
COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS

UNIVERSITI TEKNOLOGI MARA (UiTM),

CAWANGAN KEDAH, KAMPUS SUNGAI PETANI

DIPLOMA IN INFORMATICS LIBRARY

(CDIM144)

PROGRAMMING FOR LIBRARIES

(IML208)

# INDIVIDUAL ASSIGNMENT: DENTAL CLINIC APPPOINTMENT SYSTEM

PREPARED BY:

SITI HADIRAH BINTI NASIB

(2023664186)

CLASS: KCDIM1443B

PREPARED FOR:

MOHD FIRDAUS BIN MOHD HELMI

SUBMISSION DATE:

WEEK 10

# INDIVIDUAL ASSIGNMENT: DENTAL CLINIC APPPOINTMENT SYSTEM

PREPARED BY:

SITI HADIRAH BINTI NASIB

(2023664186)

CLASS: KCDIM1443B

CDIM144 – DIPLOMA IN INFORMATICS LIBRARY

**ACKNOWLEDGEMENT**

Assalamualaikum w.b.t,

first and foremost I would like to say thanks for Allah because of His Grace the journey of finishing this assignment run smoothly.

Next, I would say thank you to miss Zuraidah binti Arif for gave the tips on finding the information, gave us the guideline on how to organize this assignment in perfect format and the advice to improve the assignment. I don't know how to start this assignment without her guidance. I also won't forget to be thankful for having a supporting family. My parents always pray for my success and won't ever give up to ask about my health. They always take care of me even though I am far away from home. My love from them never be less day by day.

Other than that, I would say thanks for my classmates for kindly helping me qlarify any doubts or misunderstood throught the chat group or in the conversation. Last but not least, I would say thank you for myself because in short period of time I can finished this assignment. The improvement or mistakes that I made hopefully would be the lessons for the next time.

**Project Name:** Dental Clinic Appointment

**File name:** dentalappointment.py

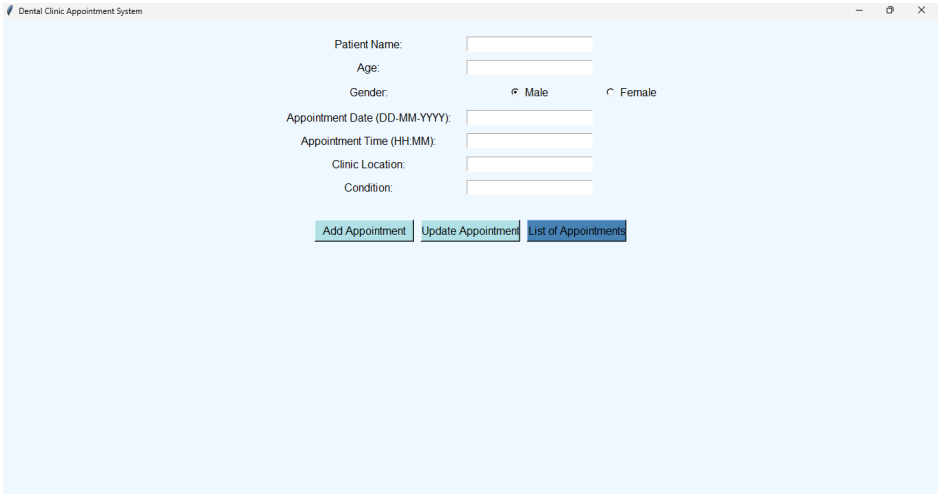
**Prompt Data:**

i. Name
ii. Age
iii. Gender (Male/Female)
iv. Phone number
v. Appointment date
vi. Appointment time
vii. Clinic location
viii. Condition

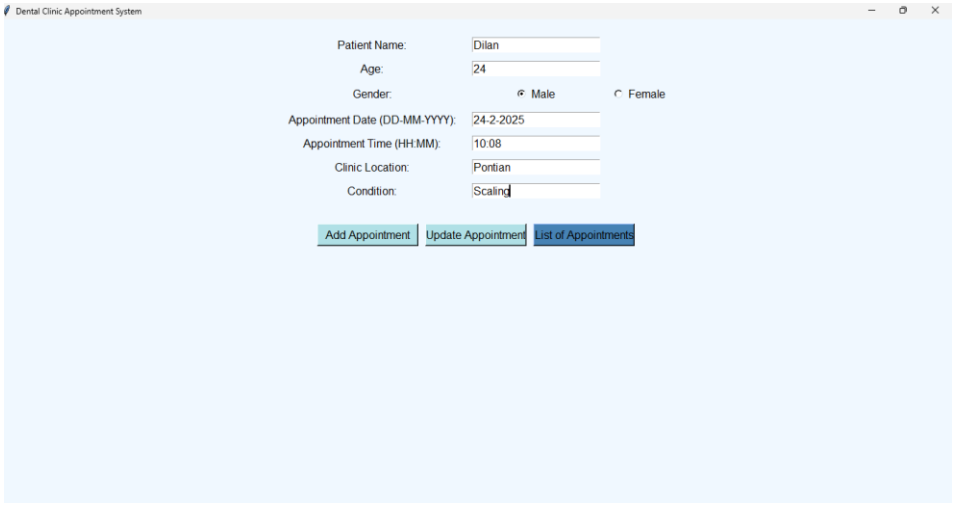
**Function :**

### i. Create data

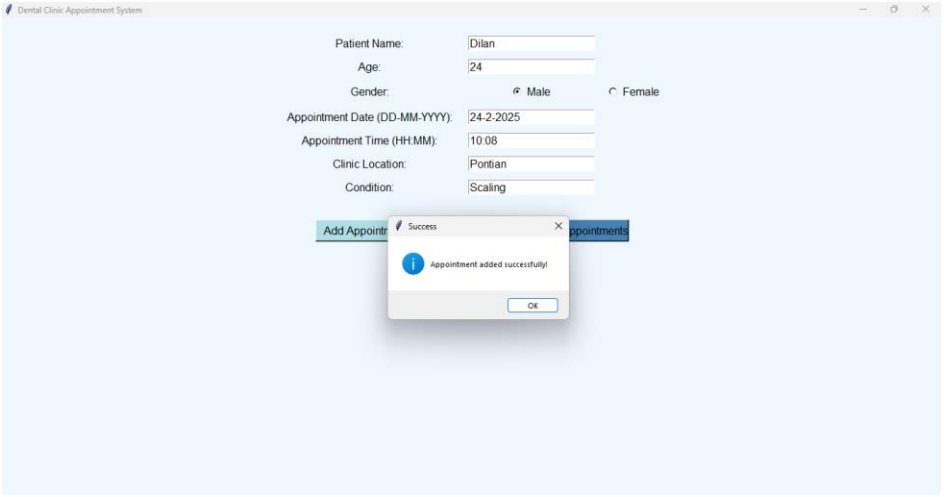
The first step is let's start with open the GUI.




