SpUStify

Use-Case Specification

Version 2.1

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 30/6/2023 | 1.0 | Initial version | Dương Trường Bình |
| 12/07/2023 | 2.0 | Revised version in PA3 | Dương Trường Bình |
| 02/08/2023 | 2.1 | Revised version from Mr.Vu’s comment | Dương Trường Bình |
|  |  |  |  |

Table of Contents

[**1. Use-case Model 4**](#_heading=h.gjdgxs)

[**2. Use-case Specifications 4**](#_heading=h.30j0zll)

[2.1 Use-case: Search songs/playlists/artists 4](#_heading=h.2et92p0)

[2.2 Use-case: Leaderboard 5](#_heading=h.e9a74gxs1vgv)

[2.3 Use-case: View Favorite Songs and Playlists 5](#_heading=h.r8gack999gkp)

[2.4 Use-case: History songs and playlists 6](#_heading=h.v0uz56cbhmn9)

[2.5 Use-case: Download a song/playlist 6](#_heading=h.rfxtqmcydyz8)

[2.6 Use-case: Play a song/ playlist 6](#_heading=h.c1xbuuujtgdf)

[2.7 Use-case: Upload songs 7](#_heading=h.r1q37mu81qyy)

[2.8 Use-case: Create a playlist 8](#_heading=h.njw102am1b1e)

[2.9 Use-case: Edit profile 9](#_heading=h.qdu9j9azdu8)

[2.10 Use-case: Edit song 9](#_heading=h.ong831ltx4jw)

[2.11 Use-case: Edit playlist 10](#_heading=h.p424aak1dz66)

[2.12 Use-case: Delete a playlist 11](#_heading=h.yeg66mywunaq)

[2.13 Use-case: Delete a song 12](#_heading=h.trh7g68fni4c)

[2.14 Use-case: Playlist and songs recommendation 13](#_heading=h.soy3b7fged6r)

[2.15 Use-case: Calculate and generate reports 14](#_heading=h.wssdnmi8dvl8)

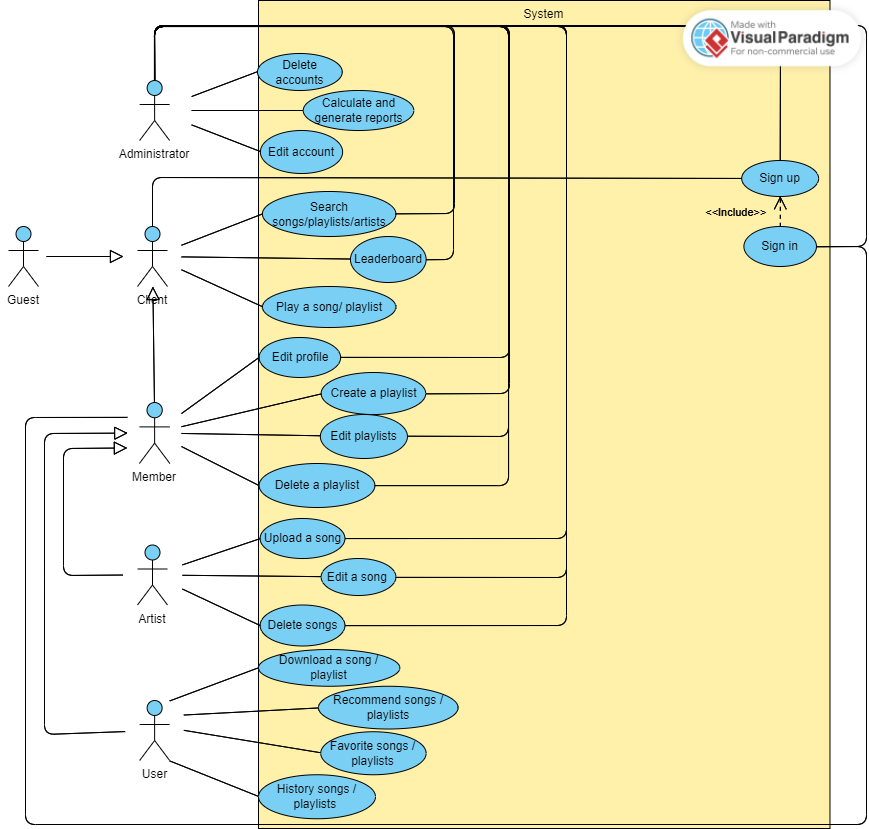
[2.16 Use-case: Sign in 15](#_heading=h.uo33q3d50dqo)

