Software Requirements Specification

for

Smart Music Player

Prepared by

Sakshi Agrawal(1812845)

Sakshi Gupta(1812846)

Shivangi(1812864)

Shreya Jain(1812869)

GROUP ID- CS 32

Banasthali Vidyapith

21-Feb-2021

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to provide a debriefed view of requirements and specifications of the project called "*Smart Music Player*".

The goal is to present a better listening experience to the user where shuffling songs is no longer random but will favor those songs the user is less likely to skip and more likely to listen through to the end.

This document discusses about whole system from backend to user interactions.

1.2 Document Conventions

- All terms in italic style.
- Main features, headings and important terms are in bold style.
- DB->Database
- ER->Entity Relationship
- TBD->"To be Decided", these are the components that are not yet decided

1.3 Intended Audience and Reading Suggestions

- Anyone with some basic knowledge of programming can understand this document. The document is intended for Developers, Software architects, Testers, Project managers and Documentation Writers. But anyone with programming background and some experience with UML can understand this document.
- This Software Requirement Specification also includes:
    - Overall description of the product
    - External interface requirements
    - System Features
    - Other non functional requirements

1.4 Product Scope

"Smart Music Player**"** is a *web-based application*. The system will be composed of server-side components and client-side components. The server-side component will manage the database operations and the client-side components will be graphical interfaces.

The project's main aim is to provide organized music library and recommendations to the user. It will provide users songs which they may like, based on the songs that they previously listened, liked /disliked or on the basis of their preferred language, artists or genre. It would be beneficial for the user as they may find songs that they may like without consuming much time. It can also play music from local music library. There is no specific audience for this software. The application is accessible by anyone who creates a profile on the system. The application is designed to meet the following needs of the users as described below:

1. Creating an account or signing up, signing in, signing out/logging out.
2. New users choose the languages, artists, genres they like.
3. Basic features of music player like play, pause, play next, play previous, loop, shuffle, seekbar, stop, adjust volume, length of the song , add to playlist in front of each song.
4. A user can create custom playlists/delete playlist.
5. History of songs played.
6. Queue to play songs.
7. Like/dislike a song.
8. Show new songs added to the server.
9. Filters for searching songs (like album, artist, etc.)
10. Display all local music and play them.
11. Recommend a user new music based on his preferred languages, artists, genres, or on the basis of his likes/dislikes or his previously played music.
12. Display trending songs based on users play count.
13. Allows all the functionalities while music plays in the application.

1.5 References

- https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc
- https://www.reqview.com/doc/iso-iec-ieee-29148-srs-example
- https://www.geeksforgeeks.org/software-requirement-specification-srs-format/
- Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave Macmillan.

1.6 Terminologies

Term	**Description**
User	Any living being who is interacting with the software is a *user*.
System	The package of all the components which takes input and gives output to demonstrate the features of the software is called System.
Local Storage	This is the collection of songs already present in the user's system.
Database	The set of data which describes and gives information about the sound track.
Recommendation System	A system which takes a track as input and outputs set of tracks closely related to the input.
SQL	The SQL queries are used to manage the database and also for the recommendation of songs.

2. Overall Description

2.1 Product Perspective:-

This system consists of four components packaged as one desktop application:

- Music player: for playing music available online and from local library.
- Recommendation: On the basis of present track playing, like/dislike, preferred language/artist/ genre, it will generate suggestions.
- Database updation: It updates all the tracks in library with their metadata tags. This is done using already available online database.
- Local Store: Users can access their local songs and play them in our web application.

With music player user can play/pause/stop/like/shuffle a track etc. It is a fully functional music player.

Recommendation system on the basis of like/dislike/currently playing song/year-wise and initial inputs it suggests similar tracks which may be already available on the Internet.

Database updater takes a sound track or list of sound tracks as input and updates their metadata information according to information available in database.

Internet is needed for functioning of this application.

2.2 Product Functions

Using this application, user can play tracks available in offline/online library. While playing music user can get a list of suggested tracks which are most closely related with the present track in terms of their metadata tags like singer, genre, release year, like/dislike etc.

