* Database
  + Use database to add new user
    - Each user has userid - that id will link to different album suggestions when they click to spotify to suggest
    - [user id (gmail or spotify account name), album, genre of music, book title]
    - Connect database through cloud?
    - postgres
  + API
    - Book API: https://www.programmableweb.com/api/goodreads
  + Decoupled architecture should be good
  + Amelia and Justin - once database is all set will print out username and list of suggested albums in playlist
* OAuth:
  + <https://developers.google.com/api-client-library/java/google-api-java-client/oauth2>
  + https://www.youtube.com/watch?v=xH6hAW3EqLk
  + User just logins with google email

Steps:

* User logs into spotify and gets oauth
  + We have access to their albums now
  + Make code that generalizes their music taste and spits out a genre
  + Make a dictionary for each genre and count how many times each song has that genre to find the most popular genre (Python - temp dictionaries)
    - Find the top two genres and with that find the equivalent of the genre in book form
    - Store the specific top two genres in the database w/ user id
  + Call the user and get their genre from the database
  + Use the book api to find the genre related to the book + return the book title
    - Store the book title in the database w/ user
  + Display the book title to the user + send link to book
  + Keep the list of books and show it to user as a dashboard - books they are going to read - give them an option of taking the book out of their to read (to do list basically) - once you finish reading the book, delete from database

<https://www.postgresqltutorial.com/postgresql-python/connect/>

<https://www.postgresqltutorial.com/postgresql-insert/>

<https://www.geeksforgeeks.org/python-postgresql-select-data/>

<https://www.postgresqltutorial.com/postgresql-python/query/>

<https://stackoverflow.com/questions/485039/mysql-insert-query-doesnt-work-with-where-clause>

<https://blog.postman.com/generate-spotify-playlists-using-a-postman-collection/>

**Wayne’s Progress on Dec 9th**

* created a virtual env for this project
  + so after you guys do a git pull (which might take some time), you should be able to cd into the project folder and type this command in terminal:
  + source 411\_env/bin/activate
  + then you should not have to configure the environment again (hopefully my understanding of virtual env is correct)
* created a postgresql.py for database related code
  + I put all the database related code there but feel free to organize them into places that are more logical
  + so far, the code will create a table named USERS, which contains:

{

ID BIGSERIAL NOT NULL PRIMARY KEY,

USER\_ID VARCHAR(100) NOT NULL,

Playlist jsonb,

Music\_Genre jsonb,

Book\_Titles jsonb

}

* one thing I am not sure about is whether I should keep everything else except userID in json format or should I parse it out before storing them in DB

- added a get\_playlist function

- see more here:

<https://developer.spotify.com/documentation/web-api/reference/#/operations/get-a-list-of-current-users-playlists>

Spotify to JSON

<https://www.youtube.com/watch?v=Bk90lT6ne3g>

Getting Genre:

<https://stackoverflow.com/questions/41224070/spotify-javascript-api-getting-the-genre-of-a-song-in-a-playlist>

Use this to get genre instead?

<https://betterprogramming.pub/how-to-extract-any-artists-data-using-spotify-s-api-python-and-spotipy-4c079401bc37>

<https://www.kaylinpavlik.com/classifying-songs-genres/#:~:text=There%20are%2012%20audio%20features,tempo%20%2C%20key%20%2C%20and%20mode%20>.

* Spotify has an audio features...feature that lets you get the liveliness, danceability, etc. of a track so I was thinking we could probably use this instead of doing a double spotify call to get the genre from the artist too