Group02

Beatify
Use-Case Specification: Beatify Cases

Version 1.1

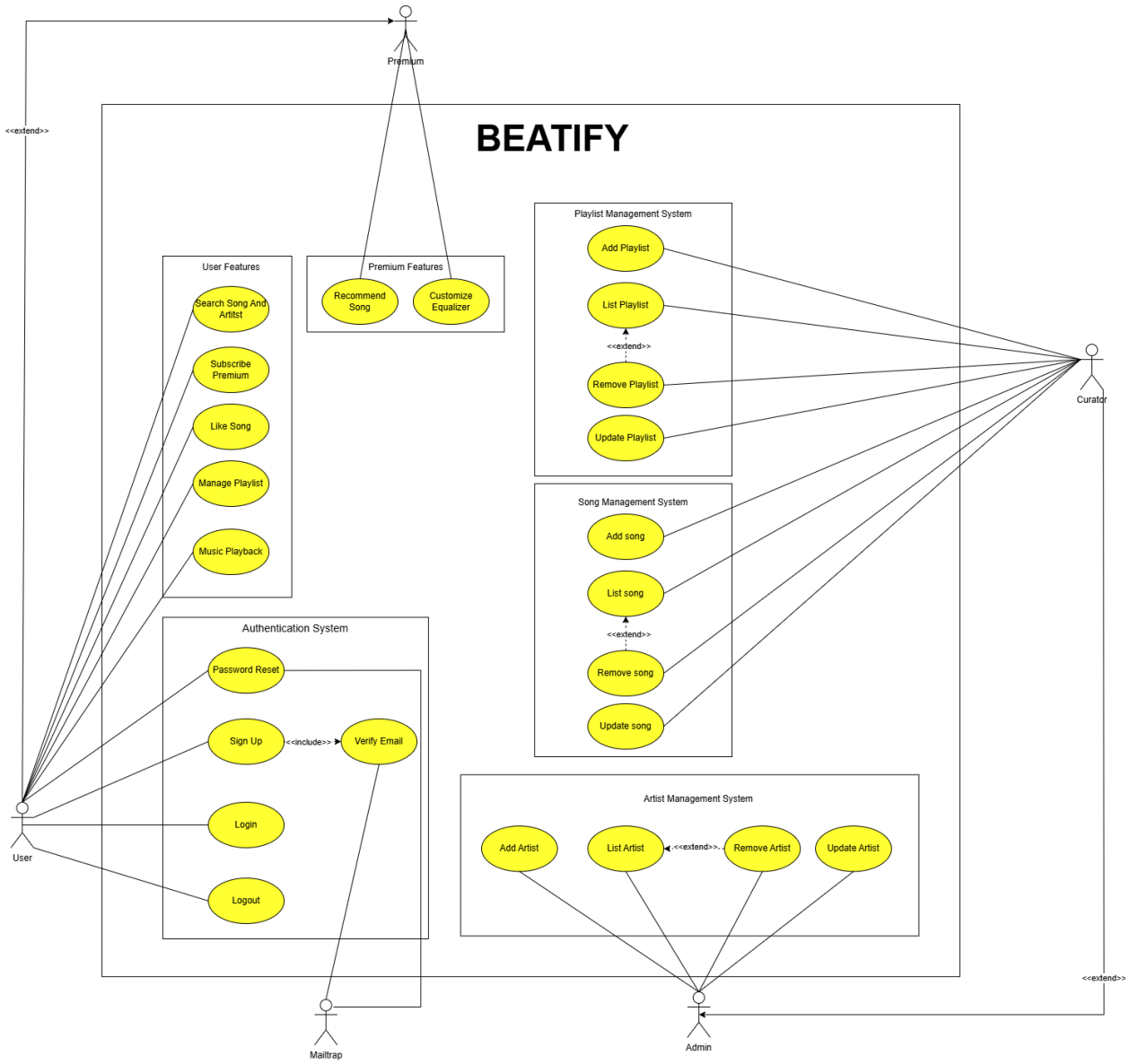
Revision History

Date	Version	Description	Author
13/11/2024	1.0	Initial version. Created use-case specifications including necessary fields, basic and alternative flows, and specific preconditions and postconditions.	- Lê Gia Huy: Customize Equalizer, Music Playback, Reset Password, Recommend Song, User Login, User Sign Up. - Nguyễn Duy Bảo: Music Playback, User Logout, Verify Email, Add Playlist, List Playlist, Remove Playlist, Update Playlist. - Phạm Hà Hiếu: Like Song, Manage User Playlist, Search Song And Artist. - Trần Trung Hiếu: Add Song, Remove Song, Subscribe Premium, Update Song. - Võ Ngọc Khoa: Add Artist, List Artist, Remove Artist, Subscribe Premium, Update Artist.
25/11/2024	1.1	Updated Use-case Model Diagram and added new alternative flows	- Lê Gia Huy, Nguyễn Duy Bảo, Phạm Hà Hiếu, Trần Trung Hiếu, Võ Ngọc Khoa

Table of Contents

Use-Case Specification: Beatify

1. Use-Case Model



2. Use-Case Specification

Use-case ID	Use-case name	Use-case ID	Use-case name
UC001	Sign Up	UC013	List Artist
UC002	Login	UC014	Remove Artist
UC003	Logout	UC015	Update Artist
UC004	Verify Email	UC016	Add Song
UC005	Reset Password	UC017	Remove Song
UC006	Like Song	UC018	Update Song
UC007	Manage User Playlist	UC019	Add Playlist
UC008	Music Playback	UC020	List Playlist
UC0009	Subscribe Premium	UC021	Remove Playlist
UC010	Search Song And Artist	UC022	Update Playlist
UC011	Customize Equalizer	UC023	Recommend Song
UC012	Add Artist		

