SQL CONNECTIVITY WITH VS CODE

Pinkeye_App > test.py > check_connection

```python
# check_db_connection.py
import pyodbc as odbc
def check_connection():
    try:
        DRIVER_NAME = 'ODBC Driver 18 for SQL Server'
        SERVER = 'SYEDA-FARWA-BAT\\CLASS'  # Use double backslashes
        DATABASE = 'Pinkeyeflu'

        connStr = (
            f"Driver={{{DRIVER_NAME}}};"
            f"Server={SERVER};"
            f"Database={DATABASE};"
            "Trusted_Connection=yes;"
            "TrustServerCertificate=yes;"  # Added to trust the server certificate
        )
        connection = odbc.connect(connStr)
        print("Connected to SQL Server Database")
        connection.close()
    except odbc.DatabaseError as e:
        print("There was a problem connecting to the database: ", e)
if __name__ == "__main__":
    check_connection()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
Connected to SQL Server Database
PS C:\Users\hp\Downloads\Pinkeye_App_updated> & "C:/Program Files/Python312/python.exe" c:/Users/hp/Downloads/Pinkeye_App_updated/Pinkeye_App/test.py
Connected to SQL Server Database
PS C:\Users\hp\Downloads\Pinkeye_App_updated> & "C:/Program Files/Python312/python.exe" c:/Users/hp/Downloads/Pinkeye_App_updated/Pinkeye_App/test.py
```

DATABASE UTILITY FUNCTIONS

Pinkeye_App > db_utils.py > get_connection

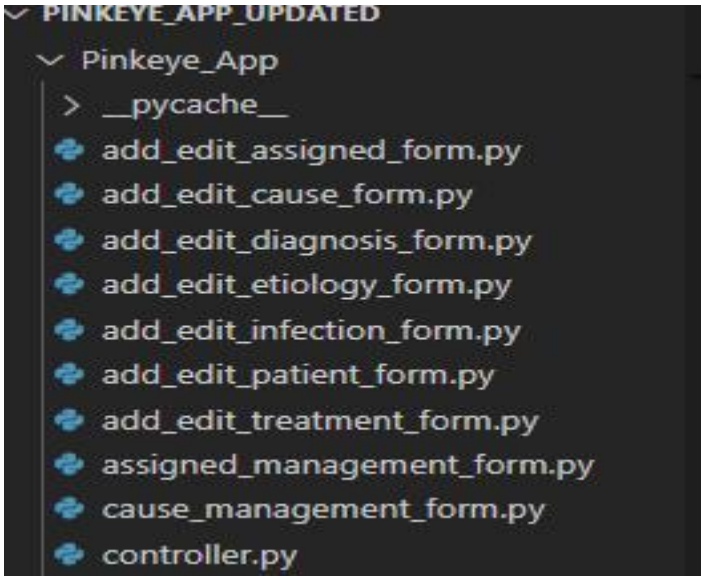
```python
import pyodbc as odbc

def get_connection():
    try:
        DRIVER_NAME = 'ODBC Driver 18 for SQL Server'
        SERVER = 'SYEDA-FARWA-BAT\\CLASS'  # Use double backslashes
        DATABASE = 'Pinkeyeflu'

        connStr = (
            f"Driver={{{DRIVER_NAME}}};"
            f"Server={SERVER};"
            f"Database={DATABASE};"
            "Trusted_Connection=yes;"
            "TrustServerCertificate=yes;"  # Added to trust the server certificate
        )
        connection = odbc.connect(connStr)
        print("Connected to SQL Server Database")
        # connection.close()
        return connection
    except odbc.DatabaseError as e:
        print("There was a problem connecting to the database: ", e)
```

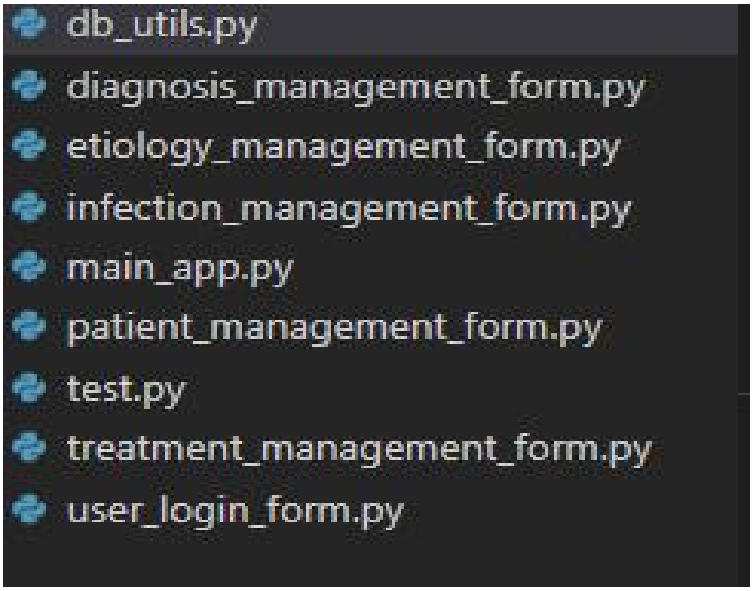
```python
def execute_query(query, params=None):
    connection = get_connection()
    cursor = connection.cursor()
    cursor.execute(query, params or [])
    result = cursor.fetchall()
    cursor.close()
    connection.close()
    return result

def execute_non_query(query, params=None):
    connection = get_connection()
    cursor = connection.cursor()
    cursor.execute(query, params or [])
    connection.commit()
    cursor.close()
    connection.close()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
Connected to SQL Server Database
PS C:\Users\hp\Downloads\Pinkeye_App_updated> & "C:/Program Files/Python312/python.exe" c:/Users/hp/Downloads/Pinkeye_App_updated/Pinkeye_App/db_utils.py
PS C:\Users\hp\Downloads\Pinkeye_App_updated>
```

FORMS :





CONTROLLER

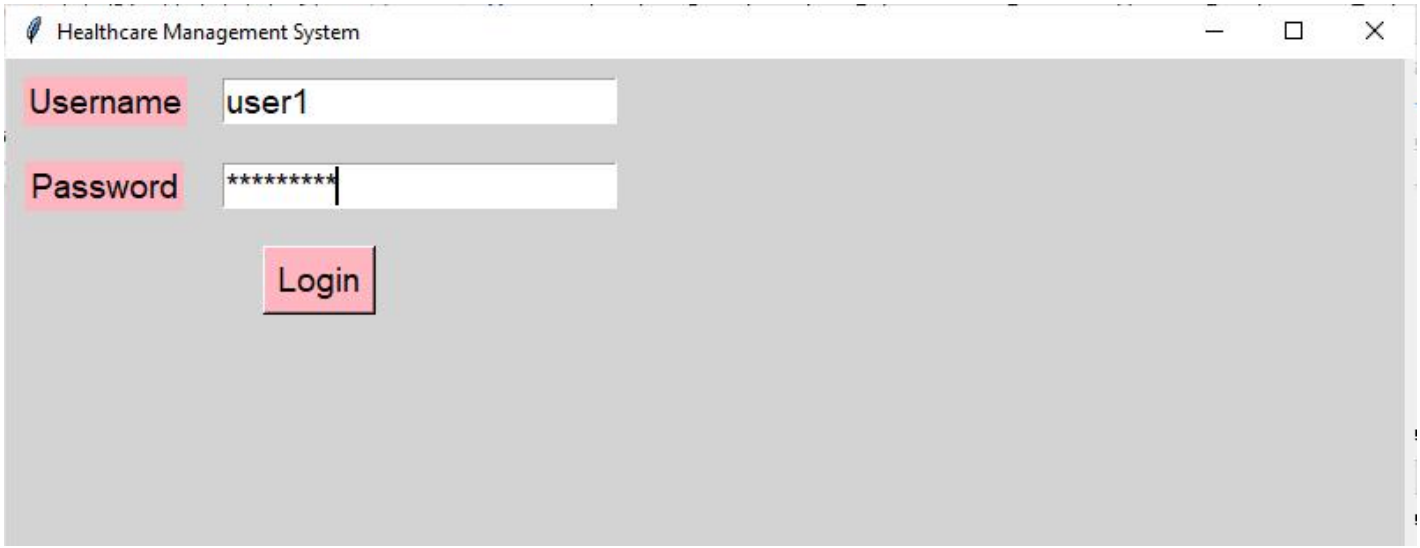
```
import tkinter as tk
from tkinter import font
from user_login_form import UserLoginForm
from add_edit_patient_form import AddEditPatientForm
from patient_management_form import PatientManagementForm
from add_edit_patient_form import AddEditPatientForm
from infection_management_form import InfectionManagementForm
from add_edit_infection_form import AddEditInfectionForm
from diagnosis_management_form import DiagnosisManagementForm
from add_edit_diagnosis_form import AddEditDiagnosisForm
from treatment_management_form import TreatmentManagementForm
from add_edit_treatment_form import AddEditTreatmentForm
from etiology_management_form import EtiologyManagementForm
from add_edit_etiology_form import AddEditEtiologyForm
from cause_management_form import CauseManagementForm
from add_edit_cause_form import AddEditCauseForm
from assigned_management_form import AssignedManagementForm
from add_edit_assigned_form import AddEditAssignedForm

class MainController(tk.Tk):
    def __init__(self, bg_color="light grey", button_color="light pink"):
        super().__init__()
        self.title("Healthcare Management System")
        self.geometry("800x525")

        # Define a custom font
        custom_font = font.Font(family="Helvetica", size=14)  # You can change the family and size as
needed

        # Apply the custom font to all widgets
        self.option_add("*Font", custom_font)

        self.frames = {}
        self.bg_color = bg_color
```

```
        self.button_color = button_color

        for F in (UserLoginForm,PatientManagementForm,
AddEditPatientForm,InfectionManagementForm,AddEditInfectionForm ,
DiagnosisManagementForm, AddEditDiagnosisForm, TreatmentManagementForm,
AddEditTreatmentForm, EtiologyManagementForm, AddEditEtiologyForm, CauseManagementForm,
AddEditCauseForm, AssignedManagementForm, AddEditAssignedForm):
            page_name = F.__name__
            frame = F(parent=self, controller=self, bg_color=self.bg_color,
button_color=self.button_color)
            self.frames[page_name] = frame
            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame("UserLoginForm")

    def show_frame(self, page_name):
        frame = self.frames[page_name]
        frame.tkraise()

if __name__ == "__main__":
    app = MainController()
    app.mainloop()
```

USER LOGIN FORM

```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_query

class UserLoginForm(tk.Frame):
    def __init__(self, parent, controller, bg_color, button_color):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.create_widgets()

    def create_widgets(self):
        self.label_username = tk.Label(self, text="Username",bg=self.button_color)
        self.label_username.grid(row=0, column=0, padx=10, pady=10)

        self.entry_username = tk.Entry(self)
        self.entry_username.grid(row=0, column=1, padx=10, pady=10)

        self.label_password = tk.Label(self, text="Password",bg=self.button_color)
        self.label_password.grid(row=1, column=0, padx=10, pady=10)

        self.entry_password = tk.Entry(self, show="*")
        self.entry_password.grid(row=1, column=1, padx=10, pady=10)

        self.button_login = tk.Button(self, text="Login", bg=self.button_color, command=self.login)
        self.button_login.grid(row=2, column=0, columnspan=2, pady=10)

    def login(self):
        username = self.entry_username.get()
        password = self.entry_password.get()
        query = "SELECT * FROM Userlogin WHERE username = ? AND password = ?"
        result = execute_query(query, (username, password))
        if result:
            self.controller.show_frame("PatientManagementForm")
```

```
        else:
            messagebox.showerror("Error", "Invalid username or password")


if __name__ == "__main__":
    root = tk.Tk()
    frame = UserLoginForm(root, None, "Light Grey", "light pink")
    frame.pack()
    root.mainloop()
```

