# **Documentation for Rhythmic Tune - Your Melodic Companion**

## **1. Introduction**

* Project Title: Rhythmic Tune - Your Melodic Companion
* Team Members:
  + Yokeshwaran - Project Lead
  + Mathiarasan - Frontend Developer
  + Sanjay - Backend Developer
  + Magesh - UI/UX Designer

## **2. Project Overview**

* Purpose: The purpose of Rhythmic Tune is to provide a user-friendly platform for musicians and music enthusiasts to create, manage, and customize their music rhythms and melodies.
* Features:
  + Intuitive interface for rhythm creation.
  + Tools for melody composition.
  + Ability to save and manage musical pieces.

## **3. Architecture**

* Component Structure: Major components include:
  + RhythmCreator: For creating and editing rhythms.
  + MelodyComposer: For composing melodies.
  + MusicLibrary: For managing saved music pieces.
* State Management: Utilizes Context API for global state management.
* Routing: Implemented using React Router for navigation between different components.

## **4. Setup Instructions**

* Prerequisites:
  1. Node.js (version 14 or higher)
  2. npm (Node package manager)
* Installation:
  1. Clone the repository:
     + git clone https://github.com/yokiii12/Rhythmic-Tune-Your-Melodic-Companion.git
  2. Navigate to the project directory:
     + cd Rhythmic-Tune-Your-Melodic-Companion
  3. Install dependencies:
     + npm install
  4. Start the application:
     + npm start

## **5. Folder Structure**

* Client:
  + src/components: Contains all React components.
  + src/pages: Contains different pages of the application.
  + src/assets: Contains images, audio files, and other assets.
* Utilities:
  + src/utils: Includes helper functions and custom hooks.

## **6. Running the Application**

* To run the application locally, use the command:
  + npm start This will start the frontend server and open the application in your default web browser.

## **7. Component Documentation**

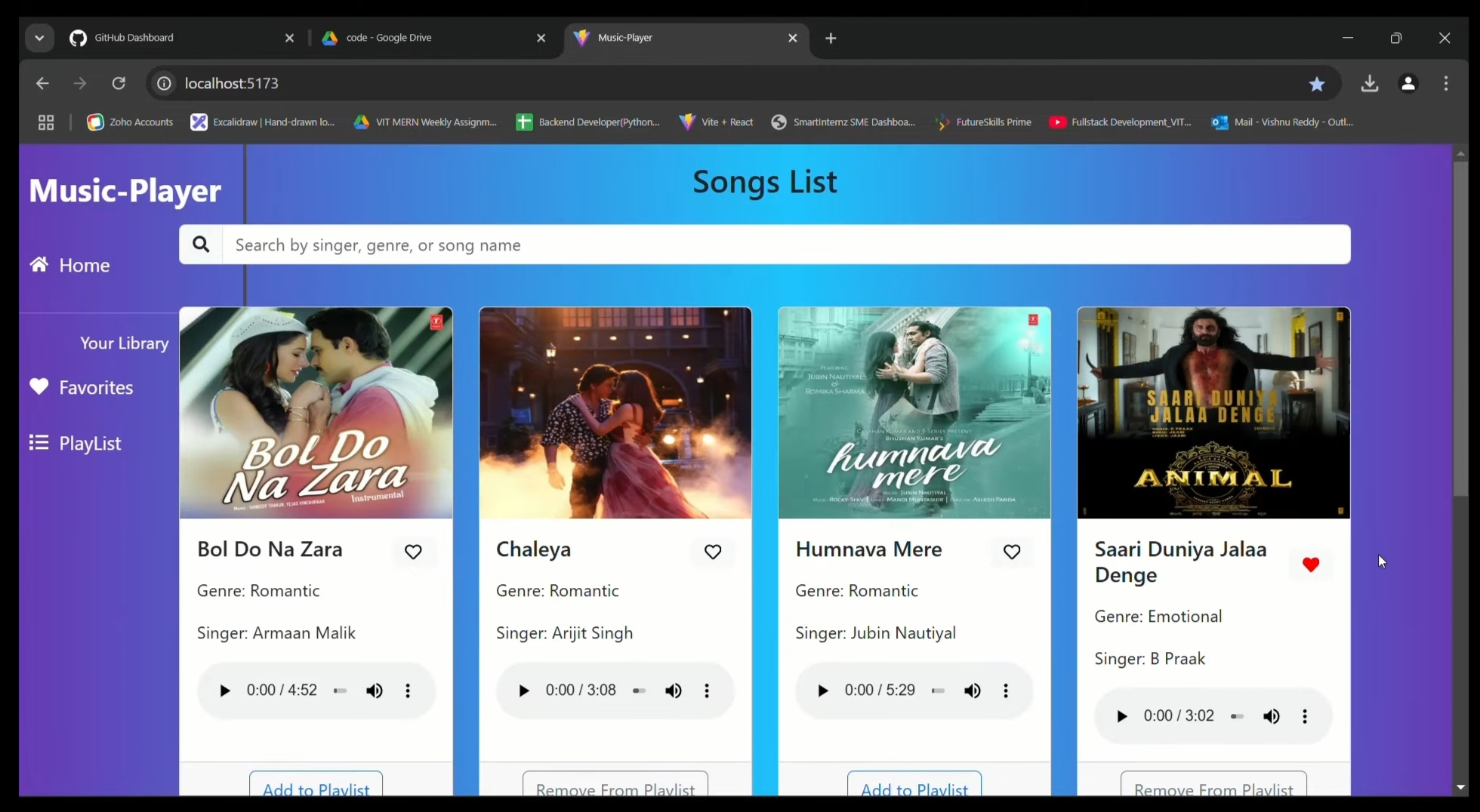
* Key Components:
  + RhythmCreator:
    - Purpose: Allows users to create and edit rhythms.
    - Props:
      * onSave: Function to save the created rhythm.
  + MelodyComposer:
    - Purpose: Enables users to compose melodies.
    - Props:
      * onSave: Function to save the composed melody.
* Reusable Components:
  + Button: A reusable button component with customizable styles and actions.

