# Software Requirements Specification

for

# Spotify Annotation Project

Version 1.2.1 approved

Prepared by Kyle Yannelli, Koby Perez, Joshua Smith, and Noah Newton

State University of New York at Oswego

September 10th, 2021

# Table of Contents

## Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| Team | 8/30/21 | Created documentation for 1.1 through 2.3 | 1.0 |
| Team | 9/01/21 | Revised and added documentation for Sections 1 & 2 | 1.1 |
| Team | 9/08/21 | Revised and added documentation for Sections 3 & 4, added diagrams | 1.2 |
| Team | 9/10/21 | Revised diagrams in section 2 & 3 | 1.2.1 |
| Team | 9/13/21 | Revised diagrams in section 2 & 3 again | 1.2.2 |
| Team | 9/17/21 | Added Class diagram to Section 2 | 1.2.3 |
| Team | 9/22/21 | Revised class diagram | 1.2.4 |
| Team | 9/27/21 | Added Behavioral Patterns to Class Diagrams | 1.2.5 |
| Team | 10/3/21 | Revised Behavioral Patterns | 1.2.6 |
| Team | 10/6/21 | Added new Pattern Diagrams | 1.2.7 |
| Team | 11/17/21 | Revised Class Diagram, Revised Documentation in all sections | 1.3 |

# 1. Introduction

## 1.1 Purpose

This project will satisfy all stakeholders expectations by allowing for the user to annotate songs from their Spotify library.

## 1.2 Document Conventions

This document will use a number of conventions. A strikethrough will denote the section is not currently needed or applicable to the project. "TBD" will denote the section relevant to the project and will be completed at a later date. Any content not written in italics is definite for the current version. Any content written in italics has not been reviewed yet.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for stakeholders and developers of the project. Section 1 contains an introduction to the project. Section 2 contains a more detailed description of the project. Section 3, 4, and 5 contain technical information about the project suited for developers.

## 1.4 Product Scope

This system will be a web application that allows a Spotify user to annotate and play songs from their playlists. This system will not allow for the user to annotate songs not in their library. This system will not allow users to annotate songs not in a playlist. This system will not allow users to search, add, like, or remove songs from their playlists.

## 1.5 References

- Project Description
- Spotify API Documentation
- Spotify Player Reference Repository
- Spotify Visualization
- Spotify Reference

# 2. Overall Description

## 2.1 Product Perspective

This is a web interface using the Spotify API to authenticate, load playlists and play songs. This interface will allow the user to input text annotations at specific times during the song's duration.

## 2.2 Product Functions

- Input and display annotation of text at specific timestamps
- Basic functionality of music player (pause, play, skip, rewind, etc.)
- List user's playlists
- Display current playing song
- Display when song was last played

## 2.3 User Classes and Characteristics

This application is intended to allow workout instructors to arrange playlists for a workout by the addition of annotating songs at specified timestamps.

## 2.4 Operating Environment

This application can run on any device with a stable internet connection and a web browser, however, this app is intended to run on a desktop web browser, limited to Google Chrome, Firefox, Microsoft Edge, or Opera. Browsers not listed, such as Safari, Internet Explorer, and etc.. are not supported.

## 2.5 Design and Implementation Constraints

Due to rate limiting, the Spotify API may limit our ability to send requests which allows the application to receive data from Spotify's servers. Although the Spotify API allows a user to add songs to the queue, it does not have the ability to remove, edit, or view the queue. Finally, the Spotify API limits the amount of playlists to 50. This should not be an issue as that is a considerable amount of playlists, but it is something to be noted.

## 2.6 User Documentation

### **2.6.1** Getting Started

1. Click the login text and sign in with a Spotify account.
2. Select a playlist from the table.
3. Select a song to play.

The user can now use the player buttons (previous, play/pause, skip). The user can click anywhere on the waveform to skip to that part of the song. The user can also click the song title or artist under the album cover image to find the song's location in the table.

The user can use the side menu to access the annotation display via Present, customize settings in Settings, or edit annotations in the Annotation Editor

### **2.6.2** Annotation Editor

1. Select a playlist from the playlists table
2. Select a song from the songs table
3. Type in the desired text in the top text area
4. Type in the time using the formatted text areas containing a colon (minutes : seconds)
5. Click the Update button

To update an annotation, select an annotation from the annotation list, and then change the parameters and click the Update button. To delete, simply click delete with an annotation selected.

To close the editor, select the annotation editor button again from the side menu.

### **2.6.3** Music Player

The player has the standard functionality of any music player: pause, play, previous, next, and shuffle buttons. Select a song from the song table to access functionality of the player.

The selected song title can be clicked on the header to bring the user to the location of the song in the playlist.

A song can be added to the queue by clicking the check mark in the row of the desired song. The queue can be viewed by selecting clicking the three-stack icon next to the Next Song button

### **2.6.4** Present Annotations

To view annotations as the song is being played, the Present option in the side menu must be selected. This will display the annotations in the center, with the time they appear before and after being adjustable in the Settings Menu. When there are no annotations being presented, ellipses will appear until it is appropriate to display annotations.

### **2.6.5** Settings Menu

The settings menu is accessed via the side menu, where the user can edit the color of the waveform, color of the annotations on the waveform, and how long before and after annotations are displayed. Options can be selected by clicking the current value and selecting from the dropdown menu to the desired value.

#### **2.6.6** Logging out

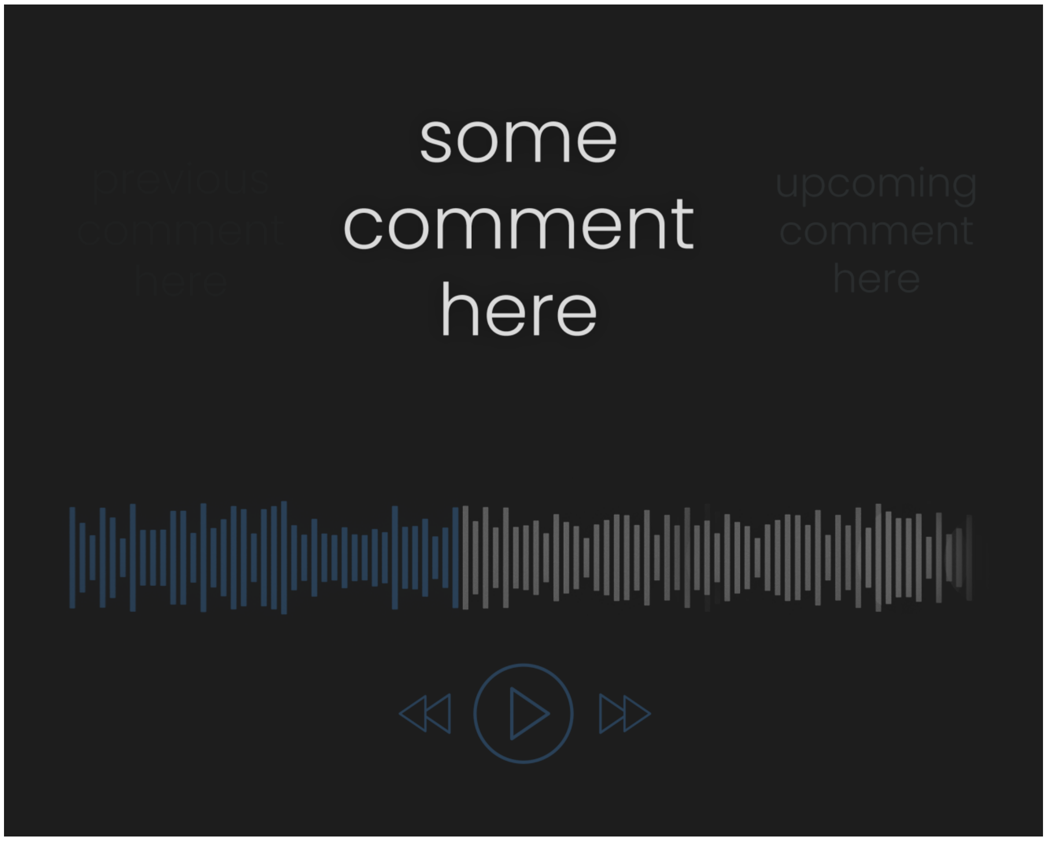
To log out, select the Logout option from the side menu. This will end the current session and will require re-authentication to access the Annotation Player again.

