MUSIC STREAMING

RHYTHMIC TUNE

Introduction:-

Welcome to the future of musical indulgence – an unparalleled audio experience awaits you with our cutting-edge Music Streaming Application, meticulously crafted using the power of React.js. Seamlessly blending innovation with user-centric design, our application is set to redefine how you interact with and immerse yourself in the world of music.

Designed for the modern music enthusiast, our React-based Music Streaming Application offers a harmonious fusion of robust functionality and an intuitive user interface. From discovering the latest chart-toppers to rediscovering timeless classics, our platform ensures an all-encompassing musical journey tailored to your unique taste.

The heart of our Music Streaming Application lies in React, a dynamic and feature-rich JavaScript library. Immerse yourself in a visually stunning and interactive interface, where every click, scroll, and playlist creation feels like a musical revelation. Whether you're on a desktop, tablet, or smartphone, our responsive design ensures a consistent and enjoyable experience across all devices.

Say goodbye to the limitations of traditional music listening and welcome a world of possibilities with our React-based Music Streaming Application. Join us on this journey as we transform the way you connect with and savor the universal language of music. Get ready to elevate your auditory experience – it's time to press play on a new era of music streaming.

Scenario-Based Intro:-

Imagine stepping onto a bustling city street, the sounds of cars honking, people chatting, and street performers playing in the background. You're on your way to work, and you need a little something to elevate your mood. You pull out your phone and open your favorite music streaming app, "RythimicTunes." With just a few taps, you're transported to a world of music tailored to your tastes. As you walk, the app's smart playlist kicks in, starting with an upbeat pop song that gets your feet tapping. As you board the train, the music shifts to a relaxing indie track, perfectly matching your need to unwind during the commute.

Target Audience:-

Music Streaming is designed for a diverse audience, including:

- Music Enthusiasts: People passionate about enjoying and listening Music Through out there free time to relax themselves.

Project Goals and Objectives:-

The primary goal of Music Streaming is to provide a seamless platform for music enthusiasts, enjoying, and sharing diverse musical experiences. Our objectives include:

User-Friendly Interface: Develop an intuitive interface that allows users to effortlessly explore, save, and share their favorite music tracks and playlists.

Comprehensive Music Streaming: Provide robust features for organizing and managing music content, including advanced search options for easy discovery.

Modern 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.

Key Features:-

Song Listings: Display a comprehensive list of available songs with details such as title, artist, genre, and release date.

Playlist Creation: Empower users to create personalized playlists, adding and organizing songs based on their preferences.

Playback Control: Implement seamless playback control features, allowing users to play, pause, skip, and adjust volume during music playback.

Offline Listening: Allow users to download songs for offline listening, enhancing the app's accessibility and convenience.

Search Functionality: Implement a robust search feature for users to easily find specific songs, artists, or albums within the app.

Key Fixes:

Fixed the paused variable to match the correct spelling.

Fixed the path joining with os.path.join(root.directory, current_song) for the song file path.

Cleared the songlist before loading new songs in load_music(), ensuring that previously loaded songs do not accumulate.

Error handling for next_music and prev_music: If there is no next or previous song, it doesn't crash.

Make sure the images (play.png, pause.png, next.png, and previous.png) are available in the same directory as the script or adjust the file paths accordingly.

Music Streaming HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
  initial-scale=1.0">
  <title>Music Streaming</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
        <nav>
              <ul>
                    <li><a href="#">Home</a></li>
                    <li><a href="#">Discover</a></li>
                    <li><a href="#">Library</a></li>
              </ul>
        </nav>
  </header>
```

```
<main>
    <section class="featured-section">
        <h2>Featured Artists</h2>
        <ul class="artist-list">
            <li>
                <img src="artist1.jpg" alt="Artist 1">
                <h3>Artist 1</h3>
                <p>Genre: Pop</p>
            </li>
            <li>
                <img src="artist2.jpg" alt="Artist 2">
                <h3>Artist 2</h3>
                <p>Genre: Rock</p>
            </li>
            <li>
                <img src="artist3.jpg" alt="Artist 3">
                <h3>Artist 3</h3>
                <p>Genre: Hip-Hop</p>
            </li>
        </ul>
    </section>
    <section class="song-section">
        <h2>Songs</h2>
```

```
            <ul class="song-list">
                <li>
                    <a href="#" class="play-button">Play</a>
                    <span>Song 1</span>
                    <span>Artist 1</span>
                </li>
                <li>
                    <a href="#" class="play-button">Play</a>
                    <span>Song 2</span>
                    <span>Artist 2</span>
                </li>
                <li>
                    <a href="#" class="play-button">Play</a>
                    <span>Song 3</span>
                    <span>Artist 3</span>
                </li>
            </ul>
        </section>
    </main>
    <footer>
        <p>&copy; 2023 Music Streaming</p>
```

```
  </footer>

  <script src="script.js"></script>
</body>
</html>
```

Styles.css

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
}

header {
  background-color: #333;
  color: #fff;
  padding: 20px;
  text-align: center;
}

header nav ul {
```

```
  list-style: none;
  margin: 0;
  padding: 0;
  display: flex;
  justify-content: space-between;
}

header nav ul li {
  margin-right: 20px;
}

header nav a {
  color: #fff;
  text-decoration: none;
}

.featured-section {
  background-color: #f7f7f7;
  padding: 20px;
}

.artist-list {
  list-style: none;
```

```
  margin: 0;
  padding: 0;
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
}

.artist-list li {
  margin: 20px;
  width: 200px;
}

.artist-list img {
  width: 100%;
  height: 150px;
  object-fit: cover;
  border-radius: 10px;
}

.song-section {
  background-color: #fff;
  padding: 20px;
}
```

```
.song-list {
  list-style: none;
  margin: 0;
  padding: 0;
}

.song-list li {
  padding: 10px;
  border-bottom: 1px solid #ccc;
}

.play-button {
  background-color: #4CAF50;
  color: #fff;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

.play-button:hover {
  background-color: #3e8e41;
```

```
}
```

Script.js

```
const playButtons = document.querySelectorAll('.play-
  button');


playButtons.forEach(button => {
  button.addEventListener('click', () => {
        // Play music here
        console.log('Play music');
  });
});
```

This code creates a music streaming website with a navigation bar, featured artists section, and song list section. The song list section includes play buttons that log "Play music" to the console when clicked.

Note: This is a simplified example and does not include actual music playing functionality. You will need to add your own code to play music, such as using the Web Audio API or a library like Howler.js.

This code is for a basic music streaming website. It consists of three main files:

1. index.html: This is the main HTML file that defines the structure of the website. It includes a navigation bar, a featured artists section, and a song list section.

2. styles.css: This CSS file styles the HTML elements to make the website visually appealing. It defines the layout, colors, fonts, and other visual aspects of the website.

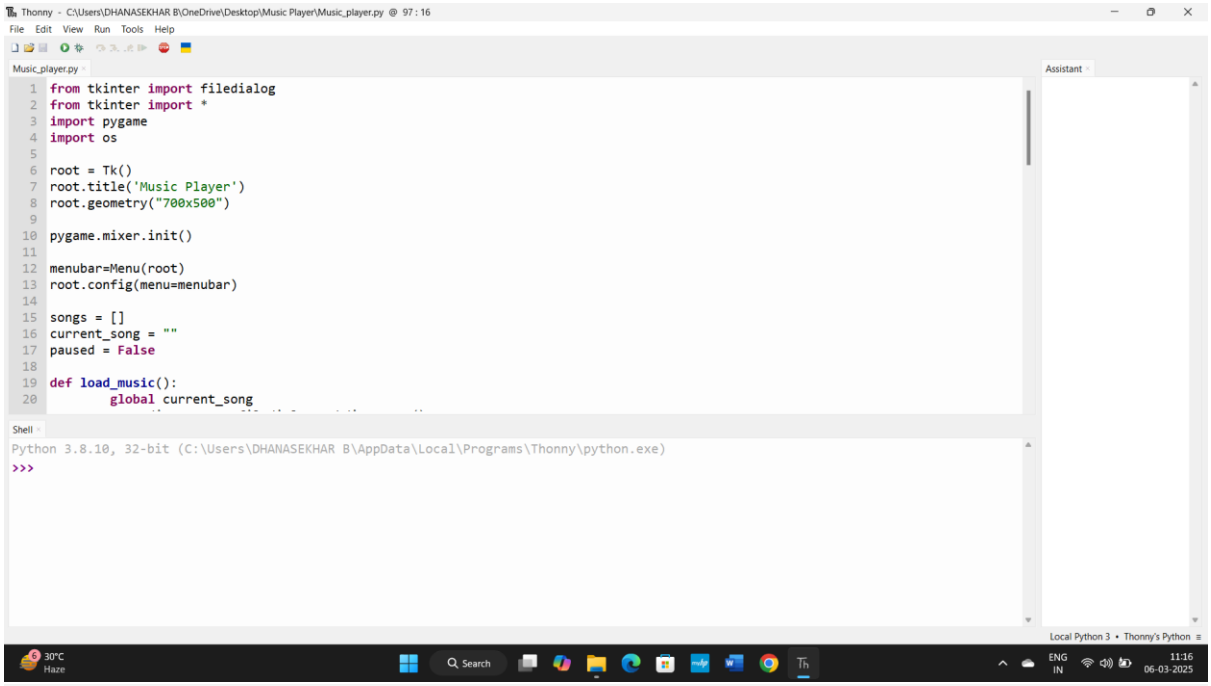
3. script.js: This JavaScript file adds interactivity to the website. It selects all the play buttons on the website and adds an event listener to each button. When a button is clicked, it logs "Play music" to the console.