2.1 User

2.1.1 Sign Up

Use-case ID	**UC001**
Use-case name	**User Sign Up**
Brief description	Allows a new user to register an account by providing an email, password, and name, and agreeing to the terms and conditions.
Actor	User
Basic flow	1. The user submits their email, password, name, and ticks the "Agree to Terms and Conditions" checkbox. 2. The system checks if all required fields are filled and if the terms and conditions checkbox is ticked. 3. The system verifies if the email already exists. 4. If the email does not exist, the system hashes the password and creates a new user record. 5. A verification token is generated, and a verification email is sent. 6. The system sets a cookie with a session token and sends a success response.

Alternative flow	**Alternative flow 1: User Already Exists** 1. If the email already exists, the system responds with an error message 2. Users can try with different email or log in if an account is already exists **Alternative flow 2: Terms and Conditions Not Agreed** 1. If the terms and conditions checkbox is not ticked, the system displays an error message requiring the user to agree before proceeding 2. Users can agree to the terms and conditions and retry **Alternative flow 3: Invalid details** 1. System displays an error message indicating the invalid details 2. Users can correct the details and retry
Special Requirements	- Password hashing using *bcryptjs* for security. - Email verification token expires in 24 hours.
Preconditions	- User is not already registered.
Postconditions	- User account is created and pending email verification
Extension Points	- None.

2.1.2 Login

Use-case ID	**UC002**
Use-case name	**User Login**
Brief description	Allows a registered user to log in by providing their email and password.
Actor	User, Admin, Curator
Basic flow	1. The user submits their email and password to log in 2. The system checks if the user exists in the database 3. If the user exists, the system compares the provided password with the stored hashed password. 4. If the password is valid, a session token is generated and set as a cookie. 5. The user's last login date is updated, and the system sends a success response.
Alternative flow	**Alternative flow 1: Invalid Credentials** 1. If the email or password is incorrect, the system responds with an error message 2. Users can retry entering their credentials.
Special Requirements	- Password comparison using *bcryptjs.compare*. - Session token management for authentication.
Preconditions	- User is registered.

Postconditions	- User is logged in and the session token is set.
Extension Points	- None.

2.1.3 Logout

Use-case ID	**UC003**
Use-case name	**User Logout**
Brief description	Allows a logged-in user to log out, clearing their session.
Actor	User, Admin, Curator
Basic flow	1. The user initiates a logout request. 2. The system clears the session token cookie. 3. The system sends a success response indicating that the user is logged out.
Alternative flow	- None
Special Requirements	- Clear session token cookie on logout.
Preconditions	- User is logged in.
Postconditions	- User is logged out, and the session token is removed
Extension Points	- None.

2.1.4 Verify Email

Use-case ID	**UC004**
Use-case name	**Verify Email**
Brief description	Allows a new user to verify their email address by entering a verification code sent to their email.
Actor	User
Basic flow	1. The user enters the verification code sent to their email. 2. The system checks if the code matches and is within the expiration period. 3. If valid, the system updates the user's status to verified, clears the verification token, and uses Mailtrap to send a welcome email.
Alternative flow	**Alternative flow 1: Invalid Code** 1. The system returns an error message: "The code you entered is invalid. Please try again."

	2. The user is prompted to re-enter the correct code. **Alternative flow 2: Expired Code** 1. The system returns an error message: "The code has expired. Please request a new verification code." 2. The user is given the option to request a new verification code. 3. The system generates a new verification code and sends the new code to the user's email address.
Special Requirements	- Verification token expiration set to 24 hours. - Mailtrap is used as the email delivery service to handle verification and welcome emails.
Preconditions	- User has signed up but is not verified
Postconditions	- User's email is verified, and they are granted full access.
Extension Points	- None.

2.1.5 Reset Password

Use-case ID	**UC005**
Use-case name	**Reset Password**
Brief description	Allows a user who has forgotten their password to request a password reset email with a link to reset their password
Actor	User
Basic flow	1. The user submits their email to request a password reset. 2. The system checks if the user exists in the database. 3. If the user exists, the system generates a reset token and sets an expiration time. 4. The system uses Mailtrap to send a password reset email containing a link with the reset token. 5. Upon clicking the link, the user is directed to a reset password page where they provide a new password. 6. The system hashes and updates the user's password, then clears the reset token.
Alternative flow	**Alternative flow 1: User Not Found** 1. If the email does not match any user, the system responds with an error message.
Special Requirements	- Password hashing using *bcryptjs*. - Reset token expiration set to 1 hour. - Mailtrap is used as the email delivery service to handle password reset emails.
Preconditions	- User is registered.

Postconditions	- User's password is reset.
Extension Points	- None.