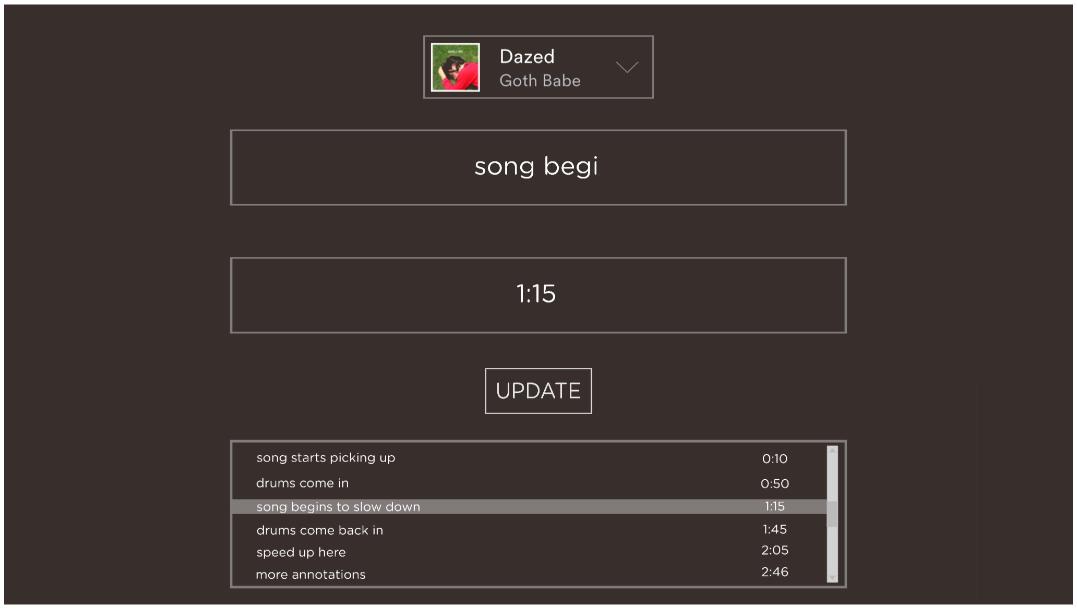
### 2.7 Assumptions and Dependencies

This application will use JavaScript to interact with the Spotify API, alongside MongoDB and AWS to handle the backend storage of user data. External Interface Requirements

