**Team:**   Ziyang Yang
           Yifan Li
           Nicholas Montoya


**Title:** Music Generation


**Project Summary:** After signing in with Spotify, we want users to select songs from their libraries and get auto-generated playlists returned back to them. Users can select whether they want a playlist to be generated based off of the songs genre or the time it was produced. After seeing the playlists, users can then add them directly into their personal Spotify accounts.

**Project Requirements:**

| Business Requirements | | | |
|---|---|---|---|
| **ID** | **Requirement** | **Actor** | **Priority** |
| **BR-01** | **All actors need to have a special user name** | **User, Admin** | **High** |
| **BR-02** | **Admin only can update eight popular lists** | **Admin** | **Medium** |
| **BR-03** | **User cannot delete or edit other user's post playlists** | **User** | **High** |


| User Requirements | | | |
|---|---|---|---|
| **ID** | **Requirement** | **Actor** | **Priority** |
| **UR-01** | **As a Music Generation web user, I need to be able to search for songs by name so I can find them efficiently** | **User** | **High** |

| UR-02 | As a Music Generation web user, I want to be able to add auto-generated playlists to my Spotify account | User | High |
|---|---|---|---|
| UR-03 | As a Music Generation user, I want to auto-generate playlists based on the year the music was made | User | Medium |
| UR-04 | As a Music Generation user, I want to be able to find music based on the genre | User | Medium |
| UR-05 | As an Admin, I want to be able to monitor backend performance in order to be notified of failed requests | Admin | Low |
| UR-06 | As a Admin of Music Generation web, I need to be able to update the popular playlists page based on what users are listening to | Admin | High |
| UR-07 | As a Music Generation user, I want to be able to view the popular music playlist | User | High |
| UR-08 | As Admin of Music Generation web, I need to be able to remove old/irrelevant playlists | Admin | High |
| UR-09 | As Admin of Music Generation web, I need to be able to remove harmful users | Admin | Medium |
| UR-10 | As a Music Generation user, I need to be able to sign up in system | User | High |
| UR-11 | As a Music Generation user, I need to be able to login into the system | User | High |

| Functional Requirements | | | |
|---|---|---|---|
| ID | Requirement | Actor | Priority |
| FR-01 | System should allow people login with their Spotify account | User, Admin | High |
| FR-02 | System should notify user if they have a long time not add any songs or playlists into their accounts | User, Admin | Medium |

| Nonfunctional Requirements | | |
|---|---|---|
| ID | Requirement | Priority |
| NFR-01 | User shall see the popular playlists page in under 5 seconds | High |
| NFR-02 | User account number and passport must store in a secure manner | High |

