

HTX Audio Application Documentation

1. Overview

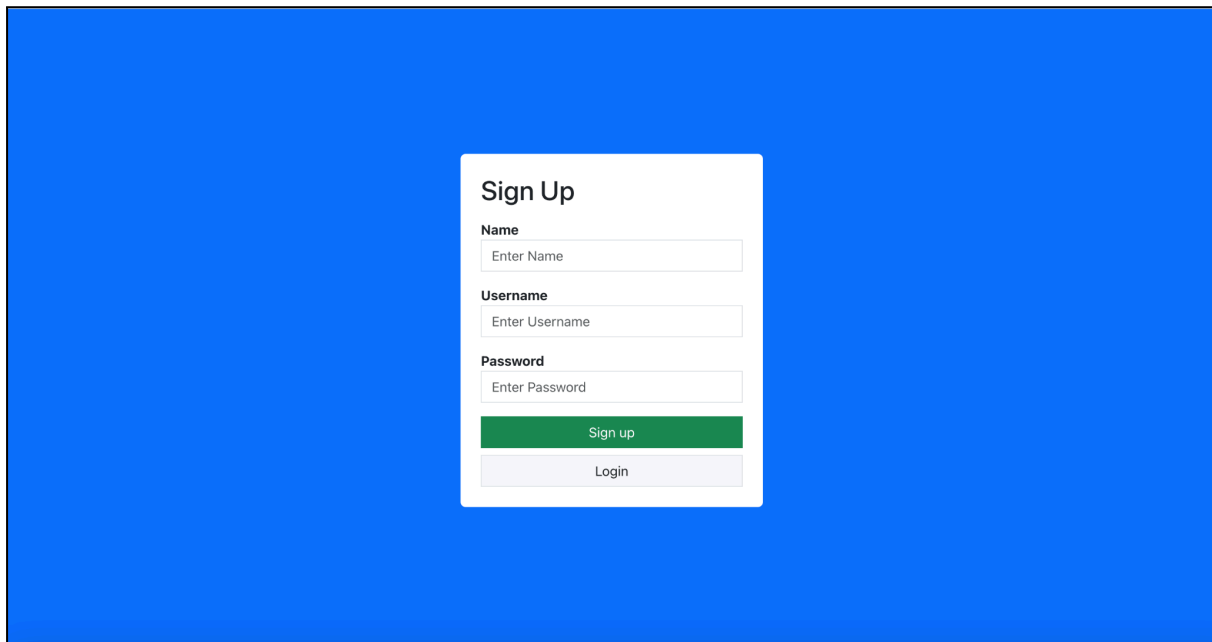
This documentation provides an overview of the application's capabilities, the technical stack that was utilised and the API references on the server-side of the application. The instructions on how to run the application can be found in a separate file attached in the email - "Instructions to Run The Application.pdf"

2. Capabilities of the Application

This section contains illustrative screenshots of all pages in the application and details the functionalities of the application, spanning from the stipulated features of the assessment to the additional features that value-add to the user experience (*highlighted in green italic font*) of the application and the security/validation mechanisms that were being implemented (*highlighted in blue italic font*).

2.1 Sign Up Page

Users can sign up for an account by entering their name, username and password. The username entered by all users should be unique. A success message will be registered in the window when the sign up is successful and users can proceed to login to the application

