Project 02 - Online Music Store - Manage your personal Music library

Initialize the Application

The version of Node in the local machine is
v16.13.0
The version of npm (node package manager) in the local machine is
8.19.1
Initialize the application by typing the following command in the terminal
npm init --yes
install the required packages (latest) by typing the following commands in the terminal
npm install mongodb express nodemon dotenv body-parser multer
nodemon automatically incorporate changes to js files without the need to stop and restart the server each time a change is made to the files
Create a config file to add environment variables to make the project dynamic

config.env
NODE_ENV=development PORT=5000

Add a command to start script to set NODE_ENV as production and create a script as dev to run server as nodemon

package.json

```
{
  "name": "backend_nodejs_v1",
  "version": "1.0.0",
  "description": "Backend for Online Music Store created using Node.js",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "SET NODE_ENV=production && node server",
    "dev": "nodemon server"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/zvdas/OnlineMusicStore.git"
  },
  "keywords": [
    "node",
    "express"
  ],
  "author": "zvdas",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/zvdas/OnlineMusicStore/issues"
  },
  "homepage": "https://github.com/zvdas/OnlineMusicStore#readme",
  "dependencies": {
    "body-parser": "^1.20.0",
    "dotenv": "^16.0.3",
    "express": "^4.18.1",
    "mongoose": "^6.6.4",
    "multer": "^1.4.5-lts.1",
    "nodemon": "^2.0.20",
    "pug": "^3.0.2"
  }
}
```

Start Express on localhost:3000

app.js

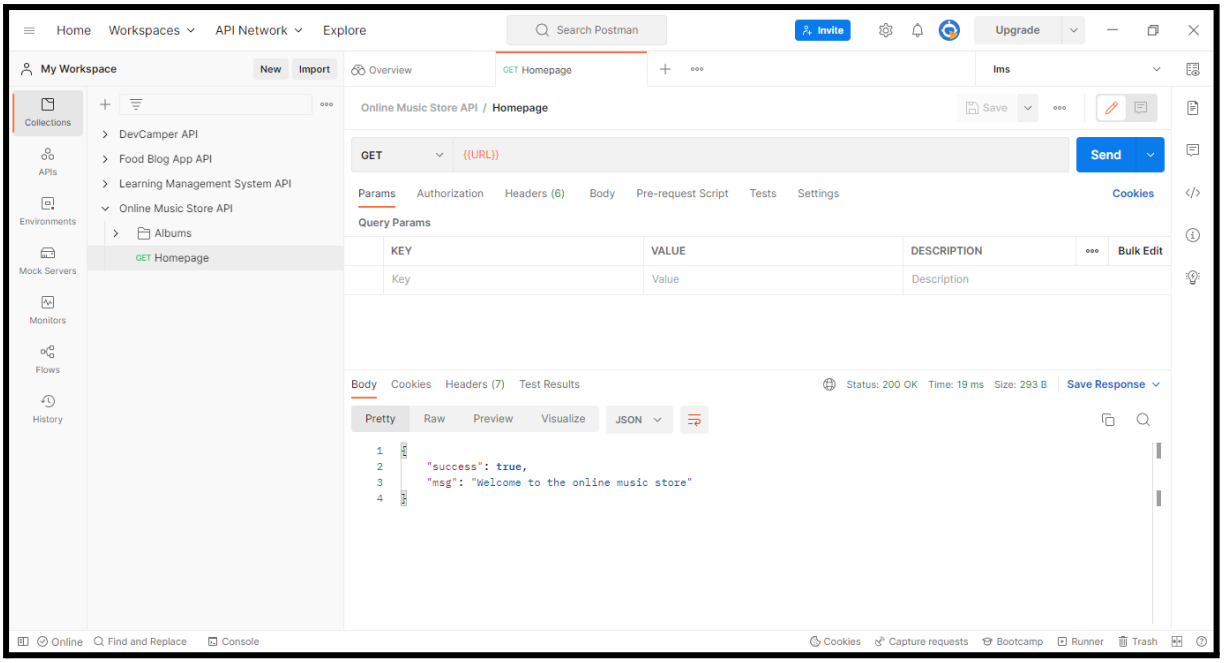
```
// import the necessary modules
const express = require('express');
const dotenv = require('dotenv');

// load environment variables
dotenv.config({ path: './config/config.env' });

// initialize express app
const app = express();

// get homepage route
app.get('/', (req, res) => {
   res.status(200).json({ success: true, msg: 'Welcome to the online music store' });
});

const PORT = process.env.PORT || 5000;

app.listen(
   PORT,
   () => console.log(`Server running in ${ process.env.NODE_ENV } mode on port ${ PORT }`)
);
```

Start express in developer mode by typing the following command in the terminal
npm run dev

vs code console output

```
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Server running in development mode on port 5000
```

Open Postman and create a collection with the name of the project. This collection will contain all the API requests created which can be run by clicking on it instead of configuring each one separately. Add localhost:3000 as an environment variable {{URL}} which will be used frequently and repeatedly. Add and save the GET {{URL}} request as Homepage. send GET {{URL}}. The following will show in the Postman console.



postman console output

```
{
   "success": true,
   "msg": "Welcome to the online music store"
}
```

Configure & Connect to Database (MongoDB)

create the configuration for MongoDB database and connect to the database. To ensure dynamic & reusable settings & avoid hardcoding, place the MongoDB URI in the config file.

config.env

```
NODE_ENV=development
PORT=5000
MONGODB_URI=**enter MongoDB URI here**
```

database.js

```
const mongoose = require('mongoose');

const connectDB = async() => {
   const conn = await mongoose.connect(process.env.MONGODB_URI, {
      useNewUrlParser: true,
      UseUnifiedTopology: true
   });

   console.log(`MongoDB connected: ${conn.connection.host}`);
}

module.exports = connectDB;
```

server.js

```
…
// load environment variables
dotenv.config({ path: './config/config.env' });

// config file
const connectDB = require('./config/database');

// initialize express app
const app = express();

// connect database
connectDB();
…
```

console output

```
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
Server running in development mode on port 5000
MongoDB connected: ac-zqrbyuq-shard-00-00.2dm5ajx.mongodb.net
```

Handle Unhandled Promise Rejections (MongoDB connection errors)

If the URI string has errors , like one letter missing from the password, then the application will hang. Hence handle unhandled promise rejections be exiting or crashing the app in such a scenario

server.js

```
…
const PORT = process.env.PORT || 5000;

const server =
app.listen(
   PORT,
   () => console.log(`Server running in ${ process.env.NODE_ENV } mode on port ${ PORT }`)
);

// handle unhandled promise rejections
process.on('unhandledRejection', (err, promise) => {
   console.log(`Error: ${err.message}`);
   // close server & exit process
   server.close(() => process.exit(1));
});
```

Create the Album Model

Create an album model using mongoose schema.

models > Album.js

```
const mongoose = require('mongoose');
const slugify = require('slugify');

const AlbumSchema =  new mongoose.Schema({
   album_name: {
      type: String,
      required: [true, 'Please add a name for the Album'],
      unique: true,
      trim: true,
      maxlength: [50, 'Name cannot exceed 50 characters']
   },
   album_slug: String,
   // to slugify the url path (ie) "Admiral Bob" => "admiral-bob" for parsing url
   cover_photo: {
      type: String,
      default: 'no-photo.jpg'
   },
   artist: {
      type: String,
      required: [true, 'Please add an artist\'s name'],
   },
   artist_slug: String,
   genre: String,
   year: Number,
   producer: String,
   averageRating: {
      type: Number,
      min: [1, 'Rating must be atleast 1'],
      max: [5, 'Rating cannot be more than 5']
   },
   description: {
      type: String,
      maxlength: [500, 'Name cannot exceed 500 characters']
   },
   album_url: String,
   createdAt: {
      type: Date,
      default: Date.now()
   }
});

// create album slug from the name
AlbumSchema.pre('save', function(next) {
   this.album_slug = slugify(this.album_name, { lower: true });
   this.artist_slug = slugify(this.artist, { lower: true });
   next();
});

module.exports = mongoose.model('Album', AlbumSchema);
```

Create the Album Controller

Create the controllers for albums which will perform the actions set by the API route

controllers > albums.js

```
const AlbumModel = require('../models/Album');

// @desc    Get all albums
// @route   GET /api/v1/albums
// @access  Public
exports.getAlbums = async(req, res) => {
   const albums = await AlbumModel.find();

   res.status(200).json({ success: true, count: albums.length, data: albums });
}

// @desc    Get album by ID
// @route   GET /api/v1/albums/:id
// @access  Public
exports.getAlbumById = async(req, res) => {
   const album = await AlbumModel.findById(req.params.id);

   res.status(200).json({ success: true, data: album });
}

// @desc    Create new album
// @route   POST /api/v1/albums
// @access  Private
exports.createAlbum = async(req, res) => {
   const albums = await AlbumModel.create(req.body);

   res.status(201).json({ success: true, msg: 'Album created successfully' });
}

// @desc    Update an album
// @route   PUT /api/v1/albums/:id
// @access  Private
exports.updateAlbumById = async(req, res) => {
   const album = await AlbumModel.findByIdAndUpdate(req.params.id, req.body, { new: true,
runValidators: true });

   res.status(200).json({ success: true, msg: `Album with id ${req.params.id} updated successfully`,
data: album });
}

// @desc    Delete an album
// @route   DELETE /api/v1/albums/:id
// @access  Private
exports.deleteAlbumById = async(req, res) => {
   const album = await AlbumModel.findByIdAndDelete(req.params.id);
```

```
   res.status(200).json({ success: true, msg: `Album with id ${req.params.id} deleted successfully`
});
}
```

Create the Album Routes

Create the API routes for albums

routes > albums.js

```
const express = require('express');

const { getAlbums, getAlbumById, createAlbum, updateAlbumById, deleteAlbumById } =
require('../controllers/albums');

const router = express.Router();

router
   .route('/')
   .get(getAlbums)
   .post(createAlbum);

router
   .route('/:id')
   .get(getAlbumById)
   .put(updateAlbumById)
   .delete(deleteAlbumById);

module.exports = router;
```

Add the routes files and mount the routes in server.js file

server.js

```
…
// config file
const connectDB = require('./config/database');

// route files
const albums = require('./routes/albums');
…
// initialize express app
const app = express();

// body-parser
app.use(express.json());
…
app.get('/', (req, res) => {
   res.status(200).json({ success: true, msg: 'Welcome to the online music store' });
});

// mount routers
```

```
app.use('/api/v1/albums', albums);
…
```

Consume the APIs using Postman

Create a requests folder in Postman for albums, and create CRUD requests for each in the folders

GET ALL ALBUMS

Send GET {{URL}}/api/v1/albums

postman console output


```
{
   "success": true,
   "count": 0,
   "data": []
}
```


CREATE NEW ALBUM

Set key ‘Content-Type’ to value ‘application/json’ in headers and send the following message in POST {{URL}}/api/v1/albums


```
{
      "album_name": "Airtone",
      "cover_photo": "airtone.jpeg",
      "artist": "Martyn Bloor",
      "genre": "Instrumental",
      "year": 2022,
      "producer": "CCMixter",
      "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
      "album_url": "http://dig.ccmixter.org/people/airtone"
}
```


postman console output


```
{
   "success": true,
   "msg": "Album created successfully"
}
```


Send again GET {{URL}}/api/v1/albums

Postman console output


```
{
   "success": true,
   "count": 1,
   "data": [
      {
         "_id": "6360c4e5bdea6f0358e910c1",
```

```
        "album_name": "Airtone",
        "cover_photo": "airtone.jpeg",
        "artist": "Martyn Bloor",
        "genre": "Instrumental",
        "year": 2022,
        "producer": "CCMixter",
        "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
        "album_url": "http://dig.ccmixter.org/people/airtone",
        "createdAt": "2022-11-01T07:03:54.389Z",
        "__v": 0
    }
  ]
}
```


GET ALBUM BY ID

Send GET {{URL}}/api/v1/albums/6360c4e5bdea6f0358e910c1

Postman Console output


```
{
  "success": true,
  "data": {
    "_id": "6360c4e5bdea6f0358e910c1",
    "album_name": "Airtone",
    "cover_photo": "airtone.jpeg",
    "artist": "Martyn Bloor",
    "genre": "Instrumental",
    "year": 2022,
    "producer": "CCMixter",
    "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
    "album_url": "http://dig.ccmixter.org/people/airtone",
    "createdAt": "2022-11-01T07:03:54.389Z",
    "__v": 0
  }
}
```


UPDATE ALBUM BY ID

Set key ‘Content-Type’ to value ‘application/json’ in headers and send the following message in PUT {{URL}}/api/v1/albums/6360c4e5bdea6f0358e910c1


```
{
  "year": "2021"
}
```


Postman console output

```
{
   "success": true,
   "msg": "Album with id '6360c4e5bdea6f0358e910c1' updated successfully",
   "data": {
      "_id": "6360c4e5bdea6f0358e910c1",
      "album_name": "Airtone",
      "cover_photo": "airtone.jpeg",
      "artist": "Martyn Bloor",
      "genre": "Instrumental",
      "year": 2021,
      "producer": "CCMixter",
      "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
      "album_url": "http://dig.ccmixter.org/people/airtone",
      "createdAt": "2022-11-01T07:03:54.389Z",
      "__v": 0
   }
}
```


DELETE ALBUM BY ID

Send DELETE {{URL}}/api/v1/albums/6360c4e5bdea6f0358e910c1

Postman console output


```
{
   "success": true,
   "msg": "Album with id 6360c4e5bdea6f0358e910c1 deleted successfully"
}
```


Send again GET {{URL}}/api/v1/albums

Postman console output


```
{
   "success": true,
   "count": 0,
   "data": []
}
```


Use a Logging Middleware

Use a logging middleware called Morgan that logs API requests to the console

Install morgan using npm

npm i morgan

server.js

```
// import the necessary modules
const express = require('express');
```

```
const dotenv = require('dotenv');
const morgan = require('morgan');
…
// route files
const albums = require('./routes/albums');

// initialize express app
const app = express();

// dev logging middleware: use in development mode
if(process.env.NODE_ENV === 'development'){
   app.use(morgan('dev'));
}
..
```

Send GET request using Postman

console output

```
[nodemon] restarting due to changes...
[nodemon] starting `node server.js`
Server running in development mode on port 5000
MongoDB connected: ac-zqrbyuq-shard-00-01.2dm5ajx.mongodb.net
GET /api/v1/albums 200 89.370 ms - 36
```

Custom Error Handler Middleware

Create a middleware to handle database errors during CRUD operation. The error handler will take errors from the route controllers via next of (req, res, next). A class errorResponse is created extending the error class, taking a custom error message from the user and assigning a user defined status code for each error case, like duplicate entry error, resource not found error, etc.

middleware > error.js

```
const ErrorResponse = require("../utils/errorResponse");

const errorHandler = (err, req, res, next) => {
   let error = { ...err };

   error.message = err.message;

   // log to console for dev
   console.log(err);

   // Mongoose bad ObjectId Error
   if(err.name === 'CastError') {
      const message = `Resource with id '${err.value}' not found`;
      error = new ErrorResponse(message, 404);
   }

   // Mongoose duplicate key error
   if(err.code === 11000) {
```

```
      const message = `Resource with same ${Object.keys(err.keyValue)} already exists` ;
      error = new ErrorResponse(message, 400);
   }

   // Mongoose validation error
   if(err.name === 'ValidationError') {
      const message = Object.values(err.errors).map(val => val.message);
      error = new ErrorResponse(message, 400);
   }

   res
      .status(error.statusCode || 500)
      .json({
         success: false,
         error: error.message || 'Internal Server Error'
      })
}

module.exports = errorHandler;
```

utils > errorResponse.js

```
class ErrorResponse extends Error {
   constructor(message, statusCode) {
      super(message);
      this.statusCode = statusCode;
   }
}

module.exports = ErrorResponse;
```

server.js

```
…
// route files
const albums = require('./routes/albums');

// middleware files
const error = require('./middleware/error');
…
// mount routers
app.use('/api/v1/albums', albums);

// use error handler middleware
app.use(errorHandler);
…
```

controllers > albums.js

```
const AlbumModel = require('../models/Album');
const ErrorResponse = require('../utils/errorResponse');
```

```
// @desc    Get all albums
// @route   GET /api/v1/albums
// @access  Public
exports.getAlbums = async(req, res, next) => {
   const albums = await AlbumModel.find();

   res
      .status(200)
      .json({
         success: true,
         count: albums.length,
         data: albums
      });
}

// @desc    Get album by ID
// @route   GET /api/v1/albums/:id
// @access  Public
exports.getAlbumById = async(req, res, next) => {
   try {
      const album = await AlbumModel.findById(req.params.id);

      // error for correctly formatted id not present in database
      if(!album) {
         return next(new ErrorResponse(`Album with id '${req.params.id}' not found`, 404));
      }

      res
         .status(200)
         .json({
            success: true,
            data: album
         });
   } catch (err) {
      // error for incorrectly formatted id
      next(err);
   }
}

// @desc    Create new album
// @route   POST /api/v1/albums
// @access  Private
exports.createAlbum = async(req, res, next) => {
   try {
      const albums = await AlbumModel.create(req.body);

      res
         .status(201)
         .json({
            success: true,
            msg: 'Album created successfully'
```

```
        });
    } catch (err) {
        next(err);
    }
}

// @desc    Update an album
// @route   PUT /api/v1/albums/:id
// @access  Private
exports.updateAlbumById = async(req, res, next) => {
    try {
        const album = await AlbumModel.findByIdAndUpdate(req.params.id, req.body, { new: true,
runValidators: true });

        // error for correctly formatted id not present in database
        if(!album) {
            return next(new ErrorResponse(`Album with id '${req.params.id}' not found`, 404));
        }

        res
            .status(200)
            .json({
                success: true,
                msg: `Album with id '${req.params.id}' updated successfully`,
                data: album
            });
    } catch (err) {
        next(err);
    }
}

// @desc    Delete an album
// @route   DELETE /api/v1/albums/:id
// @access  Private
exports.deleteAlbumById = async(req, res, next) => {
    try {
        const album = await AlbumModel.findByIdAndDelete(req.params.id);

        // error for correctly formatted id not present in database
        if(!album) {
            return next(new ErrorResponse(`Album with id '${req.params.id}' not found`, 404));
        }

        res
            .status(200)
            .json({
                success: true,
                msg: `Album with id '${req.params.id}' deleted successfully`
            });
    } catch (err) {
        next(err);
```

```
    }
}
```

Display Error Handling Messages in Postman

Object with correctly formatted id not found
Send GET to {{URL}}/api/v1/albums/6360c5c0bdea6f0358e910c7
Postman console output

{
"success": false,
"error": "Album with id '6360c5c0bdea6f0358e910c7' not found"
}

Bad ObjectId error (Object with incorrectly formatted id)
Send GET to {{URL}}/api/v1/albums/6360c5c0bdea6f0358e910c8ee
Postman console output

{
"success": false,
"error": "Resource with id '6360c5c0bdea6f0358e910c8ee' not found"
}

Duplicate key error
Send POST to {{URL}}/api/v1/albums twice with the same "album_name"
Postman console output

{
"success": false,
"error": "Resource with same name already exists"
}

Validation error
Send POST to {{URL}}/api/v1/albums with blank content { }
Postman console output

{
"success": false,
"error": "Please add an artist's name,Please add a name for the Album"
}

Custom Async Handler Middleware

Create an async handler middleware to eliminate the try-catch blocks yet implement the same functionality (adhering to the DRY principle - Don’t Repeat Yourself).

middleware > async.js

```
const asyncHandler = fn => (req, res, next) =>
   Promise
      .resolve(fn(req, res, next))
      .catch(next)

module.exports = asyncHandler;
```

controllers > albums.js

```
const AlbumModel = require('../models/Album');
const asyncHandler = require('../middleware/async');
const ErrorResponse = require('../utils/errorResponse');

// @desc    Get all albums
// @route   GET /api/v1/albums
// @access  Public
exports.getAlbums = asyncHandler(async(req, res, next) => {
   const albums = await AlbumModel.find();

   res
      .status(200)
      .json({
         success: true,
         count: albums.length,
         data: albums
      });
});

// @desc    Get album by ID
// @route   GET /api/v1/albums/:id
// @access  Public
exports.getAlbumById = asyncHandler(async(req, res, next) => {
   const album = await AlbumModel.findById(req.params.id);

   // error for correctly formatted id not present in database
   if(!album) {
      return next(new ErrorResponse(`Album with id '${req.params.id}' not found`, 404));
   }

   res
      .status(200)
      .json({
         success: true,
         data: album
      });
});

// @desc    Create new album
// @route   POST /api/v1/albums
```

```
// @access  Private
exports.createAlbum = asyncHandler(async(req, res, next) => {
   const albums = await AlbumModel.create(req.body);

   res
      .status(201)
      .json({
         success: true,
         msg: 'Album created successfully'
      });
});

// @desc    Update an album
// @route   PUT /api/v1/albums/:id
// @access  Private
exports.updateAlbumById = asyncHandler(async(req, res, next) => {
   const album = await AlbumModel.findByIdAndUpdate(req.params.id, req.body, { new: true,
runValidators: true });

   // error for correctly formatted id not present in database
   if(!album) {
      return next(new ErrorResponse(`Album with id '${req.params.id}' not found`, 404));
   }

   res
      .status(200)
      .json({
         success: true,
         msg: `Album with id '${req.params.id}' updated successfully`,
         data: album
         });
});

// @desc    Delete an album
// @route   DELETE /api/v1/albums/:id
// @access  Private
exports.deleteAlbumById = asyncHandler(async(req, res, next) => {
   const album = await AlbumModel.findByIdAndDelete(req.params.id);

   // error for correctly formatted id not present in database
   if(!album) {
      return next(new ErrorResponse(`Album with id '${req.params.id}' not found`, 404));
   }

   res
      .status(200)
      .json({
         success: true,
         msg: `Album with id '${req.params.id}' deleted successfully`
      });
});
```

results are same as “Consume APIs using Postman” & “Display error handling messages”

Mongoose Middleware (Create slug)

Create a slug from the album name to send when making a request. The spaces are replaced by ‘-’ & case is changed to lowercase.

Install slugify using npm

npm i slugify

models > Album.js

```
const mongoose = require('mongoose');
const slugify = require('slugify');

const AlbumSchema =  new mongoose.Schema({
…
});

// create album slug from the name
AlbumSchema.pre('save', function(next) {
   this.slug = slugify(this.album_name, { lower: true });
   next();
});

module.exports = mongoose.model('Album', AlbumSchema);
```

Send POST request to {{URL}}/api/v1/albums with following data


```
{
   "album_name": "Hear Me Now",
   "artist": "James Pomelo",
   "genre": "Pop",
   "year": 2003,
   "producer": "Martin Records",
   "averageRating": 3,
   "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.",
   "price": "399.99"
}
```


Postman console output


```
{
   "success": true,
   "count": 1,
   "data": [
      {
         "_id": "6356b1a49febac8392490fa0",
         "album_name": "Hear Me Now",
         "cover_photo": "no-photo.jpg",
         "artist": "James Pomelo",
```

```
        "genre": "Pop",
        "year": 2003,
        "producer": "Martin Records",
        "averageRating": 3,
        "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.",
        "price": 399.99,
        "createdAt": "2022-10-24T15:38:03.721Z",
        "slug": "hear-me-now",
        "__v": 0
    }
  ]
}
```

Database Seeder

Create a database seeder to import multiple rows from a json file into the database and delete all the data from the database using a single command

albums.json

```
[
  {
    "album_name": "Airtone",
    "cover_photo": "airtone.jpeg",
    "artist": "Martyn Bloor",
    "genre": "Instrumental",
    "year": 2022,
    "producer": "CCMixter",
    "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
    "album_url": "http://dig.ccmixter.org/people/airtone"
  },
  {
    "album_name": "Admiral Bob",
    "cover_photo": "admiral_bob.jpg",
    "artist": "Admiral Bob",
    "genre": "Instrumental",
    "year": 2020,
    "producer": "CCMixters",
    "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
    "album_url": "http://dig.ccmixter.org/people/admiralbob77"
  },
  {
    "album_name": "Mindmapthat",
    "cover_photo": "kara_square.jpg",
    "artist": "Kara Square",
    "genre": "Instrumental",
```

```
    "year": 2021,
    "producer": "CCMixter",
    "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
    "album_url": "http://dig.ccmixter.org/people/mindmapthat"
  },
  {
    "album_name": "Siobhan Dakay",
    "cover_photo": "siobhan_dakay.jpg",
    "artist": "Siobhan Dakay",
    "genre": "Instrumental",
    "year": 2007,
    "producer": "CCMixter",
    "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
    "album_url": "http://dig.ccmixter.org/people/SiobhanD"
  },
  {
    "album_name": "Radioontheshelf",
    "cover_photo": "Radioontheshelf.jpg",
    "artist": "Radioontheshelf",
    "genre": "Instrumental",
    "year": 2022,
    "producer": "CCMixter",
    "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
    "album_url": "http://dig.ccmixter.org/people/Radioontheshelf"
  }
]
```

seeder.js

```
// import the necessary modules
const fs = require('fs');
const mongoose = require('mongoose');
const colors = require('colors');
const dotenv = require('dotenv');

// load env variables
dotenv.config({ path: './config/config.env' });

// load models
const Album = require('./models/Album');

// connect to DB
mongoose.connect(process.env.MONGODB_URI, {
  useNewUrlParser: true,
  UseUnifiedTopology: true
```

```
});

// read JSON files
const albums =  JSON.parse(fs.readFileSync(`${__dirname}/_data/albums.json`, 'utf-8'));

// import into db
const importData = async () => {
   try {
      await Album.create(albums);

      console.log('data imported...'.green.inverse);
      // exit the process
      process.exit();
   } catch (err) {
      console.log(err);
   }
}

// delete from db
const deleteData = async () => {
   try {
      await Album.deleteMany();

      console.log('data deleted...'.red.inverse);
      // exit the process
      process.exit();
   } catch (err) {
      console.log(err);
   }
}

// add command line argument to import or delete data
if (process.argv[2] === '-i') {
   importData();
} else if (process.argv[2] === '-d') {
   deleteData();
}
```

run “node seeder -d” in terminal

console output

data deleted...

send GET all albums in Postman

postman console output

```
{
   "success": true,
   "count": 0,
   "data": []
```

}

run “node seeder -i” in terminal

console output

data imported...

send GET all albums in Postman

postman console output


```
{
   "success": true,
   "count": 5,
   "data": [
      {
         "_id": "6360ed4964cfb2c6e54b6f89",
         "album_name": "Admiral Bob",
         "cover_photo": "admiral_bob.jpg",
         "artist": "Admiral Bob",
         "genre": "Instrumental",
         "year": 2020,
         "producer": "CCMixters",
         "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
         "album_url": "http://dig.ccmixter.org/people/admiralbob77",
         "createdAt": "2022-11-01T09:56:25.757Z",
         "slug": "admiral-bob",
         "__v": 0
      },
      {
         "_id": "6360ed4964cfb2c6e54b6f8a",
         "album_name": "Mindmapthat",
         "cover_photo": "kara_square.jpg",
         "artist": "Kara Square",
         "genre": "Instrumental",
         "year": 2021,
         "producer": "CCMixter",
         "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
         "album_url": "http://dig.ccmixter.org/people/mindmapthat",
         "createdAt": "2022-11-01T09:56:25.757Z",
         "slug": "mindmapthat",
         "__v": 0
      },
      {
         "_id": "63627554c1648b56814fc92d",
         "album_name": "Siobhan Dakay",
         "cover_photo": "siobhan_dakay.jpg",
```

```
        "artist": "Siobhan Dakay",
        "genre": "Instrumental",
        "year": 2007,
        "producer": "CCMixter",
        "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
        "album_url": "http://dig.ccmixter.org/people/SiobhanD",
        "createdAt": "2022-11-02T13:44:22.349Z",
        "slug": "siobhan-dakay",
        "__v": 0
    },
    {
        "_id": "6360ed4964cfb2c6e54b6f88",
        "album_name": "Airtone",
        "cover_photo": "airtone.jpeg",
        "artist": "Martyn Bloor",
        "genre": "Instrumental",
        "year": 2022,
        "producer": "CCMixter",
        "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
        "album_url": "http://dig.ccmixter.org/people/airtone",
        "createdAt": "2022-11-01T09:56:25.757Z",
        "slug": "airtone",
        "__v": 0
    },
    {
        "_id": "6360ed4964cfb2c6e54b6f8c",
        "album_name": "Radioontheshelf",
        "cover_photo": "Radioontheshelf.jpg",
        "artist": "Radioontheshelf",
        "genre": "Instrumental",
        "year": 2022,
        "producer": "CCMixter",
        "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
        "album_url": "http://dig.ccmixter.org/people/Radioontheshelf",
        "createdAt": "2022-11-01T09:56:25.757Z",
        "slug": "radioontheshelf",
        "__v": 0
    }
  ]
}
```


Advanced Filtering

controllers > albums.js

```
…
// @desc    Get all albums
// @route   GET /api/v1/albums
// @access  Public
exports.getAlbums = asyncHandler(async(req, res, next) => {
   let query;

   // create query string
   let queryStr = JSON.stringify(req.query);

   // create query operators ($gt, $gte, etc...)
   queryStr = queryStr.replace(/\b(gt|gte|lt|lte|in)\b/g, match => `$${match}`);

   // finding resource
   const albums = await AlbumModel.find(JSON.parse(queryStr));

   res
      .status(200)
      .json({
         success: true,
         count: albums.length,
         data: albums
      });
});
…
```

Send GET to {{URL}}/api/v1/albums?year[lte]=2020&genre=Instrumental

Postman console output


```
{
   "success": true,
   "count": 2,
   "data": [
      {
         "_id": "6361ff4314b08a4853714b69",
         "album_name": "Admiral Bob",
         "cover_photo": "admiral_bob.jpg",
         "artist": "Admiral Bob",
         ***"genre": "Instrumental",***
         ***"year": 2020,***
         "producer": "CCMixters",
         "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
         "album_url": "http://dig.ccmixter.org/people/admiralbob77",
         "createdAt": "2022-11-02T05:25:23.074Z",
         "slug": "admiral-bob",
         "__v": 0
      },
      {
```

```
        "_id": "63627554c1648b56814fc92d",
        "album_name": "Siobhan Dakay",
        "cover_photo": "siobhan_dakay.jpg",
        "artist": "Siobhan Dakay",
        "genre": "Instrumental",
        "year": 2007,
        "producer": "CCMixter",
        "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
        "album_url": "http://dig.ccmixter.org/people/SiobhanD",
        "createdAt": "2022-11-02T13:44:22.349Z",
        "slug": "siobhan-dakay",
        "__v": 0
      }
   ]
}
```


Select

controllers > albums.js

```
…
// @desc    Get all albums
// @route   GET /api/v1/albums
// @access  Public
exports.getAlbums = asyncHandler(async(req, res, next) => {
   let query;

   // copy req.query
   const reqQuery = { ...req.query };

   // fields to exclude
   const removeFields = ['select'];

   // loop over removeFields and remove them from reqQuery
   removeFields.forEach(param => delete reqQuery[param]);

   // create query string
   let queryStr = JSON.stringify(req.query);

   // create query operators ($gt, $gte, etc...)
   queryStr = queryStr.replace(/\b(gt|gte|lt|lte|in)\b/g, match => `$${match}`);

   // const albums = await AlbumModel.find();

   // finding resource
   query = AlbumModel.find(JSON.parse(queryStr));

   // select fields
   if (req.query.select) {
```

```
    const fields = req.query.select.split(',').join(' ');
    query = query.select(fields);
  }

  // executing query
  const albums = await query;

  res
    .status(200)
    .json({
      success: true,
      count: albums.length,
      data: albums
    });
});
…
```

Send GET ‘{{URL}}/api/v1/albums?select=artist,genre’ using Postman

Postman console output


```
{
  "success": true,
  "count": 5,
  "data": [
    {
      "_id": "6361ff4314b08a4853714b69",
      "artist": "Admiral Bob",
      "genre": "Instrumental"
    },
    {
      "_id": "6361ff4314b08a4853714b6a",
      "artist": "Kara Square",
      "genre": "Instrumental"
    },
    {
      "_id": "6361ff4314b08a4853714b6b",
      "artist": "Siobhan Dakay",
      "genre": "Instrumental"
    },
    {
      "_id": "6361ff4314b08a4853714b6c",
      "artist": "Radioontheshelf",
      "genre": "Instrumental"
    },
    {
      "_id": "6361ff4314b08a4853714b68",
      "artist": "Martyn Bloor",
      "genre": "Instrumental"
    }
  ]
```

```
}
```

Send GET '{{URL}}/api/v1/albums?select=artist,genre,year&year[lte]=2010' using Postman

```
{
   "success": true,
   "count": 1,
   "data": [
      {
         "_id": "63627554c1648b56814fc92d",
         "artist": "Siobhan Dakay",
         "genre": "Instrumental",
         "year": 2007
      }
   ]
}
```

Sorting

controllers > albums.js

```
…
// @desc    Get all albums
// @route   GET /api/v1/albums
// @access  Public
exports.getAlbums = asyncHandler(async(req, res, next) => {
   let query;

   // copy req.query
   const reqQuery = { ...req.query };

   // fields to exclude
   const removeFields = ['select', 'sort'];

   // loop over removeFields and remove them from reqQuery
   removeFields.forEach(param => delete reqQuery[param]);

   // create query string
   let queryStr = JSON.stringify(req.query);

   // create query operators ($gt, $gte, etc...)
   queryStr = queryStr.replace(/\b(gt|gte|lt|lte|in)\b/g, match => `$${match}`);

   // const albums = await AlbumModel.find();

   // finding resource
   query = AlbumModel.find(JSON.parse(queryStr));

   // select fields
   if (req.query.select) {
      const fields = req.query.select.split(',').join(' ');
```

```
    query = query.select(fields);
  }

  // sort
  if (req.query.sort) {
    const sortBy = req.query.sort.split(',').join(' ');
    query = query.sort(sortBy);
  } else {
    // default sort by createdAt field if no sort specified
    query = query.sort('-createdAt');
  }

  // executing query
  const albums = await query;

  res
    .status(200)
    .json({
      success: true,
      count: albums.length,
      data: albums
    });
});
…
```

Send GET '{{URL}}/api/v1/albums?select=artist,genre,year&year[lte]=2020&sort=year' using Postman

Postman console output


```
{
  "success": true,
  "count": 2,
  "data": [
    {
      "_id": "63627554c1648b56814fc92d",
      "artist": "Siobhan Dakay",
      "genre": "Instrumental",
      "year": 2007
    },
    {
      "_id": "6361ff4314b08a4853714b69",
      "artist": "Admiral Bob",
      "genre": "Instrumental",
      "year": 2020
    }
  ]
}
```


Send GET '{{URL}}/api/v1/albums?select=artist,genre,year&year[lte]=2020&sort=-year' using Postman

Postman console output


```
{
   "success": true,
   "count": 2,
   "data": [
      {
         "_id": "6361ff4314b08a4853714b69",
         "artist": "Admiral Bob",
         "genre": "Instrumental",
         ***"year": 2020***
      },
      {
         "_id": "63627554c1648b56814fc92d",
         "artist": "Siobhan Dakay",
         "genre": "Instrumental",
         ***"year": 2007***
      }
   ]
}
```


Send GET ‘{{URL}}/api/v1/albums?select=artist,genre,year,createdAt&year[lte]=2020&sort=createdAt’ using Postman

Postman console output


```
{
   "success": true,
   "count": 2,
   "data": [
      {
         "_id": "6361ff4314b08a4853714b69",
         "artist": "Admiral Bob",
         "genre": "Instrumental",
         "year": 2020,
         ***"createdAt": "2022-11-02T05:25:23.074Z"***
      },
      {
         "_id": "63627554c1648b56814fc92d",
         "artist": "Siobhan Dakay",
         "genre": "Instrumental",
         "year": 2007,
         ***"createdAt": "2022-11-02T13:44:22.349Z"***
      }
   ]
}
```


Send GET ‘{{URL}}/api/v1/albums?select=artist,genre,year,createdAt&year[lte]=2020’ using Postman

Postman console output

```
{
  "success": true,
  "count": 2,
  "data": [
    {
      "_id": "63627554c1648b56814fc92d",
      "artist": "Siobhan Dakay",
      "genre": "Instrumental",
      "year": 2007,
      "createdAt": "2022-11-02T13:44:22.349Z"
    },
    {
      "_id": "6361ff4314b08a4853714b69",
      "artist": "Admiral Bob",
      "genre": "Instrumental",
      "year": 2020,
      "createdAt": "2022-11-02T05:25:23.074Z"
    }
  ]
}
```