[2.17 Use-case: Sign up 15](#_heading=h.yf4svzsrjc1v)

[2.18 Use-case: Delete account 15](#_heading=h.sn025dbq62ut)

[2.19 Use-case: Edit account 16](#_heading=h.xp4uc63oe1no)

# Use-case Model



# Use-case Specifications

## Use-case: Search songs/playlists/artists

| Use case Name | Search songs/playlists/artists |
| --- | --- |
| Brief description | This use case describes the client/admin searching songs, playlists or artists based on keywords. |
| Actors | Client, Admin |
| Basic Flow | 1. Client/admin enters keywords in the 'Search' field at the navigation bar. 2. Client/admin then clicks the 'Search' button to initiate the search. 3. The website displays all results related to the entered keyword, including songs, playlists or artists |
| Alternative Flows | **Alternative flow: Keyword not found**   1. In the essential flow step #2, if the entered keyword is not found, the website presents an announcement to the user or guest. 2. Client/admin can navigate back to the homepage. |
| Pre-conditions | Client/admin can be anywhere on the website |
| Post-conditions | The search results for the keyword were successfully retrieved and displayed on the website, or an announcement is made if the keyword cannot be found. |

## Use-case: Leaderboard

| Use case Name | Leaderboard |
| --- | --- |
| Brief description | Ranking of songs by likes, by listens. |
| Actors | Client, Admin |
| Basic Flow | 1. At the navigation bar, client/admin clicks on the “Leaderboard” field 2. The website navigates to the leaderboard page by number of listens. |
| Alternative Flows | 1. In the essential flow step #2, if client/admin selects the “Leaderboard” by likes, the website will navigate to the leaderboard page by number of likes |
| Pre-conditions | Client/admin can be anywhere on the website |
| Post-conditions | The website navigates to the leaderboard page |

## 

## Use-case: View Favorite Songs and Playlists

| Use case Name | View Favorite Songs and Playlists |
| --- | --- |
| Brief description | This use case allows user to view their favorite songs and playlists |
| Actors | User |
| Basic Flow | 1. The user selects the "Favorites" or "My Library" option from the navigation menu. 2. The web application displays the user's favorite songs and playlists on the screen. 3. The user can select a favorite song or playlist to view more details or play the song. |
| Alternative Flows | * No Favorite Songs or Playlists: * If the user does not have any favorite songs or playlists * The web application displays a message indicating that there are no favorite songs or playlists available |
| Pre-conditions | * The user has an active SpUStify account and is logged into the application * The user has previously liked songs or playlists as favorites |
| Post-conditions | The user can see their favorite songs and playlists |

## Use-case: History songs and playlists

| Use case Name | Showing Listening history |
| --- | --- |
| Brief description | This use case allows a user to view their listening history. The listening history feature will provide users with a comprehensive record of their previously played songs |
| Actors | User |
| Basic Flow | 1. The user logs into their account on the website 2. Upon successful login, the user navigates to the "Listening History" section of their account 3. The web application displays a list of the user's recently played songs, arranged in logical order 4. Each entry in the listening history list includes the following details:    1. Song title    2. Artist name    3. Date and time of playback 5. The user can scroll through the listening history to view additional entries beyond the initial visible list 6. The user can click on an entry in the listening history to access more details about the song, including additional metadata and lyrics |
| Alternative Flows | * No Listening History Available * If the user does not have any listening history entries, the listening history section displays a message indicating that no history is available. |
| Pre-conditions | * The user has an active SpUStify account and is logged into the application |
| Post-conditions | * The user can view their listening history, including detailed information about each played song * The listening history is continuously updated as the user plays new songs |

