



## **Tunis Business School**

Bachelor of Science in Business Administration

IT 325: Web Services

## **BeatMotion**

# **Adjusting Music to your Workout**

## **REST API**

Done by

**Ons Gargouri**

Business Analytics and Information Technology Senior Student

**Academic Year: 2023 - 2024**

# **Acknowledgement and Declaration of Plagiarism**

I would like to acknowledge the efforts, commitment, and time I invested throughout the development of this project.

I declare that this work is the result of my ideas, and all sources consulted or cited are acknowledged.

A comprehensive list of references is provided at the end of this report.

I also declare that I have followed academic honesty and integrity principles, ensuring there is no misrepresentation, fabrication, or falsification of ideas, data, facts, or sources in my submission.

Date: January 17th, 2024

Ons Gargouri

# Abstract

This is the technical report of my final project for the IT325 Web Services course. The purpose of this document is to clearly define my RESTful API namely the “BeatMotion API”. It serves as a comprehensive guide to define the functionalities, features, and technical aspects. It also defines the various endpoints included and the technologies employed.

The document starts by establishing the context behind the BeatMotion API, highlighting its necessity for enhanced fitness exercise experience. Following this introduction, we introduce the intended use of the API and underlying assumptions. Subsequently, an overview of the API Design is provided, offering insights into the development steps and the technologies used. It acts as a roadmap for developers to learn the essence of this API. This section also explores more in-depth the playlist generation algorithm and how users’ preferences are mapped to Spotify tracks audio features. Additionally, a comprehensive exploration of endpoints and data models is presented to provide a more comprehensive understanding of the API’s architecture. The document concludes with the achievements, challenges, and areas for future improvements. This final segment offers reflections on the journey of developing the BeatMotion API, emphasizing the iterative nature of the project.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>API Scope</b>	<b>2</b>
2.1	Intended Use . . . . .	2
2.2	Limitations . . . . .	2
2.3	Assumptions . . . . .	2
2.4	Definition, Acronyms, or Abbreviations . . . . .	3
<b>3</b>	<b>API Design</b>	<b>4</b>
3.1	Development Steps and Technology Stacks . . . . .	4
3.2	Endpoints . . . . .	5
3.2.1	Authorization Endpoints . . . . .	5
3.2.2	Playlist Endpoints . . . . .	7
3.3	Playlist Generation Algorithm . . . . .	11
3.4	Models . . . . .	12
3.4.1	PlaylistModel . . . . .	12
3.4.2	PlaylistTrackModel . . . . .	13
3.4.3	UserModel . . . . .	13
<b>4</b>	<b>Conclusion and Discussion</b>	<b>14</b>
4.1	Summary of Achievements . . . . .	14
4.2	Discussion . . . . .	14
4.2.1	Challenges and Solutions . . . . .	14
4.2.2	Future Improvements . . . . .	14
4.3	Conclusion . . . . .	15
<b>5</b>	<b>Appendices</b>	<b>16</b>
5.1	Documentation : Swagger . . . . .	16

5.1.1	Authorization Operations . . . . .	16
5.1.2	Playlist Operations . . . . .	17
5.1.3	User preferences . . . . .	17
5.1.4	Generate Playlist . . . . .	18

<b>References</b>		<b>20</b>
-------------------	--	-----------

# 1 Introduction

With the higher recognition of the importance of fitness, there exists a growing need for innovative solutions that enhance the overall workout experience. Recognizing this demand, BeatMotion API was inspired by a personal need for a more tailored and motivating music playlist during exercise routines. Traditional streaming music services often presented challenges – whether finding existing playlists that did not match the desired workout pace or mood, or spending excessive time creating personalized playlists, distracting attention from the workout itself.

BeatMotion API was developed to enhance users' workout experiences by generating playlists aligned with their workout types, pace, and musical preferences while discovering new music. This API aims to boost motivation, engagement, and enjoyment during exercise routines through personalized playlists.

## 2 API Scope

### 2.1 Intended Use

The API can be integrated with fitness applications, wearables, and other compatible platforms by incorporating a personalized workout playlist feature. It is suited for various fitness levels and music preferences

### 2.2 Limitations

While the BeatMotion API aims to offer an engaging music customization experience, it is essential to acknowledge certain limitations:

- **Music Streaming Services Dependency:** The API is currently integrated with Spotify as its external music streaming service. Users are required to have valid Spotify accounts to access the full functionality of generated playlists.
- **Genre Availability:** The availability of specific genres is subject to Spotify's available genres.
- **Playlist Tracks Recommendations:** While the API employs an algorithm for playlist generation, ongoing development is planned to enhance this algorithm.

### 2.3 Assumptions

The following assumptions were made during the design and development of the BeatMotion API:

- **Internet Connection:** Users must have a stable internet connection during their workouts to access their Spotify account and generate or play playlists.
- **Potential for Expansion:** The API can evolve to integrate with other music streaming services in the future.

