FINAL PROJECT REPORT

SEMESTER 1, ACADEMIC YEAR: 2024-2025

CT313H: WEB TECHNOLOGIES AND SERVICES

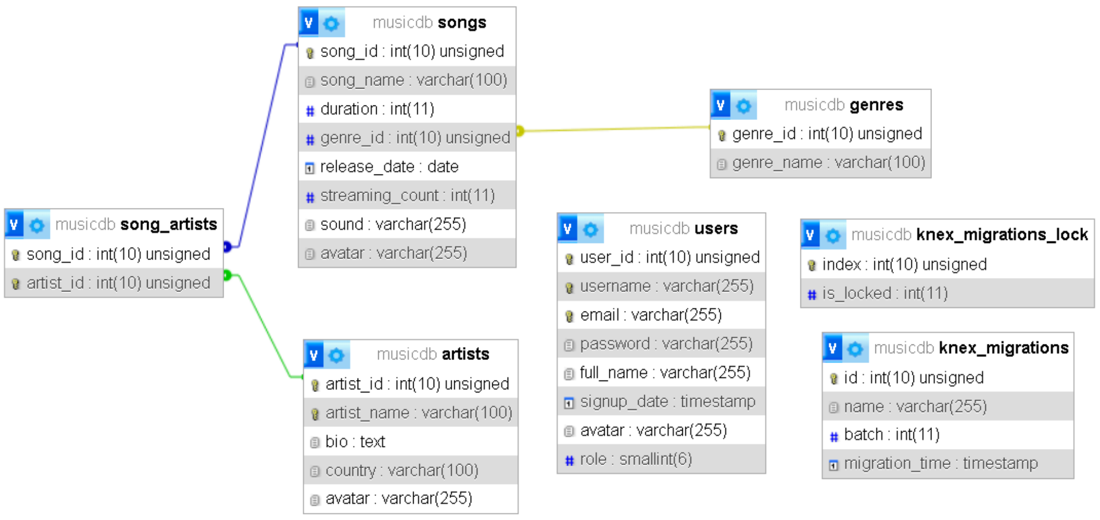
- **Project/Application name: Music app**
- **GitHub links (for both frontend and backend):** **https://github.com/24-25Sem1-Courses/ct313h03-project-LyKhai2708**
- **Youtube Link:** https://www.youtube.com/watch?v=Y9dexWWS_D0
- **Student ID 1: B2111930**
- **Student Name 1: Lý Phương Khải**
- **Student ID 2: B2111946**
- **Student Name 2: Trần Minh Quang**
- **Class/Group Number: CT313HM03**

I. Introduction

- Project/application description: A few words introducing an overview of the project/web application.

This music web application serves as a comprehensive platform for music enthusiasts, allowing users to discover, listen to, and manage their favorite songs and artists. The application features an extensive music library, enabling users to search for songs by title, artist, or genre. Users can create personalized playlists, explore trending tracks, and access detailed information about each song, including album details and artist biographies. With a user-friendly interface and responsive design, this application aims to enhance the music listening experience while fostering a vibrant community of music lovers.

- A list of tables and their structures in the database (could show a CDM/PDM diagram here).



- A task assignment sheet for each member if working in groups.

Team Member	Tasks
Lý Phương Khải	- add migration - User API - Add Song Genre API - Follow API - Songs_Artist API - Song API - Artist Update Admin - Artist Add Admin - Artist Delete Admin - Login - Register - Change Password - User Profile - User Update - User Add
Trần Minh Quang	- Setup express - Insert fake data - Create knex.js - Artist API - Albums API - Playlists API - Paginator

	- songs_playlist API - Song Update Admin - Song Add Admin - Song Delete Admin - Audio Player - Song Page - Artist Profile

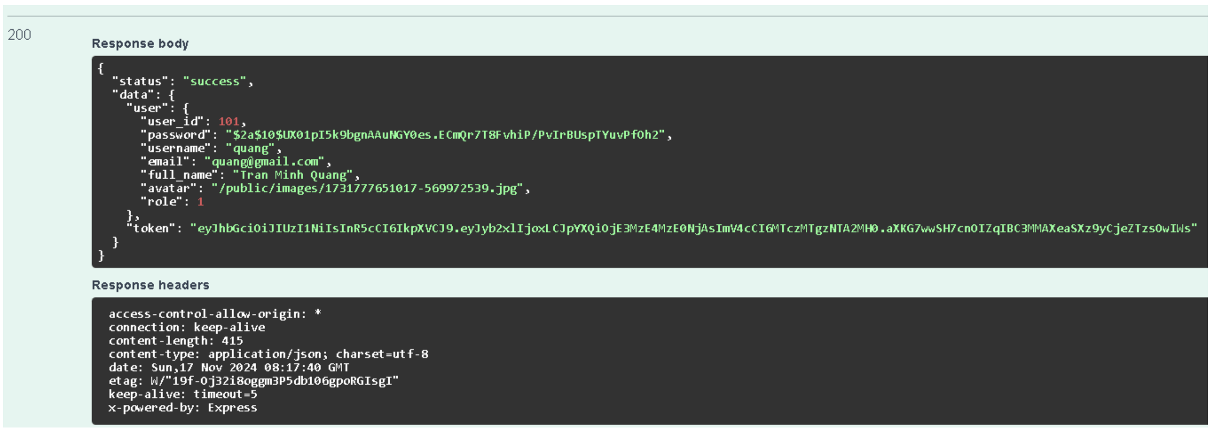
II. Details of implemented features

1. User login to get JWT token / Login

- Description:

The login functionality is used to authenticate users or admins to grant access to the website, operating based on an authentication process using email and password. When a user sends a login request, the back-end receives the information, checks the existence of the email in the database, and validates the password using bcrypt. If both are valid, the system generates a JWT token containing user information, role, and expiration time, then sends this token along with the user information to the client. This token is used to authenticate the user during API calls. If the user clicks on "Remember Me," the website will save the email and password for future logins.

