RHYTHMIC TUNES

**INTRODUCTION**:-

Project Title:-

Rhythmic Tunes.

* Team ID:-
* NM2025TMID38950
* Team Leader:-

|  |  |
| --- | --- |
| Anok K – | [Anokanok51@gmail.com](mailto:Anokanok51@gmail.com) |

* Team Members:-

|  |  |  |
| --- | --- | --- |
|  | Kamaresh.A | [Kamaresharumugam@gmail.com](mailto:Kamaresharumugam@gmail.com) |
|  | Dilli babu.R | [Dilli0052@gmail.com](mailto:Dilli0052@gmail.com) |
|  | Jaya prakash.N | [S90256965@gmail.com](mailto:S90256965@gmail.com) |
|  | Mohammed Ibrahim.A | [Mohammedmohammedz870@gmail.com](mailto:Mohammedmohammedz870@gmail.com) |

1.

**1. PROJECT OVERVIEW:**

**1.Purpose:-**  
The purpose of *Rhythmic Tunes* is to create a comprehensive and user-friendly platform that enhances the way users interact with music. It is designed to not only provide access to songs and playlists but also integrate advanced features like secure communication, feedback systems, and admin-level management. The project bridges entertainment, interaction, and security into a single music-based application.

**2.Features:-**  
 The major features of *Rhythmic Tunes* include:

* + **Project Posting and Binding** – Enables users to share music-related content, projects, or playlists with the community and bind them with relevant categories for better accessibility.
  + **Secure Chat System** – A built-in chat module that ensures private, encrypted communication between users for collaboration and interaction.
  + **Feedback and Review System** – Allows users to provide feedback on songs, playlists, or shared content, helping improve recommendations and community engagement.
  + **Admin Control Panel** – A dedicated dashboard for administrators to manage users, content, and system activities effectively, ensuring smooth operation and security.

**2.ARCHITECTURE:-**

1. **Frontend – React.js with Bootstrap and Material UI**  
   The frontend of *Rhythmic Tunes* is built using **React.js**, ensuring a highly dynamic and responsive user interface. **Bootstrap** provides a robust grid system and layout framework, while **Material UI** adds modern, pre-designed components for consistency and visual appeal. Together, they deliver a smooth, intuitive, and interactive user experience across devices.
2. **Backend – Node.js and Express.js**  
   The backend is powered by **Node.js** for fast, scalable, and event-driven server operations. **Express.js** is used to handle routing, server logic, and API endpoints, ensuring efficient communication between the frontend and database. This combination supports secure data processing and reliable application performance.
3. **Database – MongoDB**  
   **MongoDB** serves as the primary database, chosen for its flexibility in handling unstructured data. It stores user information, project postings, application details, and chat messages in a secure and scalable manner. Its document-oriented structure makes it well-suited for real-time interactions and large-scale data management.

**3.SETUP INSTRUCTIONS:-**

1. **Prerequisites**  
   Before setting up *Rhythmic Tunes*, ensure that the following tools and technologies are installed on your system:
   * **Node.js** – JavaScript runtime environment.
   * **MongoDB** – Database for storing user data, projects, and chat messages.
   * **Git** – For cloning and version control.
   * **React.js** – Frontend framework for building the user interface.
   * **Express.js** – Backend framework for handling APIs and server logic.
   * **Mongoose** – ODM (Object Data Modeling) library for MongoDB.
   * **Visual Studio Code** – Recommended IDE for development.
2. **Installation Steps**
   * **Clone the Repository**
   * git clone <repository\_url>
   * **Install Client Dependencies**
   * cd client
   * npm install
   * **Install Server Dependencies**
   * cd ../server
   * npm install

After installation, configure the environment variables (like MongoDB connection string and server port), then run both the **client** and **server** to start the application.

**4.FOLDER STRUCTURE:-**

Rhythmic tunes/

│

├── client/ # React Frontend

│ ├── components/ # Reusable UI components

│ └── pages/ # Application pages and views

│

├── server/ # Node.js Backend

│ ├── routes/ # API endpoint definitions

│ ├── models/ # Mongoose schemas for MongoDB

│ └── controllers/ # Business logic and request handling

This structure ensures a clear separation between the **frontend (React)** and **backend (Node.js + Express)**, making development and maintenance more efficient.

