

**Software Requirement Specification (SRS)**

**Document**

**JUKEBOX**

**Sprint-1**

Project Timeline: 07-12-2022 to 12-12-2022

High Level Design & Low Level Design

The purpose of this document is to provide with a template for documenting both HLD & LLD

**Document Control :**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Revision History** | | | | | | | | |
|  |  |  | |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | | **Approver Signature** | |
| 06-Dec-2022 | 0.1 | Rahul Tarkunde | Start | | | |  | |
| 06-Dec-2022 | 0.2 | Rutuja Taware | Added Acronyms | | | |  | |
| 06-Dec-2022 | 0.3 | Komal Bhosale | Corrected spellings and approved. | | | | Kanchan | |

[**1. I**ntroduction 5](#_Toc368912248)

[1.1. Intended Audience 5](#_Toc368912249)

[1.2. Acronyms/Abbreviations 5](#_Toc368912250)

[1.3. Project Purpose 5](#_Toc368912251)

[1.4. Key Project Objectives 5](#_Toc368912252)

[1.5. Project Scope and Limitation](#_Toc368912253) 6

[1.5.1. In Scope 6](#_Toc368912254)

[1.6. Functional Overview 6](#_Toc368912256)

[1.7. Assumptions, Dependencies 6](#_Toc368912257)

[2. **D**esign **O**verview 7](#_Toc368912259)

[2.1. Design Objectives 7](#_Toc368912260)

[2.1.1. Recommended Architecture 7](#_Toc368912261)

[2.2. Design Alternative 8](#_Toc368912262)

[2.2.1. User Interface Paradigms 8](#_Toc368912266)

[2.2.2. System Interface Paradigms 8](#_Toc368912267)

[2.2.3. Error Detection / Exceptional Handling 8](#_Toc368912268)

[2.2.4. Performance 8](#_Toc368912270)

[**3**. **S**ystem **A**rchitecture 9](#_Toc368912274)

[3.2. SEQUENCE FLOW Diagram.](#_Toc368912275) 9

[**4**. **D**ata **F**low **D**iagram 10](#_Toc368912281)

5. Environment Description………………………………………………………… 11

5.2[.language support 11](#_Toc368912302)

[5.3.User Desktop Requirements 11](#_Toc368912303)

[5 .4. Deployment Considerations 11](#_Toc368912304)

[5.4.1. Database Server Disk Space 12](#_Toc368912305)

[5.4.2. Integration Requirements 12](#_Toc368912306)

[5.4.3. Jobs 12](#_Toc368912307)

[5.4.4. Network 12](#_Toc368912308)

[5.5. Configuration 12](#_Toc368912312)

[5.5.1. Operating System 12](#_Toc368912313)

[6. References 12](#_Toc368912317)

[7. Appendix 3](#_Toc368912318)

# 

# Introduction:

Jukebox is a simple mini project in C and efficient architecture that models the core functionality of modern-day music player software such as Windows Media Player, VLC Media Player and Google Play Music. It's just a demonstration of the use of file handling in c language. It has been **u**sed to jump from one menu to another within the program. Also, for editing and deleting the song.

This is capable of reading with locally stored song data. Designed purely in the C programming language and features are used in day – to day life. The Linux based envelope implementation that has been written to handle various levels of runtime exceptions.

## Intended Audience

This document is intended to be read by Team and Manager. This is simple structure program

User can modified it use and etc.

## Acronyms/Abbreviations

|  |  |
| --- | --- |
| client | User |
| struct | structure |

## Project Purpose

A jukebox music software program is one that makes well-known songs from a songlist. User easily access and modify functions. So user can use jukebox in everday life.

## Key Project Objectives

* 1. Allow **User** to add, delete, display features in backlog

## 1.5. Project Scope and Limitation

The music player allows a user to play various media file formats. It can be used to play mp4 files also. The music player is a software project supporting all known media files.

