1. Introduction 3
1.1. Purpose 3
1.2. Document Conventions 3
1.3. Definitions, Acronyms, and Abbreviations 3
1.4. Intended Audience 4
1.5. Project Scope 5
1.6. Technology Challenges 5
1.7. References 6
2. General Description 7
2.1. Product Features 7
2.2. User Class and Characteristics 7
2.3. Operating Environment 8
2.4. Constraints 8
2.5. Assumptions and Dependencies 8
3. Functional Requirements 10
3.1. Primary 10

1. Introduction

1.1. Purpose

TuneTribe is a social platform designed for music enthusiasts and artists alike. Users can share their favorite songs, create music-related posts, comment on friends' posts, and recommend songs/artists to each other. Artists have a dedicated space to upload their music, gain a following, and track royalties. The platform fosters community interaction while providing exposure for emerging artists. Moderators ensure a positive user experience by managing comments and addressing any issues. Admins oversee the platform's operations and have the authority to ban users or artists if necessary.

1.2. Document Conventions

The purpose of this Software Requirements Document (SRD) is to describe the client-view and developer-view requirements for the TuneTribe social web app. Client-oriented requirements describe the system from the client's perspective. These requirements include a description of the different types of users served by the system. Developer-oriented requirements describe the system from a software developer's perspective. These requirements include a detailed description of functional, data, performance, and other important requirements.

1.3. Definitions, Acronyms, and Abbreviations

Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build the Restaurant Manager.
CSS	CSS stands for Cascading Style Sheets, and it's a style sheet language used for describing the presentation of a document written in a markup language like HTML.
JavaScript	JavaScript is a high-level, interpreted programming language used primarily to add interactivity and dynamic behavior to web pages.
MySQL	Open-source relational database management system.
.HTML	Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content.
SpringBoot	An open-source Java-based framework used to create a micro Service. This will be used to create and run our application.
MVC	Model-View-Controller. This is the architectural pattern that will be used to implement our system.
Spring Web	Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system.
Thymeleaf	A modern server-side Java template engine for our web environment. This is one of the dependencies of our system.
NetBeans	An integrated development environment (IDE) for Java. This is where our system will be created.

API	Application Programming Interface. This will be used to implement a function within the software where the current date and time is displayed on the homepage.

1.4. Intended Audience

Intended Audience:

1. Stakeholders:
   - Investors or funding agencies
   - Executives or decision-makers within the organization
   - Legal advisors or regulatory bodies involved in compliance matters

   Relevant Sections: Entire SRS document for understanding project objectives, scope, and compliance requirements.

2. Developers:
   - Architects
   - Software Engineers
   - Frontend and Backend Developers

   Relevant Sections:
   - System architecture and design details
   - Technical specifications for frontend and backend components
   - API documentation
   - Database schema and data models
   - Coding standards and guidelines

3. **Project Managers**:
   - Project Managers
   - Team Leads

   Relevant Sections:
   - Project objectives and scope
   - Timeline and milestones
   - Resource allocation and budgetary considerations
   - Risk assessment and mitigation strategies
   - Quality assurance and testing plans

4. **Users**:
   - End-users (music enthusiasts and artists)
   - Moderators
   - Administrators

   Relevant Sections:
   - User interface specifications
   - Functional requirements related to user interactions (e.g., sharing songs, commenting)
   - Guidelines for moderators (e.g., managing comments, addressing issues)
   - Admin panel functionalities and permissions (e.g., user banning, content moderation tools)

By specifying the intended audience for each part of the SRS document, stakeholders, developers, project managers, and users can easily locate the information relevant to their roles and responsibilities, facilitating effective communication and collaboration throughout the project lifecycle.

1.5. Project Scope

Reducing the amount of time that a customer needs to wait; therefore, increasing the amount of customers that are able to be served in the restaurant within a day.
The goal of TuneTribe is to create a social platform that fosters interaction between music enthusiasts and artists. This aligns with the overall business goals of providing a vibrant community space for sharing music and supporting emerging artists.

The benefits of the project to business include:

- Facilitating community engagement: TuneTribe provides a platform for users to share their favorite music, interact with fellow enthusiasts, and discover new artists, thereby enhancing user engagement and loyalty.
- Supporting emerging artists: By offering artists a dedicated space to upload their music and gain a following, TuneTribe helps to promote and support emerging talent, fostering a dynamic music ecosystem.
- Driving user growth and retention: The platform's emphasis on community interaction and music discovery enhances user satisfaction, driving user growth and increasing user retention over time.
-

1.6. Technology Challenges

Technology Challenges for a Web App:

1. Frontend Performance: Ensuring fast load times and smooth user experience by optimizing frontend code, assets, and rendering processes.

2. Browser Compatibility: Ensuring compatibility with various web browsers (e.g., Chrome, Firefox, Safari) and versions to provide a consistent experience across different platforms.

3. Responsive Design: Implementing a responsive design to ensure the web app adapts seamlessly to various screen sizes and devices, including desktops, tablets, and smartphones.

4. Data Security: Implementing robust security measures to protect user data, including encryption, secure communication protocols (HTTPS), and protection against common web vulnerabilities (e.g., XSS, CSRF).

5. Backend Scalability: Designing a scalable backend architecture capable of handling increasing user traffic and data volume without compromising performance or reliability.

6. Real-time Communication: Implementing real-time communication features such as instant messaging or live updates using technologies like WebSockets or server-sent events.

7. Offline Functionality: Providing offline functionality where possible to allow users to access certain features or content even when they're not connected to the internet.

8. Accessibility: Ensuring the web app is accessible to users with disabilities by following accessibility guidelines (e.g., WCAG) and implementing features like keyboard navigation and screen reader compatibility.

9. Search Engine Optimization (SEO): Optimizing the web app for search engines to improve visibility and discoverability, including proper HTML markup, meta tags, and structured data.

10. Continuous Deployment: Implementing a streamlined deployment process with continuous integration/continuous deployment (CI/CD) pipelines to automate testing, deployment, and updates, ensuring rapid and reliable releases.

1.7. References

2. General Description

2.1. Product Features

The product features include the ability for users, artists, and moderators to create an account and the ability for administrators to manipulate those accounts. Users can create posts sharing the music they are listening to, recommend their favorite artists to their friends and discover new music through the artist discovery feature. Users will have their top artists tracked and displayed on their page. Artists are provided with dedicated spaces to upload their music, gain a following, and track royalties. Moderators can delete comments and posts that do not follow community guidelines, they can ban users for up to a week (temp ban). Moderators can also limit comments from a specific user who is not following community guidelines. If a user is a repeat offender, moderators can request to have the user permanently banned. An admin will be able to ban users, and artists, they will also have the ability to remove moderator and artists permissions.

2.2. User Class and Characteristics

User Class and Characteristics:

TuneTribe, as a website or web application, is designed to cater to users with diverse interests and roles within the music community. Users can be categorized into the following classes based on their characteristics:

1. **Music Enthusiasts**:
   - Music enthusiasts are individuals who have a passion for music and enjoy discovering new songs, artists, and genres.
   - They actively engage with the platform by sharing their favorite songs, creating music-related posts, and recommending songs/artists to others.
   - Music enthusiasts value features that facilitate music discovery, community interaction, and sharing their musical preferences with others.