2.1.6 Like Song

Use-case ID	**UC006**
Use-case name	**Like Song**
Brief description	The Like Song enables users to mark songs as favorites for quick access and personalized recommendations.
Actor	User
Basic flow	1. The user clicks the heart or "like" icon on a song to express their preference for the content. 2. In response, the system registers the item as liked, marking it accordingly within the user's account for future reference. 3. The selected song is then added to the user's "Liked" collection, ensuring it is easily accessible for future listening and management. 4. Simultaneously, the system updates the user's preference data, incorporating this new interaction to refine future recommendations and enhance the overall personalized experience.
Alternative flow	**Alternative flow 1: Unlike Song** 1. User clicks the heart icon on an item that is already liked 2. The system removes it from the user's "Liked" collection. This update ensures the item is no longer associated with the user's preferences, maintaining an accurate and current list of liked content for a more personalized experience
Special Requirements	- Instant response to like/unlike actions.
Preconditions	- User must be logged in - Item must be available in system
Postconditions	- Liked items appear in user's library - Recommendations updated based on likes
Extension Points	- Integration with music discovery - Export liked songs

2.1.7 Manage User Playlist

Use-case ID	**UC007**

Use-case name	**Manage User Playlist**
Brief description	The Manage Playlist allows users to create, edit, and manage their music playlists. Users can add songs to playlists, remove songs, rename playlists.
Actor	User
Basic flow	1. The user initiates the playlist creation process by clicking on the "Create New Playlist" button. 2. In response, the system presents the user with a playlist creation form, allowing them to input necessary details. 3. The user proceeds to enter a name for the playlist, and optionally, includes a brief description to further define the playlist's theme or purpose. 4. The user then searches through the music library, selecting individual songs to add to the playlist based on personal preference or mood. 5. After selecting the desired tracks, the user adds them to the playlist, ensuring that the collection of songs aligns with their intended playlist concept. 6. Once the playlist is finalized, the system securely saves the playlist and provides the user with a confirmation message, confirming the successful creation of the playlist. 7. For further customization, the user can reorder the songs within the playlist.
Alternative flow	**Alternative flow 1: Edit Existing Playlist** 1. The user navigates to their playlist library and selects the playlist they wish to edit. 2. The system retrieves the selected playlist and displays its current details and song list in an editable format. 3. The user modifies the playlist details. 4. The system updates the playlist in real-time to reflect the user’s modifications. 5. The user confirms the changes by clicking a "Save" or "Update Playlist" button. 6. The system securely saves the updated playlist and displays a confirmation message, e.g., *"Your playlist has been successfully updated."*
Special Requirements	- Maximum 500 songs per playlist. - Playlist names limited to 100 characters. - Real-time sync across devices
Preconditions	- User must be logged in. - System must have songs database accessible.
Postconditions	- New playlist is saved to the user's account . - Playlist is available for playback - Changes are synced across all user's devices.

Extension Points	- None.

2.1.8 Music Playback

Use-case ID	**U008**
Use-case name	**Music Playback**
Brief description	This use case allows users to stream music directly on Beatify with essential playback controls, such as play, pause, next, previous, shuffle, and repeat. Users can adjust volume, mute/unmute, change playback speed (premium users only), and seamlessly switch between songs or playlists.
Actor	User
Basic flow	1. User initiates playback: The user selects a song or playlist and clicks the Play button; the system triggers playSong, starts the song, and sets isPlaying to true. 2. User controls playback: a. Pause: The user clicks the Pause button; the system pauses the audio and sets isPlaying to false. b. Play: The user clicks the Play button after pausing; the system resumes playback from where it was paused. 3. User navigates tracks: a. Next Track: The user clicks the Next button; the system skips to the next song or selects a random track if shuffle is enabled. b. Previous Track: The user clicks the Previous button; the system either restarts the current song or plays the previous one, depending on the playback position. 4. User adjusts volume: The user changes the volume slider; the system updates the volume and adjusts the audio element's volume. 5. Toggle Mute: The user clicks the Mute button; the system toggles the audio volume between muted and the last set volume. 6. Playback Modes: Shuffle: The user enables or disables shuffle mode; the system changes the playback order to random if shuffle is enabled. 7. Playback Speed (For Premium Users Only): The user adjusts the playback speed slider; the system updates the playback speed (1x, 1.5x, 2x, etc.) depending on the selected value.
Alternative flow	**Alternative flow 1: Track Unavailable** 1. If a track is not found or cannot be played, the system skips to the next available track and displays an error message.
Special Requirements	- The application requires a reliable internet connection for uninterrupted streaming. - The audio player must support cross-page navigation without interrupting playback. - The application must handle exceptions, such as missing audio files or unsupported audio formats. - Playback Speed is available only for premium users. Non-premium

	users cannot access this feature.
Preconditions	- The user must be logged into their Beatify account. - The user must have selected a song, album, or playlist. - A stable internet connection is required for streaming. - For playback speed control: The user must have a premium account.
Postconditions	- Playback settings, such as the last played song, volume, and playback modes (shuffle, repeat), are saved for continuity. - The user can resume playback with the previous settings upon returning. - For playback speed: The selected speed setting is saved and reapplied for subsequent sessions.
Extension Points	- The *getSongsData* and *getPlaylistsData* functions provide additional track and playlist information, which can extend functionality for dynamic playlist loading and song recommendations.