## 2.4 Definition, Acronyms, or Abbreviations

**API:** Application Programming Interface

**Playlist Generation Algorithm:** The specialized algorithm utilized by the BeatMotion API to dynamically craft personalized workout playlists.

**BPM:** Beats Per Minute

**Tempo:** Represents the estimated beats per minute (BPM) of a track.

**Energy:** represents the intensity; energetic tracks typically feel fast, loud, and noisy.

**Danceability:** A metric indicating how suitable a track is for dancing.

**Instrumentalness:** A measure describing whether a track contains vocals or is purely instrumental.

**Valence:** A metric describing the musical positiveness conveyed by a track, ranging from positive (happy, cheerful, euphoric) to negative (sad, depressed, angry).



## 3 API Design

### 3.1 Development Steps and Technology Stacks

This section provides a comprehensive explanation of the undertaken work, detailing the key activities and highlighting the diverse set of technologies employed throughout the project.

#### 1. Set Up Virtual Environment:

- Established an isolated virtual environment to ensure project dependencies are managed efficiently.

#### 2. Spotify Developer Dashboard:

- Created an account on the Spotify Developer Dashboard and created an application to obtain the Client ID and secret for Spotify Web API interactions.
- **Why Spotify:** Spotify Web API was chosen due to its free API access. It also offers detailed track audio features which are crucial for our playlist generation algorithm. In addition, python has a library called Spotipy that facilitates the interactions with this API.

#### 3. Flask App Creation:

- Implemented a Flask web application as the foundation for my playlist generation API.
- **Why Flask:** Flask offers simplicity, flexibility, and ease of use.

#### 4. Docker Setup:

- Used Docker for containerization to enhance scalability and consistent deployment across different environments.

#### 5. Playlist Generation Algorithm:

- Developed a playlist generation algorithm.

## 6. API Improvements with Flask Smorest, Blueprints, and MethodViews:

- Enhanced the API structure using Flask Smorest, Blueprints, and MethodViews for improved code organization and more readability.

## 7. Marshmallow Schemas for Response and Request Handling (Swagger):

- Implemented Marshmallow schemas to decorate responses and requests data.
- **Why Marshmallow:** Marshmallow simplifies data serialization and validation, and the integration with Swagger aids in clear API documentation.

## 8. Creating Models with SQLAlchemy (SQLite):

- Established data models using SQLAlchemy, with SQLite as the relational database backend.
- **Why SQLAlchemy and SQLite:** SQLAlchemy allows interaction with databases, and SQLite provides a lightweight database solution suitable for development.

## 9. Endpoint Creation and Testing with Insomnia:

- Developed various endpoints to cover different functionalities.
- Tested the endpoints using Insomnia.

# 3.2 Endpoints

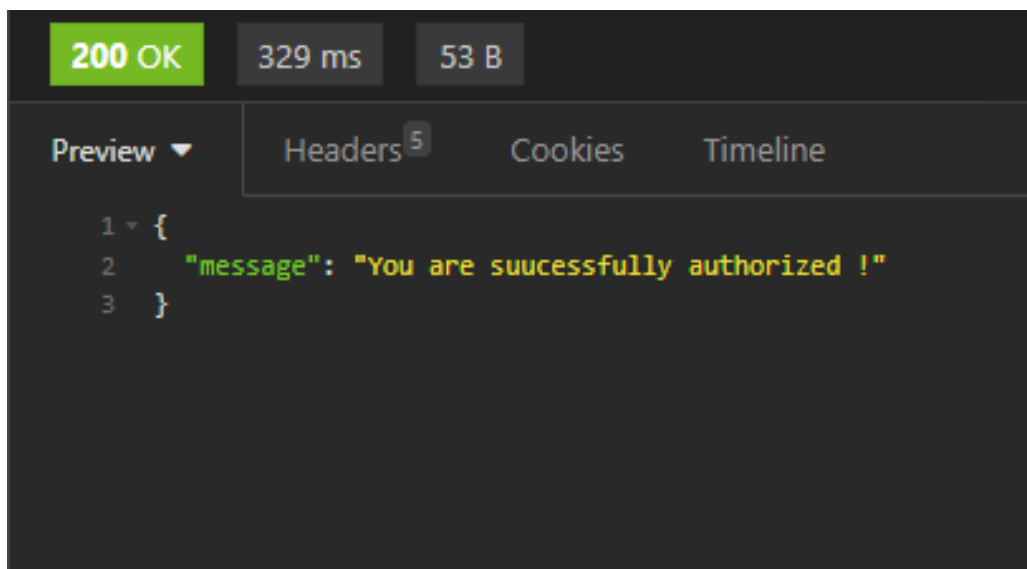
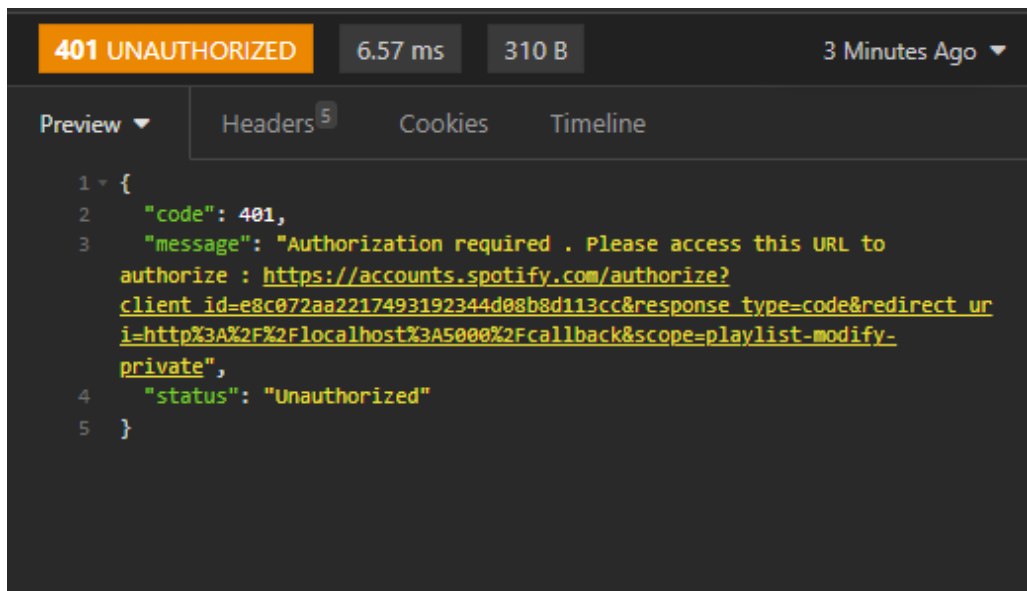
This section provides a comprehensive overview of the various API endpoints, corresponding HTTP requests, and a brief description of the functionality associated with each.