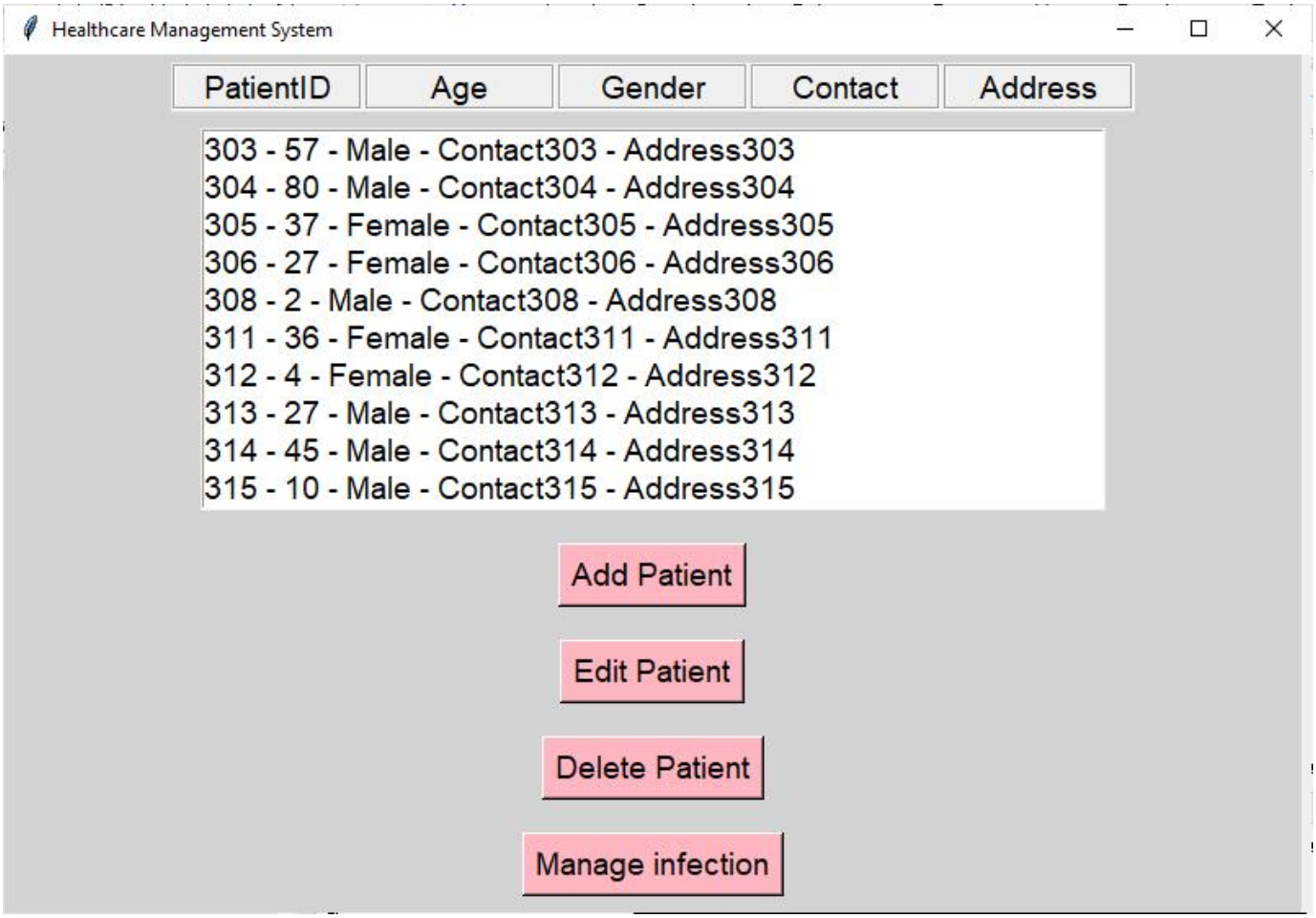


PATIENT MANAGEMENT FORM

```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_query, execute_non_query


class PatientManagementForm(tk.Frame):
    def __init__(self, parent, controller, bg_color, button_color="light pink"):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.create_widgets()


    def create_widgets(self):
        # Create a frame for the heading
        heading_frame = tk.Frame(self)
        heading_frame.grid(row=0, column=0, columnspan=1, padx=5, pady=(5, 0))
```

```
    # Add headings as labels
    headings = ["PatientID", "Age", "Gender", "Contact", "Address"]
    for col_num, heading in enumerate(headings):
        label = tk.Label(heading_frame, text=heading, borderwidth=2, relief="groove", width=10)
        label.grid(row=0, column=col_num, padx=1, pady=1)

    self.patient_listbox = tk.Listbox(self, width=50)
    self.patient_listbox.grid(row=1, column=0, columnspan=3, padx=10, pady=10)

    # Configure the Listbox frame grid
    self.grid_rowconfigure(1, weight=1)
    self.grid_columnconfigure(0, weight=1)

    self.button_add = tk.Button(self, text="Add Patient", command=lambda:
      self.controller.show_frame("AddEditPatientForm"), bg=self.button_color)
    self.button_add.grid(row=2, column=0, padx=10, pady=10)
    self.button_edit = tk.Button(self, text="Edit Patient", command=self.edit_patient,
      bg=self.button_color)
    self.button_edit.grid(row=3, column=0, padx=10, pady=10)
    self.button_delete = tk.Button(self, text="Delete Patient", command=self.delete_patient,
      bg=self.button_color)
    self.button_delete.grid(row=4, column=0, padx=10, pady=10)
    self.button_manage_infection = tk.Button(self, text="Manage infection", command=lambda:
     self.controller.show_frame("InfectionManagementForm"), bg=self.button_color)
    self.button_manage_infection.grid(row=5, column=0, padx=10, pady=10)
    self.load_patients()
def load_patients(self):
    self.patient_listbox.delete(0, tk.END)
    query = "SELECT * FROM patients"
    patients = execute_query(query)
    for patient in patients:
        self.patient_listbox.insert(tk.END, f"{patient[0]} - {patient[1]} - {patient[2]} - {patient[3]} -
        {patient[4]}")
def edit_patient(self):
    selected = self.patient_listbox.curselection()
    if not selected:
```

```
        messagebox.showerror("Error", "No patient selected")
        return
    patient_id = self.patient_listbox.get(selected).split(" - ")[0]
    self.controller.frames["AddEditPatientForm"].load_patient(patient_id)
    self.controller.show_frame("AddEditPatientForm")
  def delete_patient(self):
    selected = self.patient_listbox.curselection()
    if not selected:
        messagebox.showerror("Error", "No patient selected")
        return
    patient_id = self.patient_listbox.get(selected).split(" - ")[0]
    query = "DELETE FROM Patients WHERE patientID = ?"
    execute_non_query(query, (patient_id,))
    self.load_patients()
if __name__ == "__main__":
  root = tk.Tk()
  frame = PatientManagementForm(root, None, "Light Grey", "light pink")
  frame.pack()
  root.mainloop()
```

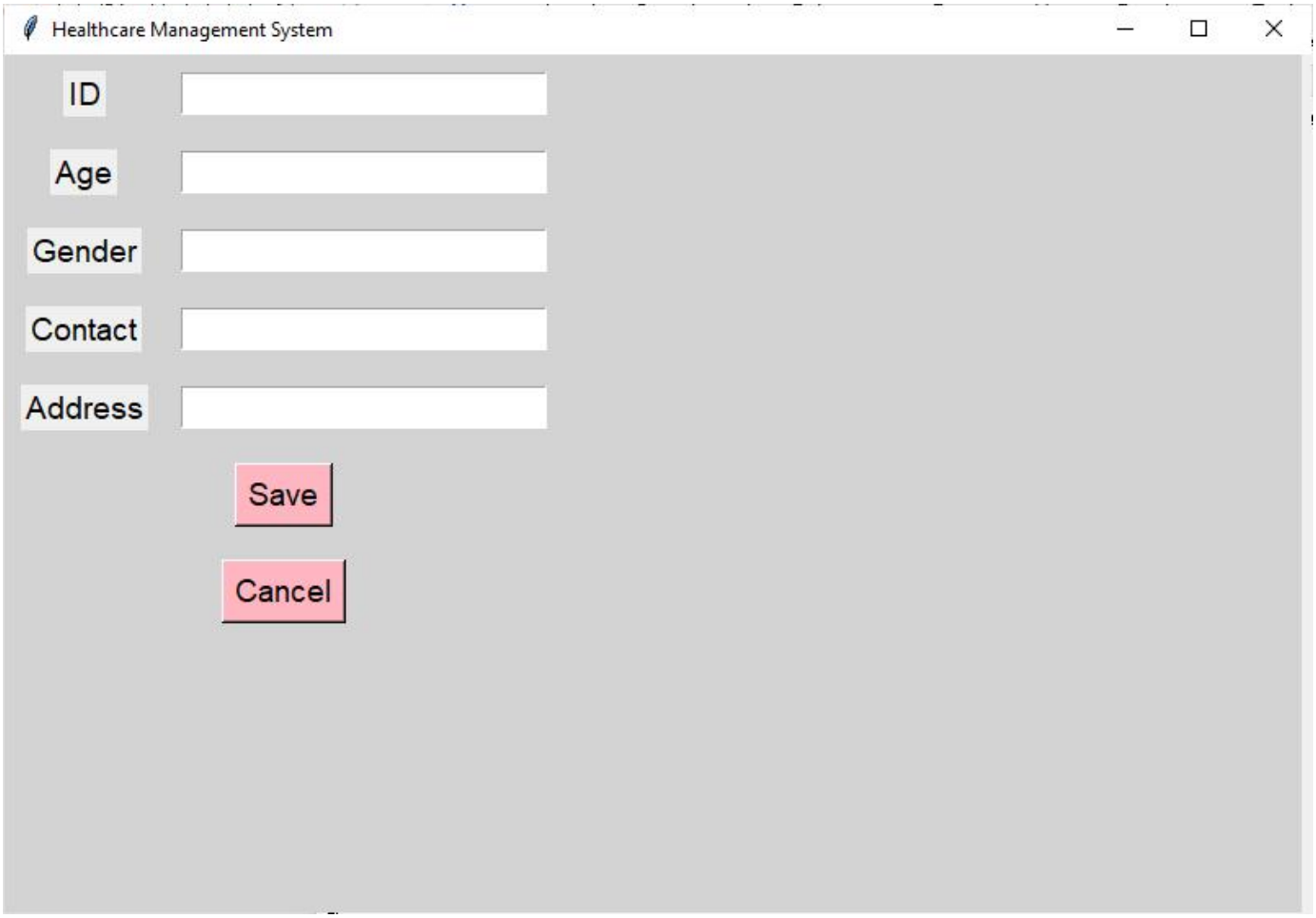


ADD EDIT PATIENT FORM

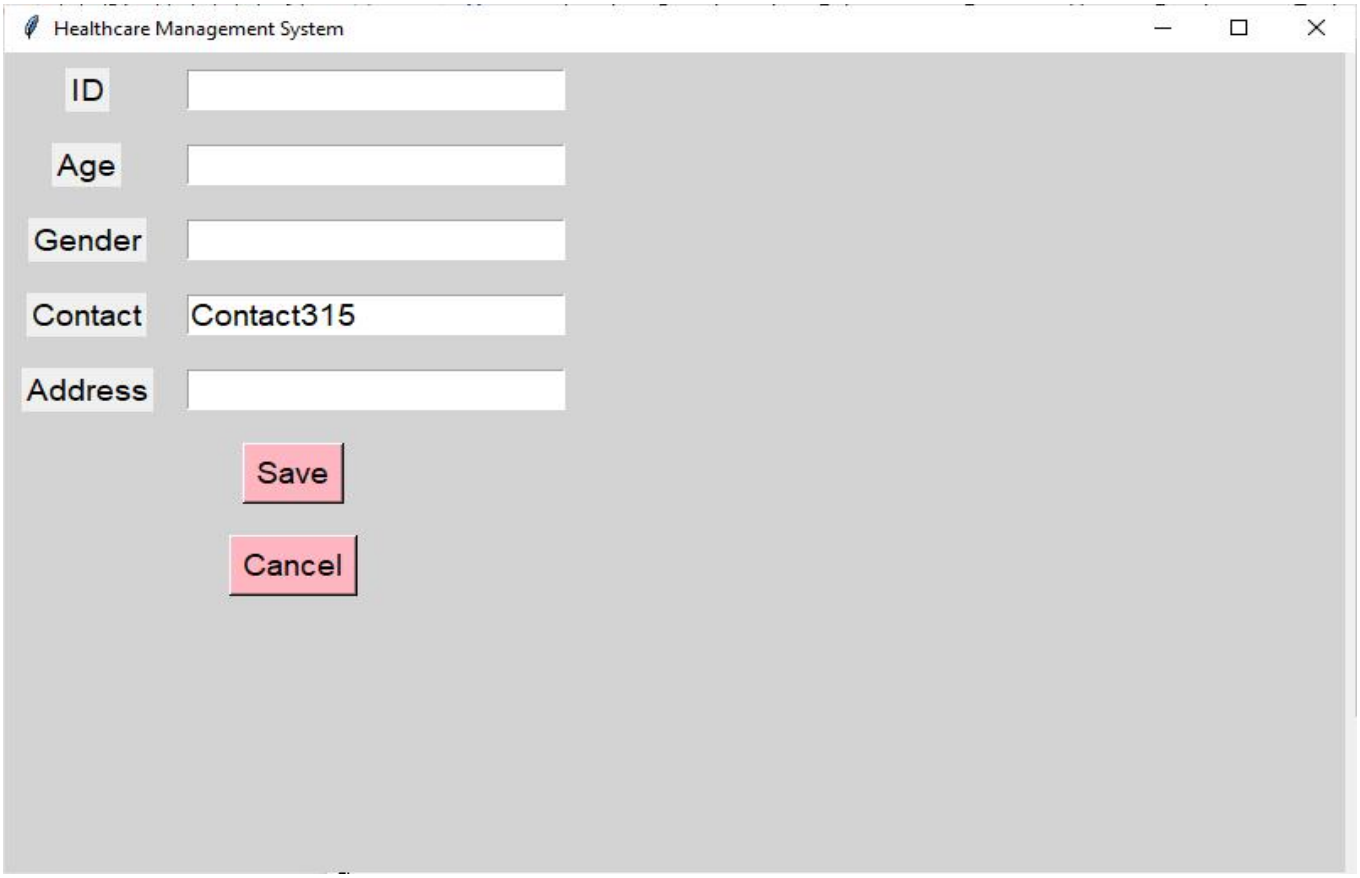
```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_non_query, execute_query

class AddEditPatientForm(tk.Frame):
    def __init__(self, parent, controller, bg_color, button_color="light pink"):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.patient_id = None
        self.create_widgets()

    def create_widgets(self):
        self.label_id = tk.Label(self, text="ID")
        self.label_id.grid(row=0, column=0, padx=10, pady=10)
        self.entry_id = tk.Entry(self)
        self.entry_id.grid(row=0, column=1, padx=10, pady=10)

        self.label_age = tk.Label(self, text="Age")
        self.label_age.grid(row=1, column=0, padx=10, pady=10)
        self.entry_age = tk.Entry(self)
        self.entry_age.grid(row=1, column=1, padx=10, pady=10)

        self.label_gender = tk.Label(self, text="Gender")
        self.label_gender.grid(row=2, column=0, padx=10, pady=10)
        self.entry_gender = tk.Entry(self)
        self.entry_gender.grid(row=2, column=1, padx=10, pady=10)

        self.label_contact = tk.Label(self, text="Contact")
        self.label_contact.grid(row=3, column=0, padx=10, pady=10)
        self.entry_contact = tk.Entry(self)
        self.entry_contact.grid(row=3, column=1, padx=10, pady=10)

        self.label_address = tk.Label(self, text="Address")
        self.label_address.grid(row=4, column=0, padx=10, pady=10)
       self.entry_address = tk.Entry(self)
```

```
        self.entry_address.grid(row=4, column=1, padx=10, pady=10)

        self.button_save = tk.Button(self, text="Save", command=self.save_patient, bg=self.button_color)
        self.button_save.grid(row=5, column=0, columnspan=2, pady=10)

        self.button_cancel = tk.Button(self, text="Cancel", command=lambda:
        self.controller.show_frame("PatientManagementForm"), bg=self.button_color)
        self.button_cancel.grid(row=6, column=0, columnspan=2, pady=10)

    def load_patient(self, patient_id):
        self.patient_id = patient_id
        query = "SELECT Contact FROM Patients WHERE PatientID = ?"
        patient = execute_query(query, (patient_id,))
        if patient:
            self.entry_contact.delete(0, tk.END)
            self.entry_contact.insert(0, patient[0][0])

    def save_patient(self):
        id = self.entry_id.get()
        age = self.entry_age.get()
        gender = self.entry_gender.get()
        contact = self.entry_contact.get()
        address = self.entry_address.get()
        if self.patient_id:
            query = "UPDATE Patients SET PatientID = ?, Age = ?, Gender = ?, Contact = ?, Address = ?  WHERE
            PatientID = ?"
            execute_non_query(query, (id, age, gender, contact, address, self.patient_id))
        else:
            query = "INSERT INTO Patients (PatientID, Age, Gender, Contact, Address) VALUES (?, ?, ?, ?, ?)"
            execute_non_query(query, (id, age, gender, contact, address))
        self.controller.show_frame("PatientManagementForm")
        self.controller.frames["PatientManagementForm"].load_patients()

if __name__ == "__main__":
    root = tk.Tk()
    frame = AddEditPatientForm(root, None, "Light Grey", "light pink")
    frame.pack()
    root.mainloop()
```