2. **Casual Listeners**:
   - Casual listeners are users who enjoy listening to music casually without deep involvement in the music industry or community.
   - They may use the platform to explore new music, discover trending songs, and connect with friends through music-related interactions.
   - Casual listeners appreciate user-friendly features and intuitive interfaces that simplify music discovery and enhance their listening experience.

3. **Serious Music Fans**:
   - Serious music fans are individuals who have a deeper level of engagement with music, including a strong interest in specific genres, artists, or music scenes.
   - They actively participate in discussions, share insights, and engage with other users who share similar musical tastes and interests.
   - Serious music fans value platforms that provide opportunities for meaningful interactions, discovery of niche music, and connections with like-minded individuals.

4. **Emerging Artists**:

- Emerging artists are aspiring musicians who use the platform to showcase their music, gain exposure, and connect with fans.
- They utilize the dedicated space provided by the platform to upload their music, build a following, and track royalties.
- Emerging artists seek features that support their growth and development as musicians, including tools for promoting their music, engaging with fans, and tracking their success on the platform.

5. **Moderators and Administrators**:
- Moderators and administrators play a crucial role in ensuring the smooth operation of the platform and maintaining a positive user experience.
- They oversee user interactions, manage comments, address issues, and enforce community guidelines to foster a safe and welcoming environment for all users.
- Moderators and administrators require access to specialized tools and privileges to effectively perform their duties and maintain platform integrity.

By catering to the needs and preferences of these user classes, TuneTribe aims to create a vibrant and inclusive community where music enthusiasts and artists can connect, discover, and share their passion for music with others.

Our website application does not expect our users to have any prior knowledge of a computer, apart from using a web browser, or knowledge of astronomy. Our website application has removed the need for them to have astronomy, math, or science knowledge and allows the user to focus on exploring the night sky.

2.3. Operating Environment

The application is designed to operate on the web across many different devices.

2.4. Constraints

Due to TuneTribe being a webapp, fluent mobile access may be limited. Interaction with website is already limited on mobile.

2.5. Assumptions and Dependencies

The software will be dependent on Spring Web in order to create and execute the MVC architecture that will be developed within NetBeans, VS Code, and Spring. These technologies will facilitate the development of a robust and scalable web application, providing essential features for handling client requests, managing server-side logic, and rendering dynamic HTML content. NetBeans and VS Code will serve as the primary integrated development environments (IDEs) for coding and debugging, offering extensive tooling support and seamless integration with the Spring framework. Additionally, Spring's comprehensive ecosystem of libraries and tools will enable efficient development, testing, and deployment of the software, ensuring high performance and reliability throughout the development lifecycle.

The software will be dependent on Spring Web and Thymeleaf in order to create and execute the MVC architecture that will be developed within NetBeans, Vs Code, Spring, .

3. Functional Requirements

3.1. Primary

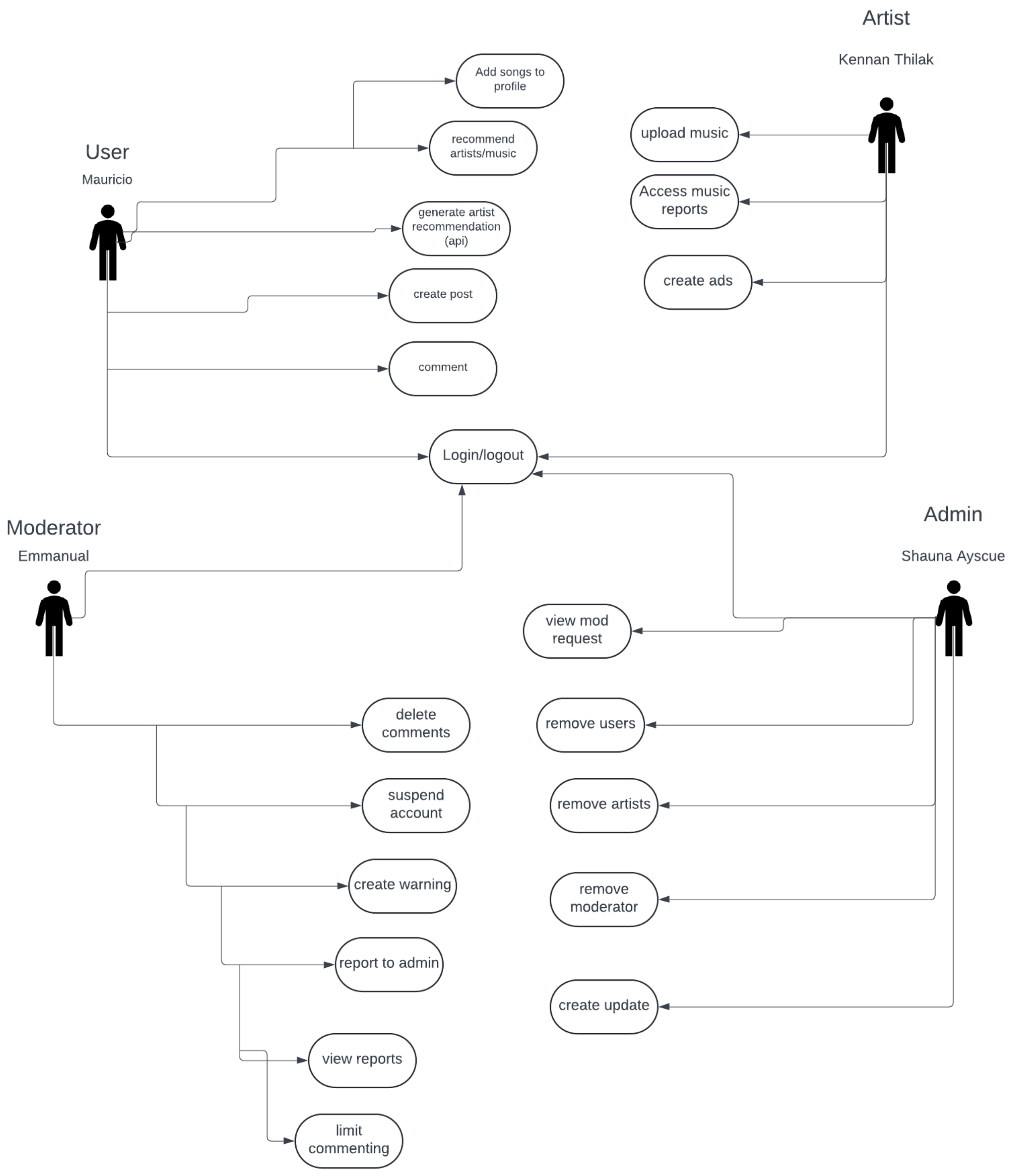
-The System will allow the user to comment on other users' posts.
-The System will allow the user to lookup other user information and see the other user's posts
-The System will allow the user to search for songs based on Song or artist name
-The system will allow users to react to other people's post using emojis
-The system will allow users to create and login to their accounts using their personal information
-The System will allow users with artist partnership to input their song information which will boost their post and their music within the app
-System will allow users to issue temporary and permanent bans, comment restrictions,

3.2. Secondary

- Password protection for information only accessible to employees, managers, and each individual table.
- Authorization scheme so that users can only see their friends and no other users posts, assuming profile is private
- System to keep track of users who have been banned or limited from

