



edunet
foundation



NEXT GEN EMPLOYABILITY PROGRAM

Creating a future-ready workforce

Team Members

Student Name : Swetha G
Student ID :311121205056

College Name

Loyola Icam College of
Engineering and Technology

CAPSTONE PROJECT SHOWCASE

Project Title

Notes Sharing Web Application using Django Framework

Abstract | Problem Statement | Project Overview | Proposed Solution |
Technology Used | Modelling & Results | Conclusion



Abstract

This project focuses on developing a music application using Django, leveraging its capabilities for efficient backend development and seamless frontend integration. The application will provide user authentication, music catalog management, playlist creation, and audio streaming features. Django's ORM will handle database interactions, ensuring data integrity and scalability. The frontend will be built using Django's templating system for dynamic content rendering and JavaScript for interactive features like audio playback controls. The application will support search functionality for songs, albums, and artists. User data security and privacy will be prioritized through Django's authentication system and custom permissions. The goal is to create a user-friendly and responsive music platform that allows users to discover, organize, and enjoy their favorite music effortlessly.

Problem Statement

Designing an efficient music application using Django poses challenges such as implementing robust user authentication, managing complex music metadata, enabling seamless audio streaming, and ensuring a responsive frontend interface. The project aims to address these challenges by developing a scalable and user-friendly platform that simplifies music discovery and organization. Key objectives include optimizing database performance for large music catalogs, integrating secure audio streaming capabilities, and delivering a visually appealing and intuitive user experience. The solution will focus on leveraging Django's features effectively to overcome these hurdles and provide a compelling music application that meets modern user expectations.

Project Overview

This project involves building a music application using Django, integrating user authentication, music catalog management, playlist creation, and audio streaming functionalities. The application will leverage Django's ORM for database operations, ensuring data integrity and scalability. Frontend development will utilize Django's templating system and JavaScript for interactive features. Key goals include delivering a responsive and visually appealing interface for users to explore and enjoy music seamlessly. Emphasis will be placed on optimizing performance, security, and user experience throughout the development process.

Proposed Solution

User Authentication and Authorization:

Implement secure user registration and login functionality using Django's authentication system. Ensure proper authorization levels for users and administrators to control access to features like song management.

Song Catalog and Management:

Develop a robust backend system to manage a catalog of songs, albums, and artists. Utilize Django models to define database schemas for storing music metadata and implement CRUD (Create, Read, Update, Delete) operations for song management.

Admin Dashboard for Song Administration:

Create an intuitive admin dashboard accessible only to authorized users (admins). This dashboard allows admins to add new songs, update existing songs (including metadata like title, artist, and genre), and delete songs as needed.

User-Facing Home Page with Song Playback:

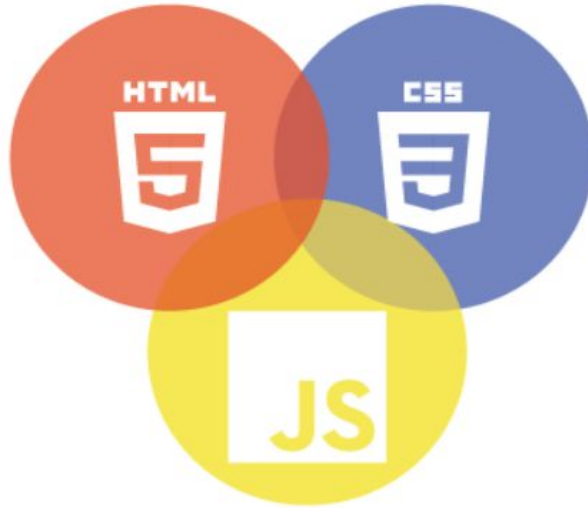
Design a user-friendly home page where authenticated users can browse through available songs and play them directly on the website. Implement audio streaming functionality to deliver a seamless listening experience.

Responsive Frontend Design:

Develop responsive frontend views using Django templates, HTML, CSS, and JavaScript to ensure a visually appealing and intuitive interface. Focus on usability and responsiveness across different devices and screen sizes to enhance user experience.

Technology Used

Front-end



Back-end

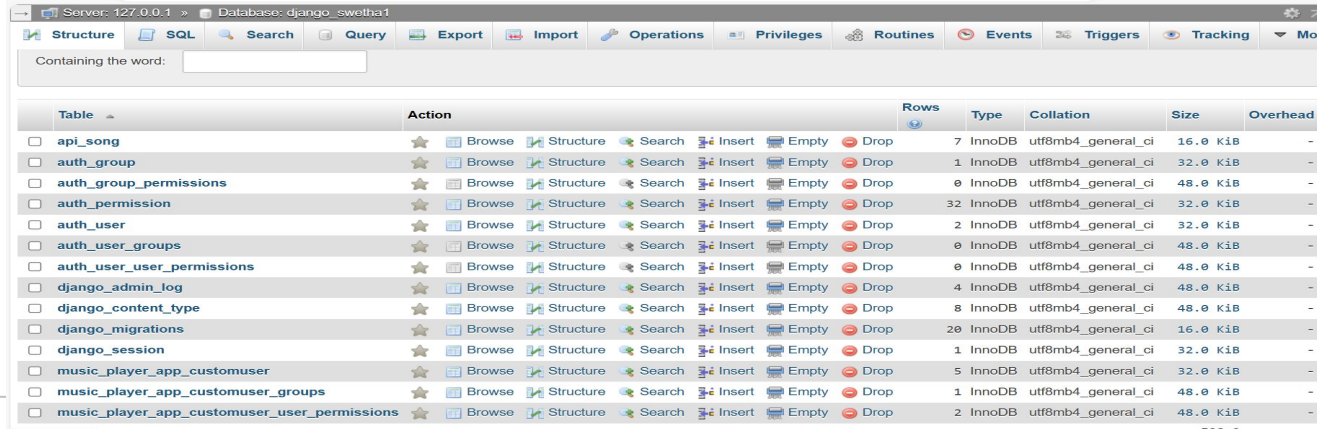


Modelling & Results

```
PS C:\Users\swethag\New folder\music_player_project> type api\models.py
# api\models.py
from django.db import models

class Song(models.Model):
    id = models.AutoField(primary_key=True)
    categorie = models.CharField(max_length=100, null=True, default=None)
    artist = models.CharField(max_length=100)
    audio_file = models.FileField(upload_to='audio_files/')
    audio_img = models.FileField(upload_to='audio_images/')

    def __str__(self):
        return self.title
PS C:\Users\swethag\New folder\music_player_project> |
```



