Software Engineering
Software Requirements Specification (SRS) Document

Spartan Jam

https://github.com/kaniisun/SpartanJam

Final Version

By: [Kanika Sun, Nehabahen Chauhan, Sebastian Del Campo]

I HAVE ABIDED BY THE UNCG ACADEMIC POLICY ON THIS ASSIGNMENT.

Student's Signature Sebastian Del Campo Date 4/23/2024

Student's Signature Nehabahen Chauhan Date 4/24/2024

Student's Signature Kanika Sun Date 4/24/2024

Table of Contents

1. Introduction

1.1 Purpose

The goal of our project is to create a music website. It allows for users to add and listen to music from their favorite artists as well as local artists. Artists can upload music with a system that allows for passage or denial of the song uploads.

1.2 Document Conventions

The purpose of this Software Requirements Document (SRD) is to describe our project and the requirements we would need for SpartanJam (SJ). There are sections that describe what our project would include, the users, and what it would do. The requirements describe how we would do it, what we would use, our accomplishments and the things we struggled with.

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.
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

The intended audience for the application is people who are attending the University of North Carolina at Greensboro, hence the name, SpartanJam. This is also intended for an audience who enjoy listening to music on the go, and for those who are interested in creating music and trying to look for a platform to post their songs.

1.5 Project Scope

The goal of the software is to allow users to listen to music they like and have people who love to create music a new website to share their music on. In hindsight, it would allow for people in the UNCG community to share their talents with others on campus.

The benefits of the project include:

- Listeners are able to like music and create a playlist of their favorite songs.
- Artists posting music and getting approval from administration for the song to pass through.
- An enclosed music space for students and staff at UNCG.
- Local artists could share their music.
- Regular listeners could still listen to their favorite artists.

1.6 Technology Challenges

This is all of our first time using APIs so it's going to be a little bit difficult for all of us to learn from the beginning. It's also our first time working with servers and learning how to make a functional website. It was a little difficult learning UI and backend.

1.7 References

References to CSC 340 powerpoints.

2. General Description

2.1 Product Features

The product features include the ability for individuals and artists to create an account and as well as an admin being able to manage those accounts. Users should be able to log in and listen to music and create a playlist of their favorite songs. Artists should be able to upload their own music and display the music they uploaded. The admin would be monitoring users as well as approving or denying new artist music.

2.2 User Class and Characteristics

Casual users should just be able to know what music they want to listen to. They should be able to know how to use a regular music website. Artists should know how to upload music. Admins should understand what is appropriate for the music application.

2.3 Operating Environment

The application is designed to operate on the web across various devices that are able to run a web browser.

2.4 Constraints

Some possible constraints that might pose a challenge to the development of the software could be testing API might be unavailable from time to time if Spotify APIs Servers were to go down. Another possible

constraint might also include the necessity of access to the internet as this is an online platform if there is no stable connection to the site the program will not function.

2.5 Assumptions and Dependencies

The software will be dependent on the Spring Web in order to execute the web application which will be developed via NetBeans, it will also be dependent on the Spotify API as it will be receiving and gaining information for music and audio from their server.