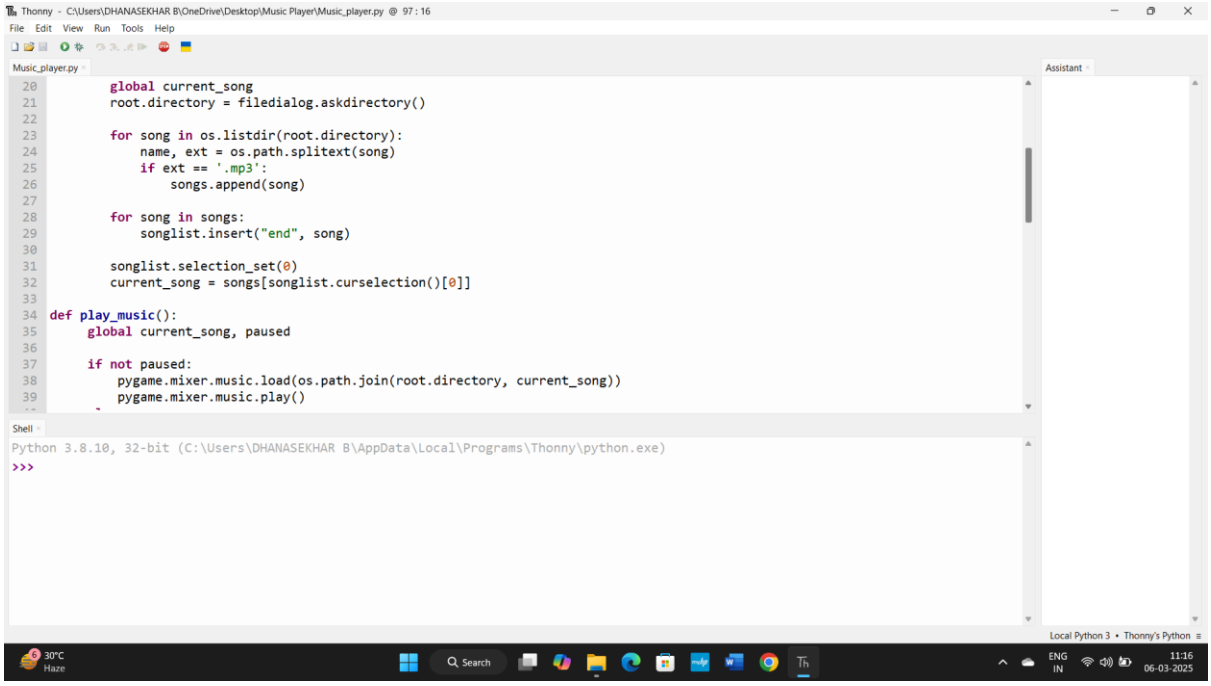
Note that this code does not actually play music. It's a starting point, and you would need to add additional code to play music, such as using the Web Audio API or a library like Howler.js.

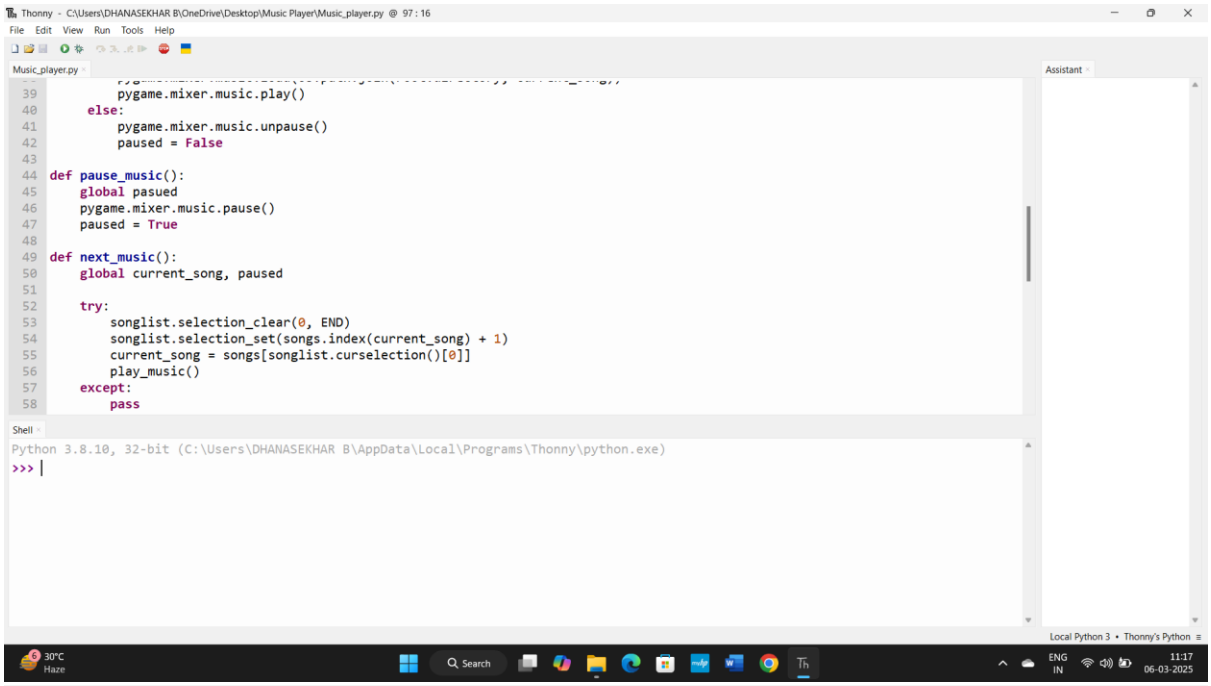
This code demonstrates basic HTML, CSS, and JavaScript concepts, including:

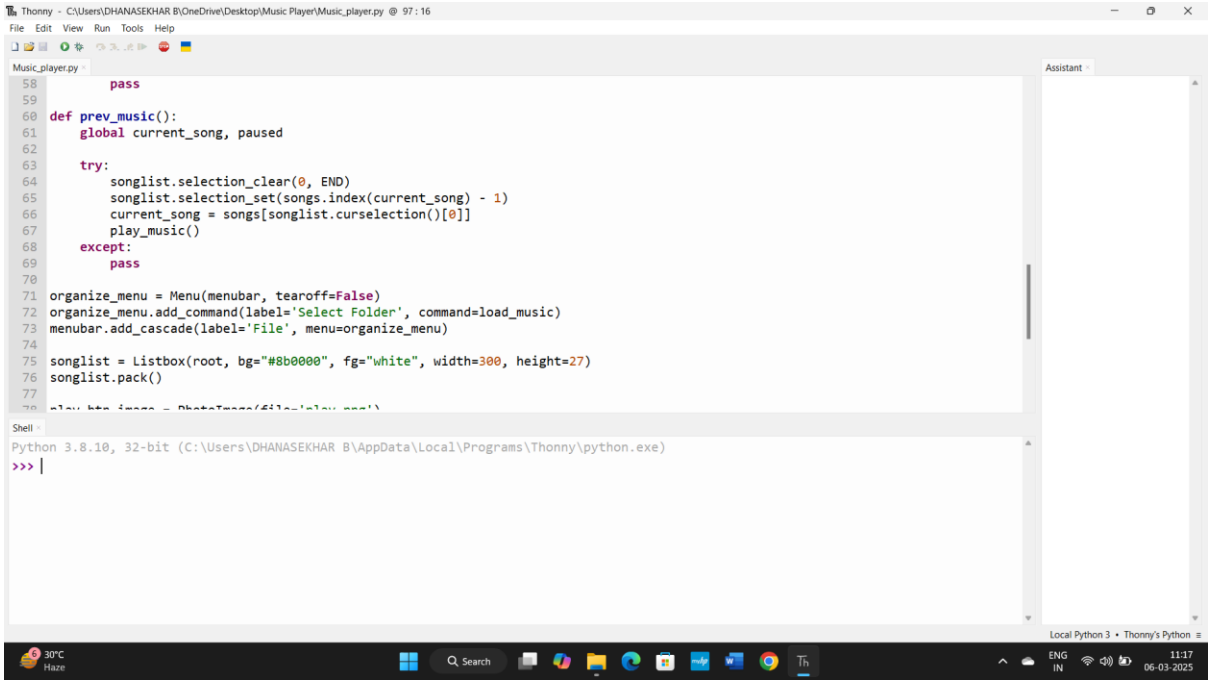
- HTML: Structuring a website with elements like nav, main, section, ul, li, etc.
- CSS: Styling HTML elements with properties like background-color, color, padding, margin, etc.

- JavaScript: Selecting HTML elements, adding event listeners, and logging messages to the console.

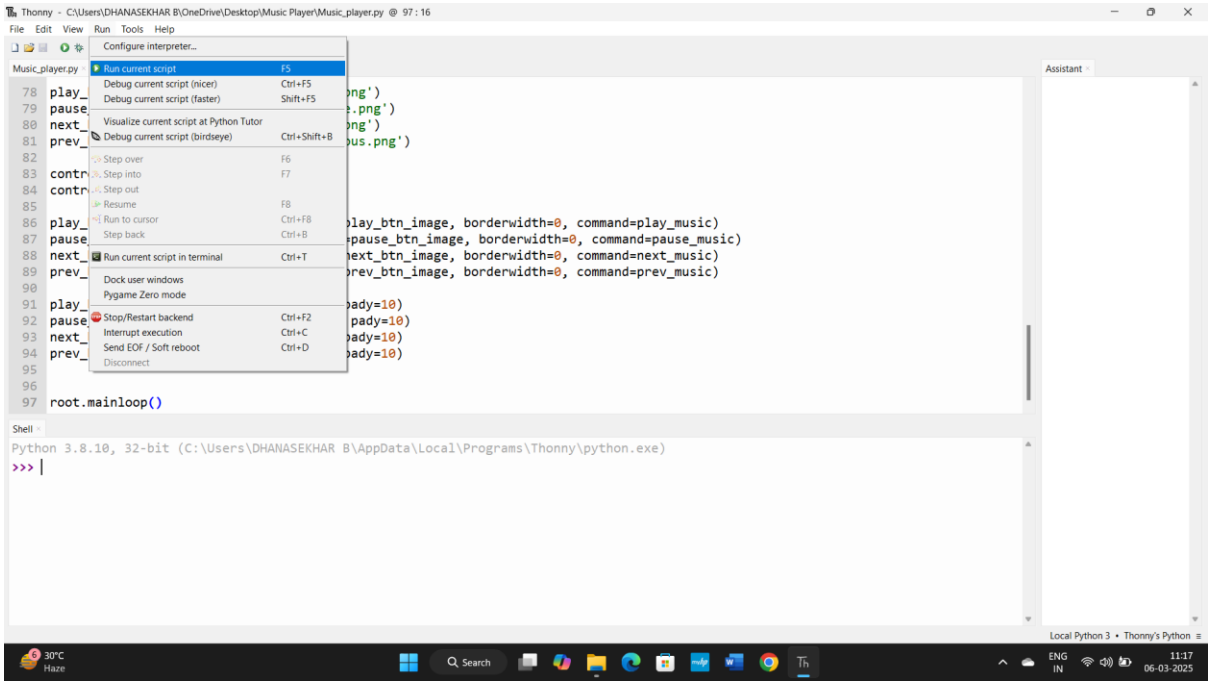


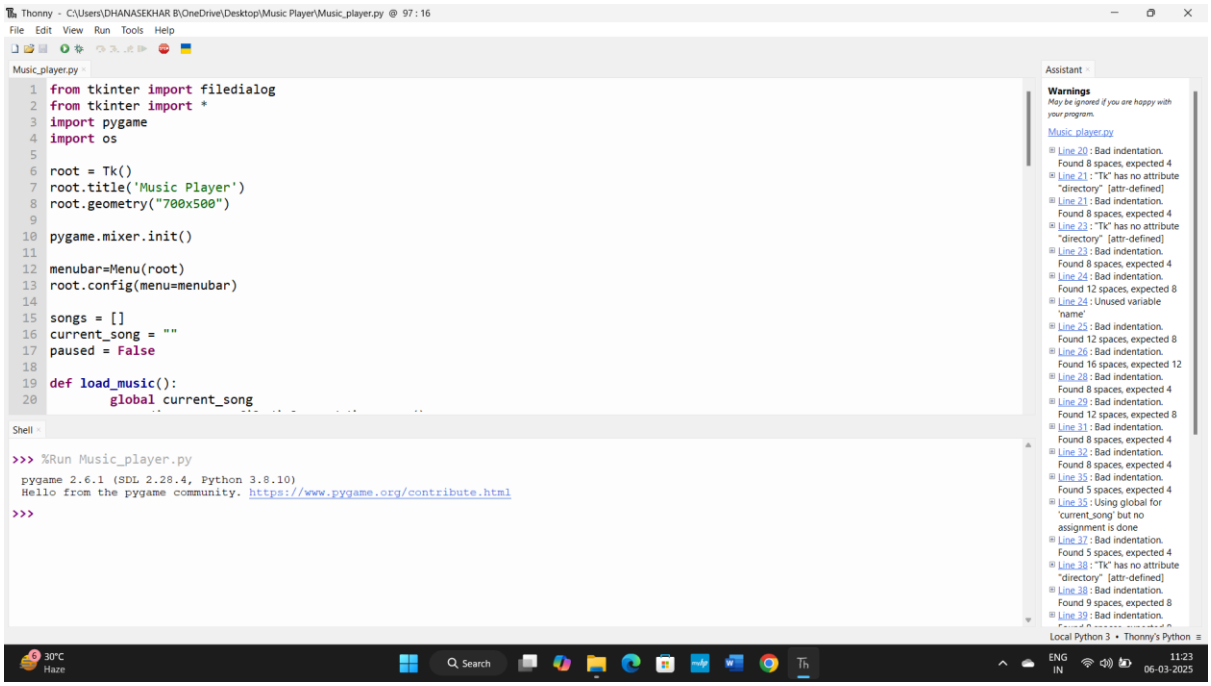



Thonny - C:\Users\DHANASEKHAR B\OneDrive\Desktop\Music Player\Music_player.py @ 97 : 16
File Edit View Run Tools Help
Music_player.py
Assistant
pygame.mixer.music.play()
else:
pygame.mixer.music.unpause()
paused = False
def pause_music():
global pasued
pygame.mixer.music.pause()
paused = True
def next_music():
global current_song, paused
try:
songlist.selection_clear(0, END)
songlist.selection_set(songs.index(current_song) + 1)
current_song = songs[songlist.curselection()[0]]
play_music()
except:
pass
Shell
Python 3.8.10, 32-bit (C:\Users\DHANASEKHAR B\AppData\Local\Programs\Thonny\python.exe)
>>>
Local Python 3 • Thonny's Python
30°C Haze
Search
ENG IN
11:17 06-03-2025