ADD FORM :



EDIT FORM :

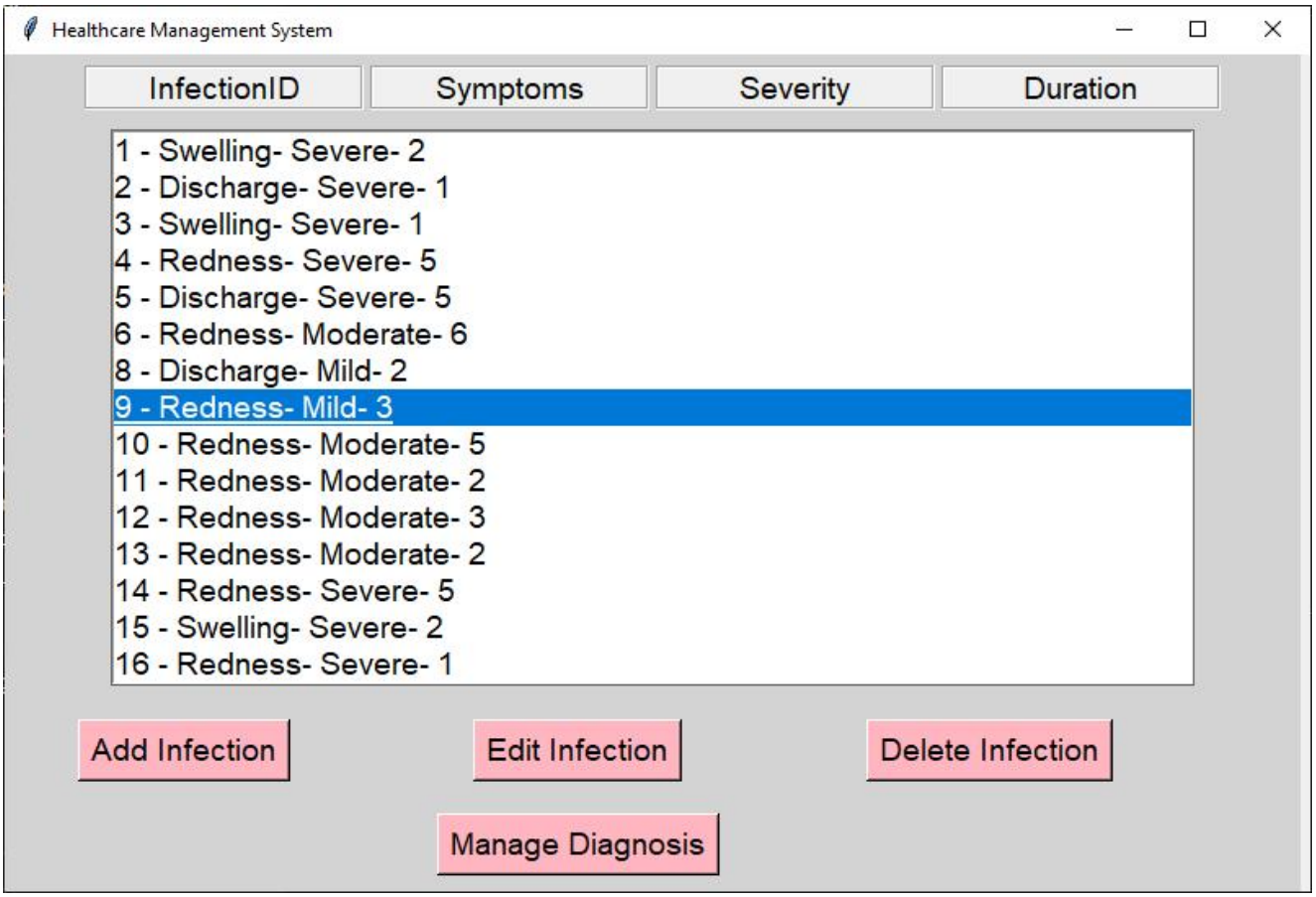


INFECTION MANAGEMENT FORM

```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_query, execute_non_query

class InfectionManagementForm(tk.Frame):
    def __init__(self, parent, controller, bg_color, button_color):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.create_widgets()

    def create_widgets(self):
        # Create a frame for the heading
        heading_frame = tk.Frame(self, bg=self["bg"])
        heading_frame.grid(row=0, column=0, columnspan=4, padx=5, pady=(5, 0))

        # Add headings as labels
        headings = ["InfectionID", "Symptoms", "Severity", "Duration"]
        for col_num, heading in enumerate(headings):
            label = tk.Label(heading_frame, text=heading, borderwidth=2, relief="groove", width=15)
            label.grid(row=0, column=col_num, padx=2, pady=2)

        # Create the Listbox to display infections
        self.infection_listbox = tk.Listbox(self, width=60, height=15)
        self.infection_listbox.grid(row=1, column=0, columnspan=4, padx=10, pady=10)

        # Create and place the buttons
        button_frame = tk.Frame(self, bg=self["bg"])
        button_frame.grid(row=5, column=0, columnspan=4, pady=10)

        self.button_add = tk.Button(self, text="Add Infection", bg=self.button_color,
command=lambda: self.controller.show_frame("AddEditInfectionForm"))
        self.button_add.grid(row=5, column=0, padx=10, pady=10)
```

```
        self.button_edit = tk.Button(self, text="Edit Infection", bg=self.button_color,
command=self.edit_infection)
        self.button_edit.grid(row=5, column=1, padx=10, pady=10)

        self.button_delete = tk.Button(self, text="Delete Infection", bg=self.button_color,
command=self.delete_infection)
        self.button_delete.grid(row=5, column=2, padx=10, pady=10)

        self.button_manage_diagnosis = tk.Button(self, text="Manage Diagnosis", command=lambda:
self.controller.show_frame("DiagnosisManagementForm"), bg=self.button_color)
        self.button_manage_diagnosis.grid(row=6, column=1, padx=10, pady=10)

        # Configure the grid
        self.grid_rowconfigure(1, weight=1)
        self.grid_columnconfigure(0, weight=1)
        self.grid_columnconfigure(1, weight=1)
        self.grid_columnconfigure(2, weight=1)
        self.grid_columnconfigure(3, weight=1)

        # Load the infections data
        self.load_infections()

    def load_infections(self):
        self.infection_listbox.delete(0, tk.END)
        query = "SELECT * FROM pinkeyeinfection"
        infections = execute_query(query)
        for infection in infections:
            self.infection_listbox.insert(tk.END, f"{infection[0]} - {infection[1]}- {infection[2]}-
{infection[3]}")

    def edit_infection(self):
        selected = self.infection_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No infection selected")
            return
        infection_id = self.infection_listbox.get(selected).split(" - ")[0]
```

```
        self.controller.frames["AddEditInfectionForm"].load_infection(infection_id)
        self.controller.show_frame("AddEditInfectionForm")

    def delete_infection(self):
        selected = self.infection_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No infection selected")
            return
        infection_id = self.infection_listbox.get(selected).split(" - ")[0]
        # Delete related entries from the child tables
        delete_diagnosis_query = "DELETE FROM diagnosis WHERE infectionID = ?"
        delete_treatment_query = "DELETE FROM treatment WHERE infectionID = ?"
        delete_assigned_query = "DELETE FROM assigned WHERE infectionID = ?"
        delete_etiology_query = "DELETE FROM etiology WHERE infectionID = ?"

        # Call execute_non_query for each deletion
        execute_non_query(delete_diagnosis_query, (infection_id,))
        execute_non_query(delete_treatment_query, (infection_id,))
        execute_non_query(delete_assigned_query, (infection_id,))
        execute_non_query(delete_etiology_query, (infection_id,))

        # Delete the entry from the parent table
        delete_infection_query = "DELETE FROM pinkeyeinfection WHERE infectionID = ?"
        execute_non_query(delete_infection_query, (infection_id,))
        self.load_infections()

if __name__ == "__main__":
    root = tk.Tk()
    frame = InfectionManagementForm(root, None, "Light Grey", "light pink")
    frame.pack(expand=True, fill=tk.BOTH)
    root.title("Infection Management")
    root.mainloop()
```



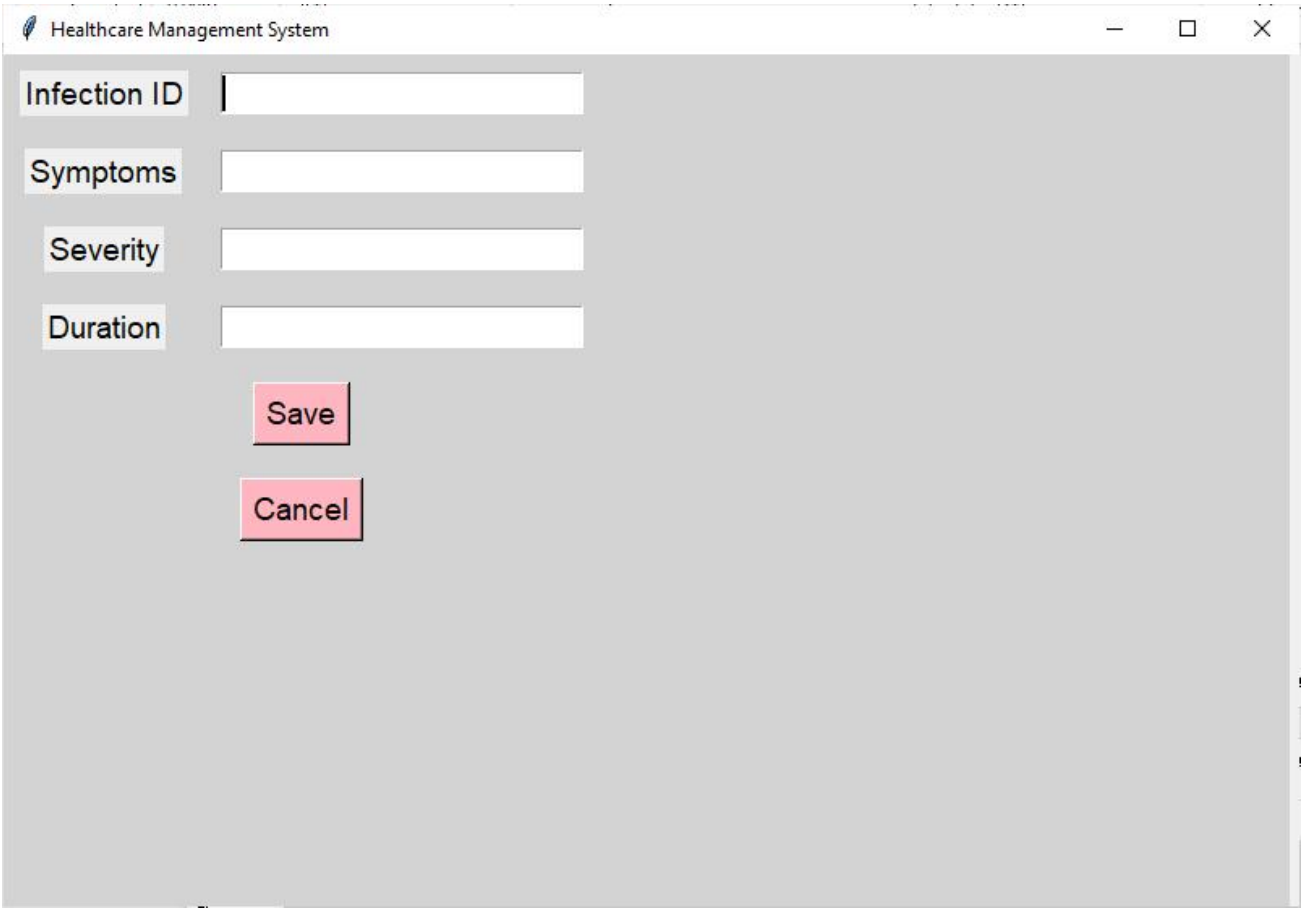
ADD EDIT INFECTION FORM

```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_non_query, execute_query

class AddEditInfectionForm(tk.Frame):
    def __init__(self, parent, controller, bg_color, button_color):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.infection_id = None
        self.create_widgets()

    def create_widgets(self):

        self.label_IID = tk.Label(self, text="Infection ID")
        self.label_IID.grid(row=0, column=0, padx=10, pady=10)

        self.entry_IID = tk.Entry(self)
        self.entry_IID.grid(row=0, column=1, padx=10, pady=10)
```

