COMPUTER SCIENCE

By Ayjot

TABLE OF CONTENT

INTRODUCTION

The project aims towards making the work of a clothes store manager easy.
It is a data management system where the cashier inputs the product details and the system shows the number of pieces available. It allows the store manager to add or remove a product or the product's details. This will be possible with the help of the tkinter library.It is the de facto way in Python to create **Graphical User interfaces(GUI's)** and it is included in all standard Python Distributions. In Fact,it's the only framework built into Python standard library.Also,all the data will be stored in the MySql Database. This project can be used in small to medium-sized standalone buisnesses and shops.

Aim & Outline

My program hopes to provide an all in one experience with an application when deployed serves the needs of the manager, warehouse employees and salespeople.

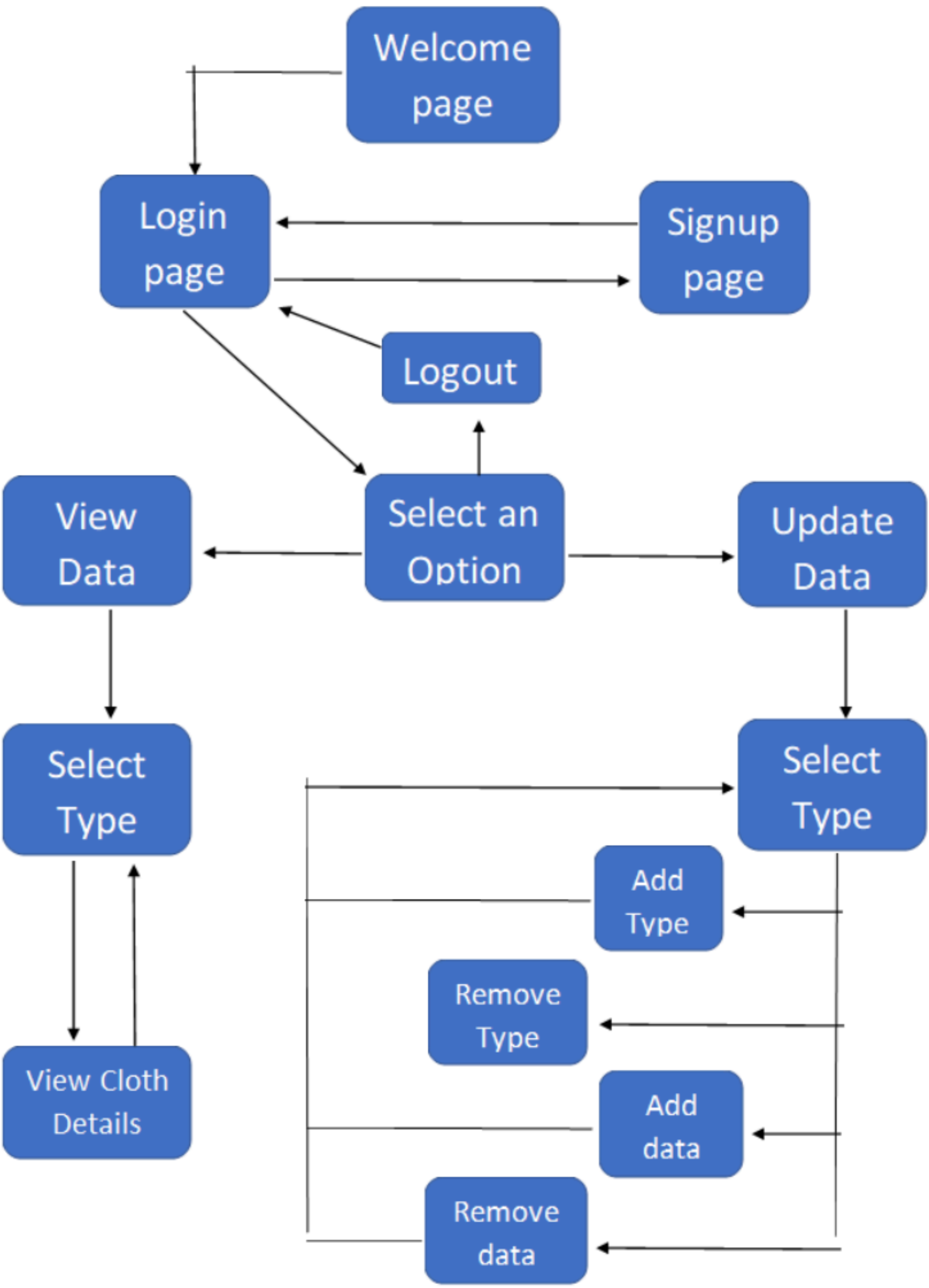
The main inspiration came from a local mall store where I once went and couldn't receive an order, only because there were different systems for selling products and maintaining inventory, and there was bound to be a problem with the integration of the two systems. I believed that there could not have been a better reason to create a program that can bring these two together.

We aim to save the data long term such that to face no problem and make the work easier for our generation.

Hardware &Software Specifications

Project Name	Cloth Store Management System
Hardware	Processor: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz RAM: 8 GB System Type: 64-bit operating system, x64-based processor
Software	Operating System: Windows 10 Front-end: Python IDLE (3.8 64-bit) Back-end: MySQL Command Line Client (8.0)

