

# **Audio Hosting Full Stack Application Documentation**

## Contents

<b>1. API Interface</b>	<b>2</b>
1.1. User / User Account	2
1.1.1. Authenticate User Login	2
1.1.2. GetAllUsersList	2
1.1.3. AddNewUser / SignUpNewUser	2
1.1.4. DeleteUser	3
1.1.5. UpdateUser	3
1.1.6. GetCurrentUser	3
1.2. Audio / Audio Upload API	3
1.2.1. GetUserAudioCollections	3
1.2.2. DeleteAudioTrack	4
1.2.3. InitAudioUpload	4
1.2.4. UploadAudioChunks	4
1.3. Middleware Definitions	4
1.3.1. VerifyToken (uses jwt token in authorization request header)	4
1.3.2. ValidateAdminUserRole	5
1.3.3. CheckUsernameAlrExist	5
<b>2. System Architecture</b>	<b>5</b>
2.1. Overview	5
2.2. Application File Directory Overview	5
2.3. Web Frontend Overview	6
2.4. Nodejs Backend Overview	6
2.5. Mongo DB Overview	6
2.6. Mongo Express Overview	6
<b>3. Instruction to Start Application</b>	<b>7</b>
<b>4. Application walkthrough</b>	<b>7</b>
4.1. Login Page (http://localhost:3000/login)	7
4.2. User Account Page (http://localhost:3000/account)	8
4.3. Audio Management Page (http://localhost:3000/audio)	9
<b>5. Source Codes and Project Artifacts</b>	<b>10</b>
<b>6. Docker Images</b>	<b>10</b>

## 1. API Interface

### 1.1. User / User Account

#### 1.1.1. Authenticate User Login

<a href="http://ServerIP:5555/user/authenticateLogin">http://ServerIP:5555/user/authenticateLogin</a>		Method=POST	Content-Type = JSON	
	Field Names	Data Type	Description	Example
<b>Input</b> ((Body))	Username	String	Username of user. Unique to each user	"User_123"
	Password	String	Password of the user.	"1Password"
<b>Return</b>	loginAuthentication StatusObj	Object	authenticateStatus → true if successful login (Boolean) message → description of the status (string) jwt → json web token is issued if authentication was successful. (string) Payload in the jwt contains the following information. (1) username (2) userID (3) role (4) timestamp jwt was issued (5) timestamp jwt expires	Response: { authenticateStatus: true, message: 'Successful Login', jwt: yJhbGciOiJIUzI1NiIs..... } }
	Status	Number	200 → Success, 500 → Error	

#### 1.1.2. GetAllUsersList

<a href="http://ServerIP:5555/user/getAllUsers">http://ServerIP:5555/user/getAllUsers</a>		Method=GET	Content-Type = JSON	Authorization-Header = JWT Token
	Field Names	Data Type	Description	Example
<b>Input</b> ((Body))	NIL	NIL	NIL	NIL
<b>Return</b>	usersList	Collection of Objects. Each object contains user information.	User Information Objects consists of (1) UserID → UUID (2) Username (3) Password (4) Role → Either "admin" or "user"	Response: { userID: "234033-234....." username: "Admin_User", password: "1Password" role: "Admin" } }
	Status	Number	200 → Success, 500 → Error	

#### 1.1.3. AddNewUser / SignUpNewUser

<a href="http://ServerIP:5555/user/addUser">http://ServerIP:5555/user/addUser</a>		Method=POST	Content-Type = JSON	Authorization-Header = JWT Token
<a href="http://ServerIP:5555/user/signUpNewUser">http://ServerIP:5555/user/signUpNewUser</a>		Method=POST	Content-Type = JSON	NO AUTHORIZATION HEADER
	Field Names	Data Type	Description	Example
<b>Input</b> (Body)	user	Object	User Object (1) UserID → UUID (2) Username (3) Password (4) Role → Either "admin" or "user"	{userID: "234033-234....." username: "Admin_User", password: "1Password" role: "Admin"}
<b>Return</b>	StatusObj	Object	Status → status of add new user	pass or fail
	Status	Number	200 → Success, 500 → Error	

