PREAPARED BY :
DevDynasty

API Documentation

Welcome to the MoodSphere API documentation. This document covers all the APIs that we used in building the application MoodSphere. The API provides endpoints for predicting moods based on images, recommending songs based on predicted moods, managing user playlists, and retrieving song information, retrieving songs based on artist and genre, deleting a song from the playlist, deleting the playlist, etc.

Base URL

https://msdev-cewl7upn6q-uc.a.run.app/

Endpoints

Save User

- **URL: '/save-user'**
- **Method: 'POST'**
- **Description:** Saves user data to the Firestore database. This API takes user_id, first_name and last_name as an input and saves the details to the firebase and we display that on the front-end.
- **Request Body:**
  Json
  {
  "user_id": "string",
  "first_name": "string",
  "last_name": "string"
  }
- **Response:**
  - **'200 OK':** User data saved successfully.
  - **'500 Internal Server Error'**: Error saving user data.

Predict Mood

- **URL: '/predict'**

- **Method: ‘POST’**
- **Description:** Predicts mood based on the provided image and recommends songs accordingly. This API takes an image as an input and sends it to the backend to predict the mood of the image and then outputs the predicted mood along with the song recommendation.
- **Request Body:**
    - Form data with an image file.
- **Response:**

Json


```
{
    “prediction”: “string”,
    “recommended_songs”: [
        {
            “YOUTUBE”: "string",
            “acousticness”: float,
            “album”: "string",
            “artist”: “string”,
            “danceability”: float,
            “energy”: float,
            “genre”: “string”,
            “id”: “string”,
            “instrumentalness”: float,
            “key”: int,
            “length”: int,
            “liveness”: float,
            “loudness”: float,
            “mood”: “string”,
            “name”: “string”,
            “popularity”: int,
            “release_date”: “string”,
            “speechiness”: float,
            “spotify”: “string”,
            “tempo”: float,
            “time_signature”: int,
            “valence”: float
        }
    ]
}
```

Search

- **URL: ‘/search’**
- **Method: ‘POST’**
- **Description:** This API searches for songs or artists in the Firestore database. For instance, if you search for “AC/DC”, it will fetch you all the top songs by the artist AC/DC.
- **Request Body:**
  Json
  ```
  {
      “query”: “string”

  }
  ```
- **Response:**
  Json
  ```
  {
      “matches”: [
          {
              “YOUTUBE”: "string",
              “acousticness”: float,
              “album”: "string",
              “artist”: “string”,
              “danceability”: float,
              “energy”: float,
              “genre”: “string”,
              “id”: “string”,
              “instrumentalness”: float,
              “key”: int,
              “length”: int,
              “liveness”: float,
              “loudness”: float,
              “mood”: “string”,
              “name”: “string”,
              “popularity”: int,
              “release_date”: “string”,
              “speechiness”: float,
              “spotify”: “string”,
              “tempo”: float,
              “time_signature”: int,
              “valence”: float

          }

      ]

  }
  ```

Songs by Artist

- **URL: '/songs-by-artist'**
- **Method: 'POST'**
- **Description:** This API retrieves songs by a specific artist from the Firestore database. For instance, if you search for "Adele", it will fetch you all the top songs sung by the artist Adele.
- **Request Body:**
  Json
  ```
  {
      "artist_name": "string"

  }
  ```
- **Response:**
  Json
  ```
  {
      "songs": [
          {
              "YOUTUBE": "string",
              "acousticness": float,
              "album": "string",
              "artist": "string",
              "danceability": float,
              "energy": float,
              "genre": "string",
              "id": "string",
              "instrumentalness": float,
              "key": int,
              "length": int,
              "liveness": float,
              "loudness": float,
              "mood": "string",
              "name": "string",
              "popularity": int,
              "release_date": "string",
              "speechiness": float,
              "spotify": "string",
              "tempo": float,
              "time_signature": int,
              "valence": float

          }
  ```

]

