RhythmiQ

Lab 2 Version 2

Team Blue | CS 411W | November 8, 2026

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

1. Introduction 4
1.1 Purpose 4
1.2 Scope 4
1.3 Definitions, Acronyms, and abbreviations 4
1.4 References 5
1.5 Overview 5
2. Overall Description 6
2.1 Product Perspective 6
2.2 Product Functions 6
2.3 User Classes and Characteristics 7
2.4 Design and Implementation Constraints 7
2.5 Assumptions and Dependencies 7
3 System Requirements 8
3.1 System Features 8
3.1.1 Account Management 8
3.1.1.1 Introduction/Purpose of Feature 8
3.1.1.2 Stimulus/Response 8
3.1.1.3 Associated Functional Requirements 8
3.1.2 Login 8
3.1.2.1 Introduction/Purpose of Feature 8
3.1.2.2 Stimulus/Response 9
3.1.2.3 Associated Functional Requirements 9
3.1.3 QR Code Creation 9
3.1.3.1 Introduction/Purpose of Feature 9
3.1.3.2 Stimulus/Response 9
3.1.3.3 Associated Functional Requirements 9
3.1.4 Venue Connection 10
3.1.4.1 Introduction/Purpose of Feature 10
3.1.4.2 Stimulus/Response 10
3.1.4.3 Associated Functional Requirements 10
3.1.5 Song Search and Validated Request Submission 10
3.1.5.1 Introduction/Purpose of Feature 10
3.1.5.2 Stimulus/Response 10
3.1.5.3 Associated Functional Requirements 11
3.1.6 Vibe Engine Evaluation 12
3.1.6.1 Introduction/Purpose of Feature 12
3.1.6.2 Stimulus/Response 12
3.1.6.3 Associated Functional Requirements 12
3.1.7 Staff Queue Management 13
3.1.7.1 Introduction/Purpose of Feature 13
3.1.7.2 Stimulus/Response 13

3.1.7.3 Associated Functional Requirements....14
3.1.8 Priority Requests....15
3.1.8.1 Introduction/Purpose of Feature....15
3.1.8.2 Stimulus/Response....15
3.1.8.3 Associated Functional Requirements....15
3.1.9 Venue Blacklist Management....16
3.1.9.1 Introduction/Purpose of Feature....16
3.1.9.2 Stimulus/Response....16
3.1.9.3 Associated Functional Requirements....16
3.2 Non-Functional Requirements....16
3.2.1 Performance and Capacity....17
3.2.2 Security and Privacy....17
3.2.3 Reliability and Recovery....18
3.2.4 Usability and Accessibility....19
3.3 Design Constraints....19
3.4 Requirements Traceability and Verification....20

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
- **MusicBrainz:** External source for baseline recording, release, title and artist information.
- **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 RhythiQ for legal audio playback in venues
- **Spotify API:** enables the creation of applications that can interact with Spotify's streaming service, such as retrieving content metadata, creating and managing playlists, or controlling playback.