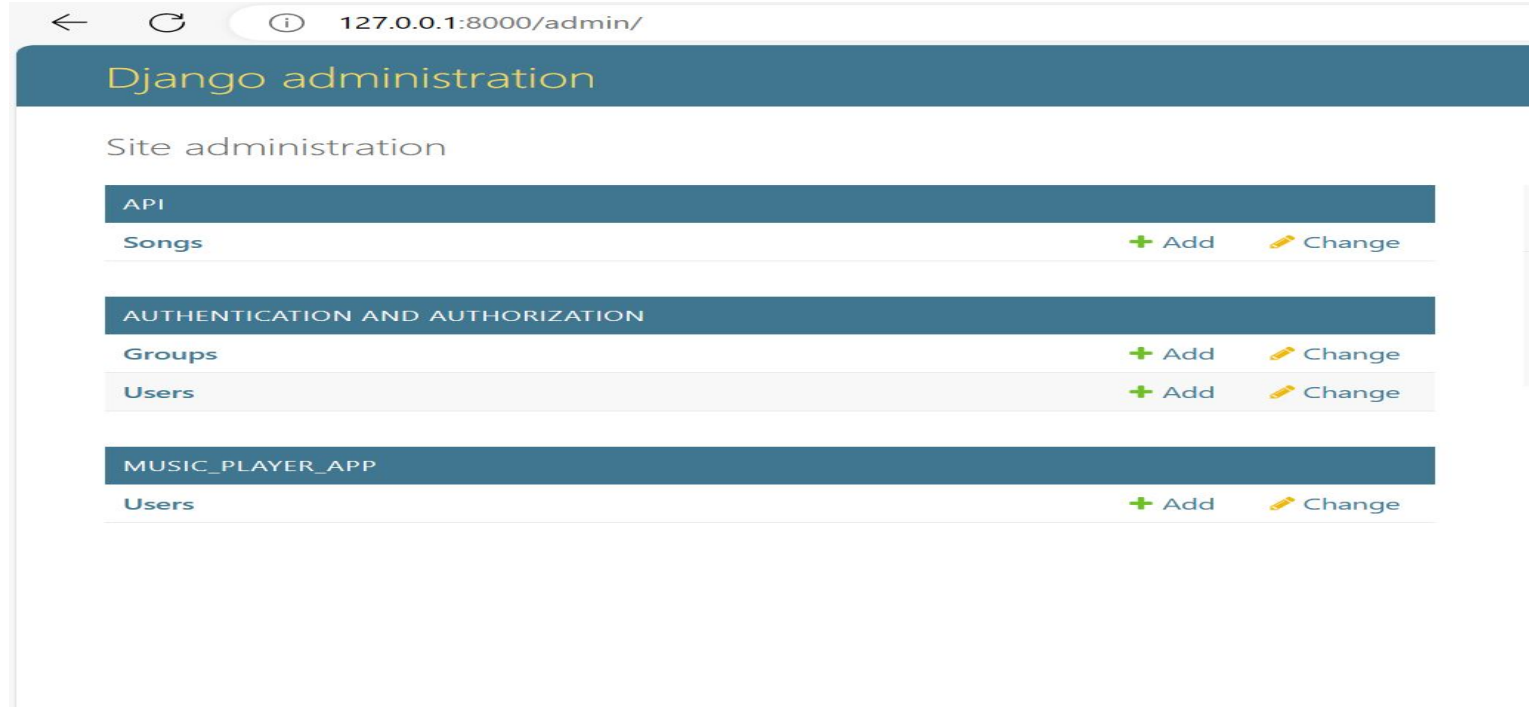
Server: 127.0.0.1 » Database: django_swetha1

Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Tracking More

Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> api_song	★ Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> auth_group	★ Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> auth_group_permissions	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> auth_permission	★ Browse Structure Search Insert Empty Drop	32	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> auth_user	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> auth_user_groups	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> auth_user_user_permissions	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> django_admin_log	★ Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> django_content_type	★ Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> django_migrations	★ Browse Structure Search Insert Empty Drop	20	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> django_session	★ Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> music_player_app_customuser	★ Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> music_player_app_customuser_groups	★ Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/> music_player_app_customuser_user_permissions	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	48.0 KiB	-


Modelling & Results



Homepage

← ↻ ⓘ 127.0.0.1:8000/home 🔊 ☆ 📄 ☆ 🗑️ 🔒 ...

