

DBMS MINI PROJECT

SPOTIFY DATABASE MANAGEMENT SYSTEM

TEAM MEMBERS :

Sajal Jaiswal – PES1UG21CS517

Samar Pratap – PES1UG21CS522

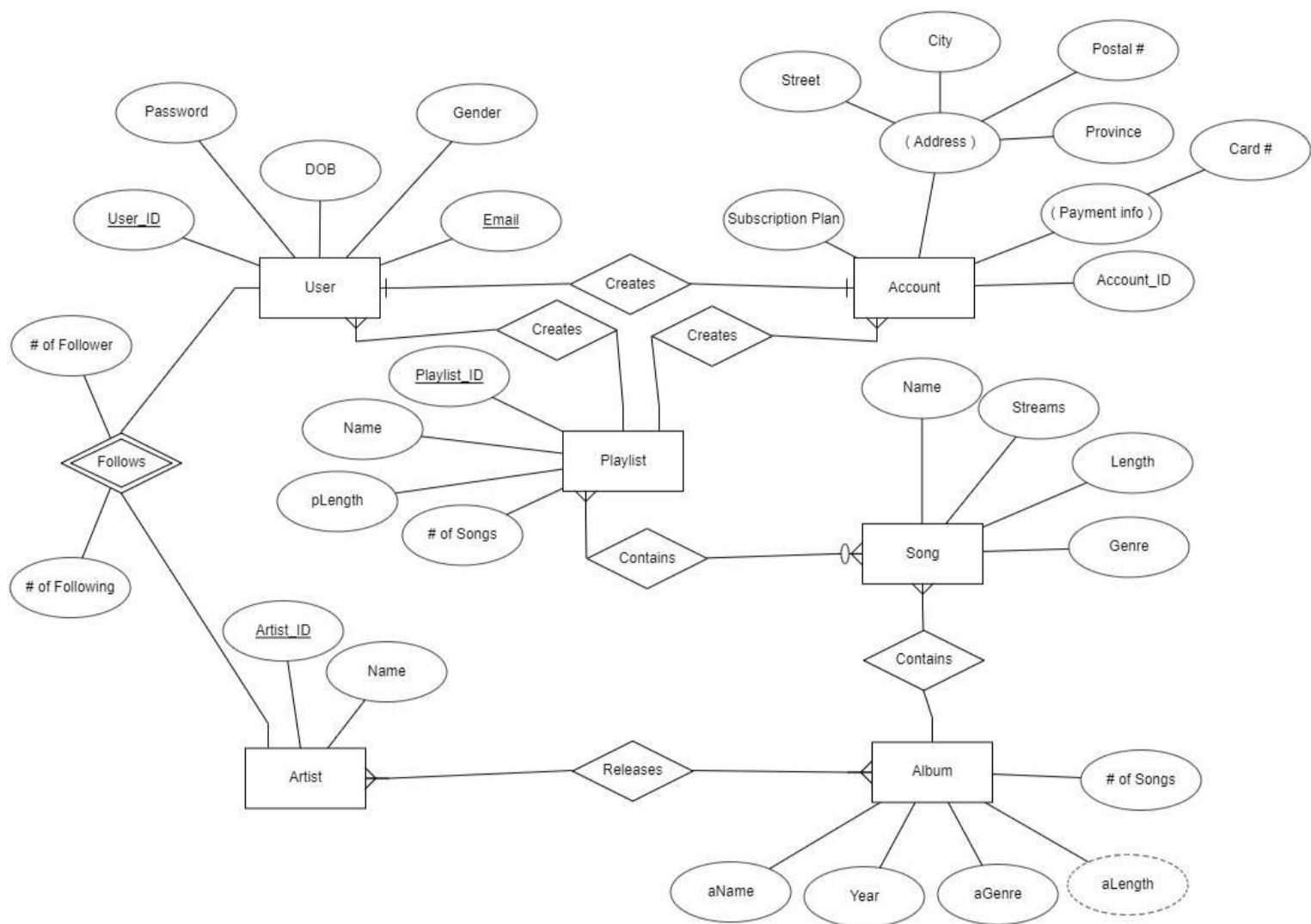
SECTION – I

CONTENTS

ER – DIAGRAM

RELATIONAL SCHEMA

SRS



User	
P_K	User_ID
	DOB
	Password
	Gender
	Email

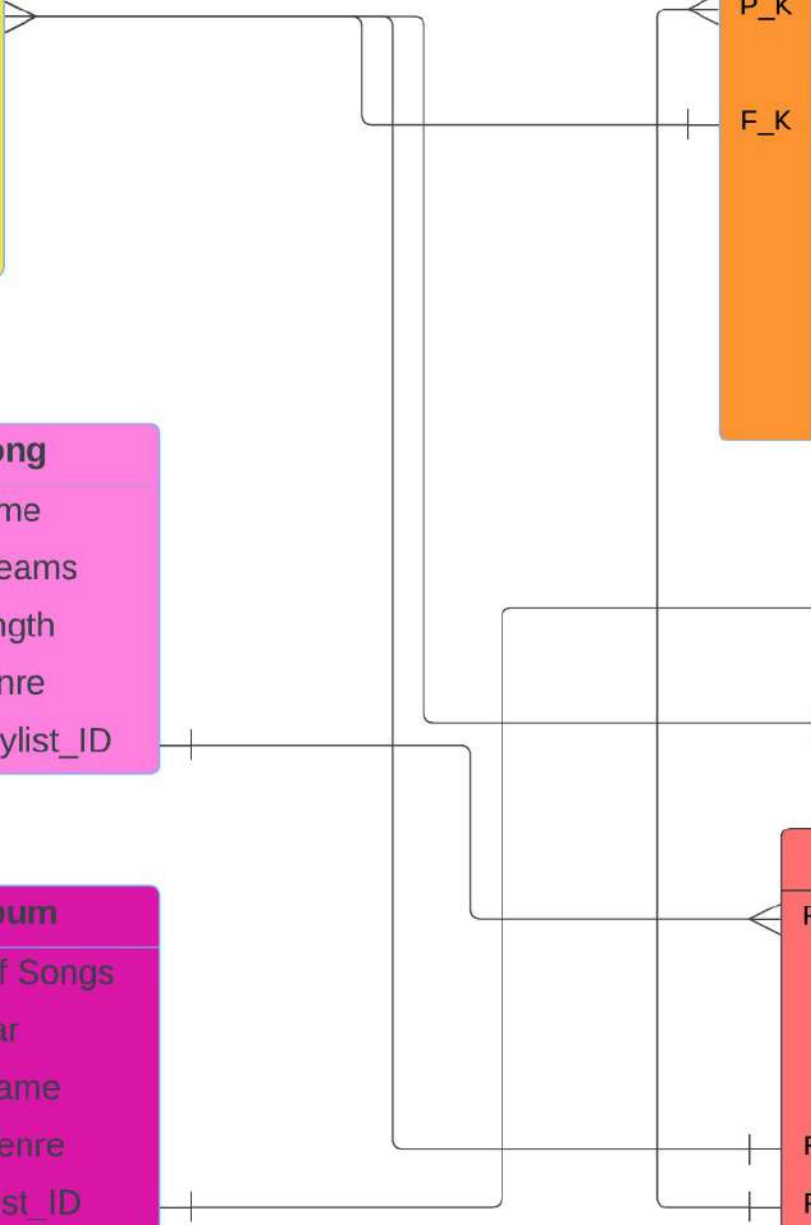
Account	
P_K	Account_ID
	Subscription_Plan
F_K	User_ID
	Card #
	City
	Postal #
	Province
	Street

Song	
	Name
	Streams
	Length
	Genre
F_K	Playlist_ID

Artist	
P_K	Artist_ID
	Name
F_K	User_ID

Album	
	# of Songs
	Year
	aName
	aGenre
F_K	Artist_ID

PlayList	
P_K	Playlist_ID
	Name
	pLength
	# of SONGs
F_K	User_ID
F_K	Account_ID



Spotify Database Management System SRS

1. Introduction

1.1 Purpose of the Project

The purpose of the Spotify Database Management System is to develop a backend system that utilizes the Spotify API to fetch user data and store it in a SQL database. The system will enable users to perform various queries and operations on the stored data, such as searching for songs, viewing and searching through playlists, and exploring artist and album information.

