

**PYTHON PROJECT**

**TOPIC- INFIRMARY MANAGEMENT SYSTEM**

**Submitted to- Deepak Kumar Sir**

**Name: Prachi Agarwalla**

**Roll no.: R2142232053**

**Sap Id: 500126504**

**Batch-41**

**All About Domain :-**

Domain name – Infirmary Management System

Domain Details:

It is a comprehensive program designed to simplify healthcare administration and optimize health care within medical facilities. This program presents a menu prompting a user to choose between admin mode and user mode. In admin mode, the user is prompted to enter a password, and upon successful authentication, they gain access to functionalities such as managing patients, doctors, and medicines. For example, in the patient management section, the admin can add, edit, display, or delete patient records stored in a text file. Similarly, functionalities for managing doctors and medicines are also provided. The user mode, on the other hand, allows users to log in or sign up. After logging in, the user can see his/her profile or can order medicines. Overall, this system serves as a rudimentary framework for managing various aspects of an infirmary, including patient records, doctor information, and medicine inventory.

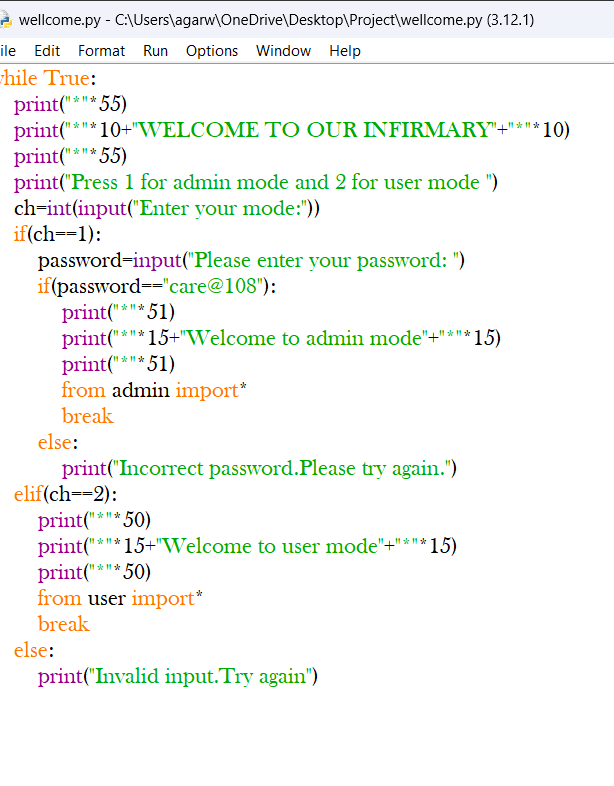
Top of Form

**Reason:**

Managing a medical facility involves handling various aspects, including patient records, doctor schedules, medicine inventory and billing. Designing an IMS project provides an opportunity to tackle complex problems and develop robust solutions.

**OVERVIEW:**

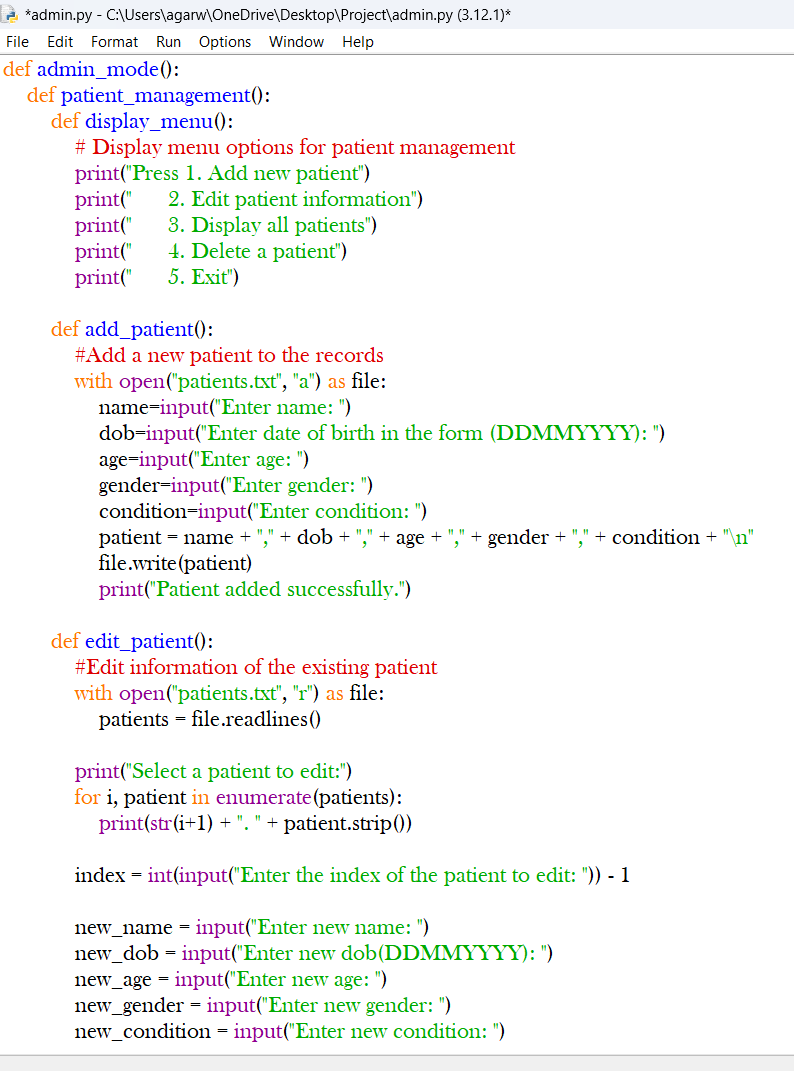
The program is structured into two main modes: admin mode and user mode.

