JetStream Streaming Service

Software Requirements Specification

03/13/2022 - Revision A1

Team 06

**SRS Authors**

| **Name** |
| --- |
| Nicholas Phan |
| Shane Arwood |
| Hyungtaek Kim |
| Anshul Kumar |

**Contents**

[1](#_heading=h.gjdgxs) Introduction 3

[1.1](#_heading=h.30j0zll) Document Purpose 3

[1.2](#_heading=h.1fob9te) Project Scope 3

[1.3](#_heading=h.3znysh7) Definitions, Acronyms, and Abbreviations 3

[1.4](#_heading=h.2et92p0) Referenced Documents 4

[1.5](#_heading=h.tyjcwt) Document Overview 4

[2](#_heading=h.3dy6vkm) Overall System Description 4

[2.1](#_heading=h.1t3h5sf) Product Functional Categories 4

[2.2](#_heading=h.4d34og8) User (Actor) Characteristics 5

[2.3](#_heading=h.2s8eyo1) Constraints 6

[2.4](#_heading=h.17dp8vu) Assumptions and Dependencies 6

[2.5](#_heading=h.3rdcrjn) Apportioning of Features or Requirements 6

[3](#_heading=h.26in1rg) Specific Requirements 7

[3.1](#_heading=h.lnxbz9) External & User Interfaces 7

[3.1.1](#_heading=h.35nkun2) GUI Screens 7

[3.1.2](#_heading=h.1ksv4uv) Service Interfaces to External Clients 10

[3.2](#_heading=h.2jxsxqh) Functional Requirements 10

[3.2.1](#_heading=h.z337ya) Formal Requirements 11

[3.3](#_heading=h.3j2qqm3) Non-Functional (Performance) Requirements 11

[3.4](#_heading=h.1y810tw) Fully Dressed Use Case Documents 11

[3.5](#_heading=h.4i7ojhp) Logical Database Requirements 12

[3.6](#_heading=h.2xcytpi) Design Constraints 12

[3.6.1](#_heading=h.1ci93xb) Standards Compliance 12

[3.7](#_heading=h.3whwml4) System Quality Attributes 12

[3.7.1](#_heading=h.2bn6wsx) Reliability / Availability 12

[3.7.2](#_heading=h.qsh70q) Security 13

[3.7.3](#_heading=h.3as4poj) Maintainability 13

[3.7.4](#_heading=h.1pxezwc) Portability 13

[4](#_heading=h.49x2ik5) Supporting Information 13

# Introduction

This section describes the purpose and contents of the SRS. It provides definitions for key terms used in the document as well as references.

## Document Purpose

The purpose of this document is to present a detailed description of the features and requirements for the JetStream Streaming Service project. This document contains an overall description of the project, its non-functional and functional requirements, and the use cases for the features the JetStream Streaming Service will provide. It also includes specifications for how the system’s quality will be assessed.

This document is targeted at the users, project management, and the system developers.

## Project Scope

The JetStream Streaming Service is a web-based application that provides on-demand music streaming services to Customers.

Subscribers pay for an account that allows them to browse for music and create a library to access and organize songs. Artists are able to add their music and receive payment for it. Administrators manage the information associated with subscriber and artist accounts.

The goal of JetStream is to provide user-friendly access to music and enable artists to share their content to subscribers. It has extensive browsing capabilities and options for users to curate their library, playlists, and listening experience. Through the project’s automatic subscription platform, account and payment management is secure and seamless.

The product is accessible to Customers through a browser-based client interface or client App that runs on any mobile platform. Internet connection on a supported device is required for access.

## Definitions, Acronyms, and Abbreviations

| **Term** | **Definition** |
| --- | --- |
| Account | The data associated with a specific customer, such as personal information, music library, and account preferences, which requires log-in to access and edit. |
| Administrator | Employee of the streaming service that manages subscriber and artist accounts. |
| Automatic Payment | A bill charged to the user’s chosen payment method every month without the user having to initiate the payment each time. |
| Artist | User that registers and uploads media to the streaming service site. |
| Browsing | Using keywords, recommendations, similar content, and other features to find new or specific media. |
| Customer | The individual who uses the streaming service and the capabilities it provides. |
| Encryption | Encoding the user’s personal information, such as their payment methods or account details, to ensure it is secured. |
| Library | The collection of songs, playlists, and other content that a user has saved to their account. |
| Media Player | The section of the streaming service responsible for user interaction with the on-demand content, which displays the content currently being consumed and provides options for volume control, pause/play, etc. |
| Streaming Service | A software product that provides on-demand access to media and options to consume and organize the media. |
| Subscriber | User that pays to access the streaming service’s media catalog and use the streaming service’s features. |
| Subscription | Recurring access to the streaming service through an automatic payment every month. |
| User | Any individual that uses the application, such as the customer or administrator. |

## Referenced Documents

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.