3. Functional Requirements

3.1 Primary

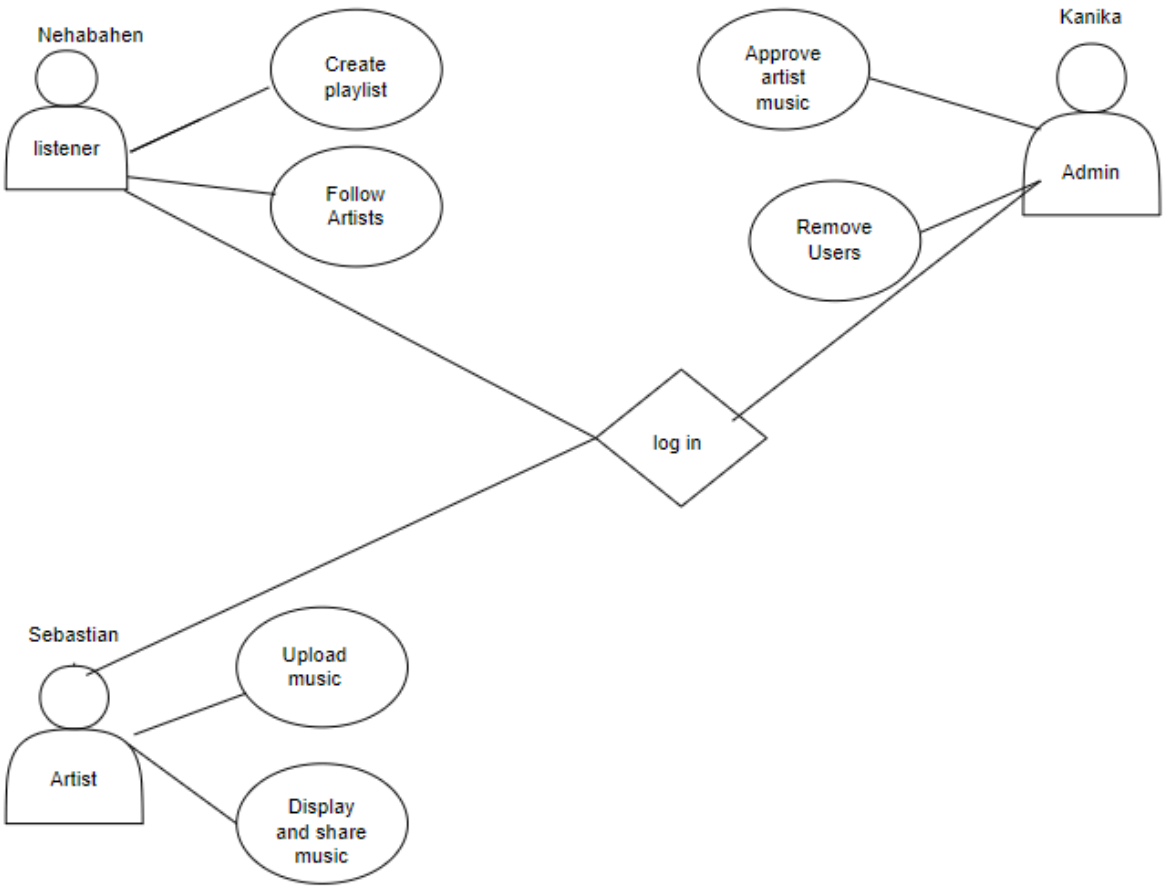
- The system will allow the user to create an account or log in, depending on what type of user they are (Listener, Artist, Admin).
- The system will allow listeners to listen to music and add songs to a playlist.
- The system will allow artists to upload music.
- The system will allow an admin to monitor artist music uploads and approve of them.
- The system will allow an admin to manage user accounts.

3.2 Secondary

- Passwords should only be accessible by individual account users.
- Only individual users should be able to see the playlists and songs they like.
- Artists should be the only ones able to upload music to their account.
- Admins are the only ones able to monitor artist music.

3.3 Use-Case Model

3.3.1 Use-Case Model Diagram



3.3.2 Use-Case Model Descriptions

3.3.2.1 Actor: Admin (Kanika)

- **Approve Artist Music**: Admins are able to approve artist music to see if its appropriate before allowing it to be uploaded onto the site.
- **Remove Users**: Admins are able to remove users off the site if there is a probable cause.

3.3.2.2 Actor: Listener (Nehabahen)

- **Create Playlist**: Allows listener to create a playlist of songs that they like
- **Follow Artists:** Allows listener to search for and follow artists that they are interested in

3.3.2.3 Actor: Artist (Sebastian)

- **Upload Music**: Artists are able to upload their music onto the platform for other listeners to listen.
- **Edit Music**: Artists are able to edit their music by taking it down or reuploading the music.

3.3.3 Use-Case Model Scenarios

3.3.3.1 Actor: Listener (Nehabahen)

- **Use-Case Name**: Creating Playlists
  - **Initial Assumption**: Listener will be able to create his/her own favorite playlist.
  - **Normal**: Listener will create his/her own account and will interact with other features like shar, comment like the music.
  - **What Can Go Wrong:** When a user logs out, the playlist can be deleted.
  - **Other Activities**: add or delete the playlist
  - **System State on Completion**:The edit button should appear on the right of music name
- **Use-Case Name**: Follow Artists
  - **Initial Assumption**: Listeners are able to follow artists
  - **Normal**: Listeners should be able to search and follow artists they find interest in
  - **What Can Go Wrong**: The followed artist doesn't appear in their following.
  - **Other Activities**: n/a
  - **System State on Completion**: The following button should appear on the client side view.

3.3.3.2 Actor: Artist (Sebastian Del Campo)

- **Use-Case Name**: Upload Music
  - **Initial Assumption**: Artists will upload an audio file of their choice which will be reviewed by the moderator and if approved will be shared on the forum.
  - **Normal**: The artist will upload an audio file into a submission box which will then be sent to the moderator for review before being uploaded onto the website.

        - **What Can Go Wrong**: A possible security threat is if a malicious file gets sent, the file might be corrupted or can't be opened on the moderator's side.
        - **Other Activities**: Create a specific page that contains an audio file along with comments and possible connections to forums.
        - **System State on Completion**: The audio file will be denied by a moderator and will need to fix the audio and resend it for approval or can be approved and get a specific link to a page that contains their audio file along with audio player options, sections to comment, and possible links to forums.
    - **Use-Case Name**: Display Music
        - **Initial Assumption**: Artists can go into their specific audio file page. This will also need approval from moderation.
        - **Normal**: The artist can click a button to show a specific audio file page.
        - **What Can Go Wrong**: Possibility to upload malicious data, music may produce bugs, and music may be lost if gone unsaved.
        - **Other Activities**: The audio file page will be updated.
        - **System State on Completion**: The audio file will be denied by a moderator.

