# Webapps Final Project Specification

---

**Team Information**
Team #: 26
Team Members: Vishal Baskar, Minji Kim, Joanne Park, Iris Wang

---

**Product Backlog**

Authentication with Spotify:

- Application registered with Spotify developers
- Ability to connect with and authenticate to a user's Spotify account
- Ability to pull the user's playlists and play songs using the API

Music Rooms:

- Music is played by a host in a room, users can join any room that is currently being hosted listen to music being played live by the host
- Unique playlist themes for each room
- Live song queue that users can view, which is managed by the host

Chat Feature

- Real time chat room functionality that updates every second
- Shows all chats of all members in a room

Following friends

- Users can follow DJ's that they want to hear more songs from
- Friends will show higher up on the list of currently playing hosts

---

**Sprint 1 Product Owner**
Iris Wang (imwang)

---

**Sprint 1 Backlog**

Work Allocation: (J: Joanne, I: Iris, M: Minji, V:Vishal)

- Set up environment for Django project utilizing Spotify API (M)
- Create an application on Spotify for Developers (I)
- Utilize spotipy package as a python wrapper for API with application credentials (M, I, J)
- Enable Spotify user account authentication (M, I, J)
- Retrieve playlists from authenticated user account (M, I, J)
- Finish wireframe mockups for the application (M)
- Decide which data models and fields are used within the application (J, V)
- Write the project specification (I)

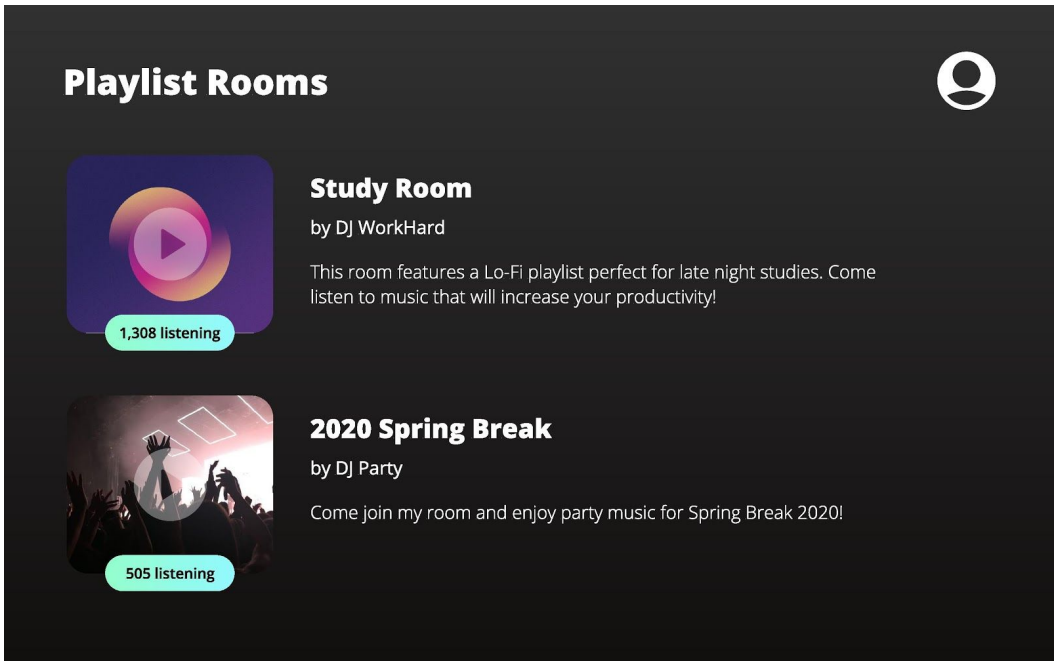
## Data Models

```
from django.db import models
from django.contrib.auth.models import User

# Create your models here.
class Profile(models.Model):
   username = models.OneToOneField(User, on_delete=models.CASCADE)
   spotify_id = models.CharField(blank=True, null=True, max_length=500)

   def __str__(self):
       return 'Post(id=' + str(self.id) + ')'
class Room(models.Model):

   dj = models.OneToOneField(Profile, on_delete=models.CASCADE)
   play_status = models.BooleanField()
   time_stamp = models.IntegerField()


class Song(models.Model):
   spotify_song_id = models.CharField(blank=True, null=True, max_length=500)
   artist = models.CharField(blank=True, null=True, max_length=500)
   title = models.CharField(blank=True, null=True, max_length=500)
   album = models.CharField(blank=True, null=True, max_length=500)
   room_song = models.OneToOneField(Room, on_delete=models.CASCADE,
related_name='current_song')
   associated_room = models.ForeignKey(Room, on_delete=models.PROTECT, blank=True,
null=True, related_name='song_queue')
   def __str__(self):
       return 'Post(id=' + str(self.id) + ')'
```

