Stupefy
Database Design Document (DDD)

Version 1.0

Prepared by:

Legaspi, Nicole Ann

Loreto, Christopher

Nofre, Jasmine

Tiquio, Anilov

Viñas, Gabriel Angelo

COM232

Revision History

Date	Version	Description	Author

Table of Contents

1 Introduction

The section introduces the Database Design Document (DDD) for Stupefy to its readers.

1.1 Document Objectives

This DDD for the Stupefy software has the following objectives:

- **Describe the database design –** Present a comprehensive structure of Stupefy database entailing its entities, relationships between them, and constraints of the data.
- **Define data storage and management –** State how a MySQL relational database stores and acquires the user data, playlists, songs, artists, and albums.
- **Serve as a reference for implementation –** Be a guideline with which developers and database administrators implement and manage the system after development.

1.2 Intended Audiences

This DDD is intended for the following audiences:

- Technical reviewers, Supervisor and Stupefy staff who must evaluate the quality, functionality, and user experience of the application as outlined in this document.
- Stupefy developers including:
- Architects, whose overall system architecture (including backend infrastructure and data flow) must meet the requirements and scalability expectations specified in this document.
- Designers, whose user interface and user experience designs must adhere to the branding, accessibility, and functional specifications set forth in this document.
- Programmers, whose implementation of features (including playlist management, music streaming, and integration with external services) must align with the technical requirements described in this document.
- Testers, whose test cases and quality assurance processes must validate the correct functionality of features such as search, playback, and account management as described in this document.

1.3 References

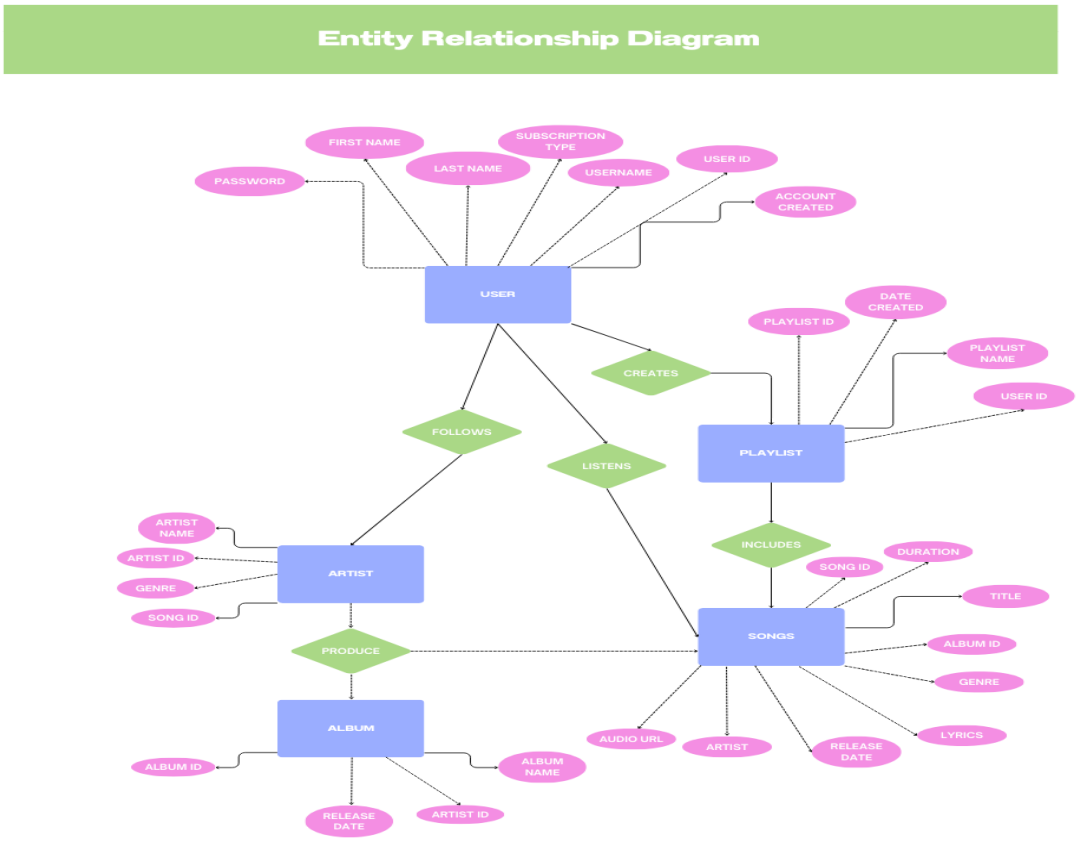
This DDD refers to the following references:

- Software requirement specification: SRS_Stupefy.docx
- Project Proposal: Project_Proposal_SS173D_V1.docx

2 Detailed Database Design

This section describes the actual design of different databases at varying levels of abstraction. A subsection for each of conceptual, internal, logical and physical levels.

2.1.1 Entity Relationship Diagram



2.1.1.1 Data dictionary for Element: User

Name	Data Type	Constrain	Description
User ID (primary key)	String	Min :1, Max:1	ID of the user
First Name	String		First name of the user
Last Name	String		Last name of the user
Password	Integer		The password of the user

Username	String		Username of the user
Subscription Type (FK)	String		The type of subscription the user has
Account Created	Integer		Joined date of the user

2.1.1.2 Data dictionary for Element: Artist

Name	Data Type	Constrain	Description
Artist ID (primary key)	Integer	Min :1, Max:1	ID to identify the Artist
Artist Name	String		Name of the Artist
Genre	String		Genre of the Artist
Song ID	Integer		ID to identify the song

2.1.1.3 Data dictionary for Element: Album

Name	Data Type	Constrain	Description
Album ID (primary key)	string	Min :1, Max:1	ID to identify the album
Album Name	string		Name of the playlist
Artist ID (FK)	Integer		ID to identify the artist

Release Date	string		When the playlist was created

2.1.1.4 Data dictionary for Element: Song

Name	Data Type	Constrain	Description
Song ID (primary key)	Integer	Min :1, Max:1	ID to identify the song
Song Title	String		Name of the song
Album ID (FK)	Integer		ID to identify the album
Song Genre	String		Genre of the song
Duration	Integer		Duration of the song
Lyrics	String		Lyrics of the song
Artist	String		Artist of the song
Release Date	Integer		Release date of the song
Audio URL	String		URL of the audio