Pagination

controllers > albums.js

```
…
// @desc    Get all albums
// @route   GET /api/v1/albums
// @access  Public
exports.getAlbums = asyncHandler(async(req, res, next) => {
  let query;

  // copy req.query
  const reqQuery = { ...req.query };

  // fields to exclude
  const removeFields = ['select', 'sort', 'page', 'limit'];

  // loop over removeFields and remove them from reqQuery
  removeFields.forEach(param => delete reqQuery[param]);

  // create query string
  let queryStr = JSON.stringify(req.query);

  // create query operators ($gt, $gte, etc...)
  queryStr = queryStr.replace(/\b(gt|gte|lt|lte|in)\b/g, match => `$${match}`);

  // const albums = await AlbumModel.find();

  // finding resource
  query = AlbumModel.find(JSON.parse(queryStr));
```

```
// select fields
if (req.query.select) {
   const fields = req.query.select.split(',').join(' ');
   query = query.select(fields);
}

// sort
if (req.query.sort) {
   const sortBy = req.query.sort.split(',').join(' ');
   query = query.sort(sortBy);
} else {
   // default sort by createdAt field if no sort specified
   query = query.sort('-createdAt');
}

// pagination
   // parse the page number from query (which is a string) to a number of base 10
   // if no page number is specified, set default as 1
const page = parseInt(req.query.page, 10) || 1;
   // if no limit is specified, set default as 1 resource per page
const limit = parseInt(req.query.limit, 10) || 1;
   // skip some resources (start index from)
const startIndex = (page - 1) * limit;
   // end index at
const endIndex = page * limit;
   // total resources
const total = await AlbumModel.countDocuments();

query = query.skip(startIndex).limit(limit);

// executing query
const albums = await query;

// pagination result
const pagination = {};

if (startIndex > 0) {
   pagination.prev = {
      page: page - 1
   }
}

pagination.curr = {
   page
}

if (endIndex < total) {
   pagination.next = {
      page: page + 1
   }
```

```
  }

  res
    .status(200)
    .json({
      success: true,
      count: albums.length,
      pagination,
      total,
      limit,
      data: albums
    });
});
…
```

Send GET '{{URL}}/api/v1/albums?select=artist' using Postman

Postman console output


```
{
  "success": true,
  "count": 1,
  "pagination": {
    "curr": {
      "page": 1
    },
    "next": {
      "page": 2
    }
  },
  "total": 5,
  "limit": 1,
  "data": [
    {
      "_id": "63627554c1648b56814fc92d",
      "artist": "Siobhan Dakay"
    }
  ]
}
```


Send GET '{{URL}}/api/v1/albums?select=artist&limit=2' using Postman (first page)

Postman console output


```
{
  "success": true,
  "count": 2,
  "pagination": {
    "curr": {
      "page": 1
    },
    "next": {
```

```
        "page": 2
      }
    },
    "total": 5,
    "limit": 2,
    "data": [
      {
        "_id": "63627554c1648b56814fc92d",
        "artist": "Siobhan Dakay"
      },
      {
        "_id": "6361ff4314b08a4853714b69",
        "artist": "Admiral Bob"
      }
    ]
}
```


Send GET '{{URL}}/api/v1/albums?select=artist&limit=2&page=2' using Postman

Postman console output


```
{
    "success": true,
    "count": 2,
    "pagination": {
      "prev": {
        "page": 1
      },
      "curr": {
        "page": 2
      },
      "next": {
        "page": 3
      }
    },
    "total": 5,
    "limit": 2,
    "data": [
      {
        "_id": "6361ff4314b08a4853714b6a",
        "artist": "Kara Square"
      },
      {
        "_id": "6361ff4314b08a4853714b69",
        "artist": "Admiral Bob"
      }
    ]
}
```


Send GET '{{URL}}/api/v1/albums?select=artist&page=5' using Postman (last page)

Postman console output


```
{
   "success": true,
   "count": 1,
   "pagination": {
      "prev": {
         "page": 4
      },
      "curr": {
         "page": 5
      }
   },
   "total": 5,
   "limit": 1,
   "data": [
      {
         "_id": "6361ff4314b08a4853714b6c",
         "artist": "Radioontheshelf"
      }
   ]
}
```


Send GET '{{URL}}/api/v1/albums' using Postman

Postman console output


```
{
   "success": true,
   "count": 1,
   "pagination": {
      "curr": {
         "page": 1
      },
      "next": {
         "page": 2
      }
   },
   "total": 5,
   "limit": 1,
   "data": [
      {
         "_id": "63627554c1648b56814fc92d",
         "album_name": "Siobhan Dakay",
         "cover_photo": "siobhan_dakay.jpg",
         "artist": "Siobhan Dakay",
         "genre": "Instrumental",
         "year": 2007,
         "producer": "CCMixter",
         "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality
```

```
sourced from an open licence music website, http://dig.ccmixter.org",
        "album_url": "http://dig.ccmixter.org/people/SiobhanD",
        "createdAt": "2022-11-02T13:44:22.349Z",
        "slug": "siobhan-dakay",
        "__v": 0
    }
  ]
}
```

Create the Track Model

models > Track.js

```
const mongoose = require('mongoose');
const slugify = require('slugify');

const TrackSchema = new mongoose.Schema({
  track_name: {
    type: String,
    trim: true,
    required: [true, 'Please add a name for the Track']
  },
  track_slug: String,
  featuring: String,
  duration: String,
  track_file: {
    type: String,
    default: 'no-track'
  },
  track_data: String,
  credit: String,
  file_size: String,
  createdAt: {
    type: Date,
    default: Date.now()
  }
});

// create track slug from the name
TrackSchema.pre('save', function(next) {
  this.track_slug = slugify(this.track_name, { lower: true });
  next();
});

module.exports = mongoose.model('Track', TrackSchema);
```

Create a relationship to Album Model from Track Model

Create a relationship to link each track to an album by referencing the album id to be linked

models > Track.js

```
  file_size: String,
  createdAt: {
    type: Date,
    default: Date.now()
  },
  album: {
    type: mongoose.Schema.ObjectId,
    // reference to model for relationship
    ref: 'Album',
    required: true
  }
});
…
```

Database Seeder

tracks.json

```
[
  {
    "track_name": "spacedust",
                "featuring": "mwic",
    "duration": "05:18",
    "track_file": "airtone_-_spacedust_1.mp3",
    "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/64741'>spacedust</a> by airtone (c) copyright 2022
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
    "file_size": "12.14MB",
    "album": "6361ff4314b08a4853714b68"
  },
  {
    "track_name": "bluenotes",
                "featuring": "Admiral Bob",
    "duration": "03:48",
    "track_file": "airtone_-_bluenotes_6.mp3",
    "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/64427'>bluenotes</a> by airtone (c) copyright 2021
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
    "file_size": "8.73MB",
    "album": "6361ff4314b08a4853714b68"
  },
  {
    "track_name": "blackSnow",
                "featuring": "zikweb, SackJo22",
    "duration": "6:00",
    "track_file": "airtone_-_blackSnow_1.mp3",
    "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/63513'>blackSnow</a> by airtone (c) copyright 2021
Licensed under a Creative Commons <a
```

```
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
      "file_size": "13.74MB",
      "album": "6361ff4314b08a4853714b68"
   },
   {
      "track_name": "reNovation",
      "featuring": "mwic",
      "duration": "03:30",
      "track_file": "airtone_-_reNovation_1.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/60674'>reNovation</a> by airtone (c) copyright 2019
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
      "file_size": "8.04MB",
      "album": "6361ff4314b08a4853714b68"
   },
   {
      "track_name": "reCreation",
      "featuring": "mwic",
      "duration": "03:40",
      "track_file": "airtone_-_reCreation_1.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/59721'>reCreation</a> by airtone (c) copyright 2019
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
      "file_size": "8.43MB",
      "album": "6361ff4314b08a4853714b68"
   },
   {
      "track_name": "Wanderer (Take 2)",
      "featuring": "SackJo22",
      "duration": "05:12",
      "track_file": "admiralbob77_-_Wanderer_(Take_2)_1.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/admiralbob77/62202'>Wanderer (Take 2)</a> by Admiral Bob
(c) copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
SackJo22</div>",
      "file_size": "11.91MB",
      "album": "6361ff4314b08a4853714b69"
   },
   {
      "track_name": "I Can't Breathe",
      "featuring": "Subhashish",
      "duration": "08:52",
      "track_file": "admiralbob77_-_I_Can_t_Breathe.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/admiralbob77/61763'>I Can't Breathe</a> by Admiral Bob (c)
copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
Subhashish</div>",
```

```
      "file_size": "12.19MB",
      "album": "6361ff4314b08a4853714b69"
   },
   {
      "track_name": "Victoria",
      "featuring": "Sascha Ende",
      "duration": "04:54",
      "track_file": "admiralbob77_-_Victoria.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/admiralbob77/59769'>Victoria</a> by Admiral Bob (c) copyright
2019 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: Sascha
Ende</div>",
      "file_size": "6.74MB",
      "album": "6361ff4314b08a4853714b69"
   },
   {
      "track_name": "Warm Vacuum Tube",
      "featuring": "starfrosch",
      "duration": "03:32",
      "track_file": "admiralbob77_-_Warm_Vacuum_Tube_.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/admiralbob77/59533'>Warm Vacuum Tube </a> by Admiral Bob
(c) copyright 2019 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
starfrosch</div>",
      "file_size": "8.12MB",
      "album": "6361ff4314b08a4853714b69"
   },
   {
      "track_name": "High Above the Darkness (My Star)",
      "featuring": "Skye Jordan",
      "duration": "03:41",
      "track_file": "admiralbob77_-_High_Above_the_Darkness_(My_Star).mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/admiralbob77/62876'>High Above the Darkness (My Star)</a>
by Admiral Bob (c) copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
      "file_size": "6.76MB",
      "album": "6361ff4314b08a4853714b69"
   },
   {
      "track_name": "Where the Moon Shines Bright",
      "featuring": "Mr_Yesterday, Stefan Kartenberg",
      "duration": "04:28",
      "track_file": "mindmapthat_-_Where_the_Moon_Shines_Bright_1.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/mindmapthat/62227'>Where the Moon Shines Bright</a> by
Kara Square (c) copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
Mr_Yesterday, Stefan Kartenberg</div>",
```

```
      "file_size": "10.24MB",
      "album": "6361ff4314b08a4853714b6a"
   },
   {
      "track_name": "Blue Sky Blues",
      "featuring": "Admiral Bob",
      "duration": "03:03",
      "track_file": "mindmapthat_-_Blue_Sky_Blues.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/mindmapthat/60675'>Blue Sky Blues</a> by Kara Square (c)
copyright 2019 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: Admiral
Bob</div>",
      "file_size": "4.21MB",
      "album": "6361ff4314b08a4853714b6a"
   },
   {
      "track_name": "Transmutation",
      "featuring": "Spinningmerkaba",
      "duration": "02:53",
      "track_file": "mindmapthat_-_Transmutation.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/mindmapthat/56527'>Transmutation</a> by Kara Square (c)
copyright 2017 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
Spinningmerkaba</div>",
      "file_size": "3.98MB",
      "album": "6361ff4314b08a4853714b6a"
   },
   {
      "track_name": "Democrazy",
      "featuring": "reusenoise",
      "duration": "03:12",
      "track_file": "mindmapthat_-_Democrazy.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/mindmapthat/55686'>Democrazy</a> by Kara Square (c)
copyright 2017 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
reusenoise</div>",
      "file_size": "4.40MB",
      "album": "6361ff4314b08a4853714b6a"
   },
   {
      "track_name": "Possibility",
      "featuring": "Admiral Bob",
      "duration": "03:36",
      "track_file": "mindmapthat_-_Possibility.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/mindmapthat/61671'>Possibility</a> by Kara Square (c)
copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: Admiral
```

```
Bob</div>",
      "file_size": "4.96MB",
      "album": "6361ff4314b08a4853714b6a"
   },
   {
      "track_name": "From Gurd To Nark",
      "featuring": "Gurdonark",
      "duration": "05:18",
      "track_file": "SiobhanD_-_From_Gurd_To_Nark_1.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/SiobhanD/64324'>From Gurd To Nark</a> by Siobhan Dakay (c)
copyright 2021 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
Gurdonark</div>",
      "file_size": "12.16MB",
      "album": "63627554c1648b56814fc92d"
   },
   {
      "track_name": "Love is my Road (Back to You)",
      "featuring": "Admiral Bob",
      "duration": "05:52",
      "track_file": "SiobhanD_-_Love_is_my_Road_(Back_to_You).mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/SiobhanD/61885'>Love is my Road (Back to You)</a> by
Siobhan Dakay (c) copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: Admiral
Bob</div>",
      "file_size": "13.46MB",
      "album": "63627554c1648b56814fc92d"
   },
   {
      "track_name": "Good Night",
      "featuring": "MyVanillaworld",
      "duration": "03:27",
      "track_file": "SiobhanD_-_Good_Night.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/SiobhanD/60107'>Good Night</a> by Siobhan Dakay (c)
copyright 2019 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
MyVanillaworld</div>",
      "file_size": "7.91MB",
      "album": "63627554c1648b56814fc92d"
   },
   {
      "track_name": "Unbury Your Heart",
      "featuring": "Madam Snowflake",
      "duration": "04:31",
      "track_file": "SiobhanD_-_Unbury_Your_Heart_1.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/SiobhanD/57378'>Unbury Your Heart</a> by Siobhan Dakay (c)
copyright 2018 Licensed under a Creative Commons <a
```

```
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: Madam
Snowflake</div>",
      "file_size": "10.37MB",
      "album": "63627554c1648b56814fc92d"
   },
   {
      "track_name": "Der Mond ist aufgegangen",
      "featuring": "Maike, Lemon Yellow Hayes",
      "duration": "04:26",
      "track_file": "SiobhanD_-_Der_Mond_ist_aufgegangen.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/SiobhanD/56771'>Der Mond ist aufgegangen</a> by Siobhan
Dakay (c) copyright 2017 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: Maike,
Lemon Yellow Hayes</div>",
      "file_size": "10.16MB",
      "album": "63627554c1648b56814fc92d"
   },
   {
      "track_name": "Yesterdays Blues",
      "featuring": "Mr Yesterday & The Lovely Ladies",
      "duration": "06:22",
      "track_file": "Radioontheshelf_-_Yesterdays_Blues.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/Radioontheshelf/63894'>Yesterdays Blues</a> by
Radioontheshelf (c) copyright 2021 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: Mr
Yesterday & The Lovely Ladies</div>",
      "file_size": "14.59MB",
      "album": "6361ff4314b08a4853714b6c"
   },
   {
      "track_name": "The Day We Went To Tesco's",
      "featuring": "Old Dog Ross",
      "duration": "04:02",
      "track_file": "Radioontheshelf_-_The_Day_We_Went_To_Tesco_s.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/Radioontheshelf/61451'>The Day We Went To Tesco's</a> by
Radioontheshelf (c) copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: Old Dog
Ross</div>",
      "file_size": "9.27MB",
      "album": "6361ff4314b08a4853714b6c"
   },
   {
      "track_name": "The Blacksmiths Horse",
      "featuring": "Skyejordan",
      "duration": "05:04",
      "track_file": "Radioontheshelf_-_The_Blacksmiths_Horse.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/Radioontheshelf/62554'>The Blacksmiths Horse</a> by
```

```
Radioontheshelf (c) copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
Skyejordan</div>",
      "file_size": "11.60MB",
      "album": "6361ff4314b08a4853714b6c"
   },
   {
      "track_name": "Crawling Back To The Sea",
      "featuring": "The Admiral",
      "duration": "04:16",
      "track_file": "Radioontheshelf_-_Crawling_Back_To_The_Sea.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/Radioontheshelf/62365'>Crawling Back To The Sea</a> by
Radioontheshelf (c) copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: The
Admiral</div>",
      "file_size": "9.77MB",
      "album": "6361ff4314b08a4853714b6c"
   },
   {
      "track_name": "Same Destinations",
      "featuring": "The Admiral",
      "duration": "03:27",
      "track_file": "Radioontheshelf_-_Same_Destinations.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/Radioontheshelf/58482'>Same Destinations</a> by
Radioontheshelf (c) copyright 2018 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: The
Admiral</div>",
      "file_size": "7.92MB",
      "album": "6361ff4314b08a4853714b6c"
   }
]
```

seeder.js

```
…
// load models
const Album = require('./models/Album');
const Track = require('./models/Track');
…
// read JSON files
const albums =  JSON.parse(fs.readFileSync(`${__dirname}/_data/albums.json`, 'utf-8'));
const tracks =  JSON.parse(fs.readFileSync(`${__dirname}/_data/tracks.json`, 'utf-8'));

// import into db
const importData = async () => {
   try {
      await Album.create(albums);
      await Track.create(tracks);
```

```
      console.log('data imported...'.green.inverse);
      // exit the process
…
}

// delete from db
const deleteData = async () => {
   try {
      await Album.deleteMany();
      await Track.deleteMany();

      console.log('data deleted...'.red.inverse);
   …
}
…
```

run “node seeder -d” in terminal

console output

data deleted...

send GET all albums in Postman

postman console output


```
{
   "success": true,
   "count": 0,
   "data": []
}
```


run “node seeder -i” in terminal

console output

data imported...

mongo db oms collection

collection name	albums	tracks
documents	5	25

first document of tracks collection


```
{
  "_id": {
    "$oid": "63634dcd406956f24d6fd7a1"
  },
```

```
  "track_name": "reCreation",
  "featuring": "mwic",
  "duration": "03:40",
  "track_file": "airtone_-_reCreation_1.mp3",
  "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/59721'>reCreation</a> by airtone (c) copyright 2019
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
  "file_size": "8.43MB",
  "createdAt": {
    "$date": {
      "$numberLong": "1667452364150"
    }
  },
  "album": {
    "$oid": "6361ff4314b08a4853714b68"
  },
  "slug": "recreation",
  "__v": {
    "$numberInt": "0"
  }
}
```

album matching document

```
{
    "_id": "6361ff4314b08a4853714b68",
    "album_name": "Airtone"
}
```

Create the Track Controller

controllers > tracks.js

```
const TrackModel = require('../models/Track');
const asyncHandler = require('../middleware/async');
const ErrorResponse = require('../utils/errorResponse');

// @desc    Get all tracks
// @route   GET /api/v1/tracks
// @route   GET /api/v1/albums/:albumId/tracks
// @access  Public
exports.getTracks = asyncHandler(async(req, res, next) => {
    let query;

    if (req.params.albumId) {
        // get all tracks for a particular album (by album id)
        query = TrackModel.find({ album: req.params.albumId });
    } else {
        query = TrackModel.find();
    }
```

```
  const tracks = await query;

  res
    .status(200)
    .json({
      success: true,
      count: tracks.length,
      data: tracks
    });
});

// @desc    Create new track
// @route   POST /api/v1/albums/:albumId/tracks
// @access  Private
exports.createTrack = asyncHandler(async(req, res, next) => {
  // set the 'album' field in TrackSchema & request body to the request albumId of params
  req.body.album = req.params.albumId;

  const album = await AlbumModel.findById(req.params.albumId);

  if (!album) {
    return next (new ErrorResponse(`Album with id ${ req.params.albumId } not found`, 404))
  }

  const track = await TrackModel.create(req.body);

  res
    .status(201)
    .json({
      success: true,
      msg: 'Track created successfully'
    });
});

// @desc    Update track by ID
// @route   PUT /api/v1/tracks/:id
// @access  Private
exports.updateTrackById = asyncHandler(async(req, res, next) => {
  let track = await TrackModel.findById(req.params.id);

  // error for correctly formatted id not present in database
  if(!track) {
    return next(new ErrorResponse(`Track with id '${req.params.id}' not found`, 404));
  }

  track = await TrackModel.findByIdAndUpdate(req.params.id, req.body, { new: true,
runValidators: true });

  res
    .status(200)
```

```
      .json({
         success: true,
         msg: `Track with id ${req.params.id} updated successfully`, data: track
      });
});

// @desc    Delete track by ID
// @route   DELETE /api/v1/tracks/:id
// @access  Private
exports.deleteTrackById = asyncHandler(async(req, res, next) => {
   let track = await TrackModel.findById(req.params.id);

   // error for correctly formatted id not present in database
   if(!track) {
      return next(new ErrorResponse(`Track with id '${req.params.id}' not found`, 404));
   }

   await TrackModel.remove();

   res
      .status(200)
      .json({
         success: true,
         msg: `Track with id ${req.params.id} deleted successfully`
      });
});
```

Create the Track Routes

routes > tracks.js

```
const express = require('express');

const { getTracks, createTrack, getTrackById, updateTrackById, deleteTrackById } =
require('../controllers/tracks');

const router = express.Router();

router
   .route('/')
   .get(getTracks)
   .post(createTrack);

router
   .route('/:id')
   .get(getTrackById)
   .put(updateTrackById)
   .delete(deleteTrackById)

module.exports = router;
```

Add the routes files and mount the routes in server.js file

server.js

```
…
// config file
const connectDB = require('./config/database');

// route files
const albums = require('./routes/albums');
const tracks = require('./routes/tracks');
…
// initialize express app
const app = express();

// body-parser
app.use(express.json());
…
app.get('/', (req, res) => {
   res.status(200).json({ success: true, msg: 'Welcome to the online music store' });
});

// mount routers
app.use('/api/v1/albums', albums);
app.use('/api/v1/tracks', tracks);
…
```

Create a Resource Router in Albums Routes

Create a resource router in albums routes to get tracks by album id in tracks routes

routes > albums.js

```
…
const { getAlbums, getAlbumById, createAlbum, updateAlbumById, deleteAlbumById } =
require('../controllers/albums');

// use other resource routers
const trackRouter = require('./tracks');

const router = express.Router();

// re-route to other resource routers
router
   .use('/:albumId/tracks', trackRouter);
…
```

routes > tracks.js

```
…
const { getTracks, createTrack, getTrackById, updateTrackById, deleteTrackById } =
require('../controllers/tracks');

const router = express.Router({ mergeParams: true });
```

…

Consume the APIs using Postman

GET ALL ALBUMS

Send GET {{URL}}/api/v1/tracks

postman console output


```
{
   "success": true,
   ***"count": 25,***
   "data": [
      {
         "_id": "63634dcd406956f24d6fd7a1",
         "track_name": "reCreation",
         "featuring": "mwic",
         "duration": "03:40",
         "track_file": "airtone_-_reCreation_1.mp3",
         "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/59721'>reCreation</a> by airtone (c) copyright 2019
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
         "file_size": "8.43MB",
         "createdAt": "2022-11-03T05:12:44.150Z",
         "album": "6361ff4314b08a4853714b68",
         "slug": "recreation",
         "__v": 0
      },
…
}
```


GET TRACKS BY ALBUM ID

Send GET {{URL}}/api/v1/albums/6361ff4314b08a4853714b68/tracks

Postman console output


```
{
   "success": true,
   "count": 5,
   "data": [
      {
         "_id": "6367457b5af941f71f9cb4a7",
         "track_name": "spacedust",
         "featuring": "mwic",
         "duration": "05:18",
         "track_file": "airtone_-_spacedust_1.mp3",
         "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/64741'>spacedust</a> by airtone (c) copyright 2022
Licensed under a Creative Commons <a
```

```
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
        "file_size": "12.14MB",
        "createdAt": "2022-11-06T05:26:18.673Z",
        ***"album": "6361ff4314b08a4853714b68",***
        "slug": "spacedust",
        "__v": 0
    },
    {
        "_id": "6367457b5af941f71f9cb4aa",
        "track_name": "reNovation",
        "featuring": "mwic",
        "duration": "03:30",
        "track_file": "airtone_-_reNovation_1.mp3",
        "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/60674'>reNovation</a> by airtone (c) copyright 2019
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
        "file_size": "8.04MB",
        "createdAt": "2022-11-06T05:26:18.673Z",
        ***"album": "6361ff4314b08a4853714b68",***
        "slug": "renovation",
        "__v": 0
    },
    {
        "_id": "6367457b5af941f71f9cb4ab",
        "track_name": "reCreation",
        "featuring": "mwic",
        "duration": "03:40",
        "track_file": "airtone_-_reCreation_1.mp3",
        "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/59721'>reCreation</a> by airtone (c) copyright 2019
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
        "file_size": "8.43MB",
        "createdAt": "2022-11-06T05:26:18.673Z",
        ***"album": "6361ff4314b08a4853714b68",***
        "slug": "recreation",
        "__v": 0
    },
    {
        "_id": "6367457b5af941f71f9cb4a9",
        "track_name": "blackSnow",
        "featuring": "zikweb, SackJo22",
        "duration": "6:00",
        "track_file": "airtone_-_blackSnow_1.mp3",
        "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/63513'>blackSnow</a> by airtone (c) copyright 2021
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
        "file_size": "13.74MB",
        "createdAt": "2022-11-06T05:26:18.673Z",
```

```
        "album": "6361ff4314b08a4853714b68",
        "slug": "blacksnow",
        "__v": 0
      },
      {
        "_id": "6367457b5af941f71f9cb4a8",
        "track_name": "bluenotes",
        "featuring": "Admiral Bob",
        "duration": "03:48",
        "track_file": "airtone_-_bluenotes_6.mp3",
        "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/64427'>bluenotes</a> by airtone (c) copyright 2021
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
        "file_size": "8.73MB",
        "createdAt": "2022-11-06T05:26:18.673Z",
        "album": "6361ff4314b08a4853714b68",
        "slug": "bluenotes",
        "__v": 0
      }
    ]
}
```


delete all data using seeder "***node seeder -d***"

CREATE NEW TRACK IN SPECIFIC ALBUM

GET ALBUM BY ID

Send GET {{URL}}/api/v1/albums/6361ff4314b08a4853714b69 using Postman

Postman console output


```
{
    "success": true,
    "data": {
        "_id": "6361ff4314b08a4853714b69",
        "album_name": "Admiral Bob",
        "cover_photo": "admiral_bob.jpg",
        "artist": "Admiral Bob",
        "genre": "Instrumental",
        "year": 2020,
        "producer": "CCMixters",
        "description": "Disclaimer: This album is not meant for sales, promotions or any other
commercial use. It is simply sample music for this website to demonstrate CRUD functionality
sourced from an open licence music website, http://dig.ccmixter.org",
        "album_url": "http://dig.ccmixter.org/people/admiralbob77",
        "createdAt": "2022-11-07T07:12:46.081Z",
        "slug": "admiral-bob",
        "__v": 0,
        "id": "6361ff4314b08a4853714b69"
    }
```

}

GET TRACKS BY ALBUM

Send GET {{URL}}/api/v1/albums/6361ff4314b08a4853714b69/tracks using Postman

Postman console output


```
{
   "success": true,
   "count": 0,
   "data": []
}
```


CREATE TRACK IN SPECIFIC ALBUM

Set key ‘Content-Type’ to value ‘application/json’ in headers and send the following message in POST {{URL}}/api/v1/albums/6361ff4314b08a4853714b69/tracks


```
{
   "track_name": "I Can't Breathe",
   "featuring": "Subhashish",
   "duration": "08:52",
   "track_file": "admiralbob77_-_I_Can_t_Breathe.mp3",
   "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/admiralbob77/61763'>I Can't Breathe</a> by Admiral Bob (c)
copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
Subhashish</div>",
   "file_size": "12.19MB",
   "album": "6361ff4314b08a4853714b69"
}
```


Postman console output


```
{
   "success": true,
   "msg": "Track created successfully"
}
```


GET TRACKS BY ALBUM

Send GET {{URL}}/api/v1/albums/6361ff4314b08a4853714b69/tracks using Postman

Postman console output


```
{
   "success": true,
   "count": 1,
   "data": [
      {
         "_id": "6368c19db5e310dfc74d94b6",
         "track_name": "I Can't Breathe",
```

```
        "featuring": "Subhashish",
        "duration": "08:52",
        "track_file": "admiralbob77_-_I_Can_t_Breathe.mp3",
        "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/admiralbob77/61763'>I Can't Breathe</a> by Admiral Bob (c)
copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
Subhashish</div>",
        "file_size": "12.19MB",
        "createdAt": "2022-11-07T07:12:46.130Z",
        "album": "6361ff4314b08a4853714b69",
        "slug": "i-can't-breathe",
        "__v": 0
      }
   ]
}
```


GET TRACK BY ID

Send GET {{URL}}/api/v1/tracks/63674371637b3b4560a9cb77

Postman Console output


```
{
   "success": true,
   "data": {
      "_id": "63674371637b3b4560a9cb77",
      "track_name": "spacedust",
      "featuring": "mwic",
      "duration": "05:18",
      "track_file": "airtone_-_spacedust_1.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/64741'>spacedust</a> by airtone (c) copyright 2022
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
      "file_size": "12.14MB",
      "createdAt": "2022-11-06T05:11:35.694Z",
      "album": "6361ff4314b08a4853714b68",
      "slug": "spacedust",
      "__v": 0
   }
}
```


UPDATE TRACK BY ID

Set key ‘Content-Type’ to value ‘application/json’ in headers and send the following message in PUT {{URL}}/api/v1/tracks/63674371637b3b4560a9cb77


```
{
   "file_size": "17.55MB"
}
```

Postman console output


```
{
   "success": true,
   "msg": "Track with id 63674371637b3b4560a9cb77 updated successfully",
   "data": {
      "_id": "63674371637b3b4560a9cb77",
      "track_name": "spacedust",
      "featuring": "mwic",
      "duration": "05:18",
      "track_file": "airtone_-_spacedust_1.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/64741'>spacedust</a> by airtone (c) copyright 2022
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
      "file_size": "17.55MB",
      "createdAt": "2022-11-06T05:11:35.694Z",
      "album": "6361ff4314b08a4853714b68",
      "slug": "spacedust",
      "__v": 0
   }
}
```


DELETE TRACK BY ID

Send DELETE {{URL}}/api/v1/tracks/63674371637b3b4560a9cb77

Postman console output


```
{
   "success": true,
   "msg": "Album with id 63674371637b3b4560a9cb77 deleted successfully"
}
```


Send again GET {{URL}}/api/v1/tracks

Postman console output


```
{
   "success": true,
   "count": 0,
   "data": []
}
```


Populate

Populate the tracks data with data from albums

controllers > tracks.js

```
…
// @desc    Get all tracks
// @route   GET /api/v1/tracks
```

```
// @route   GET /api/v1/albums/:albumId/tracks
// @access  Public
exports.getTracks = asyncHandler(async(req, res, next) => {
   let query;

   if (req.params.albumId) {
      // get all tracks for a particular album (by album id)
      query = TrackModel.find({ album: req.params.albumId });
   } else {
      query = TrackModel.find().populate('album');
   }

   const tracks = await query;

   res
      .status(200)
      .json({
         success: true,
         count: tracks.length,
         data: tracks
      });
});
…
```

Send GET {{URL}}/api/v1/tracks using Postman

Postman console output


```
{
   "success": true,
   "count": 25,
   "data": [
      {
         "_id": "6367457b5af941f71f9cb4a7",
         "track_name": "spacedust",
         "featuring": "mwic",
         "duration": "05:18",
         "track_file": "airtone_-_spacedust_1.mp3",
         "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/64741'>spacedust</a> by airtone (c) copyright 2022
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
         "file_size": "12.14MB",
         "createdAt": "2022-11-06T05:26:18.673Z",
         "album": {
            "_id": "6361ff4314b08a4853714b68",
            "album_name": "Airtone",
            "cover_photo": "airtone.jpeg",
            "artist": "Martyn Bloor",
            "genre": "Instrumental",
            "year": 2022,
```

```
            "producer": "CCMixter",
            "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
            "album_url": "http://dig.ccmixter.org/people/airtone",
            "createdAt": "2022-11-06T05:26:18.656Z",
            "slug": "airtone",
            "__v": 0
        },
        "slug": "spacedust",
        "__v": 0
    },
…
}
```

Alternatively, if only specific fields from album are to be populated in tracks

controllers > tracks.js

```
…
// @desc    Get all tracks
// @route   GET /api/v1/tracks
// @route   GET /api/v1/albums/:albumId/tracks
// @access  Public
exports.getTracks = asyncHandler(async(req, res, next) => {
  let query;

  if (req.params.albumId) {
    // get all tracks for a particular album (by album id)
    query = TrackModel.find({ album: req.params.albumId });
  } else {
    query = TrackModel.find().populate({ path: 'album', select: 'album_name album_url createdAt' });
  }

  const tracks = await query;

  res
    .status(200)
    .json({
      success: true,
      count: tracks.length,
      data: tracks
    });
});
…
```

Send GET {{URL}}/api/v1/tracks using Postman

Postman console output

```
{
```

```
  "success": true,
  "count": 25,
  "data": [
    {
      "_id": "6367457b5af941f71f9cb4a7",
      "track_name": "spacedust",
      "featuring": "mwic",
      "duration": "05:18",
      "track_file": "airtone_-_spacedust_1.mp3",
      "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/64741'>spacedust</a> by airtone (c) copyright 2022
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
      "file_size": "12.14MB",
      "createdAt": "2022-11-06T05:26:18.673Z",
      "album": {
        "_id": "6361ff4314b08a4853714b68",
        "album_name": "Airtone",
        "album_url": "http://dig.ccmixter.org/people/airtone",
        "createdAt": "2022-11-06T05:26:18.656Z"
      },
      "slug": "spacedust",
      "__v": 0
    },
…
}
```


Virtuals (Reverse Populate)

In each album, display an array of tracks

models > Album.js

```
…
  album_url: String,
  createdAt: {
    type: Date,
    default: Date.now()
  }
}, {
  toJSON: { virtuals: true },
  toObject: { virtuals: true }
});

// create album slug from the name
AlbumSchema.pre('save', function(next) {
  this.slug = slugify(this.album_name, { lower: true });
  next();
});

// reverse populate with virtuals
AlbumSchema.virtual('tracks', {
```

```
  ref: 'Track',
  localField: '_id',
  foreignField: 'album',
  justOne: false
});

module.exports = mongoose.model('Album', AlbumSchema);
```

controllers > albums.js

```
…
// @desc    Get all albums
// @route   GET /api/v1/albums
// @access  Public
exports.getAlbums = asyncHandler(async(req, res, next) => {
…
  // create query operators ($gt, $gte, etc...)
  queryStr = queryStr.replace(/\b(gt|gte|lt|lte|in)\b/g, match => `$${match}`);

  // const albums = await AlbumModel.find();

  // finding resource
  query = AlbumModel.find(JSON.parse(queryStr)).populate('tracks');
…
```

Send GET {{URL}}/api/v1/albums using Postman

Postman console output


```
{
  "success": true,
  "count": 1,
  "pagination": {
    "curr": {
      "page": 1
    },
    "next": {
      "page": 2
    }
  },
  "total": 5,
  "limit": 1,
  "data": [
    {
      "_id": "6361ff4314b08a4853714b69",
      "album_name": "Admiral Bob",
      "cover_photo": "admiral_bob.jpg",
      "artist": "Admiral Bob",
      "genre": "Instrumental",
      "year": 2020,
      "producer": "CCMixters",
```