Then, insert your data on the system and click "Add Appointment" button.

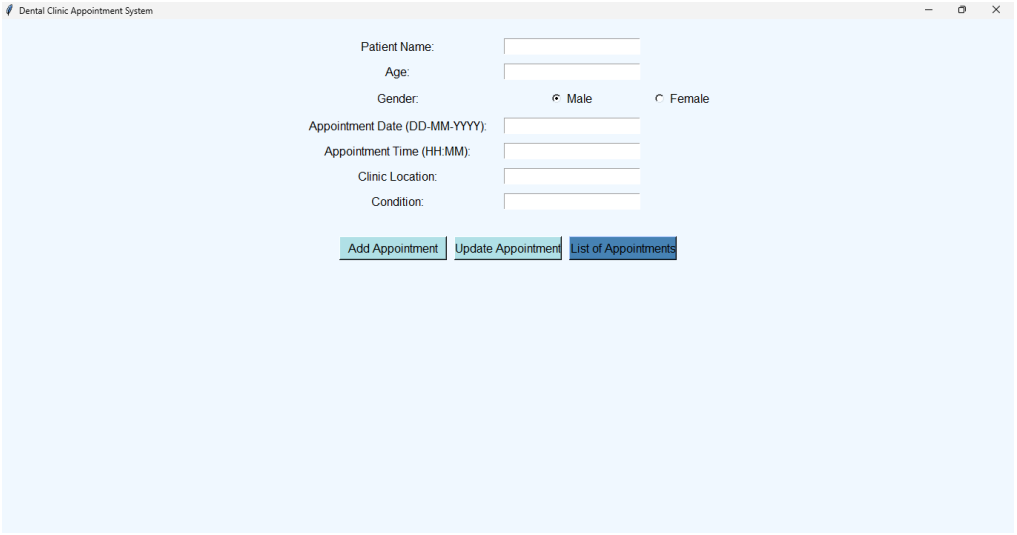



The result will be like this. Click "Ok" and the system will automatically restart into normal form.

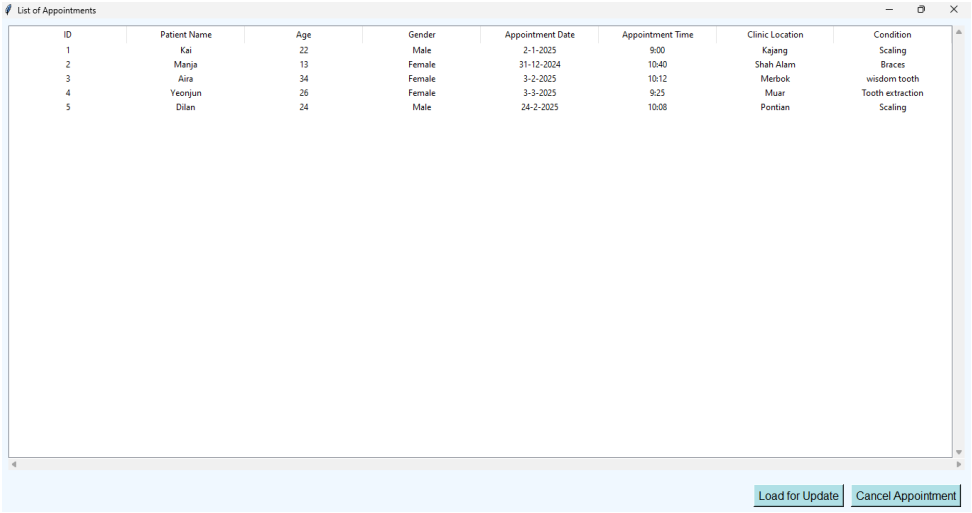



### ii. Read data

Go and click “List of Appointments” button.

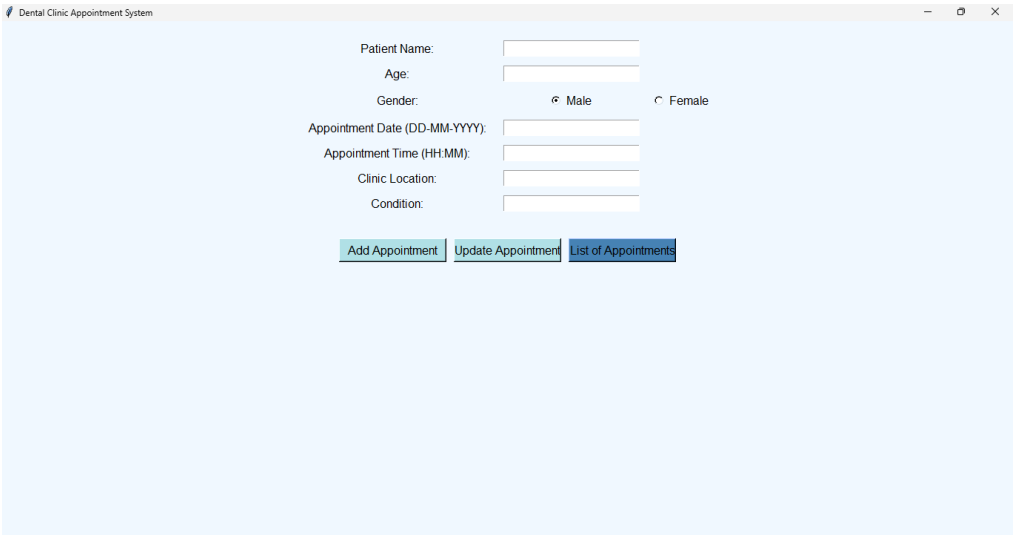



This is how user can read their appointment schedule.