**Use Cases:**

      **Use Case Overview:**

The main Use Case Diagram below depicts our whole system.
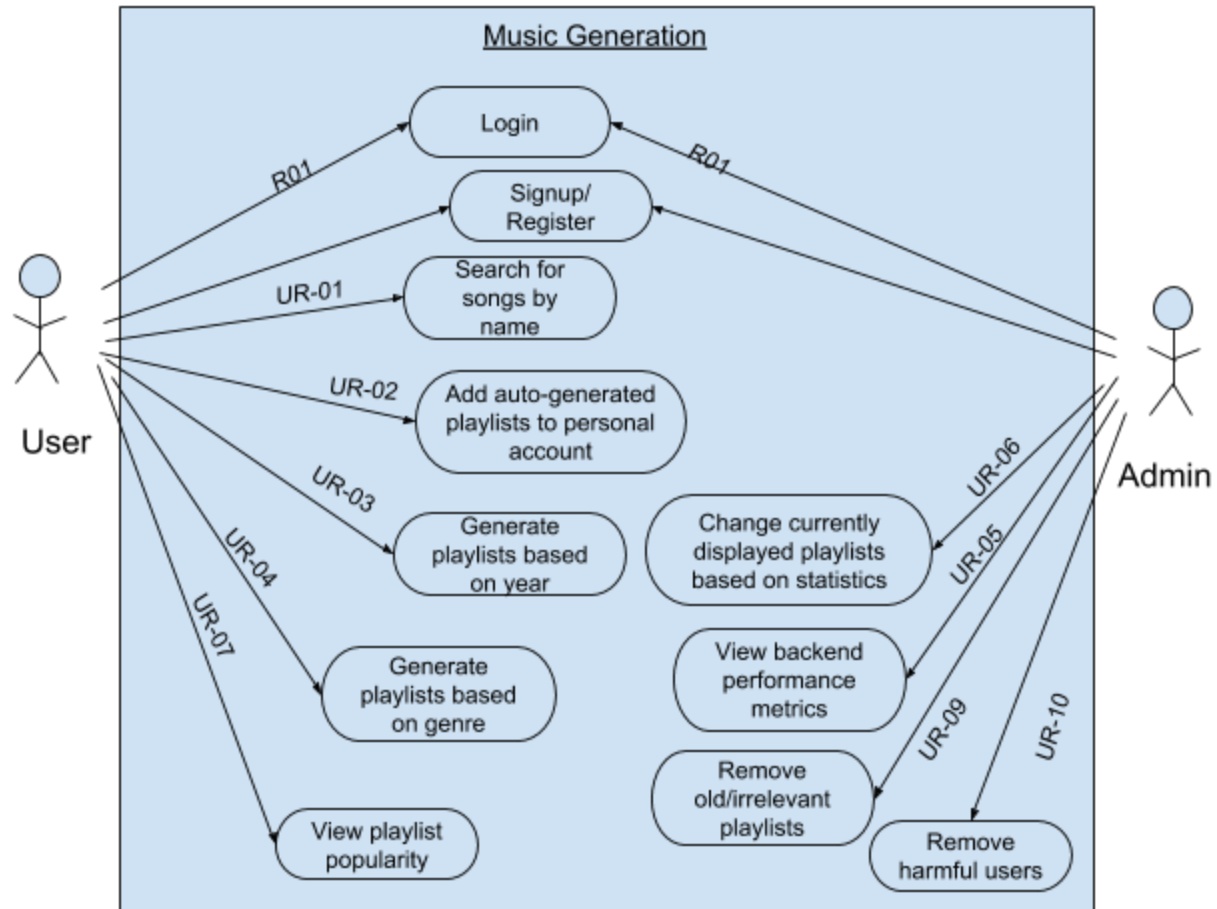
The sub use case diagram below is for generating playlists based on the year of a known song. This is exactly the same for generating a playlist based on genre, just searching for top songs based on genre.
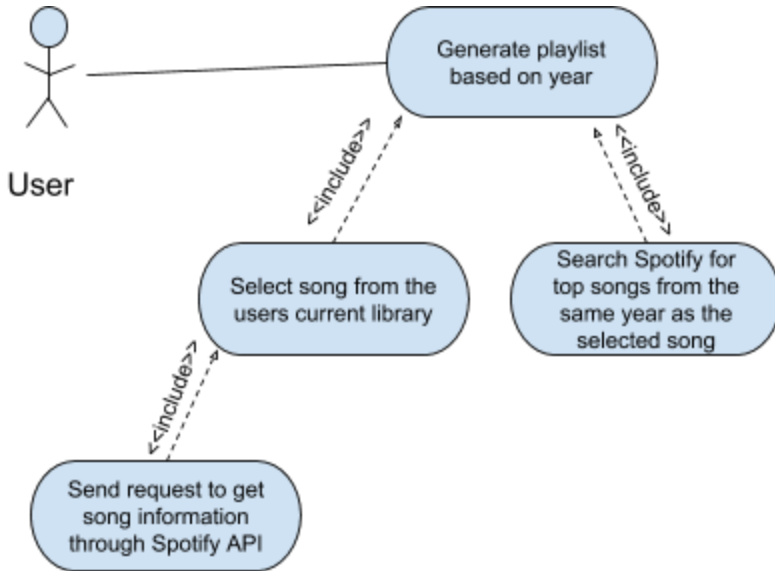
**Use Case Description:**

