UNIVERSITI TEKNOLOGI MARA (UiTM)

CAWANGAN KEDAH | KAMPUS SUNGAI PETANI

COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS

DIPLOMA OF LIBRARY INFORMATICS

(CDIM144 3B)

PROGRAMMING FOR LIBRARIES

(IML208)

GROUP PROJECT: MEDICAL CLINIC APPOINTMENT

PREPARED BY:

NO	NAME	STUDENT ID
1.	FATIN NUR AFIQAH BINTI MUHAMAD FADHIL	2023687904
2.	NUR ATIQAH AMIRA BINTI MOHD ZAMRI	2023463146
3.	NUR IMAN SHAKIRAH BINTI NUR AFFENDI	2023825284
4.	NUR ADILAH BINTI MOHD AZMI	2023875138

PREPARED FOR:

MOHD FIRDAUS BIN MOHD HELMI

SUBMISSION WEEK: 15 JANUARY 2025

GROUP PROJECT: MEDICAL CLINIC APPOINTMENT

FATIN NUR AFIQAH BINTI MUHAMMAD FADHIL

2023687904

NUR ATIQAH AMIRA BINTI MOHD ZAMRI

2023463146

NUR IMAN SHAKIRAH BINTI NUR AFFENDI

2023825284

NUR ADILAH BINTI MOHD AZMI

2023875138

UNIVERSITI TEKNOLOGI MARA (UiTM) KEDAH, KAMPUS SUNGAI PETANI

SCHOOL OF INFORMATION SCIENCE

COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS

DIPLOMA IN LIBRARY INFORMATICS

15 JANUARY 2024

ACKNOWLEDGEMENT

First of all, we are grateful to be able to finish this group project successfully with the efforts we put in and the help of the people surrounding us. This group project would not be able to be completed without His permission too.

Primarily, the biggest shout out to our teammates, Nur Atiqah Amira, Fatin Nur Afiqah, Nur Iman Shakirah and Nur Adilah who contributed to the success in this group project. We learned something new and upgraded our skills in competencies throughout this journey. It will be a bonus and help us in pursuing our careers in the future.

We would like to express our gratitude to our lecturer, Sir Mohd Firdaus Bin Mohd Helmi for his unconditional support during this group project period. His clear explanations and the lessons that he taught us in these 14 weeks helped us in many ways.

Not to forget, special thanks to our classmates who directly or indirectly helped us in some ways in completing this project. Their contribution to this project brings a great outcome. Every aid is truly appreciated.

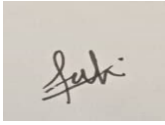
UNIVERSITI TEKNOLOGI MARA

STUDENT PLEDGE OF ACADEMIC INTEGRITY

As a student of Universiti Teknologi MARA (UiTM), it is my responsibility to act in accordance with UiTM's academic assessment and evaluation policy. I hereby pledge to act and uphold academic integrity and pursue scholarly activities in UiTM with honesty and responsible manner. I will not engage or tolerate acts of academic dishonesty, academic misconduct, or academic fraud including but not limited to:

a. **Cheating:** Using or attempt to use any unauthorized device, assistance, sources, practice or materials while completing academic assessments. This include but not limited to copying from another, allowing others to copy, unauthorized collaboration on an assignment or open book tests, or engaging in any act or conduct that can be construed as cheating.
b. **Plagiarism:** Using or attempts to use the work of others (ideas, design, words, art, music, etc.) without acknowledging the source; using or purchasing materials prepared by another person or agency or engaging in other behavior that a reasonable person would consider as plagiarism.
c. **Fabrication:** Falsifying data, information, or citations in any academic assessment and evaluation.
d. **Deception:** Providing false information with intend to deceive an instructor concerning any academic assessment and evaluation.
e. **Furnishing false information:** Providing false information or false representation to any UiTM official, instructor, or office.

With this pledge, I am fully aware that I am obliged to conduct myself with utmost honesty and integrity. I fully understand that a disciplinary action can be taken against me if I, in any manner, violate this pledge.



Name: FATIN NUR AFIQAH BINTI MUHAMMAD FADHIL

Matric Number: 2023687904

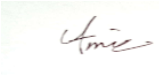
Course Code: IML 208

Programme code: CDIM144

Faculty / Campus: UiTM Campus Sungai Petani

*Students are required to sign one pledge for each course taken.

Bahagian Pentaksiran dan Penilaian Akademik 2021



Name: NUR ATIQAH AMIRA BINTI MOHD ZAMRI

Matric Number: 2023462146

Course Code: IML 208

Programme code: CDIM144

Faculty / Campus: UiTM Campus Sungai Petani

Name: NUR IMAN SHAKIRAH BINTI NUR AFFENDI

Matric Number: 2023825284

Course Code: IML 208

Programme code: CDIM144

Faculty / Campus: UiTM Campus Sungai Petani

Name: NUR ADILAH BINTI MOHD AZMI

Matric Number: 2023875138

Course Code: IML 208

Programme code: CDIM144

Faculty / Campus: UiTM Campus Sungai Petani

TABLE OF CONTENT

LIST OF FIGURES

PROJECT NAME: MEDICAL CLINIC APPOINTMENT

1.0 PROBLEM STATEMENT:

An ineffective method for managing and treating patients, failure to achieve patient satisfaction, and the clinic's inefficiency in providing patient care are some of the causes or reasons why issues with the appointment scheduling system arise.

The first issue concerns the increasing number of diseases and patients. These reasons are why medical appointments exist: to give fair services to both parties, patients and doctors. Besides, with so many diseases nowadays, medical appointments help the clinic manage services that are available in the clinic itself. For instance, there are pharmacy counters that manage medicines, an X-ray room, health check-ups in the doctor's room, and maternal treatment. By using appointments, patients can follow up without being missed to get treatment and always take note of their current health condition.

The absence of an appropriate system for patient registration and login is one of the primary issues. Such as leading to a lack of individualized care, repetitive data entry, and dispersed patient information. The clinic's administration could be improved if it takes the initiative to address this problem.

Also, manually managing a doctor's availability for clinic appointments presents challenges. Delays in receiving medical care may be owing to the patient's inability to ascertain when the doctor of their choice is available. Clinic employees also spend too much time organizing and monitoring doctors' schedules, which reduces their productivity. The lack of a method to manage and communicate physician availability impacts patient and clinic operator satisfaction.

Additionally, patients may find it difficult to postpone or cancel appointments, which could result in other patients missing out on possibilities. For instance, waiting for clinic employees to manually review schedules, update risks, or make revisions could lead to patient appointments or scheduling conflicts.

2.0 BACKGROUND OF THE SYSTEM:

We chose the topic of the medical clinic appointment booking system for this group project because we recognize the need in the health sector today and how it faces all the challenges of meeting current needs in this increasingly modern age. In particular, in the healthcare sector where patients seek treatment, patients must make an appointment in advance, either by phone or walk-in, to ensure that the process does not take too long. In addition, patients occasionally have to wait a long time to receive treatment because patient capacity is increasing, which results in a long waiting time. To better manage the system, the clinic must generally make sure that it can limit rescheduling and reduce workload.

Other than that, medical appointments help patients follow up on their current health conditions. Through medical appointments, patients can get guidance from doctors who are in charge of their health issues. As a result, patients will surely detect early signs of any diseases. They will be ready for further treatments and follow-up on healthy habits in daily life. From a doctor's perspective, a doctor can tell just by the patient’s medical history. A patient's medical appointments list helps doctors give proper treatments to the patient. This will not result in missed treatments or uncontrollable healthcare. Patients also cannot fake their health conditions without medical appointments.

Despite that, there are several functions of medical clinic appointments detected:

1. **Patient registration and login**
2. **Reducing missed visits**
3. **Manage both schedules wisely**
4. **Doctor availability management**
5. **Cancellation and rescheduling**

1. **Patient registration and login**

The first feature enables users to register for an account or log in safely for existing users without issues. Additionally, it can save patient information including name, DOB, age, gender, contact data, and medical history. This is where users begin their journey to the clinic's appointment booking system, which serves to guarantee patients' safe, individualized access and even assist them in better managing their medical care.

2. **Notify patient that will reducing missed visits**

As a patient, they do not need to worry if they happen to forget their appointment for some treatments at the clinics. In the appointment system, it will feature notifications where patients can check if they have any appointments for that day. They are also punctual with the time being scheduled. In that case, their visits will easily count and records of visits will be consistent.

3. **Track patients' medical records history**

The crucial part is the patients' past and current health conditions. Patients might get an exchange of doctors or their in-charge doctor is not available to treat them for that day. To make sure of the disease or illness they are facing, the doctor can look up the patient's appointment history. This way, quick action will be taken and the doctor can directly ask the patient for its validity.

4. **Doctor availability management**

This shows the doctor's real-time availability and schedule for that day so that patients with appointments don't have to wait around for a long time or waste time. Additionally, let the clinic staff keep track of and manage the doctors' availability so there are no conflicts with other doctors on duty.

5. **Cancellation and rescheduling**

Additionally, it enables patients to reschedule or cancel their appointments with the clinic conveniently, allows the staff to update the management system, and frees up more slots for other patients seeking treatment.

BENEFITS OF THE SYSTEM

1. Benefits to the environment

This implies that lowering paper usage can help maintain a sustainable environment. As is well known, clinics frequently use paper for appointment scheduling, patient record keeping, and appointment cancellation. If we consider the negative, it can harm the environment by, for example, causing deforestation to create paper. All patient records, timetables, and correspondence can be directly kept and handled electronically with the use of solutions like digital booking, which also helps to lessen the need for paper records.

2. Connectivity with Health Apps and Wearable technology

By integrating contemporary technology with healthcare services, the Medical Clinic Appointment System's sophisticated features can further enhance the quality of care. This integration makes personalized care, more proactive health management, and real-time health data tracking possible. As an illustration supplying tools that can assess a patient's health, including smart watches that can measure blood pressure, heart rate, etc. This all-inclusive approach to health data access can evaluate the patient's lifestyle and how it affects their health.

3. Save time and energy for both patients and doctors

Both patients and doctors will save up their time whenever they want to go to the clinic and give treatment. As for patients, they do not have to be worried because the clinic will arrange the perfect time to do the check-up or to take subscribed medicines. As for doctors, they do not have to worry about the time that patients will come to see them. With the existence of the medical appointment, doctors will use the perfect time that patients need to come without being interrupted by other patients' time.

4. **Manage both schedules wisely**

Patients do not have to feel irritated about their routine schedule just to go to the clinic anymore. They just have to come on the day and time that have been set only by the clinic authorities. This way, will lessen their worry and they can continue doing their routine schedule as usual. As for doctors, they might get an emergency leave sometimes. As the medical appointment is being introduced, their duties are manageable because it will not disturb their schedules.

5. **Reducing phone calls**

The existing appointment system makes staff's work in clinics easier. They do not have to answer multiple calls from various patients daily. It will also save clinics' telephone billing. An online appointment also prevents staff from answering unimportant calls or scams.

3.0 PSEUDOCODE:

PSEUDOCODE APPOINTMENT BOOKING

1. Start
2. Input patient ID
3. Print "Enter patient ID"
4. Input the patient's name
5. Print "Enter patient's name:"
6. Input the doctor's name
7. Print "Enter doctor's name:"
8. Input appointment date
9. Print "Enter appointment date (YYYY-MM-DD):"
10. Input appointment time
11. Print "Enter appointment date (HH:MM):"
12. Save the appointment details (patient ID, patient's name, doctor's name, appointment date, appointment time)
13. Print "Appointment booked successfully!"
14. End

Figure 1: Pseudocode Appointment Booking

PSEUDOCODE PATIENT

```
1. Start
2. Define a function 'add patient':
3. Input "Enter patient name:"
4. Input "Enter patient ID:"
5. Input "Enter patient gender (Male / Female):"
6. Input "Enter patient age:"
7. Input "Enter the patient phone:"
8. Create a dictionary 'patient':
9. Append 'patient' to 'patient list'
10. Print "Patient added successfully!"
11. If 'nurse list' is empty:
12. Print "No nurses found!"
13. Else for each nurse in 'nurse list'
14. Print "nurse name"
15. Print "patient ID"
```

Figure 2: Pseudocode Patient

```
16. Print "patient gender (Male / Female)"
17. Print "patient age"
18. Print "patient phone"
19. Create an empty list 'patient list'
20. While true:
21. Print '1. Add patient"
22. Print "2. View patient"
23. Print "3. Exit"
24. Input "choose an option:"
25. If 'choice is 1: 'add patient'
26. Elif 'choice' is 2: 'view patient'
27. Elif 'choice' is 3: 'exiting'
28. Break
29. Else: "invalid option! Please try again."
30. End
```

Figure 3: Pseudocode Patient

PSEUDOCODE NURSE

```
1. Start
2. Define a function 'add nurse':
3. Input "Enter nurse name:"
4. Input "Enter nurse ID:"
5. Input "Enter nurse gender (Male / Female):"
6. Input "Enter nurse department:"
7. Input "Enter the nurse phone:"
8. Input "Enter nurse shift time:"
9. Create a dictionary 'nurse':
10. Append 'nurse' to 'nurse list'
11. Print "Nurse added successfully!"
12. If 'nurse list' is empty:
13. Print "No nurses found!"
14. Else for each nurse in 'nurse list'
15. Print "nurse name"
16. Print "nurse ID"
```

Figure 4: Pseudocode Nurse

```
17. Print "nurse gender (Male / Female)"
18. Print "nurse department"
19. Print "nurse phone"
20. Print "nurse shift time"
21. Create an empty list 'nurse list'
22. While true:
23. Print '1. Add nurse"
24. Print "2. View nurse"
25. Print "3. Exit"
26. Input "choose an option:"
27. If 'choice is 1: 'add nurse'
28. Elif 'choice' is 2: 'view nurse'
29. Elif 'choice' is 3: 'exiting'
30. Break
31. Else: "invalid option! Please try again."
32. End
```

Figure 5: Pseudocode Nurse

PSEUDOCODE DOCTOR

1. Start
2. Define a function 'add doctor':
3. Input "Enter doctor name:"
4. Input "Enter doctor ID:"
5. Input "Enter doctor gender (Male / Female):"
6. Input "Enter doctor specialty:"
7. Input "Enter the doctor phone:
8. Input "Enter doctor working hour:"
9. Create a dictionary 'doctor':
10. Append 'doctor' to 'doctor list'
11. Print "doctor added successfully!"
12. If 'doctor list' is empty:
13. Print "No doctor found!"
14. Else for each doctor in 'doctor list'
15. Print "doctor name"

Figure 6: Pseudocode Doctor

16. Print "doctor ID"
17. Print "doctor gender (Male / Female)"
18. Print "doctor specialty"
19. Print "doctor phone"
20. Print "doctor working hour"
21. Create an empty list 'doctor list'
22. While true:
23. Print '1. Add doctor"
24. Print "2. View doctor"
25. Print "3. Exit"
26. Input "choose an option:"
27. If 'choice is 1: 'add doctor'
28. Elif 'choice' is 2: 'view doctor'
29. Elif 'choice' is 3: 'exiting'
30. Break
31. Else: "invalid option! Please try again."
32. End

Figure 7: Pseudocode Doctor

PSEUDOCODE MEDICINE

```
1. Start
2. Define a function 'add medicine':
3. Input "Enter medicine name:"
4. Input "Enter medicine type (Tablet / Syrup / Ointment):"
5. Input "Enter medicine quantity:"
6. Input "Enter medicine expired date:"
7. Input "Enter medicine price:"
8. Create a dictionary 'medicine':
9. Append 'medicine' to 'medicine list'
10. Print "Medicine added successfully!"
11. If 'medicine list' is empty:
12. Print "No medicines found!"
13. Else for each medicine in 'medicine list'
14. Print "medicine name"
```

Figure 8: Pseudocode Medicine

```
15. Print "medicine types (Tablet / Syrup / Ointment)"
16. Print "medicine quantity"
17. Print "medicine expired date"
18. Print "medicine price"
19. Create an empty list 'medicine list'
20. While true:
21. Print '1. Add medicine"
22. Print "2. View medicine"
23. Print "3. Exit"
24. Input "choose an option:"
25. If 'choice is 1: 'add medicine'
26. Elif 'choice' is 2: 'view medicine'
27. Elif 'choice' is 3: 'exiting'
28. Break
29. Else: "invalid option! Please try again."
30. End
```