User can perform following actions:

- Create profile
- Login/ logout/change password
- profile updation
- play/pause/stop/like/dislike/shuffle/previous/next/mute/song information/history/loop, control volume
- add tracks/ remove tracks from playlist
- search songs
- user manual (help)
- update database
- get recommendation
- OTP generation for verification/authentication of users if user forgets his password
- create/update/delete/view playlist
- view trending songs
- view recommended songs
- view history

2.3 User Cases and Characteristics

@Admin

@Users

Admin:

They can fix the bug.

They can create new interface, like adding new updates in database.

Users:

SignIn/SingnUp/SignOut

Changing Passwords

Play local songs

Adding songs to the playlists/Creating new playlist

Searching of songs

Removal of songs from playlist

Removing a playlist

Like/Dislike the song

Update user profile

View History

2.4 Operating Environment

Windows 7 and newer versions (TBD)

2.5 Design and Implementation Constraints

- It is a *Windows standalone application* implemented using *JAVA.*

- Database Normalization: Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divide larger tables into smaller tables and

links them using relationships. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

- SQL constraints: SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

- Username and password are used for the identification of users.

- Authentication using One Time Password if a user forgets his password.

- Admin needs to update data timely.

2.6 User Documentation

- There is a user manual that lists all the features available for the user and methods to access them. "Help" option will be available in user interface which will help understand all the features of the application.
- User need to have a basic knowledge on using a website/interacting with a website.