YOUR TOP MIXES

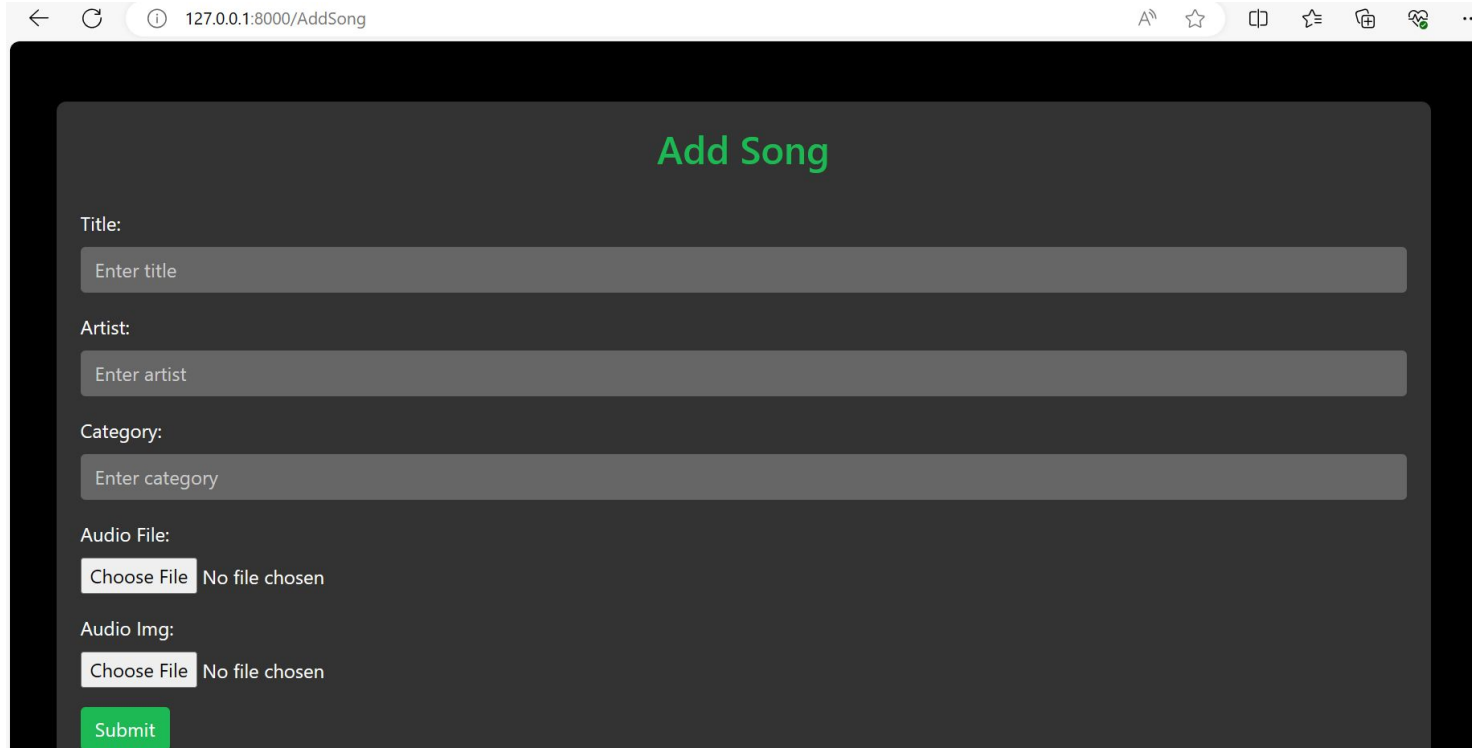


0:13 / 5:19 🔊 ⋮

Search Song:

Marudanni - Asha-Love	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
Marudanni - N R. Raohunathan-Love	<input type="button" value="Update"/>	<input type="button" value="Delete"/>

AddSong Page



A screenshot of a web browser displaying the 'AddSong' page. The browser's address bar shows '127.0.0.1:8000/AddSong'. The page has a dark theme with a black background. A large, semi-transparent dark gray rectangle contains the form. At the top of this rectangle, the text 'Add Song' is written in a bright green font. Below this, there are five input fields, each with a label to its left: 'Title:', 'Artist:', 'Category:', 'Audio File:', and 'Audio Img:'. Each input field is a light gray rectangle with its respective label text inside. The 'Audio File:' and 'Audio Img:' fields also have a 'Choose File' button and the text 'No file chosen' to their right. At the bottom left of the form rectangle is a green 'Submit' button.