**5.RUNNING THE APPLICATION:-**

The document provides commands to run a frontend and backend application:

* **Frontend**:
  + **cd client**: This command likely navigates to the frontend directory of the project.
  + **npm start**: Starts the frontend application using Node Package Manager (npm).
* **Backend**:
  + **cd server**: Navigates to the backend server directory.
  + **npm start**: Starts the backend server using npm.
* **Json-server**:
* Install json-server into the program.
* Run it with npm I –g jsonserver.
* And then again run it with json-server --watch db.json --port 3000

**Accessing the Application**:

* The application can be accessed via http://localhost:3000, which is a local address where the frontend application can be viewed in a browser.

**6.API DOCUMENTATION:-**

1. **User APIs**
   * POST /api/user/register → Register a new user account.
   * POST /api/user/login → Authenticate an existing user and return access token.
2. **Projects APIs**
   * POST /api/projects/create → Create a new project entry.
   * GET /api/projects/:id → Retrieve details of a specific project by its ID.
3. **Applications APIs**
   * POST /api/apply → Submit an application to a project.

**7.AUTHENTICATION:-**

In Rhythmic Tunes, authentication was implemented in a **basic form without JWT or third-party authentication frameworks**. The system allows users to register with a username, email, and password, which are stored in the database. Login validation simply checks user credentials against the stored records. Since advanced token-based security like JWT was not used, the current approach is simpler but less secure. Future improvements could include implementing JWT or OAuth to provide stronger protection, secure sessions, and better handling of private routes.

**8.USER INTERFACE:-**

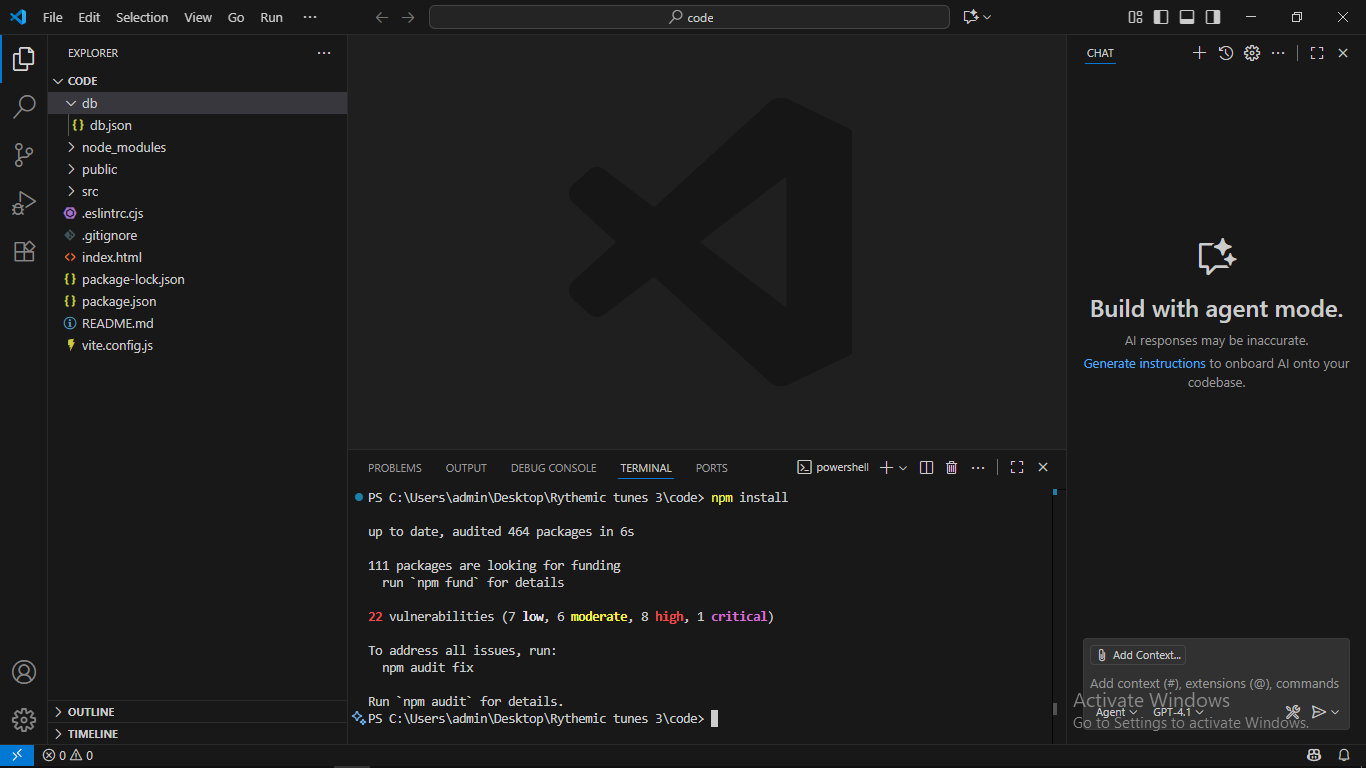
1. **Landing Page**  
   The landing page serves as the entry point to *Rhythmic Tunes*. It provides an overview of the platform, highlights key features. Its design is simple, engaging, and user-friendly to attract first-time visitors.
2. **Songs Screen**
   1. Displays the full music library.
   2. Users can browse by genre, artist, or album.
   3. Includes search bar for quick song lookup.
3. **Favorites Screen**
   1. Shows all the songs that the user has marked as favorites.
   2. Easy access to personal liked tracks.
   3. Option to remove or play directly from favorites.
4. **Playlists Screen**
   1. Allows users to create and manage playlists.
   2. View user-created and system-recommended playlists.
   3. Share playlists with friends or keep them private.

