SpUStify

Use-Case Specification

Version 1.0

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 30/6/2023 | 1.0 | Initial version | 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: User search 4](#_heading=h.2et92p0)

[2.2 Use-case: Artist search 5](#_heading=h.yafshp8z6za8)

[2.3 Use-case: View detail 5](#_heading=h.6a0mnv25ixx9)

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

[2.5 Use-case: View Favorite Songs and Playlists 6](#_heading=h.r8gack999gkp)

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

[2.7 Use-case: Play song 7](#_heading=h.c1xbuuujtgdf)

[2.8 Use-case: Uploading songs 8](#_heading=h.r1q37mu81qyy)

[2.9 Use-case: Create playlists/albums 8](#_heading=h.njw102am1b1e)

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

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

[2.12 Use-case: Delete playlists/albums 11](#_heading=h.yeg66mywunaq)

[2.13 Use-case: Delete songs 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)

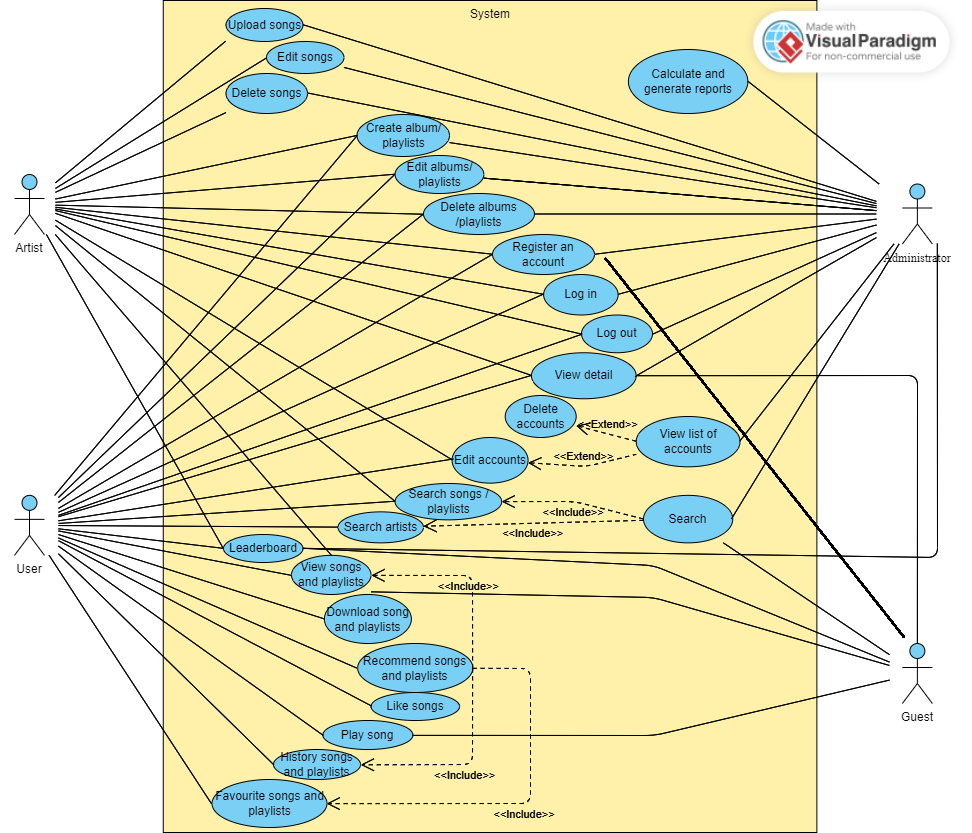
[2.16 Use-case: Log in 15](#_heading=h.uo33q3d50dqo)

[2.17 Use-case: Register an account 15](#_heading=h.yf4svzsrjc1v)

[2.18 Use-case: Log out 16](#_heading=h.w3bzpcgj24x5)