## 3. External Interface Requirements

## 3.1 User Interfaces

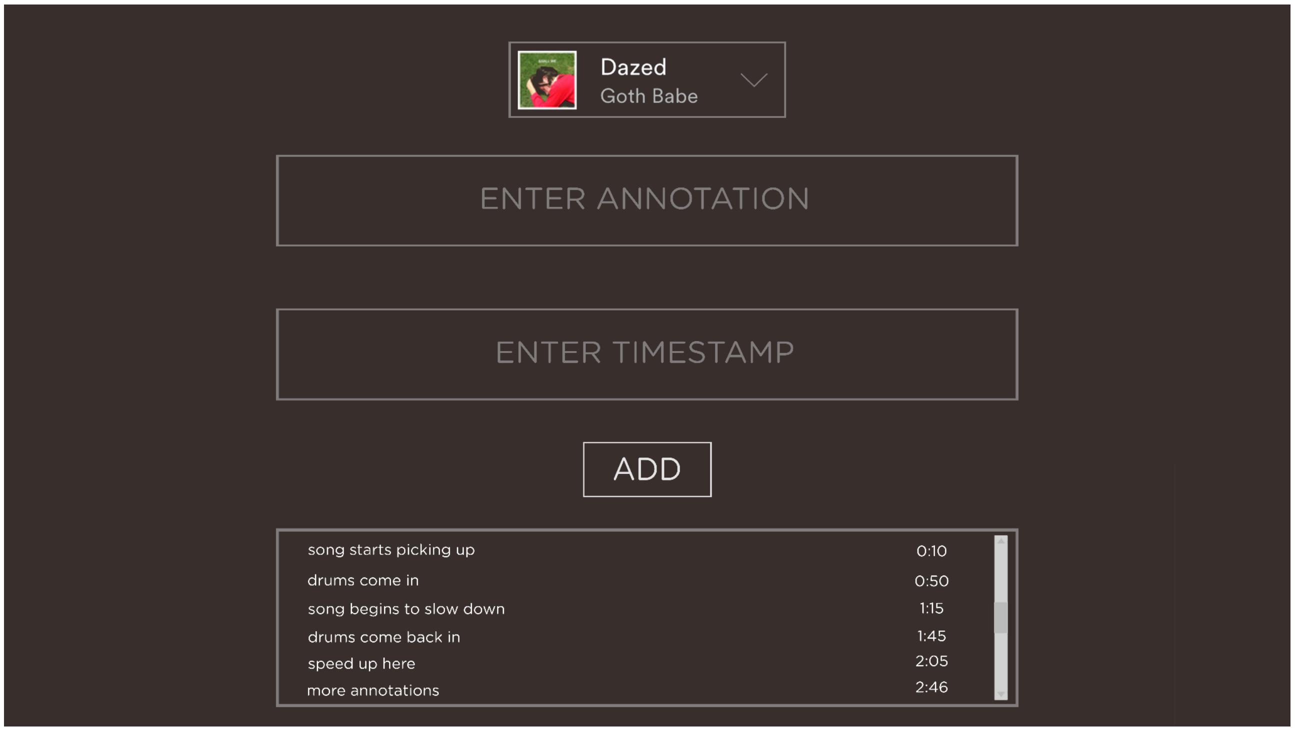




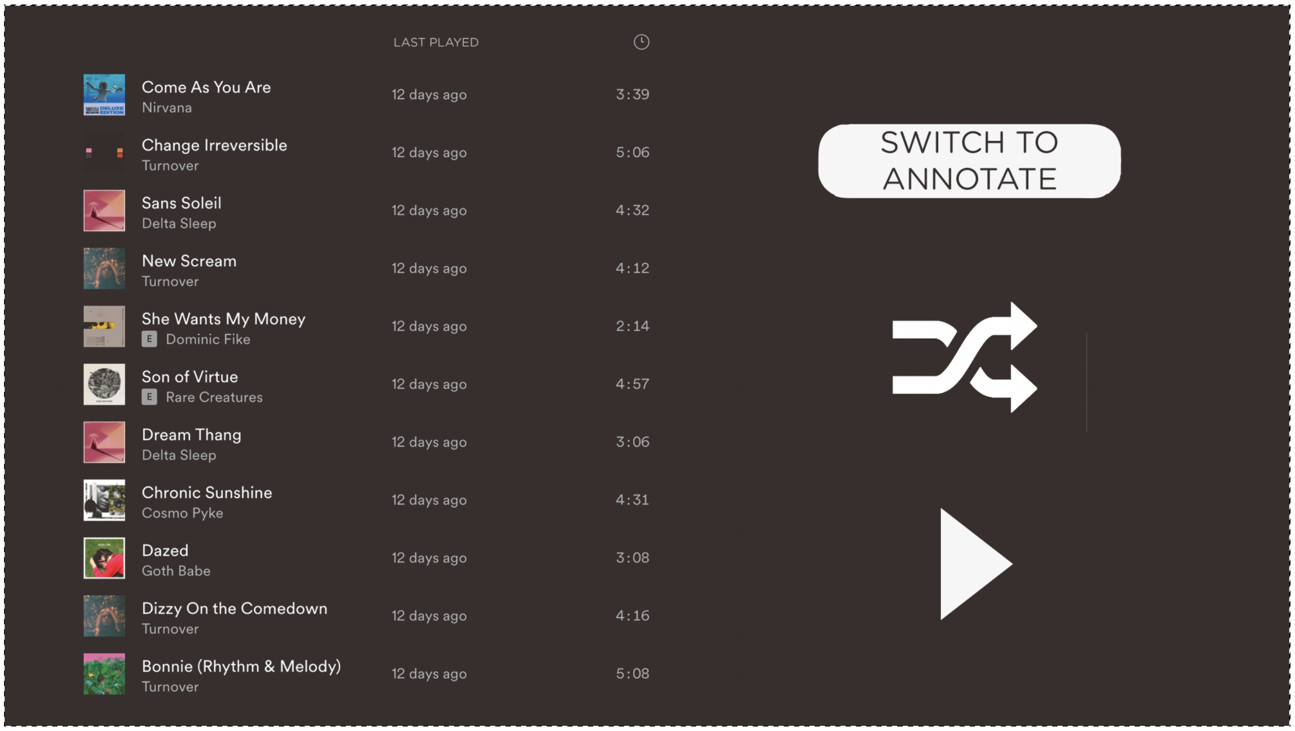
song begi

1:15

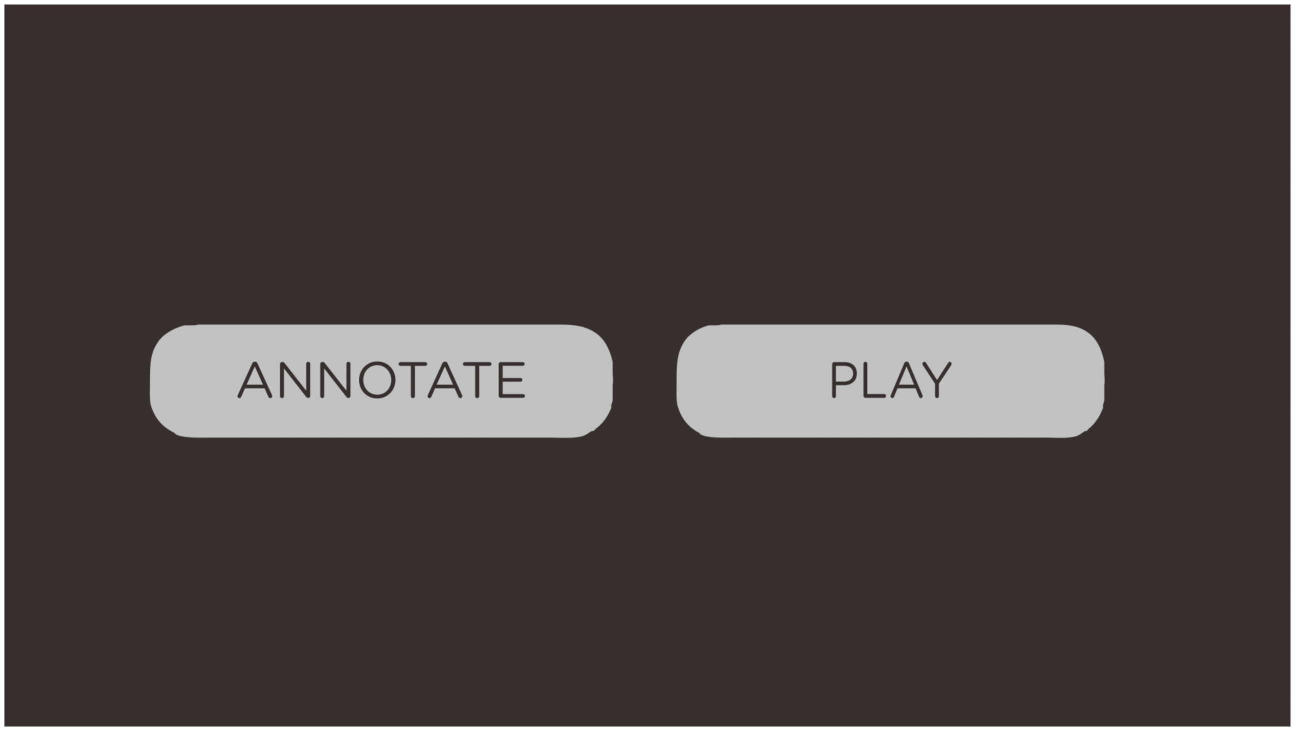
| | |
|---|---|
| song starts picking up | 0:10 |
| drums come in | 0:50 |
| song begins to slow down | 1:15 |
| drums come back in | 1:45 |
| speed up here | 2:05 |
| more annotations | 2:46 |

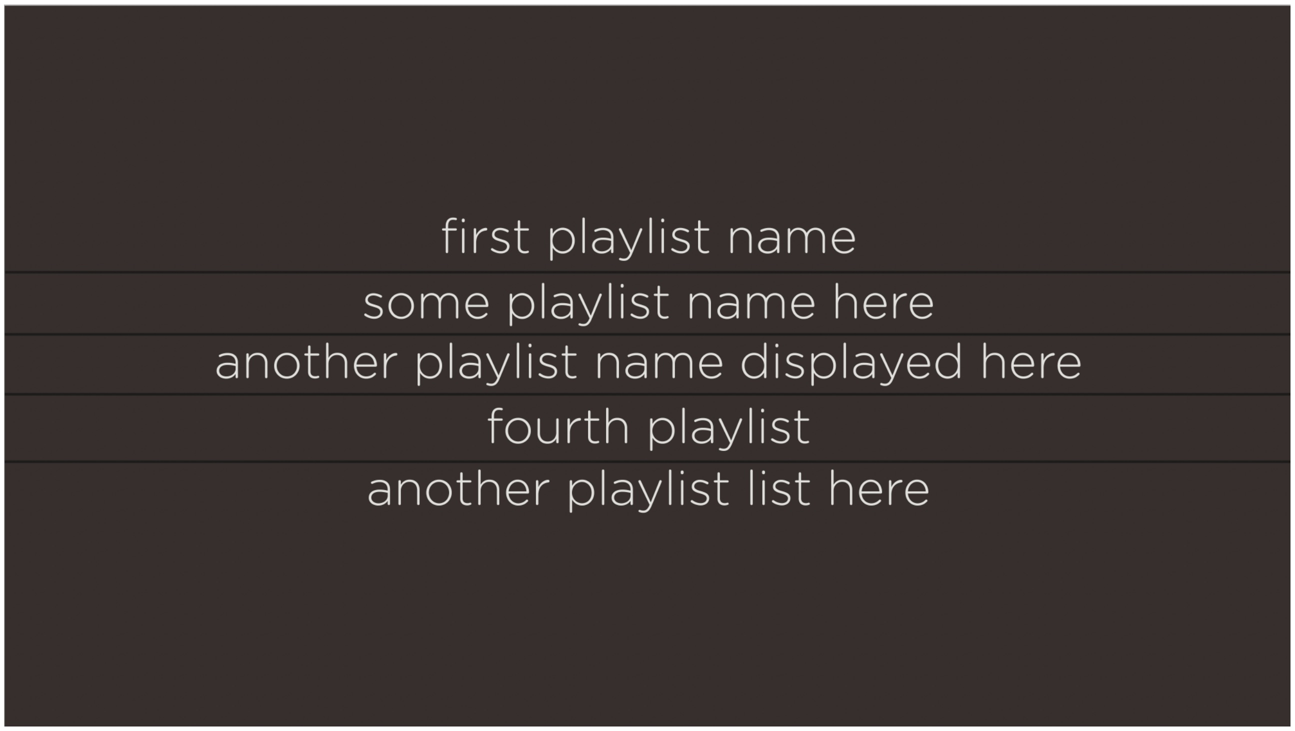


| | |
|---|---|
| song starts picking up | 0:10 |
| drums come in | 0:50 |
| song begins to slow down | 1:15 |
| drums come back in | 1:45 |
| speed up here | 2:05 |
| more annotations | 2:46 |



| | LAST PLAYED | 🕒 |
|---|---|---|
| Come As You Are<br>Nirvana | 12 days ago | 3:39 |
| Change Irreversible<br>Turnover | 12 days ago | 5:06 |
| Sans Soleil<br>Delta Sleep | 12 days ago | 4:32 |
| New Scream<br>Turnover | 12 days ago | 4:12 |
| She Wants My Money<br>E Dominic Fike | 12 days ago | 2:14 |
| Son of Virtue<br>E Rare Creatures | 12 days ago | 4:57 |
| Dream Thang<br>Delta Sleep | 12 days ago | 3:06 |
| Chronic Sunshine<br>Cosmo Pyke | 12 days ago | 4:31 |
| Dazed<br>Goth Babe | 12 days ago | 3:08 |
| Dizzy On the Comedown<br>Turnover | 12 days ago | 4:16 |
| Bonnie (Rhythm & Melody)<br>Turnover | 12 days ago | 5:08 |


ANNOTATE
PLAY


first playlist name
some playlist name here
another playlist name displayed here
fourth playlist
another playlist list here

The system will have multiple web pages as follows: The homepage displays a login button which allows users to connect their Spotify account to the application. After the user clicks the login button, they are brought to the Spotify login to authenticate their Spotify account. Once logged in, the user will automatically be redirected to the web player where they can play music and annotate as a song is being played.

