SRS DOCUMENT

Table of Contents

1. INTRODUCTION

1.1 PURPOSE

The purpose of this document is to build an online system to manage songs and music to ease the song management.

1.2 DOCUMENT CONVENTIONS

This document uses the following conventions.

DB	Database

ER	Entity Relationship
PHP	PHP language

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This project is a prototype for the music management system and it is restricted within the college premises. This has been implemented by the students of college. This project is useful for song listeners and as well as the artists who post songs.

1.4 PROJECT SCOPE

The purpose of the online music management system is to ease music management and to create a convenient and easy-to-use application for users, trying to listen to songs. The system is based on a relational database with its music management and uploading functions. We will have a database server supporting major artists around the world as well as users. Above all, we hope to provide a comfortable user experience.

2. OVERALL DESCRIPTION

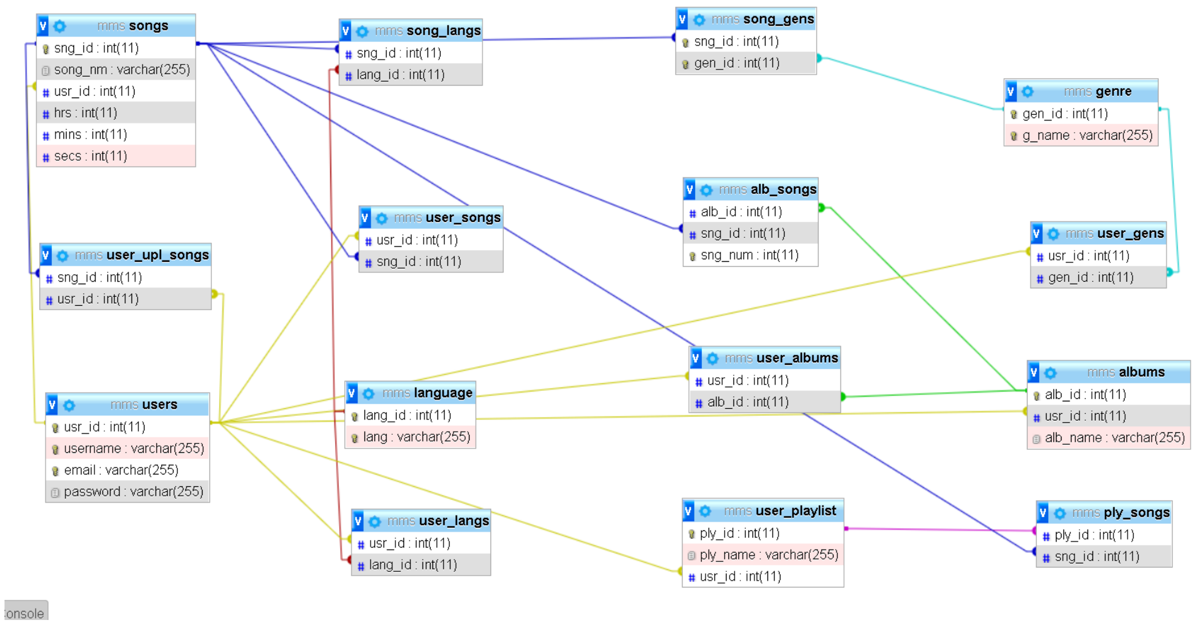
2.1 PRODUCT PERSPECTIVE

A distributed music database system stores the following information.

- Users
- Songs
- Albums
- Playlists
- Preferred
- Genres
- Languages

2.2 PRODUCT FEATURES

The major features of music database system as shown in below entity–relationship model (ER model)



2.3 USER CLASS and CHARACTERISTICS

Users of the system should be able to like songs, album. Have preferred language or Genre, create playlists of songs, upload songs or albums. The user can change their password. Users can see the songs and albums of other users.

2.4 OPERATING ENVIRONMENT

Operating environment for the music management system is as listed below.

- Database
- Client/Server system
- Operating system: Windows.
- database: SQL
- Platform: PHP

2.5 DESIGN and IMPLEMENTATION CONSTRAINTS

- The global schema, fragmentation schema, and allocation schema.
- SQL commands for above queries/applications
- How the response for application 1 and 2 will be generated. Assuming these are global queries. Explain how various fragments will be combined to do so.
- Implement the database at least using a centralized database management system.

3. SYSTEM FEATURES

DESCRIPTION and PRIORITY

The music management system maintains information on songs, albums, personal preferences etc.

FUNCTIONAL REQUIREMENTS

DATABASE:

The application has a database that stores all the data of the users, songs, albums, playlists etc. This main database is hosted on a server and is accessed via the client server architecture.

CLIENT/SERVER SYSTEM:

The term client/server refers primarily to an architecture or logical division of responsibilities, the client is the application (also known as the front-end), and the server is the DBMS (also known as the back-end).

A client/server system is a distributed system in which, some sites are client sites and others are server sites.
All the data resides at the server sites.
All applications execute at the client sites.

4. EXTERNAL INTERFACE REQUIREMENTS

4.1 USER INTERFACES

- Front-end software: HTML, CSS, Bootstrap
- Back-end software: SQL, PHP

4.2 HARDWARE INTERFACES

- Windows.
- A browser which supports CGI, HTML & JavaScript

4.3 SOFTWARE INTERFACES

Following are the software used for the music management online application.

- Operating system: We have chosen Windows operating system for its best support and user-friendliness.
- Database: To save the flight records, passengers records we have chosen SQL database.
- PHP: To implement the project we have chosen PHP language for its more interactive support.

4.4 COMMUNICATION INTERFACES

This project supports all types of web browsers. We are using simple web technologies for the login-registration, adding songs etc.

5. NONFUNCTIONAL REQUIREMENTS

5.1 PERFORMANCE REQUIREMENTS

The steps involved to perform the implementation of the music database are as listed below.

A) E-R DIAGRAM

The E-R Diagram constitutes a technique for representing the logical structure of a database in a pictorial manner. This analysis is then used to organize data as a relation, normalizing relation and finally obtaining a relation database.

- ENTITIES: Which specify distinct real-world items in an application.
- PROPERTIES/ATTRIBUTES: Which specify properties of an entity and relationships.
- RELATIONSHIPS: Which connect entities and represent meaningful dependencies between them.

B) NORMALIZATION:

The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and increase in the total size of the data stored.

If a database is not properly designed it can give rise to modification anomalies. Modification anomalies arise when data is added to, changed or deleted from a database table. Similarly, in traditional databases as well as improperly designed relational databases, data redundancy can be a problem. These can be eliminated by normalizing a database.

Normalization is the process of breaking down a table into smaller tables. So that each table deals with a single theme. There are three different kinds of modifications of anomalies and formulated the first, second and third normal forms (3NF) is considered sufficient for most practical purposes. It should be considered only after a thorough analysis and complete understanding of its implications.

5.2 SAFETY REQUIREMENTS

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

5.3 SECURITY REQUIREMENTS

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

5.4 SOFTWARE QUALITY ATTRIBUTES

- CORRECTNESS: The data uploaded must not conflict with the data fetched.
- MAINTAINABILITY: The administrators of the database make sure that there are no anomalies.
- USABILITY: The data of the songs is fetched and displayed and operations like adding to favourites etc can be performed.