2.1.9 Subscribe Premium

Use-case ID	**UC009**
Use-case name	**Subscribe Premium**
Brief description	This use case allows users to subscribe to a music website by selecting a subscription plan and completing the payment process. Upon successful payment, the user gains access to premium content and features according to the chosen plan.
Actor	Free User
Basic flow	1. The use case begins when a user selects the "Subscribe" option on the music website. 2. The system displays available subscription plans, such as monthly, annual, or family plans, with details about features and prices. 3. The user selects a subscription plan and proceeds to the payment page. 4. The user enters payment information or chooses a saved payment method. 5. The system validates the payment details and processes the transaction through a payment gateway. 6. Upon successful payment, the system activates the user's subscription. 7. The system sends a confirmation message and email, summarizing the subscription plan and payment details. 8. The user gains access to premium features.
Alternative flow	**Alternative flow 1: Insufficient Payment Information** 1. The user submits incomplete or invalid payment information. 2. The system prompts the user to correct it. 3. The user re-enters or corrects the payment details and resubmits **Alternative flow 2: Payment Failure**

	1. The payment gateway returns an error. 2. The system notifies the user of the failure. 3. The user may choose to retry with a different payment method or try again later. **Alternative flow 3: Cancellation During Subscription** 1. The user decides to cancel during the subscription process. 2. The system saves any entered information but does not process the payment. 3. The user is returned to the main page without any changes to their account.
Special Requirements	- Subscription plans and prices should be easily configurable by administrators - Subscription-related communications (confirmation emails) should align with brand guidelines and regional compliance standards. - The system must handle payment information securely.
Preconditions	- The user must have an account on the music website and be logged in. - The payment gateway service must be available.
Postconditions	- Upon successful subscription, the user's account is updated to reflect the premium membership status. - If payment fails, no changes are made to the user's account, and the subscription remains inactive.
Extension Points	- None.

2.1.10 Search Song And Artist

Use-case ID	**UC010**
Use-case name	**Search Song And Artist**
Brief description	The Search feature allows users to find songs, artists, and playlists using keywords search.
Actor	Users
Basic flow	1. The user clicks on the search bar to begin their search. 2. The user enters relevant search terms related to the content they are looking for. 3. As the user types, the system displays real-time search suggestions, helping to refine their query with relevant options. 4. The system returns categorized results, sorting content into distinct sections such as Songs, Artists, and Playlists, providing the user with a comprehensive overview of available options. 5. The user can then click on any of the displayed results to access the corresponding content directly, allowing for seamless navigation and exploration.

Alternative flow	**Alternative flow 1: No Results Found** 1. The user enters search terms and initiates the search. 2. The system attempts to match the search terms with available content. 3. If no matching results are found, the system displays a "No results found" message. 4. The system suggests alternative search terms or popular searches to help the user refine their query. 5. The user can choose from the suggested alternatives or modify their search to try again.
Special Requirements	- Search response time under 500ms. - Support for partial matching. - Multi-language search support.
Preconditions	- Search index must be updated. - Minimum 3 characters for search.
Postconditions	- Search results displayed.
Extension Points	- None.

2.1.11 Customize Equalizer

Use-case ID	**UC011**
Use-case name	**Customize Equalizer**
Brief description	This use case allows premium users of Beatify to customize the sound quality of their music playback experience through an equalizer feature. Users can adjust frequency bands (e.g., bass, midrange, treble) to enhance their listening experience based on personal preferences or to match specific genres.
Actor	Premium User
Basic flow	1. The user navigates to the Equalizer settings in the playback interface. 2. The system presents the user with multiple frequency band sliders (e.g., bass, midrange, treble) for customization. 3. User Adjusts Frequency Bands: a. The user interacts with the sliders to modify specific frequency levels. b. Adjustments include increasing or decreasing bass, midrange, and treble levels according to preference. 4. As the user adjusts each frequency band, the system immediately applies the changes to the audio output, providing real-time feedback. 5. User Saves or Resets Settings: a. The user can save the customized equalizer settings, allowing for future sessions to start with the same configuration. b. Alternatively, the user may reset to default settings if they want to return to the standard audio output.
Alternative flow	**Alternative flow 1: User Cancels Equalizer Adjustments**

	1. The user accesses the Equalizer screen to adjust audio settings. 2. The user decides not to proceed with the adjustments. 3. The user exits the Equalizer screen without saving any changes. 4. The system reverts any temporary adjustments made during the session, ensuring no changes are applied to the audio settings. **Alternative flow 2: Unsupported Device or Browser** 1. The user attempts to access the Equalizer features on their device or browser. 2. The system checks for compatibility with real-time audio adjustment. 3. If the device or browser does not support the feature, the system displays a notification to inform the user. 4. The equalizer options are disabled, but the user can still access other playback features.
Special Requirements	- The equalizer feature should only be accessible to premium users. - Real-time audio processing capabilities are required for immediate playback feedback. - The equalizer interface should be intuitive and responsive, allowing for seamless adjustments.
Preconditions	- The user must be logged into their premium Beatify account. - The user must have an active song playing or queued for playback. - The user's device/browser should support real-time audio adjustments.
Postconditions	- If the user saves the settings, the customized equalizer configuration will be stored for future sessions. - If the user resets to default, the equalizer settings revert to standard playback, and no custom settings are saved.
Extension Points	- Genre-Based Presets: Offer preset equalizer settings optimized for different music genres (e.g., Rock, Jazz, Pop), which users can choose to apply.

2.2 Admin

2.2.1 Add Artist

Use-case ID	**UC012**
Use-case name	**Add Artist**
Brief description	This use case allows an admin to add a new artist to the system. The user uploads the artist's information, including name, description, background color, and profile picture.
Actor	Admin
Basic flow	1. The admin initiates the "Add Artist" action. 2. The system prompts the user to enter Artist detail.