3.3.3.3 Actor: Admin (Kanika)

- **Use-Case Name**: Approve Artist Music
    - **Initial Assumption**: Admin can approve artist music.
    - **Normal**: When an artist uploads new music, it has to go through a process where the admin approves of music.
    - **What Can Go Wrong**: The music doesn't reach the admin for approval.
    - **Other Activities**: Deny the music uploaded by artists.
    - **System State on Completion**: Artist music uploads.
- **Use-Case Name**: Remove Users
    - **Initial Assumption**: Admin is able to remove users or artists on the platform.
    - **Normal**: Admin has the ability to delete a user.
    - **What Can Go Wrong**: Deletion of the user doesn't occur.
    - **Other Activities**: Manage user accounts.
    - **System State on Completion**: User removed.

4. Technical Requirements

4.1 Interface Requirements

4.1.1 User Interfaces

Users should be directed to a login page upon reaching the website. After logging in or creating an account, the user would be prompted to the appropriate page they created an account for. For listeners, the homepage should provide the music they saved and artists they follow. For artists, the homepage should give them options to upload and edit music. For admins, the homepage should give them options to approve music or remove users.

4.1.2 Hardware Interfaces

The web application should be able to run on computers and laptops that have access to the internet and can interact with websites.

4.1.3 Communications Interfaces

It must be able to connect to the internet. The communication protocol must be able to connect to the Spotify API.

4.1.4 Software Interfaces

- Spring Boot will be used in order to develop the website
- Spotify API will implement the Spotify Software which will be a software component towards the project

5. Non-Functional Requirements

5.1. Performance Requirements

NFR0(R): The local copy of the Spotify Music API may consume less than 15 MB of memory
NFR1(R): The system (including local copy of Spotify Music API) may consume less than 40 MB of memory
NFR2(R): The novice regular user will be able to create an account and listen to music in less than 15 minutes.
NFR3(R): The expert regular user will be able to create an account and listen to music in less than 5 minutes.
NFR4(R): The novice artist will be able to create and post music onto the platform in less than 30 minutes.
NFR5(R): The expert artist will be able to create and post music onto the platform in less than 10 minutes.

5.2. Safety Requirements

- Security Login for Website maintenance
- Cybersecurity to prevent from cyber attacks (ie ddos attack)
- Moderation Team to prevent unwanted/malicious files from entering into website.

5.3 Security Requirements

- NFR6(R): The system will only be used by authorized users
- NFR7(R): Any materials use in the project will be adhered to copyright laws

5.4. Software Quality Attributes

5.4.1. Availability

Software will be available when the project gets imported to a local computer along with an hosting software (like XAMPP) that can run the software (note: spartan_db database would need to be created prior to running the software for the first time). After that whenever the Netbeans project and XAMPP servers are running the software will be available.

5.4.2. Correctness

Peer reviews and team members reviewing each other's implementations will help with maintaining correctness in our software.

5.4.3. Maintainability

Software details and maintenance can be done through Netbeans through analysis of the project or through the XAMPP database.

5.4.4. Reusability

Software can be reused as data will be stored to database on local computer but only on the local computer, any other device will require configuration in order for project to work.

5.4.5. Portability

Project will be portable for computer devices as long there is access to the internet, Netbeans and XAMPP

5.5. Process Requirements

5.5.1. Development Process Used

We used an agile method to work on it in partial scrims as various assignments were due.

5.5.2. Time Constraints

Having our use cases completed by certain times. Presenting our progress. Having our final project completed by April 30th.