**Add auto-generated playlists Use Case**

| | |
|---|---|
| **Use Case ID:** | UC - 01 |
| **Use Case Name:** | Add Auto-generated playlists |
| **Description:** | Add auto-generated playlists into user's Spotify account. |

| | |
|---|---|
| **Actors:** | User |
| **Pre-conditions:** | Users are logged in to the system with their Spotify account, and system can create auto-generated playlists based on user's Spotify library. User is notified of auto-generated playlists and willing to listen. |
| **Post-conditions:** | User adds the playlists into their account. |

| Frequency of Use: | At Will by Users | | |
|---|---|---|---|
| **Flow of Events:** | | **Actor Action** | **System Response** |
| | 1. | User clicks on the checkbox of different auto-playlists which he/she wants to listen | System searches the Spotify database for popular songs based on user input |
| | 2. | | Forms as many playlists of 20 songs as possible ranked by popularity |
| | 3. | User clicks the specific one playlist to get in the playlist | Access the chosen playlist, and showing the songs inside of the playlist |
| | 4. | Users click the Add button to add the playlist into their accounts | Send request to add playlist through Spotify API |
| | 5. | | Add playlist to site database to show other users |
| **Variations:** | 1.     When the user accidentally select the unwanted playlists, user could click the checkbox again to cancel the selection | | |
| **Notes and Issues:** | | | |
| **Developer Notes:** | | | |

**List the popular playlists Use Cases**

| Use Case ID: | UC - 02 |
|---|---|

| Use Case Name: | List the popular playlists |
| --- | --- |
| Description: | As an Admin, I need to update to popular playlists based on what is popular. |

| Actors: | Admin, External Listing webpage |
| --- | --- |
| Pre-conditions: | System has records of the times the playlists have been played. |
| Post-conditions: | User's records is updated in the system in ordinal. The popular playlist is showing following the order. |
| Frequency of Use: | Daily update requirement |

| Flow of Events: | | Actor Action | System Response |
| --- | --- | --- | --- |
| | 1. | Admin collects the records of playlists history from the system. | System will show the number of times a playlist has been listened to. |
| | 2. | Admin views the records | |
| | 3. | Admin adds up to eight popular playlists daily by using the records | System will show the eight popular playlists in the 'Popular List' |
| | | | |

| Variations: | |
| --- | --- |
| Notes and Issues: | |
| Developer Notes: | There will only be eight popular lists updated each day. |

**Activity Diagram:**

- Requirement ID : UR -02
- Use case ID: UC - 01
- User case name: Add Auto-generated playlists (Add auto-generated playlists into user's Spotify account.)

| Actor | System |
|-------|--------|

**Click Checkbox** → Check input validity — No

Check input validity — Yes → View criteria → Search in system → Return multiple playlists

See multiple playlists → Does user like these playlists

Does user like these playlists — No → (end)

