SPOTIFY SONG DATABASE

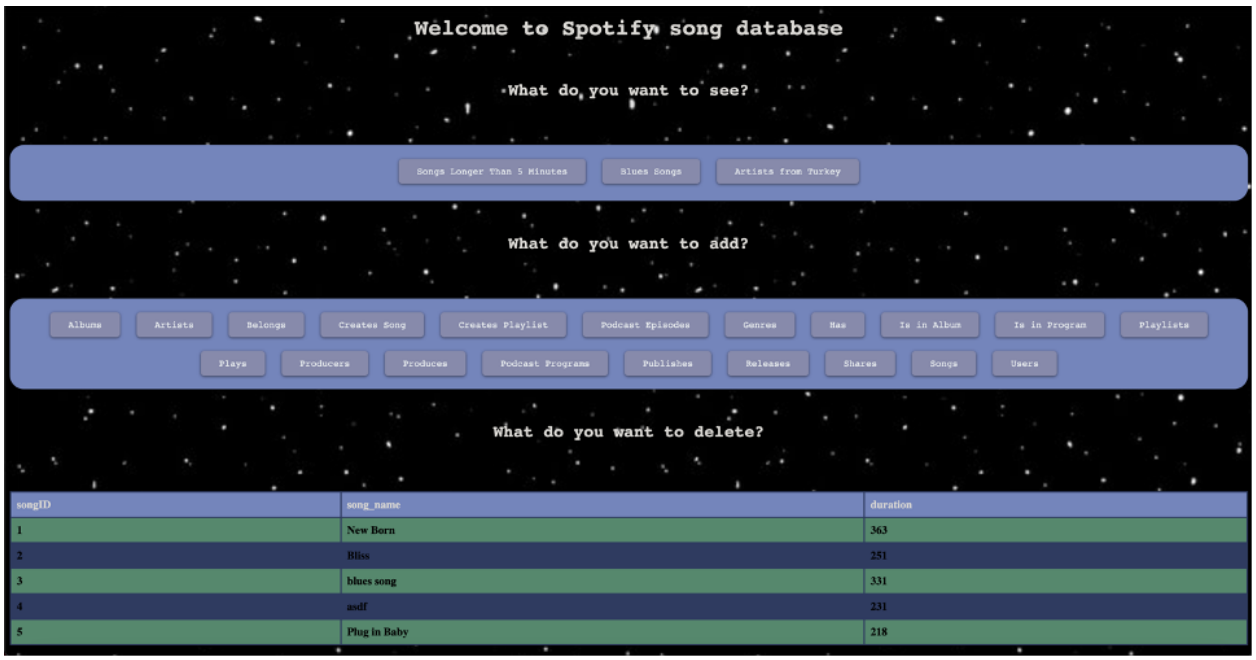
Deniz Çetin 28896
İbrahim Ege Oral 29299
Sueda Şeker 28325

Project Goal:

The original goal of our project was to create a system that would keep track of a user's music listening habits in order to recommend songs based on their playlists. However, we decided to pivot and focus on building a database that could store detailed information about various songs, albums, and artists. This database would allow users to search for and select songs for their playlists using various filters.

The project now consists of an interface that allows the user to manage and query this database of music information through a main page with buttons for adding and viewing records. This interface is intended to be used with the music streaming service Spotify.

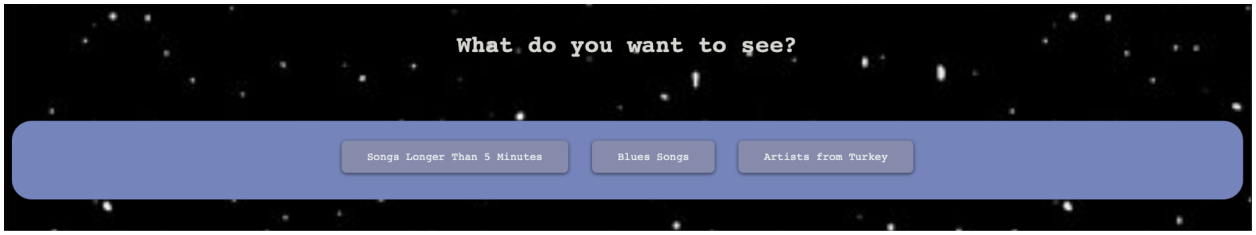
Main page:



In addition to the existing features, we added a button that allows the user to delete a song from the database using the song's ID. This gives the user the ability to remove any desired song from the database.

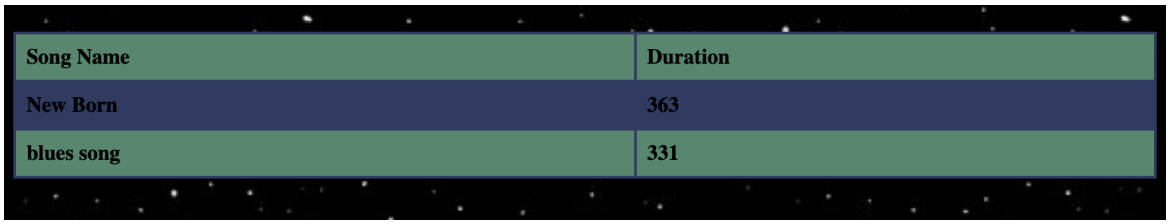
SELECTION

When the user clicks on one of the buttons in the "What do you want to see?" section, they are taken to a new page that displays the results of the relevant query in a table format. This section contains three buttons: "Songs Longer Than 5 Minutes", "Blues Songs", and "Artists from Turkey". These buttons allow the user to access the results of specific queries on the database by clicking on them. Each button corresponds to a specific query that is run on the database when it is clicked.



- The "Songs Longer Than 5 Minutes" button runs the following SELECT query:

```
SELECT * FROM songs WHERE duration > 300
```

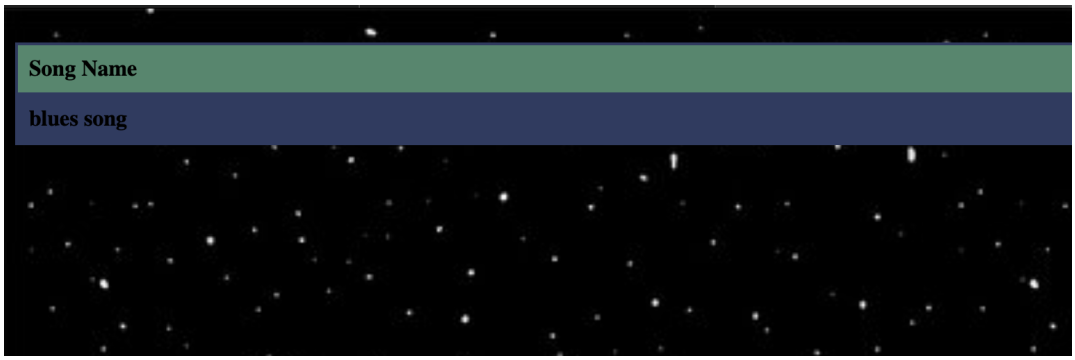


Song Name	Duration
New Born	363
blues song	331

This query selects all records from the "songs" table where the value of the "duration" attribute is greater than 300 seconds (5 minutes). The query returns the records that match this condition, and they are displayed in the table on the new page.

