C.S(083)

PROJECT

FILE

2024-2025

(Song Purchase System)

Name: Prerita Varshney

Class: 12th A

Roll No.: 29

Certificate

This is to certify that

PRERITA VARSHNEY

of XII A has accomplished

the C.S project on

‘Song Purchase System’ in year

2024-2025 under guidance

and supervision of Mr. Raj Kumar Pal.

Acknowledgement

I place my sincere thanks to my C.S teacher Mr. Raj Kumar Pal for his guidance and advices to help me complete my work successfully. I would also like to thank my Principal Ms. Meenakshi Sant for providing us all the facilities to finish the project on time.

I take this opportunity to express my deep sense of gratitude for their valuable guidance and constant encouragement. I can not forget to offer my sincere gratitude to my parents who helped me to carry out this project work successfully.

Introduction

The **Song Purchase System** is a software application developed using **Python**, integrated with **MySQL** for database management and **Tkinter** for creating a user-friendly graphical user interface (GUI). This system is designed to facilitate the purchase and management of digital music tracks, providing users with an intuitive platform for browsing, selecting, and purchasing songs.

**Key Components of the System**

1. **Python**:  
   Python serves as the core programming language, offering seamless integration with databases and GUI tools. Its versatility and extensive libraries make it an ideal choice for building applications like this.
2. **MySQL**:  
   MySQL is used as the database management system, storing essential information such as song details, user data and transaction records. Its robust query system ensures quick and reliable data retrieval and storage.
3. **Tkinter**:  
   Tkinter, Python’s standard GUI toolkit, provides an interactive and visually appealing interface. The GUI allows users to interact with the system efficiently, enabling functionalities like browsing songs and confirming purchases.

**Features of the Song Purchase System**

* **User-Friendly Interface**:  
  The system features a well-organized GUI, simplifying navigation and usage.
* **Song Catalog Management**:  
  Users can browse a catalog of songs categorized by genre.
* **Database Integration**:  
  MySQL ensures all transactions and data are logged accurately, allowing for easy retrieval and analysis.

Advantages

The Song Purchase System aims to streamline the process of buying digital music, making it more accessible and efficient for users. It bridges the gap between music providers and consumers through technology, ensuring a smooth and enjoyable experience. Additionally, the system demonstrates the practical integration of Python with database and GUI frameworks, serving as a valuable project for learning and development in software engineering.

The program is also cost effective because Python, MySQL, and Tkinter are open-source and freely available, reducing development costs.

The system is also lightweight and does not require high-end hardware, making it suitable for small businesses or startups.

This project is a robust example of combining programming, database management, and UI design to create a functional application, making it an excellent foundation for developers looking to expand their skills in these domains.

The combination of Python, MySQL, and Tkinter in the Song Purchase System provides a perfect blend of performance, usability, and scalability. It is a cost-effective solution that balances user needs with technical robustness, making it ideal for digital music distribution platforms.

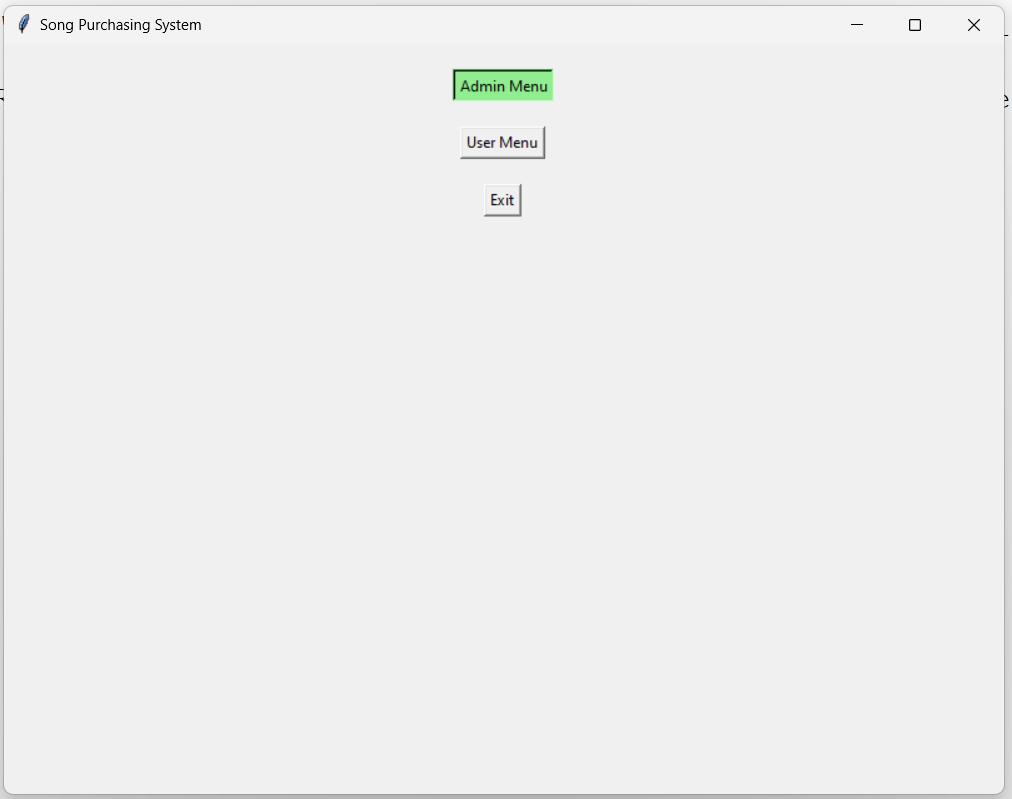
Limitations

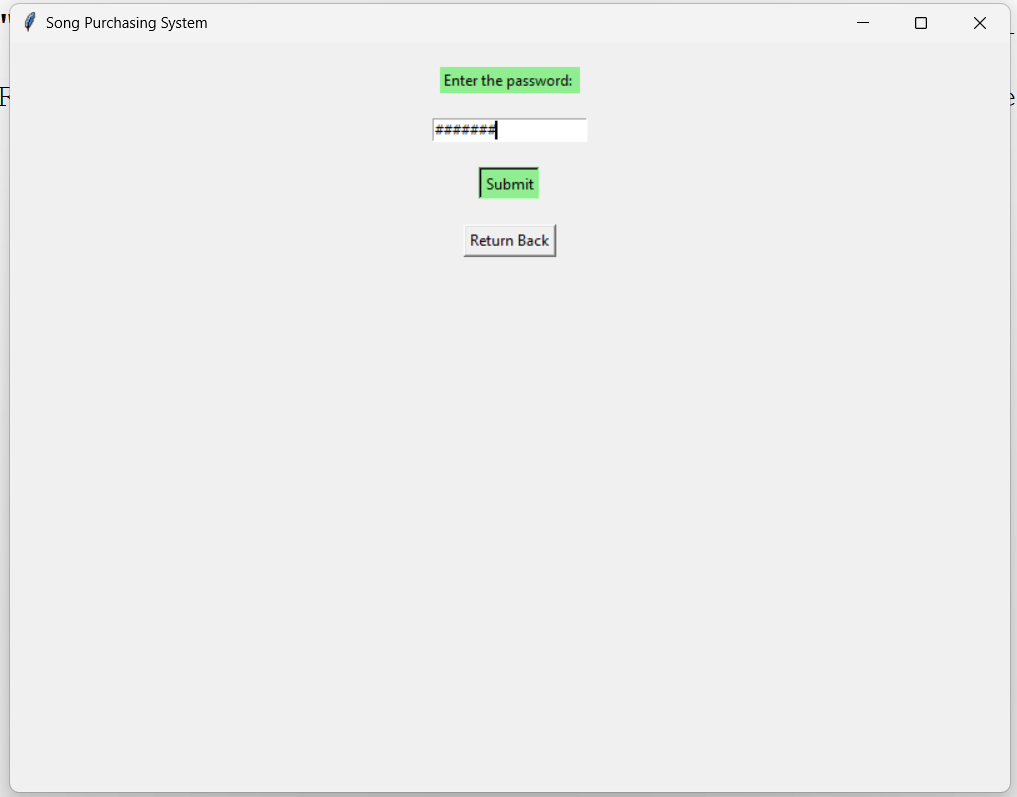
While the Song Purchase System built with Python, MySQL, and Tkinter offers several advantages, it also has certain limitations that may affect its scalability, performance, and usability in more complex scenarios. The limitations are as follows:

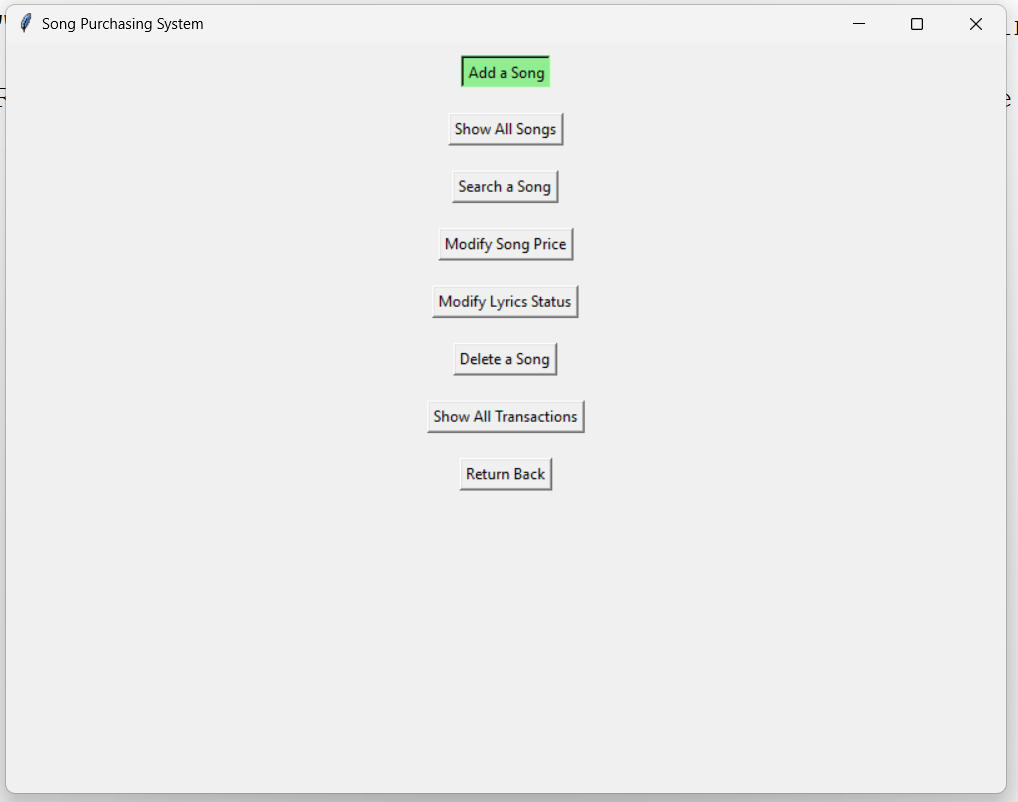
* Tkinter is a simple GUI toolkit, which may not support advanced or modern interface designs commonly expected in commercial applications.
* The system is primarily designed for desktop platforms, limiting accessibility for mobile and web users. Tkinter is not natively supported for mobile app development, which might restrict user reach.
* Python is an interpreted language and may not perform as efficiently as compiled languages like Java or C++ in high-performance applications.
* Deploying the system across multiple machines requires manual installation of Python, MySQL, and the necessary dependencies, which can be time-consuming and error-prone.
* The admin can not view the past cancelled transactions made by the user.

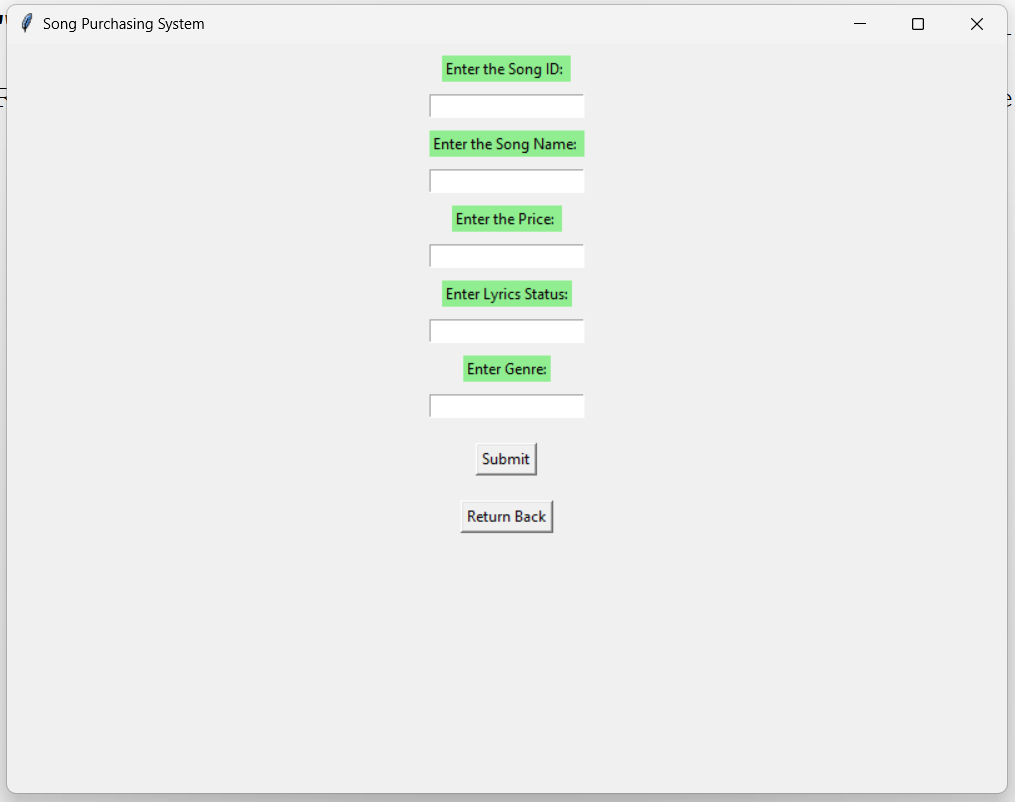
For larger, commercial-grade applications, it would require significant enhancements, including integration with web and mobile platforms, advanced GUI frameworks, cloud scalability, and robust security features.

