Amy Neeland
Lili Chen
Hunter Ruskanen

INTRODUCTION

Ths system, which we have named "MMDb", or "My Music Database," was made to emulate sites like IMDb but with a music focus instead of movies, meaning the application can function as both a search engine or database and as a social media platform. Like IMDb, users can search for songs within the database and access information such as the artist's name, featured artists, album name and release year.

This system is for the average music listener and list maker. Similar to the IMDb platform, this application allows users to access songs and generate their own unique playlists. Based on a playlist's make up, users will also be able to access song suggestions to match that playlist. While users cannot actually play songs from this application like they would from other platforms, like Spotify or YouTubeMusic, they are enabled to get direct access to streaming platforms via URLs that do offer the selected tracks.

When searching for songs, users can also see the average user ratings and rate songs themselves.

The system does not currently contain information for *all* songs and musicians; however, there is a decent variety of musicians and songs from various genres included for listeners of all genres.

TECHNICAL DESCRIPTION

Language usage:

C# - The user application was composed in C#

XAML - The applications forms and controls were constructed with XAML

SQL - The database accessed by the application was created using SQL

Platforms:

Microsoft SQL Server Management Studio and Azure Data Studio for database generation and API usage

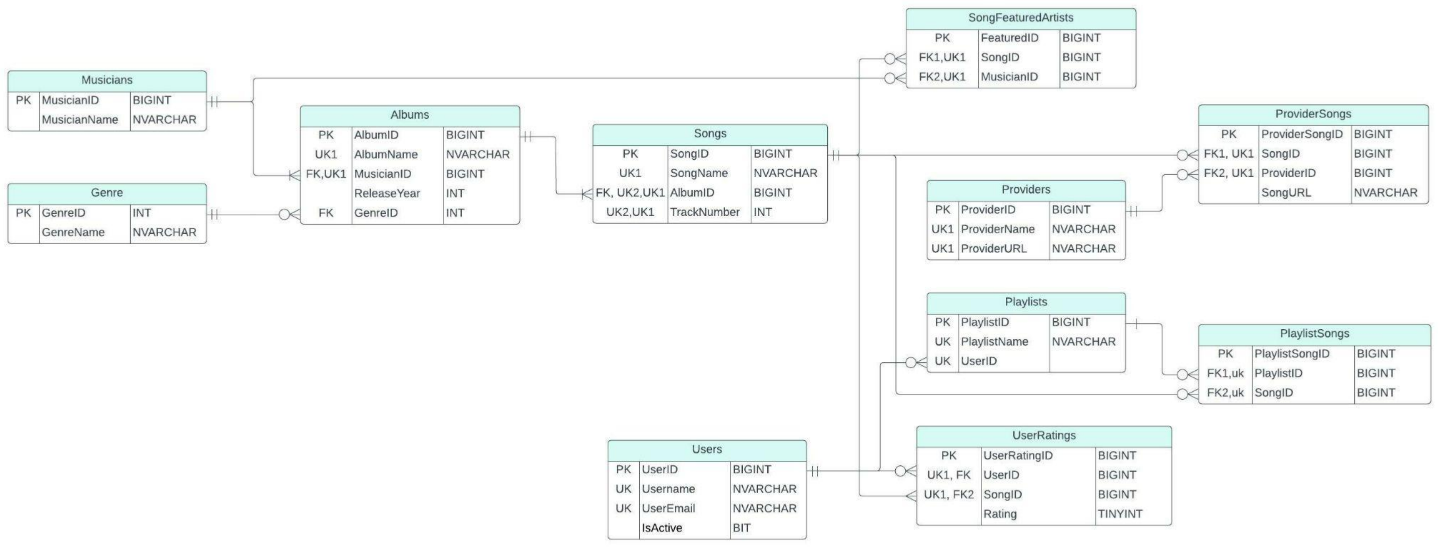
Microsoft Visual Studio WPF project for interface (ASP.Net Core)

Data Population:

For data populating purposes, we relied on some manually inputted data as well as a web crawling system called Octoparse, which helped pull the majority of our music data. We wanted to make sure our data was consistent and inclusive, which led to some manual checks and insertions as well.

We were also able to populate a vast majority of our users, including usernames and emails, as well as user playlists with the help of generatedata.com.

PHYSICAL DATABASE MODEL



Musicians		
PK	MusicianID	BIGINT
	MusicianName	NVARCHAR

Genre		
PK	GenreID	INT
	GenreName	NVARCHAR

Albums		
PK	AlbumID	BIGINT
UK1	AlbumName	NVARCHAR
FK,UK1	MusicianID	BIGINT
	ReleaseYear	INT
FK	GenreID	INT

Songs		
PK	SongID	BIGINT
UK1	SongName	NVARCHAR
FK, UK2,UK1	AlbumID	BIGINT
UK2,UK1	TrackNumber	INT

SongFeaturedArtists		
PK	FeaturedID	BIGINT
FK1,UK1	SongID	BIGINT
FK2,UK1	MusicianID	BIGINT

ProviderSongs		
PK	ProviderSongID	BIGINT
FK1, UK1	SongID	BIGINT
FK2, UK1	ProviderID	BIGINT
	SongURL	NVARCHAR

Providers		
PK	ProviderID	BIGINT
UK1	ProviderName	NVARCHAR
UK1	ProviderURL	NVARCHAR