## Document Overview

The remainder of this document contains three additional sections.

Section Two gives an overall description of the system, including functional categories, actors, and constraints.

Section Three describes requirements that the system must meet in order to function. It will provide a detailed explanation of the functional requirements, non-functional (performance) requirements, fully dressed use cases, logical database requirements, and design constraints. Additionally, it will provide some example user interfaces.

# Overall System Description

This section will provide an overview of the complete system. It shows the functional categories of the system, how actors will interact with the system, constraints, and dependencies. It will also conclude with a section about how these requirements will be implemented.

## Product Functional Categories

| **Media Player** | **Includes all options for viewing and using the media player.** |
| --- | --- |
| **Features**: | Viewing the media player |
|  | Maintaining a queue of songs |
|  | Specifying listening preferences |

| **Browsing** | **Allowing users to navigate through music they wish to listen to.** |
| --- | --- |
| **Features**: | Browsing through pages of music |
|  | A search function for quicker access |
|  | Categorization and saving music |

| **Account Management** | **All features necessary for proper account management.** |
| --- | --- |
| **Features**: | Create an account |
|  | Log in with an account |
|  | Find an account ID and password |
|  | Allow the changing of account information |
|  | Deleting an account |

| **Payment Management** | **All features related to paying for the subscription service.** |
| --- | --- |
| **Features**: | Allow customers to choose their payment method. |
|  | Update payment method |
|  | View transaction history |
|  | Bill customers monthly automatically |
|  | Ensure a secure payment process |

## User (Actor) Characteristics

The system will handle two different actors, the customer and the admin. These will have their own role and goals with the system.

| **User** | **A customer who wishes to interact with the system and listen to music** |
| --- | --- |
| **Actor’s Goals:** | Log in to the system |
|  | Manage their account settings |
|  | Manage their library of music |
|  | Browse a catalog of music |
|  | Play and listen to music |

| **Admin** | **Company users who wish to manage the system or help customers with potential problems.** |
| --- | --- |
| **Actor’s Goals:** | Manage a customer’s account settings |
|  | Manage user information |
|  | Create new customer accounts |
|  | Remove customer accounts |
|  | Assist in any problems a user may face |

## Constraints

A significant constraint on the system’s implementation is that the service’s main goal of providing on-demand music requires a large database for a consistently growing amount of media (songs). Developers will have to account for this growth.

Furthermore, because JetStream is web-based, the user experience will depend on the device and client used as well as the reliability and speed of the internet connection used to access the service. The convenience of browsing and listening to the user library on-demand might differ for each user as they access the server with different devices and internet providers.

Another constraint is that the service requires a subscription, so sensitive information about users relating to their payment methods will have to be collected. It is critical that there are enough security measures in place to protect Customer information.

## Assumptions and Dependencies

The requirements depend on the JetStream’s users having a compatible device for the streaming service. The device will need to have either built-in sound output or the ability to connect to an external sound output. It is assumed that Customers have such a device as well as access to an internet connection, as the system is web-based.

With regard to data protection, it is also assumed that there is one user per subscription, allowing for only one profile and library to be associated with a given account.

The requirements also depend on the integrity of the users when distinguishing between subscriber and artist when creating an account. It assumes artist accounts will be created by signed artists to ensure quality control of the media uploaded to the service.

## Apportioning of Features or Requirements

The first release of the JetStream Streaming Service will be largely self-contained, with artists directly uploading their content to the service and subscribers being able to access that content on-demand directly from the client. However, future releases of the system may incorporate the ability to share music, playlists, and libraries to other platforms (such as social media), and the requirements would need to change accordingly to ensure the integrity of the system and protection of subscriber information.

Future versions of the system may also expand who may create an artist account to upload media, such as indie artists not associated with a record label. This will prompt requirements related to quality control to ensure subscribers will only encounter genuinely recorded songs and not spam when browsing.

# Specific Requirements

This section contains system requirements including Interfaces, GUI Screens, Functional & Non-Functional Requirements, and Used Cases with detailed description.

## External & User Interfaces

This section contains the interfaces that will be provided by this system. It also provides a prototype of GUI Screens with description of how they work and communicate each.

