Hospital Management system

Add a short description

11:11PM

Classes
Class Hospital

Classes.py > Patient > add_to_system

```python
from tkinter import messagebox
import tkinter as Tk


# Class Hospital-------------------------------------------------------------------------------
class Hospital:
    def __init__(self):
        pass
    def add_to_system(self):
        pass
    def displayRecords(self):
        pass
```

This code sets up the structure for a Hospital class but doesn't implement any functionality yet. It includes placeholders for methods (add_to_system and displayRecords), which could later be developed to add data to the hospital system and display records. The pass statements are placeholders that allow the code to run without raising an error, even though the methods don't do anything yet.

Classes
Class Patients

Classes.py > Bill > add_to_records

```python
# Class Patient-----------------------------------------------------------------------------
class Patient(Hospital):
    records = {}
    def __init__(self,name,age,gender,phone_number,PatientID,MedicalIssues):
        self.__name = name
        self.__age = age
        self.__gender = gender
        self.__phone_number = phone_number
        self.__PatientID = PatientID
        self.__PatientMedical = MedicalIssues

    @property
    def add_to_system(self):

        if self.__PatientID in self.records:
            messagebox.showinfo("Alert","PATIENT ID ALREADY IN SYSTEM")
        else:
            self.records[self.__PatientID] = [self.__name,self.__gender,self.__age,self.__phone_number,self.__PatientMedical]
            messagebox.showinfo("Complete","PATIENT ID ADDED TO SYSTEM")
```

Classes.py > Patient

```python
class Patient(Hospital):

    @property
    def displayRecords(self):
        for key,value in records.items():
            if len(self.records) == 0:
                print("\nNo patient records available\n")
                print('----------------------------------------')
            else:
                for key, patient in self.records.items():
                    print(f'\nPatient ID: {key}')
                    print(f'Patient name: {patient[0]}')
                    print(f'Patient gender: {patient[1]}')
                    print(f'Patient age: {patient[2]}')
                    print(f"Patient Medical Issues: {patient[4]}")
                    print(f'Patient Phone number: {patient[3]}')
                    print('----------------------------------------')
```

The **Patient class** manages patient records with:

- **Add to System**: Checks if a **PatientID** already exists in **Patient_records.** If it doesn't, it adds the patient's details and confirms.

- **Display Records**: Shows all patient records. If none exist, it displays a message indicating no records are available.

The **Patient_records** dictionary stores each patient's information, ensuring unique identification by `PatientID`.

Classes
Class Doctors

Classes.py > Patient

```python
#Class Doctor-----------------------------------------------------------------
class Doctor(Hospital):
    Doc_records = {}

    def __init__(self,name,gender,phone_number,DocID):
        self.__name = name
        self.__gender = gender
        self.__phone_number = phone_number
        self.__DocID = DocID

    @property
    def add_to_system(self):
        if self.__DocID in self.Doc_records:
            print("\nDoctor already exists in the system\n")
            messagebox.showinfo("Alert","Doctor already exists in the system")
        else:
            self.Doc_records[self.__DocID] = [self.__name,self.__gender,self.__phone_number]
            print('\nDoctor added to System\n')
            messagebox.showinfo("COMPLETE","Doctor added to System")
        print('----------------------------------------')
```

Classes.py > Patient

```python
class Doctor(Hospital):

    @property
    def displayRecords(self):
        if len(self.Doc_records) == 0:
            print("\nNo Doctor records available\n")
        else:
            for key, doctor in self.Doc_records.items():
                print(f'\nDoctor ID: {key}')
                print(f'Doctor name: {doctor[0]}')
                print(f'Doctor gender: {doctor[1]}')
                print(f'Doctor Phone number: {doctor[2]}')
        print('----------------------------------------')
```

This **Doctor** class, inheriting from **Hospital**, manages a record system for doctors. It includes methods to:

- **Add to System**: Checks if a doctor's **DocID** already exists in **Doc_records**. If it does, it displays a message indicating the doctor is already in the system. Otherwise, it adds the doctor's details (name, gender, phone number) to **Doc_records** and confirms the addition.

- **Display Records**: Shows all records in **Doc_records.** For each doctor, it prints their ID, name, gender, and phone number. If there are no records, it displays a message indicating that no records are available.

The **Doc_records** dictionary stores each doctor's information using their **DocID** as the key.