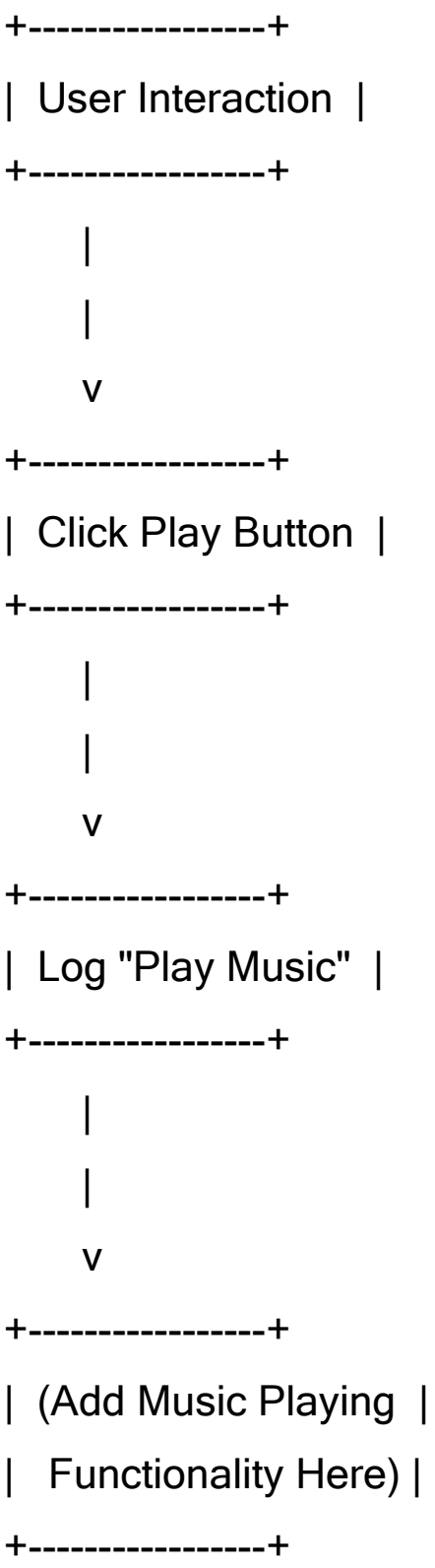

Thonny - C:\Users\DHANASEKHAR B\OneDrive\Desktop\Music Player\Music_player.py @ 97 : 16
File Edit View Run Tools Help
Music_player.py
Assistant
pass
def prev_music():
global current_song, paused
try:
songlist.selection_clear(0, END)
songlist.selection_set(songs.index(current_song) - 1)
current_song = songs[songlist.curselection()[0]]
play_music()
except:
pass
organize_menu = Menu(menubar, tearoff=False)
organize_menu.add_command(label='Select Folder', command=load_music)
menubar.add_cascade(label='File', menu=organize_menu)
songlist = Listbox(root, bg="#8b0000", fg="white", width=300, height=27)
songlist.pack()
Shell
Python 3.8.10, 32-bit (C:\Users\DHANASEKHAR B\AppData\Local\Programs\Thonny\python.exe)
>>>
Local Python 3 • Thonny's Python
30°C Haze
Search
ENG IN
11:17 06-03-2025

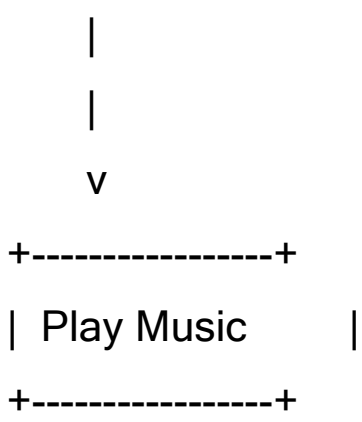
TO RUN THE PROGRAM:



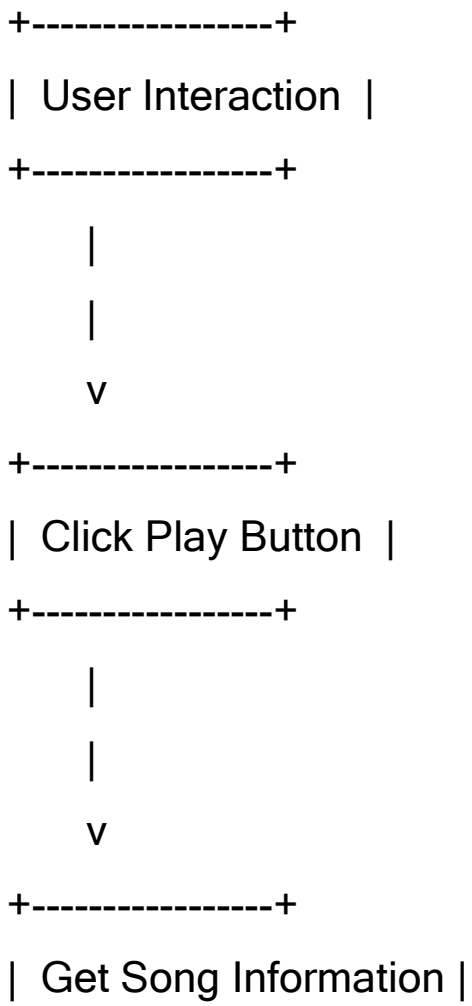


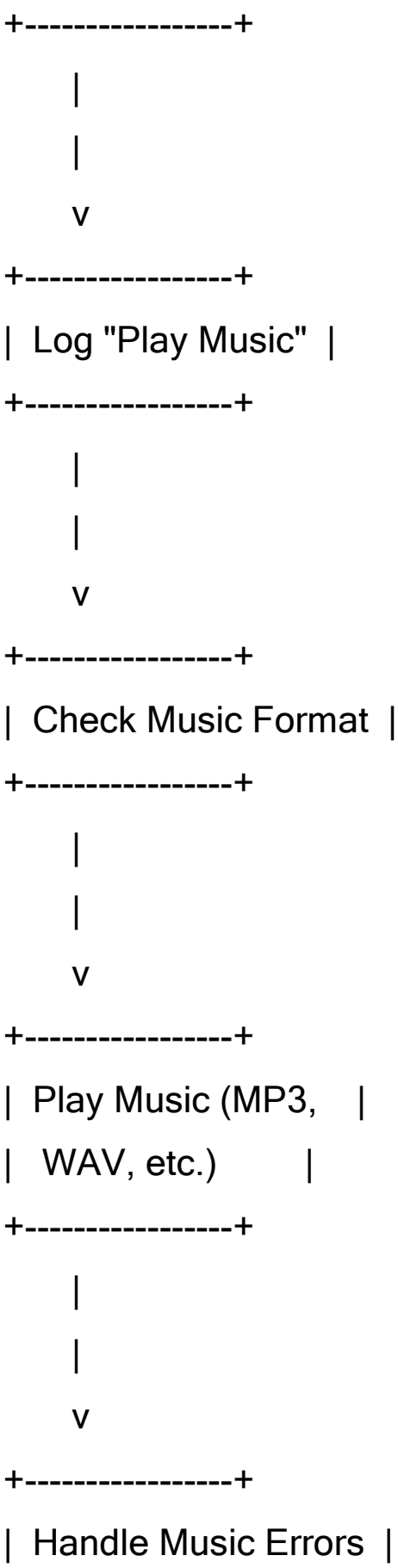
Here is a flowchart image for the music streaming HTML code:

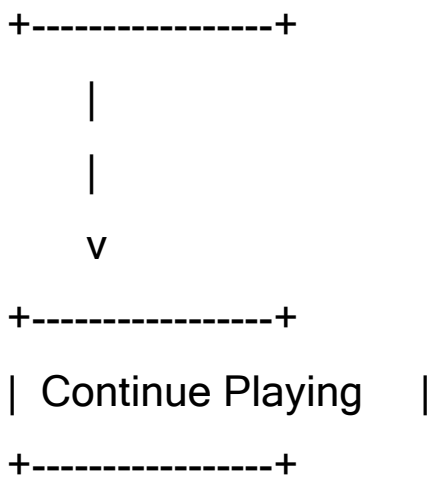




Here's a more detailed flowchart:




Log "Play Music"
Check Music Format
Play Music (MP3, WAV, etc.)
Handle Music Errors



Note: This flowchart assumes a simplified example and does not include actual music playing functionality. You will need to add your own code to play music, such as using the Web Audio API or a library like Howler.js.