- The "Blues Songs" button runs the following SELECT query:

```
SELECT * FROM songs S, belongs B, genres G
WHERE G.genre_name = 'blues' and G.genreID = B.genreID and B.songID = S.songID
```



This query selects all records from the "songs" table that belong to the "blues" genre. It does this by first joining the "songs" table with the "belongs" and "genres" tables on the "songID" and "genreID" attributes, respectively. Then, it filters the resulting records by selecting only those where the "genre_name" attribute in the "genres" table is equal to "blues". The query returns the records that match this condition, and they are displayed in the table on the new page.

- The "Artists from Turkey" button runs the following SELECT query:

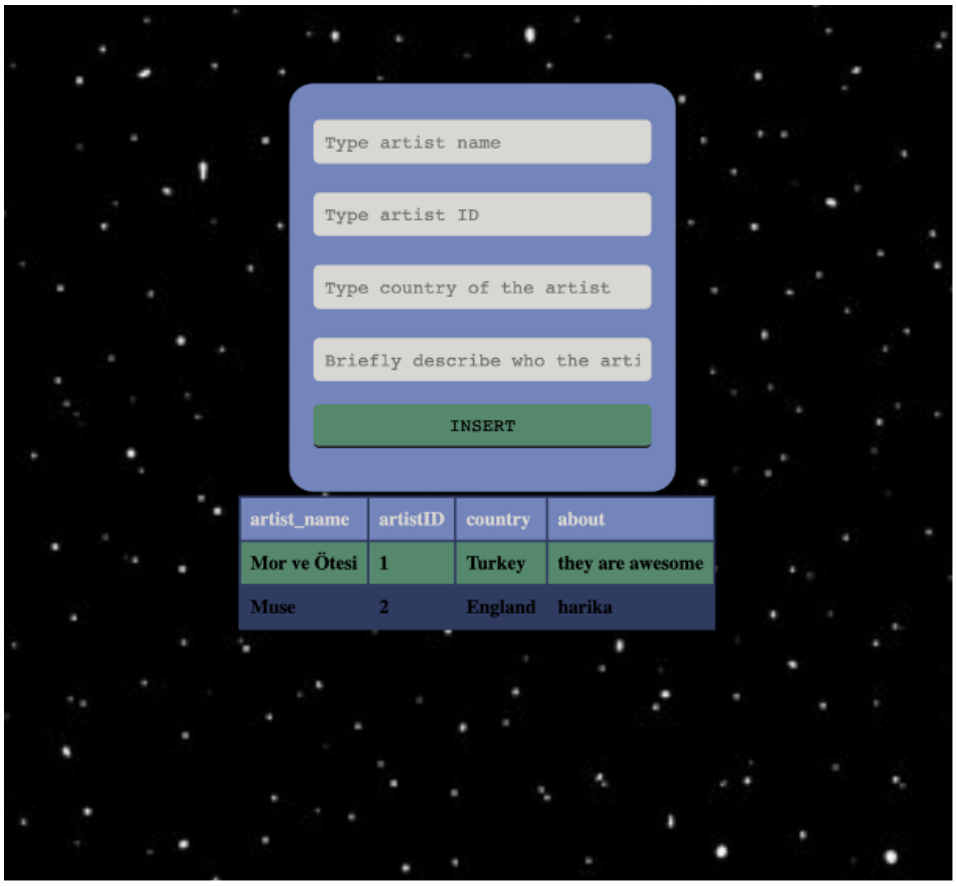
```
SELECT * FROM Artists WHERE country = 'Turkey'
```

This query selects all records from the "Artists" table where the value of the "country" attribute is equal to "Turkey". The query returns the records that match this condition, and they are displayed in the table on the new page.

INSERTION

The main page also has lots of buttons in the "What do you want to add?" section. When one of these buttons is clicked, the user is directed to a new page where they can enter data for the corresponding table in the database.

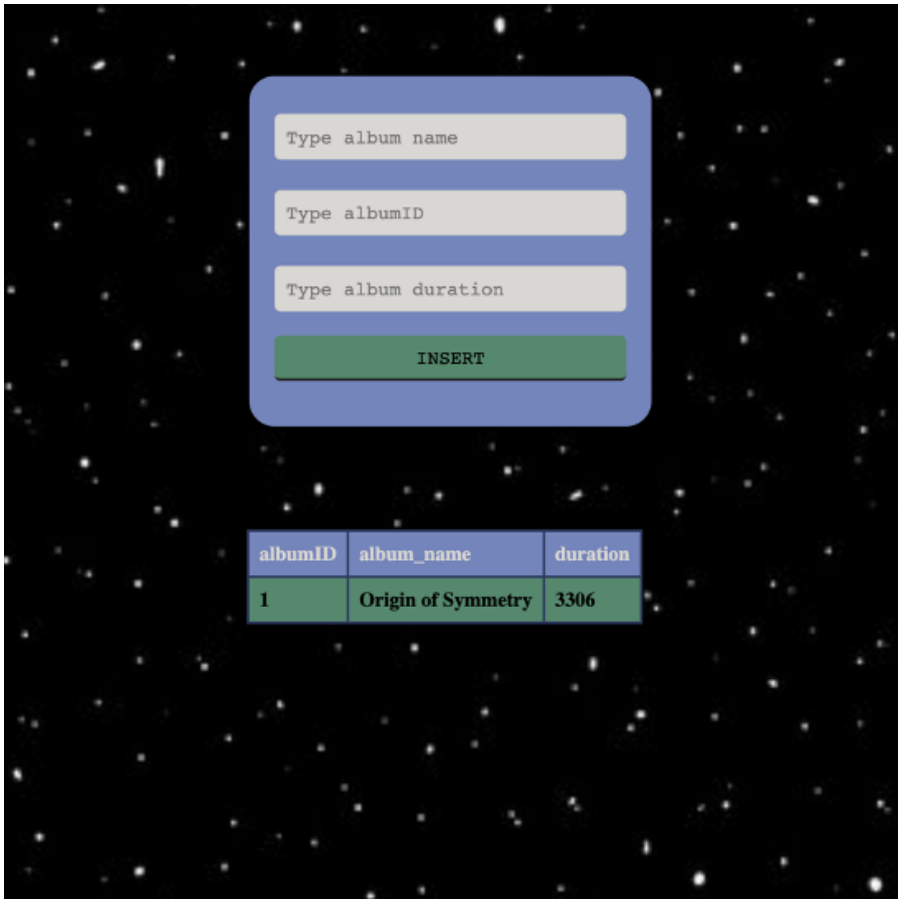
Artist insertion page:



The "Artists" button leads to a page with a form that allows the user to enter data for a new artist. The form includes fields for the artist's name, ID, country, and a brief description. When the user submits the form, the following INSERT query is run on the database:

```
INSERT INTO artists (artist_name, artistID, country, about)
VALUES ('$artist_name', '$artistID', '$country', '$about)
```

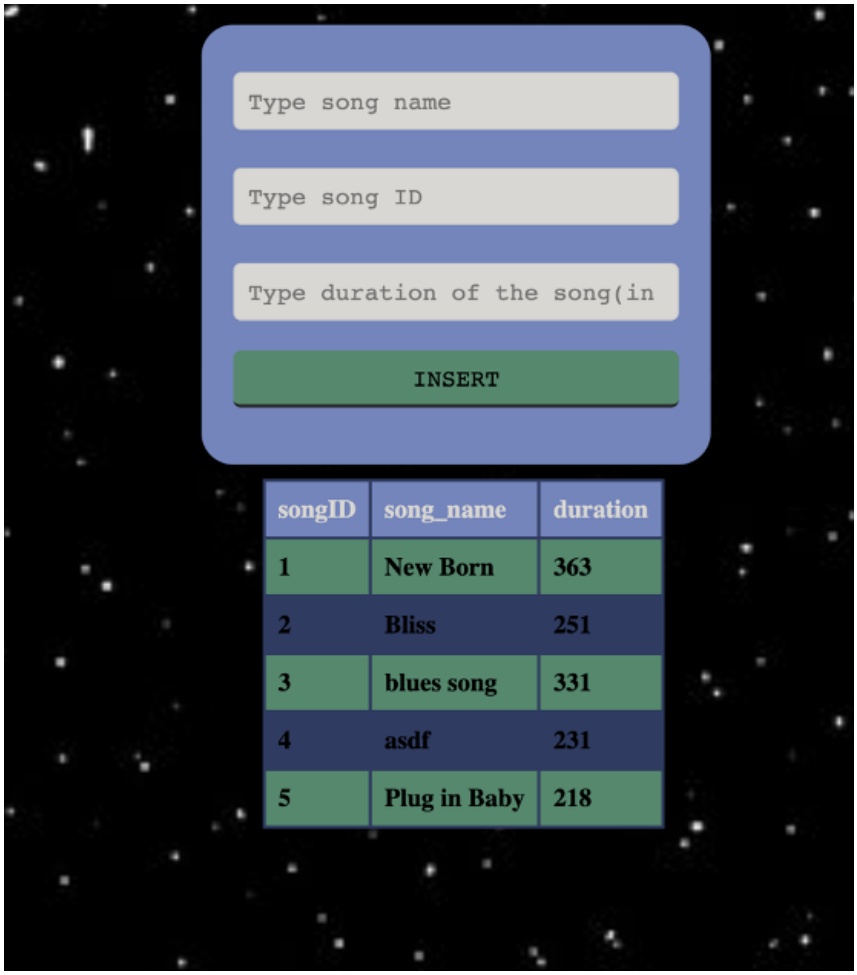
Album insertion page (Albums button):



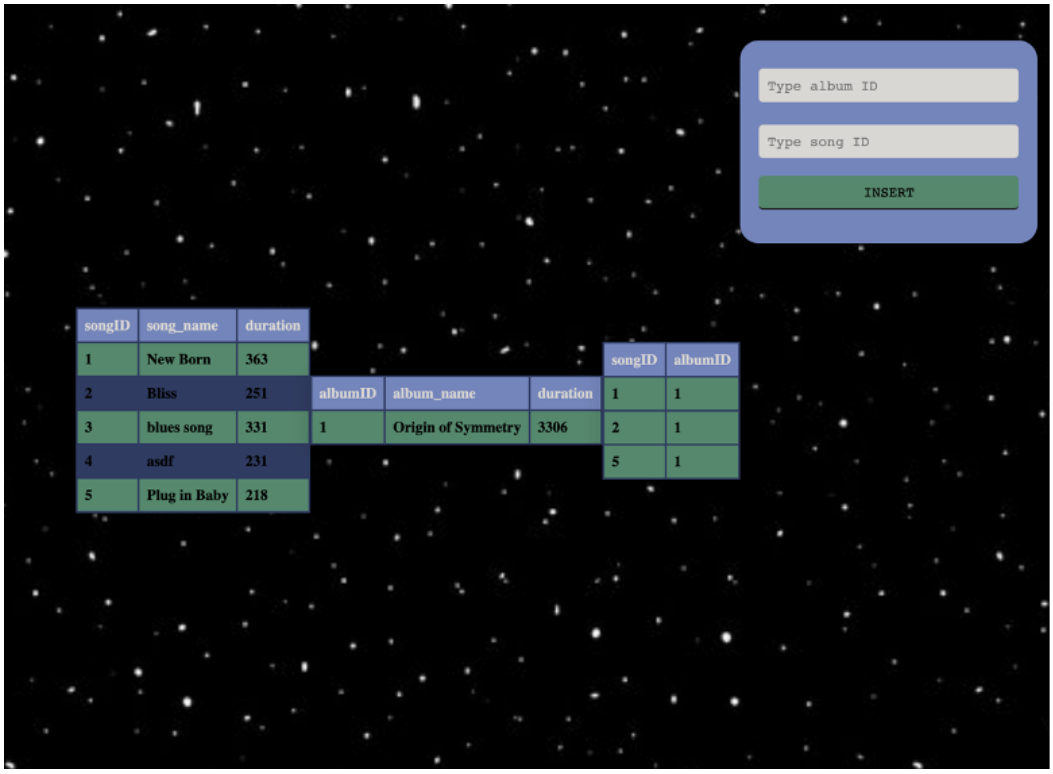
The "Albums" button leads to a page with a form that allows the user to enter data for a new album. The form includes fields for the album's name, ID, and duration. When the user submits the form, the following INSERT query is run on the database:

```
INSERT INTO albums (album_name, albumID, duration)
VALUES ('$album_name', '$albumID', '$duration')
```

Song insertion page (Songs button):