Figure 9: Pseudocode Medicine

4.0 FLOWCHART

- **FLOWCHART APPOINTMENT BOOKING**

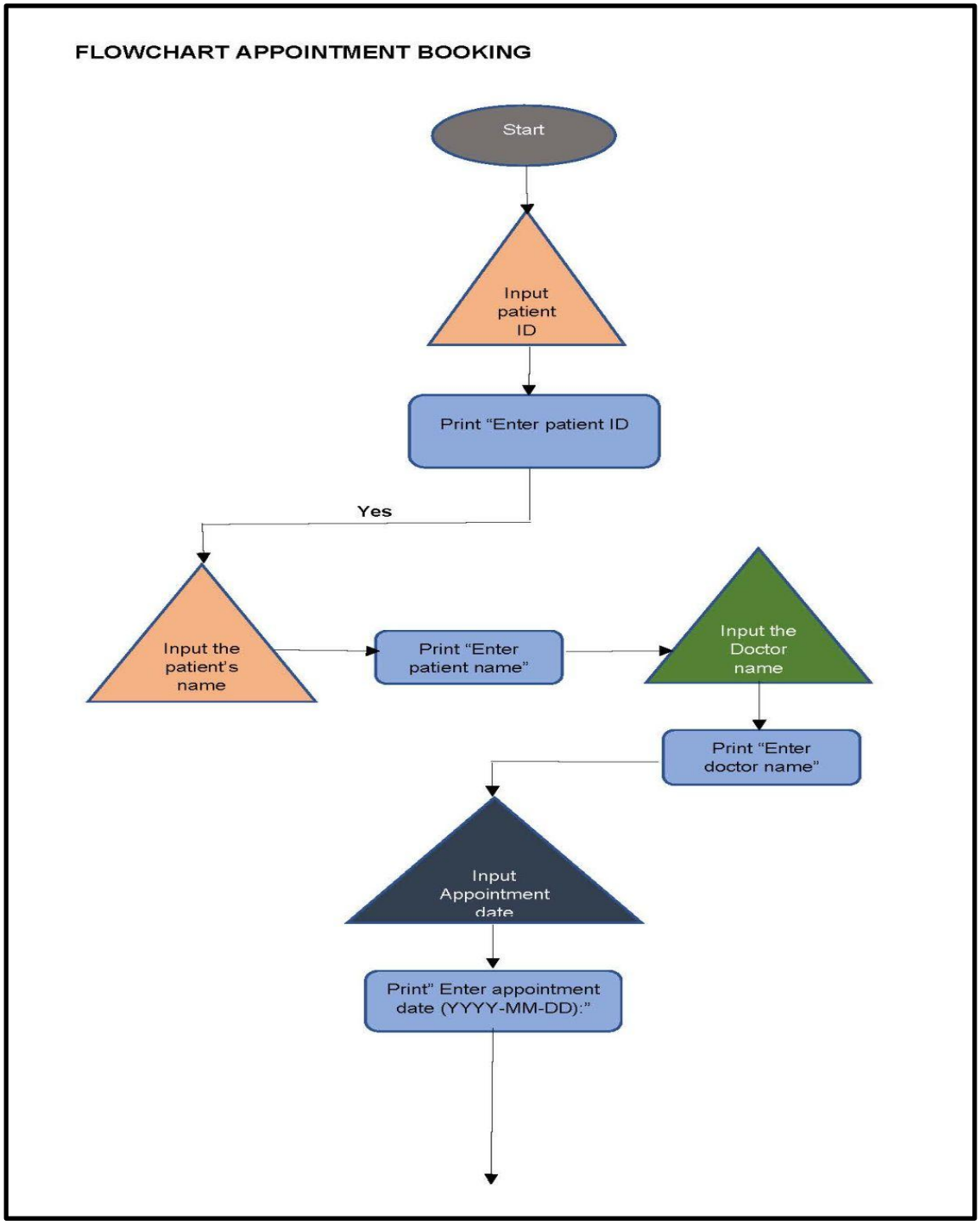


Figure 10: Flowchart Appointment Booking

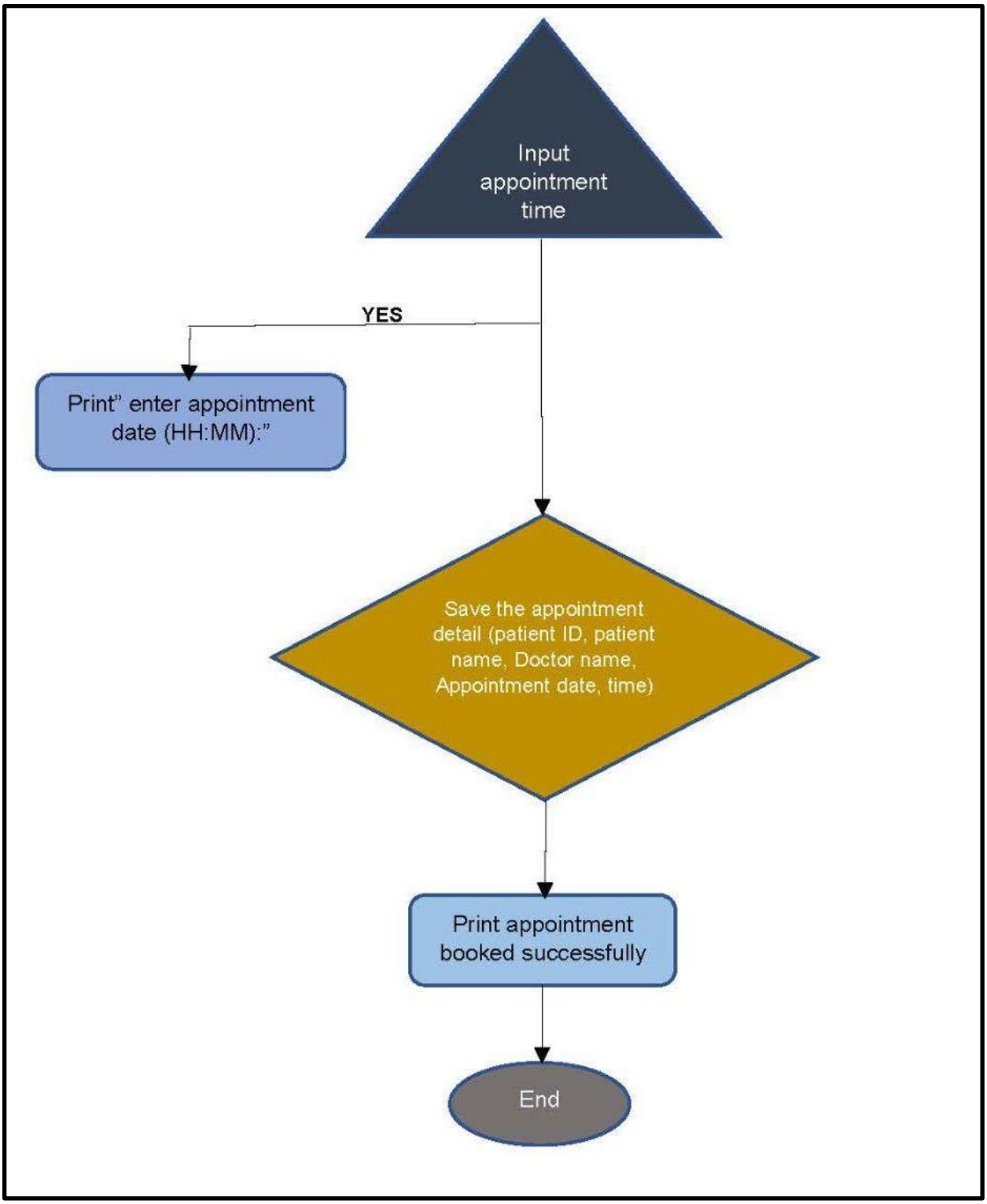


Figure 11: Flowchart Appointment Booking

- **FLOWCHART PATIENT**

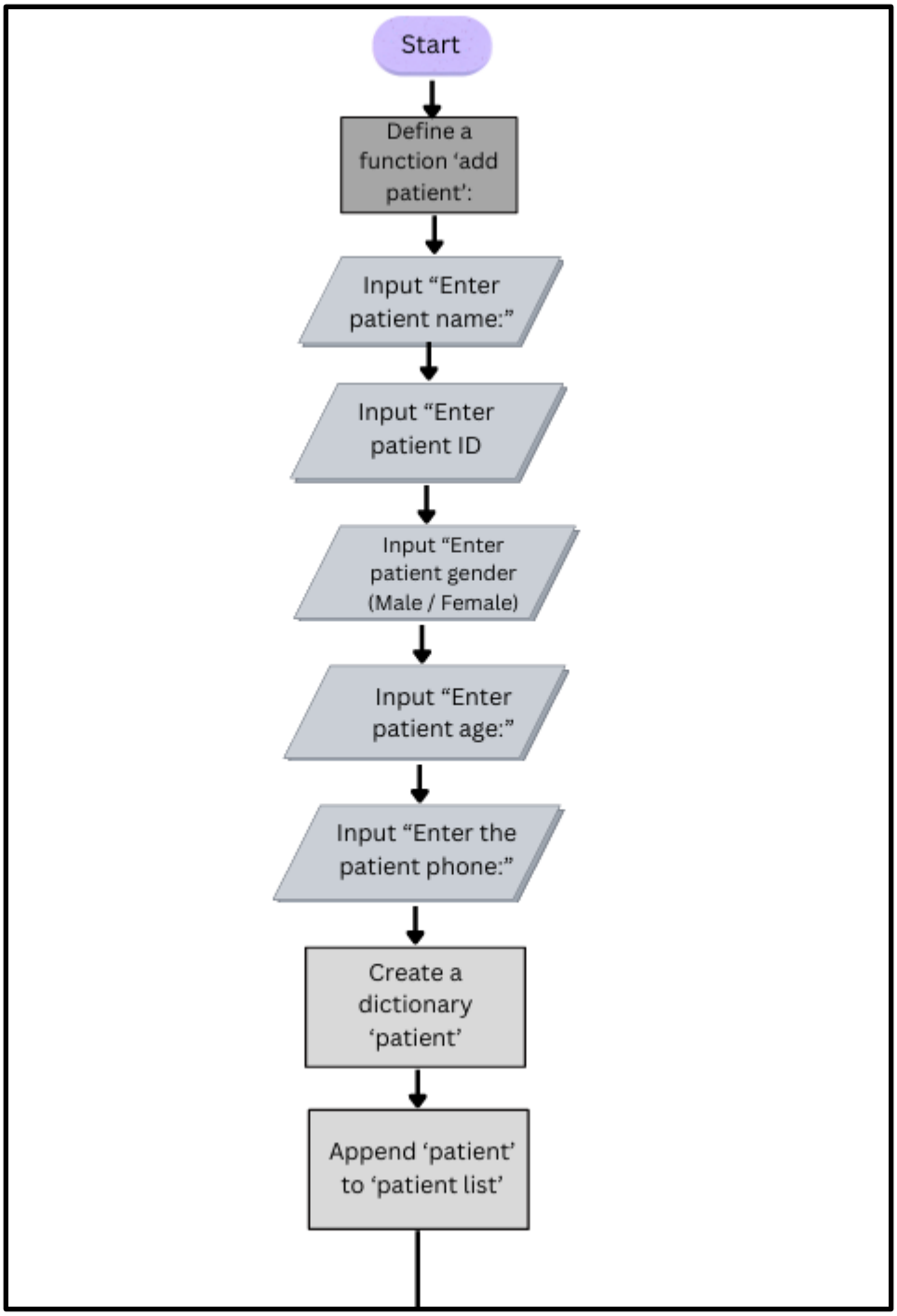


Figure 12: Flowchart Patient

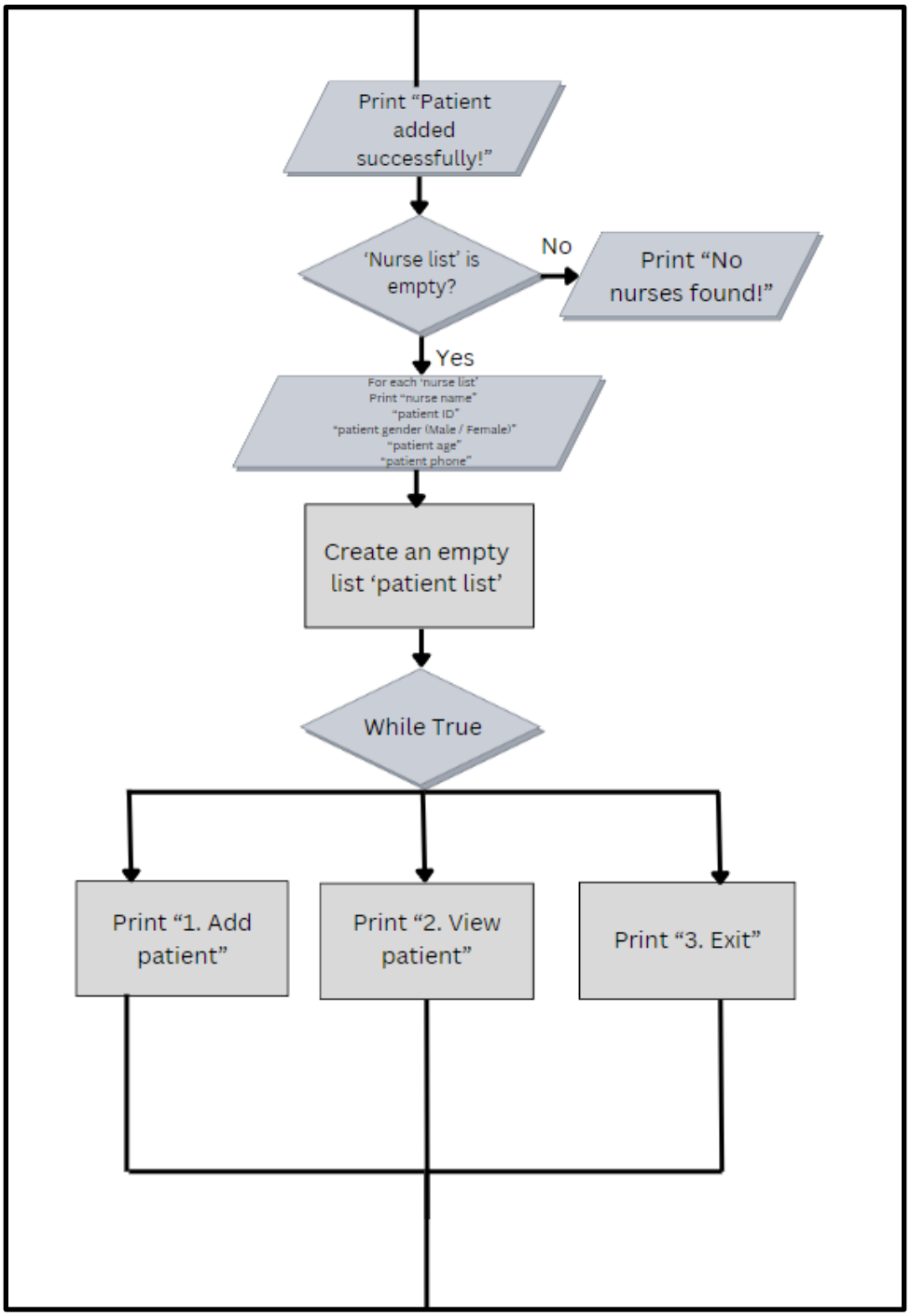


Figure 13: Flowchart Patient

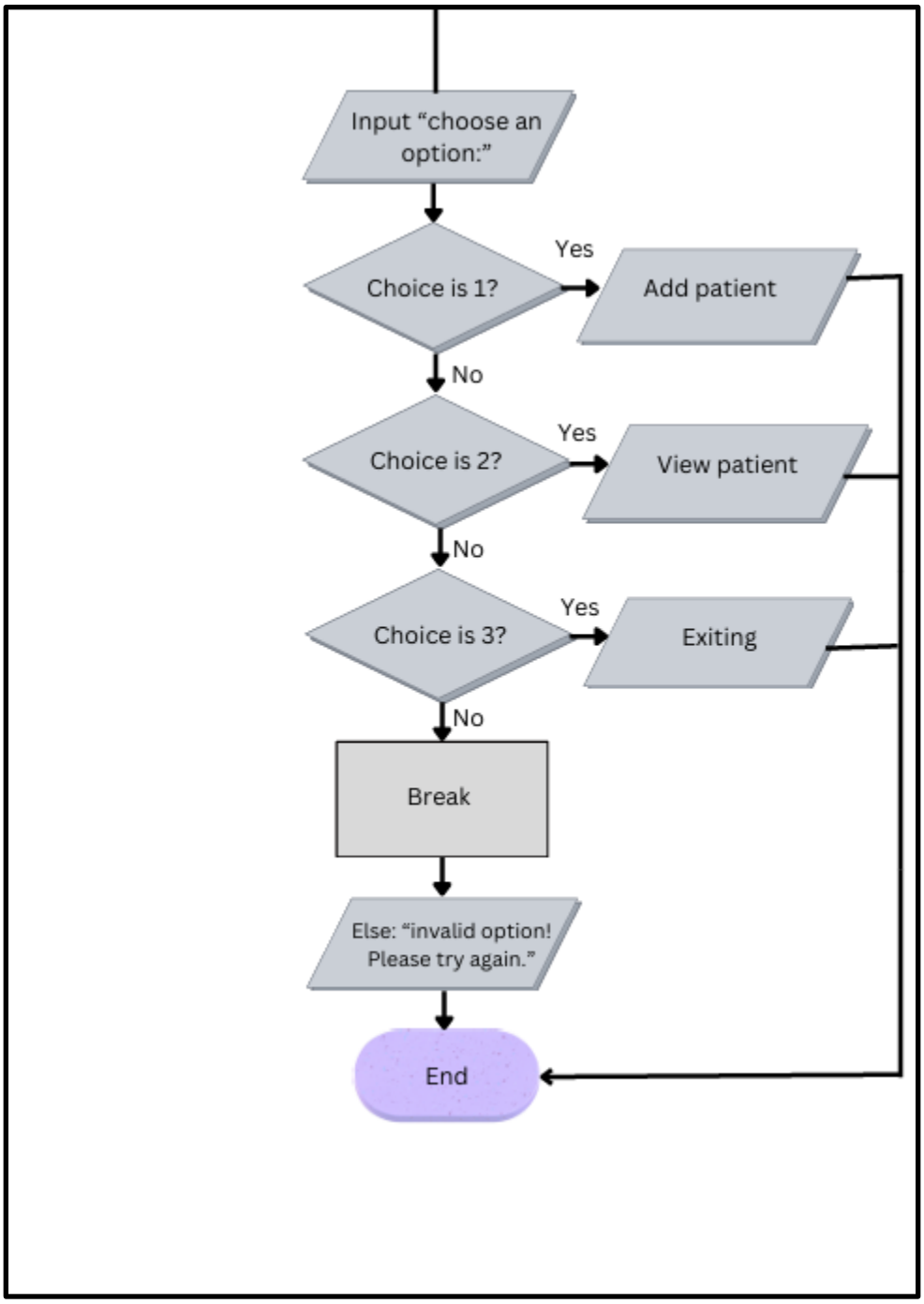


Figure 14: Flowchart Patient

- **FLOWCHART NURSE**

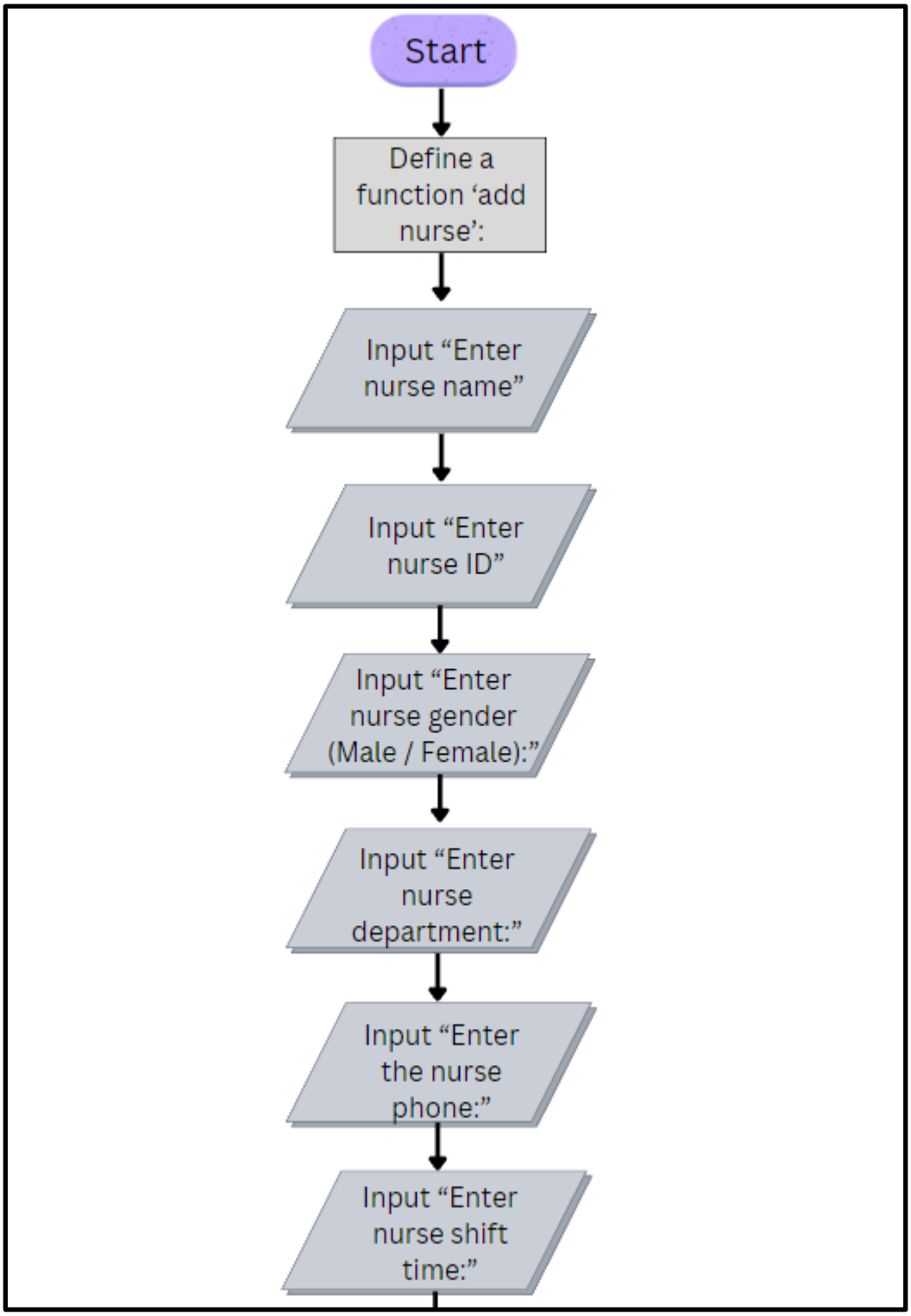


Figure 15: Flowchart Nurse

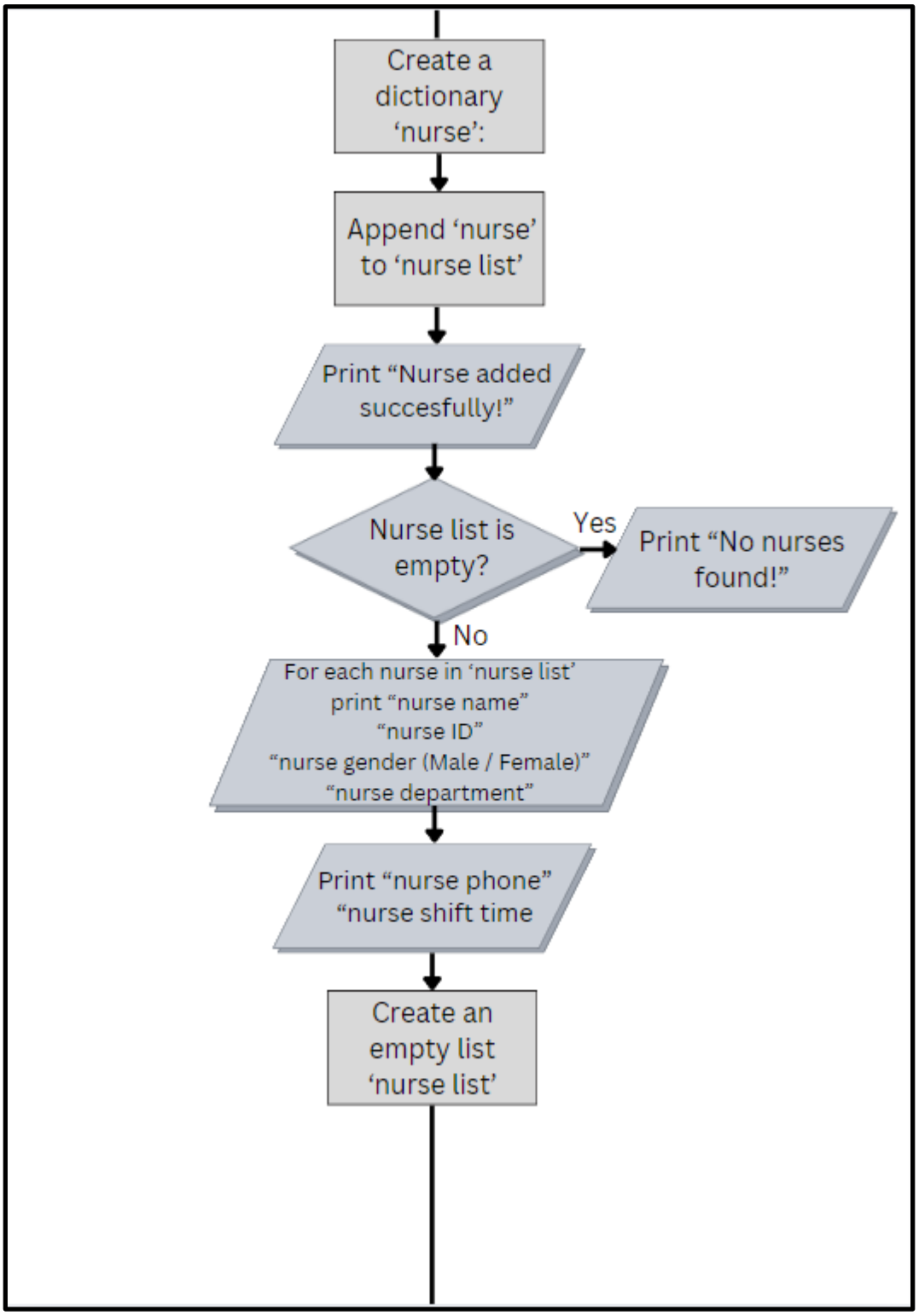


Figure 16: Flowchart Nurse

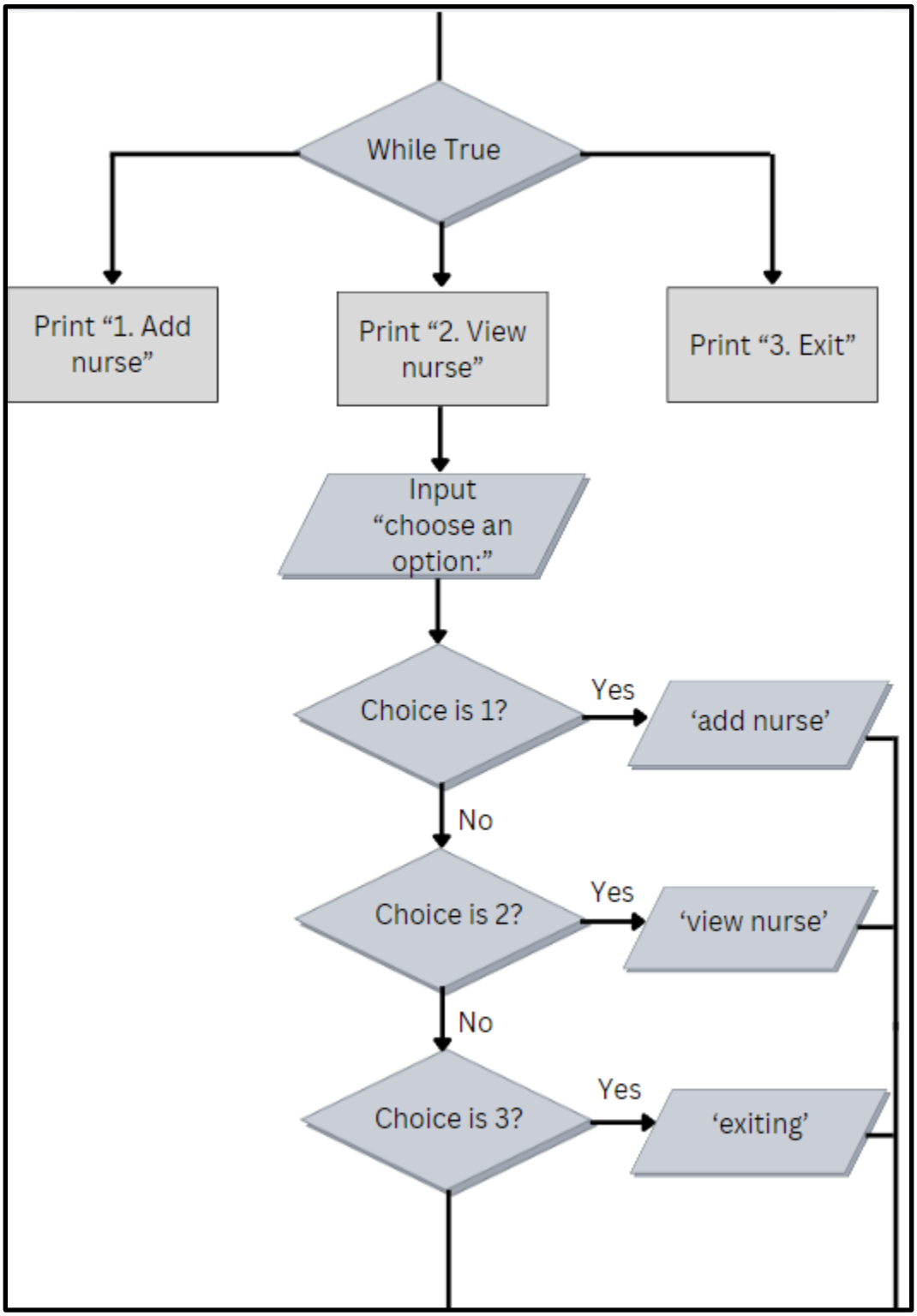


Figure 17: Flowchart Nurse

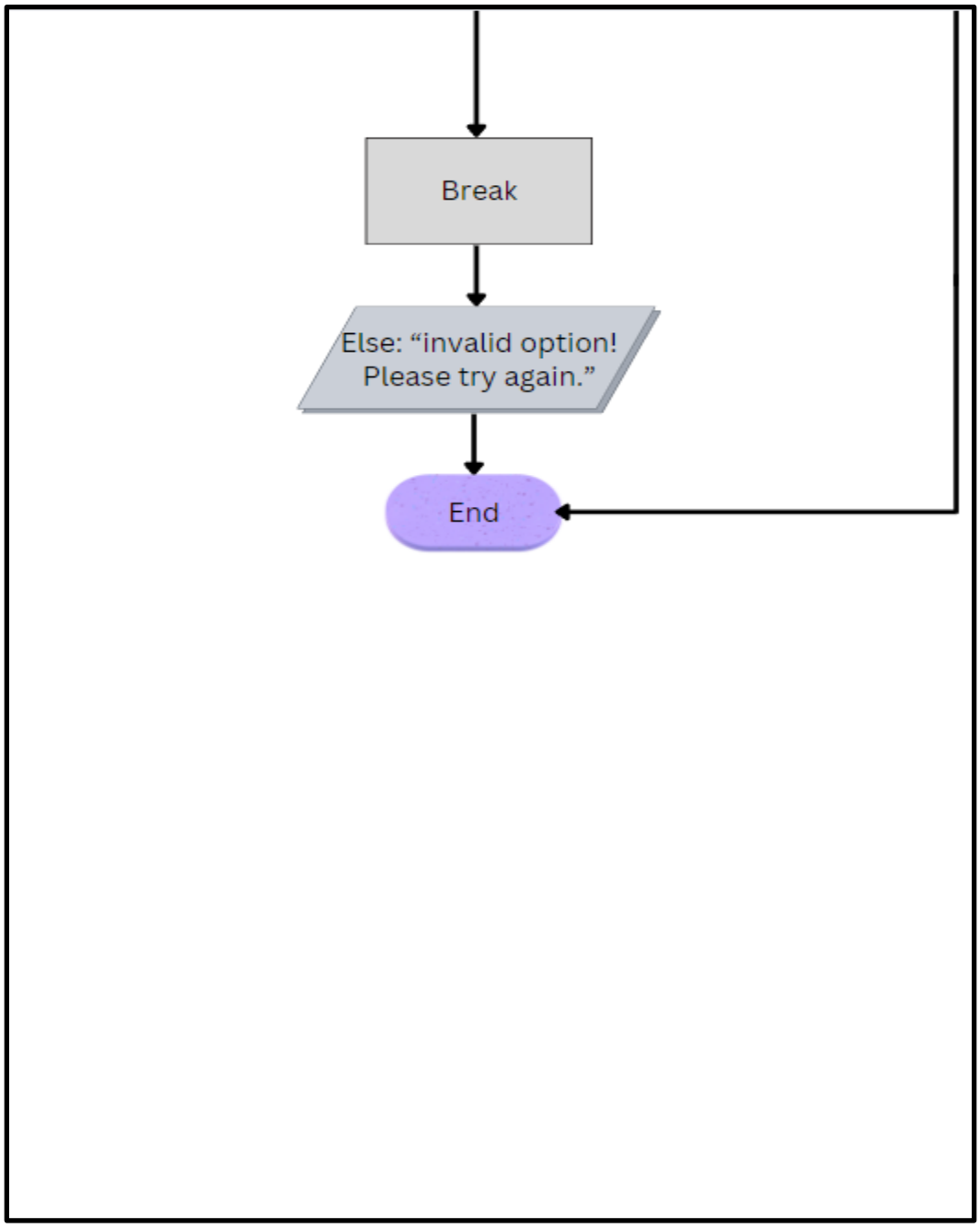


Figure 18: Flowchart Nurse

- **FLOWCHART DOCTOR**

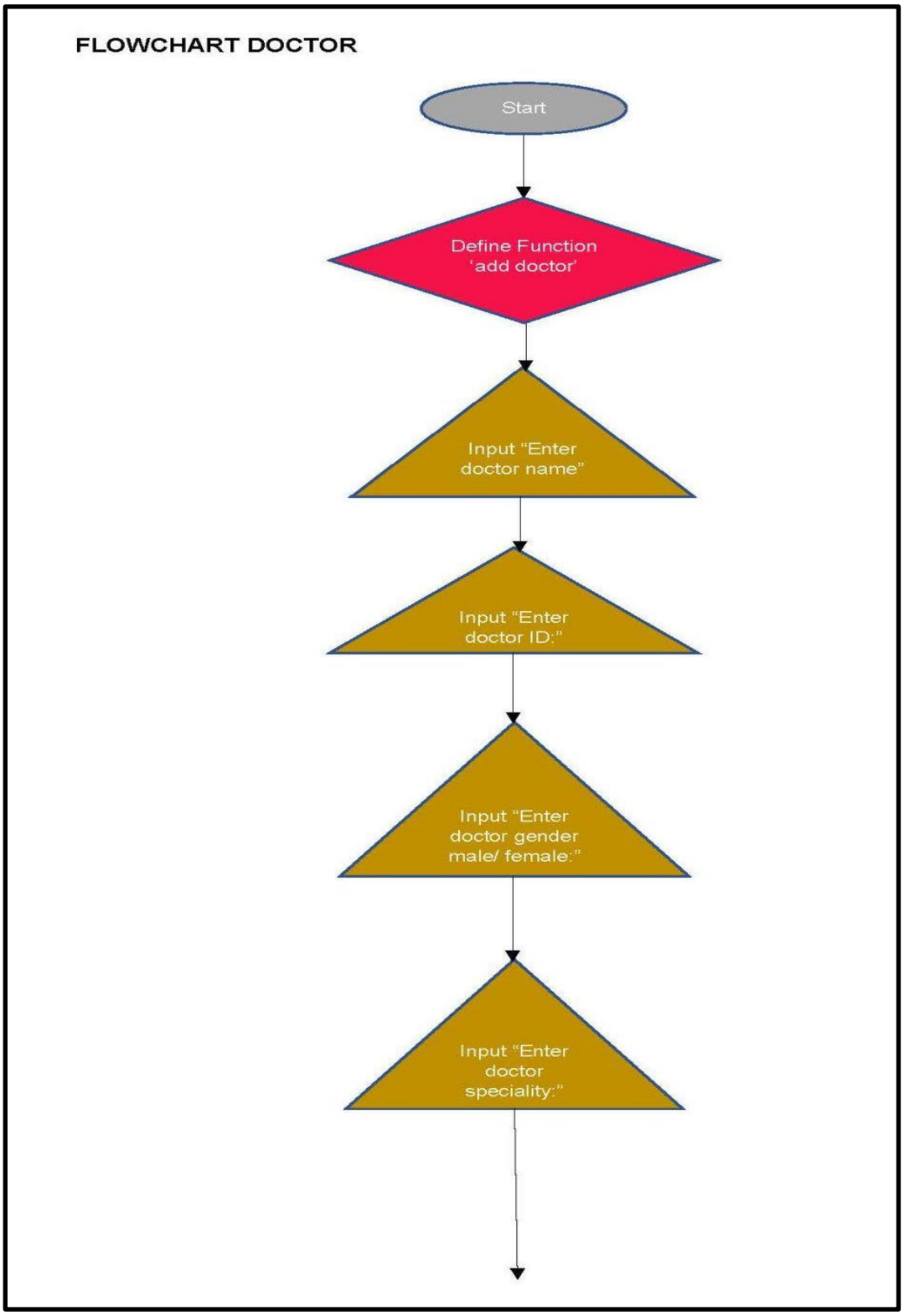


Figure 19: Flowchart Doctor

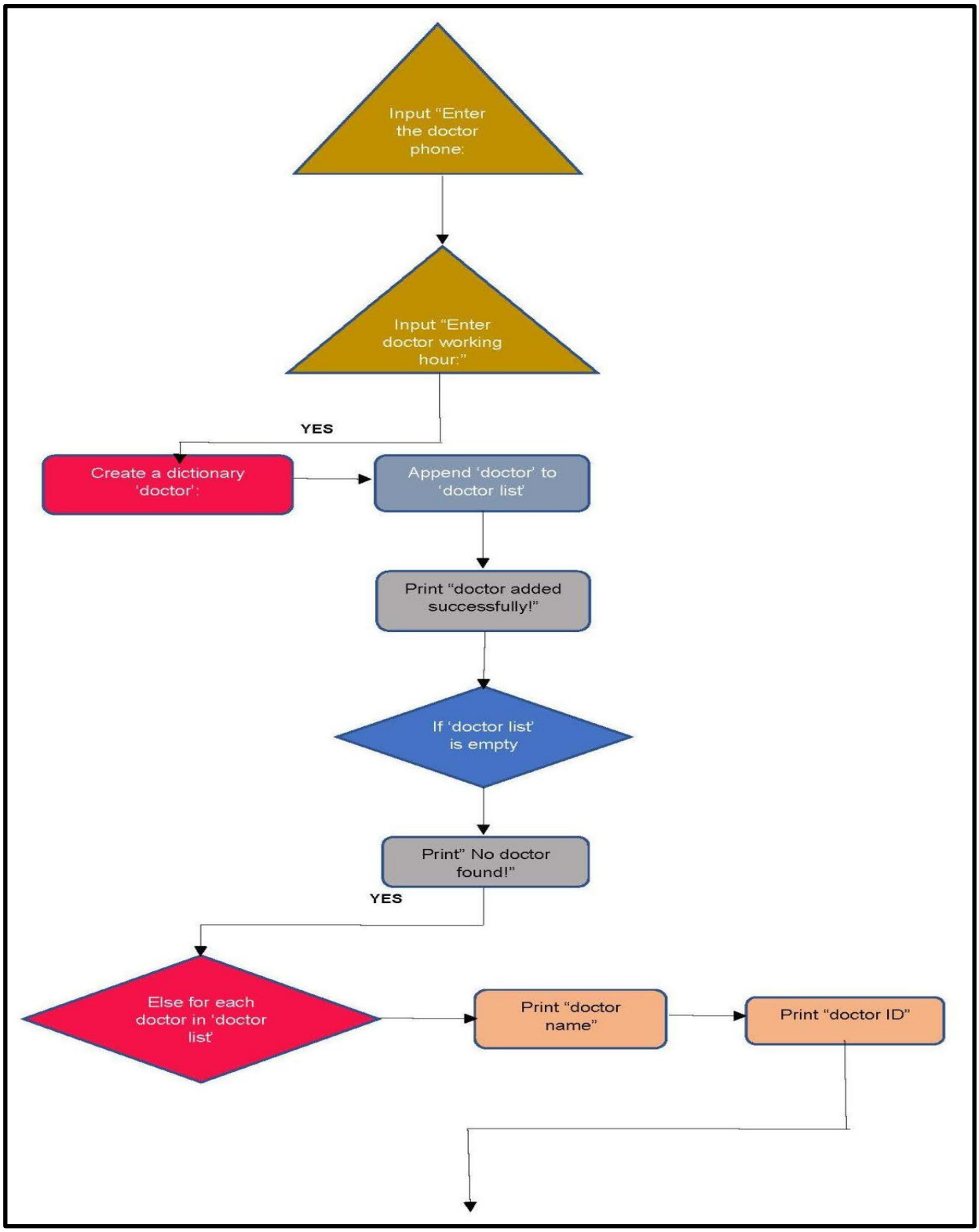


Figure 20: Flowchart Doctor

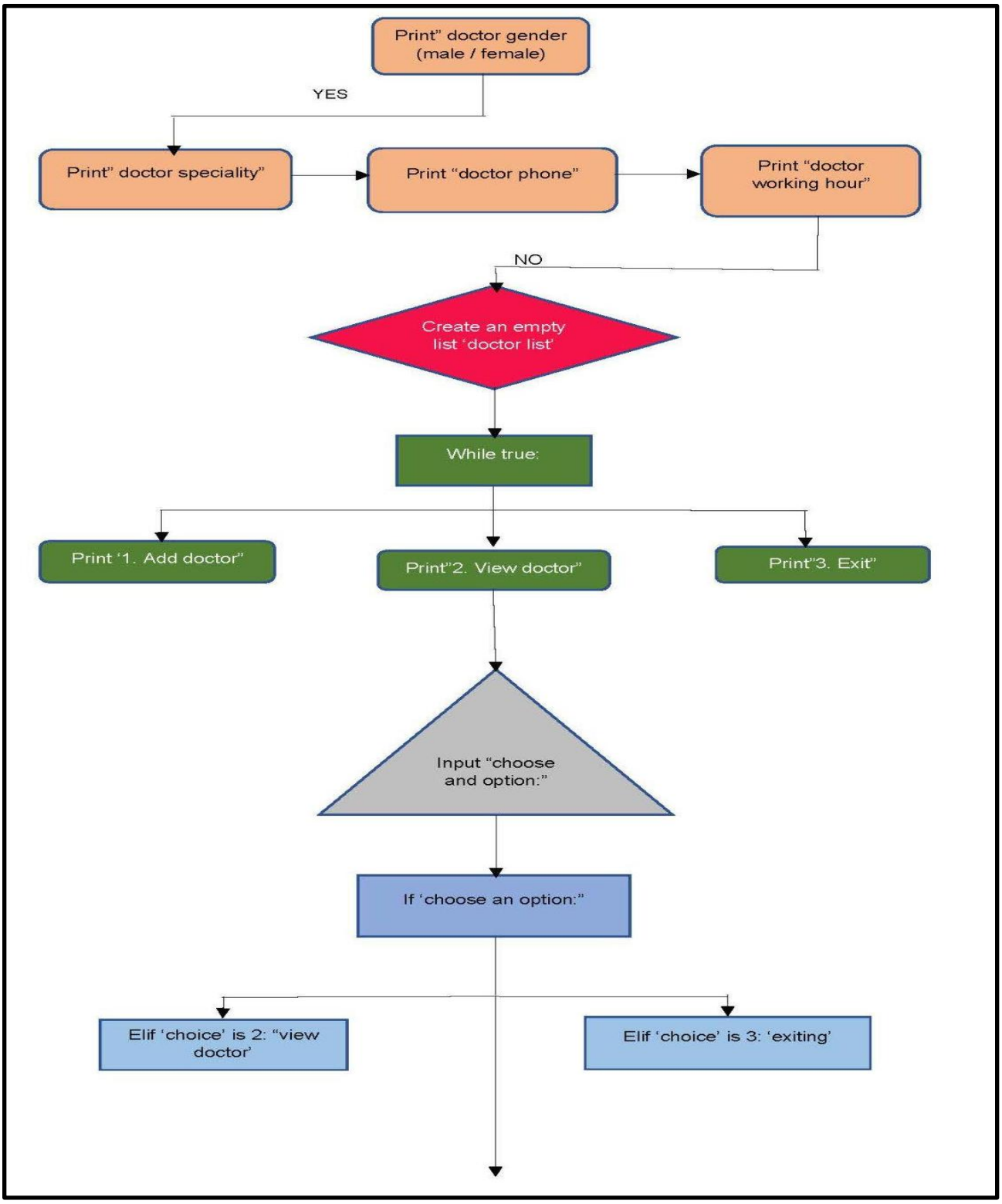


Figure 21: Flowchart Doctor

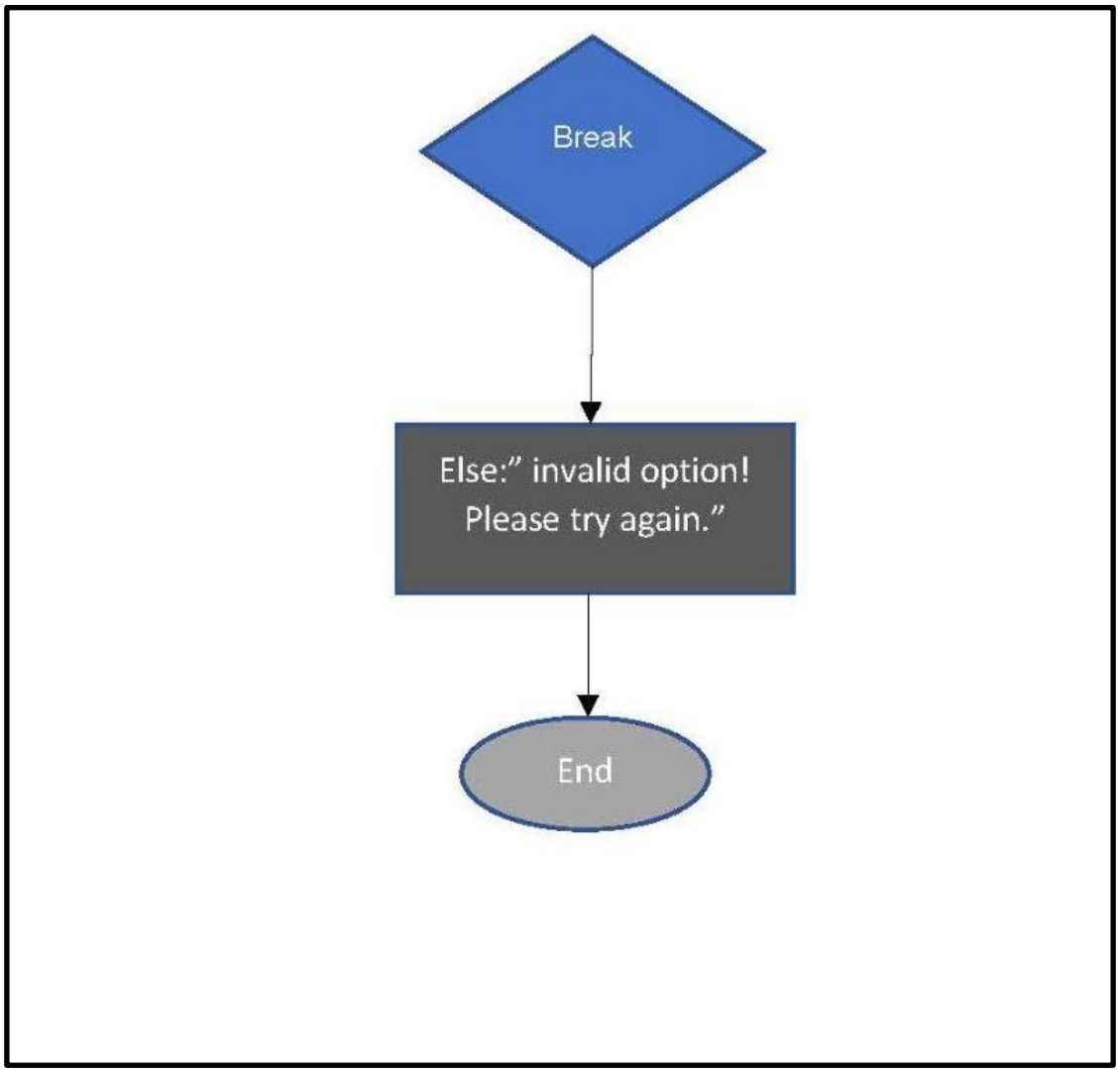


Figure 22: Flowchart Doctor

- **FLOWCHART MEDICINE**

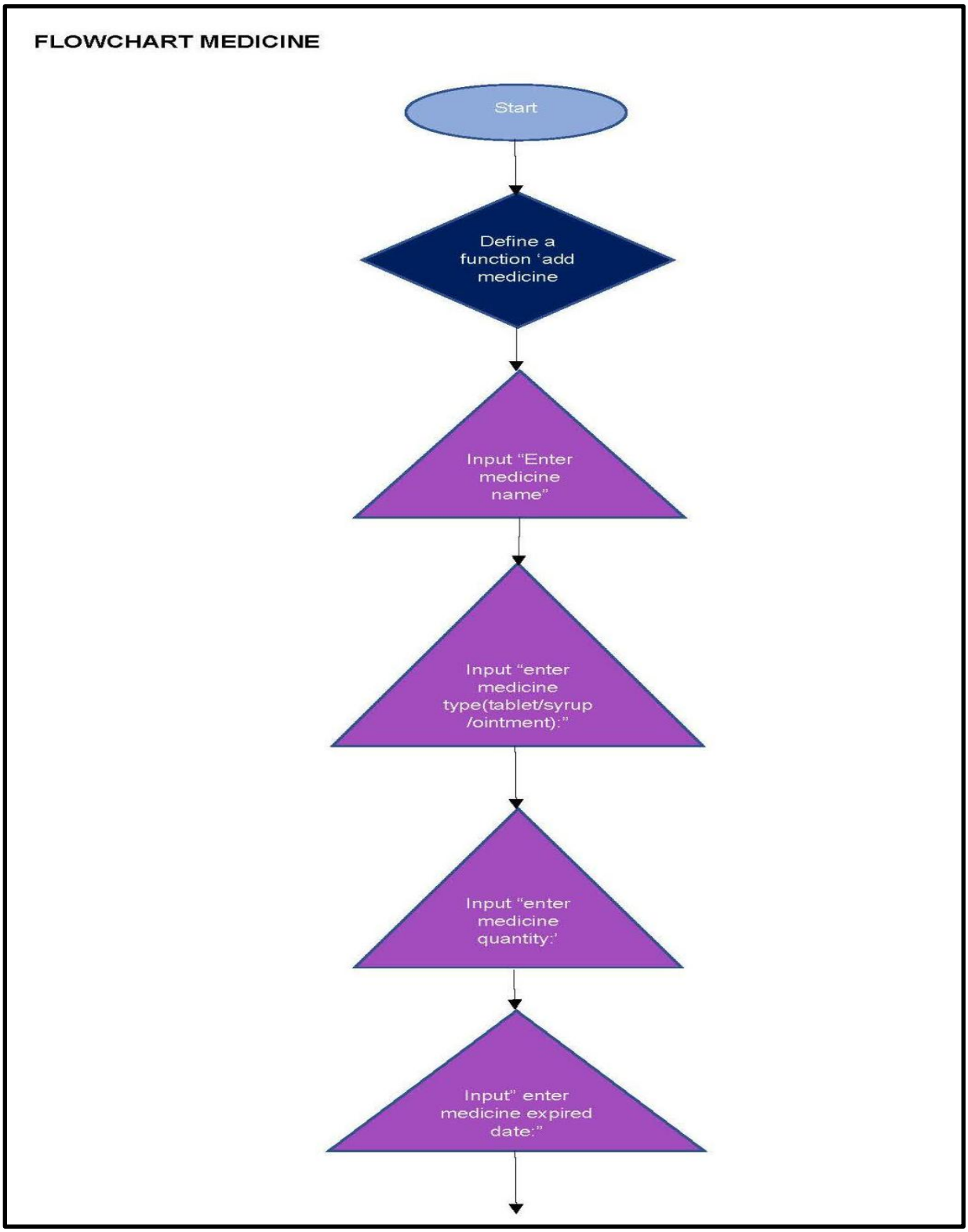


Figure 23: Flowchart Medicine

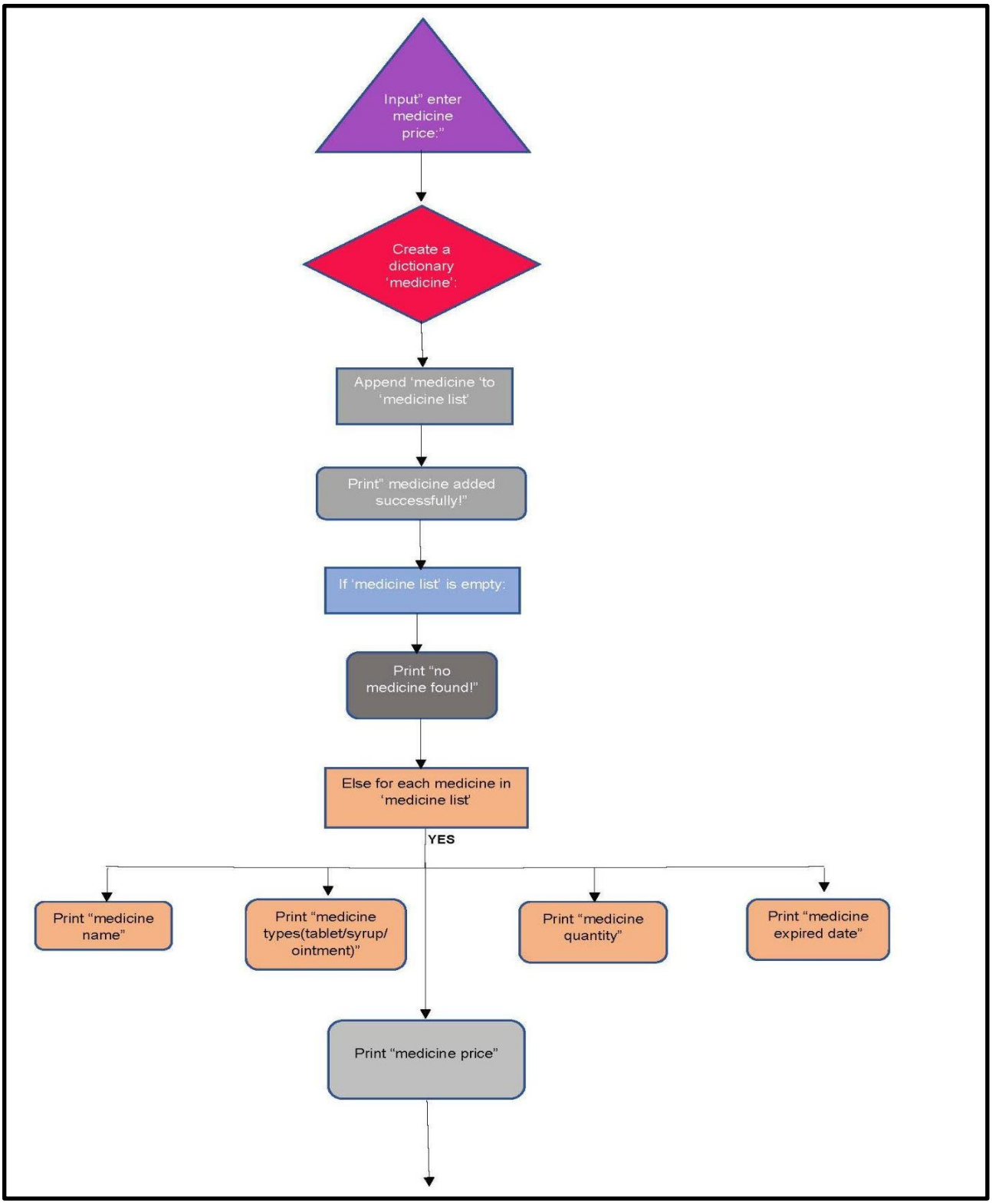


Figure 24: Flowchart Medicine

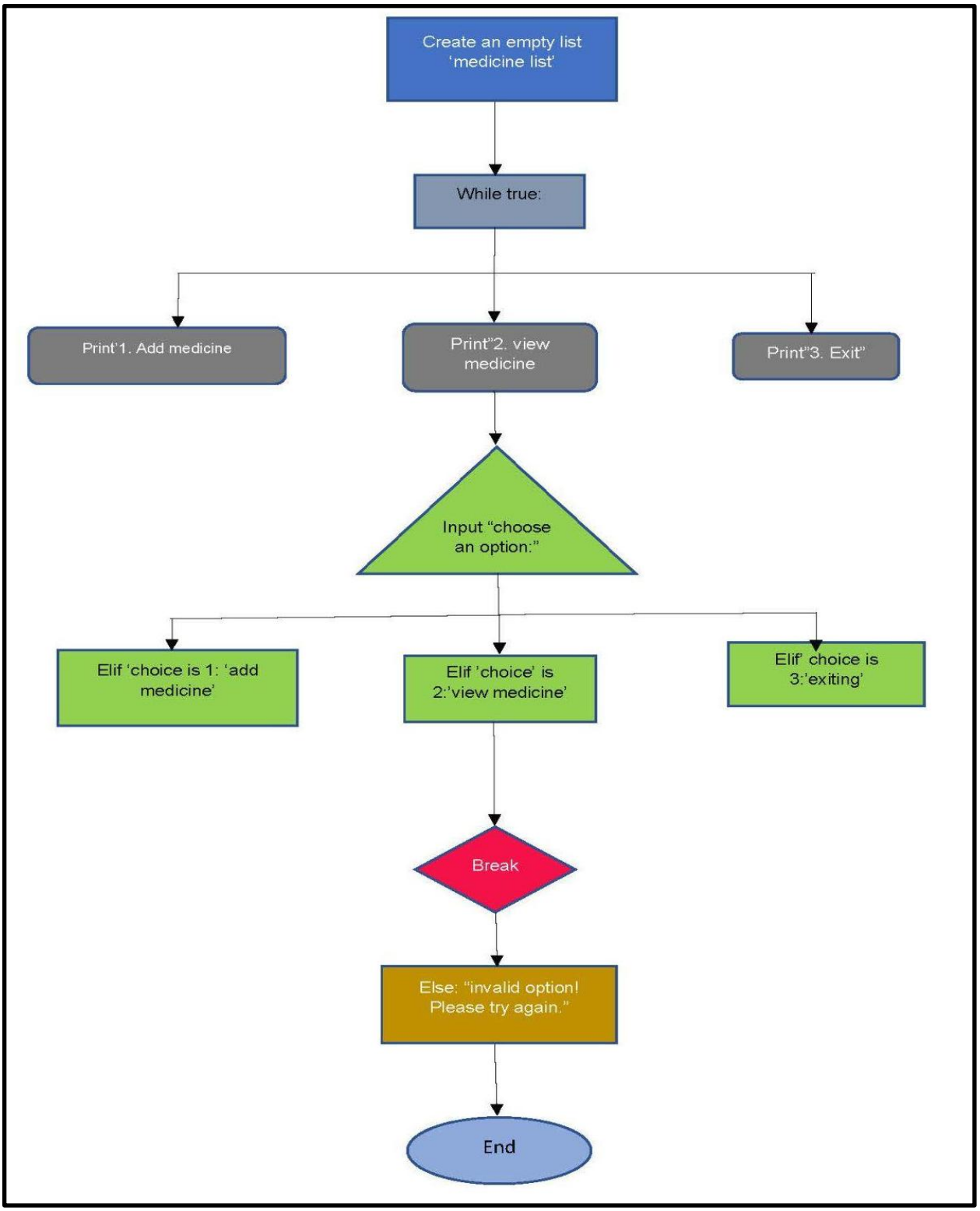


Figure 25: Flowchart Medicine

5.0 STRENGTH:

1. **Each service is separated so that every appointment leads to the right department.**