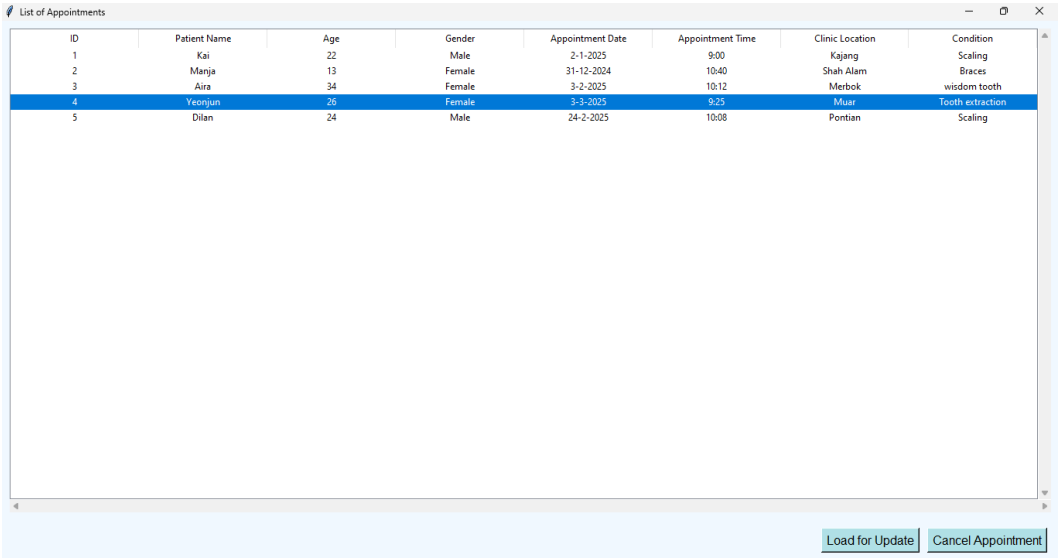



### iii. Update data

First, click on “Update Appointment” button.

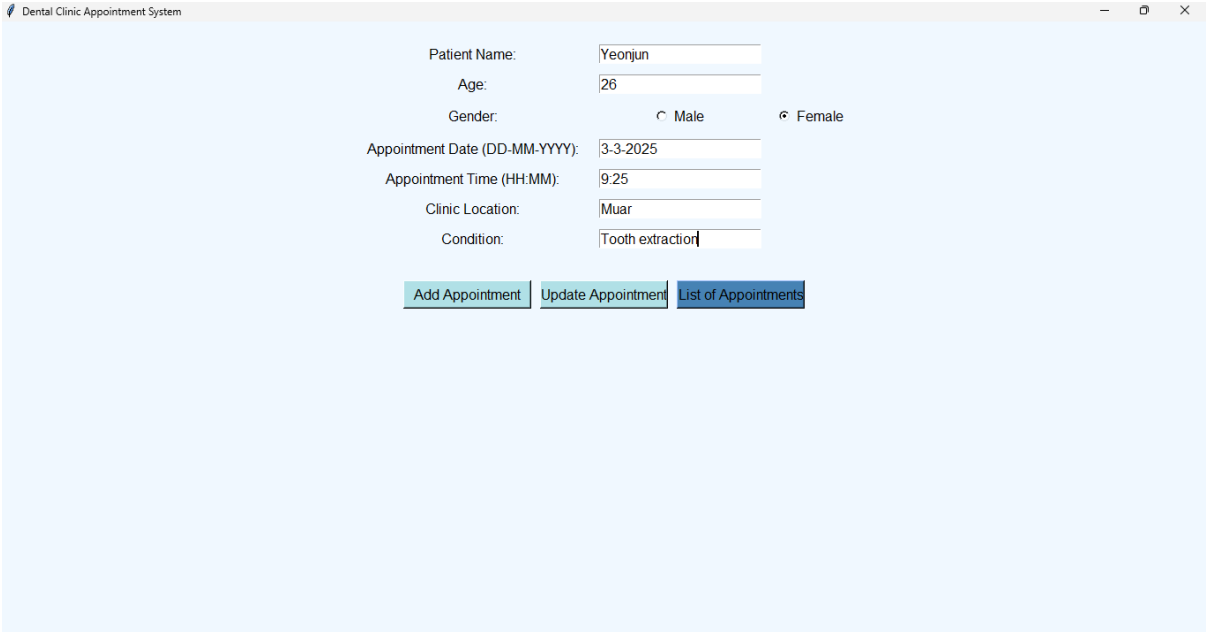


Second, choose the data you want to change and click “Load for Update” button.

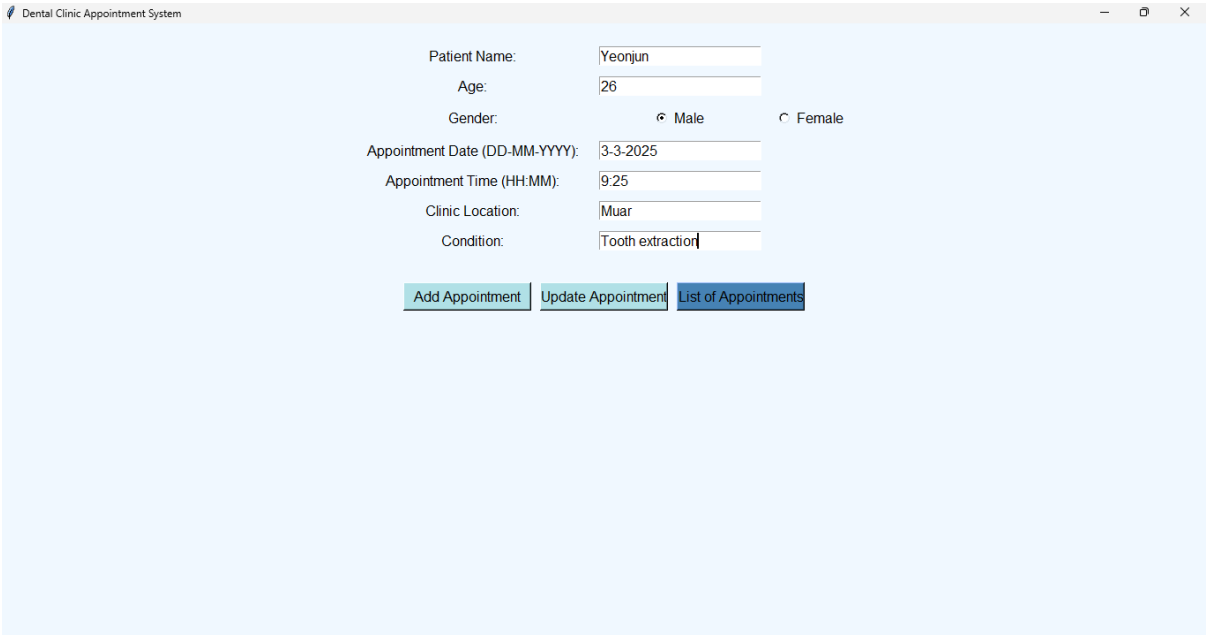



It will appear like this. Now, we need to change the gender from Female to Male.

