**PROJECT DOCUMENTATION**

**RhythmicTunes – Your Melodic Companion**

**1. Introduction**

* **Project Title: RhythmicTunes – Your Melodic Companion**
* **Team ID:NM2025TMID30370**
* **Team Leader**: **HAJISHTHA FENAZIR F - haji.fena2007@gmail.com**
* **Team Members:** 
  + **DEVADHARSHINI C -** [**devadharshini21ck@gmail.com**](mailto:devadharshini21ck@gmail.com)
  + **NARMATHA C - narmathaviji2007@gmail.com**
  + **RANJANA M -ranjanatwin12@gmail.com**
  + **MAGALYA S - magisenthi@gmail.com**

**2. Project Overview**

* **Purpose:** 
  + RhythmicTunes is a browser-based music player built using React.js and Vite. It allows users to explore and play songs through an interactive interface, with features like playlist management, playback controls, and responsive design. Styled with Tailwind CSS and Bootstrap, the platform offers a clean and modern user experience while leveraging React Router for navigation and Axios for dynamic data handling. Future enhancements include TypeScript integration, API connectivity, and advanced audio features to make RhythmicTunes a complete melodic companion.
* **Goals:**
  + Provide an intuitive and modern music player UI.
  + Allow users to browse and manage playlists easily.
  + Enable smooth playback with play, pause, next, and previous controls.
  + Ensure a fully responsive experience across devices.
  + Fetch songs dynamically from APIs or local JSON data.
  + Allow seamless navigation between pages using React Router.
  + Enhance code quality and scalability with TypeScript integration.
* **Key Features:**
  + Search and browse songs or playlists.
  + View song details with title, artist, and album cover.
  + Play, pause, skip, and control volume easily.
  + Manage and switch between playlists seamlessly.
  + Fetch songs dynamically from local JSON or APIs.
  + Responsive design for smooth experience across all devices.

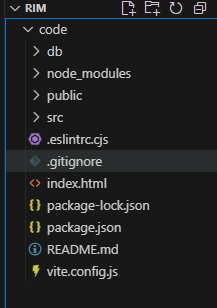
**3.Architecture**

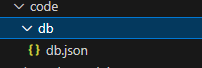
* **Component Structure**
  + App.jsx — Root component, sets up routes
  + Unavbar.jsx — Provides navigation + search
  + Uhome.jsx — Landing Hero/Home page
  + Search.jsx — Song search
  + Sidebar.jsx — Sidebar navigation
  + Songs.jsx — All songs / categories
  + Playlist.jsx — Playlist management
  + Favorites.jsx — Favorite songs
  + Wishlist.jsx — Wishlist songs
  + Uitem.jsx — Song detail view (playback page)
  + Footer.jsx — App footer / player controls
* **State Management**
  + **Local State**: Managed with useState + useEffect for songs, favorites, playlists, wishlist, UI state.
  + **API Integration**: Using Axios with db.json (via json-server) to fetch songs and metadata.
  + **Persistent State**: Favorites & Playlists stored in localStorage.
* **Routing**
  + **Library:**
    - react-router-dom
  + Routes:
    - / → **Uhome.jsx** (Home / Landing Page)
    - /songs → **Songs.jsx** (All songs list)
    - /playlist → **Playlist.jsx** (User-created playlists)
    - /favorites → **Favorites.jsx** (Liked songs)
    - /wishlist → **Wishlist.jsx** (Saved for later)
    - /song/:id → **Uitem.jsx** (Song detail page with playback)
    - /search → **Search.jsx** (Search results)

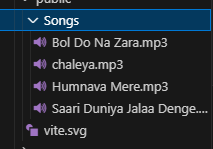
**4. Setup Instructions**

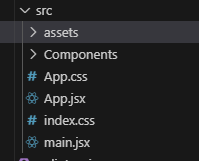
* **Prerequisites**
  + **Node.js&npm**
    - · Node.js is required to run React applications.
    - · npm (Node Package Manager) is used to install dependencies.
    - · [Download Node.js](https://nodejs.org/en/download)
  + **React.js**
    - React is the main JavaScript library used to build this project.
    - If you don’t have an existing React app, create one using:
    - npx create-react-app my-app
    - cd my-app
    - npm install
    - npm start dev
    - In SB Fitzz, the React app is already created, so you just need to install dependencies (npm install).
  + **Git**
    - Used for cloning and version control.
    - [Download Git](https://git-scm.com/downloads)
  + **Code Editor**
    - Recommended: **Visual Studio Code (VS Code)**
    - [Download VS Code](https://code.visualstudio.com)
  + **Basic Knowledge**
    - HTML, CSS, JavaScript,typescript
    - · React concepts (components, props, hooks, state, routing)
* **Installation**
  + **Get the code:**
    - Download the code from the drive link given below:
      * [https://drive.google.com/drive/folders/1fSBjgZ\_lCEDXYrKug1GuNYLAO1FAOLzX?usp=sharing](https://drive.google.com/drive/folders/1BkYWfW_K3ek_UgtXNTAsDqlhdCuqz6nT?usp=drive_link)
  + **Install Dependencies:**
    - Navigate into the cloned repository directory and install libraries:
      * cd fitness-app-react
      * npm install
    - **Start the Development Server**:
      * To start the development server, execute the following command:
        + npm start dev
    - **Access the App:**
      * Open your web browser and navigate to: <http://localhost:5173/playlist>
      * You should see the application's homepage, indicating that the installation and setup were successful.
* **Environment Variables**
  + Create a .env file with:
    - REACT\_APP\_API\_URL=<https://exercisedb.p.rapidapi.com/exercises/equipmentList>
    - REACT\_APP\_YOUTUBE\_API\_KEY=<33cf3a7616msh4c3b1e3204f24e2p1294b3jsne16a7323d732>