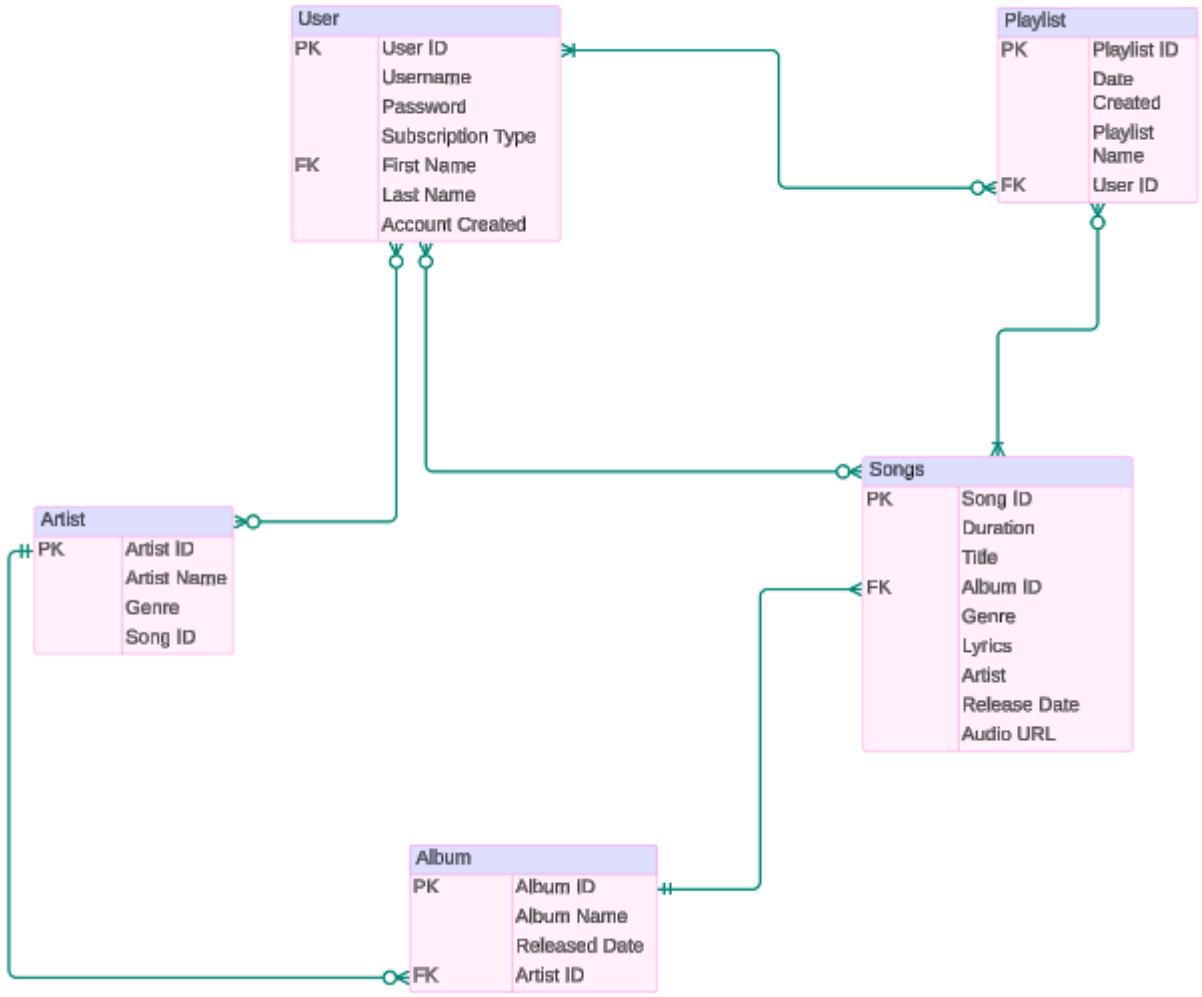
2.1.1.5 Data dictionary for Element: Playlist

Name	Data Type	Constrain	Description
Playlist ID (primary key)	string	Min :1, Max:1	ID to identify the playlist
Playlist Name	string		Name of the playlist
User ID	Integer		ID to identify the owner of the playlist

Date created	string		When the playlist was created

2.2 MySQL database design (Relational database)

2.2.1 Conceptual diagram



2.2.2 Description

This diagram displays the conceptual model of the MySQL database. This database called Stupefy is designed as a music catalogue platform. Users can create accounts, manage playlists, and play music, while admins have the authority to

manage application content and oversee the activity of a user. Users have the ability to make numerous playlists, each with a selection of songs. Albums are associated with the artists who release them, and each song is part of a particular album. A user may have several playlists, a playlist may include multiple songs, a song may be part of a single album, and an album may be linked to a single artist, all of which are enforced by the database. The streaming experience is made effective by this database format, which keeps music data accessible yet secure. It also guarantees data security by storing user information, such as passwords, securely and maintaining appropriate connections between users, playlists, and songs to prevent unwanted access.

2.2.3 Purpose of Tables

2.2.3.1 Purpose of User Table

This table stores user details and information, including username, password, first name, last name, subscription type, and joined date for those currently registered in the system. Each user will have a username and password required for log in. The primary key will be the user ID, ensuring that each record uniquely belongs to a single user in the database.