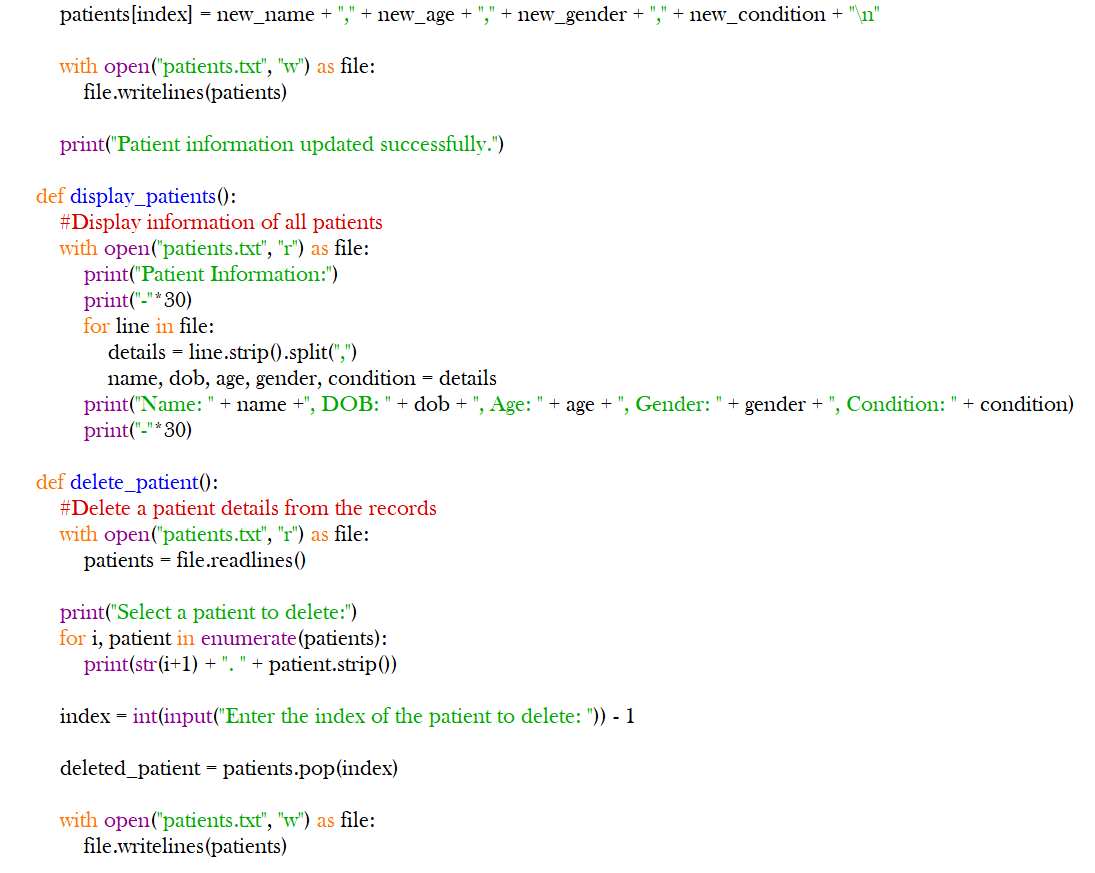


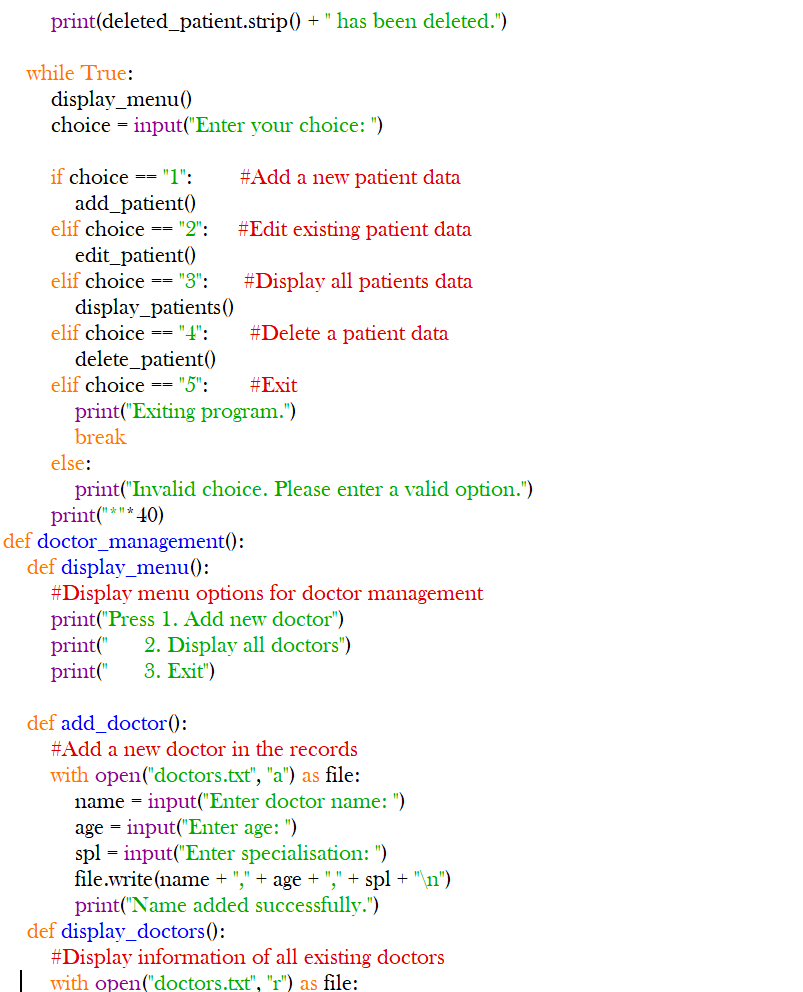
**I. ADMIN MODE:**

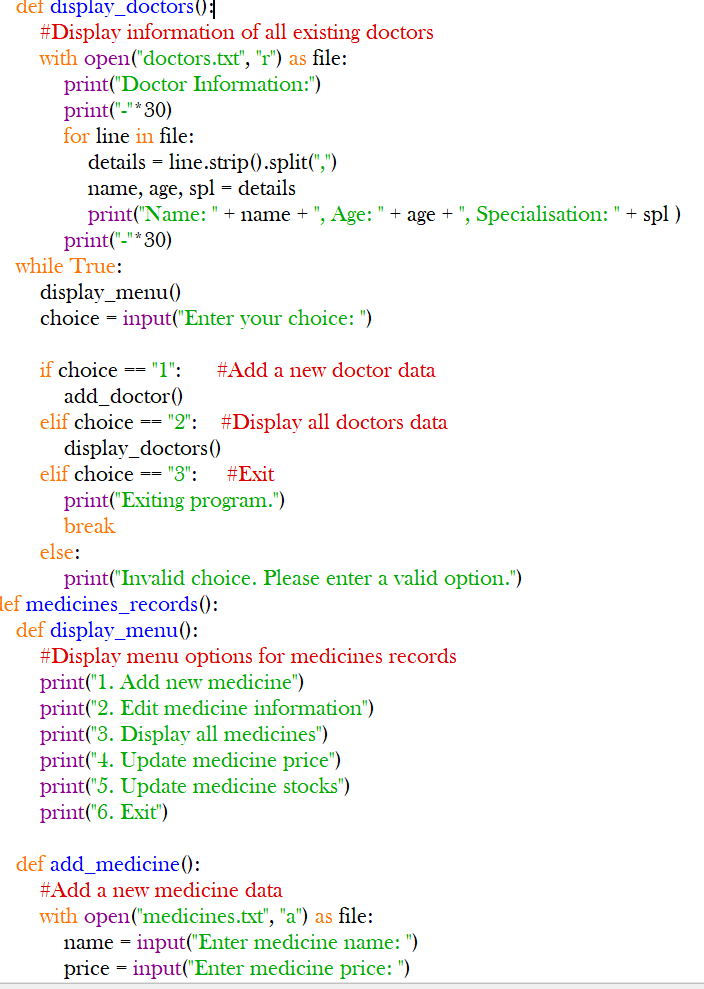
In the admin mode, administrators can perform the following tasks:

* Patient Management: Add new patients, edit patient information, display all patients, and delete patients.
* Doctor Management: Add new doctors and display all doctors.
* Medicines Records: Add new medicines, edit medicine information, display all medicines, update medicine price, and update medicine stocks.