Does user like these playlists — Yes → Click like button → Receive song list → Add them to user's account → Store in site database to display to other users → View the playlist
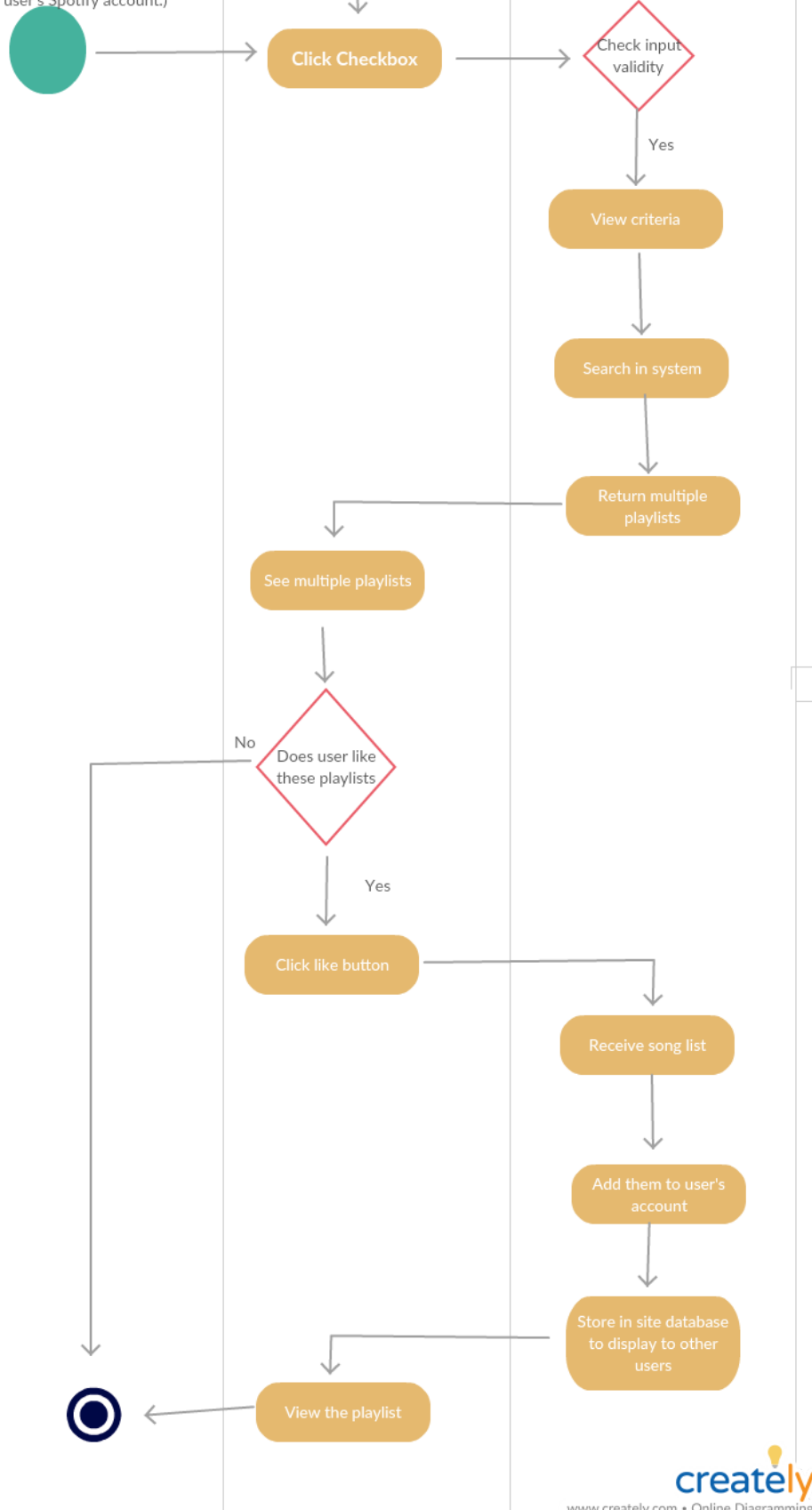
- Requirment ID: UR - 06
- Use case ID: UC - 02
- User case name: List the popular playlists (As an Admin, I need to update to popular playlists based on what is popular.)