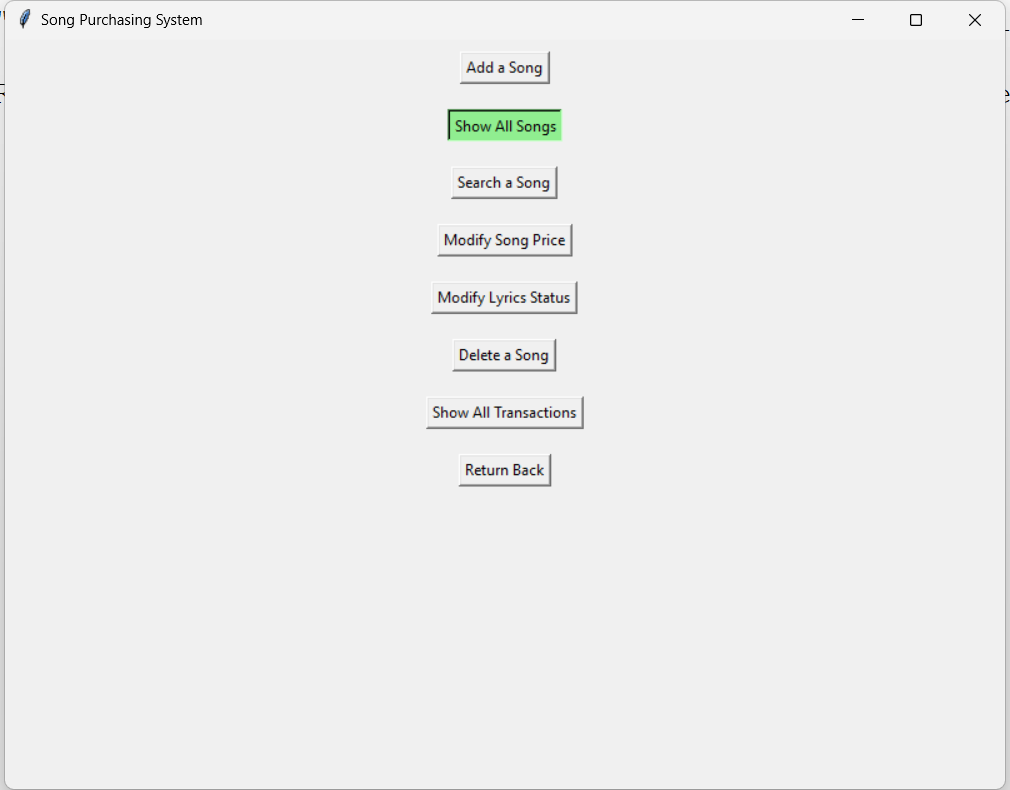
Data Flow Diagram

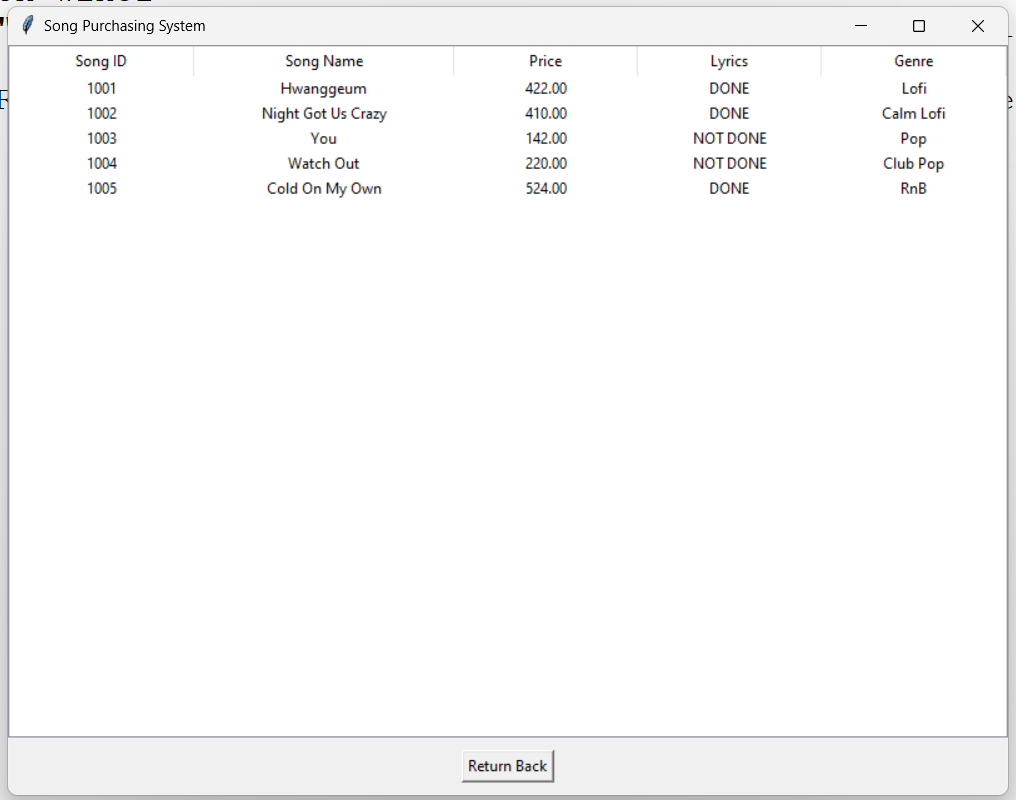


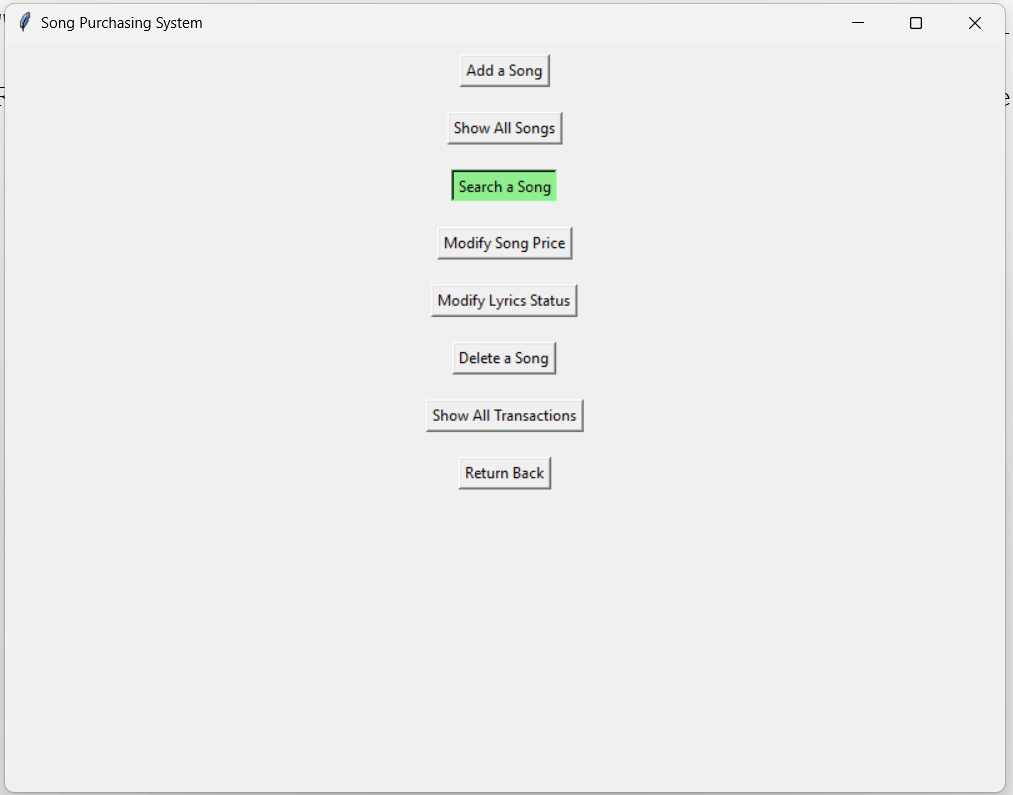


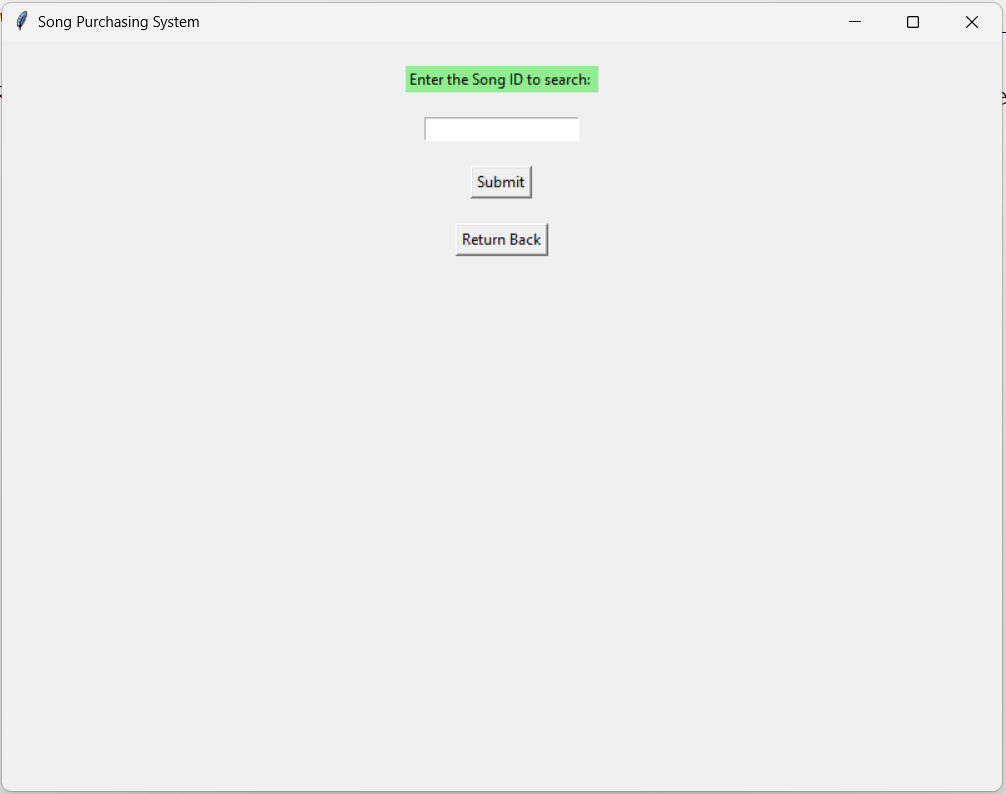


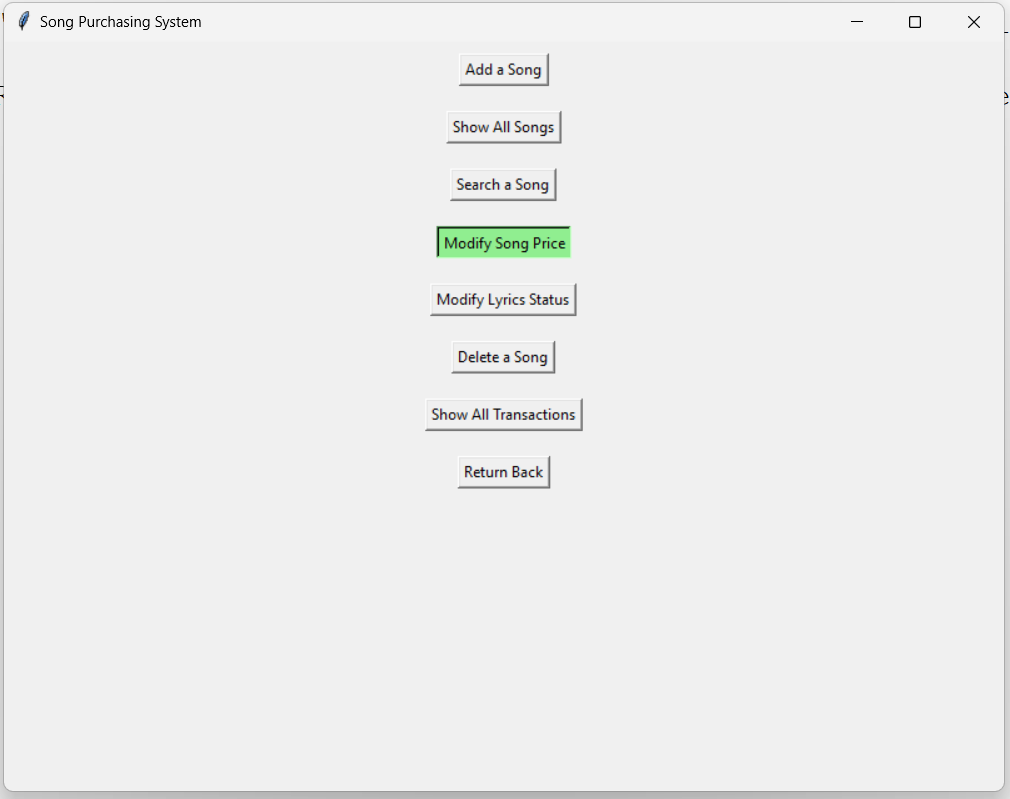


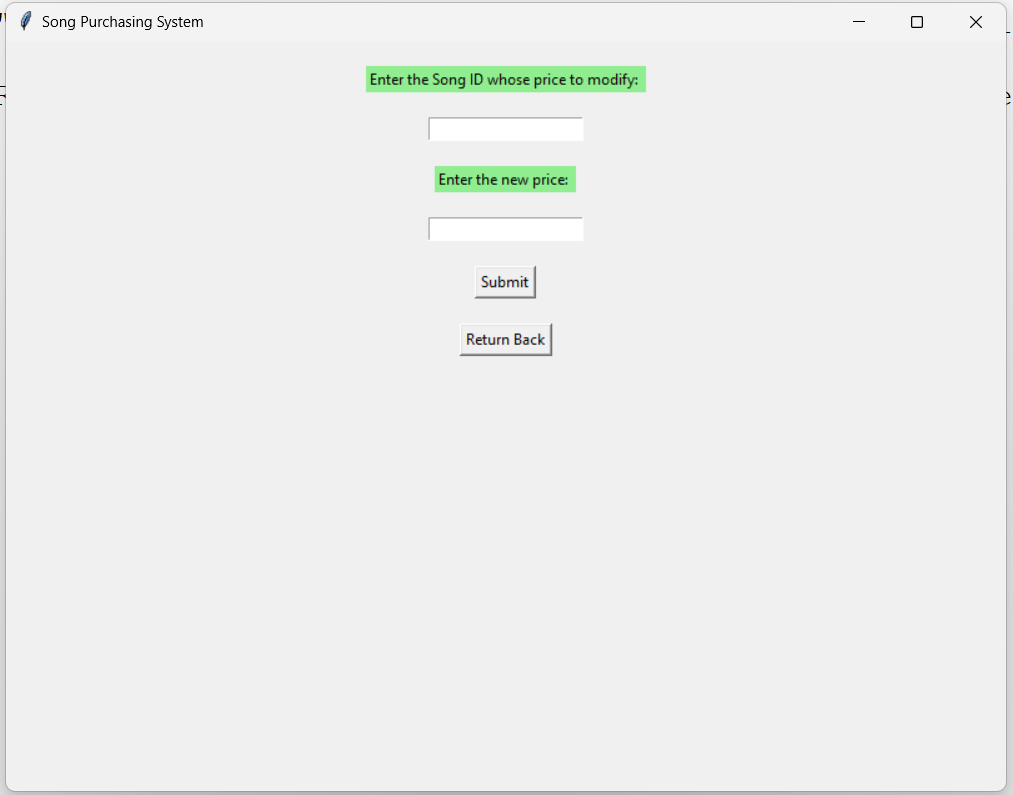


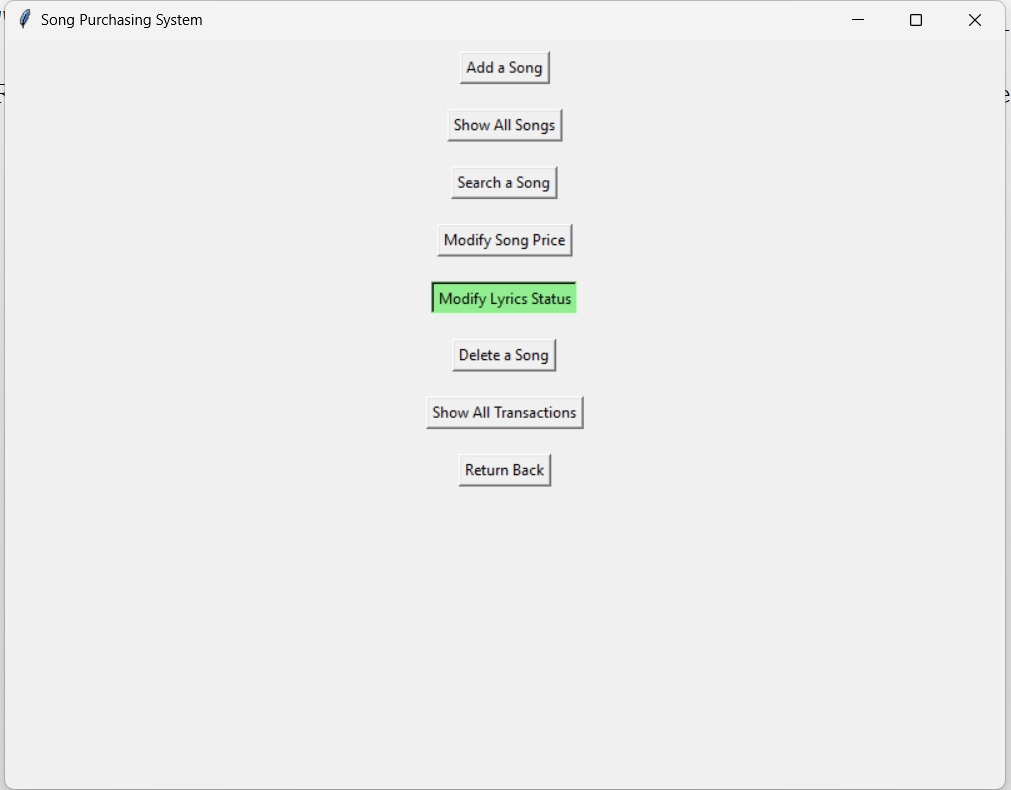


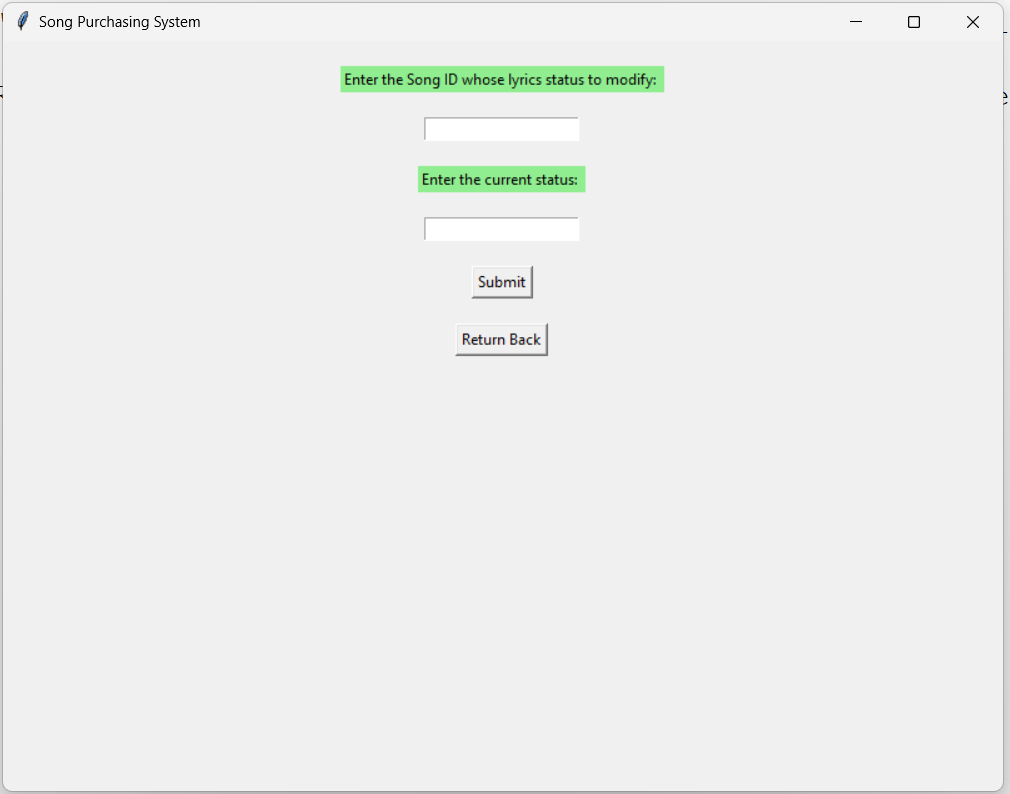


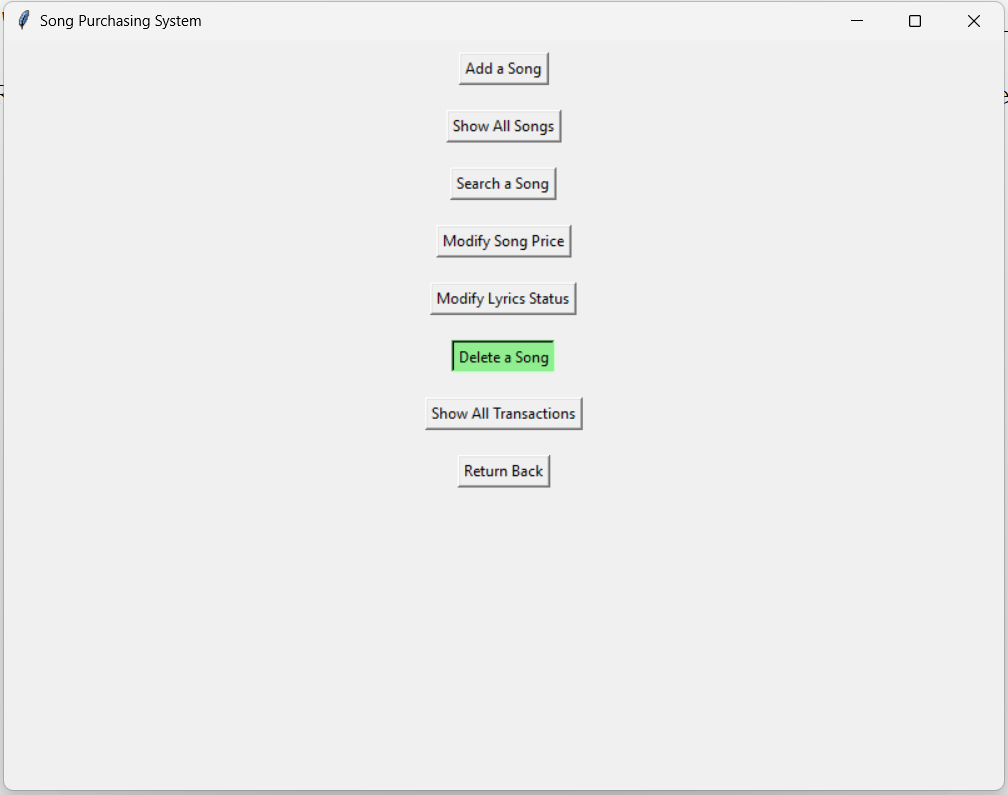


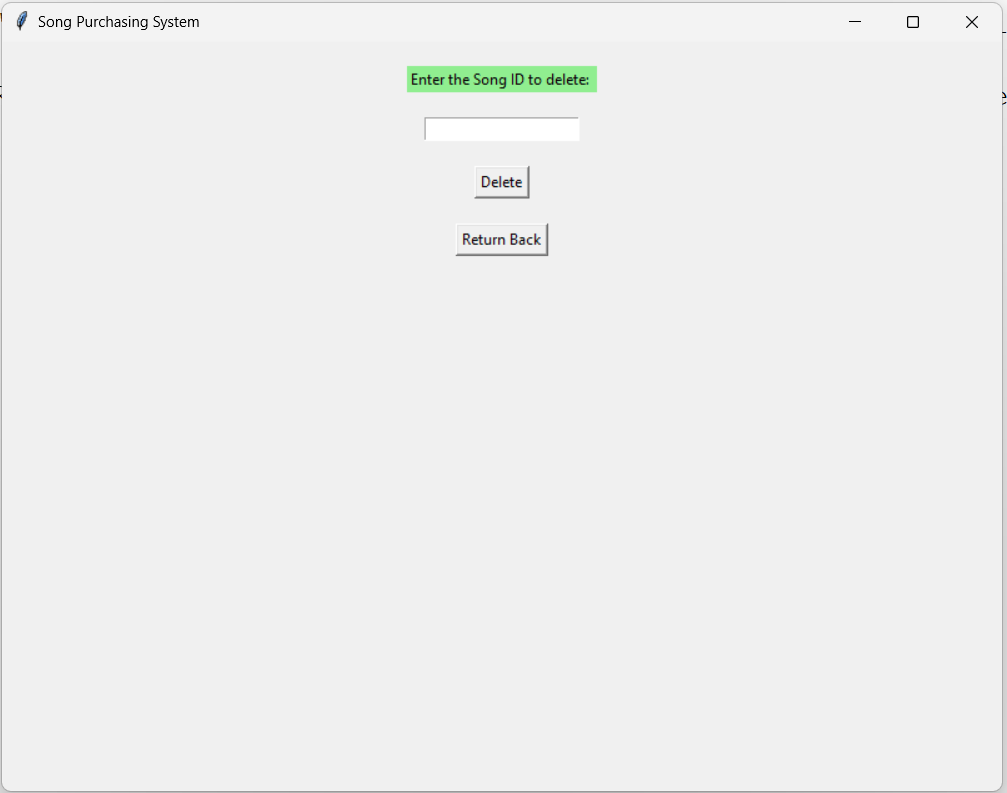


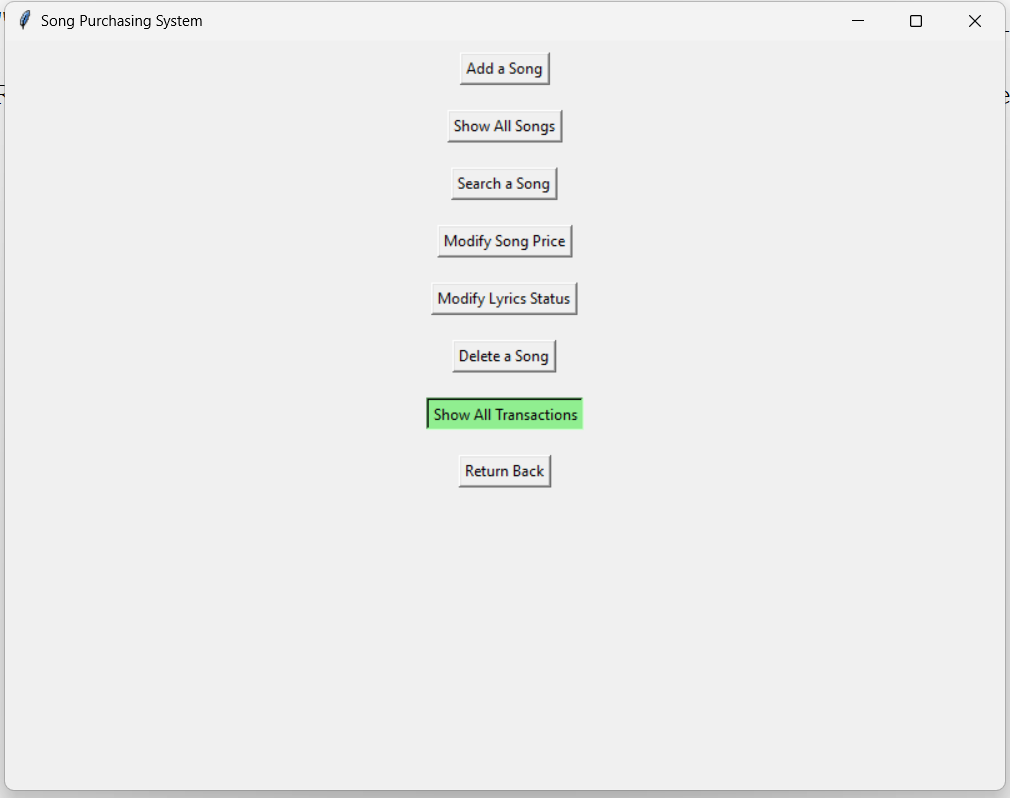


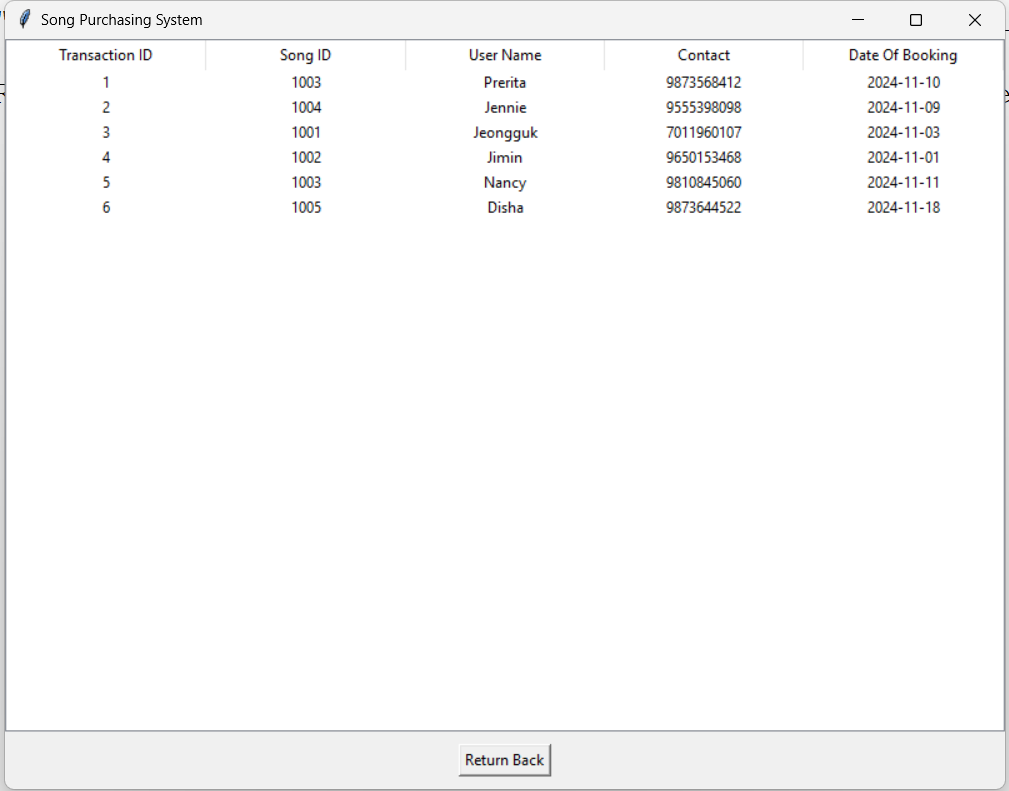


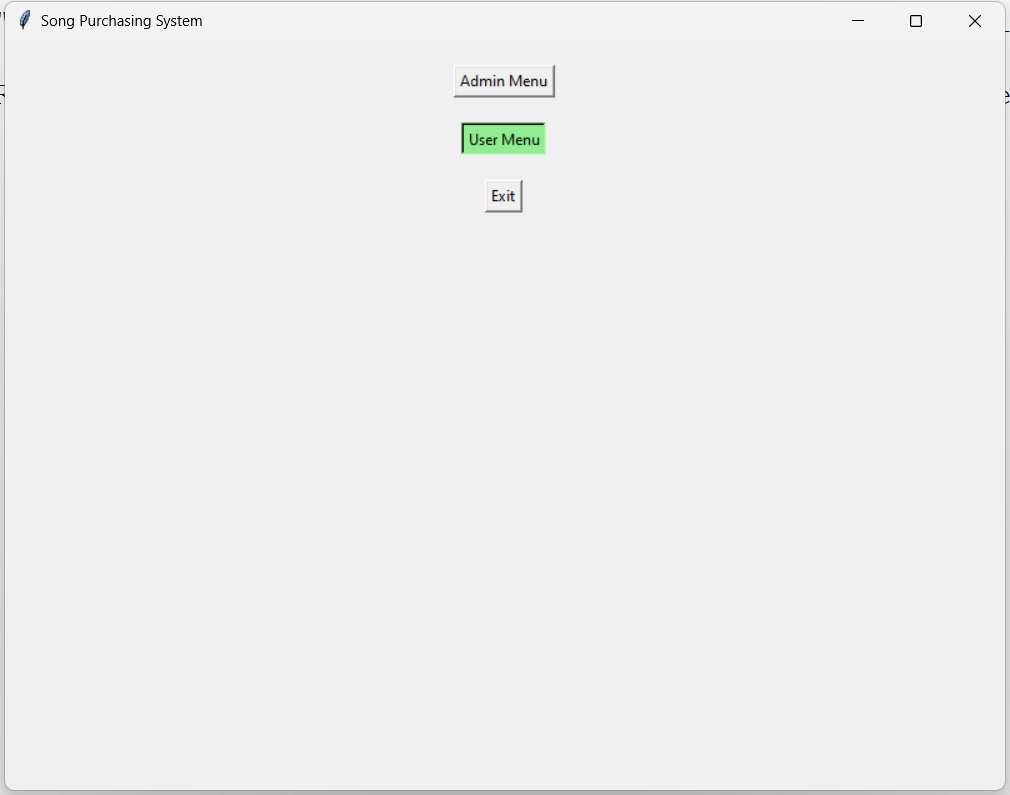


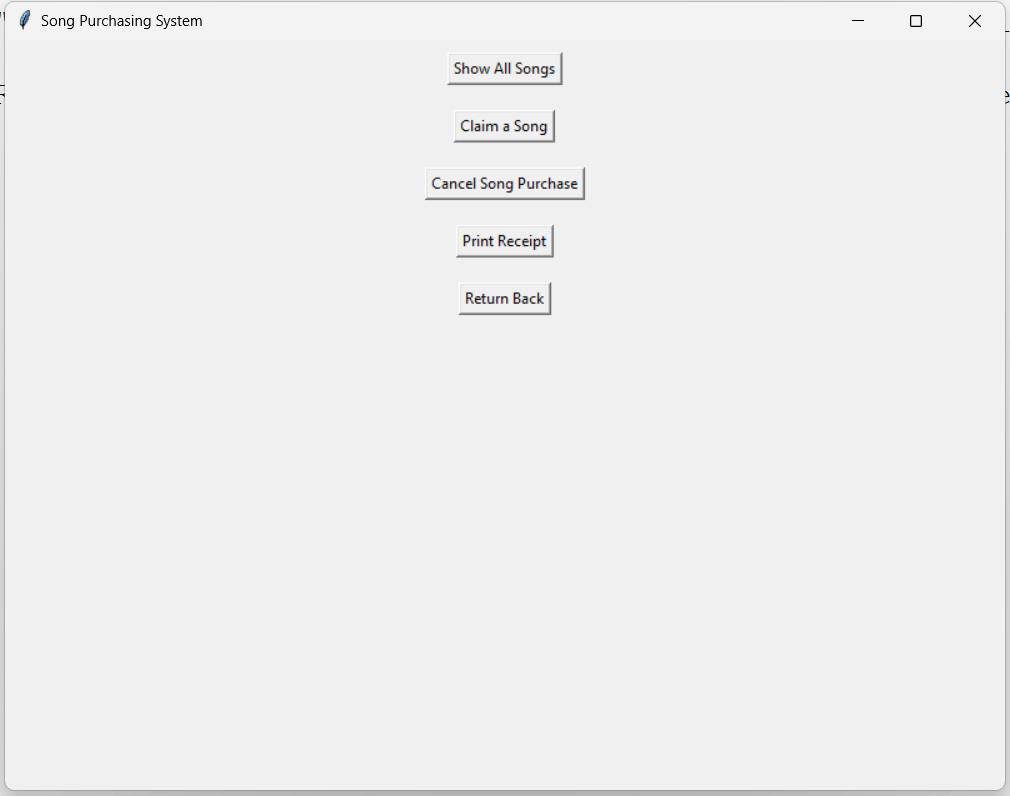


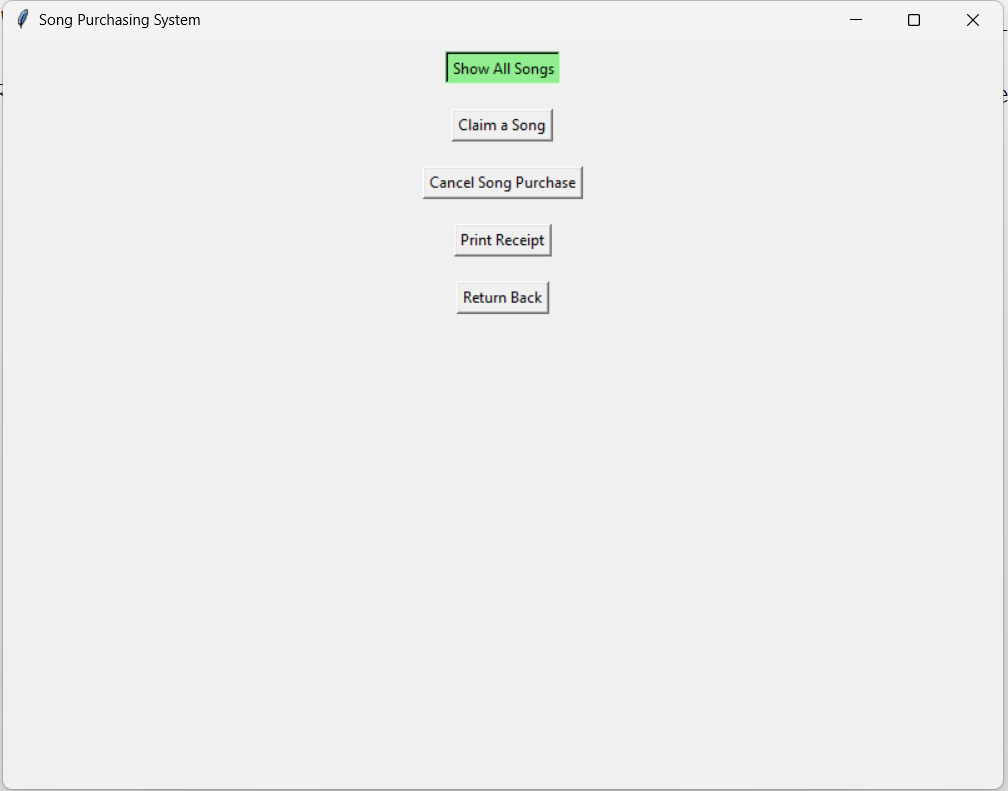


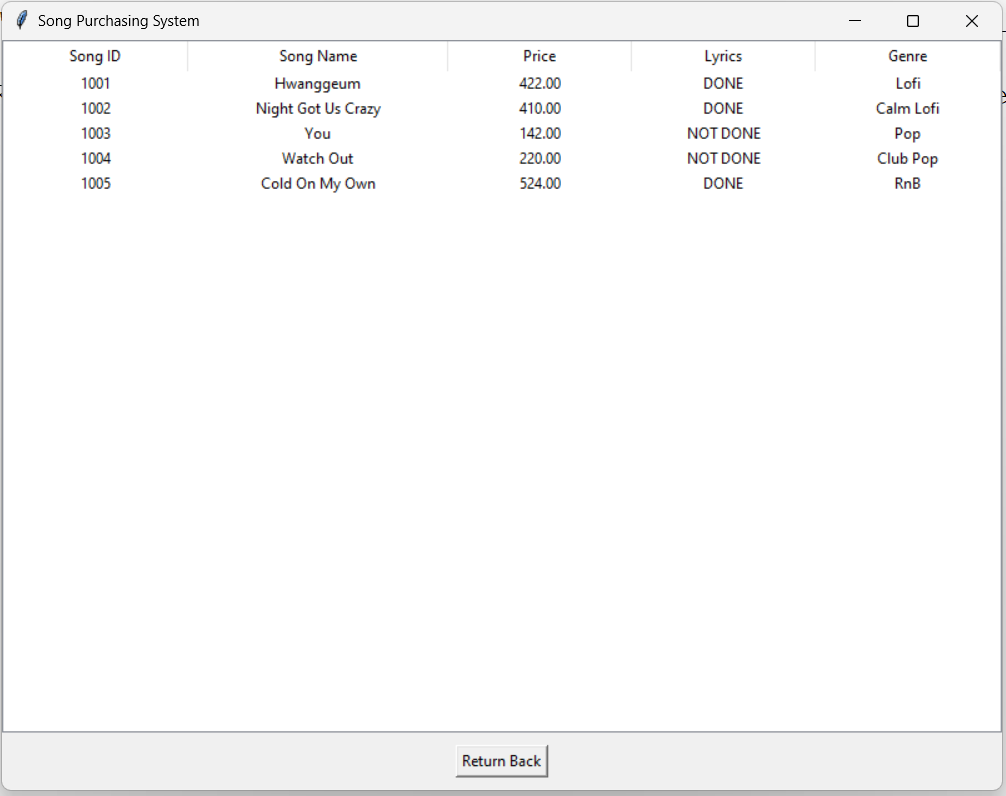


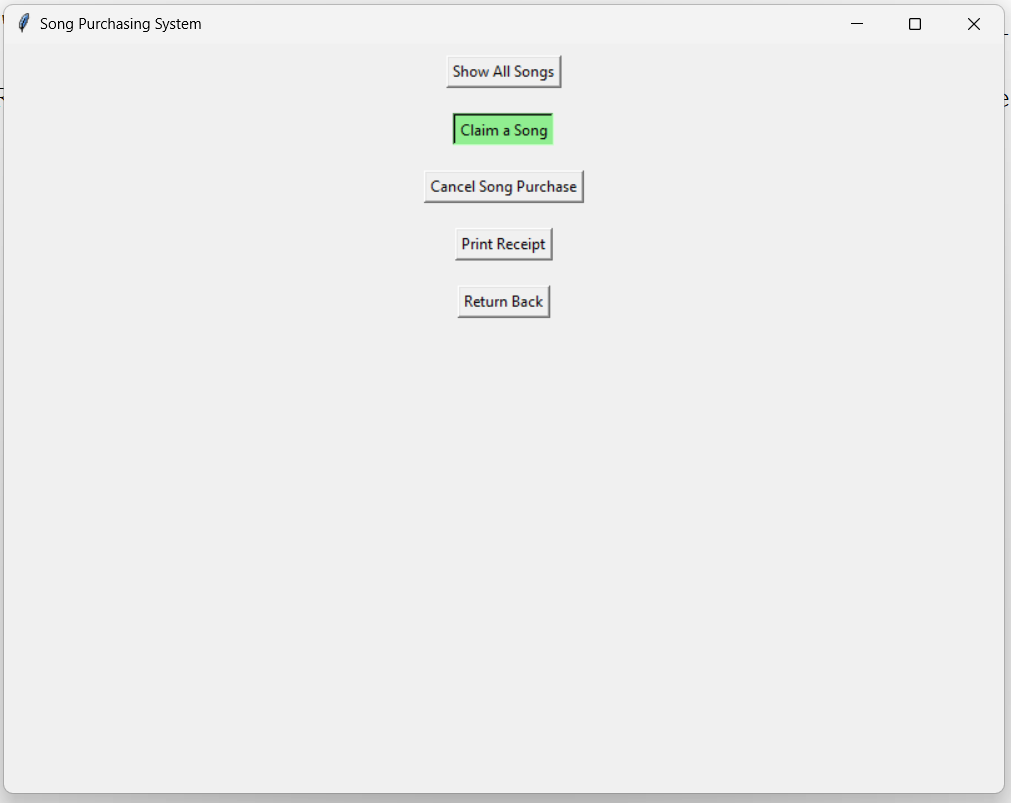


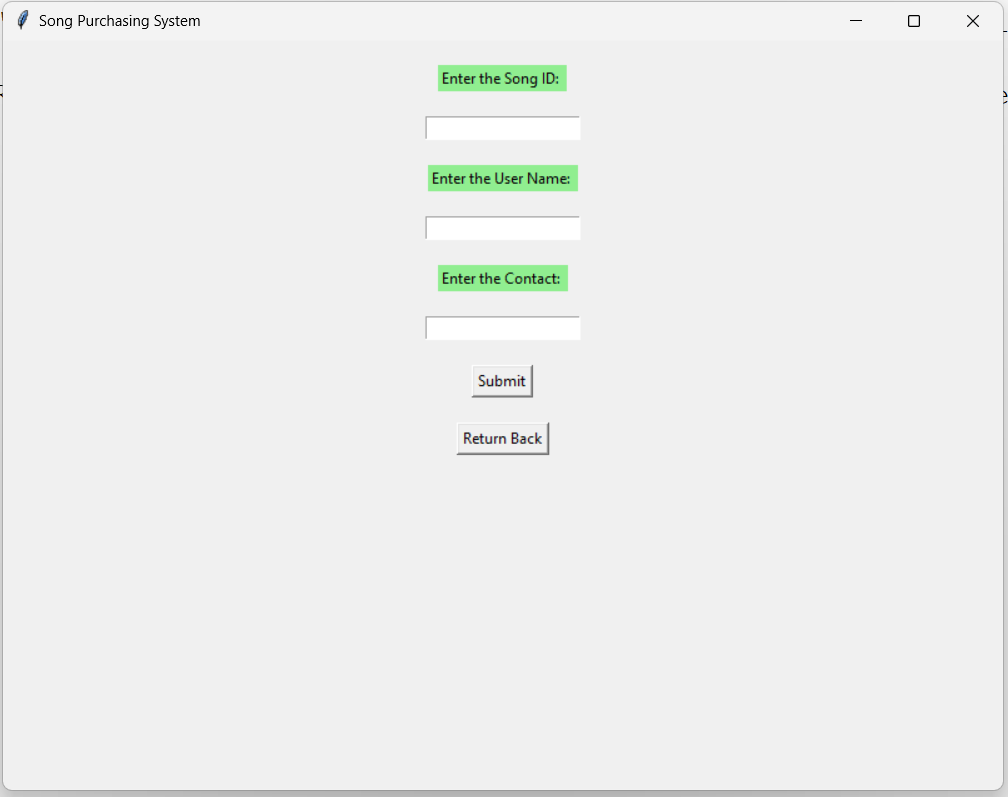


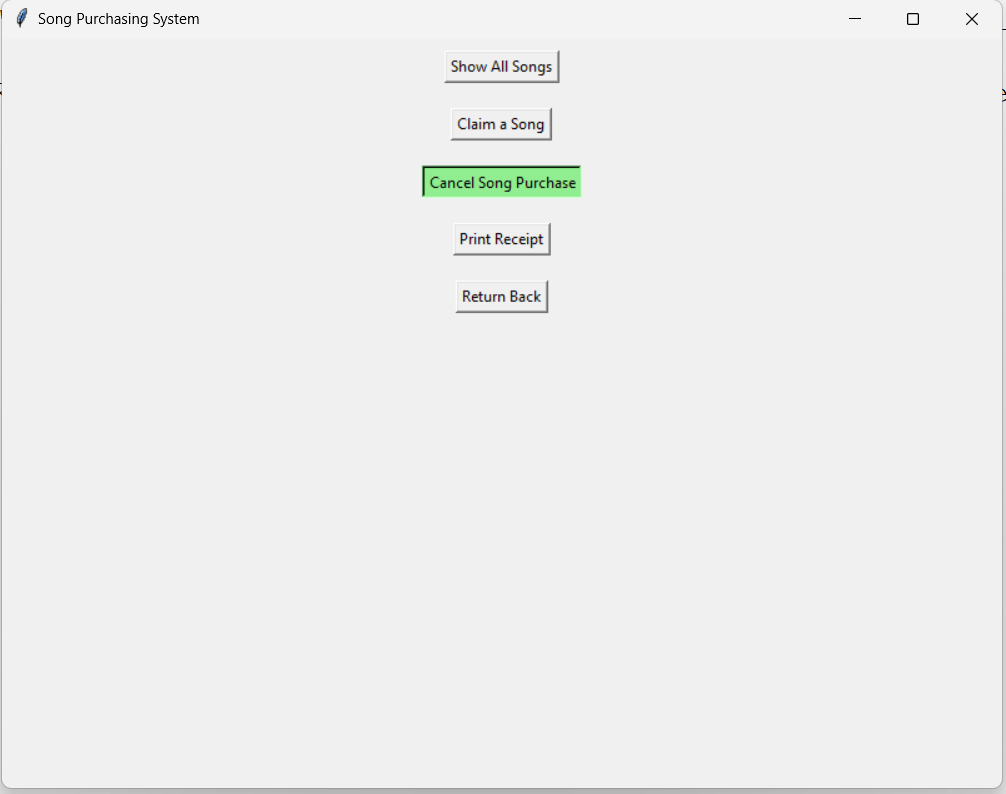


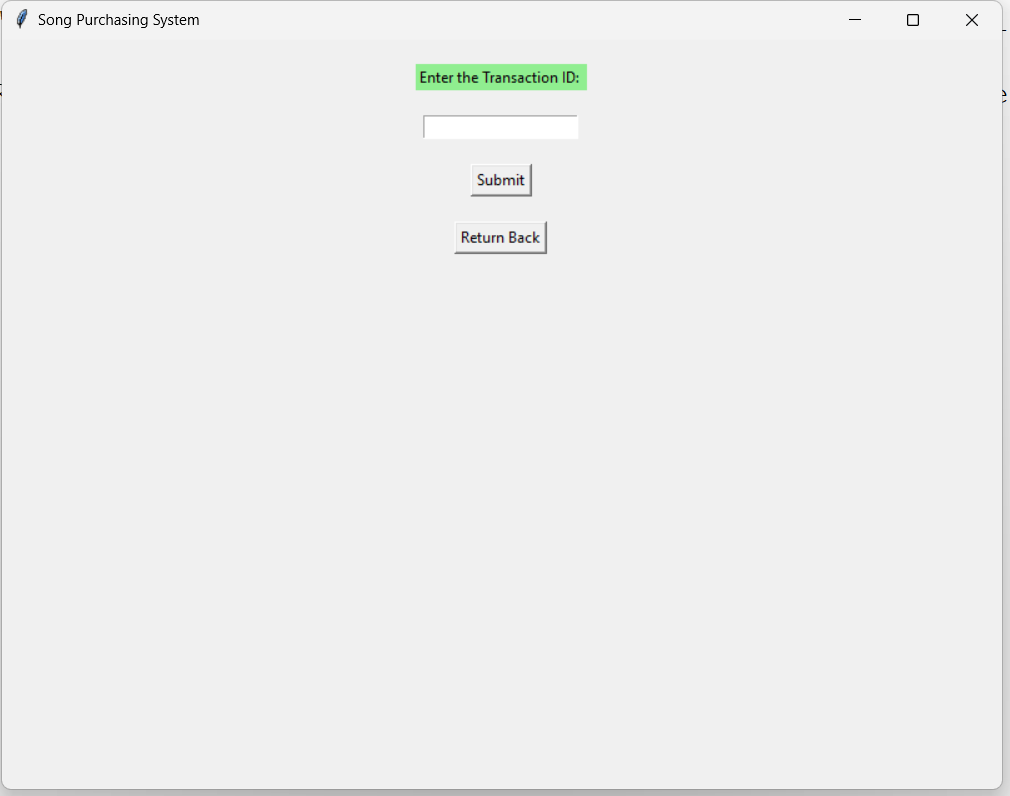


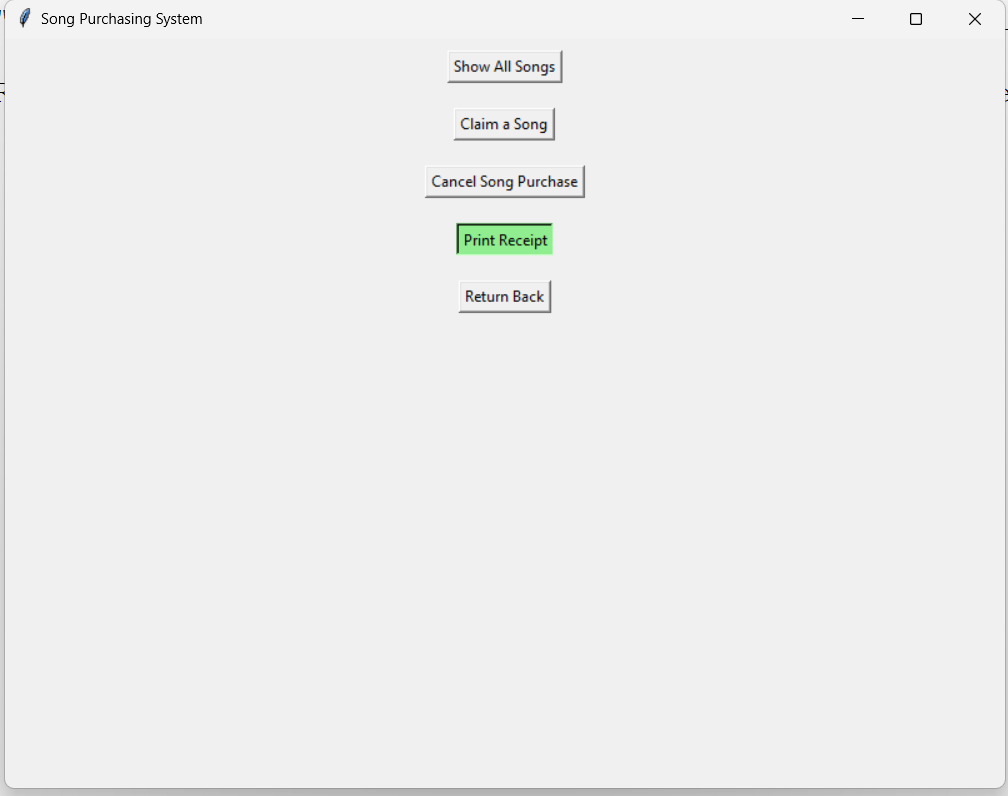


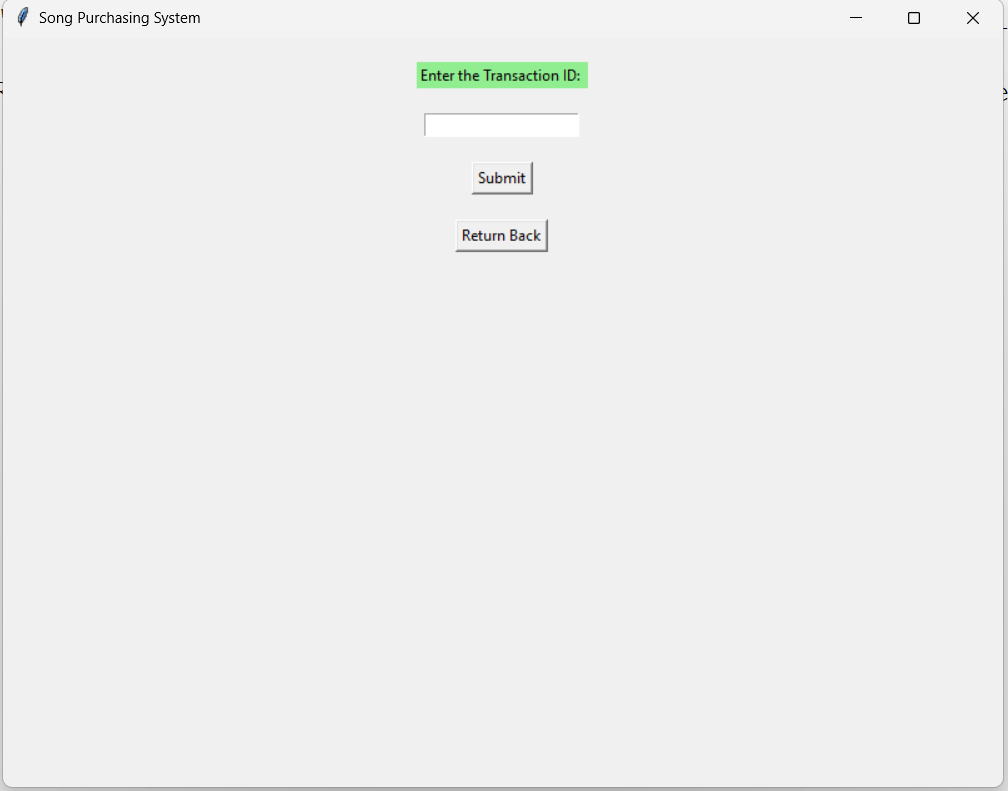












Source Code

from tkinter import \*

from tkinter import messagebox

from tkinter import ttk

import datetime as dt

import mysql.connector as ms

root = Tk()

root.title("Song Purchasing System")

root.geometry("800x600")