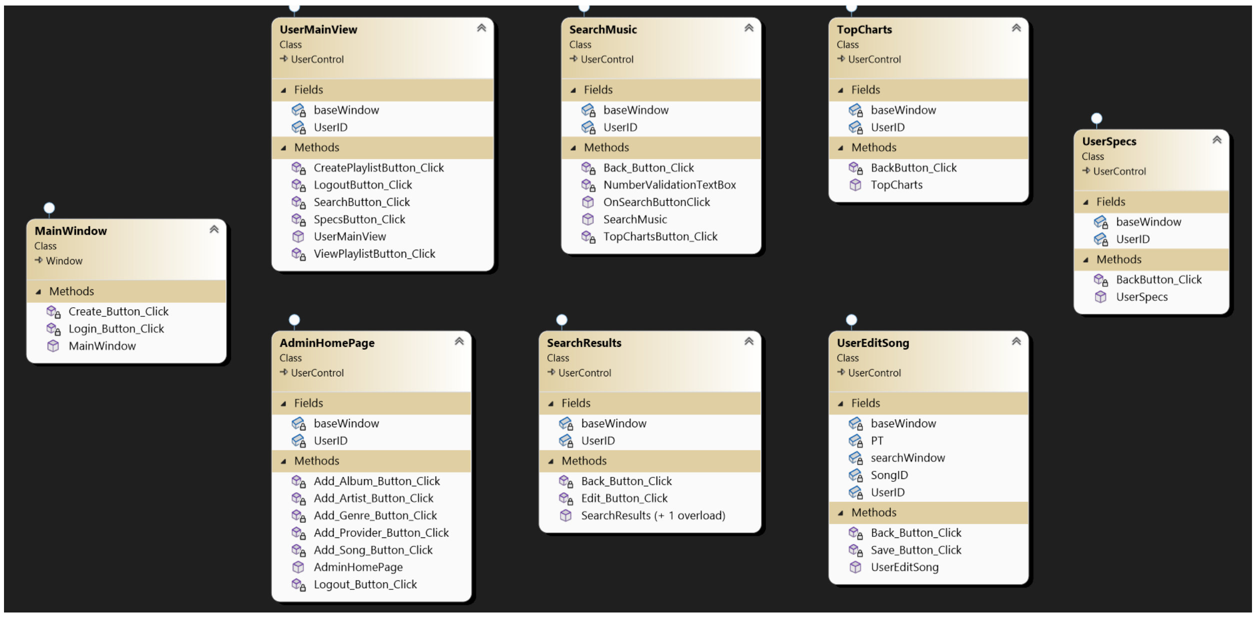
Playlists		
PK	PlaylistID	BIGINT
UK	PlaylistName	NVARCHAR
UK	UserID	

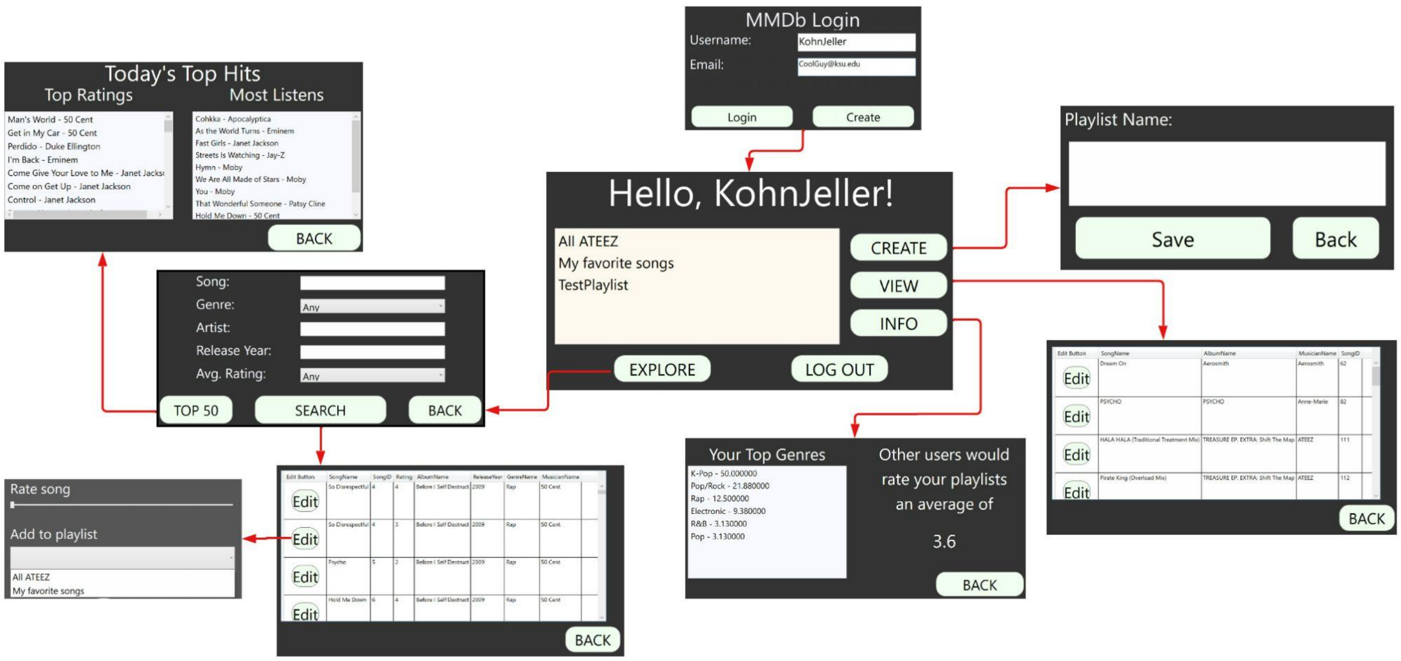
PlaylistSongs		
PK	PlaylistSongID	BIGINT
FK1,uk	PlaylistID	BIGINT
FK2,uk	SongID	BIGINT