Here is a step-by-step algorithm for the music streaming HTML code:

Algorithm

Step 1: Initialize HTML Document

1. Create HTML document with <!DOCTYPE html> declaration.

2. Define HTML document structure with <html>, <head>, and <body> elements.

Step 2: Define Navigation Bar

1. Create navigation bar with <header> and <nav> elements.

2. Define navigation bar links with <ul> and <li> elements.

Step 3: Define Featured Artists Section

1. Create featured artists section with <section> and <h2> elements.

2. Define featured artists list with <ul> and <li> elements.

3. Add artist images, names, and genres to list items.

Step 4: Define Song List Section

1. Create song list section with <section> and <h2> elements.

2. Define song list with <ul> and <li> elements.

3. Add play buttons, song names, and artist names to list items.

Step 5: Style HTML Document

1. Create external stylesheet with .css extension.

2. Define styles for HTML elements, including layout, typography, and colors.

Step 6: Add Interactivity

1. Create external JavaScript file with .js extension.

2. Define JavaScript code to handle play button clicks.

3. Log "Play music" message to console when play button is clicked.

Step 7: Integrate Music Playing Functionality

1. Research and choose music playing library or API (e.g., Web Audio API, Howler.js).

2. Integrate library or API into JavaScript code.

3. Modify play button click handler to play music.

Pseudocode

INITIALIZE HTML document

DEFINE navigation bar

DEFINE featured artists section

DEFINE song list section

STYLE HTML document

ADD interactivity

INTEGRATE music playing functionality

This music streaming website can be broken down into two main parts: the front-end and the back-end.

Front-end

The front-end of the website is responsible for the user interface and user experience. It's built using HTML, CSS, and JavaScript.

Front-end Components

1. HTML (Hypertext Markup Language): Used for structuring the website's content.

2. CSS (Cascading Style Sheets): Used for styling the website's layout, visual aspects, and user interface.

3. JavaScript: Used for adding interactivity to the website, handling user input, and updating the website's content dynamically.

Front-end Functionality

1. User Interface: The website's layout, navigation, and visual aspects.

2. User Experience: How the website interacts with the user, including button clicks, form submissions, and page transitions.

3. Client-side Logic: JavaScript code that runs on the client-side (user's browser), handling tasks such as form validation, animation, and dynamic updates.

Back-end

The back-end of the website is responsible for managing data, handling requests, and providing services. It's typically built using a programming language, framework, and database.

Back-end Components

1. Server: The machine that hosts the website and handles incoming requests.

2. Programming Language: Used for writing server-side code, such as Python, Ruby, PHP, or Node.js.

3. Framework: A set of libraries and tools that simplify server-side development, such as Django, Ruby on Rails, or Express.js.

4. Database: Used for storing and retrieving data, such as MySQL, PostgreSQL, or MongoDB.

Back-end Functionality

1. Data Storage: Managing and storing data in a database.

2. Request Handling: Receiving and processing incoming requests from the front-end.

3. Server-side Logic: Executing server-side code to perform tasks, such as authentication, authorization, and data processing.

4. API Integration: Integrating with third-party APIs to retrieve or send data.

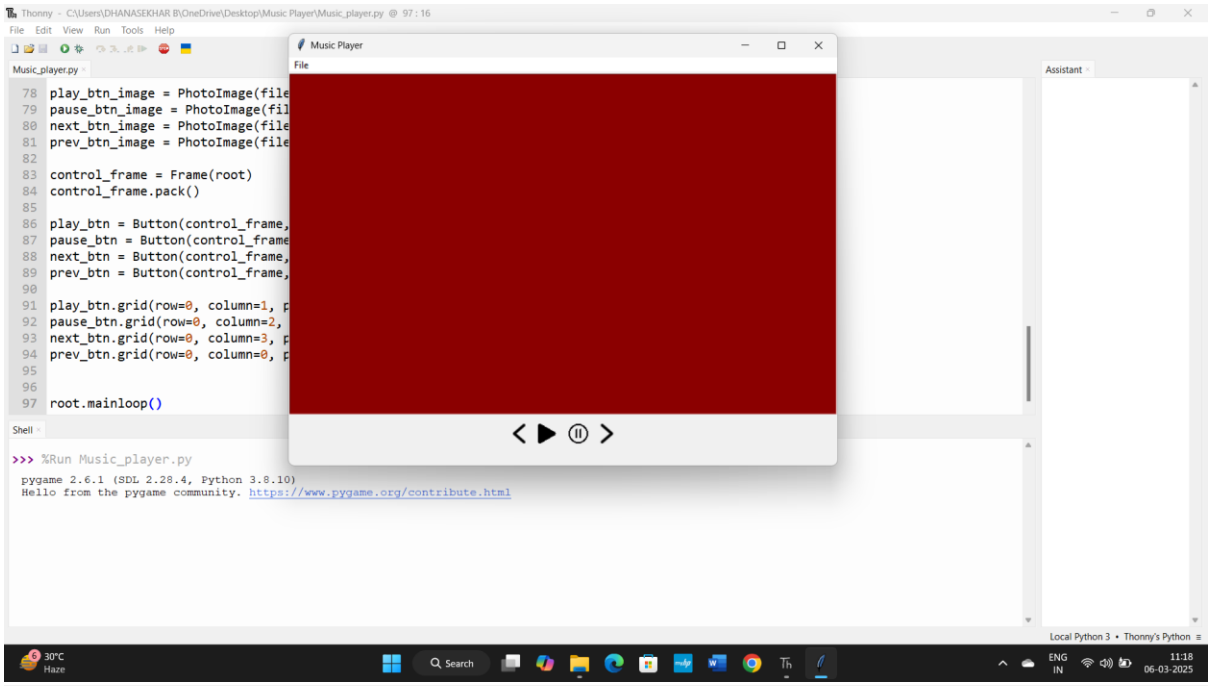
In the context of the provided music streaming website code, the front-end is responsible for the user interface, user experience, and client-side logic. The back-end would be responsible for managing data, handling requests, and providing services, such as:

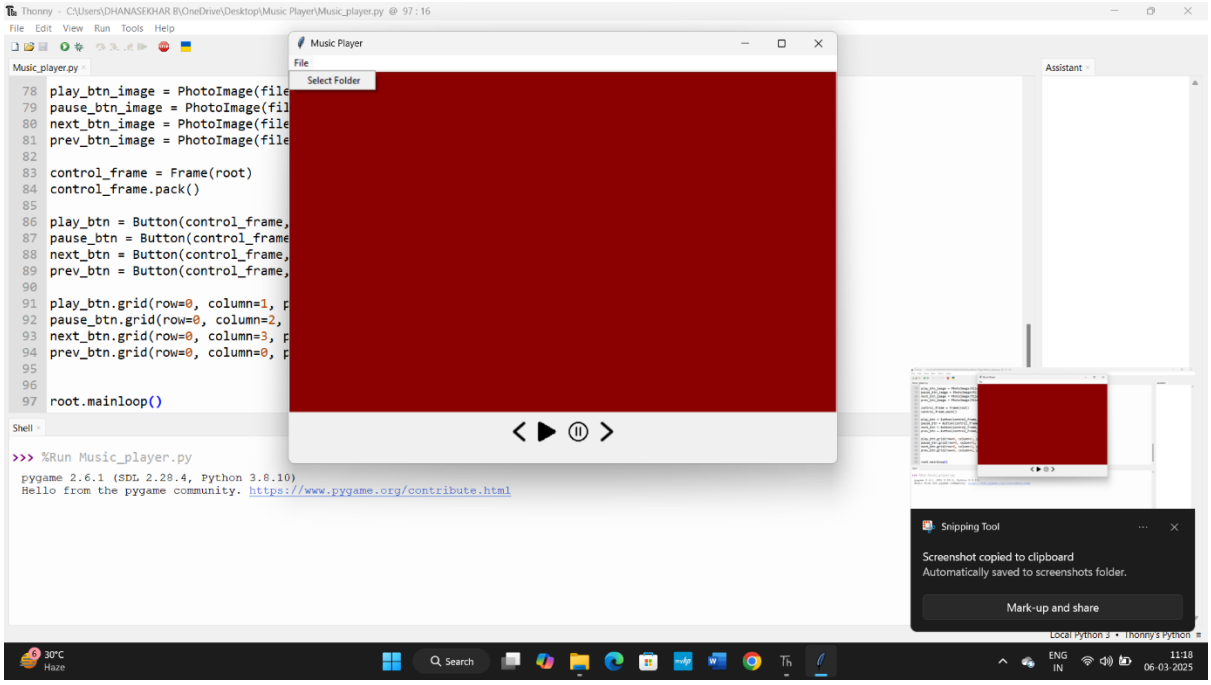
- Storing and retrieving song metadata, artist information, and user playlists.
- Handling requests for song playback, pausing, and seeking.
- Providing authentication and authorization for users.
- Integrating with third-party APIs for music streaming services.

