SQL Injection

SpoTV

Fall 2017

Overview:

SpoTV is a web app that allows users to easily convert their spotify music playlists to YouTube music video playlists through a streamlined, easy to navigate interface. This functionality could prove useful to people for a number of different purposes. It gives them a way to watch the videos that go along with their favorite songs without having to search each one individually or manually construct a youtube playlist, a tedious process, especially if the collection of songs already exists in Spotify. SpoTV could be used by party hosts to provide imagery to go along with their preconstructed party playlists. It could even help users discover music videos they were unfamiliar with before.

Besides using the highly versatile, modern framework, Django, our app's innovation comes from the integration of external services. Our use of the Spotify and YouTube API's allows users to link their respective accounts along with their SpoTV login. This feature, along with our modern, stylized UI, is a prime example of what is possible in the world of Web 2.0.

Once a user has a SpoTV account, they will need to link both their Spotify and Google accounts. After this is done, our main page will populate a sidebar with a list of their Spotify playlists. A user can then select any playlist from this sidebar and use it to generate a YouTube playlists. Once this is completed, our site will save the created playlists so they can be viewed later, without having to re-generate each one. These video playlists are saved to the user's google account. Our main page also has an embedded youtube video player that we use to load the video playlists, allowing for viewing. Thus our app has a highly dynamic homepage packed full of functionality that maintains understandability and ease of use.

Team Members:

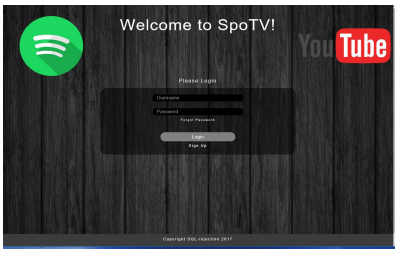
Team Member Name	Github Usernames
TEAM MEMBER #1 Harry Hause	hhause19
TEAM MEMBER #2 Trevor Brown	trevortrev11
TEAM MEMBER #3 Ruifeng Zhang	RUI123

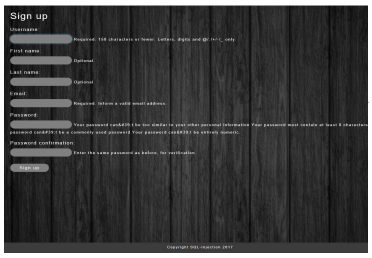
TEAM MEMBER #4 Anthony Boccadoro	aboccadoro
TEAM MEMBER #5 William Sattanuparp	willsattanuparp
TEAM MEMBER #6 Alex Sellers	acelery420

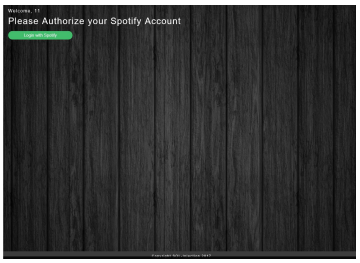
Github Repository: https://github.com/hhause19/326webprogrammingproject

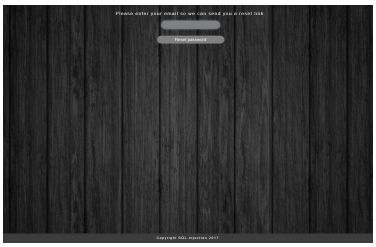
User Interface:

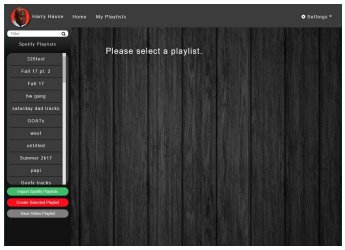
- Login page: to login to the homepage.
- Signup page: to create an account
- Signup Successfully page: to link to the link for spotify
- Forgot Password page: to reset the password
- User homepage: to allow user access all own playlists and other mock UI pieces.
- Playlist Page: to show all the playlists
- Play Page: to play the songs in the list
- Account Information Page: to change the account profile
- Preference page: to show all songs which has in SpoTV database.

