RhythmiQ

Lab 2 Version 1

Team Blue | CS 411W | November 1, 2026

Team Members: Lucas Hayes, Alanna Ivey, Spencer Hall, Andrew Behre, Bea Eggleston, Dominick Rodriguez, Gaby Macatangay

1. Introduction..... 3
1.1 Purpose..... 3
1.2 Scope..... 3
1.3 Definitions, Acronyms, and abbreviations..... 3
1.4 References..... 4
1.5 Overview..... 4
2. Overall Description..... 5
2.1 Product Perspective..... 5
2.2 Product Functions..... 5
2.3 User Classes and Characteristics..... 6
2.4 Design and Implementation Constraints..... 6
2.5 Assumptions and Dependencies..... 6

1. Introduction

1.1 Purpose

The purpose of this section is to define the system behavior by providing a clear description of what the system will and won't do. This document will also define the limits and constraints on this system. Finally, this document will ensure that the requirements written below are clear, testable, and maintainable to guide the development team effectively.

1.2 Scope

Customers prefer to hear music they enjoy when shopping, but most businesses just play the same generic pop songs or background music that is boring at best. RhytmiQ is a web app and music system that allows customers to select the music they want to be played in a business or venue.

1.3 Definitions, Acronyms, and abbreviations

- **API (Application Programming Interface):** a set of rules and protocols that allows different software applications to communicate with each other
- **ASCAP (American Society of Composers, Authors, and Publishers):** a U.S. performing rights organization that licenses music for commercial use
- **BMI (Broadcast Music Inc.):** a U.S. performing rights organization that manages commercial music licensing on behalf of songwriters and publishers
- **Business Control Center:** VibeQueue's owner facing web dashboard for configuring vibe constraints, managing the song queue, and viewing analytics
- **Hardware Agnostic:** software designed to run on any existing device without requiring specific or proprietary hardware
- **LAMP:** an open source web development stack comprised of Linux (OS), Apache (web server), MySQL (database), and PHP (scripting language)
- **MFCD (Major Functional Component Diagram):** a visual representation of a system's major components and the data flows between them, as defined in CS 410
- **QR Code (Quick Response Code):** a type of two dimensional matrix barcode that can be scanned by a smartphone camera to open a URL or application
- **SESAC (Society of European Stage Authors and Composers):** a U.S. performing rights organization providing commercial music licensing
- **SoundMachine:** a third party commercial music streaming service that holds ASCAP/BMI/SESAC licenses; used by VibeQueue for legal audio playback in venues
- **Spotify API:**

- **Vibe Engine:** RhythmiQ's proprietary, in-house song selection and validation algorithm; evaluates customer song requests against venue's configured constraints before allowing them into the queue

1.4 References

Rockbot. (n.d.). *Rockbot business music platform*. https://rockbot.com/go

SoundMachine. (n.d.). *Commercial music streaming for businesses*. https://sound-machine.com

