# Software Requirements Specification Document (CS360)

# Project Jukebox


**Group Number: 19**

**Shahrukh Nawaz**
**Wahid Ejaz**
**Hamza Hassan**
**Mohammad Obaid Ur Rehman**
**Shaheryar Faisal**

**Course:** Software Engineering CS360

**Instructor:** Suleman Shahid

**University:** Lahore University of Management Sciences (LUMS)

**Version: 3.1**
**Date: (24/02/2019)**
**Number of hours spent on this document: 20 man hours**

# Table Of Contents

# 1 Introduction

## 1.1 Document Purpose

The purpose of the document is to collect and analyze all assorted ideas that have come up to define the “Project Jukebox System” and its requirements with respect to the intended audience. This document will illustrate the complete declaration for the development of the system, it’s function, usage, scope, purpose, ideas, primary features and secondary features (that we may implement later). It will also explain system constraints, interface and interactions which the software maybe subject to. This document is primarily intended for the instructor, teacher’s assistant and our clients for its approval and to act as a reference for developing the first version of the system for the development team.

## 1.2 Product Scope

The Jukebox a system which enables users to play music at any place they want for example at restaurants. The application would be free to download from the website, android play store or similar sources. Business owners and managers would be able to subscribe to our service so that they can run the server locally at their venue thus improving customer satisfaction by exposing customers to a more customizable experience. The incentive for the customer to go to a restaurant which supports jukebox would be to enjoy their food while listening to their favourite music.

## 1.3 Intended Audience and Document Overview

This document is intended for

- Project jukebox team: To mutually agree on the requirements of the system, its implementation and to help in developing strategies for improvement, scaling, testing and deployment.
- Instructor and teacher’s assistant: To understand the basic project idea, it’s functionality, scope and implementation.
- Secondary clients (Managers at restaurants): To understand the project ideas, the potential benefits (for example improved customer experience) and its features (for example security).

The remainder of the document features an overview of the product functionality followed by the specific requirements that Project Jukebox aims to fulfill.

## 1.4 Definitions, Acronyms and Abbreviations

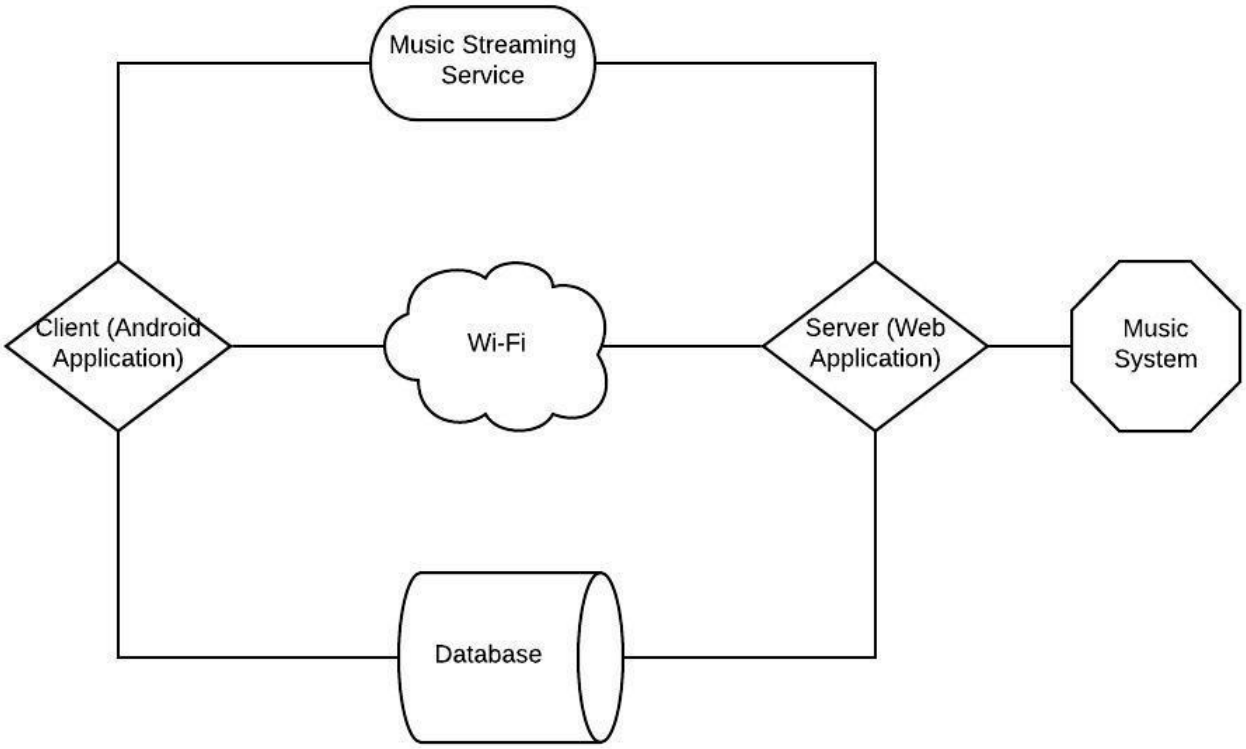
**API:** Application programming interface
**App:** android application
**DB:** database