Users		
PK	UserID	BIGINT
UK	Username	NVARCHAR
UK	UserEmail	NVARCHAR
	IsActive	BIT

UserRatings		
PK	UserRatingID	BIGINT
UK1, FK	UserID	BIGINT
UK1, FK2	SongID	BIGINT
	Rating	TINYINT

SYSTEM DESIGN





Username: KohnJeller
Email: CoolGuy@ksu.edu

All ATEEZ
My favorite songs
TestPlaylist

Genre: Any
Avg. Rating: Any

Top Ratings:
Man's World - 50 Cent
Get in My Car - 50 Cent
Perdido - Duke Ellington
I'm Back - Eminem
Come Give Your Love to Me - Janet Jacks
Come on Get Up - Janet Jackson
Control - Janet Jackson

Most Listens:
Cohkka - Apocalyptica
As the World Turns - Eminem
Fast Girls - Janet Jackson
Streets Is Watching - Jay-Z
Hymn - Moby
We Are All Made of Stars - Moby
You - Moby
That Wonderful Someone - Patsy Cline
Hold Me Down - 50 Cent

Edit Button	SongName	SongID	Rating	AlbumName	ReleaseYear	GenreName	MusicianName
Edit	So Disrespectful	4	4	Before I Self Destruct	2009	Rap	50 Cent
Edit	So Disrespectful	4	3	Before I Self Destruct	2009	Rap	50 Cent
Edit	Psycho	5	2	Before I Self Destruct	2009	Rap	50 Cent
Edit	Hold Me Down	6	4	Before I Self Destruct	2009	Rap	50 Cent

Add to playlist:
All ATEEZ
My favorite songs

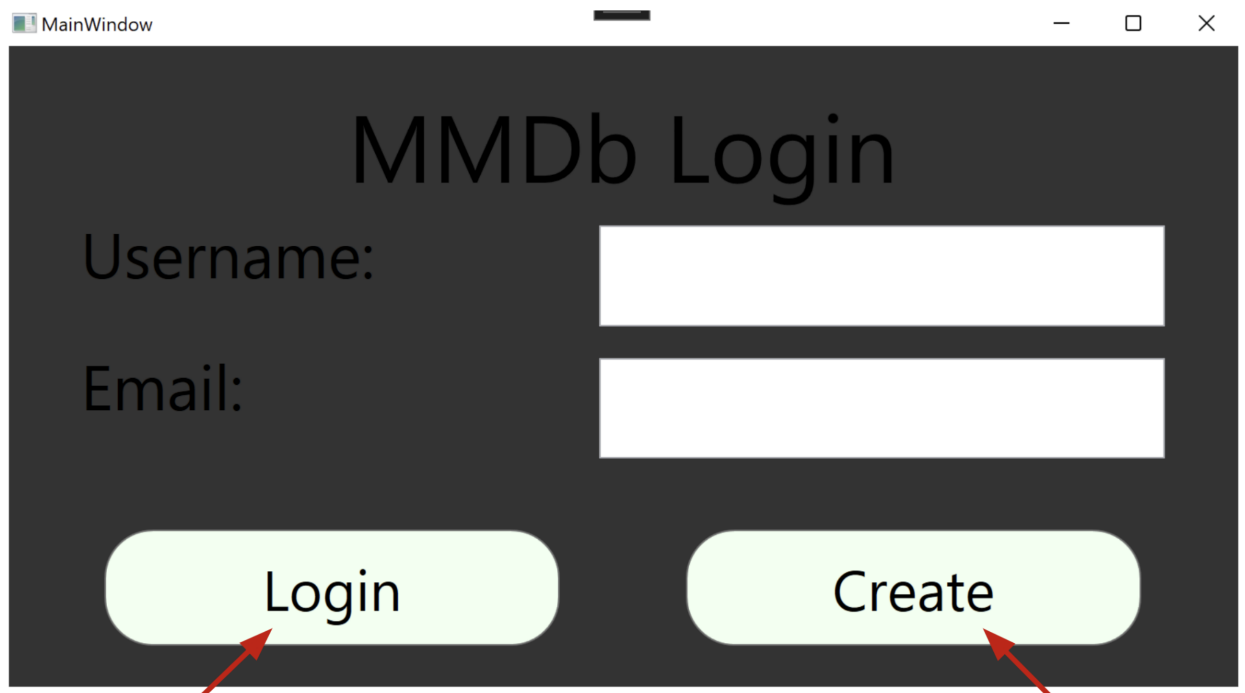
Your Top Genres:
K-Pop - 50.000000
Pop/Rock - 21.880000
Rap - 12.500000
Electronic - 9.380000
R&B - 3.130000
Pop - 3.130000

Other users would rate your playlists an average of 3.6

Edit Button	SongName	AlbumName	MusicianName	SongID
Edit	Dream On	Aerosmith	Aerosmith	62
Edit	PSYCHO	PSYCHO	Anne-Marie	82
Edit	HALA HALA (Traditional Treatment Mix)	TREASURE EP. EXTRA: Shift The Map	ATEEZ	111
Edit	Pirate King (Overload Mix)	TREASURE EP. EXTRA: Shift The Map	ATEEZ	112

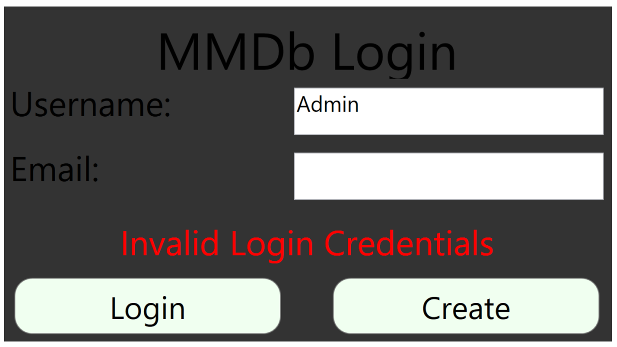
SYSTEM FEATURES AND USAGE

Login or Add User:



Users and Administrators can login if they have valid credentials

If the Username and Email entered are both unique, the user can create a new account



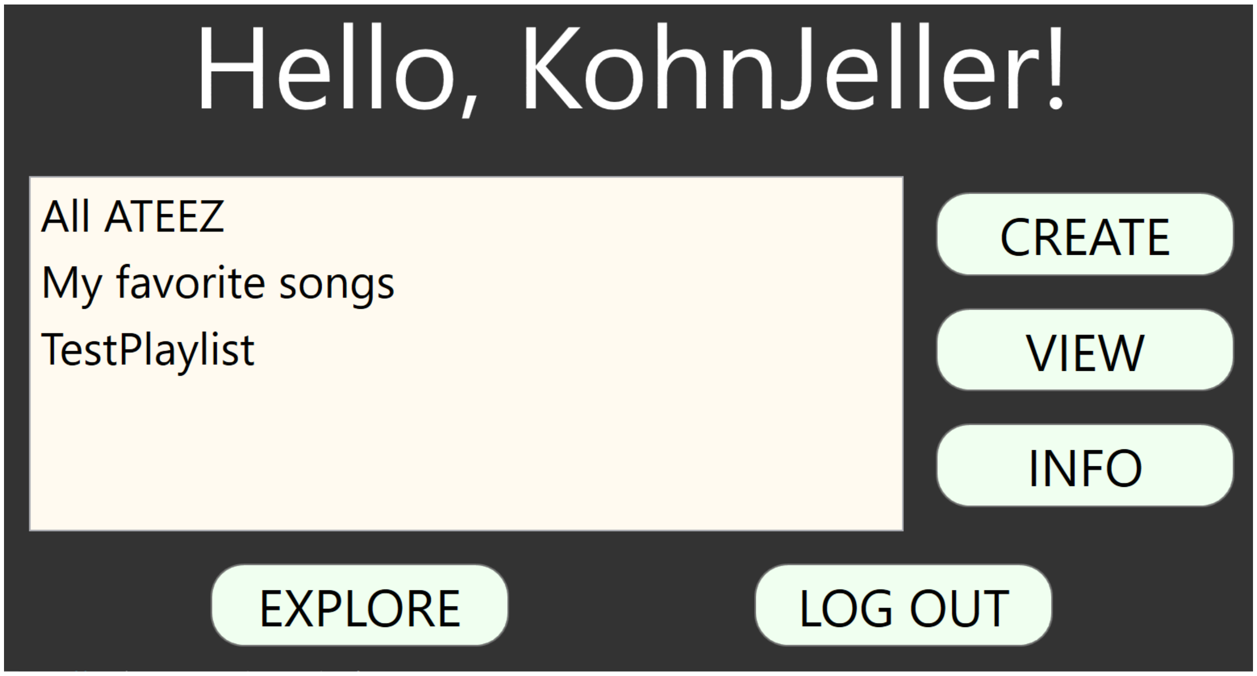
Username: Admin

Invalid Login Credentials

An error message will appear if the Username and Email combination do not exist in the system or if you try to create an already existing account.

A fault in this format of login of course comes into the fact that there is no email validating in terms of determining if the user is using a fake email or not, nor is there an ability for the user to reset their password. As such, they have unlimited attempts, which would be unsafe if these were real user accounts.

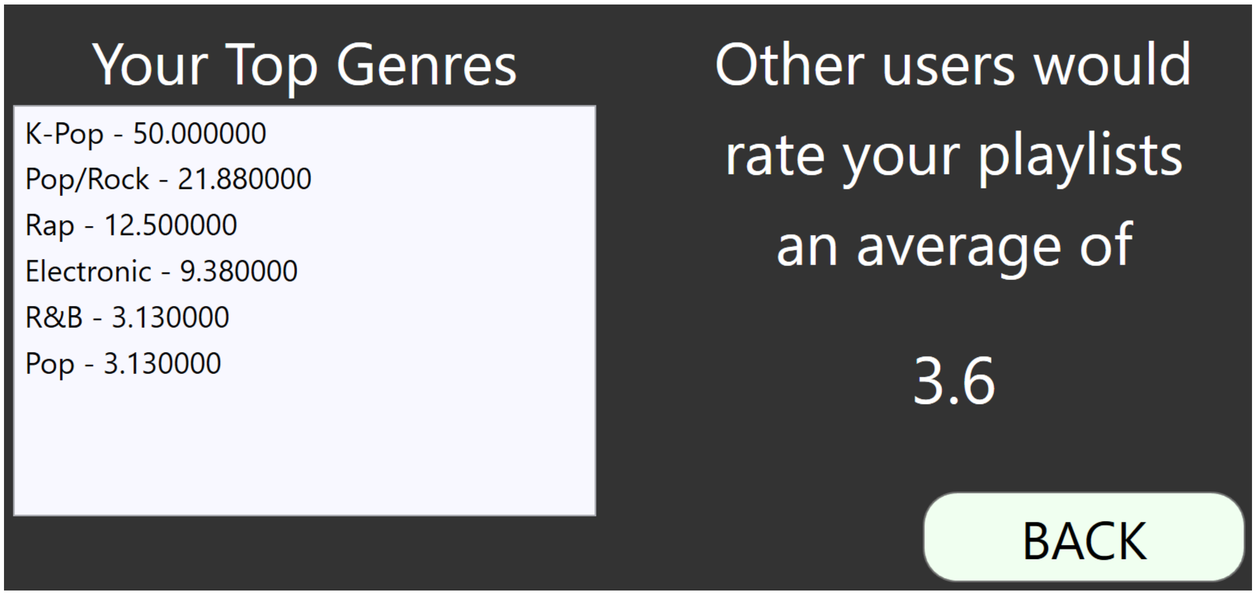
Access Your Playlists in Main Dash:



All ATEEZ
My favorite songs
TestPlaylist

This is the main user view that is used to access all of the apps features

Get User Information Regarding Playlists:


Your Top Genres

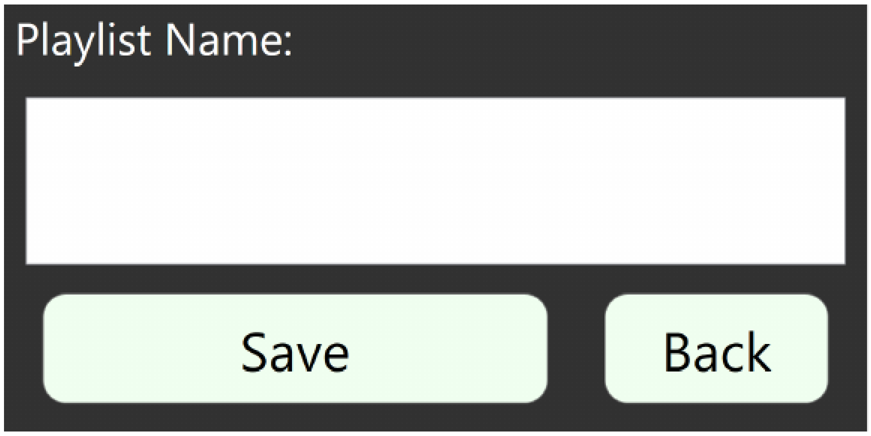
K-Pop - 50.000000
Pop/Rock - 21.880000
Rap - 12.500000
Electronic - 9.380000
R&B - 3.130000
Pop - 3.130000

Other users would rate your playlists an average of

3.6

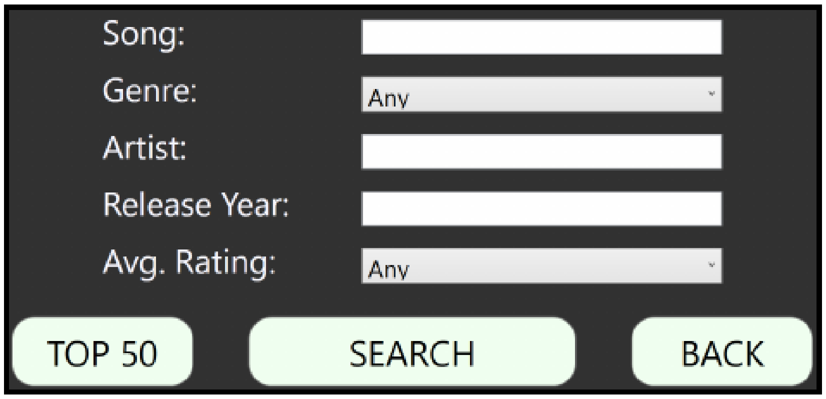
This feature gives the genres included in each playlist as well as what proportion of the entire song collection they make up. An average rating of all of the user's playlists is also displayed, calculated from the individual songs within the playlists.

Create a Playlist:


Playlist Name:

A current flaw we have in this playlist creation is an inability to delete a playlist or remove a song from it once it has been added. We focused more on the ease of accessing and adding song selections, which unfortunately led to missing this important feature.

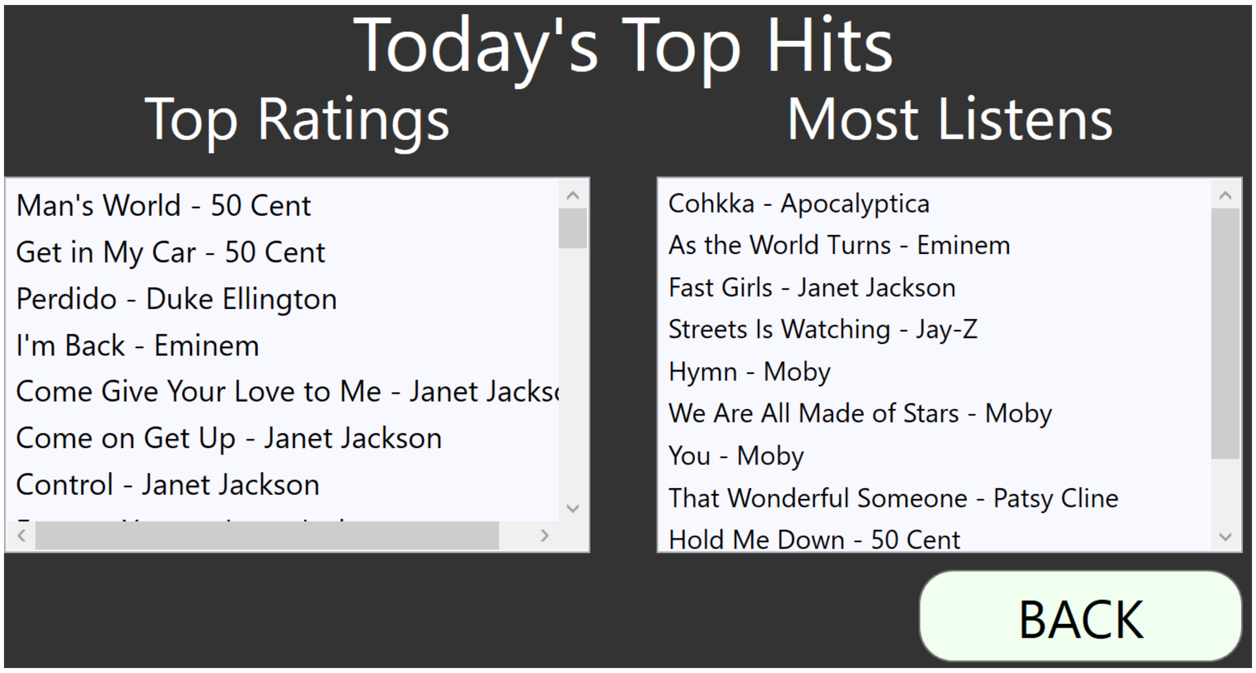
Search for Music:



Song:	
Genre:	Any
Artist:	
Release Year:	
Avg. Rating:	Any

Users are able to search for songs based on a number of identifying features. They are also able to get song suggestions based on the most popular songs rated amongst other users.

See Top Rated Songs:

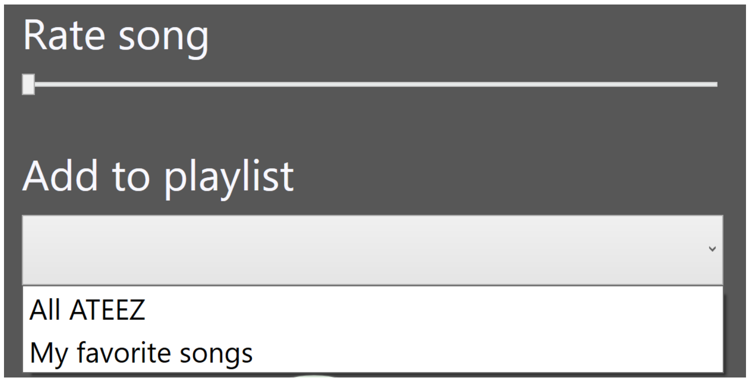


Today's Top Hits

Top Ratings	Most Listens
Man's World - 50 Cent	Cohkka - Apocalyptica
Get in My Car - 50 Cent	As the World Turns - Eminem
Perdido - Duke Ellington	Fast Girls - Janet Jackson
I'm Back - Eminem	Streets Is Watching - Jay-Z
Come Give Your Love to Me - Janet Jacks	Hymn - Moby
Come on Get Up - Janet Jackson	We Are All Made of Stars - Moby
Control - Janet Jackson	You - Moby
	That Wonderful Someone - Patsy Cline
	Hold Me Down - 50 Cent

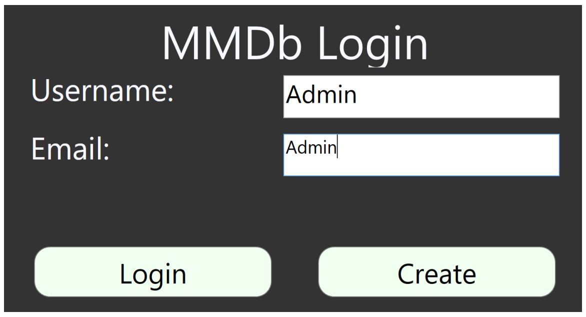
For easier viewing, this feature does not display the values that got each of these songs their place in the list. It looked too muddled when added so we relied solely on the fact that they are in ranked descending order.

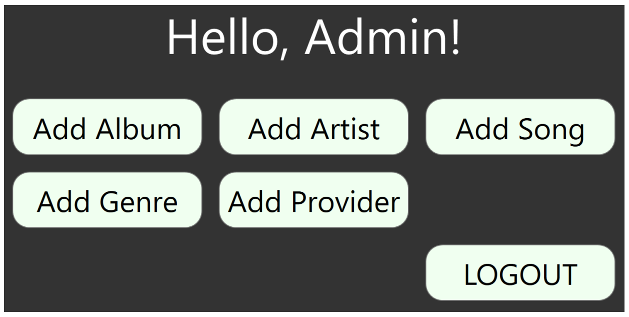
Rate Songs and Add to Existing Playlists:



If a song goes unrated, the average rating of that song will not be negatively affected, meaning the lack of user rating does not read as a zero (or one in this case of our 1-5 star rating scale). While this is a good attribute for this feature, the coupled “Add to playlist” does have a notable fault. Unlike other applications, like YouTube Music, you cannot create a playlist from the playlist selection panel. Playlist can only be made from the user’s main view.

Beyond the user’s accessible features, we do also have an admin view implemented. This allows us to add to our non-user-writable tables in our database directly from the app!





