Musa Streaming Service Design Specification

Produced by: *Lorenzo Ayala* & *Ali Musa*

March 8, 2024

CS 250 Introduction to Software Systems

Brian Donyanavard

System Description

1. Introduction and Overview

Overview

Musa Streaming Service's main goal is to provide a user-friendly service for individuals that will allow them to discover, save, and play music. Offering personalized playlists, seamless song discovery, and the ability to share what you're listening to with your friends. The interface should show user's its intuitive ability at a first glance and be easy to navigate.

Personalized Playlists

The algorithms crafted allow for us to generate tailored playlists based on the search and like history of the user. The goal of these playlists is to be able to not only show users songs that they may enjoy, but also give artists another chance to be heard by new audiences.

Song Discovery

Musa's Streaming Service's strong suit lies in the program's ability to take the information allowed from the user to create a personalized experience for them. Based on your likes and listening preferences, certain songs that are liked by people who enjoy some of the music you may enjoy can be recommended and shared to users.

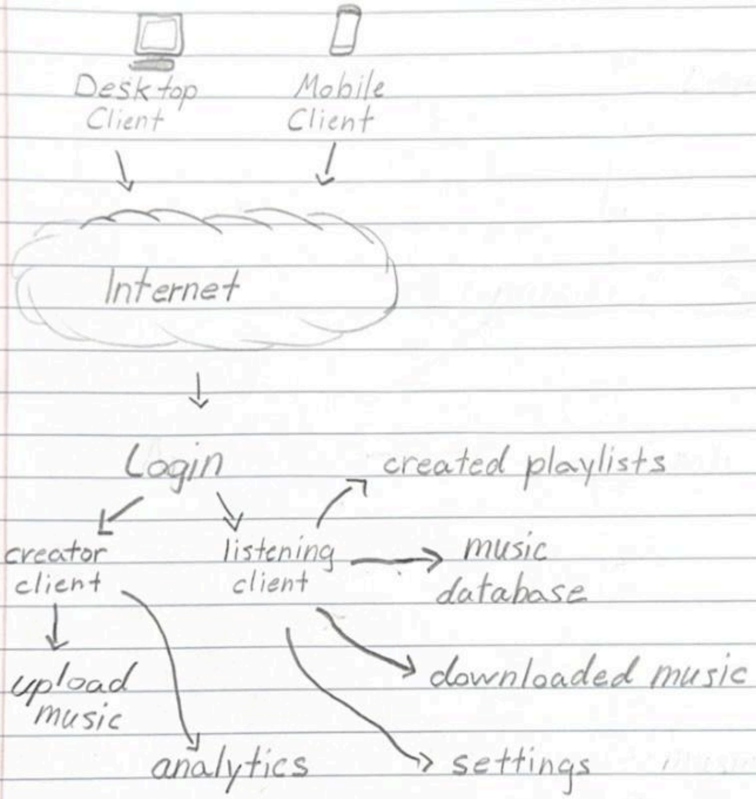
Social Sharing

Sharing your favorite songs and playlists with your friends and family, allows for groups of people to share, listen to, and promote other music, as well as creates a sense of community behind the service.

Software Architecture Overview

1. Architectural Diagram of all Components

Architectural diagram for Musa Streaming Service



created playlist / Upload Music / Database variables

- Artist string
- nameOfSong string
- mp3 file
- lengthOfSong double
- coverArt image
- search function string