conn=ms.connect(host="localhost", user="root", passwd="prerita", charset="utf8", db="songdb")

cur=conn.cursor()

#Main Frames

user\_f = Frame(root)

admin\_f = Frame(root)

#Password for admin

password\_f = Frame(root)

#Admin Frames

add\_song\_f = Frame(root)

show\_all\_songs\_f = Frame(root)

search\_song\_f = Frame(root)

modify\_song\_price\_f = Frame(root)

modify\_lyrics\_status\_f = Frame(root)

delete\_song\_f = Frame(root)

show\_transactions\_f = Frame(root)

#User Frames

show\_all\_songs\_user\_f = Frame(root)

claim\_song\_f = Frame(root)

cancel\_song\_f = Frame(root)

print\_receipt\_f = Frame(root)

#main window

def main\_menu():

admin\_f.pack\_forget()

user\_f.pack\_forget()

password\_f.pack\_forget()

main\_f.pack(pady=10)

#password window

def password\_menu():

main\_f.pack\_forget()

for widget in password\_f.winfo\_children():

widget.destroy()

password\_f.pack(pady=10)

label = Label(password\_f, text="Enter the password: ", bg="lightgreen", fg="black")

label.pack(pady=10)

entry = Entry(password\_f, show="#")

entry.pack(pady=10)

def check\_password():

password = entry.get()

if password == "prerita":

admin\_menu()

else:

messagebox.showinfo("Error", "Incorrect Password")

