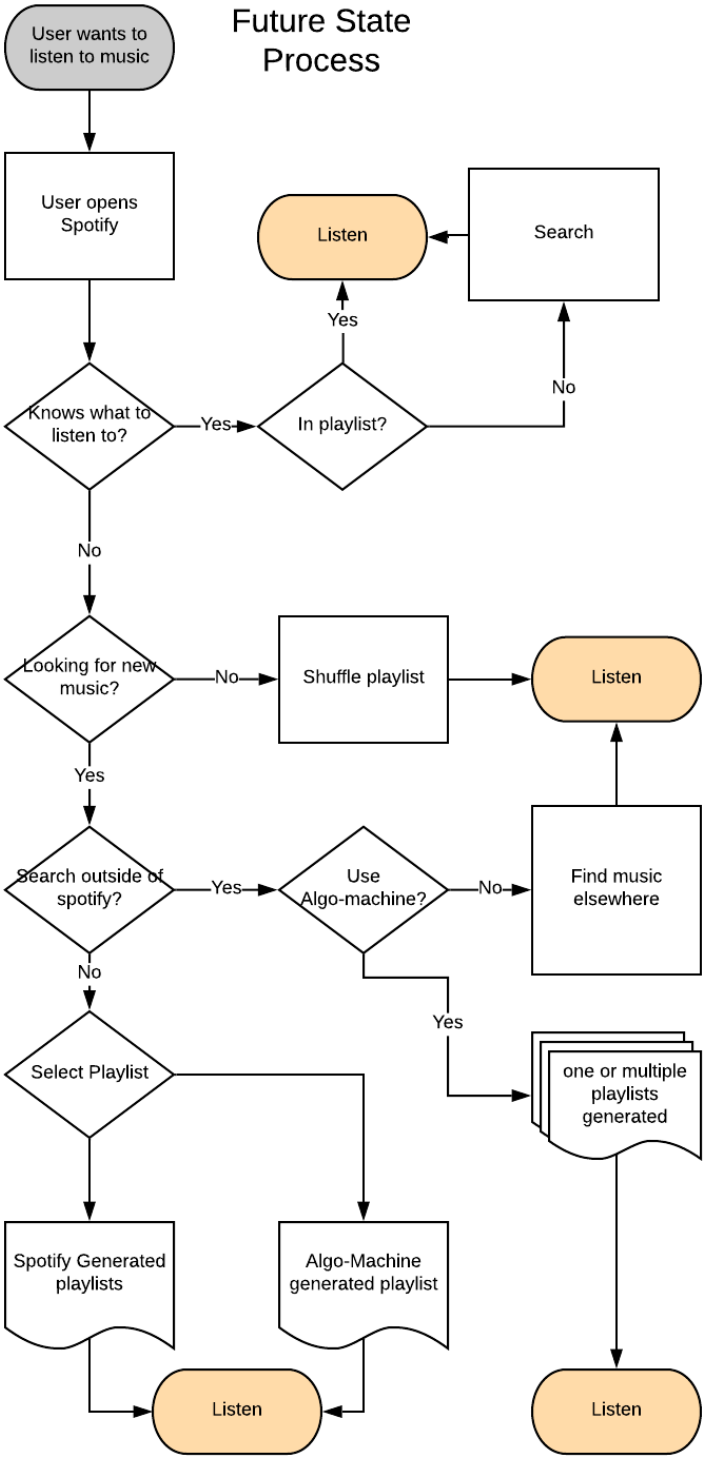
Remote Server:

- Deploy on Heroku
- Further: host on a website, schedule interval playlist generation