"description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
"album_url": "http://dig.ccmixter.org/people/admiralbob77",
"createdAt": "2022-11-06T05:26:18.656Z",
"slug": "admiral-bob",
"__v": 0,
"tracks": [
{
"_id": "6367457b5af941f71f9cb4ac",
"track_name": "Wanderer (Take 2)",
"featuring": "SackJo22",
"duration": "05:12",
"track_file": "admiralbob77_-_Wanderer_(Take_2)_1.mp3",
"credit": " Wanderer (Take 2) by Admiral Bob (c) copyright 2020 Licensed under a Creative Commons Attribution (3.0) license. Ft: SackJo22 ",
"file_size": "11.91MB",
"createdAt": "2022-11-06T05:26:18.673Z",
"album": "6361ff4314b08a4853714b69",
"slug": "wanderer-(take-2)",
"__v": 0
},
{
"_id": "6367457b5af941f71f9cb4ad",
"track_name": "I Can't Breathe",
"featuring": "Subhashish",
"duration": "08:52",
"track_file": "admiralbob77_-_I_Can_t_Breathe.mp3",
"credit": " I Can't Breathe by Admiral Bob (c) copyright 2020 Licensed under a Creative Commons Attribution (3.0) license. Ft: Subhashish ",
"file_size": "12.19MB",
"createdAt": "2022-11-06T05:26:18.673Z",
"album": "6361ff4314b08a4853714b69",
"slug": "i-can't-breathe",
"__v": 0
},
{
"_id": "6367457b5af941f71f9cb4ae",
"track_name": "Victoria",
"featuring": "Sascha Ende",
"duration": "04:54",
"track_file": "admiralbob77_-_Victoria.mp3",
"credit": " Victoria by Admiral Bob (c) copyright 2019 Licensed under a Creative Commons <a

```
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: Sascha
Ende</div>",
            "file_size": "6.74MB",
            "createdAt": "2022-11-06T05:26:18.673Z",
            "album": "6361ff4314b08a4853714b69",
            "slug": "victoria",
            "__v": 0
          },
          {
            "_id": "6367457b5af941f71f9cb4af",
            "track_name": "Warm Vacuum Tube",
            "featuring": "starfrosch",
            "duration": "03:32",
            "track_file": "admiralbob77_-_Warm_Vacuum_Tube_.mp3",
            "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/admiralbob77/59533'>Warm Vacuum Tube </a> by Admiral
Bob (c) copyright 2019 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
starfrosch</div>",
            "file_size": "8.12MB",
            "createdAt": "2022-11-06T05:26:18.673Z",
            "album": "6361ff4314b08a4853714b69",
            "slug": "warm-vacuum-tube",
            "__v": 0
          },
          {
            "_id": "6367457b5af941f71f9cb4b0",
            "track_name": "High Above the Darkness (My Star)",
            "featuring": "Skye Jordan",
            "duration": "03:41",
            "track_file": "admiralbob77_-_High_Above_the_Darkness_(My_Star).mp3",
            "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/admiralbob77/62876'>High Above the Darkness (My
Star)</a> by Admiral Bob (c) copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
            "file_size": "6.76MB",
            "createdAt": "2022-11-06T05:26:18.673Z",
            "album": "6361ff4314b08a4853714b69",
            "slug": "high-above-the-darkness-(my-star)",
            "__v": 0
          }
        ],
        "id": "6361ff4314b08a4853714b69"
      }
    ]
}
```


Alternatively, if only specific fields from tracks are to be reverse populated in albums

controllers > albums.js

```
…
```

```
// @desc    Get all albums
// @route   GET /api/v1/albums
// @access  Public
exports.getAlbums = asyncHandler(async(req, res, next) => {
…
   // create query operators ($gt, $gte, etc...)
   queryStr = queryStr.replace(/\b(gt|gte|lt|lte|in)\b/g, match => `$${match}`);

   // const albums = await AlbumModel.find();

   // finding resource
   query = AlbumModel.find(JSON.parse(queryStr)).populate({ path: 'tracks', select: 'track_name
featuring duration file_size' });
…
```

Send GET {{URL}}/api/v1/albums using Postman

Postman console output


```
{
   "success": true,
   "count": 1,
   "pagination": {
      "curr": {
         "page": 1
      },
      "next": {
         "page": 2
      }
   },
   "total": 5,
   "limit": 1,
   "data": [
      {
         "_id": "6361ff4314b08a4853714b69",
         "album_name": "Admiral Bob",
         "cover_photo": "admiral_bob.jpg",
         "artist": "Admiral Bob",
         "genre": "Instrumental",
         "year": 2020,
         "producer": "CCMixters",
         "description": "Disclaimer: This album is not meant for sales, promotions or any other
commercial use. It is simply sample music for this website to demonstrate CRUD functionality
sourced from an open licence music website, http://dig.ccmixter.org",
         "album_url": "http://dig.ccmixter.org/people/admiralbob77",
         "createdAt": "2022-11-06T05:26:18.656Z",
         "slug": "admiral-bob",
         "__v": 0,
         "tracks": [
            {
               "_id": "6367457b5af941f71f9cb4ac",
```

```
            "track_name": "Wanderer (Take 2)",
            "featuring": "SackJo22",
            "duration": "05:12",
            "file_size": "11.91MB",
            "album": "6361ff4314b08a4853714b69"
          },
          {
            "_id": "6367457b5af941f71f9cb4ad",
            "track_name": "I Can't Breathe",
            "featuring": "Subhashish",
            "duration": "08:52",
            "file_size": "12.19MB",
            "album": "6361ff4314b08a4853714b69"
          },
          {
            "_id": "6367457b5af941f71f9cb4ae",
            "track_name": "Victoria",
            "featuring": "Sascha Ende",
            "duration": "04:54",
            "file_size": "6.74MB",
            "album": "6361ff4314b08a4853714b69"
          },
          {
            "_id": "6367457b5af941f71f9cb4af",
            "track_name": "Warm Vacuum Tube",
            "featuring": "starfrosch",
            "duration": "03:32",
            "file_size": "8.12MB",
            "album": "6361ff4314b08a4853714b69"
          },
          {
            "_id": "6367457b5af941f71f9cb4b0",
            "track_name": "High Above the Darkness (My Star)",
            "featuring": "Skye Jordan",
            "duration": "03:41",
            "file_size": "6.76MB",
            "album": "6361ff4314b08a4853714b69"
          }
        ],
        "id": "6361ff4314b08a4853714b69"
      }
    ]
}
```


Cascade Delete

If an album is deleted, all the associated tracks in that album should get deleted

models > Album.js
…

```
// reverse populate with virtuals
AlbumSchema.virtual('tracks', {
   ref: 'Track',
   localField: '_id',
   foreignField: 'album',
   justOne: false
});

// cascade delete tracks when an album is deleted
AlbumSchema.pre('remove', async function (next) {
   await this.model('Track').deleteMany({ album: this._id });
   next();
});

module.exports = mongoose.model('Album', AlbumSchema);
```

controllers > albums.js

```
…
// @desc    Delete an album
// @route   DELETE /api/v1/albums/:id
// @access  Private
exports.deleteAlbumById = asyncHandler(async(req, res, next) => {
   const album = await AlbumModel.findById(req.params.id);

   // error for correctly formatted id not present in database
   if(!album) {
      return next(new ErrorResponse(`Album with id '${req.params.id}' not found`, 404));
   }

   album.remove();

   res
      .status(200)
      .json({
         success: true,
         msg: `Album with id '${req.params.id}' deleted successfully`
      });
});
```

Send DELETE {{URL}}/api/v1/albums/6361ff4314b08a4853714b69 using Postman

Postman console output

```
{
   "success": true,
   "msg": "Album with id '6361ff4314b08a4853714b69' deleted successfully"
}
```

Send GET {{URL}}/api/v1/tracks using Postman

Postman console output

```
{
   "success": true,
   "count": 20,
   "data": [
      {
         "_id": "6367457b5af941f71f9cb4a7",
         "track_name": "spacedust",
         "featuring": "mwic",
         "duration": "05:18",
         "track_file": "airtone_-_spacedust_1.mp3",
         "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/airtone/64741'>spacedust</a> by airtone (c) copyright 2022
Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. </div>",
         "file_size": "12.14MB",
         "createdAt": "2022-11-06T05:26:18.673Z",
         "album": {
            "_id": "6361ff4314b08a4853714b68",
            "album_name": "Airtone",
            "album_url": "http://dig.ccmixter.org/people/airtone",
            "createdAt": "2022-11-06T05:26:18.656Z",
            "id": "6361ff4314b08a4853714b68"
         },
         "slug": "spacedust",
         "__v": 0
      },
…
}
```


5 tracks have been deleted, from the deleted album

Cover Photo Upload

Upload cover photos using express-fileupload. Install using npm

npm i express-fileupload

config.env

```
…
PHOTO_UPLOAD_PATH=./public/uploads/albums
MAX_PHOTO_UPLOAD=2*1024*1024
```

Create a folder ‘public’, inside create a folder ‘uploads’, inside create a folder ‘images’. The cover photos will be moved inside the images folder upon upload.

Set the public folder as static so that it can be accessed by browser

server.js

```
// import the necessary modules
const express = require('express');
const path = require('path');
const dotenv = require('dotenv');
```

```
const morgan = require('morgan');
const fileupload = require('express-fileupload');
…
// dev logging middleware: use in development mode
if(process.env.NODE_ENV === 'development'){
   app.use(morgan('dev'));
}

// file upload
app.use(fileupload());

// set public folder as static folder
app.use(express.static(path.join(__dirname,'public')));
…
```

controllers > albums.js

```
const fs = require('fs');
const path = require('path');
const AlbumModel = require('../models/Album');
const asyncHandler = require('../middleware/async');
const ErrorResponse = require('../utils/errorResponse');
…
// @desc    Upload a cover photo for album
// @route   PUT /api/v1/albums/:id/photo
// @access  Private
exports.albumPhotoUpload = asyncHandler(async(req, res, next) => {
   const album = await AlbumModel.findById(req.params.id);

   // error for correctly formatted id not present in database
   if(!album) {
      return next(new ErrorResponse(`Album with id '${req.params.id}' not found`, 404));
   }

   if(!req.files) {
      return next(new ErrorResponse(`Kindly upload a file`, 400));
   }

   const file = req.files.photo;

   // ensure the image is a photo
   if(!file.mimetype.startsWith('image')) {
      return next(new ErrorResponse(`Kindly upload an image file`, 400));
   }

   let max_photo_size = process.env.MAX_PHOTO_UPLOAD.split('*')[0] *
process.env.MAX_PHOTO_UPLOAD.split('*')[1] * process.env.MAX_PHOTO_UPLOAD.split('*')[2];
   let mbs = process.env.MAX_PHOTO_UPLOAD.split('*')[1] *
process.env.MAX_PHOTO_UPLOAD.split('*')[2];

   // check file size (should be less than MAX_PHOTO_UPLOAD in config.env)
```

```
  if(file.size > max_photo_size) {
    return next(new ErrorResponse(`The uploaded image exceeds ${max_photo_size/mbs}MB.
Kindly upload a smaller image.`, 400));
  }

  // create custom filename
  file.name = `cover_photo_${album.slug}${path.parse(file.name).ext}`;

  // create folder with artist-album name if it doesn't exist
  const folder_path =
`${process.env.PHOTO_UPLOAD_PATH}/${album.artist_slug}-${album.album_slug}`;
  if(!fs.existsSync(folder_path)){
    fs.mkdir(`${folder_path}`, async err => {
      if(err) {
        return next(new ErrorResponse(`Problem with folder creation. ${err}`, 500));
      }
    });
  }

  // upload the file
  file.mv(`${folder_path}/${file.name}`, async err => {
    if(err) {
      return next(new ErrorResponse(`Problem with file upload. ${err}`, 500));
    }

    await AlbumModel.findByIdAndUpdate(req.params.id, { cover_photo: file.name })

    res
      .status(200)
      .json({
        success: true,
        msg: `Cover photo uploaded to album with id '${req.params.id}' successfully`,
        data: file.name
      });
  })
});
```

routes > albums.js

```
…
router
  .route('/:id')
  .get(getAlbumById)
  .put(updateAlbumById)
  .delete(deleteAlbumById);

router
  .route('/:id/photo')
  .put(albumPhotoUpload);

module.exports = router;
```

Upload photo in Postman

Send form data of type ‘file’ and select the image from local system in Postman. Send PUT {{URL}}/api/v1/albums/6361ff4314b08a4853714b69/photo

Postman console output


```
{
   "success": true,
   "msg": "Cover photo uploaded to album with id '6361ff4314b08a4853714b69' successfully",
   "data": "cover_photo_admiral-bob.jpg"
}
```


Send GET {{URL}}/api/v1/albums/6361ff4314b08a4853714b69

Postman console output


```
{
   "success": true,
   "data": {
      "_id": "6361ff4314b08a4853714b69",
      "album_name": "Admiral Bob",
      "cover_photo": "cover_photo_admiral-bob.jpg",
      "artist": "Admiral Bob",
      "genre": "Instrumental",
      "year": 2020,
      "producer": "CCMixters",
      "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
      "album_url": "http://dig.ccmixter.org/people/admiralbob77",
      "createdAt": "2022-11-07T07:12:46.081Z",
      "slug": "admiral-bob",
      "__v": 0,
      "id": "6361ff4314b08a4853714b69"
   }
}
```


Track Files Upload

Upload the track files to the ‘tracks’ in the ‘uploads’ folder. Also send each track as a buffer string to the database.

config.env

```
…
TRACK_UPLOAD_PATH=./public/uploads/tracks
MAX_TRACK_UPLOAD=25*1024*1024
```

Create a ‘tracks’ folder inside the ‘uploads’ folder of the ‘public’ folder. The tracks will be moved inside the tracks folder upon upload.

controllers > tracks.js

```
const path = require('path');
const TrackModel = require('../models/Track');
const asyncHandler = require('../middleware/async');
const ErrorResponse = require('../utils/errorResponse');
const AlbumModel = require('../models/Album');
…
// @desc    Upload a cover photo for album
// @route   PUT /api/v1/albums/:id/photo
// @access  Private
exports.albumPhotoUpload = asyncHandler(async(req, res, next) => {
   const album = await AlbumModel.findById(req.params.id);

   // error for correctly formatted id not present in database
   if(!album) {
      return next(new ErrorResponse(`Album with id '${req.params.id}' not found`, 404));
   }

   if(!req.files) {
      return next(new ErrorResponse(`Kindly upload a file`, 400));
   }

   const file = req.files.track;

   // ensure the image is audio
   if(!file.mimetype.startsWith('audio')) {
      return next(new ErrorResponse(`Kindly upload an audio file`, 400));
   }

   let max_photo_size = process.env.MAX_PHOTO_UPLOAD.split('*')[0] *
process.env.MAX_PHOTO_UPLOAD.split('*')[1] * process.env.MAX_PHOTO_UPLOAD.split('*')[2];
   let mbs = process.env.MAX_PHOTO_UPLOAD.split('*')[1] *
process.env.MAX_PHOTO_UPLOAD.split('*')[2];

   // check file size (should be less than MAX_PHOTO_UPLOAD in config.env)
   if(file.size > max_photo_size) {
      return next(new ErrorResponse(`The uploaded image exceeds ${max_photo_size/mbs}MB.
Kindly upload a smaller image.`, 400));
   }

   // create custom filename
   file.name = `cover_photo_${album.slug}${path.parse(file.name).ext}`;

   // upload the file
   file.mv(`${process.env.PHOTO_UPLOAD_PATH}/${file.name}`, async err => {
      if(err) {
         return next(new ErrorResponse(`Problem with file upload. ${err}`, 500));
      }

      await AlbumModel.findByIdAndUpdate(req.params.id, { cover_photo: file.name })

      res
```

```
        .status(200)
        .json({
          success: true,
          msg: `Cover photo uploaded to album with id '${req.params.id}' successfully`,
          data: file.name
        });
    })
});
```

routes > tracks.js

```
…
router
  .route('/:id')
  .get(getTrackById)
  .put(updateTrackById)
  .delete(deleteTrackById);

router
  .route('/:id/audio')
  .put(trackAudioUpload);
…
```

Send PUT {{URL}}/api/v1/tracks/636a1c6e39ae92284e312e1a/audio using Postman

Postman console output

```
{
  "success": true,
  "msg": "Track uploaded to track with id '636a1c6e39ae92284e312e1a' successfully",
  "data": "admiral-bob_-_warm-vacuum-tube.mp3"
}
```

Advanced Results Middleware

Move the advanced results (filtering, select, pagination, etc) to a middleware file which can be used by other controllers

middleware > advancedResults.js

```
const advancedResults = (model, populate) => async (req, res, next) => {
  let query;

  // copy req.query
  const reqQuery = { ...req.query };

  // fields to exclude
  const removeFields = ['select', 'sort', 'page', 'limit'];

  // loop over removeFields and remove them from reqQuery
  removeFields.forEach(param => delete reqQuery[param]);
```

```
// create query string
let queryStr = JSON.stringify(req.query);

// create query operators ($gt, $gte, etc...)
queryStr = queryStr.replace(/\b(gt|gte|lt|lte|in)\b/g, match => `$${match}`);

// finding resource
query = model.find(JSON.parse(queryStr));

// populate
if(populate) {
   query.populate(populate);
}

// select fields
if (req.query.select) {
   const fields = req.query.select.split(',').join(' ');
   query = query.select(fields);
}

// sort
if (req.query.sort) {
   const sortBy = req.query.sort.split(',').join(' ');
   query = query.sort(sortBy);
} else {
   // default sort by createdAt field if no sort specified
   query = query.sort('-createdAt');
}

// pagination
   // parse the page number from query (which is a string) to a number of base 10
   // if no page number is specified, set default as 1
const page = parseInt(req.query.page, 10) || 1;
   // if no limit is specified, set default as 1 resource per page
const limit = parseInt(req.query.limit, 10) || 1;
   // skip some resources (start index from)
const startIndex = (page - 1) * limit;
   // end index at
const endIndex = page * limit;
   // total resources
const total = await model.countDocuments();

query = query.skip(startIndex).limit(limit);

// executing query
const results = await query;

// pagination result
const pagination = {};

if (startIndex > 0) {
```

```
    pagination.prev = {
      page: page - 1
    }
  }

  pagination.curr = {
    page
  }

  if (endIndex < total) {
    pagination.next = {
      page: page + 1
    }
  }

  res.advancedResults = {
    success: true,
    count: results.length,
    pagination,
    total,
    limit,
    data: results
  }

  next();
}

module.exports = advancedResults;
```

routes > albums.js

```
…
// use other resource routers
const trackRouter = require('./tracks');

// use advancedResults middleware with album model
const AlbumModel = require('../models/Album');
const advancedResults = require('../middleware/advancedResults');
…
router
  .route('/')
  .get(advancedResults(AlbumModel, {path: 'tracks', select: 'track_name featuring duration
file_size'}), getAlbums)
  .post(createAlbum);
…
```

controllers > albums.js

```
…
// @desc    Get all albums
// @route   GET /api/v1/albums
```

```
// @access  Public
exports.getAlbums = asyncHandler(async(req, res, next) => {

   res
      .status(200)
      .json(res.avancedresults);
});
…
```

Send GET {{URL}}/api/v1/albums?limit=5&select=album_name,cover_photo using Postman

Postman console output


```
{
   "success": true,
   "count": 5,
   "pagination": {
      "curr": {
         "page": 1
      }
   },
   "total": 5,
   "limit": 5,
   "data": [
      {
         "_id": "6361ff4314b08a4853714b6c",
         "album_name": "Radioontheshelf",
         "cover_photo": "Radioontheshelf.jpg",
         "tracks": [
            {
               "_id": "6369fb35313de993fa2734c5",
               "track_name": "Yesterdays Blues",
               "featuring": "Mr Yesterday & The Lovely Ladies",
               "duration": "06:22",
               "file_size": "14.59MB",
               "album": "6361ff4314b08a4853714b6c"
            },
            {
               "_id": "6369fb35313de993fa2734c6",
               "track_name": "The Day We Went To Tesco's",
               "featuring": "Old Dog Ross",
               "duration": "04:02",
               "file_size": "9.27MB",
               "album": "6361ff4314b08a4853714b6c"
            },
            {
               "_id": "6369fb35313de993fa2734c7",
               "track_name": "The Blacksmiths Horse",
               "featuring": "Skyejordan",
               "duration": "05:04",
               "file_size": "11.60MB",
```

```
            "album": "6361ff4314b08a4853714b6c"
          },
          {
            "_id": "6369fb35313de993fa2734c8",
            "track_name": "Crawling Back To The Sea",
            "featuring": "The Admiral",
            "duration": "04:16",
            "file_size": "9.77MB",
            "album": "6361ff4314b08a4853714b6c"
          },
          {
            "_id": "6369fb35313de993fa2734c9",
            "track_name": "Same Destinations",
            "featuring": "The Admiral",
            "duration": "03:27",
            "file_size": "7.92MB",
            "album": "6361ff4314b08a4853714b6c"
          }
        ],
        "id": "6361ff4314b08a4853714b6c"
      },
      …
  ]
}
```

Apply the advanced results middleware to tracks

routes > tracks.js

```
…
// use advancedResults middleware with track model
const TrackModel = require('../models/Track');
const advancedResults = require('../middleware/advancedResults');
…
router
  .route('/')
  .get(advancedResults(TrackModel, { path: 'album', select: 'album_name album_url createdAt' }),
getTracks)
  .post(createTrack);
```

controllers > tracks.js

```
…
// @desc    Get all tracks
// @route   GET /api/v1/tracks
// @route   GET /api/v1/albums/:albumId/tracks
// @access  Public
exports.getTracks = asyncHandler(async(req, res, next) => {
  if (req.params.albumId) {
    // get all tracks for a particular album (by album id)
    const tracks = await TrackModel.find({ album: req.params.albumId });
```

```
    res
    .status(200)
    .json({
      success: true,
      count: tracks.length,
      data: tracks
    });
  } else {
    res
      .status(200)
      .json(res.advancedResults);
  }
});
…
```

Send GET {{URL}}/api/v1/tracks?limit=10&select=track_name,featuring,duration,file_size using Postman

Postman console output


```
{
  "success": true,
  "count": 10,
  "pagination": {
    "curr": {
      "page": 1
    },
    "next": {
      "page": 2
    }
  },
  "total": 25,
  "limit": 10,
  "data": [
    {
      "_id": "6369fb35313de993fa2734bc",
      "track_name": "Blue Sky Blues",
      "featuring": "Admiral Bob",
      "duration": "03:03",
      "file_size": "4.21MB",
      "album": {
        "_id": "6361ff4314b08a4853714b6a",
        "album_name": "Mindmapthat",
        "album_url": "http://dig.ccmixter.org/people/mindmapthat",
        "createdAt": "2022-11-08T06:46:12.273Z",
        "id": "6361ff4314b08a4853714b6a"
      }
    },
    {
      "_id": "6369fb35313de993fa2734bb",
```

```
    "track_name": "Where the Moon Shines Bright",
    "featuring": "Mr_Yesterday, Stefan Kartenberg",
    "duration": "04:28",
    "file_size": "10.24MB",
    "album": {
      "_id": "6361ff4314b08a4853714b6a",
      "album_name": "Mindmapthat",
      "album_url": "http://dig.ccmixter.org/people/mindmapthat",
      "createdAt": "2022-11-08T06:46:12.273Z",
      "id": "6361ff4314b08a4853714b6a"
    }
  },
  {
    "_id": "6369fb35313de993fa2734b7",
    "track_name": "I Can't Breathe",
    "featuring": "Subhashish",
    "duration": "08:52",
    "file_size": "12.19MB",
    "album": {
      "_id": "6361ff4314b08a4853714b69",
      "album_name": "Admiral Bob",
      "album_url": "http://dig.ccmixter.org/people/admiralbob77",
      "createdAt": "2022-11-08T06:46:12.273Z",
      "id": "6361ff4314b08a4853714b69"
    }
  },
  {
    "_id": "6369fb35313de993fa2734ba",
    "track_name": "High Above the Darkness (My Star)",
    "featuring": "Skye Jordan",
    "duration": "03:41",
    "file_size": "6.76MB",
    "album": {
      "_id": "6361ff4314b08a4853714b69",
      "album_name": "Admiral Bob",
      "album_url": "http://dig.ccmixter.org/people/admiralbob77",
      "createdAt": "2022-11-08T06:46:12.273Z",
      "id": "6361ff4314b08a4853714b69"
    }
  },
  {
    "_id": "6369fb35313de993fa2734b8",
    "track_name": "Victoria",
    "featuring": "Sascha Ende",
    "duration": "04:54",
    "file_size": "6.74MB",
    "album": {
      "_id": "6361ff4314b08a4853714b69",
      "album_name": "Admiral Bob",
      "album_url": "http://dig.ccmixter.org/people/admiralbob77",
      "createdAt": "2022-11-08T06:46:12.273Z",
```

```
      "id": "6361ff4314b08a4853714b69"
    }
  },
  {
    "_id": "6369fb35313de993fa2734b6",
    "track_name": "Wanderer (Take 2)",
    "featuring": "SackJo22",
    "duration": "05:12",
    "file_size": "11.91MB",
    "album": {
      "_id": "6361ff4314b08a4853714b69",
      "album_name": "Admiral Bob",
      "album_url": "http://dig.ccmixter.org/people/admiralbob77",
      "createdAt": "2022-11-08T06:46:12.273Z",
      "id": "6361ff4314b08a4853714b69"
    }
  },
  {
    "_id": "6369fb35313de993fa2734b4",
    "track_name": "reNovation",
    "featuring": "mwic",
    "duration": "03:30",
    "file_size": "8.04MB",
    "album": {
      "_id": "6361ff4314b08a4853714b68",
      "album_name": "Airtone",
      "album_url": "http://dig.ccmixter.org/people/airtone",
      "createdAt": "2022-11-08T06:46:12.273Z",
      "id": "6361ff4314b08a4853714b68"
    }
  },
  {
    "_id": "6369fb35313de993fa2734b1",
    "track_name": "spacedust",
    "featuring": "mwic",
    "duration": "05:18",
    "file_size": "12.14MB",
    "album": {
      "_id": "6361ff4314b08a4853714b68",
      "album_name": "Airtone",
      "album_url": "http://dig.ccmixter.org/people/airtone",
      "createdAt": "2022-11-08T06:46:12.273Z",
      "id": "6361ff4314b08a4853714b68"
    }
  },
  {
    "_id": "6369fb35313de993fa2734b5",
    "track_name": "reCreation",
    "featuring": "mwic",
    "duration": "03:40",
    "file_size": "8.43MB",
```

```
        "album": {
          "_id": "6361ff4314b08a4853714b68",
          "album_name": "Airtone",
          "album_url": "http://dig.ccmixter.org/people/airtone",
          "createdAt": "2022-11-08T06:46:12.273Z",
          "id": "6361ff4314b08a4853714b68"
        }
      },
      {
        "_id": "6369fb35313de993fa2734b9",
        "track_name": "Warm Vacuum Tube",
        "featuring": "starfrosch",
        "duration": "03:32",
        "file_size": "8.12MB",
        "album": {
          "_id": "6361ff4314b08a4853714b69",
          "album_name": "Admiral Bob",
          "album_url": "http://dig.ccmixter.org/people/admiralbob77",
          "createdAt": "2022-11-08T06:46:12.273Z",
          "id": "6361ff4314b08a4853714b69"
        }
      }
    ]
}
```


Send GET {{URL}}/api/v1/albums/6361ff4314b08a4853714b6c/tracks using Postman

Postman console output


```
{
    "success": true,
    "count": 5,
    "data": [
      {
        "_id": "6369fb35313de993fa2734c5",
        "track_name": "Yesterdays Blues",
        "featuring": "Mr Yesterday & The Lovely Ladies",
        "duration": "06:22",
        "track_file": "Radioontheshelf_-_Yesterdays_Blues.mp3",
        "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/Radioontheshelf/63894'>Yesterdays Blues</a> by
Radioontheshelf (c) copyright 2021 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: Mr
Yesterday & The Lovely Ladies</div>",
        "file_size": "14.59MB",
        "createdAt": "2022-11-08T06:46:12.309Z",
        "album": "6361ff4314b08a4853714b6c",
        "slug": "yesterdays-blues",
        "__v": 0
      },
      {
```

```
        "_id": "6369fb35313de993fa2734c6",
        "track_name": "The Day We Went To Tesco's",
        "featuring": "Old Dog Ross",
        "duration": "04:02",
        "track_file": "Radioontheshelf_-_The_Day_We_Went_To_Tesco_s.mp3",
        "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/Radioontheshelf/61451'>The Day We Went To Tesco's</a> by
Radioontheshelf (c) copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: Old Dog
Ross</div>",
        "file_size": "9.27MB",
        "createdAt": "2022-11-08T06:46:12.309Z",
        "album": "6361ff4314b08a4853714b6c",
        "slug": "the-day-we-went-to-tesco's",
        "__v": 0
    },
    {
        "_id": "6369fb35313de993fa2734c7",
        "track_name": "The Blacksmiths Horse",
        "featuring": "Skyejordan",
        "duration": "05:04",
        "track_file": "Radioontheshelf_-_The_Blacksmiths_Horse.mp3",
        "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/Radioontheshelf/62554'>The Blacksmiths Horse</a> by
Radioontheshelf (c) copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft:
Skyejordan</div>",
        "file_size": "11.60MB",
        "createdAt": "2022-11-08T06:46:12.309Z",
        "album": "6361ff4314b08a4853714b6c",
        "slug": "the-blacksmiths-horse",
        "__v": 0
    },
    {
        "_id": "6369fb35313de993fa2734c8",
        "track_name": "Crawling Back To The Sea",
        "featuring": "The Admiral",
        "duration": "04:16",
        "track_file": "Radioontheshelf_-_Crawling_Back_To_The_Sea.mp3",
        "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/Radioontheshelf/62365'>Crawling Back To The Sea</a> by
Radioontheshelf (c) copyright 2020 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: The
Admiral</div>",
        "file_size": "9.77MB",
        "createdAt": "2022-11-08T06:46:12.309Z",
        "album": "6361ff4314b08a4853714b6c",
        "slug": "crawling-back-to-the-sea",
        "__v": 0
    },
    {
```

```
        "_id": "6369fb35313de993fa2734c9",
        "track_name": "Same Destinations",
        "featuring": "The Admiral",
        "duration": "03:27",
        "track_file": "Radioontheshelf_-_Same_Destinations.mp3",
        "credit": "<div class='attribution-block'><a
href='http://dig.ccmixter.org/files/Radioontheshelf/58482'>Same Destinations</a> by
Radioontheshelf (c) copyright 2018 Licensed under a Creative Commons <a
href='http://creativecommons.org/licenses/by/3.0/'>Attribution (3.0)</a> license. Ft: The
Admiral</div>",
        "file_size": "7.92MB",
        "createdAt": "2022-11-08T06:46:12.309Z",
        "album": "6361ff4314b08a4853714b6c",
        "slug": "same-destinations",
        "__v": 0
      }
   ]
}
```

Create the User Model

Install bcryptjs (encrypt passwords) using npm

npm i bcryptjs

models > User.js

```
const mongoose  = require('mongoose');
const bcrypt = require('bcryptjs');

const UserSchema = mongoose.Schema({
   name: {
      type: String,
      required: [true, 'Please add a name']
   },
   email: {
      type: String,
      required: [true, 'Please add an email address'],
      unique: true,
      match: [
         /^\w+([\.-]?\w+)*@\w+([.-]?\w+)*(\.\w{2,3})+$/,
         'Please add a valid email address'
      ]
   },
   role: {
      type: String,
      enum: ['user', 'publisher'],
      default: 'user'
   },
   password: {
      type: String,
      required: [true, 'Please add a password'],
      minlength: 6,
```

```
    select: false
  },
  resetPasswordToken: String,
  resetPasswordExpire: Date,
  createdAt: {
    type: Date,
    default: Date.now()
  }
});

// encrypt password using bcrypt
UserSchema.pre('save', async function(next) {
  // higher genSalt Number, higher the load on system; 10 is recommended in documentation
  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
})

module.exports = mongoose.model('User', UserSchema);
```

Create the Auth Controllers

Auth controllers will contain logic related to authentication, registration, protecting routes, etc. different from users, which will contain logic for CRUD functionality of users for admin.

controllers > auth.js

```
const ErrorResponse = require('../utils/errorResponse');
const asyncHandler = require('../middleware/async');
const UserModel = require('../models/User');

// @desc    Register a user
// @route   POST /api/v1/auth/register
// @access  Public
exports.register = asyncHandler(async(req, res, next) => {
  const { name, email, password, role } = req.body;

  // create user
  const user = await UserModel.create({
    name,
    email,
    password,
    role
  })


  res
    .status(200)
    .json({
      success: true,
      msg: `User registered successfully`
    });
```

```
});
```

Create the Auth Routes

routes > auth.js

```
const express = require('express');

const { register } = require('../controllers/auth');

const router = express.Router();

router
  .route('/register')
  .post(register);

module.exports = router;
```

server.js

```
…
// route files
const albums = require('./routes/albums');
const tracks = require('./routes/tracks');
const auth = require('./routes/auth');
…
// mount routers
app.use('/api/v1/albums', albums);
app.use('/api/v1/tracks', tracks);
app.use('/api/v1/auth', auth);
…
```

Consume the APIs using Postman

REGISTER USER

Send POST {{URL}}/api/v1/auth/register using the following data

```
{
  "name": "publisher",
  "email": "publisher@gmail.com",
  "password": "123456",
  "role": "publisher"
}
```

Postman console output

```
{
  "success": true,
  "msg": "User registered successfully"
}
```

MongoDB output

```
{
  "_id": {
    "$oid": "636dcf59e826f7ec738ebcf2"
  },
  "name": "publisher",
  "email": "publisher@gmail.com",
  "role": "publisher",
  "password": "$2a$10$35vxqo43/stQX1bJPyZqZOfK5QOzxGtlyPJC.QBvYQmp7OzFAw2Si",
  "createdAt": {
    "$date": {
      "$numberLong": "1668140859689"
    }
  },
  "__v": {
    "$numberInt": "0"
  }
}
```


Authentication (Sign In & Get JSON Web Token)

install jsonwebtoken (for authentication) using npm

npm i jsonwebtoken

Sign in with registered user email and password and get a JSON web token to access protected routes

config.env

```
…
JWT_SECRET=onlinemusicstoreversion01
JWT_EXPIRE=6h
```

models > User.js

```
const mongoose  = require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
…
// sign JSON web token and return
UserSchema.methods.getSignedJwtToken = function() {
   return jwt
      .sign(
         {id: this._id},
         process.env.JWT_SECRET,
         {expiresIn: process.env.JWT_EXPIRE}
      )
}

module.exports = mongoose.model('User', UserSchema);
```

controllers > auth.js

```
…
// @desc    Register a user
```

```
// @route   POST /api/v1/auth/register
// @access  Public
exports.register = asyncHandler(async(req, res, next) => {
   const { name, email, password, role } = req.body;

   // create user
   const user = await UserModel.create({
      name,
      email,
      password,
      role
   })

   // create token
   const token = user.getSignedJwtToken();

   res
      .status(200)
      .json({
         success: true,
         msg: `User registered successfully`,
         token
      });
});
```

Send GET {{URL}}/api/v1/auth/register using Postman with following data

```
{
   "name": "publisher",
   "email": "publisher@gmail.com",
   "password": "123456",
   "role": "publisher"
}
```

Postman console output

```
{
   "success": true,
   "msg": "User registered successfully",
   "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzNmRlYjNmYzM2YzNmOWFkYjBhZmI5ZiIsImlhd
CI6MTY2ODE0ODAzMSwiZXhwIjoxNjY4MTY5NjMxfQ.JOUcxmtdP0x8_5xwh4hMCWVZmPSfOn0XR
9SvDHFekwE"
}
```

User Login

models > User.js

```
…
// match user entered password to hashed password in database
UserSchema.methods.matchPassword = async function(enteredPassword) {
```

```
  return await bcrypt.compare(enteredPassword, this.password);
}

module.exports = mongoose.model('User', UserSchema);
```

controllers > auth.js

```
…
// @desc    Login user
// @route   POST /api/v1/auth/login
// @access  Public
exports.login = asyncHandler(async(req, res, next) => {
  const { email, password } = req.body;

  // validate email & password
  if (!email || !password) {
    return next(new ErrorResponse('Please provide a valid email & password', 400));
  }

  // check for user
  const user = await UserModel.findOne({ email }).select('+password');

  if (!user) {
    return next(new ErrorResponse('Invalid credentials', 401));
  }

  // check if password matches
  const isMatch = await user.matchPassword(password);

  if (!isMatch) {
    return next(new ErrorResponse('Invalid credentials', 401));
  }

  // create token
  const token = user.getSignedJwtToken();

  res
    .status(200)
    .json({
      success: true,
      msg: `User logged in successfully`,
      token
    });
});
```

routes > auth.js

```
const express = require('express');

const { register, login } = require('../controllers/auth');
…
```

```
router
   .route('/login')
   .post(login);

module.exports = router;
```

CORRECT email & password

Send POST {{URL}}/api/v1/auth/login using Postman with the following data

```
{
   "email": "publisher@gmail.com",
   "password": "123456"
}
```

Postman console output


```
{
   "success": true,
   "msg": "User logged in successfully",
   "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzNmRlYjNmYzM2YzNmOWFkYjBhZmI5ZiIsImlhdCI6MTY2OTEyMDUyOSwiZXhwIjoxNjY5MTQyMTI5fQ.GczOCZH-nBHSR114knXsSG5mXW4V9x5BrrqEY436VVI"
}
```


Incorrect email

Send POST {{URL}}/api/v1/auth/login using Postman with the following data

```
{
   "email": "publisher1@gmail.com",
   "password": "123456"
}
```

Postman console output

```
{
   "success": false,
   "error": "Invalid credentials"
}
```

Incorrect password

Send POST {{URL}}/api/v1/auth/login using Postman with the following data

```
{
   "email": "publisher@gmail.com",
   "password": "1234567"
}
```

Postman console output

```
{
   "success": false,
   "error": "Invalid credentials"
}
```

Sending JWT in a Cookie

Send the JWT token in a cookie to the browser (to access protected routes).