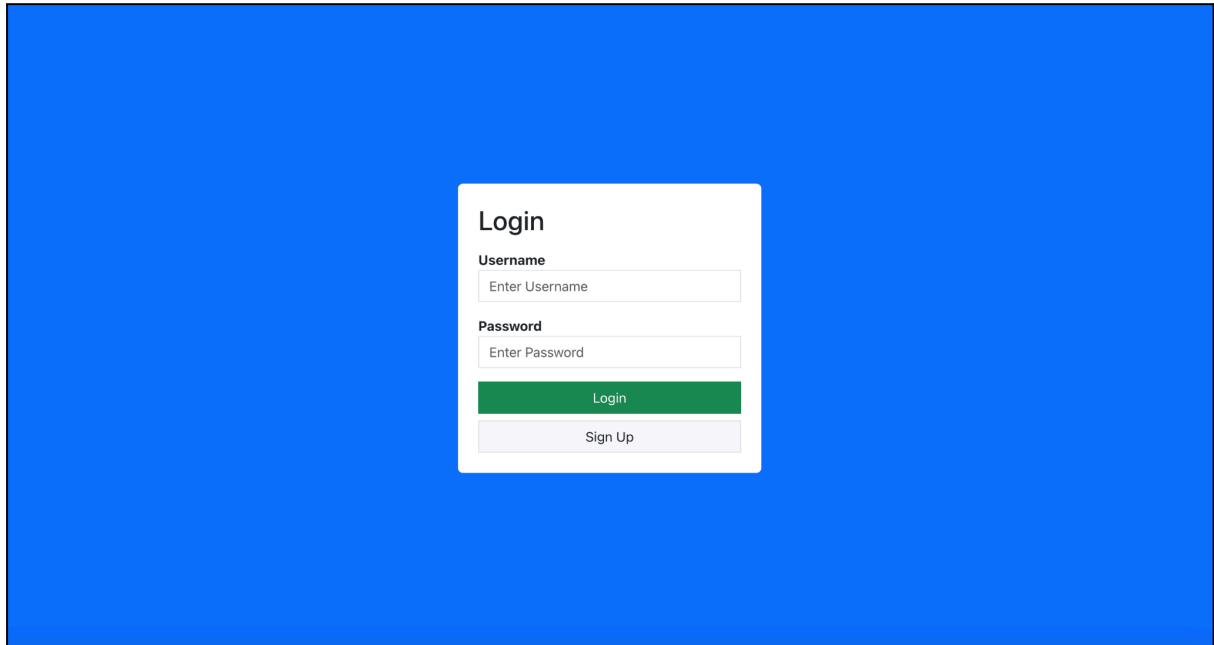
A screenshot of the 'Sign Up' page. The page has a solid blue background. In the center, there is a white rectangular form. The form is titled 'Sign Up' in bold black text. Below the title, there are three input fields: 'Name' with the placeholder text 'Enter Name', 'Username' with the placeholder text 'Enter Username', and 'Password' with the placeholder text 'Enter Password'. Below these fields are two buttons: a green button labeled 'Sign up' and a light gray button labeled 'Login'.

An error message will be triggered when:

- *The user leaves either one of the name, username and password input fields blank*
- *The user chooses a username that has been previously chosen by an existing user*

2.2 Login Page

Users can login to the application with their username and password. A success message will be registered in the window when the login is successful and users will be redirected to the Media page (also the Home page)



The image shows a login page with a solid blue background. Centered on the page is a white rectangular form titled "Login". Inside the form, there are two input fields. The first is labeled "Username" and contains the placeholder text "Enter Username". The second is labeled "Password" and contains the placeholder text "Enter Password". Below the password field, there are two buttons: a green button labeled "Login" and a light gray button labeled "Sign Up".

An error message will be triggered when:

- *The user leaves either one of the username and password input fields blank*
- *The user's username is accurate but the password does not match*
- *The username entered by the user does not exist*

2.3 Media Page (Also the Home Page)

Users can upload their audio files here with an accompanying audio category (Category A/B/C/D) and description. They can also view and manage their previous uploads. For users who have yet to upload any audio, a message “No audio files have been uploaded yet” will be displayed under the “Previous Uploads” section

The screenshot shows the HTX Audio Portal interface. The top navigation bar includes 'HTX Audio Portal', 'Media', and 'Settings'. The user is signed in as 'janelee' with a 'Logout' button. The main content area is divided into two sections. The first section, 'Upload Audio File', contains a form with three input fields: 'Audio Category' (a dropdown menu with 'Select a category' selected), 'Audio Description' (a text area with the placeholder 'Enter a description of audio file here'), and 'File Input' (a button labeled 'Choose file' and a text field showing 'No file chosen'). A blue 'Submit' button is located to the right of the 'File Input' field. The second section, 'Previous Uploads', displays the message 'No audio files have been uploaded yet'.

