PROJECT TITLE: RYTHMIC TUNES: YOUR MELODIC COMPANION

TEAM DETAILS:

TEAM ID: NM2025TMID40084

TEAM MEMBERS:

TEAM LEADER: HARINI S

NM ID: 7B5FBDF14C60BA0B18E4C0F97C566B84

TEAM MEMBER: JAYA SRI R

NM ID: 46FA5ABC4F59736812C46F90B9A0A125

TEAM MEMBER: SANDHIYA M

NM ID: 88F43933E420CCA9A7849156A8769EA1

TEAM MEMBER: HARINI S

NM ID: CB5B2C4BAC94036817C4D076E70126E5

OBJECTIVE

The primary goal of Music Streaming is to provide a seamless platform for music enthusiasts, enjoying, and sharing diverse musical experiences. Our objectives include:

User-Friendly Interface: Develop an intuitive interface that allows users to effortlessly explore, save, and share their favorite music tracks and playlists.

Comprehensive Music Streaming: Provide robust features for organizing and managing music content, including advanced search options for easy discovery.

Modern Tech Stack: Harness cutting-edge web development technologies, such as React.js, to ensure an efficient and enjoyable user experience while navigating and interacting with the music streaming application.

PLATFORM AD TECHNOLOGY USED

1. Frontend Development

- **HTML5**: For structuring the web pages.
- **CSS3**: For styling, layout, and visual design.
- **React.js**: For creating dynamic, component-based user interfaces.

2. Backend / Runtime Environment

- **Node.js**: Provides the JavaScript runtime environment needed for React development and package execution.

3. Package Management & Build Tool

- **npm (Node Package Manager)**: For installing and managing project dependencies.
- **Vite**: Used as the development server and build tool for faster performance.

4. Development Environment & Version Control

- **Visual Studio Code (VS Code)**: Code editor for building and managing the project.
- **GitHub**: For version control, storing source code, and collaboration.

5. Platform Compatibility

- The website runs entirely in **web browsers** and is fully **responsive**, making it accessible on desktops, laptops, tablets, and smartphones.

IMPLEMENTATION/PROCESS

1. Project Setup

- The project was initialized using **Vite** along with **React.js** for fast development and modular design.
- **Node.js** and **npm** were installed to set up the runtime environment and manage required dependencies.
- A GitHub repository was created to store and maintain the project code.

2. User Interface Design

- The **HTML5** structure and **CSS3** styling were used to design the layout of the music player.
- A **navigation sidebar** was implemented to provide easy access to different sections such as Home, Playlist, and Favorites.
- The **main content area** was designed to display songs, album covers, and playback controls with a clean and responsive interface.

3. Music Player Functionality

- The built-in **HTML <audio> element** was integrated for song playback.
- Core features such as **play, pause, forward, backward, and volume control** were implemented using **JavaScript**.
- A **favorites system** was added, allowing users to mark and store their preferred songs.

- A **playlist section** was created to organize and display selected tracks.

4. React.js Integration

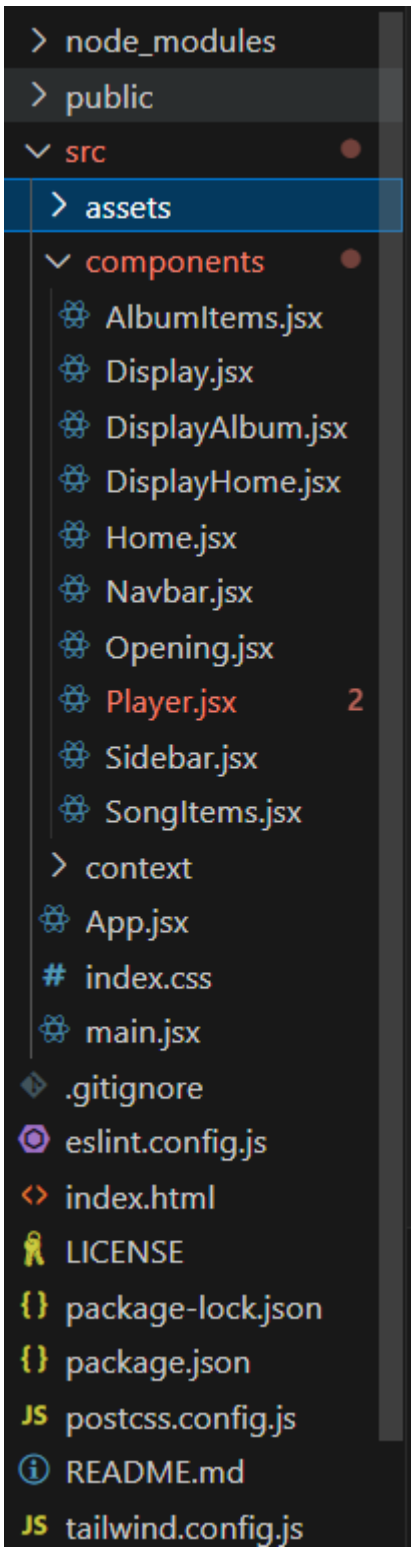
- The project was divided into **reusable React components** (Navbar, Sidebar, Player, Playlist, etc.).
- **State management** was implemented to handle user interactions like playing songs, adding favorites, and updating playlists.
- Smooth rendering and component reusability improved overall performance and code efficiency.