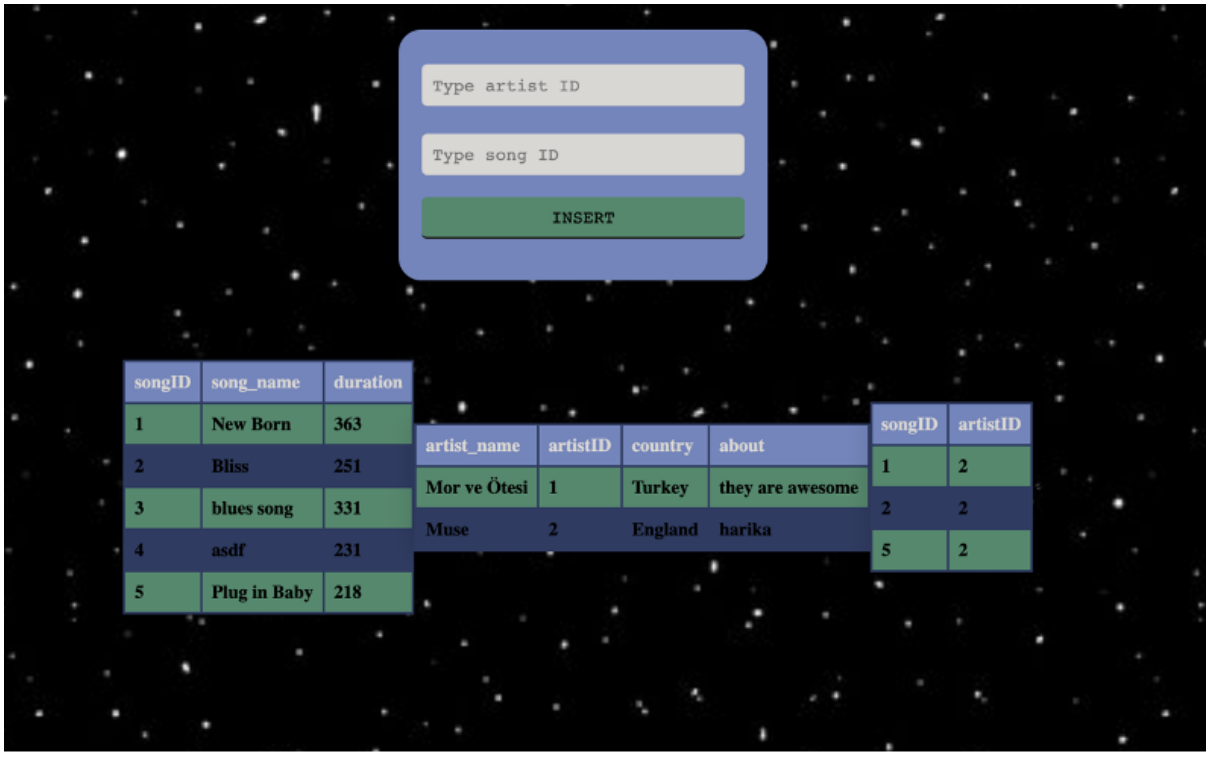


Album-Song relation insertion page (Is in Album button):



We can see which song is in which album based on the 3rd table(songId | albumID), same goes for other insertion pages, hence, we can determine which is which.

Song-Artist relation insertion page (Creates Song button):

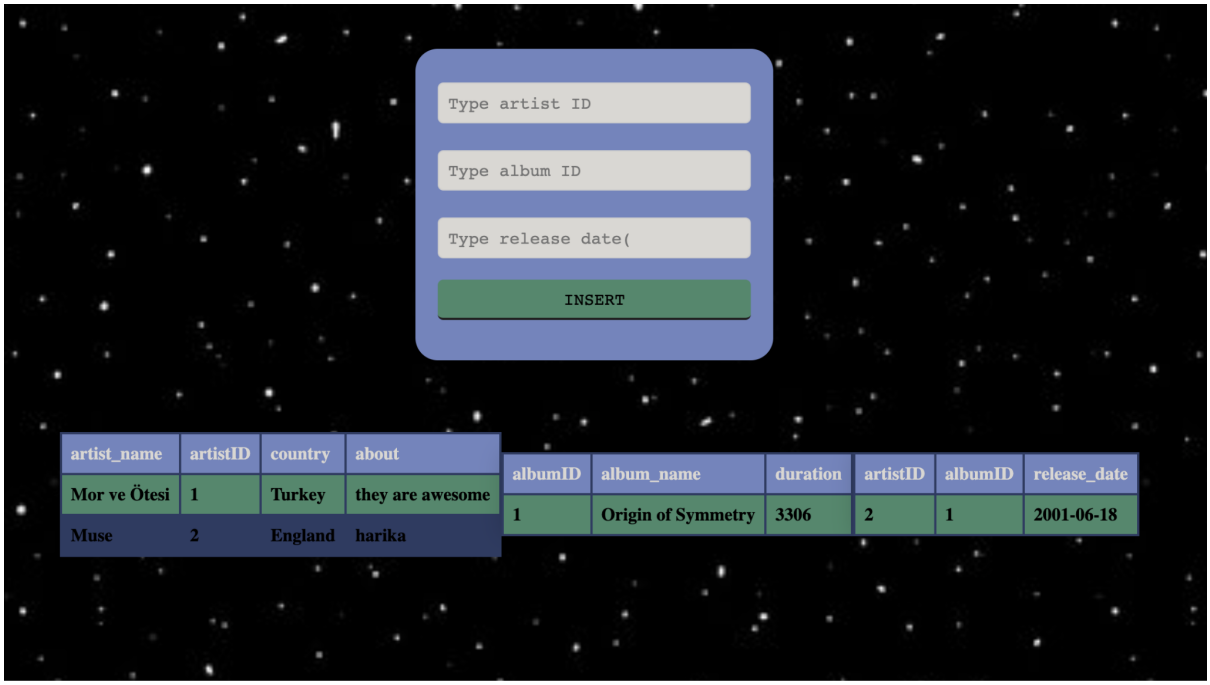


songID	song_name	duration
1	New Born	363
2	Bliss	251
3	blues song	331
4	asdf	231
5	Plug in Baby	218

artist_name	artistID	country	about
Mor ve Ötesi	1	Turkey	they are awesome
Muse	2	England	harika