## 3.2.1 Authorization Endpoints

These endpoints focus on user authorization, and general user account management within the BeatMotion API.

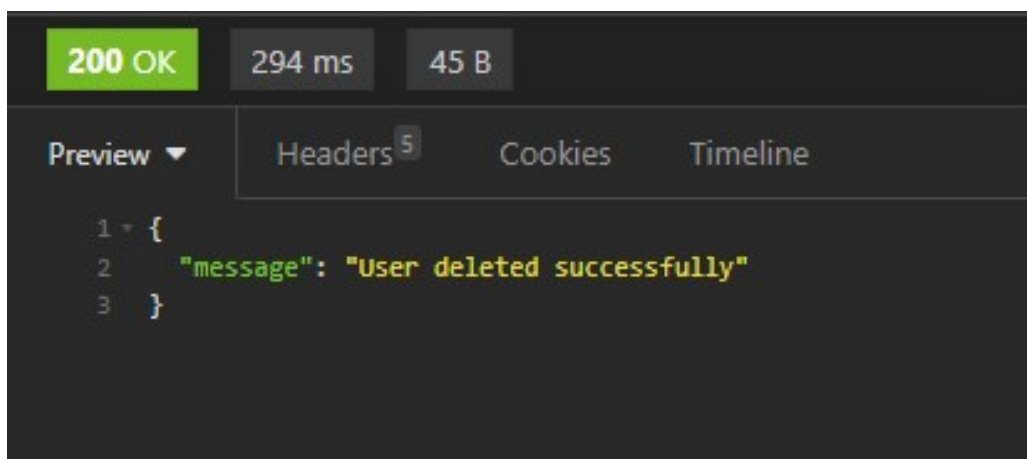
### **Authorization : GET /authorization**

This endpoint handles the process of authenticating and granting access to users' Spotify accounts.



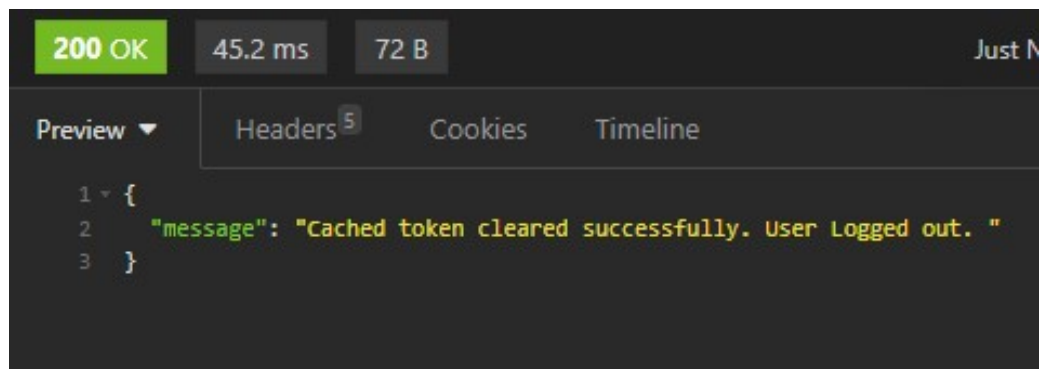
### Delete User: DELETE /user\_id

This endpoint is responsible for removing a specific user from the system.