| Actor | System |
|---|---|
| Enter Admin interface | |
| Ask system for updated songs info. | |
| | collect all information about songs for a day |
| Get all information | |
| Sort by total number of play times | |
| | Create a empty list for top 8 songs |
| | add the list to recommendation system |
| | add to user's main page |
| can access these songs | |
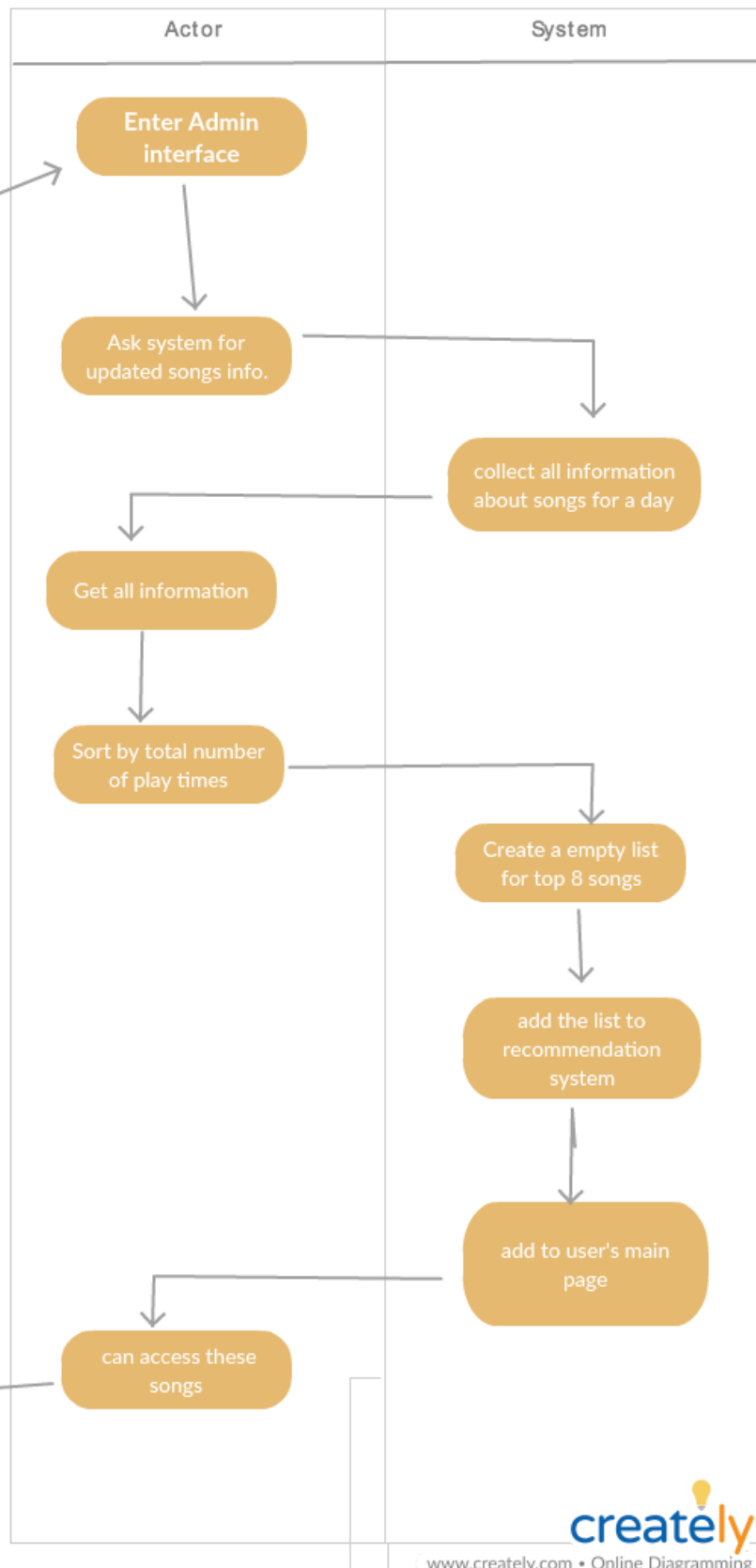
**Data Storage:**

  We will be using MySQL Database Management System to store and retrieve messages. The database will contain all the details of user, admin, playlists, time records and so on.

**UI Mockups:**

**Page before user login to the website**
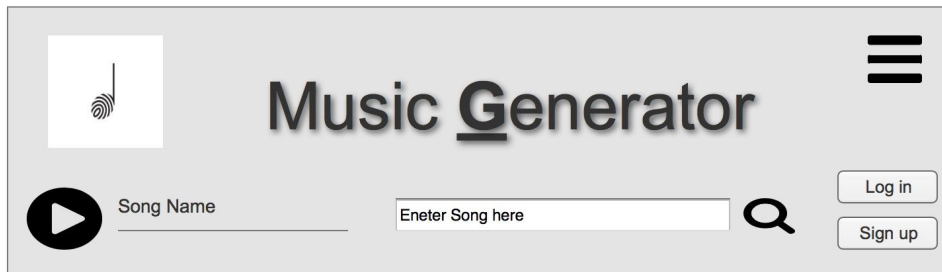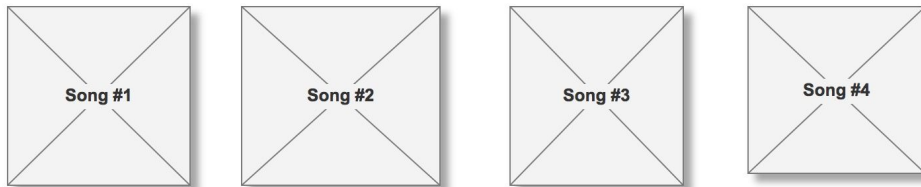
**Page after user login to the website**

# Music **G**enerator

Song Name

Eneter Song here

Log out

Sign up

**Create your own music list by one click:**

Hello, user!