2.7 Assumptions and dependencies

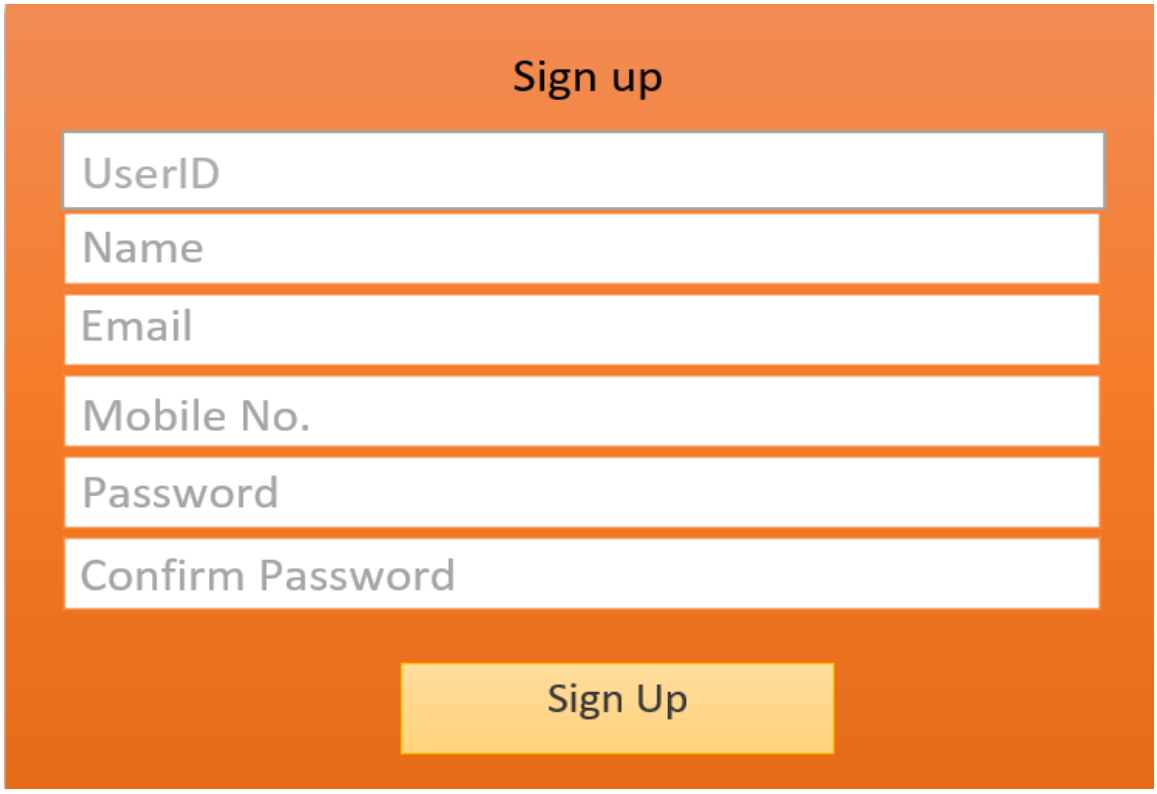
TBD

3. External Interface Requirements

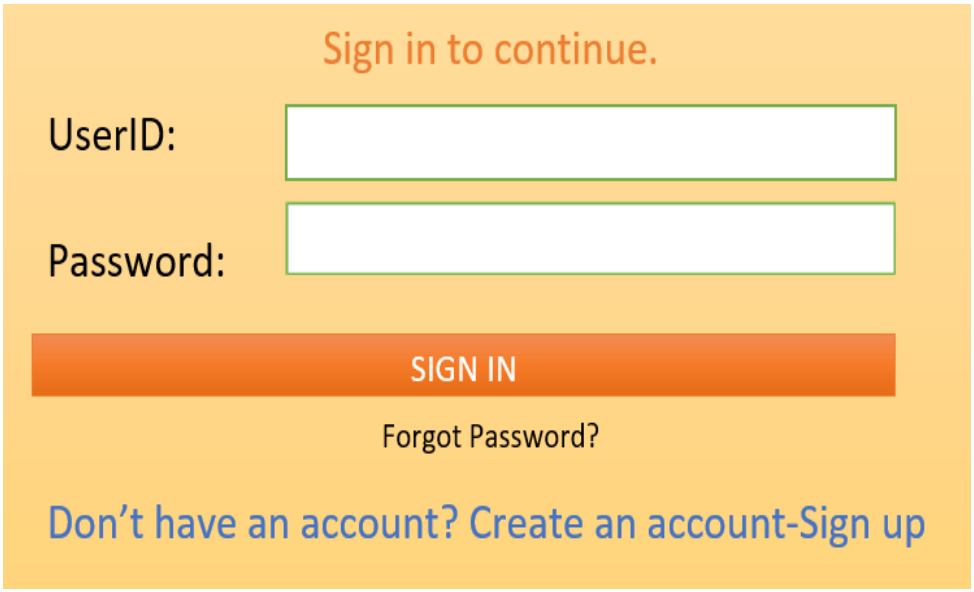
3.1 User Interfaces

User interface is implemented in *Java*.

