Rhythmic Tunes: Your Melodic Companion

# Introduction:

## Team Members:

ARUN V (TEAM LEADER) BOOBESH K

GOKULRAJ S

MADHAN S

MOHAMMED MANSOOR S

# Project Overview:

### Purpose:

The Melodic Companion, specifically referred to as "Melodica," is an affordable and inclusive music companion robot designed to help users practice mindfulness, overcome social disconnection, and enrich their musical experiences through music. It aims to bring joy and solace to everyday lives by using music as an approach to self-care, serving as a universal

language that connects people across cultures, age groups, and socio-economic backgrounds.

**Features:**

* Situated robotic agent with song and lyric search capabilities.
* Sound-activated colorful LED light display.
* Robotic movement with wheels and detachable arm parts.
* Voice user interface for verbal commands.
* Basic and advanced music search functions (e.g., song identification via humming, melody and lyrics search).
* Lyrics and song information display on an LCD screen.

1. Integration with smart lighting and health-related responses (e.g., volume adjustments, stress-relief music).
2. Personalized experience with accessories, atmospheric light projections, and voice modulation.
3. Functional modes such as education (learn about instruments), health (play specific music for distress), and "Follow Me" mode (robot follows user).
4. Robotic dance movements and environmental interaction with smart apps.

# Architecture

### Component Structure

App.js – Main entry point that includes routing and layout components Header – Navigation bar with links to home, topics, and user profile Song Feed – Displays aggregated song articles with filtering options

Article Card – Individual song article component with metadata Search Bar – Allows users to search articles by keywords

User Preferences – Manages user settings and personalized feeds

### State Management

The application uses Context API for global state management. User preferences, theme settings, and API data are shared across components using context providers.

### Routing

Implemented using React Router v6 with routes defined for:

/ – Home page with songs feed

/topics – Topic selection page

/profile – User preferences and saved articles

# Setup Instructions:

### Prerequisites

Node.js version 18 or above npm version 9 or above

### Installation

Clone the repository:

GIT CLONE: <https://github.com/Arun-hp-ai>

Navigate to the client directory:

cd melodic companion /client Install dependencies:

npm install

Create a .env file with necessary API keys: REACT\_APP\_SONGS\_API\_KEY=your\_api\_key\_here Start the development server:

Npm start

# Folder Structure:

client/

├── public/

├── src/

│ ├── components/

│ │ ├── ArticleCard.js

│ │ ├── Header.js

│ │ ├── NewsFeed.js

│ │ └── SearchBar.js

│ ├── context/

│ │ └── AppContext.js

│ ├── pages/

│ │ ├── Home.js

│ │ ├── Profile.js

│ │ └── Topics.js

│ ├── utils/

│ │ └── api.js

│ ├── App.js

│ ├── index.js

├── .env

├── package.json

└── README.md

## Utilities

api.js – Handles API requests to the news service provider

custom hooks – useFetchArticles for fetching news and useTheme for managing themes

# Running the Application:

Run the following command in the client directory: npm start

This will launch the application locally at <http://localhost:3000/>.

