Use Case Document – SOTD

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Description** |
| 1.0 | 01/10/2023 | Ryan Tavares | Initial Version |
|  |  |  |  |
|  |  |  |  |

***Please note: The term “System” is used to refer to the SOTD application.***

**UC-01: Authorize Spotify/Apple Music Account**

**Overview:**

This use case describes a user authorizing the application’s ability to access to necessary information about their Spotify or Apple Music account.

**Notes:**

The user can link either their Spotify account or their Apple Music account, but not both.

**Actors:**

User – The user who wants to give the System the authority to access necessary data from their streaming service account.

Service – The streaming service (Spotify or Apple Music) that hosts the account being authorized and provides the authorization process via their respective APIs.

**Pre-conditions:**

1. User has completed the briefing given upon opening the System for the first time.

2. User has no currently linked Service account.

**Scenario:**

|  |  |  |
| --- | --- | --- |
|  | Action | Reaction |
| 1. | User chooses which Service they would like to use with the System. | System informs User that they are giving System authority to access necessary data from User’s Service account.  System prompts User with the authorization process provided by User’s chosen Service. |
| 2. | Service confirms that User has completed the authorization process. | System requests and loads the necessary data from User’s chosen Service to begin recommending songs tailored towards User.  System navigates to Home view. |

**Scenario Notes:**

The authorization process is determined by the Service, and is not handled by the System. The System simply lets the Service know that User wants to authorize their account with the System.

**Post-conditions:**

1. The System has been given authorization by User and Service to use necessary data from User’s Service account for System functionality.

2. System navigates to Home view.

**Exceptions:**

User does not complete Sevice’s authorization process.

**Dependencies and Relations:**

None.

**UC-02: Receive an Automatic Song Recommendation**

**Overview:**

This use case describes a user automatically receiving a song recommendation.

**Notes:**

The System serves song recommendations based on a timer. One song per day.

**Actors:**

User – The user who receives a song recommendation.

Service – The streaming service (Spotify or Apple Music) that hosts and generates the song recommendations.

Timer – The timer that signals when a song recommendation should be generated and served.

**Pre-conditions:**

1. User has linked a Service account.

**Scenario:**

|  |  |  |
| --- | --- | --- |
|  | Action | Reaction |
| 1. | Timer runs out. | System uses Service API functionality in conjunction with User’s Service account data, System settings, and vote history to request a song recommendation from the Service. |
| 2. | Service generates a song recommendation for System. | System serves the song recommendation to User.  If User has push-notifications enabled, System sends a push-notification to User’s device to inform them that they have a new song recommendation. |

**Scenario Notes:**

The User does not have to perform any actions to receive a song recommendation. Timer has been included as an actor because the Timer is the only stimulus that triggers the generation of a song recommendation. The key reason that the User is using the System is to be served recommendations automatically.

**Post-conditions:**

1. Timer is reset.

2. Song is available on System’s “Home” view.

3. Song recommendation is added to History log.

**Exceptions:**

None.

**Dependencies and Relations:**

UC-01: Authorize Spotify/Apple Music Account

**UC-03: Play Song Recommendation**

**Overview:**

This use case describes a user playing a song recommendation.

**Notes:**

The user will have the choice to play the song in the System or in their streaming service’s app. Embedding the song in-System may only allow for a preview to be played instead of the whole song.

**Actors:**

User – The user who wants to play their recommended song.

Service – The streaming service (Spotify or Apple Music) that hosts the song.

**Pre-conditions:**

1. User has been served at least one song recommendation by their chosen Service.

2. User is on the “Home” view of the System.

**Scenario 1:**

|  |  |  |
| --- | --- | --- |
|  | Action | Reaction |
| 1. | User presses the play button. | System begins playing the song (or a preview of the song depending on Service API capabilities) directly.  The play button turns into a pause button. |
| 2. | User presses the pause button. | System pauses the song.  The pause button turns into a play button. |
| 3. | User uses the seek slider to seek to a new timestamp in the song. | System moves to the timestamp indicated by the User.  If the song was playing before seeking, the song continues playing from the new timestamp.  If the song was paused before seeking, the song is paused at the new timestamp. |
| 4. | User lets the song play until the song is finished. | System reminds User to vote on the song in order to improve future recommendations. |

**Scenario 2:**

|  |  |  |
| --- | --- | --- |
|  | Action | Reaction |
| 1. | User presses the “Play in [Service name]” button. | System checks if User’s device has the corresponding Service app installed on their device.  If the Service app is installed, System uses the Service API to open the Service app and begin playing the song. |

**Scenario Notes:**

None.

**Post-conditions:**

1. The song is marked as “played” by the System.

**Exceptions:**

For Scenario 2: User does not have the corresponding Service app installed on their device.

**Dependencies and Relations:**

UC-01: Authorize Spotify/Apple Music Account

UC-02: Receive an Automatic Song Recommendation

**UC-04: Vote on a Song Recommendation**

**Overview:**

This use case describes a user voting on a song recommendation to indicate whether they want more or less recommendations like it.

**Notes:**

An up-vote means the user wants future recommendations to be like the song they up-voted.

A down-vote means the user wants future recommendations to be less like the song they down-voted.

A no-vote means that the System will treat the user’s experience of the song neutrally.

**Actors:**

User – The user who wants to vote on the quality of a song recommendation.