Process Flow



Source Code

```
# importing libraries
import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
import mysql.connector as sql

# Creating a basic tkinter screen
root = tk.Tk()
root.title("Cloth Store Managment System")
root.resizable(width = False, height = False)
root.iconbitmap("T-shirt_symbol.ico")

widget_frame = tk.Frame(root, bg = "black")
widget_frame.pack(fill = "both", expand = 1)

# Connecting to the mysql database
obj = sql.connect(host = "localhost", user = "root", passwd = "root", database = "cloth_store_database")
cursor = obj.cursor()

# Making the tkinter window come up in the middle of the screen
app_width = 450
app_height = 400

screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

x = (screen_width/2) - (app_width/2)
y = (screen_height/2) - (app_height/2)

root.geometry(f"{app_width}x{app_height}+{int(x)}+{int(y)}")

# Colour variables for widgets
bg_colour = "#000000"
fg_colour = "#FDFEFE"
btn_bg = "#0D007E"
btn_fg = "#FDFEFE"
active_btn_bg = "#00B0FF"
active_btn_fg = "#000000"

# Function to remove all widgets from the screen
def clear_screen():
   for widget in widget_frame.winfo_children():
      widget.destroy()

def welcome():
   welcome_label = tk.Label(widget_frame, text = "Welcome to\n Cloth Store Managment System" , font =
("century bold",20), bg = bg_colour, fg = "#0D3EEE")
   welcome_label.place(anchor = "c", relx = 0.5, rely = 0.4)

   loading_label = tk.Label(widget_frame, text = "Loading..." , font = ("century",14), bg = bg_colour, fg =
fg_colour)
   loading_label.place(anchor = "c", relx = 0.5, rely = 0.6)

   root.after(3000,lambda: login())

# Function to login
def login():
   global user_entry
   global pass_entry
   clear_screen()

   # Entry box to enter username
   user_label = tk.Label(widget_frame, text = "Username", font = ("century",14), bg = bg_colour, fg =
fg_colour)
   user_label.grid(column = 0, row = 0, padx = (90,10), pady = (60,5))
```

```
user_entry = tk.Entry(widget_frame, bd = 4, font = ("century",10))
    user_entry.grid(column = 1, row = 0, padx = (10,90), pady = (60,5))

    # Entry box to enter password
    pass_label = tk.Label(widget_frame, text = "Password", font = ("century",14), bg =
bg_colour, fg = fg_colour)
    pass_label.grid(column = 0, row = 1, padx = (90,10), pady = (20,5))

    pass_entry = tk.Entry(widget_frame, bd = 4, show = "*", font = ("century",10))
    pass_entry.grid(column = 1, row = 1, padx = (10,90), pady = (20,5))

    # Button for logging in
    login_btn = tk.Button(widget_frame, text = "Login", command =
check_credentials, font = ("century",14), bd = 5, width = 10,
                          bg = btn_bg, fg = btn_fg, activebackground = active_btn_bg,
activeforeground = active_btn_fga
    login_btn.place(anchor = "c", relx = 0.5, rely = 0.51)

    # Asking for signup
    signup_label = tk.Label(widget_frame, text = "Don't have an account?", font =
("century",12), bg = bg_colour, fg = fg_colour)
    signup_label.place(anchor = "c", relx = 0.35, rely = 0.7)

    signup_btn = tk.Button(widget_frame, text = "Sign up",command = signup, font =
("century",12), bd = 5, bg = btn_bg, fg = btn_fg,
                           activebackground = active_btn_bg, activeforeground = active_btn_fg)
    signup_btn.place(anchor = "c", relx = 0.7, rely = 0.7)

# Function to check if the entered user exists
def check_credentials():
    cursor.execute("select * from users")
    all_users = cursor.fetchall()

    user = user_entry.get().strip()
    passwd = pass_entry.get()
    x = False

    for i in all_users:
        if user == i[1] and passwd == i[2]:
            x = True

    if x == False:
        messagebox.showerror("Error", "Username/Password incorrect.")
    else:
        select_option()

# Function to sign up
def signup():
    global name_entry
    global user_entry
    global pass_entry
    global confirm_pass_entry
    clear_screen()

    # Entry box to enter name
    name_label = tk.Label(widget_frame, text = "Name", font = ("century",14), bg =
bg_colour, fg = fg_colour)
    name_label.grid(column = 0, row = 0, padx = (50,10), pady = (60,0))

    name_entry = tk.Entry(widget_frame, bd = 4, font = ("century",10))
    name_entry.grid(column = 1, row = 0, padx = (10,50), pady = (60,0))
```

```
# Entry box to enter username
    user_label = tk.Label(widget_frame, text = "Username", font =
("century",14), bg = bg_colour, fg = fg_colour)
    user_label.grid(column = 0, row = 1, padx = (50,10), pady = (25,0))

    user_entry = tk.Entry(widget_frame, bd = 4, font = ("century",10))
    user_entry.grid(column = 1, row = 1, padx = (10,50), pady = (25,0))

    # Entry box to enter password
    pass_label = tk.Label(widget_frame, text = "Password", font =
("century",14), bg = bg_colour, fg = fg_colour)
    pass_label.grid(column = 0, row = 2, padx = (50,10), pady = (25,0))

    pass_entry = tk.Entry(widget_frame, bd = 4, show = "*", font =
("century",10))
    pass_entry.grid(column = 1, row = 2, padx = (10,50), pady = (25,0))

    # Entry box to enter confirm password
    confirm_pass_label = tk.Label(widget_frame, text = "Confirm Password",
font = ("century",14), bg = bg_colour, fg = fg_colour)
    confirm_pass_label.grid(column = 0, row = 3, padx = (50,10), pady = (25,5))

    confirm_pass_entry = tk.Entry(widget_frame, bd = 4, show = "*", font =
("century",10))
    confirm_pass_entry.grid(column = 1, row = 3, padx = (10,50), pady = (25,5))

    # Back button to go to previous page
    back_btn = tk.Button(widget_frame, text = u"\u2190", font = ("century",12),
command = login, bd = 5, bg = btn_bg, fg = btn_fg,
                         activebackground = active_btn_bg, activeforeground =
active_btn_fg)
    back_btn.place(anchor = "nw")

    # Sign up button
    signup_btn = tk.Button(widget_frame, text = "Sign up",command =
check_signup_credentials, font = ("century",14), bd = 5, width = 10,
                           bg = btn_bg, fg = btn_fg,activebackground = active_btn_bg,
activeforeground = active_btn_fg)
    signup_btn.grid(column = 0, row = 4, columnspan = 2, padx = 20, pady =
25)

# Function to check signup credentials
def check_signup_credentials():
    # Extracting existing usernames
    cursor.execute("select * from users")
    all_users = cursor.fetchall()

    # Extracting values entered
    name = name_entry.get()
    user = user_entry.get().strip()
    passwd = pass_entry.get()
    confirm_passwd = confirm_pass_entry.get()
```

```
if len(name) <= 0:
    messagebox.showerror("Error", "Please enter your Name.")
  elif len(user) <= 0:
    messagebox.showerror("Error", "Please enter your Username.")
  elif len(passwd) <= 0:
    messagebox.showerror("Error", "Please enter your Password.")
  else:
    # Checking if password matches the confirm password
    if passwd != confirm_passwd:
      messagebox.showerror("Error", "Password and Confirm
Password do not match.")
    else:
      # Checking if username already exists
      x = False
      for i in all_users:
        if user == i[0]:
          x = True

      if x == True:
        messagebox.showerror("Error", "Username already exists.")
      else:
        # Inserting username and password in the database
        cursor.execute("insert into users values
('{}','{}','{}')".format(name,user,passwd))
        obj.commit()
        login()

# Function to select an option
def select_option():
  global option_entry

  clear_screen()

  info_options = """1. View Data
2. Update Data"""

  # Label of options
  info_label = tk.Label(widget_frame, text = info_options, font =
("century",14), justify = "left", bg = bg_colour, fg = fg_colour)
  info_label.place(anchor = "c", relx = 0.5, rely = 0.35)

  # Label to input the choice
  option_label = tk.Label(widget_frame, text = "Enter your choice", font
= ("century",14), bg = bg_colour, fg = fg_colour)
  option_label.place(anchor = "c", relx = 0.32, rely = 0.5)

  # Entry box to input the choice
  option_entry = tk.Entry(widget_frame, width = 20, bd = 4, font =
("century",10))
  option_entry.place(anchor = "c", relx = 0.68, rely = 0.5)
```

```
# Next button to confirm choice
    option_btn = tk.Button(widget_frame, text = "Next", font = ("century",14),
command = check_choice, bd = 5, width = 10,
                        bg = btn_bg, fg = btn_fg, activebackground = active_btn_bg,
activeforeground = active_btn_fg)
    option_btn.place(anchor = "c", relx = 0.5, rely = 0.68)

    # logout button to go to previous page
    logout_btn = tk.Button(widget_frame, text = "Logout", font =
("century",14), command = login, bd = 5, bg = btn_bg, fg = btn_fg,
                        activebackground = active_btn_bg, activeforeground =
active_btn_fg)
    logout_btn.place(anchor = "nw")

def check_choice():
    global option_value
    # Storing the choice in a variable
    option_value = option_entry.get()

    # storing valid values in a variable
    valid_option_values = ["1","2","view data","update data"]

    # Checking the choice
    if option_value.lower() not in valid_option_values:
        messagebox.showerror("Error", "Invalid Entry.")

    else:
        select_choice()

def select_choice():
    if option_value == "1" or option_value.lower() == "view data":
        global gender_combo
        global sizes_combo
        global types_combo

        clear_screen()

        men_types, women_types = [],[]

        # Extracting types of men clothes from the database
        cursor.execute("select * from men_cloth")
        for i in cursor.fetchall():
            men_types.append(i[0])

        # Extracting types of women clothes from the database
        cursor.execute("select * from women_cloth")
        for j in cursor.fetchall():
            women_types.append(j[0])

        sizes = ["Small","Medium","Large"]

        gender = ["Men","Women"]
```

```
 # Changing value of 'types_combo' dropdown menu according to the selected
gender
    def update_types(e):
        if gender_combo.get() == "Men":
            types_combo.config(value = men_types)
            types_combo.current(0)

        elif gender_combo.get() == "Women":
            types_combo.config(value = women_types)
            types_combo.current(0)

    # Dropdown menu to select gender
    gender_label = tk.Label(widget_frame, text = "Select Gender", font = ("century",14),
bg = bg_colour, fg = fg_colour)
    gender_label.grid(column = 0, row = 0, padx = 60, pady = (60,20))

    gender_combo = ttk.Combobox(widget_frame, value = gender, state = "readonly")
    gender_combo.current(0)
    gender_combo.grid(column = 1, row = 0, pady = (60,20))

    # Dropdown menu to select size
    sizes_label = tk.Label(widget_frame, text = "Select Size", font = ("century",14), bg =
bg_colour, fg = fg_colour)
    sizes_label.grid(column = 0, row = 1, padx = 60, pady = 20)

    sizes_combo = ttk.Combobox(widget_frame, value = sizes, state = "readonly")
    sizes_combo.current(0)
    sizes_combo.grid(column = 1, row = 1, pady = 20)

    # Dropdown menu to select type
    types_label = tk.Label(widget_frame, text = "Select Type", font = ("century",14), bg =
bg_colour, fg = fg_colour)
    types_label.grid(column = 0, row = 2, padx = 60, pady = 20)

    types_combo = ttk.Combobox(widget_frame, value = men_types, state =
"readonly")
    types_combo.current(0)
    types_combo.grid(column = 1, row = 2, pady = 20)

    gender_combo.bind("<<ComboboxSelected>>", update_types)

    # Next button
    confirm_type = tk.Button(widget_frame, text = "Next", font = ("century",14),
command = view_data, bd = 5, width = 10,
                        bg = btn_bg, fg = btn_fg, activebackground = active_btn_bg,
activeforeground = active_btn_fg)
    confirm_type.place(anchor = "c", relx = 0.5, rely = 0.75)

    # Back button to go to previous page
    back_btn = tk.Button(widget_frame, text = u"\u2190", font = ("century",12),
command = select_option, bd = 5, bg = btn_bg, fg = btn_fg,
                        activebackground = active_btn_bg, activeforeground = active_btn_fg)
    back_btn.place(anchor = "nw")

  else:
    global entry_widget
    clear_screen()
    choice_variable = """1. Add Type
2. Remove Type
3. Add Data
4. Remove Data"""
```

```
 # Label to show available choices
    choice_label = tk.Label(widget_frame, text = choice_variable, font =
("century",14), justify = "left", bg = bg_colour, fg = fg_colour)
    choice_label.place(anchor = "c", relx = 0.5, rely = 0.32)

    # Entry box to input choice
    entry_label = tk.Label(widget_frame, text = "Enter your choice",
font = ("century",14), bg = bg_colour, fg = fg_colour)
    entry_label.place(anchor = "c", relx = 0.32, rely = 0.52)

    entry_widget = tk.Entry(widget_frame, width = 20, bd = 4, font =
("century",10))
    entry_widget.place(anchor = "c", relx = 0.68, rely = 0.52)

    # Next button to confirm choice
    confirm_btn = tk.Button(widget_frame, text = "Next", font =
("century",14), command = select, bd = 5, width = 10, bg = btn_bg, fg =
btn_fg,
                            activebackground = active_btn_bg, activeforeground =
active_btn_fg)
    confirm_btn.place(anchor = "c", relx = 0.5, rely = 0.7)

    # Back button to go to previous page
    back_btn = tk.Button(widget_frame, text = u"\u2190", font =
("century",12), command = select_option, bd = 5, bg = btn_bg, fg =
btn_fg,
                         activebackground = active_btn_bg, activeforeground =
active_btn_fg)
    back_btn.place(anchor = "nw")

# Function to view the data
def view_data():
  # Extracting values of gender, size and type selected
  selected_gender =  gender_combo.get()
  selected_size =  sizes_combo.get()
  selected_type =  types_combo.get()

  # Extracting the amount from the database
  if selected_gender == "Men":
    cursor.execute("select {}_amount from men_cloth where type =
'{}'".format(selected_size,selected_type))
    amount_var = "The available amount of "+selected_type+"(s) in
"+selected_size+" size is: "+str(cursor.fetchall()[0][0])
  else:
    cursor.execute("select {}_amount from women_cloth where type =
'{}'".format(selected_size,selected_type))
    amount_var = "The available amount of "+selected_type+"(s) in
"+selected_size+" size is: "+str(cursor.fetchall()[0][0])

  # Displaying the available amount of cloth
  messagebox.showinfo("Amount",amount_var)
```

```
# Function to select type of cloth
def select():
    global men_types
    global women_types