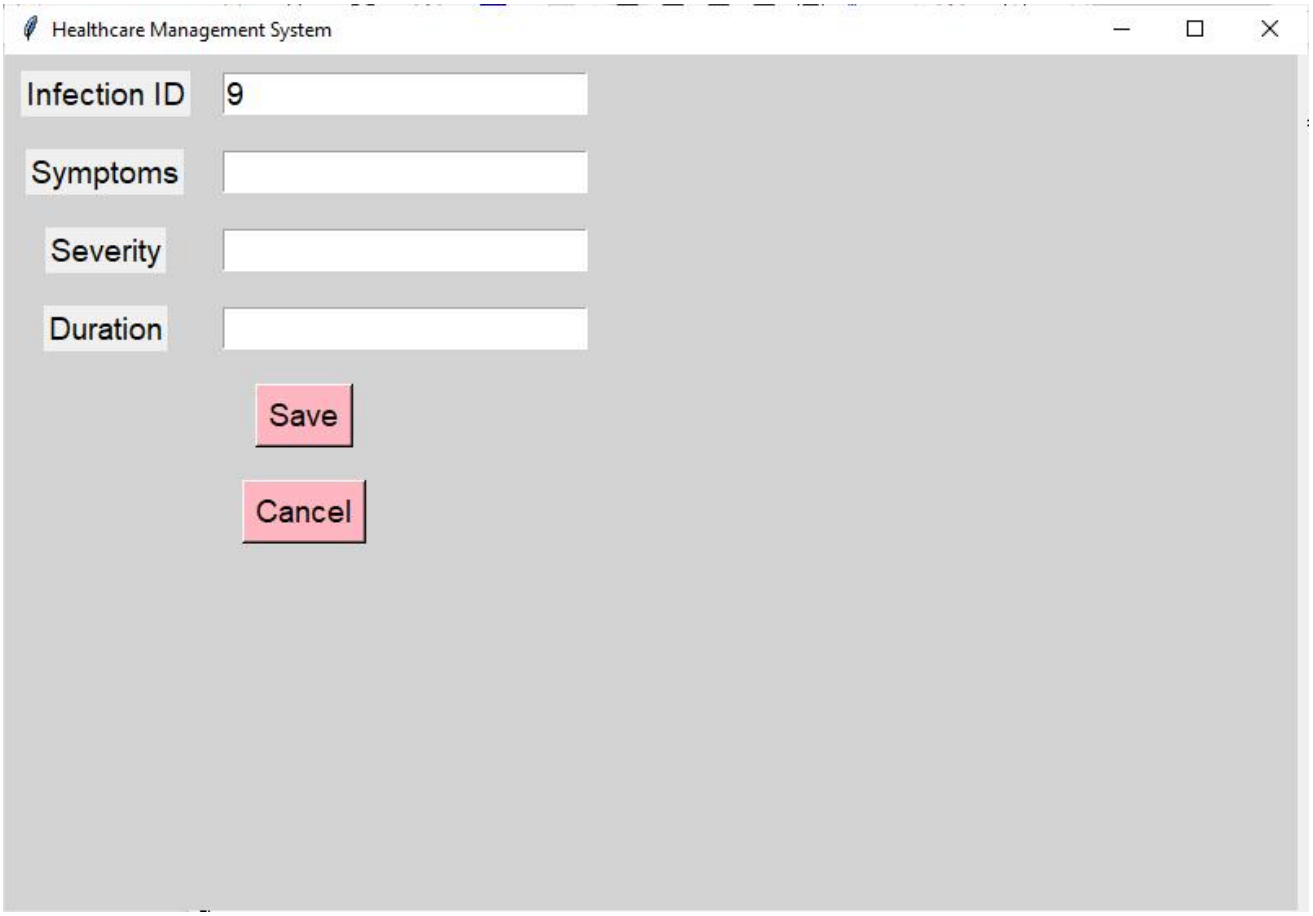
```
        self.label_symptoms = tk.Label(self, text="Symptoms")
        self.label_symptoms.grid(row=1, column=0, padx=10, pady=10)

        self.entry_symptoms = tk.Entry(self)
        self.entry_symptoms.grid(row=1, column=1, padx=10, pady=10)

        self.label_severity = tk.Label(self, text="Severity")
        self.label_severity.grid(row=2, column=0, padx=10, pady=10)

        self.entry_severity = tk.Entry(self)
        self.entry_severity.grid(row=2, column=1, padx=10, pady=10)

        self.label_duration = tk.Label(self, text="Duration")
        self.label_duration.grid(row=3, column=0, padx=10, pady=10)

        self.entry_duration = tk.Entry(self)
        self.entry_duration.grid(row=3, column=1, padx=10, pady=10)



        self.button_save = tk.Button(self, text="Save", bg=self.button_color, command=self.save_infection)
        self.button_save.grid(row=4, column=0, columnspan=2, pady=10)

        self.button_cancel = tk.Button(self, text="Cancel", bg=self.button_color, command=lambda:
self.controller.show_frame("InfectionManagementForm"))
        self.button_cancel.grid(row=5, column=0, columnspan=2, pady=10)

    def load_infection(self, infection_id):
        self.infection_id = infection_id
        query = "SELECT infectionID FROM pinkeyeinfection WHERE infectionID = ?"
        infection = execute_query(query, (infection_id,))
        if infection:
            self.entry_IID.delete(0, tk.END)
            self.entry_IID.insert(0, infection[0][0])

    def save_infection(self):
        IID = self.entry_IID.get()
        symptoms = self.entry_symptoms.get()
        severity = self.entry_severity.get()
```

```
        duration = self.entry_duration.get()
        if self.infection_id:
            query = "UPDATE pinkeyeinfection SET infectionID = ?, symptoms = ?, severity = ?, duration = ? WHERE infectionID = ?"
            execute_non_query(query, (IID, symptoms, severity, duration, self.infection_id))
        else:
            query = "INSERT INTO pinkeyeinfection (infectionID, symptoms, severity, duration) VALUES (?, ?, ?, ?)"
            execute_non_query(query, (IID, symptoms, severity, duration))
        self.controller.show_frame("InfectionManagementForm")
        self.controller.frames["InfectionManagementForm"].load_infections()

if __name__ == "__main__":
    root = tk.Tk()
    frame = AddEditInfectionForm(root, None, "Light Grey", "light pink")
    frame.pack()
    root.mainloop()
```

ADD FORM :



EDIT FORM :

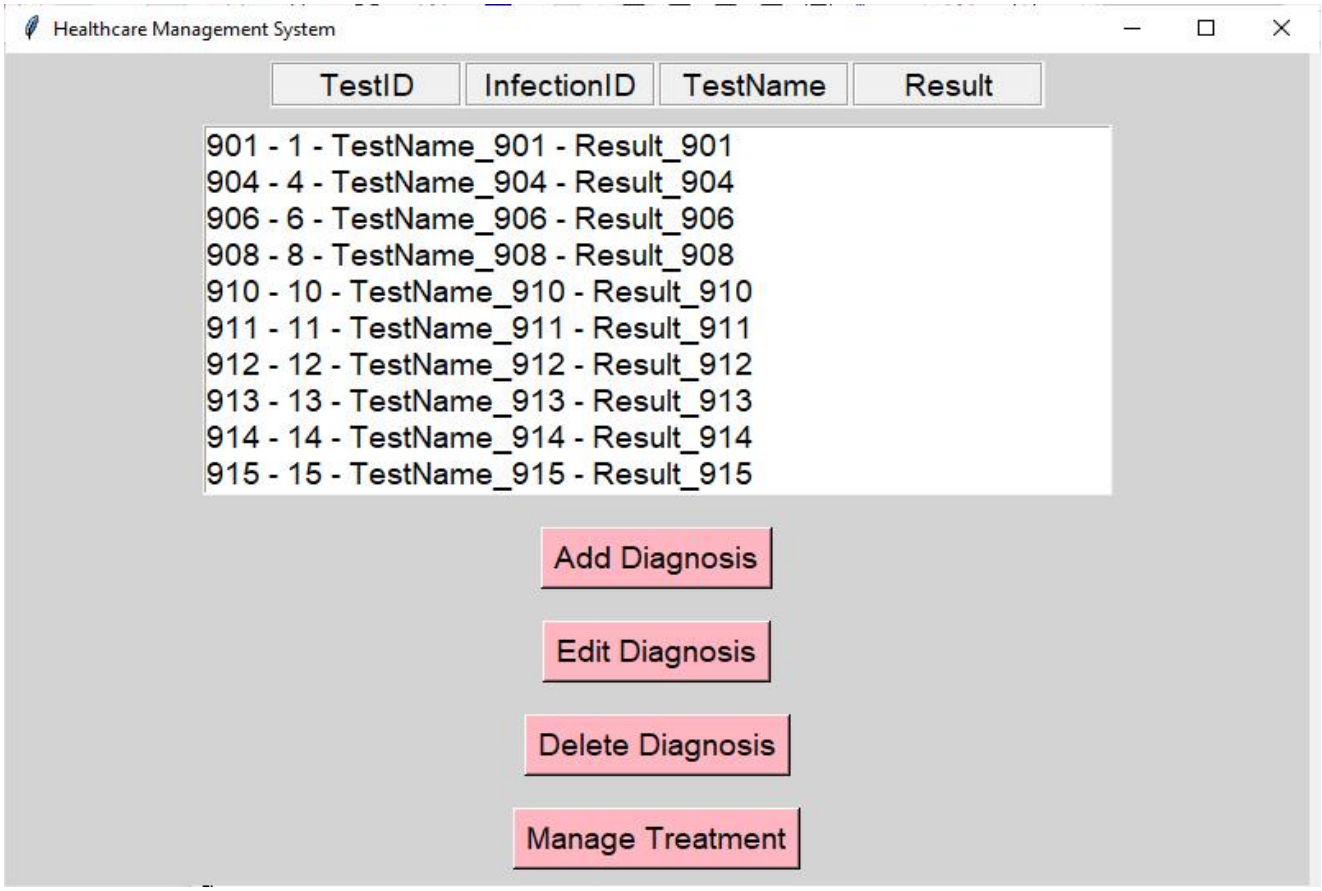


DIAGNOSIS MANAGEMENT FORM

```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_query, execute_non_query

class DiagnosisManagementForm(tk.Frame):
    def __init__(self, parent, controller, bg_color, button_color="light pink"):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.create_widgets()

    def create_widgets(self):
        # Create a frame for the heading
        heading_frame = tk.Frame(self)
        heading_frame.grid(row=0, column=0, columnspan=1, padx=5, pady=(5, 0))

        # Add headings as labels
        headings = ["TestID", "InfectionID", "TestName", "Result"]
        for col_num, heading in enumerate(headings):
```

```
            label = tk.Label(heading_frame, text=heading, borderwidth=2, relief="groove", width=10)
            label.grid(row=0, column=col_num, padx=1, pady=1)

        self.diagnosis_listbox = tk.Listbox(self, width=50)
        self.diagnosis_listbox.grid(row=1, column=0, columnspan=3, padx=10, pady=10)

        # Configure the Listbox frame grid
        self.grid_rowconfigure(1, weight=1)
        self.grid_columnconfigure(0, weight=1)

        self.button_add = tk.Button(self, text="Add Diagnosis", command=lambda:
self.controller.show_frame("AddEditDiagnosisForm"), bg=self.button_color)
        self.button_add.grid(row=2, column=0, padx=10, pady=10)

        self.button_edit = tk.Button(self, text="Edit Diagnosis", command=self.edit_diagnosis, bg=self.button_color)
        self.button_edit.grid(row=3, column=0, padx=10, pady=10)

        self.button_delete = tk.Button(self, text="Delete Diagnosis", command=self.delete_diagnosis,
bg=self.button_color)
        self.button_delete.grid(row=4, column=0, padx=10, pady=10)

        self.button_manage_treatment = tk.Button(self, text="Manage Treatment", command=lambda:
self.controller.show_frame("TreatmentManagementForm"), bg=self.button_color)
        self.button_manage_treatment.grid(row=5, column=0, padx=10, pady=10)

        self.load_diagnoses()

    def load_diagnoses(self):
        self.diagnosis_listbox.delete(0, tk.END)
        query = "SELECT * FROM diagnosis"
        diagnoses = execute_query(query)
        for diagnosis in diagnoses:
            self.diagnosis_listbox.insert(tk.END, f"{diagnosis[0]} - {diagnosis[1]} - {diagnosis[2]} - {diagnosis[3]}")

    def edit_diagnosis(self):
        selected = self.diagnosis_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No diagnosis selected")
            return
```

```
        diagnosis_id = self.diagnosis_listbox.get(selected).split(" - ")[0]
        self.controller.frames["AddEditDiagnosisForm"].load_diagnosis(diagnosis_id)
        self.controller.show_frame("AddEditDiagnosisForm")

    def delete_diagnosis(self):
        selected = self.diagnosis_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No diagnosis selected")
            return
        diagnosis_id = self.diagnosis_listbox.get(selected).split(" - ")[0]
        query = "DELETE FROM diagnosis WHERE TestID = ?"
        execute_non_query(query, (diagnosis_id,))
        self.load_diagnoses()

if __name__ == "__main__":
    root = tk.Tk()
    frame = DiagnosisManagementForm(root, None, "Light Grey", "light pink")
    frame.pack()
    root.mainloop()
```

