**Frontend Development with React.js**

**Project Documentation**

1. **Introduction**
   * **Project Title**: RhythmicTunes: Your Melodic Companion
   * **Team Members**: **NISHANTHINI S(Team Lead), ANANTHI R(Team Member), NIVETHITHA M(Team Member)**
2. **Project Overview**
   * **Purpose**:

**Rhythmic Tunes** is a music streaming and playlist creation application. The goal of the project is to provide users with a seamless platform to discover, play, and organize their favorite songs into custom playlists.

**Features**

* Music search by song, singer.
* Playlist management (create, edit, delete).
* Favourites and Wishlist functionality.
* Interactive music player (play, pause).
* Responsive UI for desktop and mobile.

1. **Architecture**
   * **Component Structure**:
     + **App.jsx** – Root component with routing.
     + **Sidebar** – Navigation (Home, Search, Playlists, Favourites, Wishlist).
     + **Search** – Search bar and results view.
     + **Playlist** – Displays playlist details and songs.
     + **Songs** – Generic song list renderer.
     + **Favourites** – User’s favourite songs.
     + **Wishlist** – Songs saved for later.
     + **Player** – Music player controls.
   * **State Management**:
     + **Global State** → Managed using **React Context API**. Handles authentication state, current playing track, and theme.
     + **Local State** → Managed with useState inside individual components (form values, modals, search input).
   * **Routing**:

 **React Router v6** is used.

 Routes:

/ → Home

/search → Search

/playlist/:id → Playlist

/favourites → Favourites

**Backend Integration**

* **Node.js + Express** backend.
* **json-server** provides a mock API with db.json.

1. **Setup Instructions**
   * **Prerequisites**:
     + **Node.js** (v16+ recommended)
     + **Visual Studio Code**
     + **Npm**
     + Json-server(for mock backend API)
   * **Installation**:

npm install

**Environment Variables**

Create .env in the root directory:

REACT\_APP\_API\_URL=http://localhost:3000

**Start JSON Server**

json-server --watch db.json --port 3000

1. **Folder Structure**

rhythmic-tunes/

│── db/ # JSON database (db.json)

│── node\_modules/ # Dependencies

│── public/ # Static assets

│── src/ # Source code

│ ├── assets/ # Images, icons

│ ├── components/ # Reusable UI (Sidebar, Songs, Player)

│ │── index.css # css file

│── .env # Environment variables

1. **Running the Application**
   * To start Frontend : npm run dev

To start Backend: json-server --watch db.json --port 3000

1. **Component Documentation**

### Key Components

* **Sidebar** → Navigation menu.
* **Search** → Search functionality.
* **Playlist** → Displays and manages playlist songs.
* **Songs** → Generic list view for songs.
* **Favourites** → User’s liked songs.

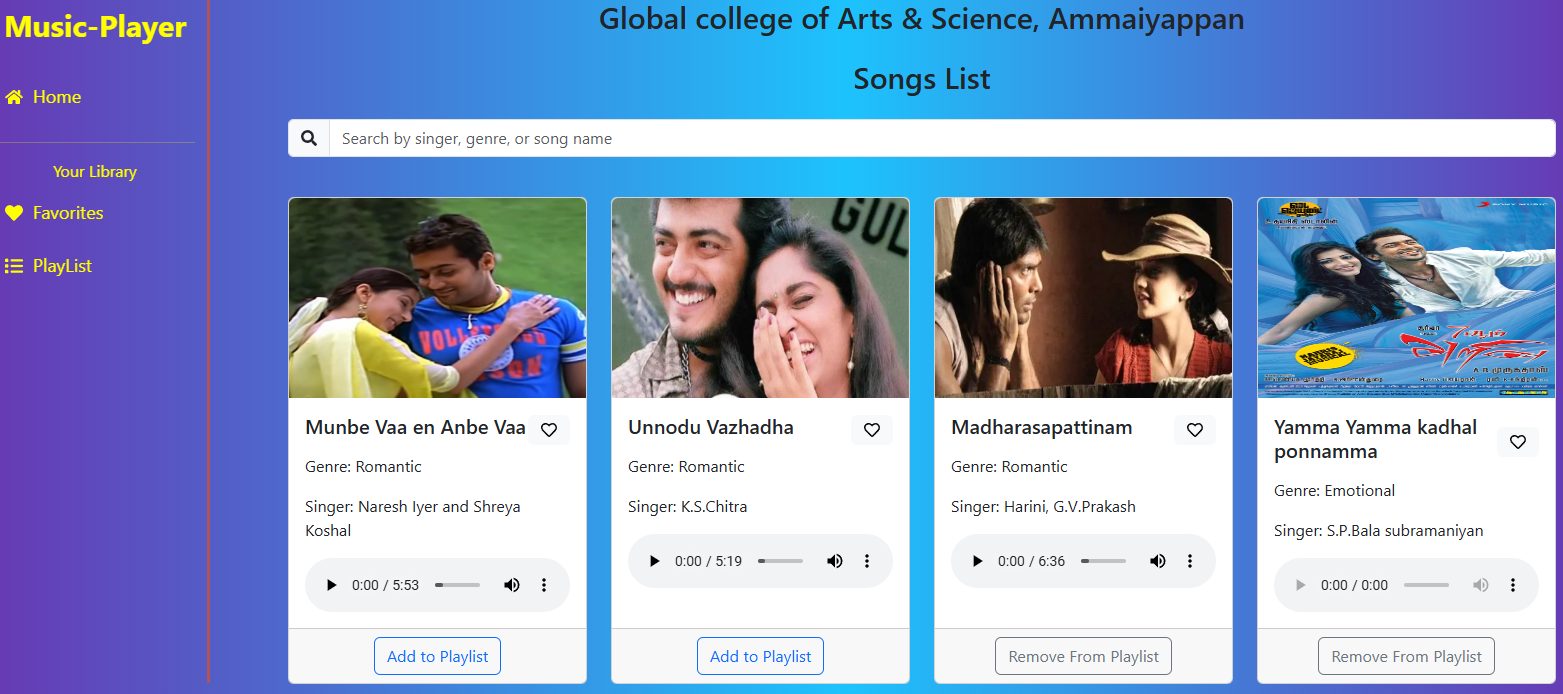
**Reusable Components**:

* + - **Player Controls** → Play, pause, skip, shuffle, repeat.
    - **SongCard** → Displays song details.
    - **Button** & **Input** → Reusable UI elements.

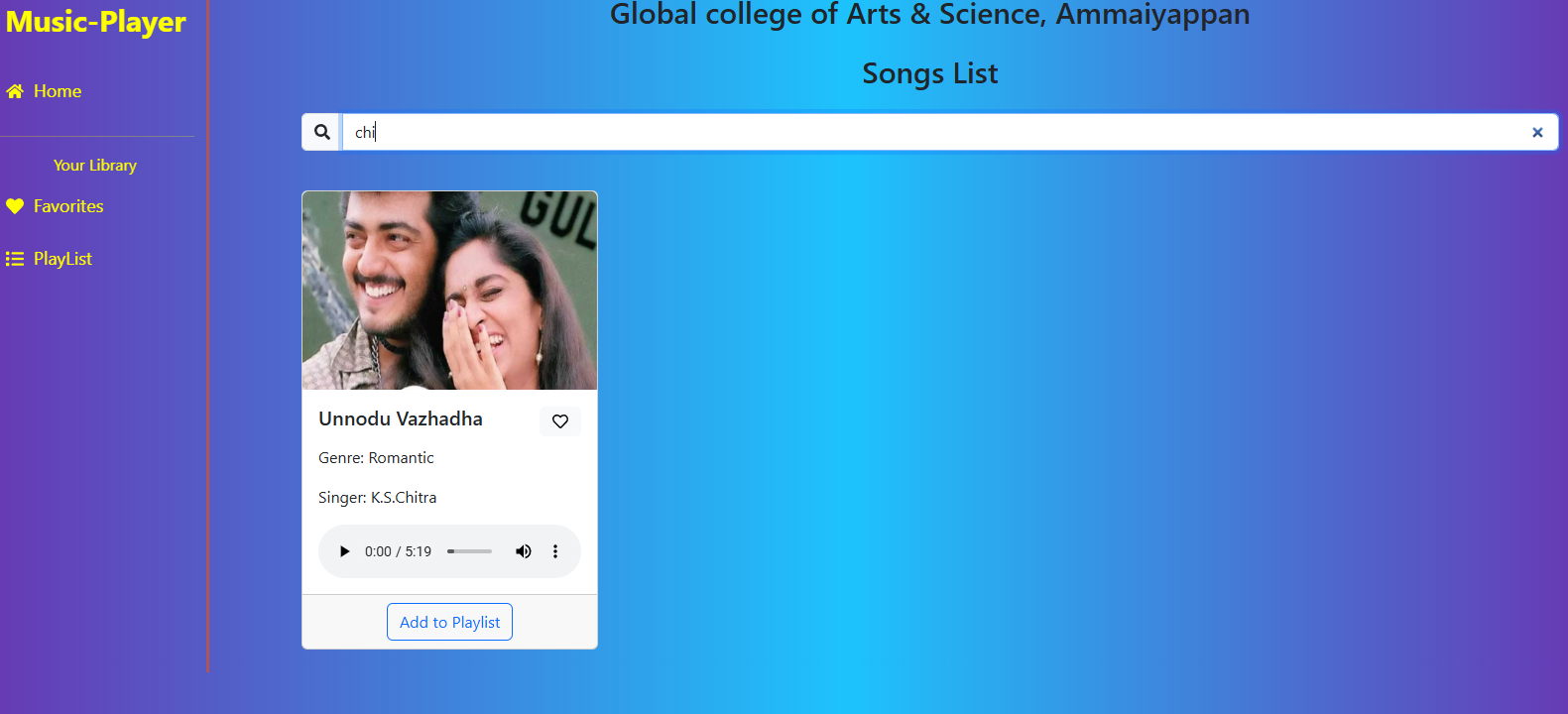
1. **State Management**
   * **Global State**: Auth, current playing song, theme handled by Context API.**Local State**: Managed within individual components for UI events.
2. **User Interface**