    global gender_combo
    global sizes_combo
    global types_combo

    men_types, women_types = [],[]

    # Extracting types of men clothes from the database
    cursor.execute("select * from men_cloth")
    for i in cursor.fetchall():
        men_types.append(i[0])

    # Extracting types of women clothes from the database
    cursor.execute("select * from women_cloth")
    for j in cursor.fetchall():
        women_types.append(j[0])

    sizes = ["Small","Medium","Large"]

    choice = entry_widget.get()
    if choice == "1" or choice.lower() == "add type":
        global entry
        clear_screen()

        gender = ["Both","Men","Women"]

        # Dropdown menu to select gender
        gender_label = tk.Label(widget_frame, text = "Select Gender", font =
("century",14), bg = bg_colour, fg = fg_colour)
        gender_label.grid(column = 0, row = 0, padx = 70, pady = (80,30))

        gender_combo = ttk.Combobox(widget_frame, value = gender, state =
"readonly")
        gender_combo.current(0)
        gender_combo.grid(column = 1, row = 0, pady = (80,30))

        # Entry box to enter the type
        entry_label = tk.Label(widget_frame, text = "Enter the type to add", font =
("century",14), bg = bg_colour, fg = fg_colour)
        entry_label.grid(column = 0, row = 2, pady = 30)

        entry = tk.Entry(widget_frame, bd = 4)
        entry.grid(column = 1, row = 2, pady = 30)

        # Save button
        confirm_btn = tk.Button(widget_frame, text = "Save", font = ("century",14),
command = add_type, bd = 5, width = 10, bg = btn_bg, fg = btn_fg,
                                activebackground = active_btn_bg, activeforeground =
active_btn_fg)
        confirm_btn.place(anchor = "c", relx = 0.5, rely = 0.75)

        # Back button to go to previous page
```

```
 back_btn = tk.Button(widget_frame, text = u"\u2190", font = ("century",12),
command = select_choice, bd = 5, bg = btn_bg, fg = btn_fg,
                    activebackground = active_btn_bg, activeforeground = active_btn_fg)
    back_btn.place(anchor = "nw")

  elif choice == "2" or choice.lower() == "remove type":
    clear_screen()

    gender = ["Men","Women"]

    # Changing value of 'types_combo' dropdown menu according to the selected
gender
    def update_types(e):
      if gender_combo.get() == "Men":
        types_combo.config(value = men_types)
        types_combo.current(0)

      elif gender_combo.get() == "Women":
        types_combo.config(value = women_types)
        types_combo.current(0)

    # Dropdown menu to select gender
    gender_label = tk.Label(widget_frame, text = "Select Gender", font =
("century",14), bg = bg_colour, fg = fg_colour)
    gender_label.grid(column = 0, row = 0, padx = 70, pady = (80,30))

    gender_combo = ttk.Combobox(widget_frame, value = gender, state =
"readonly")
    gender_combo.current(0)
    gender_combo.grid(column = 1, row = 0, pady = (80,30))

    # Dropdown menu to select type
    types_label = tk.Label(widget_frame, text = "Select Type to remove", font =
("century",14), bg = bg_colour, fg = fg_colour)
    types_label.grid(column = 0, row = 2, pady = 30)

    types_combo = ttk.Combobox(widget_frame, value = men_types, state =
"readonly")
    types_combo.current(0)
    types_combo.grid(column = 1, row = 2, pady = 30)

    gender_combo.bind("<<ComboboxSelected>>", update_types)

    # Save button
    confirm_type = tk.Button(widget_frame, text = "Save", font = ("century",14),
command = remove_type, bd = 5, width = 10, bg = btn_bg, fg = btn_fg,
                        activebackground = active_btn_bg, activeforeground =
active_btn_fg)
    confirm_type.place(anchor = "c", relx = 0.5, rely = 0.75)

    # Back button to go to previous page
    back_btn = tk.Button(widget_frame, text = u"\u2190", font = ("century",12),
command = select_choice, bd = 5, bg = btn_bg, fg = btn_fg,
                    activebackground = active_btn_bg, activeforeground = active_btn_fg)
    back_btn.place(anchor = "nw")

  elif choice == "3" or choice.lower() == "add data":
    global add_entry
    clear_screen()
```

```
gender = ["Men","Women"]

    # Changing value of 'types_combo' dropdown menu according to the
selected gender
    def update_types(e):
        if gender_combo.get() == "Men":
            types_combo.config(value = men_types)
            types_combo.current(0)

        elif gender_combo.get() == "Women":
            types_combo.config(value = women_types)
            types_combo.current(0)

    # Dropdown menu to select gender
    gender_label = tk.Label(widget_frame, text = "Select Gender", font =
("century",14), bg = bg_colour, fg = fg_colour)
    gender_label.grid(column = 0, row = 0, padx = 60, pady = (40,20))

    gender_combo = ttk.Combobox(widget_frame, value = gender, state =
"readonly")
    gender_combo.current(0)
    gender_combo.grid(column = 1, row = 0, pady = (40,20))

    # Dropdown menu to select size
    sizes_label = tk.Label(widget_frame, text = "Select Size", font = ("century",14),
bg = bg_colour, fg = fg_colour)
    sizes_label.grid(column = 0, row = 1, padx = 60, pady = 20)

    sizes_combo = ttk.Combobox(widget_frame, value = sizes, state =
"readonly")
    sizes_combo.current(0)
    sizes_combo.grid(column = 1, row = 1, pady = 20)

    # Dropdown menu to select type
    types_label = tk.Label(widget_frame, text = "Select Type", font =
("century",14), bg = bg_colour, fg = fg_colour)
    types_label.grid(column = 0, row = 2, padx = 60, pady = 20)

    types_combo = ttk.Combobox(widget_frame, value = men_types, state =
"readonly")
    types_combo.current(0)
    types_combo.grid(column = 1, row = 2, pady = 20)

    gender_combo.bind("<<ComboboxSelected>>", update_types)

    # Entry box to enter the amount
    add_label = tk.Label(widget_frame, text = "Enter amount to add", font =
("century",14), bg = bg_colour, fg = fg_colour)
    add_label.grid(column = 0, row = 3, pady = 20)

    add_entry = tk.Entry(widget_frame, bd = 4, font = ("century",10))
    add_entry.grid(column = 1, row = 3, pady = 20)

    # Save button
    confirm_type = tk.Button(widget_frame, text = "Save", font = ("century",14),
command = add_data, bd = 5, width = 10, bg = btn_bg, fg = btn_fg,
                        activebackground = active_btn_bg, activeforeground =
active_btn_fg)
    confirm_type.place(anchor = "c", relx = 0.5, rely = 0.84)
```

```
# Back button to go to previous page
    back_btn = tk.Button(widget_frame, text = u"\u2190", font = ("century",12),
command = select_choice, bd = 5, bg = btn_bg, fg = btn_fg,
                activebackground = active_btn_bg, activeforeground = active_btn_fg)
    back_btn.place(anchor = "nw")

  elif choice == "4" or choice.lower() == "remove data":
    global remove_entry
    clear_screen()

    gender = ["Men","Women"]

    # Changing value of 'types_combo' dropdown menu according to the
selected gender
    def update_types(e):
      if gender_combo.get() == "Men":
        types_combo.config(value = men_types)
        types_combo.current(0)

      elif gender_combo.get() == "Women":
        types_combo.config(value = women_types)
        types_combo.current(0)

    # Dropdown menu to select gender
    gender_label = tk.Label(widget_frame, text = "Select Gender", font =
("century",14), bg = bg_colour, fg = fg_colour)
    gender_label.grid(column = 0, row = 0, padx = 60, pady = (40,20))

    gender_combo = ttk.Combobox(widget_frame, value = gender, state =
"readonly")
    gender_combo.current(0)
    gender_combo.grid(column = 1, row = 0, pady = (40,20))

    # Dropdown menu to select size
    sizes_label = tk.Label(widget_frame, text = "Select Size", font = ("century",14),
bg = bg_colour, fg = fg_colour)
    sizes_label.grid(column = 0, row = 1, padx = 60, pady = 20)

    sizes_combo = ttk.Combobox(widget_frame, value = sizes, state = "readonly")
    sizes_combo.current(0)
    sizes_combo.grid(column = 1, row = 1, pady = 20)

    # Dropdown menu to select type
    types_label = tk.Label(widget_frame, text = "Select Type", font = ("century",14),
bg = bg_colour, fg = fg_colour)
    types_label.grid(column = 0, row = 2, padx = 60, pady = 20)

    types_combo = ttk.Combobox(widget_frame, value = men_types, state =
"readonly")
    types_combo.current(0)
    types_combo.grid(column = 1, row = 2, pady = 20)

    gender_combo.bind("<<ComboboxSelected>>", update_types)

    # Entry box to enter the amount
    remove_label = tk.Label(widget_frame, text = "Enter amount to remove", font =
("century",14), bg = bg_colour, fg = fg_colour)
    remove_label.grid(column = 0, row = 3, pady = 20)
```

```
remove_entry = tk.Entry(widget_frame, bd = 4, font = ("century",10))
    remove_entry.grid(column = 1, row = 3, pady = 20)

    # Save button
    confirm_type = tk.Button(widget_frame, text = "Save", font = ("century",14),
command = remove_data, bd = 5, width = 10, bg = btn_bg, fg = btn_fg,
                        activebackground = active_btn_bg, activeforeground =
active_btn_fg)
    confirm_type.place(anchor = "c", relx = 0.5, rely = 0.84)

    # Back button to go to previous page
    back_btn = tk.Button(widget_frame, text = u"\u2190", font = ("century",12),
command = select_choice, bd = 5, bg = btn_bg, fg = btn_fg,
                    activebackground = active_btn_bg, activeforeground = active_btn_fg)
    back_btn.place(anchor = "nw")

  else:
    messagebox.showerror("Error", "Invalid Choice.")

# Function to add the type to the database
def add_type():
  # Extracting values of gender, size and type selected
  selected_gender = gender_combo.get()
  entered_type = entry.get().title()

  # Inserting the cloth into the database
  if selected_gender == "Men" and len(entered_type) > 0:

    # Checking if type already exists
    if entered_type in men_types:
      messagebox.showerror("Error", "Type alreasy exists.")

    else:
      cursor.execute("insert into men_cloth (Type) values
('{}')".format(entered_type))
      obj.commit()
      select_choice()

  elif selected_gender == "Women" and len(entered_type) > 0:

    # Checking if type already exists
    if entered_type in women_types:
      messagebox.showerror("Error", "Type alreasy exists.")

    else:
      cursor.execute("insert into women_cloth (Type) values
('{}')".format(entered_type))
      obj.commit()
      select_choice()

  elif selected_gender == "Both" and len(entered_type) > 0:

    # Checking if type already exists
    if entered_type in men_types:
      messagebox.showerror("Error", "Type alreasy exists in men.")

    elif entered_type in women_types:
      messagebox.showerror("Error", "Type alreasy exists in women.")
```

```
 else:
        cursor.execute("insert into men_cloth (Type) values
('{}')".format(entered_type))
        cursor.execute("insert into women_cloth (Type) values
('{}')".format(entered_type))
        obj.commit()
        select_choice()

    else:
        messagebox.showerror("Error", "Invalid Type.")

# Function to remove type from the database
def remove_type():
    # Extracting values of gender, size and type selected
    selected_gender = gender_combo.get()
    selected_type = types_combo.get()

    # Removing the cloth from the database
    if selected_gender == "Men" and selected_type in str(men_types):
        cursor.execute("delete from men_cloth where type =
'{}'".format(selected_type))

    else:
        cursor.execute("delete from women_cloth where type =
'{}'".format(selected_type))

    obj.commit()
    select_choice()

# Function to add data in the database
def add_data():
    try:
        # Extracting the value of amount entered
        entered_amount = int(add_entry.get())

        if entered_amount <= 0:
            messagebox.showerror("Error", "Please enter a value greater than 0.")
        else:
            # Extracting values of gender, size and type selected
            selected_gender = gender_combo.get()
            selected_size = sizes_combo.get()
            selected_type = types_combo.get()

            # Updating the amount in the databases
            if selected_gender == "Men":
                cursor.execute("update men_cloth set {}_amount = {}_amount + {} where
type = '{}'".format(selected_size, selected_size,
                                                                    entered_amount,
selected_type))
            else:
                cursor.execute("update women_cloth set {}_amount = {}_amount + {}
where type = '{}'".format(selected_size, selected_size,
                                                                    entered_amount,
selected_type))
            obj.commit()
            select_choice()
```

```
except ValueError:
    messagebox.showerror("Error", "Please enter an integer.")

def remove_data():
  try:
    # Extracting the value of amount entered
    entered_amount = int(remove_entry.get())

    if entered_amount <= 0:
      messagebox.showerror("Error", "Please enter a value greater than 0.")
    else:
      # Extracting values of gender, size and type selected
      selected_gender = gender_combo.get()
      selected_size = sizes_combo.get()
      selected_type = types_combo.get()

      # Updating the amount in the database
      if selected_gender == "Men":
        cursor.execute("select {}_amount from men_cloth where type =
'{}'".format(selected_size, selected_type))

        if cursor.fetchall()[0][0] == 0:
          messagebox.showerror("Error", "Amount is already 0.")
        else:
          cursor.execute("update men_cloth set {}_amount = {}_amount - {}
where type = '{}'".format(selected_size, selected_size,
                                                          entered_amount,
selected_type))
          obj.commit()
          select_choice()

      elif selected_gender == "Women":
        cursor.execute("select {}_amount from men_cloth where type =
'{}'".format(selected_size, selected_type))

        if cursor.fetchall()[0][0] == 0:
          messagebox.showerror("Error", "Amount is already 0.")
        else:
          cursor.execute("update women_cloth set {}_amount = {}_amount -
{} where type = '{}'".format(selected_size, selected_size,
                                                          entered_amount,
selected_type))
          obj.commit()
          select_choice()

  except ValueError:
    messagebox.showerror("Error", "Please enter an integer.")


welcome()

root.mainloop()
obj.close()
```