Otherwise, users who have successfully uploaded audios will have their uploads presented in a tabular format under the “Previous Uploads” section. They can then manage their previous uploads by viewing or deleting them

The screenshot shows the HTX Audio Portal interface. The top navigation bar includes 'HTX Audio Portal', 'Media', and 'Settings'. The user is signed in as 'janelee' with a 'Logout' button. The main content area is divided into two sections. The first section, 'Upload Audio File', contains a form with three input fields: 'Audio Category' (a dropdown menu with 'Select a category' selected), 'Audio Description' (a text area with the placeholder 'Enter a description of audio file here'), and 'File Input' (a button labeled 'Choose file' and a text field showing 'No file chosen'). A blue 'Submit' button is located to the right of the 'File Input' field. The second section, 'Previous Uploads', displays a table of uploads.

#	Filename	Playback	Created at	Audio Category	Audio Description	Delete
1	SampleVideo_1280x720_30mb.mp4	▶ 0:00 / 2:50 ————— 🔊 ⋮	4 Apr 2025, 3:21pm	A	My first audio upload	Delete

An error message will be triggered when:

- The user leaves either one of the audio category, audio description or file input fields blank
- The user uploads a file with an incompatible format (only audio files of type .m4a, .flac, .mp3, .mp4, .wav, .wma, .aac are supported)

Value-added features:

- *Only users who are signed-in can access this page due to the use of protected routes*
- *Users can see their file upload progress as a % below the file upload component*
- *Users can delete their previously uploaded audios*
- *Users can download their previously uploaded audios*
- *The previous uploads are sorted in descending order (most to least recent) based on their time of creation, allowing for easy identification of files sharing the same name*

2.4 Account Settings Page

There are 2 tabs on this page, “Update Account” and “Delete Account”. Users may modify their name, username and password under the “Update Account” tab

HTX Audio Portal Media Settings

Signed in as: [janelee](#) [Logout](#)

Account Settings

[Update Account](#) [Delete Account](#)

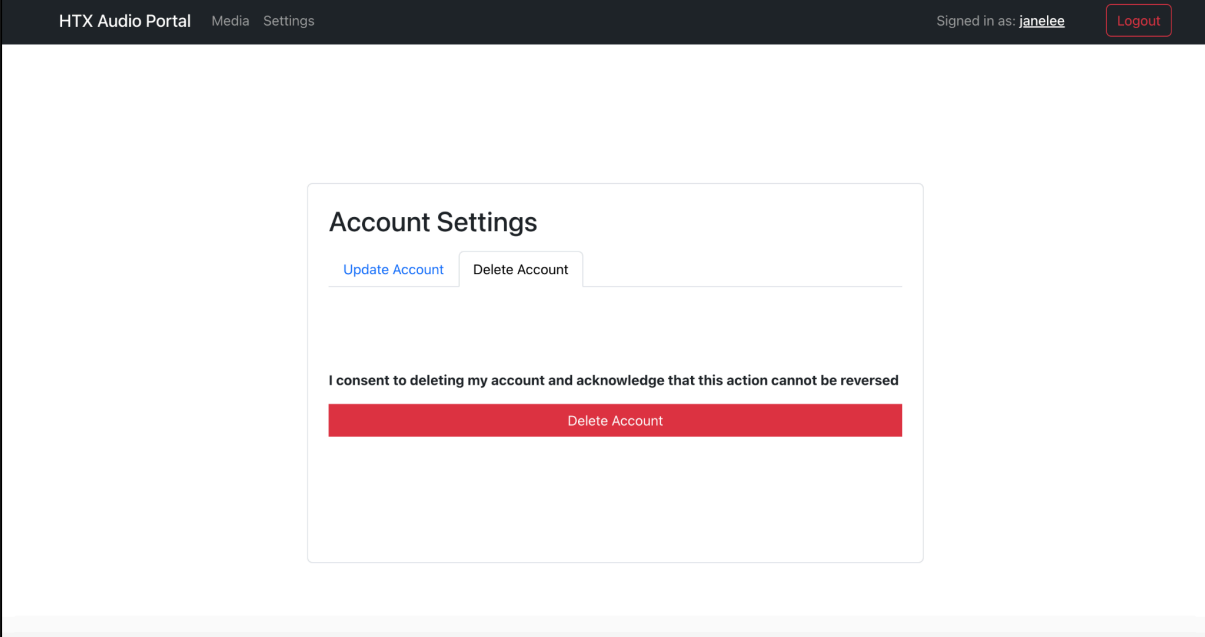
New Name
Current Name: Jane

New Username
Current Username: janelee

New Password
Enter Password

[Update](#)