← ↻ ⓘ 127.0.0.1:8000/AddSong 🔊 ☆ 📄 📌 🗑️ ⋮

Add Song

Title:
Enter title

Artist:
Enter artist

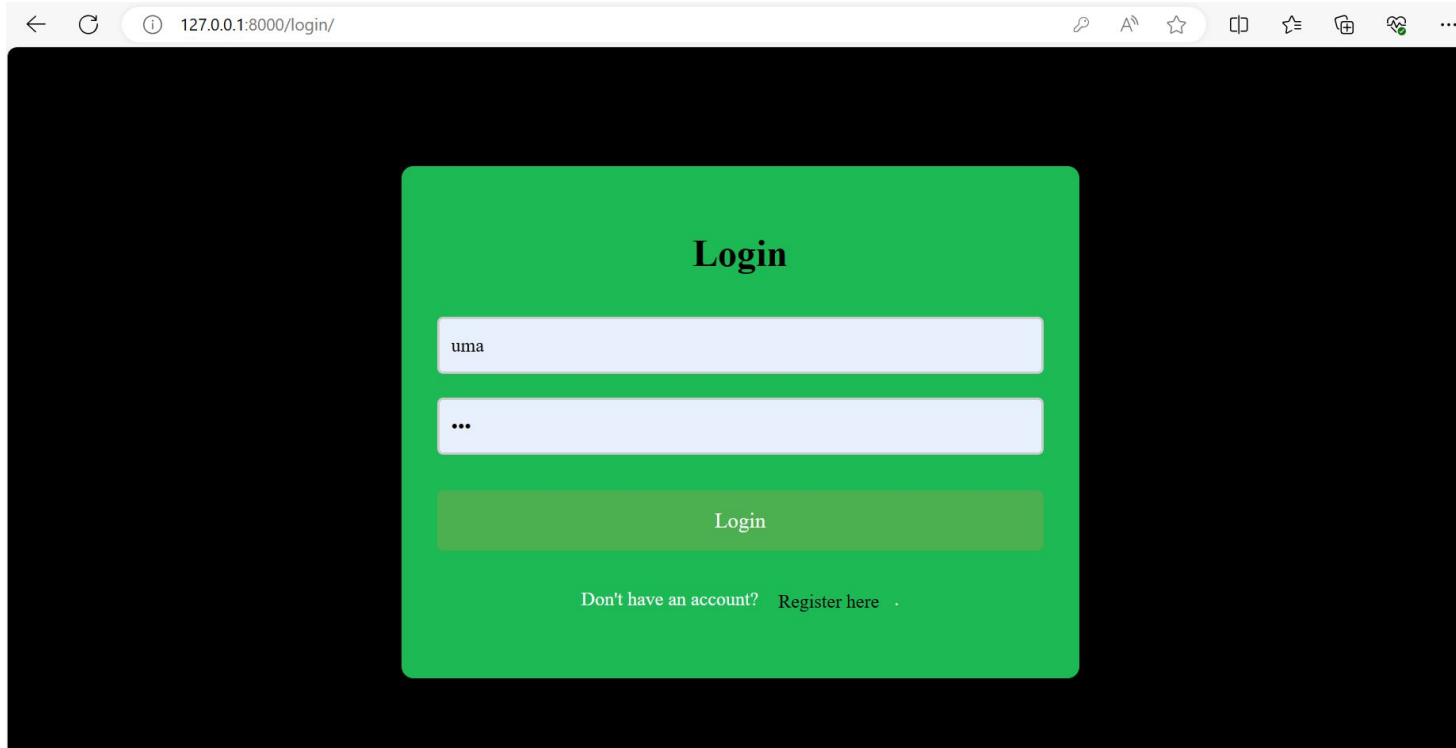
Category:
Enter category

Audio File:
Choose File No file chosen

Audio Img:
Choose File No file chosen

Submit

Login Page



← ↻ ⓘ 127.0.0.1:8000/login/ 🔍 🔊 ☆ 📄 ⚙️ 📌 📱 ...