Separate services ensure proper specialists or departments are in charge of getting accurate disease diagnoses for the patient. For instance, pregnant moms usually go to the maternal room for their pregnancy check-ups and current health conditions as pregnant moms. The patient will experience effective treatment due to the organized appointment.

2. **Efficient appointment booking method because it can update all information through the patient ID.**

Unexpected events can occur anytime. By using the appointment booking system, it can change to other free time or schedules. For example, a patient can just call the clinic to adjust the appointment to the next day or any day just by informing their identity details. This system will improve patient's participation in that particular clinic.

3. **Both doctors and nurses can leave their work on time without any additional working hours due to the leftover patients by referring to the appointment system.**

An appointment booking system will ensure doctors' and nurses' shifts during work days. They will just follow the list of appointments given without needing to work overtime. This will give them space to rest and continue to treat people in need.

4. **Simplified Patient Registration and Login features created specifically to make it easier for patients to access systems and accounts and to expedite the procedure.**

Patients can register and log in using the system's safe and intuitive platform. This can ensure speedy access to profiles, medical histories, and appointment plans, increase user convenience, reduce repetitive data entry, and guard against user data loss.

5. **Enhanced Patient Experience focuses on developing user-friendly, efficient, and personalized experiences with clinics.**

Patients can quickly access their accounts, view future appointments, and receive appointment date reminders, achieving a seamless and fulfilling user experience. This guarantees clear communication between patients and clinics, reduces waiting times, makes scheduling easier, and offers real-time information.

6.0 KAIZEN (ROOM FOR IMPROVEMENT):

1. **Build more security measures such as two-factor authentication to avoid being modified by the other employees.**

Security measures such as username and password before logging in to the system are crucial. This is to prevent any threats from the existing system. An insider can have bad intentions which can modify the system to persecute someone in the clinic. So, it is better to have security measures to avoid a bad reputation in the clinic.

2. **Display any visuals in every service offered so that patients know what exactly the treatment they will get.**

Visuals meaning here are any pictures regarding the services in the clinic. It can be from various departments to promote the operations they offer. Patients can browse them in the system so they can easily make the right appointment.

