OLD DOMINION UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

SOFTWARE REQUIREMENTS SPECIFICATION

RhytmiQ

Lab 2 | Sections 1 and 2

Prepared by

Alanna Ivey

CS 411W | Sumaya Sanober

November 9, 2026, 2026

Draft based on Lab 1. Team confirmation is required for the MFCD and prototype feature statuses.

Prepared in accordance with IEEE 830-1998 section organization

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) defines the intended behavior, system boundaries, interfaces, user classes, and implementation considerations for RhytmiQ. It is intended primarily for software developers and testers implementing and validating the prototype. Sections 1 and 2 establish the context for the detailed, testable product requirements to be specified in Section 3.

1.2 Scope

RhytmiQ is a web-based, hardware-agnostic music request and queue-management platform for small hospitality venues. Customers submit song requests through a digital interface while authorized venue personnel retain control of which requests are accepted and played. The intended system evaluates requests against venue-configured music constraints and provides a business-facing interface for queue management. The project targets venues that want customer participation without proprietary jukebox hardware or repeated verbal requests to staff [1].

The envisioned real-world product includes venue configuration, a customer request channel, queue administration, music metadata evaluation, and integration with a commercially licensed playback provider. The SRS does not assume that the prototype itself supplies commercial music licenses or unrestricted playback rights. Exact prototype inclusions and exclusions must be reconciled with the team's missing RWP/prototype comparison table before submission.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
API	Application Programming Interface; protocols allowing software applications to communicate.
ASCAP	American Society of Composers, Authors and Publishers; U.S. music performing-rights organization.
BMI	Broadcast Music Inc.; U.S. music performing-rights organization.
Business Control Center	Owner-facing dashboard for configuring music constraints, managing the queue, and viewing analytics.
Hardware-Agnostic	Designed to work on existing devices without proprietary hardware.
LAMP	Linux, Apache, MySQL, and PHP; legacy glossary term in Lab 1, not the confirmed current prototype stack.
MFCD	Major Functional Component Diagram; diagram of principal components and data flows.
QR Code	Quick Response Code; scannable code that opens a URL or application.
SESAC	Society of European Stage Authors and Composers; U.S. music licensing organization.
MusicBrainz	Commercial music streaming service named as the intended playback integration in Lab 1.

Soundcharts API	Music analytics interface named in Lab 1 for metadata such as genre, tempo, mood, popularity, and explicit-content flags.

Term	Definition
Vibe Engine	Rule-evaluation component that compares requested songs against venue constraints.

1.4 References

[1] A. Ivey, “Lab 1 – RhytmiQ ,” CS 411W, Old Dominion University, version 1, unpublished course report, 2026.

[2] R. E. Milliman, “Using background music to affect the behavior of supermarket shoppers,” Journal of Marketing, vol. 46, no. 3 , pp.86–91, 1982, doi: 10.2307/1251706.

[3] A. C. North, D. J. Hargreaves, and J. McKendrick, “The influence of in-store music on wine selections,” Journal of AppliedPsychology, vol. 84, no. 2, pp. 271–276, 1999, doi: 10.1037/0021-9010.84.2.271.

[4] National Restaurant Association, “Restaurant industry facts at a glance,” 2023. [Online]. Available: https://restaurant.org/research-and-media/research/restaurant-statistics/restaurant-industry-facts-at-a-glance/

[5] TouchTunes, “Interactive music and entertainment for venues.” [Online]. Available: https://www.touchtunes.com

[6] Rockbot, “Business music, TV & digital signage solutions.” [Online]. Available: https://rockbot.com

[7] SoundMachine, “Licensed background music for business.” [Online]. Available: https://soundmachine.com

[8] Soundcharts, “Music analytics platform.” [Online]. Available: https://soundcharts.com

[9] Spotify, “Spotify terms and conditions of use,” 2023. [Online]. Available: https://www.spotify.com/us/legal/end-user-agreement/[10] Rockbot, “Licensing,” 2026. [Online]. Available: https://rockbot.com/licensing

1.5 Overview

Section 2 describes the product perspective, major functions, user characteristics, constraints, and dependencies. Section 3, to be developed separately, will specify detailed functional and nonfunctional requirements in a form that developers can implement and testers can verify.