Use-Case Model

3.2.1. Use-Case Model Diagram



3.2.2. Use-Case Model Descriptions

3.2.2.1. Actor: User (Mauricio)

- **Generate Artist Recommendation**:
  - **User can request an artists recommendation based on the artists they listen to**
- **Comment**:
  - Users can comment on other users posts
- **Create Post**:
  - **Users can create a post**

- **Post will consist of**
    - **an emoji (how they are feeling)**
    - The song itself
- **Recommend Artists/Music**:
  - **Users can recommend a song/artist to any of their friends**.
  - **This will function like Poke on facebook where users will receive a notification**
- **Add Songs to Profile:**
  - **Users can add songs they like to their profile for others to see.**
  - **Up to 5 songs**
- **Login:**
  - **User will be able to log into their accounts and sync across other browsers**

3.2.2.2. Actor: Admin (Shauna)

- **Ban Users**: The admin can prohibit users from accessing TuneTribe due to violations of community guidelines or inappropriate behavior.
- **Ban Artists**: Artists who breach terms of service or upload unauthorized content can be banned from TuneTribe by the admin, resulting in the removal of their content and account restrictions.
- **Remove Moderator Permission**: The admin has the authority to revoke moderator privileges from users who no longer meet criteria or violate moderator guidelines, limiting their access to moderation tools.
- **Remove Artist Permission**: If a user is inactive or violates artist guidelines, the admin can withdraw artist permissions, restricting their ability to upload music or manage their artist profile.
- **Login**: Admins can access the admin dashboard by logging in with valid credentials, enabling them to perform administrative tasks and manage the TuneTribe platform.

3.2.2.3. Actor: Artist (Kannan)

- Access Music Reports: Artists can access detailed reports about the performance of their music, including streaming statistics, demographic insights, and geographical reach.
- Upload Music to App: Artists can upload their music to the application, making it available for streaming and discovery by users.
- Login: Artists can log into their accounts to manage their profile, upload new music, access reports, and engage with their audience. Their login credentials will allow them to sync their account across different browsers for seamless access.

3.2.2.4. Actor: Moderator (Emmanuel)

- **Delete Comments**: The moderator has the authority to remove comments made by users and artists deemed inappropriate or violating community guidelines.
- **Temporary Ban Users**: Moderators can impose temporary bans on specific user accounts for a predefined duration in response to rule violations or disruptive behavior.

- **Limit Users Commenting**: Moderators can restrict users' commenting capabilities if they engage in spamming or post inappropriate content, ensuring a conducive environment for discussion.
- **Request to Ban User**: In cases of severe policy violations, moderators have the ability to submit requests for the permanent ban of users, subject to approval by higher authorities or administrators.
- **Submit Reports of Users**: Moderators enable users to report instances of rule violations or misconduct by other users, facilitating community-driven enforcement of platform policies.
- **Login**: Moderators can access their moderator dashboard and perform moderation tasks upon successful authentication through the login process.

3.2.3. Use-Case Model Scenarios

3.2.3.1. Actor: User (Mauricio)

- **Use-Case Name**: Generate Artists Recommendation
  - **Initial Assumption**: User has access to artists recommendation and can have a song recommended to them
  - **Normal**: The user is able to generate an artists recommendation based on the songs on the user's profile
  - **What Can Go Wrong:** User does not have any artists/songs added to their profile
  - **Other Activities**: User may be asked to add songs to their profile
  - **System State on Completion**: A new page is loaded along with a new artist for the user to listen to.
- **Use-Case Name**: Comment
  - **Initial Assumption**: The user has access to commenting feature
  - **Normal**: The user comments on another user's post
  - **What Can Go Wrong**: User is restricted from commenting
  - **Other Activities**: N/A
  - **System State on Completion**: A comment is added to intended user's post.
- **Use-Case Name**: Create post
  - **Initial Assumption**: User has access create post feature
  - **Normal**: User will create a post which will consist of an emoji and a song
  - **What Can Go Wrong:** The song the user is looking for is not found
  - **Other Activities**: The user can check for spelling errors
  - **System State on Completion**: A new post is added to the user's timeline
- **Use-Case Name**: Login
  - **Initial Assumption**: The user has access to login feature
  - **Normal**: The user logs in an they have access to their account
  - **What Can Go Wrong:** The users credentials cannot be validated
  - **Other Activities**: User is redirected to enter credentials again or to create an account
  - **System State on Completion**: The user will have access to the web app
- **Use-Case Name**: Recommend song
  - **Initial Assumption**: User has access to recommend an artist feature

- **Normal**: User recommends a song to a friend
- **What Can Go Wrong:** The requested song is not found
- **Other Activities**: search again
- **System State on Completion**: Prompt is given that the song has been recommended

- **Use-Case Name**: Add Song to profile
  - **Initial Assumption**: User has access to this feature
  - **Normal**: The user will be able to add up to 5 songs to their profile for display purposes
  - **What Can Go Wrong:** The song the user is looking for is not found
  - **Other Activities**: prompt user to search again
  - **System State on Completion**: The requested song will appear on the users profile

3.2.3.2. Actor: Moderator (Emmanuel)

- **Use Case Name: Delete Comments**
  - **Initial Assumption**: The moderator is logged into the moderation dashboard and has the necessary permissions to delete comments.
  - **Normal**: The moderator identifies a comment that violates community guidelines, selects the comment, and deletes it from the platform.
  - **What Can Go Wrong**: Accidental deletion of a valid comment due to a misclick or misunderstanding of the guidelines. Technical issues prevent the moderator from accessing the moderation dashboard or deleting comments.
  - **Other Activities**: Reviewing multiple comments before deciding which ones to delete. Communicating with users regarding the deletion of their comments.
  - **System State on Completion**: The deleted comment is removed from the platform, and the moderation log is updated with details of the action.

- **Use Case Name: Temporary User Ban**
  - **Initial Assumption**: The moderator is logged into the moderation dashboard and has the authority to impose temporary bans on users.
  - **Normal**: The moderator identifies a user who has violated community guidelines, selects the user, and imposes a temporary ban for a specified duration.
  - **What Can Go Wrong**: Accidentally banning the wrong user due to a selection error. Technical issues prevent the moderator from accessing the moderation dashboard or applying the ban.
  - **Other Activities**: Reviewing the user's activity history before deciding on the ban duration. Communicating with the banned user regarding the reason and duration of the ban.
  - **System State on Completion**: The banned user is restricted from accessing the platform for the specified duration, and the moderation log is updated with details of the ban.

- Use Case Name: **Limit User Commenting**
    - **Initial Assumption**: The moderator is logged into the moderation dashboard and can restrict users' commenting capabilities.
    - **Normal**: The moderator identifies a user who is engaging in spamming or posting inappropriate comments, selects the user, and limits their commenting capabilities.
    - **What Can Go Wrong**: Mistakenly limiting commenting capabilities of a user who has not violated guidelines. Technical issues prevent the moderator from accessing the moderation dashboard or applying the restriction.
    - **Other Activities**: Communicating with the user regarding the reason for limiting their commenting capabilities. Monitoring the user's activity to ensure compliance with platform guidelines.
    - **System State on Completion**: The user's commenting capabilities are restricted, and the moderation log is updated with details of the restriction.