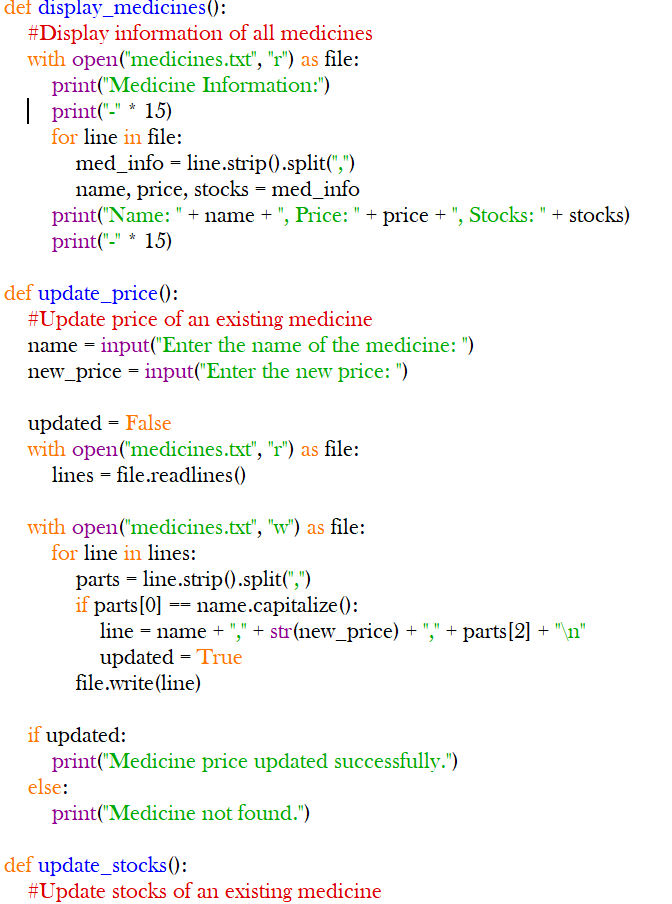


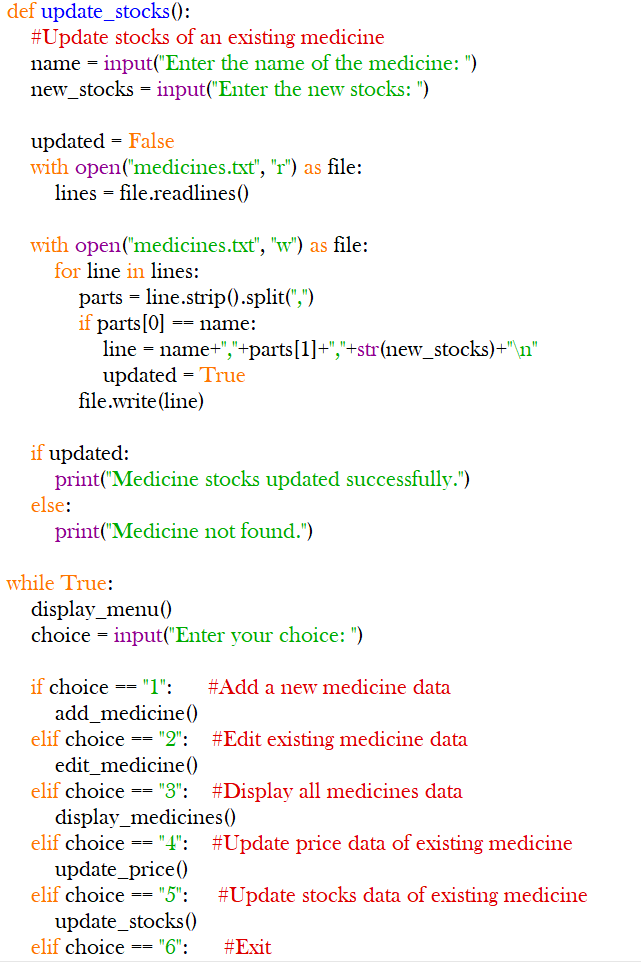


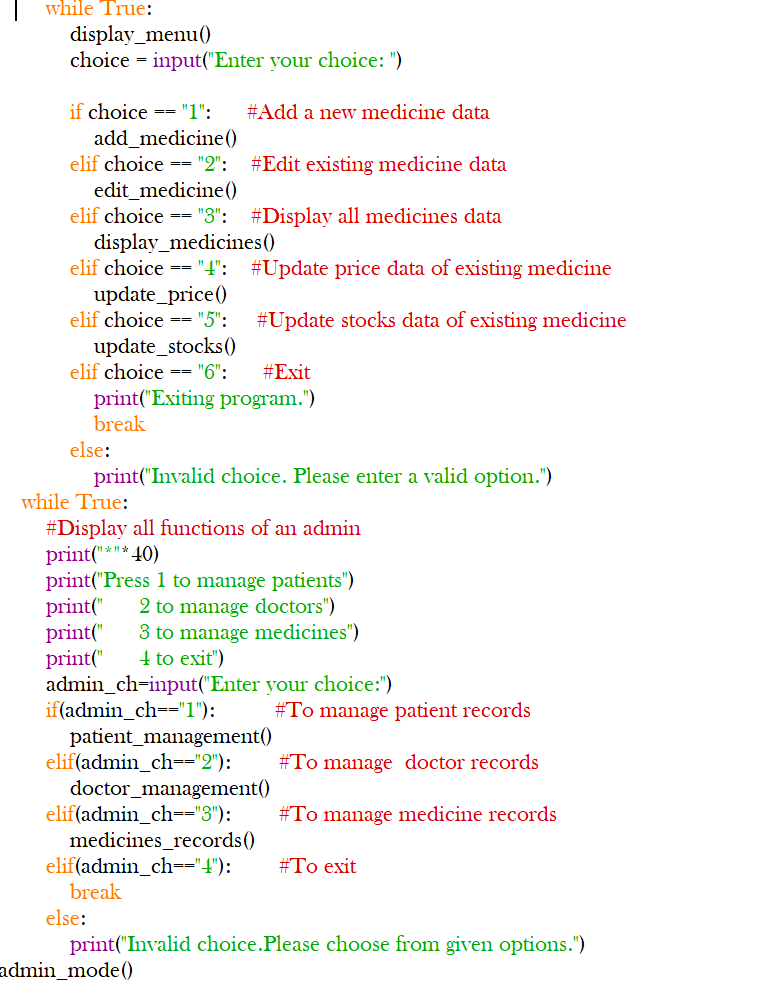








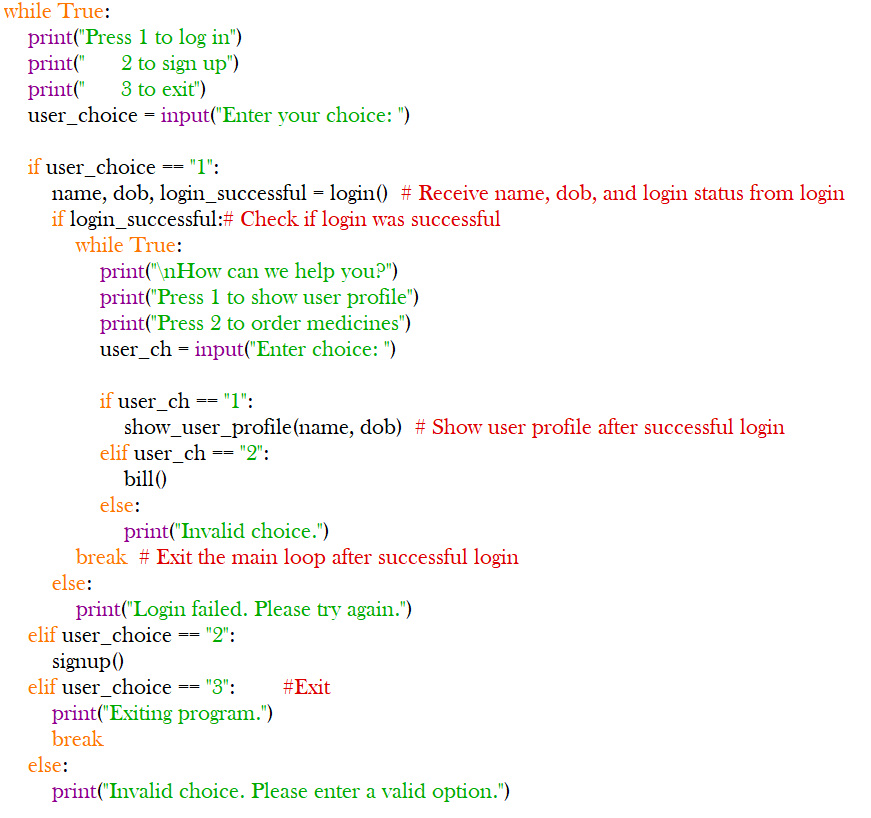
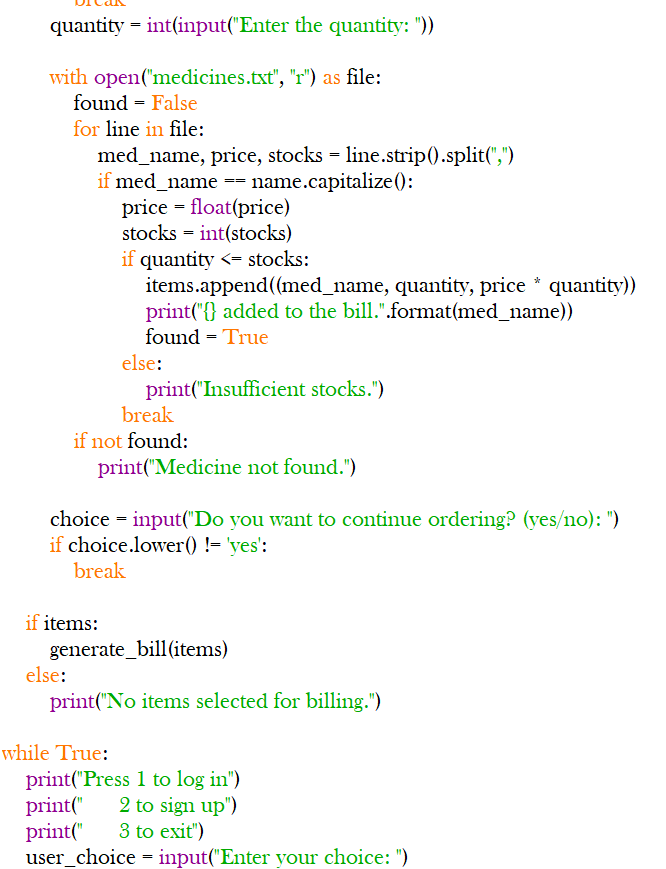




**II.USER MODE:**

In the user mode, users can perform the following tasks:

* Login: Users can log in using their name and date of birth to access personalized services.
* Signup: New users can create an account by providing their details. After signing up users have to login with their name and password (i.e. date of birth).
* After logging in, another menu appears which contains:
* Show User Profile: Users can view their profile information, including name, date of birth, age, gender, and medical condition.
* Order Medicines: Users can order medicines from the available inventory and generate a bill for the same.



**Main Code:**

while True:

print("\*"\*55)

print("\*"\*10+"WELCOME TO OUR INFIRMARY"+"\*"\*10)

print("\*"\*55)

print("Press 1 for admin mode and 2 for user mode ")

ch=int(input("Enter your mode:"))

if(ch==1):

password=input("Please enter your password: ")

if(password=="care@108"):

print("\*"\*51)

print("\*"\*15+"Welcome to admin mode"+"\*"\*15)

print("\*"\*51)

from admin import\*

break

else:

print("Incorrect password.Please try again.")

elif(ch==2):

print("\*"\*50)

print("\*"\*15+"Welcome to user mode"+"\*"\*15)

print("\*"\*50)

from user import\*

break

else:

print("Invalid input. Try again")

**Module 1:**

def admin\_mode():

def patient\_management():

def display\_menu():

# Display menu options for patient management

print("Press 1. Add new patient")

print(" 2. Edit patient information")

print(" 3. Display all patients")

print(" 4. Delete a patient")

print(" 5. Exit")

def add\_patient():

#Add a new patient to the records

with open("patients.txt", "a") as file:

name=input("Enter name: ")

dob=input("Enter date of birth in the form (DDMMYYYY): ")

age=input("Enter age: ")

gender=input("Enter gender: ")

condition=input("Enter condition: ")

patient = name + "," + dob + "," + age + "," + gender + "," + condition + "\n"

file.write(patient)

print("Patient added successfully.")

def edit\_patient():

#Edit information of the existing patient

with open("patients.txt", "r") as file:

patients = file.readlines()

print("Select a patient to edit:")

for i, patient in enumerate(patients):

print(str(i+1) + ". " + patient.strip())

index = int(input("Enter the index of the patient to edit: ")) - 1

new\_name = input("Enter new name: ")

new\_dob = input("Enter new dob(DDMMYYYY): ")

new\_age = input("Enter new age: ")

new\_gender = input("Enter new gender: ")

new\_condition = input("Enter new condition: ")

patients[index] = new\_name + "," + new\_age + "," + new\_gender + "," + new\_condition + "\n"

with open("patients.txt", "w") as file:

file.writelines(patients)

print("Patient information updated successfully.")

def display\_patients():

#Display information of all patients

with open("patients.txt", "r") as file:

print("Patient Information:")

print("-"\*30)

for line in file:

details = line.strip().split(",")

name, dob, age, gender, condition = details

print("Name: " + name +", DOB: " + dob + ", Age: " + age + ", Gender: " + gender + ", Condition: " + condition)

print("-"\*30)

def delete\_patient():

#Delete a patient details from the records

with open("patients.txt", "r") as file:

patients = file.readlines()

print("Select a patient to delete:")

for i, patient in enumerate(patients):

print(str(i+1) + ". " + patient.strip())

index = int(input("Enter the index of the patient to delete: ")) - 1

deleted\_patient = patients.pop(index)

with open("patients.txt", "w") as file:

file.writelines(patients)

print(deleted\_patient.strip() + " has been deleted.")

while True:

display\_menu()

choice = input("Enter your choice: ")

if choice == "1": #Add a new patient data

add\_patient()

elif choice == "2": #Edit existing patient data

edit\_patient()

elif choice == "3": #Display all patients data

display\_patients()

elif choice == "4": #Delete a patient data

delete\_patient()

elif choice == "5": #Exit

print("Exiting program.")

break

else:

print("Invalid choice. Please enter a valid option.")

print("\*"\*40)

def doctor\_management():

def display\_menu():

#Display menu options for doctor management

print("Press 1. Add new doctor")

print(" 2. Display all doctors")

print(" 3. Exit")

def add\_doctor():

#Add a new doctor in the records

with open("doctors.txt", "a") as file:

name = input("Enter doctor name: ")

age = input("Enter age: ")

spl = input("Enter specialisation: ")

file.write(name + "," + age + "," + spl + "\n")

print("Name added successfully.")

def display\_doctors():

#Display information of all existing doctors

with open("doctors.txt", "r") as file:

print("Doctor Information:")

print("-"\*30)

for line in file:

details = line.strip().split(",")

name, age, spl = details

print("Name: " + name + ", Age: " + age + ", Specialisation: " + spl )

print("-"\*30)

while True:

display\_menu()

choice = input("Enter your choice: ")

if choice == "1": #Add a new doctor data

add\_doctor()

elif choice == "2": #Display all doctors data

display\_doctors()

elif choice == "3": #Exit

print("Exiting program.")

break

else:

print("Invalid choice. Please enter a valid option.")

def medicines\_records():

def display\_menu():

#Display menu options for medicines records

print("1. Add new medicine")

print("2. Edit medicine information")

print("3. Display all medicines")

print("4. Update medicine price")

print("5. Update medicine stocks")

print("6. Exit")

def add\_medicine():

#Add a new medicine data

with open("medicines.txt", "a") as file:

name = input("Enter medicine name: ")

price = input("Enter medicine price: ")

stocks = input("Enter medicine stocks: ")

file.write(name.capitalize() + "," + price + "," + stocks + "\n")

print("Medicine added successfully.")

def edit\_medicine():

#Edit data of an existing medicine

with open("medicines.txt", "r") as file:

medicines = file.readlines()

print("Select a medicine to edit:")

for i, medicine in enumerate(medicines):

med\_info = medicine.strip().split(",")

print(str(i+1) + ". " + med\_info[0])

index = int(input("Enter the index of the medicine to edit: ")) - 1

new\_name = input("Enter new name: ")

new\_price = input("Enter new price: ")

new\_stocks = input("Enter new stocks: ")

medicines[index] = new\_name + "," + new\_price + "," + new\_stocks + "\n"

with open("medicines.txt", "w") as file:

file.writelines(medicines)

print("Medicine information updated successfully.")

def display\_medicines():

#Display information of all medicines

with open("medicines.txt", "r") as file:

print("Medicine Information:")

print("-" \* 15)

for line in file:

med\_info = line.strip().split(",")

name, price, stocks = med\_info

print("Name: " + name + ", Price: " + price + ", Stocks: " + stocks)

print("-" \* 15)

def update\_price():

#Update price of an existing medicine

name = input("Enter the name of the medicine: ")

new\_price = input("Enter the new price: ")

updated = False

with open("medicines.txt", "r") as file:

lines = file.readlines()

with open("medicines.txt", "w") as file:

for line in lines:

parts = line.strip().split(",")

if parts[0] == name.capitalize():

line = name + "," + str(new\_price) + "," + parts[2] + "\n"

updated = True

file.write(line)

if updated:

print("Medicine price updated successfully.")

else:

print("Medicine not found.")

def update\_stocks():

#Update stocks of an existing medicine

name = input("Enter the name of the medicine: ")

new\_stocks = input("Enter the new stocks: ")

updated = False

with open("medicines.txt", "r") as file:

lines = file.readlines()

with open("medicines.txt", "w") as file:

for line in lines:

parts = line.strip().split(",")

if parts[0] == name:

line = name+","+parts[1]+","+str(new\_stocks)+"\n"

updated = True

file.write(line)

if updated:

print("Medicine stocks updated successfully.")

else:

print("Medicine not found.")

while True:

display\_menu()

choice = input("Enter your choice: ")

if choice == "1": #Add a new medicine data

add\_medicine()

elif choice == "2": #Edit existing medicine data

edit\_medicine()

elif choice == "3": #Display all medicines data

display\_medicines()

elif choice == "4": #Update price data of existing medicine

update\_price()

elif choice == "5": #Update stocks data of existing medicine

update\_stocks()

elif choice == "6": #Exit

print("Exiting program.")

break

else:

print("Invalid choice. Please enter a valid option.")

while True:

#Display all functions of an admin

print("\*"\*40)

print("Press 1 to manage patients")

print(" 2 to manage doctors")

print(" 3 to manage medicines")

print(" 4 to exit")

admin\_ch=input("Enter your choice:")

if(admin\_ch=="1"): #To manage patient records

patient\_management()

elif(admin\_ch=="2"): #To manage doctor records

doctor\_management()

elif(admin\_ch=="3"): #To manage medicine records

medicines\_records()

elif(admin\_ch=="4"): #To exit

break

else:

print("Invalid choice.Please choose from given options.")

admin\_mode()

**Module 2:**

def login():

name = input("Enter your name: ")

dob = input("Enter your password (DD/MM/YYYY): ")

login\_successful = False # Initialize login status flag

with open("patients.txt", "r") as file:

for line in file:

details = line.strip().split(",")

if details[0] == name and details[1] == dob:

print("Login successful!")

login\_successful = True

break # Exit loop once login is successful

if not login\_successful:

print("Invalid credentials. Please try again")

return name, dob, login\_successful

def signup():

with open("patients.txt", "a") as file:

name = input("Enter your name: ")

dob = input("Enter your date of birth (DDMMYYYY): ")

age=input("Enter age: ")

gender=input("Enter gender(M/F/O): ")

condition=input("Enter condition: ")

patient = name + "," + dob + "," + age + "," + gender + "," + condition+ "\n"

file.write(str(patient) + "\n")

print("Account created successfully.")

logged\_in = False

def show\_user\_profile(name, dob):

with open("patients.txt", "r") as file:

for line in file:

details = line.strip().split(",")

if details[0] == name and details[1] == dob:

print("User Profile:")

print("Name:", details[0])

print("Date of Birth:", details[1])

print("Age:", details[2])

print("Gender:", details[3])

print("Condition:", details[4])

return

print("User profile not found.")

def bill():

def display\_medicines():

with open("medicines.txt", "r") as file:

print("Medicine Information:")

print("-" \* 30)

for line in file:

name, price, stocks = line.strip().split(",")

print("Name:" + name + ",Price:" + price + ", Stocks:" + stocks)

print("-" \* 30)

def generate\_bill(items):

total\_amount = sum(item[2] for item in items)

print("\nItems in Bill:")

print("-" \* 30)

print("Name\t\tQuantity\tPrice")

for item in items:

print(item[0]+" "+str(item[1])+" "+str(item[2]))

print("Total Amount:", total\_amount)

print("-" \* 30)

items = []

while True:

display\_medicines()

name = input("Enter the name of the medicine (or 'done' to finish adding items): ")

if name.lower() == 'done':

break

quantity = int(input("Enter the quantity: "))

with open("medicines.txt", "r") as file:

found = False

for line in file:

med\_name, price, stocks = line.strip().split(",")

if med\_name == name.capitalize():

price = float(price)

stocks = int(stocks)

if quantity <= stocks:

items.append((med\_name, quantity, price \* quantity))

print("{} added to the bill.".format(med\_name))

found = True

else:

print("Insufficient stocks.")

break

if not found:

print("Medicine not found.")

choice = input("Do you want to continue ordering? (yes/no): ")

if choice.lower() != 'yes':

break

if items:

generate\_bill(items)

else:

print("No items selected for billing.")

while True:

print("Press 1 to log in")

print(" 2 to sign up")

print(" 3 to exit")

user\_choice = input("Enter your choice: ")

if user\_choice == "1":

name, dob, login\_successful = login() # Receive name, dob, and login status from login

if login\_successful:# Check if login was successful

while True:

print("\nHow can we help you?")

print("Press 1 to show user profile")

print("Press 2 to order medicines")

user\_ch = input("Enter choice: ")

if user\_ch == "1":

show\_user\_profile(name, dob) # Show user profile after successful login

elif user\_ch == "2":

bill()

else:

print("Invalid choice.")

break # Exit the main loop after successful login

else:

print("Login failed. Please try again.")

elif user\_choice == "2":

signup()

elif user\_choice == "3": #Exit

print("Exiting program.")

break

else:

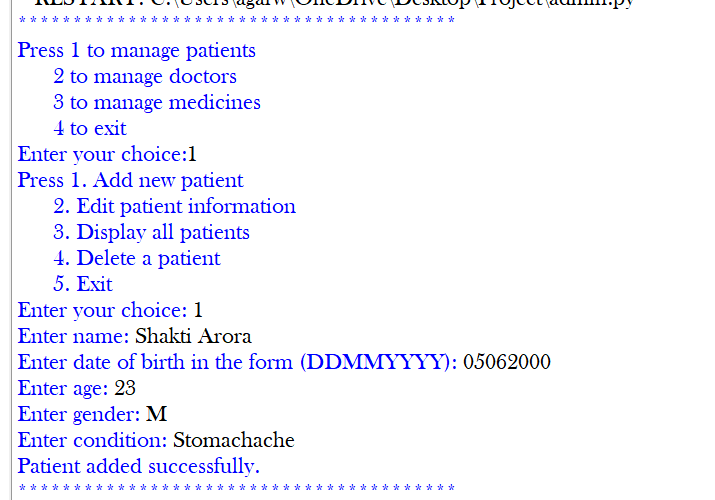
print("Invalid choice. Please enter a valid option.")

**OUTPUTS:**

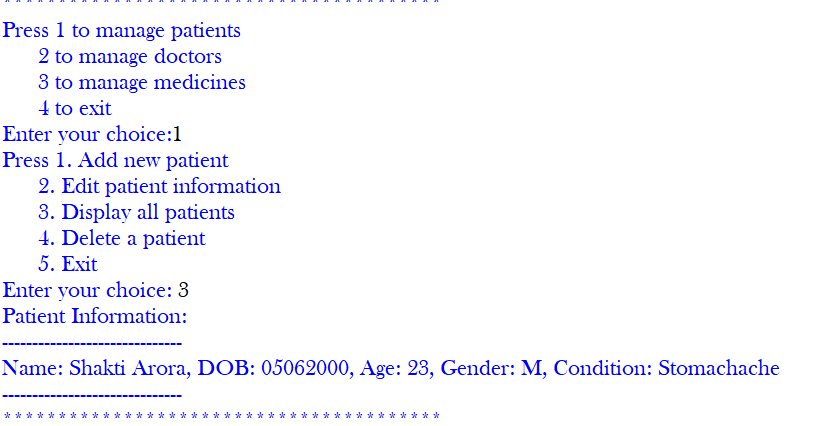
****

Choose mode

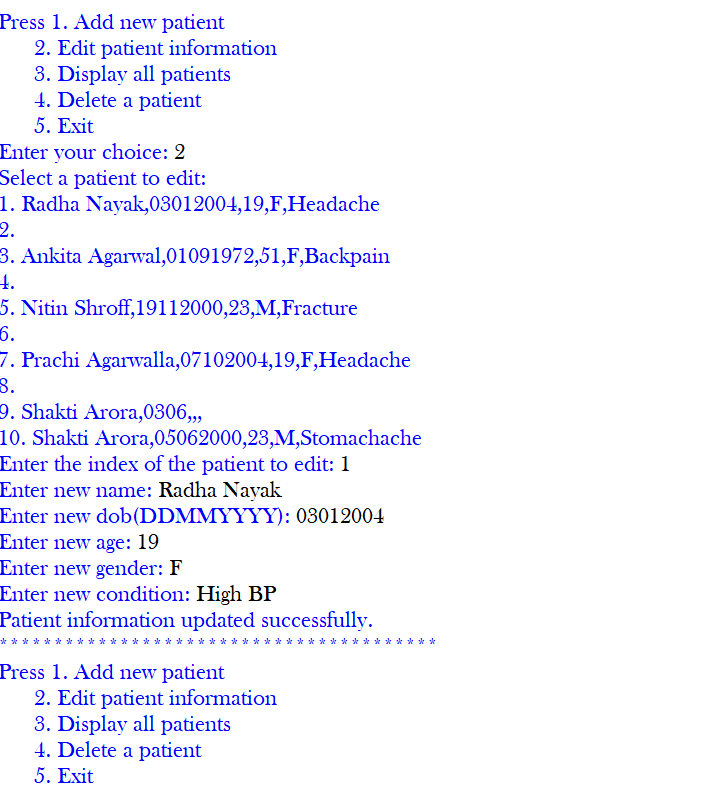
1.Admin Mode

****

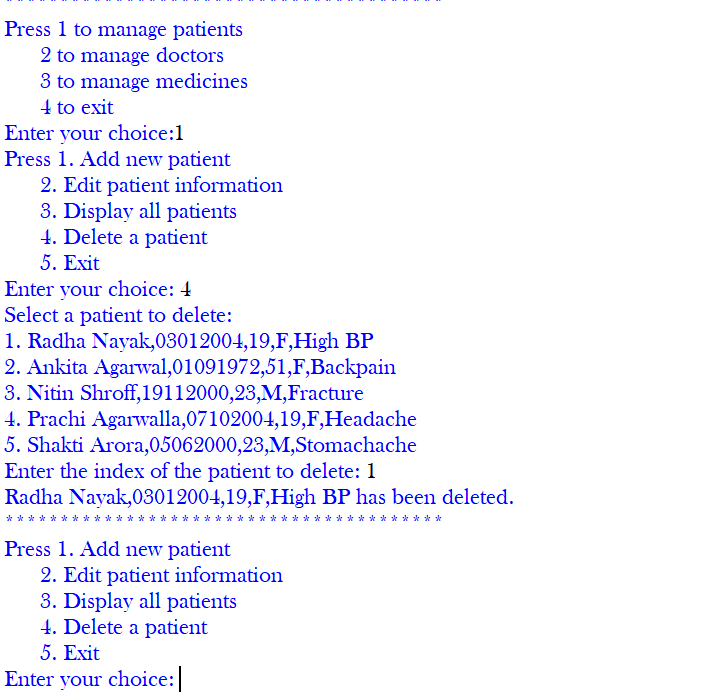
**Add new patient**

****

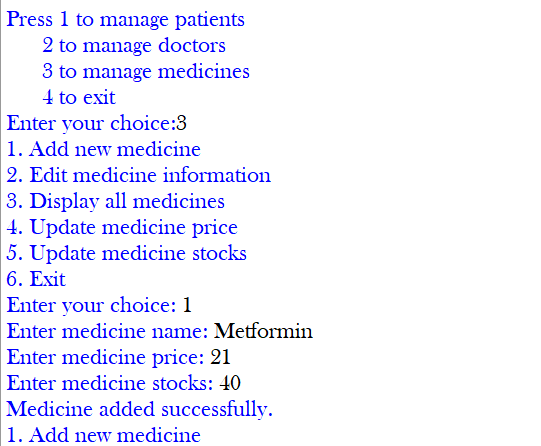
**Display patient**

****

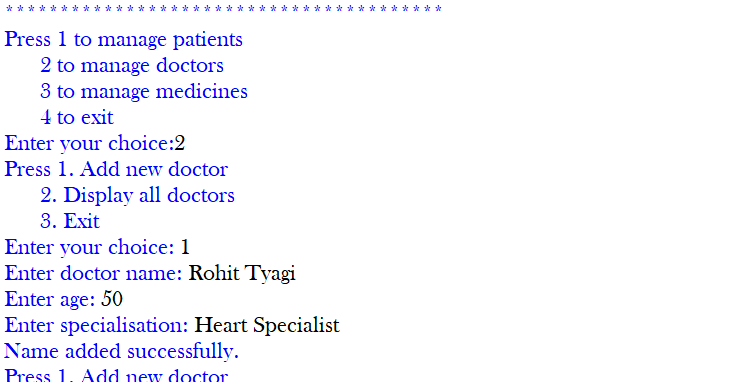
**Edit patient data**

****

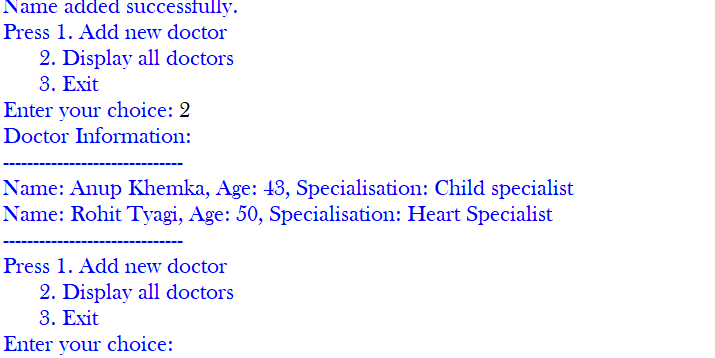
**Delete patient data**

****

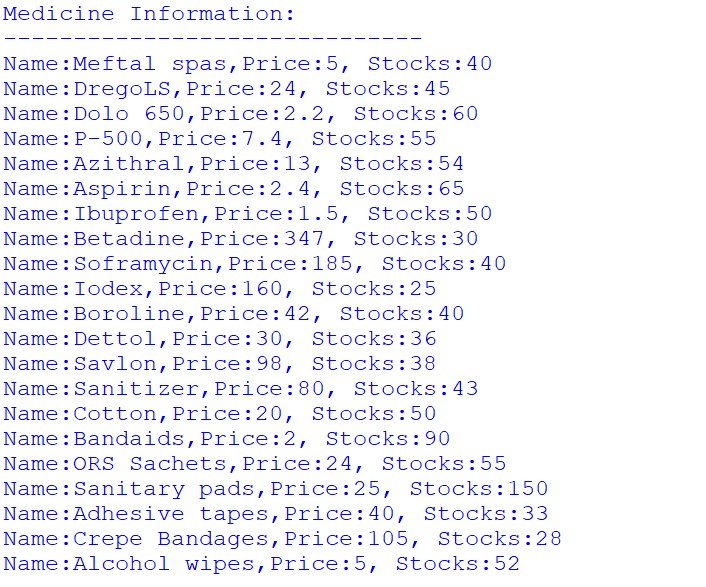
**Add new medicine**

****

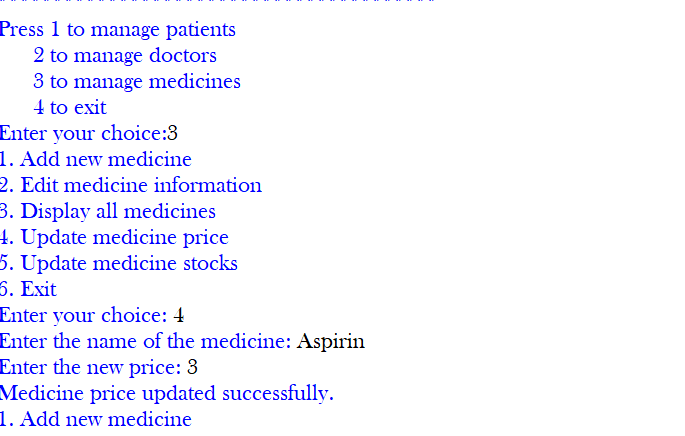
**Add doctor**

****

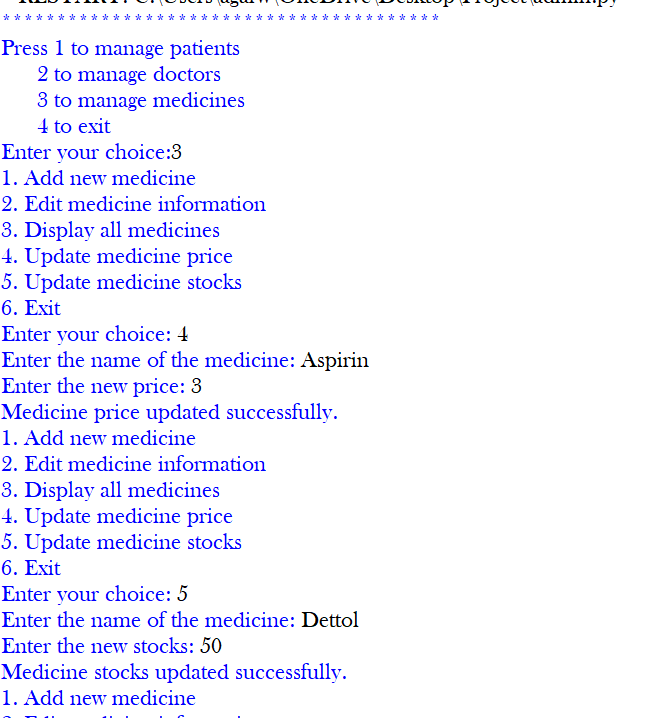
**Display doctors data**



**Display medicines**

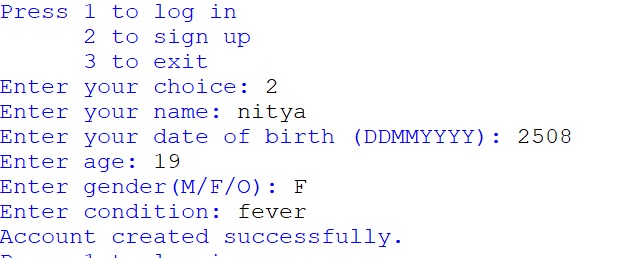
****

**Update medicine price**

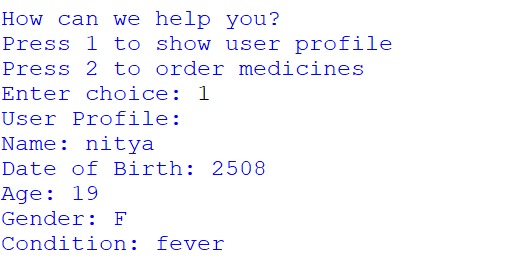
****

**Update medicine stocks**

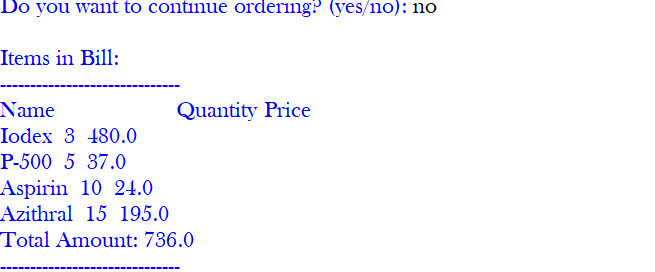
**2.User mode**



**Sign up**



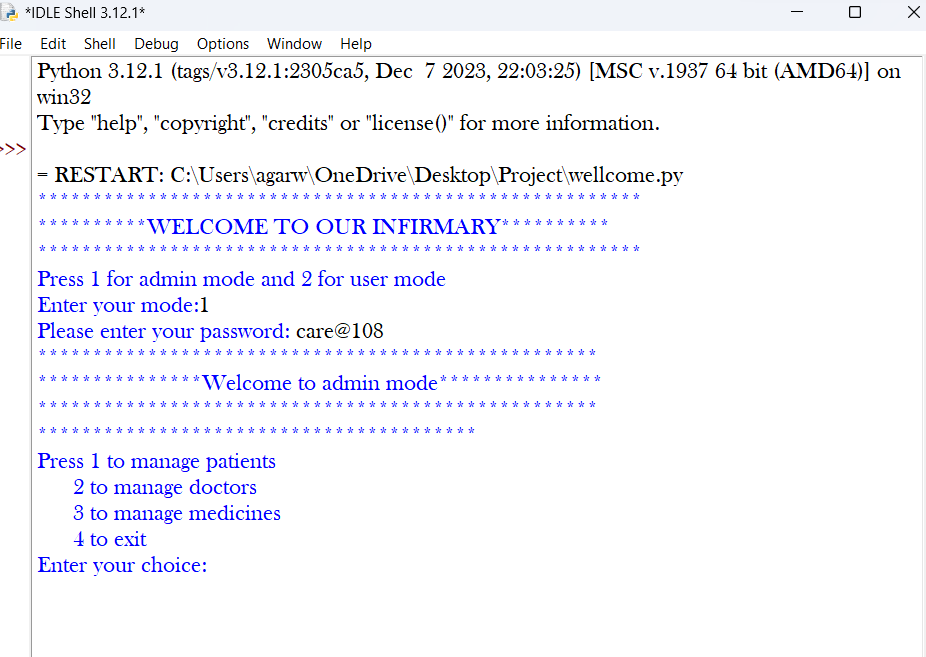
**To show user profile**



**Generate bill**

Key Features :-

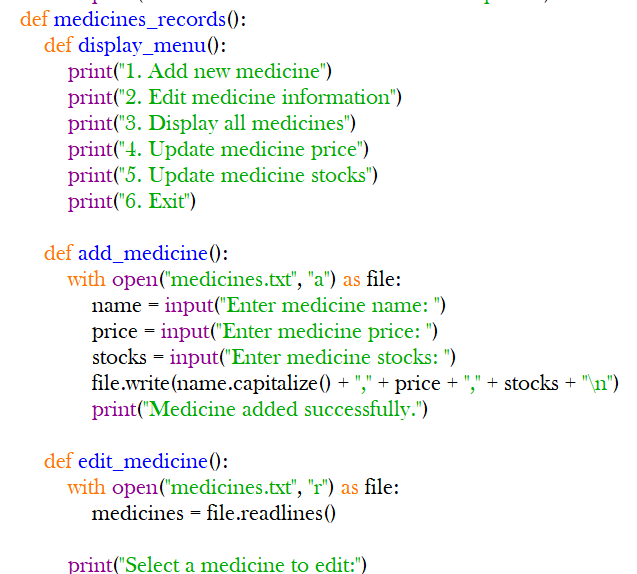
* **Authentication:**

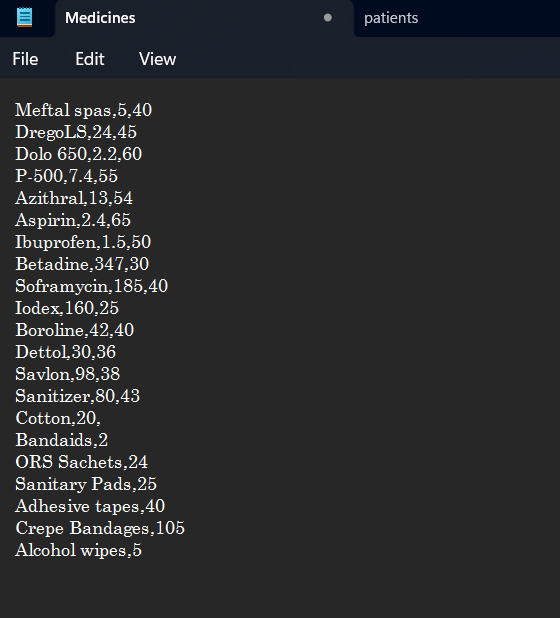




The program ensures secure access by requiring administrators to enter a password in admin mode as well as user mode.

* Data Management:





Patient records, doctor details, and medicine inventory are stored in separate text files and managed through file I/O operations.

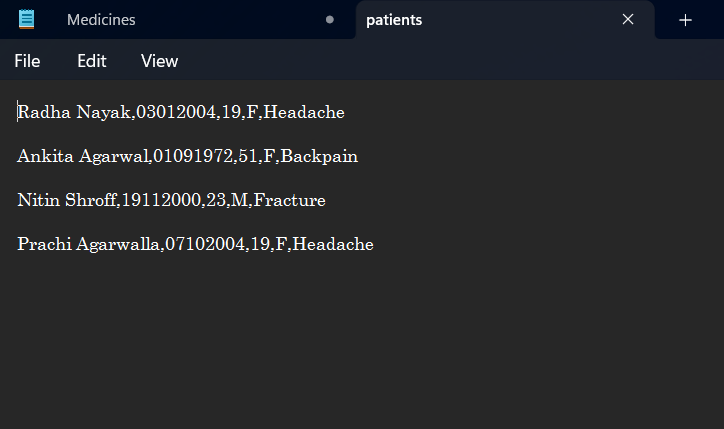
* Modular Design:



The program is modularized, with separate functions and modules for different functionalities, enhancing code organization and readability.

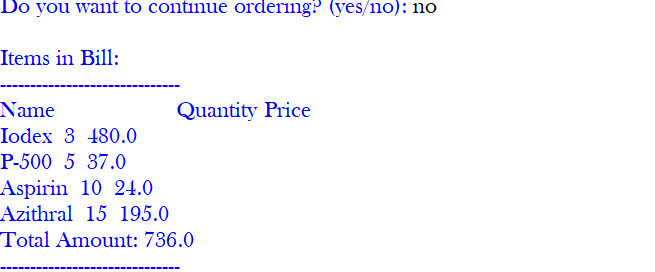
* Input Validation:





Various input validation checks are implemented to handle invalid inputs gracefully and provide a smooth user experience.

* Billing System:



Users can order medicines and generate a bill for their purchases, facilitating seamless transactions.

**CONCEPTS USED :-**

1. File Handling:

The program reads from and writes to different text files to store patient information, doctor information and medicine details, respectively. The open() function is used to open files, and the with statement ensures that the files are properly closed after use.

1. Functions and nested functions:

The **admin\_mode()** function contains nested functions (**patient\_management()**, **doctor\_management()**, **medicines\_records()**), which encapsulate different aspects of admin functionalities .The program defines several functions (**login()**, **signup()**, **show\_user\_profile()**, and **bill()**) to modularize the code and promote reusability. Functions help in organizing code and performing specific tasks.

1. Input/output:

The **input()** function is used to accept user input from the console. User input is validated and processed accordingly to perform actions like adding,editing,deleting details in the text files, logging in, signing up, and ordering medicines.

1. Conditional statements:

Conditional statements (**if**, **elif**, **else**) are used to execute different blocks of code based on certain conditions. For example, the script checks user input to determine whether to log in, sign up, or exit the program.

1. Loops:

The script uses **while** loops to repeatedly prompt the user for input until certain conditions are met. Nested loops are also used within the **bill()** function to handle user interactions related to ordering medicines. For example, the outer loop in **admin\_mode()** continues until the user chooses to exit, and inner loops within **patient\_management()**, **doctor\_management()**, and **medicines\_records()** handle the menu options for each functionality.

1. String Manipulation:

String manipulation techniques are used to process and format data. Functions like **split()** and string concatenation (**+** operator) are used to manipulate strings and extract relevant information from them. Other string manipulation techniques such as splitting, joining, and formatting are used to process and display data read from files, as well as to construct data to be written to files.

1. Variable scope:

The scope of variables is managed appropriately within functions and the main program.

1. Menu driven programs:

The menu-driven approach employed in the code allows users to interact with the program by presenting them with a series of options displayed as a menu. Users can navigate through different functionalities by selecting the corresponding menu options.

1. Modules and packages:

Data structures like dictionaries and lists are implicitly used to store or fetch data in order to perform operations.

1. Data Structures:

Depending on the selected mode (admin or user), the script imports functionality from different modules (**admin** or **user**). This allows for modular code organization and separation of concerns, making the codebase more maintainable and scalable.