Install cookie-parser using npm

npm i cookie-parser

server.js

```
…
const cookieParser = require('cookie-parser');
…
// body-parser
app.use(express.json());

// cookie-parser
app.use(cookieParser());
…
```

config > config.env

```
…
JWT_EXPIRE=6h
JWT_COOKIE_EXPIRE=6
```

controllers > auth.js

```
const ErrorResponse = require('../utils/errorResponse');
const asyncHandler = require('../middleware/async');
const UserModel = require('../models/User');

// get token from model, create cookie & send response
const sendTokenResponse = (user, statusCode, res, msg) => {
   // create token
   const token = user.getSignedJwtToken();

   // create cookie
   const options = {
      expires: new Date(Date.now() + process.env.JWT_COOKIE_EXPIRE * 60 * 60 * 1000),
      httpOnly: true
   };

   // set secure protocol (https) for cookie in production
   if (process.env.NODE_ENV === 'production') {
      options.secure = true;
   }
```

```
  res
    .status(statusCode)
    .cookie('token', token, options)
    .json({
      success: true,
      msg,
      token
    });
}

// @desc    Login user
// @route   POST /api/v1/auth/login
// @access  Public
exports.login = asyncHandler(async(req, res, next) => {
  const { email, password } = req.body;
…

  if (!isMatch) {
    return next(new ErrorResponse('Invalid credentials', 401));
  }

  sendTokenResponse(user, 200, res, 'User logged in successfully');
});
```

Auth Protect Middleware

The token generated upon login will be sent to protected routes so that they can be accessed

middleware > auth.js

```
const jwt = require('jsonwebtoken');
const asyncHandler = require('./async');
const ErrorResponse = require('../utils/errorResponse');
const UserModel = require('../models/User');

// protect routes
exports.protect = asyncHandler(async (req, res, next) => {
  let token;

  if (req.headers.authorization && req.headers.authorization.startsWith('Bearer')) {
    // get the token from request headers
    token = req.headers.authorization.split(' ')[1];
  } else if (req.cookies.token) {
    // get the token from cookie
    token = req.cookies.token;
  }

  // ensure the token exists
  if (!token) {
    return next(new ErrorResponse('Not authorized to access this route', 401));
  }
```

```
  try {
    // verify token
    const decoded = jwt.verify(token, process.env.JWT_SECRET);

    // get the currently logged in user
    req.user = await UserModel.findById(decoded.id);

    next();
  } catch (error) {
    return next(new ErrorResponse(`Error: ${error}`, 401));
  }
})
```

routes > albums.js

```
…
const router = express.Router();

// use protect middleware
const { protect } = require('../middleware/auth');
…
router
  .route('/')
  .get(advancedResults(AlbumModel, { path: 'tracks', select: 'track_name featuring duration
file_size'}), getAlbums)
  .post(protect, createAlbum);

router
  .route('/:id')
  .get(getAlbumById)
  .put(protect, updateAlbumById)
  .delete(protect, deleteAlbumById);

router
  .route('/:id/photo')
  .put(protect, albumPhotoUpload);
…
```

routes > tracks.js

```
…
const router = express.Router({ mergeParams: true });

// use protect middleware
const { protect } = require('../middleware/auth');

router
  .route('/')
  .get(advancedResults(TrackModel, { path: 'album', select: 'album_name album_url createdAt' }),
getTracks)
  .post(protect, createTrack);
```

```
router
   .route('/:id')
   .get(getTrackById)
   .put(protect, updateTrackById)
   .delete(protect, deleteTrackById);

router
   .route('/:id/audio')
   .put(protect, trackAudioUpload);
…
```

comment out ‘get the token from cookie’ from auth middleware so that token is not referenced in cookie and try to create an album with name “TEST”

Postman console output

```
{
   "success": false,
   "error": "Not authorized to access this route"
}
```

uncomment the code, login and try to create an album with name “TEST”

Postman console output

```
{
   "success": true,
   "msg": "Album created successfully"
}
```

Get Current Logged in User

controllers > auth.js

```
…
// @desc    Get current logged in user
// @route   POST /api/v1/auth/me
// @access  Private
exports.getMe = asyncHandler(async (req, res, next) => {
   const user = await UserModel.findById(req.user.id);

   res
      .status(200)
      .json({
         success: true,
         data: user
      })
});
```

routes > auth.js

```
const { register, login, getMe } = require('../controllers/auth');
…
const router = express.Router();

// use protect
const { protect } = require('../middleware/auth');
…
router
   .route('/me')
   .get(protect, getMe);

module.exports = router;
```

Sent GET {{URL}}/api/v1/auth/me using Postman

Postman console output


```
{
   "success": true,
   "data": {
      "_id": "636deb3fc36c3f9adb0afb9f",
      "name": "publisher",
      "email": "publisher@gmail.com",
      "role": "publisher",
      "createdAt": "2022-11-11T06:26:53.354Z",
      "__v": 0
   }
}
```


Role Authorization

middleware > auth.js

```
…

// grant access to specific roles
exports.authorize = (...roles) => {
   return (req, res, next) => {
      if (!roles.includes(req.user.role)) {
         return next(new ErrorResponse(`User with role '${req.user.role}' is not authorized to access this route`, 401));
      }
      next();
   }
};
```

routes > albums.js

```
…
// use protect & authorize middleware
const { protect, authorize } = require('../middleware/auth');
…
```

```
router
   .route('/')
   .get(advancedResults(AlbumModel, { path: 'tracks', select: 'track_name featuring duration
file_size'}), getAlbums)
   .post(protect, authorize('publisher', 'admin'), createAlbum);

router
   .route('/:id')
   .get(getAlbumById)
   .put(protect, authorize('publisher', 'admin'), updateAlbumById)
   .delete(protect, authorize('publisher', 'admin'), deleteAlbumById);

router
   .route('/:id/photo')
   .put(protect, authorize('publisher', 'admin'), albumPhotoUpload);

module.exports = router;
```

routes > tracks.js

```
…
// use protect & authorize middleware
const { protect, authorize } = require('../middleware/auth');

router
   .route('/')
   .get(advancedResults(TrackModel, { path: 'album', select: 'album_name album_url createdAt' }),
getTracks)
   .post(protect, authorize('publisher', 'admin'), createTrack);

router
   .route('/:id')
   .get(getTrackById)
   .put(protect, authorize('publisher', 'admin'), updateTrackById)
   .delete(protect, authorize('publisher', 'admin'), deleteTrackById);

router
   .route('/:id/audio')
   .put(protect, authorize('publisher', 'admin'), trackAudioUpload);

module.exports = router;
```

login as user and try to create an album using postman with name ‘test2’

Postman console output

```
{
   "success": false,
   "error": "User with role 'user' is not authorized to access this route"
}
```

register a user, change role to ‘admin’ manually in mongodb collection, login as admin and delete

album with id 637cddbcc3edd7daaafef5d9

Postman console output

```
{
   "success": true,
   "msg": "Album with id '637cddbcc3edd7daaafef5d9' deleted successfully"
}
```

Create a Relationship for Users in Album & Track Model

Create a relationship for users in album & track model

models > Album.js

```
…
createdAt: {
      type: Date,
      default: Date.now()
   },
   user: {
      type: mongoose.Schema.ObjectId,
      ref: 'User',
      required: true
   }
…
```

models > Track.js

```
…
   album: {
      type: mongoose.Schema.ObjectId,
      // reference to model for relationship
      ref: 'Album',
      required: true
   },
   user: {
      type: mongoose.Schema.ObjectId,
      ref: 'User',
      required: true
   }
});
…
```

Ownership Access & Permissions - Album

A user who is not an admin can only add one album. An admin user does not have any such restrictions. Additionally, a user who has created an album as well as admin has permission to update, add a cover photo to and delete that album.

controllers > albums.js

```
…
// @desc    Create new album
```

```
// @route   POST /api/v1/albums
// @access  Private
exports.createAlbum = asyncHandler(async (req, res, next) => {
  // add user to the request body
  req.body.user = req.user.id;

  // check for a published album
  const publishedAlbum = await AlbumModel.findOne({ user: req.user.id });

  // a user who is not an admin can only add one album
  if (req.user.role !== 'admin' && publishedAlbum) {
    return next(
      new ErrorResponse(
        `User with id '${req.user.id}' has already published an album`,
        400
      )
    );
  }

  const albums = await AlbumModel.create(req.body);

  res.status(201).json({
    success: true,
    // data: albums,
    msg: 'Album created successfully',
  });
});
…
// @desc    Update an album
// @route   PUT /api/v1/albums/:id
// @access  Private
exports.updateAlbumById = asyncHandler(async (req, res, next) => {
  let album = await AlbumModel.findById(req.params.id);

  // error for correctly formatted id not present in database
  if (!album) {
    return next(
      new ErrorResponse(`Album with id '${req.params.id}' not found`, 404)
    );
  }

  // ensure user is the album owner or admin
  if (album.user.toString() !== req.user.id && req.user.role !== 'admin') {
    return next(
      new ErrorResponse(
        `User with ID '${req.user.id}' is not authorized to update this album`,
        401
      )
    );
  }
```

```
  album = await AlbumModel.findByIdAndUpdate(req.params.id, req.body, {
    new: true,
    runValidators: true,
  });

  res.status(200).json({
    success: true,
    msg: `Album with id '${req.params.id}' updated successfully`,
    data: album,
  });
});

// @desc    Delete an album
// @route   DELETE /api/v1/albums/:id
// @access  Private
exports.deleteAlbumById = asyncHandler(async (req, res, next) => {
  const album = await AlbumModel.findById(req.params.id);

  // error for correctly formatted id not present in database
  if (!album) {
    return next(
      new ErrorResponse(`Album with id '${req.params.id}' not found`, 404)
    );
  }

  // ensure user is the album owner or admin
  if (album.user.toString() !== req.user.id && req.user.role !== 'admin') {
    return next(
      new ErrorResponse(
        `User with ID '${req.user.id}' is not authorized to update this album`,
        401
      )
    );
  }

  album.remove();

  res.status(200).json({
    success: true,
    msg: `Album with id '${req.params.id}' deleted successfully`,
  });
});

// @desc    Upload a cover photo for album
// @route   PUT /api/v1/albums/:id/photo
// @access  Private
exports.albumPhotoUpload = asyncHandler(async (req, res, next) => {
  const album = await AlbumModel.findById(req.params.id);

  // error for correctly formatted id not present in database
  if (!album) {
```

```
    return next(
      new ErrorResponse(`Album with id '${req.params.id}' not found`, 404)
    );
  }

  // ensure user is the album owner or admin
  if (album.user.toString() !== req.user.id && req.user.role !== 'admin') {
    return next(
      new ErrorResponse(
        `User with ID '${req.user.id}' is not authorized to update this album`,
        401
      )
    );
  }

  if (!req.files) {
    return next(new ErrorResponse(`Kindly upload a file`, 400));
  }
…
```

Login with a non-admin user using Postman, check if user logged in, create an album and try to create another album

Postman console output

```
{
    "success": false,
    "error": "User with id '636deb3fc36c3f9adb0afb9f' has already published an album"
}
```

Try to update, delete, and upload a cover photo for an album '63821dc9ccdb050a6019a180' created by another user with id '637ce76f2f034cf4de47e9c7' using Postman.

Postman console output

```
{
    "success": false,
    "error": "User with ID '636deb3fc36c3f9adb0afb9f' is not authorized to update this album"
}
```

```
{
    "success": false,
    "error": "User with ID '636deb3fc36c3f9adb0afb9f' is not authorized to delete this album"
}
```

```
{
    "success": false,
    "error": "User with ID '636deb3fc36c3f9adb0afb9f' is not authorized to upload a cover photo for this album"
}
```

Ownership Access & Permissions - Track

A user who is not an admin can only add one track. An admin user does not have any such restrictions. Additionally, a user who has created a track as well as admin has permission to update, upload a file to and delete that track.

controllers > tracks.js

```
…
// @desc    Create new track
// @route   POST /api/v1/albums/:albumId/tracks
// @access  Private
exports.createTrack = asyncHandler(async (req, res, next) => {
  // set the 'album' field in TrackSchema & request body to the request albumId of params
  req.body.album = req.params.albumId;

  // add user to the request body
  req.body.user = req.user.id;

  const album = await AlbumModel.findById(req.params.albumId);

  if (!album) {
    return next(
      new ErrorResponse(`Album with id ${req.params.albumId} not found`, 404)
    );
  }

  // ensure the user is the album owner or admin
  if (album.user.toString() !== req.user.id && req.user.role !== 'admin') {
    return next(
      new ErrorResponse(
        `User with ID '${req.user.id}' is not authorized to add tracks to album with ID '${album._id}'`,
        401
      )
    );
  }

  const track = await TrackModel.create(req.body);

  res.status(201).json({
    success: true,
    msg: 'Track created successfully',
  });
});

// @desc    Update track by ID
// @route   PUT /api/v1/tracks/:id
// @access  Private
exports.updateTrackById = asyncHandler(async (req, res, next) => {
  let track = await TrackModel.findById(req.params.id);

  // error for correctly formatted id not present in database
```

```
  if (!track) {
    return next(
      new ErrorResponse(`Track with id '${req.params.id}' not found`, 404)
    );
  }

  // ensure user is track owner or admin
  if (track.user.toString() !== req.user.id && req.user.role !== 'admin') {
    return next(new ErrorResponse(`User with ID '${req.user.id}' is not authorized to update this
track`, 401))
  }

  track = await TrackModel.findByIdAndUpdate(req.params.id, req.body, {
    new: true,
    runValidators: true,
  });

  res.status(200).json({
    success: true,
    msg: `Track with id ${req.params.id} updated successfully`,
    data: track,
  });
});

// @desc    Delete track by ID
// @route   DELETE /api/v1/tracks/:id
// @access  Private
exports.deleteTrackById = asyncHandler(async (req, res, next) => {
  let track = await TrackModel.findById(req.params.id);

  // error for correctly formatted id not present in database
  if (!track) {
    return next(
      new ErrorResponse(`Track with id '${req.params.id}' not found`, 404)
    );
  }

  // ensure user is track owner or admin
  if (track.user.toString() !== req.user.id && req.user.role !== 'admin') {
    return next(new ErrorResponse(`User with ID '${req.user.id}' is not authorized to delete this
track`, 401))
  }

  await TrackModel.remove();

  res.status(200).json({
    success: true,
    msg: `Track with id ${req.params.id} deleted successfully`,
  });
});
```

```
// @desc    Upload an audio track for a track
// @route   PUT /api/v1/tracks/:id/audio
// @access  Private
exports.trackAudioUpload = asyncHandler(async (req, res, next) => {
  const track = await TrackModel.findById(req.params.id).populate('album');

  // error for correctly formatted id not present in database
  if (!track) {
    return next(
      new ErrorResponse(`Track with id '${req.params.id}' not found`, 404)
    );
  }

  // ensure user is track owner or admin
  if (track.user.toString() !== req.user.id && req.user.role !== 'admin') {
    return next(new ErrorResponse(`User with ID '${req.user.id}' is not authorized to upload a file for this track`, 401))
  }

  if (!req.files) {
    return next(new ErrorResponse(`Kindly upload a file`, 400));
  }
…
```

Login with user id '63823151fc9ebaaba5228152' and try to create, update, delete, and upload a track in an album '6361ff4314b08a4853714b6c' created by another user with id '63823375743b2c4f10cc7d60' using Postman.

Postman console output

```
{
    "success": false,
    "error": "User with ID '636deb3fc36c3f9adb0afb9f' is not authorized to add tracks to album with ID '63821dc9ccdb050a6019a180'"
}
```

```
{
    "success": false,
    "error": "User with ID '63823151fc9ebaaba5228152' is not authorized to add tracks to album with ID '6361ff4314b08a4853714b6c'"
}
```

```
{
    "success": false,
    "error": "User with ID '63823151fc9ebaaba5228152' is not authorized to update this track"
}
```

```
{
    "success": false,
    "error": "User with ID '63823151fc9ebaaba5228152' is not authorized to upload a file for this track"
}
```

```
{
   "success": false,
   "error": "User with ID '63823151fc9ebaaba5228152' is not authorized to delete this track"
}
```

Add Users to seeder

_data > users.json

```
[
 {
  "_id": "636deb3fc36c3f9adb0afb9f",
  "name": "admin",
  "email": "admin@gmail.com",
  "role": "user",
  "password": "123456"
 },
 {
  "_id": "63823151fc9ebaaba5228152",
  "name": "publisher",
  "email": "publisher@gmail.com",
  "role": "publisher",
  "password": "123456"
 },
 {
  "_id": "638232bb5115c7c0ae4b102d",
  "name": "user",
  "email": "user@gmail.com",
  "role": "user",
  "password": "123456"
 },
 {
  "_id": "6382330f8f89608157d4a795",
  "name": "Patricia Lebsack",
  "email": "Julianne.OConner@kory.org",
  "role": "publisher",
  "password": "123456"
 },
 {
  "_id": "63823375743b2c4f10cc7d60",
  "name": "Chelsey Dietrich",
  "email": "Lucio_Hettinger@annie.ca",
  "role": "publisher",
  "password": "123456"
 }
]
```

seeder.js

```
…
// load models
```

```
const Album = require('./models/Album');
const Track = require('./models/Track');
const User = require('./models/User');
…
// read JSON files
const albums =  JSON.parse(fs.readFileSync(`${__dirname}/_data/albums.json`, 'utf-8'));
const tracks =  JSON.parse(fs.readFileSync(`${__dirname}/_data/tracks.json`, 'utf-8'));
const users =  JSON.parse(fs.readFileSync(`${__dirname}/_data/users.json`, 'utf-8'));

// import into db
const importData = async () => {
   try {
      await Album.create(albums);
      await Track.create(tracks);
      await User.create(users);
…
// delete from db
const deleteData = async () => {
   try {
      await Album.deleteMany();
      await Track.deleteMany();
      await User.deleteMany();
…
```

Forgot Password - Generate Token

When request is sent to forgot password route via user's email, a reset token is generated and received by the email, hashed and stored in the database in resetPasswordToken of User model, with an expiration date.

models > User.js

```
const crypto = require('crypto');
…
// encrypt password using bcrypt
UserSchema.pre('save', async function (next) {
  // will skip and proceed to generate salt if password is not modified
  if (!this.isModified('password')) {
    next();
  }
  // higher genSalt Number, higher the load on system; 10 is recommended in documentation
  const salt = await bcrypt.genSalt(10);
  this.password = await bcrypt.hash(this.password, salt);
});
…
// generate & hash password reset token
UserSchema.methods.getResetPasswordToken = function () {
  // generate token
  const resetToken = crypto.randomBytes(20).toString('hex');

  // hash token & set to resetPasswordToken field
  this.resetPasswordToken = crypto
```

```
    .createHash('sha256')
    .update(resetToken)
    .digest('hex');

  // set expiration time to 10 minutes
  this.resetPasswordExpire = Date.now() + 10 * 60 * 1000;

  // return the original token (unhashed)
  return resetToken;
};

module.exports = mongoose.model('User', UserSchema);
```

controllers > auth.js

```
…
// @desc    Forgot password
// @route   POST /api/v1/auth/forgotpassword
// @access  Public
exports.forgotPassword = asyncHandler(async (req, res, next) => {
  const user = await UserModel.findOne({ email: req.body.email });

  if (!user) {
    return next(
      new ErrorResponse(`There is no user with email '${req.body.email}'`, 404)
    );
  }

  // get reset token
  const resetToken = user.getResetPasswordToken();

  await user.save({ validateBeforeSave: false });

  res.status(200).json({
    success: true,
    data: user,
  });
});
```

routes > auth.js

```
const express = require('express');

const { register, login, getMe, forgotPassword } = require('../controllers/auth');
…
router
    .route('/forgotpassword')
    .get(forgotPassword);
…
```

Send POST to ‘{{URL}}/api/v1/auth/forgotpassword’ using Postman with following data

```
{
   "email": "publisher1@gmail.com"
}
```

Postman console output

```
{
   "success": false,
   "error": "There is no user with email 'publisher1@gmail.com'"
}
```

Send POST to '{{URL}}/api/v1/auth/forgotpassword' using Postman with following data

```
{
   "email": "publisher@gmail.com"
}
```

Postman console output

```
{
   "success": true,
   "data": {
      "_id": "63823151fc9ebaaba5228152",
      "name": "publisher",
      "email": "publisher@gmail.com",
      "role": "publisher",
      "createdAt": "2022-11-26T15:59:29.696Z",
      "__v": 0,
      "resetPasswordToken":
"7a0b64f6b4eb96eca69e1ac4ac2d1394860e0c347cecf41b72b572a2f4846739",
      "resetPasswordExpire": "2022-12-04T06:22:10.864Z"
   }
}
```

Forgot Password - Send Email

Send email to user's email with password token created which has an export time set to 10 minutes. Using this password token, the user can reset the password. Nodemailer package will be used to send email, using sendtrap (fake SMTP Server).

install nodemailer using npm

npm i nodemailer

config.env

```
…
SMTP_HOST=smtp.mailtrap.io [for mailtrap, else use email host smtp like smtp.gmail.com]
SMTP_PORT=2525 [from host website]
SMTP_EMAIL= [username on host website]
SMTP_PASSWORD= [password on host website]
FROM_EMAIL=noreply@oms.io [appears in email "From:"]
FROM_NAME=OnlineMusicStore [appears in email "From:"]
```

utils > sendEmail.js

```
const nodemailer = require('nodemailer');

const sendEmail = async (options) => {
  // create reusable transporter object using the default SMTP transport
  const transporter = nodemailer.createTransport({
    host: process.env.SMTP_HOST,
    port: process.env.SMTP_PORT,
    auth: {
      user: process.env.SMTP_EMAIL,
      pass: process.env.SMTP_PASSWORD,
    },
  });

  // send mail with defined transport object
  const message = {
    from: `${process.env.FROM_NAME} <${process.env.FROM_EMAIL}>`, // sender address
    to: options.email, // list of receivers
    subject: options.subject, // Subject line
    text: options.message, // plain text body
  };

  const info = await transporter.sendMail(message);

  console.log('Message sent: %s', info.messageId);
};

module.exports = sendEmail;
```

controllers > auth.js

```
const ErrorResponse = require('../utils/errorResponse');
const asyncHandler = require('../middleware/async');
const UserModel = require('../models/User');
const sendEmail = require('../utils/sendEmail');
…
// @desc    Forgot password
// @route   POST /api/v1/auth/forgotpassword
// @access  Public
exports.forgotPassword = asyncHandler(async (req, res, next) => {
  const user = await UserModel.findOne({ email: req.body.email });

  if (!user) {
    return next(
      new ErrorResponse(`There is no user with email '${req.body.email}'`, 404)
    );
  }

  // get reset token
  const resetToken = user.getResetPasswordToken();

  await user.save({ validateBeforeSave: false });
```

```
  // create reset URL
  const resetUrl = `${req.protocol}://${req.get(
    'host'
  )}/api/v1/auth/resetpassword/${resetToken}`;

  const message = `You are receiving this email because you (or someone else) has requested the
reset of a password. Please make a PUT request to:\n\n ${resetUrl}`;

  try {
    await sendEmail({
      email: user.email,
      subject: 'Password Reset Token',
      message,
    });

    res.status(200).json({
      success: true,
      msg: 'Email sent successfully',
    });
  } catch (err) {
    console.log(err);
    user.resetPasswordToken = undefined;
    user.resetPasswordTokenExpire = undefined;

    await user.save({ validateBeforeSave: false });

    return next(new ErrorResponse('Email could not be sent', 500));
  }
});
```

Send POST to '{{URL}}/api/v1/auth/forgotpassword' using Postman with following data

```
{
    "email": "publisher@gmail.com"
}
```

Postman console output

```
{
    "success": true,
    "msg": "Email sent successfully"
}
```

Mailtrap Inbox Email

From:OnlineMusicStore <noreply@oms.io>
To: <publisher@gmail.com>
Subject: Password Reset Token
Message: You are receiving this email because you (or someone else) has requested the reset of a password. Please make a PUT request to:

http://localhost:5000/api/v1/auth/resetpassword/82232226ef8c1c34303b2685b236a8098e0ed34c

Kindly note, the token is set to expire in 10 minutes.

Reset Password

Create the controllers and route-logic for the password reset URL received in the email.

controllers > auth.js

```
const ErrorResponse = require('../utils/errorResponse');
const asyncHandler = require('../middleware/async');
const UserModel = require('../models/User');
const sendEmail = require('../utils/sendEmail');
const crypto = require('crypto');
…
// @desc    Reset Password
// @route   PUT /api/v1/auth/resetpassword/:resettoken
// @access  Public
exports.resetPassword = asyncHandler(async (req, res, next) => {
  // get hashed token
  const resetPasswordToken = crypto
    .createHash('sha256')
    .update(req.params.resettoken.trim())
    .digest('hex');

  const user = await UserModel.findOne({
    resetPasswordToken,
    resetPasswordExpire: { $gt: Date.now() },
  });

  if (!user) {
    return next(new ErrorResponse('Invalid token', 400));
  }

  // set new password
  user.password = req.body.password;

  // nullify tokens
  user.resetPasswordToken = undefined;
  user.resetPasswordExpire = undefined;

  await user.save();

  sendTokenResponse(user, 200, res);
});
```

routes > auth.js

```
const express = require('express');
```

```
const {
  register,
  login,
  getMe,
  forgotPassword,
  resetPassword,
} = require('../controllers/auth');
…
router.route('/resetpassword/:resettoken').put(resetPassword);

module.exports = router;
```

send PUT {{URL}}/api/v1/auth/resetpassword/82232226ef8c1c34303b2685b236a8098e0ed34c using Postman with following data

```
{
   "password": "1234567"
}
```

Postman console output

```
{
   "success": true,
   "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzODIzMTUxZmM5ZWJhYWJhNTIyODE1MiIsImlhdCI6MTY3MDE2Nzg2MywiZXhwIjoxNjcwMTg5NDYzfQ.5Gzh4xv_FNdk8FL0a0SXIzEaYybN1aJGMpX6mm6j3xc"
}
```

MongoDb output

```
{
  "_id": {"$oid": "63823151fc9ebaaba5228152"},
  "name": "publisher",
  "email": "publisher@gmail.com",
  "role": "publisher",
  "password": "$2a$10$uX1x29Nn8f6ngfQTc7cXFevpc0YLAhCZwqIa7KxOQmivD0Fup48oy",
  "createdAt": {"$date": {"$numberLong": "1669478369696"}},
  "__v": {"$numberInt": "0"}
}
```

Update Logged in User’s name & email

Allow admin to update logged in user name & email.

controllers > auth.js

```
…
// @desc    Update logged in user's name & email
// @route   PUT /api/v1/auth/updatedetails
// @access  Private
exports.updateDetails = asyncHandler(async (req, res, next) => {
  const fieldsToUpdate = { name: req.body.name, email: req.body.email };
```

```
  await UserModel.findByIdAndUpdate(req.user.id, fieldsToUpdate, {
    new: true,
    runValidators: true,
  });

  res.status(200).json({
    success: true,
    message: `Name ${req.body.name} & email ${req.body.email} of user with id ${req.user.id}
updated successfullly`,
  });
});
```

routes > auth.js

```
const express = require('express');

const {
  register,
  login,
  getMe,
  forgotPassword,
  resetPassword,
  updateDetails,
} = require('../controllers/auth');
…
router.route('/updatedetails').put(protect, updateDetails);

module.exports = router;
```

Send PUT {{URL}}/api/v1/auth/updatedetails from Postman with following data after logging in with email “Lucio_Hettinger@annie.ca” (name: “Chelsey Dietrich”)

```
{
    "email": "Lucio_Hettinger@annie.ca",
    "name": "Chelsey Dietrich Charleston"
}
```

Postman console output

```
{
    "success": true,
    "message": "Name Chelsey Dietrich & email Lucio_Hettinger@annie.ca of user with id
63823375743b2c4f10cc7d60 updated successfullly"
}
```

Update Logged in User’s password

controllers > auth.js

```
…
```

```
// @desc    Update logged in user's password
// @route   PUT /api/v1/auth/updatepassword
// @access  Private
exports.updatePassword = asyncHandler(async (req, res, next) => {
  const user = await UserModel.findById(req.user.id).select('+password');

  // check current password
  if (!(await user.matchPassword(req.body.currentPassword))) {
    return next(new ErrorResponse('Incorrect Password', 401));
  }

  user.password = req.body.newPassword;

  await user.save();

  sendTokenResponse(user, 200, res, 'Password updated successfully');
});
```

routes > auth.js

```
const express = require('express');

const {
  register,
  login,
  getMe,
  forgotPassword,
  resetPassword,
  updateDetails,
  updatePassword,
} = require('../controllers/auth');
…
router.route('/updatepassword').put(protect, updatePassword);

module.exports = router;
```

Send PUT {{URL}}/api/v1/auth/updatepassword using Postman with following data after logging in with email “Lucio_Hettinger@annie.ca”

```
{
    "currentPassword": "123456",
    "newPassword": "1234567"
}
```

Postman console output


```
{
    "success": true,
    "msg": "Password updated successfully",
    "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzODIzMzc1NzQzYjJjNGYxMGNjN2Q2MCIsImlhd
CI6MTY3MDI0ODU5NCwiZXhwIjoxNjcwMjcwMTk0fQ.0tJhJYO3QChzWwVQNZ4XXFoOPPc3DK1tcu
```

```
XYc2sCZv4"
}
```

Logout to Clear Token Cookie

Logout a user who is logged in

controllers > auth.js

```
…
// @desc    Logout user & clear cookies
// @route   GET /api/v1/auth/logout
// @access  Private
exports.logout = asyncHandler(async (req, res, next) => {
  // delete cookie
  res.cookie('token', 'none', {
    expires: new Date(Date.now() + 10 * 1000),
    httpOnly: true,
  });

  res.status(200).json({
    success: true,
    msg: 'User logged out successfully',
  });
});
```

routes > auth.js

```
const express = require('express');

const {
  register,
  login,
  getMe,
  forgotPassword,
  resetPassword,
  updateDetails,
  updatePassword,
  logout,
} = require('../controllers/auth');
…
router.route('/me').get(protect, getMe);

router.route('/logout').get(logout);
…
```

Get the current logged in user using postman

Send GET {{URL}}/api/v1/auth/me

Postman console output

```
{
```

```
   "success": false,
   "error": "Not authorized to access this route"
}
```


Login a user using postman

Postman console output


```
{
   "success": true,
   "msg": "User logged in successfully",
   "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzNmRlYjNmYzM2YzNmOWFkYjBhZmI5ZiIsImlhdCI6MTY3MDU5Njc0NiwiZXhwIjoxNjcwNjE4MzQ2fQ.NkF5WGLA1avFoHIn1ipeamlUWZSsnosfTjl56iY6Eyk"
}
```


Get logged in user

Postman console output


```
{
   "success": true,
   "data": {
      "_id": "636deb3fc36c3f9adb0afb9f",
      "name": "admin",
      "email": "admin@gmail.com",
      "role": "user",
      "createdAt": "2022-12-08T14:21:13.028Z",
      "__v": 0
   }
}
```


Logout user

Send GET {{URL}}/api/v1/auth/logout

Postman console output


```
{
   "success": true,
   "msg": "User logged out successfully"
}
```


Get logged in user

Postman console output


```
{
   "success": false,
   "error": "Not authorized to access this route"
}
```

Create the User Controllers & Routes

Create users controllers & routes for CRUD functionality for Admin.

controllers > users.js

```
const ErrorResponse = require('../utils/errorResponse');
const asyncHandler = require('../middleware/async');
const UserModel = require('../models/User');

// @desc    Get all users
// @route   GET /api/v1/auth/users
// @access  Private/Admin
exports.getUsers = asyncHandler(async (req, res, next) => {
  res.status(200).json(res.advancedResults);
});

// @desc    Get user by ID
// @route   GET /api/v1/auth/users/:id
// @access  Private/Admin
exports.getUserById = asyncHandler(async (req, res, next) => {
  const user = await UserModel.findById(req.params.id);

  if (!user) {
    next(new ErrorResponse(`User with  id ${req.params.id} not found`, 404));
  }

  res.status(200).json({ success: true, data: user });
});

// @desc    Create a user
// @route   POST /api/v1/auth/users
// @access  Private/Admin
exports.createUser = asyncHandler(async (req, res, next) => {
  const user = await UserModel.create(req.body);

  res
    .status(201)
    .json({ success: true, msg: 'User created successfully', data: user });
});

// @desc    Update a user
// @route   PUT /api/v1/auth/users/:id
// @access  Private/Admin
exports.updateById = asyncHandler(async (req, res, next) => {
  const user = await UserModel.findByIdAndUpdate(req.params.id, req.body, {
    new: true,
    runValidators: true,
  });

  if (!user) {
```

```
    next(new ErrorResponse(`User with  id ${req.params.id} not found`, 404));
  }

  res.status(200).json({
    success: true,
    msg: `User with id ${req.params.id} updated successfully`,
    data: user,
  });
});

// @desc    Delete a user
// @route   DELETE /api/v1/auth/users/:id
// @access  Private/Admin
exports.updateById = asyncHandler(async (req, res, next) => {
  const user = await UserModel.findByIdAndDelete(req.params.id);

  if (!user) {
    next(new ErrorResponse(`User with  id ${req.params.id} not found`, 404));
  }

  res.status(200).json({
    success: true,
    msg: `User with id ${req.params.id} deleted successfully`,
  });
});
```

routes > users.js

```
const express = require('express');

const {
  getUsers,
  createUser,
  getUserById,
  updateUserById,
  deleteUserById,
} = require('../controllers/users');

const User = require('../models/User');

const advancedResults = require('../middleware/advancedResults');

const router = express.Router();

// use protect & authorize
const { protect, authorize } = require('../middleware/auth');

// all routes below will be protected & authorized
router.use(protect);
router.use(authorize('admin'));
```

```
router.route('/').get(advancedResults(User), getUsers).post(createUser);

router
  .route('/:id')
  .get(getUserById)
  .put(updateUserById)
  .delete(deleteUserById);

module.exports = router;
```

server.js

```
…
// route files
const albums = require('./routes/albums');
const tracks = require('./routes/tracks');
const auth = require('./routes/auth');
const users = require('./routes/users');
…
// mount routers
app.use('/api/v1/albums', albums);
app.use('/api/v1/tracks', tracks);
app.use('/api/v1/auth', auth);
app.use('/api/v1/auth/users', users);
…
```

test all routes on Postman after logging in as admin

note: not shown here in effort to avoid repeatability

Create the Review Model, Controllers & Routes

create model, controller & route file for reviews and add to server.js file.