## 1.5 References and Acknowledgments

*How to write a good SRS for your Project*
*https://www.geeksforgeeks.org/how-to-write-a-good-srs-for-your-project/*
*Software Requirements Specification document with example*
*https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database*
Software Requirements Specification Amazing Lunch Indicator
http://www.cse.chalmers.se/~feldt/courses/reqeng/examples/srs_example_2010_group2.pdf

# 2 Overall Description

## 2.1 Product Perspective




Our product is a new self-contained product which consists of 4 major parts. The first part consists of an **android based user applicatio**n which will allow users to access a server through login via a local Wi-Fi connection. The second consists of a **server-based application** which will allow users to access and make song requests through the local Wi-Fi system. The android application users upon connecting to the Wi-Fi service of the location will be notified that the application services are available at this location and will be prompted by the android application to connect to the local server if they wish to play a song. The server will maintain a local queue of the songs that are to be played next which will be made available to the users using the application. The third will be an **online music streaming service** that will allow the server to access this song database to fetch and play song requests that have been queued within the system. The database of songs will also be made available to the android application users so that they may search and request for songs that are available in the database. The fourth consists of a **user database** that will contain the user ID information for all users which will be used to verify user logins into different systems.

## 2.2 Product Functionality

1. Restaurant manager initiates the server and runs the server application on the local system.
2. A user downloads the android application and registers as a unique user in the online database.

3. The android application user signs into the local Wi-Fi where the system is being implemented.
4. Upon signing into the local wi-fi, the user is alerted via the application that the application and its services are available at this location.
5. The user then signs into the local application server via a login screen.
6. Upon logging in, the user is shown a list of the songs that have already been queued to play in the system and is given an option to add their own request into the system.
7. The user upon selecting the add request/song option is prompted to search for their song in the online streaming service database.
8. Upon finding the desired song from the list provided by the online streaming database, the user can select the “add song” or “play” option to have their request added to the end of the local song queue.
9. Once the song has been added to the queue, the user is unable to make additional requests to the server until their initial request has been played.

## 2.3 Users and Characteristics

The application broadly consists of 2 user categories.

The first are the ***managerial staff*** that will be using the system at the Restaurant/Cafe to initiate the server at their end and log into the user music streaming platform as a registered client which will allow them to stream music from the platform. Basic knowledge pertaining to average computer use and log in processes is required to fulfill this role.

The second category consists of the **android application user** base, which will download the application from the play store and use it in the pertinent locations. Basic knowledge regarding basic application use on android based devices, and user login processes should be sufficient to be able to make use of the application.

## 2.4 Assumptions and Dependencies

We are assuming that the local system implementing the queue has enough memory to accommodate for the maximum number of total users that it may have at a given time.

We are also using a third party (Spotify API) software-based API for our online music streaming service which we assume will be easily integrated into our system interface.

We also plan on integrating an already available login and user identification/authorization system which we again assume will be fully and easily integrated into our system.

