**TEAM 12**

**PROJECT TOPIC: Music Streaming Application Database**

**Names**

**Maruti Mohit Rayalacheruvu –** [**rayalacheruvu.m@northeastern.edu**](mailto:rayalacheruvu.m@northeastern.edu)

**Meghana Sai Vattikuti –** [**vattikuti.m@northeastern.edu**](mailto:vattikuti.m@northeastern.edu)

**Prakruthi Panakanahalli Somashekaraiah –** [**panakanahallisomas.p@northeastern.edu**](mailto:panakanahallisomas.p@northeastern.edu)

**Sumanayana Konda –** [**konda.su@northeastern.edu**](mailto:konda.su@northeastern.edu)

**TABLE OF CONTENT**

1. **Overview 2**
2. **Entity Relationship Diagram 3**
3. **Business Rules 4**
4. **Views 5**
5. **Data Flow Diagrams 8**
6. **Security and Permissions 13**
7. **Tables 15**

# Overview

The Music Database System project is a database management system that aims to provide a comprehensive platform for managing and organizing music data, including artists, albums, songs, genres, playlists, and user profiles. The system consists of three main user roles: admin, artist, and user, each with their specific permissions and privileges.

Each role in the Music Database System has its importance and purpose. The Admin role is critical for managing the database and its users, the Artist role is essential for managing artist data, and the User role is crucial for providing a user-friendly interface for music enthusiasts to interact with the music data in the database. Together, these roles ensure that the Music Database System can effectively manage music data, providing a centralized platform for the music industry to manage and organize their music data.

The database schema is designed to provide data integrity and security, with different levels of permissions and access control for each user role. The system includes tables for managing user profiles, music data, and other related information, including playlists and favorites. It has the following advantages:

1. Centralized Music Data Management: The Music Database System provides a centralized location to manage music data, which can reduce the chances of data inconsistencies and errors.
2. Access Control: The system offers different levels of permissions and access control to ensure that only authorized users can access and modify data.
3. Data Security: The Music Database System is designed to protect against data loss, corruption, and unauthorized access by implementing data backup and recovery strategies and user authentication.
4. Efficiency: With its user-friendly interface and efficient database schema, the system can streamline music-related tasks, such as searching, streaming, and organizing music collections.
5. Customizability: The Music Database System can be customized to fit the needs of different user roles, providing a flexible and adaptable platform for managing music data.

Overall, the Music Database System project is an essential tool for managing music data and streamlining music-related tasks. It provides a secure and efficient way to manage music data, ensuring that users can easily search, stream, and organize their music collections. The project's robust functionality and user-friendly interface make it a valuable resource for musicians, music industry professionals, and music enthusiasts alike.

# Entity Relationship Diagram

The entire database system has 9 main Tables and relationships respectfully as shown below.

Diagram

Description automatically generated

# Business Rules

Admin

1. Admins can create and manage user accounts.
2. Admins can add or remove songs, albums, and artists from the music streaming service.
3. Admins can view and manage all transactions.
4. Admins can view user data.
5. Admins can add ratings on songs.

Artist

1. Artists can upload their music to the music streaming service for streaming.
2. Artists can view and update their own artist profiles.
3. Artists can view ratings for their music.
4. Artists cannot make changes on other records apart from their own.

User

1. Users can create and manage their own playlists on the music streaming service.
2. Users can view songs, artists, and albums.
3. Users can mark songs as favorite.
4. Users can view and update their own user profiles.
5. Users cannot make changes on other records apart from their own.

# Views

A view is a virtual table based on the results of a SQL statement. The fields in view are fields from one or more real tables in the database.

View 1: NUMBER\_OF\_SONGS\_IN\_GENRE

* This will display the number of songs in each genre in no particular order
* The ADMIN, ARTIST have read permissions over this.

|  |
| --- |
| **Number of Songs in Genre View** |
| Genre\_Name |
| Song\_Count |

View 2: NUMBER\_OF\_SONGS\_IN\_PLAYLISTS

* This will display the number of songs in all playlists ranked by descending order
* The ADMIN will have read permissions over this

|  |
| --- |
| **Number of Songs in Playlists View** |
| Playlist\_ID |
| Playlist\_Name |
| Song\_Count |

View 3: SONGS\_BY\_RATING

* This will display all songs ranked by rating in descending order
* The ADMIN, ARTIST, USER will all have read permissions over this

|  |
| --- |
| **Songs by Rating View** |
| Song\_ID |
| Song\_Name |
| Song\_Artist\_Name |
| Song\_Album\_ID |
| Song Rating |