3.2.3.3. Actor: Artist (Kannan)

- **Use-Case Name**: Upload Music
    - **Initial Assumption**: The artists has access to the TuneTribe platform and is logged in as an artist.
    - **Normal**: The artists navigates to the upload music section and selects the music file from his device. He fills in the required details such as song title, artist name, genre, and album information. Kennan then confirms the upload, and the system processes the music file.
    - **What Can Go Wrong**: If the artists attempts to upload a music file in an unsupported format. The system should display an error message indicating the supported file formats and prompt Kennan to upload a compatible file.
    - **Other Activities**: The artists can edit the details of the uploaded music later, including updating the title, genre, or album information.
    - **System State on Completion**: The uploaded music is successfully added to Artist's profile on the platform, where it is available for other users to discover and listen to.

- **Use-Case Name**: Access Music Reports
    - **Initial Assumption**: The artist has access to the TuneTribe platform and is logged in.
    - **Normal**: the artist navigates to the music reports section to view detailed reports on the performance of their uploaded music. They can analyze metrics such as streams, downloads, likes, and comments for each track. The reports may also include insights into audience demographics and engagement patterns.
    - **What Can Go Wrong**: The artist encounters discrepancies or inaccuracies in the reported music performance metrics. The system should provide a mechanism for the artist to report the issue and request further investigation by platform administrators. Additionally, technical issues such as data synchronization errors or server downtime may affect the availability or accuracy of the reports.

- **Other Activities**: The artist can download the music reports for their records or export them in various formats for further analysis. They may also share the reports with collaborators or stakeholders to demonstrate the success and impact of their music on the platform.
- **System State on Completion**: The artist has access to comprehensive and reliable information on the performance of their music uploads, enabling them to make informed decisions, track their success, and optimize their strategies for maximizing engagement and revenue on the platform.

3.2.3.4. Actor: Admin (Shauna)

- **Ban Users**:
  - **Initial Assumption**: The admin has access to ban a user from TuneTribe if policies are violated.
  - **Normal**: The admin will select a certain user to ban and prevent them from commenting or posting on the webapp.
  - **What Can Go Wrong**: The admin could accidently ban the wrong user.
  - **Other Activities**: The admin could ban users from commenting on users post.
  - **System State on Completion**: The banned user is deleted from the platform with the admin logs being updated.
- **Ban Artists**:
  - **Initial Assumption**: The admin has access to ban an artist from TuneTribe if policies are violated.
  - **Normal**: The admin determines what artist has violate the guidelines of the webapp and selects the artist to be banned.
  - **What Can Go Wrong**: The admin could accidently permanently ban the wrong artists due to an error in the system.
  - **Other Activities**: Reviewing the artist that has been banned by searching their history.
  - **System State on Completion**: The selected banned artists are restricted from accessing the webapp permanently.
- **Remove Moderator Permissions**:
  - **Initial Assumption**: The admin has the authority to remove the moderator's permission on users and artists.
  - **Normal**: The admin can not allow a moderator to delete comments, ban users, limit commenting, and look at user reports.
  - **What Can Go Wrong**: The admin could choose the wrong task to remove from the moderator.
  - **Other Activities**: The admin could re-add a moderator that has been previously not allowed to perform a certain task.
  - **System State on Completion**: The moderators tasks are restricted and the admin's log is up to date.
- **Remove Artists Permissions**:
  - **Initial Assumption**: The admin has the authority to remove an artists at anytime if they violate any policies.
  - **Normal**: The admin can restrict an artist from uploading songs for a certain amount of time or commenting on user accounts.

- **What Can Go Wrong**: The admin could restrict the wrong artist due to misunderstanding of guidelines.
- **Other Activities**: The admin could contact the artist and determine the reasons for restricting the artist.
- **System State on Completion**: The artists tasks are restricted and the Admin’s log is up to date.

4. Technical Requirements

4.1. Interface Requirements

4.1.1. User Interfaces

Users will have access to:

- Side menu bar
    - Profile
    - Friends (Circle)
    - Settings
- Search bar
    - Here they can search
        - Other users
        - Songs
- Create post button
    - Users will be prompted to include
        - An emoji (how they are feeling)
        - A song
- Comment button
    - Users will be able to leave a short comment on their or a friends post

4.1.2. Hardware Interfaces

The web application will run on any hardware device that has access to the internet, the ability to display webpages, and the ability to interact with web pages. This includes, but is not limited to, smartphones, tablets, desktop computers, and laptops.

4.1.3. Communications Interfaces

It must be able to connect to the internet as well as the local database on phpMyAdmin. The communication protocol, HTTP, must be able to connect to the World Time API and return the current date and time.

4.1.4. Software Interfaces

We will use React and Spring Boot ThymeLeaf to help build the frontend, as well as JPA for the backend database functionality. We will also use Spring Boot with Java to connect the frontend to the backend. We will also be using a MySQL DataBase to store all data.

5. Non-Functional Requirements

5.1. Performance Requirements

-NFR0(R): The TuneTribe platform should maintain optimal memory usage, ensuring that the local copy of the music database consumes less than 50 MB of memory.

-NFR1(R): The system, including all components and databases, should operate efficiently within memory constraints, consuming less than 100 MB of memory under normal usage conditions.

-NFR2(R): Novice users should be able to browse, search for songs, and create playlists within the TuneTribe platform in less than 10 seconds.

-NFR3(R): Expert users, familiar with the platform's interface, should be able to perform advanced tasks such as uploading music, managing their artist profile, and engaging with the community in less than 30 seconds.

5.2. Safety Requirements

- Safeguards should be implemented to protect user data and privacy, including encryption of sensitive information and secure authentication mechanisms.
- Measures must be in place to prevent unauthorized access to user accounts and ensure the integrity of user-generated content.
- Banning Users to uphold the community’s enviornment and integrity.

5.3. Security Requirements

- NFR4(R): Access to the TuneTribe platform should be restricted to authorized users and their specified roles only, requiring valid authentication credentials for login and access to system functionalities.

5.4. Software Quality Attributes

5.4.1. Availability

The platform should strive for high availability, minimizing downtime and ensuring uninterrupted access for users.

5.4.2. Correctness

All functionalities and data processing should be accurate and error-free to maintain the integrity of the platform.

5.4.3. Maintainability

The codebase should be well-structured and documented, facilitating easy maintenance and updates by developers.

5.4.4. Reusability

Common components and functionalities should be designed for reusability across different modules to promote efficient development and reduce redundancy.

5.4.5. Portability

The platform should be compatible with various devices, browsers, and operating systems, ensuring a consistent user experience across different platforms.

