Rhythmic Tunes

1. Introduction

This project represents the frontend of a music player web application developed using React.js. It is designed to provide a simple, elegant, and responsive interface for playing songs, managing playlists, and enjoying an interactive music experience.

Project Title: Rhythmic Tunes I

Team Members:

1. Monika M
2. Abinaya E
3. Keerthika T
4. Dhatchayini S

2. Project Overview

Purpose: This project is a music player web application built using React.js. It allows users to play, pause, and navigate through a collection of songs with a clean and responsive UI.

Features:

1 Upload and manage songs 2 Play/Pause functionality 3 Playlist support 4 Responsive design with images and audio integration

3. Architecture

Component Structure: The project includes App.jsx as the main entry point. Components handle player controls, song list display, and navigation.

State Management: React useState is used for handling component-level state. Routing: Vite-based project with React Router support for future enhancements.

4. Setup Instructions

Prerequisites: Node.js and npm must be installed. Installation Steps:

1 Clone the repository

2 Navigate to project directory 3 Run npm install to install dependencies 4 Run npm start to start the development server

5. Folder Structure

Client: React app organized with src (components, assets), public (songs, images). Utilities: Includes helper functions and hooks for handling songs and player controls.

6. Running the Application

To run the application locally: npm start

7. Component Documentation

App.jsx: Root component handling layout and routing.

SongList: Renders available songs with props for audio files.

Player: Controls playback (play, pause, next).

8. State Management

Global State: Not implemented yet, potential use of Context API or Redux.

Local State: Player and UI handled with React useState.

9. User Interface

https://drive.google.com/file/d/1FZXFji8YTfFgkhrVg-4rnOTraM8cLMjC/view?usp=drivesdk