songID	artistID
1	2
2	2
5	2

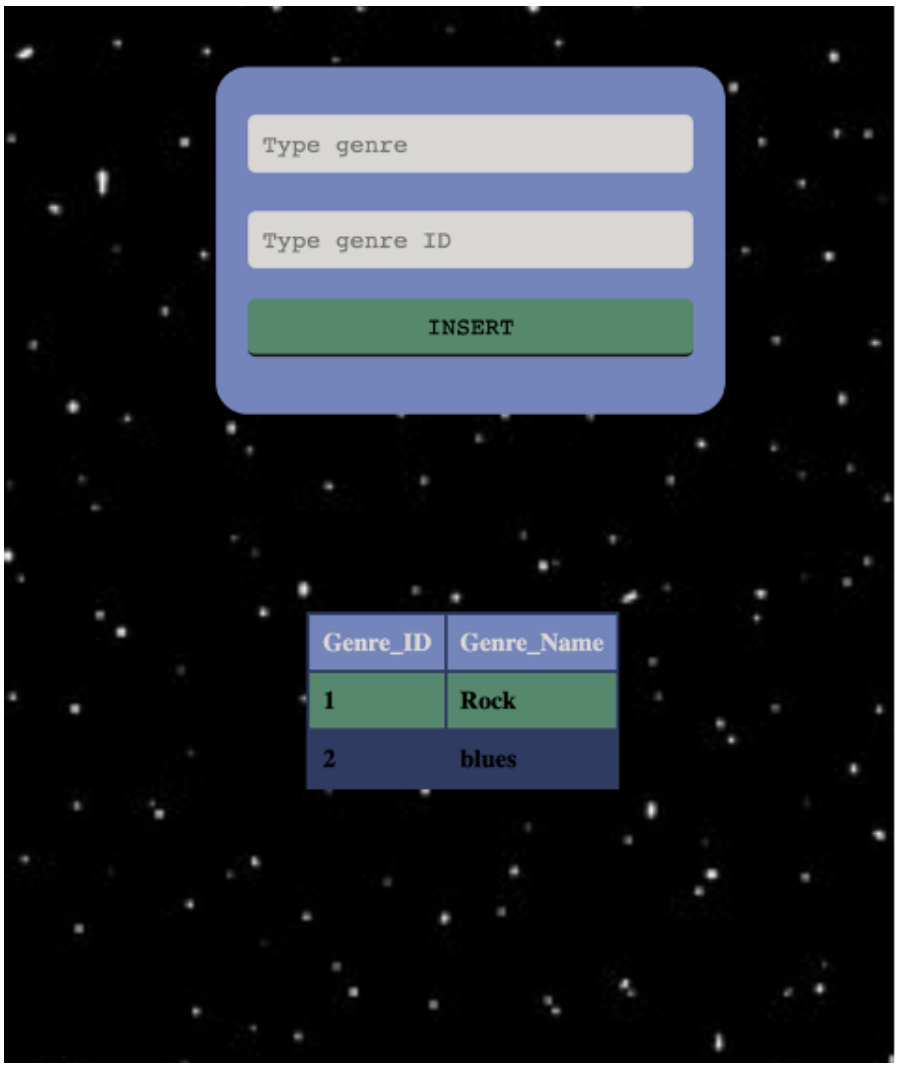
Album-Artist relation insertion page (Releases button):



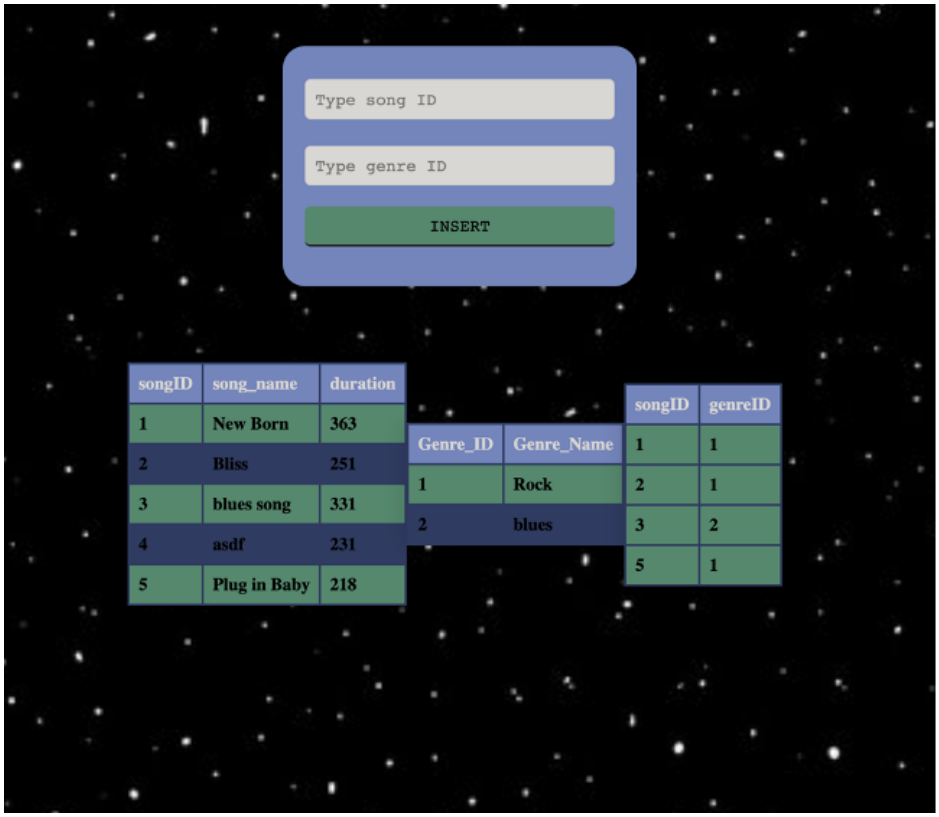
artist_name	artistID	country	about
Mor ve Ötesi	1	Turkey	they are awesome
Muse	2	England	harika

albumID	album_name	duration	artistID	albumID	release_date
1	Origin of Symmetry	3306	2	1	2001-06-18

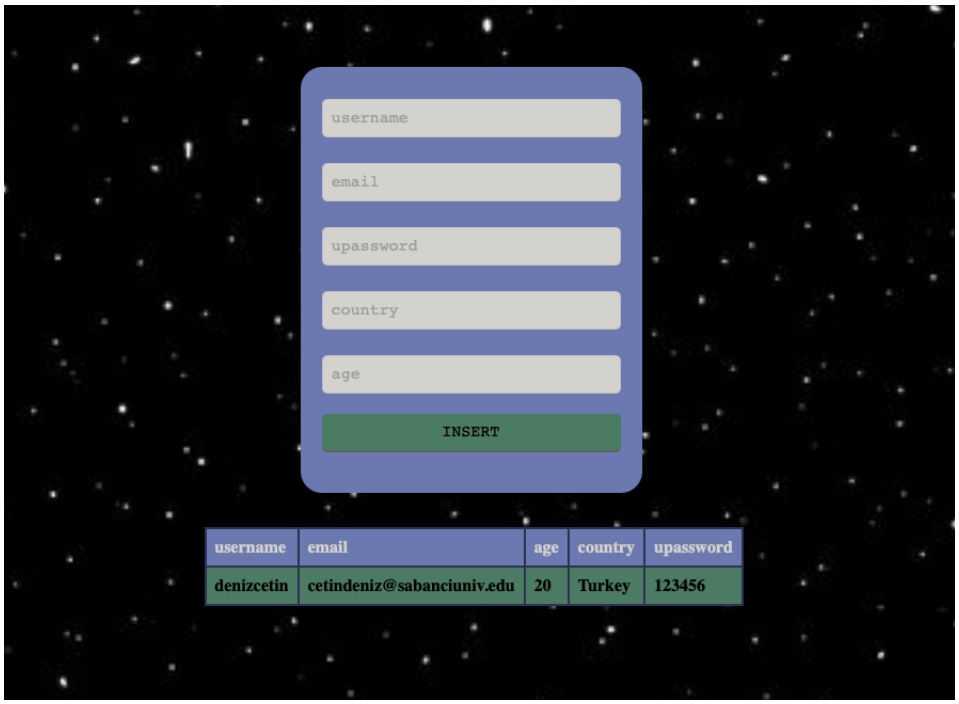
Genre insertion page (Genres button):