Login

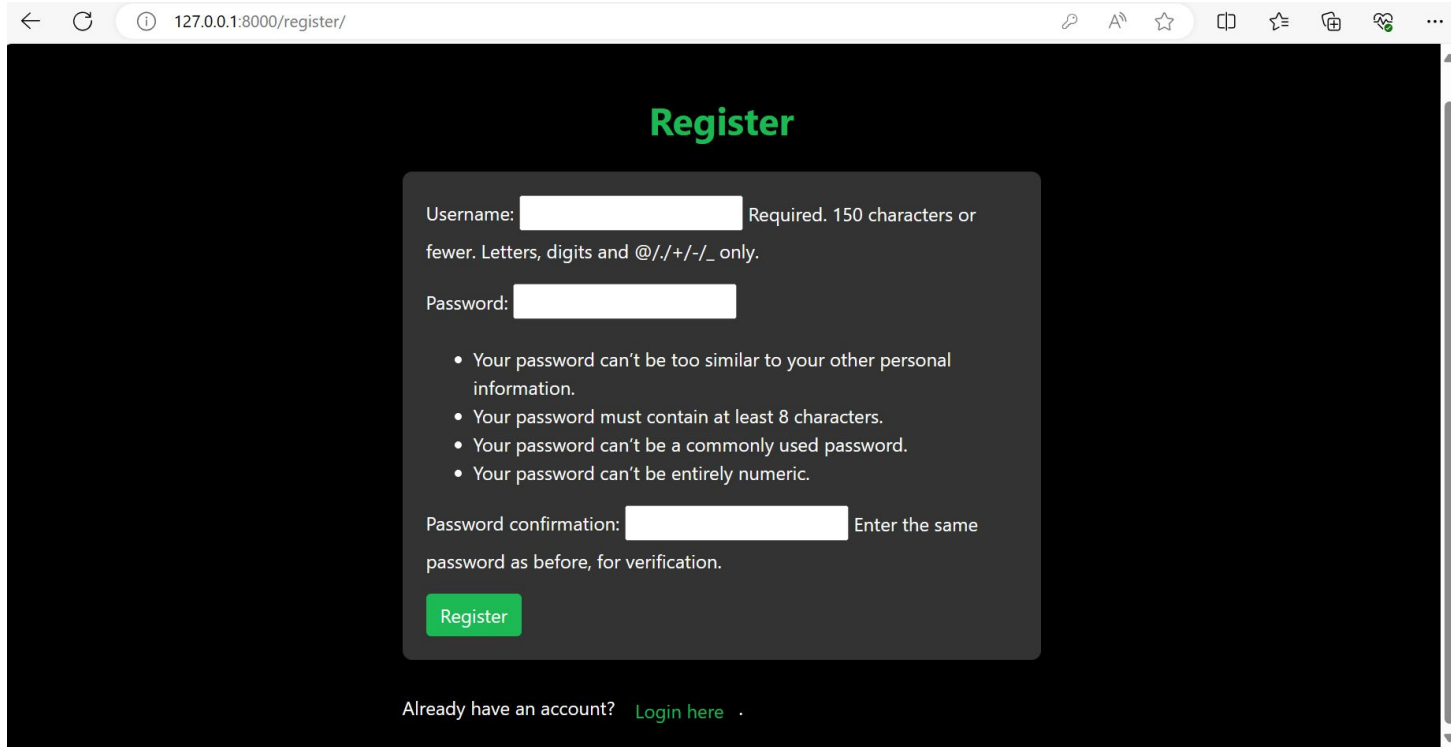
uma

...

Login

Don't have an account? [Register here](#)