- **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

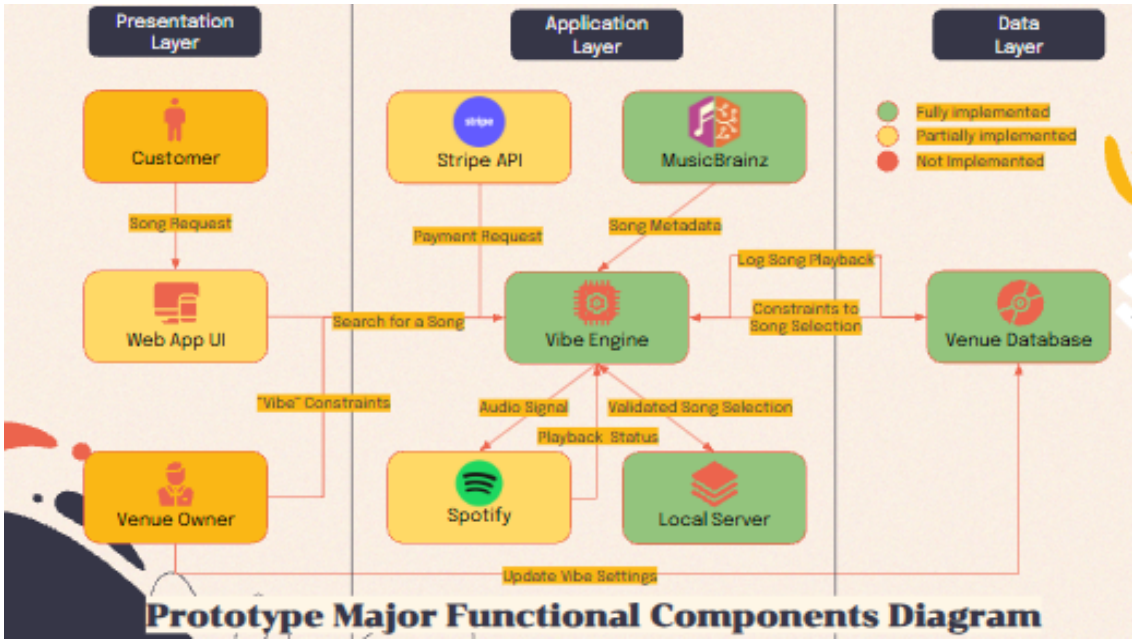
1.5 Overview

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 allows customers to open venue-specific request portals, search for eligible music and submit song requests. The system checks the active venues music requirements before adding the song to the queue. 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. Staff will also be able to manage pending requests, along with having their own login information and dashboard.

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 supports three different user classes: 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)

3 System Requirements

3.1 System Features

3.1.1 Account Management

3.1.1.1 Introduction/Purpose of Feature

This feature allows for the creation of user accounts on RhythmiQ's website.

3.1.1.2 Stimulus/Response

Stimulus: User clicks the 'Register' button and enters their credentials (email and desired password)
Response: Upon submission, the user's credentials are processed by the system, verifying that the email and password are sufficient and comply with account standards. Following verification, the credentials are securely stored in the system's database.

3.1.1.3 Associated Functional Requirements

3.1.1.3.1 Registration Page (O: Hayes)
The system shall present a registration page for users to enter relevant information. The relevant information includes the user's email, their desired password, and a second entry of their desired password for verification of accuracy.
3.1.1.3.2 Email Validation (O: Hayes)
The system shall verify that the email provided does not already have an account. If the system detects an existing account with the email provided, the user will be notified and given the option to login.
3.1.1.3.3 Password Validation (O: Hayes)
The registration form must enforce strong password creation. The system shall reject account creation for the following reasons:

1. Password length is less than 7 characters
2. Password is more than 32 characters
3. Password uses invalid characters (/,\, (,), ~, `)

3.1.1.3.4 Email Verification (O: Hayes)
The system shall send a confirmation email to the address provided at account creation. The email will contain a confirmation message and link to finish the account creation process. The user must click the link to finish account creation.

3.1.2 Login

3.1.2.1 Introduction/Purpose of Feature

The login feature allows users to access all features of RhythmiQ's website, including account information and song selection.

3.1.2.2 Stimulus/Response

Stimulus: Users initiate the login process by providing their login credentials on the login page, prompting the system's authentication process.
Response: Following authentication, the user is directed to an account page showing personal information and connected venues.

3.1.2.3 Associated Functional Requirements

3.1.2.3.1 Login Page (O: Hayes, M1: Ivey)
The system shall display a login form accepting the registered user's username and password.
3.1.2.3.2 Login Validation (O: Hayes, M1: Ivey)
The system shall reject invalid username/password combinations, show an authentication error and refrain from opening an authenticated user session.
3.1.2.3.3 Login Completion (O: Hayes, M1: Ivey)
Following successful login, the system shall create an authenticated user session and redirect the user to the home page, displaying the user at the top right of screen.