### GUI Screens

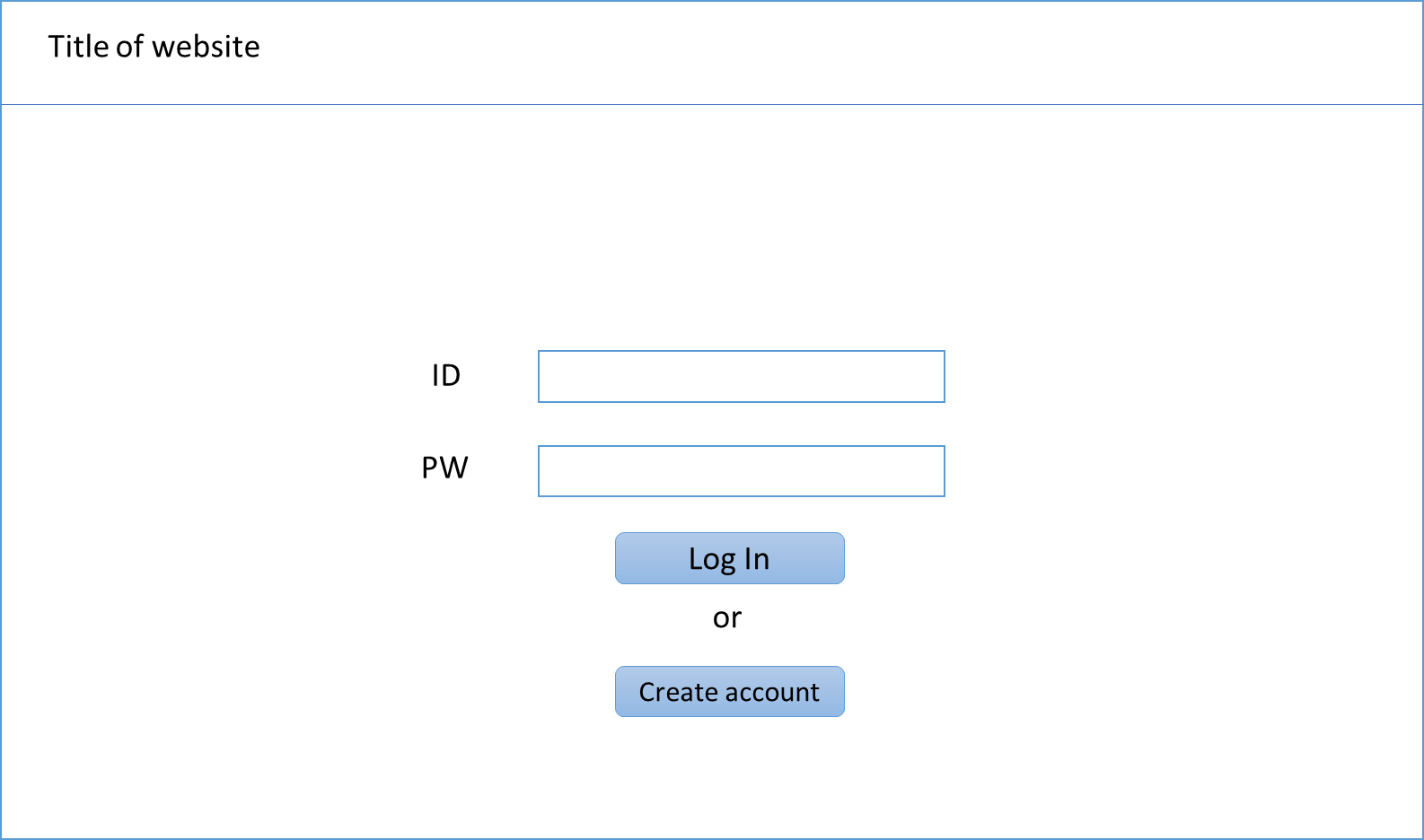


Figure 1 - Login Page

Login page is simple. Left top specifies what website here is. Users can enter ID and PW if the user has an existing account. If they are new, they can create a new account by clicking the “Create Account” button.

