****

**MUSEBOXD.**

**Software Sprint Phase – 1**

**Christ (Deemed To Be) University, Bangalore**

**Members:**

Priyak Sarkar, Kshitij Sakhuja

**Register Numbers:**

2241040, 2241028

**Dated – 16th January, 2025.**

#### ARCHITECTURAL DESIGN OF MUSEBOXD.

##### Tiered Architecture Overview:

Museboxd’s goal is to serve a broad audience with features like user-generated content, recommendations, and scalability, 3-tier architecture is the optimal choice. It provides the flexibility and robustness needed for handling large user bases and expanding functionalities.

1. **Presentation Layer (Client Tier)**

* The user interface (UI) for interaction, developed using React.Js, CSS, for a smooth and clean experience.
* Includes a huge database of every music ever, and updates regarding new music uploaded daily.

1. **Application Logic Layer (Middle Tier)**

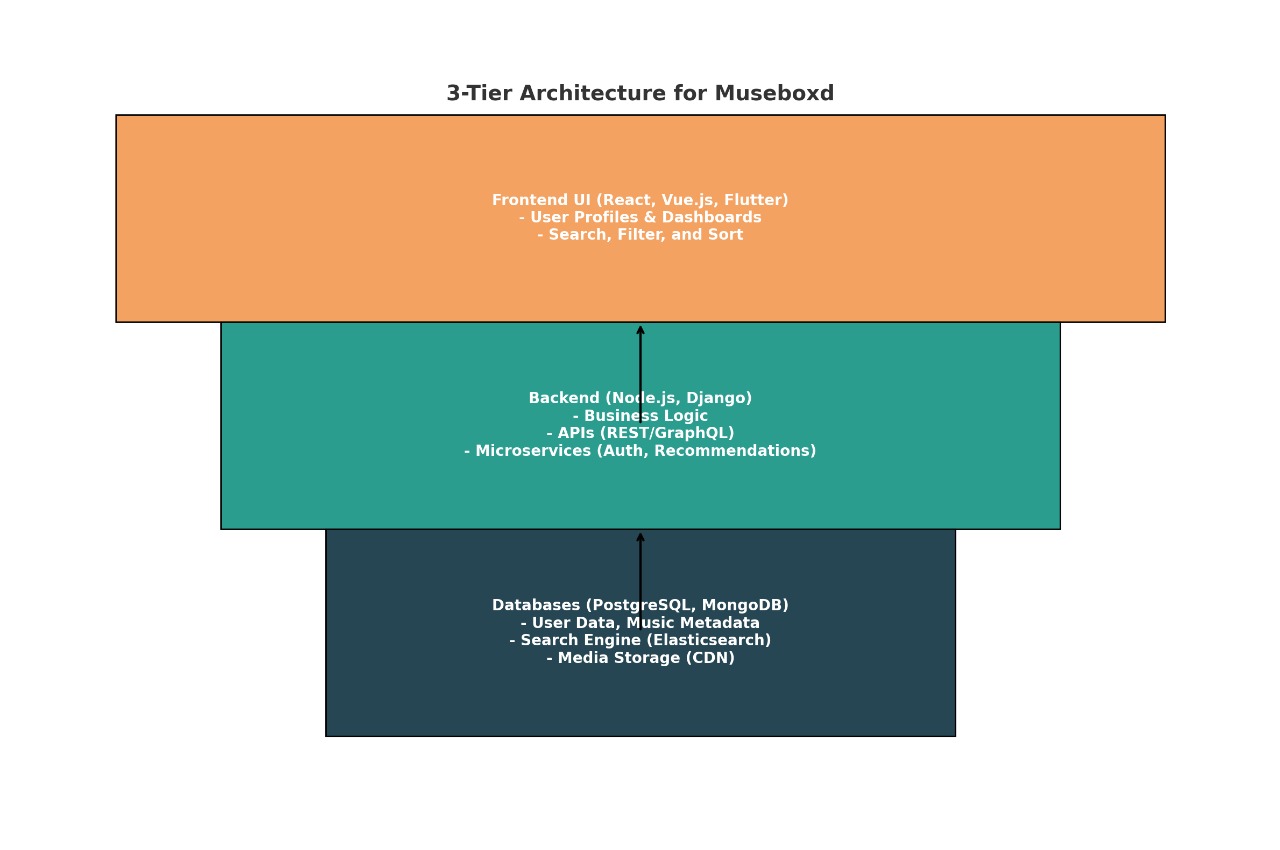
* Responsible for implementing the core functionalities like:

1. Logging feature to add your own diary of what you listen or review
2. Reviews reading and writing to get the best rating out of what people think altogether.
3. Fetching the music database from API and making API calls, for making or searching playlists.
4. Minimalistic UI elements, to keep it clean.
5. **Data Layer (Backend Tier)**

* HeidiSQL for:

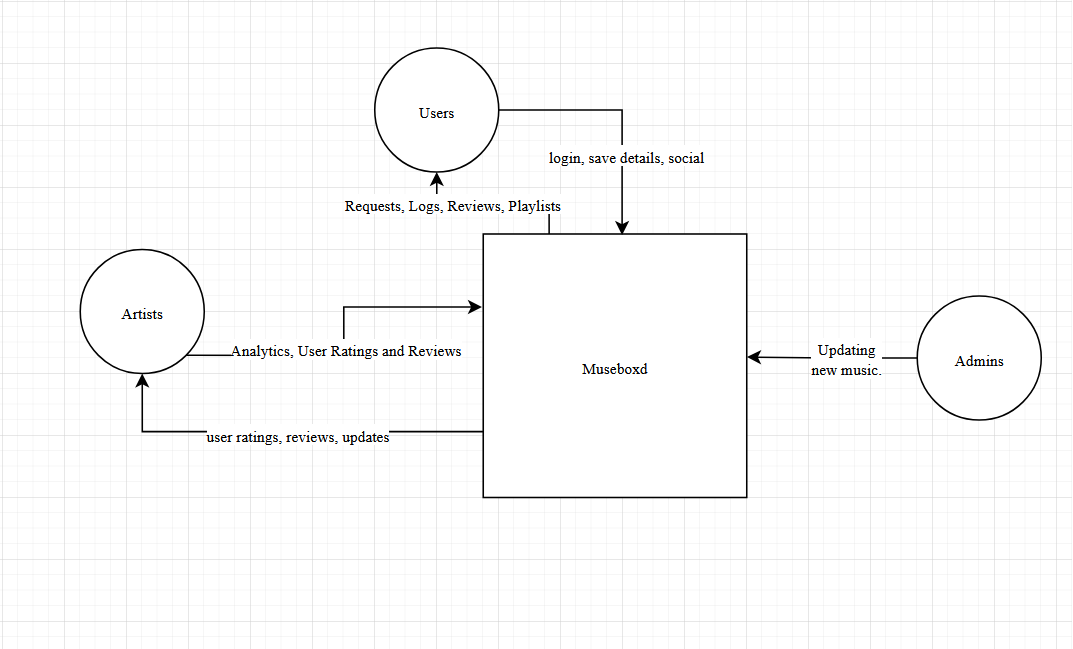
1. User Authentication (email, social logins)
2. Storage for playlists and every user diary feature.
3. Real-time database for maintaining tasks and feature logs.

* Using Node.Js to make API calls.
* API for music from thealbumoftheyear.org ‘s API.

**Three Tier Architectural Diagram**

#### DATA FLOW DIAGRAM & RESPECTIVE DATA DICTIONARY.

##### Level 0:



###### **Data Dictionary:**

**1.User**

The end user interacting with Museboxd.

**Attributes:**

* + - Music Library: An entire music library and upcoming music.
    - Preferences: User-defined settings such as accessibility preferences, language options, and volume levels.

**2.Museboxd**

The core processing system responsible for analyzing inputs, processing them, and interacting with external services.

**Attributes:**

* + - Review System: Identifies objects in the environment.
    - Logging Feature: Stores a diary like feature to store all the music that you listen or review that particular day.
    - Social Activity and Playlists Creation: Users can have a social platform with their friends and read anything about music here.