3.1.3 QR Code Creation

3.1.3.1 Introduction/Purpose of Feature

This feature allows the user to create a unique QR code for their venue, allowing others to connect to the venue through RhythmiQ.

3.1.3.2 Stimulus/Response

Stimulus: The user clicks the "Create QR code" button in the "My Venues" section of their account page.
Response: The system creates a unique QR code that points to a URL on RhythmiQ. The QR code will include the venue's unique ID. Upon users scanning the QR code, they will be taken to RhytmiQ's website. The venue ID within the QR code will allow RhytmiQ to display a page for the venue, where users will be able to see the current status, request songs, and see information about the venue.

3.1.3.3 Associated Functional Requirements

3.1.3.3.1 Venues Page (O: Hayes)
The system must provide a venues section on the account page that shows all venues owned by the user, as well as any venues the user is connected to.
3.1.3.3.2 QR Code Creation (O: Hayes)
The system must create a unique QR code when prompted.
3.1.3.3.3 QR Code Access (O: Hayes)
Venue QR codes must be accessible for any connected venue from the venues section.
3.1.3.3.4 Invalid Code Handling (O: Ivey)
If the code or token is unknown, malformed, or revoked, the system shall display invalid-code message and shall not open request form.
3.1.3.3.5 Venue Separation (O: Ivey)

Scanning code for one venue shall not connect a customer to, or submit a request for, a different venue.
3.1.3.3.6 Venue accessibility (O: Hayes)
Venues may disable song requests at any time or for only scheduled hours. If the venue is not currently active in RhytmiQ, the venue page shown upon QR code scan will notify the user via an alert window in the user's browser.

3.1.4 Venue Connection

3.1.4.1 Introduction/Purpose of Feature

This feature allows the user to connect to a venue. Connection is required for the user to request songs on that specific venue.

3.1.4.2 Stimulus/Response

Stimulus: The user scans a RhythmiQ QR code at the venue and clicks the link provided by the QR code.
Response: The link in the QR code directs the user to a venue connection page on RhytmiQ's website.

3.1.4.3 Associated Functional Requirements

3.1.4.3.1 Connection Page (O: Hayes)
The system must provide a connection page showing the to-be-connected venue and confirm/cancel buttons.
3.1.4.3.2 Venue Page (O: Hayes)
The system must provide a default venue page accessible to all users connected to the venue. It must show information about the venue such as:

1. Venue name
2. Song request status (allowed/disabled)
3. Whitelisted genres

3.1.4.3.3 Venue Confirmation (O: Ivey)
The system shall identify the target venue by name and require customer confirmation before connecting a session to it.

3.1.5 Song Search and Validated Request Submission

3.1.5.1 Introduction/Purpose of Feature

This feature allows a customer connected to a venue to search for a song and submit a request, with or without an account. Before a request is recorded, the system checks the venue's request availability, the customer's existing requests, and the venue's music rules, and it tells the customer the outcome.

3.1.5.2 Stimulus/Response

Stimulus: A customer on a venue's song search page enters one or more words from a song title, an artist name, or both; selects a search result; and selects Request song.

Response: The system returns matching songs, then checks the request against the venue's request availability, the customer's pending requests, and the Vibe Engine (Section 3.1.6). It records the request as pending, marks it for staff review, or refuses it, and displays a message stating the outcome.