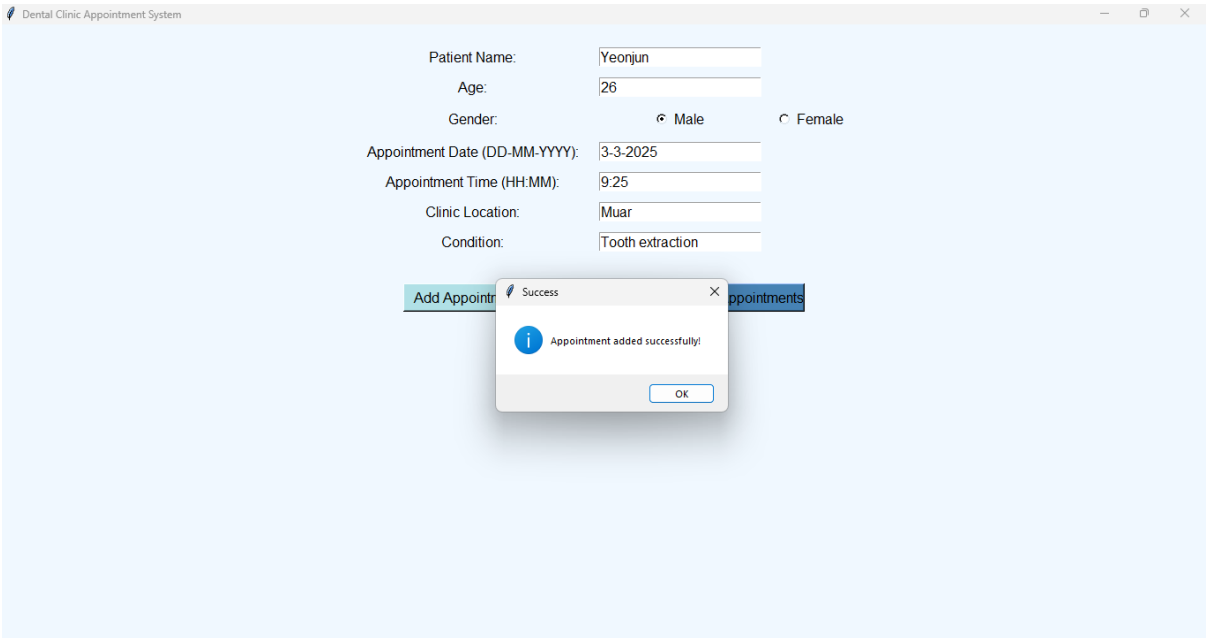
**Before:**



**After:**

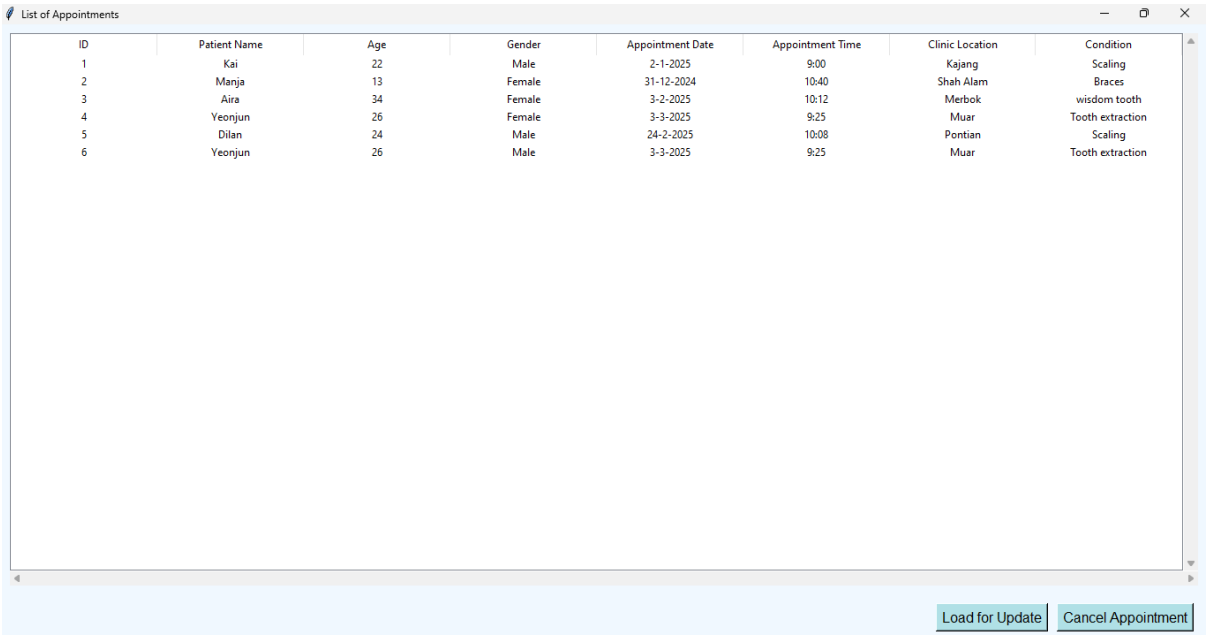



After change the data, click “Add Appointment” and it will appear like this:

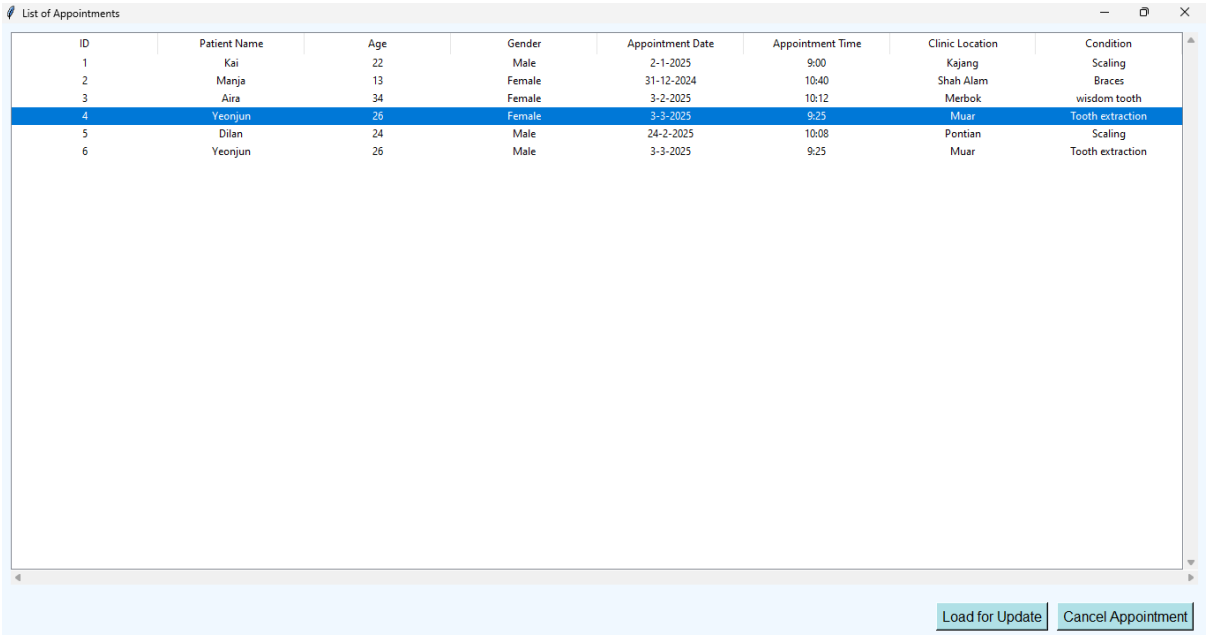


### iv. Delete existing data

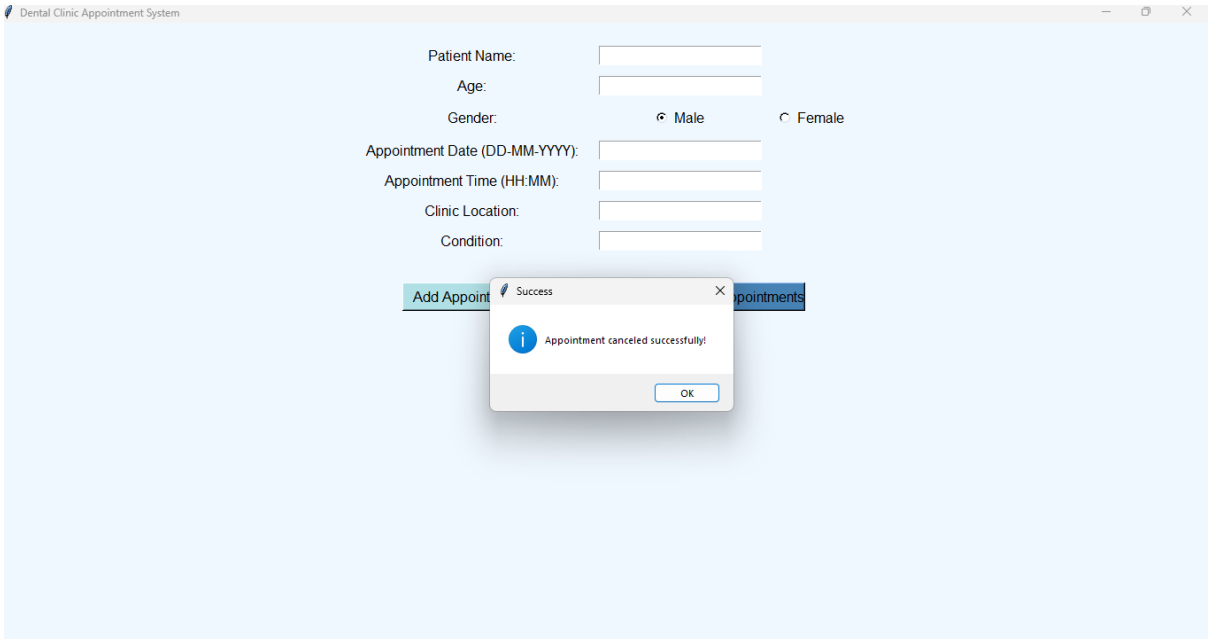
After updating the list, it turned like this:




All you have to do is choose the data you need to remove and click on the “Cancel Appointment”.



The result:


Dental Clinic Appointment System
Patient Name:
Age:
Gender:
Male
Female
Appointment Date (DD-MM-YYYY):
Appointment Time (HH:MM):
Clinic Location:
Condition:
Add Appoint
pointments
Success
Appointment canceled successfully!
OK

**Conditional Statement : Yes**

1) if

```
    if name and age and gender and date and time and location and condition:
        appointment = {
            "id": len(appointments) + 1,
            "patient_name": name,
            "age": age,
            "gender": gender,
            "appointment_date": date,
            "appointment_time": time,
            "clinic_location": location,
            "condition": condition
        }
```

```
        if not selected_item:
            messagebox.showerror('Error', 'Please select an appointment to update.')
            return
```