2.2.3.2 Purpose of Artist Table

This table stores artist's information, such as artist name, genre, albums, and songs. Each artist will have a unique artist ID as the primary key for identification. This ensures that each artist holds a unique identification in the database.

2.2.3.3 Purpose of Album Table

This table stores album information, such as album name, release date, and genre. Each album will have a unique album ID as the primary key for identification. The table will also include an artist ID as a foreign key, ensuring that each album is linked to its respective artist.

2.2.3.4 Purpose of Song Table

This table stores song information, such as song title, duration, release date, and genre. Each song will have a unique song ID as the primary key. The table will also include an album ID as a foreign key, ensuring that each song is linked to its respective album.

2.2.3.5 Purpose of Playlist Table

This table stores playlist information, such as playlist name, creation date, and songs. Each playlist will have a unique playlist ID as the primary key. The table will also include a user ID as a foreign key, ensuring that each playlist is linked to its respective user.

2.2.4 Relations

From Table	To Table	Relation

User	Artist	A user may follow an artist
User	Album	A user may view an album
User	Song	A user may view a song
User	Playlist	A user may add a song to a playlist
Artist	Album	An artist may release an album
Album	Song	An album may contain several songs
Song	Playlist	A user may combine multiple songs in a single playlist

3 References

[1] Free design tool: Presentations, video, social media | CANVA. (n.d.). https://www.canva.com/

[2] *Draw.io - free flowchart maker and diagrams online*. Flowchart Maker & Online Diagram Software. (n.d.). https://app.diagrams.net/

[3] *Er diagram_group2: Lucidchart*. ER Diagram_Group2 | Lucidchart. (n.d.). https://lucid.app/lucidchart/aa44cb0d-ec24-46cd-9a77-d9c616786ca4/edit?page=0_0&invitationId=inv_19f8d8fc-d6d0-4127-a913-740f45ca5e55

4 Appendix 1 – XML Schema

This XML schema was created to check if the schema was well formed.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>Stupefy</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>Stupefy</name>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <junit.version>5.10.2</junit.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.18</version> <!-- Ensure the version matches your MySQL setup -->
    </dependency>



    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-controls</artifactId>
      <version>21.0.2</version>
    </dependency>

    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-controls</artifactId>
```

```
    <version>21.0.2</version>
  </dependency>

  <dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-fxml</artifactId>
    <version>17.0.6</version>
  </dependency>

  <!-- Ikonli JavaFX Support -->
  <dependency>
    <groupId>org.kordamp.ikonli</groupId>
    <artifactId>ikonli-javafx</artifactId>
    <version>12.3.1</version>
  </dependency>

  <!-- Ikonli FontAwesome5 Pack -->
  <dependency>
    <groupId>org.kordamp.ikonli</groupId>
    <artifactId>ikonli-fontawesome5-pack</artifactId>
    <version>12.3.1</version>
  </dependency>

  <dependency>
    <groupId>com.dlsc.formsfx</groupId>
    <artifactId>formsfx-core</artifactId>
    <version>11.6.0</version>
    <exclusions>
      <exclusion>
        <groupId>org.openjfx</groupId>
        <artifactId>*</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>${junit.version}</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>${junit.version}</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

```
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.13.0</version>
        <configuration>
          <source>23</source>
          <target>23</target>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.openjfx</groupId>
        <artifactId>javafx-maven-plugin</artifactId>
        <version>0.0.8</version>
        <executions>
          <execution>
            <!-- Default configuration for running with: mvn clean javafx:run -->
            <id>default-cli</id>
            <configuration>
              <mainClass>com.example.stupefy/com.example.stupefy.HelloApplication</mainClass>
              <launcher>app</launcher>
              <jlinkZipName>app</jlinkZipName>
              <jlinkImageName>app</jlinkImageName>
              <noManPages>true</noManPages>
              <stripDebug>true</stripDebug>
              <noHeaderFiles>true</noHeaderFiles>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```