### 1.1.4. DeleteUser

http://ServerIP:5555/user/deleteUser		Method=DELETE	Content-Type = JSON	Authorization-Header = JWT Token
	Field Names	Data Type	Description	Example
<b>Input</b> (URL Param)	userID	String	String → UUID	'00000000-0000-0000-0000-000000000000'
<b>Return</b>	StatusObj	Object	Status → Status of deletion, statusMsg → Description	statusMsg = "At Least One Admin User is needed. Unable to delete" status: "fair" or "pass"
	Status	Number	200 → Success, 500 → Error	

### 1.1.5. UpdateUser

http://ServerIP:5555/user/updateUser		Method=PUT	Content-Type = JSON	Authorization-Header = JWT Token
	Field Names	Data Type	Description	Example
<b>Input</b> (URL Param)	userID	String	String → UUID	'00000000-0000-0000-0000-000000000000'
<b>Input</b> (Body)	user	Object	Object containing all user information (1) UserID → UUID (2) Username (3) Password (4) Role → Either "admin" or "user"	{userID: "234033-234....." username: "Admin_User", password: "1Password" role: "Admin"}
<b>Return</b>	StatusObj	Object	Status → Status of deletion, statusMsg → Description	statusMsg = "At Least One Admin User is needed. Unable to delete" status: "fair" or "pass"
	Status	Number	200 → Success, 500 → Error	

### 1.1.6. GetCurrentUser

http://ServerIP:5555/user/getCurrentUsername		Method=GET	Content-Type = JSON	Authorization-Header = JWT Token
	Field Names	Data Type	Description	Example
<b>Input</b>	NIL	NIL	NIL	NIL
<b>Return</b>	username	Object	Contains one property username	{ username: "Admin_User" }
	Status	Number	200 → Success, 500 → Error	

## 1.2. Audio / Audio Upload API

### 1.2.1. GetUserAudioCollections

http://ServerIP:5555/audio/getUserAudioCollections		Method=GET	Content-Type = JSON	Authorization-Header = JWT Token
	Field Names	Data Type	Description	Example
<b>Input</b>	NIL	NIL	NIL	NIL
<b>Return</b>	collection	Collection of Objects. Each object contains user information.	User Information Objects consists of (1) filename → name of file (2) key → string info derived from upload time/date (3) audioCategory → string description of category	{ audioCategory: "pop" audioDescription: "best of 2000s pop ..." fileContent: "//uQZAAAAAAA"

			(4) audioDescription → string description of song (5) username → username of user who uploaded file. (6) userID → user id of user who uploaded file. (7) base64 string representation of media file.	fileName: "Test_1OMB_MP3.mp3" key: "1682254380997-9" userID: "00000000-0000-00000....." username: "Super_Admin", {...}, .....]
	Status	Number	200 → Success, 500 → Error	

### 1.2.2. DeleteAudioTrack

http://ServerIP:5555/audio/deleteAudio		Method=DELETE	Content-Type = JSON	Authorization-Header = JWT Token
	Field Names	Data Type	Description	Example
<b>Input</b> (URL Param)	key	String	key → unique string representation of audio file	"1682254380997-9"
<b>Return</b>	StatusObj	Object	Status → Status of deletion	status: "fair" or "pass"
	Status	Number	200 → Success, 500 → Error	

### 1.2.3. InitAudioUpload

http://ServerIP:5555/audio/upload/init		Method=POST	Content-Type = JSON	Authorization-Header = JWT Token
	Field Names	Data Type	Description	Example
<b>Input</b> (Headers)	X-Content-Length	Number	depicts the size of the file to be uploaded	
	X-Content-Name	String	depicts the name of the file	file_01.mp3
	X-Chunks-Quantity	Number	depicts the total slices/chunks the file will be broken into and sent	2
<b>Return</b>	fileId	String	Unique ID to represent the file to be uploaded by chunks.	"sdfss"
	Status	Number	200 → Success, 500 → Error	