### User Logout : GET /clearcache

This endpoint allows users to terminate their current session.

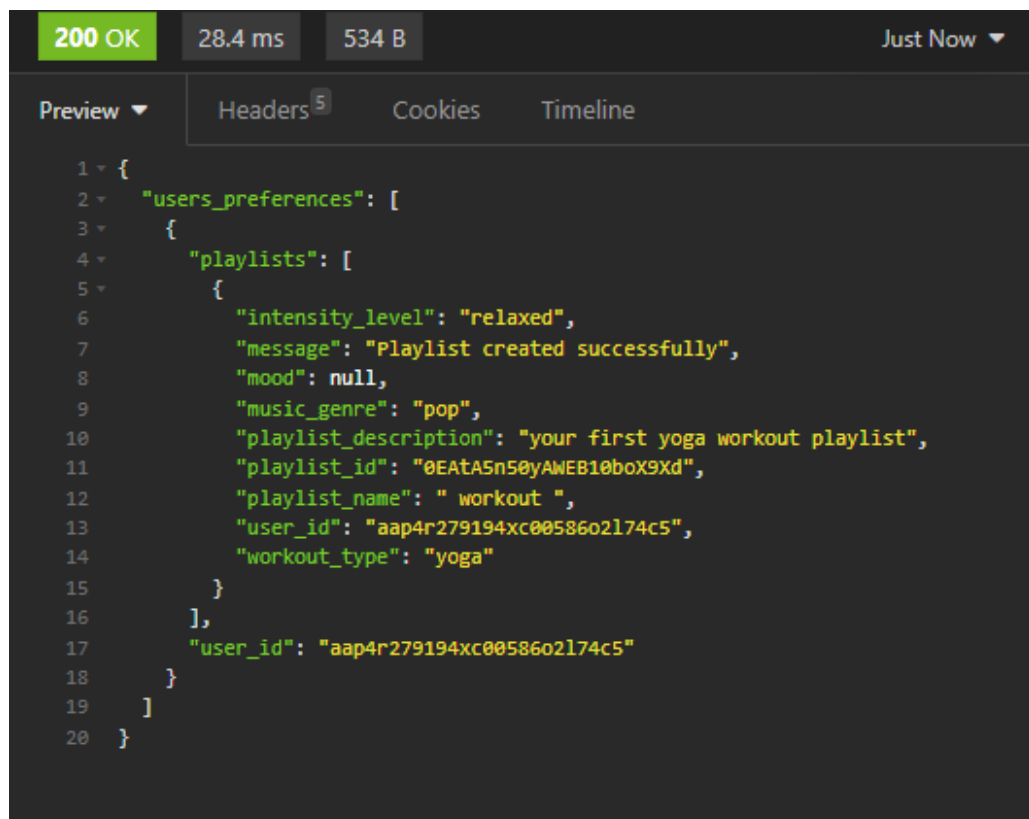


### 3.2.2 Playlist Endpoints

These endpoints illustrate various operations on playlists

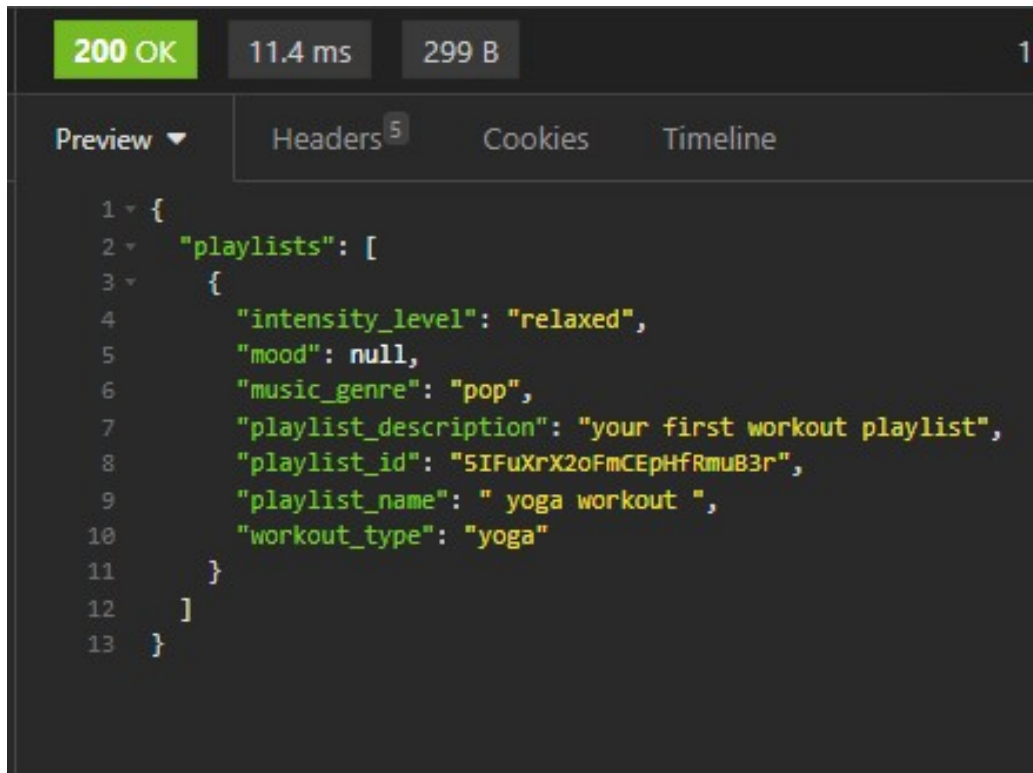
#### Display all users : GET /preferences

