RhythmiQ

Lab 2

Spencer Hall

Team Blue | CS 411 | November 1, 2026

Team Members: Lucas, Alanna, Spencer, Andrew, Bea, Dominick, Gaby

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) establishes the technical and operational criteria for the implementation and verification of the RhythmiQ system. Written directly for development and engineering teams, this document serves as the formal technical baseline defining software architectures, data models, external communication interfaces, and functional modules required to construct the working software prototype.

1.2 Scope

RhythmiQ is a cloud-hosted, hardware-agnostic music queuing and atmosphere management platform tailored for small, independent hospitality venues. The project aims to eliminate the financial barrier of expensive proprietary jukebox hardware while mitigating copyright liability by bridging patron device interaction with licensed commercial streaming backends

The system will:

1. Allow patrons to scan a location-specific QR code to access a progressive, mobile-optimized song selection web interface without requiring native application downloads.
2. Evaluate patron song requests using the server-side Vibe Engine against active venue constraints, including musical genre, tempo (BPM), acoustic mood, and explicit content flags.
3. Dispatch validated track requests to a fully compliant commercial streaming provider (Spotify) to maintain full ASCAP, BMI, and SESAC commercial licensing compliance.
4. Provide venue operators with a real-time web portal (Business Control Center) for song skipping, queue overrides, venue filter management, and crowd engagement telemetry.

The system will not:

5. Depend on, ship, or operate dedicated wall-mounted jukeboxes or vendor-locked physical hardware terminals.
6. Stream unlicensed local MP3s, client device audio files, or consumer-grade personal streaming subscriptions (e.g., standard consumer Spotify or Apple Music accounts).
7. Support global, cross-venue shared queues; active queues are isolated to their designated physical venue.

1.3 Definitions:

API (Application Programming Interface): A set of defined protocols and routines enabling structured data interchange between separate software services.

ASCAP (American Society of Composers, Authors, and Publishers): A United States performing rights organization that licenses musical public performance rights for commercial entities.

BMI (Broadcast Music, Inc.): A United States performing rights organization that manages commercial music licensing on behalf of songwriters and music publishers.

BCC (Business Control Center): The administrative dashboard utilized by venue staff and managers to configure atmospheric boundaries, veto tracks, reorder the active playback queue, and monitor analytics.

Hardware Agnostic: Software engineered to execute across standard commercial-off-the-shelf devices and standard web browsers without requiring proprietary physical enclosures or peripherals.

LAMP: An open-source web stack composed of Linux, Apache, MySQL, and PHP/Python.

MFCD (Major Functional Component Diagram): A systems architecture schematic visually illustrating internal software modules, data pipelines, and third-party external service boundaries.

QR Code (Quick Response Code): A machine-readable two-dimensional matrix barcode containing encoded URLs scannable by mobile camera modules.

RWP (Real-World Product): The fully realized, production-ready, commercial-grade software platform intended for scalable commercial market deployment.

SESAC (Society of European Stage Authors and Composers): A United States performing rights organization handling musical composition licensing for public commercial spaces.

Spotify Web API: A suite of web endpoints providing track catalog search, playback queue dispatch, audio analysis, and player state control.

MusicBrainz API: An open-source, community-maintained music encyclopedia API providing standardized track metadata, acoustic fingerprinting, and artist release taxonomies.

SRS (Software Requirements Specification): A technical deliverable describing the exact behavioral, functional, and performance requirements of a software release

Vibe Engine: RhythmiQ's proprietary server-side validation algorithm that evaluates patron-requested songs against active venue constraints prior to queue insertion.

1.4 References:

Team Blue (Lucas, Alanna, Spencer Hall, Andrew, Bea, Dominick, Gaby), "Lab 1: RhythmiQ Product Description & System Feasibility," Department of Computer Science, Old Dominion University, Lab Report, CS 411, Fall 2026.
Spotify AB, "Spotify for Developers: Web API Reference," [Online]. Available: https://developer.spotify.com/documentation/web-api. [Accessed: 18-Oct-2026].
MetaBrainz Foundation, "MusicBrainz API Documentation," [Online]. Available: https://musicbrainz.org/doc/MusicBrainz_API. [Accessed: 18-Oct-2026].

1.5 Overview:

Section 2 (Overall Description) establishes the architectural context of RhythmiQ, detailing the operational flows, prototype versus real-world product functional tables, user role specifications, system constraints, and external dependencies.

Section 3 (Specific Requirements) formalizes the low-level functional requirements across database schemas, API integrations, queue processing logic, security configurations, and user interface workflows.