AGGREGATING QUERIES

1. Description: Show most “playlisted” songs (songs that appear the most amongst user playlists)

   Parameters:

   N/A

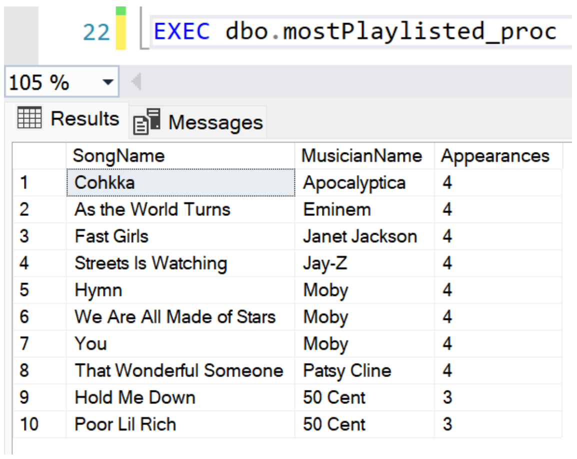
   Result Columns:

   Song (Songs.SongName)

   Musician (Musicians.MusicianName)

```
CREATE OR ALTER PROCEDURE dbo.mostPlaylisted_proc
AS
BEGIN
SELECT TOP 10
      S.SongName
      ,M.MusicianName
      ,COUNT(*) AS Appearances
      FROM dbo.PlaylistSongs PS
           INNER JOIN dbo.Songs S ON S.SongID = PS.SongID
           INNER JOIN dbo.Albums A ON A.AlbumID =S.AlbumID
           INNER JOIN dbo.Musicians M ON M.MusicianID = A.MusicianID
GROUP BY S.SongName, M.MusicianName
ORDER BY Appearances DESC

END;
```



```
EXEC dbo.mostPlaylisted_proc
```

105 %

Results | Messages

	SongName	MusicianName	Appearances
1	Cohkka	Apocalyptica	4
2	As the World Turns	Eminem	4
3	Fast Girls	Janet Jackson	4
4	Streets Is Watching	Jay-Z	4
5	Hymn	Moby	4
6	We Are All Made of Stars	Moby	4
7	You	Moby	4
8	That Wonderful Someone	Patsy Cline	4
9	Hold Me Down	50 Cent	3
10	Poor Lil Rich	50 Cent	3

2. Description: Report the “Top Charts” based on average user ratings

   Parameters:

      N/A

   Result Columns:

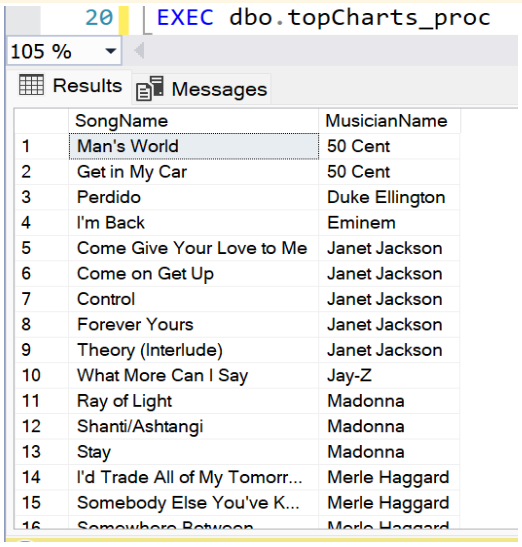
      Musician (Musicians.MusicianName)

      Song (Songs.SongName)

      AvgRating: the count of how often the song shows up in user playlists

```
CREATE OR ALTER PROCEDURE dbo.topCharts_proc
AS
BEGIN
SELECT TOP 50
      S.SongName
      ,M.MusicianName
      ,SUM (R.Rating) / COUNT(*) AS AvgRating
      FROM dbo.UserRatings R
            INNER JOIN dbo.Songs S ON S.SongID = R.SongID
            INNER JOIN dbo.Albums A ON A.AlbumID =S.AlbumID
            INNER JOIN dbo.Musicians M ON M.MusicianID = A.MusicianID
GROUP BY S.SongName, M.MusicianName
ORDER BY AvgRating DESC
```

```
END;
```



```
EXEC dbo.topCharts_proc
```

105 %

Results Messages

	SongName	MusicianName
1	Man's World	50 Cent
2	Get in My Car	50 Cent
3	Perdido	Duke Ellington
4	I'm Back	Eminem
5	Come Give Your Love to Me	Janet Jackson
6	Come on Get Up	Janet Jackson
7	Control	Janet Jackson
8	Forever Yours	Janet Jackson
9	Theory (Interlude)	Janet Jackson
10	What More Can I Say	Jay-Z
11	Ray of Light	Madonna
12	Shanti/Ashtangi	Madonna
13	Stay	Madonna
14	I'd Trade All of My Tomorr...	Merle Haggard
15	Somebody Else You've K...	Merle Haggard
16	Somewhere Between	Merle Haggard

3. Show user what their playlists' average rating would be based on the individual playlist songs' ratings by other users

   Parameters:

   UserID: the logged in user

   Result Columns:

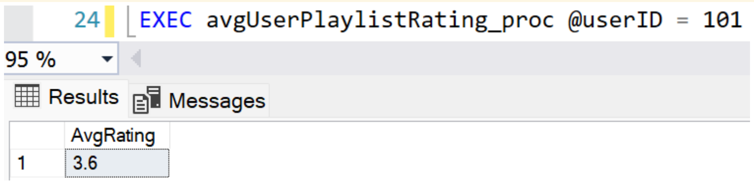
   AvgRating: This will be the overall average of the individual songs' average ratings (ie. the user (in terms of their playlists) will have a 5-point rating just like the individual songs)

```
CREATE OR ALTER PROCEDURE dbo.avgUserPlaylistRating_proc(
@userID BIGINT
)
AS
BEGIN

SELECT
      --SUM (CAST(R.Rating AS DECIMAL(3,2)))
      CAST(SUM (CAST(R.Rating AS DECIMAL(3,2))) / COUNT(*) AS
DECIMAL(2,1)) AS AvgRating
```

```
      FROM dbo.Users U
      INNER JOIN dbo.Playlists P ON P.UserID = U.UserID
      INNER JOIN dbo.PlaylistSongs PS ON PS.PlaylistID = P.PlaylistID
      LEFT JOIN dbo.Songs S ON S.SongID = PS.SongID
      JOIN dbo.UserRatings R ON R.SongID = S.SongID
WHERE U.UserID = @userID
GROUP BY U.UserID

END;
```



```
EXEC avgUserPlaylistRating_proc @userID = 101
```

	AvgRating
1	3.6

4. Show user their genre rankings based on the songs they include in their playlists (ie. “your top 3 most listened to genres are…”)

   Parameters:

      UserID: The user whose genre preferences are to be displayed.