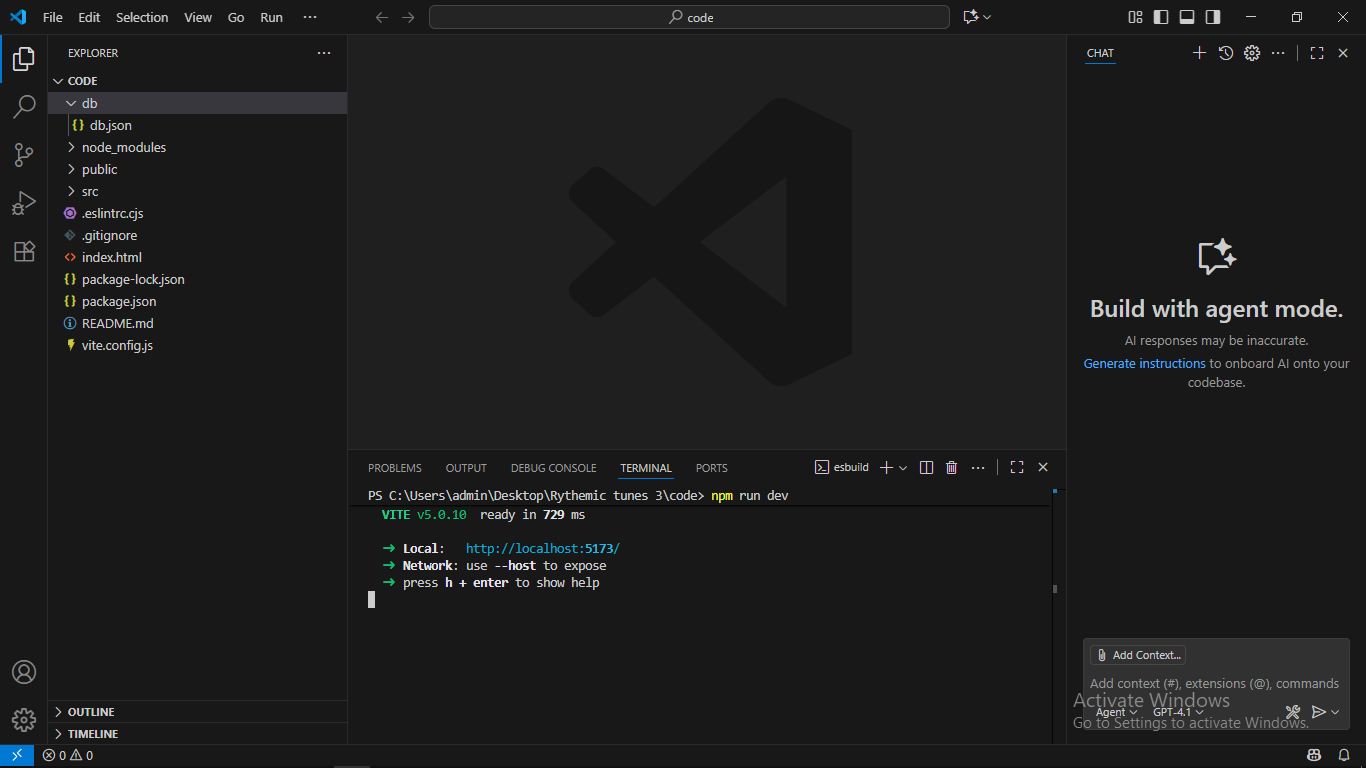
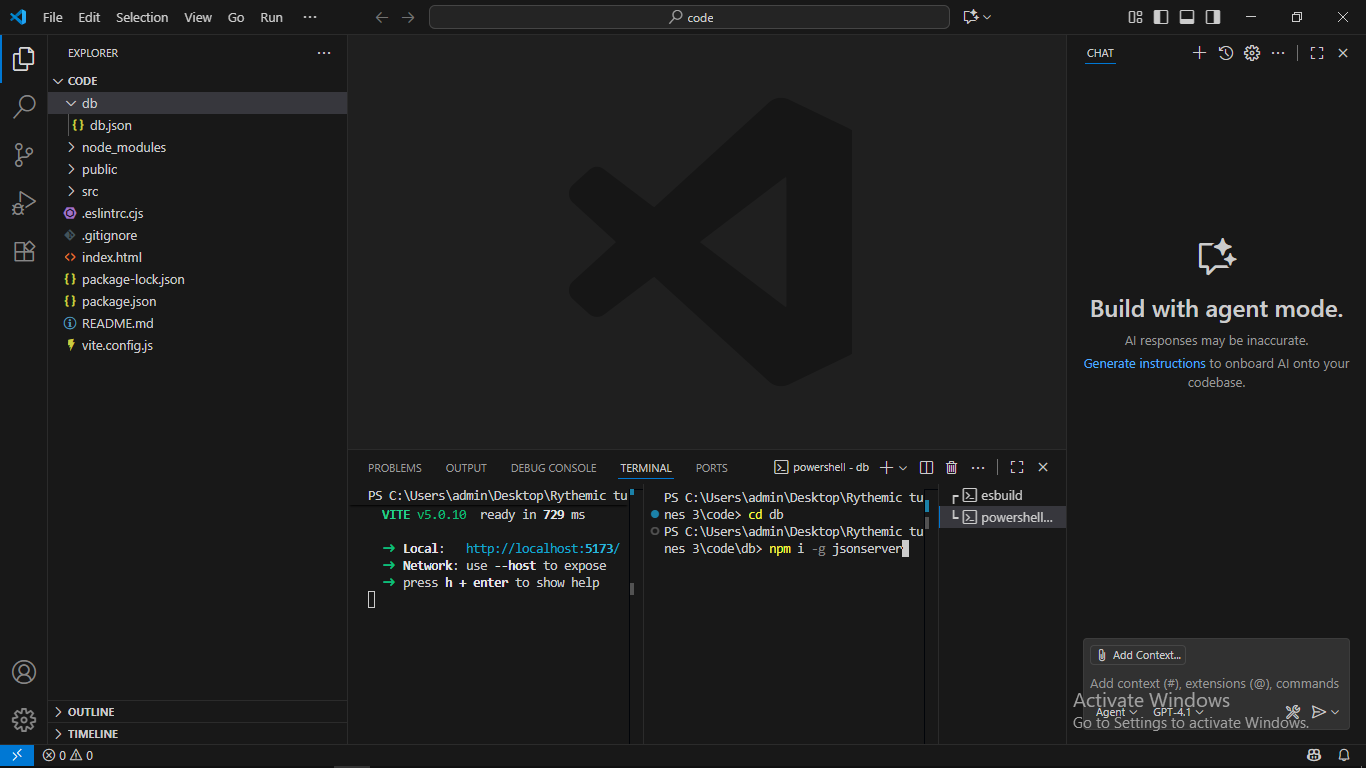