models > Review.js

```
const mongoose = require('mongoose');

const ReviewSchema = new mongoose.Schema({
  title: {
    type: String,
    required: [true, 'Please add a title for the review'],
    maxLength: 100,
  },
  text: {
    type: String,
    required: [true, 'Please add some text'],
  },
  rating: {
    type: Number,
    required: [true, 'Please add a rating between 1 and 5'],
  },
  createdAt: {
```

```
    type: Date,
    default: Date.now(),
  },
  album: {
    type: mongoose.Schema.ObjectId,
    // reference to model for relationship
    ref: 'Album',
    required: true,
  },
  user: {
    type: mongoose.Schema.ObjectId,
    ref: 'User',
    required: true,
  },
});

// prevent user from submitting more than one review per album
ReviewSchema.index({ album: 1, user: 1 }, { unique: true });

module.exports = mongoose.model('Review', ReviewSchema);
```

controllers > reviews.js

```
const ErrorResponse = require('../utils/errorResponse');
const asyncHandler = require('../middleware/async');
const ReviewModel = require('../models/Review');
const AlbumModel = require('../models/Album');

// @desc    Get all reviews
// @route   GET /api/v1/reviews
// @route   GET /api/v1/albums/:albumId/reviews
// @access  Public
exports.getReviews = asyncHandler(async (req, res, next) => {
  if (req.params.albumId) {
    const reviews = await ReviewModel.find({ album: req.params.albumId });

    return res.status(200).json({
      success: true,
      count: reviews.length,
      data: reviews,
    });
  } else {
    res.status(200).json(res.advancedResults);
  }
});

// @desc    Get review by ID
// @route   GET /api/v1/reviews/:id
// @access  Public
exports.getReviewById = asyncHandler(async (req, res, next) => {
  const review = await ReviewModel.findById(req.params.id).populate({
```

```
    path: 'album',
    select: 'album_name description',
  });

  if (!review) {
    next(new ErrorResponse(`Review with  id ${req.params.id} not found`, 404));
  }

  res.status(200).json({ success: true, data: review });
});

// @desc    Create a review
// @route   POST /api/v1/albums/:albumId/reviews
// @access  Private/User
exports.createReview = asyncHandler(async (req, res, next) => {
  req.body.album = req.params.albumId;
  req.body.user = req.user.id;

  const album = await AlbumModel.findById(req.params.albumId);

  if (!album) {
    return next(
      new ErrorResponse(`Album with id '${req.params.albumId}' not found`, 404)
    );
  }

  const review = await ReviewModel.create(req.body);

  res
    .status(201)
    .json({ success: true, msg: 'Review created successfully', data: review });
});

// @desc    Update a review
// @route   PUT /api/v1/reviews/:id
// @access  Private/User
exports.updateReviewById = asyncHandler(async (req, res, next) => {
  const review = await ReviewModel.findByIdAndUpdate(req.params.id, req.body, {
    new: true,
    runValidators: true,
  });

  if (!review) {
    next(new ErrorResponse(`Review with  id ${req.params.id} not found`, 404));
  }

  // ensure review belongs to user or user is admin
  if(review.user.toString() !== req.user.id && req.user.role !== admin) {
    next(new ErrorResponse(`User with  id ${req.user.id} not authorized to update review`, 401));
  }
```

```
  res.status(200).json({
    success: true,
    msg: `Review with id ${req.params.id} updated successfully`,
    data: review,
  });
});

// @desc    Delete a review
// @route   DELETE /api/v1/reviews/:id
// @access  Private/User
exports.deleteReviewById = asyncHandler(async (req, res, next) => {
  const review = await ReviewModel.findByIdAndDelete(req.params.id);

  if (!review) {
    next(new ErrorResponse(`Review with  id ${req.params.id} not found`, 404));
  }

  // ensure review belongs to user or user is admin
  if(review.user.toString() !== req.user.id && req.user.role !== admin) {
    next(new ErrorResponse(`User with  id ${req.user.id} not authorized to update review`, 401));
  }

  res.status(200).json({
    success: true,
    msg: `Review with id ${req.params.id} deleted successfully`,
  });
});
```

routes > reviews.js

```
const express = require('express');

const {
  getReviews,
  getReviewById,
  createReview,
  updateReviewById,
  deleteReviewById,
} = require('../controllers/reviews');

const Review = require('../models/Review');

const advancedResults = require('../middleware/advancedResults');

const router = express.Router({ mergeParams: true });

// use protect & authorize
const { protect, authorize } = require('../middleware/auth');

router
  .route('/')
```

```
  .get(
    advancedResults(Review, {
      path: 'album',
      select: 'album_name description',
    }),
    getReviews
  )
  .post(protect, authorize('user', 'admin'), createReview);

router
  .route('/:id')
  .get(getReviewById)
  .put(protect, authorize('user', 'admin'), updateReviewById)
  .delete(protect, authorize('user', 'admin'), deleteReviewById);

module.exports = router;
```

routes > albums.js

```
…
// use other resource routers
const trackRouter = require('./tracks');
const reviewRouter = require('./reviews');
…
// re-route to other resource routers
router
    .use('/:albumId/tracks', trackRouter)
    .use('/:albumId/reviews', reviewRouter);
…
```

server.js

```
…
// route files
const albums = require('./routes/albums');
const tracks = require('./routes/tracks');
const auth = require('./routes/auth');
const users = require('./routes/users');
const reviews = require('./routes/reviews');
…
// mount routers
app.use('/api/v1/albums', albums);
app.use('/api/v1/tracks', tracks);
app.use('/api/v1/auth', auth);
app.use('/api/v1/auth/users', users);
app.use('/api/v1/reviews', reviews);
…
```

Add Reviews to Seeder

__data > reviews.json

```
[
	{
		"title": "Learned a ton!",
		"text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
		"rating": "8",
		"album": "6361ff4314b08a4853714b68",
		"user": "636deb3fc36c3f9adb0afb9f"
	},
	{
		"title": "Great album",
		"text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
		"rating": "10",
		"album": "6361ff4314b08a4853714b68",
		"user": "63823151fc9ebaaba5228152"
	},
	{
		"title": "Got me a developer job",
		"text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
		"rating": "7",
		"album": "6361ff4314b08a4853714b69",
		"user": "63823151fc9ebaaba5228152"
	},
	{
		"title": "Not that great",
		"text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
		"rating": "4",
		"album": "6361ff4314b08a4853714b69",
		"user": "638232bb5115c7c0ae4b102d"
	},
	{
		"title": "Great overall experience",
		"text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae
```

```
maximus. Curabitur semper felis eu magna laoreet scelerisque",
                "rating": "7",
                "album": "6361ff4314b08a4853714b6a",
                "user": "6382330f8f89608157d4a795"
        },
        {
                "title": "Not worth the money",
                "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
                "rating": "5",
                "album": "6361ff4314b08a4853714b6a",
                "user": "63823375743b2c4f10cc7d60"
        },
        {
                "title": "Best instructors",
                "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
                "rating": "10",
                "album": "6361ff4314b08a4853714b6a",
                "user": "638232bb5115c7c0ae4b102d"
        },
        {
                "title": "Was worth the investment",
                "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
                "rating": "7",
                "album": "6361ff4314b08a4853714b6a",
                "user": "63823151fc9ebaaba5228152"
        }
]
```

seeder.js

```
…
// load models
const Album = require('./models/Album');
const Track = require('./models/Track');
const User = require('./models/User');
const Review = require('./models/Review');
…
// read JSON files
const albums = JSON.parse(
```

```
  fs.readFileSync(`${__dirname}/_data/albums.json`, 'utf-8')
);
const tracks = JSON.parse(
  fs.readFileSync(`${__dirname}/_data/tracks.json`, 'utf-8')
);
const users = JSON.parse(
  fs.readFileSync(`${__dirname}/_data/users.json`, 'utf-8')
);
const reviews = JSON.parse(
  fs.readFileSync(`${__dirname}/_data/reviews.json`, 'utf-8')
);

// import into db
const importData = async () => {
  try {
    await Album.create(albums);
    await Track.create(tracks);
    await User.create(users);
    await Review.create(reviews);
…
// delete from db
const deleteData = async () => {
  try {
    await Album.deleteMany();
    await Track.deleteMany();
    await User.deleteMany();
    await Review.deleteMany();
…
```

run node seeder -d to clear the database and then node seeder -i to populate the database

Get all reviews console output


```
{
    "success": true,
    "count": 1,
    "pagination": {
        "curr": {
            "page": 1
        },
        "next": {
            "page": 2
        }
    },
    "total": 8,
    "limit": 1,
    "data": [
        {
            "_id": "6391f2da06b0ab9e53864436",
            "title": "Not worth the money",
            "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat
```

mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
"rating": 5,
"createdAt": "2022-12-08T14:21:13.029Z",
"album": {
"_id": "6361ff4314b08a4853714b6a",
"description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
"id": "6361ff4314b08a4853714b6a"
},
"user": "63823375743b2c4f10cc7d60",
"__v": 0
}
]
}

Aggregate - Calculate Average Ratings for Reviews

Calculate average ratings for reviews after creating reviews. The average ratings will be displayed in the album model. The average will be recalculated when the rating is deleted.

models > Review.js

```
…
// prevent user from submitting more than one review per album
ReviewSchema.index({ album: 1, user: 1 }, { unique: true });

// static method to get the average rating and save
ReviewSchema.statics.getAverageRating = async function (albumId) {
  const obj = await this.aggregate([
    {
      $match: { album: albumId },
    },
    {
      $group: { _id: '$album', averageRating: { $avg: '$rating' } },
    },
  ]);

  // update rating in database and add as a field
  try {
    await this.model('Album').findByIdAndUpdate(albumId, {
      averageRating: obj[0].averageRating,
    });
  } catch (err) {
    console.log(err);
  }
};
```

```
// call getAverageRating after save
ReviewSchema.post('save', function () {
  this.constructor.getAverageRating(this.album);
});

// call getAverageRating before remove
ReviewSchema.pre('remove', function () {
  this.constructor.getAverageRating(this.album);
});
…
```

Send GET {{URL}}/api/v1/albums/6361ff4314b08a4853714b69/reviews using Postman

Postman console log


```
{
   "success": true,
   "count": 2,
   "data": [
      {
         "_id": "6391f2da06b0ab9e53864433",
         "title": "Got me a developer job",
         "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
         "rating": 7,
         "createdAt": "2022-12-08T14:21:13.029Z",
         "album": "6361ff4314b08a4853714b69",
         "user": "63823151fc9ebaaba5228152",
         "__v": 0
      },
      {
         "_id": "6391f2da06b0ab9e53864434",
         "title": "Not that great",
         "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
         "rating": 4,
         "createdAt": "2022-12-08T14:21:13.029Z",
         "album": "6361ff4314b08a4853714b69",
         "user": "638232bb5115c7c0ae4b102d",
         "__v": 0
      }
   ]
}
```


Send POST {{URL}}/api/v1/albums/6361ff4314b08a4853714b69/reviews to Postman with

following data


```
{
   "title": "Learned a ton!",
   "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
   "rating": "8"
}
```


Postman Console output


```
{
   "success": true,
   "msg": "Review created successfully",
   "data": {
      "title": "Learned a ton!",
      "text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra feugiat mauris id viverra. Duis luctus ex sed facilisis ultrices. Curabitur scelerisque bibendum ligula, quis condimentum libero fermentum in. Aenean erat erat, aliquam in purus a, rhoncus hendrerit tellus. Donec accumsan justo in felis consequat sollicitudin. Fusce luctus mattis nunc vitae maximus. Curabitur semper felis eu magna laoreet scelerisque",
      "rating": 8,
      "createdAt": "2022-12-08T15:23:22.753Z",
      "album": "6361ff4314b08a4853714b69",
      "user": "636deb3fc36c3f9adb0afb9f",
      "_id": "639203d963c9d34ba2bae6bd",
      "__v": 0
   }
}
```


Send GET {{URL}}/api/v1/albums/6361ff4314b08a4853714b69 using Postman

Postman console output


```
{
   "success": true,
   "data": {
      "_id": "6361ff4314b08a4853714b69",
      "album_name": "Admiral Bob",
      "cover_photo": "admiral_bob.jpg",
      "artist": "Admiral Bob",
      "genre": "Instrumental",
      "year": 2020,
      "producer": "CCMixters",
      "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
      "album_url": "http://dig.ccmixter.org/people/admiralbob77",
      "createdAt": "2022-12-08T14:21:12.998Z",
```

```
      "user": "63823151fc9ebaaba5228152",
      "album_slug": "admiral-bob",
      "artist_slug": "admiral-bob",
      "__v": 0,
      "averageRating": 6.333333333333333,
      "id": "6361ff4314b08a4853714b69"
    }
}
```

rating = (7+4+8)/3 = 6.33…

Dynamic View Rendering (EJS)

Render the views with the data sent to/received from the database. The EJS template engine can render the dynamic data in HTML view. Install ejs using npm. Set conditional rendering of response if ‘accept’ in headers doesn’t specify format ‘*/*’ only, which is true for postman and swagger, but false for browsers. Only GET and POST requests are permitted in HTML. For PUT and DELETE requests, method override is required. Install ejs and method-override using npm.

npm install ejs method-override

server.js

```
…
const cors = require('cors');
const ejs = require('ejs');
const methodOverride = require('method-override');
…
// cookie-parser
app.use(cookieParser());

// override with '_method' header in the request for PUT & DELETE
app.use(methodOverride('_method'));
…
// file upload
app.use(fileUpload());

// set public folder as static folder
app.use(express.static(path.join(__dirname, 'public')));

// set view render engine as 'ejs'
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

// check if signed in
app.use((req, res, next) => {
  if(req.headers.authorization === null) {
    res
      .redirect('/api/v1/auth/login');
  } else {
    next();
  }
});
```

```
// get homepage route
app.get('/', (req, res, next) => {
  res
    .status(200)
    .cookie('user', 'none')
    // .json({ success: true, msg: 'Welcome to the online music store' });
    // .redirect('https://documenter.getpostman.com/view/19419701/2s8YzTThXz');
    .render('home', {user: req.cookies.user});
});
…
```

views > home.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
      rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
      crossorigin="anonymous"
    />
    <style>
      body {
        background-image: linear-gradient(
          to right,
          #ff8177 0%,
          #ff867a 0%,
          #ff8c7f 21%,
          #f99185 52%,
          #cf556c 78%,
          #b12a5b 100%
        );
      }
      .container {
        background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
      }
    </style>
    <title>Home</title>
  </head>
  <body>
    <%- include("navbar", { user }) %>
    <div class="container p-5 m-5">
      <div class="card p-5 m-5">
        <h1 class="text-center">Welcome to Online Music Store</h1>
        <p>
          Welcome to the Online Music Store, where visitors can view information
```

```
        about albums and the tracks and reviews within each album posted by
        other users. Users can also post reviews for a particular album. Users
        can also play and pause the tracks within each album.
      </p>
      <p>
        Apart from the role of user, the roles of admin and publisher have
        also been integrated. Three accounts have been created for
        demonstration purposes.
      </p>
      <div class="d-flex flex-column align-content-between mb-3">
        <div class="row">
          <div class="col">Role: Admin</div>
          <div class="col">Email: admin@gmail</div>
        </div>
        <div class="row">
          <div class="col">Role: Publisher</div>
          <div class="col">Email: publisher@gmail</div>
        </div>
        <div class="row">
          <div class="col">Role: User</div>
          <div class="col">Email: user@gmail</div>
        </div>
        <div class="row">
          <div class="col">Password for all three accounts: 123456</div>
        </div>
      </div>
      <p>
        Users can only view albums and tracks, but can view as well as add
        reviews for albums.
      </p>
      <p>
        Publishers can only create and view albums and tracks, but cannot add
        reviews for albums.
      </p>
      <p>
        Admins have full access to all the modules: albums, tracks, reviews
        and users. Resources can be created, read, updated & deleted.
      </p>
    </div>
  </div>
 </body>
</html>
```

middleware > error.js

```
…
  if(req.header('accept')==='*/*') {
    res
      .status(error.statusCode || 500)
      .json({
        success: false,
```

```
            error: error.message || 'Internal Server Error'
          });
    } else {
       res
          .status(error.statusCode || 500)
          .render('error', {
             success: false,
             error: error.message || 'Internal Server Error'
          });
    }
}

module.exports = errorHandler;
```

views > error.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
      rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
      crossorigin="anonymous"
    />
    <style>
      body {
        background-image: linear-gradient(
          to right,
          #ff8177 0%,
          #ff867a 0%,
          #ff8c7f 21%,
          #f99185 52%,
          #cf556c 78%,
          #b12a5b 100%
        );
      }
      .container {
        background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
      }
    </style>
    <title>Error</title>
  </head>
  <body>
    <p class="text-center text-white bg-secondary mt-2">
      Click <a class="text-dark" href="https://zvdas.github.io">here</a> to go
      to portfolio website.
```

```
   </p>
   <div
    class="card m-5 col-6"
    style="margin-left: 20% !important; margin-top: 10% !important"
   >
    <div class="card-header">Error</div>
    <div class="card-body">
     <div class="d-flex justify-content-center mx-2"><%= error %></div>
    </div>
   </div>
  </body>
</html>
```

controllers > auth.js

```
…
// get token from model, create cookie & send response
const sendTokenResponse = (req, user, statusCode, res, msg, urlpath) => {
…
  if(req.header('accept')==='*/*') {
    res
    .status(statusCode)
    .cookie('token', token, options)
    .json({
      success: true,
      msg,
      token,
    });
  } else {
    res
    .status(statusCode)
    .cookie('token', token, options)
    .cookie('user', user)
    .render(urlpath, { user });
  };
};

// @desc    Get register page
// @route   GET /api/v1/auth/register
// @access  Public
exports.getRegister = asyncHandler(async (req, res, next) => {
  res
    .status(200)
    .render('register');
});

// @desc    Register a user
// @route   POST /api/v1/auth/register
// @access  Public
exports.register = asyncHandler(async (req, res, next) => {
…
```

```
  if(req.header('accept')==='*/*') {
    sendTokenResponse(user, 200, res, 'User registered successfully', '');
  } else {
    res
      .status(200)
      .redirect('login');
  }
});

// @desc    Get login page
// @route   GET /api/v1/auth/login
// @access  Public
exports.getLogin = asyncHandler(async (req, res, next) => {
  res
    .status(200)
    .render('login');
});

// @desc    Login user
// @route   POST /api/v1/auth/login
// @access  Public
exports.login = asyncHandler(async (req, res, next) => {
…
  sendTokenResponse(req, user, 200, res, 'User logged in successfully', 'home')
});

// @desc    Get current logged in user
// @route   GET /api/v1/auth/me
// @access  Private
exports.getMe = asyncHandler(async (req, res, next) => {
  const user = await UserModel.findById(req.user.id);

  if(req.header('accept')==='*/*') {
    res
      .status(200)
      .json({
        success: true,
        data: user,
      });
  } else {
    res
      .status(200)
      .render('settings', { user });
  }
});

// @desc    Get forgot password page
// @route   GET /api/v1/auth/forgotpassword
// @access  Public
exports.getForgotPassword = asyncHandler(async (req, res, next) => {
  res
```

```
   .status(200)
   .render('password-forgot');
});

// @desc    Get reset password
// @route   GET /api/v1/auth/resetpassword
// @access  Public
exports.getResetPassword = asyncHandler(async (req, res, next) => {
  res
   .status(200)
   .render('password-reset');
});
…
// @desc    Logout user & clear cookies
// @route   GET /api/v1/auth/logout
// @access  Private
exports.logout = asyncHandler(async (req, res, next) => {
  if(req.header('accept')==='*/*') {
    // delete cookie
    res
     .cookie('token', 'none', {
       expires: new Date(Date.now() + 10 * 1000),
       httpOnly: true,
     })
     .status(200)
     .json({
       success: true,
       msg: 'User logged out successfully',
     });
  } else {
    res
     .cookie('token', 'none', {
       expires: new Date(Date.now() + 10 * 1000),
       httpOnly: true,
     })
     .cookie('user', 'none')
     .status(200)
     .redirect('/');
  }
});
```

routes > auth.js

```
const express = require('express');

const {
  register,
  login,
  getMe,
  forgotPassword,
  resetPassword,
```

```
  updateDetails,
  updatePassword,
  logout,
  getLogin,
  getRegister,
  getForgotPassword,
  getResetPassword,
} = require('../controllers/auth');
…
router.route('/register').get(getRegister).post(register);
router.route('/login').get(getLogin).post(login);
router.route('/logout').get(logout);
router.route('/me').get(protect, getMe);
router.route('/forgotpassword').get(getForgotPassword).post(forgotPassword);
router.route('/resetpassword').get(getResetPassword);
…
```

views > form-template.ejs (reusable template for form input with bootstrap classes)

```
<div class="m-2 row">
  <label for="<%= form_id %>" class="col-4 col-form-label">
    <%= form_label %>
  </label>
  <div class="col">
    <input
      type="<%= form_type %>"
      id="<%= form_id %>"
      name="<%= form_name %>"
      class="form-control"
      value="<%= form_value %>"
      placeholder="<%= form_placeholder %>"
    />
  </div>
</div>
```

views > navbar.ejs (reusable template for navbar with bootstrap classes)

```
<nav class="navbar navbar-expand-lg bg-dark">
    <div class="container-fluid">
      <a class="navbar-brand text-primary" href="/">Online Music Store</a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
          <li class="nav-item active">
            <a class="nav-link text-primary" aria-current="page" href="/">Home</a>
          </li>
          <li class="nav-item">
```

```
            <a class="nav-link text-primary" href="/api/v1/albums">Albums</a>
          </li>
          <li class="nav-item">
            <a class="nav-link text-primary" href="/api/v1/tracks">Tracks</a>
          </li>
          <li class="nav-item">
            <a class="nav-link text-primary" href="/api/v1/reviews">Reviews</a>
          </li>
          <li class="nav-item">
            <a class="nav-link text-primary" href="/api/v1/auth/users">Users</a>
          </li>
          <li class="nav-item">
            <a class="nav-link text-primary" href="/docs">API Docs</a>
          </li>
        </ul>
        <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
          <li class="nav-item">
            <% if (!user.email) { %>
              <li class="nav-item">
                <a class="nav-link text-primary" href="/api/v1/auth/login">Login</a>
              </li>
              <li class="nav-item">
                <a class="nav-link text-primary" href="/api/v1/auth/register">Register</a>
              </li>
            <% } else { %>
              <a class="nav-link text-primary" href="#"><%= user.email %>(<%= user.role %>)</a>
              <li class="nav-item">
                <a class="nav-link text-primary" href="/api/v1/auth/me">Settings</a>
              </li>
            <% } %>
          </li>
          <li class="nav-item">
            <a class="nav-link text-primary" href="/api/v1/auth/logout">Logout</a>
          </li>
        </ul>
      </div>
    </div>
</nav>
<p class="text-center text-white bg-secondary mt-2">Click <a class="text-dark"
href="https://zvdas.github.io">here</a> to go to portfolio website.</p>
```

views > register.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
```

```
    rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
    crossorigin="anonymous"
  />
  <style>
    body {
      background-image: linear-gradient(
        to right,
        #ff8177 0%,
        #ff867a 0%,
        #ff8c7f 21%,
        #f99185 52%,
        #cf556c 78%,
        #b12a5b 100%
      );
    }
    .container {
      background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
    }
  </style>
  <title>Register</title>
 </head>
 <body>
  <p class="text-center text-white bg-secondary mt-2">
    Click <a class="text-dark" href="https://zvdas.github.io">here</a> to go
    to portfolio website.
  </p>
  <div
    class="card m-5 col-6"
    style="margin-left: 20% !important; margin-top: 10% !important"
  >
    <div class="card-header">Register</div>
    <div class="card-body">
      <form
        class="row g-2 m-2 p-2"
        name="loginRegister"
        action="register"
        method="POST"
      >
        <%- include("form-template", {form_id: "inputName", form_name: "name",
        form_label: "Name", form_type: "text", form_value:
        "",form_placeholder: "john Doe"}) %> <%- include("form-template",
        {form_id: "inputEmail", form_name: "email", form_label: "Email",
        form_type: "email", form_value: "",form_placeholder:
        "example@email.com"}) %> <%- include("form-template", {form_id:
        "inputPassword", form_name: "password", form_label: "Password",
        form_type: "password", form_value: "", form_placeholder: ""}) %>
        <div class="m-2 row">
          <label for="inputRole" class="col-4 col-form-label">Role</label>
          <div class="col">
```

```
          <select
            class="form-select"
            name="role"
            id="role"
            aria-label="Default select example"
          >
            <option selected>User</option>
            <option value="publisher">Publisher</option>
          </select>
        </div>
      </div>
      <div class="m-2 row">
        <input
          type="submit"
          class="btn btn-primary mx-2 col"
          value="Submit"
        />
        <a class="btn btn-primary mx-2 col" href="/api/v1/auth/login"
          >Login</a
        >
      </div>
    </form>
  </div>
 </div>
</body>
</html>
```

views > login.ejs

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link
   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
   rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
    crossorigin="anonymous"
  />
  <style>
   body {
    background-image: linear-gradient(
     to right,
     #ff8177 0%,
     #ff867a 0%,
     #ff8c7f 21%,
     #f99185 52%,
     #cf556c 78%,
```

```
      #b12a5b 100%
     );
   }
   .container {
    background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
   }
  </style>
  <title>Login</title>
</head>
<body>
  <p class="text-center text-white bg-secondary mt-2">
   Click <a class="text-dark" href="https://zvdas.github.io">here</a> to go
   to portfolio website.
  </p>
  <div
   class="card m-5 col-6"
   style="margin-left: 20% !important; margin-top: 10% !important"
  >
   <div class="card-header">Login</div>
   <div class="card-body">
    <form
     class="row g-2 m-2 p-2"
     name="loginForm"
     action="login"
     method="POST"
    >
     <%- include("form-template", {form_id: "inputEmail", form_name:
     "email", form_label: "Email", form_type: "email", form_value:
     "",form_placeholder: "example@email.com"}) %> <%-
     include("form-template", {form_id: "inputPassword", form_name:
     "password", form_label: "Password", form_type: "password", form_value:
     "", form_placeholder: ""}) %>
     <div class="row m-2">
      <a
       class="d-flex justify-content-end"
       href="/api/v1/auth/forgotpassword"
       >forgot password?</a
      >
     </div>
     <div class="m-2 row">
      <input
       type="submit"
       class="btn btn-primary mx-2 col"
       value="Submit"
      />
      <a class="btn btn-primary mx-2 col" href="/api/v1/auth/register"
       >Register</a
      >
     </div>
    </form>
   </div>
```

```
  </div>
 </body>
</html>
```

views > password-forgot.ejs

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link
   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
   rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
   crossorigin="anonymous"
  />
  <style>
   body {
    background-image: linear-gradient(
     to right,
     #ff8177 0%,
     #ff867a 0%,
     #ff8c7f 21%,
     #f99185 52%,
     #cf556c 78%,
     #b12a5b 100%
    );
   }
   .container {
    background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
   }
  </style>
  <title>Forgot Password</title>
 </head>
 <body>
  <p class="text-center text-white bg-secondary mt-2">
   Click <a class="text-dark" href="https://zvdas.github.io">here</a> to go
   to portfolio website.
  </p>
  <div
   class="card m-5 col-6"
   style="margin-left: 20% !important; margin-top: 10% !important"
  >
   <div class="card-header">Forgot Password</div>
   <div class="card-body">
    <form action="/api/v1/auth/forgotpassword" method="POST">
     <p>Enter the registered email address.</p>
     <%- include("form-template", {form_id: "email", form_name: "email",
```

```
        form_label: "Email", form_type: "email", form_value: "",
        form_placeholder: "john.doe@example.com"}) %>
        <div class="m-2 row">
         <a class="btn btn-primary mx-2 col" href="/api/v1/auth/login"
          >Login</a
         >
         <input
          class="btn btn-primary mx-2 col"
          type="submit"
          value="Submit"
         />
        </div>
       </form>
     </div>
   </div>
  </body>
</html>
```

views > password-reset.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
      rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
      crossorigin="anonymous"
    />
    <style>
      body {
        background-image: linear-gradient(
          to right,
          #ff8177 0%,
          #ff867a 0%,
          #ff8c7f 21%,
          #f99185 52%,
          #cf556c 78%,
          #b12a5b 100%
        );
      }
      .container {
        background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
      }
    </style>
    <script>
      // document.querySelector("form").onsubmit = function() {setAttribute("action",
```

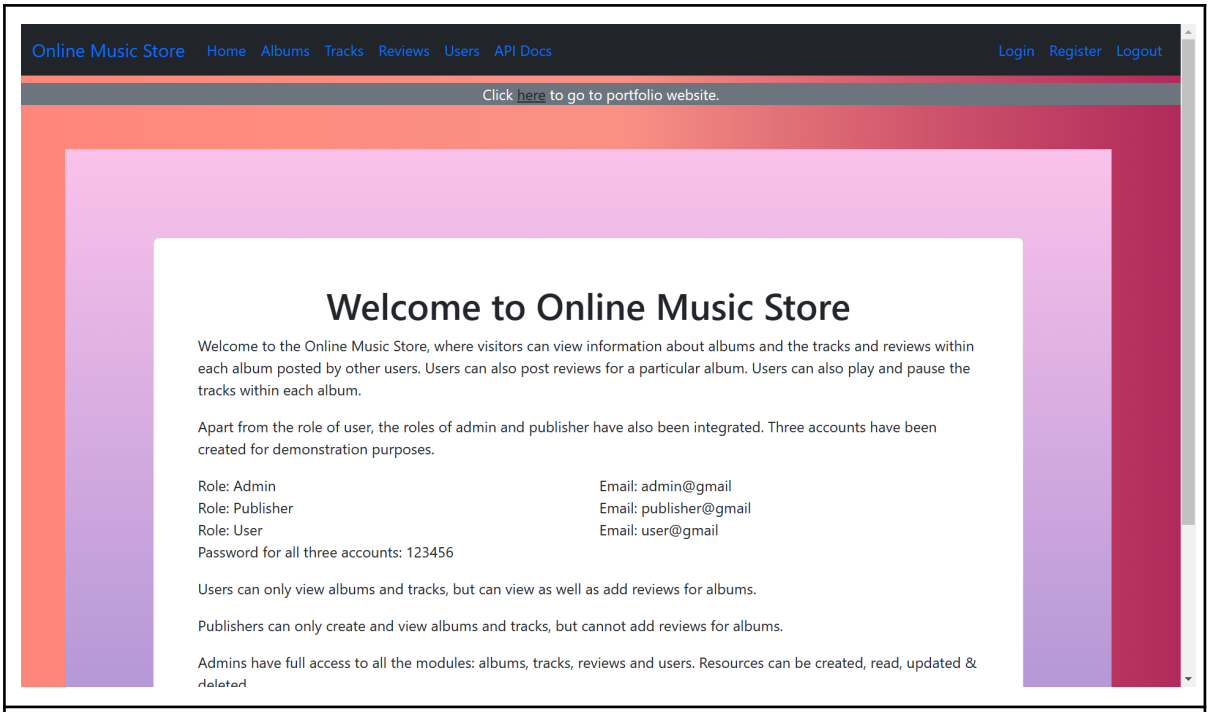
```
"/api/v1/auth/resetpassword/", document.querySelector("input[name=token]").value)}
    document.forms[0].action =
      '/api/v1/auth/resetpassword/' + document.getElementById('token').value;
    // console.log(document.forms[0].action);
  </script>
  <title>Reset Password</title>
 </head>
 <body>
  <p class="text-center text-white bg-secondary mt-2">
   Click <a class="text-dark" href="https://zvdas.github.io">here</a> to go
   to portfolio website.
  </p>
  <div
   class="card m-5 col-6"
   style="margin-left: 20% !important; margin-top: 10% !important"
  >
   <div class="card-header">Reset Password</div>
   <div class="card-body">
    <form action="/api/v1/auth/resetpassword/" method="POST">
     <p>
      Enter the new password and reset token received at the email address
      provided.
     </p>
     <%- include("form-template", {form_id: "password", form_name:
     "password", form_label: "Password", form_type: "email", form_value:
     "", form_placeholder: "Enter password"}) %> <%-
     include("form-template", {form_id: "token", form_name: "token",
     form_label: "Token", form_type: "text", form_value: "",
     form_placeholder: "Enter reset token"}) %>
     <div class="d-flex justify-content-center">
      <input class="btn btn-primary col-4" type="submit" value="Submit" />
     </div>
    </form>
   </div>
  </div>
 </body>
</html>
```

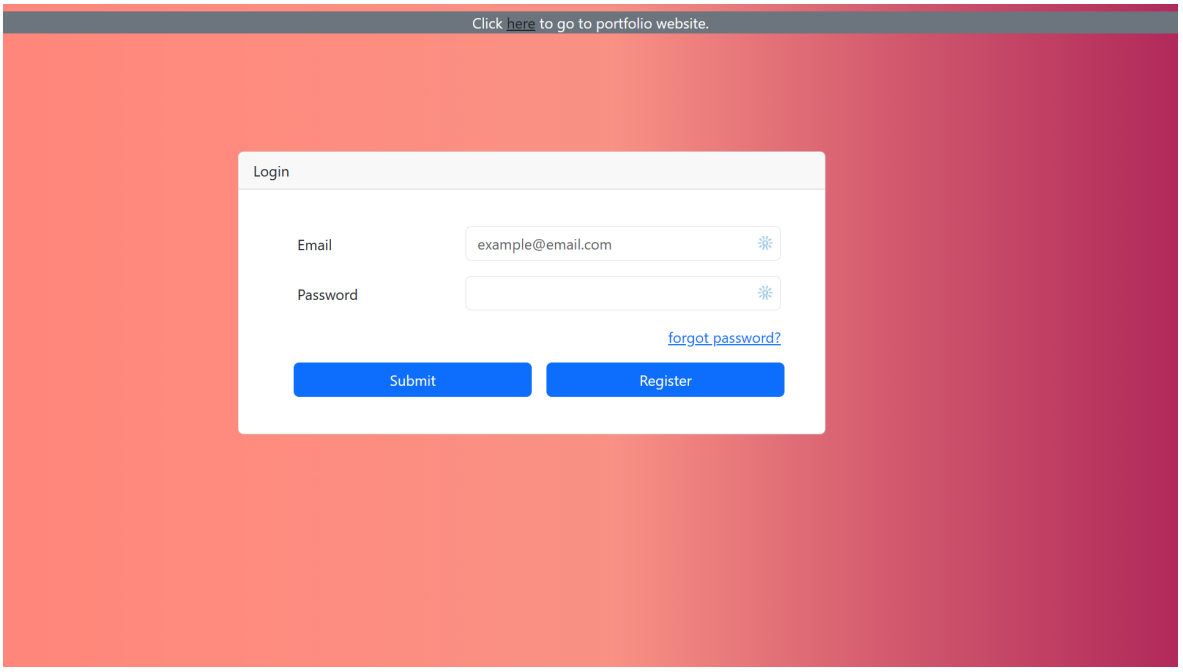
views > settings.ejs

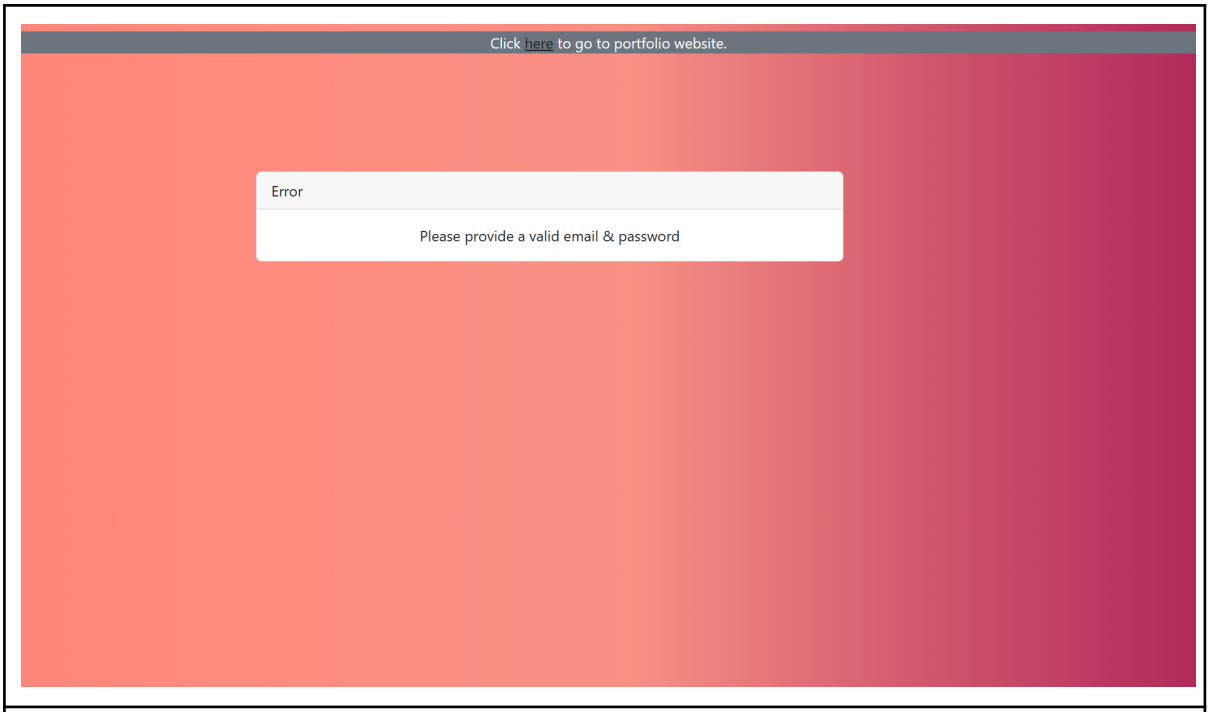
```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link
   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
   rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
```