Register-Page



← ↻ ⓘ 127.0.0.1:8000/register/ 🔍 🔊 ☆ 📄 ⌵ 📌 🌐 ⋮

Register

Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

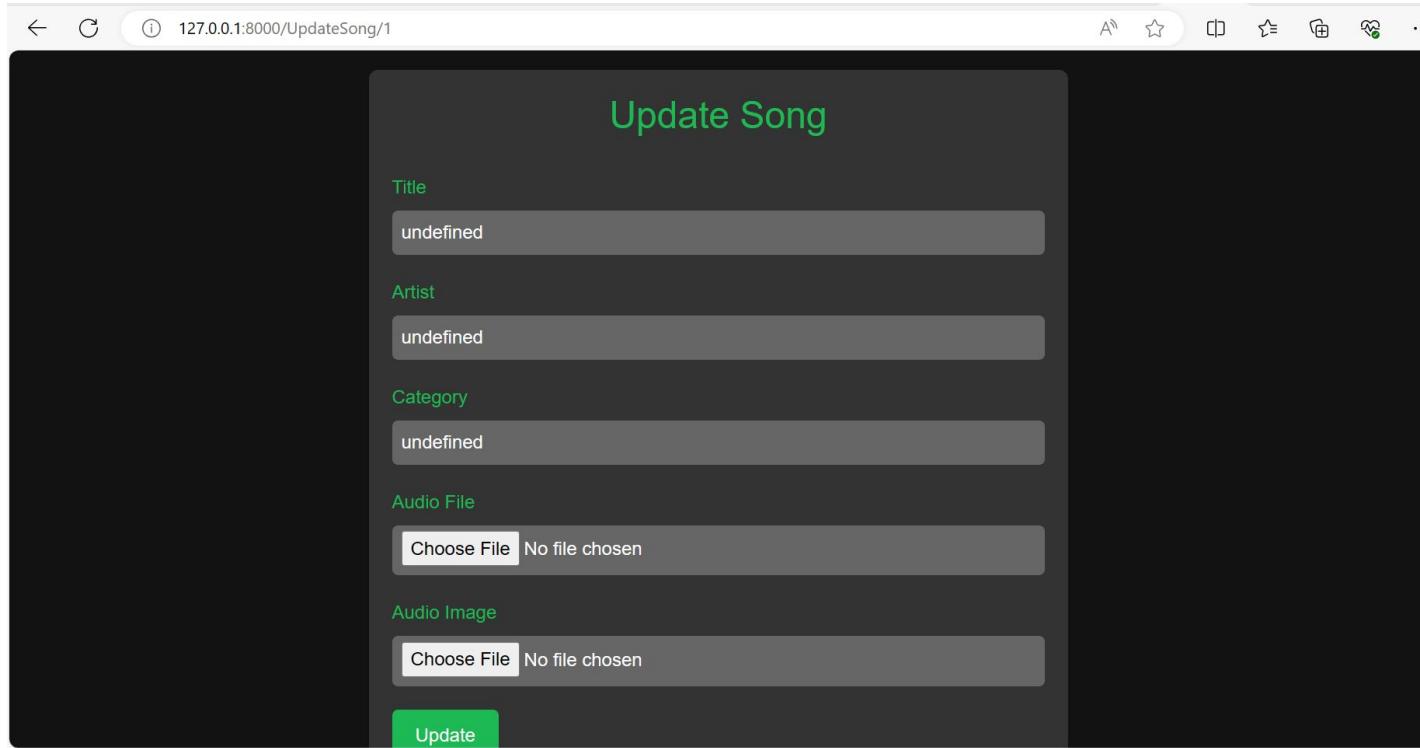
Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation: Enter the same password as before, for verification.

Already have an account? [Login here](#)

UpdateSong-Page



127.0.0.1:8000/UpdateSong/1

Update Song

Title
undefined

Artist
undefined

Category
undefined

Audio File
Choose File No file chosen

Audio Image
Choose File No file chosen

Update

Future Enhancements:

Advanced Search and Filtering: Enhance the search functionality to support advanced queries such as filtering songs by genre, artist, album, release date, and popularity.

User Profiles and Social Features: Implement user profiles where users can customize their preferences, create public playlists, follow other users, and share music recommendations.

Integration with External APIs: Integrate with external music APIs (e.g., Spotify, Apple Music) to import additional music metadata, album artwork, and potentially allow users to link their accounts for cross-platform functionality.

Cloud Storage Integration: Provide options for users to upload their music files securely to the cloud (e.g., AWS S3, Google Cloud Storage) and stream personal collections alongside the platform's catalog.

Conclusion

Developing a music application using Django offers a robust foundation for a user-friendly and scalable platform. Through secure user authentication, comprehensive song catalog management, and intuitive frontend design, we've created a functional prototype.

Looking forward, planned enhancements such as advanced search, social features, and mobile app integration promise to elevate the application's utility and appeal. Continuous integration, accessibility improvements, and machine learning-driven recommendations will further enhance the user experience.

This project highlights the versatility of Django in building complex web applications and underscores the importance of iterative development based on user feedback. With these enhancements, the music application is poised to evolve into a comprehensive and engaging platform for music enthusiasts worldwide.

Thank You!