HOSPITAL MANAGEMENT SYSTEM

WE CARE HOSPITAL

PROJECT DONE BY:

AKSHAYA SRIKRISHNA – 2022103065

ANAGHA SRIKRISHNA – 2022103066

KRISHNENDU M R – 2022103081

PROBLEM STATEMENT

The objective of this project is to develop a hospital management system that will facilitate effective management of various aspects of the hospital's operations. Using Tkinter, the system will provide a user-friendly graphical user interface (GUI) and integrate with a MySQL database to store and retrieve relevant information.

The system should address the following requirements:

User Authentication: Implement a secure login mechanism to authenticate users, including doctors, nurses, administrative staff, and other personnel. Different levels of access should be granted based on user roles to ensure appropriate data privacy and security.

Patient Management: Enable users to manage patient records, including registration, appointment scheduling, medical history, and treatment details. The system should support functionalities such as adding new patients, updating their information, and retrieving patient data when required.

Room Management: Facilitate the allocation and management of hospital rooms. Users should be able to assign patients to specific rooms, track room availability, and manage patient transfers between rooms.

Staf f Management: Assist in managing hospital personnel, including doctors, nurses, and support staff. The system should enable users to track employee information, such as work schedules, shifts, department management, and role assignation.

Billing and Payment: Provide functionality for generating and managing patient bills, tracking payments, and generating bill reports. The system should be capable of integrating with billing with room rate, latest treatment fees and medicines fees.

Database Connectivity: Establish a connection to a MySQL database to store and retrieve critical information, including patient records, inventory data, staff details, billing information, and other relevant data necessary for efficient hospital management.

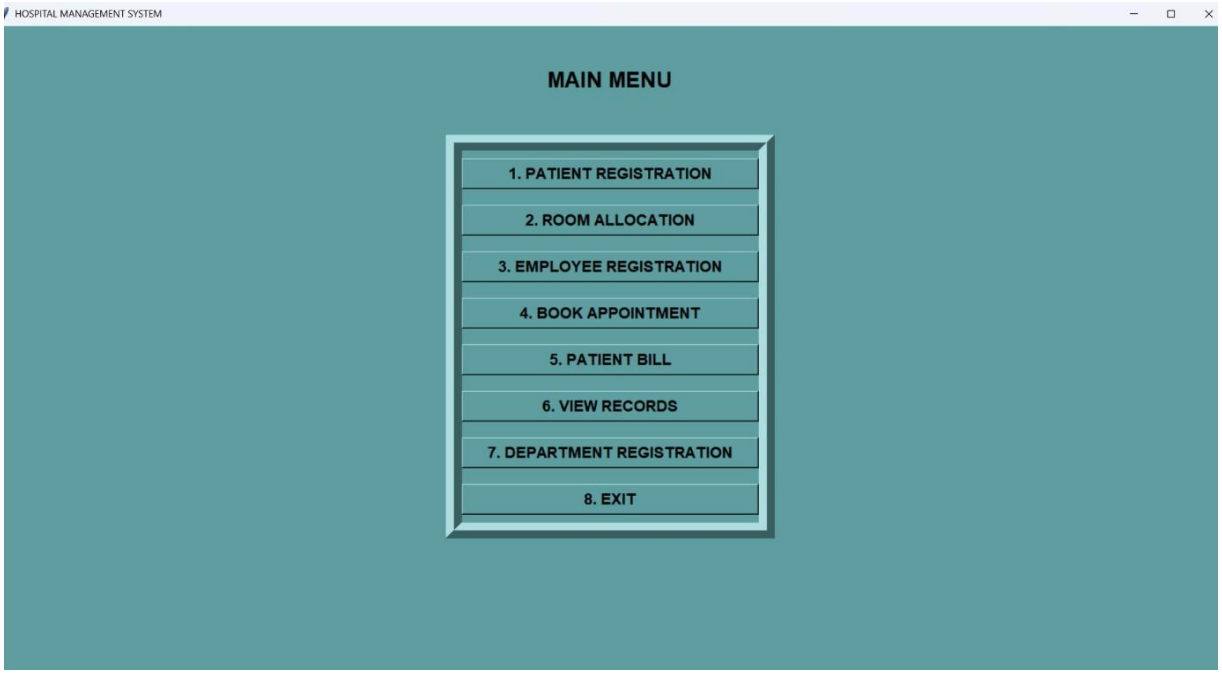
By addressing these needs, the hospital management system will streamline administrative tasks, improve patient care, and enhance overall operational efficiency within the hospital environment.