2. Overall Description

2.1 Product Perspective

RhytmiQ is envisioned as a browser-accessible system connecting customers and venue personnel to shared request and queue data. The team’s development environment has used Django for server-side application logic and PostgreSQL for persistent storage. In the intended architecture, customer-facing pages send requests to the web application; server-side authorization and the Vibe Engine determine how requests are handled; authorized users administer the queue through the Business Control Center. Lab 1 also names Soundcharts as a metadata source and SoundMachine as an intended commercial playback service [1]. Their integration status is unconfirmed.

2.1 Product Perspective (continued)

Figure 1 shows the team-provided prototype's major functional components organized into presentation, application, and data layers. The customer uses the web application to request songs. Venue owners configure music constraints and venue settings. The Vibe Engine evaluates song selections using venue rules and music metadata; the venue database stores configuration and playback information. The application layer also includes Stripe API, MusicBrainz, SoundMachine, and a local server. Integration labels and statuses below reproduce the team diagram as planned architecture, not independent evidence of completed end-to-end integrations.

Presentation layer	Application layer	Data layer
Customer Venue Owner Web App UI	Vibe Engine Stripe API MusicBrainz SoundMachine Local Server	Venue Database

Primary data flows: Customer song request → Web App UI → Vibe Engine; Owner vibe constraints → Vibe Engine; MusicBrainz song metadata → Vibe Engine; Vibe Engine ↔ Venue Database (constraints and playback logs); Stripe API → payment request handling; Vibe Engine ↔ SoundMachine (audio/playback status); Vibe Engine → Local Server (validated selection).

Component status from team diagram: Customer and venue-owner presentation components are shown as partially implemented; Web App UI, Stripe API, and SoundMachine are partially implemented; MusicBrainz, Vibe Engine, Venue Database, and Local Server are shown as fully implemented. This is a representation of the team's diagram, not a test report.

2.2 Product Functions

The following statuses reflect the team's **planned prototype** feature table, rather than an independent verification of implemented code. All listed functions are in the real-world product scope.

Category/feature	User access	Prototype (planned)
Account creation	Customer, Staff, Owner	Fully implemented
Login/authorization	Customer, Staff, Owner	Fully implemented
Payment processing (Stripe)	Customer, Staff, Owner	Partially implemented
Song selection	Customer, Staff, Owner	Fully implemented
Vibe Engine recommendations	Customer, Staff, Owner	Fully implemented
Priority boost	Customer, Staff, Owner	Partially implemented
Block songs/queue override	Staff, Owner	Fully implemented
Blacklist song/genre	Owner	Fully implemented
View revenue	Owner	Partially implemented
SoundMachine playback integration	Application service	Partially implemented

2.3 User Characteristics

Customer: A venue guest using a phone or browser to search for songs and submit requests. No specialized technical expertise is required. Customers may use song selection, account functions, payment processing, priority boost, and Vibe Engine recommendations as specified in the team feature table.

Staff: Venue personnel who can use song-selection features and override or block songs in the queue. Staff should be familiar with routine venue operations but need not have system administration expertise.

Owner: Venue administrator who configures blocked genres or songs, controls the venue music policy, and reviews revenue. Owners also have access to account and song-queue functions. Role restrictions must be enforced server-side.

2.4 Constraints

The prototype uses a browser-based interface and the team's Django/Python and PostgreSQL development environment. Docker Compose supports local development and demonstration. Commercial playback requires appropriate music licensing and compliance with third-party service terms. Availability of external service credentials, test environments, and development time can constrain payment and playback demonstrations. The product is intended to avoid proprietary jukebox hardware.

2.5 Assumptions and Dependencies

RhytmiQ depends on the web application, database, network connectivity, and configured access to external services. The team MFCD identifies MusicBrainz for song metadata, Stripe API for payment processing, SoundMachine for commercial music playback, and a local server for validated song selection. SoundMachine and Stripe integration are partially implemented according to the team materials. Deployment assumes compatible browsers, a functioning PostgreSQL instance, and valid environment configuration. Failure behavior for unavailable APIs and playback services should be specified in Section 3.