

**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. Dilli Ganesh
2. John Babu
3. Karthikeyan
4. Sanjay

**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**

The UI includes a song list, player controls, and responsive styling with Tailwind CSS.

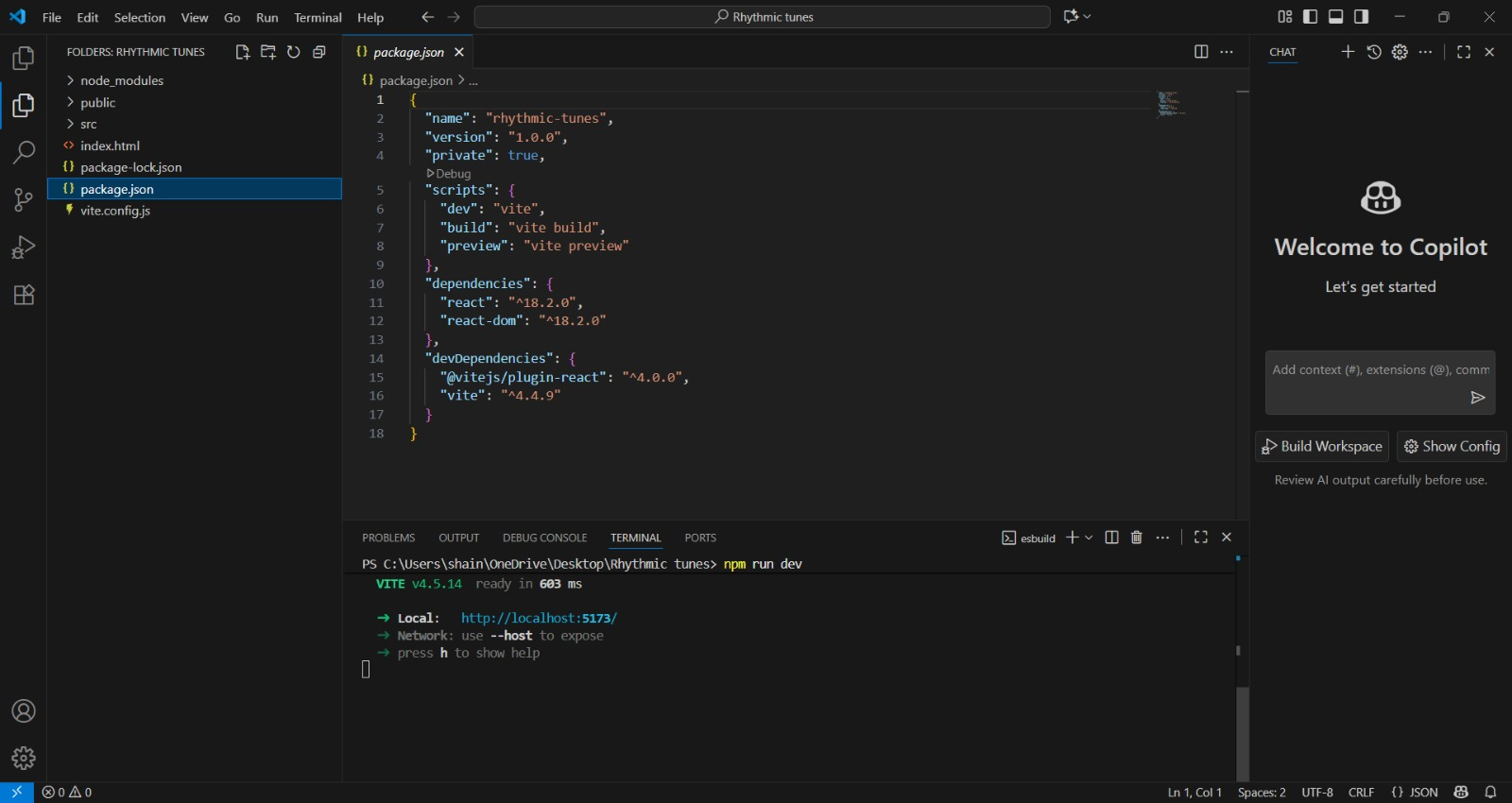
**10. Styling**

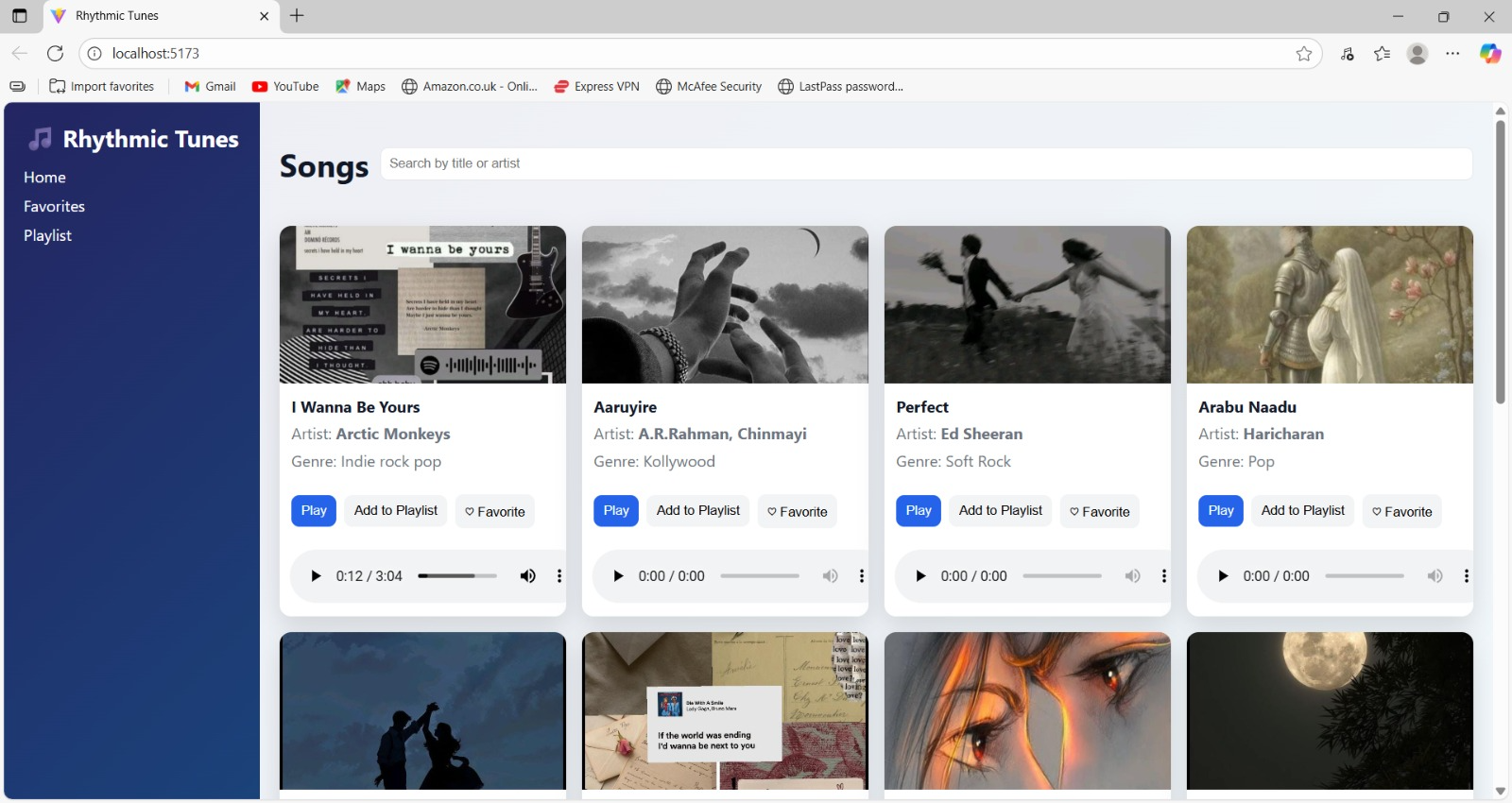
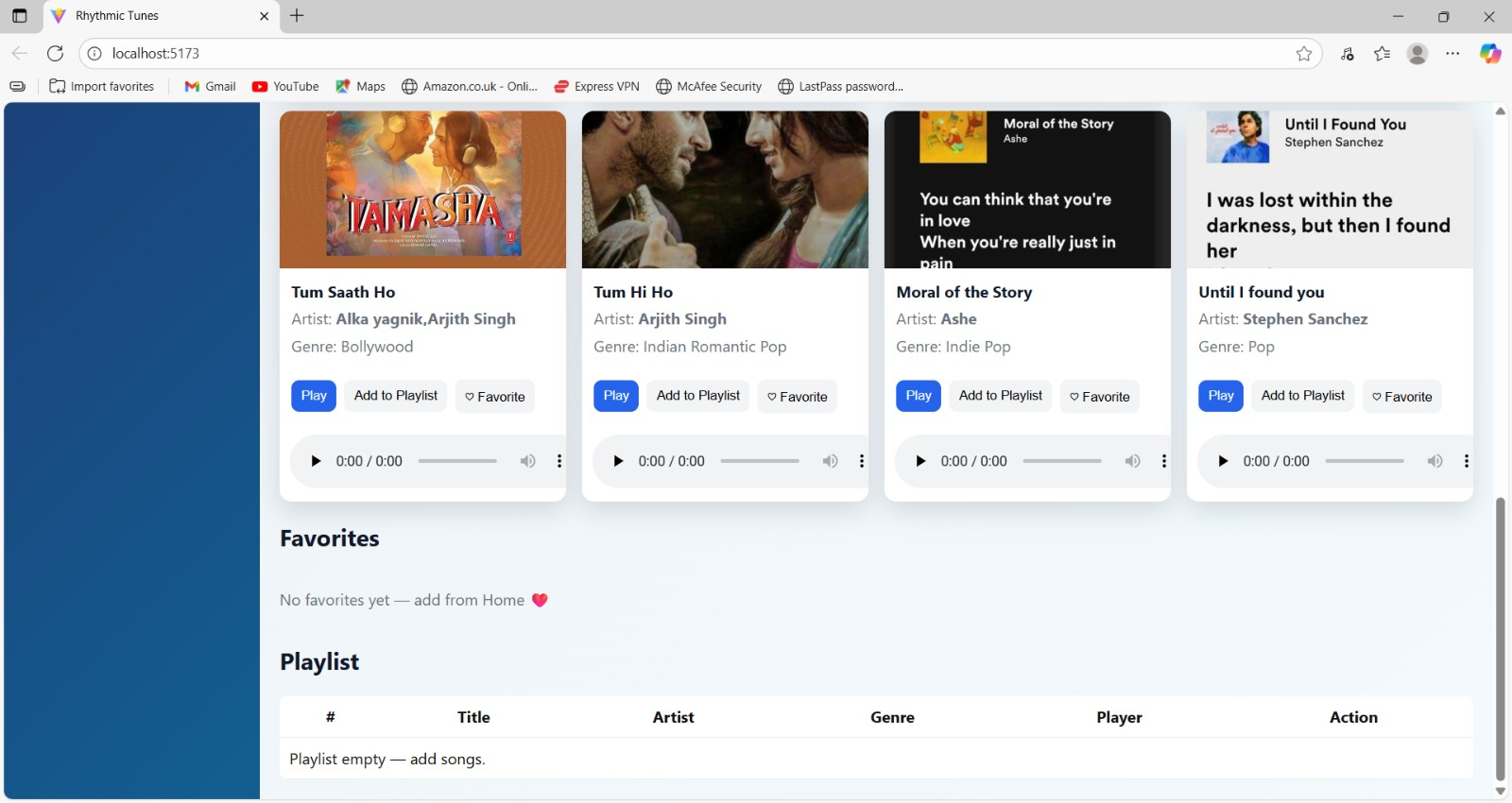
CSS Framework: Tailwind CSS is used for responsive styling.

**11. Testing**

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

**12. Screenshots or Demo**



****

**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