5.5. Process Requirements

5.5.1. Development Process Used

Scrum

5.5.2. Time Constraints

- We are given two months to complete the project

5.5.3. Cost and Delivery Date

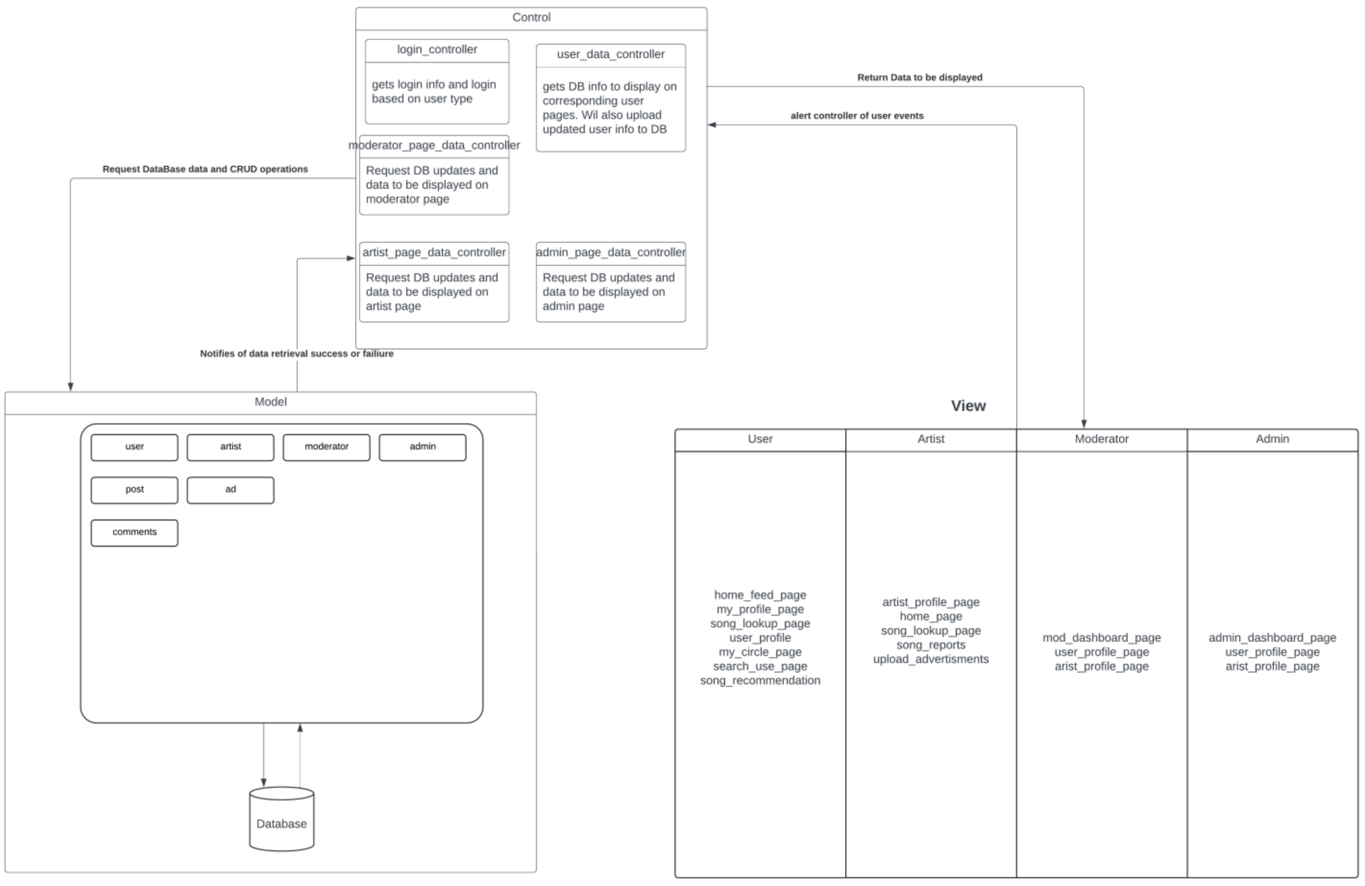
- Cost will be 0
- Delivery date by the end of the semester

5.6. Other Requirements

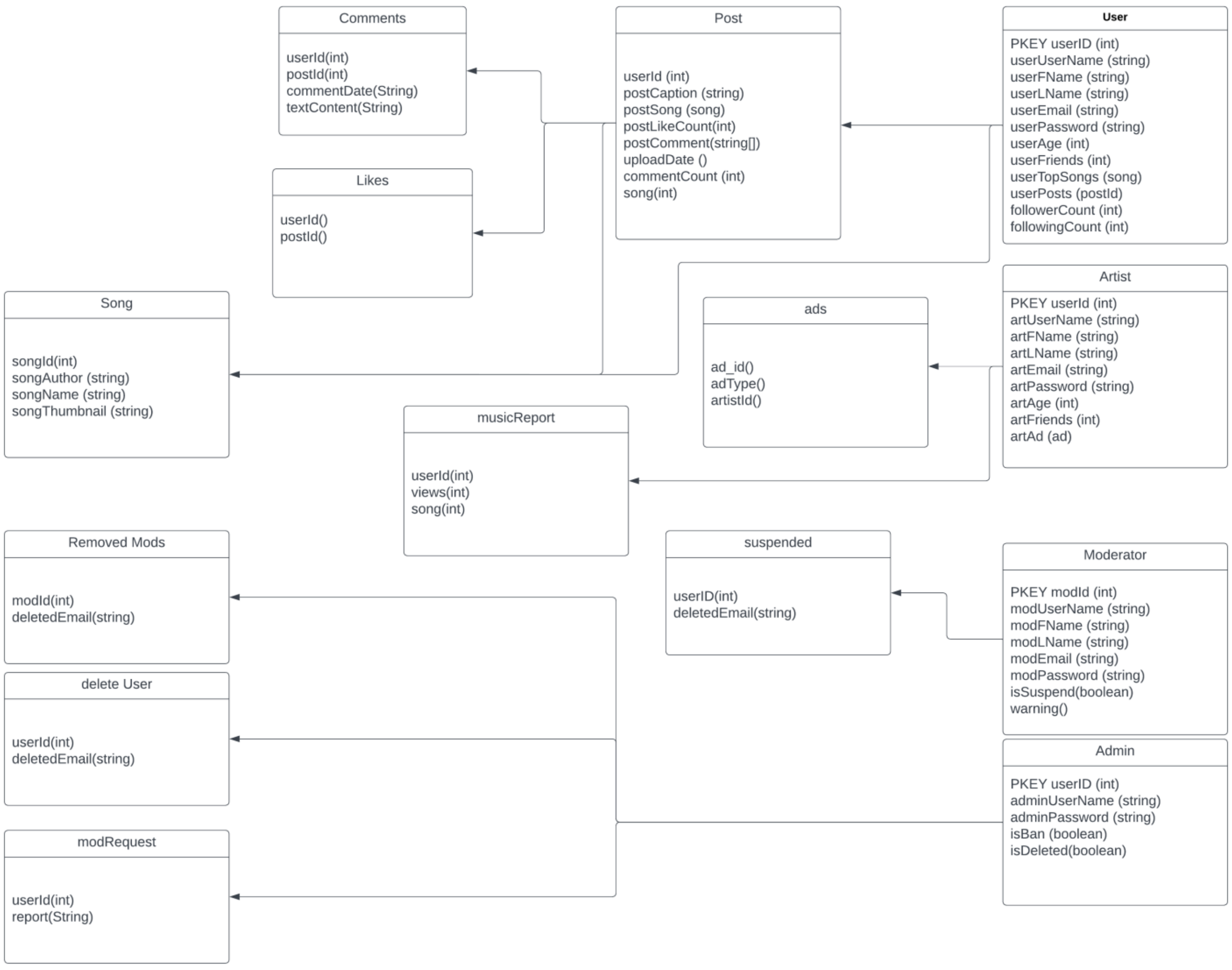
- Regular progress updates and communication among team members to ensure alignment with project objectives and requirements.
- Documentation of codebase and project progress to facilitate knowledge transfer among team members and future maintenance of the platform.