# Component Documentation:

#### Key Components

songs feed Displays songs

Accepts props: filter, search Query Article Card

Displays a single article’s title, image, and lyrics Props: article object

User Preferences

Allows users to manage their songs topics and themes Props: none

### Reusable Components

Search Bar

Handles keyword search input Props: on Search callback

Header

Navigation layout used across pages Props: none

# State Management:

#### Global State

Managed using Context API

Holds user preferences, theme data, and API responses

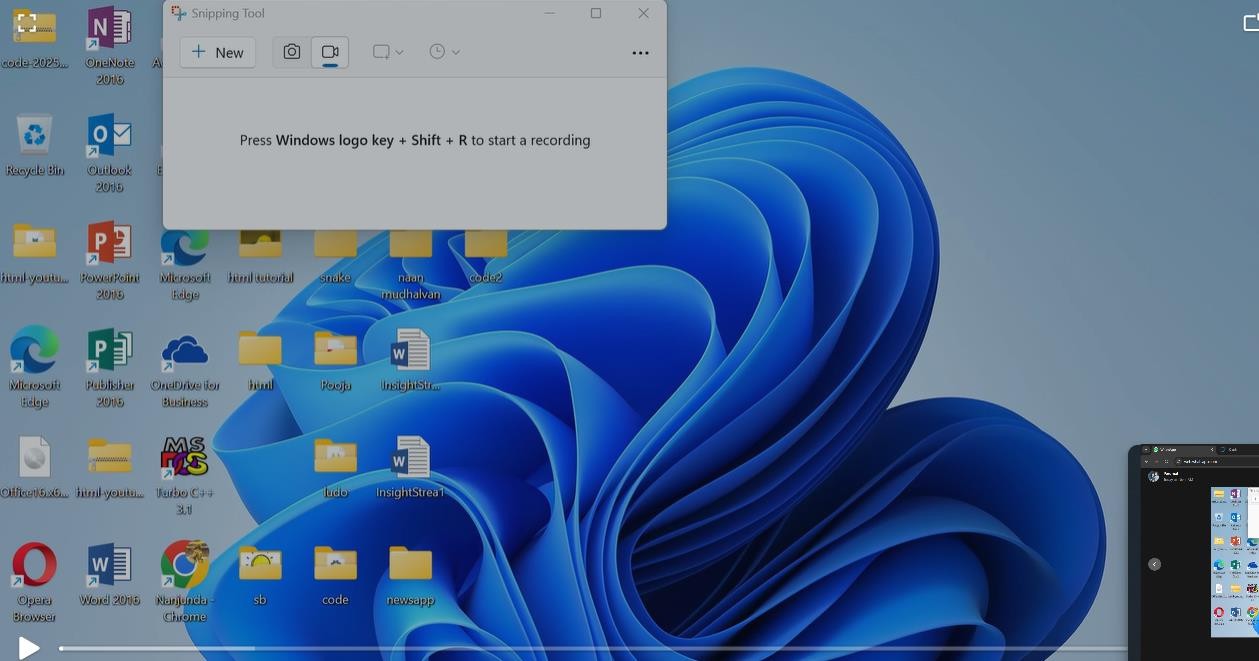
### Local State

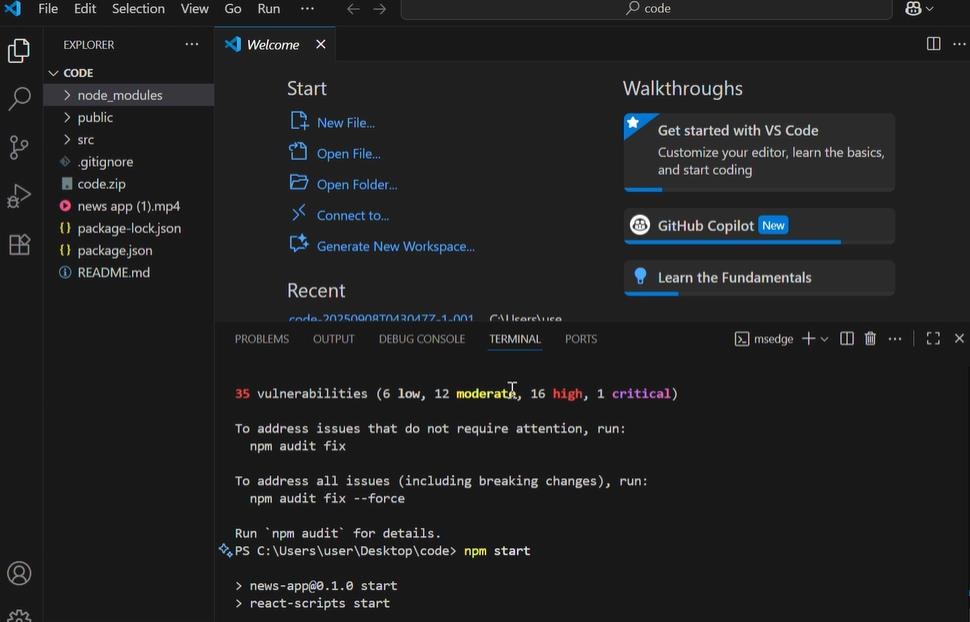
Managed using use State inside components like Search Bar for query input and Article Card for likes and comment