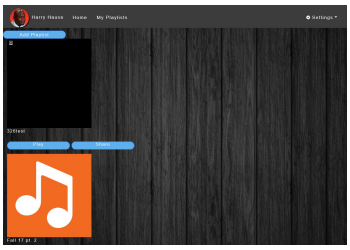


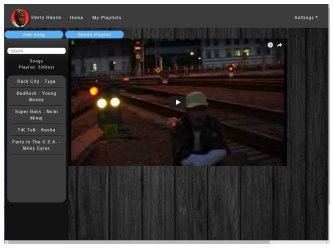


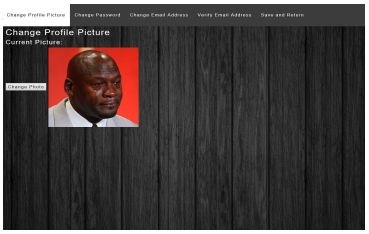




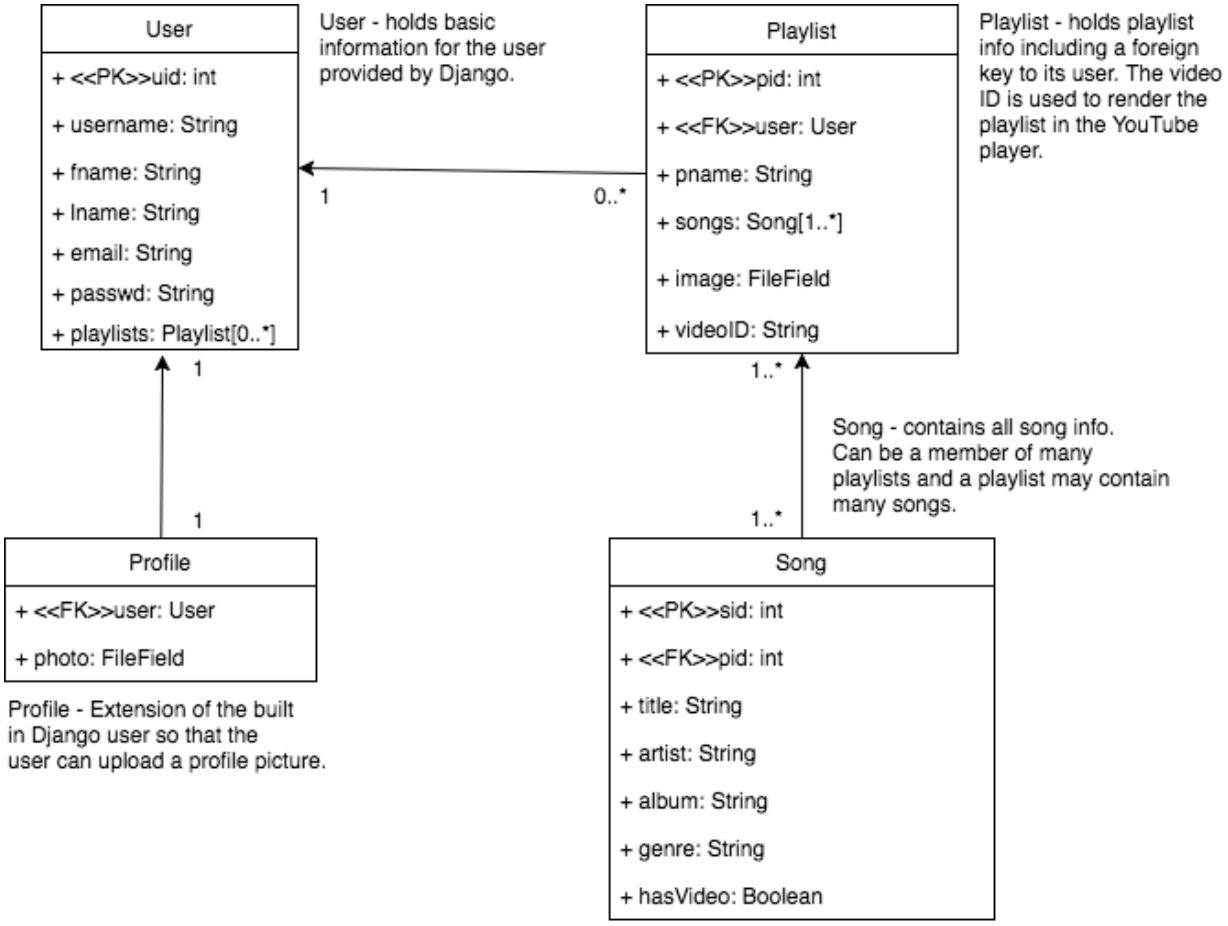








Data Model:



URL Routes/Mappings:

http://127.0.0.1:8000/SpoTV/login/
Our site's login page.
http://127.0.0.1:8000/SpoTV/login/signup/
Page that allows users to create an account with our app.
http://127.0.0.1:8000/SpoTV/

This is the homepage for our app. It contains most of the working functionality. Once a user has linked their spotify account, a sidebar will display their Spotify playlists. If a user has also linked their google account, a user will be able to generate a Youtube playlist. Once generated, the video playlist will load in the Youtube video player in the center of the screen. A user can switch between playlists once they have been created.

http://127.0.0.1:8000/SpoTV/myplaylists/

This page displays a list of playlists that have been generated by a user. From here, a user can select a playlist to listen to, and they will be brought to the following page.

http://127.0.0.1:8000/SpoTV/myplaylists/*n*/

This page is similar to the main page in look, however it pertains to the specific playlist selected in the myplaylists page, specified in the url by a number, *'n'*. Here we also have a sidebar and embedded Youtube player. Instead of a list of Spotify playlists, the sidebar contains a list of songs in the selected playlist. From here a user could add videos directly into the youtube playlist or delete the selected playlist. This page is not fully functional at this time.

http://127.0.0.1:8000/SpoTV/accinfo/

This is the account info page. It displays customizable user data, or in our case, a profile picture

http://127.0.0.1:8000/SpoTV/password/

Displays the password change form.

http://127.0.0.1:8000/SpoTV/preference/

This is the page to show all the songs, by a table of title, artist, genre, and album, in the database of this SpoTV. Users can uses checkbox to mark the song, look through the songs, and return to homepage.

http://127.0.0.1:8000/SpoTV/spotify_auth/

This page promotes a user to link their spotify account. It will bring them to an external page managed by Spotify where they log in with their credentials and authorize our app to access their account.

http://127.0.0.1:8000/SpoTV/spotify_auth_success/

This page displays a short message indicating a successful Spotify login

http://127.0.0.1:8000/SpoTV/youtube_auth/

Similarly to the Spotify authorization page, this will prompt a user to login to their Google account, again directing them to an external login page and requesting access to their Youtube information.

http://127.0.0.1:8000/SpoTV/youtube_auth_success/

Here we indicate a successful Google account link.

Authentication/Authorization:

To login and begin using SpoTv, (compsci326/compsci326). Once logged in, there will be a settings drop down tab that reveals the logout button which, when clicked, links to the login page. If the password is forgotten, the user can click the forgot password button and be linked to the appropriate page to enter the associated email address with an account in the database. If the user has no account, they can create an account by clicking the sign up button and filling in the fields. After proceeding, the user is prompted to authorize the use of their spotify account on the site, but is unimplemented. When the user is logged in, they can view mock song data and playlist data on the left panel and by clicking the playlist tab in the navigation bar. When the user is able to authorize their spotify account they will be able to add playlists and interact with actual data.

Team Choice:

Our initial choice was to implement the email verification along with the spotify login. However, as the project progressed, we realized that using the Spotify API and YouTube API were top priorities because they were necessary for the core functionality of our site. We were able to successfully implement the two of them but were unable to complete the email verification by the time of the final submission. The email verification will now be a future requirement for our project.

Conclusion:

Overall, our experience with working with our project was an extremely educational one. We learned a lot from our time working together on this project. At the most fundamental level was how we learned to work with each other. There were times at the start where we were unorganized and unsure of our own jobs. Once we started to barely meet deadlines was when we decided to organize ourselves and divide roles better. This helped us in decreasing the overall workload on the programmers that had contributed the most. Another thing we learned was the power of django. Django allows for easy creation of models, views, and urls. Django is a valuable tool to have in a programmer's arsenal and will certainly be used in our future personal projects.