We also assume that the client server that will be implementing our system and connecting to and streaming from our centralized streaming service has a stable and reliable enough internet connection to stream the music in a continuous and uninterrupted fashion. Other methods may be made available (at the client server side) such as the option to buffering if this is not the case.

# 3 Specific Requirements

## 3.1 Functional Requirements

### 3.1.1 Category: Application side

*Following are the list of requirements for the customer side of the system.*

**1-Notification**

- **Description:** The system will alert the user regarding the availability of the Jukebox application service at a location.
- **Input:** User logs into the local Wi-Fi system of the location.
- **Processing:** The application detects the presence of the application server through the local Wi-fi.
- **Output:** The user is alerted via a notification of the presence of the server and the application services at that location.

**2-Connect to Server**

- **Description:** The user should be able to log into or connect to the server where there is a server online.
- **Input:** The android application user provides the log in credentials in the form of a username and a password to log into and connect to the server/service.
- **Processing:** Log in authentication and processing.
- **Output:** The User is now logged into the system and can generate a request which will be added to the request queue for that server.

**3-Search Music**

- **Description:** The user should be able to search for a song from the database.
- **Input:** Song name query.
- **Processing:** Fetching song names from database.
- **Output:** List of songs that satisfy the users query.

**4-Request Music**

- **Description:** The user should be able to request music of his choice.
- **Input:** Song name selected from search results.
- **Processing:** Send song request to server.
- **Output:** Song plays on music system when it is at the front of the queue.

**5-Show Queue**

- **Description:** The user should be able to view the songs in the queue along with the size of the queue.
- **Input:** The user logs into the server.
- **Processing:** The app fetches the list of songs in the queue along with the number of songs in the list, which will be displayed along with an add request option on the app home screen.
- **Output:** The queue with the song names and the total number of songs in the queue.

### 3.1.2 Category: Server Side

*Following are the list of requirements for the server side of the system.*

**1- Run Server**

- **Description:** The user should be able to run a server by pressing a button.
- **Input:** Press “launch server” icon.
- **Processing:** Configure server, connect to internet.
- **Output:** The server comes online and allows android devices users to connect.

**2- Receive, Enqueue and Play Song Requests**

- **Description:** The server should accept song requests and add them to the queue.
- **Input:** Song request from the android application user.
- **Processing:** Fetch song id from database and add it to the queue.
- **Output:** The song should play when it is at the front of the queue.

**3- Display Queue**

- **Description:** The server should be able to view the queue of songs along with the size of the queue i.e. the list of songs waiting to be played.
- **Input:** User clicks the show queue button.
- **Processing:** Fetch the queue.
- **Output:** Display the queue of songs and the total number of songs in the queue.

**4- Show Available Users**

- **Description:** The server operator should be able to view the list of people connected to the server.
- **Input:** User clicks the show users button.
- **Processing:** The application loads the list of all users connected to the server.
- **Output:** Display the list of all the users connected to the server.

## 3.2 External Interface Requirements

### 3.2.1 User Interfaces

- Application User Interface

Upon opening the application, the user is prompted to log into the server for the local Wi-Fi system that the user is connected to via a login page consisting of username and a password field. For the sake of completeness, the name of the relevant server/restaurant or gym will be displayed on the login page as well. Upon entering the login credentials the user is directed to the main page for the application. The main page is minimalistic and consists of a field displaying the queue, current number of requests in the server queue as well as button to add a request to the server queue. The user can also click on the add request button, which will redirect the user to the search for song screen. The search for song interface consists of a search bar which the user can use to enter the name of a song and browse the results for the songs available on the music streaming service platform. Upon selecting a song, the user is redirected to the home screen where the user can view the position of his or her request in the queue. If the user has already added a song to the queue, there will be a mechanism in place that will lock the add request button and prevent the user from adding additional requests to the que until their first request has been played.

- Server User Interface

Upon opening the desktop application, the user is shown a login screen which allows them to log into the music streaming platform which will stream and play requests on their local music system. Upon logging in, the user is directed to the main home screen which displays the request queue for their local system and the current number of users logged into the system. There will

be an option to view the current users by username as well which will redirect the user to a screen showing a list of currently active users on the system.