5.5.3. Cost and Delivery Date

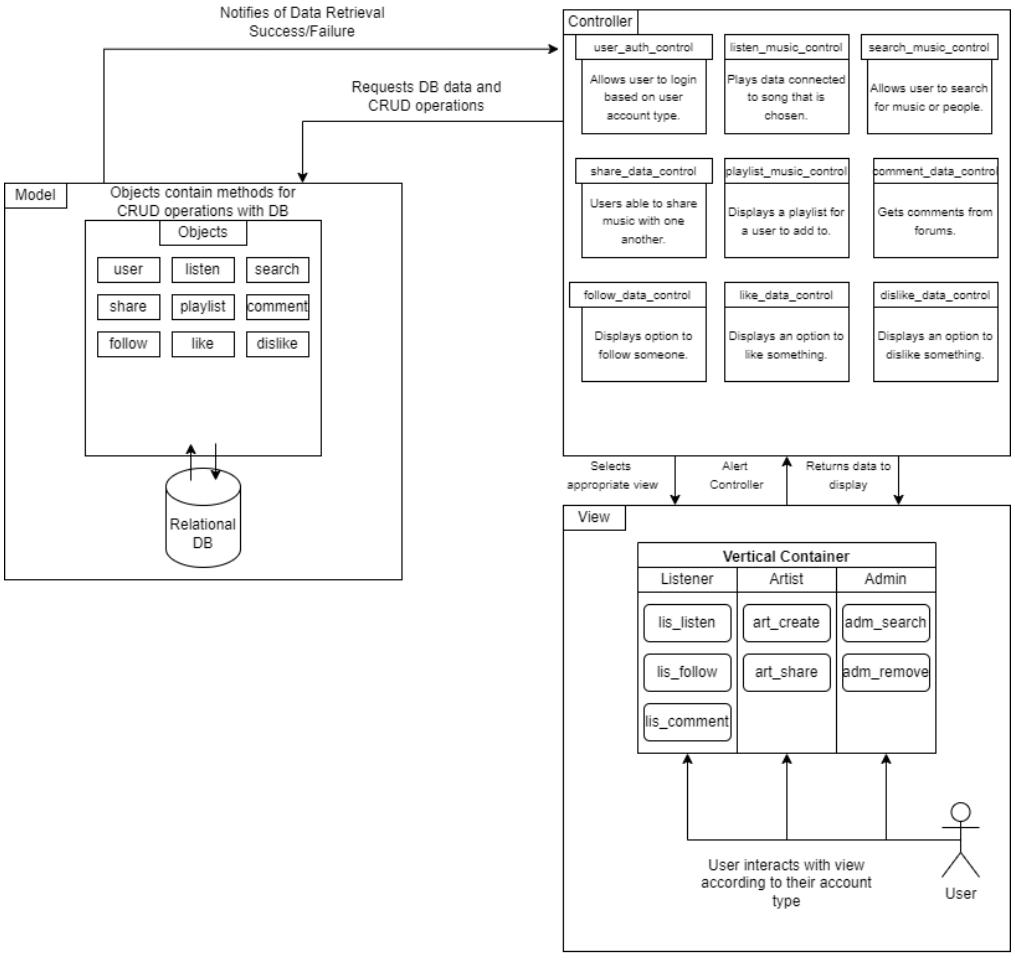
Project will be delivered on April 30, 2024 with 0 cost due to the project being hosted on localhost rather then a URL.

5.5. Other Requirements

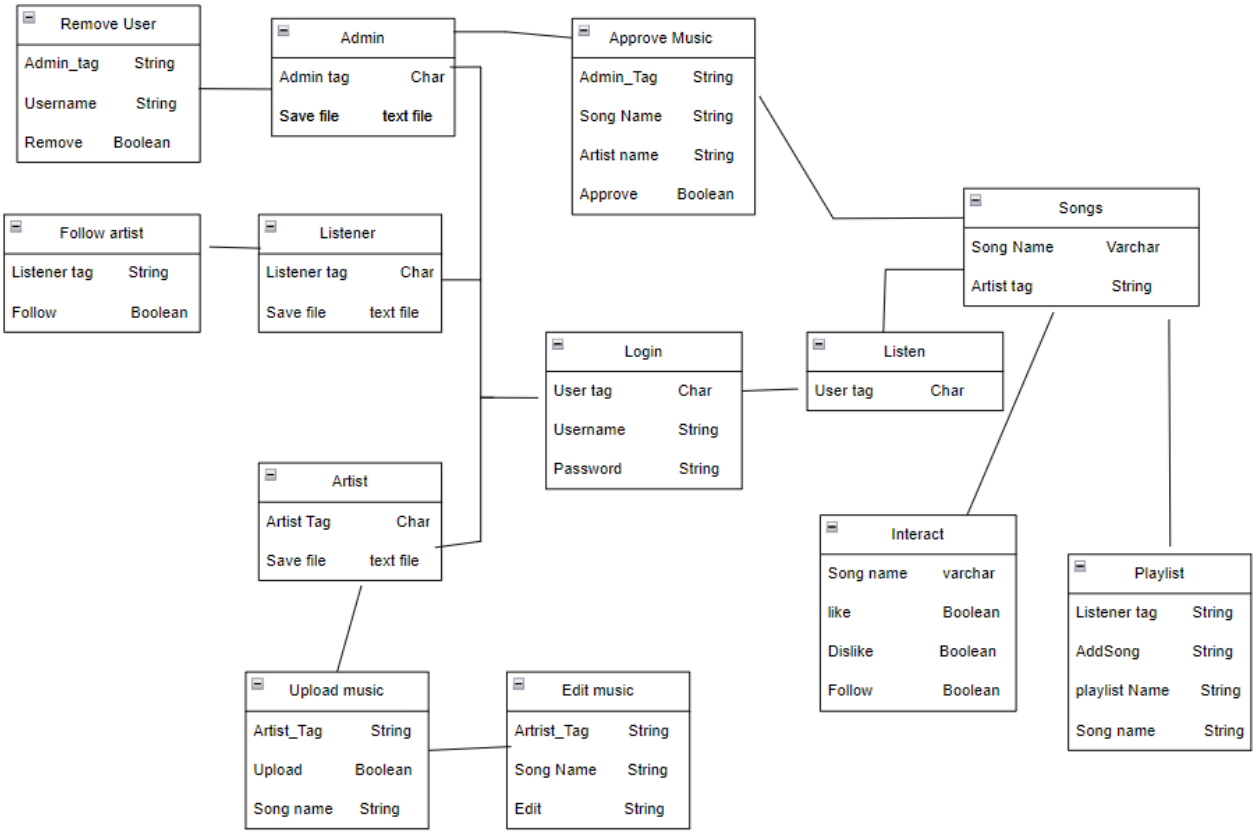
Project will need to use MP3 files in order to upload songs via the artist account.