	3. The admin inputs the artist's name, description, and background color and selects an image file to upload. 4. The admin clicks on the “ADD” button to save the Artist. 5. The system saves the new artist record in the database. 6. The system confirms that the artist has been added.
Alternative flow	**Alternative flow 1: Database Save Failure** 1. The admin attempts to save the artist's data to the database. 2. The system encounters an issue while saving the data. 3. The system captures the error and logs it for further investigation. 4. The system displays an error message to the user, indicating that the artist was not added successfully.
Special Requirements	- The request should be validated to ensure the admin is authorized to add an artist. - All image uploads must be optimized to reduce file size without compromising quality to ensure performance.
Preconditions	- The admin is authenticated and authorized to add an artist. - The required artist data (name, description, background color, and image) is available.
Postconditions	- A new artist is added to the List Artist. - In case of a failure, appropriate error logs and messages are generated for troubleshooting.
Extension Points	- None.

2.2.2 List Artist

Use-case ID	**UC013**
Use-case name	**List Artist**
Brief description	This use case allows an admin to retrieve a list of all artists from the database and view their details. It helps admins quickly access and manage artist information.
Actor	Admin
Basic flow	1. The use case begins when an admin selects the “List Artists” option on the application. 2. The system sends a request to the server to retrieve all artists. 3. The server accesses the database and retrieves all artist records. 4. If successful, the system displays a list of all artists with their details, such as name, description, profile picture, and background color. 5. The use case ends when the list of artists is successfully presented to the admin
Alternative flow	**Alternative flow 1: Database Error** 1. The system attempts to retrieve data from the database. 2. If there is an issue with the database connection (e.g., connection

	failure), the system notifies the admin that the list could not be retrieved. 3. The admin is given the option to retry the request. 4. If the issue persists, the admin is prompted to contact support for further assistance. **Alternative flow 2: Empty List** 1. The system attempts to retrieve the list of artists from the database. 2. If no artists are found in the database, the system displays a message indicating that no artists are currently available. 3. The admin can choose to add a new artist via “Add Artist” if needed.
Special Requirements	- The list of artists displayed should be up-to-date, reflecting any recent additions, deletions, or modifications. - The list of artists should load within a few seconds to maintain a smooth user experience, even with a large dataset.
Preconditions	- The admin must be logged in to access the artist list. - The database and server must be online and responsive.
Postconditions	- Upon successful completion, the admin views a list of all artists. - If unsuccessful, the system provides feedback to the admin regarding the nature of the error.
Extension Points	- None.

2.2.3 Remove Artist

Use-case ID	**UC014**
Use-case name	**Remove Artist**
Brief description	This use case allows an admin to remove an artist from the database.
Actor	Admin
Basic flow	1. The admin initiates the "List Artist" action. 2. The system displays the artist list and “Action” column. 3. The admin clicks on the “x” button to remove the artist. 4. The system prompts the user to confirm. 5. The admin confirms the action. 6. The system responds with a success message indicating that the artist has been removed.
Alternative flow	**Alternative flow 1: Admin Cancels Artist Removal** 1. The admin initiates the "List Artist" action. 2. The system displays the artist list and the “Action” column. 3. The admin clicks the “x” button to remove the artist. 4. The system prompts the admin to confirm the removal. 5. The admin decides to cancel the removal action. 6. The system cancels the action and does not remove the artist.

	7. The system displays a cancellation message or simply refreshes the list without any changes.
Special Requirements	- Only admin can remove an artist. - The artist's data, including all related records, should be completely removed from the system upon deletion. This includes ensuring that there are no orphaned records in other related data tables.
Preconditions	- The admin is authenticated and authorized to perform the action of removing an artist. - The required artist data (name, description, background color, and image) is available.
Postconditions	- The artist is successfully removed from the database. No data related to the artist should remain in the system - A success message is sent to the admin confirming the artist's removal, or an error message is sent if the operation fails.
Extension Points	- None

2.2.4 Update Artist

Use-case ID	**UC015**
Use-case name	**Update Artist**
Brief description	This use case allows an admin to update the information of an existing artist in the system. Information that can be updated includes the artist’s name, description, background color, and profile picture
Actor	Admin
Basic flow	1. The use case begins when an admin selects the “Update Artist” option for a specific artist. 2. The system retrieves the current details of the artist and displays them in an editable form. 3. The admin updates one or more fields (e.g., name, description, background color, profile picture). 4. The admin submits the updated information. 5. The system validates the input data to ensure it meets format and content requirements. 6. Upon successful validation, the system saves the updated artist information to the database. 7. The use case ends when the system shows the updated artist information to the admin.
Alternative flow	**Alternative flow 1: Admin Cancels Update** 1. The admin selects the “Update Artist” option for a specific artist. 2. The system retrieves and displays the current details of the