5. Testing and Deployment

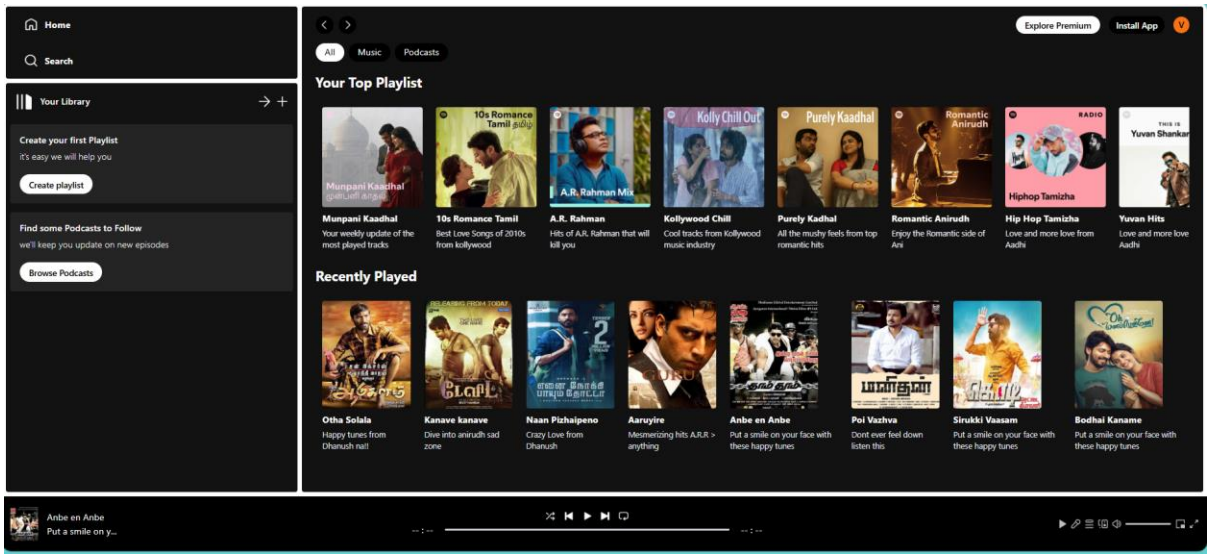
- The application was tested locally in **Visual Studio Code (VS Code)** using the Vite development server (localhost:5173).
- Debugging was carried out to fix styling issues, broken links, and functionality errors.
- The final project was committed and pushed to **GitHub** for version control and future hosting.

OUTPUT

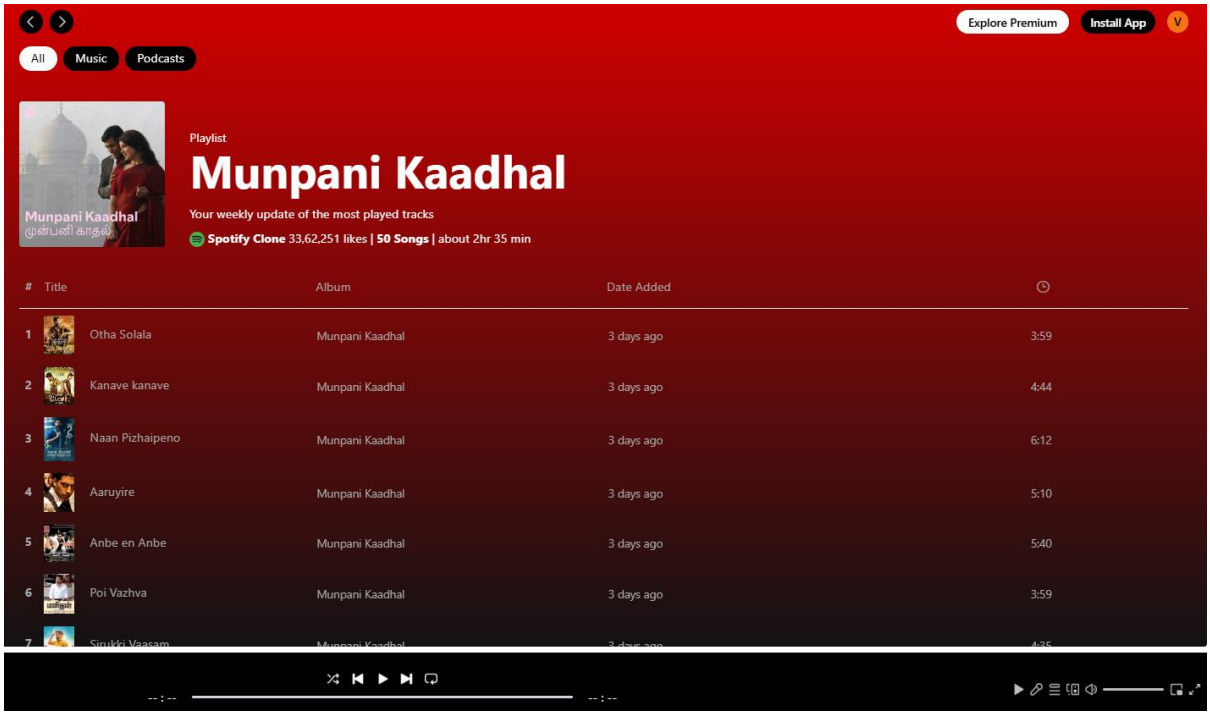
Project structure:



Homepage:



Playlist:



Player: