AudioMuse-AI API Documentation

Contents

1 Introduction

This document provides a comprehensive overview of the AudioMuse-AI backend API endpoints, their functionalities, request/response formats, and example usage.

Base URL

The base URL for all API endpoints is relative to your Flask application's root (e.g., `http://localhost:8000`).

API Endpoints Summary

Endpoint	Method	Description
/api/config	GET	Retrieves current application configuration parameters.
/api/analysis/start	POST	Initiates a background task for music analysis.
/api/clustering/start	POST	Initiates a background task for music clustering and playlist ge
/api/status/<task_id>	GET	Retrieves the real-time status and details of a specific task.
/api/cancel/<task_id>	POST	Cancels a specific active task and its child tasks.
/api/cancel_all/<task_type_prefix>	POST	Cancels all active main tasks of a specified type.
/api/last_task	GET	Retrieves the status of the most recently recorded main task.
/api/active_tasks	GET	Retrieves information about the currently active main task.
/api/playlists	GET	Retrieves all automatically generated playlists.

2 Configuration Endpoints

2.1 Get Configuration Parameters

Retrieves the current configuration parameters of the AudioMuse-AI application.

- **URL:** `/api/config`
- **Method:** `GET`
- **Description:** This endpoint provides the default and currently set configuration values for Jellyfin integration, music analysis, and clustering algorithms.
- **Request Body:** None
- **Response:** `application/json`

```
{
    "jellyfin_url": "http://your-jellyfin-server:8096",
    "jellyfin_user_id": "your_jellyfin_user_id",
    "jellyfin_token": "your_jellyfin_api_token",
    "num_recent_albums": 5,
    "max_distance": 0.5,
    "max_songs_per_cluster": 20,
    "max_songs_per_artist": 3,
    "cluster_algorithm": "kmeans",
    "num_clusters_min": 3,
    "num_clusters_max": 10,
    "dbscan_eps_min": 0.1,
    "dbscan_eps_max": 0.8,
    "dbscan_min_samples_min": 2,
    "dbscan_min_samples_max": 5,
    "gmm_n_components_min": 3,
    "gmm_n_components_max": 10,
    "pca_components_min": 2,
    "pca_components_max": 5,
    "top_n_moods": 5,
    "mood_labels": ["mood/acoustic", "mood/aggressive", "mood/ambient", "mood/chilling", "mood/dark", "
     mood/energetic", "mood/epic", "mood/happy", "mood/melancholy", "mood/party", "mood/relaxed", "mood/
     romantic", "mood/sad", "mood/serious", "mood/sexy", "mood/sleepy", "mood/warm"],
    "clustering_runs": 100
}
```

Listing 1: Example Response: Get Configuration Parameters

3 Task Management Endpoints

3.1 Start Music Analysis Task

Initiates a background task to analyze recent albums from Jellyfin.

- **URL:** `/api/analysis/start`
- **Method:** `POST`
- **Description:** This endpoint queues a new analysis task. The task will fetch recent albums from the configured Jellyfin server, download tracks, analyze their audio features (tempo, key, scale, moods), and save the analysis results to the database.
- **Request Body:** `application/json`

```
{
    "jellyfin_url": "http://your-jellyfin-server:8096",
    "jellyfin_user_id": "your_jellyfin_user_id",
    "jellyfin_token": "your_jellyfin_api_token",
    "num_recent_albums": 10,
    "top_n_moods": 5
}
```

Listing 2: Example Request Body: Start Music Analysis Task

- `jellyfin_url` (string, optional): The URL of your Jellyfin server. Defaults to the value in `config.py`.
- `jellyfin_user_id` (string, optional): Your Jellyfin user ID. Defaults to the value in `config.py`.
- `jellyfin_token` (string, optional): Your Jellyfin API token. Defaults to the value in `config.py`.
- `num_recent_albums` (integer, optional): The number of most recent albums to fetch and analyze. Defaults to the value in `config.py`.
- `top_n_moods` (integer, optional): The number of top moods to extract for each track. Defaults to the value in `config.py`.

```
{
    "task_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
    "task_type": "main_analysis",
    "status": "queued"
}
```

Listing 3: Example Response: Start Music Analysis Task

3.2 Start Music Clustering Task

Initiates a background task to perform music clustering and generate playlists based on analyzed tracks.