   Result Columns:

      <u>GenreName</u>: the ranked genres of a user based on the number of appearances of the genres’ songs within the users’ playlists.

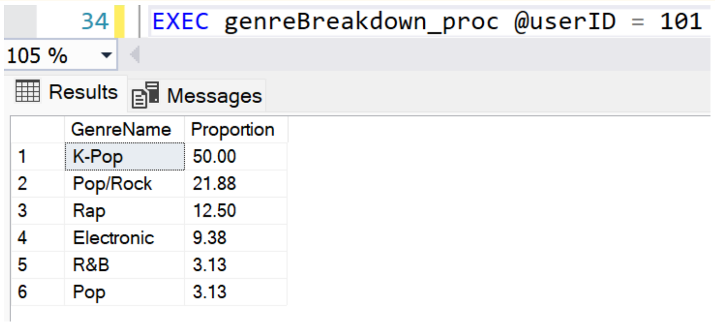
      <u>Proportion</u>: the frequency of the specified genre appearing throughout the user’s playlists.

```
CREATE OR ALTER PROCEDURE dbo.genreBreakdown_proc(
@userID BIGINT
)
AS
BEGIN

DECLARE @songCount DECIMAL(4,2) = (SELECT COUNT(*)
           FROM dbo.PlaylistSongs PS
           INNER JOIN dbo.Playlists P ON P.PlaylistID = PS.PlaylistID
           INNER JOIN dbo.Users U ON U.UserID = P.UserID
           WHERE U.UserID = @userID)
print(@songCount)

SELECT
      G.GenreName
      --,COUNT(*) AS songCount
      ,ROUND((COUNT(*) / @songCount * 100), 2) AS Proportion
      FROM dbo.Users U
      INNER JOIN dbo.Playlists P ON P.UserID = U.UserID
      INNER JOIN dbo.PlaylistSongs PS ON PS.PlaylistID = P.PlaylistID
      INNER JOIN dbo.Songs S ON S.SongID = PS.SongID
      INNER JOIN dbo.Albums A ON A.AlbumID = S.AlbumID
      INNER JOIN dbo.Genre G ON G.GenreID = A.GenreID
WHERE U.UserID = @userID
GROUP BY G.GenreName
ORDER BY Proportion DESC

END;
```



```
EXEC genreBreakdown_proc @userID = 101
```

105 %

Results | Messages

	GenreName	Proportion
1	K-Pop	50.00
2	Pop/Rock	21.88
3	Rap	12.50
4	Electronic	9.38
5	R&B	3.13
6	Pop	3.13

SUMMARY AND DISCUSSION

Completing this project proved how important good design principles are for the inception and overall completion of a project. Thankfully, we put a decent amount of thought into our overall design and how we could eventually draw connections. In the end, our final project is certainly comparable to our initial design and only relatively minor modifications to our initially proposed design were made. During the design phase, we wanted users to also have access to an artist's date of birth, origin, record label, and label address. After our initial project proposal, we decided not to implement these features due to some of the feedback we received and we also found throughout the database creation process that this in itself would have proved quite tedious and challenging.

The basic music library aspects of the program were fairly simple. Mimicking reasonable user data did prove to be a challenge. The main issue we ran into was populating tables in the sense of making the data reflect realistic playlists and user preferences. As far as music goes, everyone has a fairly distinct taste and preference in what they listen to. Creating and populating playlists in a consistent manner that also makes sense from a music preference standpoint, was not feasible for this assignment for the scale desired. In order to have enough data to compare and pull from, we needed a large number of playlists by a large number of different users as well as ratings for individual songs and so on. In the end, we relied on generators to create this information so that we could focus more on functionality.

If we were to conduct such a project again, we would certainly aim to delineate who's doing what aspects of the project much sooner. We ran into multiple instances where a couple of us were working on creating and populating our database tables at the same time without the other knowing. While this didn't ultimately end in errors constructing our system, it did cause us to waste a considerable amount of time, which later led to a notable lack of that resource.

Also, as we were warned at the inception of this project, there was in fact a learning curve with the use of some of our tools/technologies. While we initially felt fairly confident in our familiarity with the systems we ultimately utilized, we were not fully prepared to struggle with the coupling of these various technologies—alone, using XAML and SQL is fine, but accessing our data and displaying it through our UI took us more time than we anticipated.

The main development we would like to add to this application if we were to continue working on it is more inclusive data (i.e. more artists, songs, albums, etc.). While our application does have its primary desired effects, the scope is notably narrow for how vast the music industry is. Furthermore, if we were making this a real user application, we would also want to improve the visual aesthetic of our interface. Overall, our project does what we wanted, but it is certainly rough around the edges and not quite ready for prod.

In conclusion, we learned a great deal about the dynamics of working as a team, how to prioritize and delegate work, and set goals with a reasonable frame of time.