**1. Introduction**

**Project Title:** Rhythmic Tunes

**Team Leader:**

* PARTHIBAN R [email id : pa381026@gmail.com]

**Team Members:**

* Mohan Raj G [email id :[mohanraj24305@gmail.com](mailto:mohanraj24305@gmail.com)]
* Nethaji S [email id :snethaji2004@gmail.com]

**2. Project Overview**

**Purpose**

Rhythmic Tunes is a dynamic music streaming platform that allows users to explore, stream, and curate playlists of their favourite tracks. The project aims to provide an intuitive and engaging user experience through a well-structured frontend built with React.

**Features**

* User authentication and profile management
* Music search and playlist creation
* Streaming functionality with media controls
* Interactive UI with dynamic animations
* Responsive design for seamless user experience across devices

**3. Architecture**

**Component Structure**

The frontend is structured using a modular React component approach. The main components include:

* **Navbar**: Handles navigation across the application
* **Homepage**: Displays trending and recommended music
* **Search**: Allows users to find songs and artists
* **Player**: Provides music playback controls
* **Profile**: Displays user details and preferences

**State Management**

Rhythmic Tunes utilizes the **Context API** for global state management, ensuring efficient data sharing across components. Additionally, local component states manage UI interactions.

**Routing**

The application employs **React Router** for navigation. Major routes include:

* / - Homepage
* /search - Search results page
* /profile - User profile section

**4. Setup Instructions**

**Prerequisites**

Ensure the following dependencies are installed:

* Node.js
* npm

**Installation:**

1. Clone the repository:

git clone https://github.com/Sathish0116/Rhythmic-tunes

1. Navigate to the project directory:

cd Rhythmic-tunes

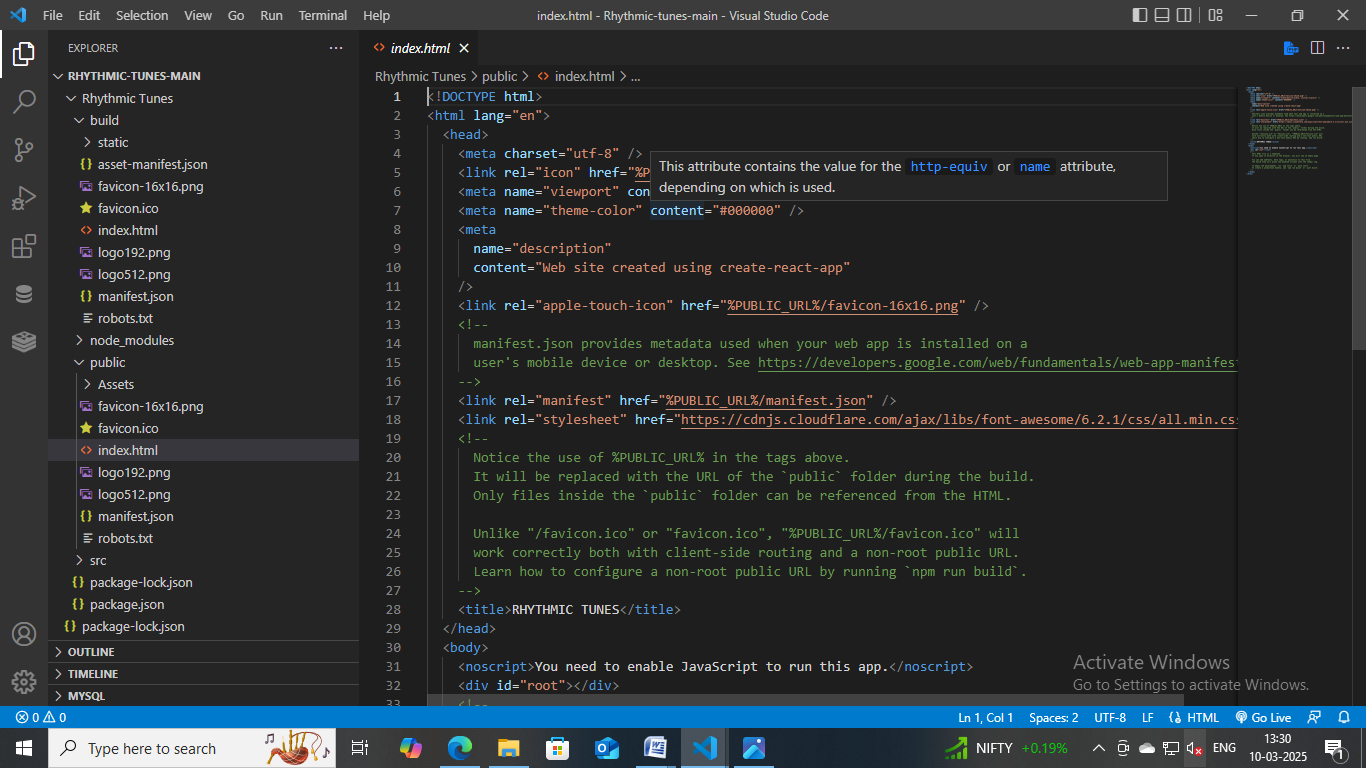
1. Install dependencies:

npm install

1. Configure environment variables in a .env file.
2. Start the development server:

npm start

**5. Project Structure:**

****

**6. Running the Application:**

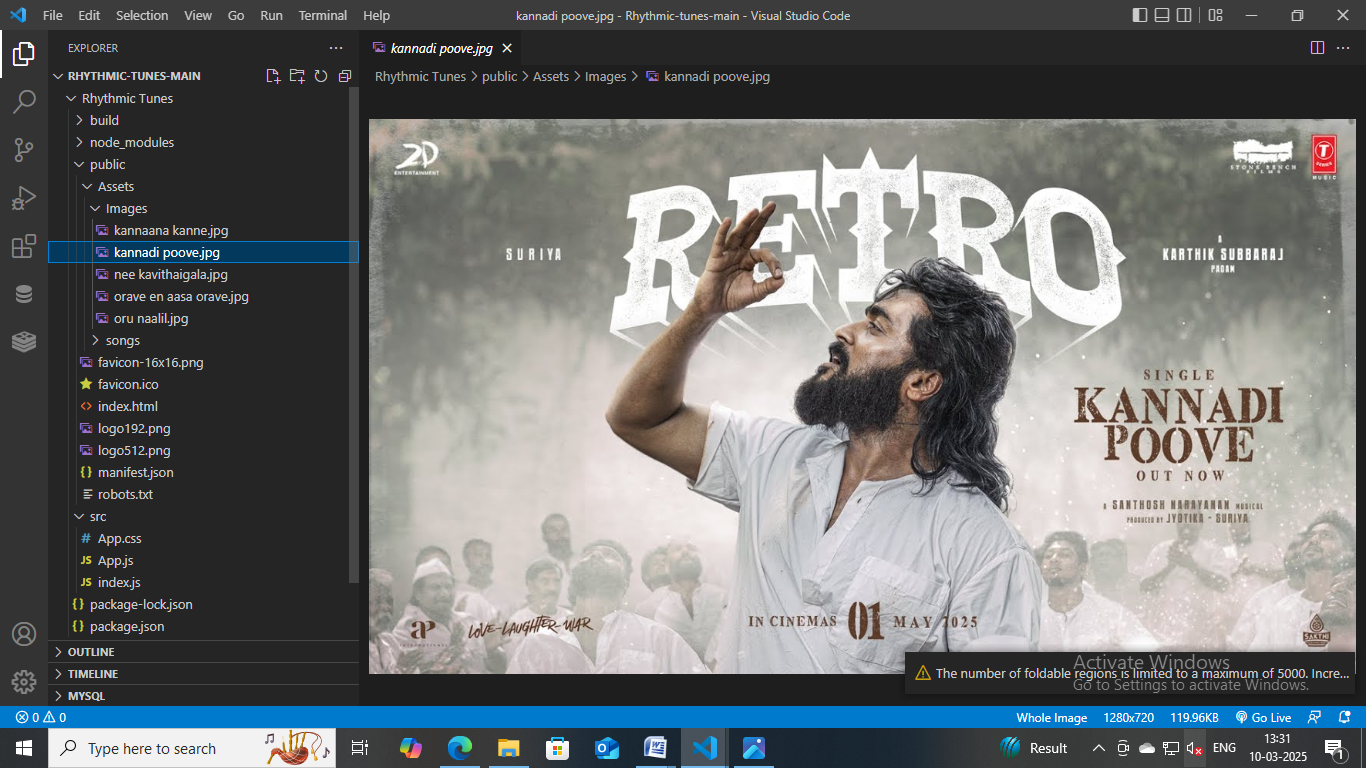
To run the frontend locally:

npm start

**7. Component Documentation:**

**Key Components**

* **Navbar** - Displays navigation links
* **SearchBar** - Handles music searches
* **MusicCard** - Represents a single song item
* **PlayerControls** - Provides music playback controls



**Reusable Components**

* **Button** - Custom button component
* **Modal** - Generic modal component

**8. State Management**

**Global State**

The Context API manages:

* User authentication status
* Music playback queue
* User preferences and playlists

**Local State**

Each component maintains its own local state for UI interactions such as toggling modals and handling form inputs.

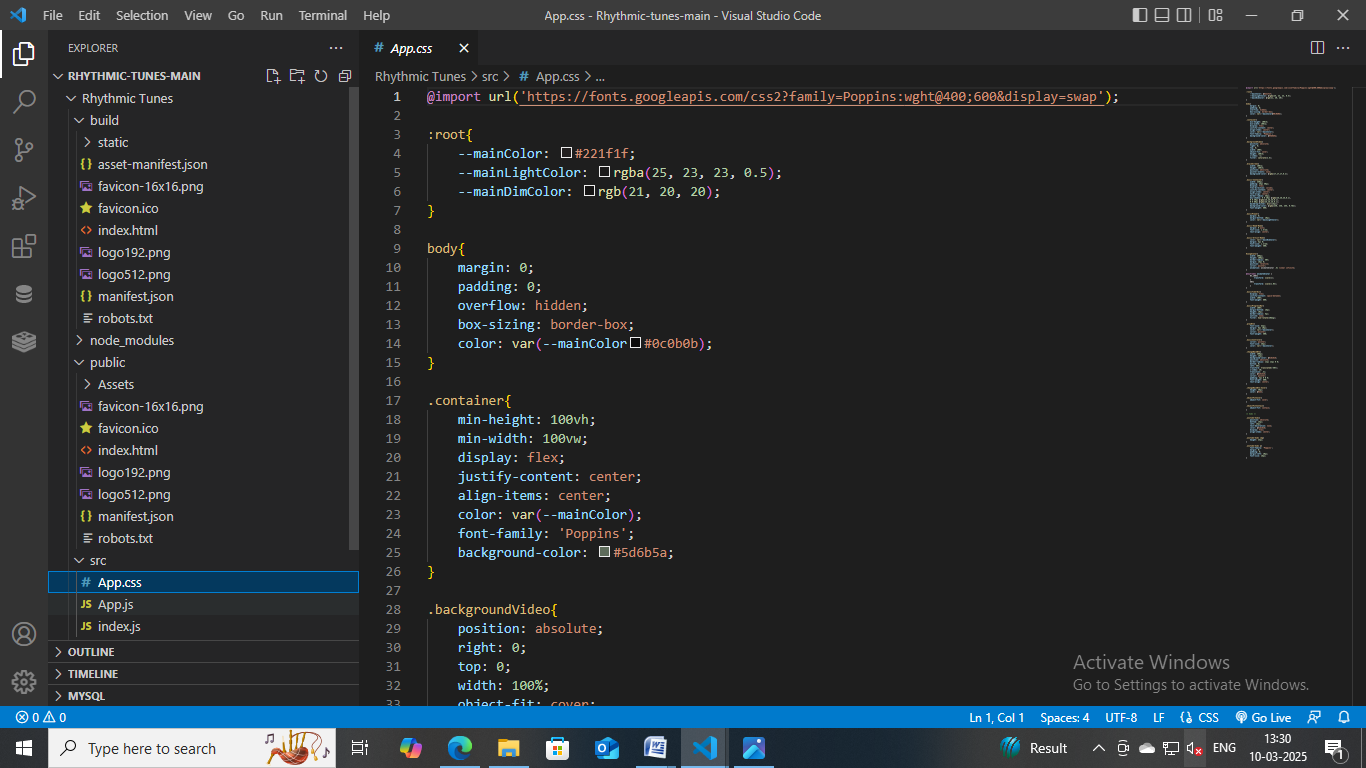
**9. User Interface**

[Include screenshots or GIFs showcasing the interface]