Meanwhile, users may delete their account under the “Delete Account” tab. This action is irreversible and users will be shown an error message to log in immediately after doing so



The screenshot shows a web application interface. At the top, a dark navigation bar contains the text "HTX Audio Portal", "Media", and "Settings" on the left, and "Signed in as: janelee" and a "Logout" button on the right. The main content area is white and contains a centered box titled "Account Settings". Inside this box, there are two tabs: "Update Account" (highlighted in blue) and "Delete Account" (highlighted in grey). Below the tabs, there is a text input field with the placeholder text "I consent to deleting my account and acknowledge that this action cannot be reversed". Below this field is a prominent red button labeled "Delete Account".

An error message will be triggered when:

- *The user clicks the “Update” button without entering information into the name, username or password fields*
- *The user chooses to update his username with a username that is held by an existing user*

Value-added features:

- *Only users who are signed-in can access this page due to the use of protected routes*
- *Users can see their current name and username as a placeholder in the name and username text input fields*

2.5 Navigation Bar

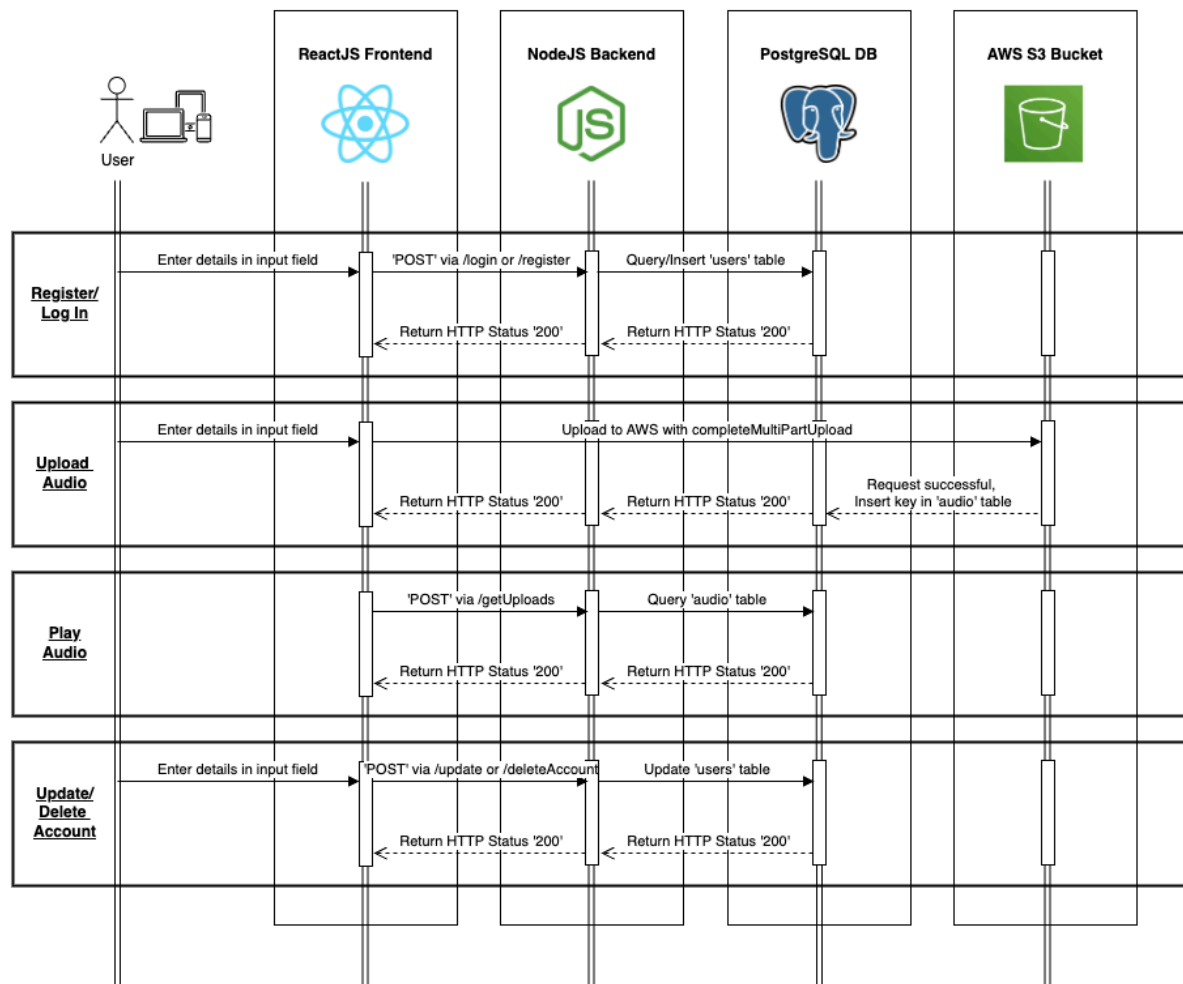
Users can access tabs in the application and log out seamlessly with the navigation bar at the top of the page. Clicking the underlined username leads to the Settings page while clicking the “HTX Audio Portal” label leads to the Media page (also the Home page)