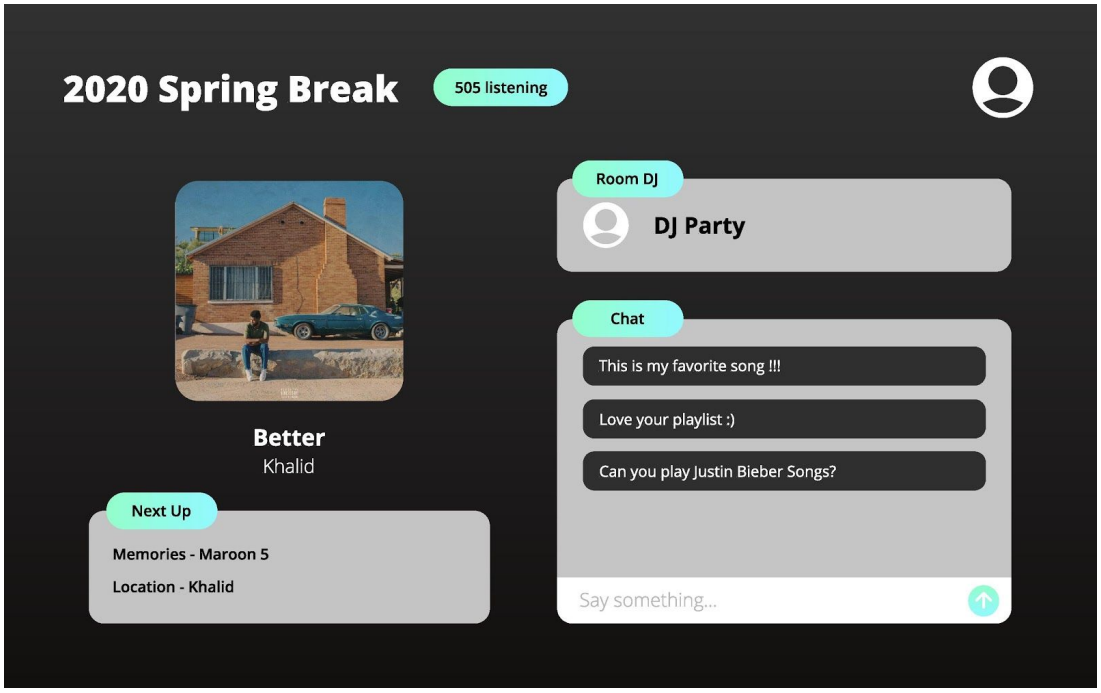
## Wireframes

- Login page

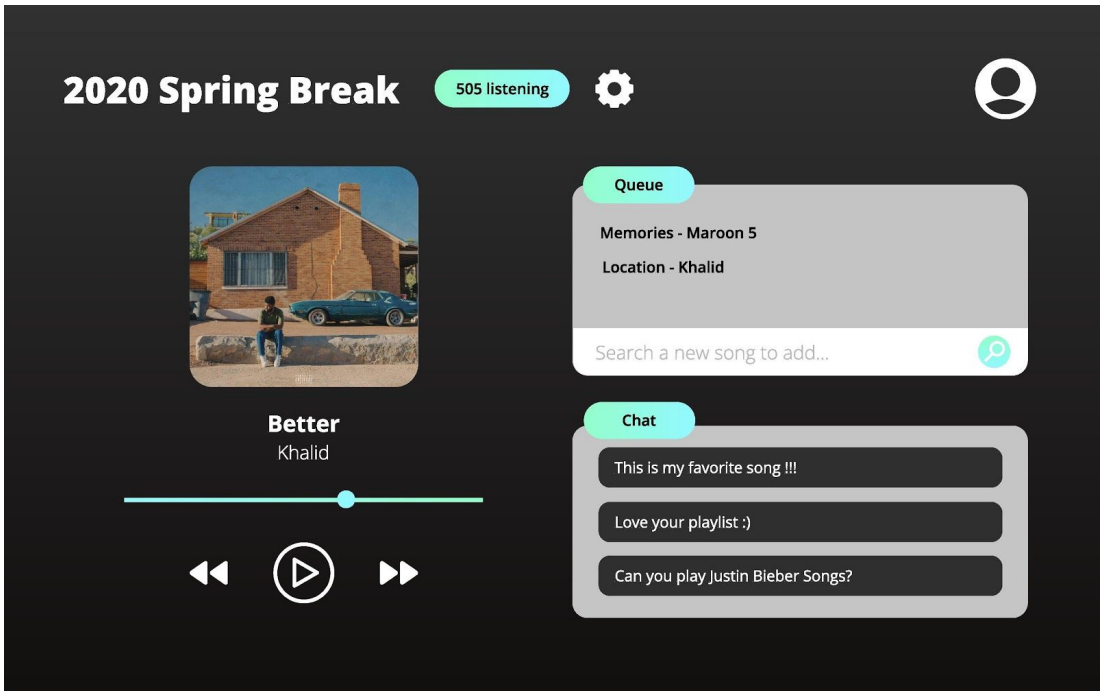
- Home page to access all of the rooms to go into



- The actual room which contains the currently playing song, chat, song queue (for normal user)




- The actual room which contains control button, spotify api search box, song queue, end/close room button, room settings (for DJ)



**Sprint 2 Product Owner**
Iris Wang (imwang)

---

**Sprint 2 Backlog**

Work Allocation: (J: Joanne, I: Iris, M: Minji, V:Vishal)

- Create a new Spotify web player connected to a logged-in user's account (M, I)
- Fully implement the player with currently playing song information displayed and play/pause functionality (M, I)
- Display the upcoming song queue next to the player, which gets updating according to the logged-in user's Spotify queue (M, I)
- Incorporated Boostrap to develop front-end features for the sign-in page and the player (M, I)
- Created the search functionality to filter out songs based on track names and artist names. (J, V)
- After getting search results, DJ is able to add to a queue of songs (J, V)