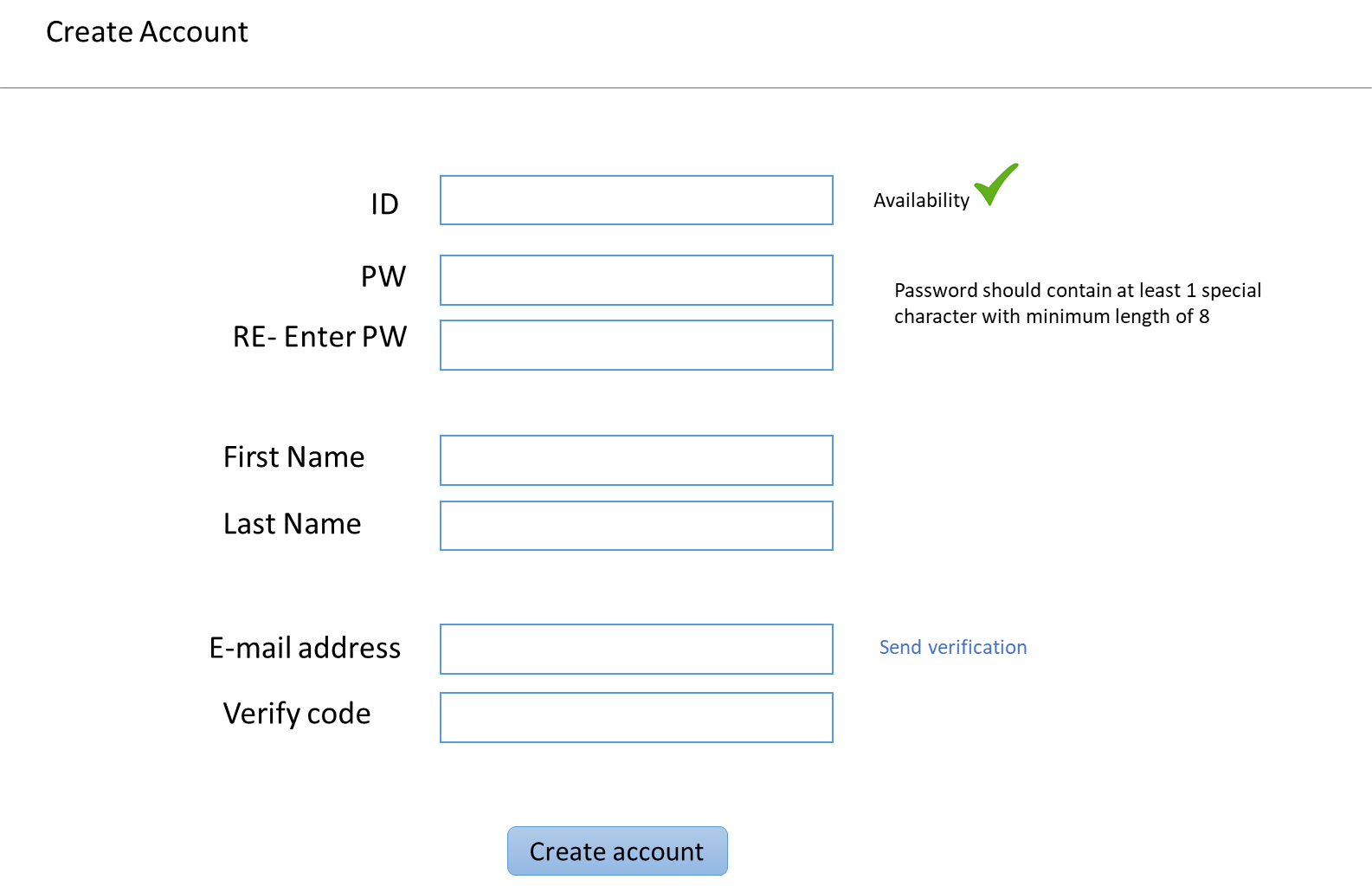


Figure 2 - Create Account

Figure 2 is a prototype of the Create Account page. Users can check if an ID already exists or not by putting the ID and clicking the Availability button. Password should meet the condition, and the user may enter the password again to double check and confirm. After entering first & last name, user send verification code to the email address and sync the email with the account.