Analytics

- monthlyListeners double
- totalStreams double

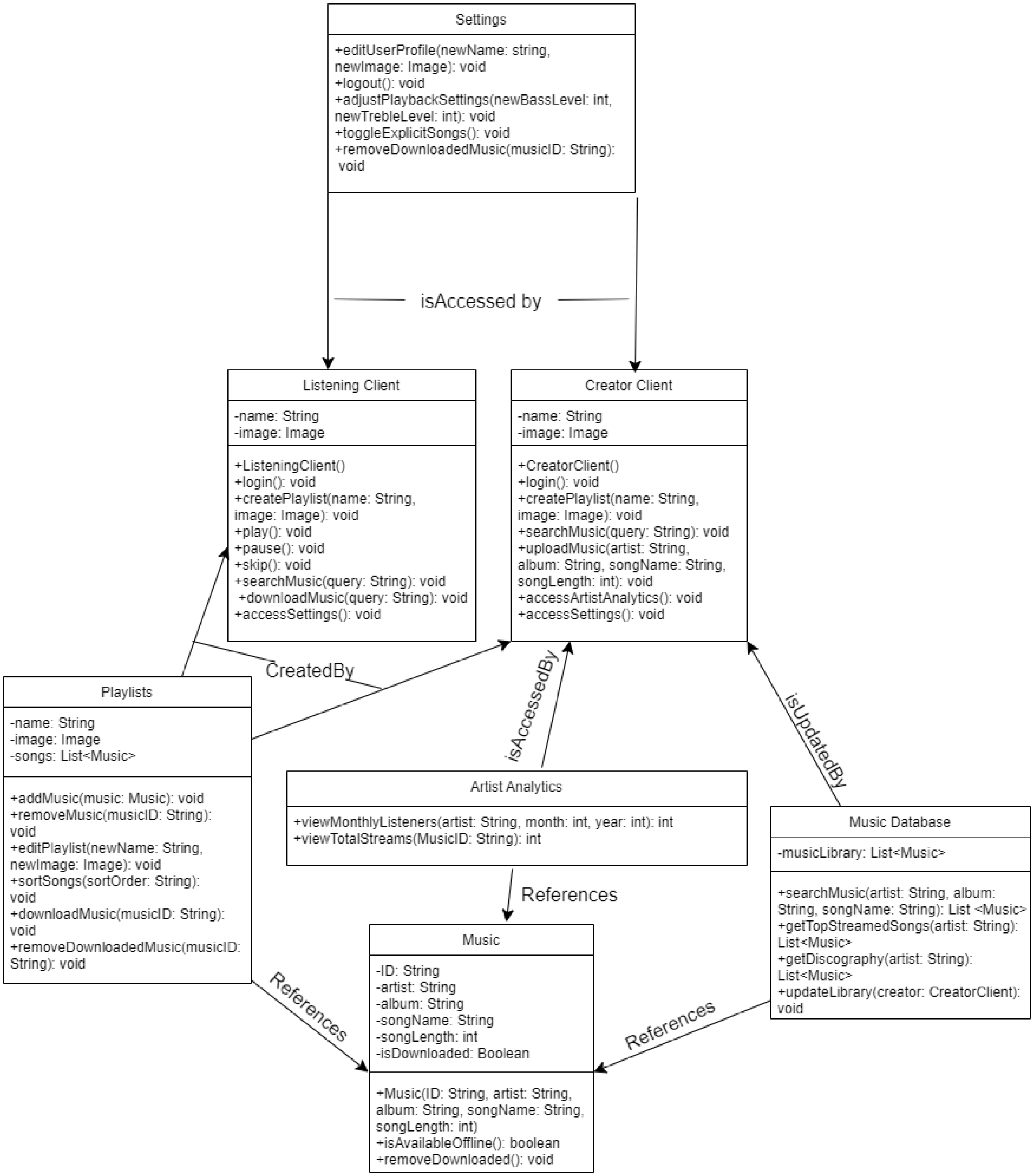
Settings

- userName string
- profilePicture Image
- explicitMusic Bool
- music playback *some type of scale

Playlists:

- playlistName string
- playlistImage image

2. UML Class Diagram


Settings
+editUserProfile(newName: string, newImage: Image): void
+logout(): void
+adjustPlaybackSettings(newBassLevel: int, newTrebleLevel: int): void
+toggleExplicitSongs(): void
+removeDownloadedMusic(musicID: String): void
isAccessed by
Listening Client
-name: String
-image: Image
+ListeningClient()
+login(): void
+createPlaylist(name: String, image: Image): void
+play(): void
+pause(): void
+skip(): void
+searchMusic(query: String): void
+downloadMusic(query: String): void
+accessSettings(): void
Creator Client
-name: String
-image: Image
+CreatorClient()
+login(): void
+createPlaylist(name: String, image: Image): void
+searchMusic(query: String): void
+uploadMusic(artist: String, album: String, songName: String, songLength: int): void
+accessArtistAnalytics(): void
+accessSettings(): void
CreatedBy
isAccessedBy
isUpdatedBy
Playlists
-name: String
-image: Image
-songs: List<Music>
+addMusic(music: Music): void
+removeMusic(musicID: String): void
+editPlaylist(newName: String, newImage: Image): void
+sortSongs(sortOrder: String): void
+downloadMusic(musicID: String): void
+removeDownloadedMusic(musicID: String): void
Artist Analytics
+viewMonthlyListeners(artist: String, month: int, year: int): int
+viewTotalStreams(MusicID: String): int
References
Music Database
-musicLibrary: List<Music>
+searchMusic(artist: String, album: String, songName: String): List <Music>
+getTopStreamedSongs(artist: String): List<Music>
+getDiscography(artist: String): List<Music>
+updateLibrary(creator: CreatorClient): void
Music
-ID: String
-artist: String
-album: String
-songName: String
-songLength: int
-isDownloaded: Boolean
+Music(ID: String, artist: String, album: String, songName: String, songLength: int)
+isAvailableOffline(): boolean
+removeDownloaded(): void
References
References

