***Project Documentation***

***Sonic Beats***

1. ***INDRODUCTION***

***PROJECT TITLE****: Sonic Beats*

* ***TEAM ID:*** *NM2025TMID38578*
* ***TEAM LEADER:*** *Kaviya G [Coder]*

***MAIL ID:***[*kaviyamalliga2007@gmail.com*](mailto:kaviyamalliga2007@gmail.com)

* ***TEAM MEMBERS :***

***NAME:*** *Keerthana M [Coder]*

***MAIL ID:***[*keerthanamoorthy20072006@gmail.com*](mailto:keerthanamoorthy20072006@gmail.com)

***NAME:*** *Mary Francina J [Documentation]*

***MAIL ID:****[maryfrancina2005@gmail.com](mailto:maryfrancina2005@gmail.com)*

***NAME:*** *Mithra S [Documentation]*

***MAIL* *ID:*** [*mithra.sundaresan.r@gmail.com*](mailto:mithra.sundaresan.r@gmail.com)

***2. PROJECT OVERVIEW***

* ***PURPOSE:***

*The primary goal of music streaming is to provide a seamless platform for*

*Music enthusiasts ,enjoying ,and sharing diverse musical experiences.*

* ***FEATURES:***
* ***User-Friendly Interface****:*

*Develop an intuitive interface that allows users to effortlessly explore, save, and share their favourite music tracks and playlists.*

* ***Comprehensive Music Streaming****:*

*Provide robust features for organizing and managing music content, including advanced search options for easy discovery.*

* ***Morden 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.*

***3. ARCHITECTURE***

* ***FRONTEND:***

*The frontend is the part of a website or application that uses interact with directly . It includes everything visible in the browser –like layout , design , and user interface—typically built with HTML,CSS and Java script ,react js,node.js and boots trap.*

* ***BACKEND:***

*The backend is the server-side part of a website or application that handles business logic, database interactions ,and user authentication .It processes requests from the frontend and sends back the appropriate data,react,node.js*

* ***DATABASE:***

*A data base is an organised collection of data store electronically for easy retrieve ,insert ,and manipulate information ,Mongo DB stores user data , project information ,applications and chat messages.*

***4. SETUP INSTRUCTION***

* ***PREREQUISITES:***
* *--Html*
* *--CSS*
* *--Java script*
* *–Mongo DB*
* *--node.js*
* *–Git*
* *–React.js*
* *Vite*
* *Express.js –Mongoose-Visual Studio Code*
* ***INSTALLATION STEPS:***

*# Clone the repository git clone*

*# Install client dependencies cd*

*#client npm install*

*# Install server dependencies cd*

*server npm install*

***5. PROJECT STRUCTURE***

*Rhythmic Tunes/*

*│*

*├── public/*

*│ ├── index.html ← Main HTML file*

*│ └── assets/ ← Images, icons, audio files*

*│*

*├──src/*

*│ ├── assets/ ← Static files like album art, sounds*

*│ ├── components/ ← Reusable React components*

*│ │ ├── Header*

*│ │ ├── Footer*

*│ │*

*│ │ ├── Home*

*│ │ └── Login*

*│ │*

*│*

*├── db/*

***6. RUNNING THE APPLICATION***

* ***FRONTEND:***

*The frontend is like the face of a website—what you see and click when you visit a page.*

* ***BACKEND****:*

*The backend is like the brain and engine of a website—it manages data, Processes logic, and responds to what the frontend (user interface) requests.*

* ***ACCESS****: Visit* [*http://localhost:5174*](http://localhost:5174)

***7. API DOCUMENTATION***

***Sign Up***

*import { useState } from 'react';*

*export default function SignUp() {*

*const [email, setEmail] = useState('');*

*const [password, setPassword] = useState('');*

*const [confirmPassword, setConfirmPassword] = useState('');*

*const [error, setError] = useState('');*

*const handleSubmit = (e) => {*

*e.preventDefault();*

*if (password !== confirmPassword) {*

*setError("Passwords don't match");*

*return;*

*}*

*setError('');*

*// Handle sign-up logic here (e.g., API call)*

*alert(Signing up with ${email});*

*};*

*return (*

*<div className="max-w-md mx-auto mt-20 p-6 border rounded shadow-lg">*

*<h2 className="text-2xl font-bold mb-6 text-center">Sign Up</h2>*

*<form onSubmit={handleSubmit} className="space-y-4">*

*<label className="block">*

*<span className="text-gray-700">Email</span>*

*<input*

*type="email"*

*required*

*className="mt-1 block w-full rounded border-gray-300 shadow-sm focus:ring focus:ring-indigo-300"*