We're introducing jukebox, a neural net that generates music, including rudimentary singing, as raw audio in a variety of genres and artist styles. We're releasing the model weights and code, along with a tool to explore the generated samples.

## 1.5.1 In Scope

We're introducing jukebox, a neural net that generates music, including rudimentary singing, as raw audio in a variety of genres and artist styles. We're releasing the model weights and code, along with a tool to explore the generated samples.

## 1.6. Functional Overview

Function used in the code are -

* fopen() - Create a new file for use
* fclose() – Closes a file which has been opened for use
* fprint() – writes a set of data values to a file
* fscanf() - Reads a set of data values from a file
* getc() – Reads a character from a file
* putc() - writes a character to a file
* rewind() – Sets the position to the beginning of the file
* ferror() – Test for an error in reading from or writing to the given stream
* remove() – Used to delete the file
* rename() – Used to change the name of the file or directory
* feof() – Pointing the manage end of the file

## 1.7. Assumptions and Dependency

* The system should have any Distribution of Linux installed.
* The Proposed System is intended to work on Terminal GUI.
* The system should have either 8GB or more RAM.
* The service is used preferably on a desktop or laptop.

# 2.Design Overview

|  |  |
| --- | --- |
| Name of the Module | Import Songs |
| Handled by | Rutuja Taware |
| Description | This function is used to import  CSV file read and append mode. |

|  |  |
| --- | --- |
| Name of the Module | Display Song list |
| Handled by | Prajakta Shinde |
| Description | This function is used to display  songs from song list. |

|  |  |
| --- | --- |
| Name of the Module | Add song Playlist |
| Handled by | Sneha Lichade |
| Description | This function is used to add song  from songlist to playlist. |

|  |  |
| --- | --- |
| Name of the Module | Delete Song Playlist |
| Handled by | Gauri Bedarkar |
| Description | This function is used to remove  songs from playlist. |

|  |  |
| --- | --- |
| Name of the Module | Display Play List |
| Handled by | Komal Bhosale |
| Description | This function is used to display  the songs from playlist . |

## 2.1. Design Objectives

Primary :

In jukebox player user anywhere, anytime it can play the song, remove song,add new song etc.

### 2.1.1. Recommended Architecture

### In this option the architect is part of the Scrum team, and this is the most embedded way that an architect would fit into a Scrum project.

### 2.2. Design Alternative

We have used file i/o because in this project we implemented no of operations like add song, play it by using id and also it plays back form current state etc. file i/o is used for this purpose.

### 2.2.1. User Interface Paradigms

The jukebox gives access to user to listen music, add song, remove from list, delete it etc. The details are stored in file. After that user can view or show songlist.

### 2.2.2. Error Detection / Exceptional Handling

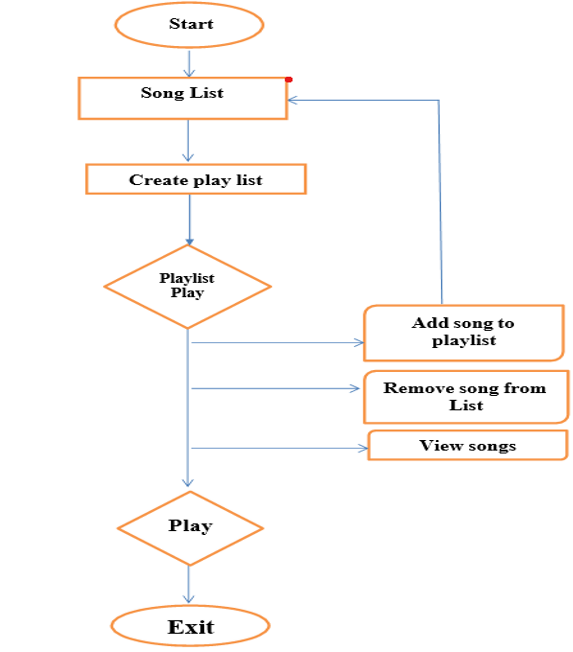
New users should register before login or else it displays the no user found. Registered users have to login with valid credentials. otherwise, they will get invalid username or password.

