Broadcast Radio- Development Project

Abstract

Your task is to complete this React-based application.

The project is an application to manage the information of musical artists. The system should allow you to list artists, create a new artist, view/update an existing artist, and delete an artist.

You are welcome to refer to resources (tutorials, guides, etc) if you're not sure how to complete a task. We ask that you attribute – for example – StackOverflow code snippets where appropriate.

The Project

You will find a bootstrapped React application and a simple NodeJS-based backend/API server, using Sequelize for simple access to a database.

The code for the React components is inside src/app. The code for the API server is inside src/server.

For your convenience, the React development server will load the API server automatically, so you will not need to configure ports/proxies/etc. The development server will also 'hot reload' your changes, highlighting any errors.

Although your solution should build without errors, deploying the application is optional.

You are welcome to extend the project with additional features.

Data

The project should access data using the Sequelize library. This has been configured for you.

This project uses a SQLite database with a single data table: Artists. Please note that the database will reset when you restart the development server.

The structure of the Artists table is:

- `id` – The Artist ID
- `name` – The artist's name
- `description` – Some pre-amble/introduction to the artist
- `label` – The name of the artist's record label

You can see/modify the definition and configure extra test data in `src/server/database.js`

Development Plan

Task 1: Implement the save functionality

At the moment, the API server does not implement the save (POST) method, and the application throws an unhandled error if you try to save the form.

Your solution should implement this method, using the appropriate Sequelize models, as per the library's documentation.

Task 2: Creating New Artists

Modify the application so you can create (and save) a new artist into the database. You may re-use/refactor the Artist component, or come up with an alternative solution.

Task 3: Deleting Artists

Modify the application so that the Delete button functions as expected.

Task 3: Refactoring

Refactor the application to use form components from the `@fluentui/react` package (version 8).

Consider if there are any other Fluent UI components that could work for displaying data.

The rest of the project template is a blank canvas! You are certainly welcome to style the application as you see fit.

Task 4: Artist Images

Modify the application so that the user can upload an image to an artist. You should use the "multer" package to handle file uploads.

It is okay to store the image as binary data inside the database, making the appropriate modifications to the table structure, or in a separate "uploads" folder (or even using a cloud storage bucket) – the choice is entirely up to you.

Solution Delivery

You should deliver your solution:

- As a .zip file *(with node_modules removed first!)*

Solutions should be submitted by **10 am** on **Friday 26th January 2024**.