6. Design Documents

6.1. Software Architecture

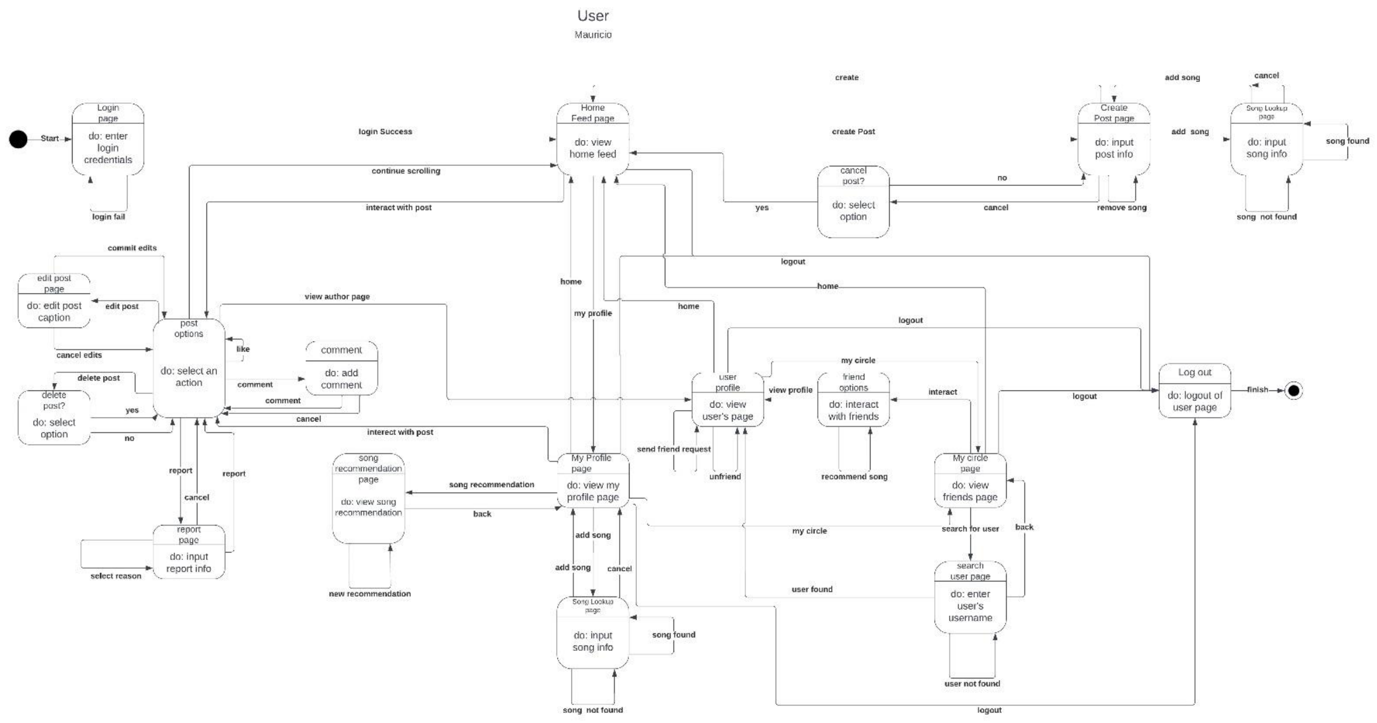

Control
login_controller
gets login info and login based on user type
user_data_controller
gets DB info to display on corresponding user pages. Wil also upload updated user info to DB
moderator_page_data_controller
Request DB updates and data to be displayed on moderator page
artist_page_data_controller
Request DB updates and data to be displayed on artist page
admin_page_data_controller
Request DB updates and data to be displayed on admin page
Return Data to be displayed
alert controller of user events
Request DataBase data and CRUD operations
Notifies of data retrieval success or failiure
Model
user
artist
moderator
admin
post
ad
comments
Database
View
User
Artist
Moderator
Admin
home_feed_page
my_profile_page
song_lookup_page
user_profile
my_circle_page
search_use_page
song_recommendation
artist_profile_page
home_page
song_lookup_page
song_reports
upload_advertisments
mod_dashboard_page
user_profile_page
arist_profile_page
admin_dashboard_page
user_profile_page
arist_profile_page