submit\_button = Button(password\_f, text="Submit", activebackground="lightgreen", activeforeground="black", command=check\_password)

submit\_button.pack(pady=10)

button5 = Button(password\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=main\_menu)

button5.pack(pady=10)

#admin menu

def admin\_menu():

main\_f.pack\_forget()

for widget in admin\_f.winfo\_children():

widget.destroy()

password\_f.pack\_forget()

for widget in password\_f.winfo\_children():

widget.destroy()

add\_song\_f.pack\_forget()

for widget in add\_song\_f.winfo\_children():

widget.destroy()

show\_all\_songs\_f.pack\_forget()

for widget in show\_all\_songs\_f.winfo\_children():

widget.destroy()

search\_song\_f.pack\_forget()

for widget in search\_song\_f.winfo\_children():

widget.destroy()

modify\_song\_price\_f.pack\_forget()

for widget in modify\_song\_price\_f.winfo\_children():

widget.destroy()

modify\_lyrics\_status\_f.pack\_forget()

for widget in modify\_lyrics\_status\_f.winfo\_children():

widget.destroy()

delete\_song\_f.pack\_forget()

for widget in delete\_song\_f.winfo\_children():

widget.destroy()

show\_transactions\_f.pack\_forget()

for widget in show\_transactions\_f.winfo\_children():

widget.destroy()

button1 = Button(admin\_f, text="Add a Song", activebackground="lightgreen", activeforeground="black", command=add\_song)

button1.pack(pady=10)

button2 = Button(admin\_f, text="Show All Songs", activebackground="lightgreen", activeforeground="black", command=show\_all\_songs)

button2.pack(pady=10)

button3 = Button(admin\_f, text="Search a Song", activebackground="lightgreen", activeforeground="black", command=search\_song)

button3.pack(pady=10)

button4 = Button(admin\_f, text="Modify Song Price", activebackground="lightgreen", activeforeground="black", command=modify\_song\_price)

button4.pack(pady=10)

button5 = Button(admin\_f, text="Modify Lyrics Status", activebackground="lightgreen", activeforeground="black", command=modify\_lyrics\_status)

button5.pack(pady=10)

button6 = Button(admin\_f, text="Delete a Song", activebackground="lightgreen", activeforeground="black", command=delete\_song)

button6.pack(pady=10)

button7 = Button(admin\_f, text="Show All Transactions", activebackground="lightgreen", activeforeground="black", command=show\_transactions)

button7.pack(pady=10)

button8 = Button(admin\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=main\_menu)

button8.pack(pady=10)

admin\_f.pack(expand=True, fill="both")