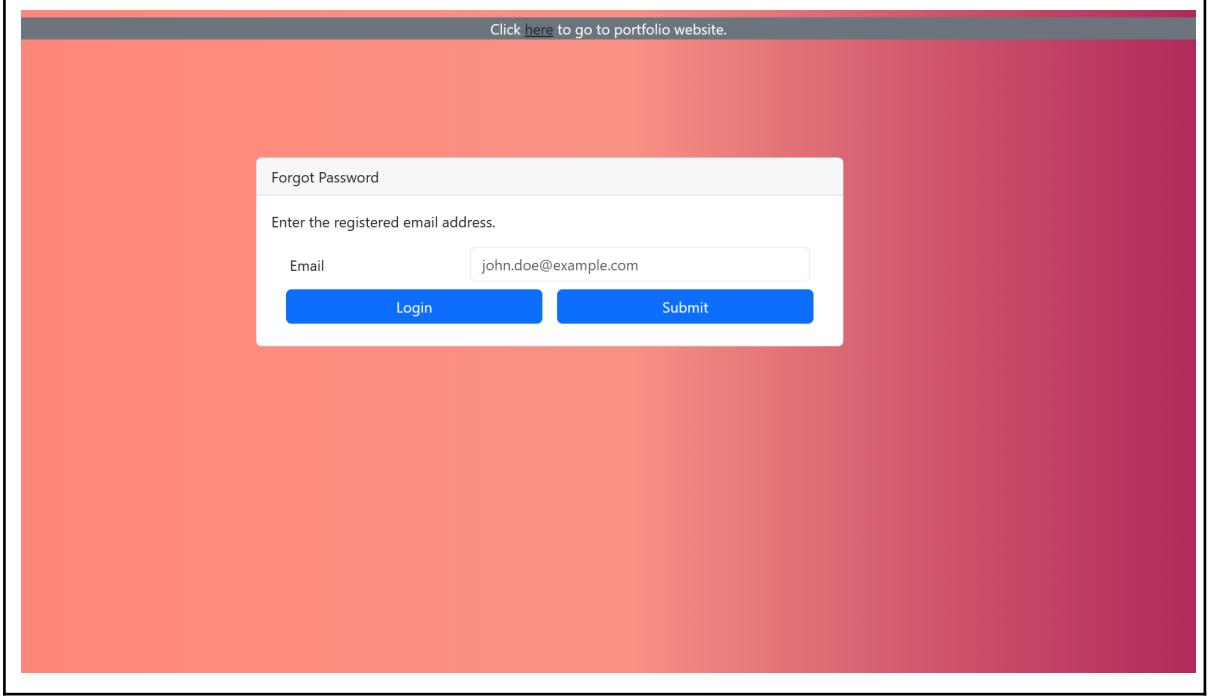
```
    crossorigin="anonymous"
   />
   <style>
    body {
     background-image: linear-gradient(
      to right,
      #ff8177 0%,
      #ff867a 0%,
      #ff8c7f 21%,
      #f99185 52%,
      #cf556c 78%,
      #b12a5b 100%
     );
    }
    .container {
     background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
    }
   </style>
   <title>Settings</title>
  </head>
  <body>
   <%- include("navbar", { user }) %>
   <div
    class="card m-5 col-6"
    style="margin-left: 20% !important; margin-top: 10% !important"
   >
    <div class="card-header">User Details</div>
    <div class="card-body">
     <div class="d-flex justify-content-center mx-2">
      <div class="col-3">
       <p>Name</p>
       <p>Email</p>
       <p>Role</p>
       <p>Created At</p>
      </div>
      <div class="col">
       <p><%= user.name %></p>
       <p><%= user.email %></p>
       <p><%= user.role %></p>
       <p><%= user.createdAt %></p>
      </div>
     </div>
    </div>
  </body>
</html>
```

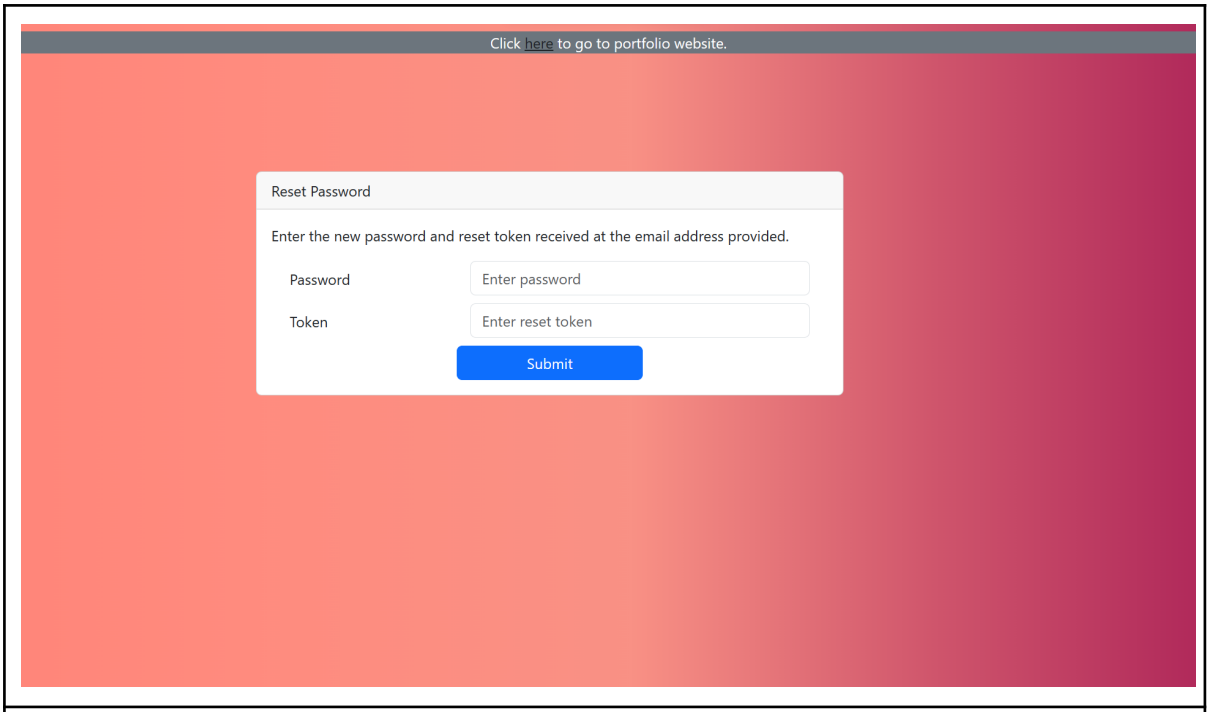
home

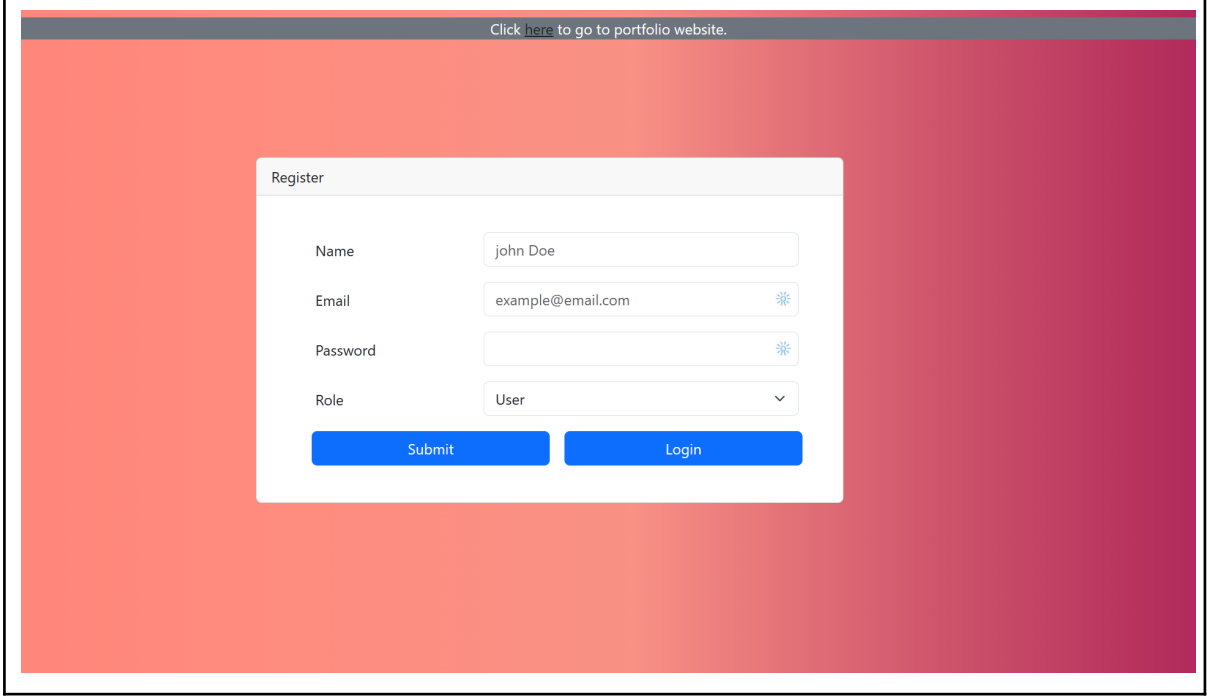

Online Music Store
Home
Albums
Tracks
Reviews
Users
API Docs
Login
Register
Logout
Click here to go to portfolio website.
Welcome to Online Music Store
Welcome to the Online Music Store, where visitors can view information about albums and the tracks and reviews within each album posted by other users. Users can also post reviews for a particular album. Users can also play and pause the tracks within each album.
Apart from the role of user, the roles of admin and publisher have also been integrated. Three accounts have been created for demonstration purposes.
Role: Admin
Email: admin@gmail
Role: Publisher
Email: publisher@gmail
Role: User
Email: user@gmail
Password for all three accounts: 123456
Users can only view albums and tracks, but can view as well as add reviews for albums.
Publishers can only create and view albums and tracks, but cannot add reviews for albums.
Admins have full access to all the modules: albums, tracks, reviews and users. Resources can be created, read, updated &

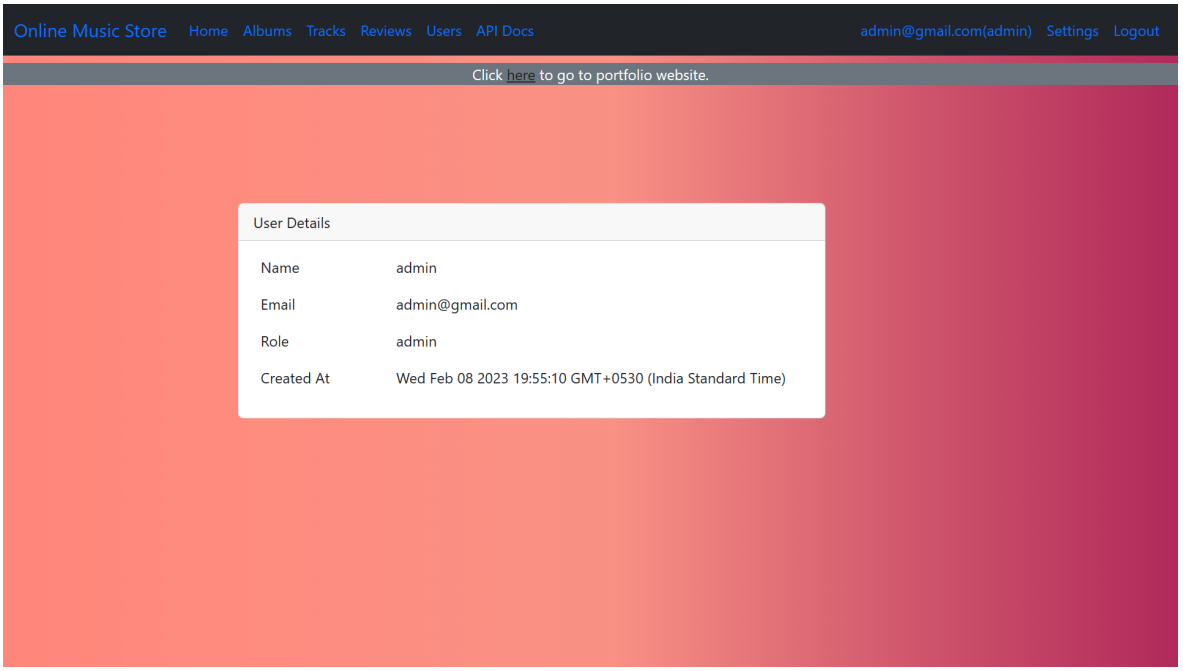

Click here to go to portfolio website.
Login
Email
example@email.com
Password
forgot password?
Submit
Register


Click here to go to portfolio website.
Error
Please provide a valid email & password


Click here to go to portfolio website.
Forgot Password
Enter the registered email address.
Email
john.doe@example.com
Login
Submit


Click here to go to portfolio website.
Reset Password
Enter the new password and reset token received at the email address provided.
Password
Enter password
Token
Enter reset token
Submit


Click here to go to portfolio website.
Register
Name
john Doe
Email
example@email.com
Password
Role
User
Submit
Login



controllers > albums.js

```
…
// @desc    Get all albums
// @route   GET /api/v1/albums
// @access  Public
exports.getAlbums = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
      .status(200)
      .json(res.advancedResults);
  } else {
    res
      .status(200)
      .cookie('albums', albums)
      .render('albums', {msg: '', results: res.advancedResults, user: req.cookies.user});
  }
});

// @desc    Get album by ID
// @route   GET /api/v1/albums/:id
// @access  Public
exports.getAlbumById = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
      .status(200)
      .json({
        success: true,
        data: album,
```

```
    });
  } else {
    res
      .status(200)
      .render('album-detail', {msg: '', result: album, user: req.cookies.user});
  }
});

// @desc    Get create new album page
// @route   GET /api/v1/albums/newalbum
// @access  Private
exports.getCreateAlbum = asyncHandler(async (req, res, next) => {
  res
    .status(200)
    .render('album-detail', {
      msg: '',
      result: {},
      user: req.cookies.user
    });
});

// @desc    Create new album
// @route   POST /api/v1/albums
// @access  Private
exports.createAlbum = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
      .status(201)
      .json({
        success: true,
        data: album,
        msg: 'Album created successfully',
      });
  } else {
    res
      .status(201)
      .render('album-detail', {
        msg: 'Album created successfully',
        result: album,
        user: req.cookies.user
      });
  }
});

// @desc    Update an album
// @route   PUT /api/v1/albums/:id
// @access  Private
exports.updateAlbumById = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
```

```
    res
      .status(200)
      .json({
        success: true,
        msg: `Album with id '${req.params.id}' updated successfully`,
        data: album,
      });
  } else {
    res
      .status(200)
      .render('album-detail', {
        msg: `Album with id '${req.params.id}' updated successfully`,
        result: album,
        user: req.cookies.user
      });
  }
});

// @desc    Delete an album
// @route   DELETE /api/v1/albums/:id
// @access  Private
exports.deleteAlbumById = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
      .status(200)
      .json({
        success: true,
        msg: `Album with id '${req.params.id}' deleted successfully`,
      });
  } else {
    res
      .status(200)
      .redirect('/api/v1/albums');
  }
});

// @desc    Upload a cover photo for album
// @route   PUT /api/v1/albums/:id/photo
// @access  Private
exports.albumPhotoUpload = asyncHandler(async (req, res, next) => {
…
    if(req.header('accept')==='*/*') {
      res
        .status(200)
        .json({
          success: true,
          msg: `Cover photo uploaded to album with id '${req.params.id}' successfully`,
          data: file.name,
        });
    } else {
```

```
    res
      .status(200)
      .render('album-detail', {
        msg: `Cover photo uploaded to album with id '${req.params.id}' successfully`,
        result: album,
        user: req.cookies.user
      });
  }
 });
});
```

routes > albums.js

```
…
router.route('/')
 .get(advancedResults(AlbumModel, [{
      path: 'tracks',
      select: 'track_name featuring duration file_size track_file',
    },
    'reviews',
    'user'
  ]),
  getAlbums).post(protect, authorize('publisher', 'admin'), createAlbum);

router.route('/newalbum').get(protect, authorize('publisher', 'admin'), getCreateAlbum);

router.route('/:id').get(getAlbumById).put(protect, authorize('publisher', 'admin'),
updateAlbumById).delete(protect, authorize('publisher', 'admin'), deleteAlbumById);

router.route('/:id/photo').put(protect, authorize('publisher', 'admin'), albumPhotoUpload);

module.exports = router;
```

views > albums.ejs

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link
   href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
   rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
    crossorigin="anonymous"
  />
  <style>
   body {
    background-image: linear-gradient(
```

```
      to right,
      #ff8177 0%,
      #ff867a 0%,
      #ff8c7f 21%,
      #f99185 52%,
      #cf556c 78%,
      #b12a5b 100%
    );
  }
  .container {
    background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
  }
  .card.m-2.p-2 {
    background-image: linear-gradient(120deg, #84fab0 0%, #8fd3f4 100%);
  }
 </style>
 <title>Albums</title>
</head>
<body>
 <%- include("navbar", { user }) %>
 <div class="container p-2">
  <h1 class="text-center">Albums</h1>
  <div class="card m-2 p-2">
   <div class="d-flex justify-content-around">
    <p>display <%= results.limit %> results per page</p>
    <p>|</p>
    <p>
     results <%= results.pagination.curr.page %> of <%= results.total %>
    </p>
    <p>|</p>
    <% if (results.pagination.prev) { %>
    <p>
     <a href="/api/v1/albums?page=<%= results.pagination.prev.page %>"
      >prev</a
     >
    </p>
    <p>|</p>
    <% } %>
    <p>
     page <%= results.pagination.curr.page %> of <%= results.total %>
    </p>
    <% if (results.pagination.next) { %>
    <p>|</p>
    <p>
     <a href="/api/v1/albums?page=<%= results.pagination.next.page %>"
      >next</a
     >
    </p>
    <% } %>
   </div>
```

```
    <% results.data.map(album => { %>
    <div class="d-flex justify-content-center mb-3">
      <img
        src="/uploads/albums/<%= album.artist_slug %>-<%= album.album_slug %>/<%=
album.cover_photo %>"
        class="w-25"
      />
    </div>
    <div class="card p-2 my-2 mx-2">
      <div class="card-title text-center">
        <strong>Album Information</strong>
      </div>
      <div class="row mx-5">
        <p class="col">Album Name</p>
        <p class="col"><%= album.album_name %></p>
      </div>
      <div class="row mx-5">
        <p class="col">Artist</p>
        <p class="col"><%= album.artist %></p>
      </div>
      <div class="row mx-5">
        <p class="col">Year</p>
        <p class="col"><%= album.year %></p>
      </div>
      <div class="row mx-5">
        <p class="col">Producer</p>
        <p class="col"><%= album.producer %></p>
      </div>
      <div class="row mx-5">
        <p class="col">Description</p>
        <p class="col"><%- album.description %></p>
      </div>
      <div class="row mx-5">
        <p class="col">Created By</p>
        <p class="col">
          <%= album.user.name %> (role: <%= album.user.name %>)
        </p>
      </div>
      <div class="row mx-5">
        <p class="col">Created At</p>
        <p class="col"><%= album.createdAt %></p>
      </div>
      <div class="row mx-5">
        <p class="col">Album URL</p>
        <p class="col">
          <a href="<%= album.album_url %>"><%= album.album_url %></a>
        </p>
      </div>
    </div>

    <div class="card p-2 my-2 mx-2">
```

```
<div class="card-title text-center">
  <strong>Track Information</strong>
</div>
<% if (album.tracks.length === 0) { %>
<p class="text-center">No tracks added</p>
<p class="text-center">
  <a
    href="/api/v1/albums/<%= album.id %>/tracks"
    class="btn btn-primary w-75"
    >Add Tracks</a
  >
</p>
<% } else { %>
<div class="row">
  <div class="col-4">
    <div id="list-example" class="list-group">
      <% album.tracks.map((track, index) => { %>
      <a
        class="list-group-item list-group-item-action text-primary bg-body-secondary"
        href="#list-item-<%= index + 1 %>"
        ><%= index + 1 %>. <%= track.track_name %></a
      >
      <% }) %>
    </div>
  </div>

  <div class="col-8">
    <div
      style="overflow-y: scroll; height: 250px"
      data-bs-spy="scroll"
      data-bs-target="#list-example"
      data-bs-smooth-scroll="true"
      data-bs-offset="50"
      class="scrollspy-example"
      tabindex="0"
    >
      <% album.tracks.map((track, index) => { %>
      <div id="list-item-<%= index + 1 %>">
        <div class="row mx-5">
          <p class="col">Track Name</p>
          <p class="col"><%= track.track_name %></p>
        </div>
        <div class="row mx-5">
          <p class="col">Featuring</p>
          <p class="col"><%= track.featuring %></p>
        </div>
        <div class="row mx-5">
          <p class="col">Duration</p>
          <p class="col"><%= track.duration %></p>
        </div>
        <div class="row mx-5">
```

```
              <p class="col">File Size</p>
              <p class="col"><%= track.file_size %></p>
            </div>
            <div class="row mx-5">
              <p class="col">File Preview</p>
              <p class="col">
                <audio controls>
                  <source
                    src="/uploads/albums/<%= album.artist_slug %>-<%= album.album_slug %>/<%=
track.track_file %>"
                    type="audio/mp3"
                  />
                </audio>
              </p>
            </div>
            <div class="row mx-5">
              <p class="col">
                <a
                  class="btn btn-primary w-100"
                  href="/api/v1/tracks/<%= track._id %>"
                >
                  Edit
                </a>
              </p>
              <p class="col">
                <a
                  class="btn btn-primary w-100"
                  href="/api/v1/tracks/<%= track._id %>?_method=DELETE"
                >
                  <input type="hidden" name="_method" value="DELETE" />
                  Delete
                </a>
              </p>
            </div>
            <hr />
          </div>
          <% }) %>
          <p class="text-center">
            <a
              href="/api/v1/albums/<%= album.id %>/tracks"
              class="btn btn-primary w-75"
              >Add Tracks</a
            >
          </p>
        </div>
      </div>
    </div>
    <% } %>
  </div>

  <div class="card p-2 my-2 mx-2">
```

```
<div class="card-title text-center">
  <strong>Review Information</strong>
</div>
<% if (album.reviews.length === 0) { %>
<p class="text-center">No reviews added</p>
<p class="text-center">
  <a
    href="/api/v1/albums/<%= album.id %>/reviews"
    class="btn btn-primary w-75"
    >Add Reviews</a
  >
</p>
<% } else { %>
<div class="row">
  <div class="col-4">
    <div id="list-example" class="list-group">
      <% album.reviews.map((track, index) => { %>
      <a
        class="list-group-item list-group-item-action text-primary bg-body-secondary"
        href="#list-item-<%= index + 1 %>"
        ><%= index + 1 %>. <%= track.title %></a
      >
      <% }) %>
    </div>
  </div>

  <div class="col-8">
    <div
      style="overflow-y: scroll; height: 250px"
      data-bs-spy="scroll"
      data-bs-target="#list-example"
      data-bs-smooth-scroll="true"
      data-bs-offset="50"
      class="scrollspy-example"
      tabindex="0"
    >
      <% album.reviews.map((review, index) => { %>
      <div id="list-item-<%= index + 1 %>">
        <div class="row mx-5">
          <p class="col">Review Title</p>
          <p class="col"><%= review.track_name %></p>
        </div>
        <div class="row mx-5">
          <p class="col">Review Text</p>
          <p class="col"><%= review.text %></p>
        </div>
        <div class="row mx-5">
          <p class="col">Rating</p>
          <p class="col"><%= review.rating %></p>
        </div>
        <div class="row mx-5">
```

```
          <p class="col">
            <a
              class="btn btn-primary w-100"
              href="/api/v1/reviews/<%= review._id %>"
            >
              Edit
            </a>
          </p>
          <p class="col">
            <a
              class="btn btn-primary w-100"
              href="/api/v1/reviews/<%= review._id %>?_method=DELETE"
            >
              <input type="hidden" name="_method" value="DELETE" />
              Delete
            </a>
          </p>
        </div>
        <hr />
      </div>
      <% }) %>
      <p class="text-center">
        <a
          href="/api/v1/albums/<%= album.id %>/reviews"
          class="btn btn-primary w-75"
          >Add Reviews</a
        >
      </p>
    </div>
  </div>
 </div>
 <% } %>
</div>

<div class="row p-2 my-2 mx-2">
 <a class="btn btn-primary col mx-1" href="/api/v1/albums/newalbum"
  >Add</a
 >
 <a
  class="btn btn-primary col mx-1"
  href="/api/v1/albums/<%= album.id %>"
  >Edit</a
 >
 <form
  class="col mx-1"
  action="/api/v1/albums/<%= album.id %>?_method=DELETE"
  method="post"
 >
  <input type="hidden" name="_method" value="DELETE" />
  <input class="btn btn-primary w-100" type="submit" value="Delete" />
 </form>
```

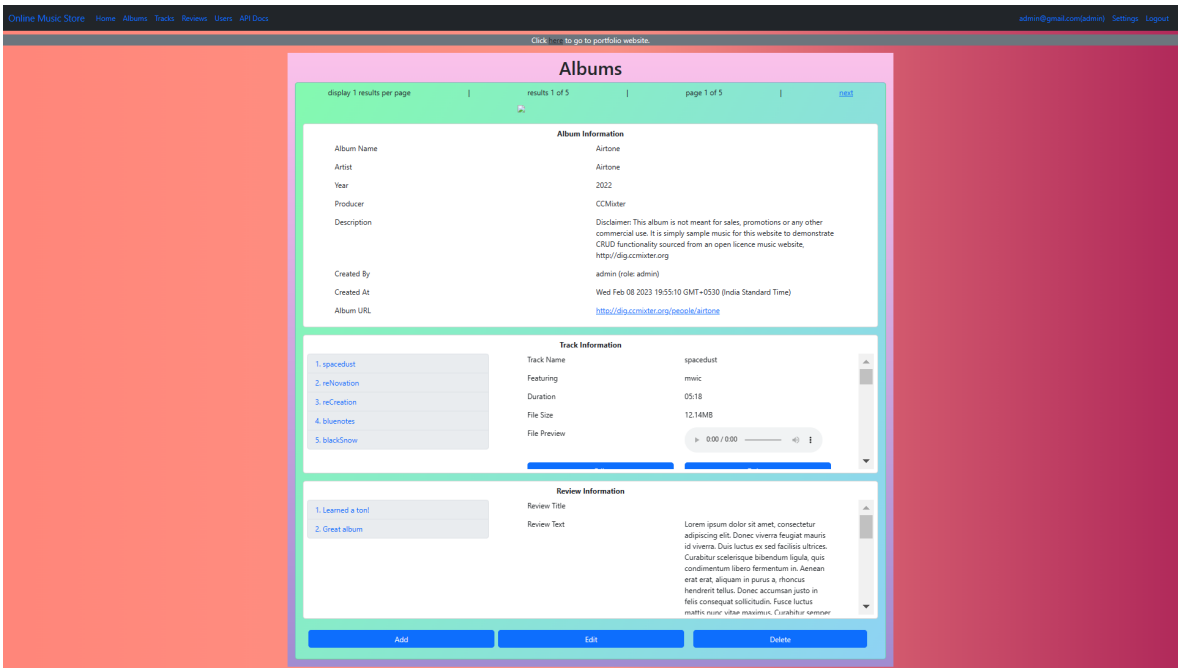
```
      </div>
      <% }) %>
    </div>
  </div>
 </body>
</html>
```

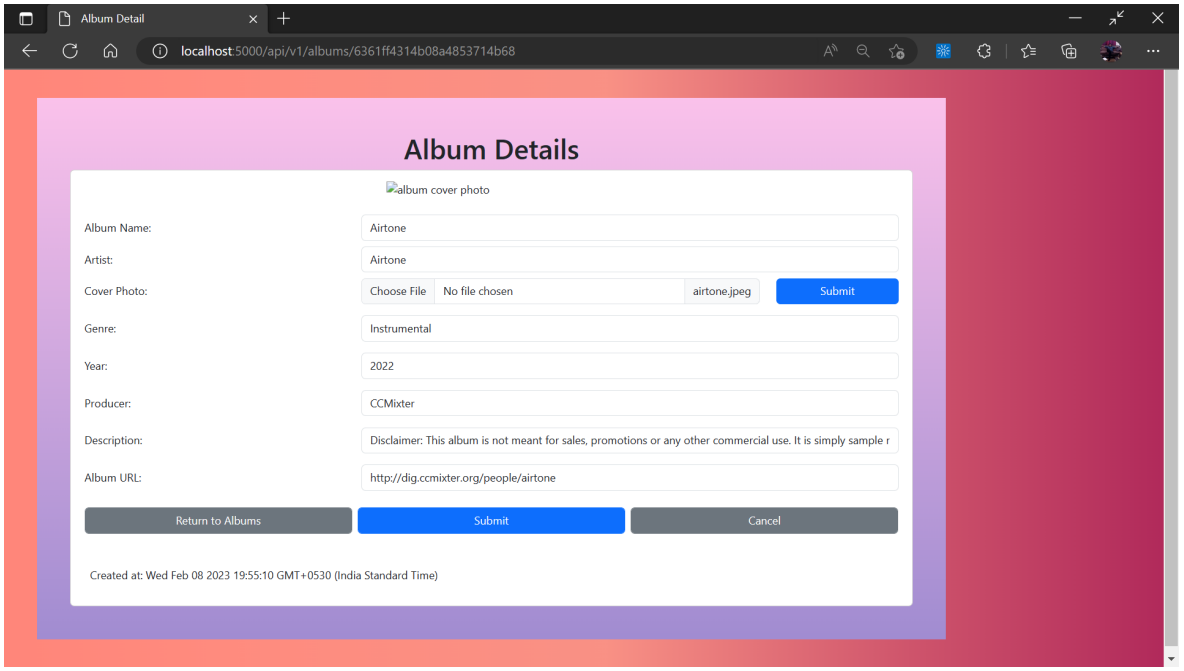
views > album-detail.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
crossorigin="anonymous">
    <style>
      body {
        background-image: linear-gradient(to right, #ff8177 0%, #ff867a 0%, #ff8c7f 21%,
#f99185 52%, #cf556c 78%, #b12a5b 100%);
      }
      .container {
        background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
      }
    </style>
    <title>Album Detail</title>
  </head>
  <body>
    <%- include("navbar", { user }) %>
    <div class="container m-5 g-5 p-5">
      <h1 class="text-center">Album Details</h1>
      <div class="card">
        <% if (result.cover_photo) { %>
          <img class="m-3 z-3 w-25 align-self-center" src="/uploads/albums/<%=
result.artist_slug %>-<%= result.album_slug %>/<%= result.cover_photo %>" alt="album cover
photo">
        <% } %>

        <% if (result.id) { %>
          <!-- put -->
          <form action="/api/v1/albums/<%= result.id %>?_method=PUT" method="POST">
            <input type="hidden" name="_method" value="PUT">
        <% } else { %>
          <!-- post -->
          <form action="/api/v1/albums" method="POST">
        <% } %>

          <%- include("form-template", {form_id: "album_id", form_name: "album_name",
```

```
form_label: "Album Name:", form_type: "text", form_value: result.album_name,
form_placeholder: ""}) %>
            <%- include("form-template", {form_id: "artist", form_name: "artist", form_label:
"Artist:", form_type: "text", form_value: result.artist, form_placeholder: ""}) %>

            <% if (result.id) { %>
              <form action="/api/v1/albums/<%= result.id %>/photo?_method=PUT"
method="POST" enctype="multipart/form-data">
                <input type="hidden" name="_method" value="PUT">
                <div class="m-2 row">
                  <label for="cover_photo" class="col-4 col-form-label">
                    Cover Photo:
                  </label>
                  <div class="col input-group">
                    <input type="file" id="photo" name="photo" class="form-control" value="<%=
result.cover_photo %>"/>
                    <span class="input-group-text"><%= result.cover_photo %></span>
                  </div>
                  <div class="col-2">
                    <input type="submit" class="btn btn-primary input-group" value="Submit"
formaction="/api/v1/albums/<%= result.id %>/photo?_method=PUT"
formenctype="multipart/form-data" formmethod="POST"/>
                  </div>
                </div>
              </form>
            <% } %>

            <%- include("form-template", {form_id: "genre", form_name: "genre", form_label:
"Genre:", form_type: "text", form_value: result.genre, form_placeholder: ""}) %>
            <%- include("form-template", {form_id: "year", form_name: "year", form_label:
"Year:", form_type: "text", form_value: result.year, form_placeholder: ""}) %>
            <%- include("form-template", {form_id: "producer", form_name: "producer",
form_label: "Producer:", form_type: "text", form_value: result.producer, form_placeholder: ""})
%>
            <%- include("form-template", {form_id: "description", form_name: "description",
form_label: "Description:", form_type: "text", form_value: result.description, form_placeholder:
""}) %>
            <%- include("form-template", {form_id: "album_url", form_name: "album_url",
form_label: "Album URL:", form_type: "text", form_value: result.album_url, form_placeholder:
""}) %>

            <div class="m-3 row">
              <a type="button" class="btn btn-secondary form-control col mx-1"
href="/api/v1/albums">Return to Albums</a>
              <% if (result.id) { %>
                <input type="submit" class="btn btn-primary form-control col mx-1"
value="Submit" formaction="/api/v1/albums/<%= result.id %>?_method=PUT"
formmethod="POST"/>
                <a type="button" class="btn btn-secondary form-control col mx-1"
href="/api/v1/albums/<%= result.id %>">Cancel</a>
              <% } else { %>
```

```
                <input type="submit" class="btn btn-primary form-control col mx-1"
value="Submit" formaction="/api/v1/albums" formmethod="POST"/>
                <% } %>
              </div>

              <div class="m-3 row">
                <p><%= msg %></p>
                <% if (result.createdAt) { %>
                  <p>Created at: <%= result.createdAt %></p>
                <% } %>
              </div>
            </form>
          </div>
        </div>
      </body>
</html>
```





controllers > tracks.js

```
…
// @desc    Get all tracks
// @route   GET /api/v1/tracks
// @route   GET /api/v1/albums/:albumId/tracks
// @access  Public
exports.getTracks = asyncHandler(async (req, res, next) => {
…
   if(req.header('accept')==='*/*') {
     res
       .status(200)
       .json({
         success: true,
         count: tracks.length,
         data: tracks,
       });
   } else {
     res
       .status(200)
       .render('tracks', { album_id: req.params.albumId, tracks, user: req.cookies.user });
   }
  } else {
   if(req.header('accept')==='*/*') {
     res
       .status(200)
       .json(res.advancedResults);
   } else {
     res
       .status(200)
       .render('tracks', { tracks: res.advancedResults, user: req.cookies.user });
```

```
  }
 }
});

// @desc    Get track by ID
// @route   GET /api/v1/tracks/:id
// @access  Public
exports.getTrackById = asyncHandler(async (req, res, next) => {
…
 if(req.header('accept')==='*/*') {
  res
   .status(200)
   .json({
    success: true,
    data: track,
   });
 } else {
  res
   .status(200)
   .render('track-detail', { msg: '', track, user: req.cookies.user });
 }
});

// @desc    Get create new track page
// @route   GET /api/v1/albums/:albumId/tracks/newtrack
// @access  Private
exports.getCreateTrack = asyncHandler(async (req, res, next) => {
 const album_id = req.params.albumId;

 res
  .status(200)
  .render('track-detail', {
   msg: '',
   track: {},
   album_id,
   albums: req.cookies.albums,
   user: req.cookies.user
  });
});

// @desc    Create new track
// @route   POST /api/v1/albums/:albumId/tracks
// @access  Private
exports.createTrack = asyncHandler(async (req, res, next) => {
…
 if(req.header('accept')==='*/*') {
  res
   .status(201)
   .json({
    success: true,
    msg: 'Track created successfully',
```

```
    });
  } else {
    res
      .status(201)
      .render('track-detail', {
        msg: 'Track created successfully',
        track,
        user: req.cookies.user
      });
  }
});

// @desc    Update track by ID
// @route   PUT /api/v1/tracks/:id
// @access  Private
exports.updateTrackById = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
      .status(200)
      .json({
        success: true,
        msg: `Track with id ${req.params.id} updated successfully`,
        data: track,
      });
  } else {
    res
      .status(200)
      .render('track-detail', {
        msg: `Track with id ${req.params.id} updated successfully`,
        track,
        user: req.cookies.user
      });
  }
});

// @desc    Delete track by ID
// @route   DELETE /api/v1/tracks/:id
// @access  Private
exports.deleteTrackById = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
      .status(200)
      .json({
        success: true,
        msg: `Track with id ${req.params.id} deleted successfully`,
      });
  } else {
    res
      .status(200)
```

```
    .redirect('/api/v1/tracks');
  }
});

// @desc    Upload an audio track for a track
// @route   PUT /api/v1/tracks/:id/audio
// @access  Private
exports.trackAudioUpload = asyncHandler(async (req, res, next) => {
…
   if(req.header('accept')==='*/*') {
     res
       .status(200)
       .json({
         success: true,
         msg: `Track uploaded to track with id '${req.params.id}' successfully`,
         data: file.name,
       });
   } else {
     res
       .status(200)
       .render('track-detail', {
         msg: `Track uploaded to track with id '${req.params.id}' successfully`,
         track,
         user: req.cookies.user
       });
   }
  });
});
```

routes > tracks.js

```
…
// use protect & authorize middleware
const { protect, authorize } = require('../middleware/auth');

router.route('/')
  .get(
    advancedResults(TrackModel, {
      path: 'album',
      select: 'album_name album_url createdAt album_slug artist_slug',
    }),
    getTracks
  )
  .post(protect, authorize('publisher', 'admin'), createTrack);

router.route('/newtrack').get(protect, authorize('publisher', 'admin'), getCreateTrack);

router.route('/:id').get(getTrackById).put(protect, authorize('publisher', 'admin'),
updateTrackById).delete(protect, authorize('publisher', 'admin'), deleteTrackById);

router.route('/:id/audio').put(protect, authorize('publisher', 'admin'), trackAudioUpload);
```

```
module.exports = router;
```

views > tracks.ejs

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
      rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
      crossorigin="anonymous"
    />
    <style>
      body {
        background-image: linear-gradient(
          to right,
          #ff8177 0%,
          #ff867a 0%,
          #ff8c7f 21%,
          #f99185 52%,
          #cf556c 78%,
          #b12a5b 100%
        );
      }
      .container {
        background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
      }
      .card.m-2.p-2 {
        background-image: linear-gradient(120deg, #84fab0 0%, #8fd3f4 100%);
      }
    </style>
    <title>Tracks</title>
  </head>
  <body>
    <%- include("navbar", { user }); %>
    <div class="container p-2">
      <h1 class="text-center">Tracks</h1>
      <div class="card m-2 p-2">
        <div class="d-flex justify-content-around">
          <% if (tracks.limit) { %>
          <p>display <%= tracks.limit %> results per page</p>
          <p>|</p>
          <% } %> <% if (tracks.pagination) { %>
          <p>
            results <%= tracks.pagination.curr.page %> of <%= tracks.total %>
```