### 3.2.2 Hardware Interfaces

**Desktop Computer:** A desktop computer with the server program needs to present at the venue, this computer would be responsible for running the server locally. It would also be connected to the music system so that song requests could be played locally.
**Android Device:** A mobile device with the android operating system would be required, on which the Jukebox application could be installed and run.
**Music System:** The music system will consist of a set of speakers. It will only be connected to the server which will play music using this system.
**Internet:** Both the devices (the android mobile device and the server computer) would need to be connected to each other via a common internet connection which will be provided through a router.

### 3.2.3 Software Interfaces

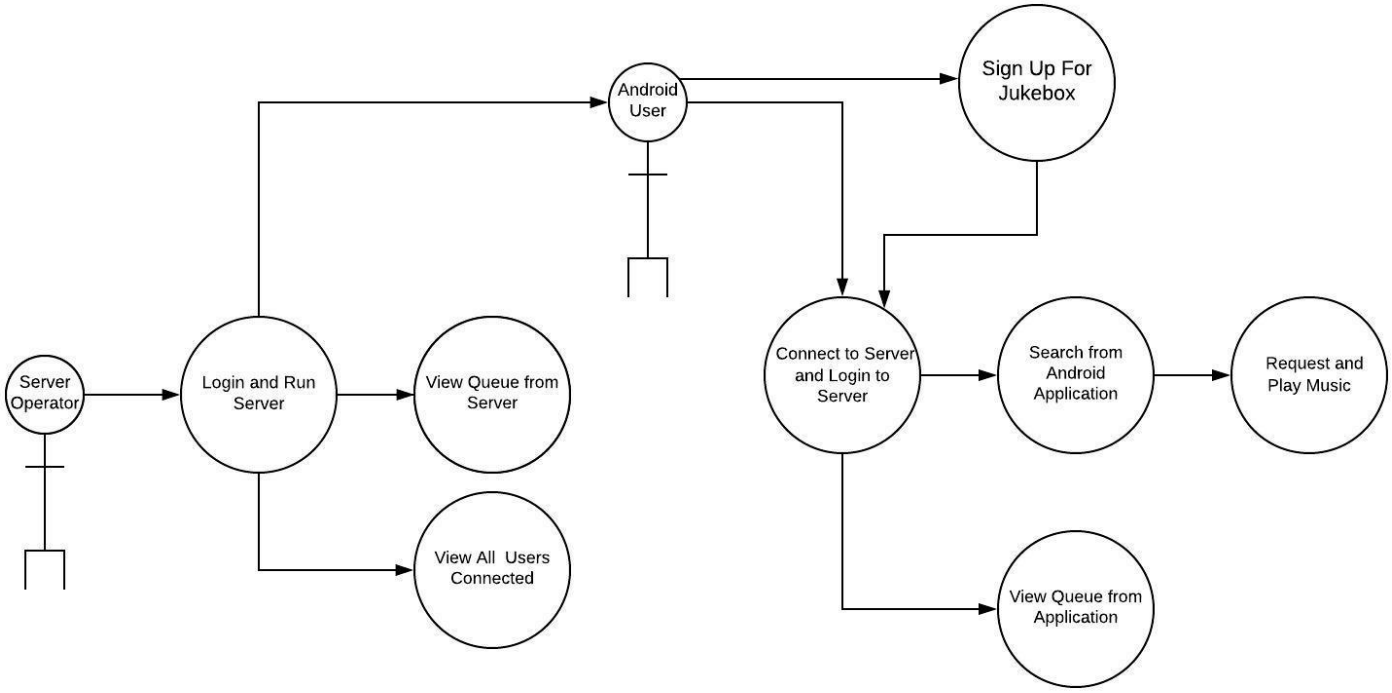
**Server:** The server will be a computer located at the venue responsible for processing user requests. This will be connected to a music system, music streaming service, database and to the application user via the internet.
**Database:** This database would be hosted using firebase or mongo-dB (depending on whichever is more feasible) and will contain login data to validate users when they sign in from the server desktop application and the user android application.
**Android Application:** The android application would be installed on mobile android devices which will be connected to the streaming service (to enable search), server (via the internet to send requests) and database (to authenticate users).
**Music Streaming Service:** The music streaming service will be connected to both the user and the server and will give them access to a database of songs. The user will mainly use this service to search for songs while the server will stream songs from this service.