#user menu

def user\_menu():

main\_f.pack\_forget()

for widget in user\_f.winfo\_children():

widget.destroy()

show\_all\_songs\_user\_f.pack\_forget()

for widget in show\_all\_songs\_user\_f.winfo\_children():

widget.destroy()

claim\_song\_f.pack\_forget()

for widget in claim\_song\_f.winfo\_children():

widget.destroy()

cancel\_song\_f.pack\_forget()

for widget in cancel\_song\_f.winfo\_children():

widget.destroy()

print\_receipt\_f.pack\_forget()

for widget in print\_receipt\_f.winfo\_children():

widget.destroy()

button0 = Button(user\_f, text="Show All Songs", activebackground="lightgreen", activeforeground="black", command=show\_all\_songs\_user)

button0.pack(pady=10)

button1 = Button(user\_f, text="Claim a Song", activebackground="lightgreen", activeforeground="black", command=claim\_song)

button1.pack(pady=10)

button2 = Button(user\_f, text="Cancel Song Purchase", activebackground="lightgreen", activeforeground="black", command=cancel\_song)

button2.pack(pady=10)

button3 = Button(user\_f, text="Print Receipt", activebackground="lightgreen", activeforeground="black", command=print\_receipt)

button3.pack(pady=10)

button4 = Button(user\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=main\_menu)

button4.pack(pady=10)

user\_f.pack(expand=True, fill="both")

#exit function

def exit\_program():

cur.close()

conn.close()

root.destroy()

#add song

def add\_song():

admin\_f.pack\_forget()

for widget in admin\_f.winfo\_children():

widget.destroy()

add\_song\_f.pack(pady=10)

label\_sid = Label(add\_song\_f, text="Enter the Song ID: ", bg="lightgreen", fg="black")

label\_sid.pack()

entry\_sid = Entry(add\_song\_f)

entry\_sid.pack(pady=10)

label\_sn = Label(add\_song\_f, text="Enter the Song Name: ", bg="lightgreen", fg="black")

label\_sn.pack()

entry\_sn = Entry(add\_song\_f)

entry\_sn.pack(pady=10)

label\_price = Label(add\_song\_f, text="Enter the Price: ", bg="lightgreen", fg="black")

label\_price.pack()

entry\_price = Entry(add\_song\_f)

entry\_price.pack(pady=10)

label\_lyrics = Label(add\_song\_f, text="Enter Lyrics Status:", bg="lightgreen", fg="black")

label\_lyrics.pack()

entry\_lyrics = Entry(add\_song\_f)

entry\_lyrics.pack(pady=10)

label\_genre = Label(add\_song\_f, text="Enter Genre:", bg="lightgreen", fg="black")

label\_genre.pack()

entry\_genre = Entry(add\_song\_f)

entry\_genre.pack(pady=10)

def add\_song\_db():

sid = int(entry\_sid.get())

sn = entry\_sn.get()

price = float(entry\_price.get())

lyrics = entry\_lyrics.get()

genre = entry\_genre.get()

sql = "INSERT INTO song VALUES(%s, %s, %s, %s, %s)"

val = (sid, sn, price, lyrics, genre)

cur.execute(sql, val)

conn.commit()

messagebox.showinfo("Success", "Song added successfully")

submit\_button = Button(add\_song\_f, text="Submit", activebackground="lightgreen", activeforeground="black", command=add\_song\_db)

submit\_button.pack(pady=10)

button5 = Button(add\_song\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=admin\_menu)

button5.pack(pady=10)

#show all songs

def show\_all\_songs():

admin\_f.pack\_forget()

for widget in show\_all\_songs\_f.winfo\_children():

widget.destroy()

show\_all\_songs\_f.pack(expand=True, fill="both")

sql = "SELECT \* FROM song"

cur.execute(sql)

res = cur.fetchall()

tree = ttk.Treeview(show\_all\_songs\_f)

tree['columns'] = ("Song ID", "Song Name", "Price", "Lyrics", "Genre")

tree.column("#0", width=0, stretch=NO)

tree.column("Song ID", anchor=CENTER, width=50)

tree.column("Song Name", anchor=CENTER, width=110)

tree.column("Price", anchor=CENTER, width=50)

tree.column("Lyrics", anchor=CENTER, width=50)

tree.column("Genre", anchor=CENTER, width=50)

tree.heading("#0", text="", anchor=CENTER)

tree.heading("Song ID", text="Song ID", anchor=CENTER)

tree.heading("Song Name", text="Song Name", anchor=CENTER)

tree.heading("Price", text="Price", anchor=CENTER)

tree.heading("Lyrics", text="Lyrics", anchor=CENTER)

tree.heading("Genre", text="Genre", anchor=CENTER)

if not res:

messagebox.showinfo("Error", "No songs found")

else:

for i in res:

tree.insert(parent="", index="end", text="", values=i)

tree.pack(fill="both", expand=True)

button5=Button(show\_all\_songs\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=admin\_menu)

button5.pack(pady=10)

#search song

def search\_song():

admin\_f.pack\_forget()

for widget in admin\_f.winfo\_children():

widget.destroy()

search\_song\_f.pack(pady=10)

label\_sid = Label(search\_song\_f, text="Enter the Song ID to search: ", bg="lightgreen", fg="black")

label\_sid.pack(pady=10)

entry\_sid = Entry(search\_song\_f)

entry\_sid.pack(pady=10)

def search\_song\_db():

sid = entry\_sid.get()

sql = "select \* from song where sid = %s"

val = (sid,)

cur.execute(sql, val)

res = cur.fetchall()

if not res:

messagebox.showinfo("Error", "Song not found")

else:

for i in res:

song\_details = f"Song Details:\nSong ID: {i[0]}\nSong Name: {i[1]}\nPrice: {i[2]}\nLyrics: {i[3]}\nGenre: {i[4]}"

label\_song\_details = Label(search\_song\_f)

label\_song\_details.pack(pady=10)

label\_song\_details.config(text=song\_details)

submit\_button = Button(search\_song\_f, text="Submit", activebackground="lightgreen", activeforeground="black", command=search\_song\_db)

submit\_button.pack(pady=10)

button5 = Button(search\_song\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=admin\_menu)

button5.pack(pady=10)

#modify song price

def modify\_song\_price():

admin\_f.pack\_forget()

for widget in admin\_f.winfo\_children():

widget.destroy()

modify\_song\_price\_f.pack(pady=10)

label\_sid = Label(modify\_song\_price\_f, text="Enter the Song ID whose price to modify: ", bg="lightgreen", fg="black")

label\_sid.pack(pady=10)

entry\_sid = Entry(modify\_song\_price\_f)

entry\_sid.pack(pady=10)

label\_nf = Label(modify\_song\_price\_f, text="Enter the new price: ", bg="lightgreen", fg="black")

label\_nf.pack(pady=10)

entry\_nf = Entry(modify\_song\_price\_f)

entry\_nf.pack(pady=10)

def modify\_song\_price\_db():

sid = entry\_sid.get()

new\_price = float(entry\_nf.get())

sql = "select \* from song where sid = %s"

val = (sid,)

cur.execute(sql, val)

res = cur.fetchall()

if not res:

messagebox.showinfo("Error", "Song not found")

else:

song\_details = f"\nOld Record\nSong ID: {res[0][0]}\nSong Name: {res[0][1]}\nPrice: {res[0][2]}\nLyrics: {res[0][3]}\nGenre: {res[0][4]}"

label\_song\_details = Label(modify\_song\_price\_f)

label\_song\_details.pack(pady=10)

label\_song\_details.config(text=song\_details)

sql = "update song set price = %s where sid = %s"

val = (new\_price, sid)

cur.execute(sql, val)

conn.commit()

sql = "select \* from song where sid = %s"

val = (sid,)

cur.execute(sql, val)

res = cur.fetchall()

song\_details = f"\n\n\nNew Record\nSong ID: {res[0][0]}\nSong Name: {res[0][1]}\nPrice: {res[0][2]}\nLyrics: {res[0][3]}\nGenre: {res[0][4]}"

label\_song\_details = Label(modify\_song\_price\_f)

label\_song\_details.pack(pady=10)

label\_song\_details.config(text=song\_details)

submit\_button = Button(modify\_song\_price\_f, text="Submit", activebackground="lightgreen", activeforeground="black", command=modify\_song\_price\_db)

submit\_button.pack(pady=10)

button5 = Button(modify\_song\_price\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=admin\_menu)

button5.pack(pady=10)

#modify lyrics status

def modify\_lyrics\_status():

admin\_f.pack\_forget()

for widget in admin\_f.winfo\_children():

widget.destroy()

modify\_lyrics\_status\_f.pack(pady=10)

label\_sid = Label(modify\_lyrics\_status\_f, text="Enter the Song ID whose lyrics status to modify: ", bg="lightgreen", fg="black")

label\_sid.pack(pady=10)

entry\_sid = Entry(modify\_lyrics\_status\_f)

entry\_sid.pack(pady=10)

label\_lf = Label(modify\_lyrics\_status\_f, text="Enter the current status: ", bg="lightgreen", fg="black")

label\_lf.pack(pady=10)

entry\_lf = Entry(modify\_lyrics\_status\_f)

entry\_lf.pack(pady=10)

def modify\_lyrics\_status\_db():

sid = entry\_sid.get()

status = entry\_lf.get()

sql = "select \* from song where sid = %s"

val = (sid,)

cur.execute(sql, val)

res = cur.fetchall()

if not res:

messagebox.showinfo("Error", "Song not found")

else:

song\_details = f"\nOld Record\nSong ID: {res[0][0]}\nSong Name: {res[0][1]}\nPrice: {res[0][2]}\nLyrics: {res[0][3]}\nGenre: {res[0][4]}"

label\_song\_details = Label(modify\_lyrics\_status\_f)

label\_song\_details.pack(pady=10)

label\_song\_details.config(text=song\_details)

sql = "update song set lyrics = %s where sid = %s"

val = (status, sid)

cur.execute(sql, val)

conn.commit()

sql = "select \* from song where sid = %s"

val = (sid,)

cur.execute(sql, val)

res = cur.fetchall()

song\_details = f"\n\n\nNew Record\nSong ID: {res[0][0]}\nSong Name: {res[0][1]}\nPrice: {res[0][2]}\nLyrics: {res[0][3]}\nGenre: {res[0][4]}"

label\_song\_details = Label(modify\_lyrics\_status\_f)

label\_song\_details.pack(pady=10)

label\_song\_details.config(text=song\_details)

submit\_button = Button(modify\_lyrics\_status\_f, text="Submit", activebackground="lightgreen", activeforeground="black", command=modify\_lyrics\_status\_db)

submit\_button.pack(pady=10)

button5 = Button(modify\_lyrics\_status\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=admin\_menu)

button5.pack(pady=10)

#delete song

def delete\_song():

admin\_f.pack\_forget()

for widget in admin\_f.winfo\_children():

widget.destroy()

delete\_song\_f.pack(pady=10)

label\_sid = Label(delete\_song\_f, text="Enter the Song ID to delete: ", bg="lightgreen", fg="black")

label\_sid.pack(pady=10)

entry\_sid = Entry(delete\_song\_f)

entry\_sid.pack(pady=10)

def delete\_song\_db():

sid = entry\_sid.get()

sql = "select \* from song where sid = %s"

val = (sid,)

cur.execute(sql, val)

res = cur.fetchall()

if not res:

messagebox.showinfo("Error", "Song not found")

else:

confirm = messagebox.askyesno("Confirm", "Do you want to delete the song?")

if not confirm:

admin\_menu()

return

sql = "delete from song where sid = %s"

cur.execute(sql, val)

conn.commit()

messagebox.showinfo("Success", "Song deleted successfully")

button\_delete = Button(delete\_song\_f, text="Delete", activebackground="lightgreen", activeforeground="black", command=delete\_song\_db)

button5 = Button(delete\_song\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=admin\_menu)

button\_delete.pack(pady=10)

button5.pack(pady=10)

#show transactions

def show\_transactions():

admin\_f.pack\_forget()

for widget in admin\_f.winfo\_children():

widget.destroy()

show\_transactions\_f.pack(expand=True, fill="both")

sql = "SELECT \* FROM transaction"

cur.execute(sql)

res = cur.fetchall()

tree = ttk.Treeview(show\_transactions\_f)

tree["columns"] = ("Trans ID", "Song ID", "User Name", "Contact", "Date Of Booking")

tree.column("#0", width=0, stretch=NO)

tree.column("Trans ID", anchor=CENTER, width=100)

tree.column("Song ID", anchor=CENTER, width=100)

tree.column("User Name", anchor=CENTER, width=100)

tree.column("Contact", anchor=CENTER, width=100)

tree.column("Date Of Booking", anchor=CENTER, width=100)

tree.heading("#0", text="", anchor=CENTER)

tree.heading("Trans ID", text="Transaction ID", anchor=CENTER)

tree.heading("Song ID", text="Song ID", anchor=CENTER)

tree.heading("User Name", text="User Name", anchor=CENTER)

tree.heading("Contact", text="Contact", anchor=CENTER)

tree.heading("Date Of Booking", text="Date Of Booking", anchor=CENTER)

if not res:

messagebox.showinfo("Error", "No transactions found")

else:

for i in res:

tree.insert(parent="", index="end", text="", values=i)

tree.pack(fill="both", expand=True)

button5=Button(show\_transactions\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=admin\_menu)

button5.pack(pady=10)

#show all songs for user

def show\_all\_songs\_user():

user\_f.pack\_forget()

for widget in show\_all\_songs\_user\_f.winfo\_children():

widget.destroy()

show\_all\_songs\_user\_f.pack(expand=True, fill="both")

sql = "SELECT \* FROM song"

cur.execute(sql)

res = cur.fetchall()

tree = ttk.Treeview(show\_all\_songs\_user\_f)

tree['columns'] = ("Song ID", "Song Name", "Price", "Lyrics", "Genre")

tree.column("#0", width=0, stretch=NO)

tree.column("Song ID", anchor=CENTER, width=50)

tree.column("Song Name", anchor=CENTER, width=110)

tree.column("Price", anchor=CENTER, width=50)

tree.column("Lyrics", anchor=CENTER, width=50)

tree.column("Genre", anchor=CENTER, width=50)

tree.heading("#0", text="", anchor=CENTER)

tree.heading("Song ID", text="Song ID", anchor=CENTER)

tree.heading("Song Name", text="Song Name", anchor=CENTER)

tree.heading("Price", text="Price", anchor=CENTER)

tree.heading("Lyrics", text="Lyrics", anchor=CENTER)

tree.heading("Genre", text="Genre", anchor=CENTER)

if not res:

messagebox.showinfo("Error", "No songs found")

else:

for i in res:

tree.insert(parent="", index="end", text="", values=i)

tree.pack(fill="both", expand=True)

button5=Button(show\_all\_songs\_user\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=user\_menu)

button5.pack(pady=10)