## Use-case: Download a song/playlist

| Use case Name | Download a song |
| --- | --- |
| Brief description | This use case describes when the users download a song/playlist |
| Actors | User |
| Basic Flow | 1. The client chooses a song and clicks on it 2. The website navigates to the song/playlist on page 3. The client clicks on the ‘Download’ button 4. The song is downloaded. |
| Alternative Flows |  |
| Pre-conditions | The client must be on the homepage or search result page to select song |
| Post-conditions | The song has been downloaded to the client side |

## Use-case: Play a song/ playlist

| Use case Name | Play a song/ playlist |
| --- | --- |
| Brief description | This use case describes the clients play(listen) a song/playlist |
| Actors | Client |
| Basic Flow | 1. The client chooses a song/playlist and clicks on it 2. The website navigates to the song/playlist on page 3. The client clicks on the ‘Play’ button 4. The selected song or playlist starts playing, and the website shows the controls (play, pause, volume, etc.) |
| Alternative Flows | 1. If the client skips to another song in the playlist, the website updates the currently playing song and continues playback from the new position 2. If the client adjusts the volume, the website modifies the audio output accordingly 3. If the client clicks on a different playlist or song while one is already playing, the website stops the current playback and starts playing the newly selected item 4. If the client wants to stop the music, they can click the pause button or navigate to a different section of the website where the music play is not active |
| Pre-conditions | The client must be on the homepage or search result page to select song or playlist |
| Post-conditions | The client can enjoy listening to music while browsing the website allows seamless control over the music playback  The client can switch songs or playlists and adjust the volume as desired |

## 

## Use-case: Upload songs

| Use case Name | Upload songs |
| --- | --- |
| Brief description | This use case specification describes the functionality and requirements for a web application that allows artists to upload songs |
| Actors | Artist |
| Basic Flow | 1. The artist navigates to the web application's song upload section. 2. The artist selects the option to upload a new song 3. The web application presents a form to provide details about the song, including the title, artist, album, and genre 4. The artist selects the audio file of the song from their local device 5. The web application associates the uploaded file with the provided song details and stores it in a secure location 6. The web application confirms the successful upload and redirects the user to the song's page |
| Alternative Flows | * Invalid File Upload: * If the artist attempts to upload a file that is not a valid audio file, the web application displays an error message and requests the user to select a valid file. * If the uploaded file does not meet the specified format requirements, the web application notifies the user and provides instructions to upload a file in the correct format. * Duplicate Song Upload: * If the artist tries to upload a song with the same title as an existing one, the web application prompts the user to confirm if they want to replace the existing entry or rename the new |
| Pre-conditions | * The artist has an active SpUStify account and is logged into the application * The artist should have a valid internet connection * The artist's device should support file upload functionality |
| Post-conditions | * The uploaded songs are securely stored in the web application's storage system * Users can access and manage the uploaded songs through the web application |

## Use-case: Create a playlist

| Use case Name | Create a playlist |
| --- | --- |
| Brief description | This use case specification describes the functionality and requirements for a web application that allows members to create their playlist and artists to create their album. |
| Actors | Member |
| Basic Flow | Create a Music Playlist   1. The member navigates to the web application's playlist upload section 2. The member selects the option to upload a new playlist 3. The web application presents a form to provide details about the playlist, including a title, description, and tags 4. The member selects the music files that constitute the playlist from their local device 5. The web application associates the uploaded files with the provided playlist details and stores them in a secure location 6. The web application confirms the successful upload and redirects the member to the playlist's page |
| Alternative Flows | * Invalid File Upload: * If the member attempts to upload a file that is not a valid audio file, the web application displays an error message and requests the user to select a valid file. * If the uploaded file does not meet the specified format requirements, the web application notifies the user and provides instructions to upload a file in the correct format. * Duplicate Playlist Upload: * If the member tries to upload a playlist with the same title as an existing one, the web application prompts the user to confirm if they want to replace the existing entry or rename the new |
| Pre-conditions | * The member has an active SpUStify account and is logged into the application * The member should have a valid internet connection * The member’s device should support file upload functionality |
| Post-conditions | * The created playlists/albums are securely stored in the web application's storage system * Members can access and manage the uploaded playlists through the web application |