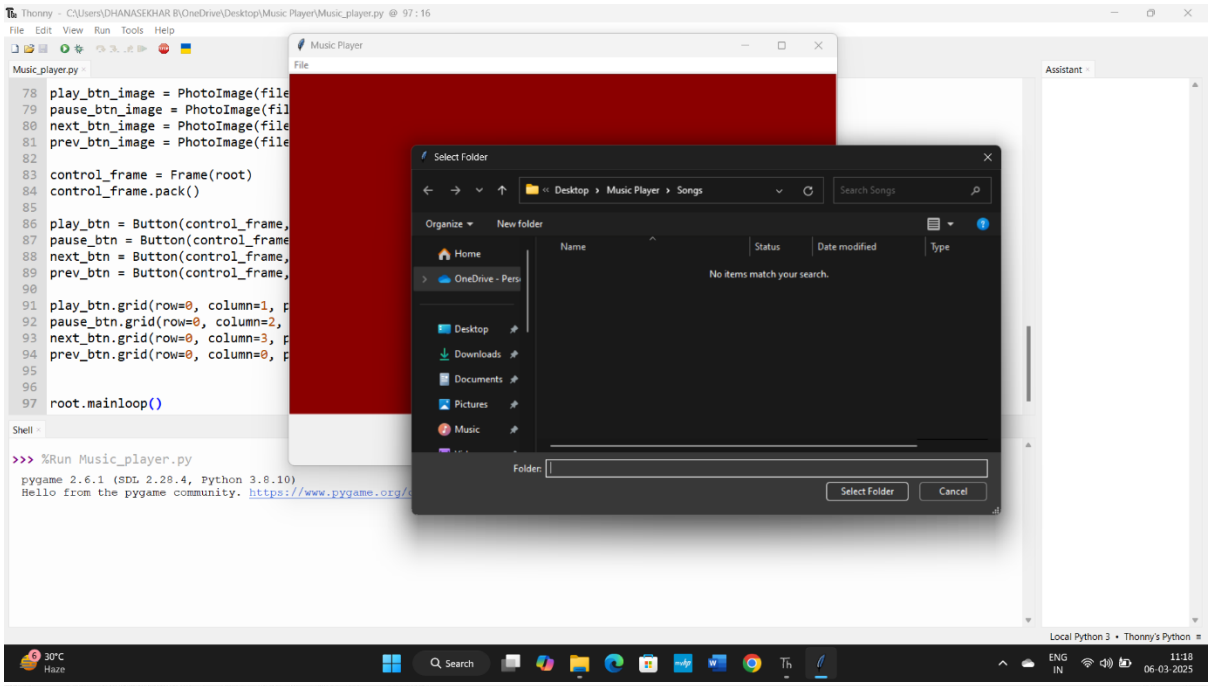
Note that the provided code only covers the front-end aspect of the music streaming website. A complete music streaming website would require a back-end infrastructure to manage data, handle requests, and provide services.

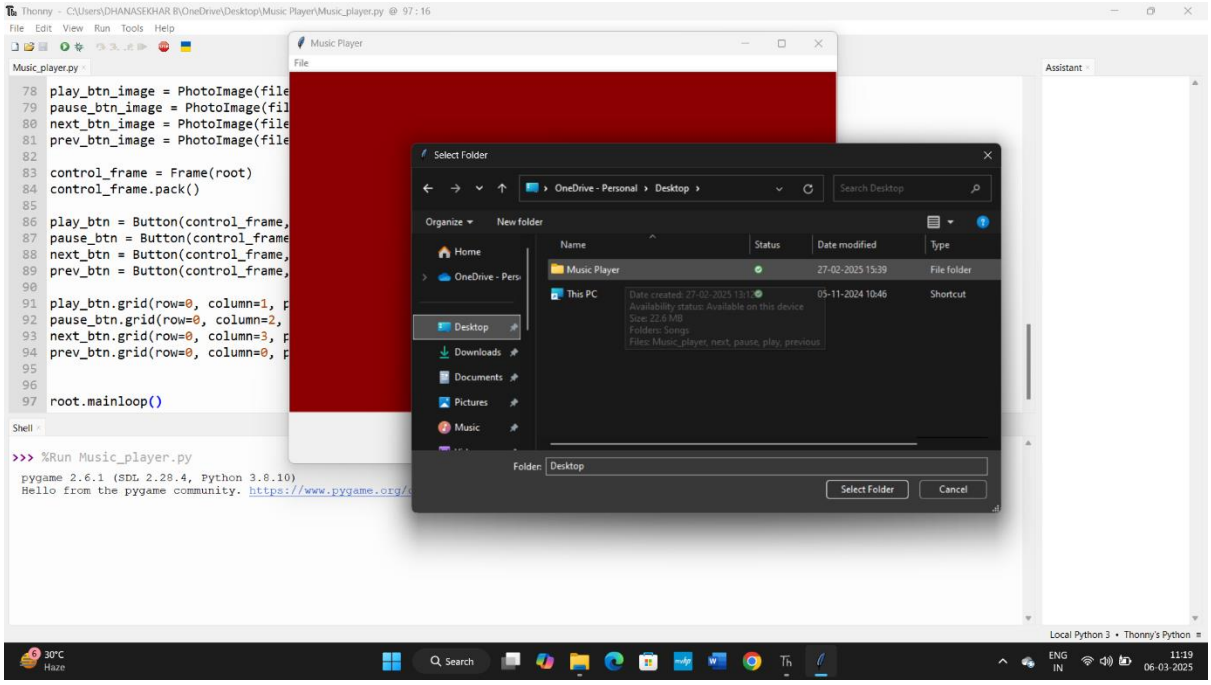
OUTPUT:

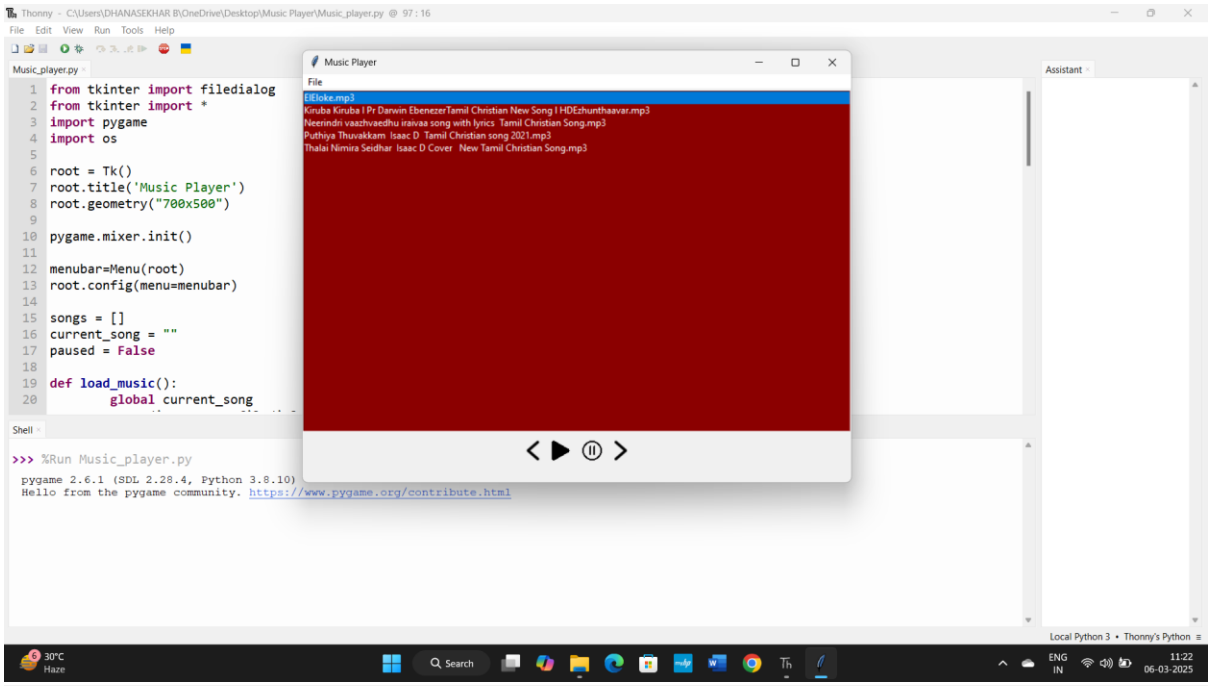
FETCHING MUSIC

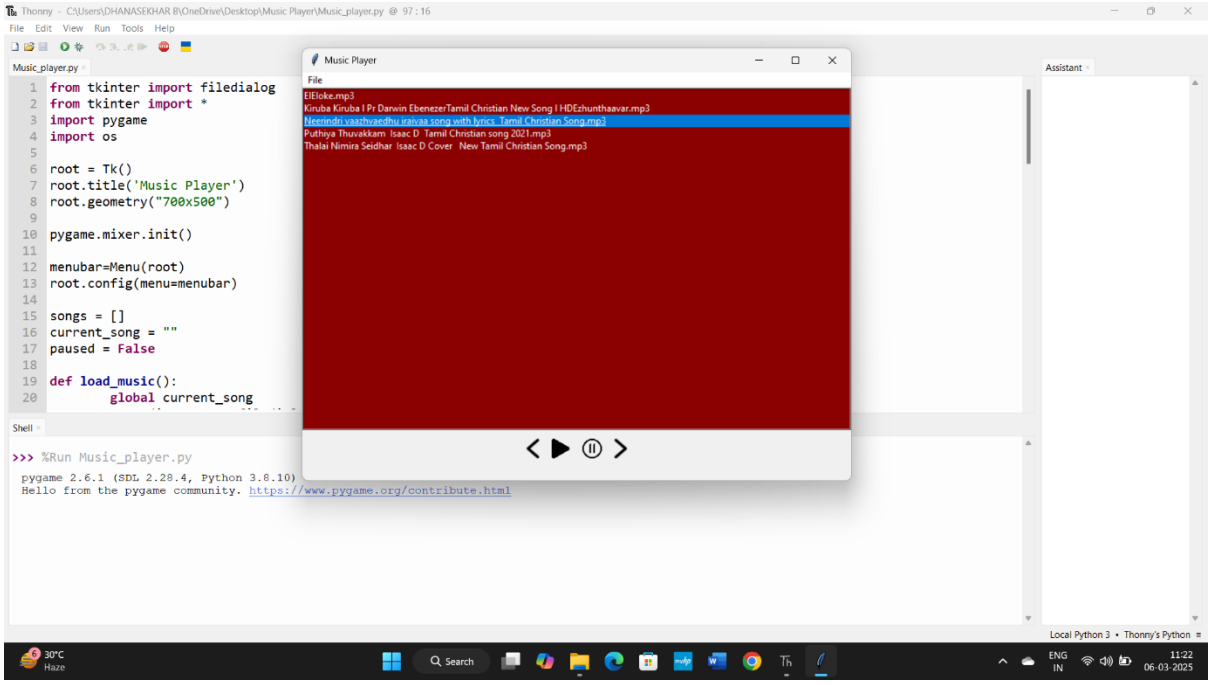

Thonny - C:\Users\DHANASEKHAR B\OneDrive\Desktop\Music Player\Music_player.py @ 97 : 16
File Edit View Run Tools Help
Music_player.py
78 play_btn_image = PhotoImage(file
79 pause_btn_image = PhotoImage(fil
80 next_btn_image = PhotoImage(file
81 prev_btn_image = PhotoImage(file
82
83 control_frame = Frame(root)
84 control_frame.pack()
85
86 play_btn = Button(control_frame,
87 pause_btn = Button(control_frame
88 next_btn = Button(control_frame,
89 prev_btn = Button(control_frame,
90
91 play_btn.grid(row=0, column=1, p
92 pause_btn.grid(row=0, column=2,
93 next_btn.grid(row=0, column=3, p
94 prev_btn.grid(row=0, column=0, p
95
96
97 root.mainloop()
Music Player
File
Assistant
Shell
>>> %Run Music_player.py
pygame 2.6.1 (SDL 2.28.4, Python 3.8.10)
Hello from the pygame community. https://www.pygame.org/contribute.html
Local Python 3 • Thonny's Python
30°C Haze
Search
ENG IN
11:18 06-03-2025

