# Project Title

Sprint Number
Date

| **Name** | **Email Address** |
|---|---|
| Name1 | WKU email address1 |
| Name2 | WKU email address2 |

CS 396
Fall 2025
Project Technical Documentation

# Contents

# List of Figures

# 1 Introduction

## 1.1 Project Overview

The Content Delivery Network (CDN) Implementation project is designed to enhance the performance, scalability, and reliability of a music streaming and download service by caching audio content at multiple geographically distributed edge locations. By leveraging edge caching, the system reduces latency, improves download speeds, and provides a seamless user experience for streaming music, even during peak usage periods. The primary objective is to ensure that users can access audio files quickly and efficiently, regardless of their location or device, while the backend intelligently manages routing, caching, and content delivery.

The system architecture was developed using Node.js for the backend, which provides efficient server-side processing, request handling, and real-time updates to cached content. The frontend was built using HTML, CSS, and JavaScript, creating a responsive and intuitive user interface where users can easily find, stream, and download music. Together, this technology stack ensures a fast, scalable, and user-friendly experience.

Functionally, the CDN caches music files in multiple formats (MP3, AAC, OGG) at edge servers to maximize compatibility with diverse devices and playback applications. Requests are routed intelligently based on a user's geographical location and network conditions, ensuring minimal streaming latency. Real-time updates allow new releases and content changes to propagate quickly, guaranteeing that users always have access to the latest music. The system also incorporates analytics to track streaming quality, download speeds, and user interactions, providing administrators with actionable insights.

Non-functional requirements focus on ensuring response times under 100 milliseconds, scalability through the addition of edge servers, and industry-standard security protocols such as encryption and authentication. The system also emphasizes seamless usability with clear feedback and error handling. With built-in monitoring and analytics, the CDN can be continuously optimized for performance and efficiency, making it a robust solution for large-scale music distribution.

## 1.2 Project Scope

Text goes here.

## 1.3 Technical Requirements

### 1.3.1 Functional Requirements

| **Mandatory Functional Requirements** |
|---|
| 1. The CDN must cache audio files at edge locations to minimize latency for users accessing the music streaming or download service. Cached content should include various formats and qualities to accommodate different user preferences. |
| 2. The system must intelligently route requests to the nearest edge location based on user geographical location and network conditions to ensure optimal performance during music streaming or downloads. |
| 3. The CDN must support real-time content updates, allowing new music releases or changes to existing files to be propagated quickly across edge locations, ensuring that users always access the latest content. |
| 4. The CDN must be capable of serving audio files in multiple formats (e.g., MP3, AAC, OGG) to accommodate various devices and playback requirements, ensuring compatibility with a wide range of client applications. |
| 5. The system must provide analytics features that track user interactions, streaming quality, and download speeds, allowing administrators to monitor performance and make data-driven improvements. |
| **Extended Functional Requirements** |
| 1. The CDN must provide users with the ability to adjust playback options, such as play, pause, and skip, during streaming. |
| 2. The system must provide users with the option to download songs for offline playback in supported formats. |
| 3. The CDN must allow users to view recently played or downloaded songs for quick access. |
| |
| |

The functional requirements of the CDN music application define the core capabilities the system should provide to deliver music efficiently and effectively to users. The CDN is designed to cache audio files at multiple edge servers, reducing latency and improving performance for both streaming and downloads. Requests are routed to the nearest edge location, ensuring that users receive the fastest possible access based on their geographic location and network conditions. The system also supports real-time updates, so new released and modified files are released across the network. Guaranteeing users always have access to the most up-to-date content. Additionally, the CDN serves audio in multiple formats such as MP3, AAC, and OGG to support a wide range of devices and applications. Users can also download songs for offline playback, access recently played or downloaded content, and control playback with basic options such as play, pause, and skip during streaming.

### 1.3.2 Non-Functional Requirements

| **Mandatory Non-Functional Requirements** |
|---|
| 1. The CDN must ensure that the average response time for streaming and download requests is under 100 milliseconds, even during peak usage periods, to provide a seamless user experience. |
| 2. The system must be designed to handle an increasing number of simultaneous users and requests without performance degradation, allowing for the addition of new edge servers as user demand grows. |
| 3. The system must implement industry-standard security protocols to protect user data and audio content, including encryption of data in transit and robust authentication mechanisms to prevent unauthorized access. |
| 4. The CDN must provide a seamless and intuitive user experience, ensuring that users can easily find, stream, and download music with minimal steps and without confusion, supported by clear feedback and error messaging throughout the process. |
| 5. The system must provide real-time monitoring and analytics capabilities, allowing administrators to track performance metrics, user interactions, and content delivery efficiency, facilitating ongoing optimization and troubleshooting. |
| **Extended Non-Functional Requirements** |
| 1. The system must maintain an uptime of at least 99 percent to ensure high availability of the music streaming and download service. |
| 2. The system must maintain error rates (e.g., failed requests) below 1 percent to ensure a consistent and reliable user experience. |
| 3. The CDN must be designed with a responsive user interface so that the service works smoothly on desktops |
| |
| |

The non-functional requirements of the CDN project establish the quality standards and operational goals that ensure the system's efficiency, scalability, and reliability. The CDN must provide a fast and seamless experience, maintaining response times under 100 milliseconds, even during peak traffic. Availability is a priority, with the system expected to maintain 99.9 percent uptime, while reliability is ensured by keeping request error rates below 1 percent. Scalability is supported by a design that allows the easy addition of new edge servers as user demand grows. The system also emphasizes security, implementing encryption for data in transit and robust authentication to protect both users and audio content. From a usability perspective, the user interface is designed to be responsive across desktops, tablets, and mobile devices, ensuring accessibility for all. Finally, administrators benefit from real-time monitoring and analytics to track performance, optimize delivery, and troubleshoot issues efficiently.

# 2 DevOps - Continuous Integration and Continuous Delivery Approach and Results

# 3 DevOps - Architecture Approach, Models, and Results

# 4 DevOps - Product and Process Approach and Results

# 5 DevOps - Product Management and Monitoring Approach and Results

# 6 DevOps - Cultural Approach and Results

# 7 Software Testing and Results

## 7.1 Software Testing Plan Template

**Test Plan Identifier:**
**Introduction:**
**Test item:**
**Features to test/not to test:**
**Approach:**
**Test deliverables:**
**Item pass/fail criteria:**
**Environmental needs:**
**Responsibilities:**
**Staffing and training needs:**
**Schedule:**
**Risks and Mitigation:**
**Approvals:**

# 8 Conclusion

Text goes here.

# 9 Appendix

## 9.1 Software Product Build Instructions

Text goes here.

## 9.2 Software Product User Guide

Text goes here.

## 9.3 Source Code with Comments

Text goes here.