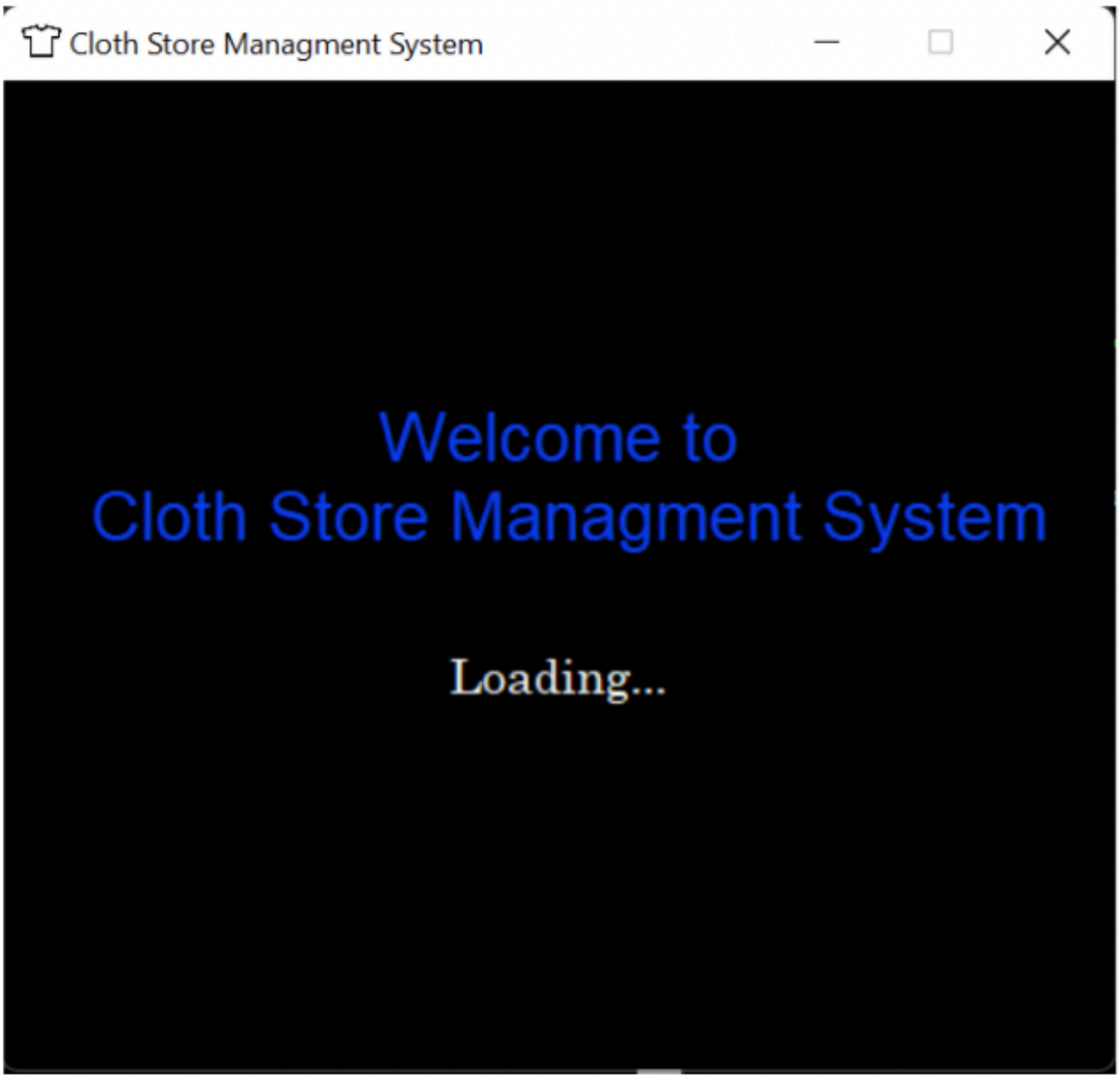
SQL Code:

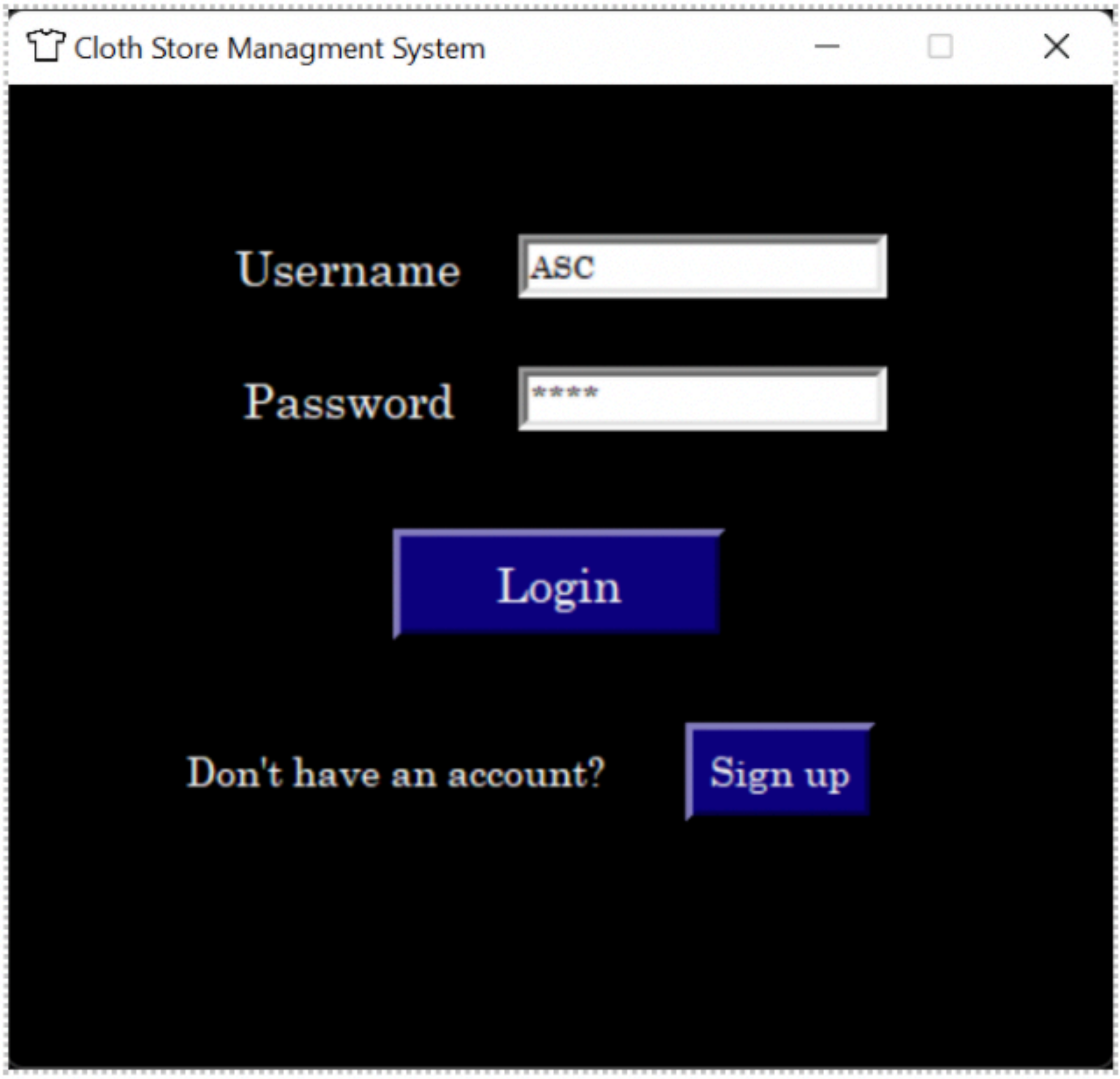
```
mysql> create database Cloth_store_database;
Query OK, 1 row affected (0.06 sec)
```

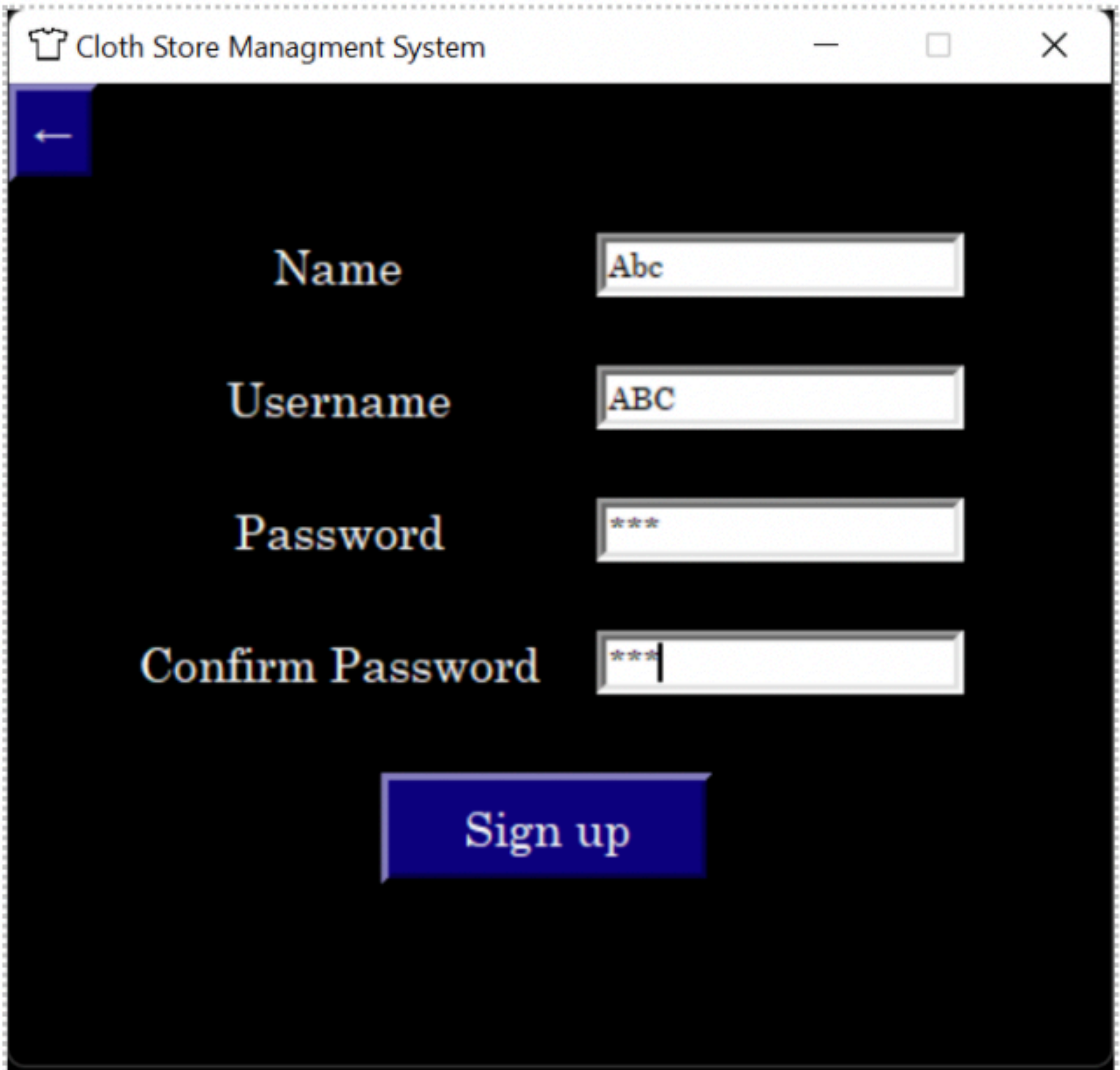
```
mysql> create table users (name varchar(20), username varchar(20),
password varchar(20));
Query OK, 0 rows affected (0.02 sec)
```

