Music Streaming Service

CN Mini Project

Satvik RK - PES1UG23CS525
Sasank Sai - PES1UG23CS522

About the Project:

This project is a simple music playback system using Python socket programming and SSL for secure communication. The server stores and plays songs locally based on requests it receives from connected clients. Clients send the name of a song they want to hear, and the server responds by playing that song on its own system. The project demonstrates core networking concepts like IP addressing, ports, and TCP connections, while also introducing the use of SSL certificate verification to ensure secure client-server communication. It serves as a practical introduction to how encrypted connections and request-response protocols work in a controlled local environment.

Code:

Client:

```python
import socket
import ssl

SERVER_IP = "192.168.0.112"
SERVER_PORT = 5000
CERT_FILE = 'server.crt'  # Path to the server's SSL certificate

def connect_to_server():
    """Connects to the server and interacts with it."""
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
        # Create SSL context for the client and verify server's certificate
        context = ssl.create_default_context(ssl.Purpose.SERVER_AUTH)
        context.load_verify_locations(CERT_FILE)

        # Wrap the client socket with SSL encryption
        client_socket = context.wrap_socket(client_socket,
server_hostname=SERVER_IP)

        client_socket.connect((SERVER_IP, SERVER_PORT))

        while True:
            # Receive song list or final message
            data = client_socket.recv(4096).decode()
            print(data)

            if "No songs available" in data or "Goodbye" in data:
                break

            # User selects song or exits
            choice = input("Your choice: ").strip()
            client_socket.sendall(choice.encode())

            if choice.lower() == "exit":
                goodbye = client_socket.recv(1024).decode()
                print(goodbye)
                break

            # Receive confirmation or error
            response = client_socket.recv(1024).decode()
            print(response)

            if "Invalid" in response or "Error" in response:
                continue

            # Receive command instructions
            response = client_socket.recv(1024).decode()
            print(response)

            # Music control loop
            while True:
                command = input("Command (pause/resume/stop): ").strip().lower()
                client_socket.sendall(command.encode())
                response = client_socket.recv(1024).decode()
                print(response)

                if "Stopped" in response or "Song finished" in response or
"Error" in response:
                    break

if __name__ == "__main__":
    connect_to_server()
```

Server:

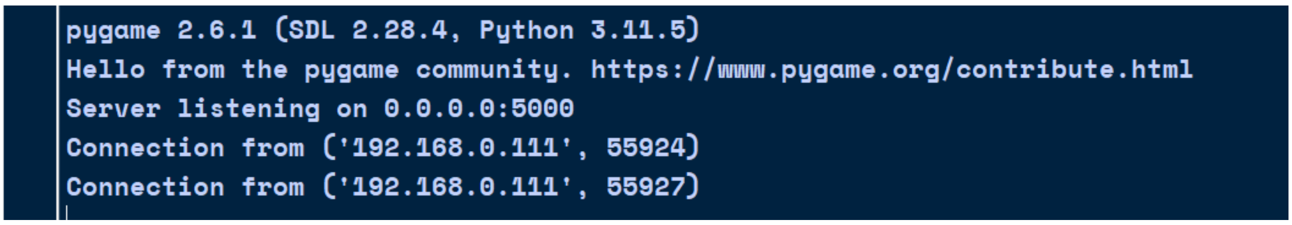
```
import socket
import os
import pygame
import ssl

MUSIC_FOLDER = "music"
HOST = "0.0.0.0"
PORT = 5000
CERT_FILE = 'server.crt'  # Path to your SSL certificate
KEY_FILE = 'server.key'   # Path to your SSL private key

def list_songs():
    """Returns a list of (title, artist, filename) tuples from MUSIC_FOLDER."""
    songs = []
    for f in os.listdir(MUSIC_FOLDER):
        if f.endswith((".mp3", ".wav", ".ogg")):
            try:
                title, artist_with_ext = f.split(" - ", 1)
                artist = os.path.splitext(artist_with_ext)[0]
                songs.append((title.strip(), artist.strip(), f))
            except ValueError:
                continue  # Skip improperly named files
    return songs

def handle_client(conn):
    """Handles communication with the client."""
    try:
        pygame.mixer.init()

        while True:
            songs = list_songs()
            if not songs:
                conn.sendall("No songs available.\n".encode())
                return
            # Show song list
            header = f"{'No.':<5}{'Title':<30}{'Artist'}\n"
            song_list = "\n".join(f"{i+1:<5}{title:<30}{artist}" for i, (title,
artist, _) in enumerate(songs))
            conn.sendall(f"Available songs:\n{header}{song_list}\nEnter song
number to play (or 'exit' to quit):\n".encode())

            choice = conn.recv(1024).decode().strip()
            if choice.lower() == "exit":
                conn.sendall("Goodbye!\n".encode())
                break

            if not choice.isdigit() or int(choice) < 1 or int(choice) >
len(songs):
                conn.sendall("Invalid choice.\n".encode())
                continue

            title, artist, filename = songs[int(choice) - 1]
            song_path = os.path.join(MUSIC_FOLDER, filename)

            conn.sendall(f"Playing: {title} by {artist}\n".encode())
            pygame.mixer.music.load(song_path)
            pygame.mixer.music.play()

            conn.sendall("Commands: 'pause', 'resume', 'stop'\n".encode())
            state = "playing"

            while True:
                if state == "playing" and not pygame.mixer.music.get_busy():
                    conn.sendall("Song finished.\n".encode())
                    break

                command = conn.recv(1024).decode().strip().lower()

                if command == "pause":
                    if state == "playing":
                        pygame.mixer.music.pause()
                        state = "paused"
                        conn.sendall("Paused. Type 'resume' to continue.
\n".encode())
                    else:
                        conn.sendall("Already paused or stopped.\n".encode())

                elif command == "resume":
                    if state == "paused":
                        pygame.mixer.music.unpause()
                        state = "playing"
                        conn.sendall("Resumed.\n".encode())
                    else:
                        conn.sendall("Music is not paused.\n".encode())

                elif command == "stop":
                    pygame.mixer.music.stop()
                    state = "stopped"
                    conn.sendall("Stopped.\n".encode())
                    break

                else:
                    conn.sendall("Unknown command. Use 'pause', 'resume', or
'stop'.\n".encode())

    except Exception as e:
        conn.sendall(f"Error: {str(e)}\n".encode())

def start_server():
    """Starts the TCP server with SSL encryption."""
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
        server_socket.bind((HOST, PORT))
        server_socket.listen(5)
        print(f"Server listening on {HOST}:{PORT}")

        # Create default SSL context for the server
        context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
        context.load_cert_chain(certfile=CERT_FILE, keyfile=KEY_FILE)

        # Wrap the server socket with SSL encryption
        server_socket_ssl = context.wrap_socket(server_socket, server_side=True)

        while True:
            conn, addr = server_socket_ssl.accept()
            print(f"Connection from {addr}")
            handle_client(conn)
            conn.close()

if __name__ == "__main__":
    start_server()
```

Input/Output:

Client Side:

```
Available songs:
No.  Title                          Artist
1    Run It Up                      Hanumankind
2    Sultans of Swing               Dire Straits
3    The Shock of The Lightning     Oasis
Enter song number to play (or 'exit' to quit):

Your choice: 2
Playing: Sultans of Swing by Dire Straits

Commands: 'pause', 'resume', 'stop'

Command (pause/resume/stop): pause
Paused. Type 'resume' to continue.

Command (pause/resume/stop): resume
Resumed.

Command (pause/resume/stop): stop
Stopped.

Available songs:
No.  Title                          Artist
1    Run It Up                      Hanumankind
2    Sultans of Swing               Dire Straits
3    The Shock of The Lightning     Oasis
Enter song number to play (or 'exit' to quit):

Your choice: 3
Playing: The Shock of The Lightning by Oasis

Commands: 'pause', 'resume', 'stop'

Command (pause/resume/stop): stop
Stopped.

Available songs:
No.  Title                          Artist
1    Run It Up                      Hanumankind
2    Sultans of Swing               Dire Straits
3    The Shock of The Lightning     Oasis
Enter song number to play (or 'exit' to quit):

Your choice: exit
Goodbye!
```

Server Side:



```
pygame 2.6.1 (SDL 2.28.4, Python 3.11.5)
Hello from the pygame community. https://www.pygame.org/contribute.html
Server listening on 0.0.0.0:5000
Connection from ('192.168.0.111', 55924)
Connection from ('192.168.0.111', 55927)
```