## Use-case: Edit profile

| Use case Name | Edit profile |
| --- | --- |
| Brief description | This use case describes the process of when the member/admin wants to edit/update his profile |
| Actors | Admin, Member |
| Basic Flow | 1. Update profile page shows up 2. The system presents the member/admin with a form to modify the profile information. 3. The member/admin makes the desired changes to the profile, such as updating name, profile picture, or contact information,.... 4. Once the member/admin has finished making the changes, submit the updated profile information. 5. The system displays a confirmation message to the member/admin, indicating that their profile has been successfully updated. |
| Alternative Flows | 1. In step 2, member/admin can click the cancel button to cancel changing the profile. 2. In step 3, if the member/admin wishes to change password,he can click the change password button. Then the member/admin has to input the password twice before submitting to the system. |
| Pre-conditions | Member/admin choose the edit account option in the user profile |
| Post-conditions | The account is or is not updated based on the validity of the submitted information |

## 

## Use-case: Edit song

| Use case Name | Edit song |
| --- | --- |
| Brief description | This function allows artists to modify the details and metadata of a song. Artists can update information such as the song title, artist name, album name |
| Actors | Artist |
| Basic Flow | 1. The artist searches or browses for the desired song they want to edit 2. The artist selects the specific song they want to edit from the search results or their library 3. The website displays the current details of the selected song, such as the title, artist name, and album name 4. The artist modifies the desired fields, such as updating the song title, artist name, or album name 5. The artist saves the changes by clicking on the "Save" or "Update" button |
| Alternative Flows | 1. If the changes made by the artist are invalid ( Duplicate song’ name) the website displays an error message indicating the validation failure |
| Pre-conditions | The artist must be on the homepage or search result page to select song |
| Post-conditions | * The selected song's details, such as title, artist name, and album name, are updated with the changes made by the user * The modified song details are saved and reflected in the artist library and any other relevant sections of the website * If the changes were successful, the website may display a confirmation message to notify the user that the song details have been successfully updated |

## Use-case: Edit playlist

| Use case Name | Edit playlist/album |
| --- | --- |
| Brief description | This use case describes the process of updating an existing playlist in the SpUStify application. It allows members/admin to modify the playlist's name, add or remove songs, and change the order of songs within the playlist. |
| Actors | Member, Admin |
| Basic Flow | 1. Member/admin accessing their playlist management section in the SpUStify application. 2. The application displays a list of the user's existing playlists. 3. Select the playlist they want to update. 4. The application presents the playlist details, including the current name and songs. 5. Choosing the option to update the playlist. 6. The application allows the user to modify the playlist's name, playlist’s image, remove existing songs, or rearrange the order of songs. 7. Making the desired changes to the playlist. 8. The application validates the changes and updates the playlist accordingly. 9. The updated playlist is saved and updated in the member's account. |
| Alternative Flows | 1. In step 3: If the playlist selected by the member/admin doesn't exist, the application displays an error message and prompts the user to choose a valid playlist. 2. In step 6: If the member/admin wants to add new songs to the playlist, they can search for songs within the SpUStify library and select the desired ones to add. 3. In step 6: If the member/admin wants to remove songs from the playlist, they can select the specific songs to remove from the playlist's song list. 4. In step 6: If the member/admin wants to change the order of songs within the playlist, they can reorder the songs using drag-and-drop or other provided methods. 5. In step 8: If any technical errors occur during the update process, the application should handle exceptions gracefully and inform the member/admin accordingly. |
| Pre-conditions | * The member/admin has an active SpUStify account and is logged into the application. * The member/admin has created at least one playlist in their SpUStify account. |
| Post-conditions | * The playlist is successfully updated with the changes made by the member/admin. * The updated playlist is saved and accessible in the member/admin's account. * Any modifications made to the playlist, such as the name, song additions, removals, or reordering, are reflected in the updated playlist. |