6. Design Documents

6.1. Software Architecture

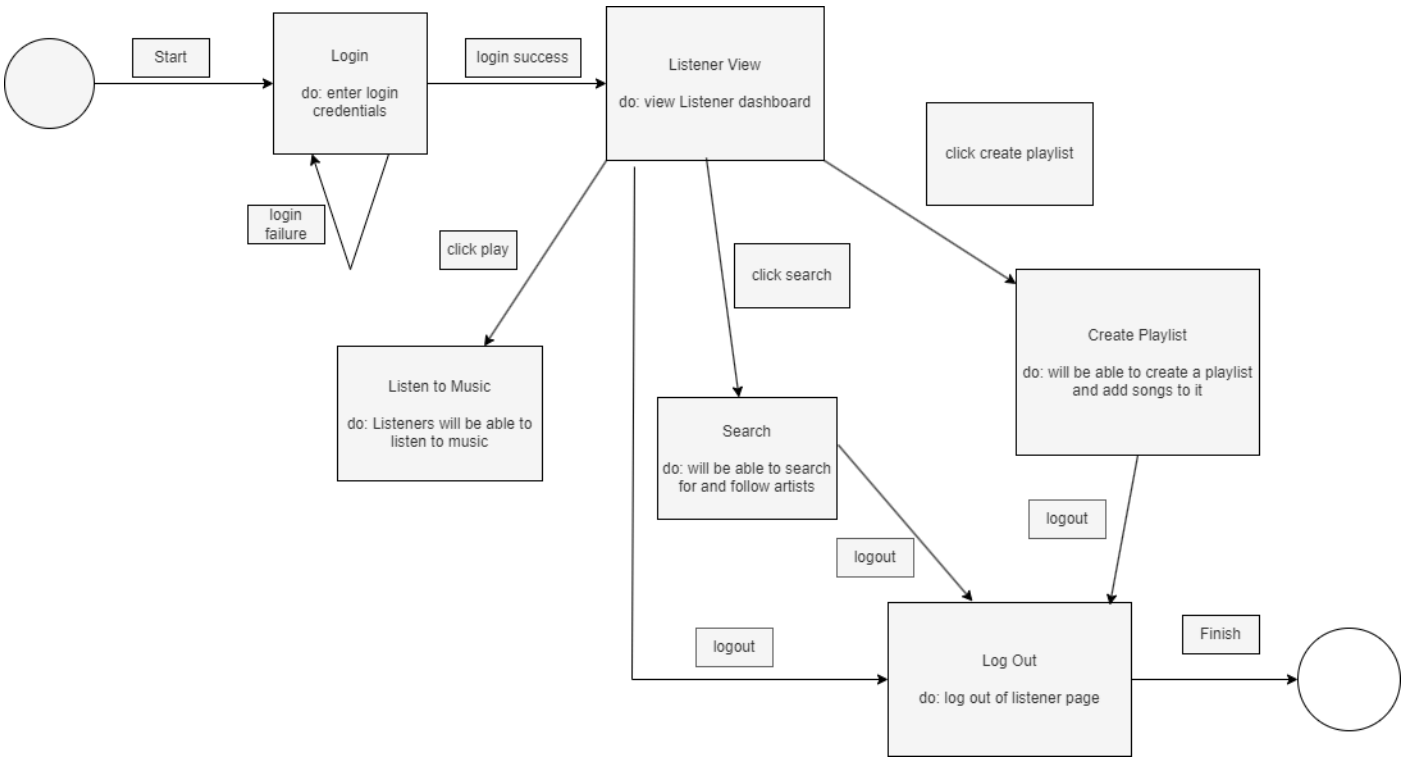


6.2. High-Level Database Schema

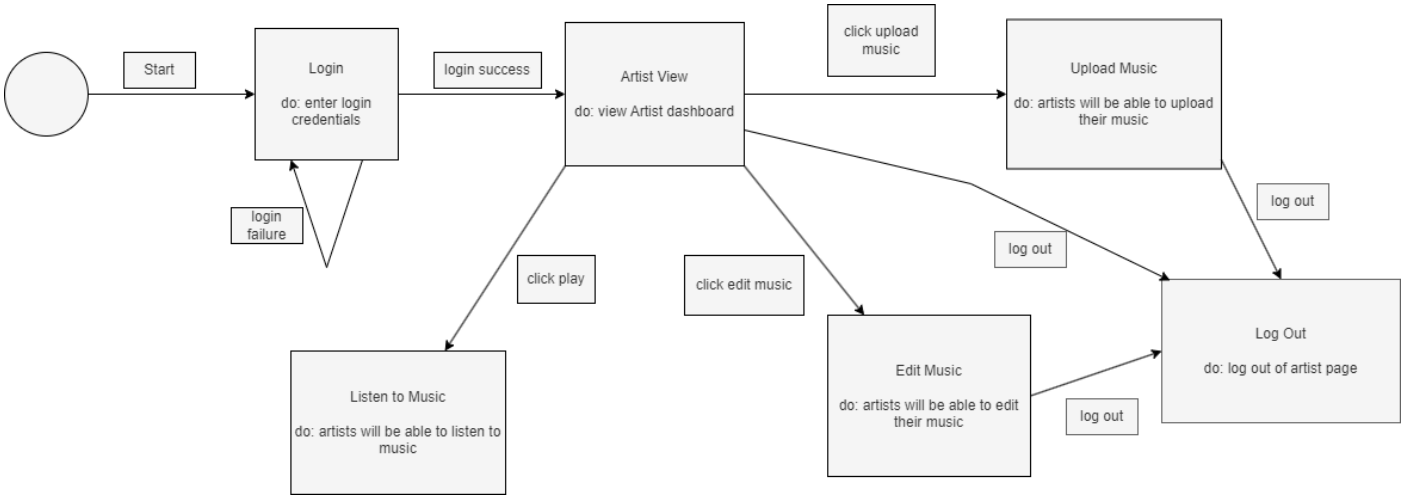


6.3. Software Design

6.3.1. State Machine Diagram: Actor Name (Nehabahen Chauhan)

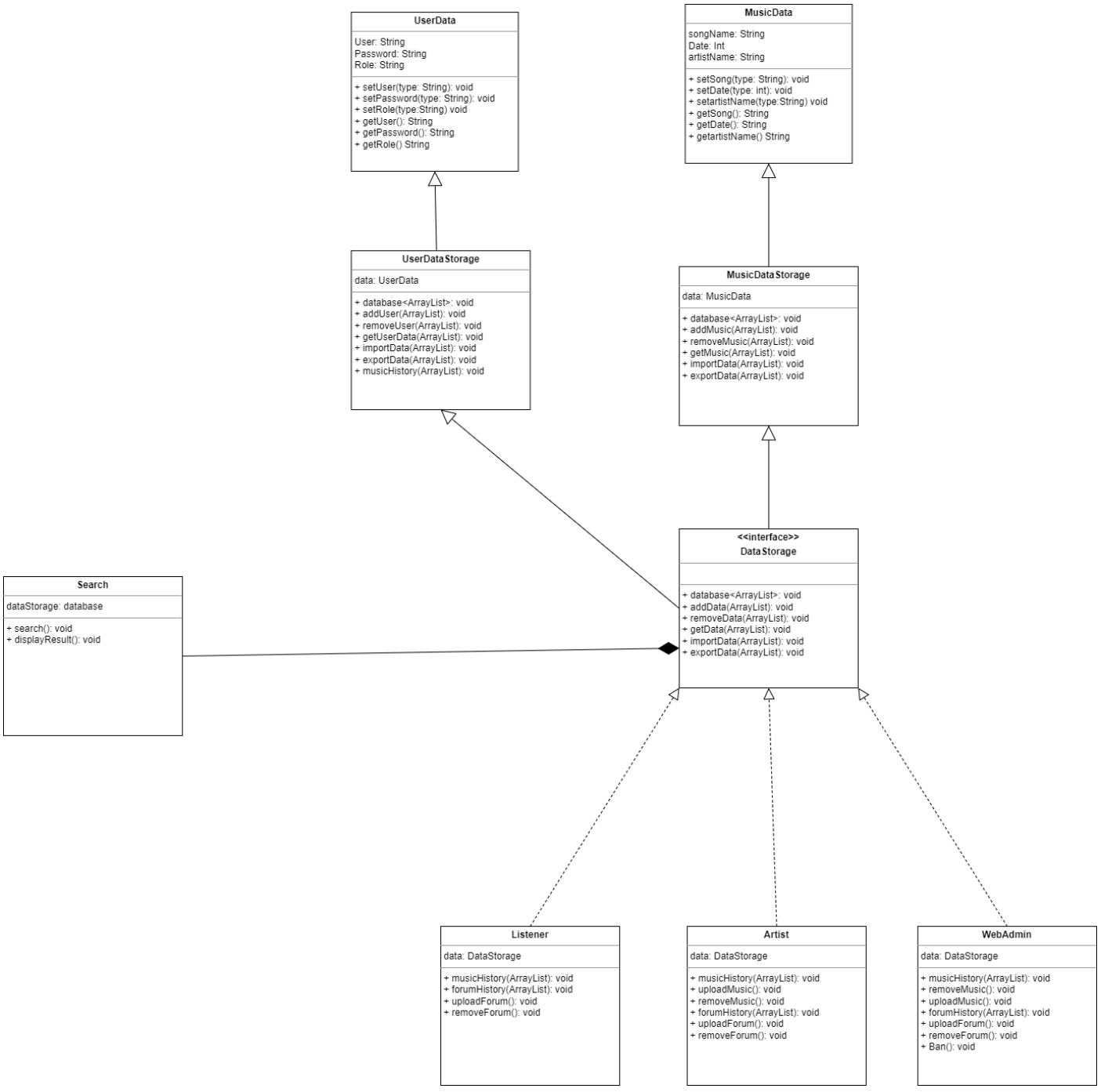


6.3.2. State Machine Diagram: Actor Name (Sebastian Del Campo)



6.3.3. State Machine Diagram: Actor Name (Kanika Sun)