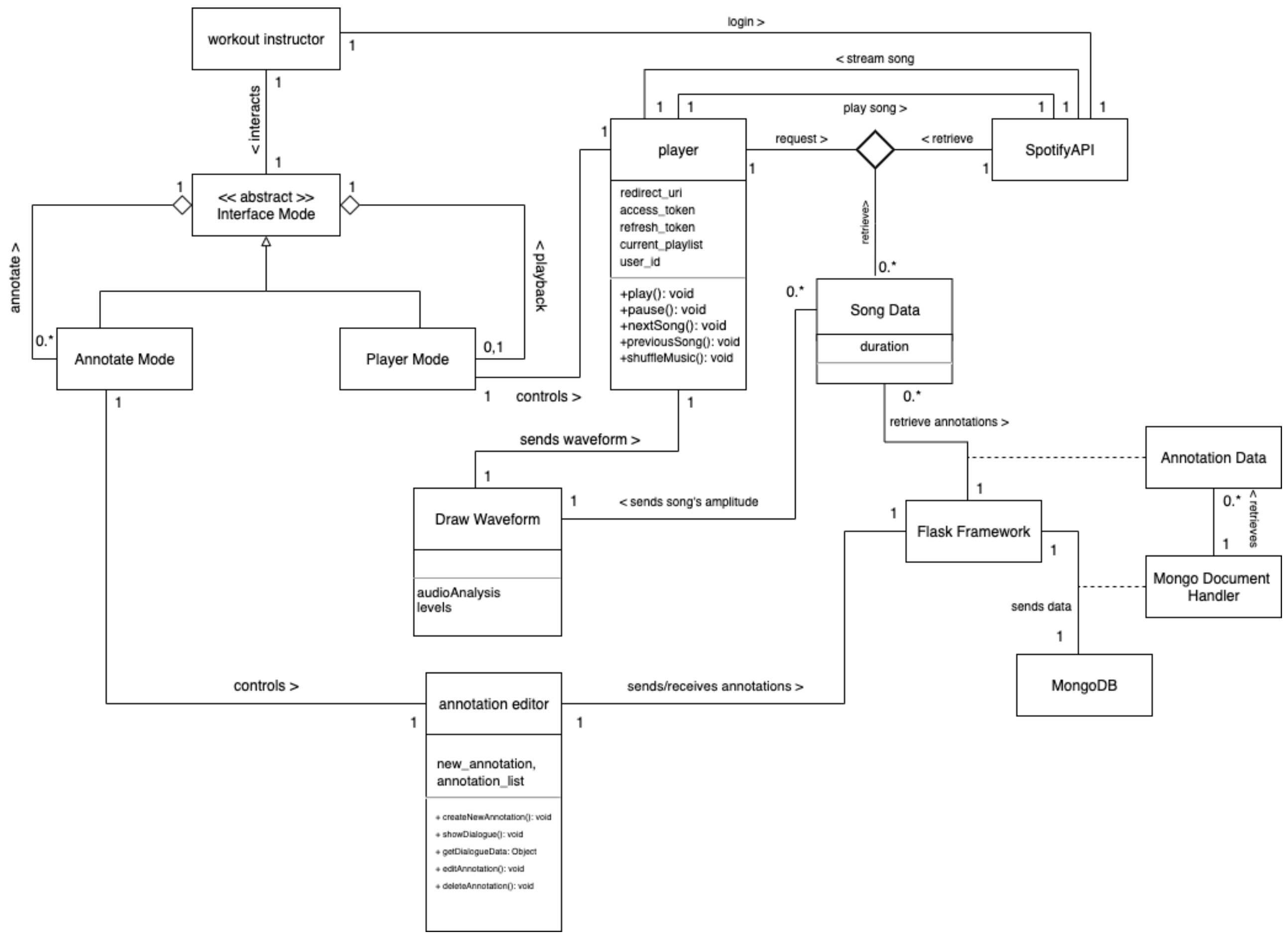
## 3.2 Hardware Interfaces

- Device with a web browser that's compatible with JavaScript
- Device with audio output functionality
- Device with a keyboard & mouse, or touchscreen.
- An internet connection fast enough to support music streaming

## 3.3 Software Interfaces

To deliver data to the web browser, the application will be running on a Linux server using OpenLiteSpeed (an alternative to Apache or Nginx). The application uses the Spotify API to retrieve and play the user's music library. To access this data from Spotify, the user authenticates their account and is given a unique access token for the session. The user's annotation data is stored using the database program, MongoDB.

## 3.4 Communications Interfaces



We have opted to not go into detail for the player due to the trivial nature of the API calls.

- Web browser
- HTTPS
- Premium Spotify Account

# 4 System Features

In all System Features sections, the importance of priority goes by ascending order.

## 4.1 Web Music Player

### 4.1.1 Description and Priority

The web player allows a user to play music.

- HIGH: **Play/Pause Button**
- HIGH: **Skip Button**
- MEDIUM: **Shuffle Button**
- MEDIUM: **Rewind Button**

### 4.1.2 Stimulus/Response Sequences

- **Play/Pause Button:** Plays the music if the music is paused, pauses the music if the music is playing.
- **Skip Button**: Skips the current playing song and begins playing the next song in the queue.
- **Shuffle Button**: Randomizes order of songs in the queue.
- **Rewind Button**: Restarts song at 0:00 timestamp for single click of the button, or goes to previous song in the queue for double click of the button.

### 4.1.3 Functional Requirements

REQ-1: The application requires a selected playlist to allow music to be played. A song does not need to be selected as the player will start from the beginning of the playlist. If a user tries to play music without a playlist selected, an error message will prompt the user to select a playlist.

REQ-2: More technical errors, such as a database error, will show the user a simple error message, such as “Sorry, something went wrong.” More simple errors such as a playlist not being selected when trying to hit the play button, will display a message such as, “Please select a playlist.”

REQ-3: User’s web browser must be compatible with JavaScript.

## 4.2 Live Track Annotation

### 4.2.1 Description and Priority

Live Playlist Annotation allows the user to annotate songs as they are playing.

- HIGH: **Update Button**
- HIGH: **Remove Button**

4.2.2 Stimulus/Response Sequences

- **Update Button:** Adds an annotation to the database and displays it in the table of annotations for the selected song. If the annotation is changed at the same time stamp, the button will update the given annotation.
- **Delete Button:** Deletes the annotation that is currently populated in the annotation editor from the database and removes it from the table.

4.2.3 Functional Requirements

REQ-1: The application requires valid inputs for the time stamp that the annotation will be set to. If a user tries to input anything but a valid timestamp in the duration of the song, the annotation will not be added to the database.
REQ-2: The application requires the user to not enter any annotations longer than 5 lines into the editor. If a user does so the presentation mode will cut it off. The application will expand to fit the text up to 4 full lines.

## 4.3 Playlist Selection Interface

4.3.1 Description and Priority
Allows user to select playlists, and song from those playlists to be played in the web player or annotated in the Playlist Annotation

- HIGH: **Playlist Table**

4.3.2 Stimulus/Response Sequences

- **Playlist Table:** Displays all of the user's playlists and will load the playlist once the user clicks on a Playlist in a row on the table.

4.3.3 Functional Requirements

REQ-1: The application requires the user to have playlists in their Spotify premium account. If there are no playlists the table will display blank and the user will not be able to access the player or annotation editor.
REQ-2: The application requires the user to have songs in the playlist to be loaded into the player.html. If the user does not have any playlists the player will not load any tracks or play anything. Also, the annotation editor will not display or allow the adding of annotations.

## 4.4 Present Annotations

### 4.4.1 Description and Priority

Displays annotations for each track above the waveform.