TouchTunes. (n.d.). *Our story*. https://www.touchtunes.com/about/our-story

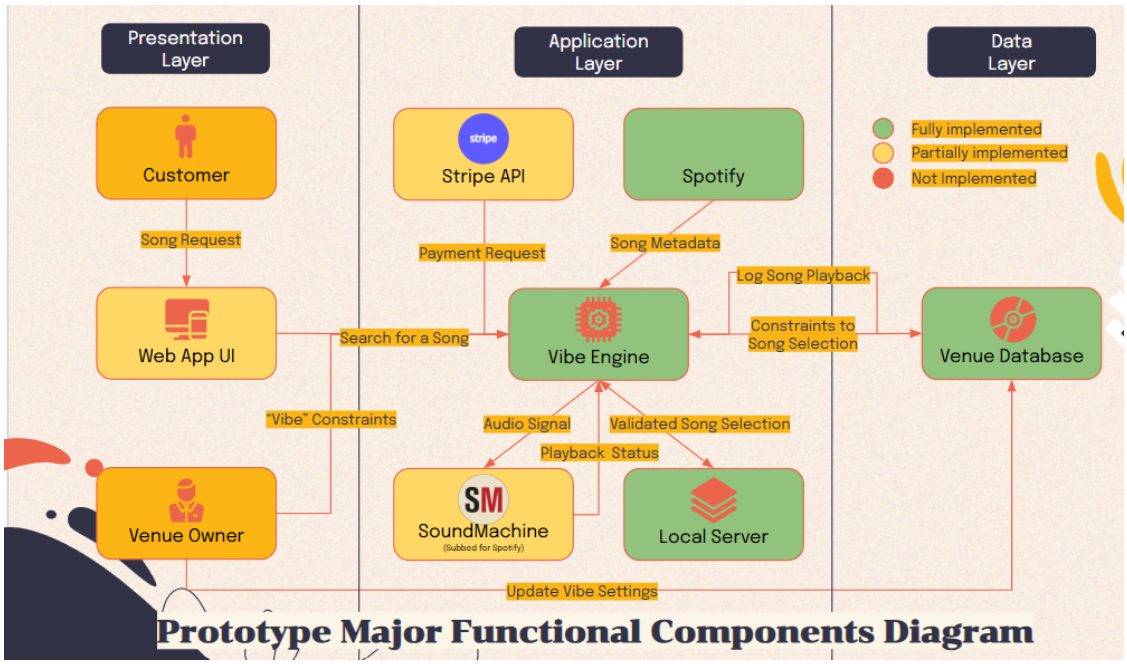
1.5 Overview

Section 2 of this document provides a brief overview of RhythmiQ
Section 3 of this document contains specific requirements

2. Overall Description

2.1 Product Perspective

RhythmiQ is a web app that provides business owners with a platform to manage their music system and provide consumers with the ability to play their favorite music. It aims to give consumers an improved shopping experience. For business owners, it aims to provide an easy-to-use platform to manage their music and ensure the happiness of their customers.



2.2 Product Functions

RhythmiQ as a platform allows consumers to search for their favorite music and request songs to be played. For business owners, it allows them to blacklist specific songs or genres, as well as enable or disable certain settings. Owners can also give control access to their staff to block or allow specific songs or disable song requests.

Category	Feature	Customer	Staff	Owner	RWP	Prototype (Planned)
Account	Account Creation	✔	✔	✔	Creation of user accounts	Fully Implemented
Account	Login	✔	✔	✔	Login screen and authorization	Fully Implemented
Account	Payment Processing	✔	✔	✔	Payment processing through Stripe API	Partially Implemented

Song Queue	Song Selection	✔	✔	✔	Song selection from list	Fully Implemented
Song Queue	Vibe Engine	✔	✔	✔	Song recommendations based on venue music preferences	Fully Implemented
Song Queue	Priority Boost	✔	✔	✔	One or more tiers that push the request higher in the queue	Partially Implemented
Song Queue	Block Songs (Queue Override)		✔	✔	Prevent songs that don't fit the vibe from playing	Fully Implemented
Venue	Blacklist Song/Genre			✔	Choose which genres or moods to block from being playable	Fully Implemented
Venue	View Revenue			✔	Live tracking of revenue from song requests	Partially Implemented

2.3 User Classes and Characteristics

RhytmiQ has three different classes of users: Consumers, Staff, and Owners. Consumers interact with RhythmiQ through a QR code scannable at the venue. Consumers can then create an account or log in, search for a song or genre, and request a specific song or a song from a specific genre to be played. Staff are capable of doing everything a Consumer can, but also block or blacklist specific songs or genres. If needed, they are also capable of whitelisting songs or overriding a block/blacklist temporarily. Owners have all of the capabilities of Staff and Consumers, but with added control over the venue. Owners can control authority of users with respect to the venue. Owners are also able to activate or deactivate RhytmiQ on the venue.

2.4 Design and Implementation Constraints

Because RhythmiQ mainly handles music, a heavily regulated media in the United States, music licensing is a concern. As this is merely a prototype and a school project, in the interest of cost savings, the team decided to use Spotify's API for playing music. This decision was made knowing that, if this project were ever to become a legitimate business and product, a different legal licensing option would be required.

2.5 Assumptions and Dependencies

- Tech stack: LAMP (Linux, Apache, MySQL, PHP)
- Spotify API
- SoundMachine API licensed audio playback
- QR code generation library (venue specific portal linking)
- Web based front end (mobile first responsive design)