#claim song

def claim\_song():

user\_f.pack\_forget()

for widget in user\_f.winfo\_children():

widget.destroy()

claim\_song\_f.pack(pady=10)

label\_sid = Label(claim\_song\_f, text="Enter the Song ID: ", bg="lightgreen", fg="black")

label\_sid.pack(pady=10)

entry\_sid = Entry(claim\_song\_f)

entry\_sid.pack(pady=10)

label\_un = Label(claim\_song\_f, text="Enter the User Name: ", bg="lightgreen", fg="black")

label\_un.pack(pady=10)

entry\_un = Entry(claim\_song\_f)

entry\_un.pack(pady=10)

label\_contact = Label(claim\_song\_f, text="Enter the Contact: ", bg="lightgreen", fg="black")

label\_contact.pack(pady=10)

entry\_contact = Entry(claim\_song\_f)

entry\_contact.pack(pady=10)

def claim\_song\_db():

sid = entry\_sid.get()

un = entry\_un.get()

contact = entry\_contact.get()

if (len(contact) != 10):

messagebox.showinfo("Error", "Invalid contact number")

return

dob = dt.datetime.now().strftime("%Y-%m-%d")

sql = "SELECT \* FROM song WHERE sid = %s"

val = (sid,)

cur.execute(sql, val)

res = cur.fetchall()

if not res:

messagebox.showinfo("Error", "Song not found")

else:

song\_details = f"Song Name: {res[0][1]}\nPrice: {res[0][2]}"

label\_song\_details = Label(claim\_song\_f)

label\_song\_details.pack(pady=10)

label\_song\_details.config(text=song\_details)

confirm = messagebox.askyesno("Confirm", "Do you want to claim the song?")

if not confirm:

user\_menu()

return

sql = "INSERT INTO transaction(sid, pname, contact, dob) VALUES(%s, %s, %s, %s)"

val = (sid, un, contact, dob)

cur.execute(sql, val)

conn.commit()

sql = "SELECT tid FROM transaction WHERE sid = %s and pname = %s and contact = %s"

val = (sid, un, contact)

cur.execute(sql, val)

res = cur.fetchall()

messagebox.showinfo("Success", "Claimed song successfully. Transaction ID: "+ str(res[0][0]))

submit\_button = Button(claim\_song\_f, text="Submit", activebackground="lightgreen", activeforeground="black", command=claim\_song\_db)

submit\_button.pack(pady=10)

button5 = Button(claim\_song\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=user\_menu)

button5.pack(pady=10)

#cancel song

def cancel\_song():

user\_f.pack\_forget()

for widget in user\_f.winfo\_children():

widget.destroy()

cancel\_song\_f.pack(pady=10)

label\_tid = Label(cancel\_song\_f, text="Enter the Transaction ID: ", bg="lightgreen", fg="black")

label\_tid.pack(pady=10)

entry\_tid = Entry(cancel\_song\_f)

entry\_tid.pack(pady=10)

def cancel\_song\_db():

tid = entry\_tid.get()

sql = "SELECT \* FROM transaction WHERE tid = %s"

val = (tid,)

cur.execute(sql, val)

res = cur.fetchall()

if not res:

messagebox.showinfo("Error", "Transaction not found")

else:

confirm = messagebox.askyesno("Confirm", "Do you want to cancel the ticket?")

if not confirm:

user\_menu()

return

sql = "DELETE FROM transaction WHERE tid = %s"

val = (tid,)

cur.execute(sql, val)

conn.commit()

messagebox.showinfo("Success", "Song purchase cancelled successfully")

submit\_button = Button(cancel\_song\_f, text="Submit", activebackground="lightgreen", activeforeground="black", command=cancel\_song\_db)

submit\_button.pack(pady=10)

button5 = Button(cancel\_song\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=user\_menu)

button5.pack(pady=10)

#print receipt

def print\_receipt():

user\_f.pack\_forget()

for widget in user\_f.winfo\_children():

widget.destroy()

print\_receipt\_f.pack(pady=10)

label\_tid = Label(print\_receipt\_f, text="Enter the Transaction ID: ", bg="lightgreen", fg="black")

label\_tid.pack(pady=10)

entry\_tid = Entry(print\_receipt\_f)

entry\_tid.pack(pady=10)

def print\_it():

tid = entry\_tid.get()

sql = "SELECT \* FROM transaction WHERE tid = %s"

val = (tid,)

cur.execute(sql, val)

res = cur.fetchall()

if not res:

messagebox.showinfo("Error", "Transaction not found")

else:

sql = "SELECT t.tid, s.sname, t.pname, t.contact, t.dob FROM song as s, transaction as t WHERE tid = %s and t.sid = s.sid"

val = (tid,)

cur.execute(sql, val)

res = cur.fetchall()

label\_tid = Label(print\_receipt\_f, text="Transaction ID: "+str(res[0][0]))

label\_tid.pack(pady=10)

label\_sname = Label(print\_receipt\_f, text="Song Name: "+str(res[0][1]))

label\_sname.pack(pady=10)

label\_uname = Label(print\_receipt\_f, text="User Name: "+str(res[0][2]))

label\_uname.pack(pady=10)

label\_contact = Label(print\_receipt\_f, text="Contact: "+str(res[0][3]))

label\_contact.pack(pady=10)

label\_dob = Label(print\_receipt\_f, text="Date of Booking: "+str(res[0][4]))

label\_dob.pack(pady=10)

submit\_button = Button(print\_receipt\_f, text="Submit", activebackground="lightgreen", activeforeground="black", command=print\_it)

submit\_button.pack(pady=10)

button5 = Button(print\_receipt\_f, text="Return Back", activebackground="lightgreen", activeforeground="black", command=user\_menu)

button5.pack(pady=10)

main\_f=Frame(root)

main\_f.pack(side="top", expand=True, fill="both")

button\_create = Button(main\_f, text="Admin Menu", activebackground="lightgreen", activeforeground="black", command=password\_menu)

button\_create.pack(pady=10)

button\_create = Button(main\_f, text="User Menu", activebackground="lightgreen", activeforeground="black", command=user\_menu)

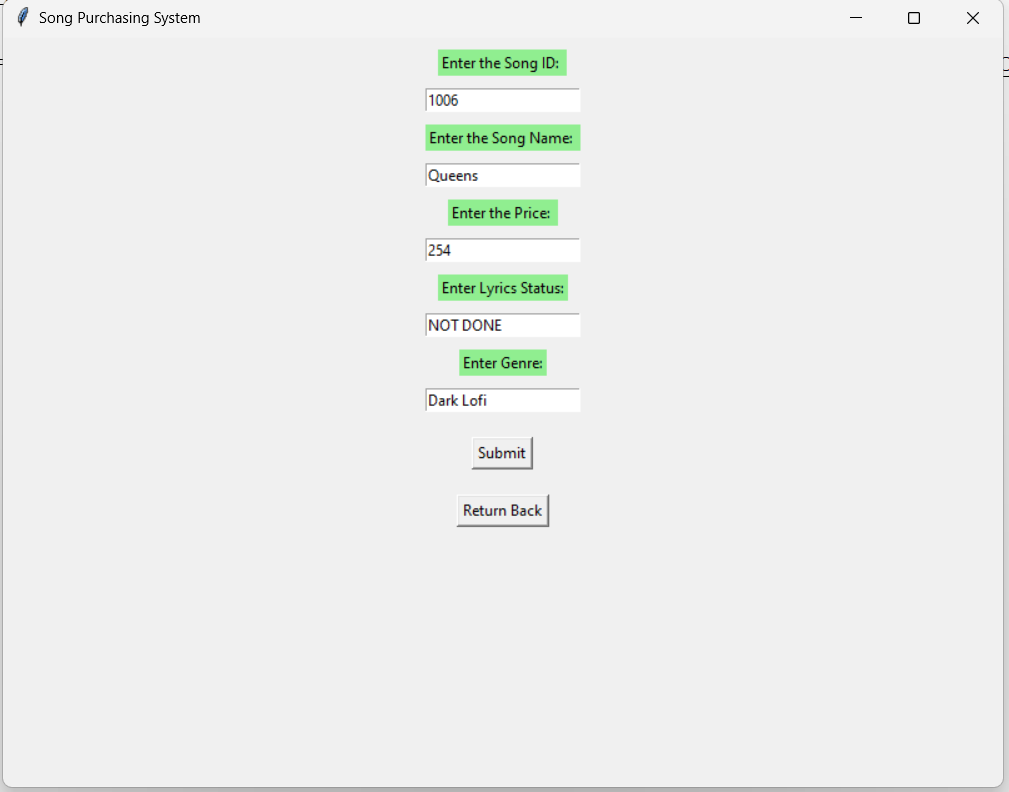
button\_create.pack(pady=10, anchor=CENTER)

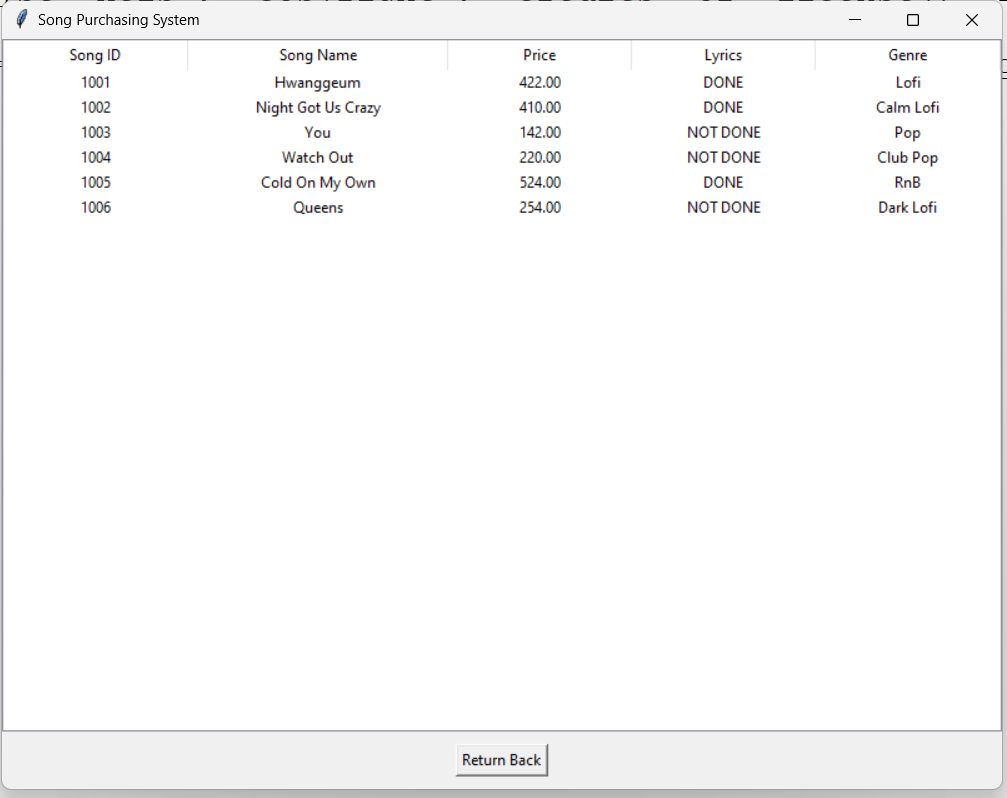
button\_exit = Button(main\_f, text="Exit", activebackground="lightgreen", activeforeground="black", command=exit\_program)

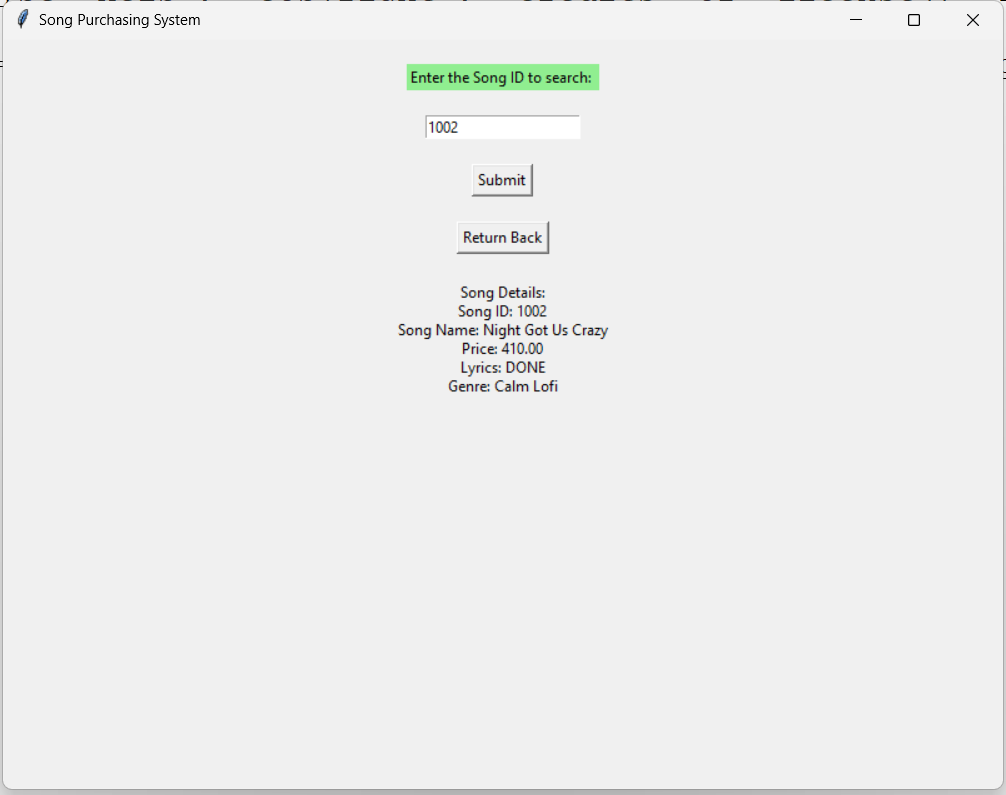
button\_exit.pack(pady=10, anchor=CENTER)