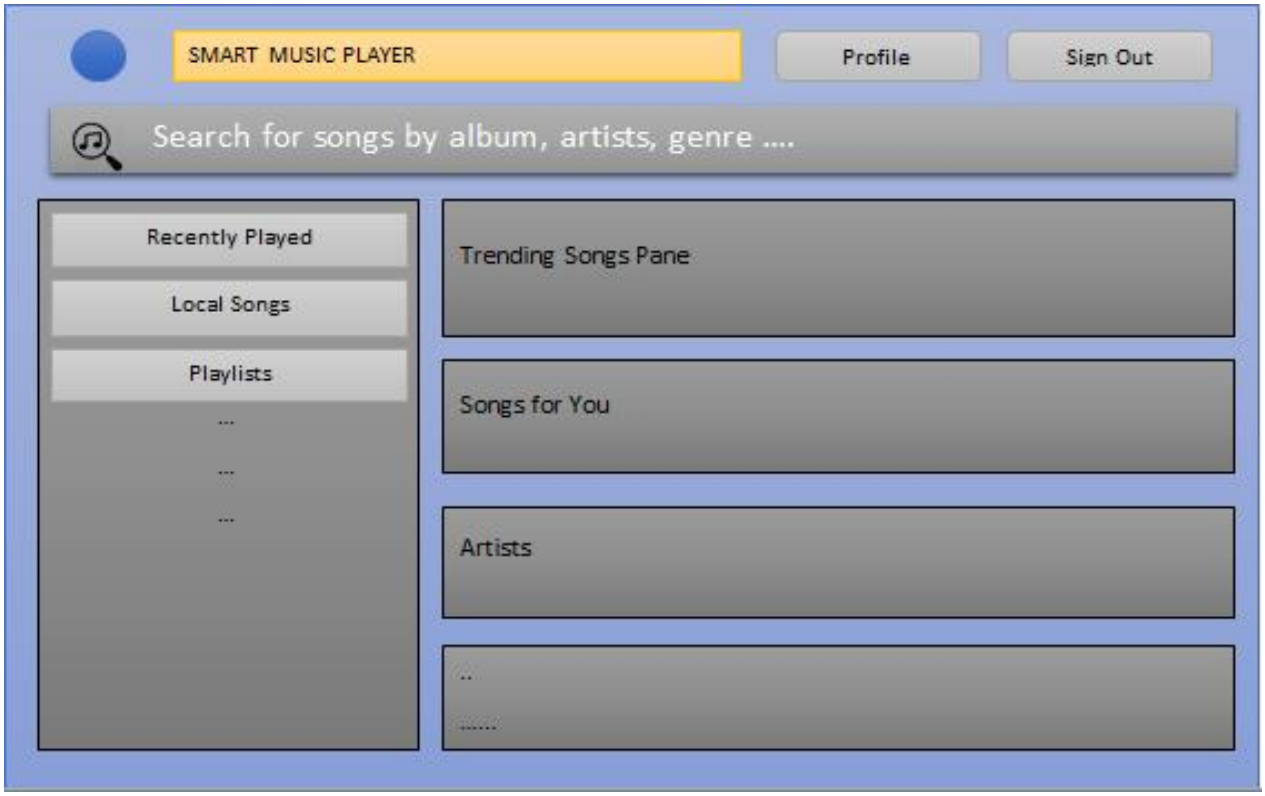
1. CREATE AN ACCOUNT



2. LOGIN



The front page which interacts with user. It is divided into frames for different functions.



3.2 Hardware Interfaces

- Input device is needed for user to interact with system.
- Software needs a display device to interact with user.
- Music player needs playback device for sound output.
- User needs a mobile phone for OTP verification.
- Internet connection is needed to run the application.