**2.2.3. Performance**

The system will work on the terminal. The performance depends on the hardware component of the user’s system.

**3.System Architecture**

3.1. SEQUENCDE FLOW DIAGRAM



**Step 1:**

When user start the Jukebox player, it will show the Song List . In Song List this list of songs are stored song list this list is in .csv file which having Track\_Id , Track\_Name, Artist\_ Name, Path.

# Step2:

Crete a Play list in this step user can create list of songs which he want to add into the list. Then new playlist is ready for use . this list of songs are stored song list this list is in .csv file which having Track\_Id , Track\_Name, Artist\_ Name, Path.

**Step3:**

**3.1.ADD:**

Load the playlist when user want to add some new song from the song library . that’s for user go song list select new song and add in to the play list.

# 3.2.REMOVE:

When the user want to remove any song from the play list . he select It and say to remove that from current list.

# 3.3.VIEW:

When user want to view all the song list from playlist .then he can do it by using show or view song from list.

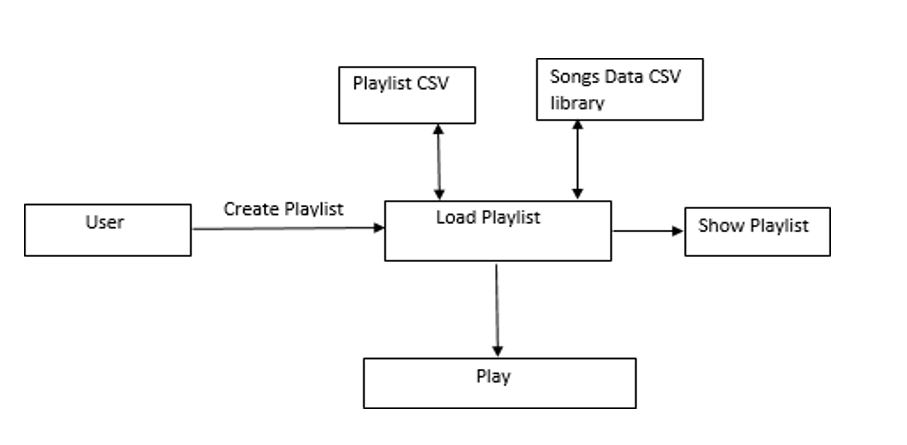
# PLAY:

Play song from playlist by using track\_id. when user exit from jukebox and again start it that time it can play from current song .also user change song bytrack\_id.

**Step5**:

Exit from the jukebox play list when he want to close or stop the jukebox that that time it will exit form it.

##### **4. DATA FLOW DIIAGRAM**



**5. Environment Description**

## 5.1.Time Zone Support

* **IST- Kolkata, IST-Mumbai**

## 5.2. Language Support

* **English**

## 5.3. User Desktop Requirements

* 64-bit processor, 1 GHz or faster
* At least 2 GB free hard drive space
* At least 1 GB RAM

### 5.4. Deployment Considerations

* Easy setup
* Local storage is used.
* No network latency to consider.
* To scale buys a bigger CPU, more memory, larger hard drive, or additional hardware.

### 5.4.1 Database Server Disk Space

No such disk space is required as the program is fully functional on online IDE(s) as well. The Local Operating System is required and one text file to store the records of processes.

### 5.4.2. Integration Requirements

No such disk space is required as the program is fully functional on online IDE(s) as well. The Local Operating System is required and one text file to store the records of processes.

**5.4.3. Jobs**

We can establish connections between users who are connected to the server. And we can search the song history of the user.

### 5.4.4.Network

* + **End to End**

# 5.5. Configuration

### 5.5.1. Operating System

* + **Linux environment**