## Use-case: Delete a playlist

| Use case Name | Delete a playlist/album |
| --- | --- |
| Brief description | This use case describes the process of deleting playlists in the SpUStify application. It allows members/admins to remove unwanted playlists from their accounts. |
| Actors | Member, Admin |
| Basic Flow | 1. Member/admin accesses their playlist management section in the SpUStify application. 2. The application displays a list of the user's existing playlists. 3. Member/admin selects the playlist they want to delete or remove songs from. 4. The application presents the playlist details, including the current name and songs. 5. Member/admin chooses the option to delete the entire playlist or remove specific songs.    1. If the member/admin wants to delete the entire playlist:    2. The application prompts the member/admin for confirmation.    3. The member/admin confirms the deletion.    4. The application removes the playlist and all its associated songs from the member/admin account.    5. If the member/admin wants to remove specific songs from the playlist:    6. Member/admin selects the songs they want to remove from the playlist.    7. The member/admin confirms the removal.    8. The application removes the selected songs from the playlist, but the songs remain in the user's library. 6. The updated playlist (if any songs were removed) or the deleted playlist is no longer displayed in the member’s account or platform. |
| Alternative Flows | 1. In step 3: If the playlist selected by the member/admin doesn't exist, the application displays an error message and prompts the user to choose a valid playlist. 2. In step 6/7: If the member/admin cancels the deletion or removal process during confirmation, the application retains the playlist and songs without making any changes. |
| Pre-conditions | * The member/admin has an active SpUStify account and is logged into the application. * The member has created at least one playlist in their SpUStify account. |
| Post-conditions | * If the member/admin deletes an entire playlist, the playlist and all its associated songs are removed (from the member's account if the member does it). * If the member removes specific songs from a playlist, the playlist is updated to exclude the removed songs, but the songs remain in the member’s library. * The updated playlist or the deleted playlist is no longer displayed in the member's account. |

## Use-case: Delete a song

| Use case Name | Delete a song |
| --- | --- |
| Brief description | This use case describes the process of deleting songs in the SpUStify application. It allows artists/admin to delete their songs. |
| Actors | Artist, Admin |
| Basic Flow | 1. Artist/admin accesses the playlist management section in the SpUStify application. 2. The application displays a list of the artist's existing songs. 3. Artist/admin selects the songs they want to delete or remove. 4. The application presents the song’s details, including the current name. 5. Artist/admin chooses the option to remove selected songs.    1. Artist/admin selects the songs they want to remove from the playlist.    2. Artist/admin confirms the removal.    3. The application removes the selected songs on the platform. |
| Alternative Flows | 1. In step 3: If the song selected by the artist/admin doesn't exist, the application displays an error message and prompts the user to choose a valid song. 2. In step 6/7: If the artist/admin cancels the deletion or removal process during confirmation, the application retains the songs without making any changes. |
| Pre-conditions | * The artist/admin has an active SpUStify account and is logged into the application. * The artist has created at least a song in their SpUStify account. |
| Post-conditions | * If the artist/admin removes specific songs from a playlist, the playlist is updated to exclude the removed songs, but the songs remain in the user's library. * The updated song or the deleted song no longer exists. |

## Use-case: Playlist and songs recommendation

| Use case Name | Playlists and songs recommendation |
| --- | --- |
| Brief description | This use case describes the process of providing recommendations for playlists and songs to users in the SpUStify application. The recommendations are based on the user's frequent listening habits and popular songs from the ranking table. The feature suggests playlists that have a high total number of song plays. |
| Actors | User |
| Basic Flow | 1. The user accesses the recommendation section in the SpUStify application. 2. The system shows a list of recommended playlists based on the user's listening habits. 3. The recommendations include playlists that align with the user's preferred genres, similar artists, or related moods. 4. Users can browse through the recommended playlists and select the ones they are interested in. 5. The application presents detailed information about each recommended playlist, including its title, curator, and description. 6. Users can preview the songs within a playlist to get a sense of its content and decide whether to listen to it. 7. User chooses to play a recommended playlist. 8. The application streams the songs from the playlist in the user's playback queue, providing a seamless listening experience. |
| Alternative Flows | 1. In step 2: If the user doesn't have a significant listening history or has limited preferences, the application can provide general recommendations based on popular playlists and songs from the ranking table 2. In step 5: If the user is not satisfied with the initial recommendations, they can provide feedback to the application by indicating preferences for specific genres, artists, or moods. The application can then refine the recommendations based on the provided feedback. |
| Pre-conditions | * The user has an active SpUStify account and is logged into the application. * The user has a history of listening to songs and playlists on the SpUStify application. |
| Post-conditions | * The user has access to a curated list of recommended playlists based on their listening habits and preferences. * The user can explore and discover new songs and playlists that align with their interests. * The user can listen to the recommended playlists seamlessly through the SpUStify application. |