This endpoint is used to display all user's preferences and their corresponding playlists



#### Display a user : GET /preferences/user\_id

This endpoint is used to display specific user preferences and their corresponding playlists



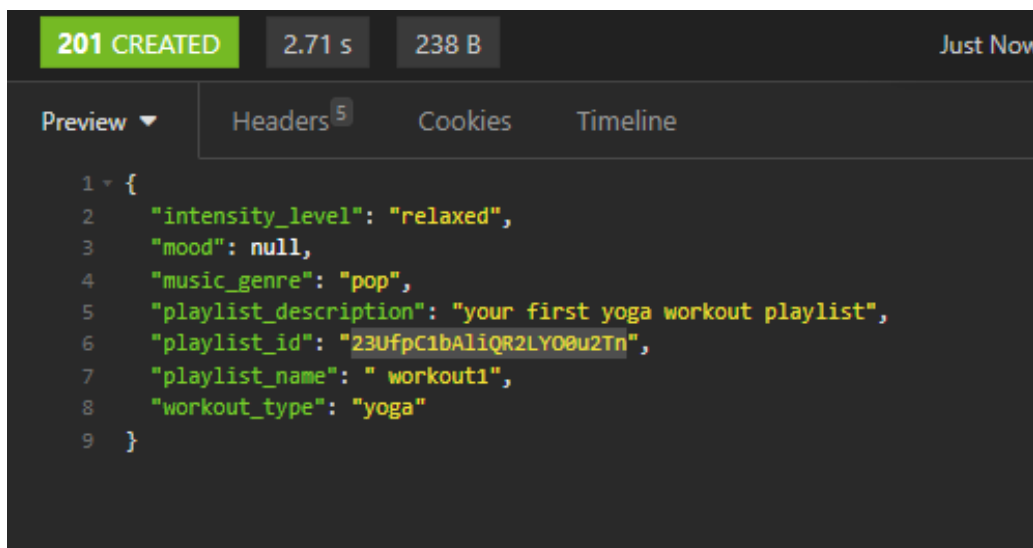
```
200 OK 11.4 ms 299 B 15

Preview ▾ Headers 5 Cookies Timeline

1 {
2   "playlists": [
3     {
4       "intensity_level": "relaxed",
5       "mood": null,
6       "music_genre": "pop",
7       "playlist_description": "your first workout playlist",
8       "playlist_id": "5IFuXrX2oFmCEpHfRmuB3r",
9       "playlist_name": " yoga workout ",
10      "workout_type": "yoga"
11    }
12  ]
13 }
```

### Generate a playlist: POST /generate\_playlist/user\_id

This endpoint creates personalized workout playlists based on user preferences. In this endpoint, we included the playlist generation algorithm.



```
201 CREATED 2.71 s 238 B Just Now

Preview ▾ Headers 5 Cookies Timeline

1 {
2   "intensity_level": "relaxed",
3   "mood": null,
4   "music_genre": "pop",
5   "playlist_description": "your first yoga workout playlist",
6   "playlist_id": "23UfpC1bAliQR2LY00u2Tn",
7   "playlist_name": " workout1",
8   "workout_type": "yoga"
9 }
```