## 3.3 Use Case View

### 3.3.1 Use Case Table

| Primary Actor | Associated Use Cases |
|---|---|
| Android Application Users (Customers at Restaurants) | 1. Connect to Server and Login to Server.<br>2. Search from Android Application.<br>3. View Queue from Application.<br>4. Request and Play Music. |
| Server Operators (e.g. Managers at Restaurants) | 1. Login and Run Server.<br>2. View Queue from Server.<br>3. Views all users connected to server. |

### 3.3.2 Use Case Diagram



### 3.3.3 Use Case Description

| Use Case 1: Login and Run Server |
|---|
| Actors:<br>• Time<br>• Server Operator (manager at restaurant) |
| Preconditions:<br>• The restaurant is about to open for the day.<br>• The computer responsible for hosting the server is connected to the internet.<br>• The device is also connected to music system. |
| Flow of events:<br>1. The use case starts when the manger opens the restaurant for business.<br>2. The manager opens the desktop application on the restaurant PC and presses the application icon.<br>3. The application launches and asks the operator for login details (username and password).<br>1. Upon incorrect entry of login credentials, the application gives an error message and asks the user to try again.<br>2. Upon successful entry the server comes online, and the operator can view the desktop application interface. |
| Postcondition:<br>• The server comes online and becomes available for other android application users to connect to. |

| Use Case 2: Sign Up for Jukebox |
|---|
| Actors:<br>• Android Application User |
| Preconditions:<br>• The mobile device has installed the application. |
| Flow of events:<br>1. The use case starts when the user opens his application.<br>2. The user can opt to sign up by pressing the sign up button.<br>3. The application then prompts the user for his details.<br>1. If the details entered are correct then he can login and use in the application.<br>2. If incorrect details are entered, then the user is allowed to re-enter the details. |
| Postcondition<br>• The Jukebox account has been created and the user can then login and use the service |

| Use Case 3: Connect to Server and Login to Server |
|---|
| Actors:<br>• Android Application User |
| Preconditions:<br>• The mobile device has installed the application.<br>• The user along with his mobile device is in proximity of the server hosted at the venue such that he can receive the signals broadcasted by the server. |
| Flow of events:<br>4. The use case starts when the user opens his application and realizes that a Jukebox server is close-by.<br>5. The user presses the connect button.<br>6. The application then prompts the user for his login credentials.<br>3. Upon successful authentication the user gains access to application.<br>4. If authentication fails an error message shows up indicating this and the user is allowed to try again. |
| Postcondition: |

| |
|---|
| • The android user is connected to the server and can send song requests to the server. |

| Use Case 4: Search from Android Application |
|---|
| Actors:<br>• Android Application User |
| Preconditions:<br>• Internet access.<br>• The user has signed into the Wi-Fi/Server.<br>• The application is open. |
| Flow of Events:<br>1. The user queries a song in the search bar.<br>1. If no matching result is found a message is displays which indicates this.<br>2. Otherwise it displays a list of songs which match the queries. |
| Postcondition:<br>• The user can look at the results and select the most appropriate song. |

| Use Case 5: View Queue from Server |
|---|
| Actors:<br>• Android Application Users<br>• Server operators (manager at restaurant) |
| Preconditions:<br>• The server is up and running. |
| Flow of Events:<br>1. The server operator presses the designated button to view the queue.<br>1. If there are no songs in the queue a message shows up indicating this.<br>2. If the queue contains songs then the list of songs in the queue is shown. |
| Postcondition:<br>• The queue of songs awaiting playback can be seen by the operator of the server. |

| Use Case 6: View Queue from Application |
|---|
| Actor:<br>• Android Application User |
| Preconditions:<br>• Server is up and running.<br>• The android device has internet access.<br>• The android device has the application open and is connected to the server. |
| Flow of Events:<br>1. The User presses the designated button to view the queue.<br>1. If there are no songs in the queue, then a message shows up indicating this.<br>2. If the queue is non empty, then the list of songs in the queue is shown. |
| Postcondition:<br>• The android application user can view the list of songs in the queue. |