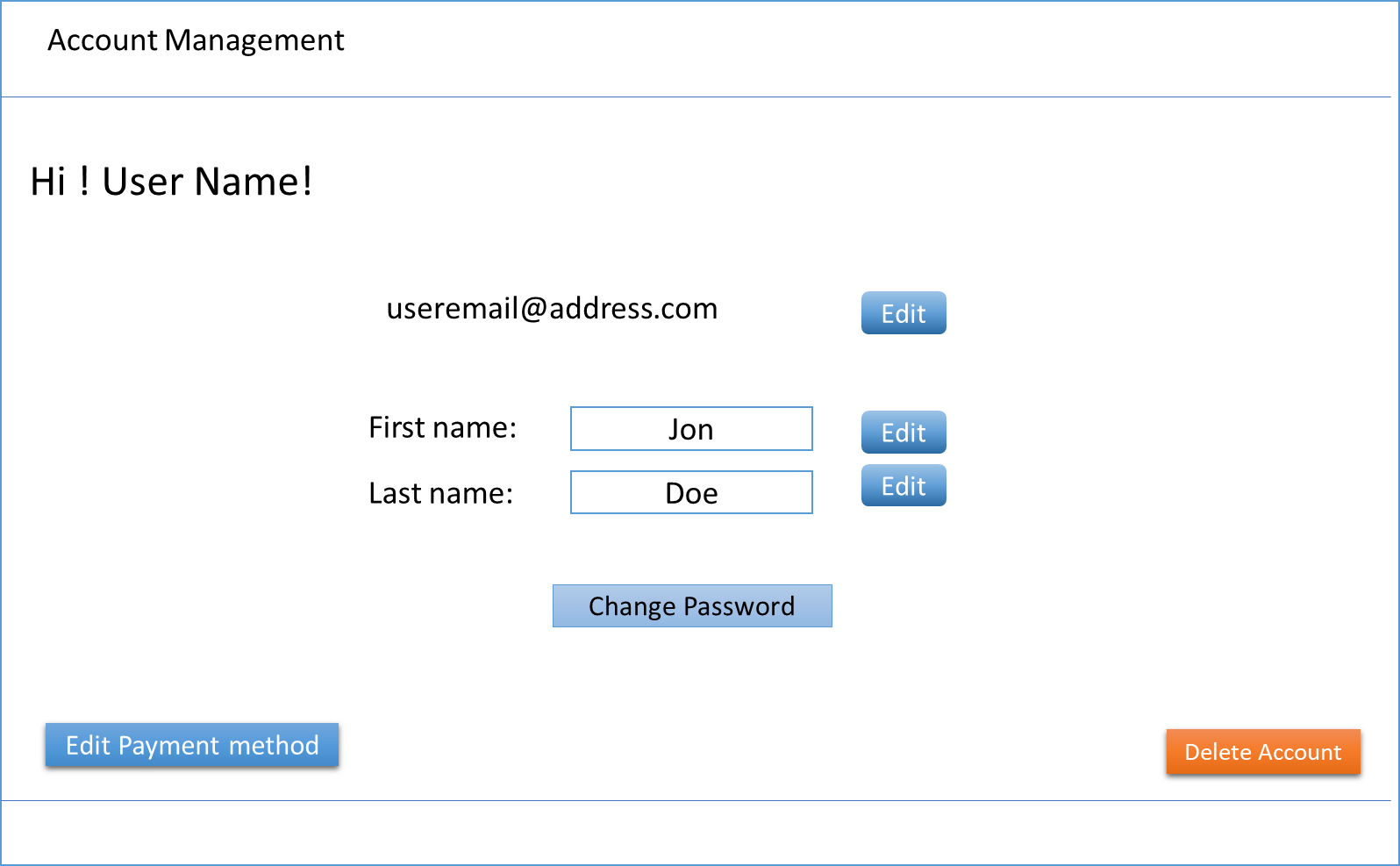


Figure 3 - Account Management

In the Account Management page, users may change their email address, first & l;ast name, and password. However, the user may not change its ID. Also this page is associated with the payment method page, so user may edit payment method at payment method page.