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

# Use-case Model



# Use-case Specifications

## Use-case: User search

| Use case Name | User Search |
| --- | --- |
| Brief description | This use case describes the user or guest searching songs and playlists based on keywords. |
| Actors | User, Guest,Admin |
| Basic Flow | 1. Users or guests enter keywords in the 'Search' field at the navigation bar. 2. The user or guest then clicks the 'Search' button to initiate the search. 3. The website displays all results related to the entered keyword, including songs and playlists. |
| 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. The user or guest can navigate back to the homepage. |
| Pre-conditions | The user (Guest, 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: Artist search

| Use case Name | Artist search |
| --- | --- |
| Brief description | This use case describes the artist searching for songs of them or playlists of them based on keywords. |
| Actors | Artist |
| Basic Flow | 1. At the navigation bar, the artist enters keywords in the 'Search' field. 2. The artist then clicks the 'Search' button to initiate the search. 3. The website displays all results related to the entered keyword, including songs and playlists of the artist. |
| 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 artist. 2. The artist can navigate back to the homepage. |
| Pre-conditions | Artist can be anywhere on the website |
| Post-conditions | The search results for the keyword was successfully retrieved and displayed on the website, or an announcement is made if the keyword cannot be found. |

## 

## Use-case: View detail

| Use case Name | View detailed information about song, playlist, artitst |
| --- | --- |
| Brief description | This use case describes the process of accessing information about a song (playlist, artist). |
| Actors | User, Artist, Guest,Admin |
| Basic Flow | 1. At the search result or homepage, the user or guest selects a specific playlist, or artist by clicking on it. (except Artist just views detailed information of its song, playlist, and artist.) 2. The website navigates to the corresponding page and displays comprehensive information about the selected item. |
| Alternative Flows |  |
| Pre-conditions | The user or guest or artist must be on the homepage or search result page |
| Post-conditions | The user or guest or artist successfully goes to the playlist (song, artist) page with information about it |

## Use-case: Leaderboard

| Use case Name | Leaderboard |
| --- | --- |
| Brief description | Ranking of songs by likes, by listens. |
| Actors | User, Guest, Artist, Admin |
| Basic Flow | 1. At the navigation bar, the user (guest, artist, 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 the user (guest, artist) selects the “Leaderboard” by likes, the website will navigate to the leaderboard page by number of likes |
| Pre-conditions | The user (guest, artist, admin) can be anywhere on the website |
| Post-conditions | The website navigates to leaderboard page |

## 

## Use-case: View Favorite Songs and Playlists

| Use case Name | View Favorite Songs and Playlists |
| --- | --- |
| Brief description | This use case allows a user to view their favorite songs and playlists and download music from website |
| Actors | User |
| Basic Flow | 1. The user selects the "Favorites" or "My Library" option from the navigation menu. 2. The web application retrieves the user's favorite songs and playlists from the database. 3. The web application displays the user's favorite songs and playlists on the screen. 4. 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 retrieves and 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. Album name    4. 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: Play song

| Use case Name | Play song |
| --- | --- |
| Brief description | This use case describes the user, guest, admin play(listen) a song |
| Actors | User, Guest, Admin |
| Basic Flow | 1. The user (guest, admin) chooses a song and clicks on it 2. The website navigates to the song on page 3. The user (guest, admin) 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 user (guest, admin) skips to another song in the playlist, the website updates the currently playing song and continues playback from the new position. 2. If the user (guest, admin) adjusts the volume, the website modifies the audio output accordingly. 3. If the user (guest, admin) 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 user 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 user (Guest, Admin) must be on the homepage or search result page to select song or playlist |
| Post-conditions | The user can enjoy listening to music while browsing the website.  The website allows seamless control over the music playback.  The user can switch songs or playlists and adjust the volume as desired. |

## 

## Use-case: Uploading songs

| Use case Name | Uploading songs |
| --- | --- |
| Brief description | This use case specification describes the functionality and requirements for a web application that allows users to upload songs |
| Actors | Artist, Admin |
| Basic Flow | 1. The user navigates to the web application's song upload section. 2. The user 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 user selects the audio file of the song from their local device 5. The web application validates the uploaded file to ensure it is a valid audio file and adheres to any specified format requirements 6. The web application associates the uploaded file with the provided song details and stores it in a secure location 7. The web application confirms the successful upload and redirects the user to the song's page |
| Alternative Flows | * Invalid File Upload: * If the user 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 user 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 user has an active SpUStify account and is logged into the application * The user should have a valid internet connection * The user's device should support file upload and download 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 playlists/albums

| Use case Name | Create playlists/albums |
| --- | --- |
| Brief description | This use case specification describes the functionality and requirements for a web application that allows users to create their playlists and artists to create their albums. |
| Actors | Artist, User, Admin |
| Basic Flow | 1. Create a Music Playlist 2. The user navigates to the web application's playlist upload section 3. The user selects the option to upload a new playlist 4. The web application presents a form to provide details about the playlist, including a title, description, and tags 5. The user selects the music files that constitute the playlist from their local device 6. The web application validates the uploaded files to ensure they are valid audio files and comply with any specified format requirements 7. The web application associates the uploaded files with the provided playlist details and stores them in a secure location 8. The web application confirms the successful upload and redirects the user to the playlist's page |
| Alternative Flows | * Invalid File Upload: * If the user attempts to upload a file that is not a valid audio file, the web application displays an error message 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 user 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 user has an active SpUStify account and is logged into the application * The user should have a valid internet connection * The user's device should support file upload and download functionality |
| Post-conditions | * The created playlists/albums are securely stored in the web application's storage system * Users can access and manage the uploaded playlists through the web application |