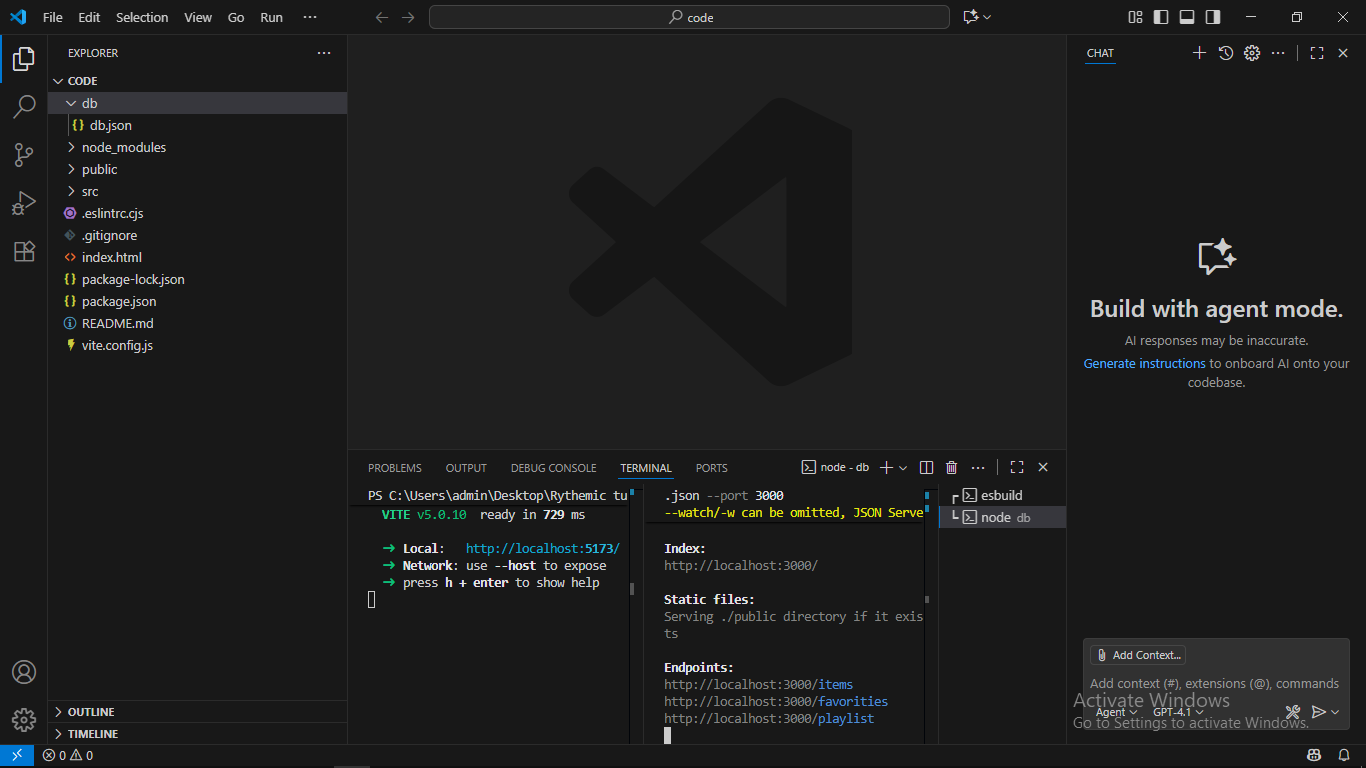
**9.TESTING:-**