6.4. UML Class Diagram

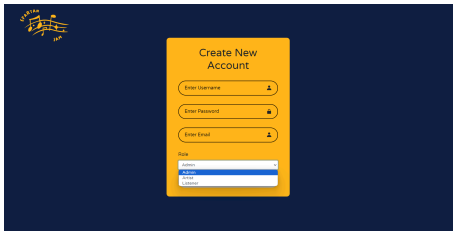


7. Scenario

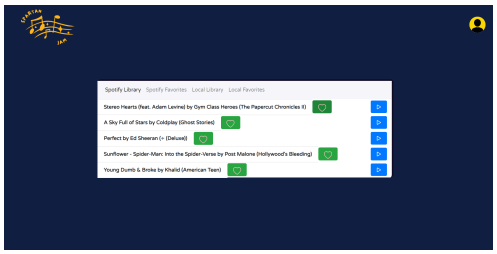
7.1. Brief Written Scenario with Screenshots

7.1.1. Listener: Add Songs to Playlist

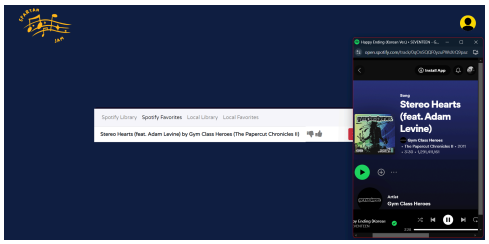
- Listener creates an account that has the listener role.



- Listener logs into account and presents the listener view.

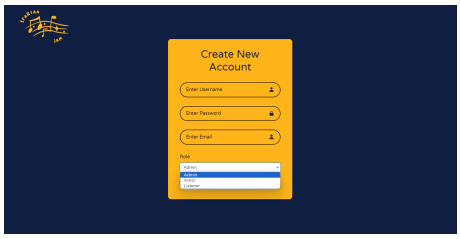