2. Overall Description

2.1 Product Perspective:

RhythmiQ operates as a distributed web platform that bridges customer mobile devices, venue administrative consoles, and licensed commercial playback systems. By shifting queue interaction to personal smartphones, the system functions without dedicated hardware kiosks

2.2 Product Functions:

Functional Area	Feature Description	Real-World Product (RWP)	CS 411 Prototype Build
Patron Web Portal	Frictionless QR code onboarding without app download	Fully Implemented	Fully Implemented
Catalog Metadata Ingestion	Querying track attributes, BPM, genre, and explicit markers	Fully Implemented (MusicBrainz live feed)	Fully Implemented (MusicBrainz API / cached metadata store)
Vibe Engine Validation	Real-time filtering against dynamic venue atmospheric rules	Fully Implemented (Continuous multi-variable scoring)	Fully Implemented (Deterministic checks: BPM range, genre, explicitness)
Administrative Control (BCC)	Web interface for track vetoes, reordering, and skips	Fully Implemented (Multi-role granular dashboard)	Fully Implemented (Core skip, override, and rule controls)
Audio Playback Engine	Continuous licensed commercial music streaming	Fully Implemented (Direct enterprise commercial stream integration)	Fully Implemented (Spotify Web API playback pipeline / virtual driver)
Deployment Scale	Multi-venue tenant isolation and load balancing	Fully Implemented (Multi-tenant cloud architecture)	Partially Implemented
Operational Analytics	Historical engagement, dwell times, and genre trends	Fully Implemented (Complete business intelligence reporting)	Partially Implemented (Basic queue transaction logs and track counters)
Content Moderation	Automated lyric scanning and patron rate limits	Fully Implemented (NLP lyric analysis and dynamic blacklisting)	Partially Implemented (Metadata-based explicit tagging and static blocks)

2.3 User Characteristics:

Patron (Guest User):

Expertise: Novice. Familiar with standard mobile camera usage and consumer web applications.

Usage Scenario: Enters the establishment, scans a table QR code, searches for songs, views real-time queue position, and submits tracks.

Constraints: Connects via varied mobile browsers (Safari, Chrome) using cellular networks or venue Wi-Fi; bounded by venue session proximity.

Floor Staff / Barista:

Expertise: Novice to Intermediate. Accustomed to commercial touch-screen POS systems.

Usage Scenario: Observes the live playback display on a counter device; uses a one-touch interface to veto inappropriate tracks or advance the queue during customer service operations.

Constraints: Fast-paced service setting requiring minimal clicks and immediate UI execution without entering configuration menus.

Venue Manager / Owner:

Expertise: Intermediate. Familiar with web configuration dashboards and SaaS software management.

Usage Scenario: Uses the Business Control Center to set up hours, configure daytime vs. evening vibe profiles, manage song request pricing, and view daily request metrics.

Constraints: Requires secure authentication and reliable storage of venue configuration parameters.

System Administrator:

Expertise: Advanced. Professional software and system administration background.

Usage Scenario: Configures database structures, manages third-party API credentials, audits uptime telemetry, and monitors deployment health.

Constraints: Accessible through secure authenticated administrator roles.

2.4 Constraints:

Copyright and Licensing Regulations: The application cannot stream audio from unverified sources, local files, or personal consumer accounts. All commercial streaming output must be routed through legally certified providers (Spotify) carrying valid ASCAP, BMI, and SESAC commercial performance licenses.

Hardware Neutrality: The software cannot require proprietary jukebox terminals, custom microcontrollers, or vendor-locked wall units. All end-user interactions must remain web accessible.

Response Latency: Track search retrieval and Vibe Engine validation cycles must be executed within 1.5 seconds under standard broadband and cellular conditions to maintain real-time responsiveness.

Third-Party Rate Limits: Ingestion from APIs is constrained by external rate caps, requiring server-side caching of song metadata to prevent rate exhaustion.

2.5 Assumptions and Dependencies:

Third-Party Metadata Service: Assumes regular availability and stable endpoint schema from the MusicBrainz API.

Commercial Streaming Availability: Relies on consistent network availability and streaming uptime from the Spotify Web API service infrastructure.

Venue Hardware and Network: Assumes the venue provides stable internet service and possesses standard commodity audio equipment (PC/tablet connected via auxiliary or Bluetooth to venue speakers).

Client Browser Capabilities: Assumes customer smartphones support standard HTML5, modern ECMAScript/JavaScript, and camera-based QR code scanning.