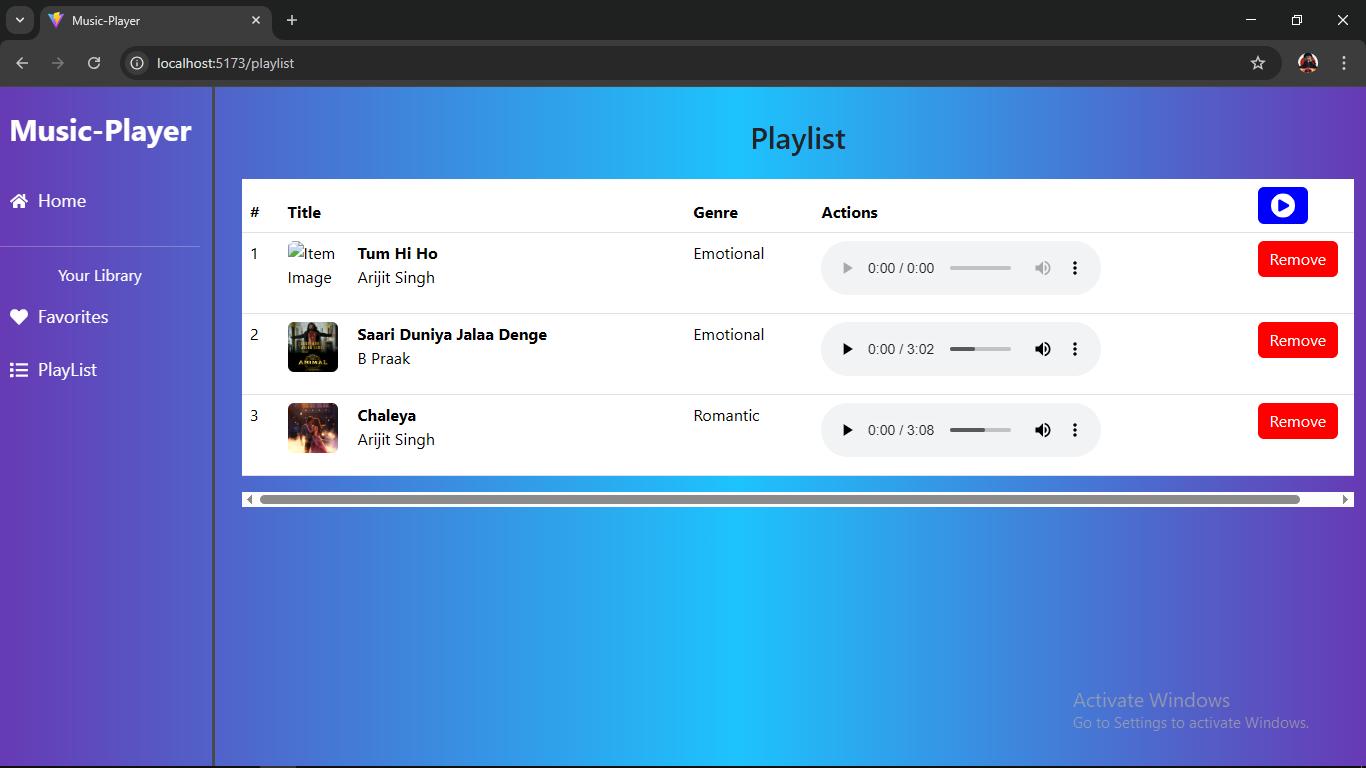