- Listener adds songs to their playlist and plays the songs.

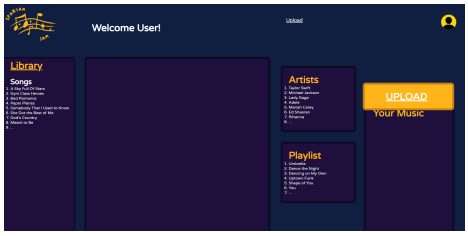


7.1.2. Artist: Upload Music

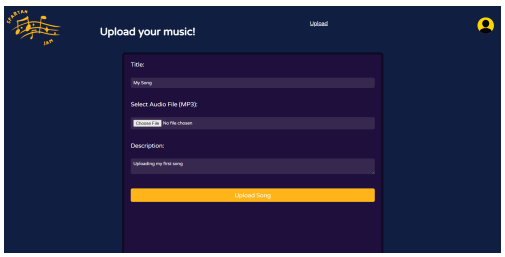
- Artist creates an account with a artist role.



- Artist logs into account and presents the artist's view.



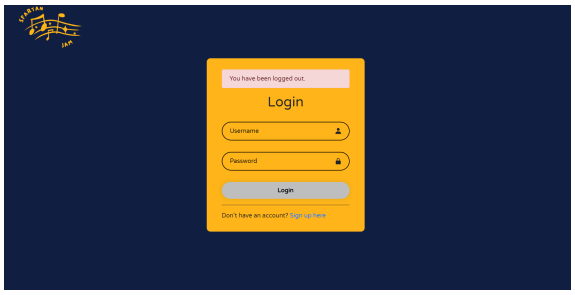
- Artist clicks on the upload feature and uploads songs S1, S2, S3, which will go to the admin for approval.