### 1.2.4. UploadAudioChunks

http://ServerIP:5555/audio/upload		Method=POST	Content-Type = application/octet-stream	Authorization-Header = JWT Token
	Field Names	Data Type	Description	Example
<b>Input</b> (Headers)	Content-Length	Number	depicts the size of the chunk to be stream	
	Content-Length	Number	Size of file	
	X-Chunk-Id	String	Used to identify the file the chunk uploaded	file_01.mp3
	Content-Info	Object	Additional information of file uploaded, information to be saved in database together with file content.	username, userID, audioDescription, ..
<b>Return</b>	Size	Number	Size of chunk	
	Status	Number	200 → Success,	

## 1.3. Middleware Definitions

### 1.3.1. VerifyToken (uses jwt token in authorization request header)

Status	Action
No Token Found	Return 401 status and statusMsg: "Authorization Token not send in header. Please login first", status: "fail"
Token is not a valid JWT	Return 403 and statusMsg: "Unauthorized transaction"
Token is valid	extract the username, userID, role and pass it to next middleware.

### 1.3.2. ValidateAdminUserRole

Used to verify if user is admin, handle access control requirements associated with the api request.

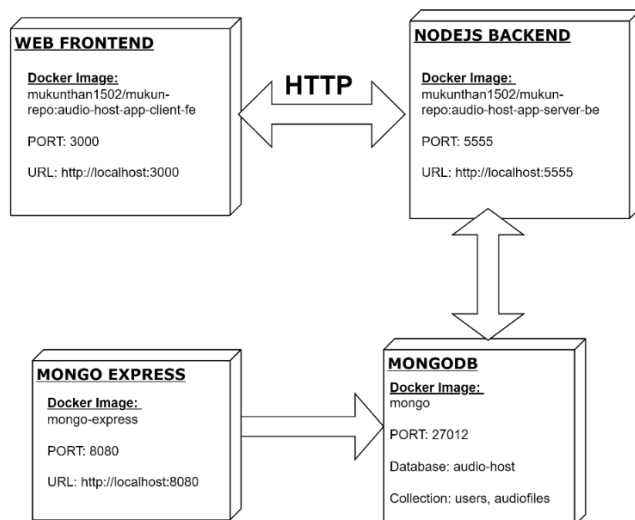
Status	Action
User Role is Admin	Call the next middleware
User Role is not Admin	Return 500 and StatusMsg: <username> current role has no access rights for current action. Current role: <role>. status: "fail"

### 1.3.3. CheckUsernameAlrExist

Status	Action
"username" is not unique	Return 409 and StatusMsg: <username> username is already taken. Try another username., status: "fail"
"username" is unique	Call the next middleware

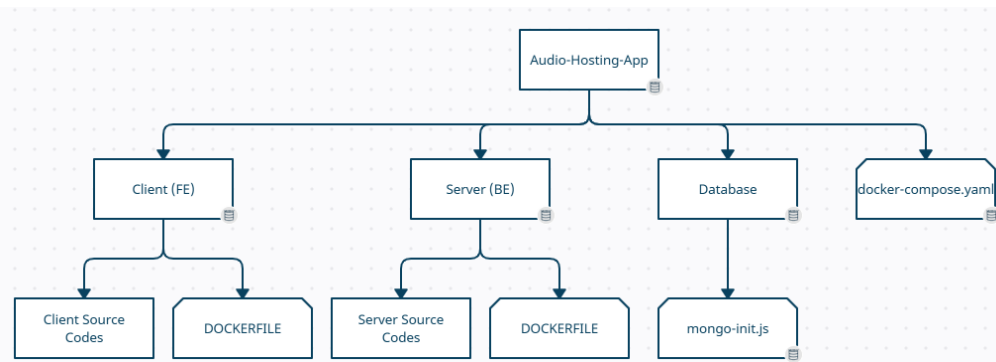
## 2. System Architecture

### 2.1. Overview



### 2.2. Application File Directory Overview

The 3 directories within Audio-Hosting-App folder contain codes and artifacts relating to the frontend, backend and database. The frontend and backend folders each contain a Dockerfile used to create their respective docker image. The mongo-init.js script resides in the Database folder. This script will initialize the mongo DB with some initial data when the mongo dB container is spawned and had no data initialize before.