- HIGH: **Current Annotation**
- HIGH: **Next Annotation**
- MEDIUM: **Loading Dots**

### 4.4.2 Stimulus/Response Sequences

- **Current Annotation:** Displays the current annotation in the center of the screen for a specified amount of time.
- **Next Annotation:** Displays the next annotation to the right of the screen for a specified amount of time before.
- **Loading Dots:** Displays loading dots with color changing keyframes when there currently isn't an annotation in the specified time range.

### 4.4.3 Functional Requirements

REQ-1: The application requires that it is able to receive annotations for a specified user access token from MongoDB. This will ensure that the annotations are fetched for the current user and that other user's annotations are not received for the same song.

## 4.5 Track Waveform Timeline

### 4.5.1 Description and Priority

Displays a unique waveform for each track and displays the progress in a song.

- MEDIUM: **Waveform**
- MEDIUM: **Overlay of Progress**

### 4.5.2 Stimulus/Response Sequences

- **Waveform:** Displays the track amplitudes over the duration of the song.
- **Overlay of Progress:** Displays an overlay of a color on top of the waveform to show the current progress. If the waveform is clicked, the progress will seek to that point in the song.

### 4.5.3 Functional Requirements

REQ-1: The application requires that the Audio Analysis Spotify endpoint is obtained for each song.

REQ-2: The application requires the linux server to have node.js and vue.js to be installed to generate the waveform and draw the graphs.

## 4.6 Settings

### 4.6.1 Description and Priority

Allows the user to change the color of the waveform, the annotation on the waveform, and the time before and after an annotation.

- LOW: **Color of Waveform**
- LOW: **Color of Annotation**
- LOW: **Time Before the Annotation**
- LOW: **Time After the Annotation**

### 4.6.2 Stimulus/Response Sequences

- **Color of Waveform:** When a user selects a color from the dropdown the waveform overlay color changes.
- **Color of Annotations:** When a user selects a color from the dropdown the annotation color changes.
- **Time Before the Annotation:** When a user selects a time from the dropdown, the annotation will pop up in the current annotation at time selected seconds before the actual timestamp.
- **Time After the Annotation:** When a user selects a time from the dropdown, the annotation will disappear in the current annotation at time selected seconds after the actual timestamp.

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

A loading page was added after the user logins to ensure the dates of the last time a song was played are loaded.

## 5.2 Safety Requirements

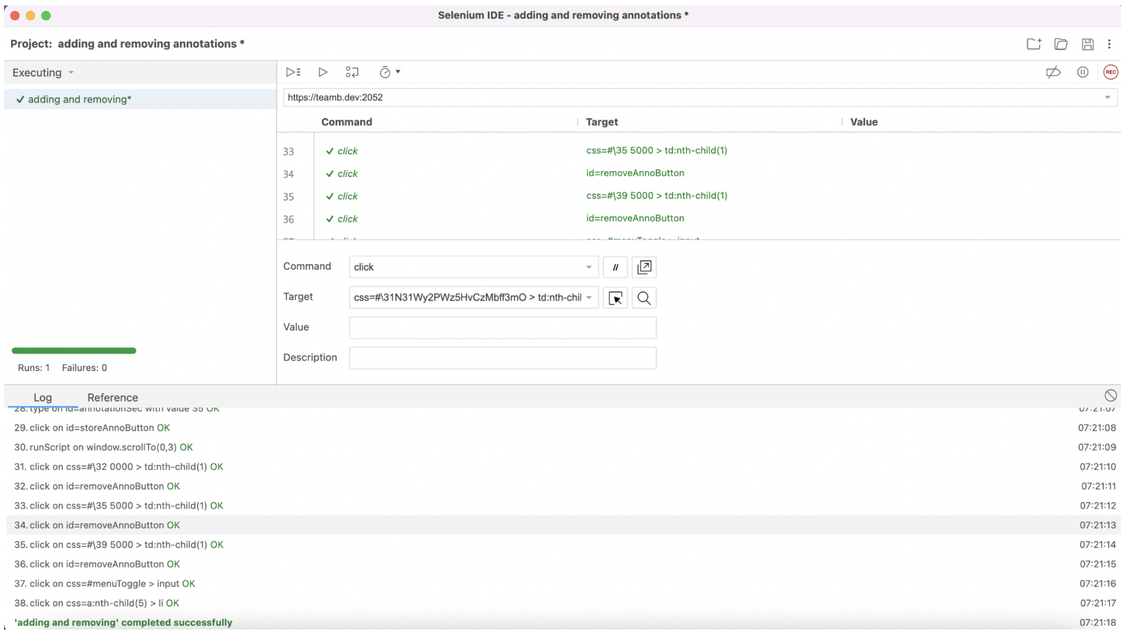
No actions must be taken to ensure the safety of the user. This system is safe and poses no safety threats to its users.

## 5.3 Security Requirements

In this system, authentication is taken care of by the Spotify API making the authentication process inherently secure. The storing of keys will be handled server side using Flask, keeping any sensitive information of this web application out of the client's browser. As well as handling the keys, Flask will protect the insertion of unwanted data from unauthenticated users by using the user's access token to permit insertion. This will also keep the user's annotation data from being retrieved by another user.

## 5.4 Software Quality Attributes

Selenium test results are as follows:



Selenium IDE - adding and removing annotations *

Project: adding and removing annotations *

Executing

✓ adding and removing*

https://teamb.dev:2052

| | Command | Target | Value |
|---|---|---|---|
| 33 | ✓ click | css=#\35 5000 > td:nth-child(1) | |
| 34 | ✓ click | id=removeAnnoButton | |
| 35 | ✓ click | css=#\39 5000 > td:nth-child(1) | |
| 36 | ✓ click | id=removeAnnoButton | |

Command: click

Target: css=#\31N31Wy2PWz5HvCzMbff3mO > td:nth-chil

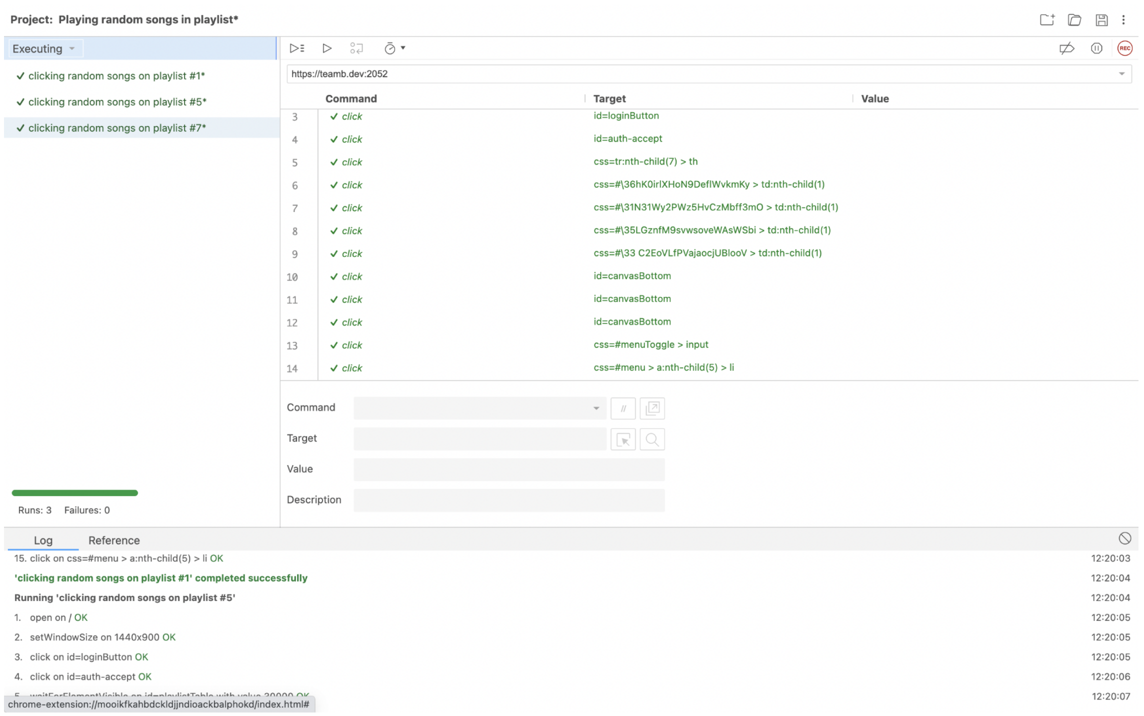
Value:

Description:

Runs: 1 Failures: 0

Log Reference

```
29. click on id=storeAnnoButton OK                                07:21:08
30. runScript on window.scrollTo(0,3) OK                          07:21:09
31. click on css=#\32 0000 > td:nth-child(1) OK                   07:21:10
32. click on id=removeAnnoButton OK                               07:21:11
33. click on css=#\35 5000 > td:nth-child(1) OK                   07:21:12
34. click on id=removeAnnoButton OK                               07:21:13
35. click on css=#\39 5000 > td:nth-child(1) OK                   07:21:14
36. click on id=removeAnnoButton OK                               07:21:15
37. click on css=#menuToggle > input OK                           07:21:16
38. click on css=a:nth-child(5) > li OK                           07:21:17
'adding and removing' completed successfully                      07:21:18
```



Project: Playing random songs in playlist*

Executing

✓ clicking random songs on playlist #1*

✓ clicking random songs on playlist #5*

✓ clicking random songs on playlist #7*

https://teamb.dev:2052

| | Command | Target | Value |
|---|---|---|---|
| 3 | ✓ click | id=loginButton | |
| 4 | ✓ click | id=auth-accept | |
| 5 | ✓ click | css=tr:nth-child(7) > th | |
| 6 | ✓ click | css=#\36hK0irlXHoN9DeflWvkmKy > td:nth-child(1) | |
| 7 | ✓ click | css=#\31N31Wy2PWz5HvCzMbff3mO > td:nth-child(1) | |
| 8 | ✓ click | css=#\35LGznfM9svwsoveWAsWSbi > td:nth-child(1) | |
| 9 | ✓ click | css=#\33 C2EoVLfPVajaocjUBlooV > td:nth-child(1) | |
| 10 | ✓ click | id=canvasBottom | |
| 11 | ✓ click | id=canvasBottom | |
| 12 | ✓ click | id=canvasBottom | |
| 13 | ✓ click | css=#menuToggle > input | |
| 14 | ✓ click | css=#menu > a:nth-child(5) > li | |

Command

Target

Value

Description

Runs: 3 Failures: 0

Log Reference

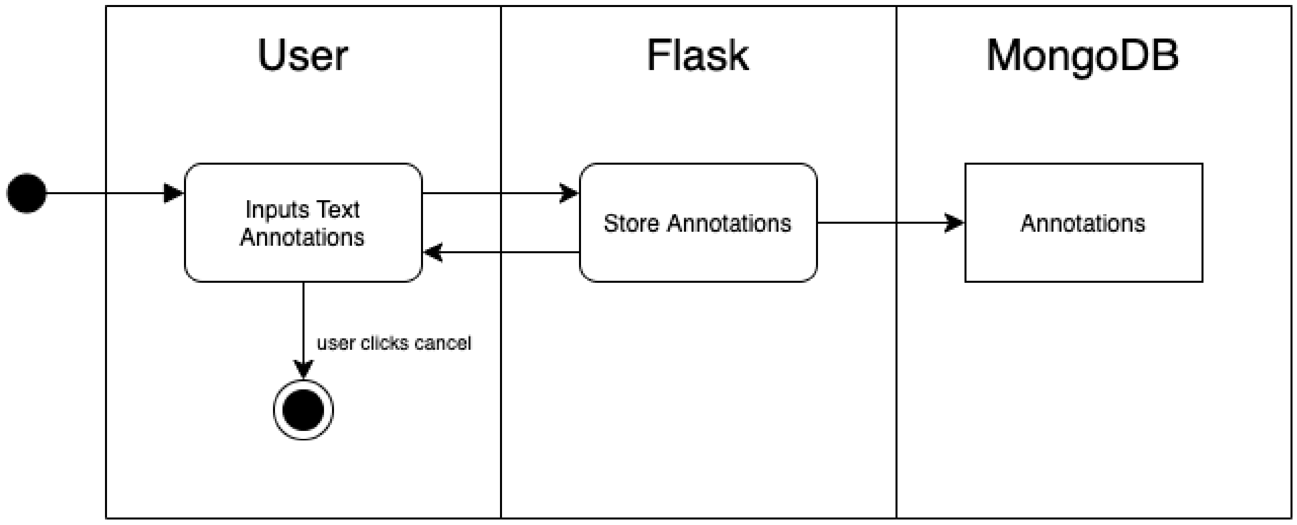
```
15. click on css=#menu > a:nth-child(5) > li OK                   12:20:03
'clicking random songs on playlist #1' completed successfully     12:20:04
Running 'clicking random songs on playlist #5'                    12:20:04
1.  open on / OK                                                  12:20:05
2.  setWindowSize on 1440x900 OK                                  12:20:05
3.  click on id=loginButton OK                                    12:20:05
4.  click on id=auth-accept OK                                    12:20:06
                                                                  12:20:07
```

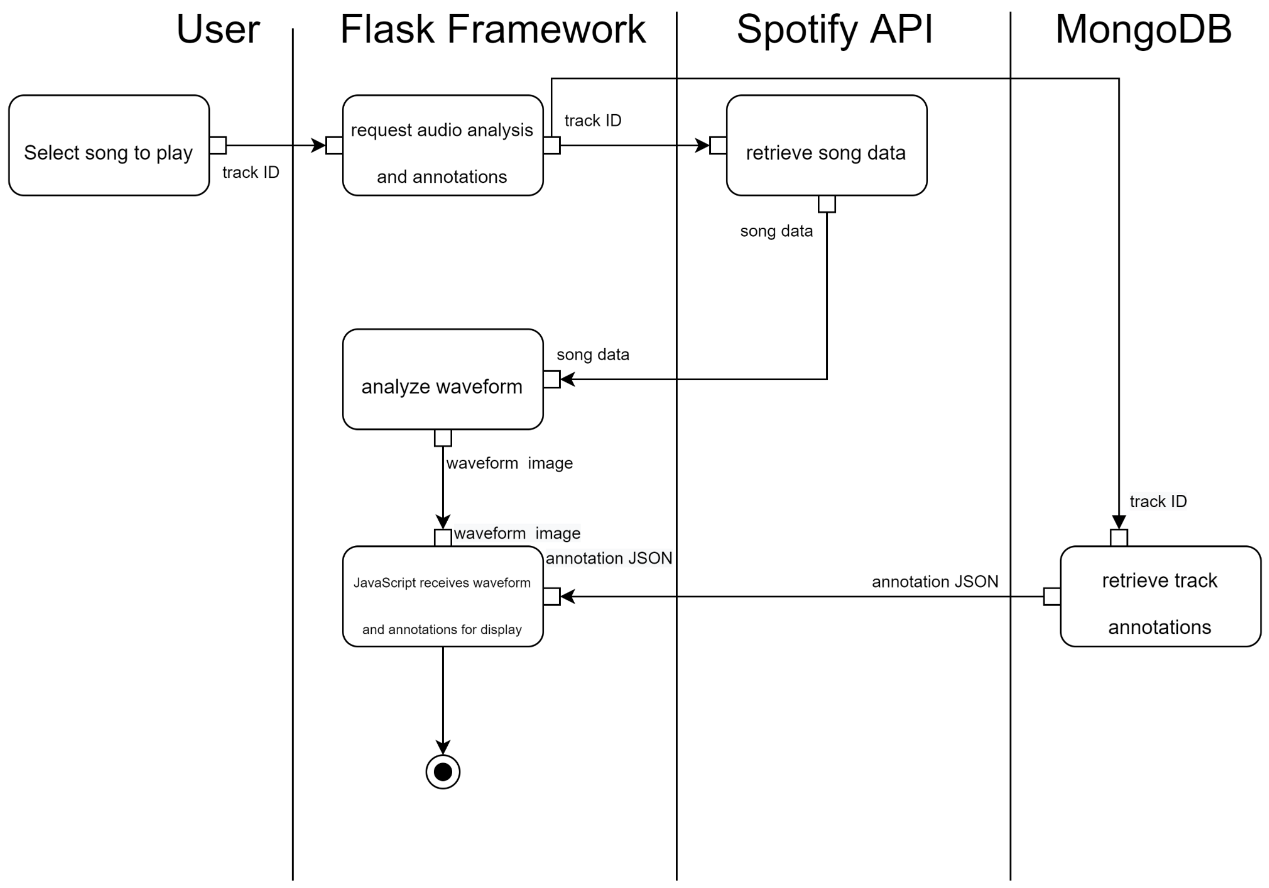
chrome-extension://mooikfkahbdckldjjndioackbalphokd/index.html#

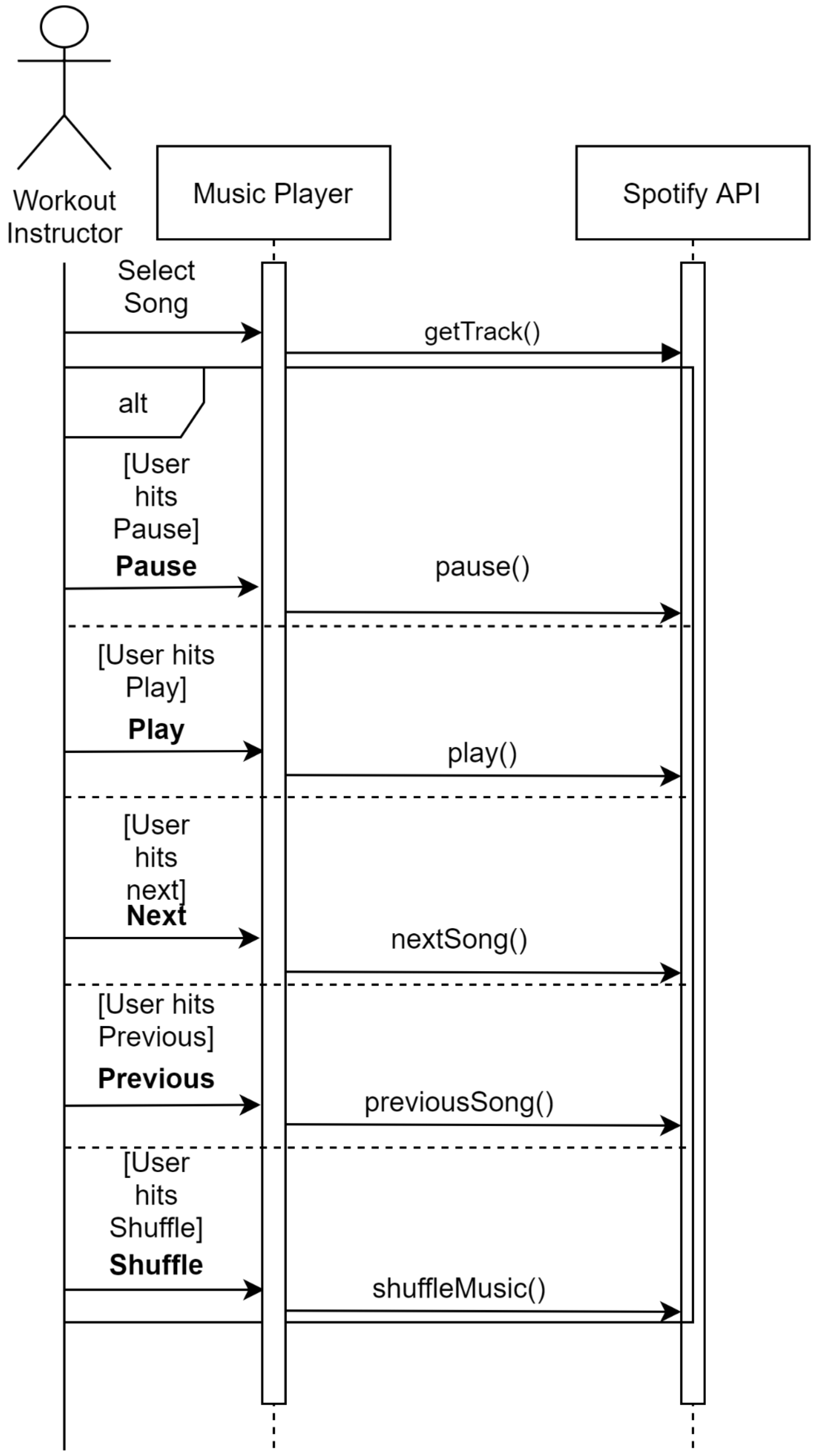
## Appendix A: Glossary

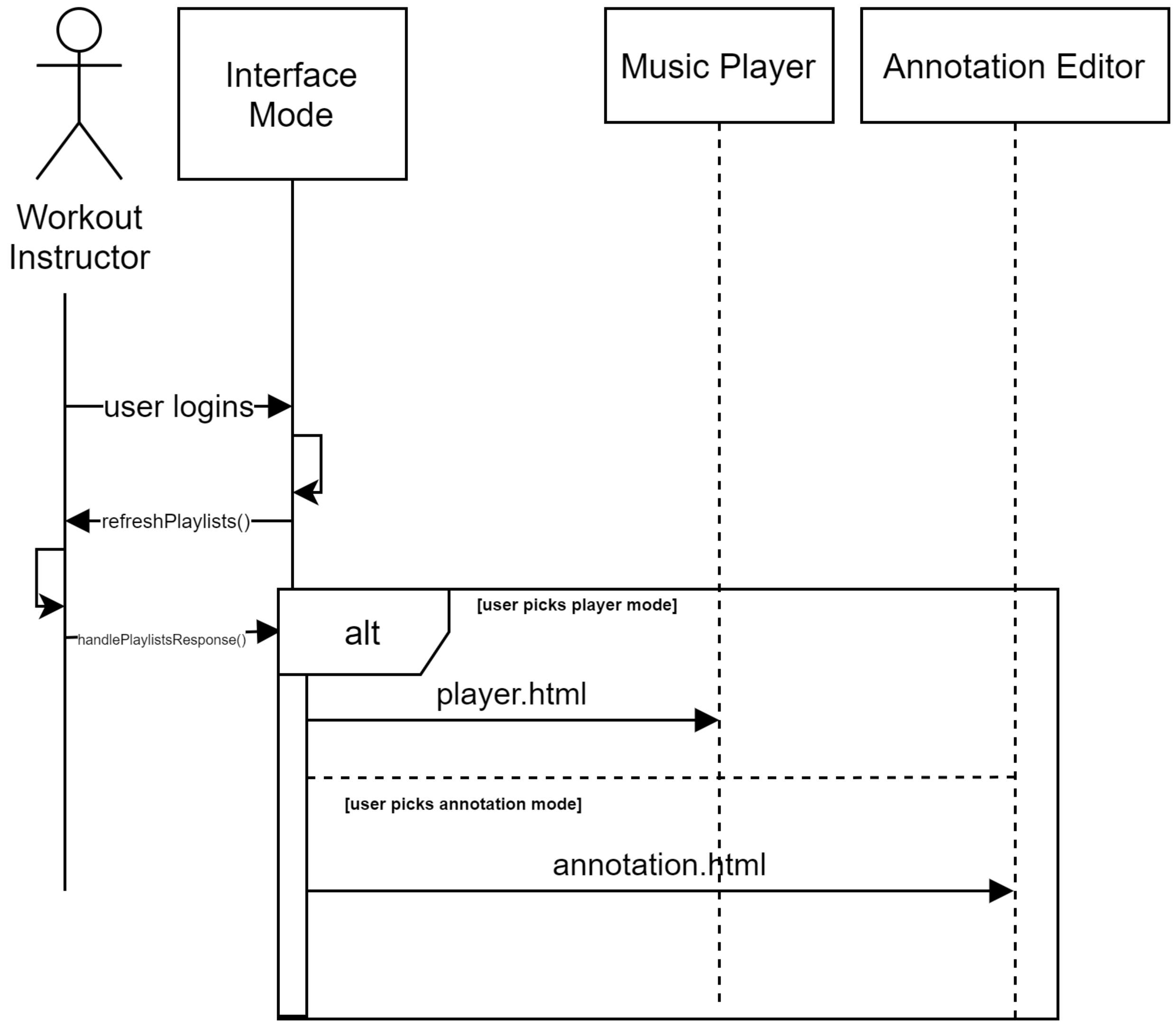
- MongoDB - Database program that uses Json styled documents
- Flask - Web Framework written in Python
- API - Application Programming Interface
- AWS - Amazon Web Services
- HTTPS - Hypertext Transfer Protocol Secure
- Waveform - The visualizations of the loudness of a song