3. Classes, Attributes, and Operations

Descriptions should be detailed and specify datatypes, function interfaces, parameters, etc..

Classes:

- Listening client
- Creator client
- Settings
- Playlists
- Music database
- Music (general and downloaded)
- Artist Analytics

Attributes-

-User: Listening and Creator clients

- Listening-create Listener profile with name and image, login, create/edit Listener playlist with name and image, play/pause/skip music, search and download music from database to Listener system, access Settings
- Creator- create Artist profile with name and image, login, create/edit Artist playlist with name and image, search and upload music to database with Artist/album/song name and song length, access Artist analytics, access Settings

-Settings: edit User profile name and image, logout of User profile, adjust music playback settings such as bass/treble, toggle explicit songs, delete downloaded/uploaded music from User system

-Playlists: add/delete music, edit name and image, sort songs in alphabetical order by title/artist/album or chronological order by date added, download music

-Music database: holds music library and can be searched through by artist/album/song name, music organized by artist, within Artist we see top streamed songs and entire discography by date released, updated by Creator users who may add/remove music

-Downloaded music: downloaded to Listener system, available without wifi/data connection, downloaded music indicated by downloaded symbol

-Artist Analytics: view Creator monthly listeners, and total streams on each song

Operations-

- Listening Client Operations::

```
class ListeningClient {
  // Constructor for creating a listener profile
  ListeningClient(String name, Image image);
  // Method for logging in
  void login();
```

```
    // Method for creating/editing a listener playlist
    void createPlaylist(String name, Image image);
    // Methods for controlling music playback
    void play();
    void pause();
    void skip();
    // Method for searching music
    void searchMusic(String query);
// Method for downloading music
    void downloadMusic(String query);
    // Method for accessing settings
    void accessSettings();
}
```

- Creator Client Operations::

```
class CreatorClient {
    // Constructor for creating an artist profile
    CreatorClient(String name, Image image);
    // Method for logging in
    void login();
    // Method for creating/editing an artist playlist
    void createPlaylist(String name, Image image);
    // Method for searching music
    void searchMusic(String artist, String album, String songName, int songLength);
 // Method for uploading music
    void uploadMusic(String artist, String album, String songName, int songLength);
    // Method for accessing artist analytics
    void accessArtistAnalytics();
    // Method for accessing settings
    void accessSettings();
}
```

- Settings Operations::

```
class Settings {
    // Method for editing user profile
    void editUserProfile(String newName, Image newImage);
    // Method for logging out
    void logout();
    // Method for adjusting music playback settings
    void adjustPlaybackSettings(int newBassLevel, int newTrebleLevel);
    // Method for toggling explicit songs
    void toggleExplicitSongs();
    // Method for removing downloaded music from the user system
    void removeDownloadedMusic(String musicID);
}
```

- Playlists Operations::

```
class Playlists {
    // Method for adding music to the playlist
    void addMusic(Music music);
    // Method for deleting music from the playlist
    void deleteMusic(String musicID);
    // Method for editing playlist name and image
    void editPlaylist(String newName, Image newImage);
    // Method for sorting songs in the playlist
    void sortSongs(String sortOrder);
    // Method for downloading music from the playlist
    void downloadMusic(String musicID);
 // Method for removing downloaded music from the user system
    void removeDownloadedMusic(String musicID);
}
```

- Music(Download) Operations::

```
class DownloadedMusic {
    // Method for checking if music is available offline
    boolean isAvailableOffline(String musicID);
    // Method for removing downloaded music
    void removeDownloadedMusic(String musicID);
}
```

- Artist Analytics Operations::

```
class ArtistAnalytics {
    // Method for viewing monthly listeners for a creator
    int viewMonthlyListeners(CreatorClient creator, int month, int year);
    // Method for viewing total streams on a specific song
    int viewTotalStreams(String songID);
```

Development Plan and Timeline

Planning / Partitioning of Tasks

1. Before taking steps of action, we'll allocate our human resources to build a team, including designers, programmers, testers, and project managers.