Classes
Class Appointment

```python
class Appointment(Patient):
    appointments = {}
    def __init__(self, date, time, docname, DocID, Patientname, AppoID):

        self.__date = date
        self.__time = time
        self.__docname = docname
        self.__DocID = DocID
        self.__Patientname = Patientname
        self.__AppoID = AppoID

    @property
    def appointment_scheduling(self):
        if self.__AppoID in self.appointments and self.__DocID in self.appointments and self.__date in self.appointments:
            print("Appointment already set")
        else:
            self.appointments[self.__AppoID] = [self.__date, self.__time, self.__Patientname, self.__docname, self.__DocID]
            print('\nAppointment scheduled\n')
        print('-----------------------------------------')
```

Classes.py > Appointment > __init__

```python
class Appointment(Patient):
    @property
    def appointments_display(self):
        if len(self.appointments) == 0:
            print("\nNo appointments Scheduled\n")
        else:
            for key, appointment in self.appointments.items():
                print(f'\nDoctor ID: {key}')
                print(f'Patient ID: {appointment[2]}')
                print(f'Appointment date: {appointment[0]}')
                print(f'Appointment time: {appointment[1]}')
        print('-----------------------------------------')
```

The **Appointment** class, inheriting from **Patient**, manages appointment scheduling for patients and doctors. It includes methods to:

- **Appointment Scheduling**: Checks if an appointment with the same **AppoID** , **DocID**, and date already exists in **appointments**. If it does, it displays a message that the appointment is already set. Otherwise, it adds a new entry with the appointment details (date, time, patient name, doctor name, and doctor ID) to the **appointments** dictionary and confirms the scheduling.

- **Appointments Display**: Lists all scheduled appointments. For each appointment, it displays the doctor ID, patient name, appointment date, and time. If there are no appointments, it displays a message indicating no appointments are scheduled.

The **appointments** dictionary stores each appointment, with **AppoID** as the key.

Classes
Class Rooms

```python
#Class Room-------------------------------------------------------------------------------
class Room:
    room_records = {}
    def __init__(self, RoomNumber, PatientID):
        self.__room_number = RoomNumber
        self.__roomPatient = PatientID

    @property
    def assign_room(self):
        if self.__room_number in self.room_records:
            print("The room is already occupied")
            messagebox.showinfo("Alert", "The room is already occupied")
        else:
            self.room_records[self.__room_number] = [self.__roomPatient]
            print("Room added to system")
            messagebox.showinfo("COMPLETE", "Patient assigned room")
        print('----------------------------------------')
```

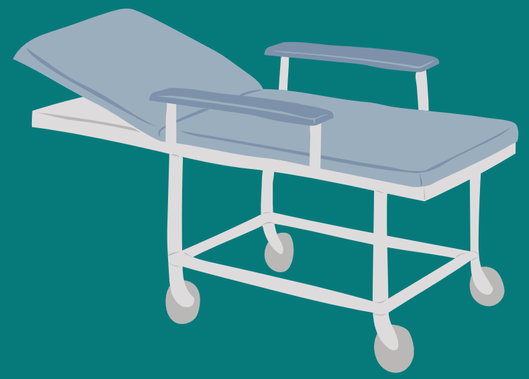
The **Room** class handles room assignments for patients. It includes the following:

- Attributes: Each room is assigned a **RoomNumber** and a **PatientID.**
- **Static Dictionary**: **room_records = {}** stores the room assignments, with each entry keyed by **RoomNumber.**

Method:

- **Assign Room**: Checks if a room (**RoomNumber**) is already in **room_records**:
- If the room is occupied, it displays a message indicating that.
- If the room is available, it adds the room to **room_records** with the assigned **PatientID,** confirming the assignment.

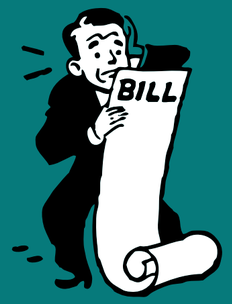
The **room_records** dictionary keeps track of each room and its assigned patient, ensuring only one patient per room.

