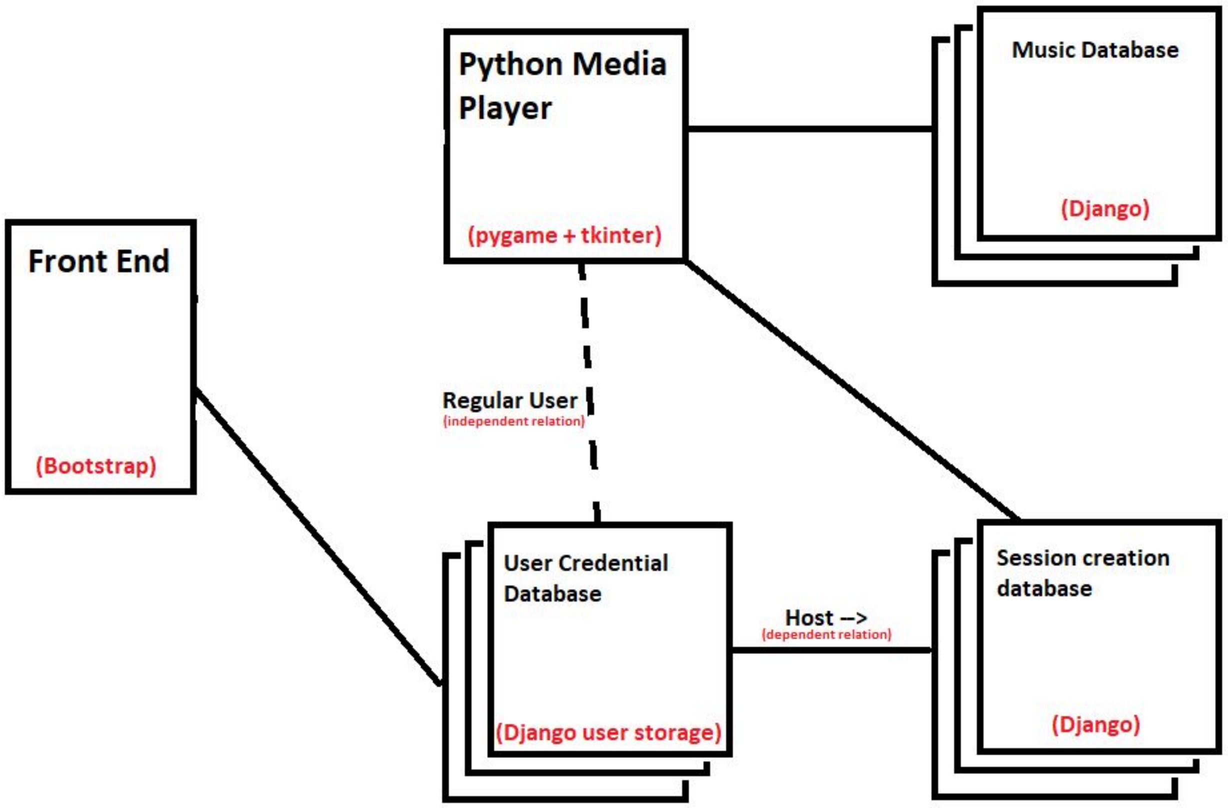
Updated FEATURES list:

(removed Personal Preferences/Genre Grouping)

1. **Saving and updating the music resources into the database(API)** - the front end of the application will be connected with the music resources in the database. We are starting out with a static database with a limited song pool in order to get a working music sharing application and then we could try to piggyback off of another services API to get a larger song pool.
2. **Security** - The application asks the users to login in order to receive all functions of the application. All the login information will be encoding and saved in the server.
3. **Generate temporary sessions using code**- The application can produce group codes/ QR codes for multiple users to join a protected group session which is terminated after use.
4. **Queuing music in a priority queue** - this is determined by the people connected

Architecture Diagram:



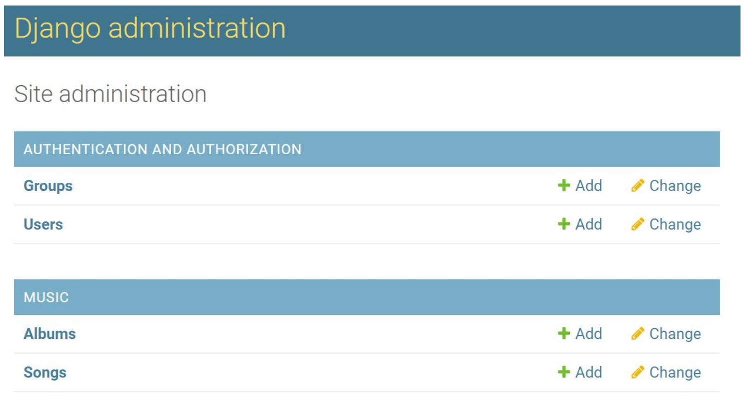
Front End Design:

- We've created a Figma model to base our front-end on. We will be creating the front end with Bootstrap and HTML.

Database Design:

- The database is created in Django. Django is an open source python extension web-framework that takes a lot of the complexities out of designing database driven websites. The reason for this is to keep everything within our framework contained in the environment. This environment allows us to include the necessary python and html or any other front-end scripts.
- The database itself is actually very simple thanks to Django's unique design. Built into a local server is an automatically generated '/admin' URL extension that allows very easy access to the database.

Admin Page



Django administration

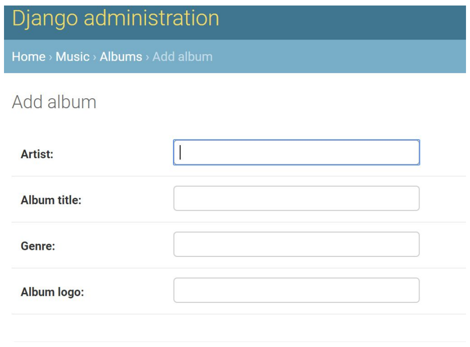
Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

MUSIC		
Albums	+ Add	Change
Songs	+ Add	Change

- Currently the database is set up with two tables: 'Albums' and 'Songs' which have a foreign key relationship through the album name. Each album has an 'artist' column, an 'album_title' column, a 'genre' column and an 'album_logo' (this will be an image file for the album that will be used in the html scripts) column. Each song table has an album foreign key, a 'file_type' (the mp3 file of the song) column and a 'song_title' column.

Add to album table



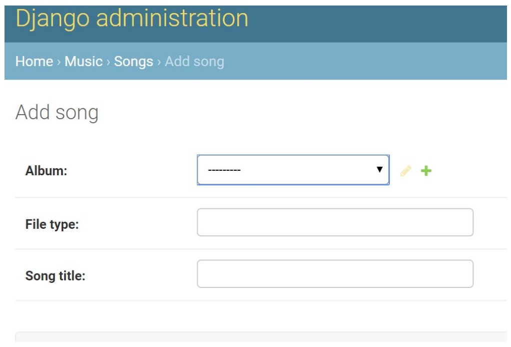
Django administration

Home › Music › Albums › Add album

Add album

Artist:	
Album title:	
Genre:	
Album logo:	

Add to song table



Django administration

Home › Music › Songs › Add song

Add song

Album:	---------
File type:	
Song title:	