3. **Display categories for each patient ID in selected services too.**

By displaying categories, doctors and nurses can know their targeted patients. To illustrate, they can identify if the patient is categorized as children, adults, or elderly people. This system is run to give way to people in need.

4. **Advanced Appointment Management**

should concentrate on streamlining the scheduling procedure so that both patients and clinic employees can manage appointments in a timely and efficient manner. To automatically fill canceled spaces, for instance, a waiting list function may be implemented. Additionally recommended Additionally, incorporate predictive bookings to recommend the best timeslots based on historical data, enabling group reservations for health programs or family checks.

5. **Improved communication, including two-way channels.**

To guarantee a more seamless and open healthcare experience, medicine focuses on improving relationships between the clinic and its patients. For instance, patients can seek clarification regarding medical procedures, ask questions regarding their appointments, or the other way around. The second channel is the live chat function, which allows patients to post queries in the comments section and broadcast live on social media networks.

7.0 PYTHON

FILE NAME: medical.py

GUI: YES

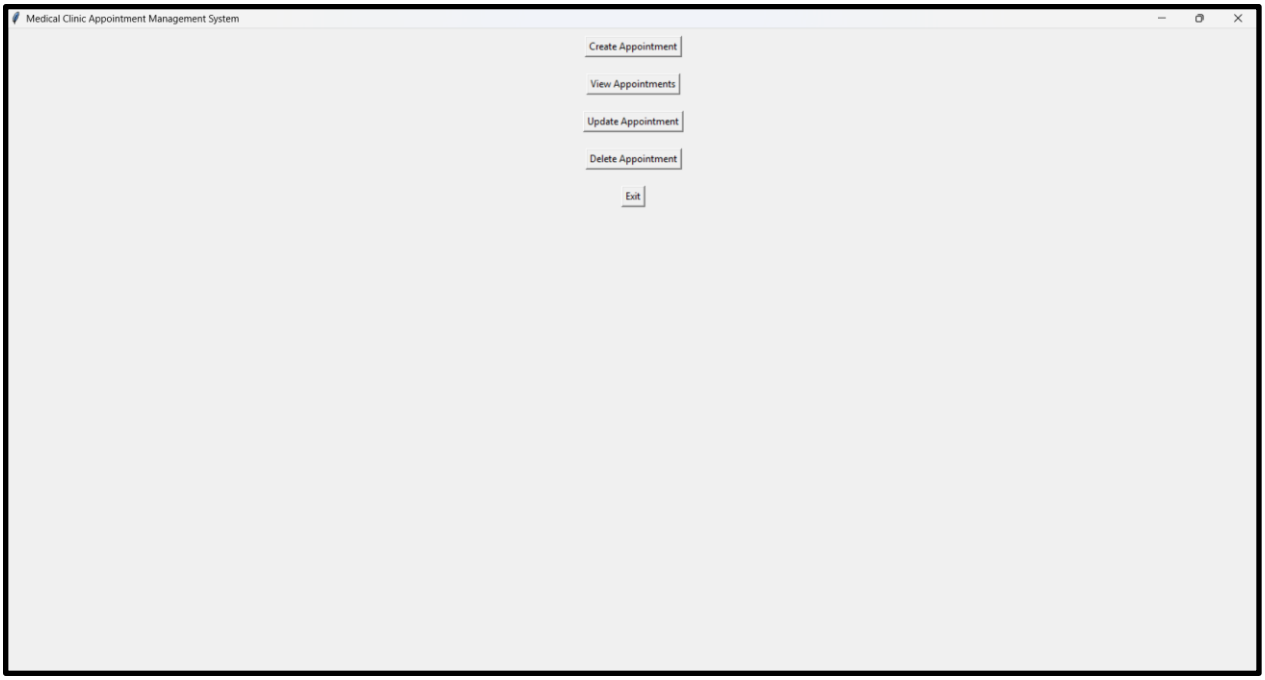


Figure 26: GUI

RESULT:

- **CREATE APPOINTMENT BOOKING**

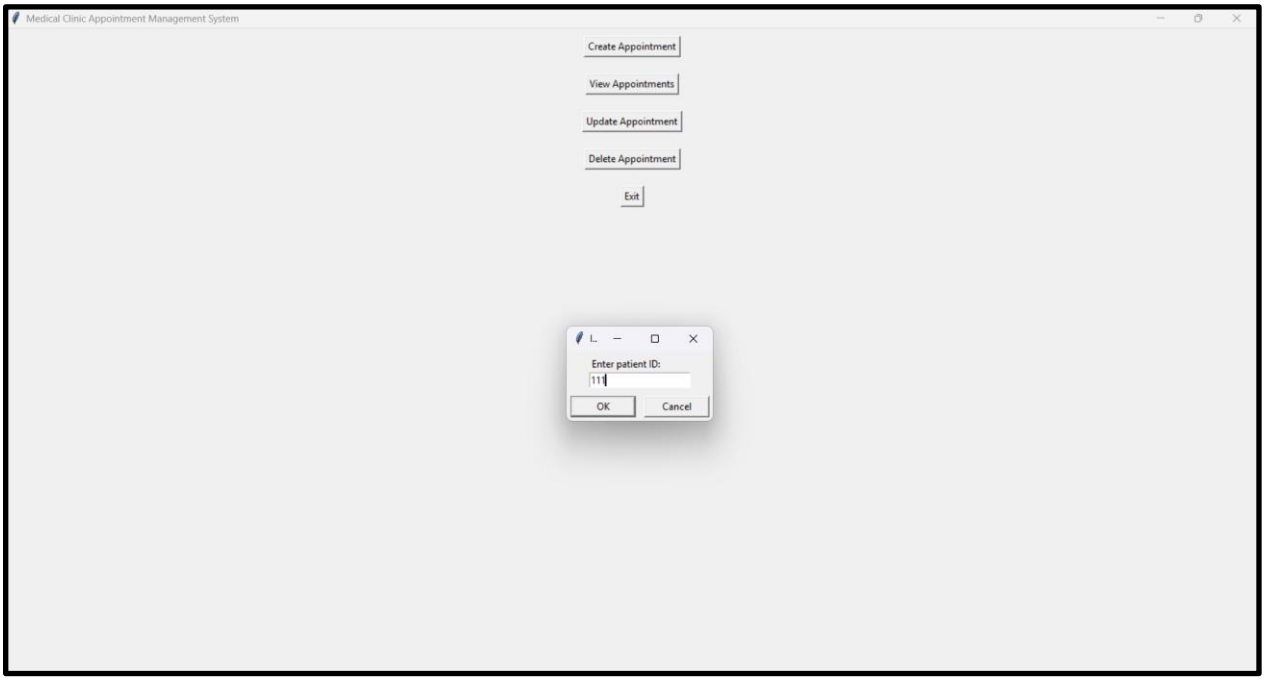


Figure 27: Result of Create Appointment Booking

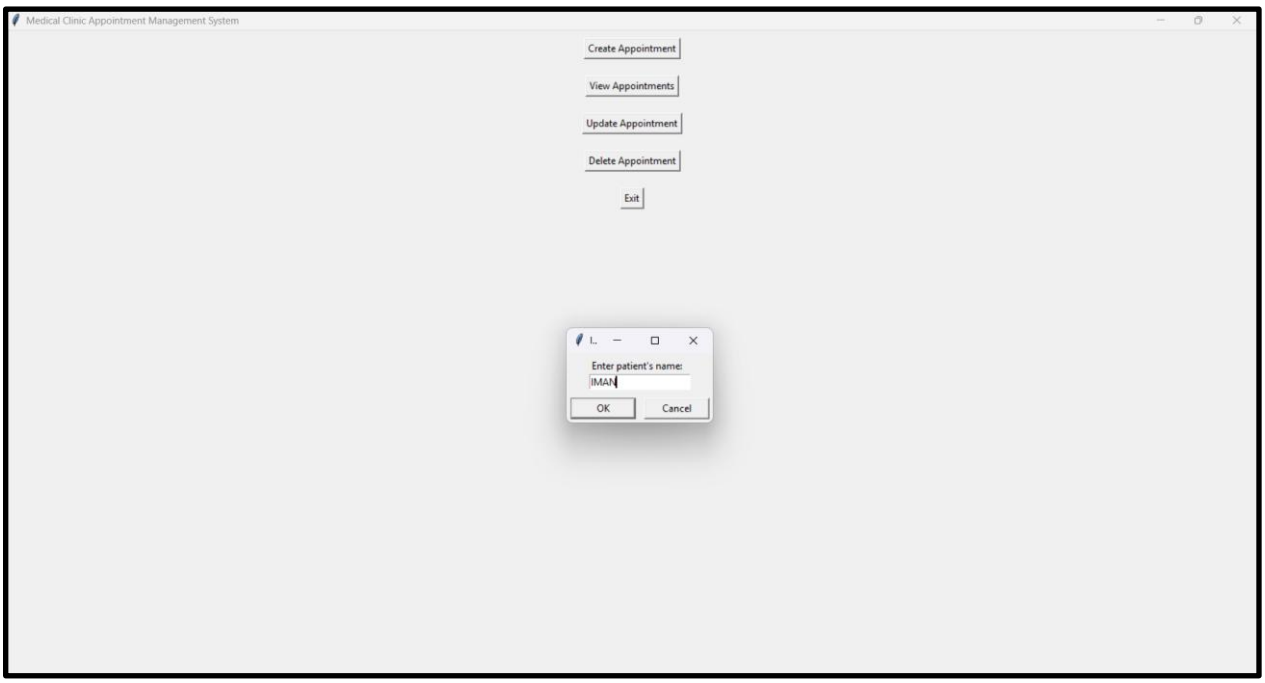


Figure 28: Result of Create Appointment Booking

Figure 29: Result of Create Appointment Booking

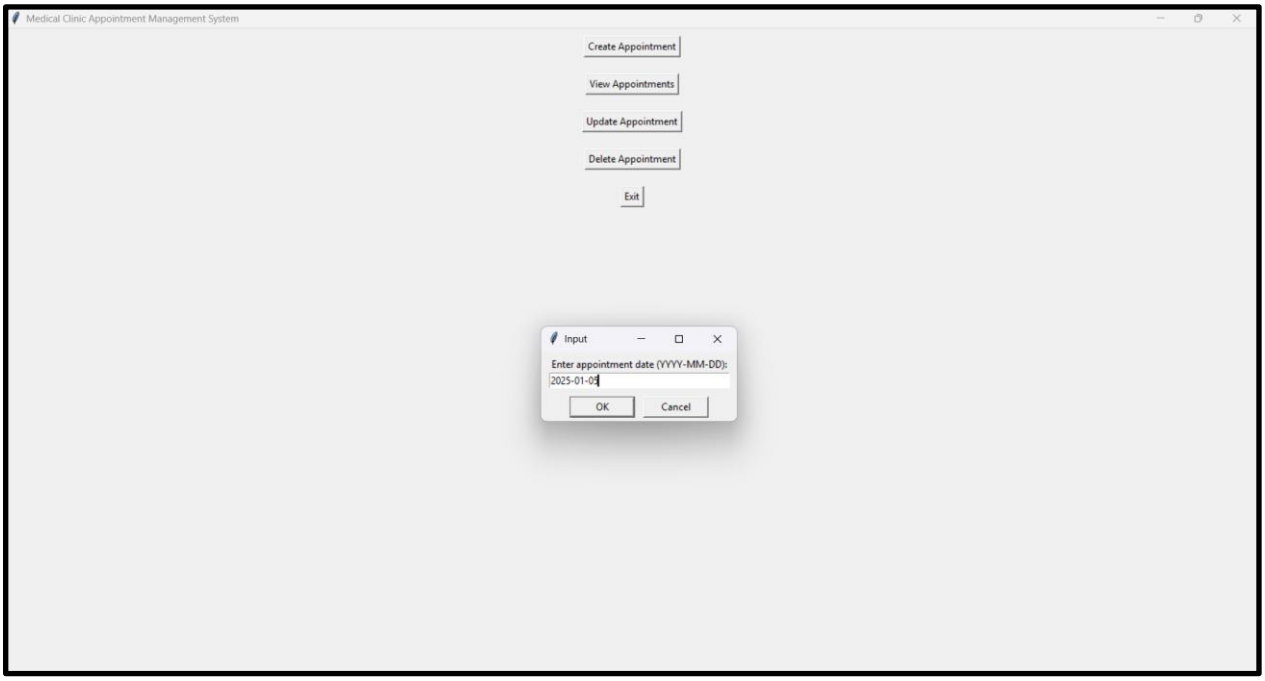


Figure 30: Result of Create Appointment Booking

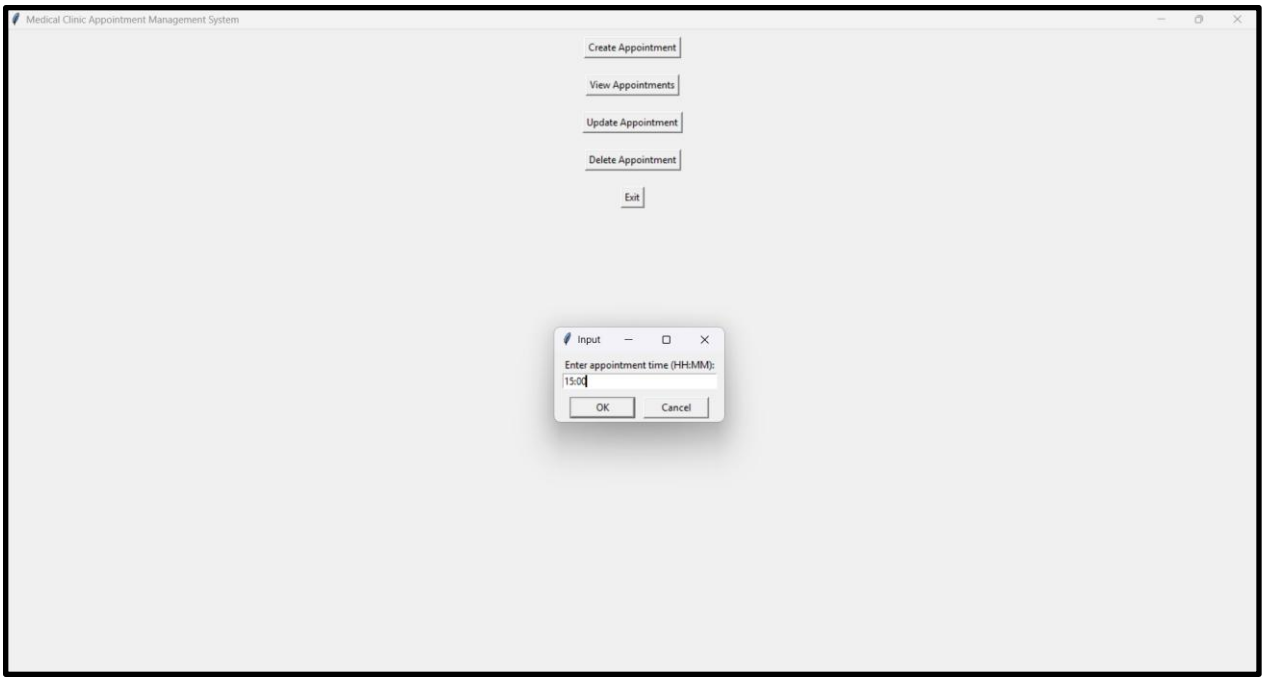


Figure 31: Result of Create Appointment Booking

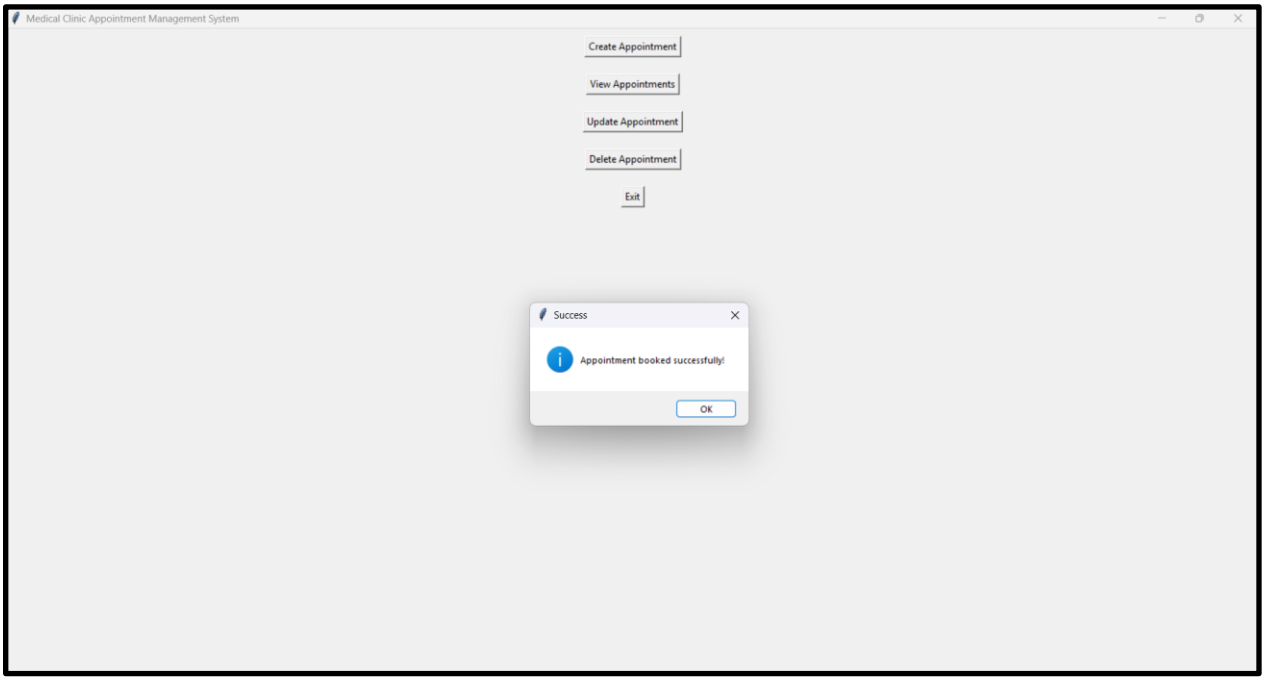


Figure 32: Result of Create Appointment Booking

- **READ/VIEW APPOINTMENT BOOKING**

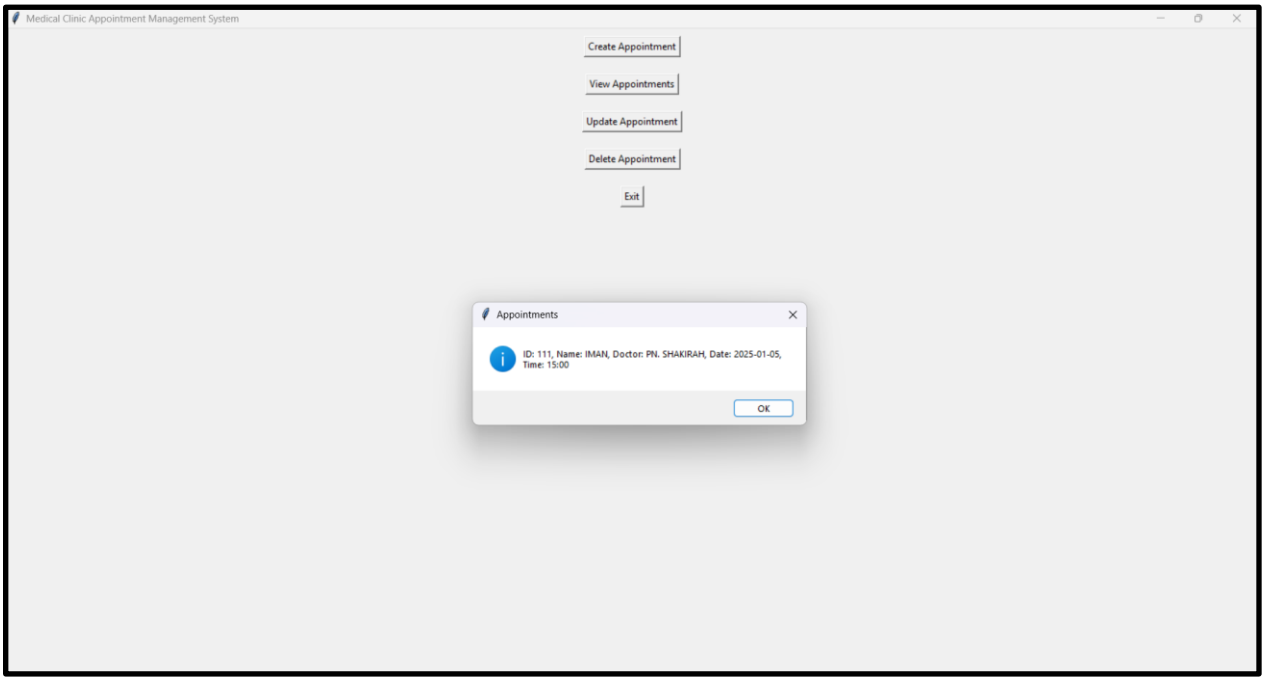


Figure 33: Result of Read/View Appointment Booking

- **UPDATE APPOINTMENT BOOKING**

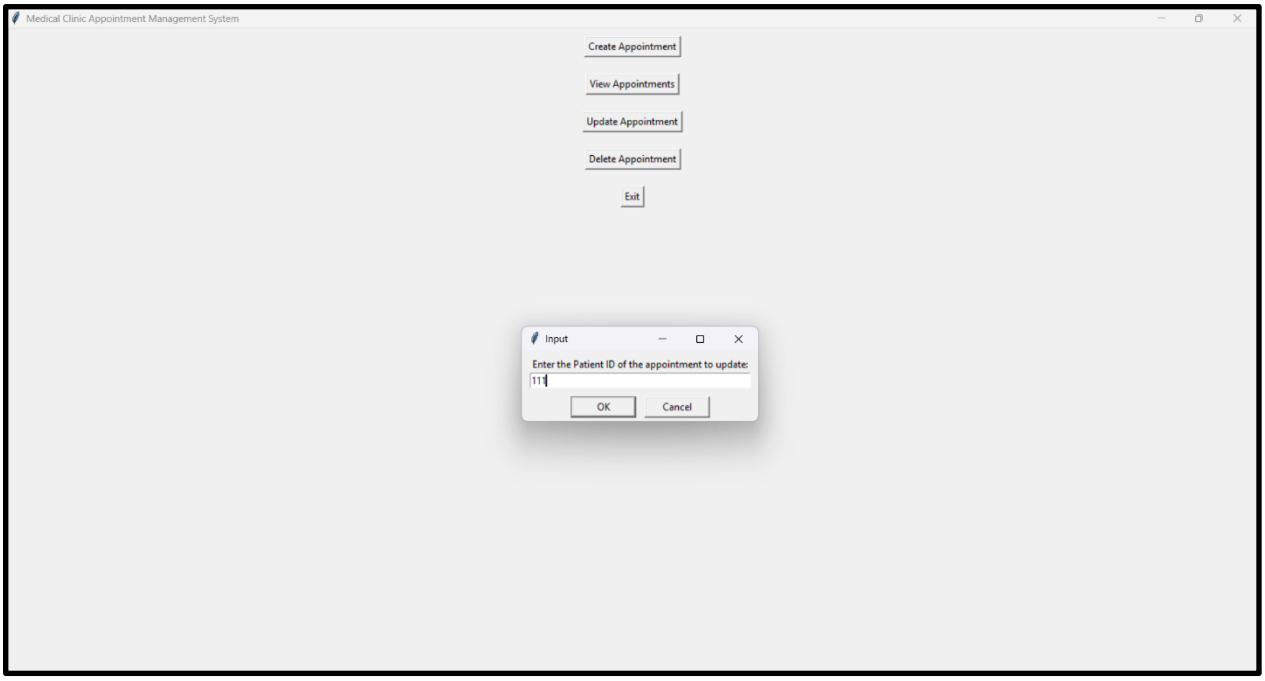


Figure 34: Result of Update Appointment Booking

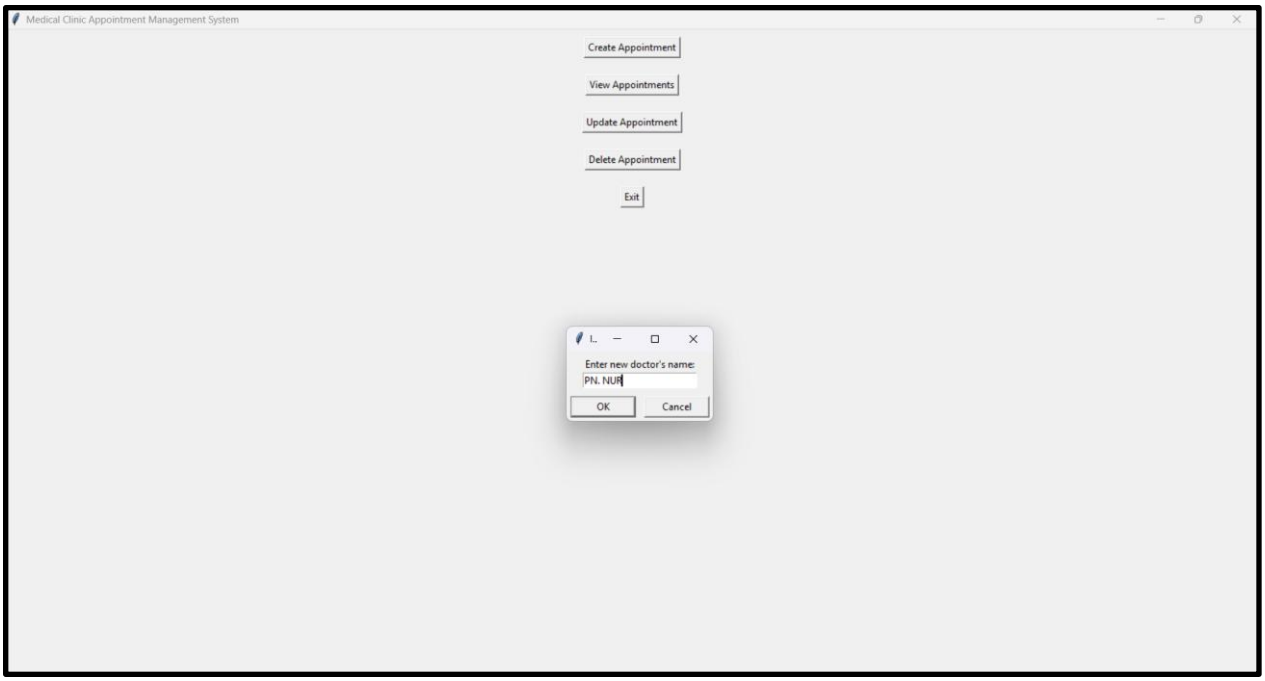


Figure 35: Result of Update Appointment Booking

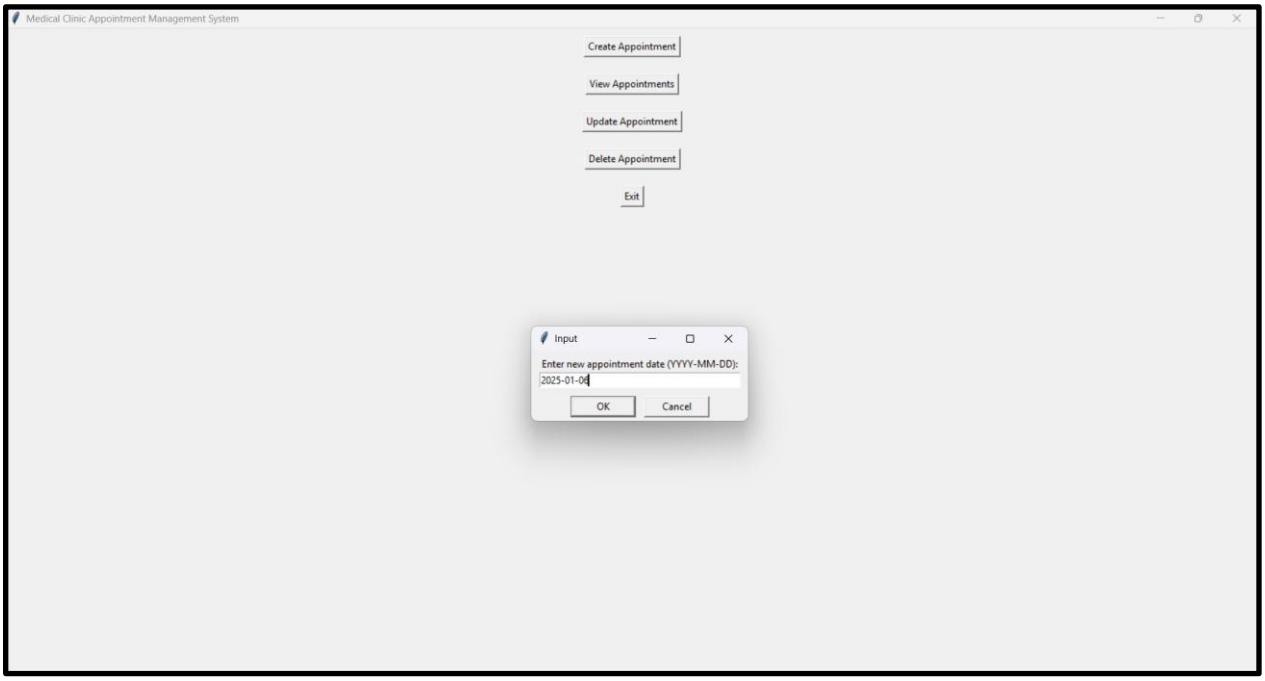


Figure 36: Result of Update Appointment Booking

Figure 37: Result of Update Appointment Booking

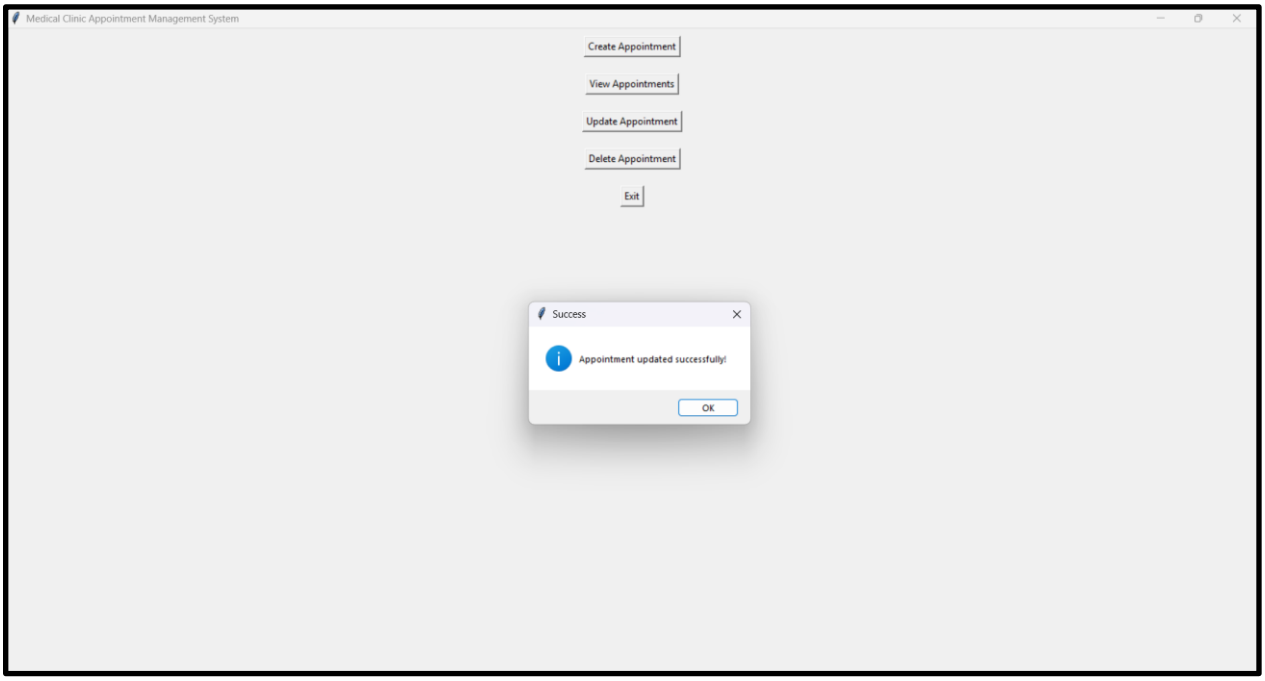


Figure 38: Result of Update Appointment Booking

- **DELETE APPOINTMENT BOOKING**

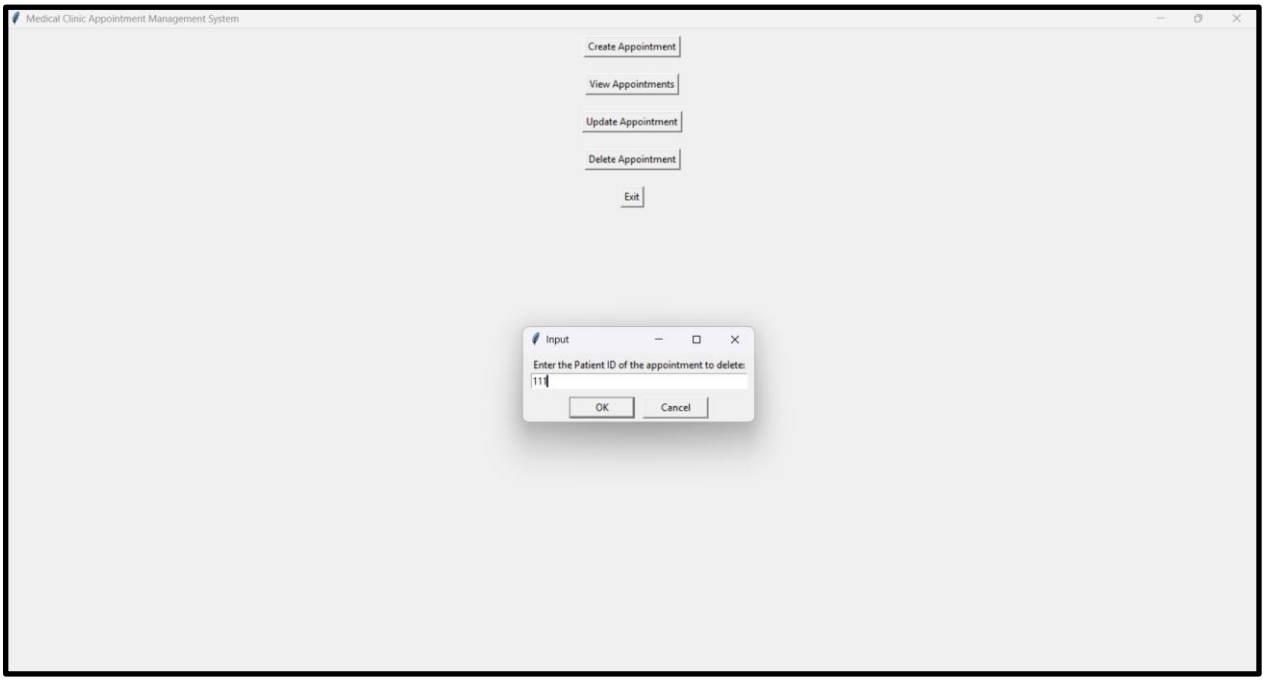


Figure 39: Result of Delete Appointment Booking

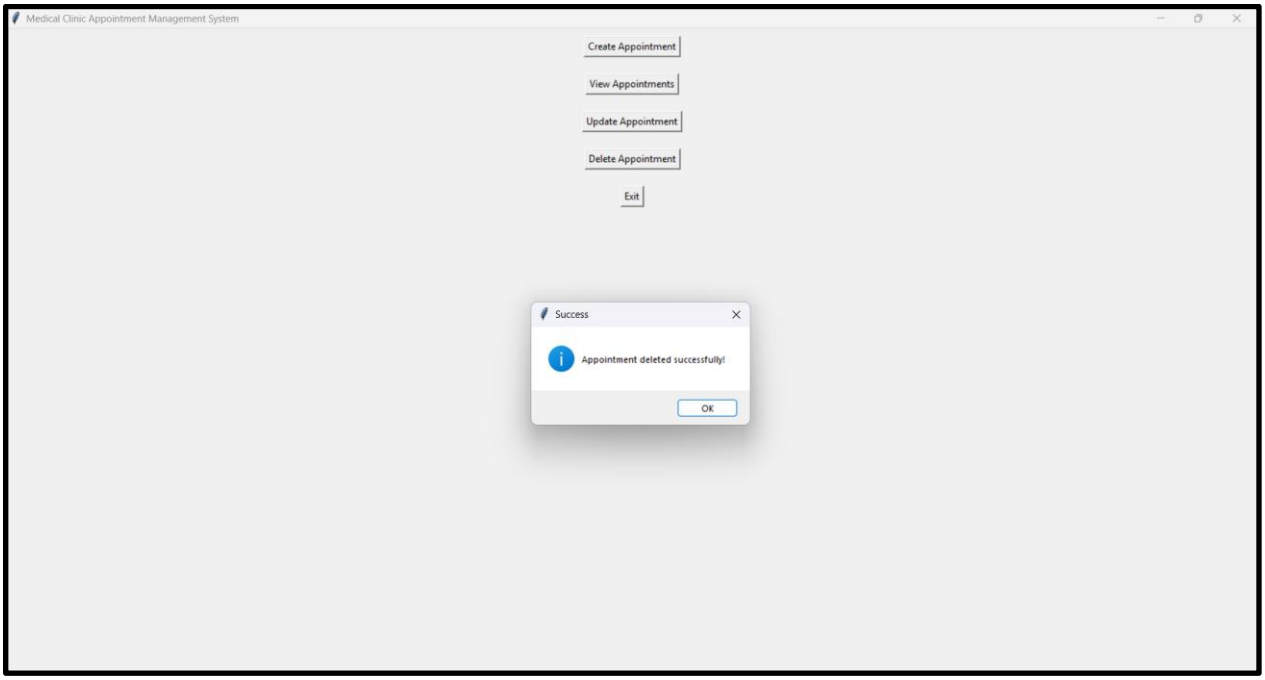


Figure 40: Result of Delete Appointment Booking

PROMPT DATA: LIST ALL

1) **APPOINTMENT BOOKING**
   i. Patient ID
   ii. Patient Name
   iii. Doctor Name
   iv. Appointment Date
   v. Appointment Time

2) **PATIENT**
   i. Patient Name
   ii. Patient ID
   iii. Patient Gender
   iv. Patient Age
   v. Patient Phone

3) **NURSE**
   i. Nurse Name
   ii. Nurse ID
   iii. Nurse Gender
   iv. Nurse Department
   v. Nurse Phone
   vi. Nurse Shift Time

4) **DOCTOR**
   i. Doctor Name
   ii. Doctor ID
   iii. Doctor Gender
   iv. Doctor Specialty
   v. Doctor Phone
   vi. Doctor Working Hour

5) **MEDICINE**

i. Medicine Name

ii. Medicine Type

iii. Medicine Quantity

iv. Medicine Expiry Date

v. Medicine Price

FUNCTION

i). CREATE DATA

- **BOOKING**

```
def appointment_booking():
    appointment_list = []

    def create_appointment():
        patient_id = input("Enter patient ID:")
        patient_name = input("Enter patient's name:")
        doctor_name = input("Enter doctor's name:")
        appointment_date = input("Enter appointment date (YYYY-MM-DD):")
        appointment_time = input("Enter appointment time (HH:MM):")

        appointment_details = {
            "Patient ID": patient_id,
            "Patient Name": patient_name,
            "Doctor Name": doctor_name,
            "Appointment Date": appointment_date,
            "Appointment Time": appointment_time
        }
        appointment_list.append(appointment_details)
        print("Appointment booked successfully!")
```

Figure 41: Create Data of Appointment Booking

- **PATIENT RECORD**

```python
def patient_management():
    patient_list = []

    def create_patient():
        patient_name = input("Enter patient name:")
        patient_id = input("Enter patient ID:")
        patient_gender = input("Enter patient gender (Male / Female):")
        patient_age = input("Enter patient age:")
        patient_phone = input("Enter patient phone:")

        patient = {
            "Name": patient_name,
            "ID": patient_id,
            "Gender": patient_gender,
            "Age": patient_age,
            "Phone": patient_phone
        }
        patient_list.append(patient)
        print("Patient added successfully!")
```

Figure 42: Create Data of Patient Record

- **NURSE RECORD**

```python
def nurse_management():
    nurse_list = []

    def create_nurse():
        nurse_name = input("Enter nurse name:")
        nurse_id = input("Enter nurse ID:")
        nurse_gender = input("Enter nurse gender (Male / Female):")
        nurse_department = input("Enter nurse department:")
        nurse_phone = input("Enter nurse phone:")
        nurse_shift_time = input("Enter nurse shift time:")

        nurse = {
            "Name": nurse_name,
            "ID": nurse_id,
            "Gender": nurse_gender,
            "Department": nurse_department,
            "Phone": nurse_phone,
            "Shift Time": nurse_shift_time
        }
        nurse_list.append(nurse)
        print("Nurse added successfully!")
```

Figure 43: Create Data of Nurse Record

- **DOCTOR RECORD**

```python
def doctor_management():
    doctor_list = []

    def create_doctor():
        doctor_name = input("Enter doctor name:")
        doctor_id = input("Enter doctor ID:")
        doctor_gender = input("Enter doctor gender (Male / Female):")
        doctor_specialty = input("Enter doctor specialty:")
        doctor_phone = input("Enter doctor phone:")
        doctor_working_hour = input("Enter doctor working hour:")

        doctor = {
            "Name": doctor_name,
            "ID": doctor_id,
            "Gender": doctor_gender,
            "Specialty": doctor_specialty,
            "Phone": doctor_phone,
            "Working Hour": doctor_working_hour
        }
        doctor_list.append(doctor)
        print("Doctor added successfully!")
```

Figure 44: Create Data of Doctor Record

- **MEDICINE**

```python
def medicine_management():
    medicine_list = []

    def create_medicine():