	artist in an editable form. 3. The admin makes changes to one or more fields (e.g., name, description, background color, profile picture). 4. The admin decides not to proceed with the update and cancels the action. 5. The system prompts the admin to confirm the cancellation. 6. The admin confirms the cancellation. 7. The system discards any changes made and returns the admin to the previous screen without saving the updates. **Alternative flow 2: Validation Error** 1. The admin selects the “Update Artist” option for a specific artist. 2. The system retrieves and displays the current details of the artist in an editable form. 3. The admin updates one or more fields (e.g., name, description, background color, profile picture). 4. The admin submits the updated information. 5. The system validates the entered data to check for format and content requirements. 6. If any data does not meet the validation requirements (e.g., missing name or invalid image format), the system displays an error message specifying the issue. 7. The admin corrects the data based on the error message. 8. The admin resubmits the update. **Alternative flow 3: Database Error** 1. The admin selects the “Update Artist” option for a specific artist. 2. The system retrieves and displays the current details of the artist in an editable form. 3. The admin updates one or more fields (e.g., name, description, background color, profile picture). 4. The admin submits the updated information. 5. The system validates the entered data. 6. Upon successful validation, the system attempts to save the updated information to the database. 7. If there is an error while saving the updates to the database, the system displays an error message to the admin, indicating the failure.
Special Requirements	- Only authorized admin users should be allowed to update artist information. - Ensure the name, description, and other fields adhere to expected formats. Profile picture should meet specific size and format requirements.
Preconditions	- The admin must be logged in with sufficient permissions to update artist details. - The artist to be updated must already exist in the database.
Postconditions	- If successful, the artist’s information is updated in the system,

	and any page displaying this artist will reflect the new information. - If unsuccessful, the system maintains the original artist details, and the admin is informed of the failure.
Extension Points	- None.

2.3 Curator

2.3.1 Add Song

Use-case ID	**UC016**
Use-case name	**Add Song**
Brief description	This use case allows a curator/admin to add a new song to the system by entering all the required song information. The system validates the input and saves the new song to the database upon successful validation.
Actor	Curator (Admin)
Basic flow	1. Curator/Admin navigates to the "Add New Song" section. 2. System displays the "Add New Song" form. 3. Curator enters song information (title, artist, description, playlist, image). 4. Curator submits the new song. 5. System validates the song information. 6. System checks for duplicate songs. 7. System saves the new song to the database. 8. System confirms the addition of the new song.
Alternative flow	**Alternative flow 1: Incomplete Information Entry** 1. The curator/admin initiates the process of adding or updating information. 2. The curator/admin attempts to submit the information. 3. The system checks for required fields. 4. If any required fields are missing, the system prompts the curator/admin to complete all required fields. 5. The curator/admin provides the missing information. 6. The system validates the information and proceeds with the process. **Alternative flow 2: Duplicate Song Entry** 1. The curator/admin attempts to add a new song to the system. 2. The system checks for duplicate entries in the database. 3. If a duplicate song is found, the system warns the curator/admin about the duplicate. 4. The curator/admin is given the option to cancel the action. 5. The curator/admin chooses to cancel the addition of the duplicate song. 6. The system does not add the duplicate song and returns to the previous screen or action.

Special Requirements	- Only authenticated curators can add new songs. - Proper session management to prevent unauthorized access.
Preconditions	- Account must have an admin or curator role and be logged in.
Postconditions	- New songs are stored in the database. - Notification for curator.
Extension Points	- None.

2.3.2 Remove Song

Use-case ID	**UC017**
Use-case name	**Remove Song**
Brief description	This use case allows a curator/admin to remove an existing song from the system. The system attempts to delete the song from the database and returns a success or failure response based on the outcome.
Actor	Curator (Admin)
Basic flow	1. Curator/Admin navigates to the "List Song" section. 2. System displays the list of existing songs. 3. Curator/Admin clicks on the “x” button for the desired song. 4. The system prompts the user to confirm. 5. Curator/Admin confirms action. 6. System removes the song from the database. 7. The system responds with a success message indicating that the song has been removed.
Alternative flow	**Alternative flow 1: Admin Cancels Song Removal** 1. The admin initiates the "List Song" action. 2. The system displays the song list and the “Action” column. 3. The admin clicks the “x” button to remove a song. 4. The system prompts the admin to confirm the song removal. 5. The admin decides to cancel the removal action. 6. The system cancels the action and does not remove the song. 7. The system displays a cancellation message or simply refreshes the list without any changes.
Special Requirements	- Only authenticated curators can delete songs - Proper session management to prevent unauthorized access.
Preconditions	- Account must have an admin or curator role and be logged in.
Postconditions	- The song is successfully removed from the database. No data related to the song should remain in the system. - A success message is sent to the curator confirming the song’s removal, or an error message is sent if the operation fails.

Extension Points	- None.

2.3.3 Update Song

Use-case ID	**UC018**
Use-case name	**Update Song**
Brief description	This use case allows a curator/admin to update the information of an existing song in the system. The curator/admin selects a song to edit, modifies the necessary details, and the system saves the changes after validation.
Actor	Curator (Admin)
Basic flow	1. Curator/Admin navigates to the "List Song" section. 2. System displays the list of existing songs. 3. Curator/Admin clicks on the “update” button for the desired song. 4. System displays the current information of the selected song in an editable form. 5. Curator/Admin modifies the song information as needed. 6. The system prompts the user to confirm. 7. Curator/Admin submits the updated song information. 8. System validates the updated information. 9. System saves the changes to the database. 10. The system responds with a success message indicating that the song has been updated.
Alternative flow	**Alternative flow 1: Incomplete Information Entry** 1. The curator/admin initiates the process of adding or updating information. 2. The curator/admin attempts to submit the information. 3. The system checks for required fields and validates the data. 4. If any required fields are left empty or contain invalid data, the system highlights the problematic fields. 5. The system prompts the curator/admin to correct the errors. 6. The curator/admin corrects the information. 7. The system validates the corrected information and proceeds with the process. **Alternative flow 2: Admin Cancels Song Update** 1. The admin selects the “Update Song” option for a specific song. 2. The system retrieves and displays the current details of the song in an editable form. 3. The admin makes changes to one or more fields (e.g., song title, artist, genre, album cover). 4. The admin decides not to proceed with the update and cancels the action. 5. The system prompts the admin to confirm the cancellation. 6. The admin confirms the cancellation. 7. The system discards any changes made and returns the admin to the previous screen without saving the updates.