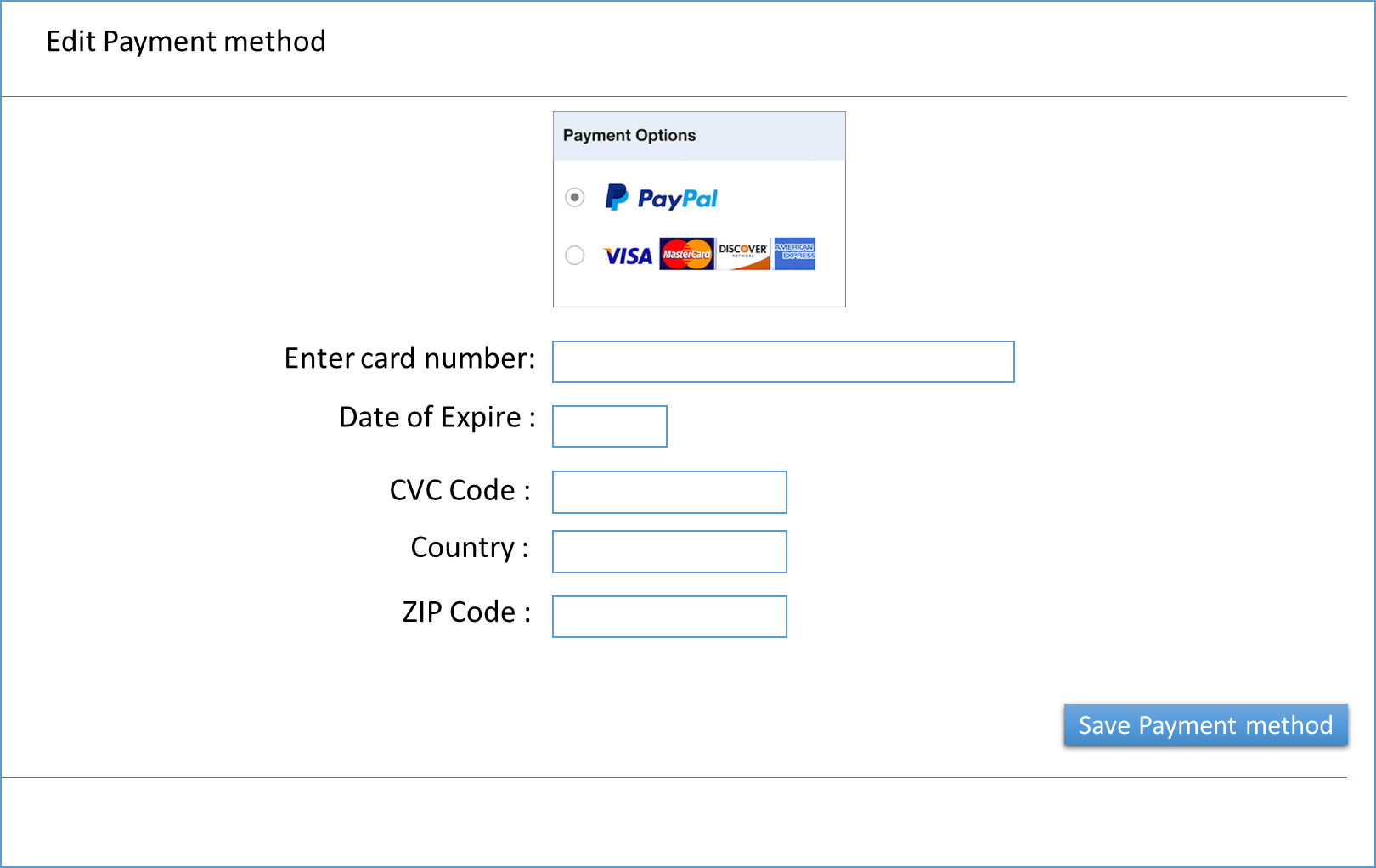


Figure 4 - Edit Payment Method

At this page, users can edit payment methods. Users can choose paypal or card for their payment method, and they provide appropriate information. Once user click save payment method button, system will save user payment info to the server

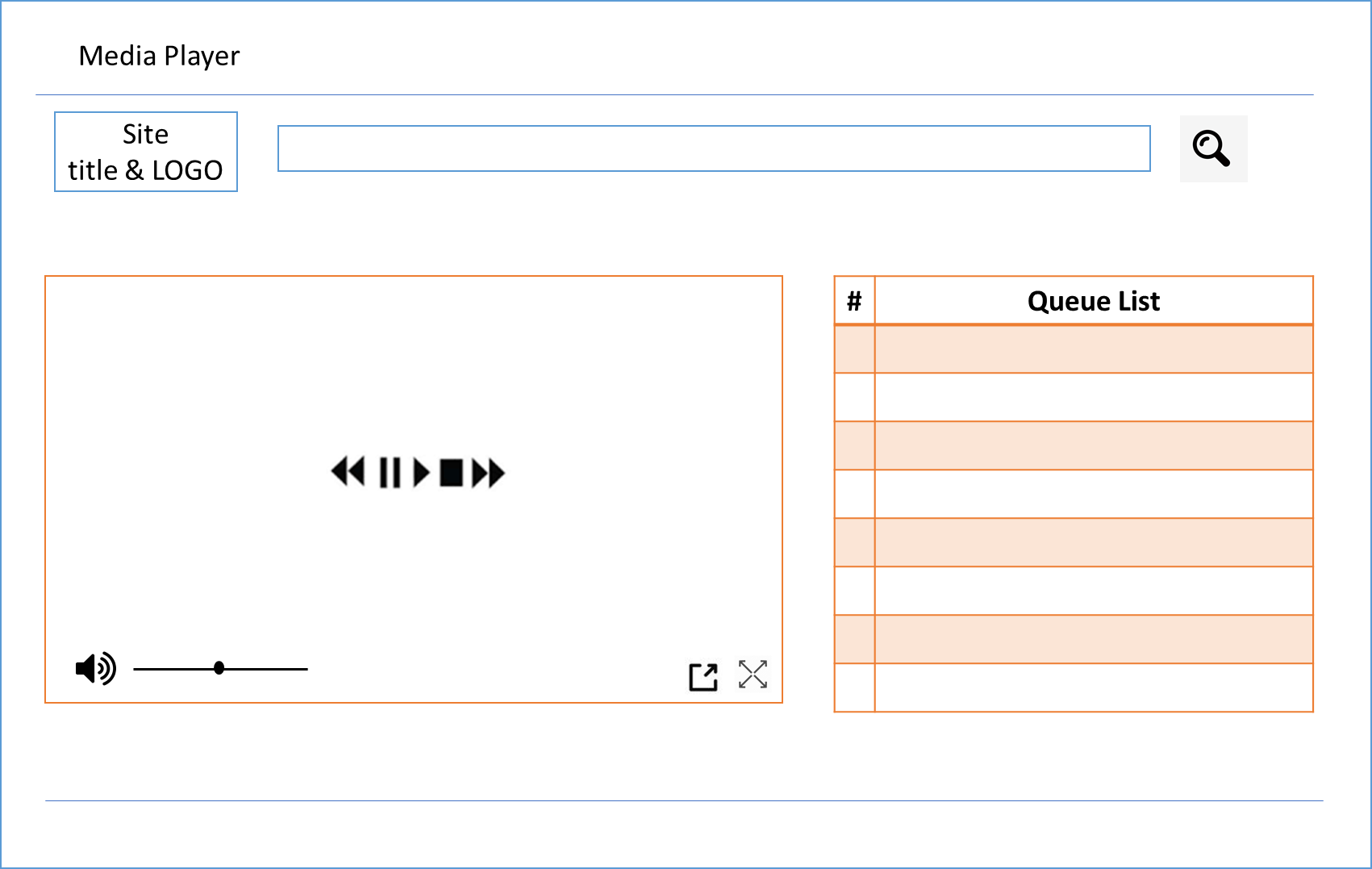


Figure 5 - Media Player

When a user plays a stream or any kind of sound, the user moves to this page. Users still can search other streams at the search bar. The box at the left represents a currently running stream. Users can play, pause, and stop the stream. Double arrows move front and back of the queue list. The symbols at the right bottom of the stream box are pop-up mini player and full screen feature.