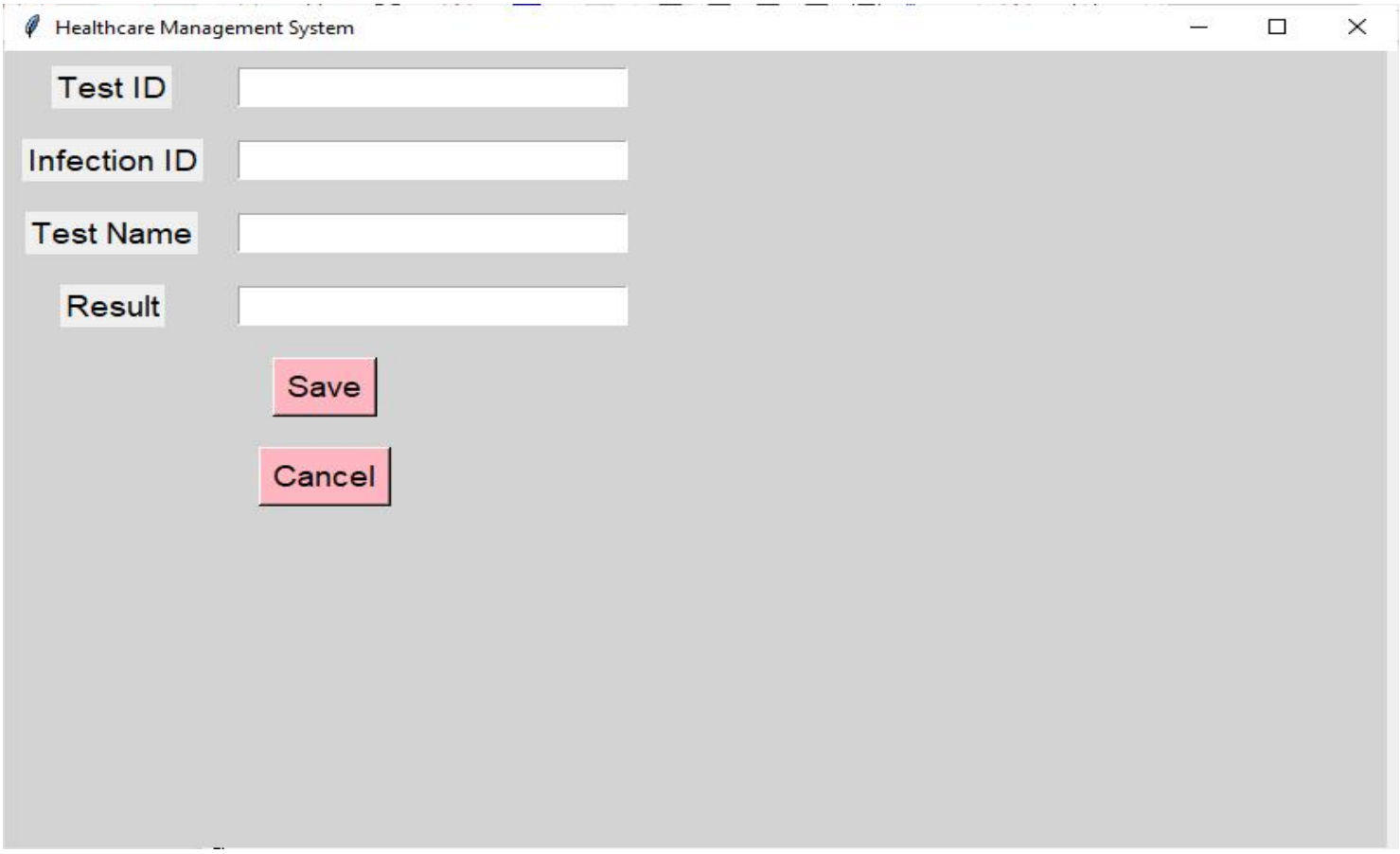


ADD EDIT DIAGNOSIS FORM

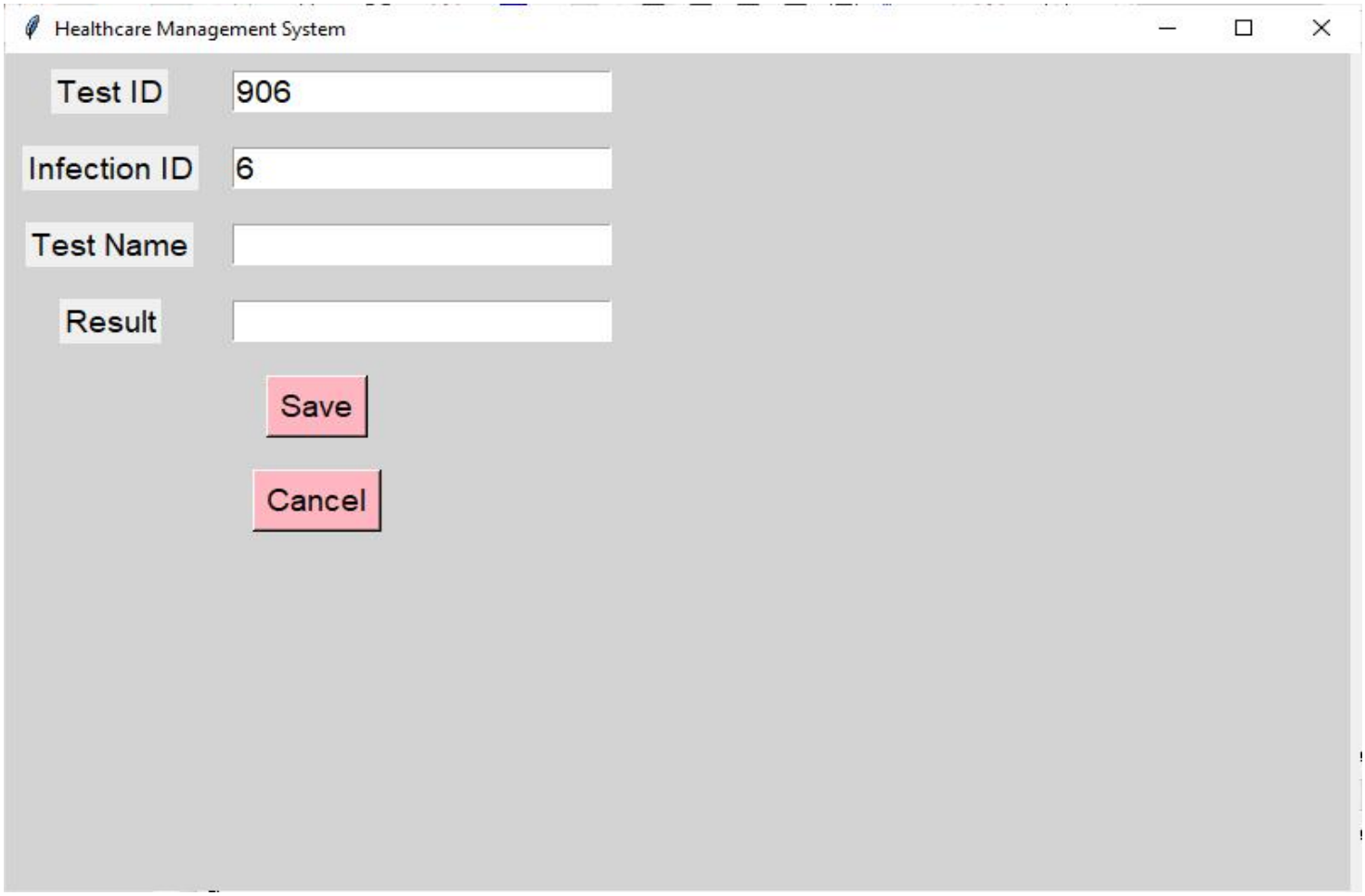
```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_non_query, execute_query

class AddEditDiagnosisForm(tk.Frame):
    def __init__(self, parent, controller, bg_color, button_color="light pink"):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.diagnosis_id = None
        self.create_widgets()

    def create_widgets(self):
        self.label_TID = tk.Label(self, text="Test ID")
        self.label_TID.grid(row=0, column=0, padx=10, pady=10)

        self.entry_TID = tk.Entry(self)
        self.entry_TID.grid(row=0, column=1, padx=10, pady=10)

        self.label_IID = tk.Label(self, text="Infection ID")
        self.label_IID.grid(row=1, column=0, padx=10, pady=10)

        self.entry_IID = tk.Entry(self)
        self.entry_IID.grid(row=1, column=1, padx=10, pady=10)

        self.label_name = tk.Label(self, text="Test Name")
        self.label_name.grid(row=2, column=0, padx=10, pady=10)

        self.entry_name = tk.Entry(self)
        self.entry_name.grid(row=2, column=1, padx=10, pady=10)

        self.label_result = tk.Label(self, text="Result")
        self.label_result.grid(row=3, column=0, padx=10, pady=10)

        self.entry_result = tk.Entry(self)
        self.entry_result.grid(row=3, column=1, padx=10, pady=10)
```

```
        self.button_save = tk.Button(self, text="Save", command=self.save_diagnosis, bg=self.button_color)
        self.button_save.grid(row=4, column=0, columnspan=2, pady=10)

        self.button_cancel = tk.Button(self, text="Cancel", command=lambda:
self.controller.show_frame("DiagnosisManagementForm"), bg=self.button_color)
        self.button_cancel.grid(row=5, column=0, columnspan=2, pady=10)

    def load_diagnosis(self, diagnosis_id):
        self.diagnosis_id = diagnosis_id
        query = "SELECT testID, infectionID FROM diagnosis WHERE TestID = ?"
        diagnosis = execute_query(query, (diagnosis_id,))
        if diagnosis:
            self.entry_TID.delete(0, tk.END)
            self.entry_TID.insert(0, diagnosis[0][0])
            self.entry_IID.delete(0, tk.END)
            self.entry_IID.insert(0, diagnosis[0][1])

    def save_diagnosis(self):
        TestID = self.entry_TID.get()
        infectionID = self.entry_IID.get()
        Testname = self.entry_name.get()
        result = self.entry_result.get()
        if self.diagnosis_id:
            query = "UPDATE diagnosis SET TestID = ?, infectionID = ?, testname = ?, result = ? WHERE TestID = ?"
            execute_non_query(query, (TestID, infectionID, Testname, result, self.diagnosis_id))
        else:
            query = "INSERT INTO diagnosis (TestID, infectionID, testname, result) VALUES (?, ?, ?, ?)"
            execute_non_query(query, (TestID, infectionID, Testname, result))
        self.controller.show_frame("DiagnosisManagementForm")
        self.controller.frames["DiagnosisManagementForm"].load_diagnoses()

if __name__ == "__main__":
    root = tk.Tk()
    frame = AddEditDiagnosisForm(root, None, "Light Grey", "light pink")
    frame.pack()
    root.mainloop()
```

ADD FORM :



EDIT FORM :

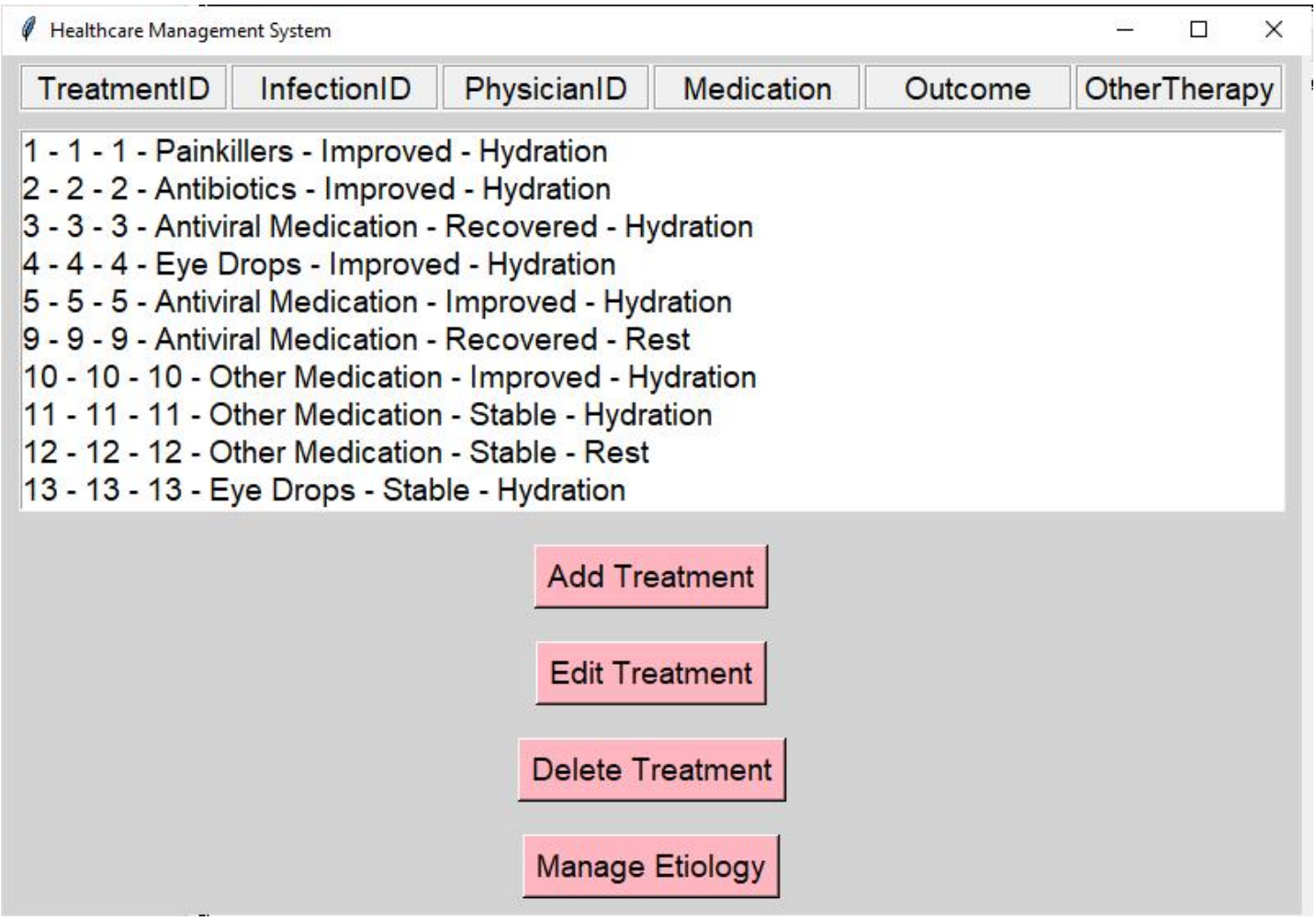


TREATMENT MANAGEMENT FORM

```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_query, execute_non_query

class TreatmentManagementForm(tk.Frame):
    def __init__(self, parent, controller, bg_color, button_color="light pink"):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.create_widgets()

    def create_widgets(self):
        # Create a frame for the heading
        heading_frame = tk.Frame(self)
        heading_frame.grid(row=0, column=0, columnspan=1, padx=5, pady=(5, 0))

        # Add headings as labels
        headings = ["TreatmentID", "InfectionID", "PhysicianID", "Medication", "Outcome", "OtherTherapy"]
        for col_num, heading in enumerate(headings):
            label = tk.Label(heading_frame, text=heading, borderwidth=2, relief="groove", width=11)
            label.grid(row=0, column=col_num, padx=1, pady=1)

        self.treatment_listbox = tk.Listbox(self, width=70)
        self.treatment_listbox.grid(row=1, column=0, columnspan=3, padx=10, pady=10)

        # Configure the Listbox frame grid
        self.grid_rowconfigure(1, weight=1)
        self.grid_columnconfigure(0, weight=1)

        self.button_add = tk.Button(self, text="Add Treatment", command=lambda:
self.controller.show_frame("AddEditTreatmentForm"), bg=self.button_color)
        self.button_add.grid(row=2, column=0, padx=10, pady=10)

        self.button_edit = tk.Button(self, text="Edit Treatment", command=self.edit_treatment, bg=self.button_color)
        self.button_edit.grid(row=3, column=0, padx=10, pady=10)
```

```
        self.button_delete = tk.Button(self, text="Delete Treatment", command=self.delete_treatment,
bg=self.button_color)
        self.button_delete.grid(row=4, column=0, padx=10, pady=10)

        self.button_manage_etiology = tk.Button(self, text="Manage Etiology", command=lambda:
self.controller.show_frame("EtiologyManagementForm"), bg=self.button_color)
        self.button_manage_etiology.grid(row=5, column=0, padx=10, pady=10)

        self.load_treatments()

    def load_treatments(self):
        self.treatment_listbox.delete(0, tk.END)
        query = "SELECT * FROM treatment"
        treatments = execute_query(query)
        for treatment in treatments:
            self.treatment_listbox.insert(tk.END, f"{treatment[0]} - {treatment[1]} - {treatment[2]} - {treatment[3]} -
{treatment[4]} - {treatment[5]}")

    def edit_treatment(self):
        selected = self.treatment_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No treatment selected")
            return
        treatment_id = self.treatment_listbox.get(selected).split(" - ")[0]
        self.controller.frames["AddEditTreatmentForm"].load_treatment(treatment_id)
        self.controller.show_frame("AddEditTreatmentForm")

    def delete_treatment(self):
        selected = self.treatment_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No treatment selected")
            return
        treatment_id = self.treatment_listbox.get(selected).split(" - ")[0]
        query = "DELETE FROM treatment WHERE treatmentID = ?"
        execute_non_query(query, (treatment_id,))
        self.load_treatments()

if __name__ == "__main__":
    root = tk.Tk()
```

```
frame = TreatmentManagementForm(root, None, "Light Grey", "light pink")
frame.pack()
root.mainloop()
```