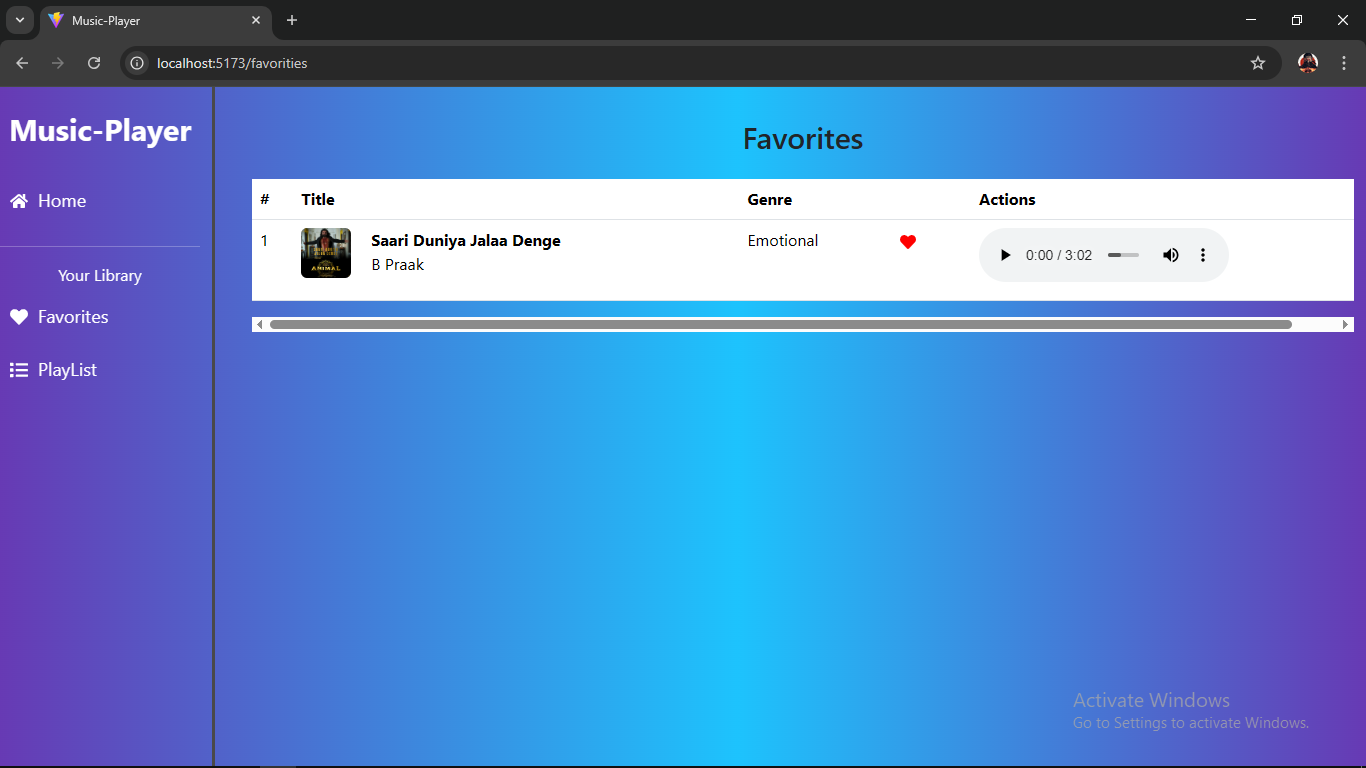
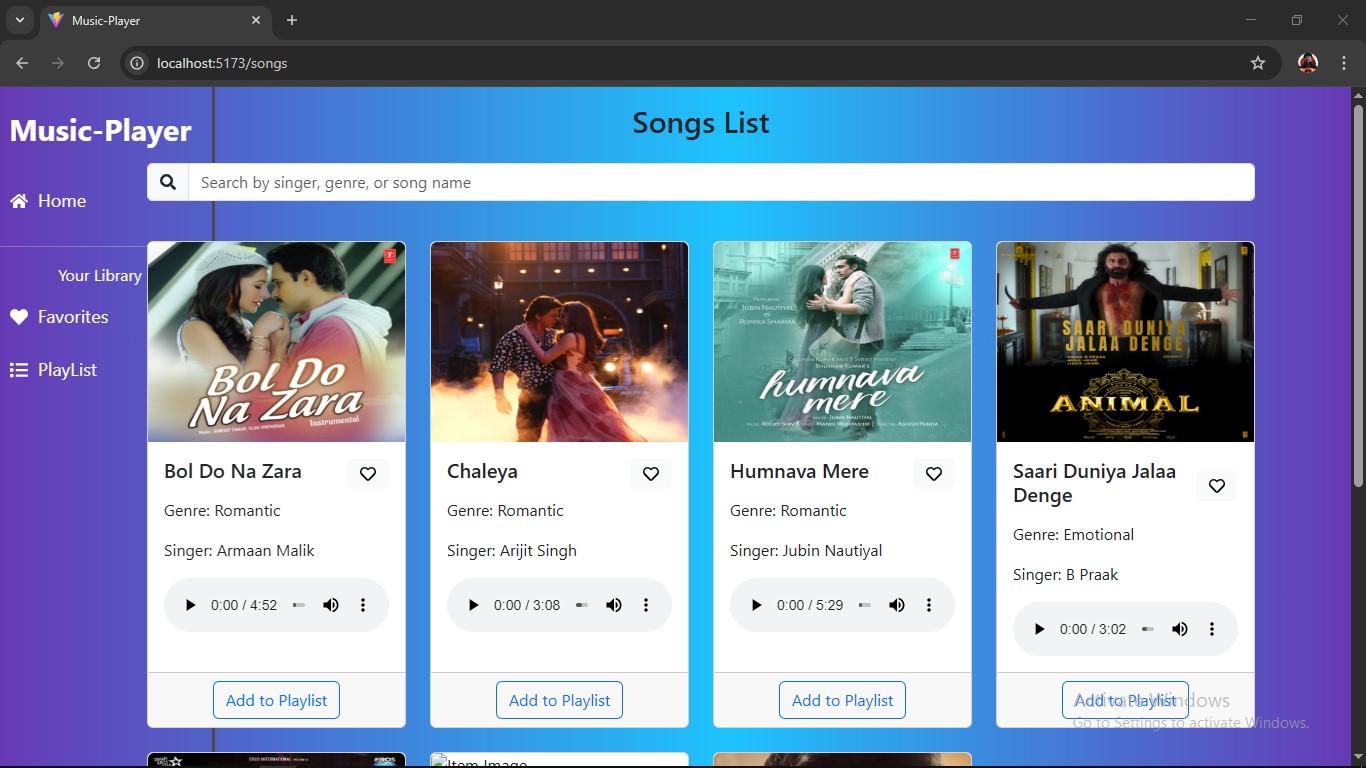
1. **Manual Testing During Milestones**