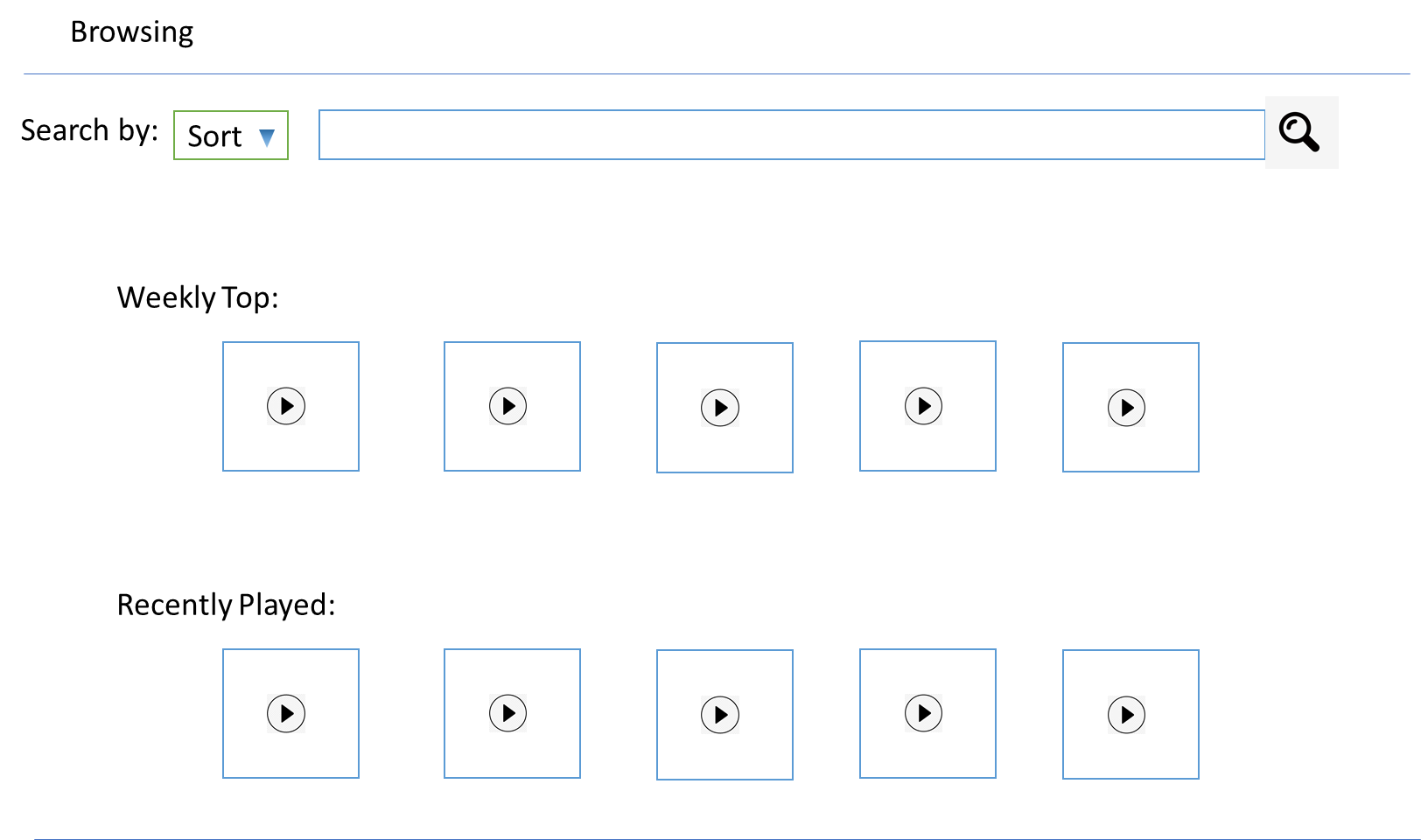


Figure 6 - Browsing

Users may search up any kind of stream at the browsing page. It displays weekly top streams which are selected by number of views. At the section underneath the Weekly Top, recent played streams are placed. Users can search any sound or stream at the top of the page, and the search result can be sorted by categories.

### External Clients Service Interfaces

N/A

## Functional Requirements

This section presents the system’s functional requirements.