        medicine_name = input("Enter medicine name:")
        medicine_type = input("Enter medicine type (Tablet / Syrup / Ointment):")
        medicine_quantity = input("Enter medicine quantity:")
        medicine_expiry_date = input("Enter medicine expiry date:")
        medicine_price = input("Enter medicine price:")

        medicine = {
            "Name": medicine_name,
            "Type": medicine_type,
            "Quantity": medicine_quantity,
            "Expiry Date": medicine_expiry_date,
            "Price": medicine_price
        }
        medicine_list.append(medicine)
        print("Medicine added successfully!")
```

Figure 45: Create Data of Medicine Record

ii). READ DATA

- **READ APPOINTMENT BOOKING**

```python

def view_appointments():
    if not appointment_list:
        print("No appointments found!")
    else:
        for appointment in appointment_list:
            print(appointment)

```

Figure 46: Read Data of Appointment Booking

- **READ PATIENT**

```python

def view_patients():
    if not patient_list:
        print("No patients found!")
    else:
        for patient in patient_list:
            print(patient)

```

Figure 47: Read Data of Patient Record

- **READ NURSE**

```python

def view_nurses():
    if not nurse_list:
        print("No nurses found!")
    else:
        for nurse in nurse_list:
            print(nurse)

```

Figure 48: Read Data of Nurse Record

- **READ DOCTOR**

```python

def view_doctors():
    if not doctor_list:
        print("No doctors found!")
    else:
        for doctor in doctor_list:
            print(doctor)

```

Figure 49: Read Data of Doctor Record

- **READ MEDICINE**

```python

def view_medicines():
    if not medicine_list:
        print("No medicines found!")
    else:
        for medicine in medicine_list:
            print(medicine)

```

Figure 50: Read Data of Medicine Record

iii). UPDATE DATA

- **UPDATE APPOINTMENT BOOKING**

```python

def update_appointment():
    patient_id = input("Enter the Patient ID of the appointment to update:")
    for appointment in appointment_list:
        if appointment["Patient ID"] == patient_id:
            print("Existing details:", appointment)
            appointment["Doctor Name"] = input("Enter new doctor's name:")
            appointment["Appointment Date"] = input("Enter new appointment date (YYYY-MM-DD):")
            appointment["Appointment Time"] = input("Enter new appointment time (HH:MM):")
            print("Appointment updated successfully!")
            return
    print("Appointment not found!")

```

Figure 51: Update Data of Appointment Booking

- **UPDATE PATIENT BOOKING**

```python

def update_patient():
    patient_id = input("Enter the Patient ID to update:")
    for patient in patient_list:
        if patient["ID"] == patient_id:
            print("Existing details:", patient)
            patient["Name"] = input("Enter new patient name:")
            patient["Gender"] = input("Enter new gender (Male / Female):")
            patient["Age"] = input("Enter new age:")
            patient["Phone"] = input("Enter new phone:")
            print("Patient updated successfully!")
            return
    print("Patient not found!")

```

Figure 52: Update Data of Patient Record

- **UPDATE NURSE**

```python

def update_nurse():
    nurse_id = input("Enter the Nurse ID to update:")
    for nurse in nurse_list:
        if nurse["ID"] == nurse_id:
            print("Existing details:", nurse)
            nurse["Name"] = input("Enter new nurse name:")
            nurse["Gender"] = input("Enter new gender (Male / Female):")
            nurse["Department"] = input("Enter new department:")
            nurse["Phone"] = input("Enter new phone:")
            nurse["Shift Time"] = input("Enter new shift time:")
            print("Nurse updated successfully!")
            return
    print("Nurse not found!")

```

Figure 53: Update Data of Nurse Record

- **UPDATE DOCTOR**

```python

def update_doctor():
    doctor_id = input("Enter the Doctor ID to update:")
    for doctor in doctor_list:
        if doctor["ID"] == doctor_id:
            print("Existing details:", doctor)
            doctor["Name"] = input("Enter new doctor name:")
            doctor["Gender"] = input("Enter new gender (Male / Female):")
            doctor["Specialty"] = input("Enter new specialty:")
            doctor["Phone"] = input("Enter new phone:")
            doctor["Working Hour"] = input("Enter new working hour:")
            print("Doctor updated successfully!")
            return
    print("Doctor not found!")

```

Figure 54: Update Data of Doctor Record

- **UPDATE MEDICINE**

```python

def update_medicine():
    medicine_name = input("Enter the Medicine Name to update:")
    for medicine in medicine_list:
        if medicine["Name"] == medicine_name:
            print("Existing details:", medicine)
            medicine["Type"] = input("Enter new type (Tablet / Syrup / Ointment):")
            medicine["Quantity"] = input("Enter new quantity:")
            medicine["Expiry Date"] = input("Enter new expiry date:")
            medicine["Price"] = input ("Enter new price: ")
            print("Medicine updated successfully!")
            return
    print("Medicine not found!")