- Screenshots:

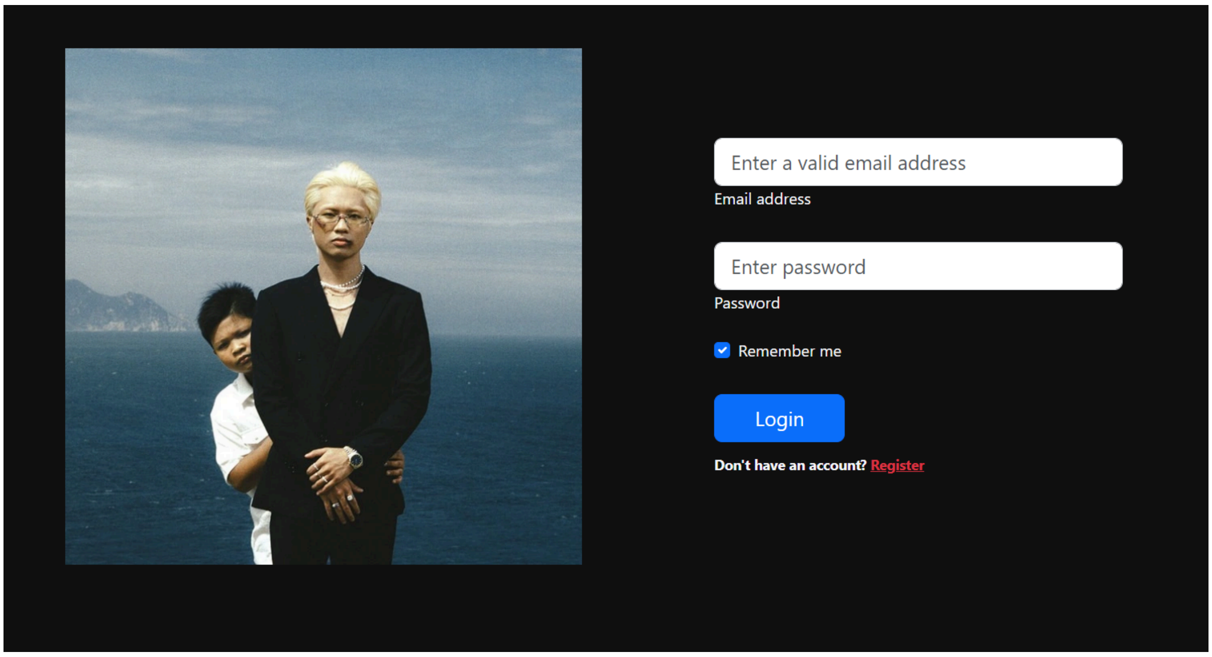

200

Response body

```
{
  "status": "success",
  "data": {
    "user": {
      "user_id": 101,
      "password": "$2a$10$UX01pI5k9bgnAAuNGY0es.ECmQr7T8FvhiP/PvIrBUspTYuvPf0h2",
      "username": "quang",
      "email": "quang@gmail.com",
      "full_name": "Tran Minh Quang",
      "avatar": "/public/images/1731777651017-569972539.jpg",
      "role": 1
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoxLCJpYXQiOjE3MzE4MzE0NjAsImV4cCI6MTczMTgzNTA2MH0.aXKG7wwSH7cnOIZqIBC3MMAXeaSXz9yCjeZTzsOwIWs"
  }
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 415
content-type: application/json; charset=utf-8
date: Sun,17 Nov 2024 08:17:40 GMT
etag: W/"19f-Oj32i8oggm3P5db106gpoRGIsgI"
keep-alive: timeout=5
x-powered-by: Express
```



- Implementation details:

● Describe the API:

○ Endpoint: http://localhost:3000/api/v1/login/

○ Method: POST

● This feature create a new user to the users table.