2. Allocate hardware resources we'll need to create the streaming service.

3. Allocate budget range; communicate expected results with all teams/divisions of the project. Here we should finalize the software design specifications and create a timeline to abide by.

4. Monitoring division's processes', mitigating risks to ensure project requirements are met by their deadlines. There will be backend development, frontend development, and a content managing team to oversee maintenance of certain licenses and certificates to ensure our services from this streaming platform are legal. We also want to keep open communication with stakeholders that may be involved in the project.

5. Assure the quality of each piece of the project. Once we are ready to run this system, leaders of each team (Executive, Backend/Frontend Development, Content Management, Marketing, and Legal) will come together to deploy the service, execute test cases, and release our streaming service.

6. The tail end of our partition will end with the maintenance of our streaming service. Here we will maintain valid licenses and certifications to stream music through our software legally.

Timeline / Team Member Responsibilities

Key:

- Highlighted in blue: Frontend/Backend Development team's responsibilities
- Highlighted in green: Executive team's responsibilities
- Highlighted in yellow: Legal/Content Management team's responsibilities
- Highlighted in red: Marketing/Content Management team's responsibilities

Year 1- Foundation and Initial Development (Mar 2024)

Months 1-3: *Foundation, Inception, and Planning*

Months 4-6: *Requirements Analysis and Specification*

Months 7-9: *System Design and Architecture*

Months 10-12: *Prototyping and Iterative Development*

- Establish Musa Streaming Ltd., including legal and financial groundwork - Formulate the initial concept for Musa Streaming - Gather stakeholder input to define high-level requirements - Conduct marketing research to identify user needs and competitor	- Develop a detailed requirements document based on gathered information - Engage with potential users and stakeholders to refine and validate - Begin designing use cases and user stories	- Develop the system architecture based on the specified requirements - Design the database schema, considering scalability and data integrity - Create prototypes to visualize the user interfaces - Begin discussions with artists and labels for licensing rights	- Build a prototype of the Musa Streaming platform - Conduct usability testing and gather feedback for iterative development - Implement core functionalities of the platform

analysis			

Year 2: Testing/Feedback and Expansion/Content Acquisition

Months 13-15: *Beta Testing and Feedback*

Months 16-18: *Content Acquisition and Database Optimization*

Months 19-21: *Regional Expansion Preparation*

Months 22-24: *Internationalization and Localization*

- Launch a limited beta version for internal testing - Gather feedback from beta users to identify bugs and areas for improvement - Refine the system architecture based on performance and scalability testing	- Secure licensing agreements with major record labels and independent artists - Optimize the database structure for efficient storage and retrieval - Implement content management features for artists and labels	- Plan for official launch Musa Streaming in the USA, ensuring a smooth user experience - Develop localization features for language and regional preferences - Initiate marketing campaigns to build brand awareness - Begin preparations for international expansion	- Conduct market research in North America, Europe, South America, Asia, Africa, and Australia - Enhance security measures to protect user data - Implement language support and regional content options - Enhance user interfaces to accommodate diverse cultural expectations

Year 3 and beyond: Official Launch and Post-Launch Optimization

Months 25-27: *Regulatory Compliance and Security*

Months 28-30: *International Pre-launch Marketing and Load Testing*

Months 31-36: *Official International Launch and Monitoring*

Beyond: *Post-Launch Optimization*

- Ensure compliance with music licensing and privacy regulations/international data protection/ in the target countries - Conduct security audits and implement measures to protect user data - Obtain necessary approvals and licenses for each region	- Develop and execute marketing campaigns to build anticipation - Collaborate with local artists for exclusive content and promotions - Conduct load testing to ensure the platform can handle increased traffic	- Launch Musa Streaming officially all throughout North America - Celebrate the milestone with promotional events, partnerships, and exclusive content releases - Monitor user feedback, system performance, user engagement, and address any initial challenges	- Continuously optimize the platform based on user feedback and emerging trends - Implement targeted marketing strategies to grow the user base - Explore additional features, partnerships, and prepare for launch in Europe, Africa, Asia, South America, Australia.

We plan to complete and launch Musa Streaming in every marketable region by the end of 2027.