## Use-case: Calculate and generate reports

| Use case Name | Calculate and generate reports |
| --- | --- |
| Brief description | This use case describes the process of calculating and generating reports on website visits, music streams, and music downloads. |
| Actors | Admin |
| Basic Flow | 1. The admin logs into the system and navigates to the reporting module. 2. The system presents the options for generating reports on website visits, music streams, and music downloads. 3. The admin selects the desired report type. 4. The system calculates the necessary metrics, such as total website visits, total music streams, and total music downloads, based on the selected report type. 5. The system displays a comprehensive report containing the calculated metrics. 6. The admin can export the report in a preferred format, such as PDF or CSV if needed. |
| Alternative Flows | If there is no data available for the specified time period or report type:   * 1. The website displays an appropriate message to the admin , indicating the unavailability of data.   2. The admin can choose to modify the report parameters or proceed with an empty report. |
| Pre-conditions | * The system has collected and stored data on website visits, music streams, and music downloads. * The admin has appropriate access privileges to generate reports Administrator accesses the web page |
| Post-conditions | The admin has access to the generated reports for analysis and decision-making purposes. |

## Use-case: Sign in

| Use case Name | Sign in |
| --- | --- |
| Brief description | This use case describes the process of when the visitors want to sign in |
| Actors | Member, Admin |
| Basic Flow | 1. Sign in page shows up 2. Member/admin inputs username and password then click sign in button 3. User is signed in, return to the homepage |
| Alternative Flows | 1. In step 2, guests can use enter instead of clicking the sign-in button |
| Pre-conditions | Member/admin hit the sign-in button on the homepage |
| Post-conditions | Member/admin become active users and go to the homepage if successfully signed in otherwise stay on the sign-in page |

## Use-case: Sign up

| Use case Name | Sign up |
| --- | --- |
| Brief description | This use case describes the process of when the visitors want to sign up |
| Actors | Client, Admin |
| Basic Flow | 1. The sign-up page shows up 2. Client/admin inputs all required information for a new account then click the sign-up button 3. Account created, the client/admin is signed in, return to homepage |
| Alternative Flows | 1. In step 2, client/admin can use enter instead of clicking the sign-up button 2. In step 3, if the username exists, notify the client/admin the account already exists. |
| Pre-conditions | Client/admin hits the sign-up button on the homepage |
| Post-conditions | Client/admin become users and go to the homepage if successfully signed up otherwise stay in Sign up page |

## Use-case: Delete account

| Use case Name | Delete account |
| --- | --- |
| Brief description | This use case describes the process of when the admin wants to delete the accounts |
| Actors | Admin |
| Basic Flow | 1. Delete account page shows up 2. The system presents the admin with a list of accounts to delete. 3. The admin can search for an account to delete in the search bar. 4. After choosing an account for deletion, the system will ask the admin for confirmation. If the admin confirms, the account will be deleted. |
| Alternative Flows | 1. In step 2, admin can click the cancel button to cancel changing the profile. |
| Pre-conditions | Admin choose the delete account option in the user profile |
| Post-conditions | An account is deleted |

## Use-case: Edit account

| Use case Name | Edit account |
| --- | --- |
| Brief description | This use case describes the process of when the admin wants to edit/update the accounts |
| Actors | Admin |
| Basic Flow | 1. Update profile page shows up 2. The system presents the admin with a form or interface to modify their profile information. 3. The admin makes the desired changes to the profile, such as updating the name, profile picture, or contact information,.... 4. Once the admin has finished making the changes, he submits the updated profile information. 5. The system displays a confirmation message to the admin, indicating that their profile has been successfully updated. |
| Alternative Flows | 1. In step 2, admin can click the cancel button to cancel changing the profile. 2. In step 3, if the admin wishes to change the password, he can click the change password button. Then the admin has to input the password twice before submitting to the system. |
| Pre-conditions | Admin choose the edit account option in the user profile |
| Post-conditions | The account is or is not updated based on the validity of the submitted information |

## 