View 4: SONGS\_RELEASED\_BY\_ARTIST

* This will display the number of songs released by each artist ranked descending order
* The ADMIN, ARTIST, USER will all have read permissions over this

|  |
| --- |
| **Songs Released by Artist View** |
| Artist\_ID |
| Artist\_Name |
| Song\_Count |

View 5: ACTIVE\_USERS

* This will display the user names of all users who are active
* The ADMIN will all have read permissions over this

|  |
| --- |
| **Active Users View** |
| User\_ID |
| User\_FName |
| User\_LName |

View 6: USERS \_GENRE\_PREFERENCE

* This will display the genre preference of a particular
* The ADMIN, USER will have read permissions over this

|  |
| --- |
| **Users Genre Preference View** |
| User\_ID |
| User\_Username |
| Genre\_Prefence\_Count |

View 7: DISPLAY\_FAVOURITE\_SONGS\_FOR\_A\_USER

* It will display all the songs that are marked as a favorite by a user
* THE ADMIN, USER will have read permissions over this

|  |
| --- |
| **Display Favorite Songs For a User View** |
| Song\_ID |
| Song\_Name |
| Song\_Artist\_ID |
| Song\_Genre\_ID |
| Song\_Rating |

View 8: RECOMMENDATIONS\_FOR\_USER

* This will show the songs that are recommended to a user based on their listening abilities
* The ADMIN, USER will have read permissions over this

|  |
| --- |
| **Recommendation for User View** |
| Song\_ID |
| Song\_Artist\_Name |

View 9: MOST\_RATED\_ALBUM

* This will show the most rated album by the rating
* THE USER, ADMIN, ARTIST all have read permissions

|  |
| --- |
| **Most Rated Album View** |
| Album\_ID |
| Album\_Name |
| Average\_Song\_Rating |

# Data Flow Diagrams and Flowcharts

The Overall flow of the database system is as follows:

Chart, diagram, box and whisker chart

Description automatically generated with medium confidence

Role-based

1. Admin: The functioning and flow all the activities Admin can perform is as follows

Diagram

Description automatically generated

1. Artist: The functioning and flow all the activities Artist can perform is as follows

Diagram

Description automatically generated

1. User: The functioning and flow all the activities User can perform is as follows

Diagram

Description automatically generated

Process-based

1. Albums module: The data flow of Creation, Updating and Deletion of ‘Albums’ by ‘Artist’ is as follows.

Diagram

Description automatically generated

1. Create artist: The dataflow of Creation of the Role ‘Artist’ is as follows.

Diagram

Description automatically generated

1. Create user: The dataflow of the Creation of the role ‘User’ is as follows.

Diagram

Description automatically generated

1. Favorites module: The data flow for Creation, Updating and Deletion of ‘Favorites’ is as follows.

Diagram

Description automatically generated

1. Playlists module: The dataflow for Creating, Adding songs, Deleting and Updating ‘Playlists’ is as follows.

Diagram

Description automatically generated

1. Ratings module: The data flow for Adding, Updating and Deleting ‘Ratings’ for ‘Songs’ is as follows.

Diagram

Description automatically generated

1. Songs module: The data flow for Adding, Updating and Deleting ‘Songs’ is as follows:

Diagram

Description automatically generated

**Security and Permissions**

1. Admin

Admins have full access to all tables and functions in the database. They can add, modify, and delete data. Admins can also create new users and assign roles. Here are the specific permissions of the admin role:

* SELECT, INSERT, UPDATE, DELETE data in tables (User, Artist, Album, Songs, Genre)
* View data in Playlist table
* Create new users and assign roles
* Modify roles and permissions of other users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **SELECT** | **INSERT** | **UPDATE** | **DELETE** |
| **User** | ✔️ | ✔️ | ✔️ | ✔️ |
| **Artist** | ✔️ | ✔️ | ✔️ | ✔️ |
| **Album** | ✔️ | ✔️ | ✔️ | ✔️ |
| **Songs** | ✔️ | ✔️ | ✔️ | ✔️ |
| **Genre** | ✔️ | ✔️ | ✔️ | ✔️ |
| **Playlist** | ✔️ | X | X | X |

1. Artist

They can add new artists, albums, and songs to the database. They can also modify and delete data for the entities they created. Here are the specific permissions of the Artist role.

* SELECT, UPDATE, DELETE and INSERT data for Album, and Song tables for the entities they created
* SELECT, UPDATE and DELETE on own Artist data
* Cannot modify or delete data for other entities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **SELECT** | **INSERT** | **UPDATE** | **DELETE** |
| **User** | X | X | X | X |
| **Artist** | ✔️ | X | ✔️ | ✔️ |
| **Album** | ✔️ | ✔️ | ✔️ | ✔️ |
| **Favorite** | ✔️ | X | X | X |
| **Songs** | ✔️ | ✔️ | ✔️ | ✔️ |
| **Genre** | ✔️ | X | X | X |
| **Playlist** | X | X | X | X |

1. User

Users have limited access to the database, primarily to search for and stream music, create and manage playlists, and mark songs as favorites. Here are the specific permissions of the User role.