**Library**:

~ Genre ☐
~ Year ☐
~ Singer ☐
~ Location ☐
~ Popular List ☐

**Playlist created:**

~ Playlist 1 ◯

**Saved List:** ⊞

~ List 1 ✖
~ List 2 ✖

Every day songs

Random user
posted list 2

Random user
posted list 1

Random user
posted list 3

**Login page**

| Music Generator | Click link to your Spotify account |

Email

Password

☐ Remember you?

Log in

Don'r have an account? **Sign up**

**Sign Up page:**

## Music **G**enerator

**Sign up with your Email account**

Email

Confirm Email

Password

User Name

DATE OF BIRTH

| Month ▼ | Day ▼ | Year ▼ |

○ Male   ○ Female

☐ I agree with all the requirement and conditions by Music Generator

**Sign Up**

Already have a account? **Log in**

**After you search a song**

# Music Generator

Song Name

Eneter Song here

Log out

| Name | Singer | Album | Time |
|------|--------|-------|------|

**User enters a playlist**

# Music Generator

Song Name

Eneter Song here

Log out

Hello, user!

**Name of playlist**                                    Created by Creator

**Library:**

~ Genre ☐

~ Year ☐

~ Singer ☐

~ Location ☐

~ Popular List ☐

**Playlist created:**

~ Playlist 1 ○

**Saved List:** ⊞

~ List 1

~ List 2

**User select their own playlist**

# Music **G**enerator

Song Name
_____

| Eneter Song here |

Log out

Hello, user!

**Library**:

~ Genre ☐
~ Year ☐
~ Singer ☐
~ Location ☐
~ Popular List ☐

**Playlist created:**

~ Playlist 1 ○

**Saved List:** ⊞

~ **List 1**
~ **List 2**

| | Name | Singer | Time | Delete | Share |
|---|---|---|---|---|---|
| ▶ | ⨯ | ⨯ | ⨯ | ☐ | ↱ |
| ▶ | ⨯ | ⨯ | ⨯ | ☐ | ↱ |
| ▶ | ⨯ | ⨯ | ⨯ | ☐ | ↱ |

.
.
.
.
.
.
.
.
.

**User Interactions:**

The first sequence diagram depicts Requirement ID UR-02, Use Case ID UC-01, and the purpose is to add auto-generated playlists into user's Spotify account.
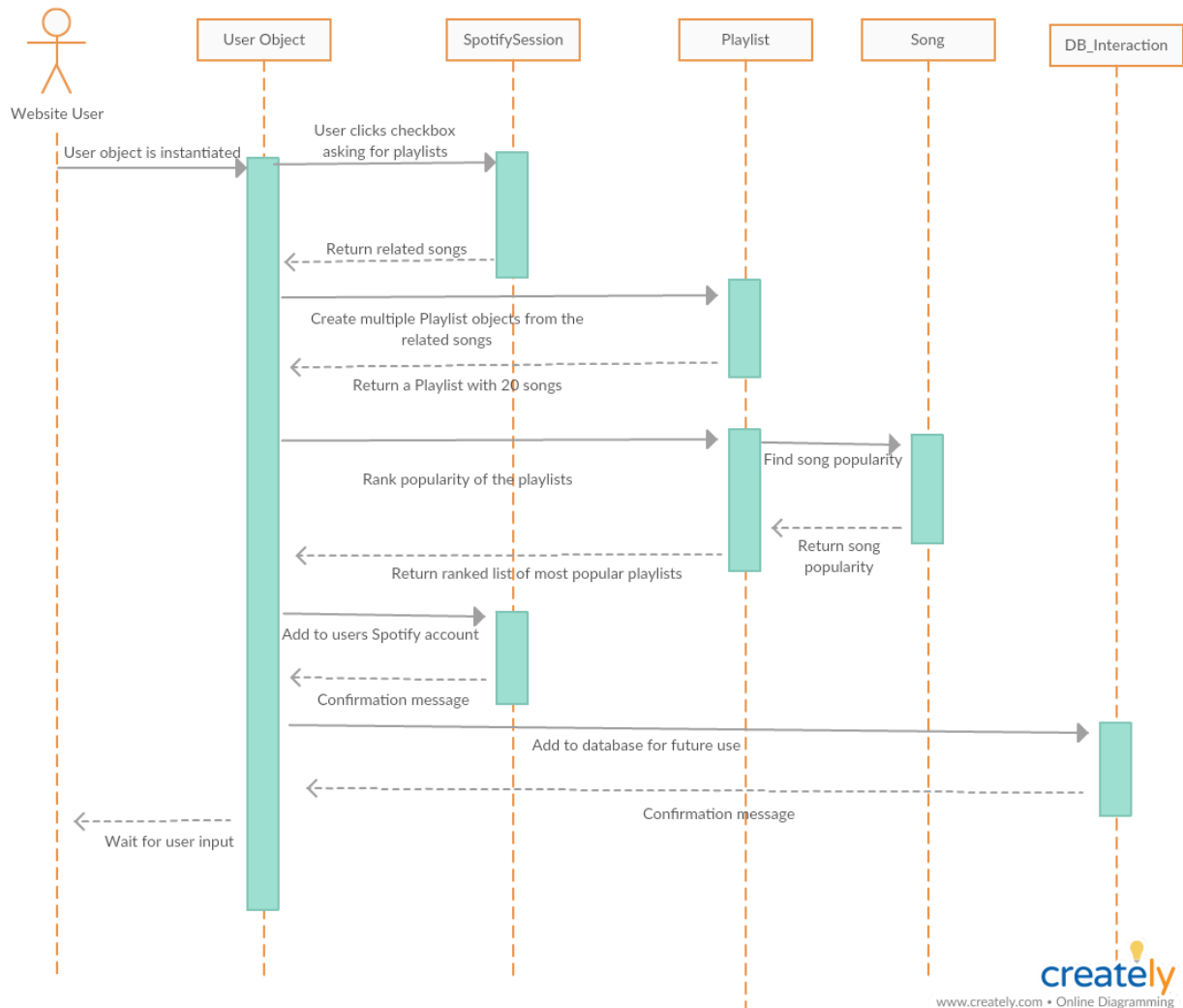
The second sequence diagram depicts Requirement ID UR-06, Use Case ID UC-02, and the purpose is to update to popular playlists based on what is popular.