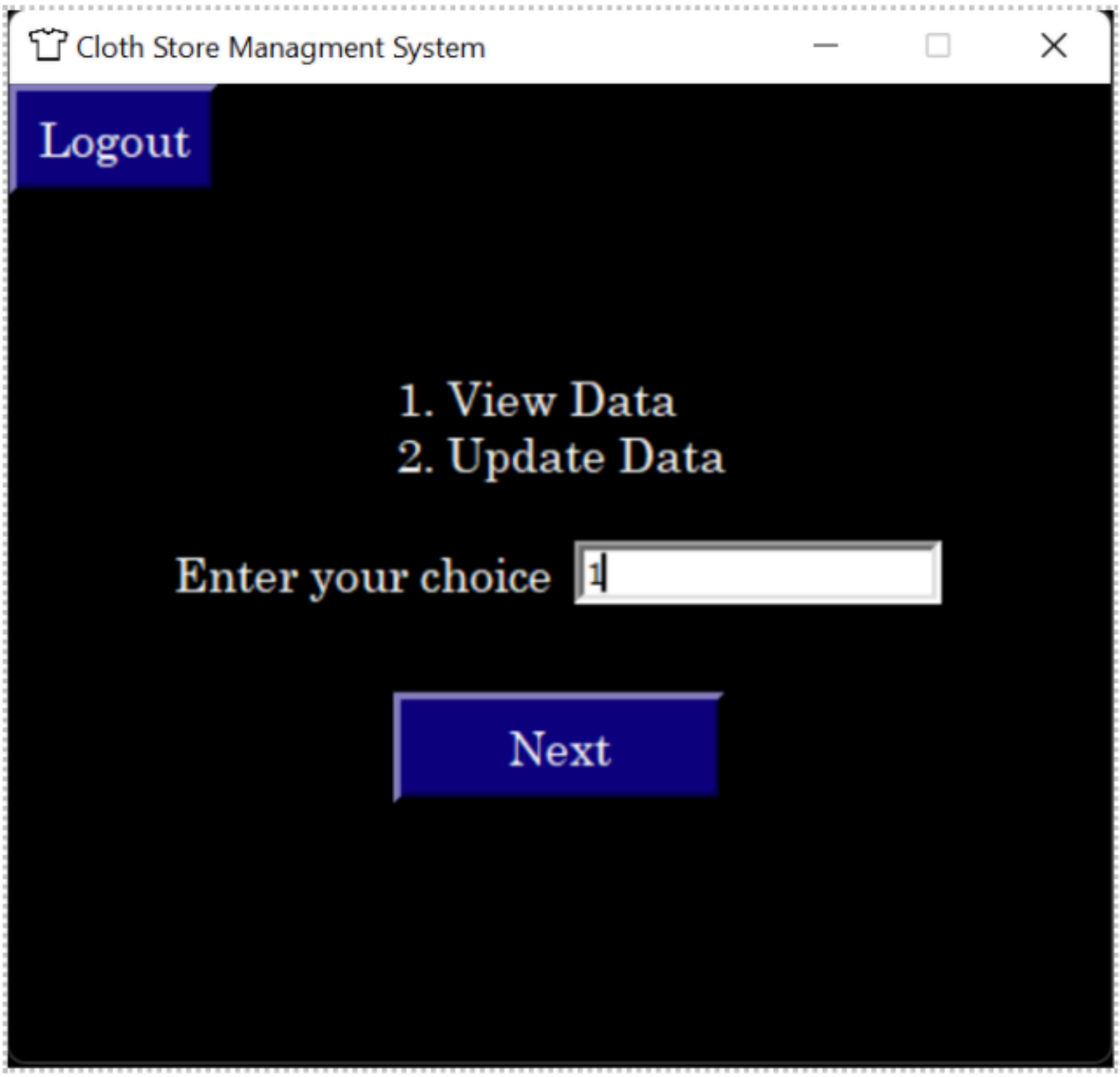
```
mysql> create table men_cloth(Type varchar(20), Small_Amount int(5) default 0,
Medium_Amount int(5) default 0, Large_Amount int(5) default 0);
Query OK, 0 rows affected, 3 warnings (0.03 sec)

mysql> create table women_cloth(Type varchar(20), Small_Amount int(5) default 0
, Medium_Amount int(5) default 0, Large_Amount int(5) default 0);
Query OK, 0 rows affected, 3 warnings (0.03 sec)
```