```
  </p>
  <p>|</p>
  <% if (tracks.pagination.prev) { %>
  <p>
   <a href="/api/v1/tracks?page=<%= tracks.pagination.prev.page %>"
    >prev</a
   >
  </p>
  <p>|</p>
  <% } %>
  <p>page <%= tracks.pagination.curr.page %> of <%= tracks.total %></p>
  <% if (tracks.pagination.next) { %>
  <p>|</p>
  <p>
   <a href="/api/v1/tracks?page=<%= tracks.pagination.next.page %>"
    >next</a
   >
  </p>
  <% } %> <% } %>
</div>

<% if (!tracks.data) { %>
<p class="text-center">No tracks added</p>
<p class="text-center">
 <a
  href="/api/v1/albums/<%= album_id %>/tracks/newtrack"
  class="btn btn-primary w-75"
  >Add</a
 >
</p>
<% } else { %> <% tracks.data.map(track => { %>
<div class="card p-2 my-2 mx-2">
 <div class="card-title text-center">
  <strong>Track Information</strong>
 </div>
 <div class="row mx-5">
  <p class="col">Track Name</p>
  <p class="col"><%= track.track_name %></p>
 </div>
 <div class="row mx-5">
  <p class="col">Featuring</p>
  <p class="col"><%= track.featuring %></p>
 </div>
 <div class="row mx-5">
  <p class="col">Track File</p>
  <p class="col"><%= track.track_file %></p>
 </div>
 <div class="row mx-5">
  <p class="col">Preview</p>
  <p class="col">
   <audio controls>
```

```
          <source
            src="/uploads/albums/<%= track.album.artist_slug %>-<%= track.album.album_slug
%>/<%= track.track_file %>"
            type="audio/mp3"
          />
        </audio>
      </p>
    </div>
    <div class="row mx-5">
      <p class="col">Duration</p>
      <p class="col"><%= track.duration %></p>
    </div>
    <div class="row mx-5">
      <p class="col">File Size</p>
      <p class="col"><%= track.file_size %></p>
    </div>
    <div class="row mx-5">
      <p class="col">Credits</p>
      <p class="col"><%- track.credit %></p>
    </div>
    <div class="row mx-5">
      <p class="col">created At</p>
      <p class="col"><%= track.createdAt %></p>
    </div>
    <div class="row mx-5">
      <p class="col">Album Name</p>
      <p class="col"><%= track.album.album_name %></p>
    </div>
    <div class="row mx-5">
      <p class="col">Album URL</p>
      <p class="col">
        <a href="<%= track.album.album_url %>"
          ><%= track.album.album_url %></a
        >
      </p>
    </div>
    <div class="row mx-5">
      <p class="col">Album Created At</p>
      <p class="col"><%= track.album.createdAt %></p>
    </div>
  </div>

  <div class="row p-2 my-2 mx-2">
    <a
      class="btn btn-primary col mx-2"
      href="/api/v1/albums/<%= track.album.id %>/tracks/newtrack"
      >Add</a
    >
    <a
      class="btn btn-primary col mx-2"
      href="/api/v1/tracks/<%= track.id %>"
```

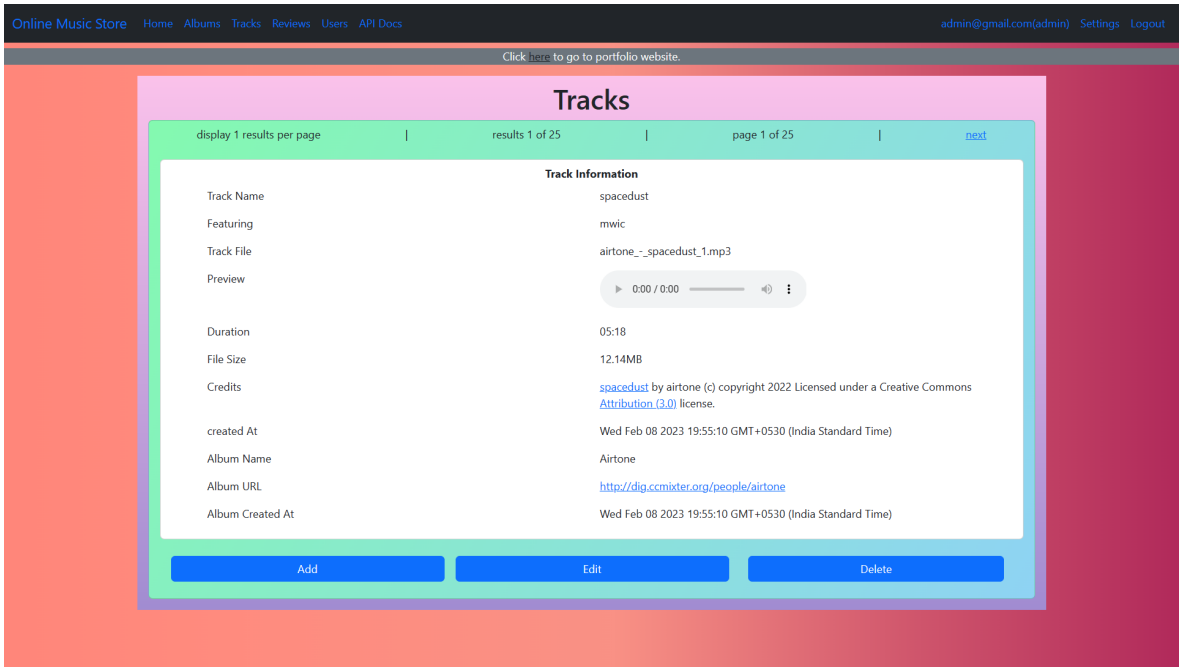
```
        >Edit</a
       >
       <form
        class="col mx-2"
        action="/api/v1/tracks/<%= track.id %>?_method=DELETE"
        method="post"
       >
        <input type="hidden" name="_method" value="DELETE" />
        <input class="btn btn-primary w-100" type="submit" value="Delete" />
       </form>
      </div>
      <% }) %> <% } %>
    </div>
   </div>
  </body>
</html>
```

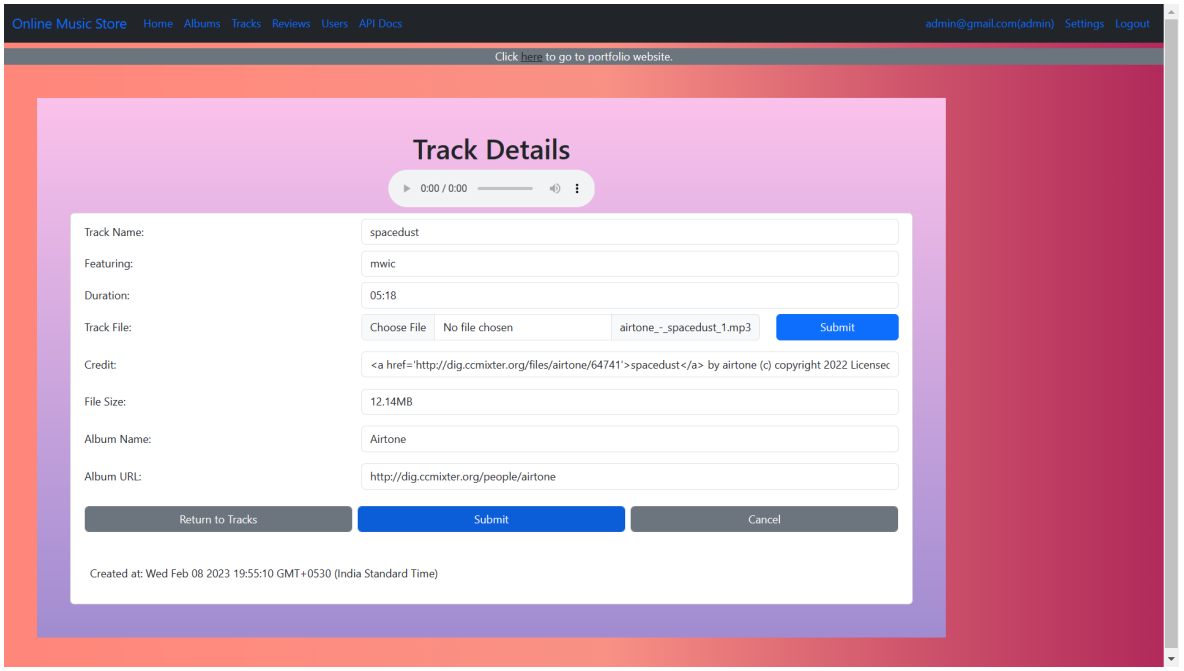
views > track-detail.ejs

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
crossorigin="anonymous">
      <style>
         body {
            background-image: linear-gradient(to right, #ff8177 0%, #ff867a 0%, #ff8c7f 21%,
#f99185 52%, #cf556c 78%, #b12a5b 100%);
         }
         .container {
            background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
         }
      </style>
      <title>Track Detail</title>
   </head>
   <body>
      <%- include("navbar", { user }); %>
      <div class="container m-5 g-5 p-5">
         <h1 class="text-center">Track Details</h1>
         <% if (track.track_file) { %>
            <div class="d-flex justify-content-center my-2">
               <audio controls>
                  <source src="/uploads/albums/<%= track.album.artist_slug %>-<%=
track.album.album_slug %>/<%= track.track_file %>" type="audio/mp3">
               </audio>
            </div>
```

```
        <% } %>
        <div class="card">
          <% if (track.id) { %>
            <!-- put -->
            <form action="/api/v1/tracks/<%= track.id %>?_method=PUT" method="POST">
              <input type="hidden" name="_method" value="PUT">
          <% } else { %>
            <!-- post -->
            <form action="/api/v1/albums/<%= album_id %>/tracks" method="POST">
          <% } %>
            <%- include("form-template", {form_id: "track_name", form_name: "track_name",
form_label: "Track Name:", form_type: "text", form_value: track.track_name, form_placeholder:
""}) %>
            <%- include("form-template", {form_id: "featuring", form_name: "featuring",
form_label: "Featuring:", form_type: "text", form_value: track.featuring, form_placeholder: ""})
%>
            <%- include("form-template", {form_id: "duration", form_name: "duration",
form_label: "Duration:", form_type: "text", form_value: track.duration, form_placeholder: ""}) %>
            <%# include("form-template", {form_id: "track_file", form_name: "track_file",
form_label: "Track File:", form_type: "text", form_value: track.track_file, form_placeholder: ""})
%>
            <% if (track.id) { %>
              <form action="/api/v1/tracks/<%= track.id %>/audio?_method=PUT"
method="POST" enctype="multipart/form-data">
                <input type="hidden" name="_method" value="PUT">
                <div class="m-2 row">
                  <label for="audio" class="col-4 col-form-label">
                    Track File:
                  </label>
                  <div class="col input-group">
                    <input type="file" id="audio" name="audio" class="form-control" value="<%=
track.track_file %>"/>
                    <span class="input-group-text"><%= track.track_file %></span>
                  </div>
                  <div class="col-2">
                    <input type="submit" class="btn btn-primary input-group" value="Submit"
formaction="/api/v1/tracks/<%= track.id %>/audio?_method=PUT"
formenctype="multipart/form-data" formmethod="POST"/>
                  </div>
                </div>
              </form>
            <% } %>
            <%- include("form-template", {form_id: "credit", form_name: "credit", form_label:
"Credit:", form_type: "text", form_value: track.credit, form_placeholder: ""}) %>
            <%- include("form-template", {form_id: "file_size", form_name: "file_size", form_label:
"File Size:", form_type: "text", form_value: track.file_size, form_placeholder: ""}) %>
            <% if (!track.album) { %>
              <% if (!album_id) { %>
                <div class="m-2 row">
                  <label for="album" class="col-4 col-form-label">Album:</label>
                  <div class="col">
```

```
                        <select class="form-select" id="album" name="album" aria-label="select">
                          <option selected>Select Album</option>
                          <% albums.map(album => { %>
                            <option value="<%= album.id %>"><%= album.album_name %></option>
                          <% }) %>
                        </select>
                      </div>
                    </div>
                  <% } else { %>
                    <%- include("form-template", {form_id: "album", form_name: "album",
form_label: "Album ID:", form_type: "text", form_value: album_id, form_placeholder: ""}) %>
                  <% } %>
                <% } else { %>
                  <%- include("form-template", {form_id: "album_name", form_name: "album_name",
form_label: "Album Name:", form_type: "text", form_value: track.album.album_name,
form_placeholder: ""}) %>
                  <%- include("form-template", {form_id: "album_url", form_name: "album_url",
form_label: "Album URL:", form_type: "text", form_value: track.album.album_url,
form_placeholder: ""}) %>
                <% } %>

                <div class="m-3 row">
                  <a type="button" class="btn btn-secondary form-control col mx-1"
href="/api/v1/tracks">Return to Tracks</a>
                  <% if (track.id) { %>
                    <input type="submit" class="btn btn-primary form-control col mx-1"
value="Submit" formaction="/api/v1/tracks/<%= track.id %>?_method=PUT"
formmethod="POST"/>
                    <a type="button" class="btn btn-secondary form-control col mx-1"
href="/api/v1/tracks/<%= track.id %>">Cancel</a>
                  <% } else { %>
                    <input type="submit" class="btn btn-primary form-control col mx-1"
value="Submit" formaction="/api/v1/albums/<%= album_id %>/tracks" formmethod="POST"/>
                  <% } %>
                </div>

                <div class="m-3 row">
                  <p><%= msg %></p>
                  <% if (track.createdAt) { %>
                    <p>Created at: <%= track.createdAt %></p>
                  <% } %>
                </div>
              </form>
            </div>
          </div>
        </body>
      </html>
```





controllers > reviews.js

```
...
// @desc    Get all reviews
// @route   GET /api/v1/reviews
// @route   GET /api/v1/albums/:albumId/reviews
// @access  Public
exports.getReviews = asyncHandler(async (req, res, next) => {
...
   if(req.header('accept')==='*/*') {
     return res
```

```
      .status(200)
      .json({
        success: true,
        count: reviews.length,
        data: reviews,
      });
   } else {
     res
      .status(200)
      .render('reviews', { album_id: req.params.albumId, reviews, user: req.cookies.user });
   }
  } else {
   if(req.header('accept')==='*/*') {
     res
      .status(200)
      .json(res.advancedResults);
   } else {
     res
      .status(200)
      .render('reviews', { reviews: res.advancedResults, user: req.cookies.user });
   }
  }
});

// @desc    Get review by ID
// @route   GET /api/v1/reviews/:id
// @access  Public
exports.getReviewById = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
     .status(200)
     .json({
       success: true,
       data: review
     });
  } else {
    res
     .status(200)
     .render('review-detail', { msg: '', review, user: req.cookies.user });
  }
});

// @desc    Get create new review page
// @route   GET /api/v1/albums/:albumId/reviews/newreview
// @access  Private/User
exports.getCreateReview = asyncHandler(async (req, res, next) => {
  const album_id = req.params.albumId;

  res
    .status(200)
```

```
    .render('review-detail', {
      msg: '',
      review: {},
      album_id,
      albums: req.cookies.albums,
      user: req.cookies.user
    });
});

// @desc    Create a review
// @route   POST /api/v1/albums/:albumId/reviews
// @access  Private/User
exports.createReview = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
      .status(201)
      .json({
        success: true,
        msg: 'Review created successfully',
        data: review
      });
  } else {
    res
      .status(201)
      .render('review-detail', {
        msg: 'Review created successfully',
        review,
        user: req.cookies.user
      });
  }
});

// @desc    Update review by ID
// @route   PUT /api/v1/reviews/:id
// @access  Private/User
exports.updateReviewById = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
      .status(200)
      .json({
        success: true,
        msg: `Review with id ${req.params.id} updated successfully`,
        data: review,
      });
  } else {
    res
      .status(200)
      .render('review-detail', {
        msg: `Review with id ${req.params.id} updated successfully`,
```

```
      review,
      user: req.cookies.user
    });
  }
});

// @desc    Delete a review
// @route   DELETE /api/v1/reviews/:id
// @access  Private/User
exports.deleteReviewById = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
      .status(200)
      .json({
        success: true,
        msg: `Review with id ${req.params.id} deleted successfully`,
      });
  } else {
    res
      .status(200)
      .redirect('/api/v1/reviews');
  }
});
```

routes > reviews.js

```
…
// use protect & authorize
const { protect, authorize } = require('../middleware/auth');

router.route('/')
  .get(
    advancedResults(ReviewModel, {
      path: 'album',
      select: 'album_name description',
    }),
    getReviews
  )
  .post(protect, authorize('user', 'admin'), createReview);

router.route('/newreview').get(protect, authorize('user', 'admin'), getCreateReview);

router.route('/:id').get(getReviewById).put(protect, authorize('user', 'admin'),
updateReviewById).delete(protect, authorize('user', 'admin'), deleteReviewById);

module.exports = router;
```

views > reviews.ejs

```
<!DOCTYPE html>
```

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
      rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
      crossorigin="anonymous"
    />
    <style>
      body {
        background-image: linear-gradient(
          to right,
          #ff8177 0%,
          #ff867a 0%,
          #ff8c7f 21%,
          #f99185 52%,
          #cf556c 78%,
          #b12a5b 100%
        );
      }
      .container {
        background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
      }
      .card.m-2.p-2 {
        background-image: linear-gradient(120deg, #84fab0 0%, #8fd3f4 100%);
      }
    </style>
    <title>Reviews</title>
  </head>
  <body>
    <%- include("navbar", { user }) %>
    <div class="container p-2">
      <h1 class="text-center">Reviews</h1>
      <div class="card m-2 p-2">
        <div class="d-flex justify-content-around">
          <% if (reviews.limit) { %>
          <p>display <%= reviews.limit %> results per page</p>
          <p>|</p>
          <% } %> <% if (reviews.pagination) { %>
          <p>
            results <%= reviews.pagination.curr.page %> of <%= reviews.total %>
          </p>
          <p>|</p>
          <% if (reviews.pagination.prev) { %>
          <p>
            <a href="/api/v1/reviews?page=<%= reviews.pagination.prev.page %>"
              >prev</a
```

```
    >
  </p>
  <p>|</p>
  <% } %>
  <p>
    page <%= reviews.pagination.curr.page %> of <%= reviews.total %>
  </p>
  <% if (reviews.pagination.next) { %>
  <p>|</p>
  <p>
    <a href="/api/v1/reviews?page=<%= reviews.pagination.next.page %>"
      >next</a
    >
  </p>
  <% } %> <% } %>
</div>

<% if (!reviews.data) { %>
<p class="text-center">No reviews added</p>
<p class="text-center">
  <a
    href="/api/v1/albums/<%= album_id %>/reviews/newreview"
    class="btn btn-primary w-75"
    >Add</a
  >
</p>
<% } else { %> <% reviews.data.map(review => { %>
<div class="card p-2 my-2 mx-2">
  <div class="card-title text-center">
    <strong>Review Information</strong>
  </div>
  <div class="row mx-5">
    <p class="col">Review Title</p>
    <p class="col"><%= review.title %></p>
  </div>
  <div class="row mx-5">
    <p class="col">Review text</p>
    <p class="col"><%= review.text %></p>
  </div>
  <div class="row mx-5">
    <p class="col">Rating</p>
    <p class="col"><%= review.rating %></p>
  </div>
  <div class="row mx-5">
    <p class="col">created At</p>
    <p class="col"><%= review.createdAt %></p>
  </div>
  <div class="row mx-5">
    <p class="col">Album Name</p>
    <p class="col"><%= review.album.album_name %></p>
  </div>
```

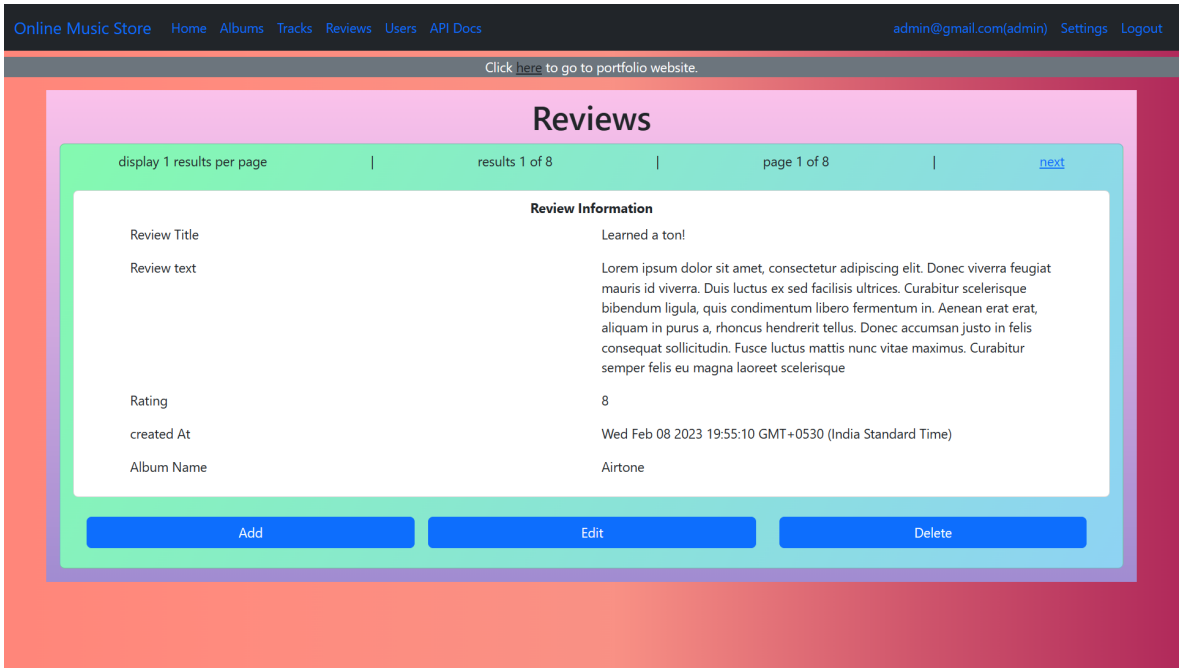
```
      </div>

      <div class="row p-2 my-2 mx-2">
       <a class="btn btn-primary col mx-2" href="/api/v1/reviews/newreview"
        >Add</a
       >
       <a
        class="btn btn-primary col mx-2"
        href="/api/v1/reviews/<%= review.id %>"
        >Edit</a
       >
       <form
        class="col mx-2"
        action="/api/v1/reviews/<%= review.id %>?_method=DELETE"
        method="post"
       >
        <input type="hidden" name="_method" value="DELETE" />
        <input class="btn btn-primary w-100" type="submit" value="Delete" />
       </form>
      </div>
      <% }) %> <% } %>
    </div>
   </div>
  </body>
</html>
```

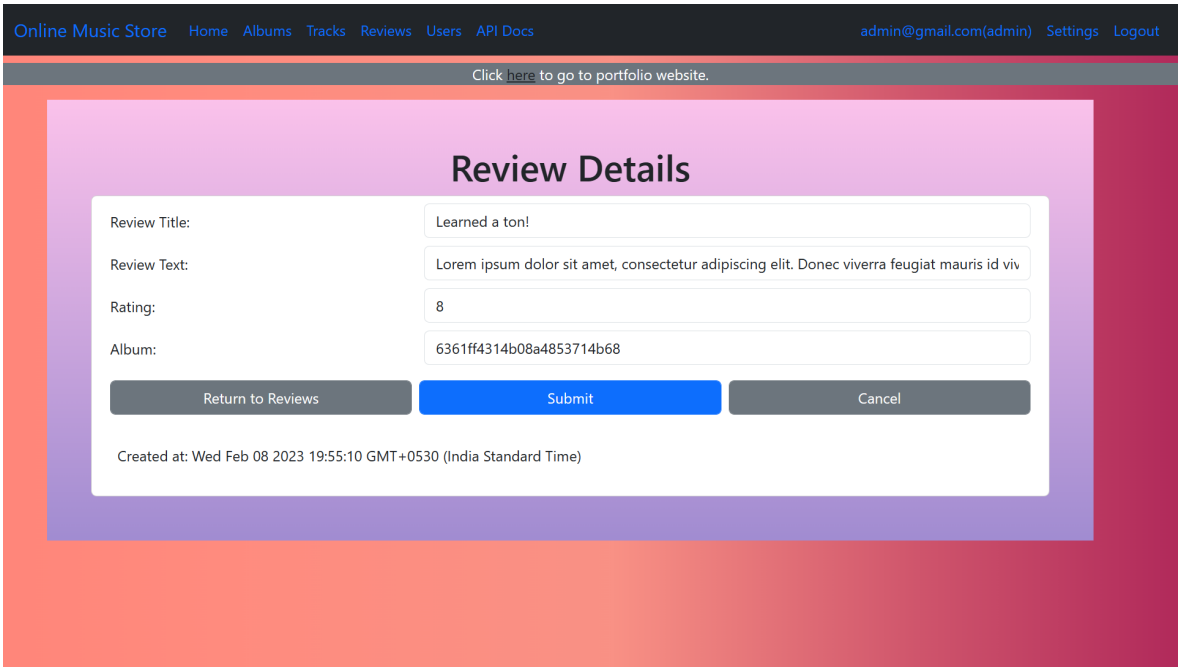
views > review-detail.ejs

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
crossorigin="anonymous">
      <style>
         body {
            background-image: linear-gradient(to right, #ff8177 0%, #ff867a 0%, #ff8c7f 21%,
#f99185 52%, #cf556c 78%, #b12a5b 100%);
         }
         .container {
            background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
         }
      </style>
      <title>Review Detail</title>
   </head>
   <body>
      <%- include("navbar", { user }) %>
```

```
    <div class="container mx-5 gx-5 p-5">
      <h1 class="text-center">Review Details</h1>
      <div class="card">
        <% if (review.id) { %>
          <!-- put -->
          <form action="/api/v1/reviews/<%= review.id %>?_method=PUT" method="POST">
            <input type="hidden" name="_method" value="PUT">
        <% } else { %>
          <!-- post -->
          <form action="/api/v1/albums/<%= album_id %>/reviews/newreview"
method="POST">
        <% } %>
          <%- include("form-template", {form_id: "title", form_name: "title", form_label: "Review
Title:", form_type: "text", form_value: review.title, form_placeholder: ""}) %>
          <%- include("form-template", {form_id: "text", form_name: "text", form_label: "Review
Text:", form_type: "text", form_value: review.text, form_placeholder: ""}) %>
          <%- include("form-template", {form_id: "rating", form_name: "rating", form_label:
"Rating:", form_type: "number", form_value: review.rating, form_placeholder: ""}) %>
          <% if (review.id) { %>
            <%- include("form-template", {form_id: "album", form_name: "album", form_label:
"Album:", form_type: "text", form_value: review.album.id, form_placeholder: ""}) %>
          <% } else { %>
            <% if (!review.album) { %>
              <% if (!album_id) { %>
                <div class="m-2 row">
                  <label for="album" class="col-4 col-form-label">Album:</label>
                  <div class="col">
                    <select class="form-select" id="album" name="album" aria-label="select">
                      <option selected>Select Album</option>
                      <% albums.map(album => { %>
                        <option value="<%= album.id %>"><%= album.album_name
%></option>
                      <% }) %>
                    </select>
                  </div>
                </div>
              <% } else { %>
                <%- include("form-template", {form_id: "album", form_name: "album",
form_label: "Album ID:", form_type: "text", form_value: album_id, form_placeholder: ""}) %>
              <% } %>
            <% } %>
          <% } %>

          <div class="m-3 row">
            <a type="button" class="btn btn-secondary form-control col mx-1"
href="/api/v1/reviews">Return to Reviews</a>
            <% if (review.id) { %>
              <input type="submit" class="btn btn-primary form-control col mx-1"
value="Submit" formaction="/api/v1/reviews/<%= review.id %>?_method=PUT"
formmethod="POST"/>
              <a type="button" class="btn btn-secondary form-control col mx-1"
```

```
href="/api/v1/reviews/<%= review.id %>">Cancel</a>
                <% } else { %>
                  <input type="submit" class="btn btn-primary form-control col mx-1"
value="Submit" formaction="/api/v1/albums/<%= album_id %>/reviews" formmethod="POST"/>
                <% } %>
              </div>

              <div class="m-3 row">
                <p><%= msg %></p>
                <% if (review.createdAt) { %>
                  <p>Created at: <%= review.createdAt %></p>
                <% } %>
              </div>
            </form>
          </div>
        </div>
      </body>
</html>
```





controllers > users.js

```
const ErrorResponse = require('../utils/errorResponse');
const asyncHandler = require('../middleware/async');
const UserModel = require('../models/User');

// @desc    Get all users
// @route   GET /api/v1/auth/users
// @access  Private/Admin
exports.getUsers = asyncHandler(async (req, res, next) => {
  if(req.header('accept')==='*/*') {
    res
      .status(200)
      .json(res.advancedResults);
  } else {
    res
      .status(200)
      .render('users', {results: res.advancedResults, user: req.cookies.user});
  }
});

// @desc    Get user by ID
// @route   GET /api/v1/auth/users/:id
// @access  Private/Admin
exports.getUserById = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
      .status(200)
      .json({
        success: true,
```

```
      data: result
    });
  } else {
   res
    .status(200)
    .render('user-detail', { msg: '', result, user: req.cookies.user });
  }
});

// @desc    Create a user
// @route   POST /api/v1/auth/users
// @access  Private/Admin
exports.createUser = asyncHandler(async (req, res, next) => {
 …
  if(req.header('accept')==='*/*') {
   res
    .status(201)
    .json({
      success: true,
      msg: 'User created successfully',
      data: user
    });
  } else {
   res
    .status(201)
    .render('user-detail', {
      msg: 'User created successfully',
      result,
      user: req.cookies.user
    });
  }
});

// @desc    Update a user
// @route   PUT /api/v1/auth/users/:id
// @access  Private/Admin
exports.updateUserById = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
   res
    .status(200)
    .json({
      success: true,
      msg: `User with id ${req.params.id} updated successfully`,
      data: user,
    });
   } else {
    res
    .status(200)
    .render('user-detail', {
      msg: `User with id ${req.params.id} updated successfully`,
```

```
      result,
      user: req.cookies.user
    });
  }
});

// @desc    Delete a user
// @route   DELETE /api/v1/auth/users/:id
// @access  Private/Admin
exports.deleteUserById = asyncHandler(async (req, res, next) => {
…
  if(req.header('accept')==='*/*') {
    res
      .status(200)
      .json({
        success: true,
        msg: `User with id ${req.params.id} deleted successfully`,
      });
  } else {
    res
      .status(200)
      .redirect('/api/v1/users');
  }
});
```

routes > users.js

```
…
router.route('/').get(advancedResults(User), getUsers).post(createUser);
router.route('/:id').get(getUserById).put(updateUserById).delete(deleteUserById);

module.exports = router;
```

views > users.js

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
      rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
      crossorigin="anonymous"
    />
    <style>
      body {
        background-image: linear-gradient(
```

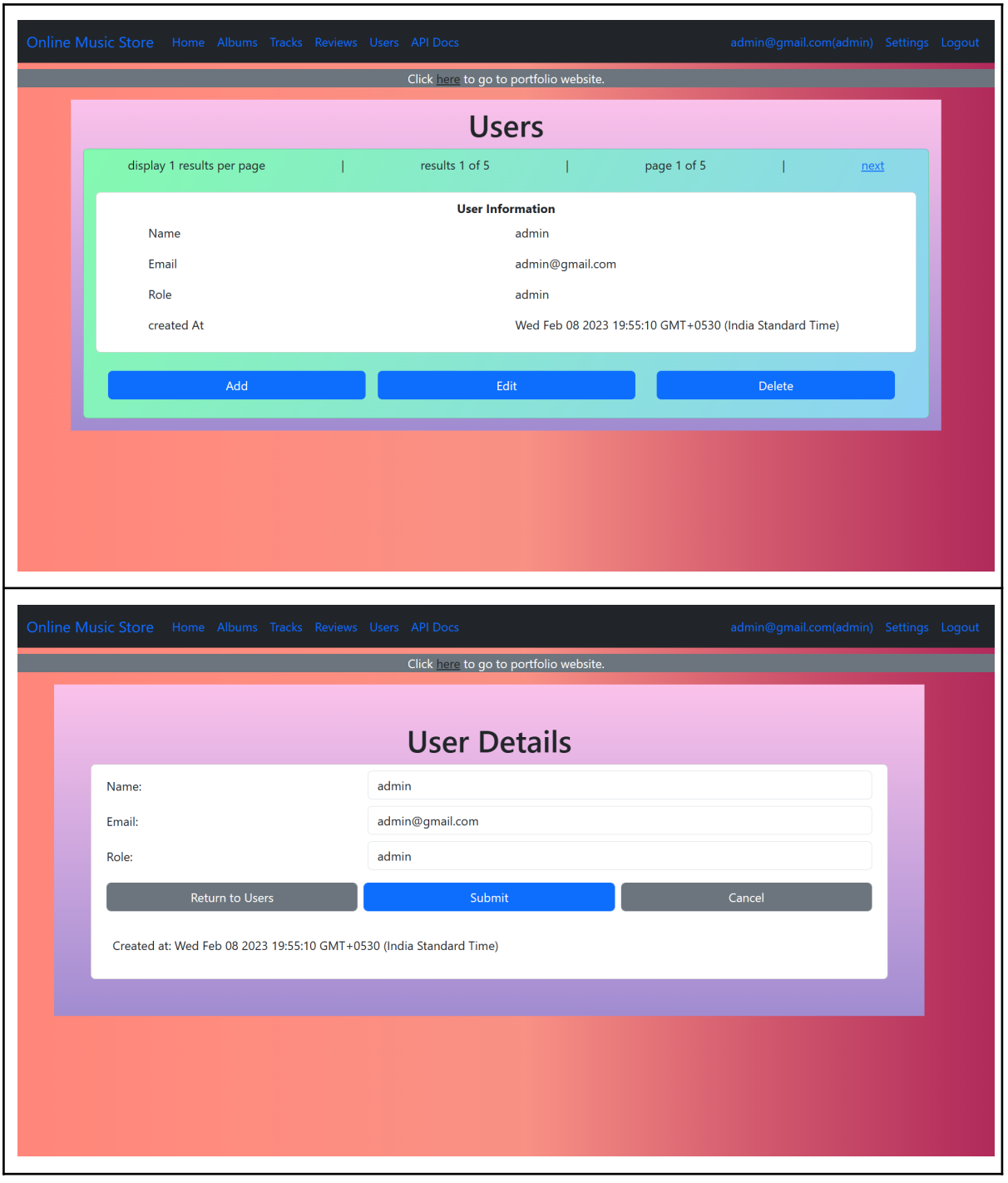
```
      to right,
      #ff8177 0%,
      #ff867a 0%,
      #ff8c7f 21%,
      #f99185 52%,
      #cf556c 78%,
      #b12a5b 100%
    );
  }
  .container {
    background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
  }
  .card.m-2.p-2 {
    background-image: linear-gradient(120deg, #84fab0 0%, #8fd3f4 100%);
  }
 </style>
 <title>Users</title>
</head>
<body>
 <%- include("navbar", { user }) %>
 <div class="container p-2">
  <h1 class="text-center">Users</h1>
  <div class="card m-2 p-2">
   <div class="d-flex justify-content-around">
    <p>display <%= results.limit %> results per page</p>
    <p>|</p>
    <p>
     results <%= results.pagination.curr.page %> of <%= results.total %>
    </p>
    <p>|</p>
    <% if (results.pagination.prev) { %>
    <p>
     <a
      href="/api/v1/auth/users?page=<%= results.pagination.prev.page %>"
      >prev</a
     >
    </p>
    <p>|</p>
    <% } %>
    <p>
     page <%= results.pagination.curr.page %> of <%= results.total %>
    </p>
    <% if (results.pagination.next) { %>
    <p>|</p>
    <p>
     <a
      href="/api/v1/auth/users?page=<%= results.pagination.next.page %>"
      >next</a
     >
    </p>
    <% } %>
```

```
    </div>

    <% results.data.map(result => { %>
    <div class="card p-2 my-2 mx-2">
      <div class="card-title text-center">
        <strong>User Information</strong>
      </div>
      <div class="row mx-5">
        <p class="col">Name</p>
        <p class="col"><%= result.name %></p>
      </div>
      <div class="row mx-5">
        <p class="col">Email</p>
        <p class="col"><%= result.email %></p>
      </div>
      <div class="row mx-5">
        <p class="col">Role</p>
        <p class="col"><%= result.role %></p>
      </div>
      <div class="row mx-5">
        <p class="col">created At</p>
        <p class="col"><%= result.createdAt %></p>
      </div>
    </div>

    <div class="row p-2 my-2 mx-2">
      <a class="btn btn-primary col mx-2" href="/api/v1/auth/users/newuser"
        >Add</a
      >
      <a
        class="btn btn-primary col mx-2"
        href="/api/v1/auth/users/<%= result.id %>"
        >Edit</a
      >
      <form
        class="col mx-2"
        action="/api/v1/auth/users/<%= result.id %>?_method=DELETE"
        method="post"
      >
        <input type="hidden" name="_method" value="DELETE" />
        <input class="btn btn-primary w-100" type="submit" value="Delete" />
      </form>
    </div>
    <% }) %>
   </div>
  </div>
 </body>
</html>
```

views > user-detail.js

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
      rel="stylesheet"

integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
      crossorigin="anonymous"
    />
    <style>
      body {
        background-image: linear-gradient(
          to right,
          #ff8177 0%,
          #ff867a 0%,
          #ff8c7f 21%,
          #f99185 52%,
          #cf556c 78%,
          #b12a5b 100%
        );
      }
      .container {
        background-image: linear-gradient(to top, #a18cd1 0%, #fbc2eb 100%);
      }
    </style>
    <title>User Detail</title>
  </head>
  <body>
    <%- include("navbar", { user }) %>
    <div class="container mx-5 gx-5 p-5">
      <h1 class="text-center">User Details</h1>
      <div class="card">
        <form
          action="/api/v1/auth/users/<%= result.id %>?_method=PUT"
          method="POST"
        >
          <input type="hidden" name="_method" value="PUT" />
          <%- include("form-template", {form_id: "name", form_name: "name",
          form_label: "Name:", form_type: "text", form_value: result.name,
          form_placeholder: ""}) %> <%- include("form-template", {form_id:
          "email", form_name: "email", form_label: "Email:", form_type: "text",
          form_value: result.email, form_placeholder: ""}) %> <%-
          include("form-template", {form_id: "role", form_name: "role",
          form_label: "Role:", form_type: "text", form_value: result.role,
          form_placeholder: ""}) %>

          <div class="m-3 row">
```

```
        <a
          type="button"
          class="btn btn-secondary form-control col mx-1"
          href="/api/v1/auth/users"
          >Return to Users</a
        >
        <input
          type="submit"
          class="btn btn-primary form-control col mx-1"
          value="Submit"
          formaction="/api/v1/auth/users/<%= result.id %>?_method=PUT"
          formmethod="POST"
        />
        <a
          type="button"
          class="btn btn-secondary form-control col mx-1"
          href="/api/v1/auth/users/<%= result.id %>"
          >Cancel</a
        >
       </div>

       <div class="m-3 row">
        <p><%= msg %></p>
        <% if(result.createdAt) { %>
          <p>Created at: <%= result.createdAt %></p>
        <% } %>
       </div>
      </form>
    </div>
   </div>
  </body>
</html>
```



API Security

Prevent NoSQL Injection and Sanitize Data

Prevent unauthorized users from hacking into the application using $gt fields.