## **8. State Management**

* Global State: Managed using Context API to allow state sharing across components.
* Local State: Handled within components using React's useState hook.

## **9. User Interface**

* Visuals: Screenshots or GIFs showcasing the UI features will be added here.



## **10. Styling**

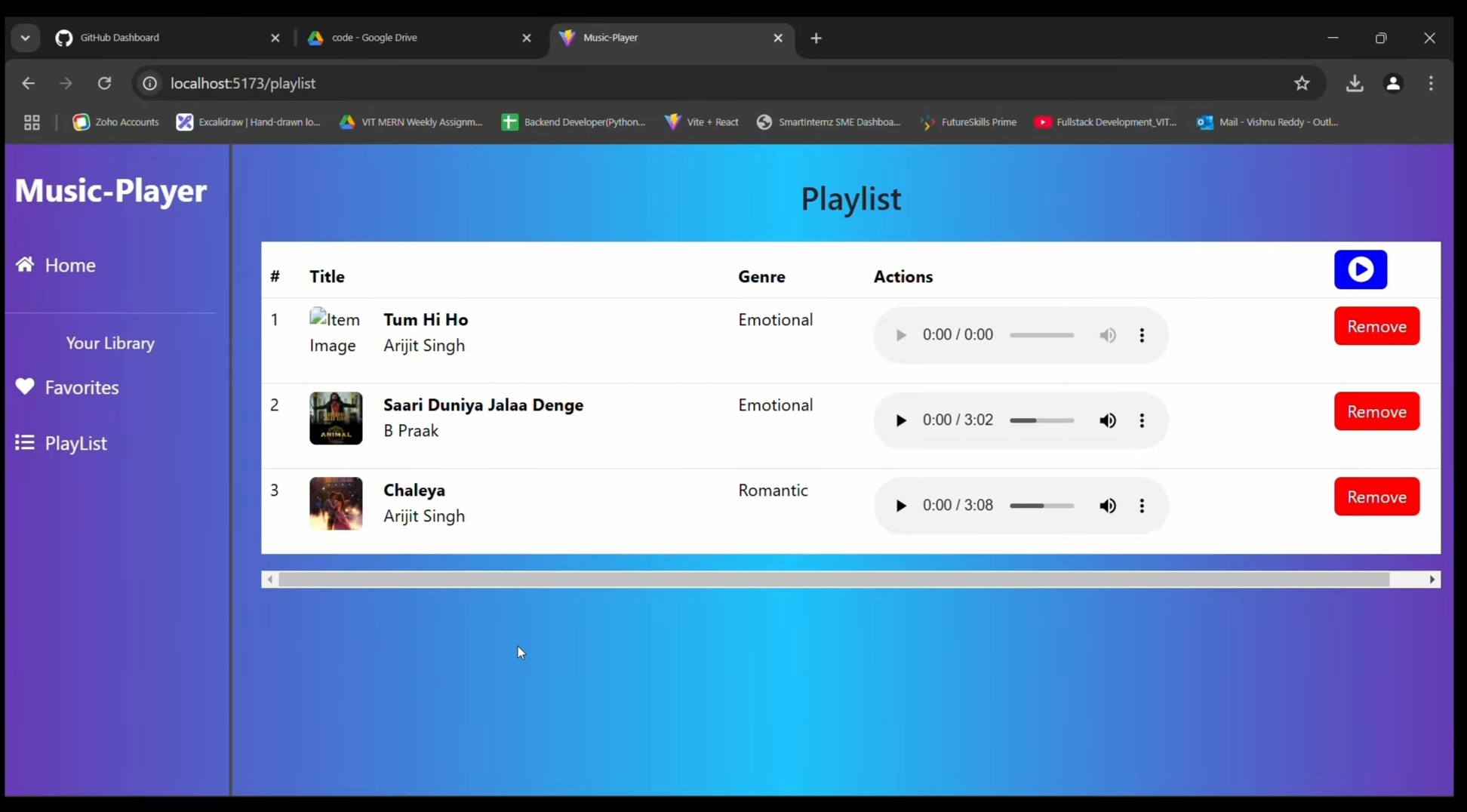
* CSS Frameworks/Libraries: Styled using CSS Modules for scoped styles.
* Theming: Custom design system implemented for consistent styling across the application.