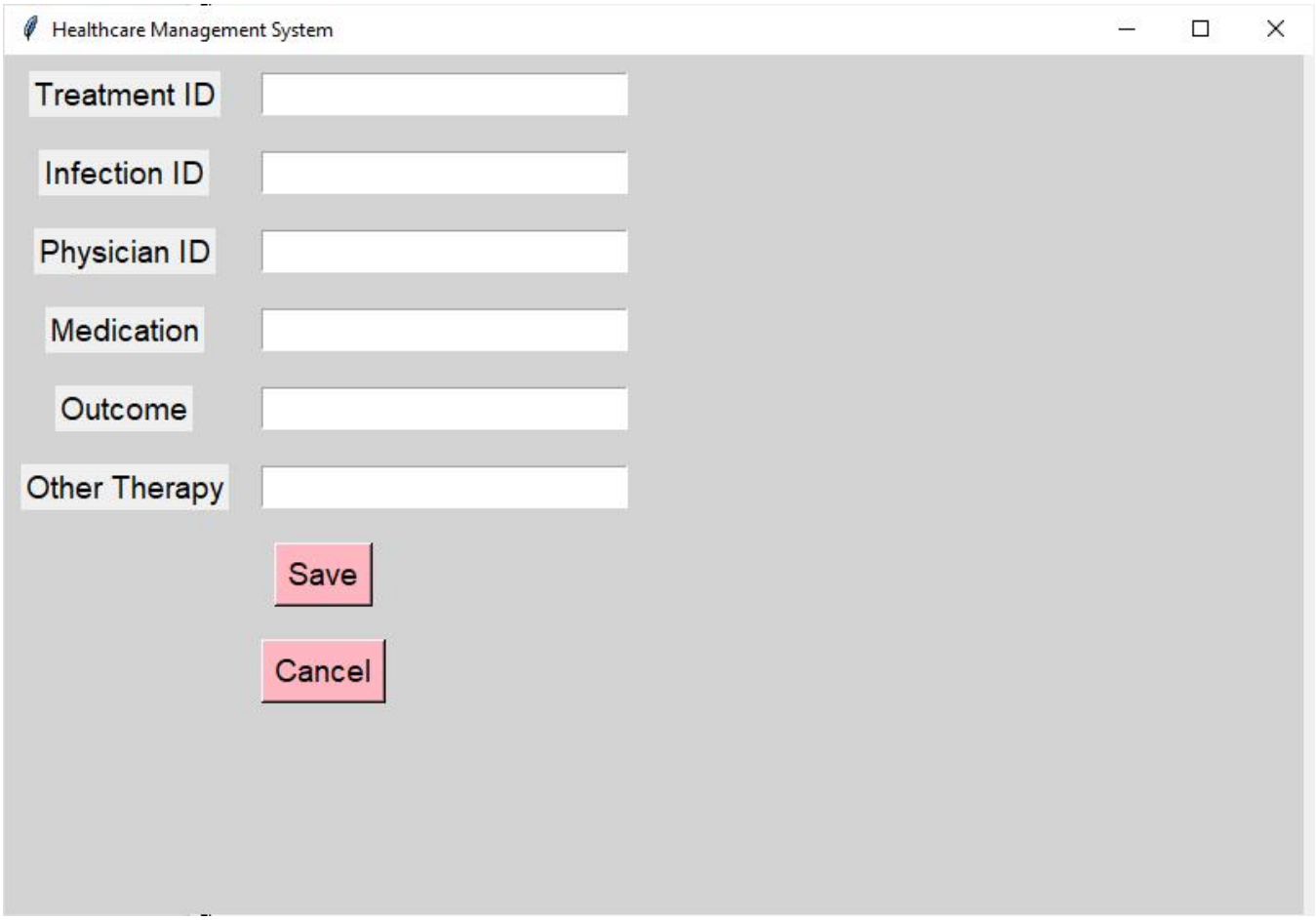
ADD EDIT TREATMENT FORM

```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_non_query, execute_query

class AddEditTreatmentForm(tk.Frame):
    def __init__(self, parent, controller, bg_color,button_color="light pink"):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.treatment_id = None
        self.create_widgets()

    def create_widgets(self):
        self.label_TID = tk.Label(self, text="Treatment ID")
```

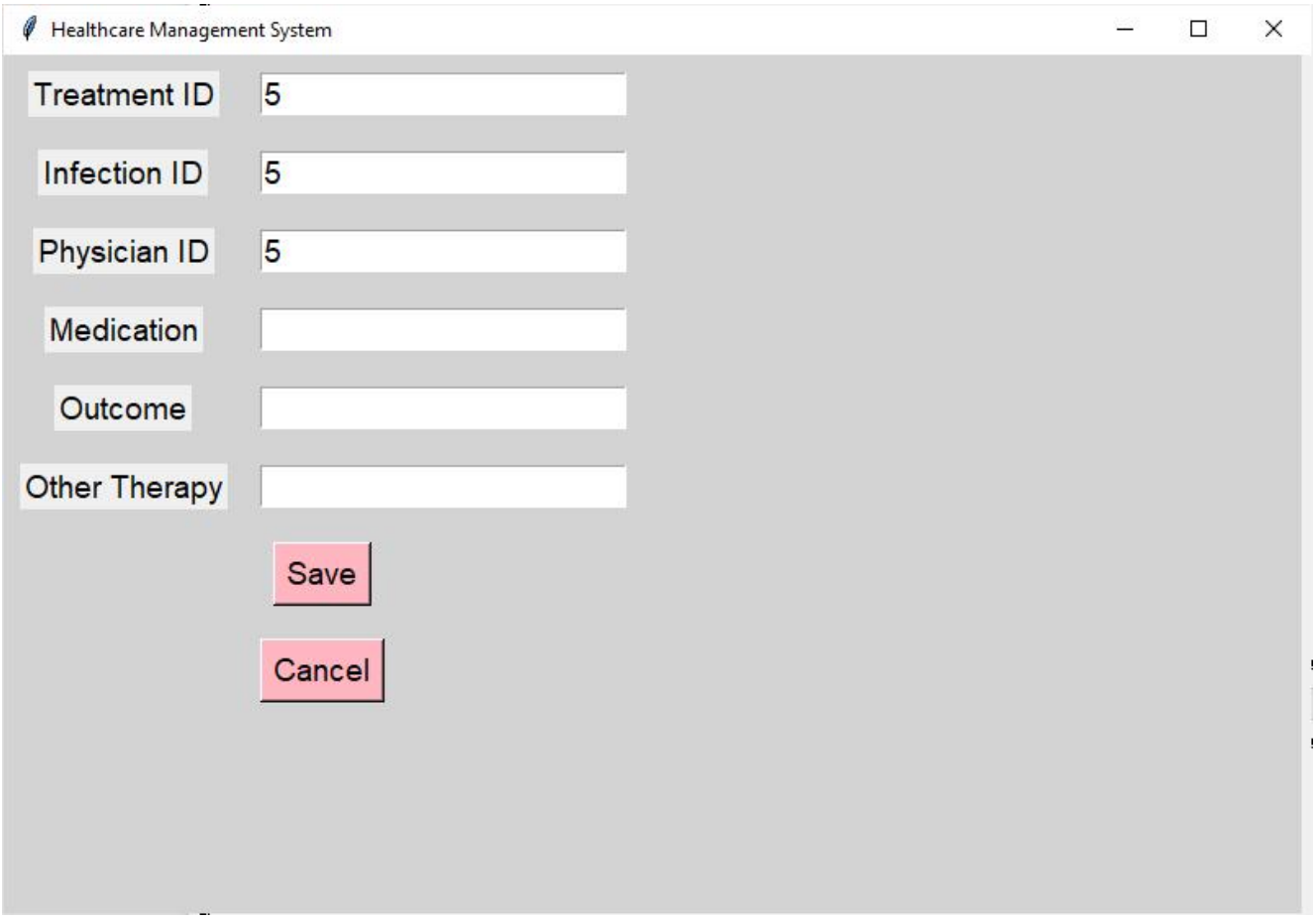
```
self.label_TID.grid(row=0, column=0, padx=10, pady=10)

self.entry_TID = tk.Entry(self)
self.entry_TID.grid(row=0, column=1, padx=10, pady=10)

self.label_IID = tk.Label(self, text="Infection ID")
self.label_IID.grid(row=1, column=0, padx=10, pady=10)

self.entry_IID = tk.Entry(self)
self.entry_IID.grid(row=1, column=1, padx=10, pady=10)

self.label_PID = tk.Label(self, text="Physician ID")
self.label_PID.grid(row=2, column=0, padx=10, pady=10)

self.entry_PID = tk.Entry(self)
self.entry_PID.grid(row=2, column=1, padx=10, pady=10)

self.label_medication = tk.Label(self, text="Medication")
self.label_medication.grid(row=3, column=0, padx=10, pady=10)

self.entry_medication = tk.Entry(self)
self.entry_medication.grid(row=3, column=1, padx=10, pady=10)

self.label_outcome = tk.Label(self, text="Outcome")
self.label_outcome.grid(row=4, column=0, padx=10, pady=10)

self.entry_outcome = tk.Entry(self)
self.entry_outcome.grid(row=4, column=1, padx=10, pady=10)

self.label_other = tk.Label(self, text="Other Therapy")
self.label_other.grid(row=5, column=0, padx=10, pady=10)

self.entry_other = tk.Entry(self)
self.entry_other.grid(row=5, column=1, padx=10, pady=10)

self.button_save = tk.Button(self, text="Save", command=self.save_treatment,bg=self.button_color)
self.button_save.grid(row=6, column=0, columnspan=2, pady=10)
```

```
        self.button_cancel = tk.Button(self, text="Cancel", command=lambda:
self.controller.show_frame("TreatmentManagementForm"),bg=self.button_color)
        self.button_cancel.grid(row=7, column=0, columnspan=2, pady=10)

    def load_treatment(self, treatment_id):
        self.treatment_id = treatment_id
        query = "SELECT treatmentID, infectionID, physicianID FROM treatment WHERE treatmentID = ?"
        treatment = execute_query(query, (treatment_id,))
        if treatment:
            self.entry_TID.delete(0, tk.END)
            self.entry_TID.insert(0, treatment[0][0])
            self.entry_IID.delete(0, tk.END)
            self.entry_IID.insert(0, treatment[0][1])
            self.entry_PID.delete(0, tk.END)
            self.entry_PID.insert(0, treatment[0][2])

    def save_treatment(self):
        TID = self.entry_TID.get()
        # IID = self.entry_IID.get()
        # PID = self.entry_PID.get()
        medication = self.entry_medication.get()
        outcome = self.entry_outcome.get()
        other = self.entry_other.get()
        if self.treatment_id:
            query = "UPDATE treatment SET treatmentID = ?, medication = ?, outcome = ?, otherTherapy = ? WHERE
treatmentID = ?"
            execute_non_query(query, (TID, medication, outcome, other, self.treatment_id))
        else:
            query = "INSERT INTO treatment (treatmentID, medication, outcome, otherTherapy) VALUES (?, ?, ?, ?)"
            execute_non_query(query, (TID, medication, outcome, other))
        self.controller.show_frame("TreatmentManagementForm")
        self.controller.frames["TreatmentManagementForm"].load_treatments()

if __name__ == "__main__":
    root = tk.Tk()
    frame = AddEditTreatmentForm(root, None,"Light Grey", "light pink")
    frame.pack()
    root.mainloop()
```

ADD FORM :



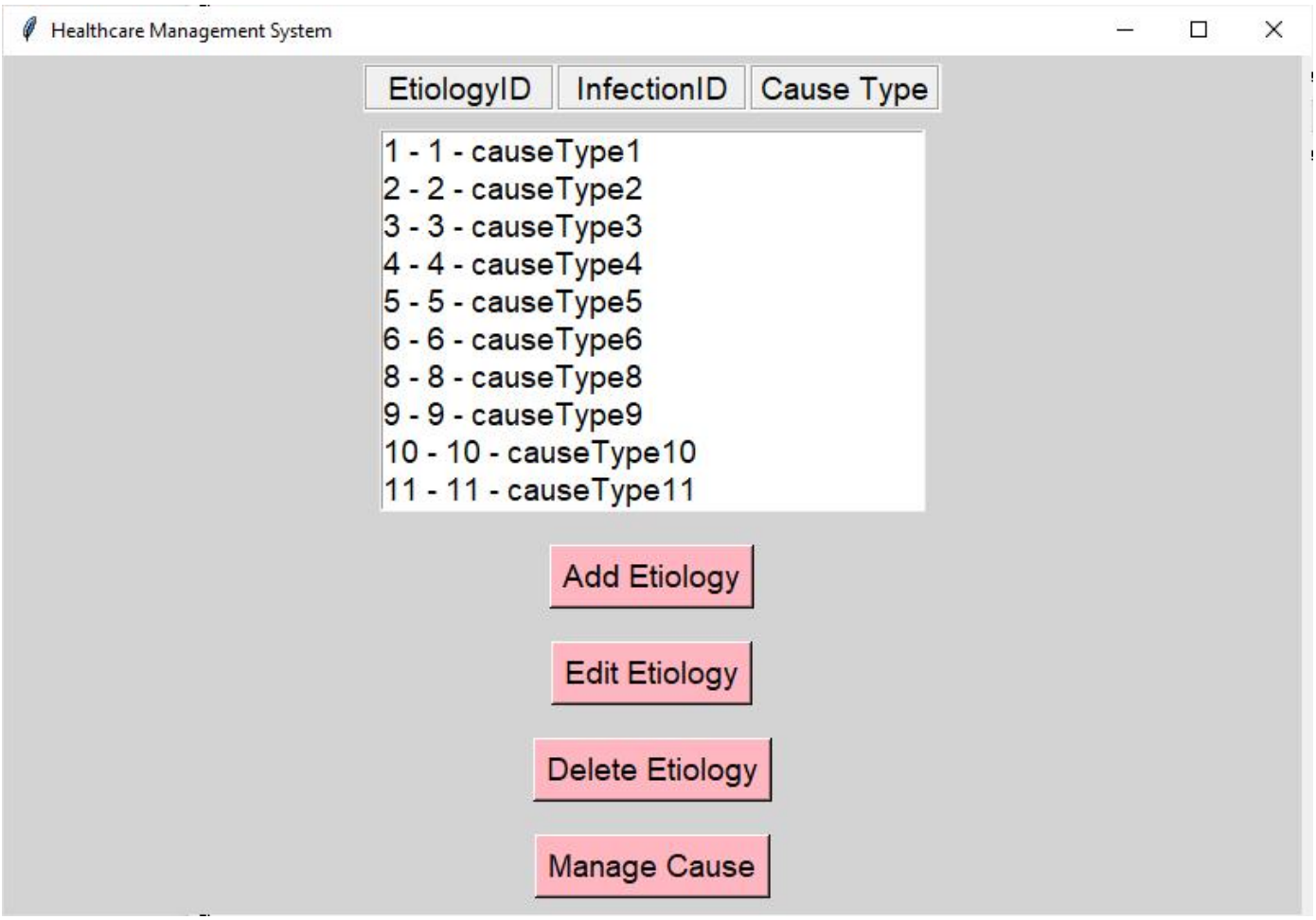
EDIT FORM :



