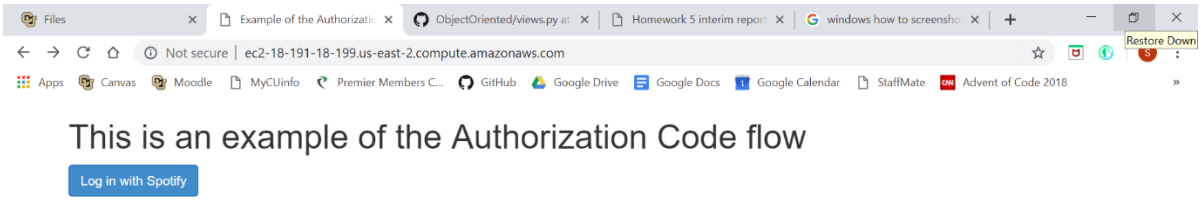
Homework 5: Interim Project Report

Team Members: Sam Berger, Sam Busser, Jasmine Bascom

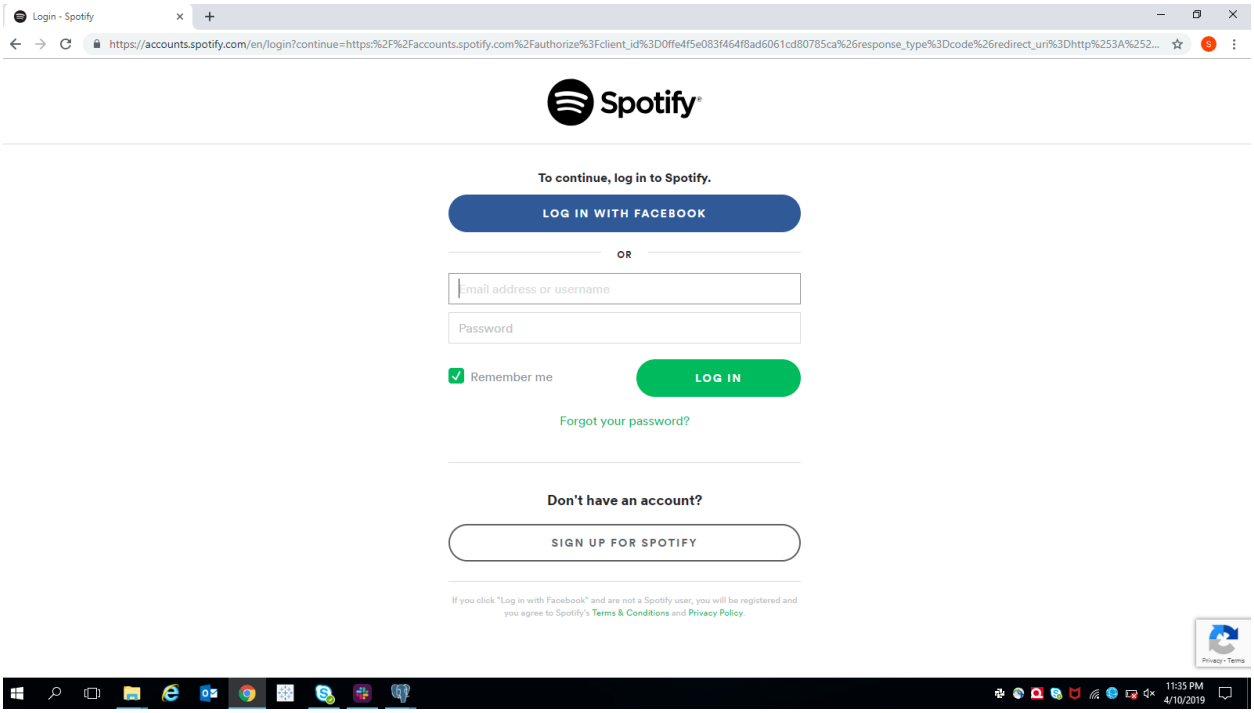
Summary:

- A large part of the first two weeks of our project have been spent setting up our web server. We decided to use Django to run our web application, so we have spent time learning how to use this. In addition to setting up the framework of our site, we have also added code that allows the users to sign in to Spotify and authorize our application to access their profile information. After users log in, we are able to access their profile and get all of their playlists using the Spotify API, Python requests library, and Python Spotipy library. Sam Berger set up the AWS server and the initial site, as well as worked on the logic of signing in the user to their Spotify account and allowing our site to access their info. Sam Busser has helped build out the website through Django and implemented the current python logic that gets the users playlists. Jasmine has been working on the HTML side of the site, dealing with design.
- The next steps for the project are to allow the user to select one of their playlists to be optimized. From there, our app will use the Spotify API and Echo Nest to recommend songs to be added and removed from the selected playlist. We can then add or delete these songs using the Python Spotipy library. In addition to adding this functionality, we need to add some CSS to our HTML code in order to make our site more visually pleasing.
- We have been unable to incorporate design patterns into our current system, as the majority of our work so far has been setting up the web page and accessing user's Spotify accounts.
- Screenshots (keep in mind, as we mentioned earlier, we have not yet added any CSS to make this visually appealing):

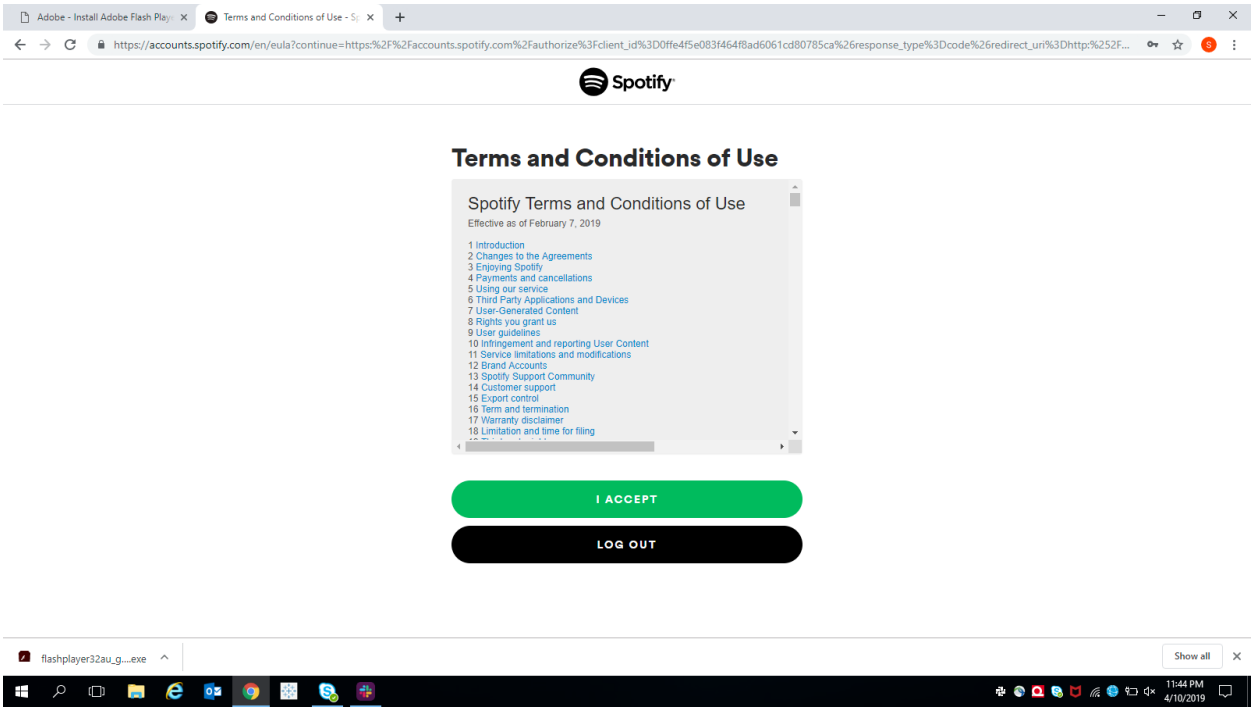
The first page the user is presented



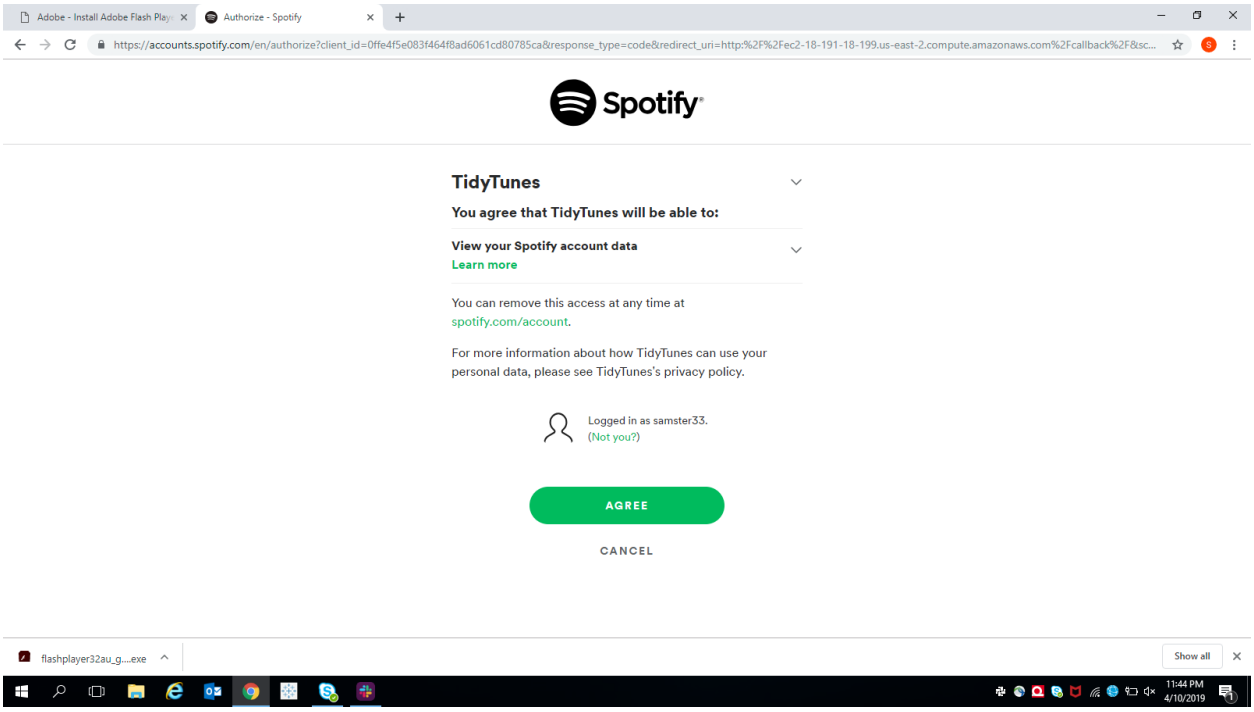
Once the user chooses to log in, they are taken to the Spotify log in page.