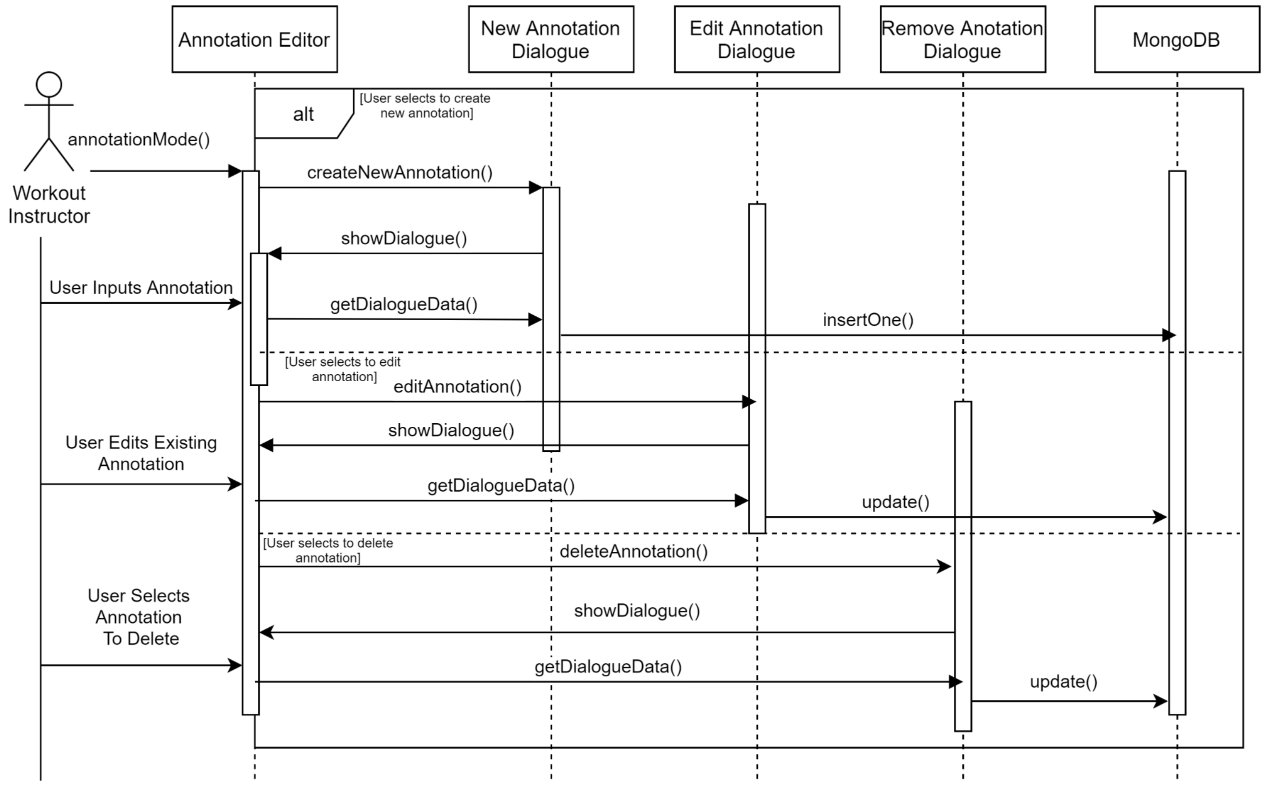
## Appendix B: Analysis Models

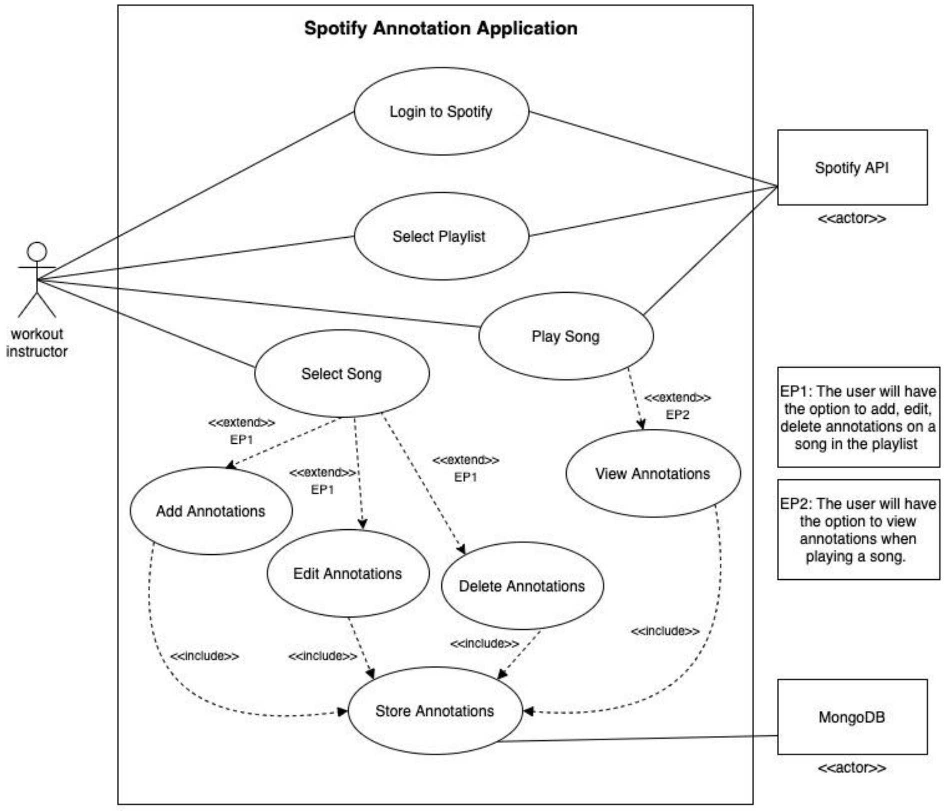



User
Flask Framework
Spotify API
MongoDB
Select song to play
track ID
request audio analysis and annotations
track ID
retrieve song data
song data
analyze waveform
song data
waveform image
waveform image
annotation JSON
JavaScript receives waveform and annotations for display
track ID
annotation JSON
retrieve track annotations


Workout Instructor
Music Player
Spotify API
Select Song
getTrack()
alt
[User hits Pause]
Pause
pause()
[User hits Play]
Play
play()
[User hits next]
Next
nextSong()
[User hits Previous]
Previous
previousSong()
[User hits Shuffle]
Shuffle
shuffleMusic()


Workout Instructor
Interface Mode
Music Player
Annotation Editor
user logins
refreshPlaylists()
handlePlaylistsResponse()
alt
[user picks player mode]
player.html
[user picks annotation mode]
annotation.html


Annotation Editor
New Annotation Dialogue
Edit Annotation Dialogue
Remove Anotation Dialogue
MongoDB
Workout Instructor
alt
[User selects to create new annotation]
annotationMode()
createNewAnnotation()
showDialogue()
User Inputs Annotation
getDialogueData()
insertOne()
[User selects to edit annotation]
editAnnotation()
showDialogue()
User Edits Existing Annotation
getDialogueData()
update()
[User selects to delete annotation]
deleteAnnotation()
User Selects Annotation To Delete
showDialogue()
getDialogueData()
update()


Spotify Annotation Application
Login to Spotify
Select Playlist
Play Song
Select Song
Add Annotations
Edit Annotations
Delete Annotations
View Annotations
Store Annotations
workout instructor
Spotify API
<<actor>>
MongoDB
<<actor>>
<<extend>> EP1
<<extend>> EP1
<<extend>> EP1
<<extend>> EP2
<<include>>
<<include>>
<<include>>
<<include>>
EP1: The user will have the option to add, edit, delete annotations on a song in the playlist
EP2: The user will have the option to view annotations when playing a song.