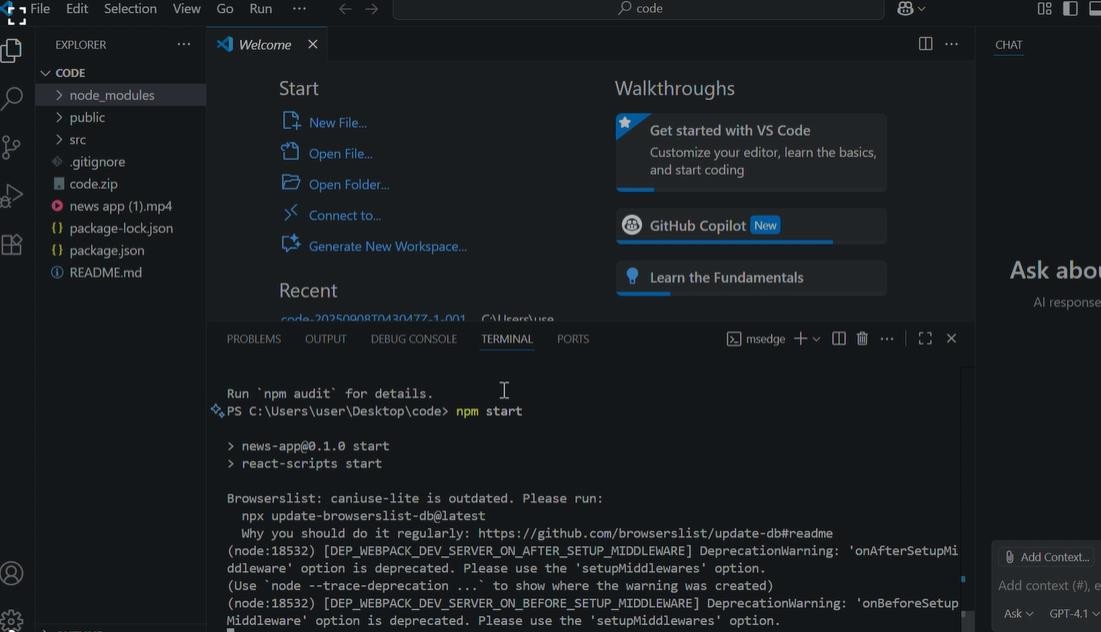
# User Interface:

### Screenshots

Home Page: Displays songs feed with filters Topics Page: Allows selection of preferred topics Profile Page: User preferences and saved articles









1. **Styling:**

### CSS Frameworks/Libraries

Uses Tailwind CSS for layout and styling Custom styles for buttons, cards, and inputs