● The client-side states required to implement the login functionality is:

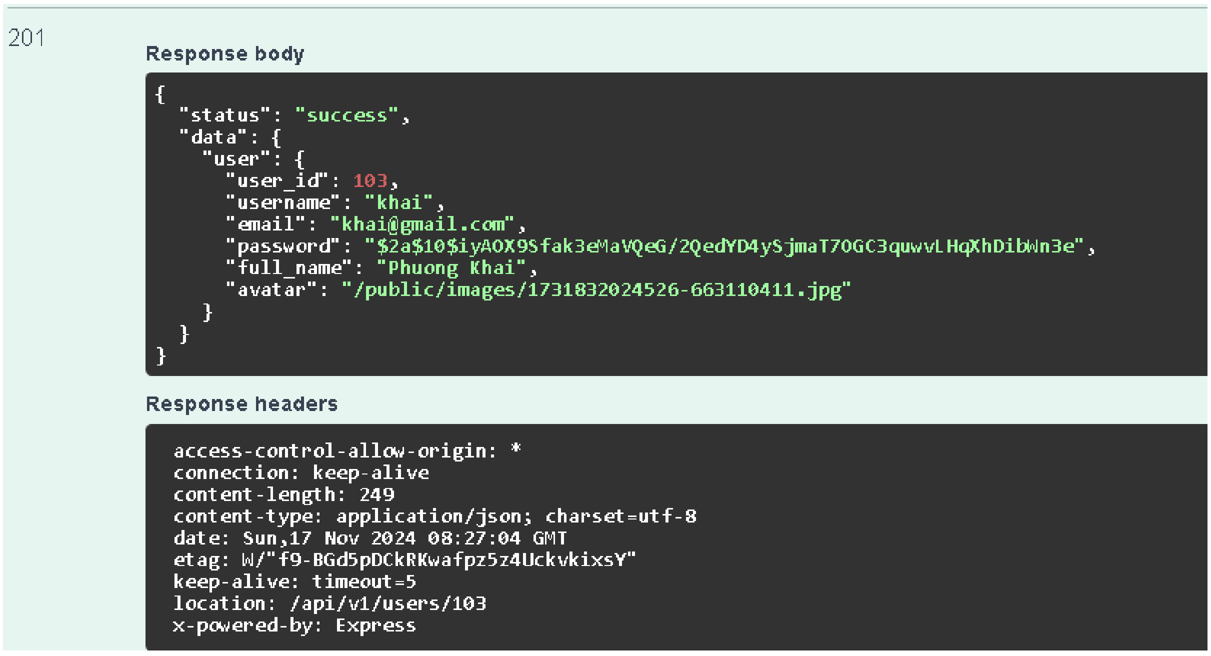
1. email: Stores the email entered by the user.
2. password: Stores the password entered by the user.
3. formErrors: An object to track validation errors for the email and password fields, as well as general errors during login.
4. message: A state for displaying a feedback message (success or error).
5. alertType: Indicates the type of alert (e.g., success or error) for styling the feedback message.
6. rememberMe: Tracks whether the "Remember Me" checkbox is selected.

2. Create new user / Register

- Description:

This Register feature allows users to create a new account on the web application. Users are required to enter information such as a username, email, password, and full name. The system will validate the input data, ensuring that the username, email, and password meet security requirements. If there are any errors during the data entry, the system will display corresponding error messages. Once the user clicks submit, a request will be sent to the API, and the system will verify and create the account for the user. After that, the website will display a success or failure notification to inform the user, and during this process, the submit button will be automatically disabled while the request is being processed.

- Screenshots:



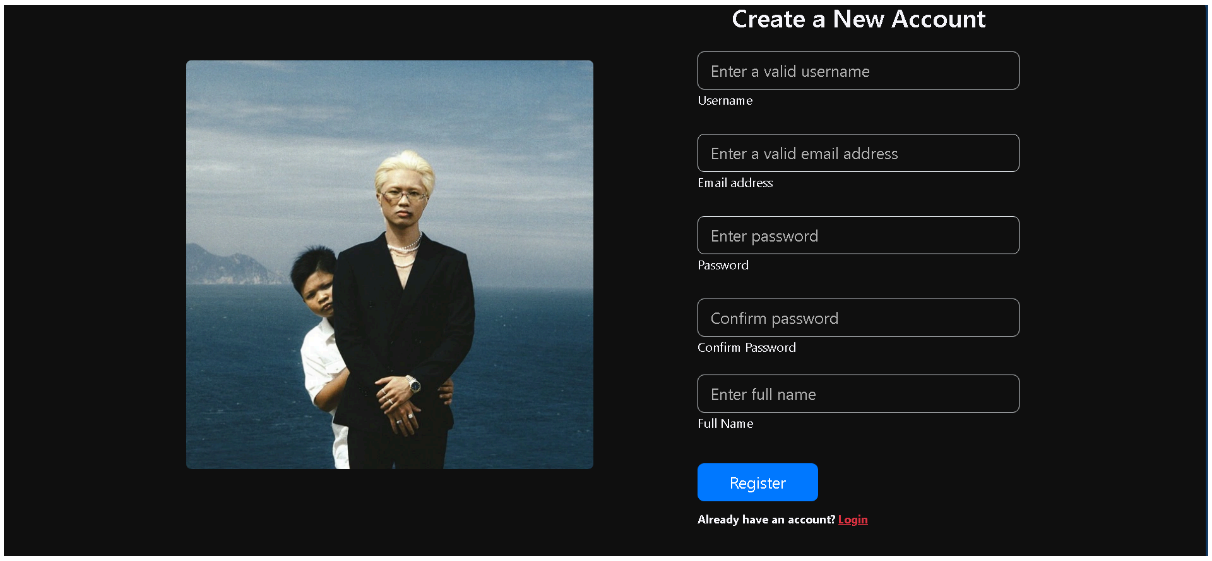
201

Response body

```
{
  "status": "success",
  "data": {
    "user": {
      "user_id": 103,
      "username": "khai",
      "email": "khai@gmail.com",
      "password": "$2a$10$iyAOX9Sfak3eMaVQeG/2QedYD4ySjmaT7OGC3quwvLHqXhDibWn3e",
      "full_name": "Phuong Khai",
      "avatar": "/public/images/1731832024526-663110411.jpg"
    }
  }
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 249
content-type: application/json; charset=utf-8
date: Sun,17 Nov 2024 08:27:04 GMT
etag: W/"f9-BGd5pDCkRKwafpz5z4UckvkixsY"
keep-alive: timeout=5
location: /api/v1/users/103
x-powered-by: Express
```



