Rhythmic Tunes: Your Melodic Companion

|  |  |  |
| --- | --- | --- |
| **TEAM** | **NAME** | **EMAIL** |
| **Team leader** | **Manasa devi N** | **ncas2225sz1028@ncas.in** |
| **Team member1** | **Angel I** | **ncas2225sz1058@ncas** |
| **Team member2** | **Deepika V** | **ncas2225sz1432@ncas.in** |
| **Team member3** | **Devi C** | **ncas2225sz1180@ncas.in** |
| **Team member4** | **Gayathri M** | **ncas2225sz1695@ncas.in** |

**Team ID SWTID1741324396155224**

**Introduction:**

Welcome to Rhythmic Tunes: Your Melodic Companion, a revolutionary music streaming service designed to harmonize your life with the perfect melody. Imagine a world where music seamlessly adapts to your emotions, activities, and surroundings. Rhythmic Tunes makes this vision a reality. Problem Statement: Traditional music streaming services often rely on static playlists and algorithms that fail to capture the nuances of human emotion. This can lead to a disjointed listening experience, where the music doesn't quite match the mood or activity. Solution: Rhythmic Tunes addresses this challenge by incorporating advanced AI-powered algorithms that learn your listening habits, preferences, and emotional state. Our platform creates personalized playlists that dynamically adapt to your needs, ensuring a melodic companion that's always in tune with your life.

**Project overview**

**Benefits**

1. Enhanced Music Experience: Rhythmic Tunes provides a unique music experience that adapts to the user's emotions and activities.

2. Music Discovery: The platform helps users discover new music, artists, and genres.

3. Convenience: Rhythmic Tunes is easily accessible across multiple devices.

4. Social Connection: Users can share their music experiences and connect with like-minded individuals.

**Target Audience:**

1. Music Lovers: Individuals who enjoy listening to music and discovering new artists.

2. Fitness Enthusiasts: People who use music to enhance their workout experiences.

3. Stress Relief: Individuals seeking relaxing music to reduce stress and anxiety.

**Technical Requirements:**

1. Music Library: A vast library of licensed songs, albums, and playlists.

2. AI-Powered Algorithm\*: A sophisticated algorithm that analyzes user behavior and preferences.

3. Scalable Infrastructure\*: A robust infrastructure that supports high traffic and seamless playback.

**Design Requirements:**

1. User-Friendly Interface: An intuitive and visually appealing interface that simplifies music discovery.

2. Personalized Experience: A customized experience that reflects the user's preferences and listening habits.

**Architecture**

**Overview**:

Rhythmic Tunes is a music streaming service that utilizes AI-powered algorithms to provide personalized music recommendations. The architecture will consist of the following components:

**1. Frontend:**

Web Application: Built using React, Redux, and CSS

Mobile Application: Built using React Native for iOS and Android

API Gateway: Handles incoming requests from clients, routes them to appropriate services

**2. Backend:**

User Service: Handles user authentication, profile management, and preferences Music Service: Responsible for music metadata management, recommendation algorithms, and playlist generation

Search Service: Provides search functionality for music, artists, and playlists

Notification Service: Handles notifications for new music releases, recommendations, and social interactions

**3. Data Storage:**

Relational Database (MySQL): Stores user data, music metadata, and playlist information

NoSQL Database (MongoDB): Stores music features, user listening history, and recommendation models

Object Storage (Amazon S3): Stores music files, album artwork, and user-uploaded content

**4. AI/ML Components:**

Recommendation Engine: Built using TensorFlow, PyTorch, or Scikit-learn, this engine generates personalized music recommendations based on user listening history, preferences, and music features

Music Feature Extraction: Utilizes libraries like Librosa, Essentia, or Music21 to extract features from music files, such as genre, mood, tempo, and instrumentation

**5. APIs and Integrations:**

Music APIs: Integrates with music streaming APIs like Spotify, Apple Music, or Google Play Music to access music metadata and streaming functionality

**6. Deployment:**

Cloud Provider: Deploys on a cloud provider like AWS, Google Cloud, or Microsoft Azure for scalability, reliability, and cost-effectiveness

Containerization: Utilizes containerization using Docker to ensure consistent and reliable deployment across environments

Orchestration: Employs orchestration tools like Kubernetes to manage and automate container deployment, scaling, and management

**Testing Plan**

**Unit Testing**:

**1. User Service:**

Test user registration and login functionality.