ETIOLOGY MANAGEMENT FORM

```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_query, execute_non_query

class EtiologyManagementForm(tk.Frame):
    def __init__(self, parent, controller, bg_color, button_color="light pink"):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.create_widgets()

    def create_widgets(self):
        # Create a frame for the heading
        heading_frame = tk.Frame(self)
        heading_frame.grid(row=0, column=0, columnspan=1, padx=5, pady=(5, 0))

        # Add headings as labels
        headings = ["EtiologyID", "InfectionID", "Cause Type"]
        for col_num, heading in enumerate(headings):
            label = tk.Label(heading_frame, text=heading, borderwidth=2, relief="groove", width=10)
            label.grid(row=0, column=col_num, padx=1, pady=1)

        self.etiology_listbox = tk.Listbox(self, width=30)
        self.etiology_listbox.grid(row=1, column=0, columnspan=3, padx=10, pady=10)

        # Configure the Listbox frame grid
        self.grid_rowconfigure(1, weight=1)
        self.grid_columnconfigure(0, weight=1)

        self.button_add = tk.Button(self, text="Add Etiology", command=lambda:
self.controller.show_frame("AddEditEtiologyForm"), bg=self.button_color)
        self.button_add.grid(row=2, column=0, padx=10, pady=10)

        self.button_edit = tk.Button(self, text="Edit Etiology", command=self.edit_etiology, bg=self.button_color)
        self.button_edit.grid(row=3, column=0, padx=10, pady=10)

        self.button_delete = tk.Button(self, text="Delete Etiology", command=self.delete_etiology, bg=self.button_color)
```

```
        self.button_delete.grid(row=4, column=0, padx=10, pady=10)

        self.button_manage_cause = tk.Button(self, text="Manage Cause", command=lambda:
self.controller.show_frame("CauseManagementForm"), bg=self.button_color)
        self.button_manage_cause.grid(row=5, column=0, padx=10, pady=10)

        self.load_etiologies()

    def load_etiologies(self):
        self.etiology_listbox.delete(0, tk.END)
        query = "SELECT * FROM Etiology"
        etiologies = execute_query(query)
        for etiology in etiologies:
            self.etiology_listbox.insert(tk.END, f"{etiology[0]} - {etiology[1]} - {etiology[2]}")

    def edit_etiology(self):
        selected = self.etiology_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No etiology selected")
            return
        etiology_id = self.etiology_listbox.get(selected).split(" - ")[0]
        self.controller.frames["AddEditEtiologyForm"].load_etiology(etiology_id)
        self.controller.show_frame("AddEditEtiologyForm")

    def delete_etiology(self):
        selected = self.etiology_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No etiology selected")
            return
        etiology_id = self.etiology_listbox.get(selected).split(" - ")[0]
        query = "DELETE FROM Etiology WHERE etiologyID = ?"
        execute_non_query(query, (etiology_id,))
        self.load_etiologies()

if __name__ == "__main__":
    root = tk.Tk()
    frame = EtiologyManagementForm(root, None, "Light Grey", "light pink")
    frame.pack()
    root.mainloop()
```



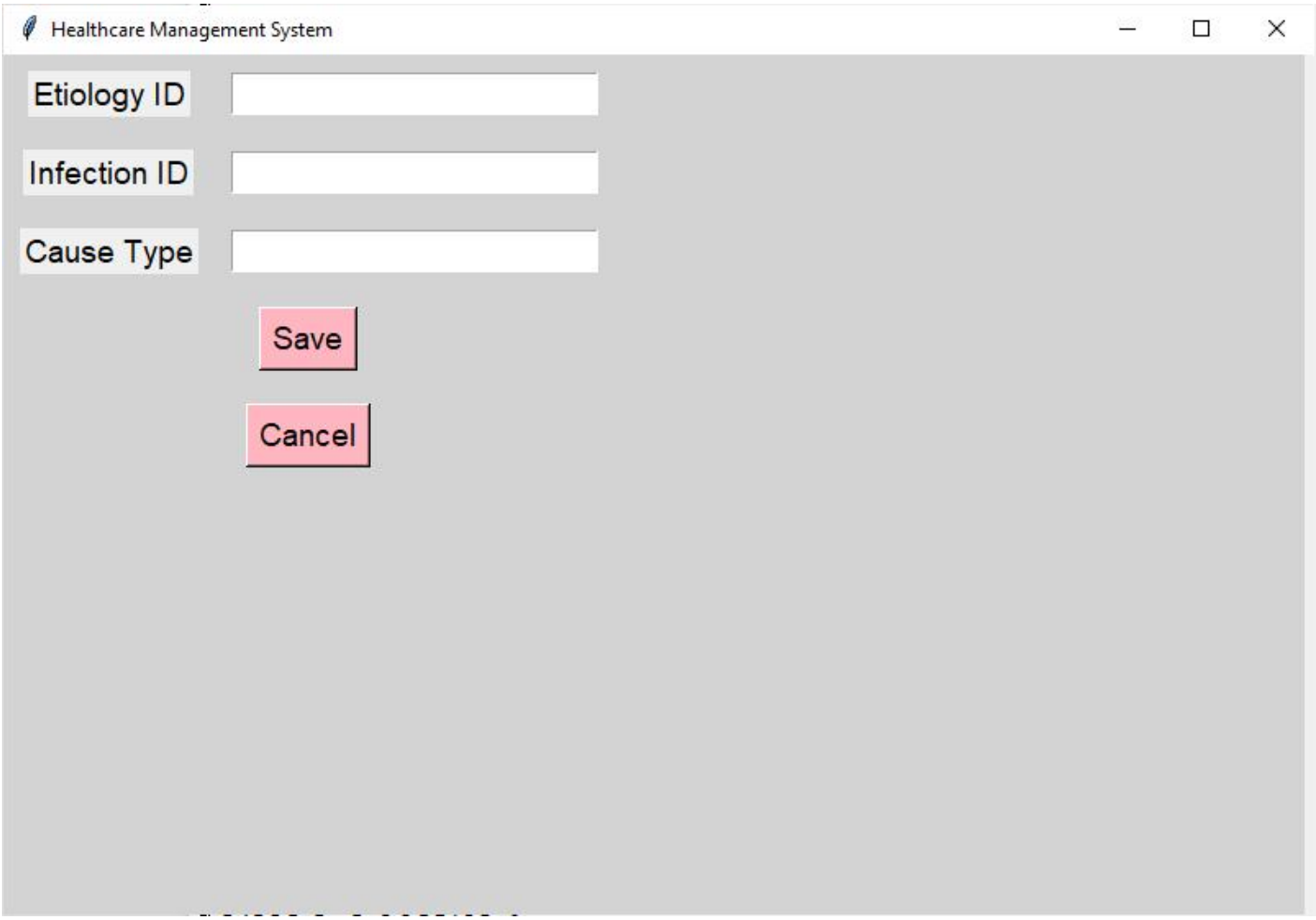
ADD EDIT ETIOLOGY FORM

```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_non_query, execute_query

class AddEditEtiologyForm(tk.Frame):
    def __init__(self, parent, controller, bg_color, button_color="light pink"):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.etiology_id = None
        self.create_widgets()

    def create_widgets(self):
        self.label_EID = tk.Label(self, text="Etiology ID")
        self.label_EID.grid(row=0, column=0, padx=10, pady=10)

        self.entry_EID = tk.Entry(self)
```

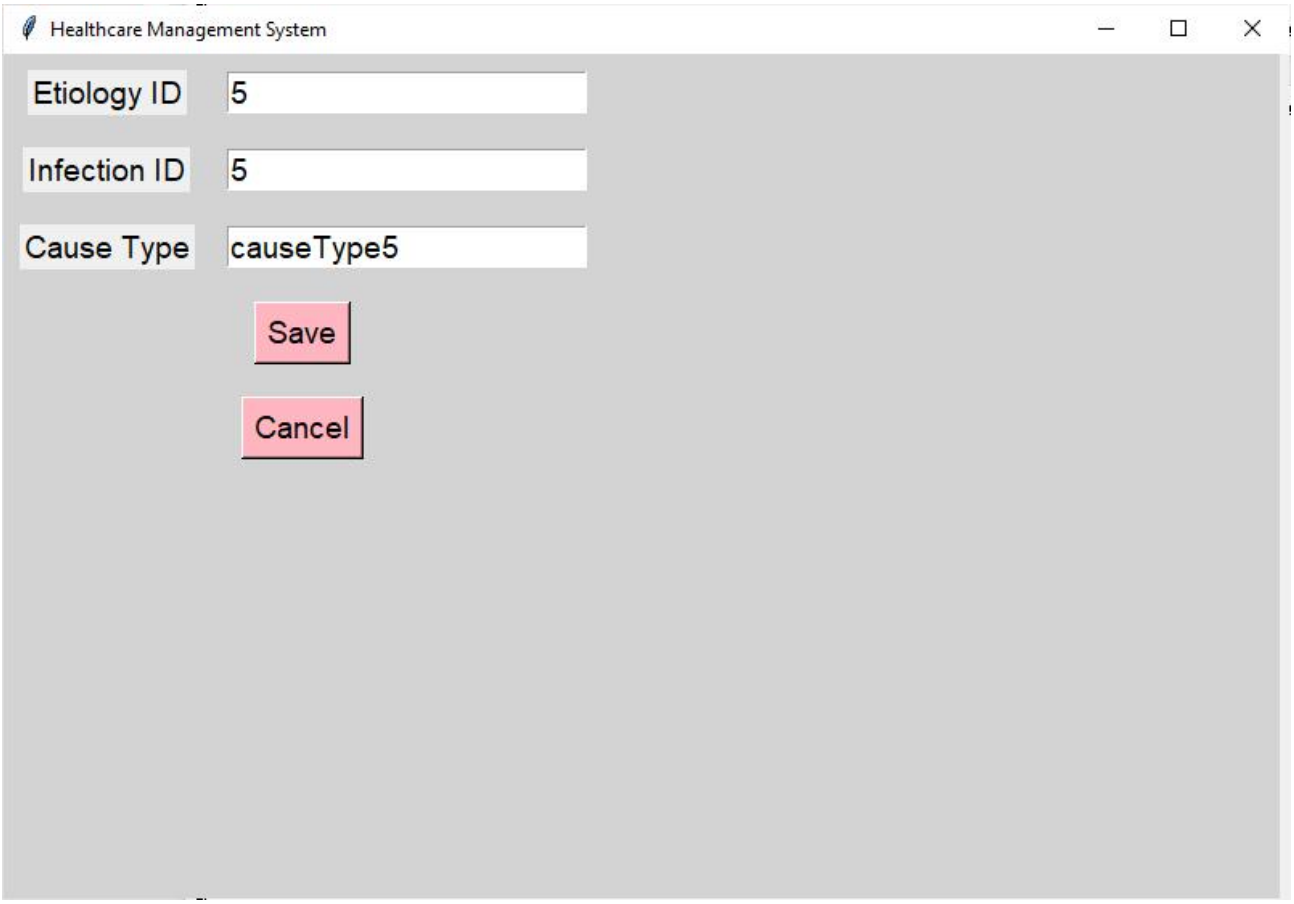
```
        self.entry_EID.grid(row=0, column=1, padx=10, pady=10)

        self.label_IID = tk.Label(self, text="Infection ID")
        self.label_IID.grid(row=1, column=0, padx=10, pady=10)

        self.entry_IID = tk.Entry(self)
        self.entry_IID.grid(row=1, column=1, padx=10, pady=10)

        self.label_cause_type = tk.Label(self, text="Cause Type")
        self.label_cause_type.grid(row=2, column=0, padx=10, pady=10)

        self.entry_cause_type = tk.Entry(self)
        self.entry_cause_type.grid(row=2, column=1, padx=10, pady=10)

        self.button_save = tk.Button(self, text="Save", command=self.save_etiology, bg=self.button_color)
        self.button_save.grid(row=3, column=0, columnspan=2, pady=10)

        self.button_cancel = tk.Button(self, text="Cancel", command=lambda:
self.controller.show_frame("EtiologyManagementForm"), bg=self.button_color)
        self.button_cancel.grid(row=4, column=0, columnspan=2, pady=10)

    def load_etiology(self, etiology_id):
        self.etiology_id = etiology_id
        query = "SELECT * FROM etiology WHERE etiologyID = ?"
        etiology = execute_query(query, (etiology_id,))
        if etiology:
            self.entry_EID.delete(0, tk.END)
            self.entry_EID.insert(0, etiology[0][0])
            self.entry_IID.delete(0, tk.END)
            self.entry_IID.insert(0, etiology[0][1])
            self.entry_cause_type.delete(0, tk.END)
            self.entry_cause_type.insert(0, etiology[0][2])

    def save_etiology(self):
        EID = self.entry_EID.get()
        IID = self.entry_IID.get()
        cause_type = self.entry_cause_type.get()
        if self.etiology_id:
            query = "UPDATE Etiology SET etiologyID = ?, infectionID = ?, cause_type = ? WHERE etiologyID = ?"
```

```
        execute_non_query(query, (EID, IID, cause_type, self.etiology_id))
    else:
        query = "INSERT INTO Etiology (etiologyID, infectionID, cause_type) VALUES (?, ?, ?)"
        execute_non_query(query, (EID, IID, cause_type))
    self.controller.show_frame("EtiologyManagementForm")
    self.controller.frames["EtiologyManagementForm"].load_etiologies()


if __name__ == "__main__":
    root = tk.Tk()
    frame = AddEditEtiologyForm(root, None, "Light Grey", "light pink")
    frame.pack()
    root.mainloop()
```