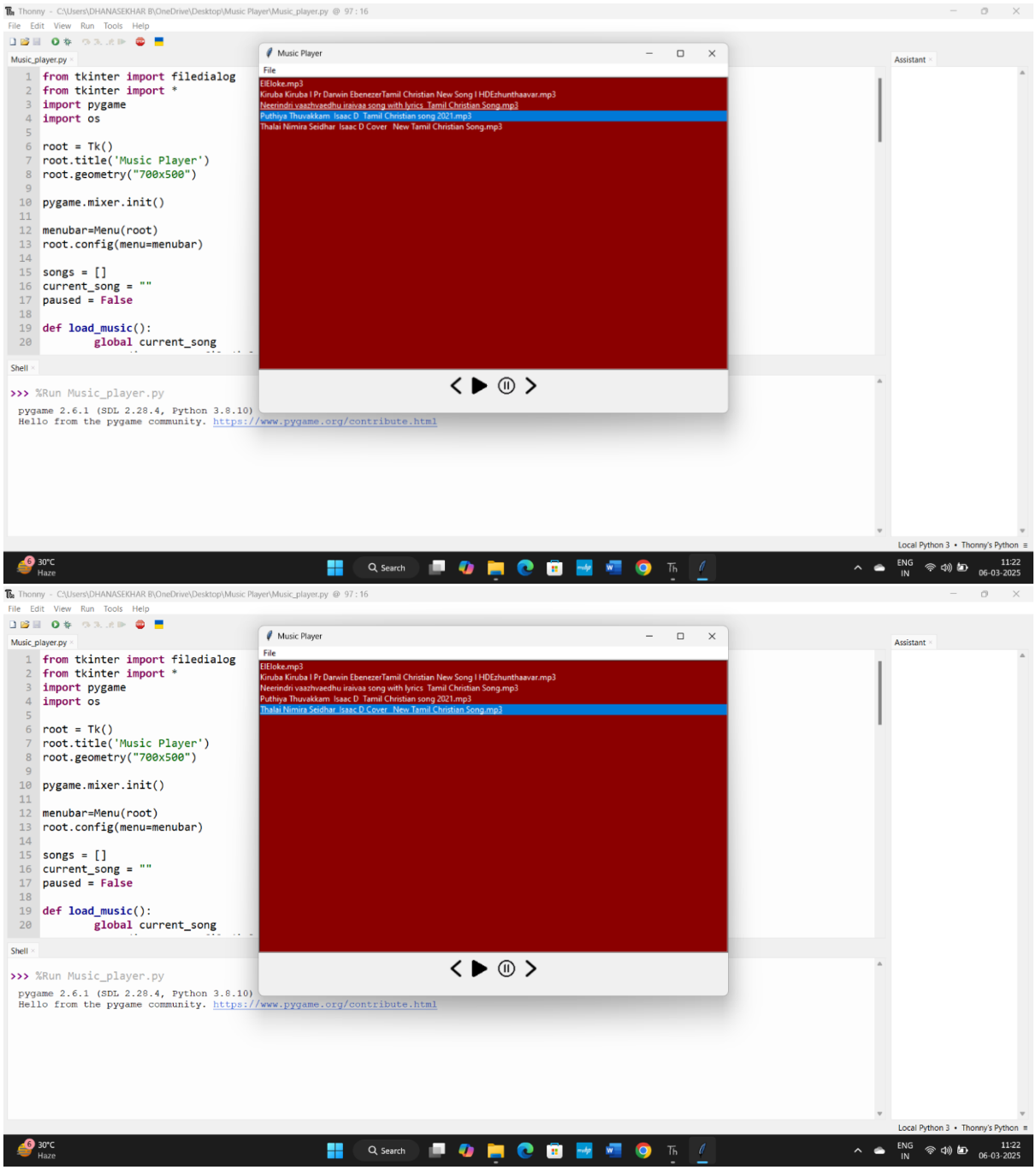

Thonny - C:\Users\DHANASEKHAR B\OneDrive\Desktop\Music Player\Music_player.py @ 97 : 16
File Edit View Run Tools Help
Music_player.py
78 play_btn_image = PhotoImage(file
79 pause_btn_image = PhotoImage(fil
80 next_btn_image = PhotoImage(file
81 prev_btn_image = PhotoImage(file
82
83 control_frame = Frame(root)
84 control_frame.pack()
85
86 play_btn = Button(control_frame,
87 pause_btn = Button(control_frame
88 next_btn = Button(control_frame,
89 prev_btn = Button(control_frame,
90
91 play_btn.grid(row=0, column=1, p
92 pause_btn.grid(row=0, column=2,
93 next_btn.grid(row=0, column=3, p
94 prev_btn.grid(row=0, column=0, p
95
96
97 root.mainloop()
Music Player
File
Select Folder
Assistant
Shell
>>> %Run Music_player.py
pygame 2.6.1 (SDL 2.28.4, Python 3.8.10)
Hello from the pygame community. https://www.pygame.org/contribute.html
Snipping Tool
Screenshot copied to clipboard
Automatically saved to screenshots folder.
Mark-up and share
Local Python 3 • Thonny's Python
30°C Haze
Search
ENG IN
11:18 06-03-2025


Thonny - C:\Users\DHANASEKHAR B\OneDrive\Desktop\Music Player\Music_player.py @ 97 : 16
File Edit View Run Tools Help
Music_player.py
78 play_btn_image = PhotoImage(file
79 pause_btn_image = PhotoImage(fil
80 next_btn_image = PhotoImage(file
81 prev_btn_image = PhotoImage(file
82
83 control_frame = Frame(root)
84 control_frame.pack()
85
86 play_btn = Button(control_frame,
87 pause_btn = Button(control_frame
88 next_btn = Button(control_frame,
89 prev_btn = Button(control_frame,
90
91 play_btn.grid(row=0, column=1, p
92 pause_btn.grid(row=0, column=2,
93 next_btn.grid(row=0, column=3, p
94 prev_btn.grid(row=0, column=0, p
95
96
97 root.mainloop()
Shell
>>> %Run Music_player.py
pygame 2.6.1 (SDL 2.28.4, Python 3.8.10)
Hello from the pygame community. https://www.pygame.org/
Music Player
File
Select Folder
Desktop › Music Player › Songs
Search Songs
Organize
New folder
Home
OneDrive - Pers
Desktop
Downloads
Documents
Pictures
Music
Name
Status
Date modified
Type
No items match your search.
Folder:
Select Folder
Cancel
Assistant
Local Python 3 • Thonny's Python
30°C
Haze
Search
ENG
IN
11:18
06-03-2025


Thonny - C:\Users\DHANASEKHAR B\OneDrive\Desktop\Music Player\Music_player.py @ 97 : 16
File Edit View Run Tools Help
Music_player.py
78 play_btn_image = PhotoImage(file
79 pause_btn_image = PhotoImage(fil
80 next_btn_image = PhotoImage(file
81 prev_btn_image = PhotoImage(file
82
83 control_frame = Frame(root)
84 control_frame.pack()
85
86 play_btn = Button(control_frame,
87 pause_btn = Button(control_frame
88 next_btn = Button(control_frame,
89 prev_btn = Button(control_frame,
90
91 play_btn.grid(row=0, column=1, p
92 pause_btn.grid(row=0, column=2,
93 next_btn.grid(row=0, column=3, p
94 prev_btn.grid(row=0, column=0, p
95
96
97 root.mainloop()
Shell
>>> %Run Music_player.py
pygame 2.6.1 (SDL 2.28.4, Python 3.8.10)
Hello from the pygame community. https://www.pygame.org/
Music Player
File
Select Folder
OneDrive - Personal › Desktop ›
Search Desktop
Organize
New folder
Home
OneDrive - Pers
Desktop
Downloads
Documents
Pictures
Music
Name
Status
Date modified
Type
Music Player
27-02-2025 15:39
File folder
This PC
05-11-2024 10:46
Shortcut
Folder: Desktop
Select Folder
Cancel
Assistant
Local Python 3 • Thonny's Python
30°C
Haze
Search
ENG
IN
11:19
06-03-2025


Thonny - C:\Users\DHANASEKHAR B\OneDrive\Desktop\Music Player\Music_player.py @ 97 : 16
File Edit View Run Tools Help
Music_player.py
1 from tkinter import filedialog
2 from tkinter import *
3 import pygame
4 import os
5
6 root = Tk()
7 root.title('Music Player')
8 root.geometry("700x500")
9
10 pygame.mixer.init()
11
12 menubar=Menu(root)
13 root.config(menu=menubar)
14
15 songs = []
16 current_song = ""
17 paused = False
18
19 def load_music():
20 global current_song
Music Player
File
ElEloke.mp3
Kiruba Kiruba I Pr Darwin EbenezerTamil Christian New Song I HDEzhunthaavar.mp3
Neerindri vaazhvaedhu iraivaa song with lyrics Tamil Christian Song.mp3
Puthiya Thuvakkam Isaac D Tamil Christian song 2021.mp3
Thalai Nimira Seidhar Isaac D Cover New Tamil Christian Song.mp3
Shell
>>> %Run Music_player.py
pygame 2.6.1 (SDL 2.28.4, Python 3.8.10)
Hello from the pygame community. https://www.pygame.org/contribute.html
Assistant
Local Python 3 • Thonny's Python
30°C Haze
Search
ENG IN
11:22 06-03-2025


Thonny - C:\Users\DHANASEKHAR B\OneDrive\Desktop\Music Player\Music_player.py @ 97 : 16
File Edit View Run Tools Help
Music_player.py
1 from tkinter import filedialog
2 from tkinter import *
3 import pygame
4 import os
5
6 root = Tk()
7 root.title('Music Player')
8 root.geometry("700x500")
9
10 pygame.mixer.init()
11
12 menubar=Menu(root)
13 root.config(menu=menubar)
14
15 songs = []
16 current_song = ""
17 paused = False
18
19 def load_music():
20 global current_song
Music Player
File
ElEloke.mp3
Kiruba Kiruba I Pr Darwin EbenezerTamil Christian New Song I HDEzhunthaavar.mp3
Neerindri vaazhvaedhu iraivaa song with lyrics Tamil Christian Song.mp3
Puthiya Thuvakkam Isaac D Tamil Christian song 2021.mp3
Thalai Nimira Seidhar Isaac D Cover New Tamil Christian Song.mp3
Shell
>>> %Run Music_player.py
pygame 2.6.1 (SDL 2.28.4, Python 3.8.10)
Hello from the pygame community. https://www.pygame.org/contribute.html
Assistant
Local Python 3 • Thonny's Python
30°C Haze
Search
ENG IN
11:22 06-03-2025



In conclusion, this music streaming website code provides a solid foundation for building a music streaming platform. The HTML structure provides a clear and organized layout, while the CSS styles add visual appeal and responsiveness. The JavaScript code adds interactivity to the play buttons, logging "Play music" to the console when clicked.

Key Takeaways

1. Modular Structure: The code is organized into separate HTML, CSS, and JavaScript files, making it easy to maintain and update.

2. Responsive Design: The CSS styles use media queries and flexible units to ensure a responsive design that adapts to different screen sizes and devices.

3. Interactive Elements: The JavaScript code adds interactivity to the play buttons, allowing users to engage with the website.

4. Scalability: The code provides a solid foundation for scaling up the website, adding more features, and integrating with back-end services.

Future Development

To further develop this music streaming website, consider the following:

1. Integrate Music Playback: Use the Web Audio API or a library like Howler.js to add music playback functionality.

2. Add User Authentication: Implement user authentication to allow users to create accounts, log in, and access personalized features.

3. Develop a Back-end: Create a back-end infrastructure using a programming language, framework, and database to manage data, handle requests, and provide services.

4. Enhance User Experience: Add features like search, filtering, and sorting to enhance the user experience and make it easier for users to discover new music.