**10. Styling**

**CSS Frameworks/Libraries**

* **Tailwind CSS** for utility-based styling
* **Styled-Components** for dynamic theming



**Theming**

A custom dark/light theme system is implemented with styled-components.

**11. Testing**

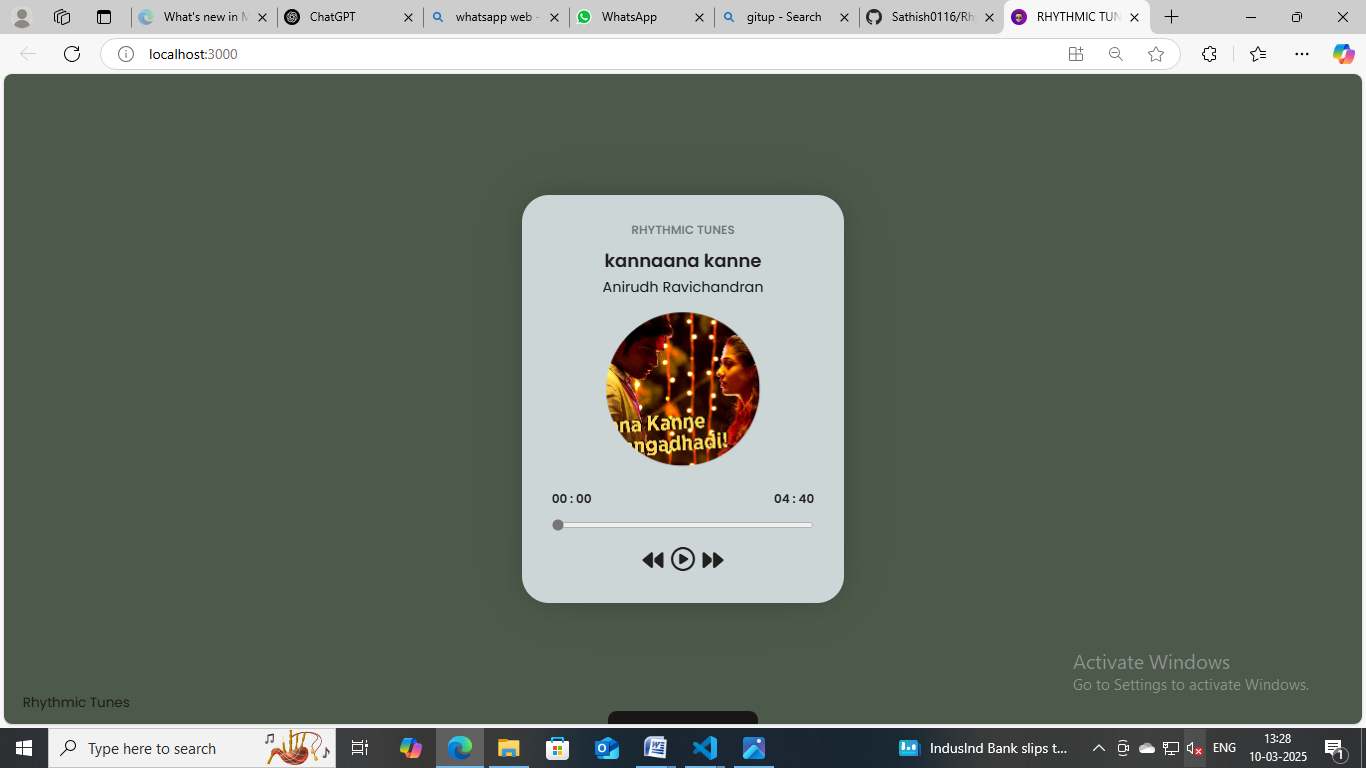
**Testing Strategy**

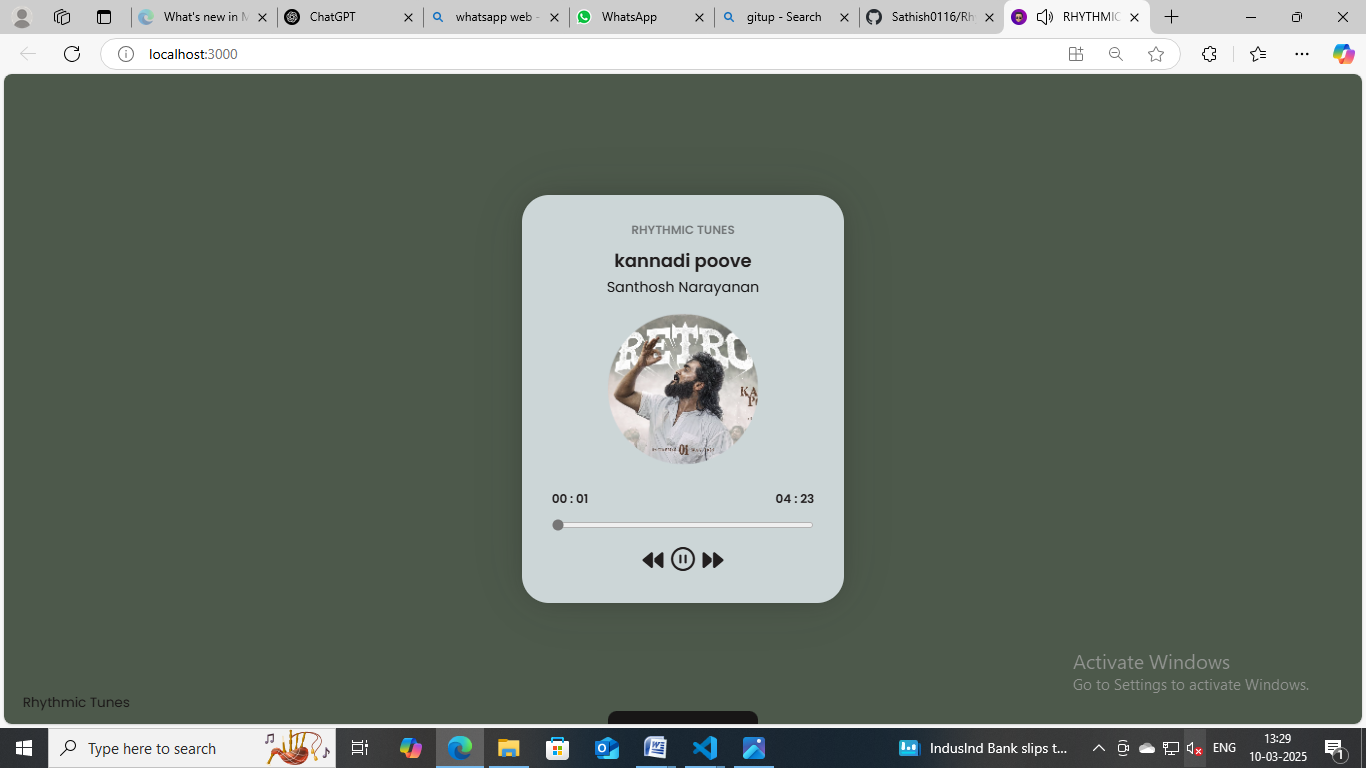
* **Unit Testing** - Jest and React Testing Library for individual components
* **Integration Testing** - Ensuring seamless interactions between components
* **End-to-End Testing** - Cypress for testing user flows

**Code Coverage**

Jest provides coverage reports to ensure comprehensive test coverage.

**12. Screenshots or Demo**





**13. Known Issues**

* Audio playback may have minor delays on slow networks.
* Mobile UI needs further optimization for smaller screens.

**14. Future Enhancements**

* AI-driven music recommendations
* Offline playback support
* Enhanced social features such as following users and sharing playlists