| Use Case 7: Request and Play Music |
|---|
| Actors:<br>• Android Application User |
| Preconditions:<br>• Internet access.<br>• The user has installed the application and signed into the Wi-Fi/Server.<br>• The user has found the song using the search feature. |
| Flow of events:<br>1. This use case starts after the user queries the song using the search feature.<br>2. He then selects the song from the resulting songs list displayed on the screen. |
| Postcondition:<br>• The song plays on the music system when all the other songs in the queue have been played. |

| Use Case 8: View User Connected To Server |
|---|
| Actors:<br>• Android Application User<br>• Server Operator |
| Preconditions:<br>• Internet access.<br>• The Server is online.. |
| Flow of events:<br>1. The user clicks the view users button on the homepage of the server application.<br>2. A list of active android users connected to the server are shown. |
| Postcondition:<br>• The Server operator can view all the users connected to the server. |

# 4 Other Non-functional Requirements

## 4.1 Performance Requirements

1. Lightweight server and android application which consumes minimal system resources so that it doesn't hinder other activities.
2. Server and android application take up little space, to save space on the devices.
3. Server and android application should be easy to use and intuitive.
4. Server should run within a minute.
5. Android application should open and connect to the server within a minute.
6. The hosting of server should not degrade internet speed significantly, customers who choose to use the local Wi-Fi for other task, apart from streaming music, should be able to do so with ease.

## 4.2 Safety and Security Requirements

- Privacy should be kept intact and only the most necessary information should be provided to the application.
- Hackers should not be able to gain access to the system and compromise security.
- No user should be allowed to jump ahead of the queue.
- The system should allow all users to listen to their music and not just allow one user to add music one after another.

## 4.3 Software Quality Attributes

### 4.3.1 Scalability

The application is intended to be used by multiple businesses that rely on stable user-base and therefore it must be compatible with different businesses that follow this business model (restaurants, cafes, gyms etc.) and thus should be scalable. We plan to achieve this by making the software cross compatible across platforms and by making it easy to use for the masses.

### 4.3.2 Portability

The application, as we intend it, is not supposed to work outside of our designated venues. The zones that we will create will be the only zones where the application will function, to cater for optimum customer satisfaction. However, the system (server) setup would be simple allowing a high distribution of servers and thus android device users would be able to make use of the Jukebox service from a variety of places.

### 4.3.3 Minimalistic

We aim to make the user interfaces for both applications as intuitive and simple as possible, by remove unnecessary features, in order to make the process natural and easy.

### 4.3.4 Maintainability

Our idea of a minimalistic application will substantially reduce potential maintenance issues. The application will only support a simple song-queue mechanism. However, the security protocols and will have to be updated at regular intervals.

### 4.3.5 Easy to use:

The application does not require a lot of technical knowledge to operate at both the server and user ends apart from basic knowledge of how to operate a computer or an android device.

# Appendix A - Group Log

Mohammad Obaid Ur Rehman: Coordinated and arranged meetings between group members. Also worked on typing of the document.
Shaheryar Faisal: Worked on idea development, anticipating potential problems and on the design of the system.
Wahid Ejaz: Overall planning on how to go approach problems faced.
Shahrukh Nawaz: Worked on noting down features that the application may offer.
Hamza Hassan: Worked on graphics and document formatting.

# Appendix B – Contribution Statement

| Name | Contributions in this phase | Approx. Number of hours | Remarks |
|---|---|---|---|
| Mohammad Obaid Ur Rehman | Section 1 | 4 | Wrote introduction and some parts of section 3 |
| Wahid Ejaz | Section 2 & 3 | 4 | Wrote overall description and some parts of section 3 |
| Shahrukh Nawaz | Section 3 | 4 | Wrote Functional Requirements |
| Shaheryar Faisal | Section 4 | 4 | Overall proof reading and writing work in section 4 |
| Hamza Hassan | Section 4 | 4 | Wrote non-functional requirements |