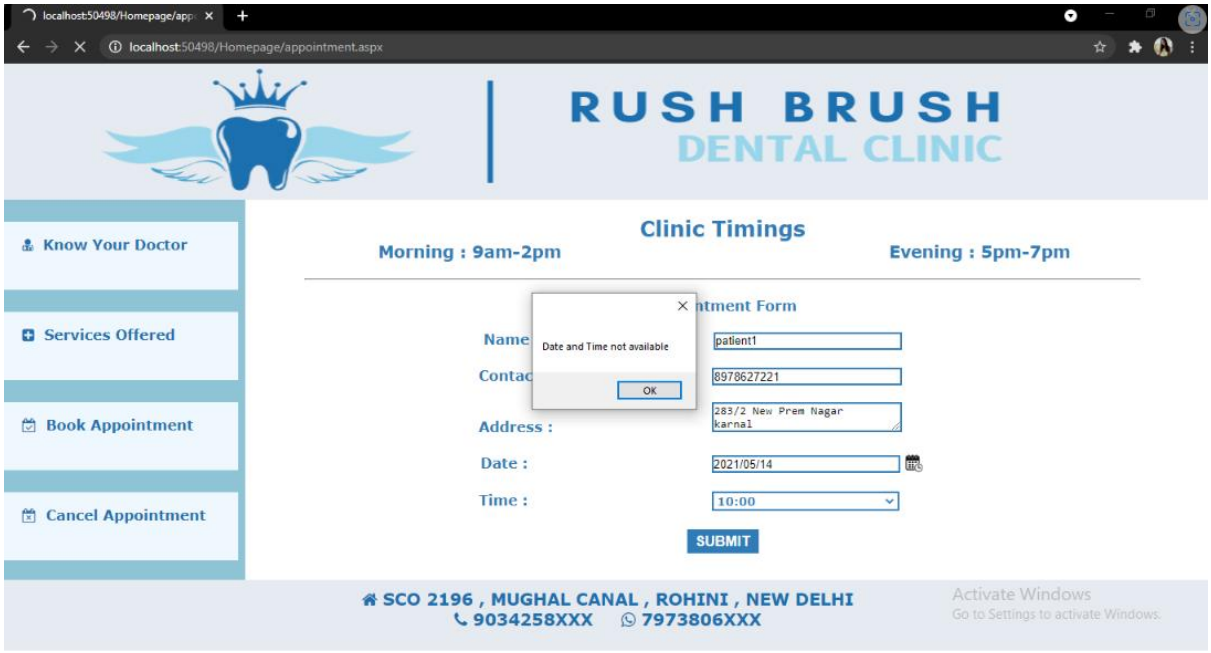
```
            if appointment["id"] == selected_id:
                entry_name.delete(0, tk.END)
                entry_name.insert(0, appointment["patient_name"])
                entry_age.delete(0, tk.END)
                entry_age.insert(0, appointment["age"])
                gender_var.set(appointment["gender"])
                entry_date.delete(0, tk.END)
                entry_date.insert(0, appointment["appointment_date"])
                entry_time.delete(0, tk.END)
                entry_time.insert(0, appointment["appointment_time"])
                entry_location.delete(0, tk.END)
                entry_location.insert(0, appointment["clinic_location"])
                entry_condition.delete(0, tk.END)
                entry_condition.insert(0, appointment["condition"])
                appointments_window.destroy()
                return
```

```
        if not selected_item:
            messagebox.showerror('Error', 'Please select an appointment to delete.')
            return
```

2) else

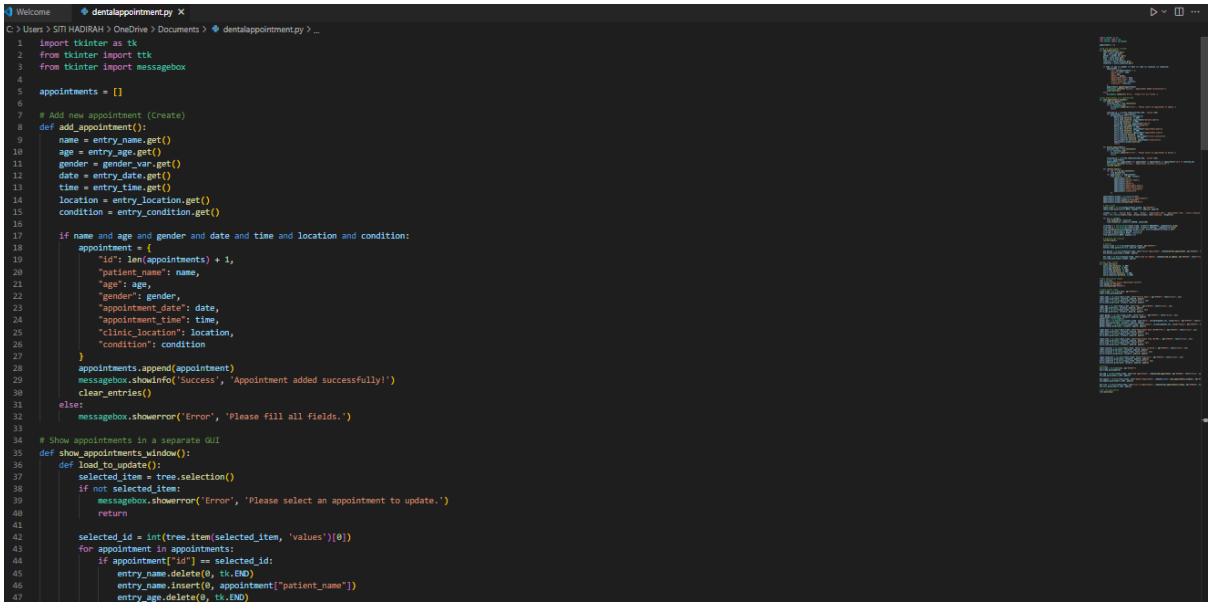
```
    else:
        messagebox.showerror('Error', 'Please fill all fields.')
```

**GUI : Yes (add screenshot)**




**Result:**

Python code



```python
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

appointments = []

# Add new appointment (Create)
def add_appointment():
    name = entry_name.get()
    age = entry_age.get()
    gender = gender_var.get()
    date = entry_date.get()
    time = entry_time.get()
    location = entry_location.get()
    condition = entry_condition.get()

    if name and age and gender and date and time and location and condition:
        appointment = {
            "id": len(appointments) + 1,
            "patient_name": name,
            "age": age,
            "gender": gender,
            "appointment_date": date,
            "appointment_time": time,
            "clinic_location": location,
            "condition": condition
        }
        appointments.append(appointment)
        messagebox.showinfo('Success', 'Appointment added successfully!')
        clear_entries()
    else:
        messagebox.showerror('Error', 'Please fill all fields.')

# Show appointments in a separate GUI
def show_appointments_window():
    def load_to_update():
        selected_item = tree.selection()
        if not selected_item:
            messagebox.showerror('Error', 'Please select an appointment to update.')
            return

        selected_id = int(tree.item(selected_item, 'values')[0])
        for appointment in appointments:
            if appointment["id"] == selected_id:
                entry_name.delete(0, tk.END)
                entry_name.insert(0, appointment["patient_name"])
                entry_age.delete(0, tk.END)
```