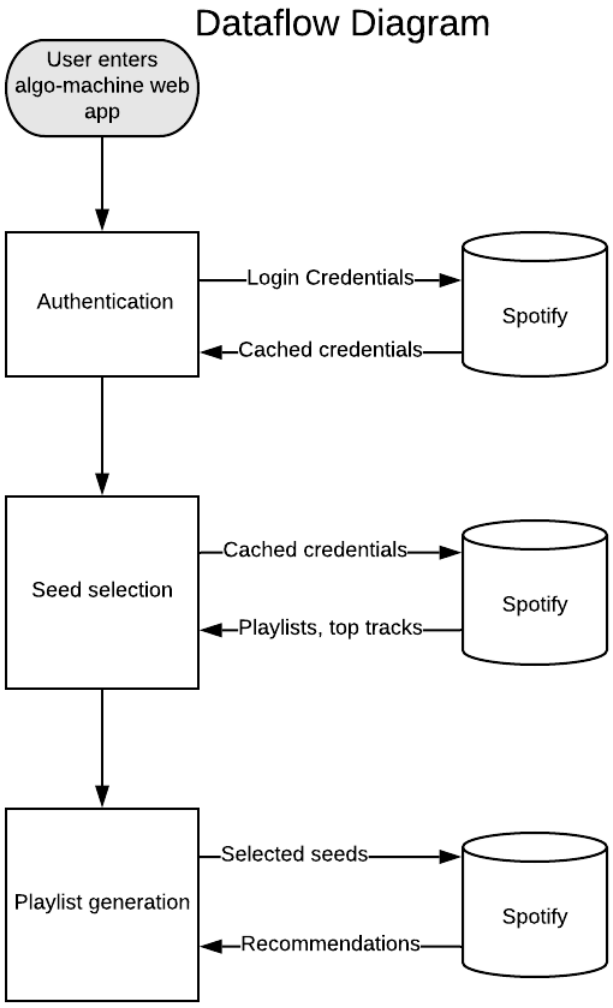
APPENDIX A:


Current State Process
User wants to listen to music
User opens Spotify
Knows what to listen to?
Yes
In playlist?
Yes
Listen
No
Search
Listen
No
Looking for new music?
No
Shuffle playlist
Listen
Yes
Search outside of spotify?
Yes
Find Music
Listen
No
Spotify Generated playlists
Listen

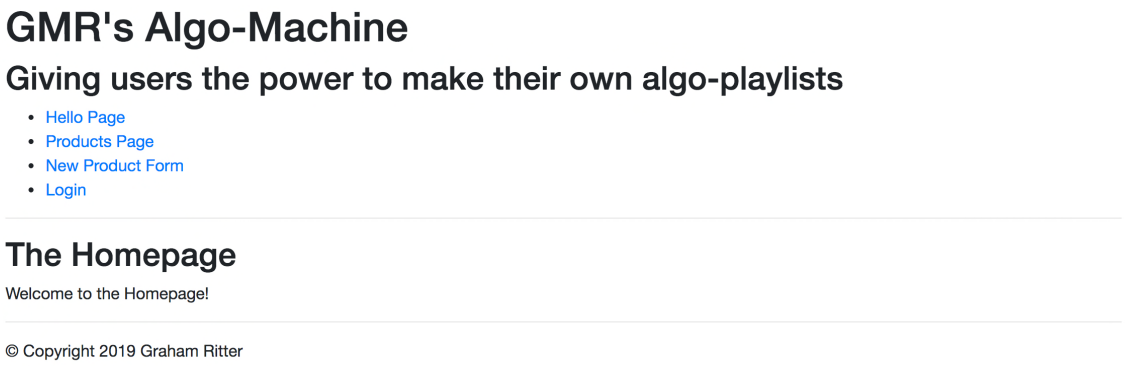
APPENDIX B:


Future State Process
User wants to listen to music
User opens Spotify
Knows what to listen to?
Yes
In playlist?
Yes
Listen
Search
No
No
Looking for new music?
No
Shuffle playlist
Listen
Yes
Search outside of spotify?
Yes
Use Algo-machine?
No
Find music elsewhere
No
Yes
Select Playlist
one or multiple playlists generated
Spotify Generated playlists
Algo-Machine generated playlist
Listen
Listen

APPENDIX C:


Dataflow Diagram
User enters algo-machine web app
Authentication
Login Credentials
Cached credentials
Spotify
Seed selection
Cached credentials
Playlists, top tracks
Spotify
Playlist generation
Selected seeds
Recommendations
Spotify

APPENDIX D:



APPENDIX E: Prototype login page