* View data in all tables (Songs, Artist, Album, Genre)
* SELECT, UPDATE and DELETE on own User data
* SELECT, INSERT, UPDATE and DELETE data in Playlist, Favorite tables for the entities they created
* Cannot modify or delete data for other entities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **SELECT** | **INSERT** | **UPDATE** | **DELETE** |
| **User** | ✔️ | X | ✔️ | ✔️ |
| **Artist** | ✔️ | X | X | X |
| **Album** | ✔️ | X | X | X |
| **Favorite** | ✔️ | ✔️ | ✔️ | ✔️ |
| **Songs** | ✔️ | X | X | X |
| **Genre** | ✔️ | X | X | X |
| **Playlist** | ✔️ | ✔️ | ✔️ | ✔️ |

# TABLES

## **Users:**

This table will store the details of Users. It has basic details of the user using the application.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| UsersID | NUMBER | Primary Key | User Identity Number |
| FirstName | VARCHAR (45) | NOT NULL | User First Name |
| LastName | VARCHAR (45) | NOT NULL | User Last Name |
| E-mail | VARCHAR (45) | NOT NULL, UNIQUE | User Email Address |
| Username | VARCHAR (45) | NOT NULL | Username of User |
| RegistrationDate | DATE | NOT NULL | Registration Date of the User |
| Status | VARCHAR (45) | NOT NULL | The status (Active or not) of the User |

## **Songs:**

The table will store the details of Songs that users listen to.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| SongID | NUMBER | Primary Key | Song Identity Number |
| SongName | VARCHAR (45) | NOT NULL | Song Name |
| SongArtistName | VARCHAR (45) | NOT NULL | Artist Name of the Song |
| Rating | VARCHAR (45) | NOT NULL | Rating of the Song |
| Length | TIME STAMP | NOT NULL | Length of the Song |
| Language | VARCHAR (45) | NOT NULL | Language of the Song |
| ReleaseDate | DATE | NOT NULL | Release Date of the Song |
| TrackNumber | NUMBER | NOT NULL | Track Number of the Song in Album |
| GenreID | NUMBER | Foreign Key | Genre ID of the Genre Song belongs to |
| AlbumID | NUMBER | Foreign Key | Album ID of the Album Song belongs to |
| ArtistID | NUMBER | Foreign Key | Artist ID of the Artist Song belongs to |

## **Album:**

The table will store the details of Album

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| AlbumID | NUMBER | Primary Key | Album Identity Number |
| AlbumName | VARCHAR (45) | NOT NULL | Album Name |
| Rating | NUMBER | NOT NULL | Rating of the Album |

## **Playlist:**

The table will store the details of Playlist

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| PlaylistID | NUMBER | Primary Key | Playlist Identity Number |
| PlaylistName | VARCHAR (45) | NOT NULL | Playlist Name |
| UsersID | Foreign Key | NOT NULL | User ID of the User Playlist belongs to |

## **Artist:**

The table will store the details of Artist

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| ArtistID | NUMBER | Primary Key | Artist Identity Number |
| FirstName | VARCHAR (45) | NOT NULL | Artist First Name |
| LastName | VARCHAR (45) | NOT NULL | Artist Last Name |
| Bio | VARCHAR (45) | NOT NULL | Description of the Artist |

## **Genre:**

The table will store the details of Genre

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| GenreID | NUMBER | Primary Key | Genre Identity Number |
| GenreName | VARCHAR (45) | NOT NULL | Genre Name |

## **Artist Album:**

The table will store the details of Artist Album

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| ArtistID | NUMBER | Foreign Key | Artist ID of the Artist Album Belongs to |
| AlbumID | NUMBER | Foreign Key | Album ID of the Album that belongs to Artist |

## **Songs Playlist:**

The table will store the details of Songs Playlist

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| SongID | NUMBER | Foreign Key | Songs ID of the Song that belongs to Playlist |
| PlaylistID | NUMBER | Foreign Key | Playlist ID that consists the Song |

## **Favorites:**

The table will store the details of Favorite songs.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |

|  |  |  |  |
| --- | --- | --- | --- |
| SongID | NUMBER | Foreign Key | Song ID of the Song User made as ‘Favorite’ |
| UsersID | NUMBER | Foreign Key | User ID of the User ‘Favorite’ belongs to |