Group02

Beatify
Use-Case Specification: Music Playback

Version 1.0

Revision History

Date	Version	Description	Author
13/11/2024	1.0	Initial version. Created use-case specifications for Music Playback including necessary fields, basic and alternative flows, and specific preconditions and postconditions.	- Lê Gia Huy: Flows, Special Requirements, Preconditions. - Nguyễn Duy Bảo: Postconditions, Extension Points.

Table of Contents

Use-Case Specification: Music Playback

1. Use-Case Name: Music Playback
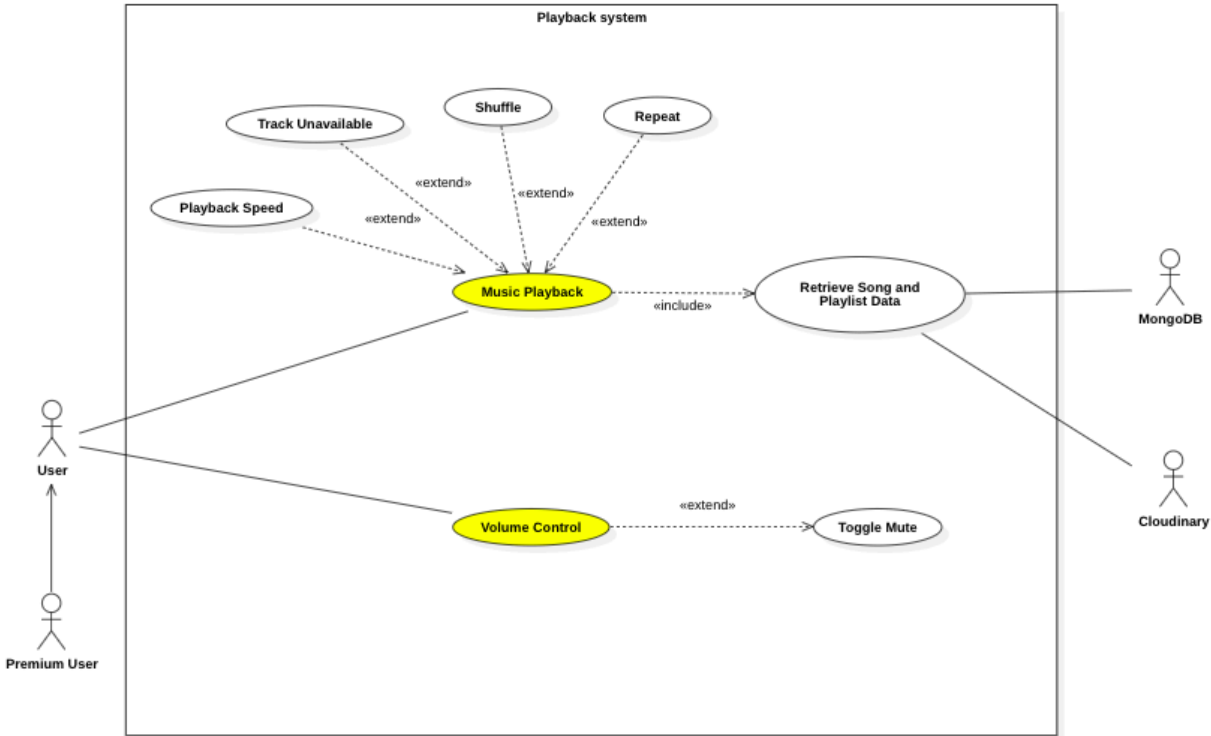
1.1 Brief Description

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.

2. Flow of Events

2.1 Basic Flow

2.1.1 *User initiates playback: The user selects a song or playlist and clicks the Play button.*

- System Response: The *playSong* function is triggered, creating a new audio instance if the selected song is different from the current one. The song begins playing, and *isPlaying* is set to *true*.

2.1.2 *User controls playback:*

- **Pause:** The user clicks the Pause button.
  - System Response: The *pauseSong* function pauses the audio, setting *isPlaying* to *false*.
- **Play:** The user clicks the Play button after pausing.

○ System Response: The song resumes from where it was paused.

2.1.3 User navigates tracks:

- **Next Track:** The user clicks the Next button.
  ○ System Response: The system skips to the next song in the playlist. If shuffle is enabled, it selects a random track.
- **Previous Track:** The user clicks the Previous button.
  ○ System Response: The system either restarts the current song or plays the previous one, depending on playback position.

2.1.4 User adjusts volume:

- **Volume Control:** The user changes the volume slider.
  ○ System Response: The *setVolume* function updates the volume, storing the new level and adjusting the audio element's volume.

2.1.5 Toggle Mute:

- **Mute/Unmute:** The user clicks the Mute button.
  ○ System Response: The *toggleMute* function toggles the audio volume between muted and the last set level.

2.1.6 Playback Modes:

- **Shuffle:** The user enables or disables shuffle mode.
  ○ System Response: The playback order changes to random if enabled.
- **Repeat:** The user enables or disables repeat mode.
  ○ *System Response*: The system repeats the song or playlist based on the selected mode.

2.1.7 Playback Speed (For Premium Users Only):

- **Change Playback Speed**: The user adjusts the playback speed slider.
  ○ **System Response**: The setPlaybackSpeed function updates the speed of the currently playing song (e.g., 1x, 1.5x, 2x, etc.) depending on the selected value.

2.2 Alternative Flows

2.2.1 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.

2.2.2 Network Disruption

1. If the connection is lost, playback pauses, and an error message is shown. The user can resume playback after reconnection.

3. 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.

4. 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.

5. 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.

6. Extension Points

- The *getSongsData* and *getPlaylistsData* functions provide additional track and playlist information, which can extend functionality for dynamic playlist loading and song recommendations.