3.1.5.3 Associated Functional Requirements

3.1.5.3.1 Song Search Access (O: Dominick)
The system shall provide a song search function on each venue's search page to every visitor, whether or not the visitor is logged in.
3.1.5.3.2 Song Search Query (O: Dominick)
The system shall accept a query of one or more words and shall return songs from the local song library in which every word appears in the song title or the artist name; when no local song matches, the system shall return the music metadata provider's results for the query.
3.1.5.3.3 Search Results (O: Dominick)
The system shall display up to 10 results, each showing the song title, the primary artist, the genre when one is known, an "E" marker when the song is flagged explicit, and a Request song option, and shall list each song only once.
3.1.5.3.4 Local Song Library (O: Dominick)
The system shall save every song returned by the music metadata provider in the local song library, with its title, artist, and any genre and explicit flag the provider's data includes, and shall answer later searches that match saved songs without contacting the provider.
3.1.5.3.5 No Matches Notice (O: Dominick)
When neither the local song library nor the music metadata provider returns a match, the system shall display "No matches found" and suggest trying a different song title or artist.
3.1.5.3.6 Search Unavailable Notice (O: Dominick)
When the music metadata provider returns an error or does not respond within 5 seconds and no local song matches, the system shall display that music search is temporarily unavailable, distinct from the no-matches notice, instead of an error page.
3.1.5.3.7 Request Availability Enforcement (O: Dominick)
The system shall not record a song request for a venue whose song requests are disabled and shall hide the Request song option on that venue's search page.
3.1.5.3.8 Request Availability Notice (O: Dominick)
When song requests are disabled at a venue, the system shall display "Song requests are currently unavailable at this venue." on the search page and after any attempted request.
3.1.5.3.9 Duplicate Request Prevention (O: Dominick)
The system shall not record a second request for a song the same customer already has pending at the same venue and shall display that the customer already has a pending request for that song. A customer is identified by their account when logged in and otherwise by their browser session.
3.1.5.3.10 Pending Request Limit (O: Dominick)
The system shall limit each customer to five pending requests per venue and shall refuse further requests with a message asking the customer to wait until some are reviewed.
3.1.5.3.11 Simultaneous Submissions (O: Dominick)

The system shall enforce 3.1.5.3.9 and 3.1.5.3.10 even when one customer submits several requests at the same moment, so that simultaneous submissions never create duplicate requests or more than five pending requests.
3.1.5.3.12 Vibe Engine Check (O: Dominick)
Before recording a request, the system shall evaluate the selected song with the Vibe Engine (Section 3.1.6) and shall save the evaluation with the request.
3.1.5.3.13 Rejected Request Handling (O: Dominick)
When the Vibe Engine rejects a song, the system shall not record a song request and shall display "[song title] does not meet this venue's music settings."
3.1.5.3.14 Missing Metadata Handling (O: Dominick)
When the song's genre, tempo in beats per minute (BPM), or explicit-content data needed by a venue rule is missing, the system shall apply the venue's unknown-song setting (3.1.6.3.6); under the default setting it shall record the request as pending and display "[song title] needs staff review."
3.1.5.3.15 Accepted Request Handling (O: Dominick)
When the Vibe Engine accepts a song, the system shall record a pending song request containing the song, venue, requester, request time, and status, and shall display a confirmation that names the requested song.
3.1.5.3.16 Request Status Display (O: Dominick)
The system shall show each customer a My Requests page listing their requests at the current venue, newest first, with each song's title, artist, and current status (Pending, Approved, Rejected, Played, or Removed).
3.1.5.3.17 Queue Entry (O: Dominick)
The system shall not place a pending request in the venue queue until staff approve it (Section 3.1.7); on approval, the system shall add it to the queue as described in Section 3.1.8.

3.1.6 Vibe Engine Evaluation

3.1.6.1 Introduction/Purpose of Feature

The Vibe Engine evaluates each requested song against the venue's music settings so that only songs that fit the venue's atmosphere can enter the queue. In the prototype it evaluates the venue's song blacklist, explicit-content setting, allowed and blocked genres, and BPM range. Language and mood restrictions belong to the real-world product and are not evaluated by the prototype.

3.1.6.2 Stimulus/Response

Stimulus: A customer submits a song request at a venue that is accepting requests.
Response: The Vibe Engine compares the song's stored data with the venue's settings and returns Accept, Reject, or Staff review together with the reasons. The system records the decision and applies it to the request (Section 3.1.5).