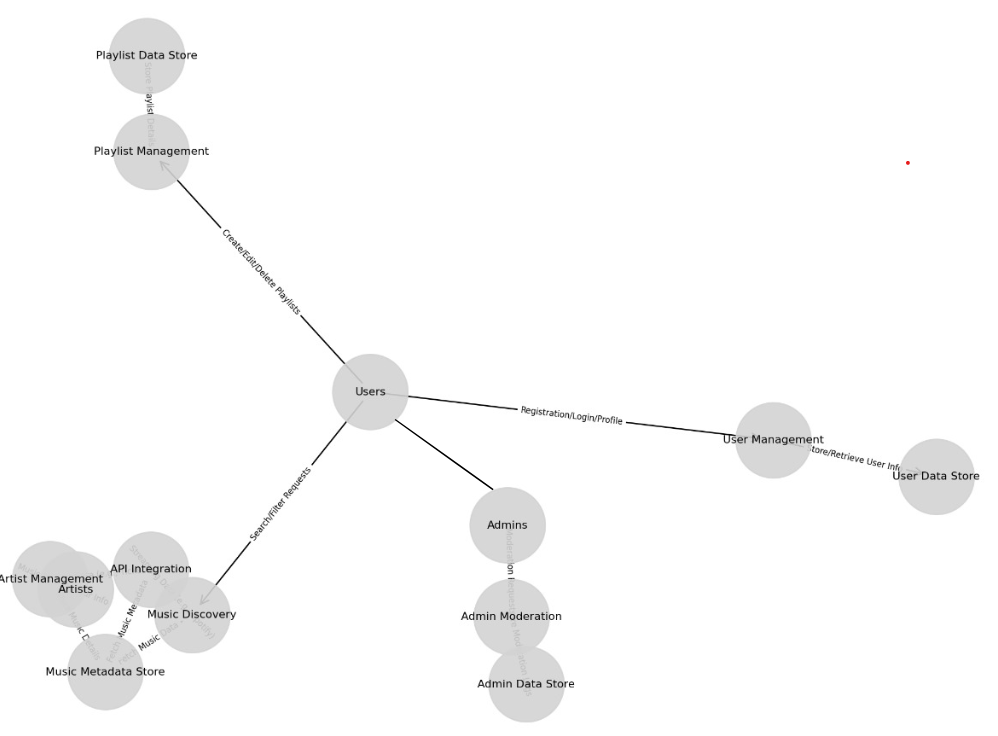
**3.External Services**

Third-party services used to enhance system functionality.

**Attributes:**

* + - API Service: For fetching and reading the database.
    - External Links: For direct link to stream or watch videos.

##### Level 1:



###### **Data Dictionary:**

1. **User**

The end user interacting with Museboxd.

**Attributes:**

* **Music Library Access:** Provides access to the entire music library, including past listens, favorites, and upcoming releases.
* **Preferences:** User-configurable settings such as accessibility options, language selection, notification preferences, and volume control.
* **Interaction:** Users can search, filter, and interact with the music database, leave reviews, and connect socially.

1. **Museboxd**

The core processing system responsible for analyzing user inputs, managing data flows, and interacting with external services.

**Attributes:**

* **Music Data Processing:** Handles the retrieval, categorization, and presentation of music data.
* **Review and Diary System:** Allows users to log and review their daily music activities in a diary-like format.
* **Social Platform:** Facilitates community interaction through playlists sharing, social updates, and forums for music discussion.
* **Playlist Management:** Enables users to create, edit, and share personalized playlists.
* **Analytics Engine:** Tracks user behavior and preferences to offer recommendations and insights.

1. **External Services**

Third-party services integrated into Museboxd to expand functionality and enrich user experience.

**Attributes:**

* **Music and Video APIs:** Fetches data about artists, albums, songs, and videos from external music platforms and streaming services.
* **Streaming Links:** Provides direct links for users to stream or purchase music and watch related videos.
* **Social Media APIs:** Facilitates sharing playlists, reviews, and updates on social platforms.
* **Payment Gateways:** Supports purchasing concert tickets, albums, or other merchandise directly through the platform.