3.3 Software Interfaces

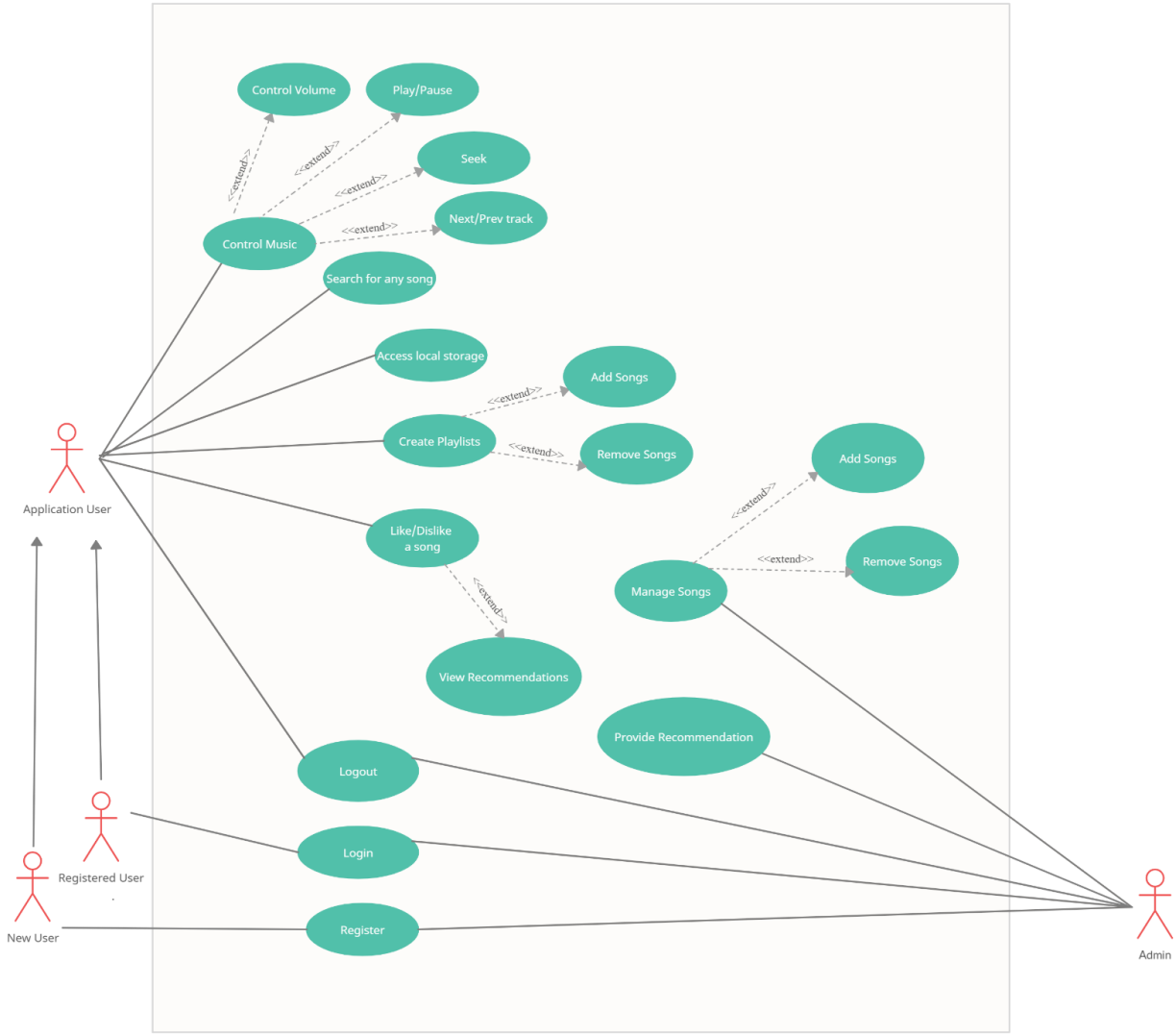
TBD

3.4 Communications Interfaces

- The Internet connection is use to communicate with database.
- All Network Communications will use HTTPS/TLS. It is used for secure communication over a computer network, and is widely used on the Internet. In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS) or, formerly, Secure Sockets Layer (SSL).
- Importing packages from javafx/swing to our website.

4. System Features

Following is the use case diagram for the application.



Functional Requirements

New users must register on the website with their mobile phone and user id. Then they can login and listen to their favourite music.

1. Manage Songs-Admin manages songs by adding the latest songs into the database and removing songs if not required.
2. Control Music-Users can play, pause, seek, stop, control volume, move to next/previous song etc.
3. Music Information-Users can view all the information for the current song.
   Specifically, the window album cover art shows the image of the cover of album and shows the following information:

- title
- artist
- movie name
- album
- genre
- year

4. Search any song-Users can search any song based on artists, albums, genres, etc.
5. Access local storage-Users can access songs on their local device.
6. Create Playlists-Users can create personalized playlists.
7. Like/dislike a song-Users can like or dislike a song based on their taste to get recommendation.
8. Provide recommendation-Admin provides recommendation of songs to the users based on their taste, on the basis of currently listening songs, or on the basis of their preferred language, artists and genre.
9. View recommendation-Users can view recommended songs. After listening music, the user will be able to view music recommendations selected using by their past actions and preferences.
10. View history-Users can view the recently listened songs.

5. Other Nonfunctional Requirements

The non-functional requirements of the system are explained below.

5.1 Performance Requirements:

- Quickness - System should be fast enough to play music and respond to any of the user action in any way without any shattering or buffering, else it will be not be a good experience.
- Robustness - System should be robust to deal and act accordingly with common error scenarios like unavailable metadata, unsupported file types.
- Failure Handling - In case of failures it should be able to handle failures and recover quickly.

5.2 Safety Requirements:

- Exception Handling - The software should be able to restrict or warn (in the first place) the user from doing things not suitable, like, increasing volume beyond threshold, or exiting the software without saving the changed data.

5.3 Security Requirements:

- Encrypted Connection - Connection between user and servers should be Encrypted (HTTPS/TLS).

- Verification – If the user forgets the password, an OTP is sent on the registered mobile number.

5.4 Software Quality Attributes:

- Memory Management - System should not leak memory.
- Compatibility - System should peacefully co-exist with other software
- Error Handling - System should not cause or trigger any events that will leave Operating System in unrecoverable state

6. Other Requirements

TBD