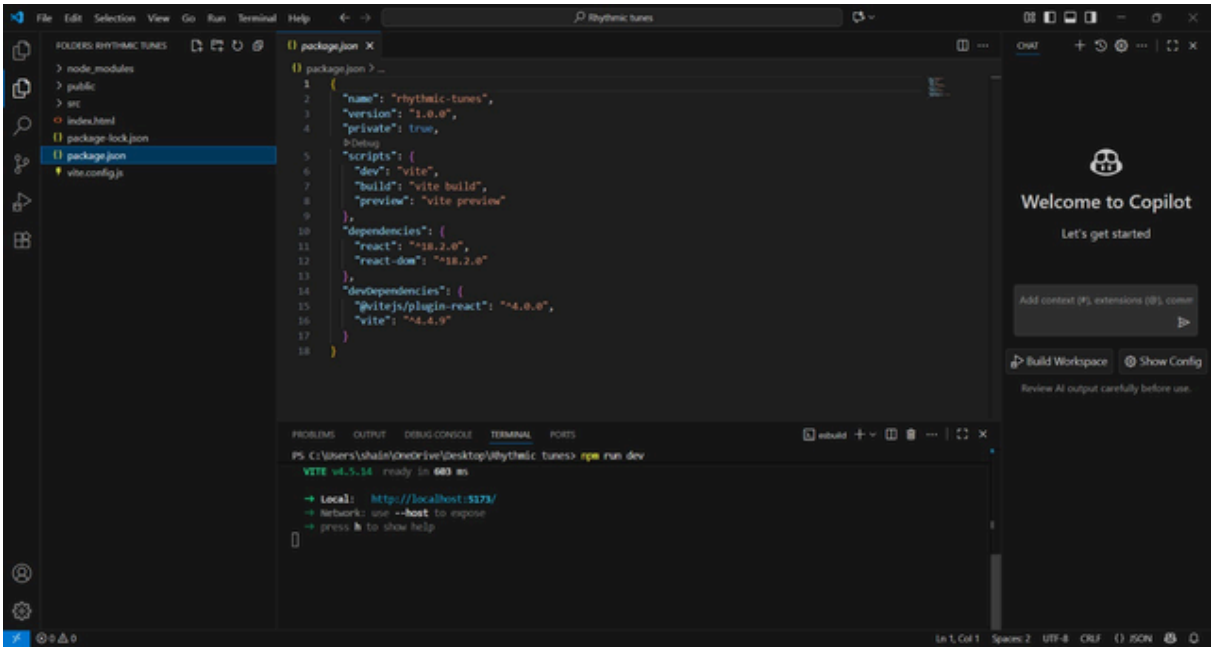
10. Styling

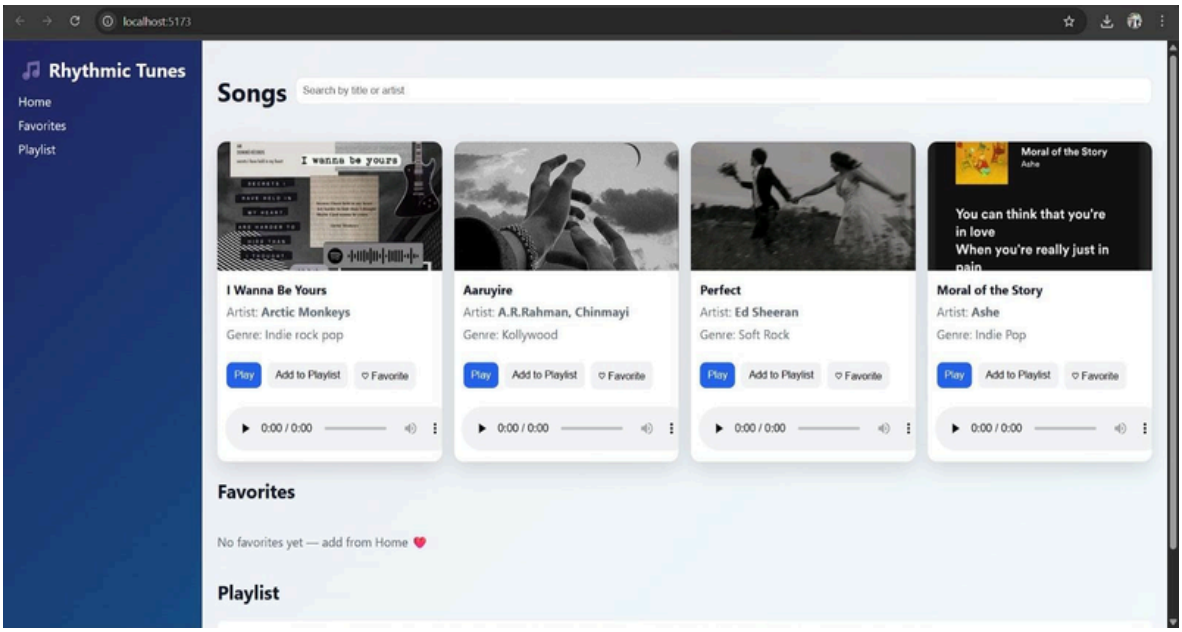
CSS Framework: Tailwind CSS is used for responsive styling.

11. Testing

Testing Strategy: Unit testing planned with Jest and React Testing Library.

12. Screenshots or Demo


File Edit Selection View Go Run Terminal Help
Rhythmic tunes
FOLDERS: RHYTHMIC TUNES
node_modules
public
src
index.html
package-lock.json
package.json
vite.config.js
package.json
package.json > ...
{
"name": "rhythmic-tunes",
"version": "1.0.0",
"private": true,
Debug
"scripts": {
"dev": "vite",
"build": "vite build",
"preview": "vite preview"
},
"dependencies": {
"react": "^18.2.0",
"react-dom": "^18.2.0"
},
"devDependencies": {
"@vitejs/plugin-react": "^4.0.0",
"vite": "^4.4.9"
}
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
esbuild
PS C:\Users\shain\OneDrive\Desktop\Rhythmic tunes> npm run dev
VITE v4.5.14 ready in 603 ms
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h to show help
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), comm
Build Workspace
Show Config
Review AI output carefully before use.
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
JSON



13. Known Issues

Some audio formats may not be supported across all browsers.

14. Future Enhancements

1 Implement Redux for state management

2 Add user authentication

3 Enhance UI with animations

4 Persist playlists with backend integration