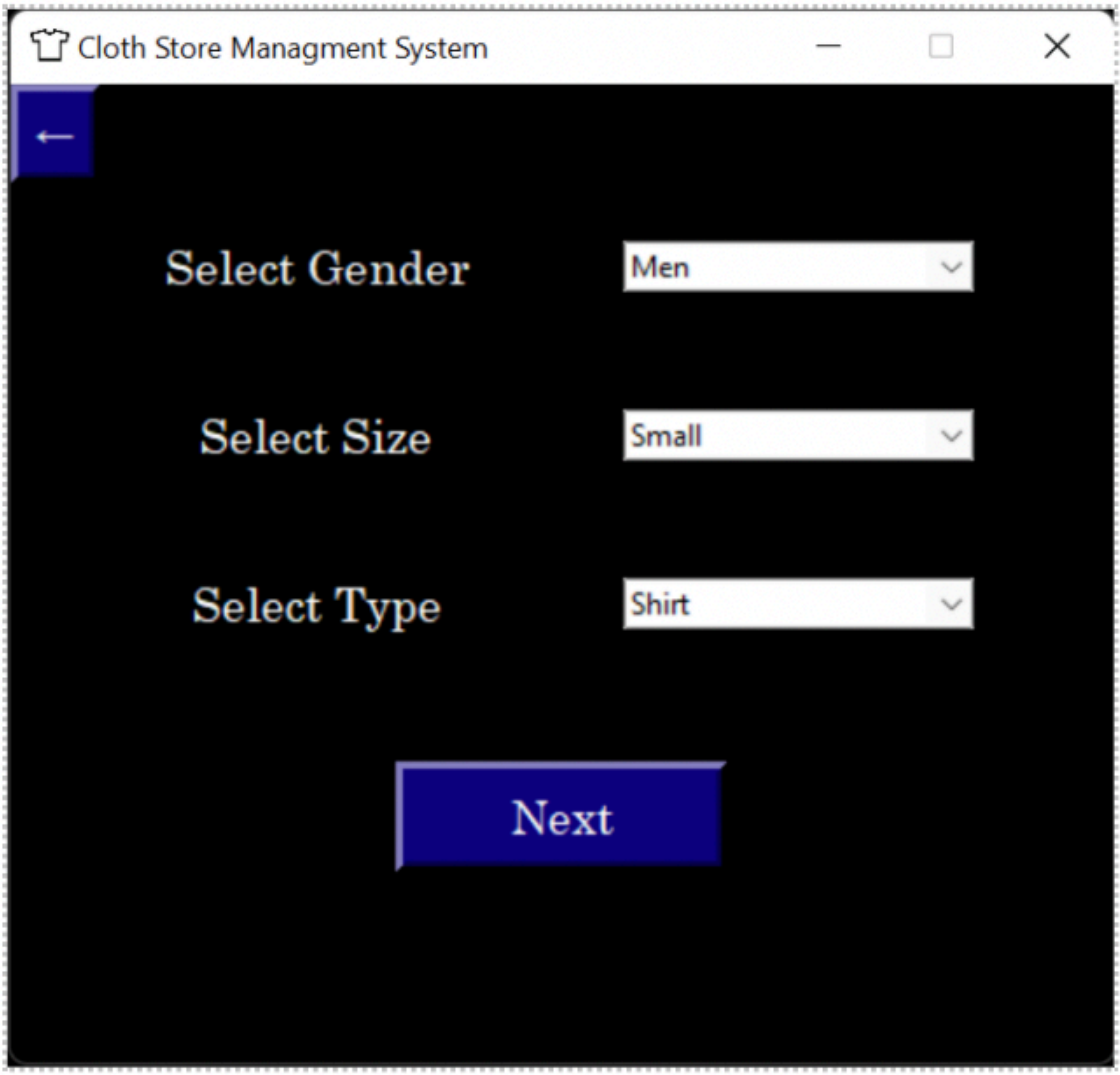
OUTPUT:-



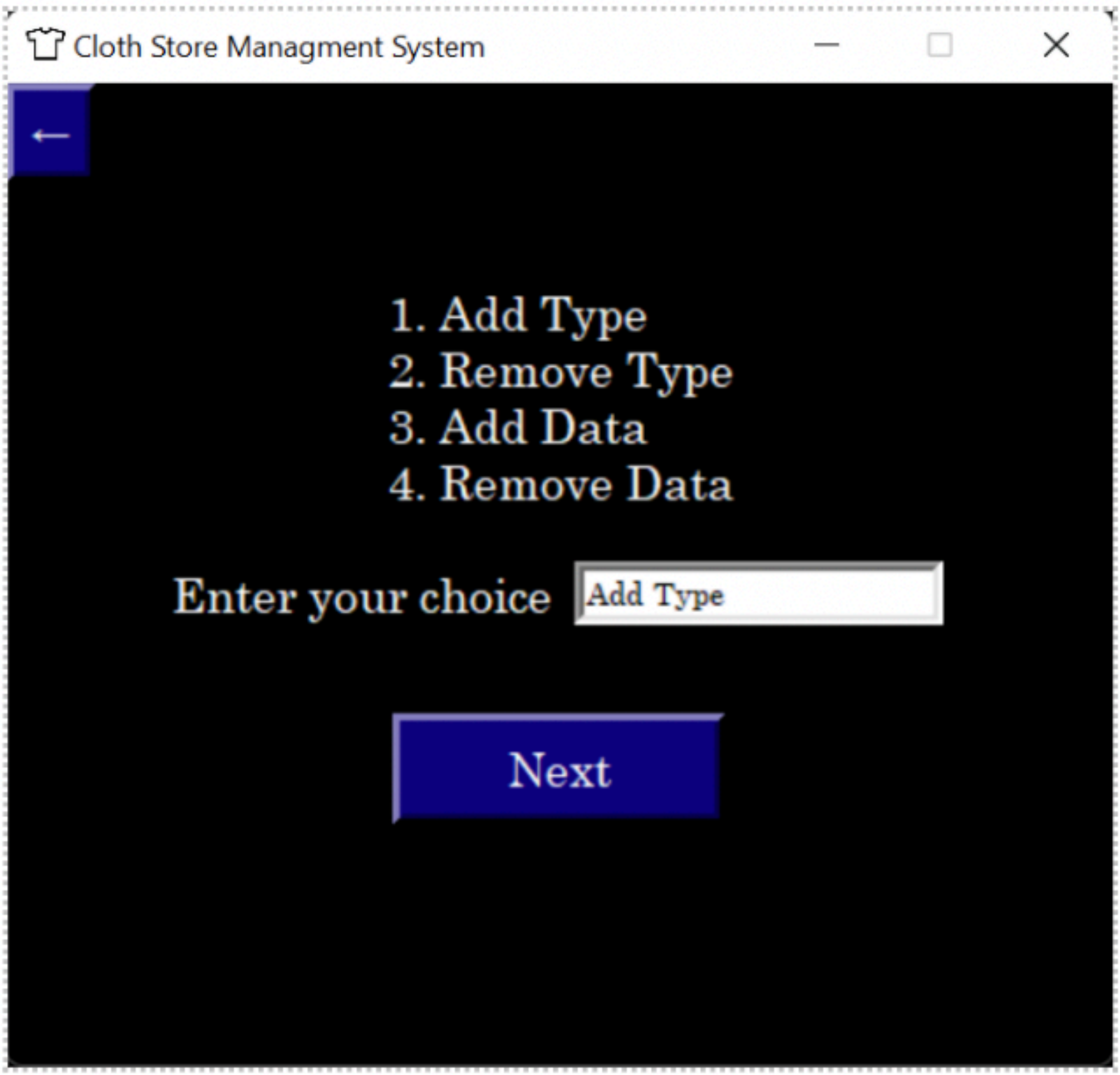

Cloth Store Managment System
Username
ASC
Password
****
Login
Don't have an account?
Sign up

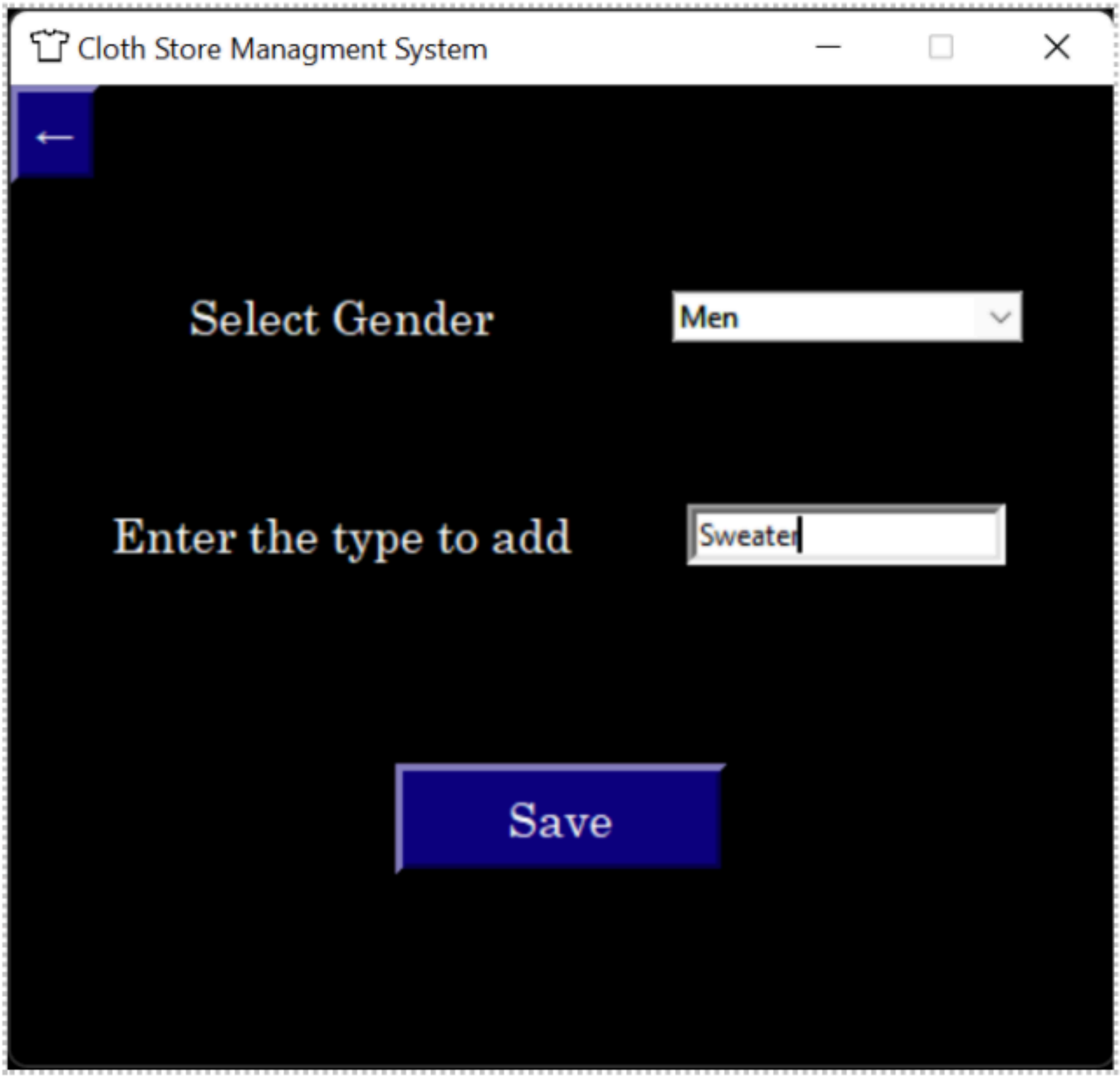

Cloth Store Managment System
←
Name
Abc
Username
ABC
Password
***
Confirm Password
***
Sign up

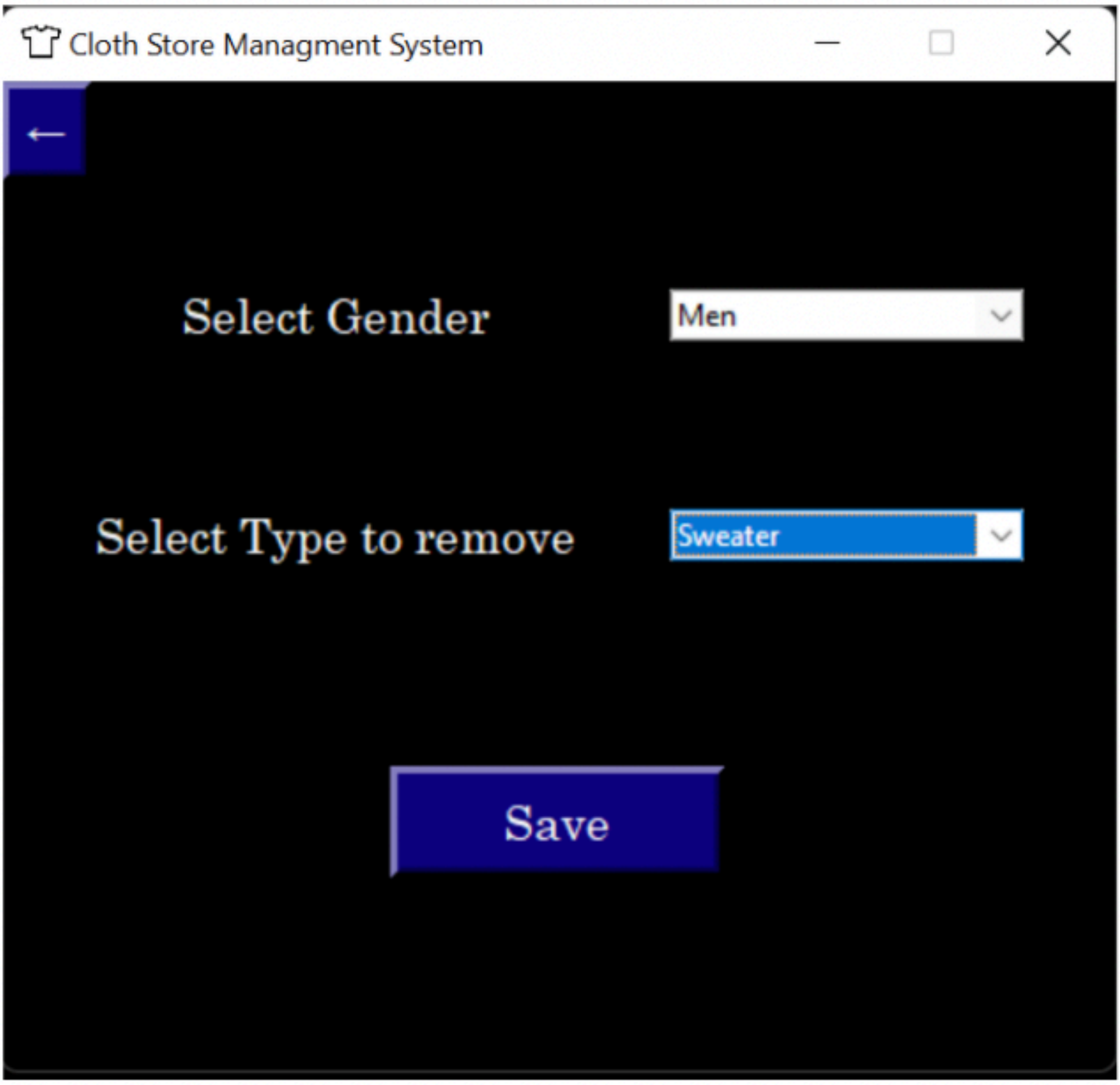

Cloth Store Managment System
Logout
1. View Data
2. Update Data
Enter your choice
1
Next