3.1.6.3 Associated Functional Requirements

3.1.6.3.1 Song Blacklist Rule (O: Dominick)
The Vibe Engine shall reject a song that is on the venue's song blacklist.
3.1.6.3.2 Explicit Content Rule (O: Dominick)

When the venue does not allow explicit content, the Vibe Engine shall reject a song marked explicit.
3.1.6.3.3 Genre Rule (O: Dominick)
The Vibe Engine shall reject a song whose genre is on the venue's blocked-genre list and, when the venue has an allowed-genre list, a song whose genre is not on that list. Genre comparison shall ignore letter case and surrounding spaces.
3.1.6.3.4 BPM Rule (O: Dominick)
When the venue sets a minimum or maximum BPM, the Vibe Engine shall reject a song whose BPM is below the minimum or above the maximum.
3.1.6.3.5 Valid BPM Range (O: Dominick)
The system shall not save a venue setting with a BPM value below 1 or a minimum BPM greater than the maximum BPM.
3.1.6.3.6 Unknown Song Setting (O: Dominick)
When no rule rejects a song but a rule that applies to the venue lacks the song data it needs (explicit status, genre, or BPM), the Vibe Engine shall return the outcome chosen in the venue's unknown-song setting: Staff review (the default), Accept, or Reject.
3.1.6.3.7 Unverified Data (O: Dominick)
The Vibe Engine shall treat song data whose source is marked unverified as missing.
3.1.6.3.8 Accept Outcome (O: Dominick)
The Vibe Engine shall return Accept only when no rule rejects the song and every applicable rule has the data it needs, or when the unknown-song setting is Accept.
3.1.6.3.9 Decision Record (O: Dominick)
The system shall record every Vibe Engine decision with the venue, song, request, outcome, reasons (for example, song_blacklisted or genre_not_allowed), the settings and song data used, and the time. Recorded decisions shall not be editable or deletable through the administration site.
3.1.6.3.10 Reasons for Staff (O: Dominick)
The system shall display the latest decision outcome and its reasons beside each request on the staff dashboard.
3.1.6.3.11 Rejected Songs Stay Out (O: Dominick)
The system shall prevent staff from approving a request whose latest Vibe Engine decision is Reject.

3.1.7 Staff Queue Management

3.1.7.1 Introduction/Purpose of Feature

This feature lets a venue's owner and assigned staff review incoming song requests and control the approved queue. Every status change is reflected in what each customer sees on their My Requests page..

3.1.7.2 Stimulus/Response

Stimulus: A staff member or owner opens the venue's staff dashboard and selects Approve, Reject, Mark played, Remove, move up, move down, or Set priority for a request.

Response: The system checks that the user may manage the venue, applies the change if the request's current status allows it, updates the queue and the request's status, and redisplays the staff dashboard with the result.

3.1.7.3 Associated Functional Requirements

3.1.7.3.1 Staff Dashboard Access (O: Dominick)
The system shall allow only the venue's owner, staff assigned to the venue, and site administrators to open a venue's staff dashboard or change its requests; it shall send visitors who are not logged in to the login page and refuse all other users.

3.1.7.3.2 Request List (O: Dominick)
The staff dashboard shall list the venue's requests, newest first, with each song's title, artist, status, and latest Vibe Engine outcome and reasons.
3.1.7.3.3 Approve Request (O: Dominick)
The system shall allow staff to approve a Pending request, changing its status to Approved and adding it to the venue queue (Section 3.1.8).
3.1.7.3.4 Reject Request (O: Dominick)
The system shall allow staff to reject a Pending request, changing its status to Rejected; a rejected request shall not enter the queue.
3.1.7.3.5 Mark Played (O: Dominick)
The system shall allow staff to mark an Approved request as Played, changing its status to Played and removing it from the queue.
3.1.7.3.6 Remove Request (O: Dominick)
The system shall allow staff to remove an Approved request from the queue, changing its status to Removed.
3.1.7.3.7 Invalid Status Change (O: Dominick)
The system shall refuse every status change not listed in 3.1.7.3.3 through 3.1.7.3.6, such as approving a Rejected or Played request, and shall display "Invalid request status transition."
3.1.7.3.8 Reorder Queue (O: Dominick)
The system shall allow staff to move a queued request up or down one place among requests with the same priority; moving the first entry up or the last entry down shall leave the queue unchanged.
3.1.7.3.9 Customer Status Update (O: Dominick)
After any change in 3.1.7.3.3 through 3.1.7.3.6, the system shall show the request's new status on the customer's My Requests page the next time that page is loaded.
3.1.7.3.10 Simultaneous Staff Changes (O: Dominick)
The system shall apply simultaneous queue changes for the same venue one at a time, so that no two queue entries share a position and no request appears in the queue twice.
3.1.7.3.11 Request Availability Control (O: Dominick)
The staff dashboard shall show whether new song requests are enabled and shall allow the owner and assigned staff to enable or disable them; the change shall apply to the next request submitted.
3.1.7.3.12 Action Messages (O: Dominick)
The system shall display a confirmation message after each successful approve, reject, or mark-played action and each request-availability change; shall redisplay the updated queue after each successful move, priority change, or removal; and shall display a readable error message after each refused change.

3.1.8 Priority Requests

3.1.8.1 Introduction/Purpose of Feature

This feature decides the order of each venue's queue. Every approved request has a priority, and higher-priority requests play before lower-priority ones. Standard requests start at priority 0, and in the prototype only staff change a request's priority.

3.1.8.2 Stimulus/Response

Stimulus: A staff member approves a request or enters a new priority for a queued request.
Response: The system places or reorders the request in the queue by priority and then by position. The staff dashboard shows the new order immediately, and customers see status changes the next time their page loads.

3.1.8.3 Associated Functional Requirements

3.1.8.3.1 Standard Priority (O: Dominick)
The system shall assign priority 0 to each request when it is approved.
3.1.8.3.2 Queue Position Assignment (O: Dominick)
On approval, the system shall give the request a queue position one greater than the highest existing position in that venue's queue.
3.1.8.3.3 Queue Order (O: Dominick)
The system shall order each venue's queue by priority from highest to lowest and, within the same priority, by queue position from lowest to highest.
3.1.8.3.4 Set Priority (O: Dominick)
The system shall allow staff to set a queued request's priority to any whole number from −10 to 10; a request with higher priority shall move ahead of every request with lower priority.
3.1.8.3.5 Priority Validation (O: Dominick)
The system shall reject a priority that is not a whole number from −10 to 10, shall display that priority must be between −10 and 10, and shall leave the queue unchanged.
3.1.8.3.6 Position Updates (O: Dominick)
When a request is played or removed, the system shall delete its queue entry, and the remaining entries shall keep their relative order.
3.1.8.3.7 Unique Positions (O: Dominick)
The system shall never give two requests at the same venue the same queue position and shall never place one request in the queue more than once.

3.1.9 Venue Blacklist Management

3.1.9.1 Introduction/Purpose of Feature

This feature lets a venue restrict which songs can be requested. In the prototype, a user with administration-site access sets the venue's song blacklist, allowed and blocked genres, explicit-content setting, BPM range, and unknown-song setting, and the Vibe Engine enforces them on every new request.

3.1.9.2 Stimulus/Response

Stimulus: A user with administration-site access adds or removes a song on a venue's blacklist, or changes the venue's genre, explicit-content, BPM, or unknown-song settings, and saves the venue.
Response: The system stores the change, and the Vibe Engine applies it to every request submitted afterward.

3.1.9.3 Associated Functional Requirements

3.1.9.3.1 Venue Music Settings (O: Dominick)
The system shall store for each venue an explicit-content setting (default: not allowed), an optional minimum and maximum BPM, and an unknown-song setting (default: Staff review). When a venue has no saved settings, the Vibe Engine shall use these defaults.
3.1.9.3.2 Song Blacklist (O: Dominick)
The system shall allow a user with administration-site access to add any song in the song library to a venue's blacklist and to remove it; a song shall appear on a venue's blacklist at most once.
3.1.9.3.3 Genre Filters (O: Dominick)
The system shall allow a user with administration-site access to mark genres as allowed or blocked for a venue; each genre shall appear at most once per venue.
3.1.9.3.4 Venue Separation (O: Dominick)
A venue's blacklist and genre filters shall apply only to requests at that venue.
3.1.9.3.5 Enforcement (O: Dominick)
The system shall apply blacklist and genre changes to every request submitted after the change is saved, through the Vibe Engine rules in Section 3.1.6.
3.1.9.3.6 Existing Requests (O: Dominick)
A blacklist or genre change shall not change the status of requests already recorded; staff remove such requests through the staff dashboard (Section 3.1.7).
3.1.9.3.7 Settings Access (O: Dominick)
Only users with administration-site access shall be able to change a venue's music settings.

3.2 Non-Functional Requirements

Non-functional requirements specify measurable properties of the system rather than additional features. The following are proposed acceptance criteria, not observed results.

3.2.1 Performance and Capacity

3.2.1.1 Introduction/Purpose of Feature

This feature establishes the performance and capacity requirements of RhytmiQ. These requirements ensure that the system can efficiently process song searches, request submissions, and queue updates while supporting multiple users.

3.2.1.2 Stimulus/Response

Stimulus: Multiple customers access RhytmiQ simultaneously to search for songs, submit requests, while staff and venue owners manage requests and venue settings.

Response: The system processes each interaction within established response time limits while maintaining accurate queue information. The system supports the expected number of concurrent requests and provides appropriate feedback if increased activity prevents requests from being completed within a specified time.

3.2.1.3 Associated Functional Requirements

3.2.1.3.1 Song Search Response Time (O: Gaby)
The system shall return song search results within 2 seconds for at least 95% of requests under normal operating conditions

3.2.1.3.2 Concurrent User Support (O: Gaby)
The system shall support at least 20 concurrent users searching for songs or submitting requests without creating duplicate requests or compromising queue accuracy.

3.2.1.3.3 Queue Update Performance (O: Gaby)
The system shall save successful staff queue changes within 3 seconds under normal operating conditions. Updated queue information shall be displayed when the staff dashboard is refreshed.

3.2.2 Security and Privacy

3.2.2.1 Introduction/Purpose of Feature

This feature defines the security and privacy requirements necessary to protect user information and maintain appropriate access to RhytmiQ's features. These requirements ensure that customer accounts, payment information and venue management functions are protected against unauthorized access.

3.2.2.2 Stimulus/Response

Stimulus: A customer, staff member, or venue owner attempts to access account information, submit a payment, or perform an action that requires specific user permissions.

Response: The system verifies the user's identity and permissions before allowing access to protect information or restricted functions, unauthorized actions are denied.

3.2.2.3 *Associated Functional Requirements*

3.2.2.3.1 Role Authorization (O: Ivey)
The system shall deny Staff or Owner restricted actions when the current authenticated user lacks required venue-specific role
3.2.2.3.2 Credential Protection (O: Ivey)
The system shall have a limit to how many times a password can be incorrect to prevent brute testing for passwords.
3.2.2.3.3 Payment Data Protection (O: Ivey)
The RhymiQ application database shall not store full payment-card numbers or card security codes.

3.2.3 Reliability and Recovery

3.2.3.1 Introduction/Purpose of Feature

This feature establishes the reliability and recovery requirements of RhytmiQ. These requirements ensure that the system maintains accurate song request, payment records, and queue information during normal operation hours and can respond appropriately to unexpected errors or service interruptions.

3.2.3.2 Stimulus/Response

Stimulus: The system experiences a network interruption, database connection failure, external service outage or unexpected error while processing a song request, payment, or queue update.

Response: The system identifies the unsuccessful operation and provides an appropriate error message to the affected user. The system preserves previously confirmed information, prevents duplicate requests or transactions and restores access to available functions once the interrupted service becomes operational.

