HARMONIX - APP DEVELOPMENT PROJECT

INTRODUCTION

Vikash P
22f3001758@ds.study.iitm.ac.in
I am a second-year student studying Computer Science Engineering at SRM Institute of Science and Technology (Ramapuram) and currently at the diploma level in IITM BS degree. I have titled this project as **Harmonix.**

DESCRIPTION

The primary goal of this project is to develop a music-streaming web application using Flask, a lightweight Python web framework. The application aims to provide an efficient platform for users to listen to songs created by various artists and create personalized playlists. The system will have three main user roles: Admin, Users, and Creators. The Admin will have privileges to view overall statistics and performance metrics, including song performances and member counts (users and creators). Users will be able to log in, search for songs and creators, add songs to their playlists, and play songs along with their lyrics. Creators will have the ability to add new songs/lyrics, edit existing song details, and delete songs as needed.

TECHNOLOGIES USED

- *HTML:* To build the front-end framework.
- *CSS:* To add styling components to the front-end code.
- *JS:* To define the functionality of the website.
- *FLASK:* Creating a connection between the front end and back end. Used to create database table structures to store values.
- *JINJA2:* Used to get Python-based website templates.
- *BOOTSTRAP:* Source to get prebuilt components required for the website building
- *SQLITE:* Database used to store and get values in the table. This database engine is less space-consuming and fast in execution.

DB SCHEMA DESIGN

https://drive.google.com/file/d/1QJ0GzoWXlhyk5AsFlBs4ji3R2GVVB3iV/view?usp=sharing

API DESIGN

The Music Streaming App's API serves as a robust foundation, providing secure access for Admin, User, and Creator logins. It focuses on Song Management, enabling CRUD operations to efficiently handle songs, including details such as creator, genre, and lyrics. Users can explore, play, and manage songs seamlessly. The API enhances the overall music streaming experience by simplifying song-related operations and ensuring a user-friendly interface within the application.

ARCHITECTURE AND FEATURES

1. User Authentication: The app supports multi-user login with distinct roles for Admin, User, and Creator.
2. User Functionality: Users can play/pause songs, adjust volume settings, read lyrics, and create and manage personalized playlists, tailoring their music experience.
3. Creator Functionality: Creators possess the ability to contribute to the platform by adding new songs, including lyrics and associated details. They also have the option to edit song lyrics, allowing for continuous refinement of content.
4. Admin Functionality: Administrators gain access to comprehensive app statistics, providing insights into top-performing songs, creators, and other relevant metrics.
5. Additional Features: The app incorporates a robust search functionality, enabling users to discover songs and creators based on various criteria easily. Furthermore, the automatic display of new songs ensures a constantly refreshed and dynamic content experience.

In summary, the Music Streaming App strives to deliver a seamless and user-friendly music experience. Using Flask and diverse technologies, the application effectively handles multi-user authentication, song categorization, and product data management. The project's well-organized structure, adherence to MVC architecture, and careful implementation of features collectively contribute to the creation of a practical and user-centric music streaming solution.

https://drive.google.com/file/d/1avgZZ0y7jE68Lhy4AXzYfJDtqv71SdiB/view?usp=sharing