**Pre-conditions:**

1. User has been served at least one song recommendation by their chosen Service.

**Scenario:**

|  |  |  |
| --- | --- | --- |
|  | Action | Reaction |
| 1. | User presses the up-vote button or the down-vote button next to a song. | The vote button that the User pressed becomes highlighted and remains highlighted.  If User pressed the up-vote button, System will make the prevalent properties of the song become more favored in the User’s recommendation profile.  If User pressed the down-vote button, System will make the prevalent properties of the song become less favored in the User’s recommendation profile. |
| 2. | User presses the opposite of the highlighted vote button next to a song they previously voted on. | The highlighted vote button becomes un-highlighted, and the opposite vote button becomes highlighted instead.  Changes made to the User’s recommendation profile are inverted. |
| 3. | User presses the highlighted vote button next to a song they previously voted on. | The highlighted vote button becomes un-highlighted.  Changes made to the User’s recommendation profile are reverted. |

**Scenario Notes:**

The scenario details a user that gives a fresh vote to a song recommendation, changes their vote to the opposite vote, and undoes their vote. It may be possible to write these cases more elegantly, or perhaps the scenario should be split into separate use cases.

Also, a recommendation profile is a dataset that the System will keep based on the User’s streaming service account data, System settings, and vote history.

**Post-conditions:**

1. The System updates the User’s recommendation profile based on the User’s vote.

2. The User’s vote is visually recorded by highlighting selected vote buttons.

**Exceptions:**

None.

**Dependencies and Relations:**

UC-01: Authorize Spotify/Apple Music Account

UC-02: Receive an Automatic Song Recommendation

**UC-05: Access Song Recommendation History**

**Overview:**

This use case describes a user accessing a log of songs they have been recommended in the past.

**Notes:**

The History view will be limited in the number of past recommendations it shows. Tentatively, let’s say the user can only view the last 30 recommendations.

**Actors:**

User – The user who wants to access a log of past recommendations.

**Pre-conditions:**

1. User has been served at least one song recommendation by their chosen Service.

2. User is on the System’s Home view.

**Scenario:**

|  |  |  |
| --- | --- | --- |
|  | Action | Reaction |
| 1. | User presses the “History” button. | System loads past song recommendation data and navigates to the History view. |
| 2. | User scrolls to the beginning of available song recommendation history. | System informs User that there are no more past song recommendations available to be viewed.  System informs User of the 30-recommendation limit on the history log. |

**Scenario Notes:**

When the System informs the User, it won’t be some type of pop-up. It will likely be text that indicates there is no more log to view, and text that says “30/30” if the log is full (assuming 30 recommendations remains the limit) or “??/30” if it’s not full.

If the User has only been served one song prior to accessing the History view, their history log will just be empty.

**Post-conditions:**

None.

**Exceptions:**

None.

**Dependencies and Relations:**

UC-01: Authorize Spotify/Apple Music Account

UC-02: Receive an Automatic Song Recommendation

**UC-06: Adjust Priority of a Song Property**

**Overview:**

This use case describes a user who wants to adjust the priority with which a song property factors into future song recommendations.

**Notes:**

Available song properties are determined by the streaming service API. For example, Spotify API can access information about a song’s “danceability, valence, energy, tempo, loudness, speechiness, instrumentalness, liveness, acousticness,” and more.

**Actors:**

User – The user who wants to adjust the prevalence of song properties in future song recommendations.

**Pre-conditions:**

1. User has linked a Service account.

2. User is on the System’s Home view.

**Scenario:**

|  |  |  |
| --- | --- | --- |
|  | Action | Reaction |
| 1. | User presses the “Settings” button. | System navigates to the Settings view. |
| 2. | User presses the “Recommendations” button. | System navigates to the Settings-Recommendations view. |
| 3. | User presses the “Adjust Priority of Song Properties” button. | System navigates to the Song-Property-Priority view. |
| 4. | User drags the “Priority” slider next to one of the available song properties towards their preference. | If User decreases the priority of the song property, System updates will be more likely to serve recommendations that are weak in that property.  If User increases the priority of the song property, System will be more likely to serve recommendations that are strong in that property. |

**Scenario Notes:**

It may be unnecessary or inappropriate to detail the User’s navigation to the setting page they want to access. I’m still learning how to write proper use cases.

**Post-conditions:**

1. The User’s recommendation profile is updated according to the adjustments made by the User.

**Exceptions:**

None.

**Dependencies and Relations:**

UC-01: Authorize Spotify/Apple Music Account

**UC-07: Toggle Genre Presence**

**Overview:**

This use case describes a user who wants to toggle the presence of a music genre in future song recommendations.

**Notes:**

Available music genres are determined by the streaming service API.

**Actors:**

User – The user who wants to give the app the authority to access necessary data from their streaming service account.

**Pre-conditions:**

1. User has linked a Service account.

2. User is on the System’s Home view.

**Scenario:**

|  |  |  |
| --- | --- | --- |
|  | Action | Reaction |
| 1. | User presses the “Settings” button. | System navigates to the Settings view. |
| 2. | User presses the “Recommendations” button. | System navigates to the Settings-Recommendations view. |
| 3. | User presses the “Toggle Genres” button. | System navigates to the Toggle-Genre view. |
| 4. | User finds the genre they’re looking for and toggles it. | If User toggled the genre to “off,” then System will remove that genre from the pool of genres that song recommendations are pulled from.  If User toggled the genre to “on,” then System will add that genre to the pool of genres that song recommendations are pulled from. |

**Scenario Notes:**

None.

**Post-conditions:**

1. The User’s recommendation profile is updated according to the adjustments made by the User.

**Exceptions:**

None.

**Dependencies and Relations:**

UC-01: Authorize Spotify/Apple Music Account