1.2 Scope of the Project

The project will focus on the development of a backend system that interacts with the Spotify API to retrieve and manage user data. It includes the creation and management of entities like users, accounts, songs, playlists, artists, and albums, along with their respective attributes and relationships. The scope also encompasses the development of a set of system features and functionalities that allow users to interact with the stored data through queries and operations.

2. Project Description

2.1 Project Overview

The Spotify Database Management System aims to provide users with a seamless experience for accessing and managing their Spotify-related data. It involves the integration of the Spotify API to fetch real-time data and store it in a SQL database. Users can then perform various operations on this data, such as creating playlists, following artists, and exploring song details.

2.2 Major Project Functionalities

User Authentication: Users can log in to their Spotify accounts and grant access to their data.

Data Retrieval: The system fetches user-specific data from the Spotify API, including playlists, songs, artists, and albums.

Data Storage: Retrieved data is stored in a SQL database, preserving the relationships between entities.

Query and Search: Users can perform queries to search for songs, artists, playlists, and more.

Playlist Creation: Users can create and manage their playlists, adding songs from their library.

Artist and Album Exploration: Detailed information about artists and albums can be accessed.

User Account Management: Users can manage their accounts, update personal information, and change preferences.

Follow/Favorite: Users can follow artists, albums, and other users, and mark songs as favorites.

3. System Features and Functional Requirement

System Feature 1: User Authentication

Entities: User, Authentication Service, Spotify API

The system shall provide a user authentication mechanism, allowing users to log in using their Spotify credentials.

The system shall use OAuth 2.0 for secure authentication and authorization with the Spotify API.

Users shall be able to grant access to their Spotify data during the authentication process.

System Feature 2: Data Retrieval

Entities: User, Spotify API, Database (Song, Playlist, Artist, Album)

The system shall make API calls to the Spotify API to fetch user-specific data, including playlists, songs, artists, and albums.

The system shall handle rate limiting and pagination when fetching large datasets.

Retrieved data shall be processed and prepared for storage in the database.

System Feature 3: Data Storage

Entities: Database (User, Song, Playlist, Artist, Album)

The system shall store fetched data in a SQL database, maintaining the integrity of the data and relationships between entities.

Each entity (user, song, playlist, artist, album, etc.) shall have a corresponding database table.

The database schema shall support efficient querying and retrieval of data.

System Feature 4: Query and Search

Entities: User, Database (Song, Playlist, Artist, Album)

Users shall be able to perform text-based queries to search for songs, playlists, artists, and albums.

The system shall provide an advanced search feature that allows users to filter results based on attributes such as genre, release date, and popularity.

Queries shall return relevant results, and the system shall display results in a user-friendly manner.

System Feature 5: Playlist Management

Entities: User, Playlist, Song

Users shall be able to access their playlists.

Users shall also be able to search in their playlists and look for additional information.

Playlists shall support custom names, descriptions, and privacy settings (public or private).

System Feature 6: Artist and Album Exploration

Entities: User, Artist, Album

Users shall be able to view detailed information about artists and albums, including biographies, top tracks, and related artists.

System Feature 7: User Account Management

Entities: User, Account

Users shall be able to update their personal information, including profile picture, display name, and password.

System Feature 8: Follow/Favorite

Entities: User, Artist (Follows relation)

Users shall be able view their followed artists, albums, and other users.

They will also be able to check the people who follow them.

Users shall have a "favorites" list where they can check the songs they have marked as favorites.

