Application Architecture

Overview

The Audiobook Review and Rating System is a full-stack web application with a clear separation between the frontend and backend. The frontend is built using React, and the backend is built using Node.js and Express. MongoDB is used for data storage.

Frontend Architecture

Components

- **App.js**: The main entry point of the React application. It includes routing and the overall layout.
- **Components**:
  - `BackToHomeButton.js`: A reusable button component that navigates back to the home page.
- **Pages**:
  - `HomePage.js`: Displays a list of audiobooks and provides navigation to other pages.
  - `AudiobookPage.js`: Displays detailed information about a selected audiobook, including reviews and ratings.
  - `NewBookPage.js`: A form to add a new audiobook.
  - `EditBookPage.js`: A form to edit existing audiobook details.
- **Styling**:
  - CSS files like `App.css`, `HomePage.css`, etc., for styling individual components and pages.
- **Assets**:
  - `defaultCoverImage.jpg`: A default image for audiobooks without a cover image.

Routing

- **React Router**: Used for navigating between different pages like home, audiobook details, new book form, and edit book form.

Backend Architecture

Server.js

- The main entry point of the backend application. It initializes the Express app and connects to the MongoDB database.

Config

- **db.js**: Contains the configuration for connecting to the MongoDB database.

Controllers

- **audiobookController.js**: Contains logic for handling requests related to audiobooks, such as fetching, adding, and updating audiobooks.
- **reviewController.js**: Contains logic for handling requests related to reviews, such as adding and fetching reviews.

Middleware

- **errorHandler.js**: Custom error-handling middleware to handle and respond to errors gracefully.

Models

- **Audiobook.js**: Mongoose schema and model for audiobooks.
- **Review.js**: Mongoose schema and model for reviews.

Routes

- **audiobooks.js**: Defines API endpoints for audiobook-related operations.

Utility Scripts

- **audiobookData.js**: Utility script for seeding initial audiobook data.
- **migrateReviews.js**: Utility script for migrating review data.
- **seed.js**: Script for seeding initial data into the database.

Application Flow

1. **User Interaction**: Users interact with the frontend by navigating through the application, searching for audiobooks, and submitting reviews.
2. **API Requests**: The frontend sends API requests to the backend to fetch data or submit user inputs.
3. **Backend Processing**: The backend processes these requests, interacts with the database, and sends appropriate responses.
4. **Database Operations**: The database stores and retrieves data as per the requests from the backend.
5. **Response Handling**: The frontend receives the data from the backend and updates the UI accordingly.

API Usage Overview The API provides endpoints for managing audiobooks and reviews. Below are the available endpoints along with their methods, paths, and descriptions.

Audiobook Endpoints

- **Add a new audiobook:**
  - Method: POST
  - Endpoint: /
  - Description: Adds a new audiobook to the collection.
  - Request Body:

    ```
    {
      "title": "string",
      "author": "string",
      "description": "string",
      "genre": "string",
      "coverImage": "string (URL)"
    }
    ```

  - Response:

    ```
    {
      "id": "string",
      "title": "string",
    ```

```
  "author": "string",
  "description": "string",
  "genre": "string",
  "coverImage": "string",
  "createdAt": "date"
}
```

- **Get all audiobooks:**
  - Method: GET
  - Endpoint: /
  - Description: Retrieves a list of all audiobooks.
  - Response:

    ```
    [
      {
        "id": "string",
        "title": "string",
        "author": "string",
        "description": "string",
        "genre": "string",
        "coverImage": "string",
        "createdAt": "date"
      }
    ]
    ```

- **Get an audiobook by ID:**
  - Method: GET
  - Endpoint: /:id
  - Description: Retrieves details of a specific audiobook by its ID, including its reviews.
  - Response:

    ```
    {
      "id": "string",
      "title": "string",
      "author": "string",
      "description": "string",
      "genre": "string",
      "coverImage": "string",
      "reviews": [
        {
          "id": "string",
          "description": "string",
          "rating": "number",
          "createdAt": "date"
        }
      ],
      "createdAt": "date"
    }
    ```

- **Delete an audiobook by ID:**
    - Method: DELETE
    - Endpoint: /:id
    - Description: Deletes a specific audiobook by its ID.
- **Update an audiobook by ID:**
    - Method: PUT
    - Endpoint: /:id
    - Description: Updates details of a specific audiobook by its ID.
    - Request Body:

    ```
    {
      "title": "string",
      "author": "string",
      "description": "string",
      "genre": "string",
      "coverImage": "string (URL)"
    }
    ```

    - Response:

    ```
    {
      "id": "string",
      "title": "string",
      "author": "string",
      "description": "string",
      "genre": "string",
      "coverImage": "string",
      "createdAt": "date"
    }
    ```

- *Review Endpoints**
- **Add a review to an audiobook:**
    - Method: POST
    - Endpoint: /:id/reviews
    - Description: Adds a new review to a specific audiobook.
    - Request Body:

    ```
    {
      "description": "string",
      "rating": "number"
    }
    ```

    - Response:

    ```
    {
      "id": "string",
      "description": "string",
      "rating": "number",
      "createdAt": "date",
    ```

```
  "audiobook": "string (audiobookId)"
}
```

- **Get all reviews for an audiobook:**
  - Method: GET
  - Endpoint: /:id/reviews
  - Description: Retrieves all reviews for a specific audiobook.
  - Response:

```
[
  {
    "id": "string",
    "description": "string",
    "rating": "number",
    "createdAt": "date",
    "audiobook": "string (audiobookId)"
  }
]
```

- **Delete a review by ID:**
  - Method: DELETE
  - Endpoint: /:audiobookId/reviews/:reviewId
  - Description: Deletes a specific review by its ID.

Schemas

- **Audiobook Schema:**
  - title: The title of the audiobook (string, required).
  - author: The author of the audiobook (string, required).
  - coverImage: URL to the cover image of the audiobook (string, default value).
  - description: A brief description of the audiobook (string, required).
  - genre: The genre of the audiobook (string, required).
  - reviews: Array of reviews related to the audiobook, referencing the Review model.
  - createdAt: The date when the audiobook was added (date, default to current date).
- **Review Schema:**
  - description: The content of the review (string, required).
  - rating: The rating given in the review, between 1 and 5 (number, required).
  - createdAt: The date when the review was created (date, default to current date).
  - audiobook: Reference to the audiobook the review is associated with (ObjectId, required).

Deployment:

1. **Clone the repository:**

```bash git clone [https://github.com/sriramyogi99/Audiobook-Review-and-Rating-System.git](https://github.com/sriramyogi99/Audiobook-Review-and-Rating-System.git)

2. **Navigate to project directory:** ```bash cd your-repo

3. **Install dependencies:** ```bash cd client npm install cd ../server npm install

4. **Set up environment variables:** Update the .env file under the server folder of your project and add the following:

   ```
   MONGO_URI=your_mongodb_connection_string
   ```

   Replace `your_mongodb_connection_string` with your actual MongoDB connection URI.

5. **Running the application:**

   - Open two separate command prompts or terminal windows.
   - In the first terminal, navigate to the client directory and run:

     ```bash
     cd client
     npm start
     ```

   - In the second terminal, navigate to the server directory:

     ```bash
     cd server
     npm run start-all
     ```

Getting Started:

- Once both servers are running, you can access the application in your browser at http://localhost:3000 (or the specified port in the `PORT` variable inside `.env` file.).
     ```