**Data Management Plan (Iteration 2)**

**Link to wiki Page:** [**https://github.com/ShivaniPandula/MusicMix\_GDP-05/wiki/Data-Management-Plan-(Iteration-2)#song-play-list**](https://github.com/ShivaniPandula/MusicMix_GDP-05/wiki/Data-Management-Plan-(Iteration-2)#song-play-list)

Data Management Plan for MusicMix: Summary of Data to Store:

**Users Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| User ID | Unique identifier for each user. |
| Song ID | Unique identifier for each song |
| Username | User's chosen username for the platform. |
| Email | User's email address for communication/login. |
| Password | Hashed and salted password for authentication. |
| Gender | Gender of the Person |
| DOB | Date of Birth of person |
| PhoneNumber | Phone Number of a Person |

### **Playlist Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Playlist ID | Unique identifier for each playlist. |
| User ID | Foreign key linking to the playlist creator. |
| Details | Details of the Playlist. |

### **Song Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Song ID | Unique identifier for each song. |
| Song Name | Name of the song. |
| Album ID | Foreign key linking to the album the song belongs to. |
| Artist ID | Foreign key linking to the artist who performed the song. |
| Genre | Genre classification of the song. |
| Lyrics | Text containing the lyrics of the song. |
| Duration | Length of the song in seconds. |

### **Album Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Album ID | Unique identifier for each album. |
| Album Name | Name of the album. |
| Release Date | Date when the album was released. |

### **Artist Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Artist ID | Unique identifier for each artist. |
| Artist Name | Name of the artist. |
| Bio | Biography or description of the artist. |

### **Admin Data**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Admin ID | Unique identifier for each admin user. |
| User Name | Username of the admin user. |
| Password | Hashed and salted password for authentication. |

### **Song Artist**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Song Artist ID | Unique identifier for each song Artist. |
| Song ID | Unique identifier for each song. |
| Artist ID | Unique identifier for each Artist. |

### **Album Artist**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Album Artist | Name of the Artist in the Album. |
| Album ID | Unique Identifier for each album. |
| Artist ID | Unique identifier for each Artist. |

### **Song Play List**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Song Play List ID | Unique identifier for the song playlist. |
| Song ID | Unique identifier for each song. |
| Play List ID | Unique identifier for each playlist. |

### **Feedback**

|  |  |
| --- | --- |
| **Field** | **Description** |
| Feedback ID | Unique identifier for the Feedback. |
| Feedback Text | Textual content of the feedback provided by the user. |
| User Id | Foreign key linking to the User. |

A diagram of a computer

Description automatically generated with medium confidence

# Relationships

## **User - Playlist**

* One-to-Many relationship: One user can have multiple playlists.
* Foreign key: User ID in the Playlist table.

## **User - Song**

* Many- to-one relationship: Many Users can have single song.

## **Playlist - SongPlayList**

* one-to-Many relationship: A playlist can contain multiple songs, and a song can belong to multiple playlists.

## **Song - SongPlayList**

* One-to-Many relationship: A song can be in multiple SongPlayList and a Song can belong

## **Song - Album**

* Many-to-One relationship: Many songs can belong to one album.
* Foreign key: Album ID in the Song table.

## **Song - SongArtist**

* One-to-Many relationship: Many songs can be performed by one artist.
* Foreign key: Artist ID in the Song table.

## **Artist - SongArtist**

* One-to-Many relationship: An Artist can Sing many Songs.

## **Album - AlbumArtist**

* One-to-Many relationship: One album can be created by Many artist.
* Foreign key: Artist ID in the Album table.

## **Artist - AlbumArtist**

* one-to-Many relationship: one Artist Can be in many in AlbumArtist Table.

## **User - feedback**

* one-to-Many relationship: one user can have zero or many feedbacks.

# Initial Plans to Secure Data

## **Authentication and Authorization:**

* Implement secure authentication mechanisms like OAuth or JWT for user login.
* Use role-based access control (RBAC) to restrict access to certain features or data based on user roles (e.g., admin, regular user).

## **Data Encryption:**

* Encrypt sensitive data like passwords and any other personal information using industry-standard encryption algorithms (e.g., bcrypt for passwords).
* Ensure that data transmission between the client and server is encrypted using HTTPS.

## **Input Validation and Sanitization:**

* Implement input validation and sanitization to prevent SQL injection, XSS (Cross-Site Scripting), and other common security vulnerabilities.
* Validate user inputs on the client-side and server-side to ensure data integrity.

## **Database Security:**

* Apply database security measures such as using strong database credentials, restricting database access to authorized users only, and regularly updating the database management system (DBMS) to the latest version with security patches.

## **Backup and Disaster Recovery:**

* Regularly backup the database to prevent data loss in case of system failures, cyber attacks, or other unforeseen events.
* Implement disaster recovery plans to quickly restore the system and data in case of emergencies.

## **Logging and Monitoring:**

* Enable logging of user activities, database transactions, and system events to track suspicious activities and detect security breaches.
* Set up real-time monitoring tools to continuously monitor system performance and security metrics.

## **Security Training and Awareness:**

* Provide security training and awareness programs for developers, administrators, and users to educate them about common security threats, best practices, and compliance requirements.
* Encourage users to use strong, unique passwords and enable multi-factor authentication (MFA) for added security.

# Mapping of Functional Requirements to Data Storage

### **User Authentication Management:**

* Store user data (e.g., username, email, password hash) in the User table.
* Session tokens for "Remember Me" functionality can be stored in the User table or a separate Session table.
* Store password reset tokens and associated expiration timestamps in the User table.
* Store User profile information (e.g., name, address) in the User table.

### **Music Management:**

* Store music content information (e.g., artists, albums, playlists) in respective tables (Artist, Album, Playlist).
* Store user-created playlists and their track associations in a Playlist table.
* Additional playlist metadata (e.g. details) may be stored in the Playlist table.

### **Social Interaction Management:**

* Store user-follow relationships in a Follow table with references to the User table.