- Implementation details:

● Describe the API:

○ Endpoint: http://localhost:3000/api/v1/users/

○ Method: POST

○ The response is a JSON object containing status and detailed information about

created users.

○ The request body is multipart/formdata type and should include fields for the username, email, password, confirm password, fullname, and an optional avatar.

● The **Create a New User** API allows users to register by providing essential details such as their username, email, password, confirm password, full name, and an optional avatar. This endpoint enables new users to be added to the system.

● The client-side states required to implement the register functionality is:

1. FormValues:

- username: Stores the value of the username field.
- email: Stores the value of the email field.
- password: Stores the value of the password field.
- confirmpass: Stores the value of the confirm password field.
- full_name: Stores the value of the full name field

2. message: A state for displaying a feedback message (success or error).

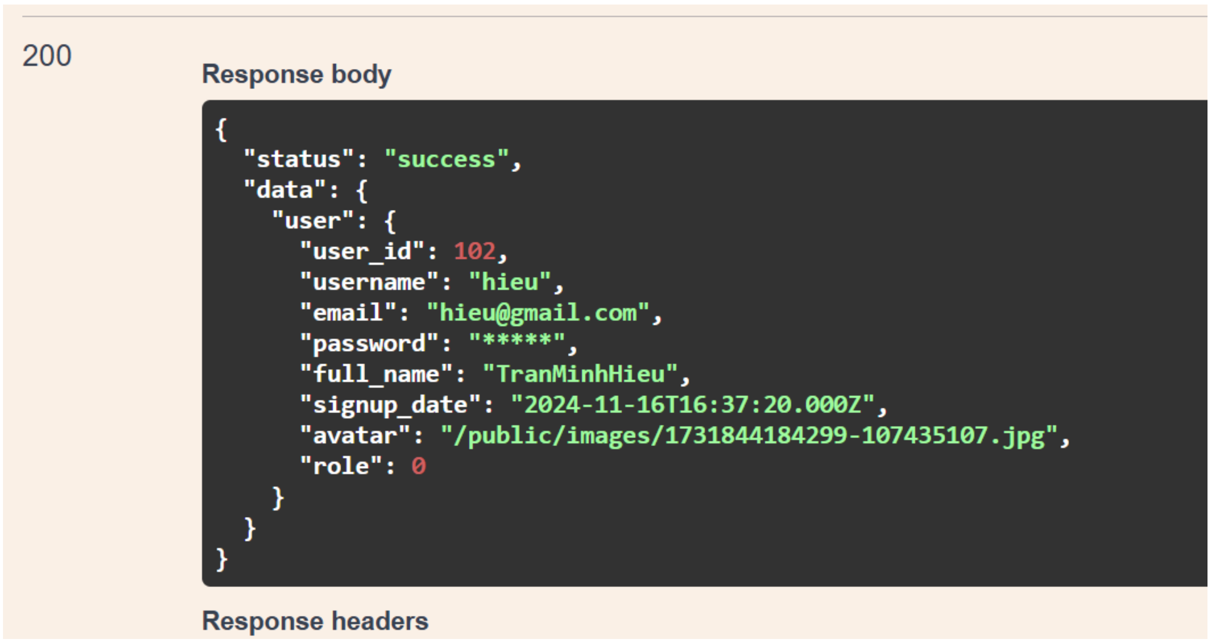
3. alertType: Indicates the type of alert (e.g., success or error) for styling the feedback message.

4. isSubmitting: A boolean that tracks whether the form is currently being submitted to prevents multiple submissions by disabling the submit button

3. Change user's password by ID/Change Password

-Description: The Change Password feature allows users to update their personal passwords by entering the current password, a new password (meeting security requirements), and confirming the new password. The system validates the input fields, and if valid, it calls the API, hashes the new password, updates it in the database, and notifies the user.

-Screenshots:

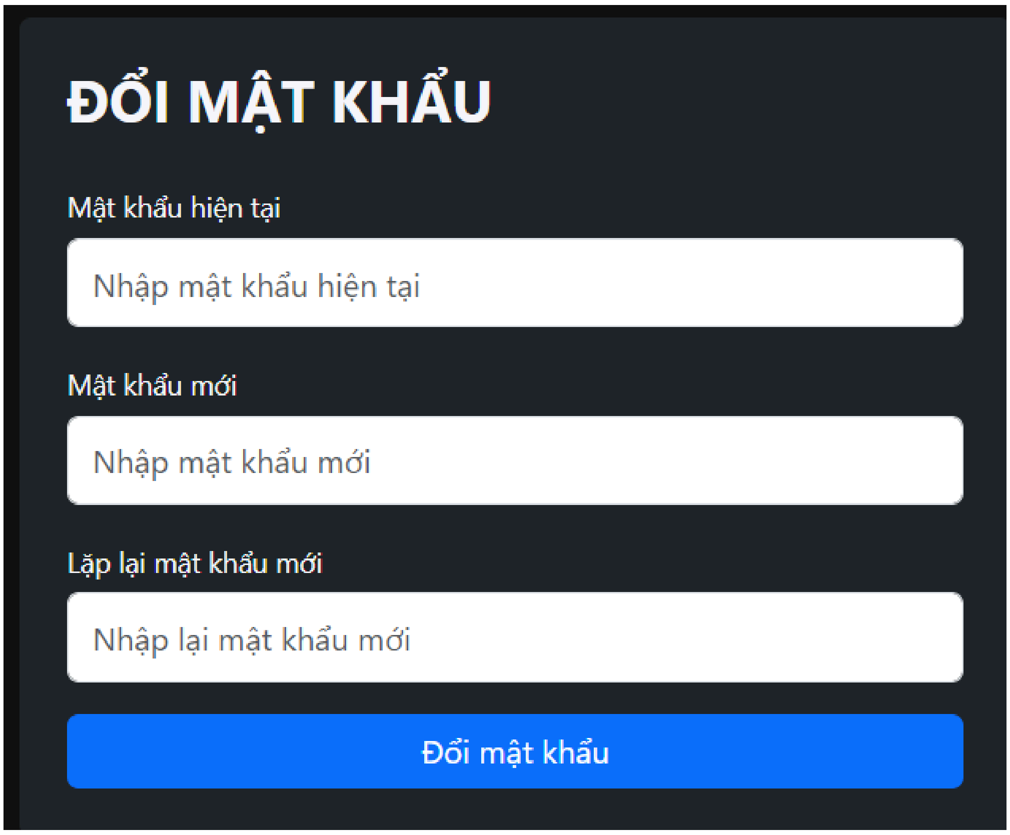


200

Response body

```
{
  "status": "success",
  "data": {
    "user": {
      "user_id": 102,
      "username": "hieu",
      "email": "hieu@gmail.com",
      "password": "*****",
      "full_name": "TranMinhHieu",
      "signup_date": "2024-11-16T16:37:20.000Z",
      "avatar": "/public/images/1731844184299-107435107.jpg",
      "role": 0
    }
  }
}
```

Response headers



ĐỔI MẬT KHẨU

Mật khẩu hiện tại

Nhập mật khẩu hiện tại

Mật khẩu mới

Nhập mật khẩu mới

Lặp lại mật khẩu mới

Nhập lại mật khẩu mới

- Implement detail:

- Describe the API:
  - Endpoint: http://localhost:3000/api/v1/users/{id}/password
  - Method: PUT
  - The data format send/received is : application/json
- This feature read the email and password in the users table

- The client-side states required to implement the Change Password functionality is:
  1. currentPassword: Stores the current password input.

  2. newPassword: Stores the new password input.

  3. confirmPassword: Stores the confirmation of the new password input.

  4. formErrors: An object to store validation error messages for each field

  5. message: A state for displaying a feedback message (success or error).

  6. alertType: Indicates the type of alert (e.g., success or error) for styling the feedback message.