- **URL:** `/api/clustering/start`
- **Method:** `POST`
- **Description:** This endpoint queues a new clustering task. It uses the previously analyzed track data to group songs into clusters based on their audio features and generates playlists from these clusters. Various clustering algorithms and parameters can be configured.
- **Request Body:** `application/json`

```
{
    "clustering_method": "kmeans",
    "num_clusters_min": 3,
    "num_clusters_max": 10,
    "dbscan_eps_min": 0.1,
    "dbscan_eps_max": 0.8,
    "dbscan_min_samples_min": 2,
    "dbscan_min_samples_max": 5,
    "gmm_n_components_min": 3,
    "gmm_n_components_max": 10,
    "pca_components_min": 2,
    "pca_components_max": 5,
    "clustering_runs": 100,
```

```
    "max_songs_per_cluster": 20
}
```

Listing 4: Example Request Body: Start Music Clustering Task

- `clustering_method` (string, optional): The clustering algorithm to use (`kmeans`, `dbscan`, or `gmm`). Defaults to the value in `config.py`.
- `num_clusters_min` (integer, optional): (K-Means/GMM) Minimum number of clusters to try. Defaults to `config.NUM_CLUSTERS_MIN`.
- `num_clusters_max` (integer, optional): (K-Means/GMM) Maximum number of clusters to try. Defaults to `config.NUM_CLUSTERS_MAX`.
- `dbscan_eps_min` (float, optional): (DBSCAN) Minimum epsilon value for DBSCAN. Defaults to `config.DBSCAN_EPS_MIN`.
- `dbscan_eps_max` (float, optional): (DBSCAN) Maximum epsilon value for DBSCAN. Defaults to `config.DBSCAN_EPS_MAX`.
- `dbscan_min_samples_min` (integer, optional): (DBSCAN) Minimum samples for DBSCAN. Defaults to `config.DBSCAN_MIN_SAMPLES_MIN`.
- `dbscan_min_samples_max` (integer, optional): (DBSCAN) Maximum samples for DBSCAN. Defaults to `config.DBSCAN_MIN_SAMPLES_MAX`.
- `gmm_n_components_min` (integer, optional): (GMM) Minimum number of components for GMM. Defaults to `config.GMM_N_COMPONENTS_MIN`.
- `gmm_n_components_max` (integer, optional): (GMM) Maximum number of components for GMM. Defaults to `config.GMM_N_COMPONENTS_MAX`.
- `pca_components_min` (integer, optional): Minimum number of PCA components to use. Defaults to `config.PCA_COMPONENTS_MIN`.
- `pca_components_max` (integer, optional): Maximum number of PCA components to use. Defaults to `config.PCA_COMPONENTS_MAX`.
- `clustering_runs` (integer, optional): The total number of clustering iterations to perform to find the best solution. Defaults to `config.CLUSTERING_RUNS`.
- `max_songs_per_cluster` (integer, optional): The maximum number of songs to include in a single generated playlist. Defaults to `config.MAX_SONGS_PER_CLUSTER`.

```
{
    "task_id": "b5c6d7e8-f9a0-1234-5678-90abcdef1234",
    "task_type": "main_clustering",
    "status": "queued"
}
```

Listing 5: Example Response: Start Music Clustering Task

3.3 Get Task Status

Retrieves the current status and details of a specific task.