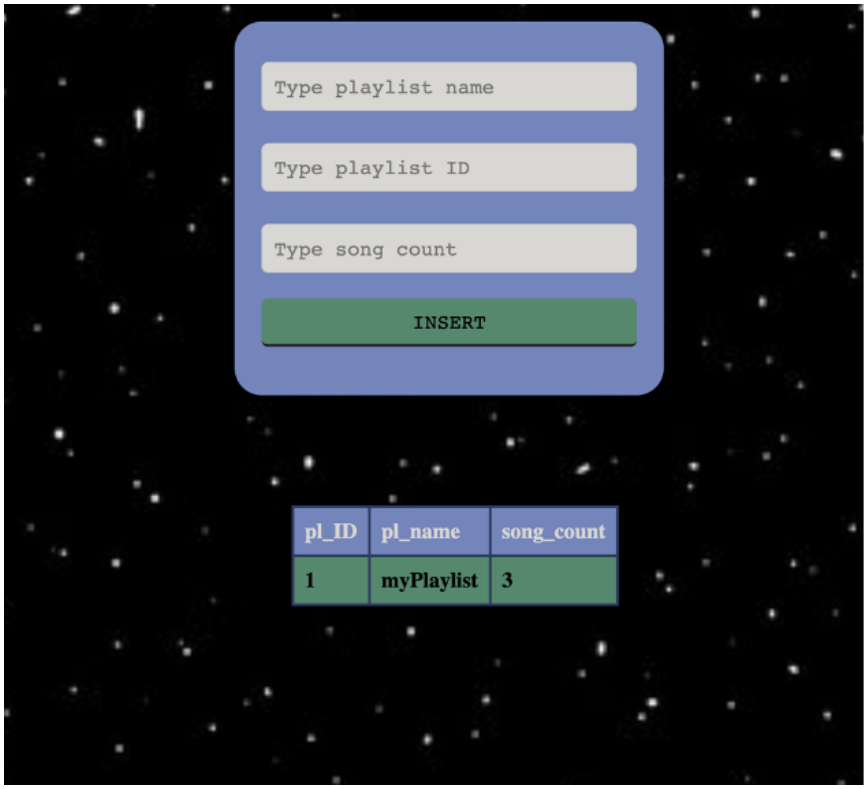
Song-Genre relation insertion page (Belongs button):



User insertion page (Users button):

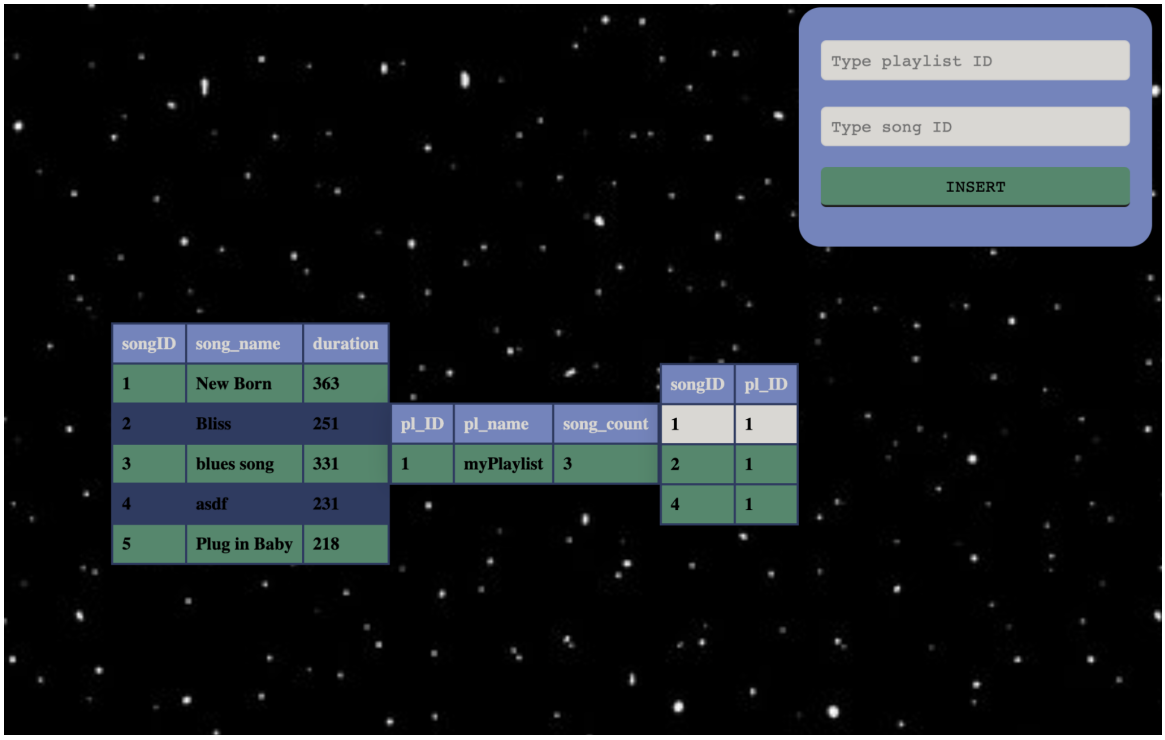


Playlist insertion page (Playlists button):



User Creates a Playlist relation insertion page (Creates Playlist button):

Song-Playlist relation insertion page (Has button):



songID	song_name	duration
1	New Born	363
2	Bliss	251
3	blues song	331
4	asdf	231
5	Plug in Baby	218

pl_ID	pl_name	song_count
1	myPlaylist	3

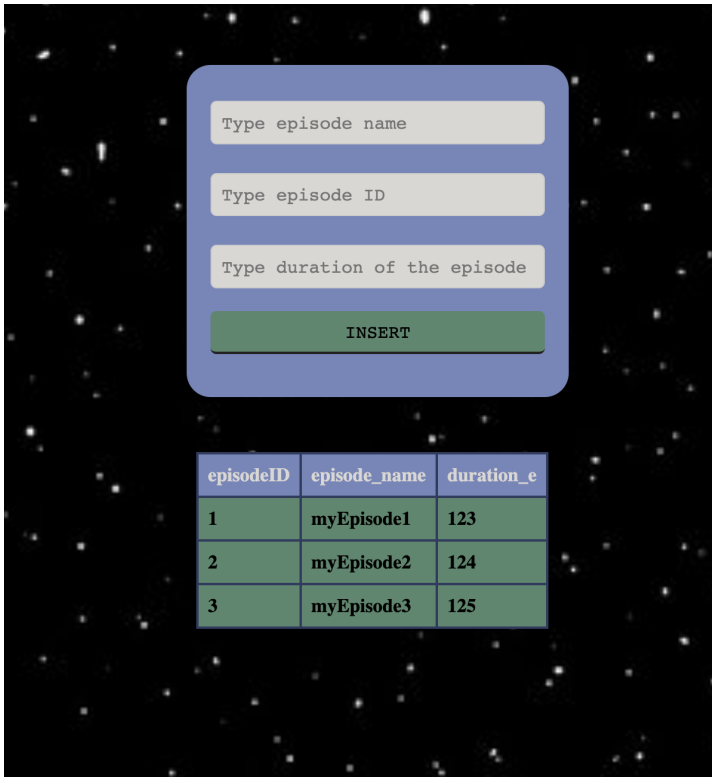
songID	pl_ID
1	1
2	1
4	1

User Plays Playlist relation insertion page (Plays button):