### theming

Supports light and dark themes

Uses CSS variables and Context API for theme switching

# Testing:

#### Testing Strategy

Unit tests with Jest

Integration tests using React Testing Library

Mock API responses for end-to-end tests

### Code Coverage

Ensured via Jest’s coverage reports

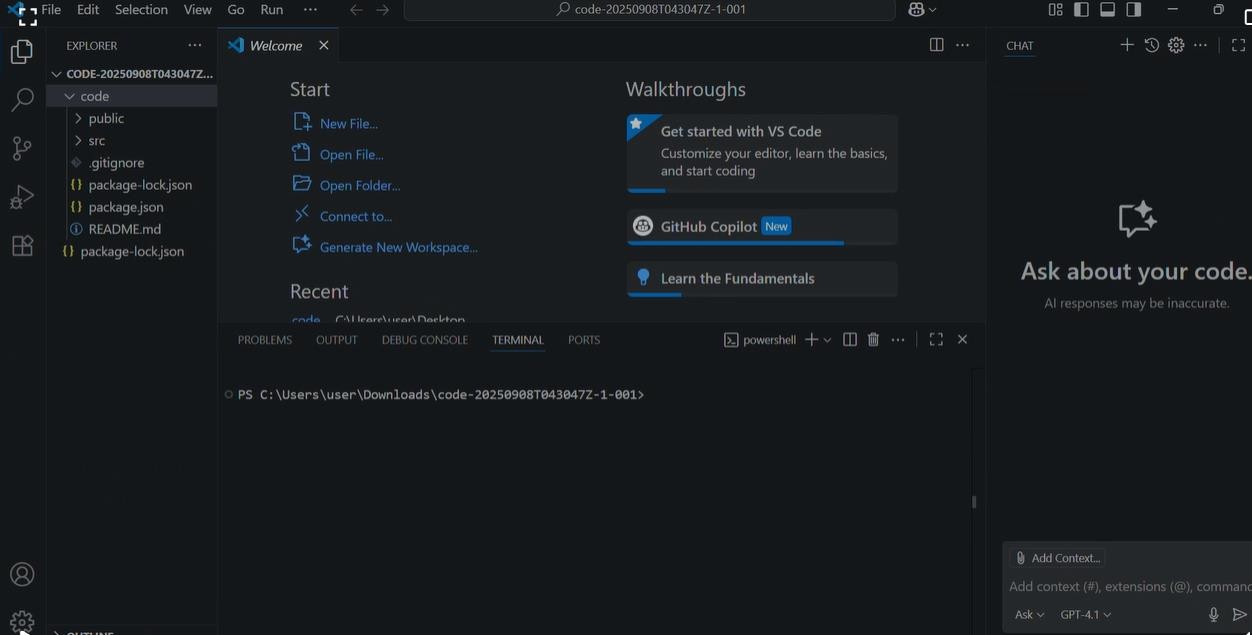
Focus on critical components like Songs Feed and Search Bar

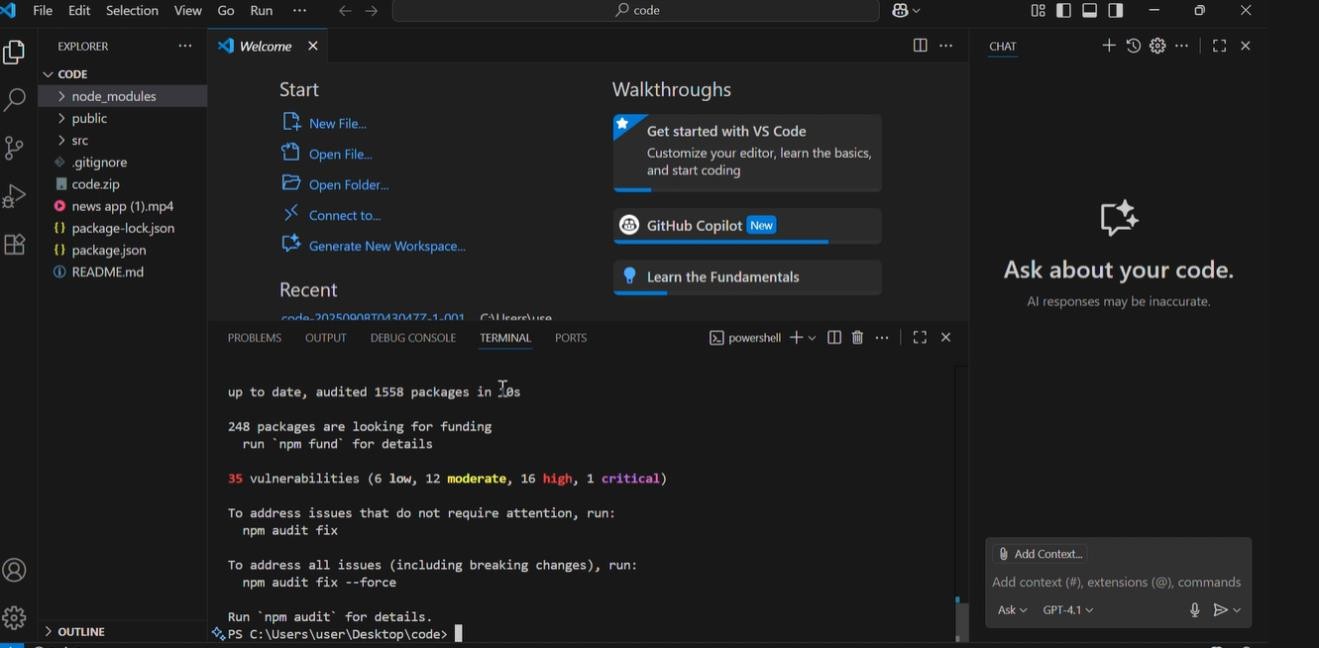
# Screenshots or Demo:

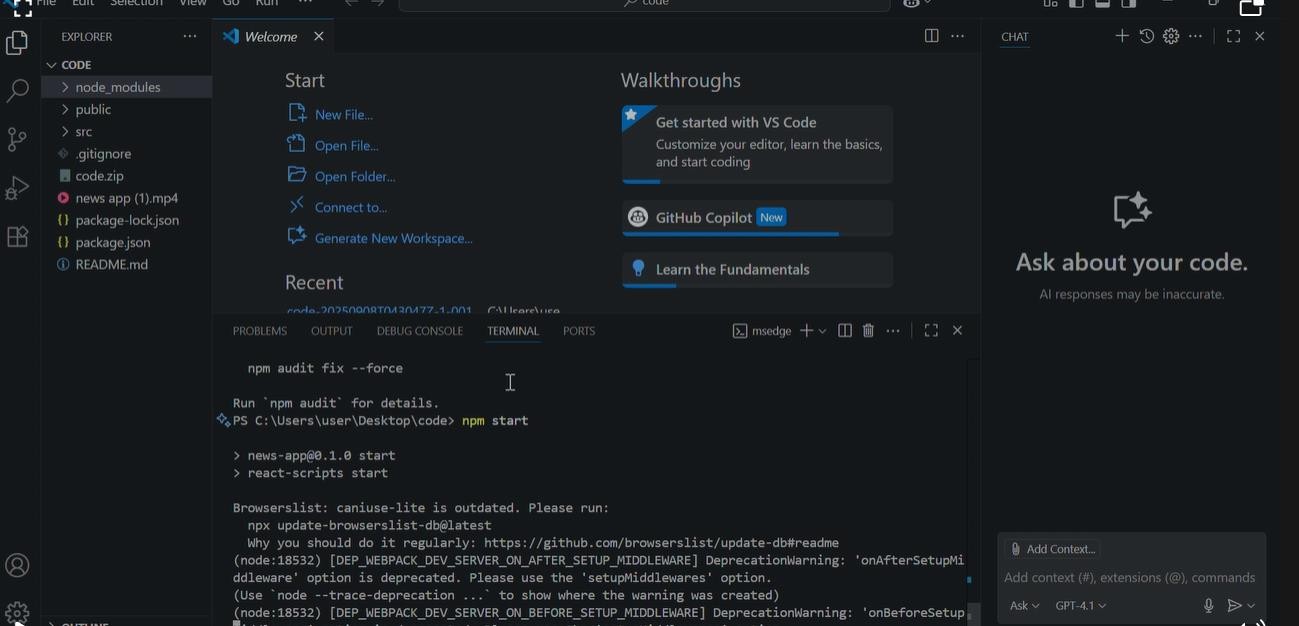
You can view the live demo at:

https://youtu.be/P9hLaSWuvQw?si=xY6PfFDsk1r3X7HO

<https://drive.google.com/drive/folders/1E9QxUsJfs_W1pTK3qt8lB_yWaYAecCWd?usp=drive_link>







# Known Issues:

Occasionally slow API responses due to third-party rate limits Mobile responsiveness needs minor tweaks on smaller devices Some saved preferences may not persist after browser refresh

# Future Enhancements:

Implement advanced filtering by date, location, and sentiment Add user authentication and article bookmarking

Introduce animations and transitions for smoother interactions Optimize API calls and caching for better performance