- **URL:** `/api/status/<task_id>`
- **Method:** `GET`
- **Description:** This endpoint provides real-time updates on the progress, status, and detailed logs for a given task, whether it's a main task or a sub-task (e.g., album analysis).
- **URL Parameters:**
  - `task_id` (string, required): The unique ID of the task.
- **Response:** `application/json`

```
{
    "task_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
    "state": "PROGRESS",
    "status_message": "Processing albums: 3/10 completed.",
    "progress": 30,
    "details": {
        "album_name": "Album Title Example",
        "current_track": "Song Name by Artist",
```

```
        "current_track_analysis": {
            "name": "Song Name by Artist",
            "tempo": 120.5,
            "key": "C",
            "scale": "major",
            "moods": {
                "mood/energetic": 0.85,
                "mood/happy": 0.72
            }
        },
        "log": [
            "[2023-10-27 10:00:00] Main analysis task started.",
            "[2023-10-27 10:00:05] Found 10 albums to process.",
            "[2023-10-27 10:01:15] Analyzing track: Song Name by Artist (1/5)",
            "[2023-10-27 10:01:30] Analyzed 'Song Name by Artist'. Tempo: 120.50, Key: C major. Moods:
     mood/energetic:0.85, mood/happy:0.72"
        ],
        "albums_completed": 3,
        "total_albums": 10,
        "status_message": "Processing albums: 3/10 completed."
    },
    "task_type_from_db": "main_analysis"
}
```


Listing 6: Example Response: Get Task Status

Possible `state` values:

- `QUEUED`: Task is waiting in the queue.
- `STARTED`: Task has begun execution. `PROGRESS`: Task is actively working (granular updates).
- `SUCCESS` / `FINISHED`: Task completed successfully.
- `FAILURE` / `FAILED`: Task encountered an error and failed.
- `REVOKED` / `CANCELED`: Task was explicitly cancelled.
- `UNKNOWN`: Task ID not found or its state is not recognized.

3.4 Cancel Task

Cancels a specific active task and its associated child tasks.

- **URL:** `/api/cancel/<task_id>`
- **Method:** `POST`
- **Description:** Sends a cancellation signal to the specified task. This attempts to stop the task gracefully and marks it as `REVOKED` in the database. Child tasks spawned by the main task will also be targeted for cancellation.
- **URL Parameters:**
  - `task_id` (string, required): The unique ID of the task to cancel.
- **Response:** `application/json`

Success (Status: 200 OK):


```
{
    "message": "Task a1b2c3d4-e5f6-7890-1234-567890abcdef and its children cancellation initiated. 5 total
      jobs affected.",
    "task_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
    "cancelled_jobs_count": 5
}
```


Listing 7: Example Response: Cancel Task (Success)

Not Found (Status: 404 Not Found):


```
{
    "message": "Task a1b2c3d4-e5f6-7890-1234-567890abcdef not found in database.",
    "task_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef"
}
```


Listing 8: Example Response: Cancel Task (Not Found)

Bad Request (Status: 400 Bad Request):

```
{
    "message": "Task could not be cancelled (e.g., already completed or not found in active state).",
    "task_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef"
}
```

Listing 9: Example Response: Cancel Task (Bad Request)

3.5 Cancel All Tasks by Type

Cancels all active main tasks of a specific type.

- **URL:** `/api/cancel_all/<task_type_prefix>`
- **Method:** `POST`
- **Description:** This endpoint allows for bulk cancellation of tasks. It will find all active main tasks (those without a `parent_task_id`) matching the given `task_type_prefix` and attempt to cancel them and their children.
- **URL Parameters:**
  - `task_type_prefix` (string, required): The type of tasks to cancel (e.g., `main_analysis`, `main_clustering`).
- **Response:** `application/json`

Success (Status: 200 OK):

```
{
    "message": "Cancellation initiated for 2 main tasks of type 'main_analysis' and their children. Total
     jobs affected: 12.",
    "cancelled_main_tasks": [
        "a1b2c3d4-e5f6-7890-1234-567890abcdef",
        "f0e9d8c7-b6a5-4321-9876-543210fedcba"
    ]
}
```

Listing 10: Example Response: Cancel All Tasks by Type (Success)

Not Found (Status: 404 Not Found):

```
{
    "message": "No active tasks of type 'main_analysis' found to cancel."
}
```

Listing 11: Example Response: Cancel All Tasks by Type (Not Found)

3.6 Get Last Overall Task Status

Retrieves the status of the most recently recorded main task.