### Playlist Tracks : GET playlist\_tracks/playlist\_id

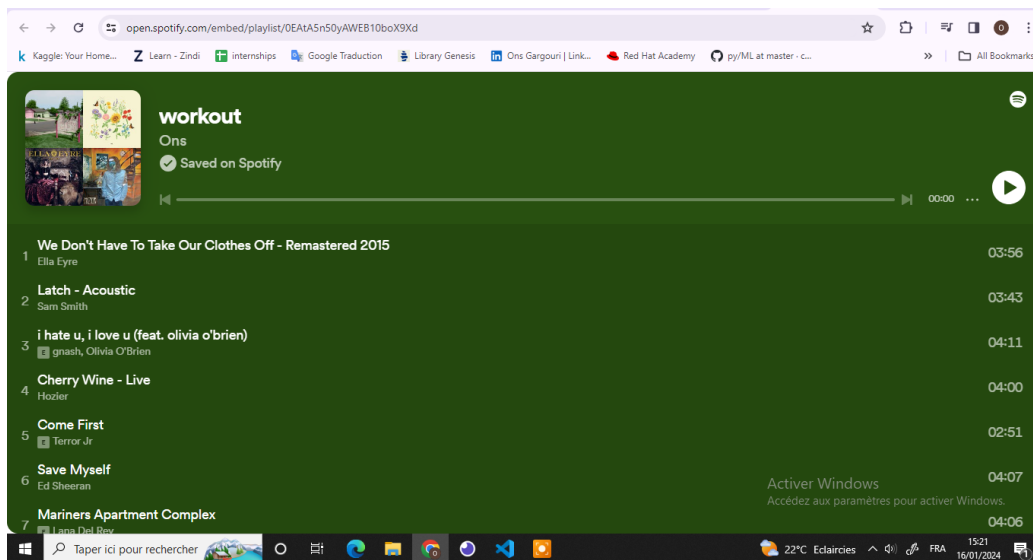
This endpoint is used to retrieve playlist tracks.

```
200 OK 92.9 ms 1641 B Just Now ▾
Preview ▾ Headers 5 Cookies Timeline
1 {
2   "playlist_tracks": [
3     {
4       "artist_name": "Ella Eyre",
5       "id": 1,
6       "track_name": "We Don't Have To Take Our Clothes Off - Remastered
7         2015",
8       "uri": "spotify:track:49GpGYG1i1xcxovgYf0k4c"
9     },
10    {
11      "artist_name": "Sam Smith",
12      "id": 2,
13      "track_name": "Latch - Acoustic",
14      "uri": "spotify:track:2c4dgwC42CRH1XOSFshSPf"
15    },
16    {
17      "artist_name": "gnash",
18      "id": 3,
19      "track_name": "i hate u, i love u (feat. olivia o'brien)",
20      "uri": "spotify:track:7vRriwrloYVaoAe3a9wJHe"
21    },
22    {
23      "artist_name": "Hozier",
24      "id": 4,
25      "track_name": "Cherry Wine - Live",
26      "uri": "spotify:track:5tbJvz7EzXc9Z8odKvfvJm"
27    },
28    {
29      "artist_name": "Terror Jr",
30      "id": 5,
31      "track_name": "Come First",
32      "uri": "spotify:track:5fo4LbJ810aB54wCM5wPMv"
33    }
34  ]
35 }
```

### Play Playlist : GET /playlist/playlist\_id

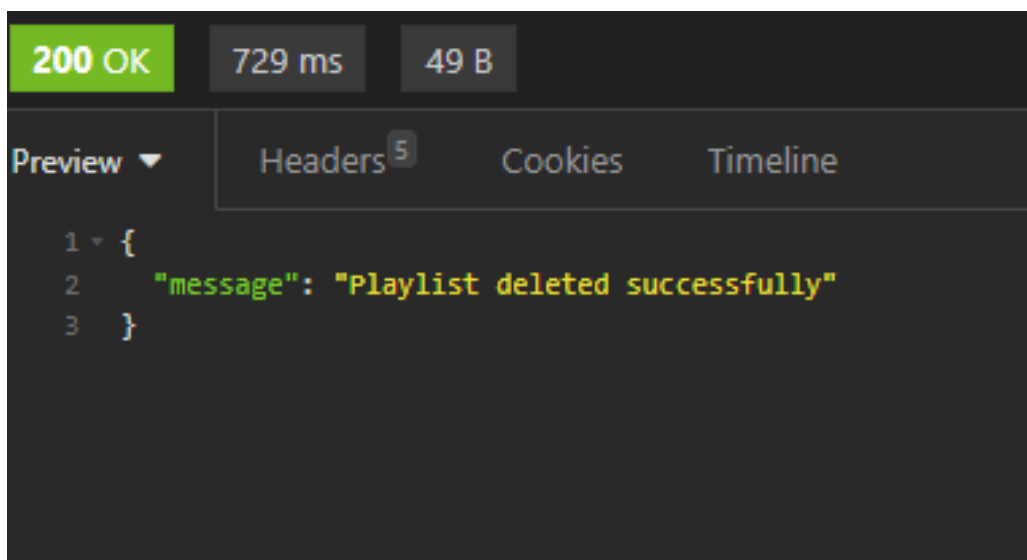
This endpoint is used to play a specific playlist.

```
200 OK 33.3 ms 104 B Just Now ▾
Preview ▾ Headers 5 Cookies Timeline
1 {
2   "Link_to_play_Spotify_Playlist":
3     "https://open.spotify.com/embed/playlist/0EAtA5n50yAWEb10boX9Xd"
4 }
```



### Delete Playlist : DELETE /delete\_playlist/playlist\_id

This endpoint is used to delete a specific playlist.



### Tracks features : GET /tracks

This endpoint is mainly for testing purposes during the development phase. It was used to verify that the audio features match the mapping of the user preferences.

```

200 OK 473 ms 7.5 KB Just Now
Preview Headers Cookies Timeline
1 {
2   "message": "Audio features retrieved successfully",
3   "tracks_info": [
4     {
5       "audio_features": {
6         "acousticness": 0.889,
7         "analysis_url": "https://api.spotify.com/v1/audio-
analysis/1rbuWPnmxmN16sN5fexycY",
8         "danceability": 0.748,
9         "duration_ms": 202920,
10        "energy": 0.284,
11        "id": "1rbuWPnmxmN16sN5fexycY",
12        "instrumentalness": 5.77e-06,
13        "key": 2,
14        "liveness": 0.155,
15        "loudness": -11.865,
16        "mode": 1,
17        "speechiness": 0.0676,
18        "tempo": 89.955,
19        "time_signature": 4,
20        "track_href":
21        "https://api.spotify.com/v1/tracks/1rbuWPnmxmN16sN5fexycY",
22        "type": "audio_features",
23        "uri": "spotify:track:1rbuWPnmxmN16sN5fexycY",
24        "valence": 0.533
25      },
26      "uri": "spotify:track:1rbuWPnmxmN16sN5fexycY"
27    },
28    {
29      "audio_features": {
30        "acousticness": 0.951,
31        "analysis_url": "https://api.spotify.com/v1/audio-

```

### 3.3 Playlist Generation Algorithm

This section explains BeatMotion Playlist Generation Algorithm: Users input their preferences, which include workout type, intensity levels, and music genre. It may also include danceability, instrumentalness, and mood. The algorithm receives user preferences, maps them to audio features, and uses the Spotify API Recommendations endpoint to request for tracks filtered by these audio features. If tracks are found, an empty playlist is created on the user's Spotify account. The algorithm adds the tracks obtained from the Spotify API recommendations to the user's newly created playlist.

This table shows the possible users' preferences and their corresponding mapping.



User Preferences	Mapping to Audio Features
Workout Type	Tempo Ranges (BPM)
running	(120, 160)
walking	(80, 120)
cycling	(120, 160)
HIIT	(160, 180)
weightlifting	(100, 120)
yoga	(60, 100)
boxing	(160, 180)
Intensity Level	Energy
energetic	(0.7, 0.9)
moderate	(0.4, 0.6)
relaxed	(0.1, 0.3)
Music Genre	Spotify Genre
pop	pop
rock	rock
hipHop	hip-hop
electronic	electronic
classical	classical
Instrumentalness	Instrumentalness
instrumental	(0.6, 1)
vocals	(0.1, 0.5)
Danceability	Danceability
danceable	(0.5, 1)
normal	(0.1, 0.4)
Mood Preference	Valence
positive	(0.5, 1)
negative	(0.1, 0.4)

Table 3.1: User Preferences and Corresponding Mapping to Spotify Audio Features

## 3.4 Models

This section provides an overview of the core models within the API namely **PlaylistModel**, **PlaylistTrackModel**, and **UserModel**. These models work together to organize and store information about users, their playlists, and the individual tracks within those playlists. The relationships established between the models ensure data integrity and provide a structured way to navigate and retrieve information.

### 3.4.1 PlaylistModel

The PlaylistModel represents a workout playlist in the application. Each playlist has a unique identifier (playlist\_id) and includes information such as the playlist name, description, workout type, intensity level, music genre, and mood. The model is connected to the UserModel through a foreign key relationship (user\_id), establishing a link between users and their playlists (many-to-

one relationship). Additionally, each playlist may have multiple tracks associated with it, which are managed through the relationship of the `playlist_tracks` (one-to-many relationship).

### **3.4.2 PlaylistTrackModel**

The `PlaylistTrackModel` corresponds to individual tracks within a playlist. It stores details such as the track name, artist name, and URI (Uniform Resource Identifier) for easy retrieval. The model is linked to the `PlaylistModel` through the `playlist_id` foreign key, creating a relationship that allows tracks to be associated with specific playlists (many-to-one relationship).

### **3.4.3 UserModel**

The `UserModel` represents a user in the system and is linked to the `PlaylistModel` through the `user_id` attribute. Each user has a unique identifier (`user_id`). The `UserModel` includes a relationship with the `PlaylistModel`, indicating that users can have multiple playlists(one to many relationships).

## 4 Conclusion and Discussion

### 4.1 Summary of Achievements

In this project, I successfully designed and implemented the BeatMotion API, aiming to enhance the workout experience by providing generated playlists tailored to users' preferences and workout types.

Key achievements include:

- **Playlist Generation Algorithm:** Developed an algorithm that interprets user preferences, maps them to audio features, and generates personalized playlists on the user's Spotify account.
- **API Structure and Endpoints:** Implemented a robust API structure with various endpoints for user authorization, playlist generation, play, and management.
- **Integration with Spotify:** Successfully integrated the API with Spotify, leveraging its free API access and detailed track audio features for playlist customization.

### 4.2 Discussion

#### 4.2.1 Challenges and Solutions

Throughout the development process, several challenges were encountered. One notable challenge involved unfound tracks when a user selected all preferences, leading to an impractical number of potential combinations. The solution involved refining the playlist generation algorithm by making only certain preferences, such as workout type, intensity level, and genre, mandatory.

#### 4.2.2 Future Improvements

While the BeatMotion API has achieved its initial goals, there are opportunities for future improvements and enhancements:

- **API deployment** : Future work should focus on deploying the BeatMotion API to a production environment that is accessible to a broader user base.
- **Enhanced User Authentication**: Strengthening user authentication is essential for ensuring the security of user data and preventing unauthorized access.
- **Algorithm Improvement**: Continuous refinement of the playlist generation algorithm to provide more accurate and personalized recommendations.
- **Integration with Additional Platforms**: Explore possibilities for integrating the API with other music streaming services, expanding user accessibility.

## 4.3 Conclusion

In conclusion, BeatMotion API represents a significant step towards enhancing the workout experience through personalized music playlists. The successful implementation of key features and the ongoing commitment to improvement make the API a valuable tool for users seeking engaging and motivating playlists.

The journey from ideation to implementation has been both challenging and rewarding. The knowledge acquired during this project will guide future initiatives, ensuring continued innovation and improvement.

# 5 Appendices

## 5.1 Documentation : Swagger

### 5.1.1 Authorization Operations

authentication Authentication Operations		^
GET	/clearcache	▼
GET	/authorization	▼
GET	/callback	▼
DELETE	/user_id	▼

### authorization

Code	Description
200	<div>OK</div> <div>Media type</div> <div>application/json ▼</div> <div>Controls Accept header.</div> <div>Example Value   Schema</div> <div><pre>{   "message": "string",   "auth_url": "string" }</pre></div>

### Delete a user

Parameters	
Name	Description
user_id * required string (path)	user_id

## 5.1.2 Playlist Operations

playlists Operations on playlists		^
GET	/preferences	▼
GET	/preferences/{user_id}	▼
POST	/generate_playlist/{user_id}	▼
DELETE	/delete_playlist/{user_id}/{playlist_id}	▼
GET	/playlist/{playlist_id}	▼
GET	/playlist_tracks/{playlist_id}	▼
GET	/tracks	▼

## 5.1.3 User preferences

Code	Description
200	<div><div>OK</div><div>Media type</div><div>application/json</div><div>Controls Accept header.</div><div>Example Value   Schema</div><div><pre>{   "user_id": "string",   "playlists": [     {       "playlist_id": "string",       "playlist_name": "string",       "playlist_description": "string",       "workout_type": "string",       "intensity_level": "string",       "music_genre": "string",       "mood": "string",       "message": "string"     }   ],   "message": "string" }</pre></div></div>

# 5.1.4 Generate Playlist

Parameters

Name	Description
<b>user_id</b> * required string (path)	<div>user_id</div>

Request body required

Example Value | Schema

```
{
  "playlist_name": "string",
  "playlist_description": "string",
  "workout_type": "string",
  "intensity_level": "string",
  "music_genre": "string",
  "mood": "string"
}
```

Code

Description

201

Created

Media type  

application/json

Controls Accept header.

Example Value | Schema

```
{
  "playlist_id": "string",
  "playlist_name": "string",
  "playlist_description": "string",
  "workout_type": "string",
  "intensity_level": "string",
  "music_genre": "string",
  "mood": "string",
  "message": "string"
}
```

## Play Playlist

Code	Description
200	<div>OK</div> <div>Media type</div> <div><div>application/json</div></div> <div>Controls Accept header.</div> <div>Example Value   Schema</div> <div><pre>{   "Link_to_play_Spotify_Playlist": "string" }</pre></div>

## Playlist tracks

Code	Description
200	<div>OK</div> <div>Media type</div> <div><div>application/json</div></div> <div>Controls Accept header.</div> <div>Example Value   Schema</div> <div><pre>{   "playlist_tracks": [     {}   ] }</pre></div>



# References

1. <https://developer.spotify.com/documentation/web-api/tutorials/getting-started>
2. <https://spotipy.readthedocs.io/en/2.22.1/>
3. <https://asifulalam.me/articles/Add%20Spotify%20Playlist%20on%20Your%20Website%20in%20Just%20a%20Few%20Simple%20Steps#:~:text=Process%2D1%3A%20navigate%20to%20your,size%20of%20your%20embedded%20player.>
4. <https://medium.com/@ishita.joshi6.ij/from-basics-to-beats-understanding-apis-and->
5. <https://github.com/juani1499/spotify-playlist-sorter>