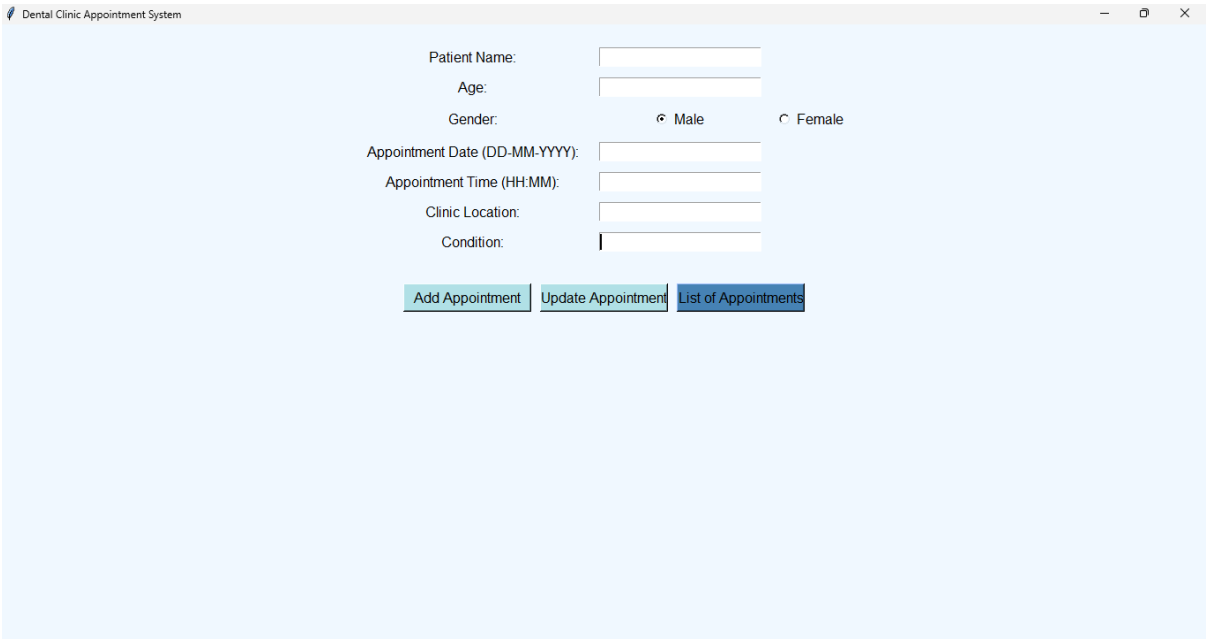
```
                entry_age.delete(0, tk.END)
                entry_age.insert(0, appointment["age"])
                gender_var.set(appointment["gender"])
                entry_date.delete(0, tk.END)
                entry_date.insert(0, appointment["appointment_date"])
                entry_time.delete(0, tk.END)
                entry_time.insert(0, appointment["appointment_time"])
                entry_location.delete(0, tk.END)
                entry_location.insert(0, appointment["clinic_location"])
                entry_condition.delete(0, tk.END)
                entry_condition.insert(0, appointment["condition"])
                appointments_window.destroy()
                return

    def delete_appointment():
        selected_item = tree.selection()
        if not selected_item:
            messagebox.showerror('Error', 'Please select an appointment to delete.')
            return

        selected_id = int(tree.item(selected_item, 'values')[0])
        global appointments
        appointments = [appointment for appointment in appointments if appointment["id"] != selected_id]
        messagebox.showinfo('Success', 'Appointment canceled successfully!')
        refresh_table()

    def refresh_table():
        for row in tree.get_children():
            tree.delete(row)
        for appointment in appointments:
            tree.insert('', tk.END, values=(
                appointment["id"],
                appointment["patient_name"],
                appointment["age"],
                appointment["gender"],
                appointment["appointment_date"],
                appointment["appointment_time"],
                appointment["clinic_location"],
                appointment["condition"]
            ))

    appointments_window = tk.Toplevel(root)
    appointments_window.title("List of Appointments")
    appointments_window.geometry("800x500")
    appointments_window.configure(bg="#f0f8ff")
```

```
    # Table Frame
    table_frame = tk.Frame(appointments_window, bg="#f0f8ff")
    table_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

    columns = ('ID', 'Patient Name', 'Age', 'Gender', 'Appointment Date', 'Appointment Time', 'Clinic Location', 'Condition')
    tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=15)

    for col in columns:
        tree.heading(col, text=col)
        tree.column(col, anchor=tk.CENTER, width=100)

    scrollbar_x = ttk.Scrollbar(table_frame, orient=tk.HORIZONTAL, command=tree.xview)
    scrollbar_y = ttk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
    tree.configure(xscrollcommand=scrollbar_x.set, yscrollcommand=scrollbar_y.set)
    scrollbar_x.pack(side=tk.BOTTOM, fill=tk.X)
    scrollbar_y.pack(side=tk.RIGHT, fill=tk.Y)
    tree.pack(fill=tk.BOTH, expand=True)

    # Populate the Treeview
    refresh_table()

    # Buttons
    button_frame = tk.Frame(appointments_window, bg="#f0f8ff")
    button_frame.pack(fill=tk.X, padx=10, pady=10)

    btn_delete = tk.Button(button_frame, text="Cancel Appointment", command=delete_appointment, bg="#b0e0e6", font=("Arial", 12))
    btn_delete.pack(side=tk.RIGHT, padx=5)

    btn_load = tk.Button(button_frame, text="Load for Update", command=load_to_update, bg="#b0e0e6", font=("Arial", 12))
    btn_load.pack(side=tk.RIGHT, padx=5)

# Clear input fields
def clear_entries():
    entry_name.delete(0, tk.END)
    entry_age.delete(0, tk.END)
    entry_date.delete(0, tk.END)
    entry_time.delete(0, tk.END)
    entry_location.delete(0, tk.END)
    entry_condition.delete(0, tk.END)

# Main application window
root = tk.Tk()
root.title("Dental Clinic Appointment System")
root.geometry("600x600")
root.configure(bg="#f0f8ff")
```