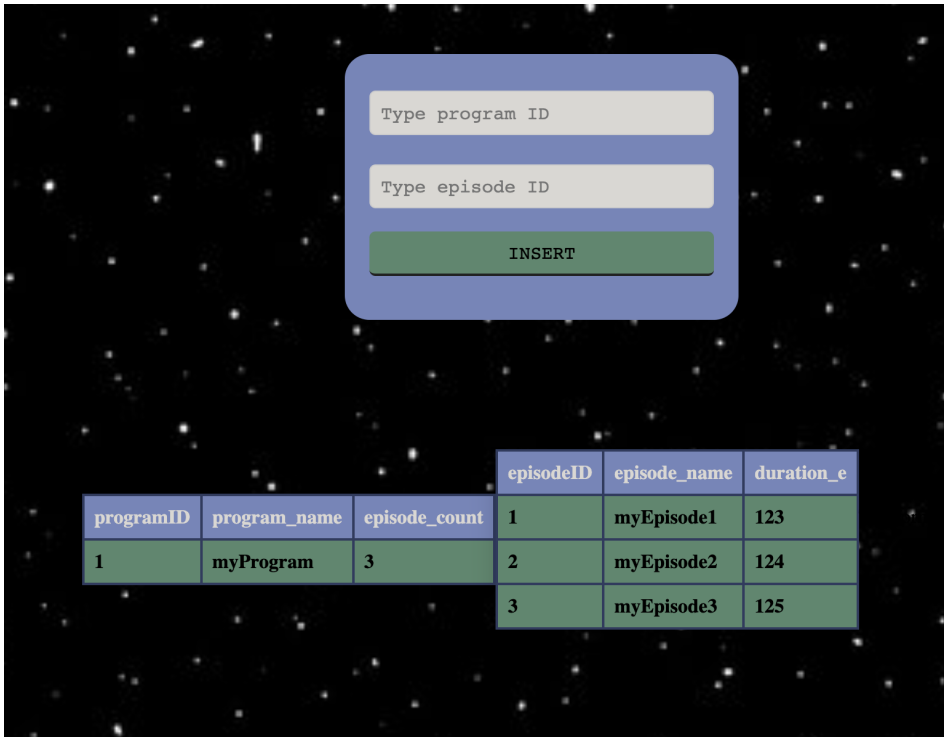
username	email	age	country	upassword	pl_ID	pl_name	song_count	username	pl_ID
denizcetin	cetindeniz@sabanciuniv.edu	20	Turkey	123456	1	myPlaylist	3	denizcetin	1

Podcast Program insertion page (Podcast Programs button):

Podcast Episode insertion page (Podcast Episodes button):



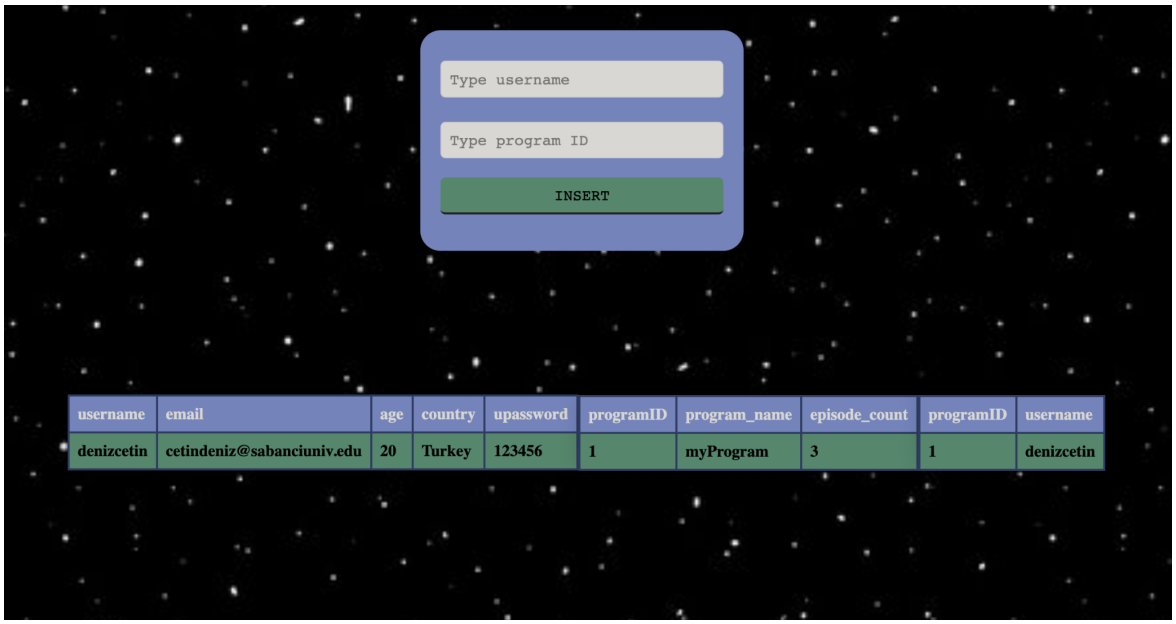
Episode-Program relation insertion page (Is in Program button):



programID	program_name	episode_count
1	myProgram	3

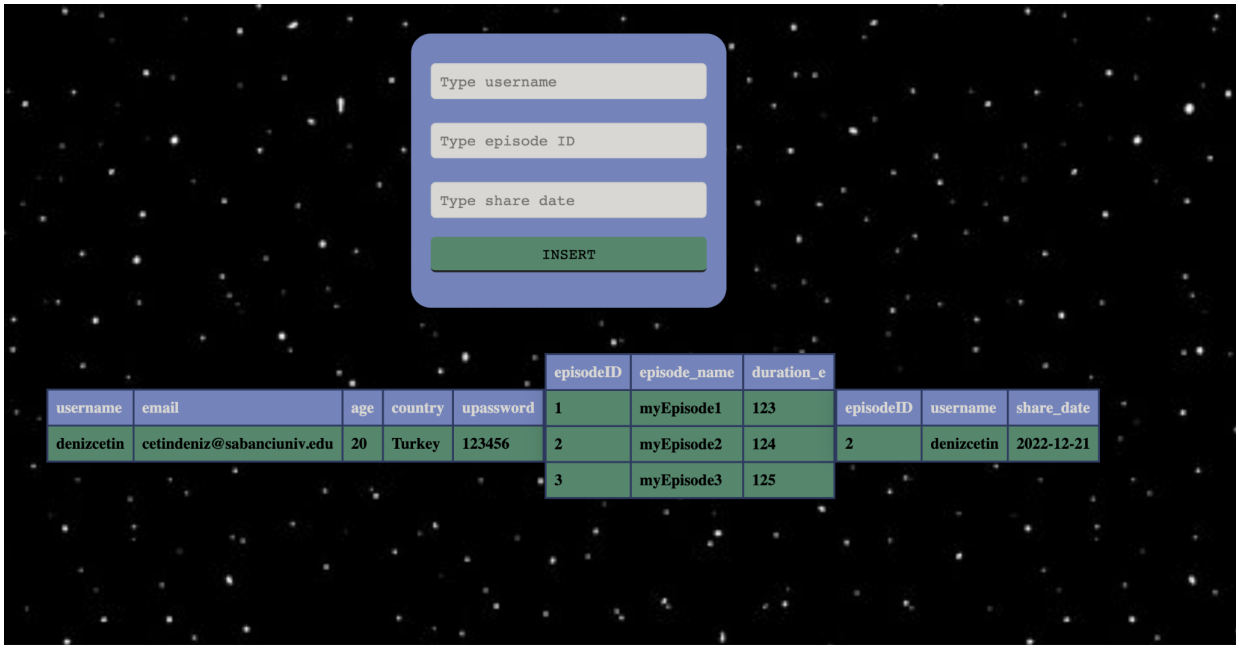
episodeID	episode_name	duration_e
1	myEpisode1	123
2	myEpisode2	124
3	myEpisode3	125