Verify user data is stored correctly in the database.

**2. Music Service:**

Test music metadata retrieval and storage.

Verify music recommendation algorithms.

**3. Search Service:**

Test search functionality for music, artists, and playlists.

**Integration Testing:**

**1. User-Music Interaction:**

Test user playback, pause, and skip functionality.

Verify music playback history is updated correctly.

**2. Search-Music Integration:**

Test search results for music, artists, and playlists.

Verify search results are relevant and accurate.

**3. User-Playlist Interaction:**

Test user playlist creation, editing, and deletion.

Verify playlist changes are reflected correctly.

**Code coverage**

**Source Code:**

**public/index.html:**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>RhythmicTunes</title>**

**<link rel="stylesheet" href="styles.css">**

**</head>**

**<body>**

**<div id="root"></div>**

**<script src="bundle.js"></script>**

**</body>**

**</html>**

**UI Testing:**

**1. User Interface:**

Test UI components, such as buttons, forms, and navigation.

Verify UI is responsive and works correctly on different devices.

**2. User Experience:**

Test user workflow, such as searching, playing, and creating playlists.

Verify user experience is smooth and intuitive.

**Integration Testing:**

**API Integration:**

**1. Music API:**

Test integration with music streaming APIs.

Verify music metadata and streaming functionality.

**2. Social Media API:**

Test integration with social media platforms.

Verify social sharing and login functionality.

**Database Integration:**

**1. User Data:**

Test user data storage and retrieval.

Verify user data is consistent across the application.

**2. Music Data:**

Test music metadata storage and retrieval.

Verify music data is consistent across the application.

**Deployment Integration:**

**1. Cloud Deployment:**

Test deployment on cloud providers.

Verify application scalability and reliability.

**2. Containerization:**

Test containerization using Docker.

Verify application consistency across environments.