*value={email}*

*onChange={(e) => setEmail(e.target.value)}*

*placeholder="you@example.com"*

*/>*

*</label>*

*<label className="block">*

*<span className="text-gray-700">Password</span>*

*<input*

*type="password"*

*required*

*className="mt-1 block w-full rounded border-gray-300 shadow-sm focus:ring focus:ring-indigo-300"*

*value={password}*

*onChange={(e) => setPassword(e.target.value)}*

*placeholder="Enter your password"*

*/>*

*</label>*

*<label className="block">*

*<span className="text-gray-700">Confirm Password</span>*

*<input*

*type="password"*

*required*

*className="mt-1 block w-full rounded border-gray-300 shadow-sm focus:ring focus:ring-indigo-300"*

*value={confirmPassword}*

*onChange={(e) => setConfirmPassword(e.target.value)}*

*placeholder="Confirm your password"*

*/>*

*</label>*

*{error && <p className="text-red-600">{error}</p>}*

*<button*

*type="submit"*

*className="w-full bg-green-600 text-white py-2 rounded hover:bg-green-700 transition"*

*>*

*Sign Up*

*</button>*

*</form>*

*</div>*

*);*

*}*

***Sign In***

*import{useState} from 'react';*

*export default function SignIn() {*

*const [email, setEmail] = useState('');*

*const [password, setPassword] = useState('');*

*const handleSubmit = (e) => {*

*e.preventDefault();*

*// Handle sign-in logic here (e.g., API call)*

*alert(Signing in with ${email});*

*};*

*return (*

*<div className="max-w-md mx-auto mt-20 p-6 border rounded shadow-lg">*

*<h2 className="text-2xl font-bold mb-6 text-center">Sign In</h2>*

*<form onSubmit={handleSubmit} className="space-y-4">*

*<label className="block">*

*<span className="text-gray-700">Email</span>*

*<input*

*type="email"*

*required*

*className="mt-1 block w-full rounded border-gray-300 shadow-sm focus:ring focus:ring-indigo-300"*

*value={email}*

*onChange={(e) => setEmail(e.target.value)}*

*placeholder="you@example.com"*

*/>*

*</label>*

*<label className="block">*

*<span className="text-gray-700">Password</span>*

*<input*

*type="password"*

*required*

*className="mt-1 block w-full rounded border-gray-300 shadow-sm focus:ring focus:ring-indigo-300"*

*value={password}*

*onChange={(e) => setPassword(e.target.value)}*

*placeholder="Enter your password"*

*/>*

*</label>*

*<button*

*type="submit"*

*className="w-full bg-indigo-600 text-white py-2 rounded hover:bg-indigo-700 transition"*