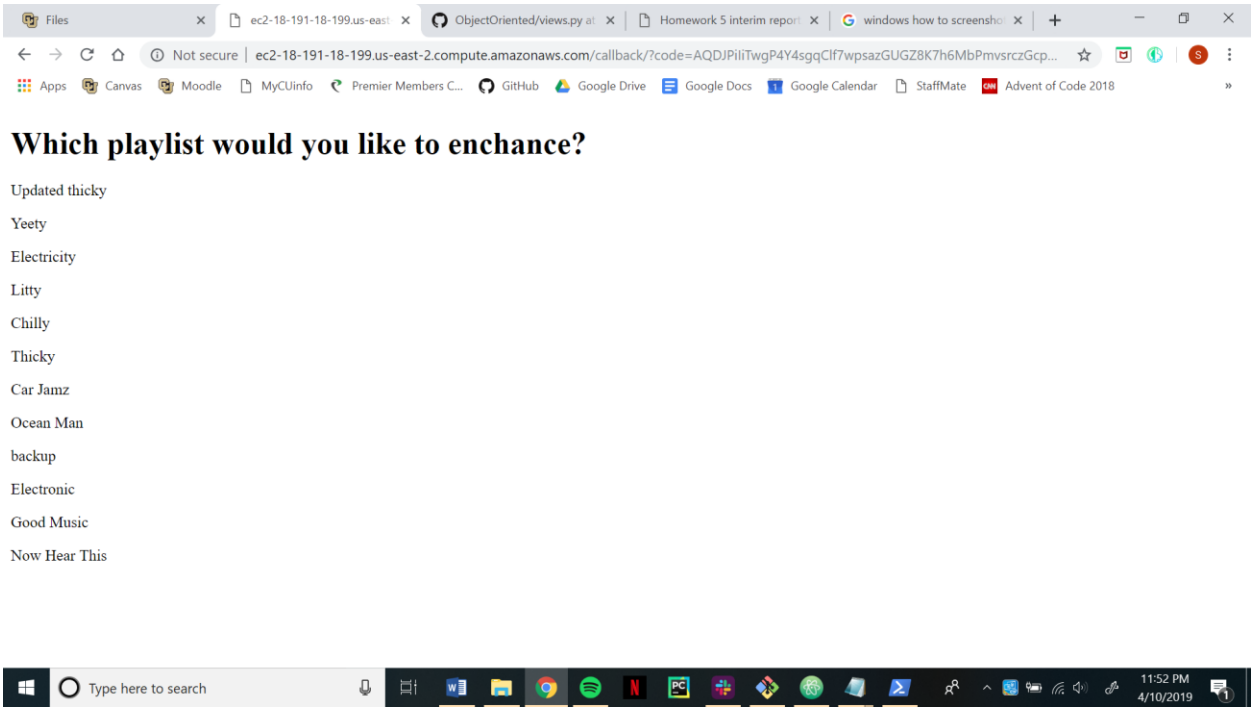
Then they must agree to the terms and conditions