**3. Tools and Frameworks:**

1. Jest: For unit testing and integration testing.

2. Cypress: For UI testing and end-to-end testing.

3. Postman: For API testing and integration testing.

4. Docker: For containerization and deployment.

**4. Testing Schedule:**

1. Unit Testing: 2 weeks

2. Integration Testing: 3 weeks

3. UI Testing: 2 weeks

4. Deployment Testing: 1 week

**Setup Instructions:**

1. Clone the repository: git clone https://github.com/your-username/RhythmicTunes.git

2. Install dependencies: npm install or yarn install

3. Create a new file named .env in the root directory and add your API keys and other environment variables.

4. Run the application: npm start or yarn start

**Folder Structure:**

**RhythmicTunes/**

**├── public/**

**│ ├── index.html**

**│ └── favicon.ico**

**├── src/**

**│ ├── components/**

**│ │ ├── App.js**

**│ │ ├── Header.js**

**│ │ ├── Footer.js**

**│ │ ├── MusicPlayer.js**

**│ │ └── ...**

**│ ├── containers/**

**│ │ ├── AppContainer.js**

**│ │ └── ...**

**│ ├── actions/**

**│ │ ├── musicActions.js**

**│ │ └── ...**

**│ ├── reducers/**

**│ │ ├── musicReducer.js**

**│ │ └── ...**

**│ ├── utils/**

**│ │ ├── api.js**

**│ │ └── ...**

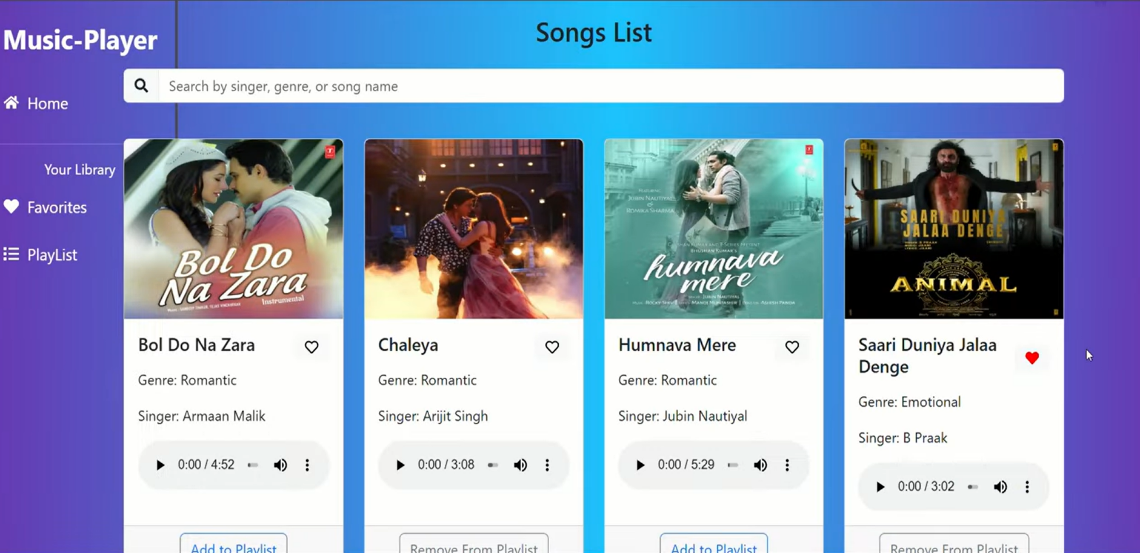
**│ ├── index.js**

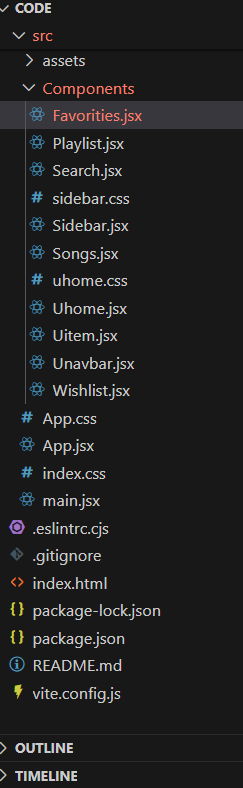
**│ └── store.js**

**├── package.json**

**└── README.md**

**Screenshots or Demo**





**Running the Application:**

1. Start the development server: npm start or yarn start

2. Open your web browser: Navigate to http://localhost:3000

3. Test the application: Verify that the application is working as expected

**Component Documentation:**

**App.js:**

Description: The main application component.

Props: None

State: music Player Open, current Song

Methods: handle Music Player Open, handle Current Song Change

**MusicPlayer.js:**

Description: The music player component.

Props: current Song, music Player Open

State: None

Methods: handle Play Pause, handle Next Song, handle Previous Song

**State Management:**

Rhythmic Tunes uses Redux for state management.

**Store:**

The store is created in store.js and is composed of multiple reducers.

**Reducers:**

music Reducer: Handles music-related state changes.

user Reducer: Handles user-related state changes.

**Actions:**

music Actions: Contains action creators for music-related actions.

user Actions: Contains action creators for user-related actions.

**Styling:**

Rhythmic Tunes features a modern and sleek design, with a focus on usability and accessibility. The application uses a combination of CSS and JavaScript to create a responsive and interactive interface.

**Color Scheme:**

Primary color: #4CAF50 (a bright and calming green)

Secondary color: #FF9800 (a vibrant and energetic orange)

Background color: #F7F7F7 (a neutral and clean gray)

**Typography:**

Font family: Open Sans

Font sizes: 16px (body), 24px (headings), 48px (titles)

**Known Issues:**

Here are some known issues with the application:

The music player may not work correctly in older browsers.

The search functionality may not return accurate results.

The application may not be fully accessible for users with disabilities.

**Future Enhancements:**

Here are some future enhancements planned for the application:

Implementing a recommendation algorithm to suggest music based on user listening habits.

Adding support for multiple music streaming services.

Improving the application's accessibility features ,Adding support for offline playback.