Special Requirements	- Only authenticated curators can update songs. - Proper session management to prevent unauthorized access.
Preconditions	- Account must have an admin or curator role and be logged in.
Postconditions	- The song's information is updated in the database with the new details - A success message is sent to the curator/admin confirming the song's update, or an error message is sent if the operation fails.
Extension Points	- None.

2.3.4 Add Playlist

Use-case ID	**UC019**
Use-case name	**Add Playlist**
Brief description	This use case allows an admin or curator to add a new playlist to the system. The user provides the playlist's information, including name, description, background color, and cover image.
Actor	Curator (Admin)
Basic flow	1. The admin/curator initiates the "Add Playlist" action. 2. The system prompts the user to enter Playlist details. 3. The admin/curator inputs the playlist's name, description, and background color and selects a cover image file to upload. 4. The admin/curator clicks on the "Add" button to save the Playlist. 5. The system saves the new playlist record in the database. 6. The system confirms that the playlist has been added.
Alternative flow	**Alternative flow 1: Database Save Failure** 1. The user initiates the process to save or update a playlist. 2. The system attempts to save the playlist's data to the database. 3. If the system encounters an issue saving the playlist data (e.g., database connection error or other issues), the system captures the error. 4. The system returns an error message to the user, indicating that the playlist was not added or updated successfully. 5. The user may attempt to retry the operation or contact support if needed.
Special Requirements	- The request should be validated to ensure the user is authorized (admin or curator) to add a playlist. - All image uploads must be optimized to reduce file size without compromising quality to ensure performance. - The system should validate the playlist data for completeness and format.
Preconditions	- The admin/curator is authenticated and authorized to add a playlist. - The required playlist data (name, description, background color, and

	cover image) is available.
Postconditions	- A new playlist is added to the List Playlist - The playlist is properly stored in the database with all its attributes. - In case of a failure, appropriate error logs and messages are generated for troubleshooting.
Extension Points	- None.

2.3.5 List Playlist

Use-case ID	**UC020**
Use-case name	**List Playlist**
Brief description	This use case allows an admin or curator to retrieve a list of all playlists from the database and view their details. It helps admins and curators quickly access and manage playlist information.
Actor	Curator (Admin)
Basic flow	1. The use case begins when an admin/curator selects the "List Playlists" option on the application. 2. The system sends a request to the server to retrieve all playlists. 3. The server accesses the database and retrieves all playlist records. 4. If successful, the system displays a list of all playlists with their details, such as name, description, cover image, and background color. 5. The use case ends when the list of playlists is successfully presented to the admin/curator.
Alternative flow	**Alternative flow 1: Database Error** 1. The system attempts to retrieve the list of playlists from the database. 2. If there is an issue with the database connection (e.g., connection failure), the system notifies the admin that the playlist list could not be retrieved. 3. The admin is given the option to retry the request. 4. If the issue persists, the admin is prompted to contact support for further assistance. **Alternative flow 2: Empty List** 1. The system attempts to retrieve the list of playlists from the database. 2. If no playlists are found in the database, the system displays a message indicating that no playlists are currently available. 3. The admin can choose to add a new playlist via the “Add Playlist” option if needed.
Special Requirements	- The list of playlists displayed should be up-to-date, reflecting any recent additions, deletions, or modifications. - The list of playlists should load within a few seconds to maintain a smooth user experience, even with a large dataset.

	- The system should display relevant playlist metrics (number of songs, total duration, etc.)
Preconditions	- The admin/curator must be logged in to access the playlist list - The database and server must be online and responsive.
Postconditions	- Upon successful completion, the admin/curator views a list of all playlists. - If unsuccessful, the system provides feedback to the admin/curator regarding the nature of the error.
Extension Points	- None.

2.3.6 Remove Playlist

Use-case ID	**UC021**
Use-case name	**Remove Playlist**
Brief description	This use case allows admins or curators to remove a playlist from the system.
Actor	Curator (Admin)
Basic flow	1. The admin/curator initiates the "List Playlist" action. 2. The system displays the playlist list and "Action" column. 3. The admin/curator clicks on the "x" button to remove the playlist. 4. The system prompts for confirmation. 5. The admin/curator confirms the action. 6. The system responds with a success message indicating that the playlist has been removed.
Alternative flow	**Alternative flow 1: Admin Cancels Playlist Removal** 1. The admin initiates the "List Playlist" action. 2. The system displays the playlist list and the “Action” column. 3. The admin clicks the “x” button to remove a playlist. 4. The system prompts the admin to confirm the playlist removal. 5. The admin decides to cancel the removal action. 6. The system cancels the action and does not remove the playlist. 7. The system displays a cancellation message or simply refreshes the list without any changes.
Special Requirements	- Only authorized users (admins and curators) can remove playlists. - The playlist's data should be completely removed from the database. - The system must maintain logs of playlist removals for auditing purposes. - The system should ensure data integrity by properly removing all references to the playlist
Preconditions	- The admin/curator is authenticated and authorized to perform playlist removal actions.