Broadly, the project functional requirements are divided into 4 chunks, Account Management, Media Player, Browsing, and Payment Method. And for each chunk, specifies all detailed requirements and fundamentals of the system.

### Formal Requirements

Refer to JetStream Requirements Team 06.xlsx

## Non-Functional (Performance) Requirements

Refer to JetStream Requirements Team 06.xlsx

## Use Case Documents Overview

The streaming service has a variety of use cases. Each use case is unique and uses different systems within the application. For example, one set of use cases focuses specifically on the payment system and the use cases an actor may encounter.

| **Anshul Kumar, AXK190117** | |
| --- | --- |
| **Use Case ID** | **Use Case Description** |
| UC 1 | The user chooses from the payment methods available and then inputs their payment information. |
| UC 2 | The actor updates their payment information, either switching between credit card and PayPal or inputting new credit card information. |
| UC 3 | The actor clicks from the main page to view the transaction history. The system builds a table of the transaction history with dates and displays the table to the user. |
| UC 4 | The actor clicks on cancel payment plan, which then prompts the user to try other plans which maybe cheaper. If the user denies, then cancel the payment plan. |
| UC 5 | The system should automatically bill the customer monthly. With each month, the actor should be notified via text and/or email (depending on preference). If the actor wishes to manually bill, then they may switch the setting. |

| **Shane Arwood, SBA190007** | |
| --- | --- |
| **Use Case ID** | **Use Case Description** |
| UC 6 | User adds a song in the queue |
| UC 7 | User changes the sound output for the application. |
| UC 8 | User chooses to view song information. |
| UC 9 | User chooses to shuffle the queue of songs |
| UC 10 | User adjusts the size of the media player display. |

| **Hyungtaek Kim, HXK151830** | |
| --- | --- |
| **Use Case ID** | **Use Case Description** |
| UC 11 | User creates an account |
| UC 12 | User log-in with an account |
| UC 13 | User finds user ID |
| UC 14 | User changes user PW |
| UC 15 | User deletes account |

| **Nicholas Phan, NTP180002** | |
| --- | --- |
| **Use Case ID** | **Use Case Description** |
| UC 16 | User opens the program |
| UC 17 | User creates a playlist |
| UC 18 | User searches for a song to add to a playlist. |
| UC 19 | User moves to a previous page |
| UC 20 | User likes a song |

## Logical Database Requirements

One important piece of information that needs to be maintained persistently is whether the user is logged in or not. If the user is not logged in, then they should not be allowed to use the service. If the user is logged in and a server shuts down, then the user should not be negatively impacted. In addition, another piece of information to be maintained is the song currently being played. If the system does not maintain this information, then other services such as song recommendations may be impacted.

## Design Constraints

An industry standard that should be enforced in the design of the system is PCI compliance to ensure the security of using credit cards. In addition, the system should utilize servers tailored specifically for sending audio, such as servers CPU-optimized instead of GPU-optimized. A high amount of RAM is unnecessary. However, a large database for storing songs is required.

### Standards Compliance

As previously mentioned, since the system handles credit card transactions, PCI compliance is a must.

## System Quality Attributes

The purpose of this streaming service is to provide great music to customers. The best way to ensure this is to provide a highly available system with an emphasis on reliability so our users can enjoy music at any point without problems. Less emphasis should be placed on security, maintainability, and portability, although they are not to be overlooked.

### Reliability / Availability

The reliability of the system is paramount to ensure customers are satisfied. If the system is unavailable to clients, it is not life-critical since the system only deals with streaming. However, it will negatively impact our business and our brand. The high availability of the system will increase its fault tolerance, decreasing the likelihood of outages. The system should be available 24/7 for customers so they can listen to music at any time. If the system is not available, then there is a risk that if something goes wrong then the whole system will shut down, as there is no backup.

### Security

Security should be emphasized on account security and credit card security. Other than these two systems, security should be utilized (such as having firewalls in place into our systems to ensure only trusted entities may enter our systems), but is not the primary focus.

PCI compliance should be used since the system handles credit card transactions. In addition, account security, such as the storage of usernames and passwords, should also be encrypted and not stored in plain text.

### Maintainability

The system should be simple in design. Therefore, the maintainability of the system should be easier than the other aspects discussed. In addition, modifying code will be easy. New features can easily be added since the system is simple. Any elements the end user sees should be easily modifiable to give quality of life updates.

Furthermore, a database must be maintained in order to store a growing number of songs. This database is critical and maintenance on this may be harder than the rest of the systems. The customer’s payment information must also be maintained.

### Portability

The system should be able to be ported easily. Since the service should be available on multiple operating systems, such as Windows, macOS, iOS, and more, having a system which is easy to port will be extremely helpful. However, since the system is not open source, the system’s portability should be restricted to conditions we make.

# Supporting Information

N/A