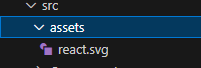
**5.Folder Structure**

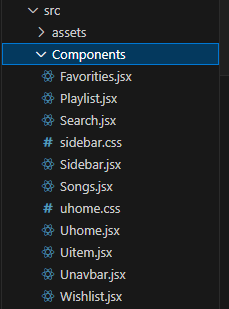












**6. Running the Application**

* **Start development server:**
  + npm start dev
* **Build for production:**
  + npm run build
* **Run tests:**
  + npm test

**7. Component Documentation**

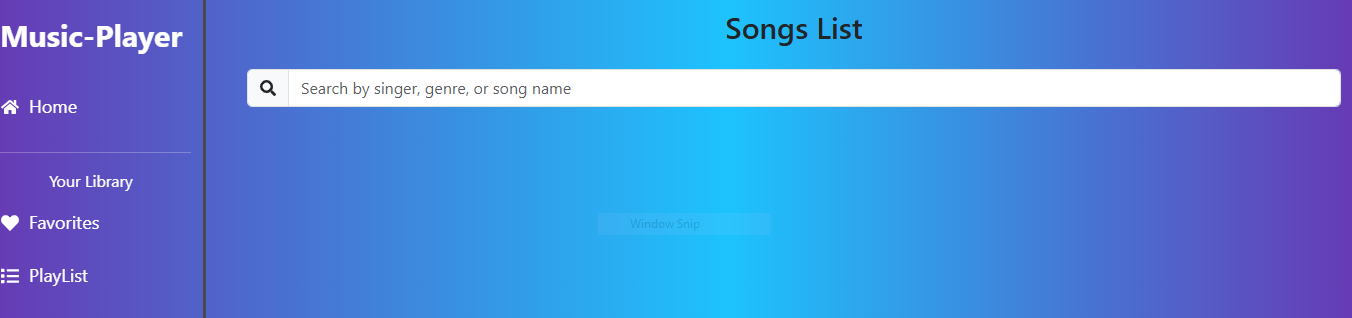
* **Key Components** 
  + **Unavbar.jsx — Top navigation with app links + search**
  + **Sidebar.jsx — Category navigation (Songs, Playlist, Favorites)**
  + **Search.jsx — Search songs by title/artist**
  + **Uitem.jsx — Song detail page with audio player**
  + **Footer.jsx — Footer with copyright & controls**
* **Pages** 
  + **Uhome.jsx — Landing page with featured songs**
  + **Songs.jsx — Displays all songs**
  + **Playlist.jsx — Displays user playlists**
  + **Favorites.jsx — Displays liked songs**
  + **Wishlist.jsx — Displays wishlist songs**
  + **Song/:id → Uitem.jsx (Detail view)**

**8. State Management**

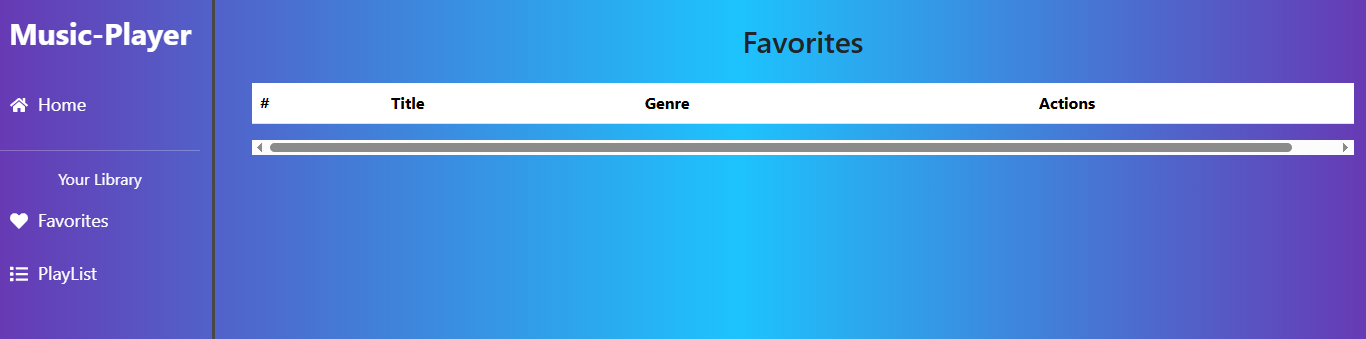
* **Local State:**
  + **o Search queries stored in HomeSearch.**
  + **o API data fetched and stored per-page.**
* **Global State:**
  + **Not implemented — app uses component-level state.**