User-Program relation insertion page (Publishes button):



username	email	age	country	upassword	programID	program_name	episode_count	programID	username
denizcetin	cetindeniz@sabanciuniv.edu	20	Turkey	123456	1	myProgram	3	1	denizcetin

User-Episode relation insertion page (Shares button):

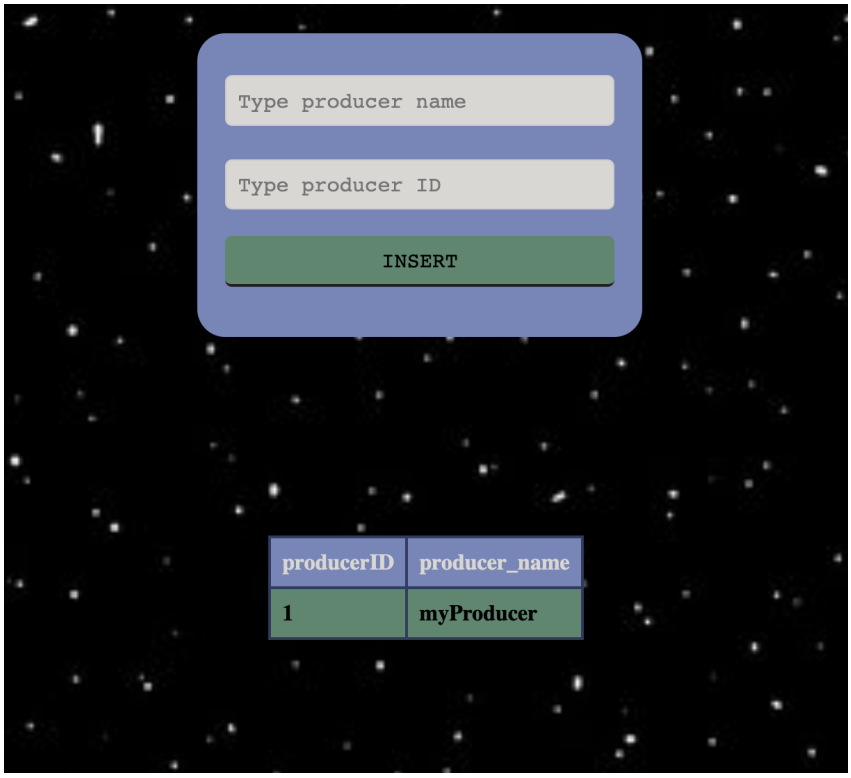


username	email	age	country	upassword
denizcetin	cetindeniz@sabanciuniv.edu	20	Turkey	123456

episodeID	episode_name	duration_e
1	myEpisode1	123
2	myEpisode2	124
3	myEpisode3	125

episodeID	username	share_date
2	denizcetin	2022-12-21

Producer insertion page (Producers button):



producerID	producer_name
1	myProducer

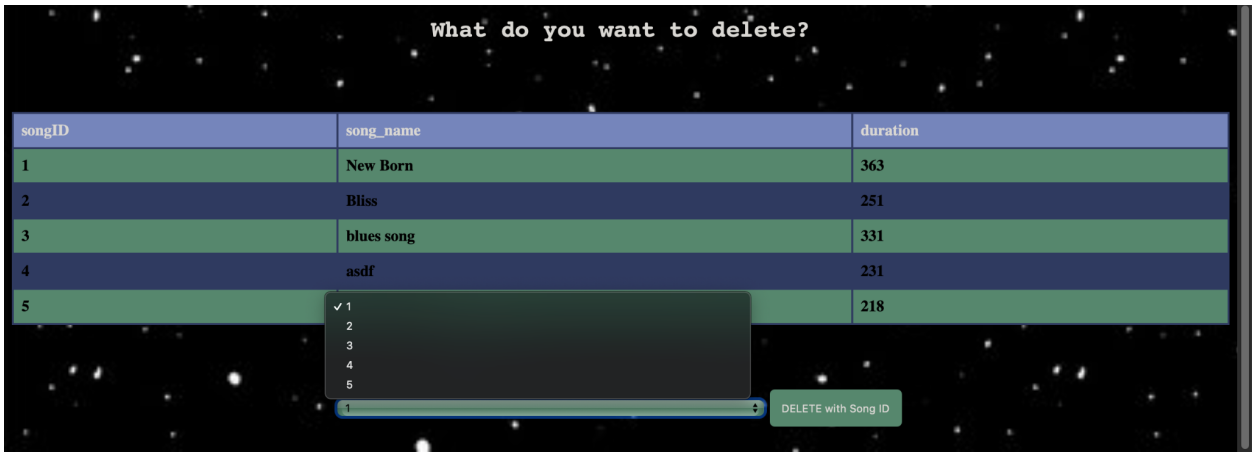
Album-Producer relation insertion page (Produces button):

DELETION

If the 'ids' parameter is present, the script will execute a DELETE statement on the 'songs' table, using the provided ID as a WHERE clause to specify which record should be deleted.

```
DELETE FROM songs WHERE songID = $selection_id
```

Main page's delete section before deletion:



After deletion:

Overall, this website allows users to view and modify the database of songs and artists by executing various SQL queries based on the user's input.

In the end, we were able to achieve the initial goals for this step of the project. We implemented a web-based administration panel for managing our website, which allows us to perform various operations on the database using web forms, including insertion, deletion, and selection (using filters applied through web forms that generate SQL statements to list relevant entries from the database on the browser). In addition to the administration panel, we also implemented other specified parts of the project according to the initial project specifications. This included making the website functional and enhancing its appearance with HTML and CSS (though we plan to continue improving the design before the final submission). Finally, we added features that covered at least the number of tables equal to the number of group members.