Install express-mongo-sanitize using the following command

npm i express-mongo-sanitize

Login a user using Postman
Send POST {{URL}}/api/v1/auth/login with following data
{

```
   "email": {"$gt": ""},
   "password": "123456"
}
```

Postman console output

```
{
   "success": true,
   "msg": "User logged in successfully",
   "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzODIzMzBmOGY4OTYwODE1N2Q0YTc5NSIsImlhdCI6MTY3MDU5Nzg3NCwiZXhwIjoxNjcwNjE5NDc0fQ.nLYAFTrnzYU5P9_FxmIplpyILtIgDDUklemHqOXrVsE"
}
```

server.js

```
// import the necessary modules
…
const cookieParser = require('cookie-parser');
const mongoSanitize = require('express-mongo-sanitize');
…
// cookie-parser
app.use(cookieParser());

// express-mongo-sanitize prevent noSQL injections
app.use(mongoSanitize());
…
```

Send POST {{URL}}/api/v1/auth/login with following data

```
{
   "email": {"$gt": ""},
   "password": "123456"
}
```

Postman console output

```
{
   "success": false,
   "error": "Resource with id '[object Object]' not found"
}
```

XSS Protection and Security Headers

Add security headers to HTTP requests to improve security using helmet, an npm package.
Prevent cross side scripting using xss-clean, an npm package. In cross side scripting, script within tags can be posted to database, which could include malicious code, and XSS clean will stringify any scripts sent in the database via APIs.
Install helmet & xss-clean using the following command
npm i helmet xss-clean

server.js

```
…
const mongoSanitize = require('express-mongo-sanitize');
const helmet = require('helmet');
…
// express-mongo-sanitize prevent noSQL injections
app.use(mongoSanitize());

// secure HTTP headers
app.use(helmet());
…
```

Create a new album using Postman with the following data

```
{
   "album_name": "Siobhan Dakay<script>alert(1)</script>",
   "cover_photo": "siobhan_dakay.jpg",
   "artist": "Siobhan Dakay",
   "genre": "Instrumental",
   "year": 2007,
   "producer": "CCMixter",
   "description": "Disclaimer: This album is not meant for sales, promotions or any other commercial use. It is simply sample music for this website to demonstrate CRUD functionality sourced from an open licence music website, http://dig.ccmixter.org",
   "album_url": "http://dig.ccmixter.org/people/SiobhanD"
}
```

Postman console output

```
{
   "success": true,
   "msg": "Album created successfully"
}
```

```
{
   "success": true,
  …
   "data": [
      {
         "_id": "639352e40fdc0696c8eefa86",
         "album_name": "Siobhan Dakay<script>alert(1)</script>",
         "cover_photo": "siobhan_dakay.jpg",
         "album_slug": "siobhan-dakaylessscriptgreateralert(1)lessscriptgreater",
         "artist_slug": "siobhan-dakay",
         "tracks": [],
         "id": "639352e40fdc0696c8eefa86"
      },
…
```

server.js

```
const helmet = require('helmet');
const xss = require('xss-clean');
```

```
…
// secure HTTP headers
app.use(helmet());

// prevent cross-side-scripting attacks
app.use(xss());
…
```

Create a new album using Postman with the above data once again

Postman console output

```
{
   "success": true,
   "msg": "Album created successfully"
}
```

```
{
   "success": true,
…
   "data": [
      {
         "_id": "63935378db06c972b2ed7890",
         "album_name": "Siobhan Dakay&lt;script>alert(1)&lt;/script>",
         "cover_photo": "siobhan_dakay.jpg",
         "album_slug": "siobhan-dakayandltscriptgreateralert(1)andltscriptgreater",
         "artist_slug": "siobhan-dakay",
         "tracks": [],
         "id": "63935378db06c972b2ed7890"
      },
…
```

Rate Limiting, HPP and CORS

Limit the rate of requests sent by users of the API using express-rate-limit, an npm package. Prevent HTTP param pollution (populating if HTTP request params with same name in an array by attackers) using hpp, an npm package. Enable Cross Origin Resource Sharing (CORS) using cors, an npm package. Cross Origin allows the API to be accessed from any domain (making the API versatile to use with any frontend application.

Install express-rate-limit, hpp and cors using the following command

npm i express-rate-limit hpp cors

server.js

```
…
const xss = require('xss-clean');
const rateLimit = require('express-rate-limit');
…
// prevent cross-side-scripting attacks
app.use(xss());

// limit API requests to 5
const limiter = rateLimit({
```

```
  windowMs: 15 * 60 * 1000, // 15 Minutes
  max: 5 // limit each IP upto 100 requests per windowMs
});

app.use(limiter);
```

Send GET {{URL}}/api/v1/albums?limit=5 using Postman 6 times in a row

Postman console output

Too many requests, please try again later.

server.js

```
…
const rateLimit = require('express-rate-limit');
const hpp = require('hpp');
…
app.use(limiter);

// prevent HTTP param pollution
app.use(hpp());
…
```

server.js

```
…
const rateLimit = require('express-rate-limit');
const hpp = require('hpp');
const cors = require('cors');
…
app.use(limiter);

// prevent HTTP param pollution
app.use(hpp());

//enable CORS
app.use(cors());
…
```

Note: Set the API request rate limit to 100 for conservative usage.

Documentation - Swagger

install using npm swagger-jsdoc, which will create the swagger API docs and swagger-ui-express, which will publish the swagger docs to the UI using express and @types for swagger-jsdoc and swagger-ui-express for types.

npm i swagger-jsdoc swagger-ui-express

npm i --save-dev @types/swagger-jsdoc @types/swagger-ui-express

utils > swagger.js

```
const swaggerJSDoc = require('swagger-jsdoc');
const swaggerUi = require('swagger-ui-express');
```

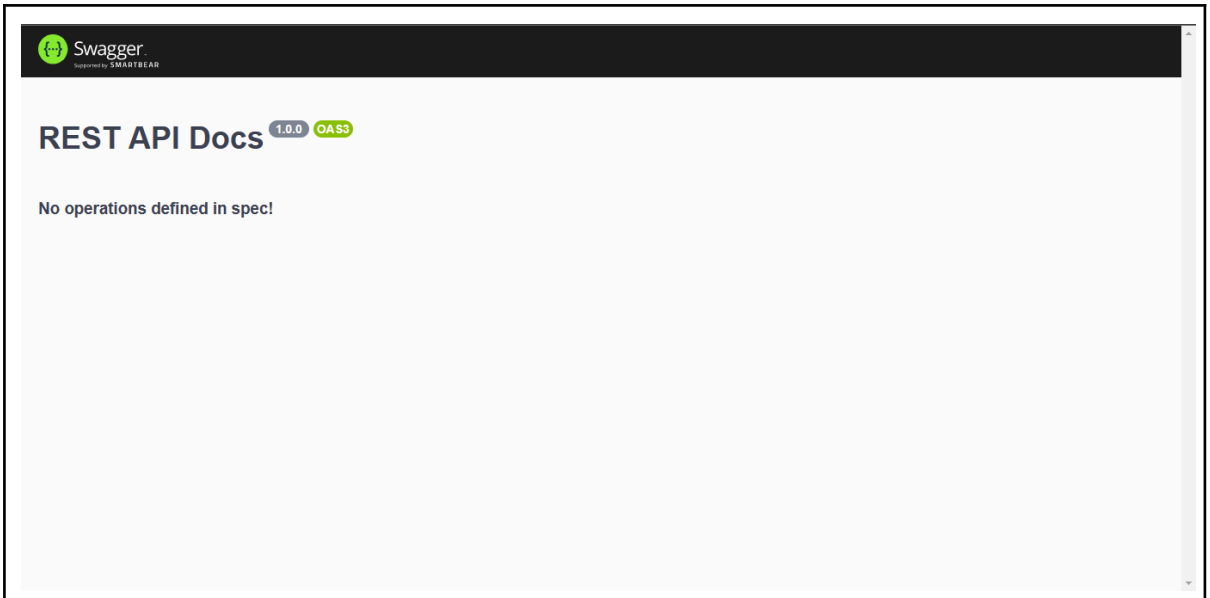
```
const version = require('../package-lock.json').version;

// const options = {
const options = swaggerJSDoc.Options = {
   definition: {
      openapi: '3.0.0',
      info: {
         title: 'REST API Docs',
         version,
         description: 'This is a REST API application made with Express. It retrieves data from a
MongoDB database.',
         license: {
            name: 'Licensed Under MIT',
            url: 'https://spdx.org/licenses/MIT.html',
         },
         contact: {
            name: 'Return to Online Music Store',
            // url: 'http://localhost:5000',
            url: 'https://oms-n7hw.onrender.com',
         },
      },
      servers: [
         {
            url: 'https://oms-n7hw.onrender.com',
            description: 'Production server'
         },
         {
            url: 'http://localhost:5000',
            description: 'Development server',
         },
      ],
      components: {
         securitySchemas: {
            bearerAuth: {
               type: 'http',
               scheme: 'bearer',
               bearerFormat: 'JWT',
            },
         },
      },
      security: [
         {
            bearerAuth: [],
         },
      ],
   },
   apis: ['./routes/*.js', './models/*.js'],
};

const swaggerSpec = swaggerJSDoc(options);
```

```
const swaggerDocs = (app, port) => {
   // swagger page
   app.use('/docs', swaggerUi.serve, swaggerUi.setup(swaggerSpec));

   // docs in JSON format
   app.get('docs.json', (req, res) => {
      res.setHeader('Content-Type', 'application/json')
      res.send(swaggerSpec);
   });

   console.log(`Documentation available at http://localhost:${port}/docs`);
};

module.exports = swaggerDocs;
```

server.js

```
…
// route files
const albums = require('./routes/albums');
const tracks = require('./routes/tracks');
const auth = require('./routes/auth');
const users = require('./routes/users');
const reviews = require('./routes/reviews');

// utils files
const swaggerDocs = require('./utils/swagger');
..
const PORT = process.env.PORT || 5000;

const server = app.listen(PORT, () => {
  console.log(`Server running in ${process.env.NODE_ENV} mode on port ${PORT}`);
  swaggerDocs(app, PORT);
});
```

The Swagger documentation is displayed on http://localhost:5000/docs.



models > Album.js

```
const mongoose = require('mongoose');
const slugify = require('slugify');

/**
 * @openapi
 * components:
 *   schemas:
 *     Album:
 *       type: object
 *       properties:
 *         album_name:
 *           type: string
 *           required: true
 *         cover_photo:
 *           type: string
 *         artist_slug:
 *           type: string
 *         genre:
 *           type: string
 *         year:
 *           type: number
 *         producer:
 *           type: string
 *         description:
 *           type: string
 *         album_url:
 *           type: string
 *       additionalProperties: false
 */
const AlbumSchema = new mongoose.Schema(
…
```

models > Track.js

```
const mongoose = require('mongoose');
const slugify = require('slugify');

/**
 * @openapi
 * components:
 *   schemas:
 *     Track:
 *       type: object
 *       properties:
 *         track_name:
 *           type: string
 *           required: true
 *         featuring:
 *           type: string
 *         duration:
 *           type: string
 *         track_file:
 *           type: string
 *         credit:
 *           type: string
 *         file_size:
 *           type: string
 *         album:
 *           type: string
 *       additionalProperties: false
 */
const TrackSchema = new mongoose.Schema({
…
```

models > Review.js

```
const mongoose = require('mongoose');

/**
 * @openapi
 * components:
 *   schemas:
 *     Review:
 *       type: object
 *       properties:
 *         title:
 *           type: string
 *         text:
 *           type: string
 *         rating:
 *           type: string
 *       additionalProperties: false
 */
```

```
const ReviewSchema = new mongoose.Schema({
…
```

models > User.js

```
const crypto = require('crypto');
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

/**
 * @openapi
 * components:
 *   schemas:
 *     User:
 *       type: object
 *       properties:
 *         name:
 *           type: string
 *           required: true
 *         email:
 *           type: string
 *           required: true
 *         role:
 *           type: string
 *           enum:
 *             - user
 *             - publisher
 *         password:
 *           type: string
 *           required: true
 *       additionalProperties: false
 *     Login:
 *       type: object
 *       properties:
 *         email:
 *           type: string
 *           required: true
 *         password:
 *           type: string
 *           required: true
 *       additionalProperties: false
 *     UserDetails:
 *       type: object
 *       properties:
 *         name:
 *           type: string
 *           required: true
 *         email:
 *           type: string
 *           required: true
```

```
 *     ForgotPassword:
 *       type: object
 *       properties:
 *         email:
 *           type: string
 *           required: true
 *       additionalProperties: false
 *     ResetPassword:
 *       type: object
 *       properties:
 *         password:
 *           type: string
 *           required: true
 *       additionalProperties: false
 */
const UserSchema = mongoose.Schema({
…
```

routes > albums.js

```
…
// re-route to other resource routers
router
  .use('/:albumId/tracks', trackRouter)
  .use('/:albumId/users', userRouter)
  .use('/:albumId/reviews', reviewRouter);

/**
 * @openapi
 * tags:
 *   name: Albums
 *   description: APIs to perform CRUD operations on albums
 * /api/v1/albums:
 *   get:
 *     tags:
 *       - Albums
 *     description: Retrieve a list of all the albums from the database. The list is paginated and one
album is displayed per page.
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *   post:
 *     tags:
 *       - Albums
 *     description: Add an album to the database.
 *     requestBody:
 *       content:
 *         application/json:
 *           schema:
```

```
 *           $ref: '#/components/schemas/Album'
 *     responses:
 *       201:
 *         description: Success
 *       401:
 *         description: User not authorized to access this route
*/
router
  .route('/')
…
router
  .route('/newalbum')
  .get(protect, authorize('publisher', 'admin'), getCreateAlbum);

/**
 * @openapi
 * /api/v1/albums/{id}:
 *   get:
 *     tags:
 *       - Albums
 *     description: Get Albums By ID
 *     parameters:
 *       - in: path
 *         name: id
 *         required: true
 *         description: Numeric id of the album to retrieve
 *         schema:
 *           type: string
 *           example: 6361ff4314b08a4853714b68
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *   put:
 *     tags:
 *       - Albums
 *     description: Update albums by id. User needs to login (under "Authorization") before
executing this endpoint.
 *     parameters:
 *       - in: path
 *         name: id
 *         required: true
 *         description: Numeric id of the album to update
 *         schema:
 *           type: string
 *           example: 6361ff4314b08a4853714b68
 *     requestBody:
 *       content:
 *         application/json:
 *           schema:
```

```
 *           $ref: '#/components/schemas/Album'
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *       401:
 *         description: User not authorized to access this route
 *   delete:
 *     tags:
 *       - Albums
 *     parameters:
 *       - in: path
 *         name: id
 *         required: true
 *         description: Numeric id of the album to delete
 *         schema:
 *           type: string
 *           example: 6361ff4314b08a4853714b68
 *     description: Delete albums by id. User needs to login (under "Authorization") before executing
this endpoint.
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *       401:
 *         description: User not authorized to access this route
*/
router
  .route('/:id')
…
/**
 * @openapi
 * /api/v1/albums/{id}/photo:
 *   put:
 *     tags:
 *       - Albums
 *     description: Upload a Cover Photo for the Albums By ID. User needs to login (under
"Authorization") before executing this endpoint.
 *     parameters:
 *       - in: path
 *         name: id
 *         required: true
 *         description: Numeric id of the album for which cover photo is to be uploaded
 *         schema:
 *           type: string
 *           example: 6361ff4314b08a4853714b68
 *     requestBody:
 *       required: true
 *       content:
```

```
 *       multipart/form-data:
 *         schema:
 *           type: object
 *           properties:
 *             photo:
 *               type: string
 *               format: binary
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *       401:
 *         description: User not authorized to access this route
 */
router
  .route('/:id/photo')
…
```

routes > tracks.js

```
…
// use protect & authorize middleware
const { protect, authorize } = require('../middleware/auth');

/**
 * @openapi
 * tags:
 *   name: Tracks
 *   descriptions: APIs to perform CRUD operations on tracks
 * /api/v1/tracks:
 *   get:
 *     tags:
 *       - Tracks
 *     descriptions: Retrieve a list of all the tracks from the database. The list is paginated and one
track is displayed per page.
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Response not found
 * /api/v1/albums/{album_id}/tracks:
 *   post:
 *     tags:
 *       - Tracks
 *     description: Add a track to a selected album in the database.
 *     parameters:
 *       - in: path
 *         name: album_id
 *         required: true
 *         description: Numeric id of the album to which the track is to be added.
```

```
 *         schema:
 *           type: string
 *           example: 6361ff4314b08a4853714b69
 *     requestBody:
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/Track'
 *     responses:
 *       201:
 *         description: Success
 *       401:
 *         description: User not authorized to access this route
 */
router
  .route('/')
…
/**
 * @openapi
 * /api/v1/tracks/{id}:
 *   get:
 *     tags:
 *       - Tracks
 *     description: Get Tracks By ID
 *     parameters:
 *       - in: path
 *         name: id
 *         required: true
 *         description: Numeric id of the track to retrieve
 *         schema:
 *           type: string
 *           example: 63674371637b3b4560a9cb77
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *   put:
 *     tags:
 *       - Tracks
 *     description: Update tracks by id. User needs to login (under "Authorization") before executing
this endpoint.
 *     parameters:
 *       - in: path
 *         name: id
 *         required: true
 *         description: Numeric id of the track to update
 *         schema:
 *           type: string
 *           example: 63674371637b3b4560a9cb77
 *     requestBody:
```

```
 *      content:
 *        application/json:
 *          schema:
 *            $ref: '#/components/schemas/Track'
 *    responses:
 *      200:
 *        description: Success
 *      404:
 *        description: Resource not found
 *      401:
 *        description: User not authorized to access this route
 *   delete:
 *    tags:
 *      - Tracks
 *    parameters:
 *      - in: path
 *        name: id
 *        required: true
 *        description: Numeric id of the track to delete
 *        schema:
 *          type: string
 *          example: 63674371637b3b4560a9cb77
 *    description: Delete tracks by id. User needs to login (under "Authorization") before executing
this endpoint.
 *    responses:
 *      200:
 *        description: Success
 *      404:
 *        description: Resource not found
 *      401:
 *        description: User not authorized to access this route
 */
router
  .route('/:id')
…
/**
 * @openapi
 * /api/v1/tracks/{id}/audio:
 *   put:
 *    tags:
 *      - Tracks
 *    description: Upload a music track to a selected album by id. User needs to login (under
"Authorization") before executing this endpoint.
 *    parameters:
 *      - in: path
 *        name: id
 *        required: true
 *        description: Numeric id of the album to which the track is to be uploaded
 *        schema:
 *          type: string
 *          example: 63674371637b3b4560a9cb77
```

```
 *     requestBody:
 *       required: true
 *       content:
 *         multipart/form-data:
 *           schema:
 *             type: object
 *             properties:
 *               audio:
 *                 type: string
 *                 format: binary
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *       401:
 *         description: User not authorized to access this route
 */
router
  .route('/:id/audio')
…
```

routes > reviews.js

```
…
// use protect & authorize
const { protect, authorize } = require('../middleware/auth');

/**
 * @openapi
 * tags:
 *   name: Reviews
 *   description: APIs to perform CRUD operations on reviews
 * /api/v1/reviews:
 *   get:
 *     tags:
 *       - Reviews
 *     description: Retrieve a list of all the reviews from the database. The list is paginated and one
album is displayed per page.
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 * /api/v1/albums/{album_id}/reviews:
 *   post:
 *     tags:
 *       - Reviews
 *     description: Add a review to a selected album in the database.
 *     parameters:
 *       - in: path
```

```
 *        name: album_id
 *        required: true
 *        description: Numeric id of the album to which the review is to be added.
 *        schema:
 *          type: string
 *          example: 6361ff4314b08a4853714b69
 *     requestBody:
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/Review'
 *     responses:
 *       201:
 *         description: Success
 *       401:
 *         description: User not authorized to access this route
 */
router
  .route('/')
…
/**
 * @openapi
 * /api/v1/reviews/{id}:
 *   get:
 *     tags:
 *       - Reviews
 *     description: Get reviews by id
 *     parameters:
 *       - in: path
 *         name: id
 *         required: true
 *         description: Numeric id of the review to retrieve
 *         schema:
 *           type: string
 *           example: 6391f2da06b0ab9e53864436
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *   put:
 *     tags:
 *       - Reviews
 *     description: Update reviews by id. User needs to login (under "Authorization") before
executing this endpoint.
 *     parameters:
 *       - in: path
 *         name: id
 *         required: true
 *         description: Numeric id of the review to update
 *         schema:
```

```
 *         type: string
 *         example: 6391f2da06b0ab9e53864436
 *     requestBody:
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/Review'
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *       401:
 *         description: User not authorized to access this route
 *   delete:
 *     tags:
 *       - Reviews
 *     parameters:
 *       - in: path
 *         name: id
 *         required: true
 *         description: Numeric id of the review to delete
 *         schema:
 *           type: string
 *           example: 6391f2da06b0ab9e53864436
 *     description: Delete reviews by id. User needs to login (under "Authorization") before
executing this endpoint.
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *       401:
 *         description: User not authorized to access this route
*/
router
  .route('/:id')
…
```

routes >auth.js

```
…
// use protect
const { protect } = require('../middleware/auth');

/**
 * @openapi
 * tags:
 *   name: Authorization
 *   description: APIs to perform authorization operations on users
 * /api/v1/auth/register:
```

```
 *   post:
 *     tags:
 *       - Authorization
 *     description: Register a new user. User needs to login (under "Authorization") before executing
this endpoint.
 *     requestBody:
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#components/schemas/User'
 *     responses:
 *       200:
 *         description: Success
 *       401:
 *         description: User not authorized to access this route
 */
router
  .route('/register')
…
/**
 * @openapi
 * /api/v1/auth/login:
 *   post:
 *     tags:
 *       - Authorization
 *     description: Login a new user. User needs to login (under "Authorization") before executing
this endpoint.
 *     requestBody:
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#components/schemas/Login'
 *     responses:
 *       200:
 *         description: Success
 */
router
  .route('/login')
…
/**
 * @openapi
 * /api/v1/auth/logout:
 *   get:
 *     tags:
 *       - Authorization
 *     description: Logout the currently logged in user.
 *     responses:
 *       200:
 *         description: Success
 */
router
```

```
 .route('/logout')
…
/**
 * @openapi
 * /api/v1/auth/me:
 *   get:
 *     tags:
 *       - Authorization
 *     description: Retrieve the details of the currently logged in user from the database.
 *     responses:
 *       200:
 *         description: Success
 *       401:
 *         User not authorized to access this route
 */
router
 .route('/me')
…
/**
 * @openapi
 * /api/v1/auth/forgotpassword:
 *   post:
 *     tags:
 *       - Authorization
 *     description: Send the email for resetting the password to the database.
 *     requestBody:
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/ForgotPassword'
 *     responses:
 *       201:
 *         description: Success
 */
router
 .route('/forgotpassword')
…
/**
 * @openapi
 * /api/v1/auth/resetpassword/{resettoken}:
 *   put:
 *     tags:
 *       - Authorization
 *     description: Send the email for resetting the password to the database.
 *     parameters:
 *       - in: path
 *         name: resettoken
 *         description: Password reset token
 *         schema:
 *           type: string
 *     requestBody:
```

```
 *      content:
 *        application/json:
 *          schema:
 *            $ref: '#/components/schemas/ResetPassword'
 *    responses:
 *      200:
 *        description: Success
 */
router
  .route('/resetpassword/:resettoken')
…
/**
 * @openapi
 * /api/v1/auth/updatedetails:
 *   put:
 *     tags:
 *       - Authorization
 *     description: Update the details of the currently logged in user.
 *     requestBody:
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/UserDetails'
 *           type: string
 *     responses:
 *       200:
 *         description: Success
 *       401:
 *         description: User not authorized to access this route
 */
router
  .route('/updatedetails')
 …
/**
 * @openapi
 * /api/v1/auth/updatepassword:
 *   put:
 *     tags:
 *       - Authorization
 *     description: Update the details of the currently logged in user.
 *     requestBody:
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/UserDetails'
 *     responses:
 *       200:
 *         description: Success
 *       401:
 *         description: User not authorized to access this route
 */
```

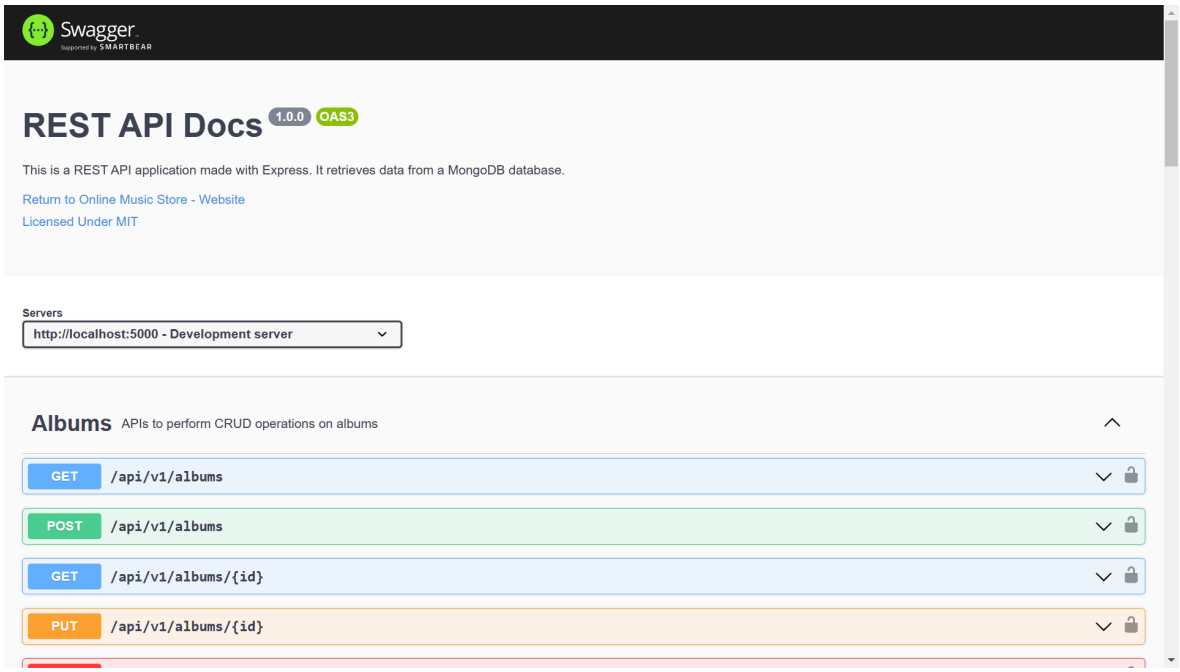
```
router
  .route('/updatepassword')
…
```

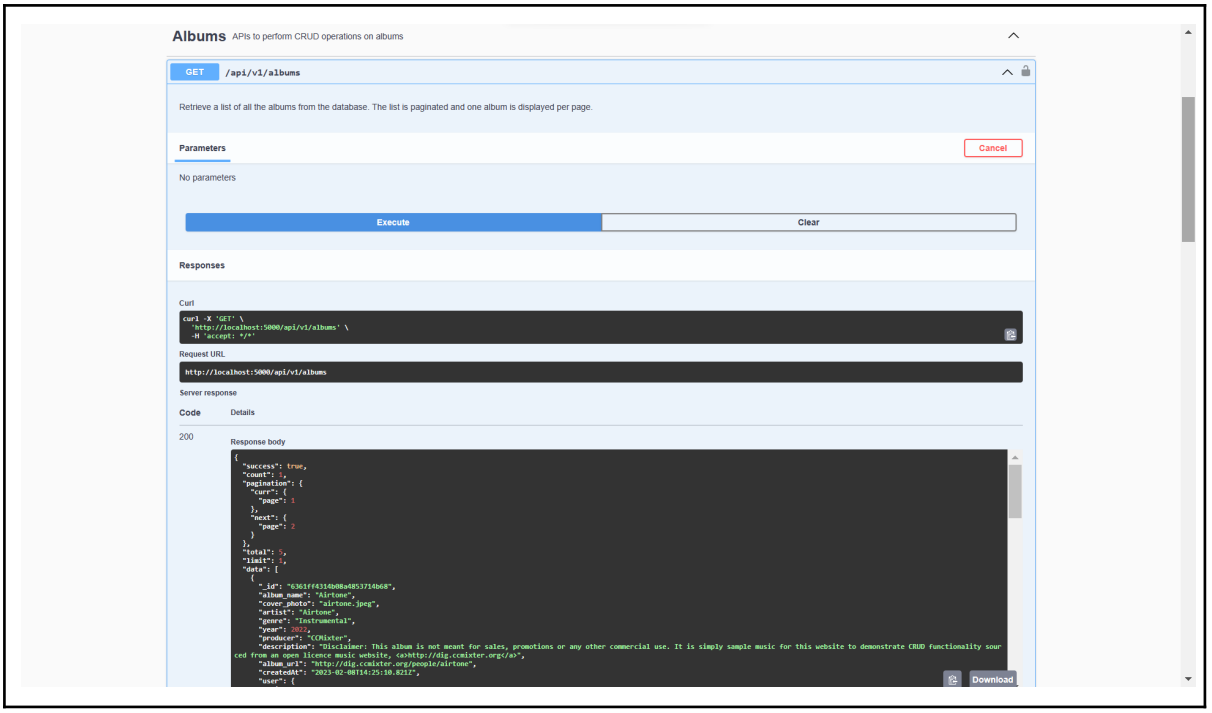
routes > users.js

```
…
// all routes below will be protected & authorized
router.use(protect);
router.use(authorize('admin'));

/**
 * @openapi
 * tags:
 *   name: Users
 *   description: APIs to perform CRUD operations on users
 * /api/v1/users:
 *   get:
 *     tags:
 *       - Users
 *     description: Retrieve a list of all the users from the database. The list is paginated and one
user is displayed per page.
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *   post:
 *     tags:
 *       - Users
 *     description: Add a user to the database. User needs to login (under "Authorization") before
executing this endpoint.
 *     requestBody:
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/User'
 *     responses:
 *       201:
 *         description: Success
 *       401:
 *         description: User not authorized to access this route
 */
router
  .route('/')
…
/**
 * @openapi
 * /api/v1/users/{id}:
 *   get:
 *     tags:
```

```
*      - Users
*    description: Retrieve a user by id from the database.
*    parameters:
*      - in: path
*        name: id
*        required: true
*        description: Numeric ID of the album to retrieve
*        schema:
*          type: string
*          example: 6361ff4314b08a4853714b68
*    responses:
*      200:
*        description: Success
*      404:
*        description: Resource not found
*   put:
*    tags:
*      - Users
*    description: Update users By id. User needs to login (under "Authorization") before executing
this endpoint.
*    parameters:
*      - in: path
*        name: id
*        required: true
*        description: Numeric ID of the user to update
*        schema:
*          type: string
*          example: 6361ff4314b08a4853714b68
*    requestBody:
*      content:
*        application/json:
*          schema:
*            $ref: '#/components/schemas/User'
*    responses:
*      200:
*        description: Success
*      404:
*        description: Resource not found
*      401:
*        description: User not authorized to access this route
*   delete:
*    tags:
*      - Users
*    parameters:
*      - in: path
*        name: id
*        required: true
*        description: Numeric id of the user to delete
*        schema:
*          type: string
*          example: 6361ff4314b08a4853714b68
```

```
 *     description: Delete users by id. User needs to login (under "Authorization") before executing this endpoint.
 *     responses:
 *       200:
 *         description: Success
 *       404:
 *         description: Resource not found
 *       401:
 *         description: User not authorized to access this route
 */
router
  .route('/:id')
…
```





Deployment

Website is deployed on render.com, by selecting the repository from the list, providing the root folder address and entering environment variables for the project in the input fields. The project is set to automatically deploy when changes are pushed to the repository.

X --- END OF PROJECT --- X