### 2.3. Web Frontend Overview

The web frontend is developed using react framework. The frontend client runs on port 3000 and communicates with the backend service on port 5555 (configurable in docker-compose).

### 2.4. Nodejs Backend Overview

The backend runs on NodeJS and uses express framework. The server listens for incoming requests from clients (port 5555 – configurable), processes them, and responds with the appropriate data or actions. Express provides a set of middleware functions that can be used to handle requests and responses. Middleware functions are used to perform tasks such as request access control by examining json web token sent together with request header.

The backend services also interfaces with a mongo DB database for persistent storage on port 27017.

### 2.5. Mongo DB Overview

The database is used for persistent storage in our application. The database name used is “audio-host”. There are 2 collections present in the database. Communicates on port 27017. Uses latest official mongo docker image from docker hub.

Collection	Description
users	Collection of user details information. Each document consists of username, password, userID and role.
audiofiles	Collection of audio details information. Each document consists of filename, key, audioCategory, audioDescription, username, userID and fileContent.

Document definition for user collection		
Property	Data Type	Example
username	String	“userxx”
password	String	“1Password”
userID	String (UUID)	1234-2342...
role	String	“admin”or “user”

Document definition for audiofiles collection		
Property	Data Type	Example
filename	String	“song1.mp3”
key	String	“123-234324....”
audioCategory	String	“Rock”
audioDescription	String	“admin” or “user”
username	String	“userxx”
userID	String (UUID)	1234-2342...
fileContent	base64 string	“//uQZAAAAAAA”

### 2.6. Mongo Express Overview

A web-based user interface (UI) tool for MongoDB that allows us to easily manage and interact with MongoDB database using port 8080. Uses latest official mongo-express docker image from docker hub.

### 3. Instruction to Start Application


1. Run docker-compose command on docker-compose.yaml file.
  - 1.1. In root folder, enter cmd and type "docker compose -f docker-compose.yaml up"
  - 1.2. This will start up all 4 containers.
2. Once all containers are started up. Navigate to <http://localhost:3000/login> to navigate to application.
  - 2.1. If this is the first time initializing the database, then 2 default user entries will populate the database. Initial application login will use these initial user entries. The initial user entries are defined in ./Database/mongo-init.js. Current initial user entries are as follows.


```
{
  username: "Super_Admin",
  password: "1Password",
  userID: "00000000-0000-0000-0000-000000000000",
  role: "admin",
},
{
  username: "User",
  password: "1Password",
  userID: "8770db80-0d4b-4edc-bf84-4076d41259c6",
  role: "user",
},
}
```

3. Navigate to <http://localhost:8080> to navigate to the web interface of the mongo database. This interface allows for manual manipulation of the database. (add/remove. modify users, audio).
  - 3.1. Database used for this application is called "audio-host"
  - 3.2. Collection used are "users" and "audiofiles"
4. Backend service can be accessed using <http://localhost:5555/>.

### 4. Application walkthrough

- 4.1. Login Page (<http://localhost:3000/login>)
  - 4.1.1 Enter username and password to log into application.
  - 4.1.2 No maximum attempts penalty

 Please Log In

 Login

---

## Login Page

\* Username

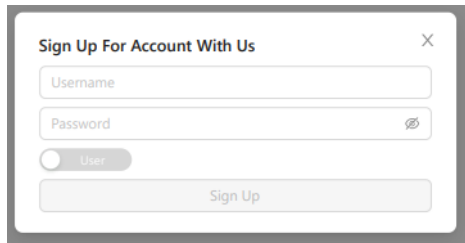
\* Password

☒ Remember me

Submit

[Create Account](#)

- 4.3 Click on “Create Account” to create a new Account.
  - 4.3.1 Enter Username, password and click “Sign Up”.
  - 4.3.2 New Signups from login page will not be able to choose the admin option.