**9. User Interface**

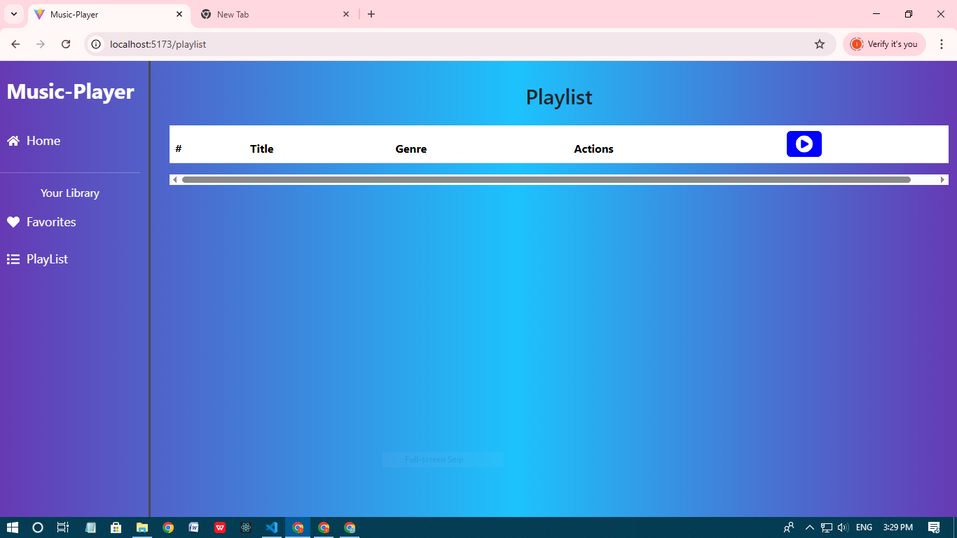
* **Pages include:**
  + **Home**



* + Favorites



* + Playlist



**10. Styling**

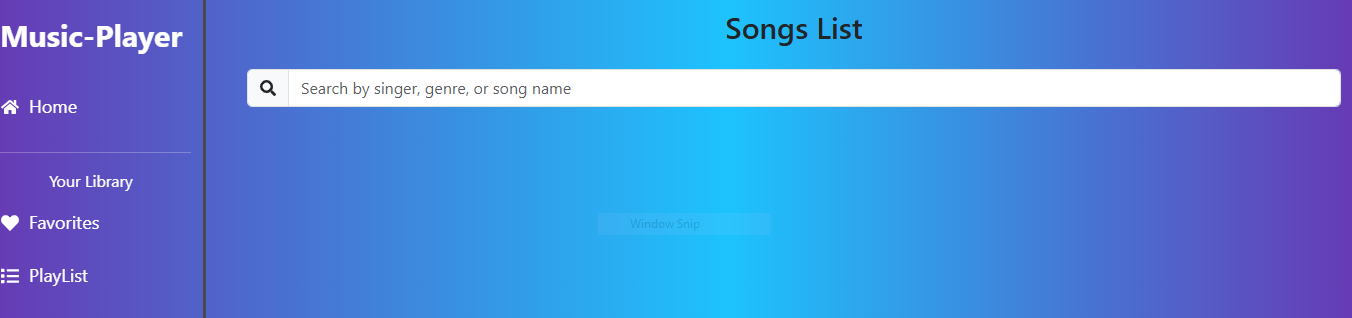
* **Frameworks Used:** Tailwind CSS / Bootstrap.
* **Custom CSS:** Stored in src/styles/.
* Each page/component has a dedicated CSS file for modularity.

**11. Testing**

* **Libraries Used:** Jest, React Testing Library.
* **Unit Tests:** Written in App.test.js.
* **Setup:** Configured with setupTests.js.

**12.Screenshots / Demo**

* **Demo**
  + <https://drive.google.com/file/d/1MZQliD6ItuuuPx5dqObR8wvUOz4iXZCx/view?usp=drive_link>
* Screenshot:



**13. Known Issues**

* API rate-limit may cause some exercises not to load.
* YouTube API sometimes fails to fetch related videos.

**14. Future Enhancements**

* Add login and user profile.
* Save favorite songs permanently.
* Create and share custom playlists.
* Add lyrics and detailed song information.
* Implement offline caching with service workers.
* Enhance animations and transitions.
* Integrate external music APIs for recommendations.
* Improve mobile responsiveness and add dark mode.