6.2. High-Level Database Schema

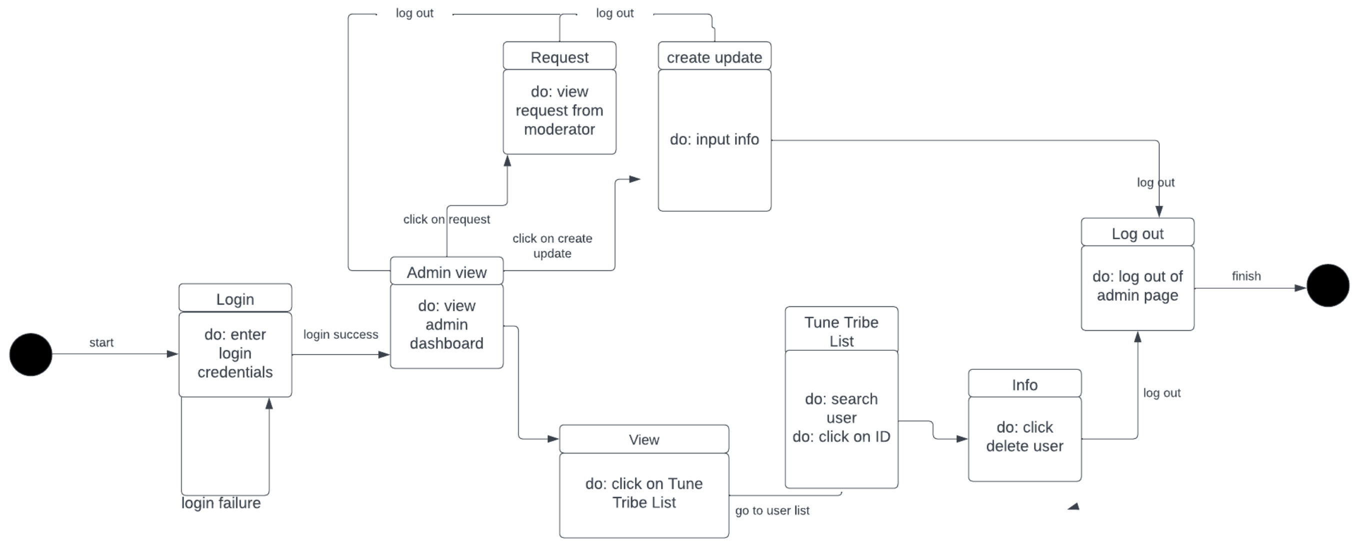


6.3. Software Design

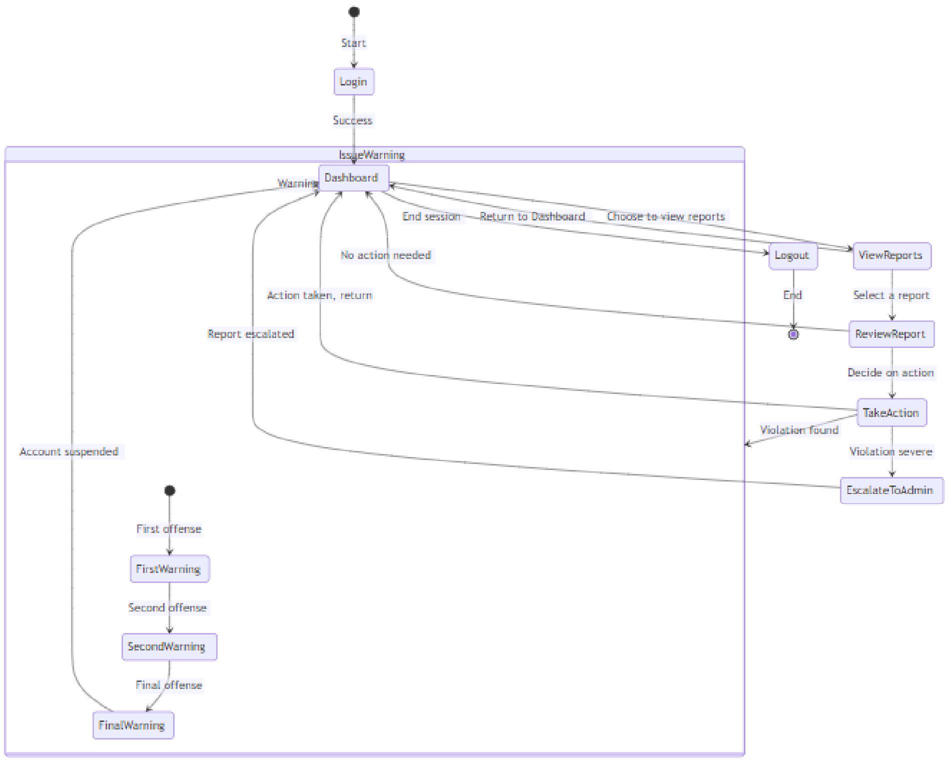
6.3.1. State Machine Diagram: User (Mauricio)



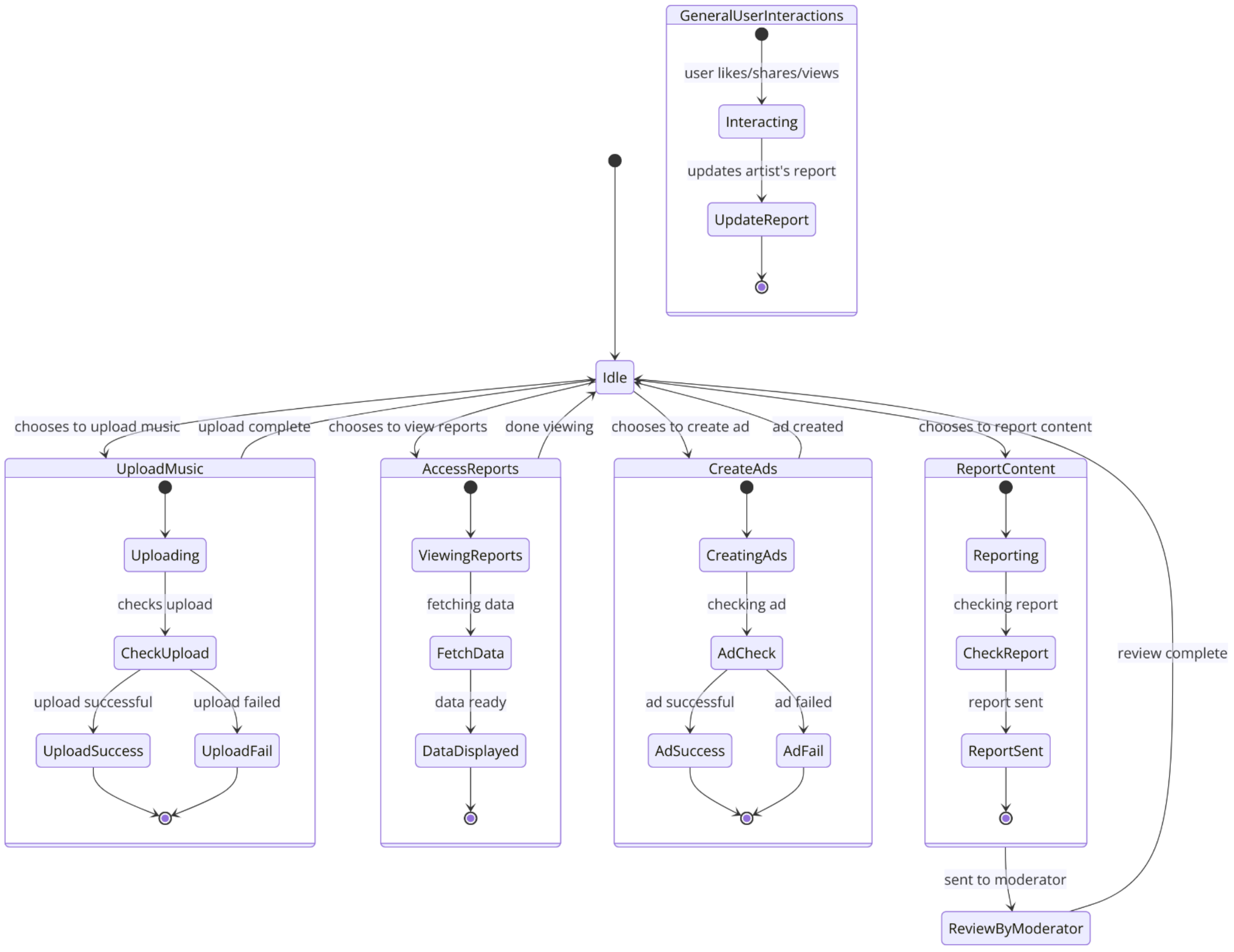
6.3.2. State Machine Diagram: Shauna Ayscue (Admin)



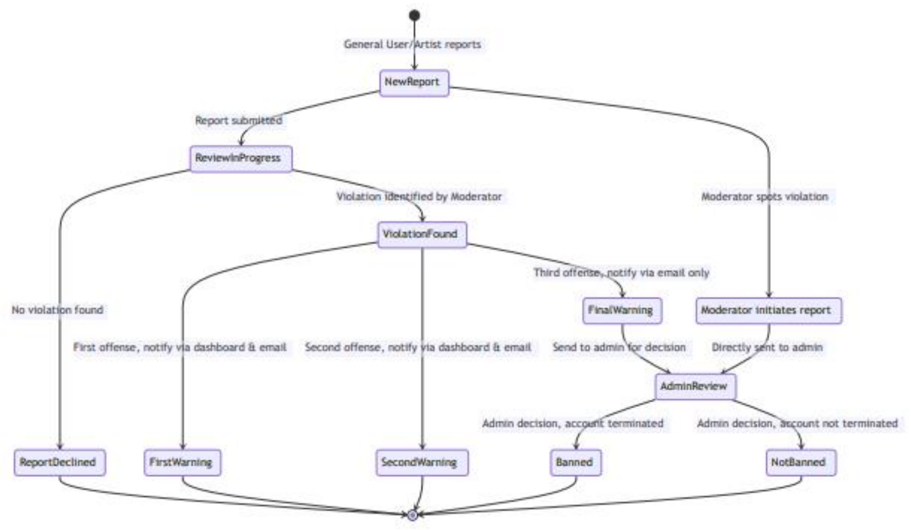
6.3.3. State Machine Diagram: Emmanuel (Mod)