## Use-case: Edit song

| Use case Name | Edit song |
| --- | --- |
| Brief description | This function allows users to modify the details and metadata of a song. Users can update information such as the song title, artist name, album name |
| Actors | Artist, Admin |
| Basic Flow | 1. User searches or browses for the desired song they want to edit 2. User 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 (admin) modifies the desired fields, such as updating the song title, artist name, or album name 5. The artist (admin) saves the changes by clicking on the "Save" or "Update" button |
| Alternative Flows | 1. If the changes made by the user are invalid ( Duplicate song’ name) the website displays an error message indicating the validation failure |
| Pre-conditions | The Artist (Admin) 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/album

| 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 users to modify the playlist's name, add or remove songs, and change the order of songs within the playlist. |
| Actors | User, Artist, Administrator |
| Basic Flow | 1. 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, add new songs, 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 reflected in the user's account. |
| Alternative Flows | 1. In step 3: If the playlist selected by the user doesn't exist, the application displays an error message and prompts the user to choose a valid playlist. 2. In step 6: If the user 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 user 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 user 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 user accordingly. |
| Pre-conditions | * The user (Artist, Administrator) has an active SpUStify account and is logged into the application. * The user (Artist, Administrator) has created at least one playlist in their SpUStify account. |
| Post-conditions | * The playlist is successfully updated with the changes made by the user. * The updated playlist is saved and accessible in the user'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 playlists/albums

| Use case Name | Delete playlists/albums |
| --- | --- |
| Brief description | This use case describes the process of deleting playlists in the SpUStify application. It allows users to remove unwanted playlists from their accounts. |
| Actors | User, Artist, Admin |
| Basic Flow | 1. User accesses their playlist management section in the SpUStify application. 2. The application displays a list of the user's existing playlists. 3. User 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. User chooses the option to delete the entire playlist or remove specific songs.    1. If the user wants to delete the entire playlist:    2. The application prompts the user for confirmation.    3. The user confirms the deletion.    4. The application removes the playlist and all its associated songs from the user's account.    5. If the user wants to remove specific songs from the playlist:    6. User selects the songs they want to remove from the playlist.    7. The user 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 user's account. |
| Alternative Flows | 1. In step 3: If the playlist selected by the user 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 user cancels the deletion or removal process during confirmation, the application retains the playlist and songs without making any changes. |
| Pre-conditions | * The user has an active SpUStify account and is logged into the application. * The user has created at least one playlist in their SpUStify account. |
| Post-conditions | * If the user deletes an entire playlist, the playlist and all its associated songs are permanently removed from the user's account. * If the user 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 playlist or the deleted playlist is no longer displayed in the user's account. |

## Use-case: Delete songs

| Use case Name | Delete Songs |
| --- | --- |
| Brief description | This use case describes the process of deleting songs in the SpUStify application. It allows users to delete individual songs from their playlists. |
| Actors | Artist, Admin |
| Basic Flow | 1. Artist (admin) accesses their playlist management section in the SpUStify application. 2. The application displays a list of the artist's existing playlists. 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 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 user's account. |
| Alternative Flows | 1. In step 3: If the song selected by the user 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 user cancels the deletion or removal process during confirmation, the application retains the playlist and 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 user deletes the last song, the playlist of the songs are permanently removed from the user's account. * If the user 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 is no longer displayed in the user's account. |