# Class Diagram:

## SpotifySession
- credentials: file
- session_ID: str
- session_info: Spotify Object

---
+ setCredentials ( username, password ): file
+ login( file )
+ logout( )
+ getInfo( request ): str
+ searchSongs( search_term ): [Song] (list of Song objects)
+ exportPlaylist( Playlist ): str

## User
- user_ID : str
- connection_info: SpotifySession
- user_playlists: [ str ] (list of strings)
- new_playlists: [ str ] (list of strings)

---
+ getUserID( )
+ getPlaylists( )
+ createPlaylists( name, type, [Song] ): [Playlist]
+findRelated( str, SpotifySong ): SpotifySong
+ delPlaylist( name )
+ addPlaylist( name )
+ organizePlaylists( [Playlist] ): [Playlist]

## Admin
curr_displayed: [ Playlist ] (list of playlists)

---
+ addDisplayPlaylist( Playlist )
+ delDisplayPlaylist( Playlist )
+ viewPerformance( ): str
+ viewPlaylistStats( ): { Playlists:  stats} (dict of playlists with statistics as values)
+ updateDisplayedPlaylists( )
+ updateDatabase( )

## Song
+ name : str
+ genre: str
+ artist: str

## Playlist
- name : str
- length : int
- songs : [ Song ] (list of Song objects)
- type:  str

---
+ getName ( ) : str
+ setName ( name )
+ addSong ( Song )
+ delSong( Song )
+ getLength( )
+ calculatePopularity( ): int

