Project: Music Playlist Management System

Objective: Develop a Music Playlist Management System where users can create, manage, and share their music playlists. The system should integrate with the Spotify API to fetch music details and provide an intuitive and responsive user interface.

Requirements:

1. **Frontend:**
    - **React.js with TypeScript**
    - **Material UI for components**
    - **HTML/CSS for custom styling**
    - **Responsive design**
2. **Backend:**
    - **Node.js with Express.js**
    - **MongoDB for database operations**
    - **JWT for authentication**
3. **Features:**

    User Authentication:

    - Users should be able to register and log in.
    - Passwords should be hashed using bcrypt.
    - Use JWT for session management.
4. **CRUD Operations:**
    - **Create:** Users can create new playlists.
    - **Read:** Users can view their playlists.
    - **Update:** Users can update playlist details (name, description).
    - **Delete:** Users can delete playlists.
5. **Spotify API Integration:**
    - Allow users to search for songs using the Spotify API.
    - Display song details (title, artist, album) retrieved from Spotify.
    - Users can add songs from the search results to their playlists.
6. **Responsive UI:**
    - Ensure the application is fully responsive and works seamlessly across different devices and screen sizes.
7. **Database Operations:**
    - Use MongoDB to store user data, playlists, and song details.
    - Implement necessary CRUD operations for managing data.

Assignment Details:

1. User Authentication:

- Create a registration page where users can sign up.
- Create a login page where users can log in.
- Protect routes using JWT to ensure only authenticated users can access certain pages.

2. Playlist Management:

- Create a dashboard where users can view all their playlists.
- Implement forms to add new playlists and update existing ones.
- Enable users to delete playlists.

3. Spotify API Integration:

- Create a search bar that allows users to search for songs.
- Display search results with song details.
- Allow users to add songs from the search results to their playlists.

4. Responsive Design:

- Use Material UI and custom CSS to ensure the application is visually appealing and responsive.
- You can use TailwindCSS as well

5. Database Schema:

- Design a schema for users, playlists, and songs in MongoDB.
- Implement necessary database operations to handle CRUD actions.

Submission Guidelines:

- **Source Code:** Provide a link to a GitHub repository with the complete source code.
- **README:** Include a README file with instructions on how to set up and run the project locally.
- **Demo:** Deploy the application to a cloud service (e.g., Heroku, Vercel) and provide a live demo link.
- **Documentation:** Include inline comments and documentation for major functions and components.

Evaluation Criteria:

- **Code Quality:** Clean, well-structured, and commented code.
- **Functionality:** Complete and correctly implemented features.
- **UI/UX:** Intuitive and responsive user interface.
- **Integration:** Proper integration with Spotify API.
- **Authentication:** Secure and functional user authentication.
- **Database Management (Optional):** Efficient and correct database operations. (you can use Redux state/Local-storage if you are not aware of any database)