

# Change History

|  |  |
| --- | --- |
| Version | Notes |
| 0.1 | * Draft 1 * Initial format was proposed * Initial functional requirements defined * Some non-functional requirements were loosely defined. * Key Actors defined * Overall description added * Some external interface requirements added |

Glossary

|  |  |
| --- | --- |
| Term | Definition |
| Framework | A platform for developing software applications. Contains in-built functionality to assist in the development of software. |
| Front-end | The front-end of an application is defined as what the users can see and directly interact with. |
| Back-end | The back-end of an application relates to the server-side tasks and database communication. |
| Database | A structured set of data held in a computer or server. |
| SDK  (Software Development Kit) | A collection of software development tools in one package. |
| API  (Application Programming Interface) | A software interface which offers a service to other pieces of software. |
| WebSocket | A web communication protocol allowing two-way communication between a client and a server. |
| RDBMS  (Relational Database Management System) | Software that enables the creation and management of a relational database. A relational database is a structured database supporting the ability for data to link from one table to another, to form relationships. This enables ease of multi-table queries. |
| SRID  (Spatial Reference Identifier) | Indicates a specific spatial reference system (i.e. a set of parameters to describe geometry). e.g. Longitude and Latitude. |
| ACID  (Atomicity Consistency, Isolation, Durability) | Principles for data reliability.  Atomicity - uninterruptible operations,  Consistency - data integrity after deletion, insertion and updating,  Isolation - transactions will not affect others,  Durability - changes guaranteed to remain. |
| ORM  (Object-relational mapper) | Object-relational mappers (ORMs) are software libraries that enable object-relational mapping when interacting with data from a database system. Object-relational mapping refers to modelling and accessing database schemas using objects within an object-oriented programming language. |
| SQL  (Structured Query Language) | A programming language used to interact with databases, providing the ability to directly query for data, insert data, update, and remove data. Typically used within relational databases. |
| NoSQL | A database paradigm that stores data in an unstructured manner, typically without a predefined schema, as opposed to the relational database paradigm. |
| JSX  (JavaScript Syntax Extension) | An extension to the JavaScript programming language which provides methods to render React components. |
| MVC  (Model-View-Controller) | A software design pattern which separates software into three interconnected components. |

Table of Contents

[Change History 2](#_Toc101547775)

[1. Introduction 5](#_Toc101547776)

[1.1. Purpose 5](#_Toc101547777)

[1.2. Scope 5](#_Toc101547778)

[1.3. Overview 5](#_Toc101547779)

[2. Overall Description 5](#_Toc101547780)

[2.1. Product Perspective 5](#_Toc101547781)

[2.2. Product Functions 5](#_Toc101547782)

[2.3. Assumptions and Dependencies 6](#_Toc101547783)

[3. External Interface Requirements 6](#_Toc101547784)

[3.1. User Interfaces 6](#_Toc101547785)

[3.2. Hardware Interfaces 6](#_Toc101547786)

[3.3. Software Interfaces 6](#_Toc101547787)

[3.4. Communications Interfaces 6](#_Toc101547788)

[3.5. Key Actors 6](#_Toc101547789)

[4. Functional Requirements 7](#_Toc101547790)

[5. Use Cases 10](#_Toc101547791)

[6. Non-Functional Requirements 10](#_Toc101547792)

[6.1. Performance 10](#_Toc101547793)

[6.2. Reliability 11](#_Toc101547794)

[6.3. Usability 11](#_Toc101547795)

[6.4. Privacy and Security 11](#_Toc101547796)

[6.5. Legal Requirements 11](#_Toc101547797)

[7. References 11](#_Toc101547798)

# Introduction

## Purpose

This document describes the software requirements and features of Virtual Jukebox. This specification is intended for the designers, developers, and maintainers of the application.

## Scope

The scope of this document includes describing the functional and non-functional requirements of the Virtual Jukebox application, alongside the software’s key actors and use cases. It will not describe project tasks or technical software details explicitly, nor will it outline task resourcing per team member.

## Overview

Several functional requirements are defined in section 4, whereas several non-functional requirements are defined in section 5 under categories including performance, reliability, usability, privacy/security, and legal requirements. Use cases are defined in section 5**.** The key actors for the software include guests, hosts, database system, map system, all of which are described in section 3.5.

# Overall Description

## Product Perspective

Virtual Jukebox is a web application designed for use in social settings such as cafes and restaurants, which aims to mimic the functionality of a jukebox via user interaction with a host device acting as a music source. Users are able to queue songs from a playlist defined by the host by joining Virtual Jukebox sessions and spending credits which they earn by listening to songs over time. Songs are placed into a queue, in which other users can vote to increase the priority of specific songs within the queue to be played sooner than others. Additionally, all users within a session have access to a chat room, enabling textual communication among users in the same session. Users are able to view all active Virtual Jukebox sessions on a map, but may only join sessions in connection range of them. Joining sessions will require a password which may be entered manually, or via scanning a QR code.