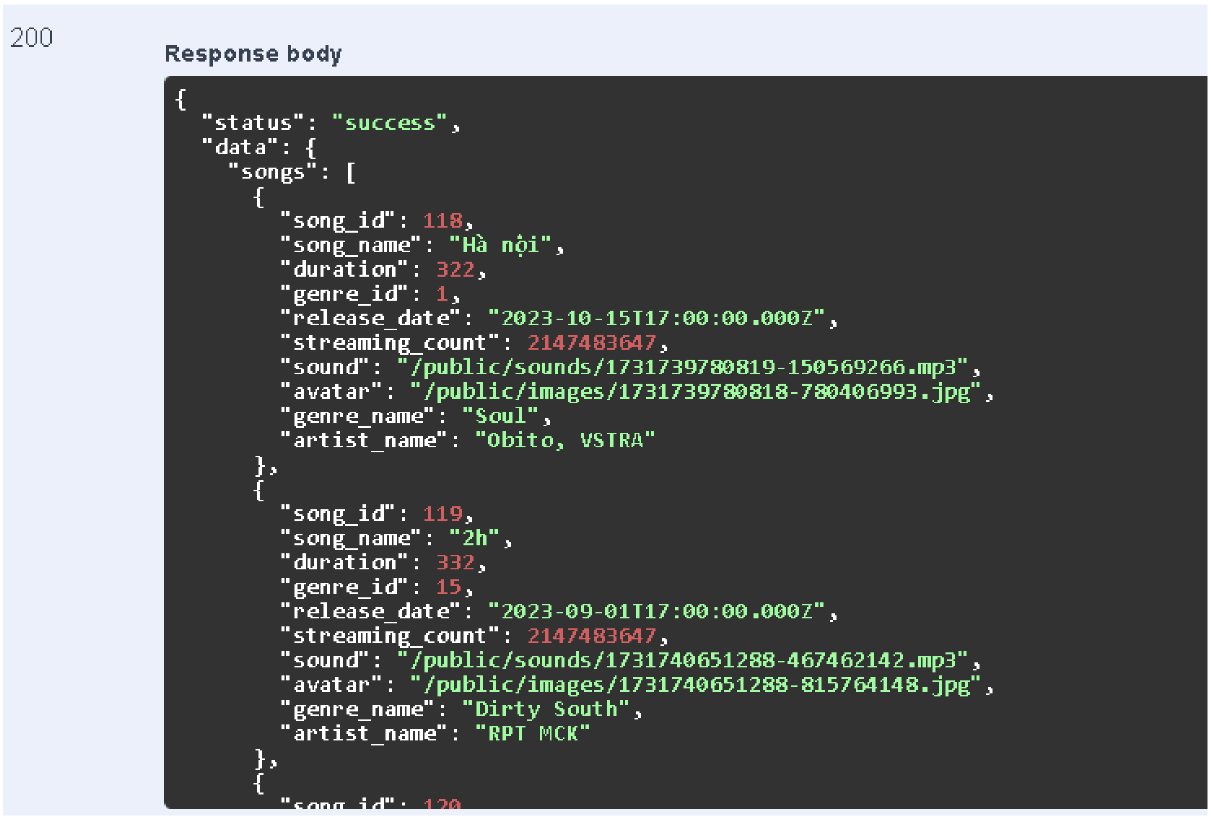
4. Get all songs / Audio Player

- Description:

The **Get All Songs** feature enables users to fetch a comprehensive list of songs from the music database. Users can optionally filter results by song name, genre ID, or both, or retrieve all songs without filters. The system validates the input criteria, and upon confirmation, it calls the API to retrieve the song list. Each song entry includes details such as the song ID, name, duration, album ID, genre ID, release date, and streaming count. The filtered or complete song list is displayed to the user.

The **Audio Player** feature allows users to play songs by interacting with the music database. When a user selects a song, the system calls the API to load the song data, which includes the song ID, name, artist name, cover image, and audio file URL. The application updates the selected song and plays the audio using the Audio object. Users can control playback with features like play/pause, track switching, and volume adjustment. The progress bar and timestamps are synchronized in real-time by listening to the timeupdate event. The system ensures a seamless experience by managing track navigation, volume control, and displaying current song information.

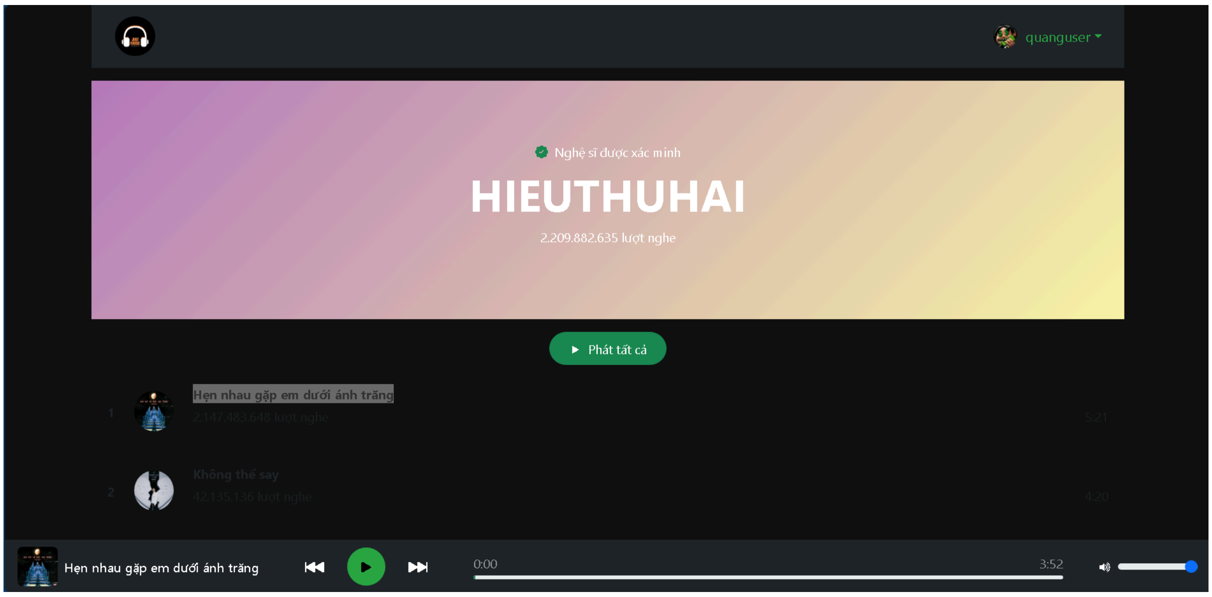
- Screenshots:



200

Response body

```
{
  "status": "success",
  "data": {
    "songs": [
      {
        "song_id": 118,
        "song_name": "Hà nội",
        "duration": 322,
        "genre_id": 1,
        "release_date": "2023-10-15T17:00:00.000Z",
        "streaming_count": 2147483647,
        "sound": "/public/sounds/1731739780819-150569266.mp3",
        "avatar": "/public/images/1731739780818-780406993.jpg",
        "genre_name": "Soul",
        "artist_name": "Obito, VSTRA"
      },
      {
        "song_id": 119,
        "song_name": "2h",
        "duration": 332,
        "genre_id": 15,
        "release_date": "2023-09-01T17:00:00.000Z",
        "streaming_count": 2147483647,
        "sound": "/public/sounds/1731740651288-467462142.mp3",
        "avatar": "/public/images/1731740651288-815764148.jpg",
        "genre_name": "Dirty South",
        "artist_name": "RPT MCK"
      },
      {
        "song_id": 120
```



- Implementation details:

- Describe the API:
    - Endpoint: http://localhost:3000/api/v1/songs/
    - Method: GET

○ The response is a JSON object containing detailed information about all songs in the database.