#### DATABASE DESIGN FOR MUSEBOXD.

**1. Users Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| UserID | INT | PRIMARY KEY, AUTO\_INCREMENT |
| FullName | VARCHAR(255) | NOT NULL |
| Email | VARCHAR(255) | UNIQUE, NOT NULL |
| Password | VARCHAR(255) | NOT NULL |
| Preferences | JSON | Stores user-defined settings (e.g., accessibility, language, volume) |
| AccountCreated | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**2. MusicLibrary Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| MusicID | INT | PRIMARY KEY, AUTO\_INCREMENT |
| Title | VARCHAR(255) | NOT NULL |
| Artist | VARCHAR(255) | NOT NULL |
| Album | VARCHAR(255) |  |
| Genre | VARCHAR(100) |  |
| ReleaseDate | DATE |  |
| ExternalLink | VARCHAR(255) | URL to stream or watch the music video |

**3. UserLogs Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| LogID | INT | PRIMARY KEY, AUTO\_INCREMENT |
| UserID | INT | FOREIGN KEY REFERENCES Users(UserID) |
| MusicID | INT | FOREIGN KEY REFERENCES MusicLibrary(MusicID) |
| PlayTimestamp | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |
| Review | TEXT | Stores user-written reviews for songs |

**4. Playlists Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| PlaylistID | INT | PRIMARY KEY, AUTO\_INCREMENT |
| UserID | INT | FOREIGN KEY REFERENCES Users(UserID) |
| Name | VARCHAR(255) | NOT NULL |
| CreationDate | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**5. PlaylistSongs Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| PlaylistSongID | INT | PRIMARY KEY, AUTO\_INCREMENT |
| PlaylistID | INT | FOREIGN KEY REFERENCES Playlists(PlaylistID) |
| MusicID | INT | FOREIGN KEY REFERENCES MusicLibrary(MusicID) |

**6. SocialActivity Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| ActivityID | INT | PRIMARY KEY, AUTO\_INCREMENT |
| UserID | INT | FOREIGN KEY REFERENCES Users(UserID) |
| ActivityType | ENUM('Like', 'Comment', 'Share') |  |
| RelatedMusicID | INT | FOREIGN KEY REFERENCES MusicLibrary(MusicID) |
| ActivityTimestamp | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

**7. ExternalAPILinks Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| APIID | INT | PRIMARY KEY, AUTO\_INCREMENT |
| ServiceName | VARCHAR(255) | NOT NULL |
| APIEndpoint | VARCHAR(255) | NOT NULL |
| Description | TEXT |  |

#### NORMALIZATION OF THE DATABASE.

1. **First Normal Form (1NF)**

* All tables have a primary key (e.g., UserID in Users, MusicID in MusicLibrary, etc.).
* Each column holds atomic values. For instance, preferences are stored as JSON, and a playlist contains rows in PlaylistSongs for each song.