A modal form titled "Sign Up For Account With Us" with a close button (X) in the top right corner. It contains two input fields: "Username" and "Password" (with a toggle icon). Below the password field is a radio button labeled "User". At the bottom is a "Sign Up" button.

#### 4.2. User Account Page (<http://localhost:3000/account>)

- 4.2.1 Upon successful login, the application will be re-directed to the accounts page.
- 4.2.2 Only admin users can add new users, edit and delete existing users.
- 4.2.3 This page can also be accessed by clicking on Account Management on navigation bar.

Logged in as Super\_Admin

Accounts Management Audio Management Logout

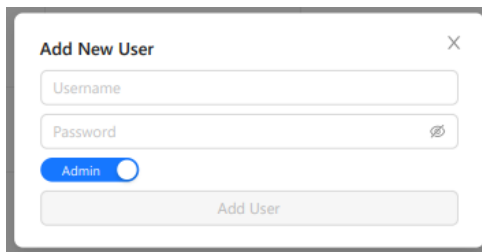
### Accounts Management

Add User

UserID	Username	Password	Account Type	Actions
00000000-0000-0000-0000-000000000000	Super_Admin	*****	admin	Edit Delete
84eba93d-ddd8-4e9d-951e-ab3b5935f8c6	ABC	*****	user	Edit Delete
2813308d-f3ff-4569-8bcd-b667c5397c01	User2	*****	user	Edit Delete

< 1 >

#### 4.2.4 Clicking On “Add User”.



A modal form titled "Add New User" with a close button (X) in the top right corner. It contains two input fields: "Username" and "Password" (with a toggle icon). Below the password field is a radio button labeled "Admin". At the bottom is an "Add User" button.

#### 4.2.5 View when click on edit.

00000000-0000-0000-0000-000000000000

Super\_Admin

Admin

Save Cancel

#### 4.2.6 Click on “Save” button to save changes, “Cancel” button to revert changes.



#### 4.3. Audio Management Page (<http://localhost:3000/audio>)

- 4.3.1 Page displays all audio files uploaded by current user only.
- 4.3.2 Additional audio files can be uploaded with corresponding category and description.
  - 4.3.2.1 “Click on Upload” to upload audio file.
  - 4.3.2.2 “Add to Collection” to save file to audio collection.
- 4.3.3 Click on “Delete” to remove audio file from collection.
- 4.3.4 Media player on each row allows for
  - 4.3.4.1 Playback and seeking of media.
  - 4.3.4.2 Volume and playback speed adjustment.
  - 4.3.4.3 Downloading of audio file to local system.
- 4.3.5 This page can also be accessed by clicking on Audio Management on navigation bar.

Logged in as Super\_Admin

Accounts Management

**Audio Management**

Logout

### Audio Manangement Page

Add Audio

File Name	Category	Description	Action
Free_Test_Data_1OMB_MP3.mp3	POP	Super_Admin_10mb	<div>Delete</div> <div><div>4:02 / 7:13</div><div></div><div></div><div></div></div>
file_example_MP3_700KB.mp3	ROCK	Super_Admin_700kb	<div>Delete</div> <div><div>0:22 / 0:27</div><div></div><div></div><div></div></div>

< 1 >

4.4 Clicking “Add Audio” will show this pop-put. Enter description, category and upload audio.

#### Add Audio Track

×

Test

Test TEST

Click to Upload

Free\_Test\_Data\_1OMB\_MP3.mp3

Add To Collection

## **5. Source Codes and Project Artifacts**

The source code and artifacts can be cloned from the following GitHub repo

<https://github.com/mukunthan1502/Audio-Hosting-Application.git/>

## **6. Docker Images**

The docker images for the frontend and backend are hosted at Docker hub repository at the following repo mukunthan1502/mukun-repo. The mongo and mongo-express images are the latest official release on Docker hub.

- Client (FE) tag name: audio-host-app-client-fe
- Server (BE) tag name: audio-host-app-server-be