## Product Functions

**User Accounts and Storage:** The Virtual Jukebox application will allow users to create a Virtual Jukebox account, and login using their credentials. A user can authenticate an existing Spotify account to their Virtual Jukebox account.  
  
**Song Queue:** Hosted Virtual Jukebox sessions will utilise a song queue which is viewable by all guest users as well as the host. The queue will control the ordering of song playback on the host device. The user queued songs will be ordered in a first-in-first-out manner by default, but songs with votes will be prioritized in playback order. Songs with the same amount of votes will remain in a FIFO order. When no songs are queued by users, the host device will randomly play songs in the host defined playlist.

**Chat Room:** The Virtual Jukebox application will support a chat room in which users connected to the same session can send and receive chat messages. Messages will be filtered for profanity.

**Virtual Jukebox Map:** The Virtual Jukebox application will support an interactive map which will show the location of nearby sessions. Users can join a session from the map if they are in a certain range.

**Music Streaming:** The Virtual Jukebox application will support music streaming through the Spotify SDK. Streaming will be played through the host’s device, and the host can control playback by skipping, pausing or playing the current song.

## Assumptions and Dependencies

For the software to remain fully functional, it will require a web server with a defined database to store user account data. It will also require access to the Spotify SDK to enable music streaming functionality in Virtual Jukebox sessions. Additionally, the Mapworks API will be necessary for Virtual Jukebox map functionality.

# External Interface Requirements

## User Interfaces

The user interface for the Virtual Jukebox application shall be compatible with both mobile and desktop devices, and will support a scalable UI for a consistent user experience between desktop and mobile devices alike.

## Hardware Interfaces

To operate the software, users will need access to a mobile phone, laptop, tablet, or desktop computer. Hosts will require access to devices with speakers (either connected internally, via cable, or wirelessly) to act as a music source.

## Software Interfaces

The software interfaces utilized for the application’s functionality include the Spotify SDK and its API, the Mapworks API, and the Django ORM for database accessibility.

## Communications Interfaces

Users will require access to a web-browser, specifically Microsoft Edge, Safari, Google Chrome, or Mozilla Firefox within the latest two versions.

Other communication interfaces include client-server communication which will be achieved via the Django back-end framework, HTTPS, encryption, and WebSocket for synchronisation.

## Key Actors

**Database System:** The database system will store user credentials for login, chat messages for chat room sessions, and session information such as its geographical location. The database system will frequently be queried for this data to enable functional software requiring persistent or semi-persistent data. Temporary data will be stored in a caching system instead. The database system in use is PostgreSQL, whereas the caching system in use is Redis.

**Map System:** The map system will supply the software with geographical data to represent a map. This will enable the portrayal of Virtual Jukebox host locations on a visual map enabling guests to identify them. The map system in use is Mapworks.

**Music Streaming System:** Supplies music stream to enable song playback on host devices. The music streaming system in use is Spotify SDK.

**Server:** The (web) server will store the database system and caching system. It will communicate with the software system via a back-end framework.

**Hosts:** Type of user. Hosts must create and log into an account to be able to host sessions. Hosts may create a session, and can control the music playback.

**Guests:** Type of user. Guests can create and log into an account, join a session, queue songs in a session, send messages, and vote on the current song queue. Guests are not required to log into an account to join sessions.

**Devices:** Devices will serve as the hardware interface between users (hosts and guests) and the Virtual Jukebox application. Devices may include desktop computers, laptops, mobile phones, and tablets. Users will input requests including voting and song queue requests via a device, and will view output of events via the devices. These events include operations undergone by the Virtual Jukebox application, such as song order changes or newly queued songs.

# Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Description | Difficulty | Business Value |
| VJ-1 | The system must allow users to create a Virtual Jukebox account with an email and password. | High | High |
| VJ-2 | The system must allow users to login to their Virtual Jukebox account with an email and password. | Medium | High |
| VJ-3 | The system must allow users to connect their Virtual Jukebox account with a Spotify account. | Low | High |
| VJ-4 | The system must allow users to select a location on a map to start a session. | Low | Medium |
| VJ-5 | The system could allow users to select a connection radius for a hosted session. | Low | Low |
| VJ-6 | The system must allow users to view the playlists of a connected Spotify account. | Low | Medium |
| VJ-7 | The system must allow users to choose a new or existing playlist to use as the session playlist. | Low | High |
| VJ-8 | The system must allow users to create either a public or password required private session. | Low | High |
| VJ-9 | The system must play the session’s music through the host’s device (Clearer description needed). | Low | High |
| VJ-10 | The system must allow the session host to pause, play or skip the current playing song. | Low | Medium |
| VJ-11 | The system must allow the session host to generate and share a QR code which will allow users to connect. | Medium | Medium |
| VJ-12 | The system must allow users to login with a guest account, which will generate a random username. | Low | High |
| VJ-13 | The system could allow users to set a username for their guest account. | Low | Low |
| VJ-14 | The system must allow all users with accounts to view all jukebox sessions on an interactive map. | Medium | Medium |
| VJ-15 | The system must allow users to join a public session from the interactive map, provided they are within the joinable range. | Medium | Medium |
| VJ-16 | The system must allow users to join a private session from the interactive map by supplying the correct password, provided they are within the joinable range. | Medium | Medium |
| VJ-17 | The system must allow users to join a public or private session with a QR code. | Medium | Medium |
| VJ-18 | The system must allow all users connected to a session to view the current music queue. | Low | High |
| VJ-19 | The system must allow all users connected to a session to view the playlist for the session. | Low | High |
| VJ-20 | The system must allow all users connected to a session to view the currently playing song. | Low | High |
| VJ-21 | The system must allow all users connected to a session to view the chat page and message history. | Low | Medium |
| VJ-22 | The system must allow all users connected to a session to vote on songs in the queue. | Medium | High |
| VJ-23 | The system must allow all users connected to a session to earn credits over time. | Low | Medium |
| VJ-24 | The system must allow all users connected to a session to use credits to add a song from the session playlist into the queue. | Medium | High |
| VJ-25 | The system must allow all users connected to a session to send and receive messages in the chat room. | High | Medium |
| VJ-26 | The system must sort the current queue by number of votes. | Low | High |
| VJ-27 | The system must sort the current queue by the order songs were added, provided the number of votes are equal. | Low | Medium |
| VJ-28 | The system must play the session playlist on shuffle playback if no songs are currently in the queue. | Low | High |
| VJ-29 | The system must not allow users to add a song to the queue that is currently in the queue. | Low | Medium |
| VJ-30 | The system must not allow users to have more than 5 current credits at any given time. | Low | Medium |
| VJ-31 | The system should enforce a minimum length for a session playlist. | High | Medium |
| VJ-32 | The system must filter profanity from all chat rooms. | Medium | High |
| VJ-33 | The system should display a popup for a user if a message they send is filtered by the profanity filter. | Low | Low |
| VJ-34 | The system could limit the number of messages a user can send in a period of time. | Low | Medium (Could help performance and prevent spam) |
| VJ-35 | The system must store sensitive data such as user passwords securely (Maybe define further?) | Medium | High |
| VJ-36 | The system should allow users to reset their password from the login page. | Medium | Medium |
| VJ-37 | The system must support users removing votes on the queue once they have been placed | Low | Low |
| VJ-38 | The system could support two factor authentication for logging in | Medium | Low |

# Use Cases

**TBA - Will be expanded**

Logging in

Using guest account

Making a room

Joining through the map

Joining through a QR code

Voting for a song

Sending a chat message

**Possible format?**

Goal: To allow a user to...  
Primary Actor:  
Secondary Actor:  
precondition:  
Trigger:

Flow of Events:

Extensions

# Non-Functional Requirements

## Performance

* The system must create user accounts within **six** seconds of the creation form being submitted (maybe this is too prescriptive).
* The system must update the music queue within **two** seconds of a vote being placed.
* The system must update the chat-room within **two** seconds of a message being sent.
* The system must generate a QR code within **five** seconds of a room being created.
* The system must be able to process up to **1000** messages per second.
* The system must be able to process up to **3000** votes per second.
* The system must retrieve and validate user details from the database within **four** seconds after a login request.
* The system must load each page within **two** seconds (to be evaluated on an internet connection of **5MBps speed)**.
* The system must support queue sizes of up to **10 000** songs.
* The system must support playlist sizes of up to **10 000** songs.
* The system must send a password reset email within one minute of a password reset request being received.

## Reliability

* The system must be available for **99.9%** of any given month.
* The system’s profanity filter should fail for no more than **one in 1000** messages.
* The system must play the most voted song **99.99%** of the time.
* The system must display the jukebox session on the correct map location **99.99%** of the time.

## Usability

* The system must allow users to join a session as a guest within **three** clicks (minus passcode entry).
* The system must allow users to queue a song while in a session within **three** clicks.
* The system must display the queue and the chat on one screen.
* The system must display media controls to the host within a session at all times.
* The system must provide a consistent UI across common desktop and mobile screens.
* The system must support the following sizes, ratios and browsers (1920x1080 - 4k, 16:9 - 21:9, Chrome, Firefox, MS Edge, and Safari).
* The system must provide a UI such that **95%** of users can navigate to join a session without help.

## Privacy and Security

* The system must not store unencrypted user passwords.
* The system must not store unencrypted session passwords.

## Legal Requirements

* The system should display a disclaimer stating that the application is not to be used commercially due to Spotify’s terms and conditions.

# References

-