● This feature retrieves all songs from the songs table with or without any filters applied.

The client-side states required to implement the audio player feature are:

1. **currentTime**: Stores the current time of the song.
2. **duration**: Stores the total duration of the song.
3. **volume**: Holds the volume level of the song.
4. **audioPlaying**: Tracks whether the song is playing or paused.
5. **selectedSong**: Holds the data of the currently playing song.
6. **audio**: Reference to the `Audio` object.
7. **selectedIndex**: Tracks the index of the currently selected song.

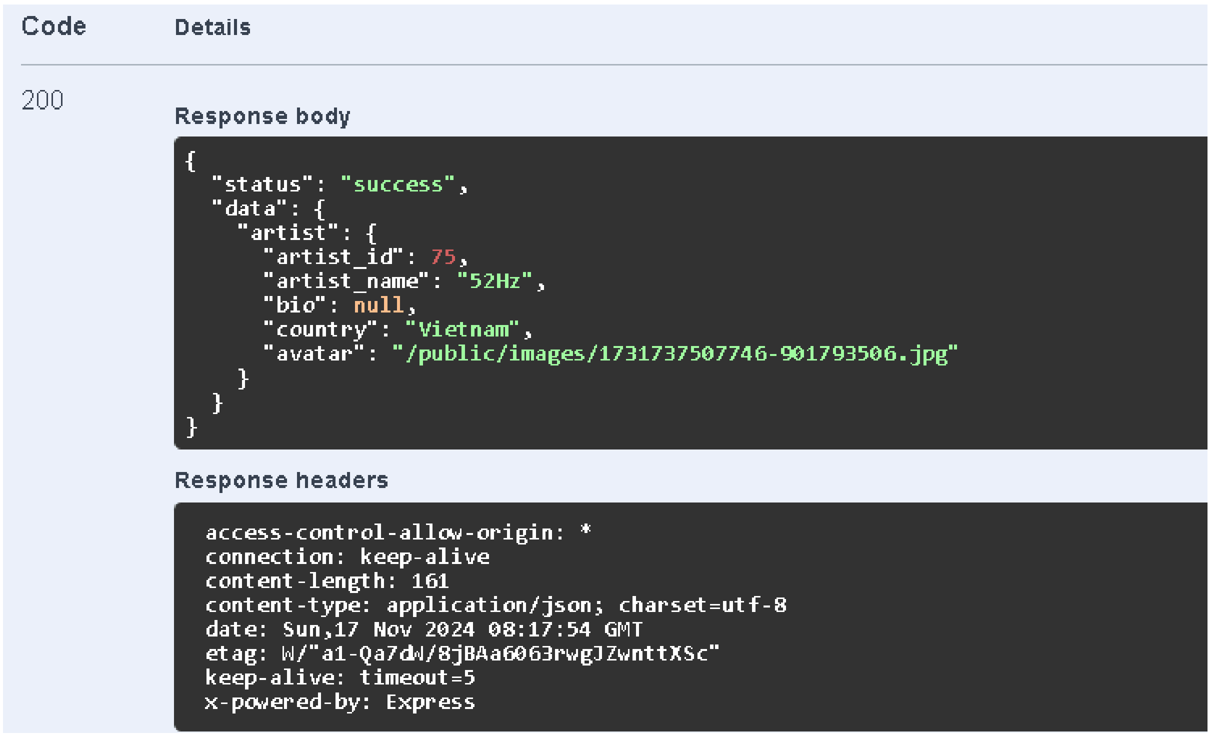
5. Get artist by ID / Artist profile

- Description:

The **Get Artist by ID** feature allows users to retrieve detailed information about a specific artist from the music database using the artist's unique ID. This functionality provides essential details such as the artist ID, name, biography, and country, enabling users to access accurate and personalized information about the artist.

The **Artist Profile** feature displays an artist's detailed profile, including their name, biography, background image, and music statistics such as monthly listeners. Users can browse through the artist's song list, play individual songs, or queue all songs for playback. The profile page also includes an "About" section that highlights the artist's background and bio, providing a rich and engaging experience for fans exploring their favorite artists.