Screenshots from the demo:

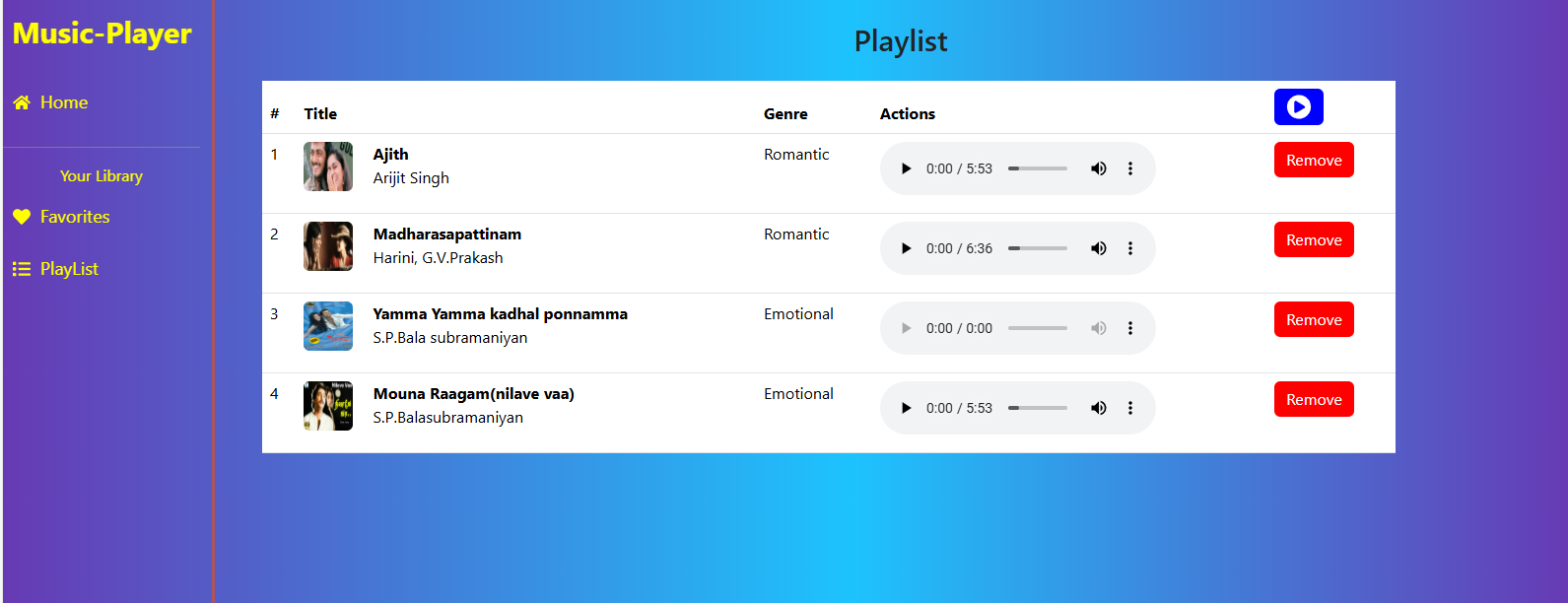
* frame\_0.jpg → Home page.



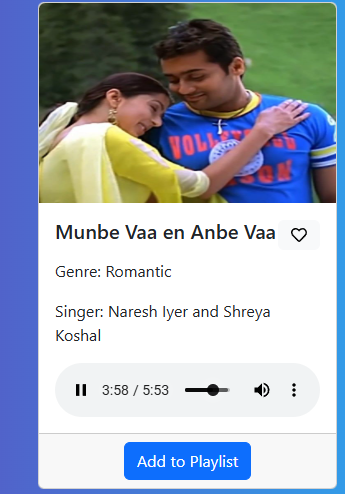
* frame\_150.jpg → Search page.



* frame\_300.jpg → Playlist view.



* frame\_450.jpg → Player controls.



1. **Styling**

 Uses **Custom CSS** (no frameworks).

 Component-specific CSS files.

 Global styles in App.css and index.css.

 Responsive design with Flexbox and Grid.

1. **Testing**

**Strategy**

* **Unit tests** for components.
* **Integration tests** for flows like playlist creation.
* **E2E tests** with Cypress (future).

**Tools**

* Jest
* React Testing Library

1. **Screenshots or Demo**

Demo video: rhythmic tunes.mp4

Extracted screenshots: frame\_0.jpg, frame\_150.jpg, frame\_300.jpg, frame\_450.jpg.

1. **Known Issues**

 Playback limited to basic controls.

 Search performance slows with larger dataset.

 Mobile responsiveness not fully optimized.

 Data resets when JSON server restarts.

 Authentication not fully secure (no JWT).

1. **Future Enhancements**

 JWT authentication & social logins.

 Collaborative playlists.

 Offline mode with local storage.

 Advanced player (equalizer, lyrics, queue).

 Push notifications for updates.

 Smooth animations & micro-interactions.

 Scalable backend (Node.js + DB like MongoDB).