	- The playlist exists in the system. - The playlist data is accessible in the database.
Postconditions	- The playlist is successfully removed from the database. - All associated data (songs, playlist metadata, featured locations) are properly cleaned up. - A success message is displayed to the admin/curator confirming the playlist's removal. - The system's playlist count is updated accordingly. - The action is logged in the system's audit trail
Extension Points	- None.

2.3.7 Update Playlist

Use-case ID	**UC022**
Use-case name	**Update Playlist**
Brief description	This use case allows an admin or curator to update the information of an existing playlist in the system. Information that can be updated includes the playlist's name, description, visibility status, and cover image
Actor	Curator (Admin)
Basic flow	1. The use case begins when an admin/curator selects the "Update Playlist" option for a specific playlist. 2. The system retrieves the current details of the playlist and displays them in an editable form. 3. The admin/curator updates one or more fields (e.g., name, description, visibility status, cover image). 4. The admin/curator submits the updated information. 5. The system validates the input data to ensure it meets format and content requirements. 6. Upon successful validation, the system saves the updated playlist information to the database. 7. The use case ends when the system shows the updated playlist information to the admin/curator.
Alternative flow	**Alternative flow 1: Admin Cancels Update** 1. The admin selects the “Update Playlist” option for a specific playlist. 2. The system retrieves and displays the current details of the playlist in an editable form. 3. The admin makes changes to one or more fields (e.g., name, description, cover image). 4. The admin decides not to proceed with the update and cancels the action. 5. The system prompts the admin to confirm the cancellation. 6. The admin confirms the cancellation. 7. The system discards any changes made and returns the admin to the previous screen without saving the updates.

	Alternative flow 2: Validation Error 1. The admin selects the “Update Playlist” option for a specific playlist. 2. The system retrieves and displays the current details of the playlist in an editable form. 3. The admin updates one or more fields (e.g., name, description, cover image). 4. The admin submits the updated information. 5. The system validates the entered data to check for format and content requirements. 6. If any data does not meet the validation requirements (e.g., missing name or invalid image format), the system displays an error message specifying the issue. 7. The admin corrects the data based on the error message. 8. The admin resubmits the update. **Alternative flow 3: Database Error** 1. The admin selects the “Update Playlist” option for a specific playlist. 2. The system retrieves and displays the current details of the playlist in an editable form. 3. The admin updates one or more fields (e.g., name, description, cover image). 4. The admin submits the updated information. 5. The system validates the entered data. 6. Upon successful validation, the system attempts to save the updated information to the database. 7. If there is an error while saving the updates to the database, the system displays an error message to the admin, indicating the failure.
Special Requirements	- Only authorized admin and curator accounts should be able to update playlist information. - Cover image should meet specific size and format requirements. - Ensure the name, description, and other fields stick to expected formats
Preconditions	- The admin/curator must be logged in with sufficient permissions to update playlist details. - The playlist to be updated must already exist in the database.
Postconditions	- If successful, the playlist's information is updated in the system, and any page displaying this playlist will reflect the new information. - If unsuccessful, the system maintains the original playlist details, and the admin/curator is informed of the failure.
Extension Points	- None.

2.4 System

2.4.1 Recommend Song

Use-case ID	**UC023**
Use-case name	**Recommend Song**

Brief description	This use case allows the Beatify system to recommend new songs based on the description of the premium user's previously listened songs. The recommendation engine identifies songs with similar descriptions, such as genres or styles (e.g., "Hip-Hop"), to suggest new songs that align with the user's preferences
Actor	System
Basic flow	1. System Analyzes Last Listened Songs: The system retrieves the description of the user's most recently listened songs, which includes general categories or genre labels. 2. System Identifies Matching Songs: The system searches the song database for other songs with the same description as the last listened songs. 3. Display Recommendations to User: a. The top matches are presented to the user in the "Recommended for You" section. b. Each recommended song includes relevant details such as the song title, artist, and description, giving context to the recommendation. 4. User Interacts with Recommendations: a. The user can play, save each recommended song. b. Listening to a recommended song updates the user's last listened song and refines future
Alternative flow	**Alternative flow 1: Limited Listening History** 1. The user logs into the system. 2. The system checks the user's listening history. 3. If the user has limited listening history (e.g., new account or few plays), the system defaults to general recommendations. 4. The system generates recommendations based on popular songs, trending artists, or songs from commonly enjoyed genres. 5. The user can browse and select from the recommended content. **Alternative flow 2: No New Recommendations Available** 1. The user logs into the system. 2. The system checks for new song recommendations based on the user's listening history. 3. If no new songs match the user's listening history, the system detects the lack of relevant new recommendations. 4. The system displays a message indicating no new recommendations are currently available. 5. The system suggests exploring other sections like trending playlists or new releases for content.
Special Requirements	- The recommendation system requires regular updates to accurately match new songs to user preferences. - The system should be optimized to handle large amounts of user data and recommend songs with minimal latency.
Preconditions	- The user must have an active premium account with at least some listening history.

	- The song database should contain song descriptions and metadata relevant for matching user preferences, such as genre, mood, and artist.
Postconditions	- Recommended songs are updated in the user's interface. - The user's interactions with the recommendations (e.g., playing, saving, dismissing) influence future recommendations.
Extension Points	- Genre-Based Playlists: The system could create a playlist based on the user's preferred genre.