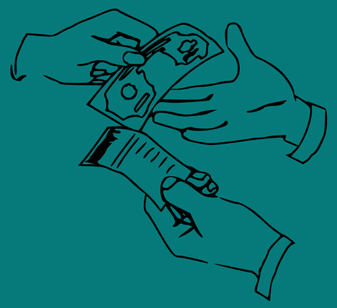


Classes
Class Bill

Classes.py > Room

```python


#Class Bill-------------------------------------------------------------------------
class Bill:
    bill_records = {}
    def __init__(self, PatientName, PatientID,services, TotalCharges):
        self.__PatientName = PatientName
        self.__PatientID = PatientID
        self.__services = services
        self.__TotalCharges = TotalCharges

    @property
    def add_to_records(self):
        if self.__PatientID in self.bill_records:
            print("Patient already billed")
            messagebox.showinfo("Alert","Patient already billed")
            print('-----------------------------------------')
        else:
            print("PATIENT BILLED")
            self.bill_records[self.__PatientID] = [self.__PatientName, self.__services, self.__TotalCharges]
            messagebox.showinfo("COMPLETE","PATIENT BILLED ✔")
            print('-----------------------------------------')



start = '*'*20
print(f'\n{start}WELCOME TO THE HOSPITAL MANAGEMENT SYSTEM🏥{start}\n')
```





The **Bill** class manages billing records for patients. Here's a breakdown:

- **Attributes**: Each bill includes a **PatientName**, **PatientID**, **services** (list of services rendered), and `TotalCharges` (total billing amount).
- **Static Dictionary**: **bill_records = {}** stores billing information for each patient, with `**PatientID**` as the key.

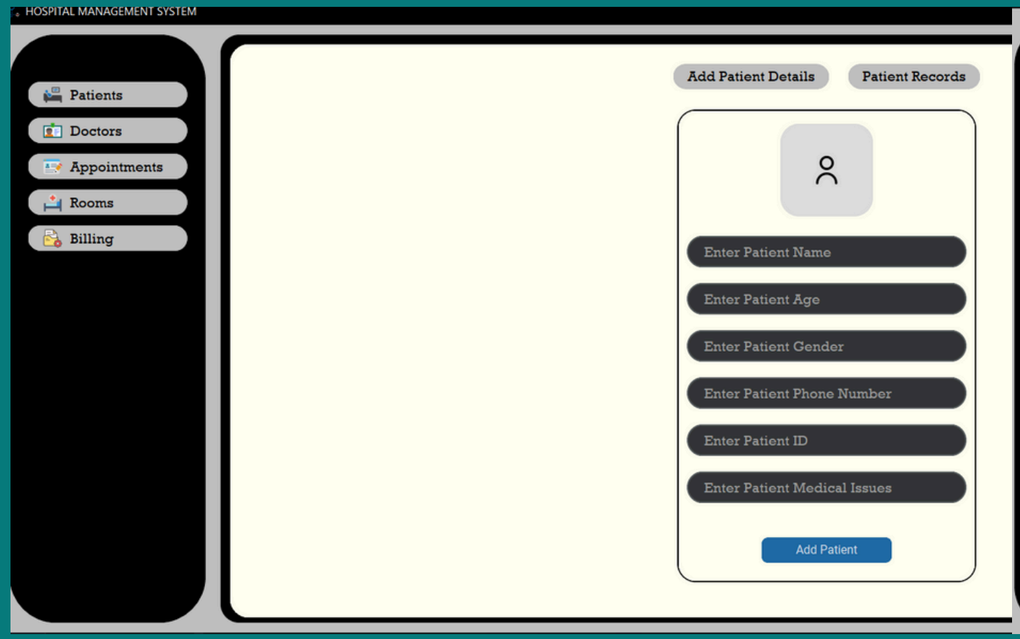
Method:
Add to Records: Checks if the **PatientID** already exists in **bill_records**:

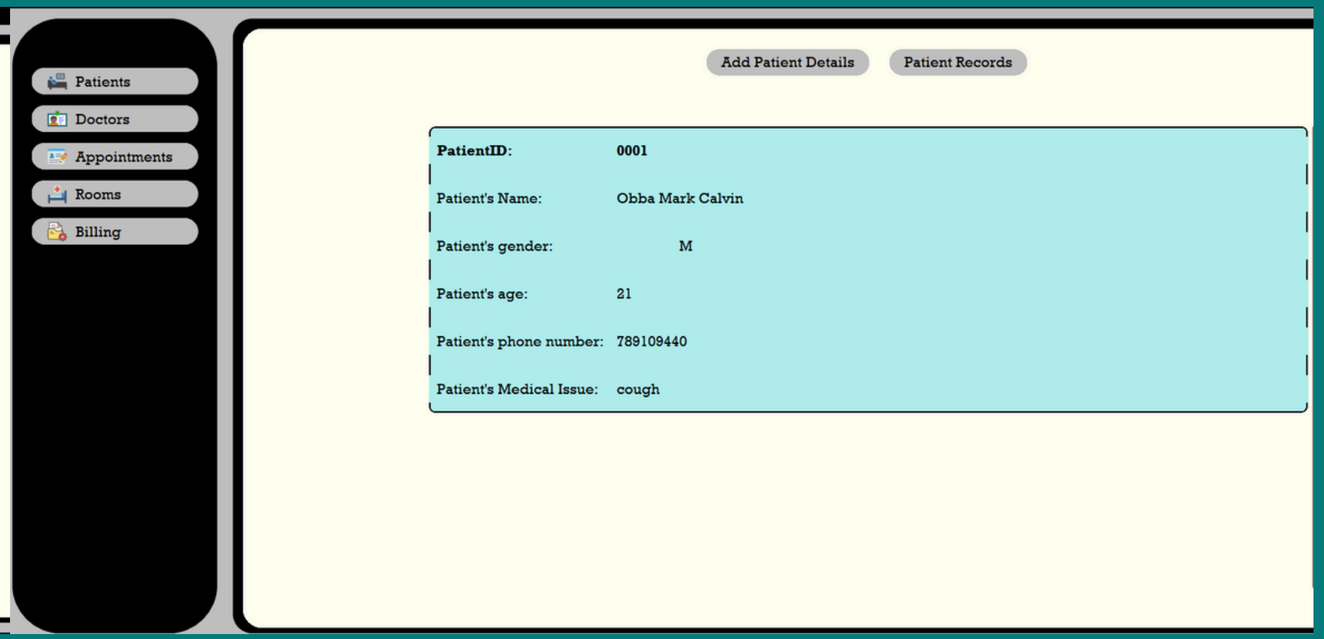
- If the patient is already billed, it displays a message indicating this.
- If not, it adds the patient's billing information (name, services, and charges) to **bill_records** and confirms billing.

The **bill_records** dictionary keeps track of each patient's billing details, ensuring each patient is billed only once.

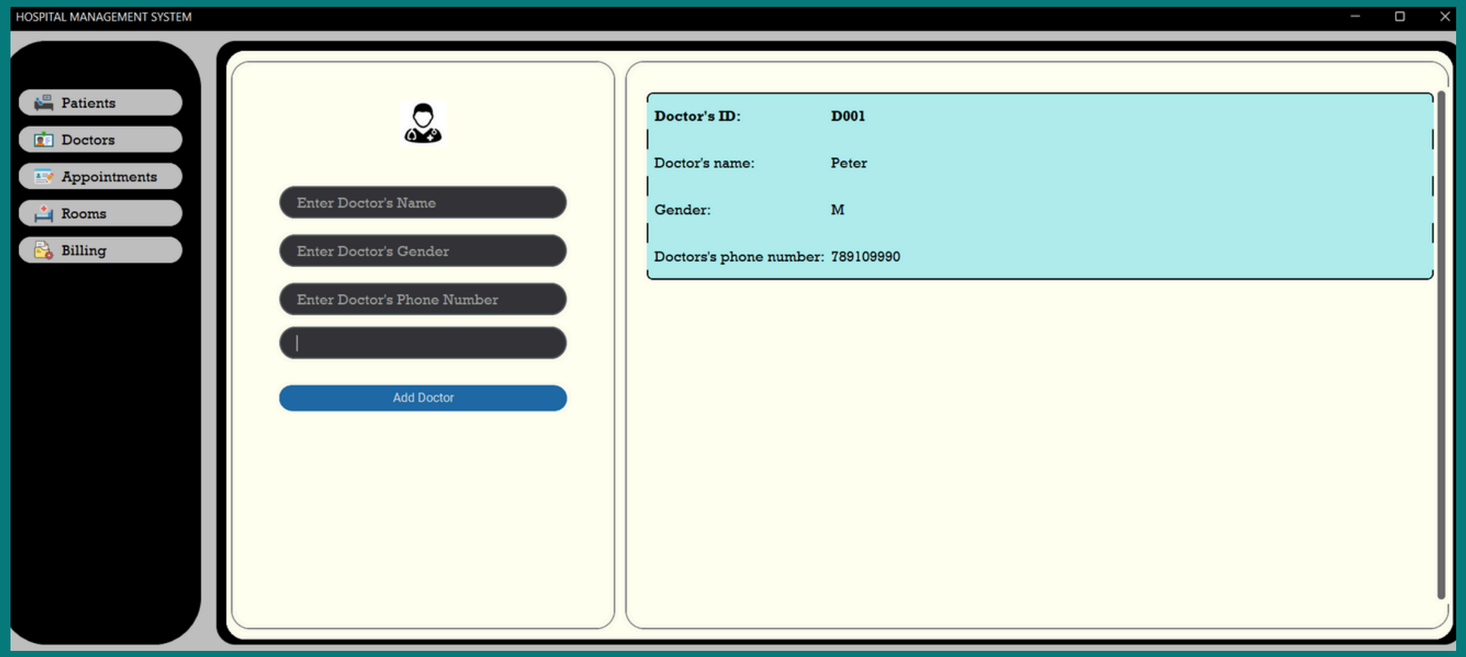
The code ends with a welcome message for the hospital management system, which is printed to the consol