This screenshot shows a close-up of the dark navigation bar. On the left, it displays "HTX Audio Portal", "Media", and "Settings". On the right, it shows "Signed in as: janelee" and a "Logout" button.

3. Technical Stack Utilised and Considerations During Development

The following technologies were used in the implementation of the full-stack audio web application (see system architecture diagram below). Full separation between the application and frontend was achieved using the RESTful API.



3.1 Frontend

ReactJS was utilised as the front-end framework for its speed, flexibility and extensive support with modern development libraries. React-bootstrap was chosen as the component library and was pivotal to the styling of the application. HTTP requests were sent to the backend to post and retrieve information that were essential to the functioning of the application.

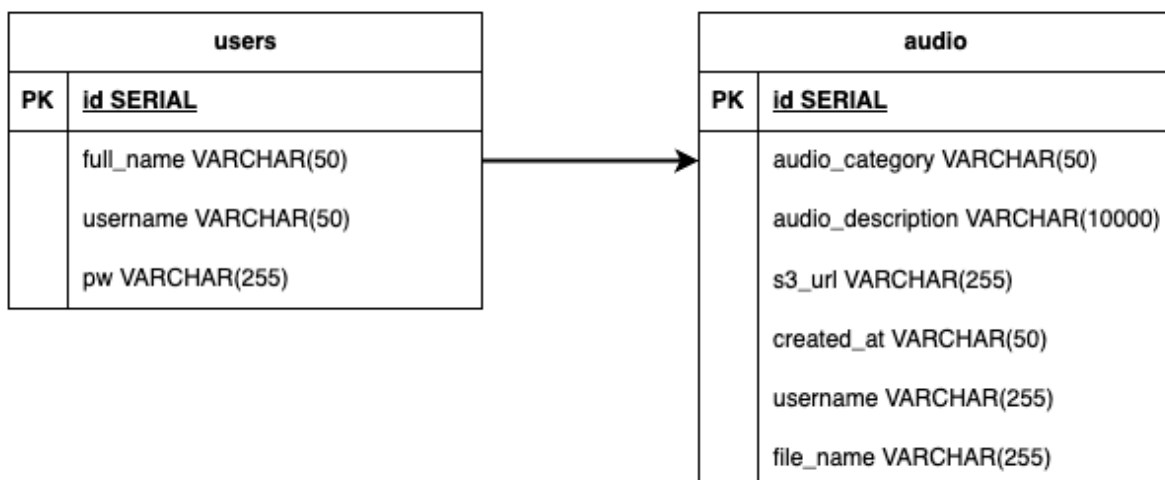
3.2 Backend

NodeJS (Express) was being utilised to develop a RESTful API. Client-server interactions are optimised using REST, and the client-server separation promotes a flexible approach to scaling the application, or integrating the application with additional features such as a Large Language Model (LLM) to process users' audio uploads.