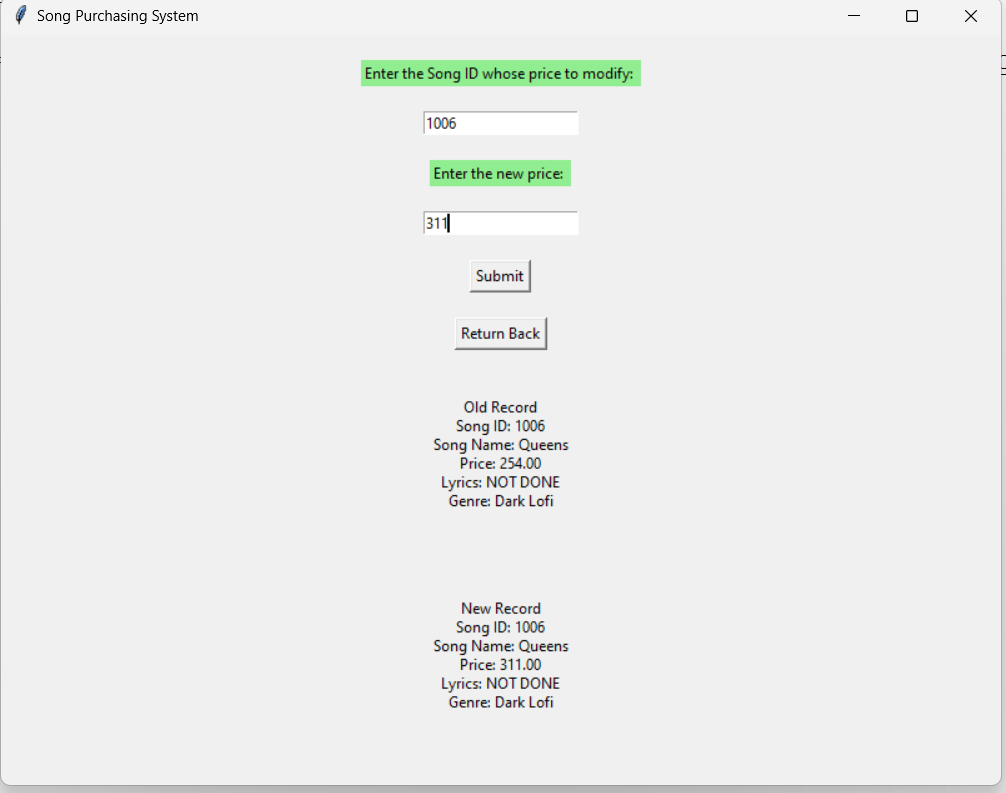
root.mainloop()

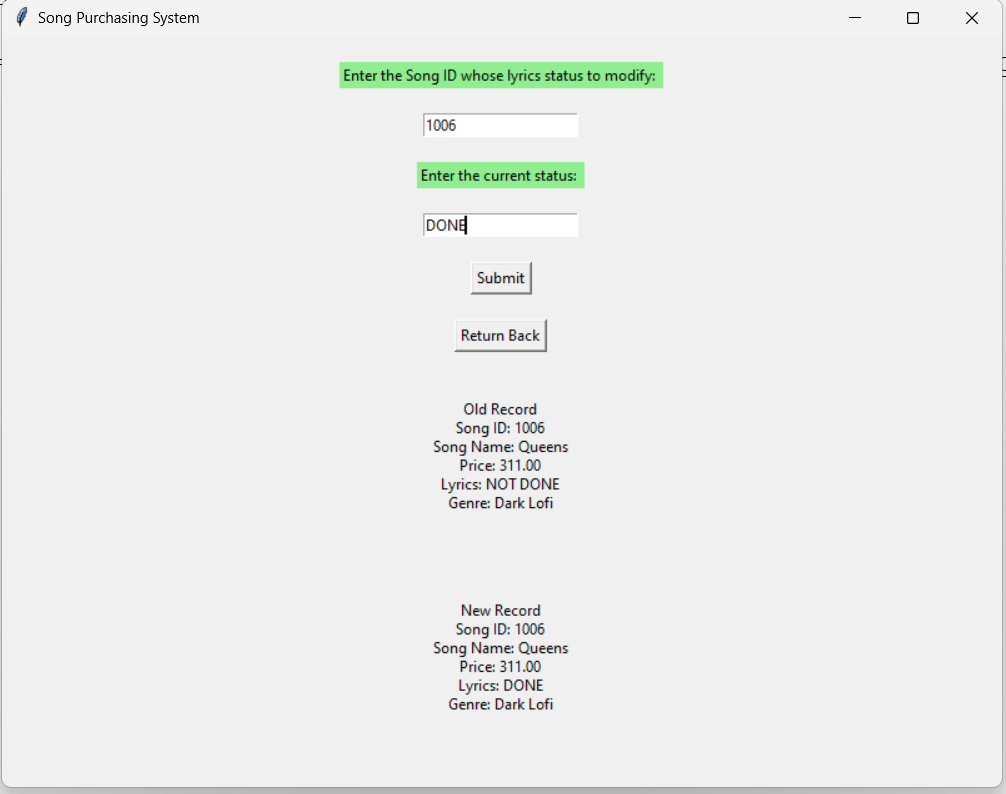
Output

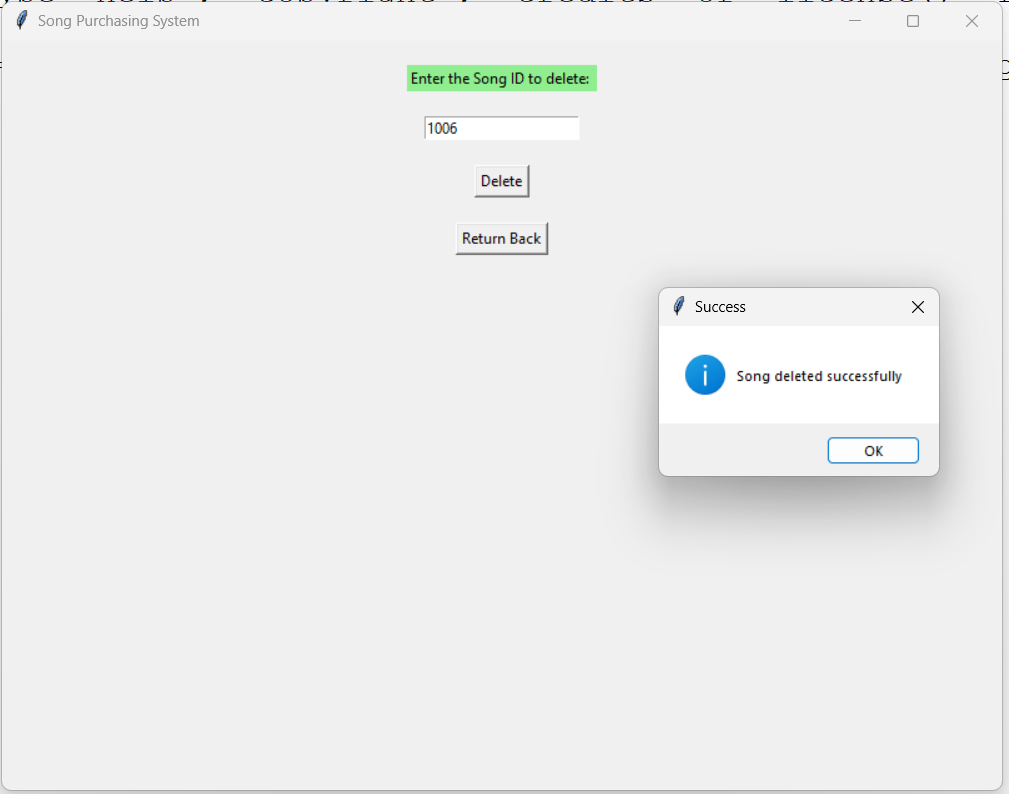


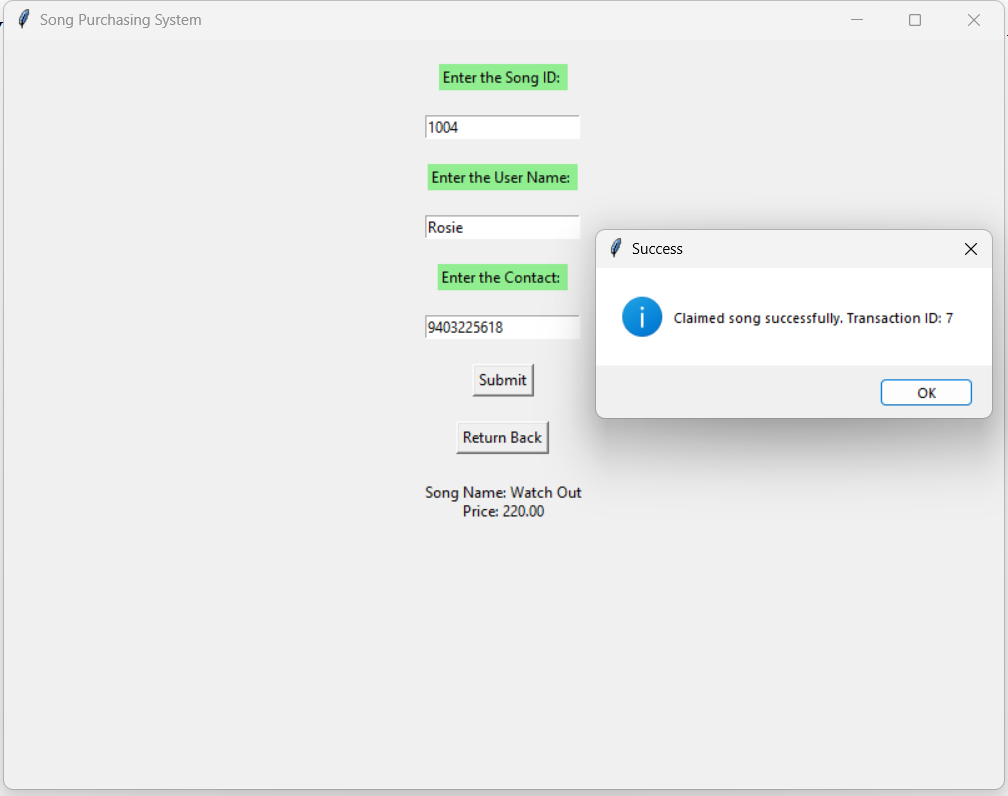


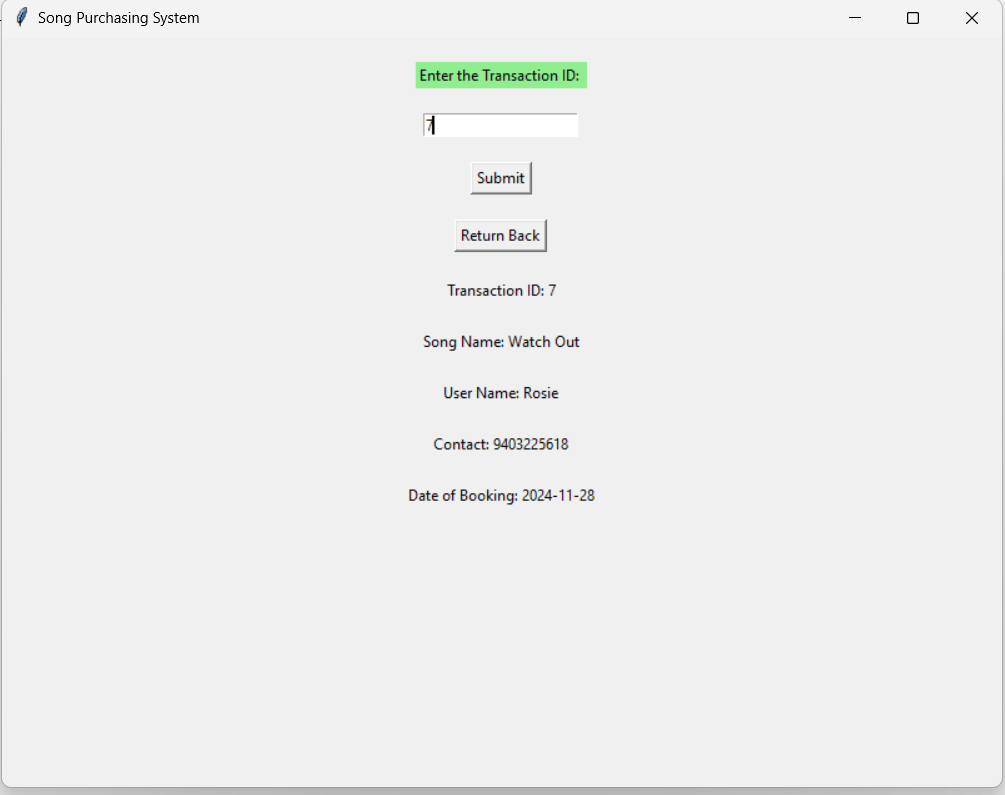


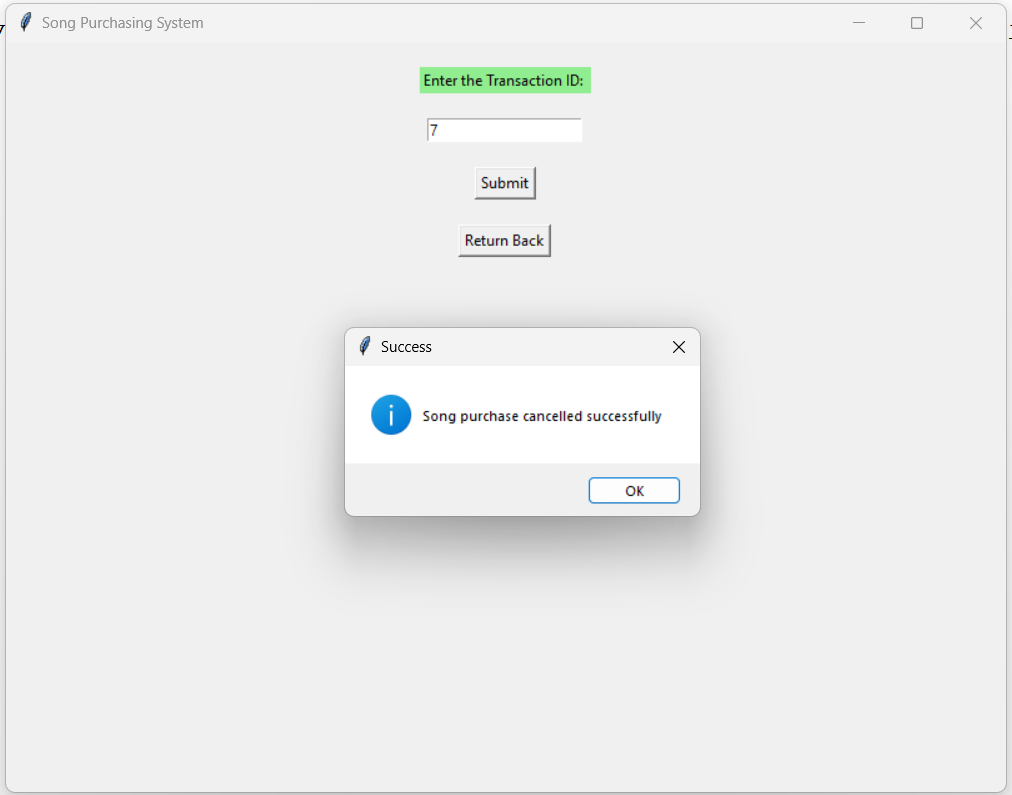












Bibliography

* [www.google.com](http://www.google.com)
* [www.wikipedia.com](http://www.wikipedia.com)
* [www.soundtrap.com](http://www.soundtrap.com)
* IDLE Shell 3.10.11
* MYSQL