## SpotifySong
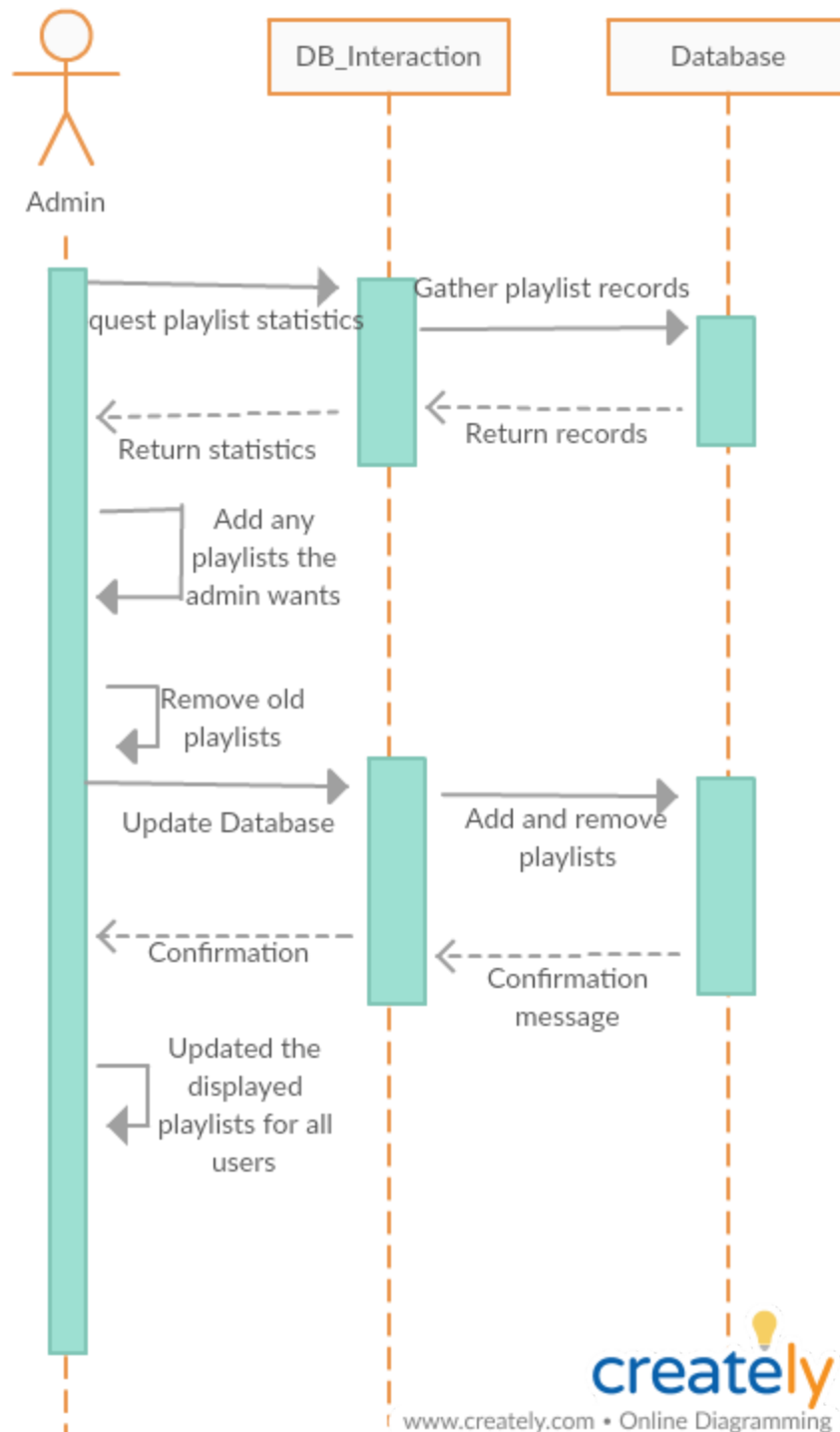- spotify_id : str
- information: {str} (dict of strings)
- extended_info: str
- alt_id: str
- popularity: int

---
+ getInfo( spotify_id ): {str}  (dict of strings)
+ setInfo( {str} )
+ addGenre( )
+ addYear( )
+ addPopularity( )
+ getPopularity( ): int

## DB_Interaction

---
+ addPlaylist( Playlist ): str
+ delPlaylist( Playlist): str
+ addUser( User ): str
+ delUser( User ): str
+ gatherPlaylistRecords( ): { Playlist: times_played }

## Database
- connection_info: file
- users: [ User ] (list of User objects)
- displayed_playlists: [ Playlist ] (list of Playlist objects)

---
+ beginSession( file ): DB_Interaction Object
+ endSession( )

0..1  1..*
0..1
0..1
0..1
1..*
0..1
1..*
1..*
1..*
1..*
0..1
1..*
0..1
1..*
1..*
0..1
1..*