7.1.5. Artist: Display and Share Music

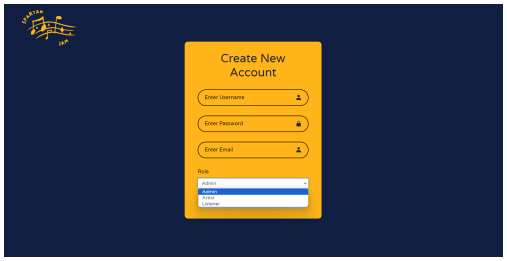
- Artist shows their share feature.

- Produces a link to share their song.
- Artist logs out.

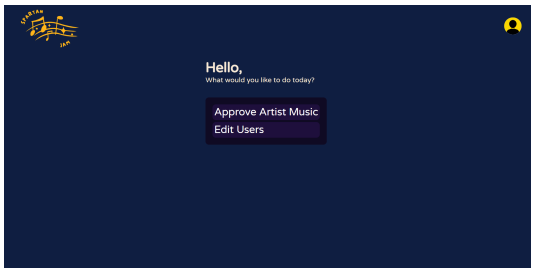


7.1.3. Admin: Approve Artist Music

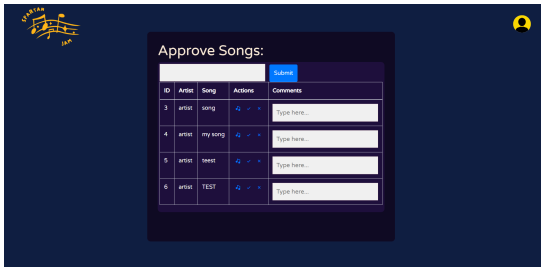
- Admin creates an account that has the admin role.



- Admin logs into account and presents the admin view.

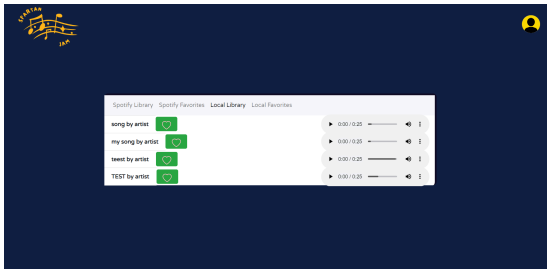


- Admin reviews S1, S2, S3 that was uploaded by the artist and chooses to approve S1, S2 and deny S3.

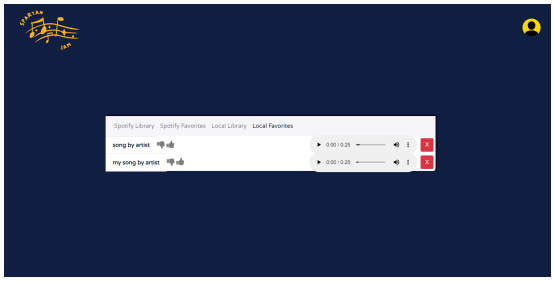


7.1.4. Listener: Like/Follow Artists

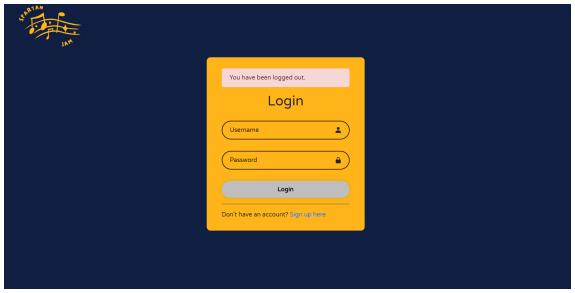
- Listener goes to their local library where songs S1 and S2 uploaded by the artist will be shown.



- Listener favorites S1 and S2.
- Listener plays the songs.
- Listener removes S1 from their local playlist.

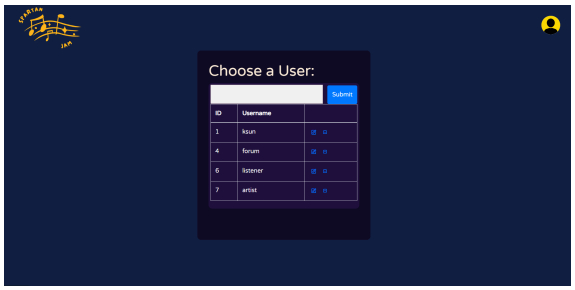


- Listener logs out.



7.1.6. Admin: Edit Users

- Admin goes to their edit users tab where user U1, U2, U3, U4 will appear.



- Admin will delete U1.
- Admin will edit U2.
- U1 will try to login but their account is deleted.
- U2 will login with their different credentials.
- Admin logs out.