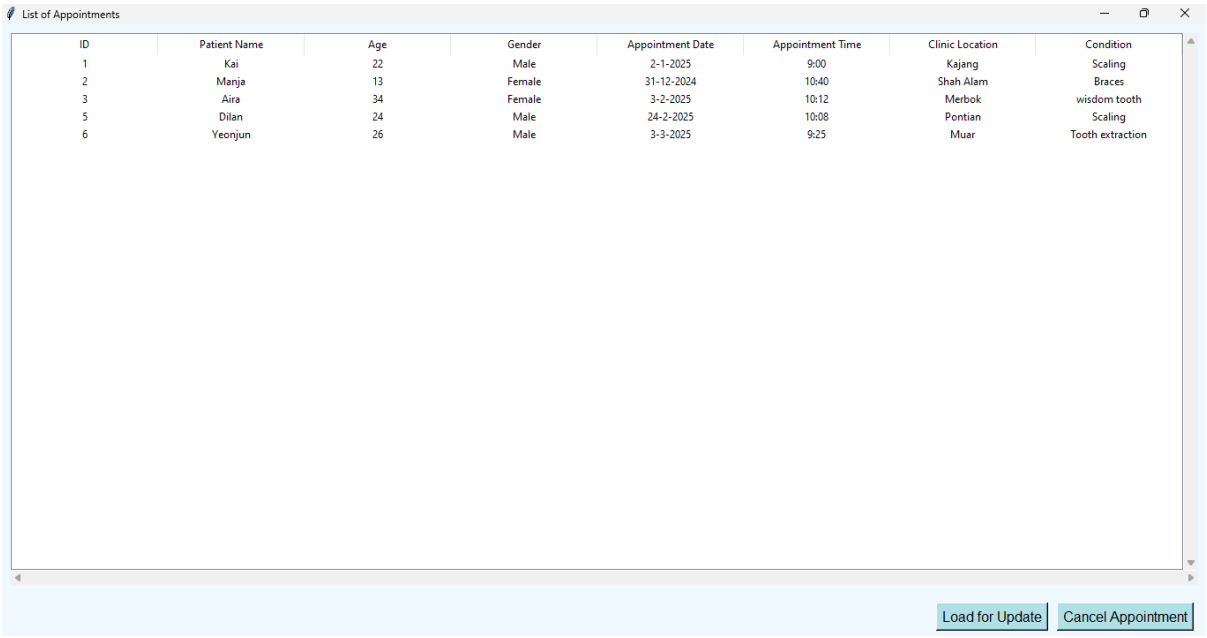
```
# Input fields frame
input_frame = tk.Frame(root, bg="#f0f8ff")
input_frame.pack(pady=20)

label_name = tk.Label(input_frame, text="Patient Name:", bg="#f0f8ff", font=("Arial", 12))
label_name.grid(row=0, column=0, padx=10, pady=5)
entry_name = tk.Entry(input_frame, font=("Arial", 12))
entry_name.grid(row=0, column=1, padx=10, pady=5)

label_age = tk.Label(input_frame, text="Age:", bg="#f0f8ff", font=("Arial", 12))
label_age.grid(row=1, column=0, padx=10, pady=5)
entry_age = tk.Entry(input_frame, font=("Arial", 12))
entry_age.grid(row=1, column=1, padx=10, pady=5)

label_gender = tk.Label(input_frame, text="Gender:", bg="#f0f8ff", font=("Arial", 12))
label_gender.grid(row=2, column=0, padx=10, pady=5)
gender_var = tk.StringVar()
gender_male = tk.Radiobutton(input_frame, text="Male", variable=gender_var, value="Male", bg="#f0f8ff", font=("Arial", 12))
gender_male.grid(row=2, column=1, padx=5, pady=5)
gender_female = tk.Radiobutton(input_frame, text="Female", variable=gender_var, value="Female", bg="#f0f8ff", font=("Arial", 12))
gender_female.grid(row=2, column=2, padx=5, pady=5)

label_date = tk.Label(input_frame, text="Appointment Date (DD-MM-YYYY):", bg="#f0f8ff", font=("Arial", 12))
label_date.grid(row=3, column=0, padx=10, pady=5)
entry_date = tk.Entry(input_frame, font=("Arial", 12))
entry_date.grid(row=3, column=1, padx=10, pady=5)

label_time = tk.Label(input_frame, text="Appointment Time (HH:MM):", bg="#f0f8ff", font=("Arial", 12))
label_time.grid(row=4, column=0, padx=10, pady=5)
entry_time = tk.Entry(input_frame, font=("Arial", 12))
entry_time.grid(row=4, column=1, padx=10, pady=5)

label_location = tk.Label(input_frame, text="Clinic Location:", bg="#f0f8ff", font=("Arial", 12))
label_location.grid(row=5, column=0, padx=10, pady=5)
entry_location = tk.Entry(input_frame, font=("Arial", 12))
entry_location.grid(row=5, column=1, padx=10, pady=5)

label_condition = tk.Label(input_frame, text="Condition:", bg="#f0f8ff", font=("Arial", 12))
label_condition.grid(row=6, column=0, padx=10, pady=5)
entry_condition = tk.Entry(input_frame, font=("Arial", 12))
entry_condition.grid(row=6, column=1, padx=10, pady=5)
```

```python
# Buttons
btn_frame = tk.Frame(root, bg="#f0f8ff")
btn_frame.pack(pady=10)

btn_add = tk.Button(btn_frame, text="Add Appointment", command=add_appointment, bg="#b0e0e6", font=("Arial", 12), width=15)
btn_add.pack(side=tk.LEFT, padx=5)

btn_update = tk.Button(btn_frame, text="Update Appointment", command=lambda: show_appointments_window(), bg="#b0e0e6", font=("Arial", 12), width=15)
btn_update.pack(side=tk.LEFT, padx=5)

btn_list = tk.Button(btn_frame, text="List of Appointments", command=show_appointments_window, bg="#4682b4", font=("Arial", 12), width=15)
btn_list.pack(side=tk.LEFT, padx=5)

# Run the application
root.mainloop()
```

## GUI for Dental Clinic Appointment System




## Final result:



**Strength:**

1. Patients would be alert to the schedule

2. It is user-friendly for anyone who needs to make an appointment

3. Save patients' times from making an appointment directly to the clinic

4. Patient can choose to change or update their data

**Kaizen (Room for improvement)**

1. This GUI is too basic and did not have any character or emoji that make it interesting.

2. After updating the data into new, it should be add instead of automatically replace the old data.

3. Need insert detail information such as patient's address to sent letter or any important reason. It also need to add patient's phone number and email-address to be extra careful for their health.