- **URL:** `/api/last_task`
- **Method:** `GET`
- **Description:** This endpoint provides the status of the last main task that was started, regardless of its current state (queued, running, or completed/failed/revoked). This is useful for displaying the last known activity.
- **Request Body:** None
- **Response:** `application/json`

Task Found (Status: 200 OK):

```
{
    "task_id": "c1d2e3f4-g5h6-7890-1234-567890abcdef",
    "task_type": "main_clustering",
    "status": "SUCCESS",
    "progress": 100,
    "details": {
        "message": "Playlists generated and updated on Jellyfin! Best diversity score: 0.75.",
        "best_score": 0.75,
        "best_params": {
```

```
            "clustering_method_config": {
                "method": "kmeans",
                "params": {
                    "n_clusters": 5
                }
            },
            "pca_config": {
                "enabled": true,
                "components": 3
            },
            "max_songs_per_cluster": 20,
            "run_id": 42
        },
        "num_playlists_created": 8,
        "log": ["Task completed successfully. Final status: Playlists generated and updated on Jellyfin!
     Best diversity score: 0.75."]
    }
}
```


Listing 12: Example Response: Get Last Overall Task Status (Task Found)

No Previous Task (Status: 200 OK):


```
{
    "task_id": null,
    "task_type": null,
    "status": "NO_PREVIOUS_MAIN_TASK",
    "details": {
        "log": ["No previous main task found."]
    }
}
```


Listing 13: Example Response: Get Last Overall Task Status (No Previous Task)

3.7 Get Active Tasks

Retrieves information about the currently active main task, if any.

- **URL:** `/api/active_tasks`
- **Method:** `GET`
- **Description:** This endpoint checks for any main tasks that are currently in a non-terminal state (i.e., `PENDING`, `STARTED`, `PROGRESS`, `QUEUED`). It returns the most recent active main task.
- **Request Body:** None
- **Response:** `application/json`

Active Task Found (Status: 200 OK):


```
{
    "task_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef",
    "parent_task_id": null,
    "task_type": "main_analysis",
    "sub_type_identifier": null,
    "status": "PROGRESS",
    "progress": 60,
    "details": {
        "message": "Processing albums: 6/10 completed.",
        "albums_completed": 6,
        "total_albums": 10,
        "status_message": "Processing albums: 6/10 completed.",
        "log": [
            "[2023-10-27 10:00:00] Main analysis task started.",
            "[2023-10-27 10:00:05] Found 10 albums to process.",
            "[2023-10-27 10:05:30] Processing albums: 6/10 completed."
        ]
    },
    "timestamp": "2023-10-27T10:05:30.123456"
}
```


Listing 14: Example Response: Get Active Tasks (Active Task Found)

No Active Task (Status: 200 OK):

```
{}
```

Listing 15: Example Response: Get Active Tasks (No Active Task)

4 Playlists Endpoint

4.1 Get Generated Playlists

Retrieves all automatically generated playlists stored in the database.

- **URL:** `/api/playlists`
- **Method:** `GET`
- **Description:** This endpoint fetches all playlists that were created by the clustering process and saved to the application's database. Playlists are grouped by their name.
- **Request Body:** None
- **Response:** `application/json`

```
{
    "Energetic_Medium_automatic": [
        {
            "item_id": "song1_id",
            "title": "Upbeat Track",
            "author": "Artist A"
        },
        {
            "item_id": "song2_id",
            "title": "Dancing Beat",
            "author": "Artist B"
        }
    ],
    "Relaxed_Slow_automatic": [
        {
            "item_id": "song3_id",
            "title": "Calm Evening",
            "author": "Artist C"
        }
    ]
}
```

Listing 16: Example Response: Get Generated Playlists

- The keys in the JSON object are the playlist names (e.g., "Energetic_Medium_automatic").
- Each value is an array of song objects, where each song object contains `item_id`, `title`, and `author`.