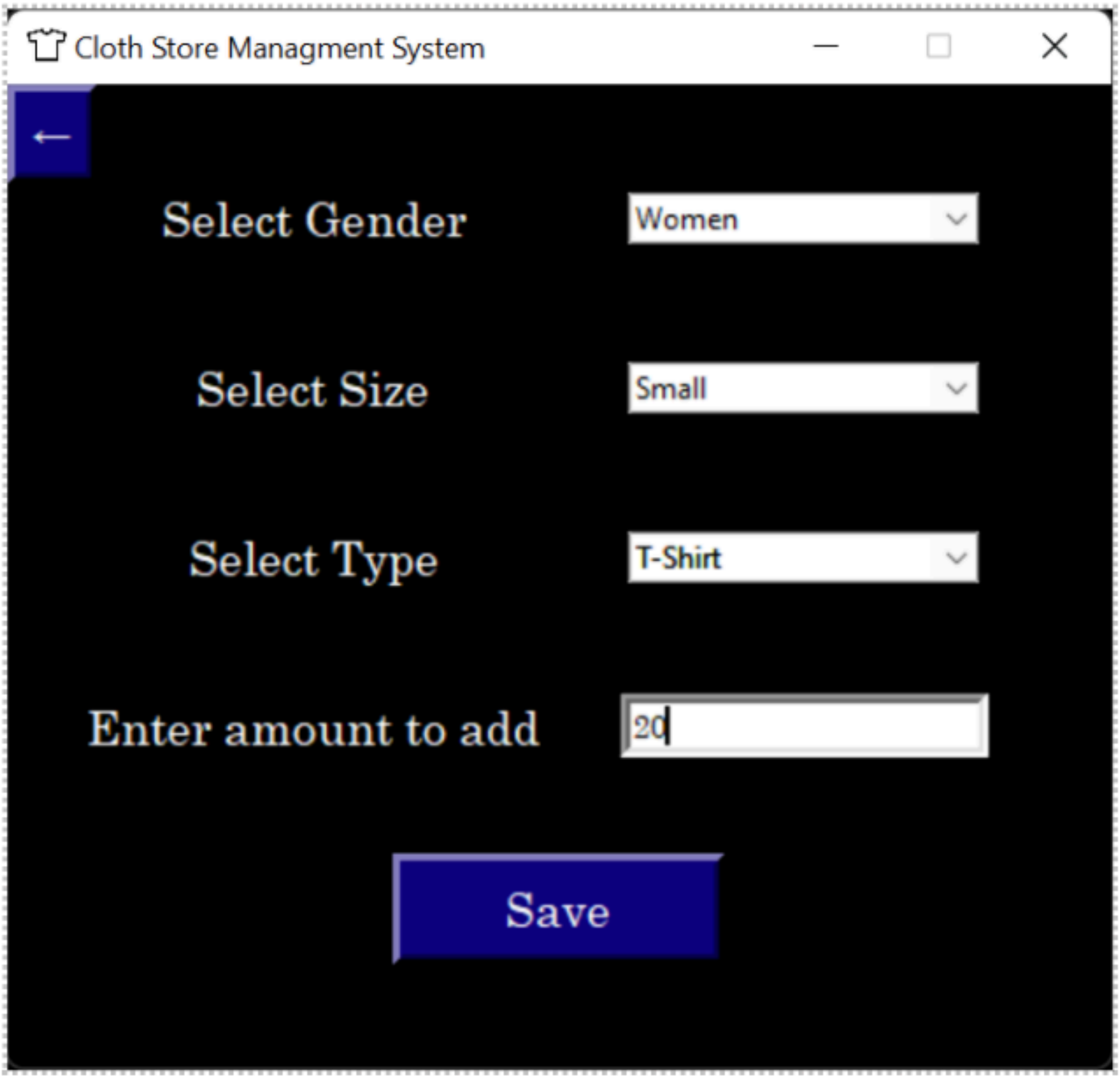

Cloth Store Managment System
Select Gender
Men
Select Size
Small
Select Type
Shirt
Next

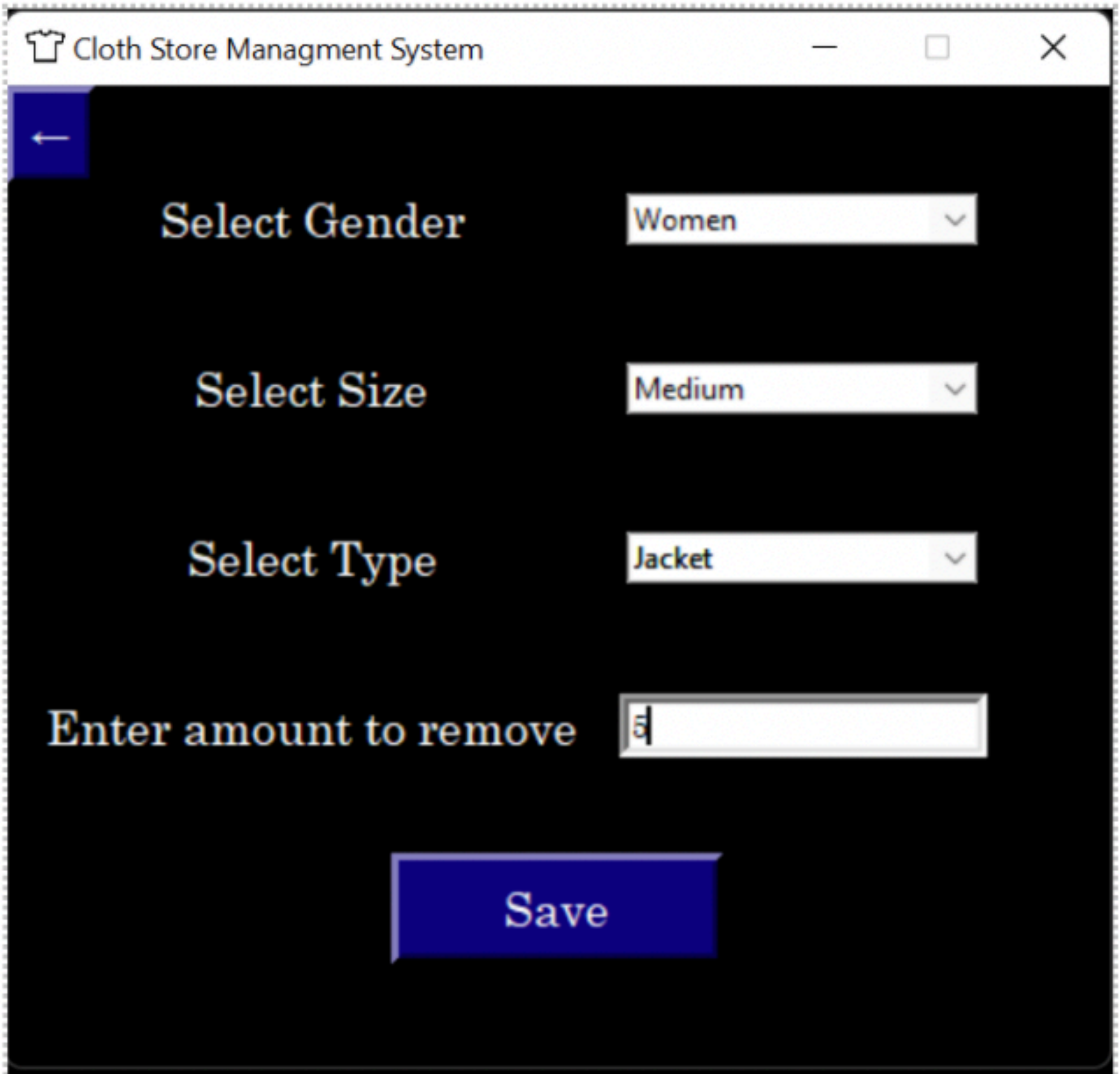

Amount
The available amount of Shirt(s) in Small size is: 80
OK


Cloth Store Managment System
1. Add Type
2. Remove Type
3. Add Data
4. Remove Data
Enter your choice
Add Type
Next


Cloth Store Managment System
Select Gender
Men
Enter the type to add
Sweater
Save


Cloth Store Managment System
Select Gender
Men
Select Type to remove
Sweater
Save


Cloth Store Managment System
Select Gender
Women
Select Size
Small
Select Type
T-Shirt
Enter amount to add
20
Save


Cloth Store Managment System
Select Gender
Women
Select Size
Medium
Select Type
Jacket
Enter amount to remove
5
Save

Future Enhancements

In future we plan to make the program more efficient by adding more cloth types and even add where the products are placed on the shelf, Manager Column where all employees can be added and attendance can be added to the system. Much Much More.....

BIBLIOGRAPHY

GOOGLE.COM

WIKIPEDIA.COM

BOTSCHOOL.COM

ZIGYA.COM

COMPUTER SCIENCE BY SUMITA ARORA