**Frontend Development with React.js**

Project Documentation format

1. **Introductions**:

Project Title: [ Rhythmic tunes ]

Team Members: List team members and their roles.

Name : RAJASRI V (Team leader)

Roles : Coding

Name : PUPPY A (Team member)

Roles : Documentation

Name : RADHIKA S (Team member)

Roles : Demo video

Name : PUVI ARASI R (Team member)

Roles : Editing and voice over

Name : PRIYA DHARSHINI V (Team member)

Roles : documentation and coding

**2.Project Over view**:

**Purpose**: •Provide a seamless listening experience with a clean and responsive interface

•Help users explore new songs, genres, and artists with personalized recommendations.

•Enable music lovers to create and manage playlists tailored to their preferences.

•Offer a lightweight alternative to existing music apps, focusing on simplicity and accessibility.

•Build a platform that is scalable for future features like social sharing, offline access, and music videos.

**Key Features**: 🎵 Music Streaming – Play songs directly from the website.

🔍 Search & Filter – Quickly find songs, albums, or artists.

📂 Playlists – Create, edit, and manage personal playlists.

❤️ Favorites – Save liked songs for quick access.

📱 Responsive Design – Works smoothly on desktop, tablet, and mobile

**3. Architecture :**

Component Structure

* App.js – Root component handling routing and global layout.
* Navbar – Provides navigation across the application.
* MusicPlayer – Controls music playback (play, pause, skip, seek).
* Playlist – Displays user playlists with options to add/remove songs.
* SongCard – Reusable component to display song details (title, artist, album art).
* SearchBar – Handles search queries for songs and artists.
* UserAuth – Login and Signup forms.
* Profile – User details and preferences.

**4. Setup Instructions :**

Prerequisites

Node.js (>=16.x)

npm or yarn package manager

Git

Installation

1. Clone repository:

git clone https://github.com/your-repo/blue-music.git

cd blue-music

2. Install dependencies:

npm install

3. Configure environment variables (e.g., API keys in .env).

**5.Project Structure :**

Rhythmic tunes

│── public/ # Static assets

│── src/

│ ├── components/ # Reusable components

│ ├── pages/ # Page-level components

│ ├── hooks/ # Custom hooks

│ ├── utils/ # Helper functions

│ ├── context/ # Global state management

│ ├── App.js # Root component

│ └── index.js # Entry point

│── package.json

Utilities:

Custom Hooks: useAuth, useMusicPlayer.

Helper Functions: API request handler, format time for music duration.

**6.Running the Application:**

To start the frontend server locally:

npm start

Runs on http://localhost:3000 by default.

**7.Component Documentation :**

Key Components

* MusicPlayer – Handles song playback and controls.

Props: song, isPlaying, onPlayPause, onSkip.

* SongCard – Displays song information.

Props: title, artist, albumArt, onClick.

* Playlist – Displays playlist details.

Props: songs, onRemoveSong.

Reusable Components

* Button – Custom styled button.
* Modal – Used for dialogs (e.g., adding songs).
* SearchBar – Input for searching songs.

**8.State management** :

1. Global State

Authentication State (logged-in user).

Current Song State (currently playing song).

Playlist State (user playlists).

1. Local State

Controlled inputs (search, login forms).

UI toggles (modal open/close).

**9. User Interfaces :**

(Insert screenshots or GIFs here)

* Home Page with featured playlists.
* Music Player controls.
* Search results page.
* Playlist management UI.
* Profile page.

**10. Styling :**

* CSS Framework: Tailwind CSS for utility-first styling.
* Additional Libraries: Styled-components for component-level customization.
* Theming: Dark and light theme support with theme context.

**11. Testing :**

**Testing Strategy**

* **Unit Testing: Jest + React Testing Library for components.**
* **Integration Testing: Ensuring components work together (e.g., MusicPlayer + Playlist).**
* **E2E Testing: Cypress for user workflows (login, search, play song).**

**Code Coverage**

**Configured with Jest for tracking test coverage.**

**12. Future Enhancements :**

🎧 Offline Mode – Save songs for offline listening.

💬 Community Features – Comments, ratings, and reviews.

📺 Music Videos – Integration with video playback.

🧑‍🤝‍🧑 Social Sharing – Share playlists on social media.

**13.Conclusion:**

[ Rhythmic tunes ] designed to be a sleek, modern, and enjoyable music streaming platform. With its intuitive design, customizable playlists, and seamless music experience, it aims to become a go-to destination for music lovers worldwide.