HOW IT WORKS FOR CLASS PATIENT



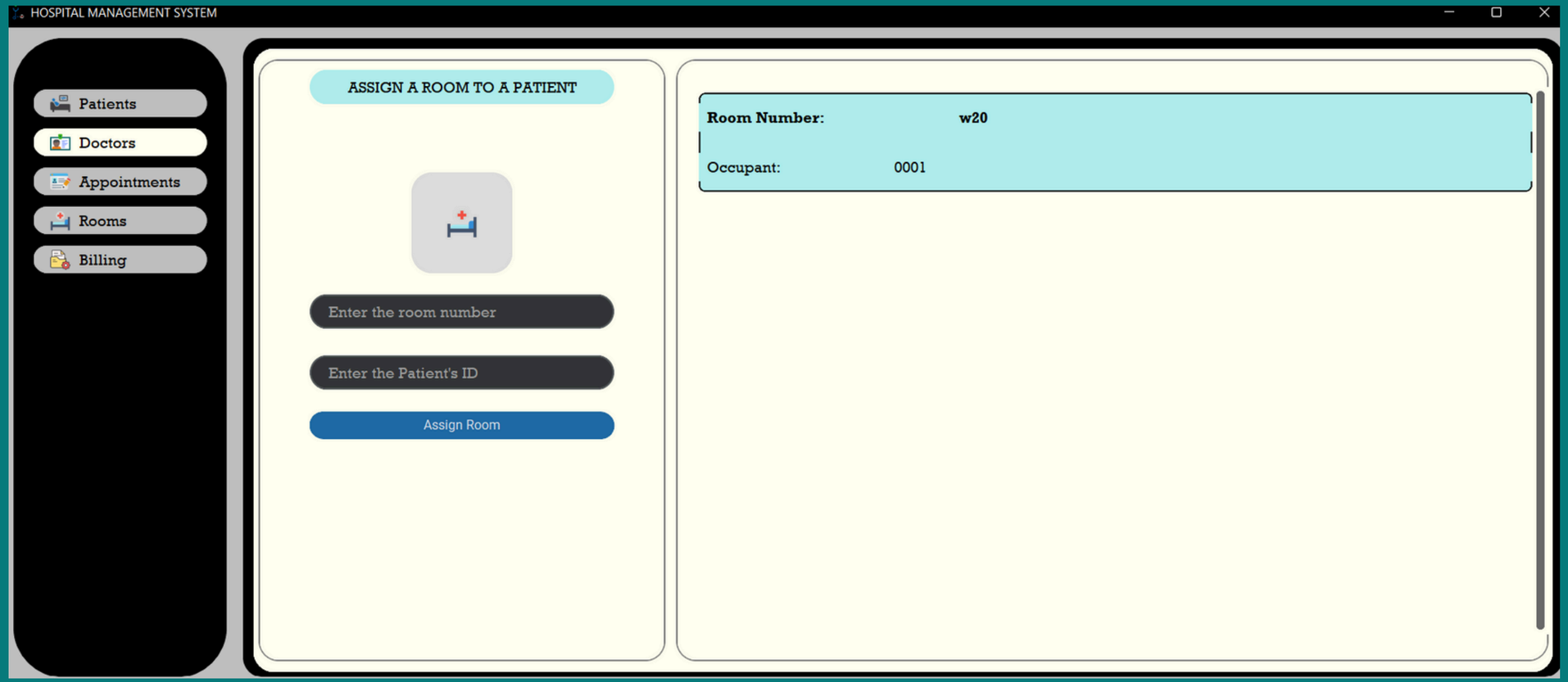


HOW IT WORKS FOR CLASS Doctor



HOW IT WORKS
FOR CLASS Appointment

HOW IT WORKS FOR CLASS Room



HOW IT WORKS FOR CLASS Bill