```

Figure 55: Read Data of Medicine Record

iv). EXISTING DATA

- **APPOINTMENT BOOKING**

```python
def delete_appointment():
    patient_id = input("Enter the Patient ID of the appointment to delete:")
    for appointment in appointment_list:
        if appointment["Patient ID"] == patient_id:
            appointment_list.remove(appointment)
            print("Appointment deleted successfully!")
            return
    print("Appointment not found!")
```

Figure 56: Delete Data of Appointment Booking

- **PATIENT**

```python
def delete_patient():
    patient_id = input("Enter the Patient ID to delete:")
    for patient in patient_list:
        if patient["ID"] == patient_id:
            patient_list.remove(patient)
            print("Patient deleted successfully!")
            return
    print("Patient not found!")
```

Figure 57: Delete Data of Patient Record

- **NURSE**

```python
def delete_nurse():
    nurse_id = input("Enter the Nurse ID to delete:")
    for nurse in nurse_list:
        if nurse["ID"] == nurse_id:
            nurse_list.remove(nurse)
            print("Nurse deleted successfully!")
            return
    print("Nurse not found!")
```

Figure 58: Delete Data of Nurse Record

- **DOCTOR**

```python
def delete_doctor():
    doctor_id = input("Enter the Doctor ID to delete:")
    for doctor in doctor_list:
        if doctor["ID"] == doctor_id:
            doctor_list.remove(doctor)
            print("Doctor deleted successfully!")
            return
    print("Doctor not found!")
```

Figure 59: Delete Data of Doctor Record

- **MEDICINE**

```python
def delete_medicine():
    medicine_name = input("Enter the Medicine Name to delete: ")
    for medicine in medicine_list:
        if medicine["Name"] == medicine_name:
            medicine_list.remove(medicine)
            print("Medicine deleted successfully!")
            return
    print("Medicine not found!")
```

Figure 60: Delete Data of Medicine Record

GUI PYTHON

- **APPOINTMENT BOOKING**

```python
# Appointment Management System
class AppointmentManagementSystem:
    def __init__(self, root):
        self.root = root
        self.root.title("Medical Clinic Appointment Management System")
        self.appointment_list = []

        # Buttons for CRUD operations
        tk.Button(root, text="Create Appointment", command=self.create_appointment).pack(pady=10)
        tk.Button(root, text="View Appointments", command=self.view_appointments).pack(pady=10)
        tk.Button(root, text="Update Appointment", command=self.update_appointment).pack(pady=10)
        tk.Button(root, text="Delete Appointment", command=self.delete_appointment).pack(pady=10)
        tk.Button(root, text="Exit", command=root.quit).pack(pady=10)

    def create_appointment(self):
        patient_id = simpledialog.askstring("Input", "Enter patient ID:")
        patient_name = simpledialog.askstring("Input", "Enter patient's name:")
        doctor_name = simpledialog.askstring("Input", "Enter doctor's name:")
        appointment_date = simpledialog.askstring("Input", "Enter appointment date (YYYY-MM-DD):")
        appointment_time = simpledialog.askstring("Input", "Enter appointment time (HH:MM):")

        if patient_id and patient_name and doctor_name and appointment_date and appointment_time:
            appointment_details = {
                "Patient ID": patient_id,
                "Patient Name": patient_name,
                "Doctor Name": doctor_name,
                "Appointment Date": appointment_date,
                "Appointment Time": appointment_time
            }
            self.appointment_list.append(appointment_details)
            messagebox.showinfo("Success", "Appointment booked successfully!")
        else:
            messagebox.showwarning("Warning", "All fields are required!")
```

Figure 61: GUI Python for Appointment Booking

```python
def view_appointments(self):
    if not self.appointment_list:
        messagebox.showinfo("Info", "No appointments found!")
    else:
        appointments = "\n".join(
            [f"ID: {a['Patient ID']}, Name: {a['Patient Name']}, Doctor: {a['Doctor Name']}, "
             f"Date: {a['Appointment Date']}, Time: {a['Appointment Time']}"
             for a in self.appointment_list]
        )
        messagebox.showinfo("Appointments", appointments)
```

Figure 62: GUI Python for Appointment Booking

```python
def update_appointment(self):
    patient_id = simpledialog.askstring("Input", "Enter the Patient ID of the appointment to update:")
    for appointment in self.appointment_list:
        if appointment["Patient ID"] == patient_id:
            new_doctor = simpledialog.askstring("Input", "Enter new doctor's name:")
            new_date = simpledialog.askstring("Input", "Enter new appointment date (YYYY-MM-DD):")
            new_time = simpledialog.askstring("Input", "Enter new appointment time (HH:MM):")
            if new_doctor and new_date and new_time:
                appointment["Doctor Name"] = new_doctor
                appointment["Appointment Date"] = new_date
                appointment["Appointment Time"] = new_time
                messagebox.showinfo("Success", "Appointment updated successfully!")
            else:
                messagebox.showwarning("Warning", "All fields are required!")
            return
    messagebox.showerror("Error", "Appointment not found!")
```

Figure 63: GUI Python for Appointment Booking

```python
def delete_appointment(self):
    patient_id = simpledialog.askstring("Input", "Enter the Patient ID of the appointment to delete:")
    for appointment in self.appointment_list:
        if appointment["Patient ID"] == patient_id:
            self.appointment_list.remove(appointment)
            messagebox.showinfo("Success", "Appointment deleted successfully!")
            return
    messagebox.showerror("Error", "Appointment not found!")
```

Figure 64: GUI Python for Appointment Booking

```python
if __name__ == "__main__":
    root = tk.Tk()
    app = AppointmentManagementSystem(root)
    root.mainloop()
```

Figure 65: GUI Python for Appointment Booking

- **PATIENT**

```python
# Patient Record Management System
class PatientManagementSystem:
    def __init__(self, root):
        self.root = root
        self.root.title("Patient Record Management System")
        self.patient_list = []

        # Buttons for CRUD operations
        tk.Button(root, text="Add Patient", command=self.create_patient).pack(pady=10)
        tk.Button(root, text="View Patients", command=self.view_patients).pack(pady=10)
        tk.Button(root, text="Update Patient", command=self.update_patient).pack(pady=10)
        tk.Button(root, text="Delete Patient", command=self.delete_patient).pack(pady=10)
        tk.Button(root, text="Exit", command=root.quit).pack(pady=10)
```

Figure 66: GUI Python for Patient Record

```python
def create_patient(self):
    name = simpledialog.askstring("Input", "Enter patient name:")
    patient_id = simpledialog.askstring("Input", "Enter patient ID:")
    gender = simpledialog.askstring("Input", "Enter patient gender (Male / Female):")
    age = simpledialog.askstring("Input", "Enter patient age:")
    phone = simpledialog.askstring("Input", "Enter patient phone:")

    if name and patient_id and gender and age and phone:
        patient = {
            "Name": name,
            "ID": patient_id,
            "Gender": gender,
            "Age": age,
            "Phone": phone
        }
        self.patient_list.append(patient)
        messagebox.showinfo("Success", "Patient added successfully!")
    else:
        messagebox.showwarning("Warning", "All fields are required!")
```

Figure 67: GUI Python for Patient Record

```python
def view_patients(self):
    if not self.patient_list:
        messagebox.showinfo("Info", "No patients found!")
    else:
        patients = "\n".join(
            [f"Name: {p['Name']}, ID: {p['ID']}, Gender: {p['Gender']}, Age: {p['Age']}, Phone: {p['Phone']}"
             for p in self.patient_list]
        )
        messagebox.showinfo("Patients", patients)
```

Figure 68: GUI Python for Patient Record

```python
def update_patient(self):
    patient_id = simpledialog.askstring("Input", "Enter the Patient ID to update:")
    for patient in self.patient_list:
        if patient["ID"] == patient_id:
            new_name = simpledialog.askstring("Input", "Enter new patient name:")
            new_gender = simpledialog.askstring("Input", "Enter new gender (Male / Female):")
            new_age = simpledialog.askstring("Input", "Enter new age:")
            new_phone = simpledialog.askstring("Input", "Enter new phone:")

            if new_name and new_gender and new_age and new_phone:
                patient["Name"] = new_name
                patient["Gender"] = new_gender
                patient["Age"] = new_age
                patient["Phone"] = new_phone
                messagebox.showinfo("Success", "Patient updated successfully!")
            else:
                messagebox.showwarning("Warning", "All fields are required!")
            return
    messagebox.showerror("Error", "Patient not found!")
```

Figure 69: GUI Python for Patient Record

```python
def delete_patient(self):
    patient_id = simpledialog.askstring("Input", "Enter the Patient ID to delete:")
    for patient in self.patient_list:
        if patient["ID"] == patient_id:
            self.patient_list.remove(patient)
            messagebox.showinfo("Success", "Patient deleted successfully!")
            return
    messagebox.showerror("Error", "Patient not found!")
```

Figure 70: GUI Python for Patient Record

```python
if __name__ == "__main__":
    root = tk.Tk()
    app = PatientManagementSystem(root)
    root.mainloop()
```

Figure 71: GUI Python for Patient Record

- **NURSE**

```python
class NurseRecordManagement:
    def __init__(self, master):
        self.master = master
        self.master.title("Nurse Record Management")

        self.nurse_list = []

        self.create_widgets()
```

Figure 72: GUI Python for Nurse Record

```python
def create_widgets(self):
    self.add_button = tk.Button(self.master, text="Add Nurse", command=self.add_nurse)
    self.add_button.pack(pady=10)

    self.view_button = tk.Button(self.master, text="View Nurses", command=self.view_nurses)
    self.view_button.pack(pady=10)

    self.update_button = tk.Button(self.master, text="Update Nurse", command=self.update_nurse)
    self.update_button.pack(pady=10)

    self.delete_button = tk.Button(self.master, text="Delete Nurse", command=self.delete_nurse)
    self.delete_button.pack(pady=10)

    self.exit_button = tk.Button(self.master, text="Exit", command=self.master.quit)
    self.exit_button.pack(pady=10)
```

Figure 73: GUI Python for Nurse Record

```python
def add_nurse(self):
    nurse_name = simpledialog.askstring("Input", "Enter nurse name:")
    nurse_id = simpledialog.askstring("Input", "Enter nurse ID:")
    nurse_gender = simpledialog.askstring("Input", "Enter nurse gender (Male / Female):")
    nurse_department = simpledialog.askstring("Input", "Enter nurse department:")
    nurse_phone = simpledialog.askstring("Input", "Enter nurse phone:")
    nurse_shift_time = simpledialog.askstring("Input", "Enter nurse shift time:")

    nurse = {
        "Name": nurse_name,
        "ID": nurse_id,
        "Gender": nurse_gender,
        "Department": nurse_department,
        "Phone": nurse_phone,
        "Shift Time": nurse_shift_time
    }
    self.nurse_list.append(nurse)
    messagebox.showinfo("Success", "Nurse added successfully!")
```

Figure 74: GUI Python for Nurse Record

```python
def view_nurses(self):
    if not self.nurse_list:
        messagebox.showwarning("Warning", "No nurses found!")
    else:
        nurses_info = "\n".join([str(nurse) for nurse in self.nurse_list])
        messagebox.showinfo("Nurses List", nurses_info)
```

Figure 75: GUI Python for Nurse Record

```python
def update_nurse(self):
    nurse_id = simpledialog.askstring("Input", "Enter the Nurse ID to update:")
    for nurse in self.nurse_list:
        if nurse["ID"] == nurse_id:
            nurse["Name"] = simpledialog.askstring("Input", "Enter new nurse name:", initialvalue=nurse["Name"])
            nurse["Gender"] = simpledialog.askstring("Input", "Enter new gender (Male / Female):", initialvalue=nurse["Gender"])
            nurse["Department"] = simpledialog.askstring("Input", "Enter new department:", initialvalue=nurse["Department"])
            nurse["Phone"] = simpledialog.askstring("Input", "Enter new phone:", initialvalue=nurse["Phone"])
            nurse["Shift Time"] = simpledialog.askstring("Input", "Enter new shift time:", initialvalue=nurse["Shift Time"])
            messagebox.showinfo("Success", "Nurse updated successfully!")
            return
    messagebox.showwarning("Warning", "Nurse not found!")
```

Figure 76: GUI Python for Nurse Record

```python
def delete_nurse(self):
    nurse_id = simpledialog.askstring("Input", "Enter the Nurse ID to delete:")
    for nurse in self.nurse_list:
        if nurse["ID"] == nurse_id:
            self.nurse_list.remove(nurse)
            messagebox.showinfo("Success", "Nurse deleted successfully!")
            return
    messagebox.showwarning("Warning", "Nurse not found!")
```

Figure 77: GUI Python for Nurse Record

```python
if __name__ == "__main__":
    root = tk.Tk()
    app = NurseRecordManagement(root)
    root.mainloop()
```

Figure 78: GUI Python for Nurse Record

- **DOCTOR**

```python
class DoctorRecordManagement:
    def __init__(self, master):
        self.master = master
        self.master.title("Doctor Record Management")

        self.doctor_list = []

        self.create_widgets()
```

Figure 79: GUI Python for Doctor Record

```python
def create_widgets(self):
    self.add_button = tk.Button(self.master, text="Add Doctor", command=self.add_doctor)
    self.add_button.pack(pady=10)

    self.view_button = tk.Button(self.master, text="View Doctors", command=self.view_doctors)
    self.view_button.pack(pady=10)

    self.update_button = tk.Button(self.master, text="Update Doctor", command=self.update_doctor)
    self.update_button.pack(pady=10)

    self.delete_button = tk.Button(self.master, text="Delete Doctor", command=self.delete_doctor)
    self.delete_button.pack(pady=10)

    self.exit_button = tk.Button(self.master, text="Exit", command=self.master.quit)
    self.exit_button.pack(pady=10)
```

Figure 80: GUI Python for Doctor Record

```python
def view_doctors(self):
    if not self.doctor_list:
        messagebox.showwarning("Warning", "No doctors found!")
    else:
        doctors_info = "\n".join([str(doctor) for doctor in self.doctor_list])
        messagebox.showinfo("Doctors List", doctors_info)
```

Figure 81: GUI Python for Doctor Record

```python
def update_doctor(self):
    doctor_id = simpledialog.askstring("Input", "Enter the Doctor ID to update:")
    for doctor in self.doctor_list:
        if doctor["ID"] == doctor_id:
            doctor["Name"] = simpledialog.askstring("Input", "Enter new doctor name:", initialvalue=doctor["Name"])
            doctor["Gender"] = simpledialog.askstring("Input", "Enter new gender (Male / Female):", initialvalue=doctor["Gender"])
            doctor["Specialty"] = simpledialog.askstring("Input", "Enter new specialty:", initialvalue=doctor["Specialty"])
            doctor["Phone"] = simpledialog.askstring("Input", "Enter new phone:", initialvalue=doctor["Phone"])
            doctor["Working Hour"] = simpledialog.askstring("Input", "Enter new working hour:", initialvalue=doctor["Working Hour"])
            messagebox.showinfo("Success", "Doctor updated successfully!")
            return
    messagebox.showwarning("Warning", "Doctor not found!")
```

Figure 82: GUI Python for Doctor Record

```python
def delete_doctor(self):
    doctor_id = simpledialog.askstring("Input", "Enter the Doctor ID to delete:")
    for doctor in self.doctor_list:
        if doctor["ID"] == doctor_id:
            self.doctor_list.remove(doctor)
            messagebox.showinfo("Success", "Doctor deleted successfully!")
            return
    messagebox.showwarning("Warning", "Doctor not found!")
```

Figure 83: GUI Python for Doctor Record

```python
if __name__ == "__main__":
    root = tk.Tk()
    app = DoctorRecordManagement(root)
    root.mainloop()
```

Figure 84: GUI Python for Doctor Record

- **MEDICINE**

```python
class MedicineRecordManagement:
    def __init__(self, master):
        self.master = master
        self.master.title("Medicine Record Management")

        self.medicine_list = []

        self.create_widgets()
```

Figure 85: GUI Python for Medicine Record

```python
def create_widgets(self):
    self.add_button = tk.Button(self.master, text="Add Medicine", command=self.add_medicine)
    self.add_button.pack(pady=10)

    self.view_button = tk.Button(self.master, text="View Medicines", command=self.view_medicines)
    self.view_button.pack(pady=10)

    self.update_button = tk.Button(self.master, text="Update Medicine", command=self.update_medicine)
    self.update_button.pack(pady=10)

    self.delete_button = tk.Button(self.master, text="Delete Medicine", command=self.delete_medicine)
    self.delete_button.pack(pady=10)

    self.exit_button = tk.Button(self.master, text="Exit", command=self.master.quit)
    self.exit_button.pack(pady=10)
```

Figure 86: GUI Python for Medicine Record

```python
def add_medicine(self):
    medicine_name = simpledialog.askstring("Input", "Enter medicine name:")
    medicine_type = simpledialog.askstring("Input", "Enter medicine type (Tablet / Syrup / Ointment):")
    medicine_quantity = simpledialog.askstring("Input", "Enter medicine quantity:")
    medicine_expiry_date = simpledialog.askstring("Input", "Enter medicine expiry date:")
    medicine_price = simpledialog.askstring("Input", "Enter medicine price:")

    medicine = {
        "Name": medicine_name,
        "Type": medicine_type,
        "Quantity": medicine_quantity,
        "Expiry Date": medicine_expiry_date,
        "Price": medicine_price
    }
    self.medicine_list.append(medicine)
    messagebox.showinfo("Success", "Medicine added successfully!")
```

Figure 87: GUI Python for Medicine Record

```python
def view_medicines(self):
    if not self.medicine_list:
        messagebox.showwarning("Warning", "No medicines found!")
    else:
        medicines_info = "\n".join([str(medicine) for medicine in self.medicine_list])
        messagebox.showinfo("Medicines List", medicines_info)
```

Figure 88: GUI Python for Medicine Record

```python
def update_medicine(self):
    medicine_name = simpledialog.askstring("Input", "Enter the Medicine Name to update:")
    for medicine in self.medicine_list:
        if medicine["Name"] == medicine_name:
            medicine["Type"] = simpledialog.askstring("Input", "Enter new type (Tablet / Syrup / Ointment):", initialvalue=medicine["Type"])
            medicine["Quantity"] = simpledialog.askstring("Input", "Enter new quantity:", initialvalue=medicine["Quantity"])
            medicine["Expiry Date"] = simpledialog.askstring("Input", "Enter new expiry date:", initialvalue=medicine["Expiry Date"])
            medicine["Price"] = simpledialog.askstring("Input", "Enter new price:", initialvalue=medicine["Price"])
            messagebox.showinfo("Success", "Medicine updated successfully!")
            return
    messagebox.showwarning("Warning", "Medicine not found!")
```

Figure 89: GUI Python for Medicine Record

```python
def delete_medicine(self):
    medicine_name = simpledialog.askstring("Input", "Enter the Medicine Name to delete:")
    for medicine in self.medicine_list:
        if medicine["Name"] == medicine_name:
            self.medicine_list.remove(medicine)
            messagebox.showinfo("Success", "Medicine deleted successfully!")
            return
    messagebox.showwarning("Warning", "Medicine not found!")
```

Figure 90: GUI Python for Medicine Record

```python
if __name__ == "__main__":
    root = tk.Tk()
    app = MedicineRecordManagement(root)
    root.mainloop()
```

Figure 91: GUI Python for Medicine Record

CONDITIONAL STATEMENT: YES