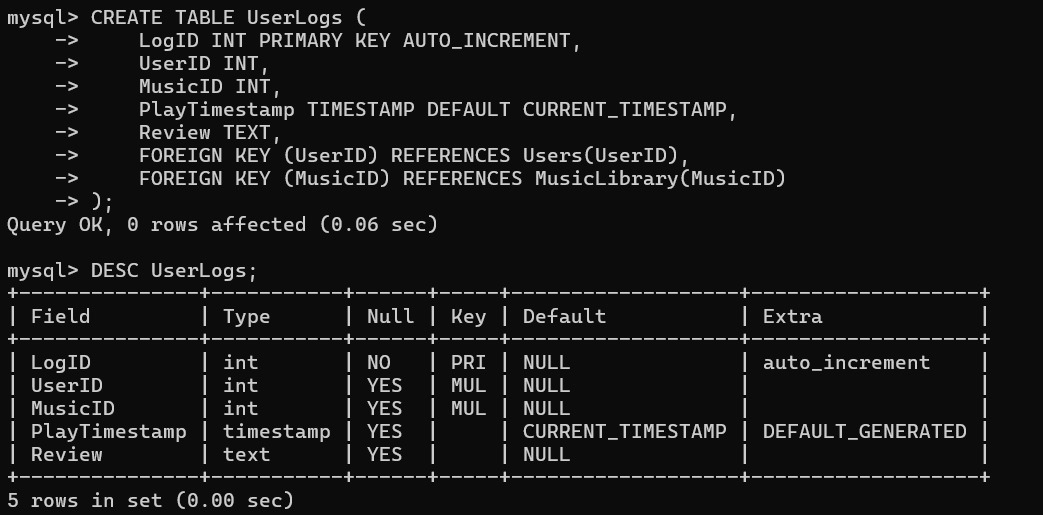
1. **Second Normal Form (2NF)**

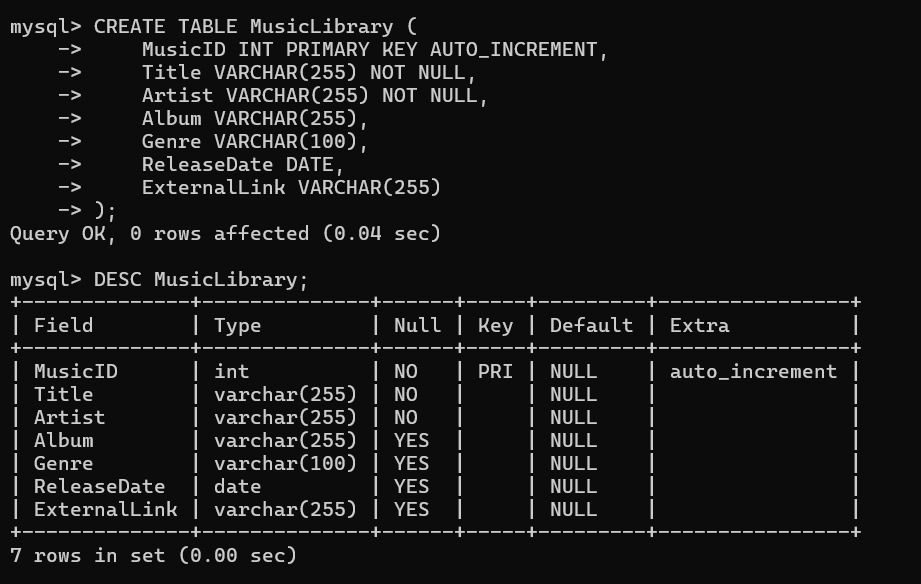
* Each table ensures no partial dependency:
  + In UserLogs, MusicID and Review depend entirely on LogID, which is the primary key.
  + In PlaylistSongs, PlaylistID and MusicID both depend on the composite primary key PlaylistSongID.
  + No column is dependent on a part of a composite key.

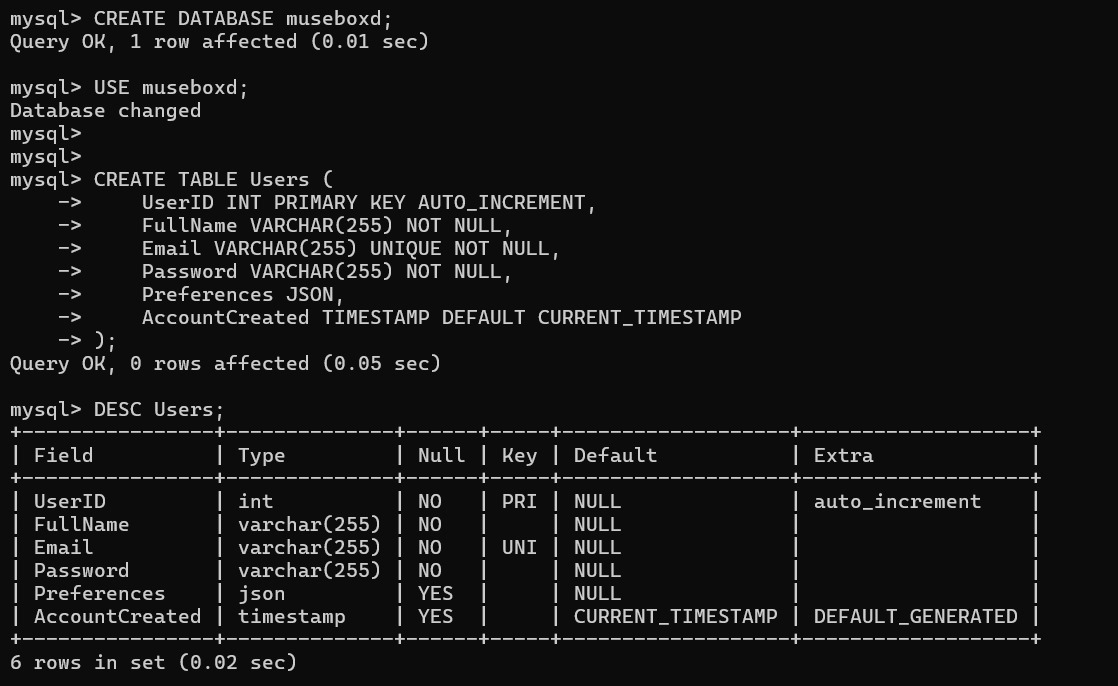
1. **Third Normal Form (3NF)**

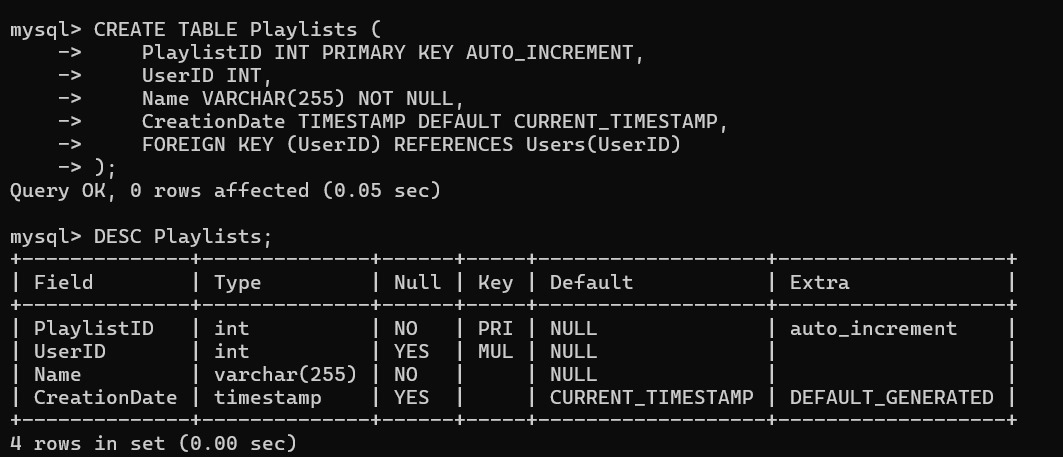
* No transitive dependencies:
  + In Users, non-key attributes (e.g., FullName, Email, Preferences) depend only on UserID.
  + In MusicLibrary, non-key attributes (e.g., Title, Artist, Genre) depend only on MusicID.
  + No attributes in any table depend on other non-key attributes.

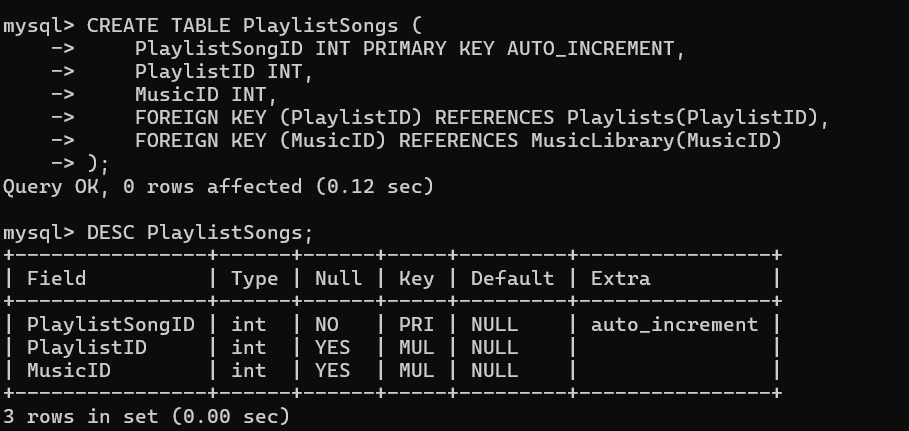
##### **Creation of tables and insertion of records:**