Now they will authorize our app to access their profile



Finally, they are shown a list of their playlists

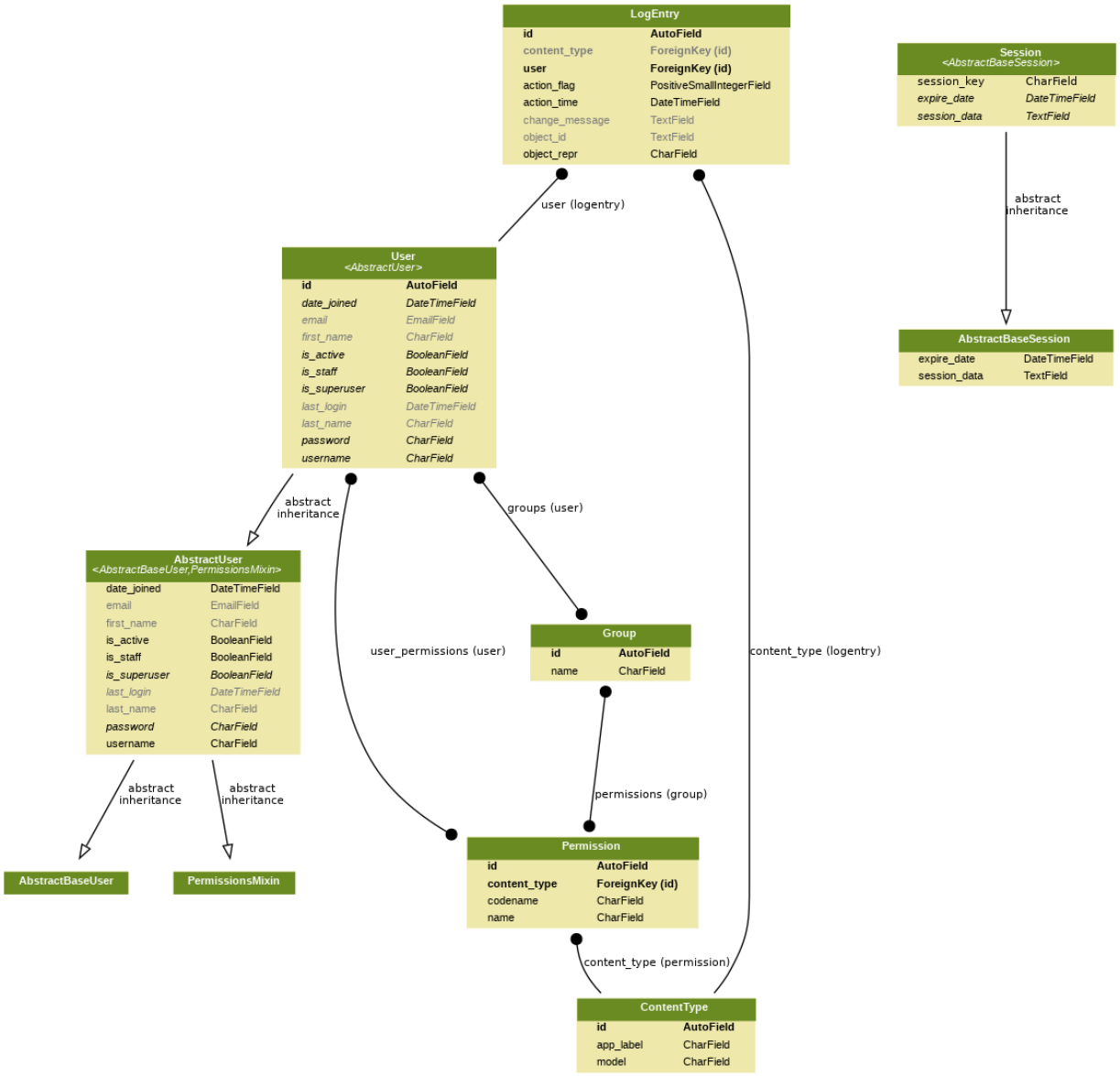


Plan for next two weeks:

As stated in summary, the next step for our application is to give the user the choice of what playlist they'd like to enhance and allow them to click on that playlist. From here we will have to implement our code that runs through the songs on that playlist and recommend a song to be added to removed. We will then present this song to the user and give them the option to accept or deny the suggestion. Most of this logic will be implemented with the help of the Spotify API as well as the Python Spotipy library, while continuing to build out our application with Django and HTML.

Updated Class Diagram:

One thing to note here is that, as stated earlier, a lot of our work has gone into establishing the web site and gaining access to user's Spotify accounts. Thus much of the object oriented work has been in Django itself in how the web app is hosted. Once we get more into the actual logic of the app, we will be able to create code that will follow our original class diagram.


LogEntry
id AutoField
content_type ForeignKey (id)
user ForeignKey (id)
action_flag PositiveSmallIntegerField
action_time DateTimeField
change_message TextField
object_id TextField
object_repr CharField
Session
<AbstractBaseSession>
session_key CharField
expire_date DateTimeField
session_data TextField
user (logentry)
abstract inheritance
User
<AbstractUser>
id AutoField
date_joined DateTimeField
email EmailField
first_name CharField
is_active BooleanField
is_staff BooleanField
is_superuser BooleanField
last_login DateTimeField
last_name CharField
password CharField
username CharField
AbstractBaseSession
expire_date DateTimeField
session_data TextField
abstract inheritance
groups (user)
AbstractUser
<AbstractBaseUser,PermissionsMixin>
date_joined DateTimeField
email EmailField
first_name CharField
is_active BooleanField
is_staff BooleanField
is_superuser BooleanField
last_login DateTimeField
last_name CharField
password CharField
username CharField
user_permissions (user)
Group
id AutoField
name CharField
content_type (logentry)
abstract inheritance
abstract inheritance
permissions (group)
AbstractBaseUser
PermissionsMixin
Permission
id AutoField
content_type ForeignKey (id)
codename CharField
name CharField
content_type (permission)
ContentType
id AutoField
app_label CharField
model CharField