For security purposes, user passwords were hashed using bcrypt prior to being stored in the PostgreSQL database. bcrypt is a computationally expensive algorithm which safeguards user accounts against resistant brute-force attacks. A random string was generated using a cost factor of 10 before the password was being hashed, thus even users who have the same password have different hashed values. Furthermore, to ensure that users will not be able to playback audio files from other users, nanoid was used to generate a random string of characters that served as the audio key in the S3 bucket, and was stored in the PostgreSQL database. Throughout the application, protected routes were enforced through cookies in the user's browser. Each time a user logs in to the application, a cookie with an expiry date of 1 day will be generated as proof of his identity and deleted upon logging out, preventing undesirable access to the user's account information.

3.3 Database

A PostgreSQL database (users_auth) was utilised, with 2 tables being created to store user login credentials (users table) and audio upload information (audio table) respectively. The database diagram is as follows



3.4 Mounted Storage

AWS S3 was utilised to store the audio files that were being successfully uploaded. With its virtually unlimited storage capacity, AWS is vital to the scaling of the application as an increased number of registered users translates to higher demand for storage to accommodate users' file uploads

For increased processing speed in uploading large audio files, AWS' multipart upload functionality was implemented.

3.5 Deployment

Docker was being used to containerise the frontend and backend of the application, which was organised into 2 separate folders for this use case. By dockerising the application, hardware and package conflicts across different devices can be eliminated, and the application can be swiftly deployed on cloud platforms for widespread use

4. API References

The API references have been segmented by page and are as follows

4.1 Sign Up Page

The API calls executed in the Sign Up page are listed below

4.1.1 POST <http://localhost:3000/register>

- Purpose
 - Parses the name, username and password entered by the user and registers an account for the user if all 3 fields are present with valid inputs. Otherwise, an error is thrown.
- Request body parameters
 - Object { name: "jane", username: "janelee", password: "abcd123" }
 - name: A string, represents the user's account name
 - username: A string, represents the user's account username
 - password: A string, represents the user's account password
- Response object
 - Object
 - message: A string about the registration status of the user e.g. "User registered successfully", "Internal server error" etc.

4.2 Login Page

The API calls executed in the Login page are listed below

4.2.1 POST <http://localhost:3000/login>

- Purpose
 - Parses the username and password entered by the user. The user is logged in if all 2 fields are present with the correct inputs and a cookie containing the user's credentials is added to the browser. Otherwise, an error is thrown.
- Request body parameters
 - Object { username: "janelee", password: "abcd123" }
 - username: A string, represents the user's account username
 - password: A string, represents the user's account password
- Response object
 - Object
 - message: A string about the login status of the user e.g. "User logged in successfully", "Invalid password" etc.

4.3 Media Page (Also the Home Page)

The API calls executed in the Media page are listed below

4.3.1 GET <http://localhost:3000>

- Purpose
 - Verifies whether the user is logged into the application by reading the cookie containing the user's credentials in the browser. Returns the user's name and username if the user is logged in.

- Request body parameters
 - Not applicable for a GET request
- Response object
 - Object { `status: "Success", name: "Jane", username: "janelee" }`
 - `status`: A string about the status of verifying whether the user is logged into the application e.g. "Success" etc.
 - `name`: A string, represents the user's account name
 - `username`: A string, represents the user's account username

4.3.2 POST <http://localhost:3000/getUploads>

- Purpose
 - Retrieves information from the PostgreSQL database about the user's past audio uploads
- Request body parameters
 - Object { `username: "janelee" }`
 - `username`: A string, represents the user's account username
- Response object
 - Object
 - `message`: A string about the status of retrieving the user's audio uploads e.g. "No history of audio uploads", "Previous history of audio uploads", "Error with retrieving past uploads" etc.
 - `uploads`: Only returned when the user has a history of audio uploads, is an array that could contain multiple objects of the format { `id: 1, audio_category: "A", audio_description: "my first audio", s3_url: "https://AWS_BUCKET.s3.amazonaws.com/filename", created_at: "4 Apr 2025, 3:15pm", username: "janelee", file_name: "file.mp4" }`. With the exception of `id`, all fields in the object are strings