ADD FORM :



EDIT FORM :

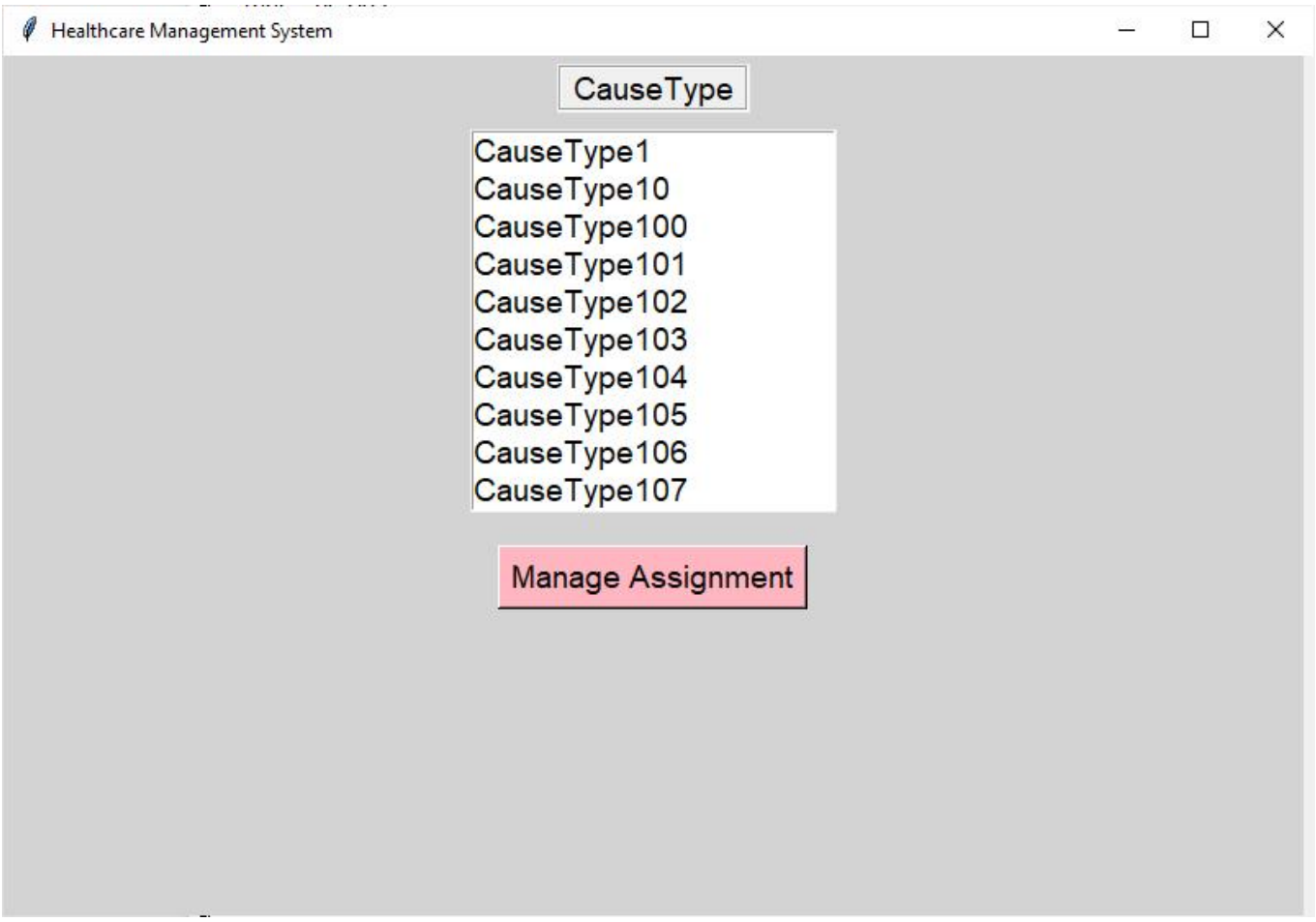


CAUSE MANAGEMENT FORM

```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_query, execute_non_query

class CauseManagementForm(tk.Frame):
    def __init__(self, parent, controller, bg_color, button_color="light pink"):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.create_widgets()

    def create_widgets(self):
        # Create a frame for the heading
        heading_frame = tk.Frame(self)
        heading_frame.grid(row=0, column=0, columnspan=1, padx=5, pady=(5, 0))

        # Add headings as labels
```

```
        headings = ["CauseType"]
        for col_num, heading in enumerate(headings):
            label = tk.Label(heading_frame, text=heading, borderwidth=2, relief="groove", width=10)
            label.grid(row=0, column=col_num, padx=1, pady=1)

        self.cause_listbox = tk.Listbox(self)
        self.cause_listbox.grid(row=1, column=0, columnspan=3, padx=10, pady=10)

        # Configure the Listbox frame grid
        self.grid_columnconfigure(0, weight=1)

        #self.button_add = tk.Button(self, text="Add Cause", command=lambda:
self.controller.show_frame("AddEditCauseForm"), bg=self.button_color)
        #self.button_add.grid(row=2, column=0, padx=10, pady=10)

        #self.button_edit = tk.Button(self, text="Edit Cause", command=self.edit_cause, bg=self.button_color)
        #self.button_edit.grid(row=2, column=1, padx=10, pady=10)

        #self.button_delete = tk.Button(self, text="Delete Cause", command=self.delete_cause, bg=self.button_color)
        #self.button_delete.grid(row=2, column=2, padx=10, pady=10)

        self.button_manage_assignment = tk.Button(self, text="Manage Assignment", command=lambda:
self.controller.show_frame("AssignedManagementForm"), bg=self.button_color)
        self.button_manage_assignment.grid(row=3, column=0, padx=10, pady=10)

        self.load_causes()

    def load_causes(self):
        self.cause_listbox.delete(0, tk.END)
        query = "SELECT causeType FROM Cause"
        causes = execute_query(query)
        for cause in causes:
            self.cause_listbox.insert(tk.END, f"{cause[0]}")

    def edit_cause(self):
        selected = self.cause_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No cause selected")
            return
```

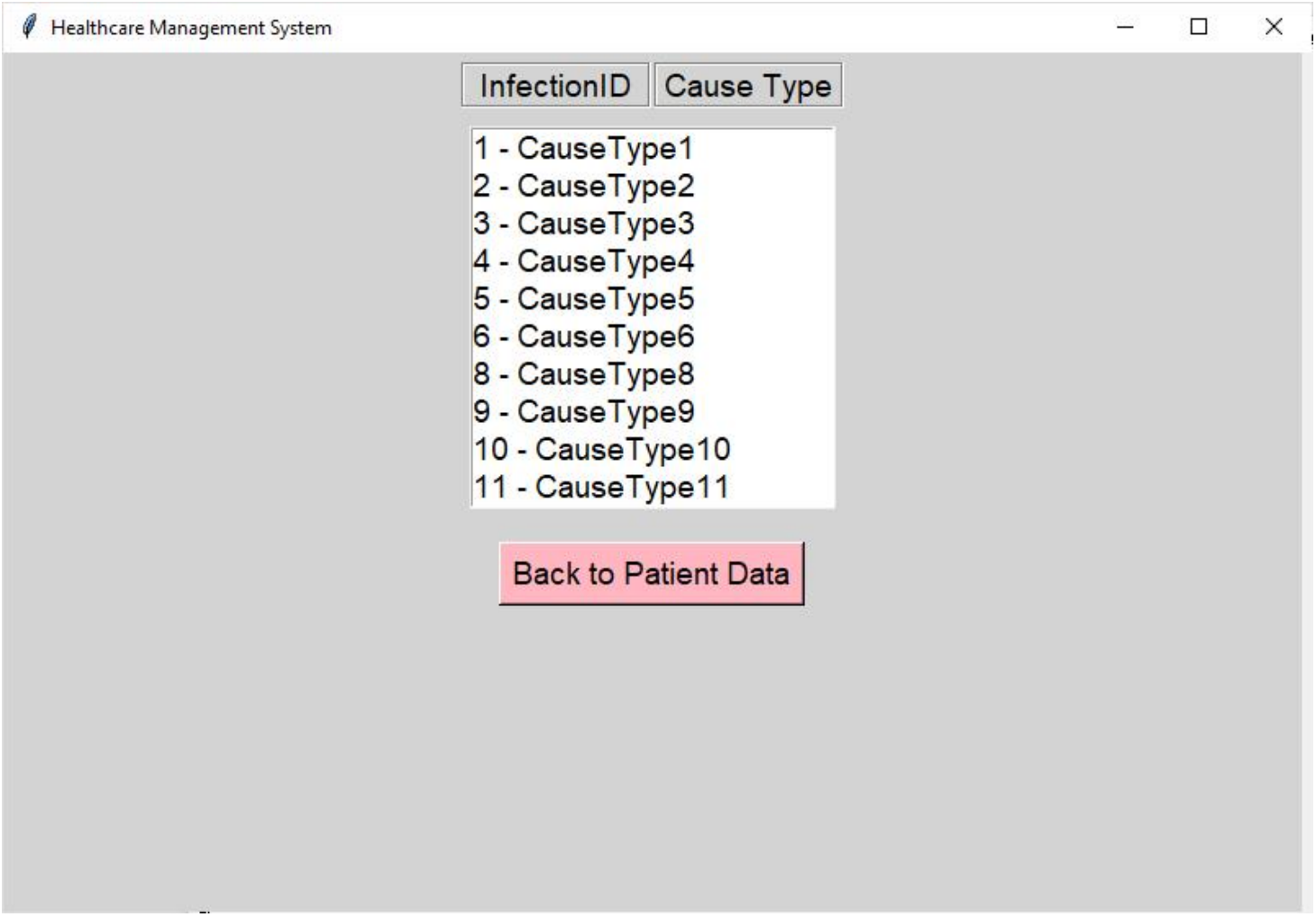
```
        cause_id = self.cause_listbox.get(selected).split(" - ")[0]
        self.controller.frames["AddEditCauseForm"].load_cause(cause_id)
        self.controller.show_frame("AddEditCauseForm")

    def delete_cause(self):
        selected = self.cause_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No cause selected")
            return
        cause_id = self.cause_listbox.get(selected).split(" - ")[0]
        query = "DELETE FROM cause WHERE causeType = ?"
        execute_non_query(query, (cause_id,))
        self.load_causes()

if __name__ == "__main__":
    root = tk.Tk()
    frame = CauseManagementForm(root, None, "Light Grey", "light pink")
    frame.pack()
    root.mainloop()
```



ASSIGNED MANAGEMENT FORM

```
import tkinter as tk
from tkinter import messagebox
from db_utils import execute_query, execute_non_query

class AssignedManagementForm(tk.Frame):
    def __init__(self, parent, controller, bg_color="pink", button_color="light pink"):
        super().__init__(parent, bg=bg_color)
        self.controller = controller
        self.button_color = button_color
        self.create_widgets()

    def create_widgets(self):
        # Create a frame for the heading
        heading_frame = tk.Frame(self, bg=self["bg"])
        heading_frame.grid(row=0, column=0, columnspan=3, padx=5, pady=(5, 0))

        # Add headings as labels
        headings = ["InfectionID", "Cause Type"]
        for col_num, heading in enumerate(headings):
            label = tk.Label(heading_frame, text=heading, borderwidth=2, relief="groove", width=10, bg=self["bg"])
            label.grid(row=0, column=col_num, padx=1, pady=1)

        self.assigned_listbox = tk.Listbox(self)
        self.assigned_listbox.grid(row=1, column=0, columnspan=3, padx=10, pady=10)

        # Configure the Listbox frame grid
        self.grid_columnconfigure(0, weight=1)
        #self.button_add = tk.Button(self, text="Add Assignment", bg=self.button_color, command=lambda:
self.controller.show_frame("AddEditAssignedForm"))
        #self.button_add.grid(row=2, column=0, padx=10, pady=10)

        #self.button_edit = tk.Button(self, text="Edit Assignment", bg=self.button_color,
command=self.edit_assignment)
        #self.button_edit.grid(row=2, column=1, padx=10, pady=10)

        #self.button_delete = tk.Button(self, text="Delete Assignment", bg=self.button_color,
command=self.delete_assignment)
```

```
        #self.button_delete.grid(row=2, column=2, padx=10, pady=10)

        self.button_back = tk.Button(self, text="Back to Patient Data", bg=self.button_color, command=lambda:
self.controller.show_frame("PatientManagementForm"))
        self.button_back.grid(row=3, column=0, columnspan=3, padx=10, pady=10)

        self.load_assignments()

    def load_assignments(self):
        self.assigned_listbox.delete(0, tk.END)
        query = "SELECT infectionID, causeType FROM Assigned"
        assignments = execute_query(query)
        for assignment in assignments:
            self.assigned_listbox.insert(tk.END, f"{assignment[0]} - {assignment[1]}")

    def edit_assignment(self):
        selected = self.assigned_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No assignment selected")
            return
        assigned_id = self.assigned_listbox.get(selected).split(" - ")[0]
        self.controller.frames["AddEditAssignedForm"].load_assignment(assigned_id)
        self.controller.show_frame("AddEditAssignedForm")

    def delete_assignment(self):
        selected = self.assigned_listbox.curselection()
        if not selected:
            messagebox.showerror("Error", "No assignment selected")
            return
        assigned_id = self.assigned_listbox.get(selected).split(" - ")[0]
        query = "DELETE FROM assigned WHERE infectionID = :1"
        execute_non_query(query, (assigned_id,))
        self.load_assignments()

if __name__ == "__main__":
    root = tk.Tk()
    frame = AssignedManagementForm(root, None, "Light Grey", "light pink")
    frame.pack()
    root.mainloop()
```


Healthcare Management System
InfectionID
Cause Type
1 - CauseType1
2 - CauseType2
3 - CauseType3
4 - CauseType4
5 - CauseType5
6 - CauseType6
8 - CauseType8
9 - CauseType9
10 - CauseType10
11 - CauseType11
Back to Patient Data