*>*

*Sign In*

*</button>*

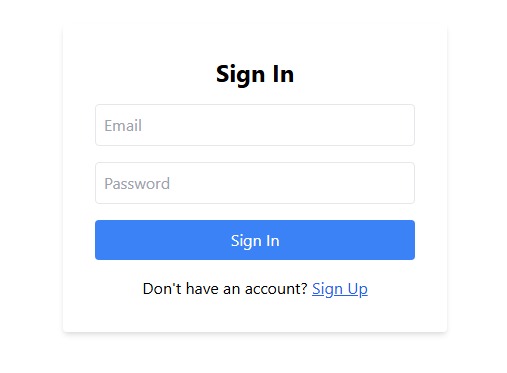
*</form>*

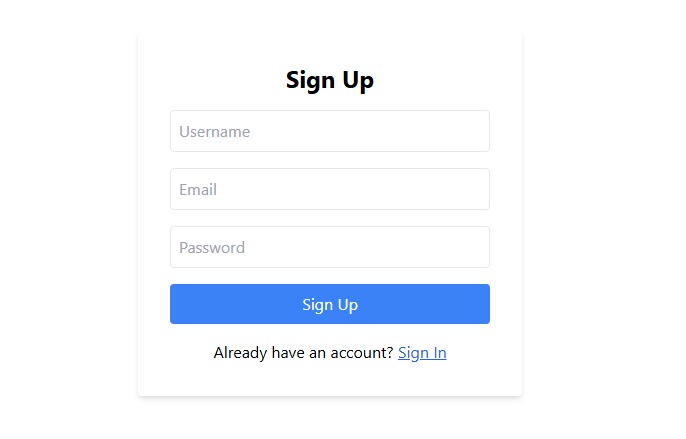
*</div>*

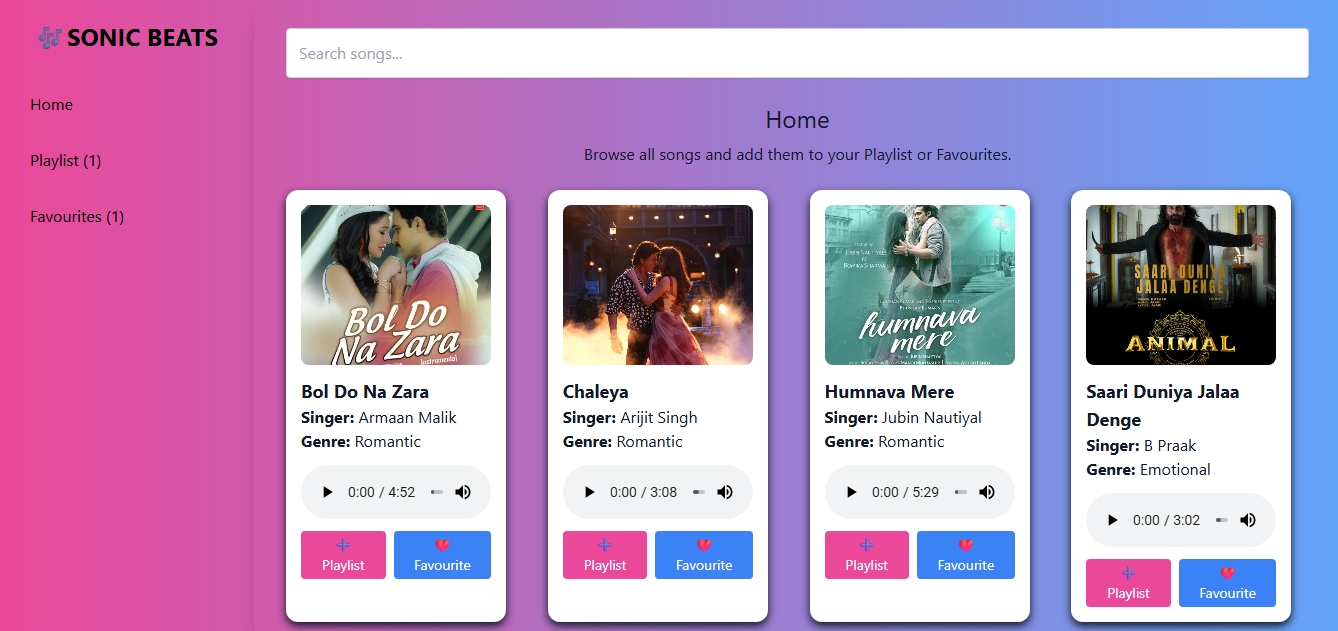
*);*

*}*

***8. USER-INTERFACE***

******

******

******

***9. FUTURE ENHANCEMENT***

* ***AI –BASED PERSONALIZED RHYTHM GENERATION***

This concept involves using AI to analyse an individual's musical preferences, physical movements, or emotional states to create personalized rhythmic patterns. It could adapt to a user’s mood, style, and even physiological signals, making rhythm a deeply personalized element of music creation.

* ***INTERACTIVE RHYTHM GAMES***

Interactive rhythm games involve players interacting with rhythm-based challenges, where their performance directly influences the game’s progression. With advancements in technology, future rhythm games could incorporate real-time feedback, motion-sensing, and AI, taking music games to a new level.

* ***COLLABORATIVE MUSIC CREATION PLATFORM***

This platform would allow musicians, producers, and enthusiasts worldwide to collaborate in creating music, specifically focusing on rhythm. With an emphasis on shared creative workflows and real-time collaboration, this platform would facilitate seamless interaction, no matter the user's location.

* ***GLOBAL RHYTHM TREANDS VISUALIZATION***

This concept revolves around analysing global rhythm patterns and visualizing them to identify trends, cultural patterns, and emerging rhythms across different genres and countries. With advanced data analytics and AI, this system could create visual representations of rhythm patterns from the global music industry.

* ***DYNAMIC LYRICS SYNCHRONIZATION***

Dynamic lyrics synchronization involves syncing the rhythm with the lyrics of a song in a more adaptive, real-time manner. This could extend to live performances or digital platforms where the rhythm and lyrics shift seamlessly to match each other’s dynamics. It would allow the lyrics to flow naturally with the rhythm, enhancing the music’s emotional depth.