4.3.3 POST <http://localhost:3000/deleteAudio>

- Purpose
 - Deletes audio file selected by user from the AWS S3 bucket and the audio file's corresponding information from the PostgreSQL database
- Request body parameters
 - Object { `filename: "https://AWS_BUCKET.s3.amazonaws.com/filename" }`
 - `filename`: A string, representing the URL where the audio file is stored in the S3 bucket
- Response object
 - Object
 - `message`: A string about the status of deleting the audio file and its corresponding information e.g. "File has been deleted", "Error with deleting audio file" etc.

4.3.4 POST <http://localhost:3000/uploadFile>

- Purpose
 - Saves the audio category (A/B/C/D), audio description, original filename and S3 URL associated with the audio file in the PostgreSQL database under the user's username.
- Request body parameters
 - Object { `username: "janelee", fileName: "file.mp4", category: "A", description: "my first audio", s3URL: "https://AWS_BUCKET.s3.amazonaws.com/filename" }`
 - `username`: A string, represents the user's account username
 - `fileName`: A string, represents the actual name of the file that has been uploaded
 - `category`: A string, represents one of the 4 categories that the audio is classified under (A/B/C/D)
 - `description`: A string, represents the description of the audio file
 - `s3URL`: A string, represents the URL where the audio file is stored in the S3 bucket
- Response object
 - Object
 - `message`: A string about the status of the audio file information being uploaded to the PostgreSQL database e.g. "Audio information has been saved successfully", "Error with saving audio information" etc.

4.4 Account Settings Page

The API calls executed in the Account Settings page are listed below

4.4.1 GET <http://localhost:3000>

- Purpose
 - Verifies whether the user is logged into the application by reading the cookie containing the user's credentials in the browser. Returns the user's name and username if the user is logged in.
- Request body parameters
 - Not applicable for a GET request
- Response object
 - Object { `status: "Success", name: "Jane", username: "janelee" }`
 - `status`: A string about the status of verifying whether the user is logged into the application e.g. "Success" etc.
 - `name`: A string, represents the user's account name
 - `username`: A string, represents the user's account username

4.4.2 POST <http://localhost:3000/update>

- Purpose
 - Updates the user's name, username and password based on his input, the username field in the audios database table is also updated to the new username so that the user's audio upload history is retained. An error is thrown if all fields are empty, or if the user wants to change his username to a username that is already taken by an existing user

- Request body parameters
 - Object { name: "mary", username: "marytan", password: "new_password", cookiesName: "Jane", cookiesUsername: "janelee" }
 - name: A string, represents the name that the user wants to switch to
 - username: A string, represents the username that the user wants to switch to
 - password: A string, represents the password that the user wants to switch to
 - cookiesName: A string, represents the user's current name
 - cookiesUsername: A string, represents the user's current username
- Response object
 - Object
 - message: A string about the status of the user's credentials being updated e.g. "User details have been successfully updated", "This username is taken. Please choose another username" etc.

4.4.3 POST <http://localhost:3000/deleteAccount>

- Purpose
 - Deletes the user's account based on the username provided
- Request body parameters
 - Object { usernameToDelete: "janelee" }
 - usernameToDelete: A string, represents the user's account username
- Response object
 - Object
 - message: A string about the status of the user's account being updated e.g. "User account has been deleted" etc.

4.5 Navigation Bar

The API calls executed in the Navigation Bar are listed below

4.5.1 GET <http://localhost:3000/logout>

- Purpose
 - Clears the cookie containing the user's credentials in the user's browser, logging the user out of the application
- Request body parameters
 - Not applicable for a GET request
- Response object
 - Object
 - status: A string about the logout status of the user e.g. "Success"

-End of HTX Audio App Documentation-