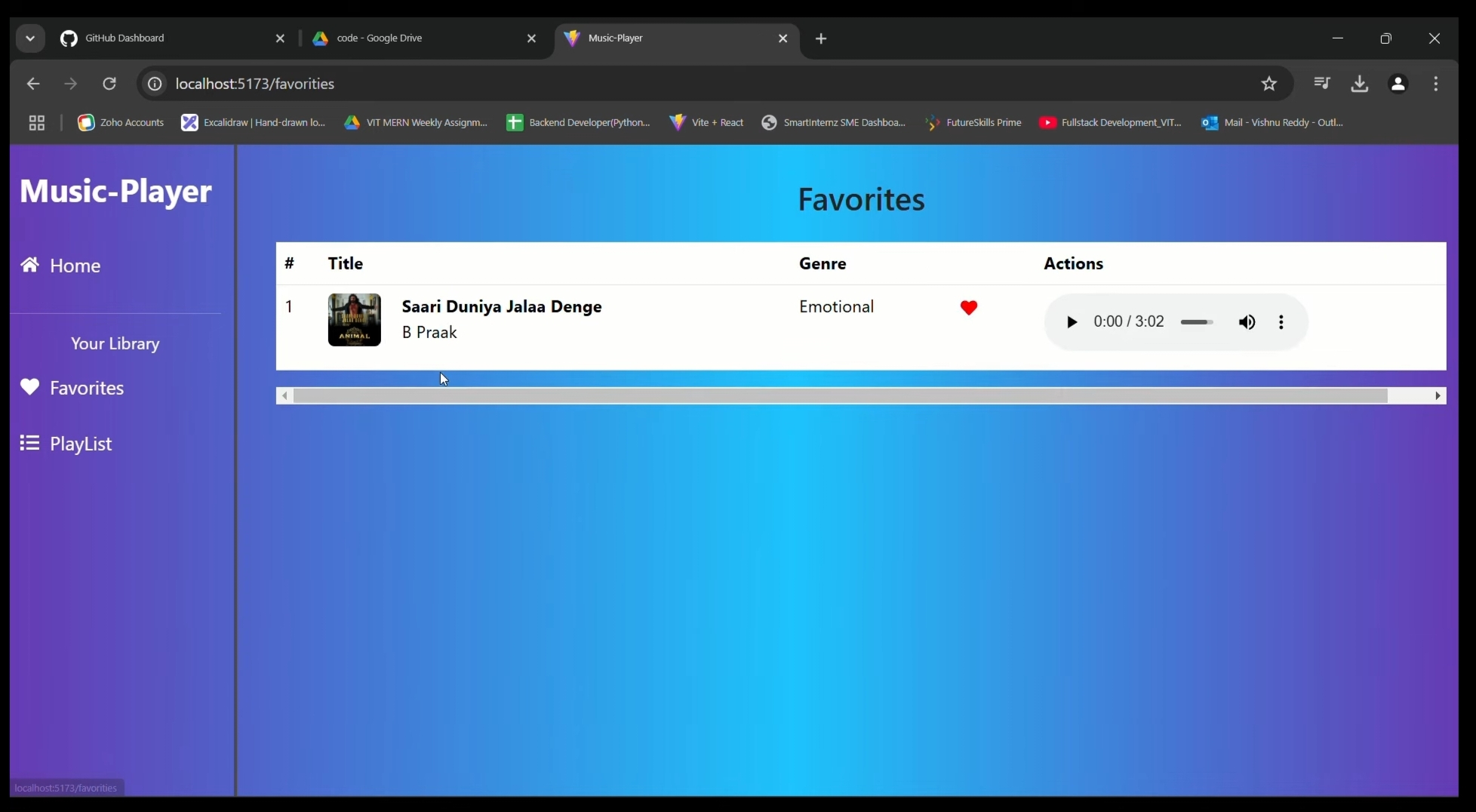
## **11. Testing**

* Testing Strategy: Unit testing using Jest and React Testing Library. Integration testing for component interactions.
* Code Coverage: Code coverage is monitored using Jest's built-in coverage tool.

## **12. Screenshots or Demo**

* Visuals: Screenshots or a link to a demo showcasing the application’s features and design will be added here.





## **13. Known Issues**

* Bugs: Document any known bugs or issues that users should be aware of, such as:
  + Minor UI glitches on mobile devices.
  + Performance issues with large music libraries.

## **14. Future Enhancements**

* Improvements: Potential features include:
  + Adding support for different musical instruments.
  + Implementing a sharing feature for users to share their compositions.
  + Enhancing the UI with animations and transitions.