System Feature 9: Data Synchronization

Entities: User, Spotify API, Database (Song, Playlist, Artist, Album)

The system shall periodically synchronize user data with the Spotify API to ensure it remains up to date.

Users shall have the option to manually trigger data synchronization when needed.

System Feature 10: Reporting and Analytics

Entities: User, Spotify API, Database (Song, Playlist, Artist, Album)

The system shall collect user data to generate analytics reports, including user activity, popular songs, and user demographics.

Administrators shall have access to these reports to gain insights into user behavior.



Select a Page

Login/Register



User Authentication


Login

Register



Register

Email

Password



User ID

Date of Birth

Username

Register

Login

Email

samarpratapald27@gmail.com

Password

.....



Login

Login successful! Hello, samarofl

Select a Page

Login/Register |



Login/Register

Albums

Artists

Songs

Playlists

Admin

Admin Accounts

User Accounts



samarofl's Albums:

Album Name: Made In The A.M. (Deluxe Edition)

Release Date: 2015-11-13

Album Type: album

Artist Name: One Direction

Album Name: UTOPIA

Release Date: 2023-07-28

Album Type: album

Artist Name: Travis Scott

Create Album

Album ID

1

—

+

Album Name

kkk

Release Date

2023/11/28

Album Type

single

Artist Name

j jonas

Press Enter to apply

Create Album

Delete Album

Select Album to Delete

Made In The A.M. (Deluxe Edition) (ID: 1) ▼

Delete Album

samarofl's Artists:

Artist ID: 1

Artist Name: Bruno Mars

Artist ID: 2

Artist Name: Weston Estate

samarofl's Songs:

Track Name: Show Me How

Artist: 6

Album Name: 1

Track Duration (ms): 215110

Popularity: 85

Track Name: Tree Among Shrubs

samarofl's Playlists:

Playlist ID: 1

Playlist Name: a little bit colder

Owner: samarofl

Total Tracks: 35

Update Name for Playlist 1

Update Playlist 1

Select a Page

Admin Accounts



You don't have permission to access this page.

Login History

Select Date

2023/11/28

Date: 2023-11-28

Total Logins: 1

Login

Email

admin2@pes.edu

Password

admin2



Login

Login successful! Hello, admin2

Admin View: Admin Accounts

List of Admin Usernames:

Admin Username: admin2

Admin View: User Accounts

List of Usernames:

Username: samarofl

Username: divyanshu

Username: sajalJ

Username: pipo

```

mysql> use spotify;
Database changed
mysql> SELECT ua.UserID, ua.UserName, ua.UserRole
-> FROM useraccount ua
-> WHERE ua.UserID IN (
->     SELECT f.UserID
->     FROM follows f
->     WHERE f.ArtistID IN (
->         SELECT ar.ArtistID
->         FROM album al
->         JOIN artist ar ON al.ArtistName = ar.ArtistName
->         WHERE al.ReleaseDate >= '2000-01-01'
->     )
-> );

```

UserID	UserName	UserRole
1	Divyanshu	user
2	Samar Pratap	user

```

2 rows in set (0.07 sec)

mysql> |

```

```
mysql> SELECT s.SongID, s.SongName, s.ArtistName, s.Popularity, a.ArtistID
-> FROM song s
-> JOIN artist a ON s.ArtistName = a.ArtistName;
```

SongID	SongName	ArtistName	Popularity	ArtistID
16	Nancy Mulligan	Ed Sheeran	67	19
25	Sixty	Weston Estate	54	22
44	Fighting Back	John Paesano	48	45
46	The Great Hunter	John Paesano	55	45

4 rows in set (0.00 sec)


```
mysql> SELECT * FROM song WHERE popularity = (SELECT MAX(popularity) FROM song WHERE UserID = 2);
+-----+-----+-----+-----+-----+-----+
| SongID | Popularity | SongName | SongLength | UserID | ArtistName |
+-----+-----+-----+-----+-----+-----+
| 24 | 90 | FE!N (feat. Playboi Carti) | 191700 | 2 | Travis Scott, Playboi Carti |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)
```