- Screenshots:

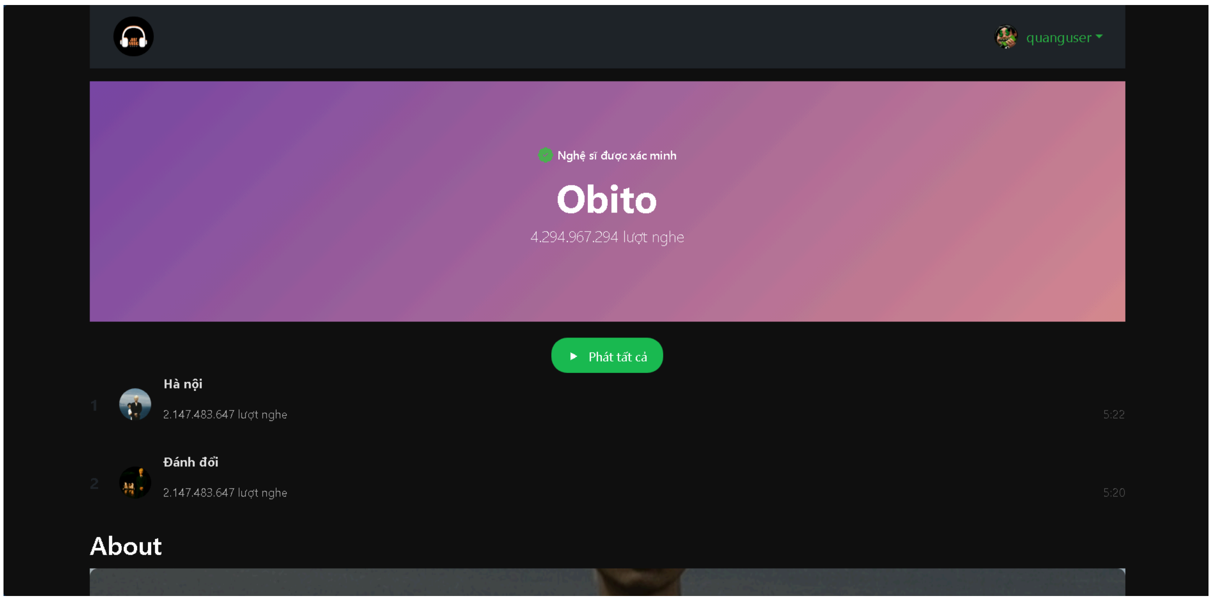


Code	Details
200	**Response body**

```
{
  "status": "success",
  "data": {
    "artist": {
      "artist_id": 75,
      "artist_name": "52Hz",
      "bio": null,
      "country": "Vietnam",
      "avatar": "/public/images/1731737507746-901793506.jpg"
    }
  }
}
```

Response headers

```
 access-control-allow-origin: *
 connection: keep-alive
 content-length: 161
 content-type: application/json; charset=utf-8
 date: Sun,17 Nov 2024 08:17:54 GMT
 etag: W/"a1-Qa7dW/8jBAa6063rwgJZwnttXSc"
 keep-alive: timeout=5
 x-powered-by: Express
```



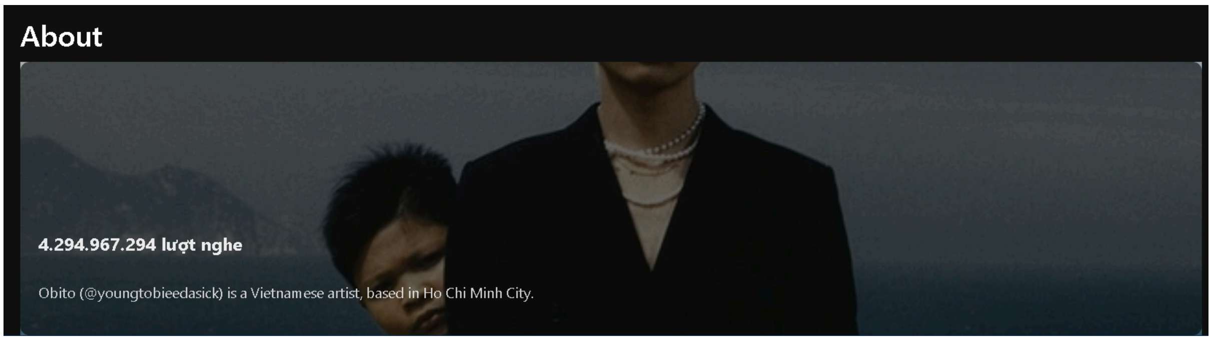
Obito

4.294.967.294 lượt nghe

1 Hà nội 2.147.483.647 lượt nghe 5:22

2 Đánh đổi 2.147.483.647 lượt nghe 5:20

About



About

4.294.967.294 lượt nghe

Obito (@youngtobieedasick) is a Vietnamese artist, based in Ho Chi Minh City.

- Implementation details:

● Describe the API:

○ Endpoint: http://localhost:3000/api/v1/artists/{id}

○ Method: GET

○ The response is a JSON object containing the details of the specified artist, including their artist ID, artist name, bio, and country.

● This feature retrieves information from the artists table, specifically for the artist identified by the provided artist ID.

To implement the Artist profile feature, the following client-side states are needed:

1. **artistId**: The ID of the artist, passed as a prop to the component.
2. **selectedSong**: Stores the current song being played.
3. **backgroundStyle**: Holds the background style for the artist profile container.
4. **songs**: Stores the list of songs associated with the artist.
5. **artist**: Contains the artist's profile information (e.g., name, bio).
6. **audio**: Refers to the audio player object for controlling playback.
7. **audioPlaying**: Tracks whether the audio is currently playing or paused.
8. **backgroundImage**: Holds the background image for the artist's profile, which may be based on a song's avatar.

.