Manual testing was carried out during each milestone to verify the functionality of both frontend and backend components. During these test phases, several issues were encountered. The most significant glitch was when the website unexpectedly turned into a **blank page**, caused by frontend rendering errors. Another challenge was faced while setting up the backend, where the **JSON-server could not be installed successfully on the first attempt**, delaying initial API testing. Additionally, during early builds, the **songs and their images were not visible**, which was traced back to incorrect data linking and frontend display issues. These bugs were identified, documented, and progressively resolved, ensuring better stability in later milestones.

**10.SCREENSHOTS OR DEMO:-**

****

** **

****

**11.KNOWN ISSUES:-**

During development and testing, several issues were identified. At times, the **website became a blank page** due to frontend rendering problems, which disrupted the user experience. Another issue was faced during setup, where the **JSON-server could not be installed on the first attempt**, delaying backend integration. In early stages, the **songs and their images were not visible** on the interface, caused by incorrect linking between the database and frontend display. Even after fixes, there can still be **occasional buffering problems** on slower internet connections, and offline downloads may lead to **high storage consumption** on user devices. Additionally, the **music catalog is still limited**, with some regional songs not yet available in the current version.

**12.FUTURE ENHANCEMENTS:-**

In the future, Rhythmic Tunes can be improved by adding more features and refinements to enhance the overall user experience. Planned improvements include expanding the **music library** with a wider variety of songs and ensuring **all images and songs load properly** without glitches. The app could also introduce a **search filter by genre, artist, or album**, making it easier for users to find music. A **better favorites management system** will allow users to organize their liked songs more effectively, while **playlist customization options** such as cover images, descriptions, and sharing with friends can make the app more interactive. On the technical side, enhancing **performance to prevent blank page glitches** and ensuring **smoother installation processes** (like avoiding JSON-server setup issues) will improve reliability. Over time, adding **offline listening support** and **basic recommendation features** could further strengthen the platform.