}

Songs by Genre

- **URL: '/songs-by-genre'**
- **Method: 'POST'**
- **Description:** This API retrieves songs by a specific genre from the Firestore database. For instance, if you enter "POP", it will fetch you all the songs based on genre "POP".
- **Request Body:**
  Json
  {
  "genre_name": "string"
  }
- **Response:**
  Json
  {
  "songs": [
  {
  "YOUTUBE": "string",
  "acousticness": float,
  "album": "string",
  "artist": "string",
  "danceability": float,
  "energy": float,
  "genre": "string",
  "id": "string",
  "instrumentalness": float,
  "key": int,
  "length": int,
  "liveness": float,
  "loudness": float,
  "mood": "string",
  "name": "string",
  "popularity": int,
  "release_date": "string",
  "speechiness": float,
  "spotify": "string",
  "tempo": float,
  "time_signature": int,
  "valence": float

```
                }
            ]
        }
```

Create Playlist

- **URL: '/create-playlist'**
- **Method: 'POST'**
- **Description:** Creates a new playlist for a user in the Firestore database. It takes user_id and playlist_name as an input and creates a playlist for that particular user with the playlist name they enter. This playlist gets saved in the firebase.
- **Request Body:**

```
Json
{
        "user_id": "string",
        "playlist_name": "string",
        "songs": [
                {
                        "title": "string",
                        "artist": "string",
                        "url": "string"

                }
        ]
}
```

- **Response:**
  - **'200 OK':** Playlist created successfully.
  - **'500 Internal Server Error':** Error creating playlist.

Get Playlist

- **URL: '/get-playlist'**
- **Method: 'GET'**
- **Description:** Retrieves playlists for a specific user from the Firestore database. Once the playlist has been created, this API retrieves the playlist that was created by the user.

The user can add songs to the playlist and when they view their playlist, they can see the songs that they added to the playlist.

- **Parameters:**
  - 'user_id': User ID (required)
- **Response:**
  - **'200 OK':** Playlists retrieved successfully.
  - **'404 Not Found':** No playlists found for the user.
  - **'500 Internal Server Error':** Error retrieving playlists.

Add Song to Playlist

- **URL: '/add-song-to-playlist'**
- **Method: 'POST'**
- **Description:** This API adds a song or a list of songs to a user's playlist in the Firestore database.
- **Request Body:**

```
Json
{
        "user_id": "string",
        "playlist_id": "string",
        "song_data": {
                "title": "string",
                "artist": "string",
                "url": "string"
        }
}
```

- **Response:**
  - **'200 OK':** Song added successfully.
  - **'400 Bad Request':** Song is already in the playlist.
  - **'404 Not Found':** Playlist not found.
  - **'500 Internal Server Error':** Error adding song.

Remove Song from Playlist

- **URL: '/remove-song-from-playlist'**

- **Method: ‘DELETE’**
- **Description:** This API when called removes a song from a user's playlist in the Firestore database. For instance, if the user has added the song “Someone like you by Adele” to their playlist “Favorite”. This API will remove that song from the playlist.
- **Parameters:**
  - **‘user_id’:** User ID (required)
  - **‘playlist_id’:** Playlist ID (required)
  - **‘key’:** Song key (required)
- **Response:**
  - **‘200 OK’:** Song removed successfully.
  - **‘404 Not Found’:** Playlist not found.
  - **‘500 Internal Server Error’:** Error removing song.

Delete Playlist

- **URL: ‘/delete-playlist’**
- **Method: ‘DELETE’**
- **Description:** This API deletes a user's playlist from the Firestore database. For instance, if a user has created a playlist called “My favs”, if the user wishes to delete that playlist and create a new playlist. This API will delete the selected playlist from the Firestore database. This API requires ‘user_id’ and ‘playlist_id’ as the input.
- **Parameters:**
  - ‘user_id’: User ID (required)
  - ‘playlist_id’: Playlist ID (required)
- **Response:**
  - **‘200 OK’:** Playlist deleted successfully.
  - **‘500 Internal Server Error’:** Error deleting playlist.

NOTES

- All responses are in JSON format.
- Error responses include an **‘error’** field describing the encountered error.
- Ensure proper authentication and authorization mechanisms are implemented before using these endpoints in a production environment.
- For detailed usage examples and request/response formats, please refer to the provided API documentation.