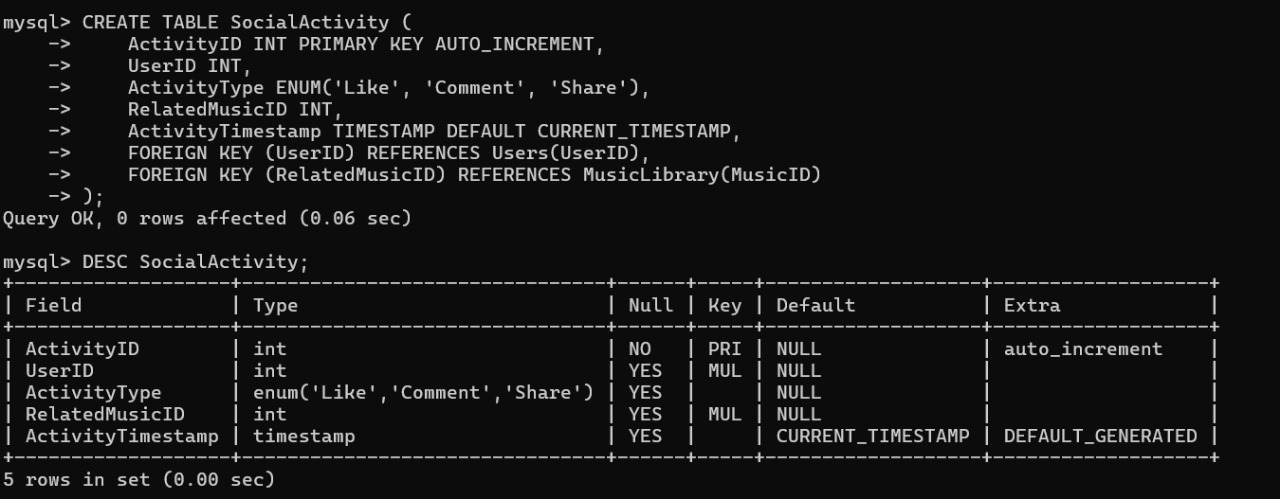


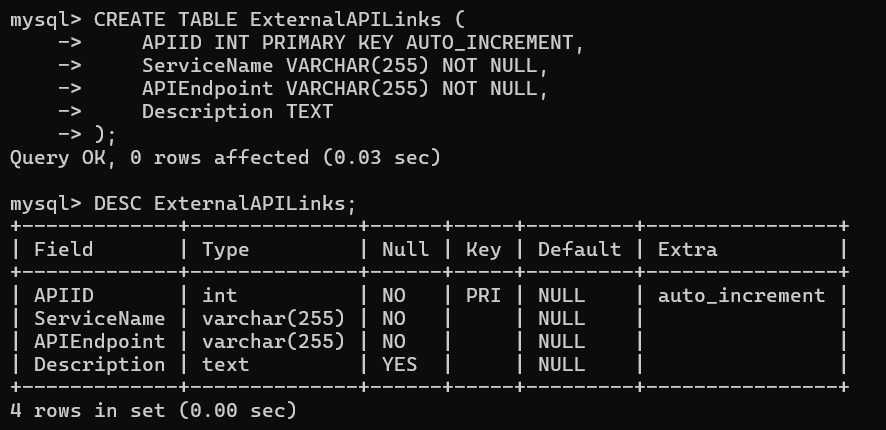












#### INTERFACE DESIGN.