6.3.4. State Machine Diagram: Artist (Kennan)

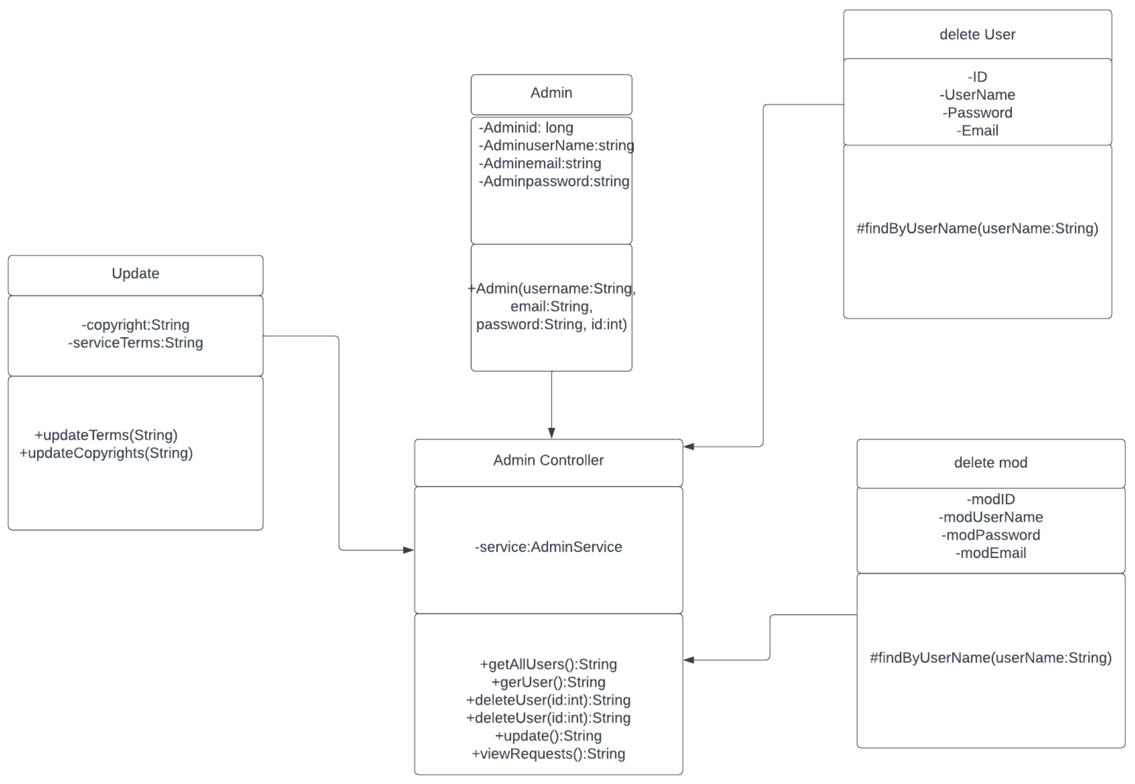


6.4. UML Class Diagram
Mod – Emmanuel(?)



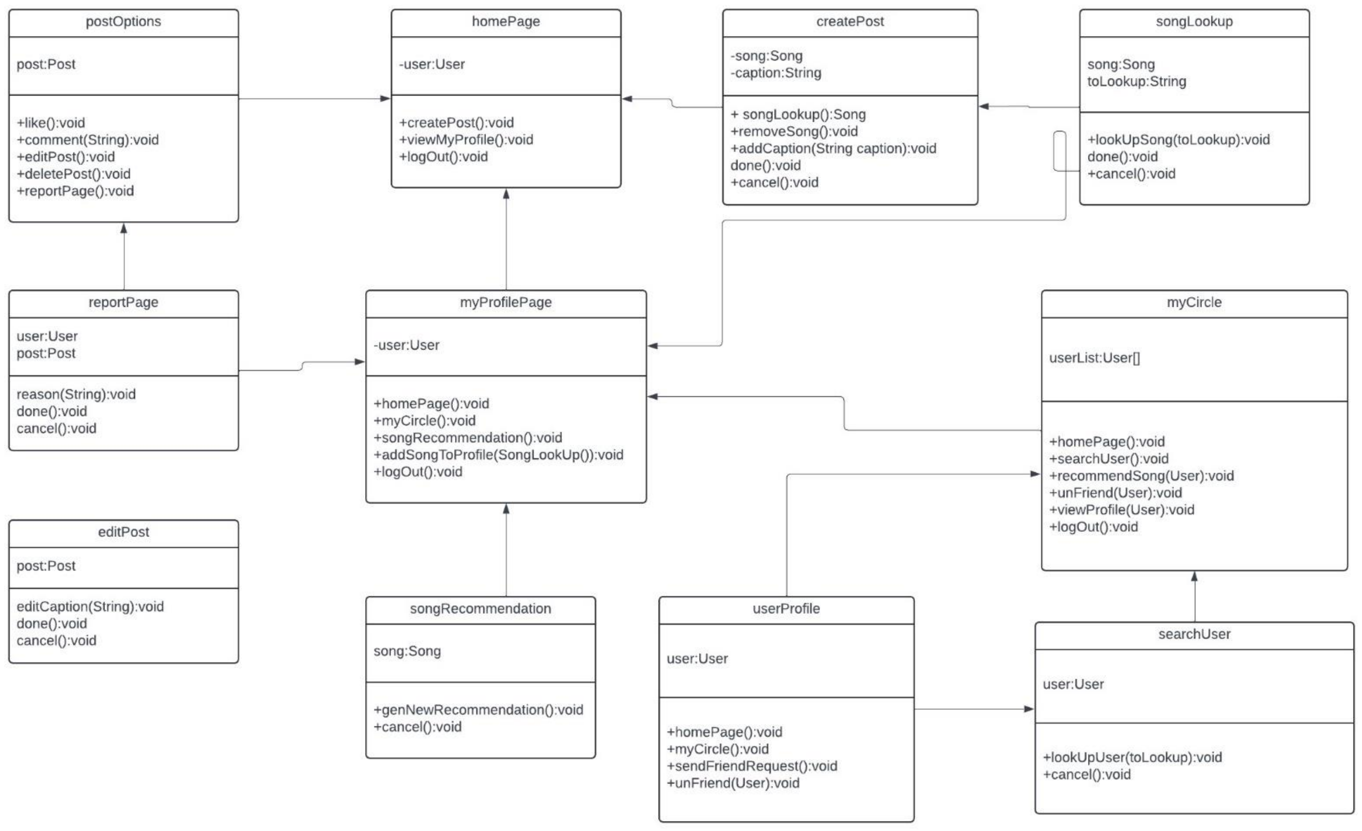
General User/Artist reports
NewReport
Report submitted
ReviewInProgress
Violation identified by Moderator
Moderator spots violation
ViolationFound
Third offense, notify via email only
No violation found
FinalWarning
Moderator initiates report
First offense, notify via dashboard & email
Second offense, notify via dashboard & email
Send to admin for decision
Directly sent to admin
AdminReview
Admin decision, account terminated
Admin decision, account not terminated
ReportDeclined
FirstWarning
SecondWarning
Banned
NotBanned

Admin- Shauna Ayscue



Artist-Kennan

User-Mauricio



ss

7. Scenario

7.1. Brief Written Scenario with Screenshots