## 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, Admin |
| Basic Flow | 1. The user accesses the recommendation section in the SpUStify application. 2. The application analyzes the user's listening history, including frequently played songs and preferred genres. 3. The application generates a list of recommended playlists based on the user's listening habits. 4. The recommendations include playlists that align with the user's preferred genres, similar artists, or related moods. 5. Users can browse through the recommended playlists and select the ones they are interested in. 6. The application presents detailed information about each recommended playlist, including its title, curator, and description. 7. Users can preview the songs within a playlist to get a sense of its content and decide whether to listen to it. 8. User chooses to play a recommended playlist. 9. 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 | Administrator |
| Basic Flow | 1. The administrator 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 administrator selects the desired report type. 4. The system retrieves the relevant data from the database. 5. The system calculates the necessary metrics, such as total website visits, total music streams, and total music downloads, based on the selected report type. 6. The system generates a comprehensive report containing the calculated metrics. 7. The report is displayed to the administrator on the system interface. 8. The administrator 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 administrator, indicating the unavailability of data.   2. The administrator 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 administrator has appropriate access privileges to generate reports Administrator accesses the web page |
| Post-conditions | The administrator has access to the generated reports for analysis and decision-making purposes. |

## Use-case: Log in

| Use case Name | Sign in |
| --- | --- |
| Brief description | This use case describes the process of when the visitors want to sign in |
| Actors | User, Artist, Guest, Admin |
| Basic Flow | 1. Sign in page shows up 2. Visitor inputs username and password then click sign in button 3. Website checks for the validity of account 4. User is signed in, return to the homepage |
| Alternative Flows | 1. In step 1, guests can choose to sign up or update the password to go to sign up or update password page 2. In step 2, guests can use enter instead of clicking the sign-in button 3. In step 3, if the username doesn’t exist, notify the guests the account doesn’t exist. If the username exists but the password is wrong, notify the guests the password is wrong |
| Pre-conditions | Guests hit the sign-in button on the homepage |
| Post-conditions | Guests become users and go to the homepage if successfully signed in otherwise stay on the sign-in page |

## Use-case: Register an account

| Use case Name | Sign up |
| --- | --- |
| Brief description | This use case describes the process of when the visitors want to sign up |
| Actors | User, Artist, Guest, Admin |
| Basic Flow | 1. The sign-up page shows up 2. Guests input all required information for a new account then click the sign-up button 3. The website checks for the validity of the account and adds the account to the database 4. Account created, the user is signed in, return to homepage |
| Alternative Flows | 1. In step 1, guests can choose to sign in or update the password to go to sign in or update password page 2. In step 2, guests can use enter instead of clicking the sign-up button 3. In step 3, if the username exists, notify the guests the account already exists. |
| Pre-conditions | Guests hits the sign-up button on the homepage |
| Post-conditions | Guests become users and go to the homepage if successfully signed up otherwise stay in Sign up page |

## Use-case: Log out

| Use case Name | Log out |
| --- | --- |
| Brief description | This use case describes the process of logging out of the website, allowing the user to securely end their session and protect their account. |
| Actors | User, Artist, Admin |
| Basic Flow | 1. The users open the website 2. The users navigate to their account settings or profile section. 3. The user locates the “Log out” option with the account setting menu. 4. The user selects the “Log out” option. 5. Website terminates the user’s current session and logs them out of the application or website. 6. The user is redirected to the login page or a “Logged out” confirmation screen. |
| Alternative Flows | None. |
| Pre-conditions | * The user must have an active account. * The user must be logged in to the account. |
| Post-conditions | * The user’s session is terminated, and they are logged out of the website. * The user’s personal information, playlists, and preferences are protected from unauthorized access. * The user will need to log in again to access their account. |

## Use-case: Edit account

| Use case Name | Edit account |
| --- | --- |
| Brief description | This use case describes the process of when the visitors want to edit/update their accounts |
| Actors | User, Artist, Admin |
| Basic Flow | 1. Update profile page shows up 2. The system presents the user with a form or interface to modify their profile information. 3. The user makes the desired changes to their profile, such as updating their name, profile picture, or contact information,.... 4. Once the user has finished making the changes, they submit the updated profile information. 5. The system processes the updated profile information and saves it in the database. 6. The system displays a confirmation message to the user, indicating that their profile has been successfully updated. |
| Alternative Flows | 1. In step 2, users can click the cancel button to cancel changing their profile. 2. In step 3, the system will check if the submitted information is valid or not. If not valid, the system will cancel updating the user's profile and send a message to notify the user their submitted information is not valid and has been canceled. 3. In step 3, if the users wish to change their password, they can click the change password button. Then the users have to input their password twice before submitting to the system. The system checks for the validity of their new passwords and updates the password in the database. |
| Pre-conditions | Users choose the edit account option in their user profile |
| Post-conditions | Users’ account is or is not updated based on the validity of their submitted information |

## 