3.2.3.3 *Associated Functional Requirements*

3.2.3.3.1 Safe Metadata Failure (O: Ivey)
If a required metadata lookup fails, the system shall not mark the request accepted and shall present a service unavailable status.
3.2.3.3.2 Safe Payment Failure (O: Ivey)
If a payment attempt is declined or cannot be confirmed, the system shall not activate a paid request.
3.2.3.3.3 Queue Consistency (O: Ivey)
After a saved queue change, the customer view and staff view shall resolve the same request status and order.

3.2.4 Usability and Accessibility

3.2.4.1 Introduction/Purpose of Feature

This feature defines the usability and accessibility requirements of RhytmiQ, to ensure that customer, staff, and venue owners can navigate the platform, understand features and complete necessary actions through an accessible interface.

3.2.4.2 Stimulus/Response

Stimulus: A customer accesses RhytmiQ through venue's QR code using a mobile device, or a staff member or venue owner accesses the platform to manage song requests and venue settings.

Response: The system presents a responsive interface appropriate for the user's device and assigned role. Navigation options, instructions, and status messages are displayed clearly to guide users through available functions.

3.2.4.3 *Associated Functional Requirements*

3.2.4.3.1 Responsive Access (O: Ivey)
The customer page, song search, and request status screen shall remain usable.
3.2.4.3.2 Action Feedback (O: Ivey)
Following a request submission, the interface shall show one of the statuses accepted, rejected, or pending with a reason for rejection or pending processing.
3.2.4.3.3 Keyboard operation (O:Ivey)
Core login, search, submission and staff queue controls shall be operable with a keyboard.

3.3 Design Constraints

3.3.1 Introduction/ Purpose of Feature

This feature identifies the technical limitations, external service dependencies and implementation restrictions that influence the development and operation of RhytmiQ.

3.3.2 Stimulus/Response

Stimulus: A customer searches for or requests a song, or venue owner modifies the venue's music preferences, requiring the system to communicate with its application components, database, and external music services.

Response: The system uses its configured services, including MusicBrainz for song metadata, to retrieve and process the information needed to support song selection and request management. If required external service is unavailable, the system shall notify the user and prevent incomplete or invalid requests from being processed.

3.3.3 *Associated Functional Requirements*

3.3.3.1 Web-Based Customer Access (O: Ivey)
The customer facing request flow shall be accessible through venue specific web link encoded in a QR code and shall not require installation of a dedicated mobile application.
3.3.3.2 Song Information Service (O: Ivey)
The configured MusicBrainz integration shall be used for baseline recording and artist identification.
3.3.3.3 External Payment Boundary (O: Ivey)
When paid requests are enabled, the system shall use a configured external payment provider for financial authorization and shall not directly process raw payment card credentials.
3.3.3.4 Licensing Boundary (O: Ivey)
Commercial venue audio playback, if added to a deployable product, shall use an appropriately licensed playback integration rather than relying on song metadata access alone.

3.4 Requirements Traceability and Verification

Use Case/ User Need	Relevant Requirements	Interface → Logic→ Data	Verification Example
UC-01 Customer submits and tracks request	3.1.6; 3.2.2.3.3; 3.2.3.3.2	Request/status pages→ validation and queue service → request table	Submit, duplicate, disable requests, and recalculate position after changes
UC-02 Customer purchases priority	3.1.8; 3.1.6	Tier selection → paid priority → payment and request records	Compare tier order and FIFO within tier after successful payment
UC-03 Staff manages requests	3.1.9; 3.2.2.3.1	Staff queue screen → permission and queue logic	Approve, reject, remove, reorder; check customer status and audit
UC-04 Owner enforces music restrictions	3.1.7	Owner settings → restriction evaluation → blacklist and venue rules	Block a recording/artist and verify rejection of matching request
UC-05 Users can authenticate	3.1.1; 3.1.3; 3.2.2	Registration/login→authentication service→ hashed credentials and roles	Try valid/invalid login and unauthorized owner-page access