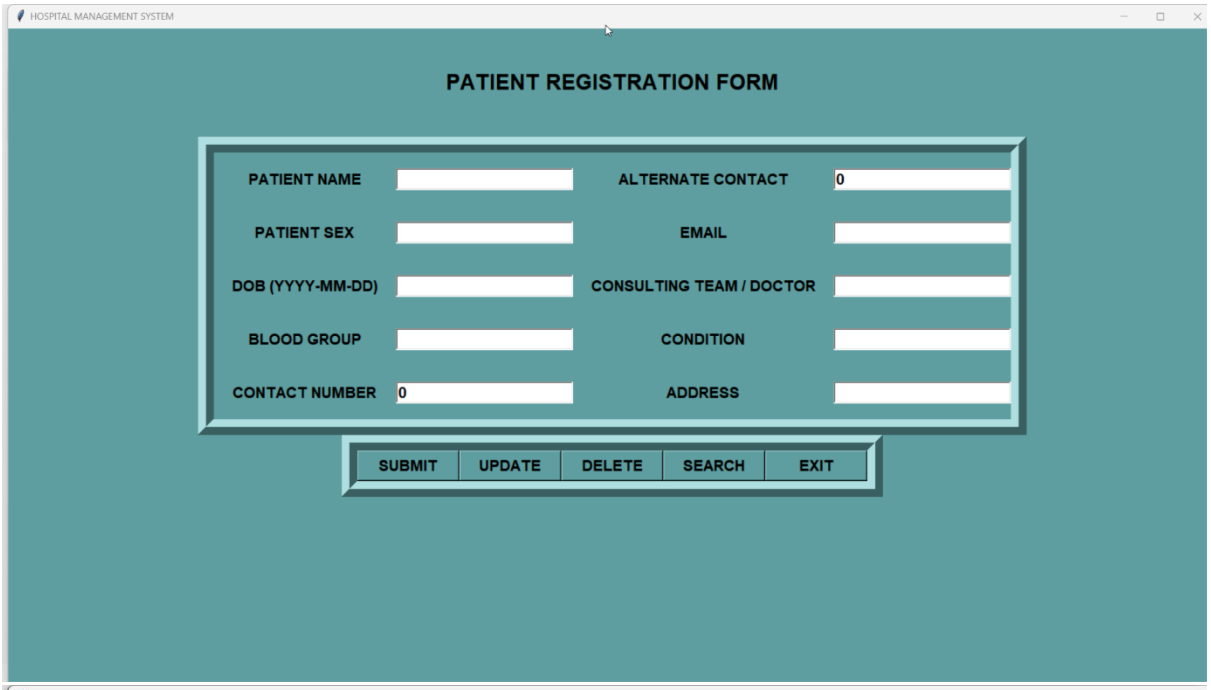
DESCRIPTION AND PRIORITY

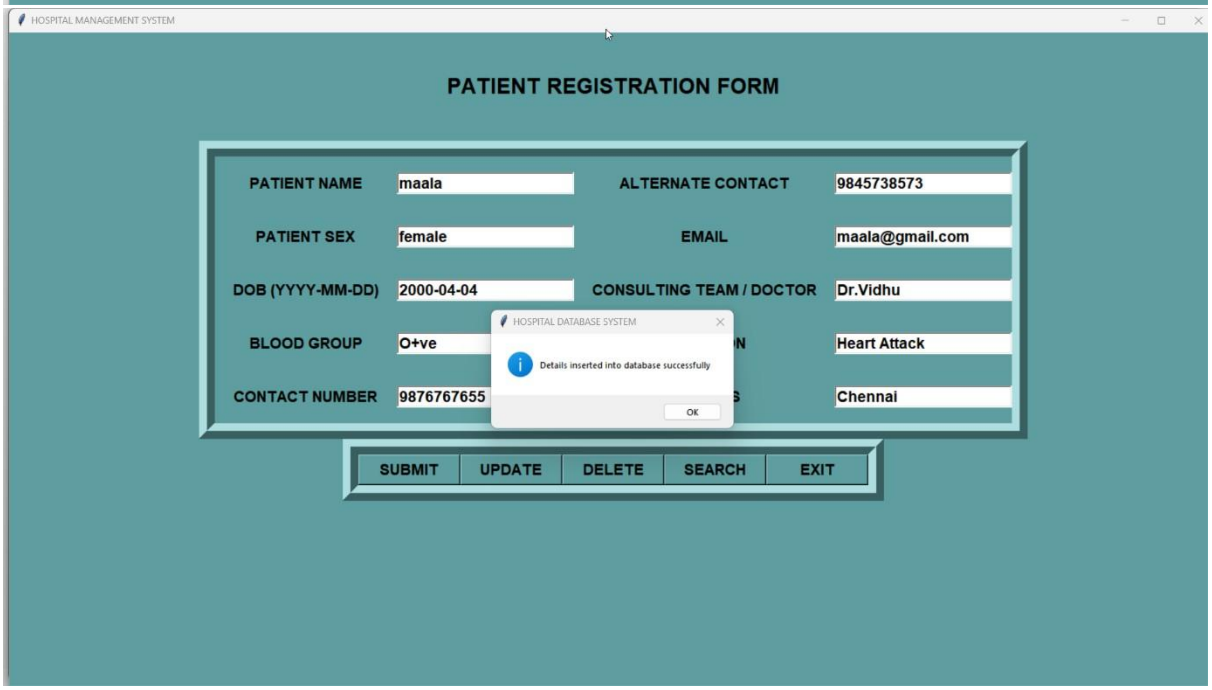
The Hospital Management System plays a crucial role in streamlining and enhancing the efficiency of healthcare facilities. This project holds an even higher priority as it significantly reduces paperwork and eliminates the fear of losing critical information or data. The Hospital Management System automates various administrative and operational tasks within a hospital, such as patient registration, appointment scheduling, billing and invoicing, inventory management, staff management, and medical record-keeping. By digitizing and centralizing these processes, the system minimizes the reliance on manual paperwork, ensuring a more streamlined and accurate workflow. One of the primary advantages of the Hospital Management System is the reduction of paperwork. By eliminating the need for physical forms, registers, and documents, healthcare providers can save time, resources, and storage space. Additionally, the system enables easy retrieval and access to patient information, medical records, and diagnostic reports, facilitating faster decision-making and improving patient care.

RESULTS


CARE & CURE NURSING
CARE & CURE NURSING
Username
admin
Password
****
Login
Exit


HOSPITAL MANAGEMENT SYSTEM
MAIN MENU
1. PATIENT REGISTRATION
2. ROOM ALLOCATION
3. EMPLOYEE REGISTRATION
4. BOOK APPOINTMENT
5. PATIENT BILL
6. VIEW RECORDS
7. DEPARTMENT REGISTRATION
8. EXIT

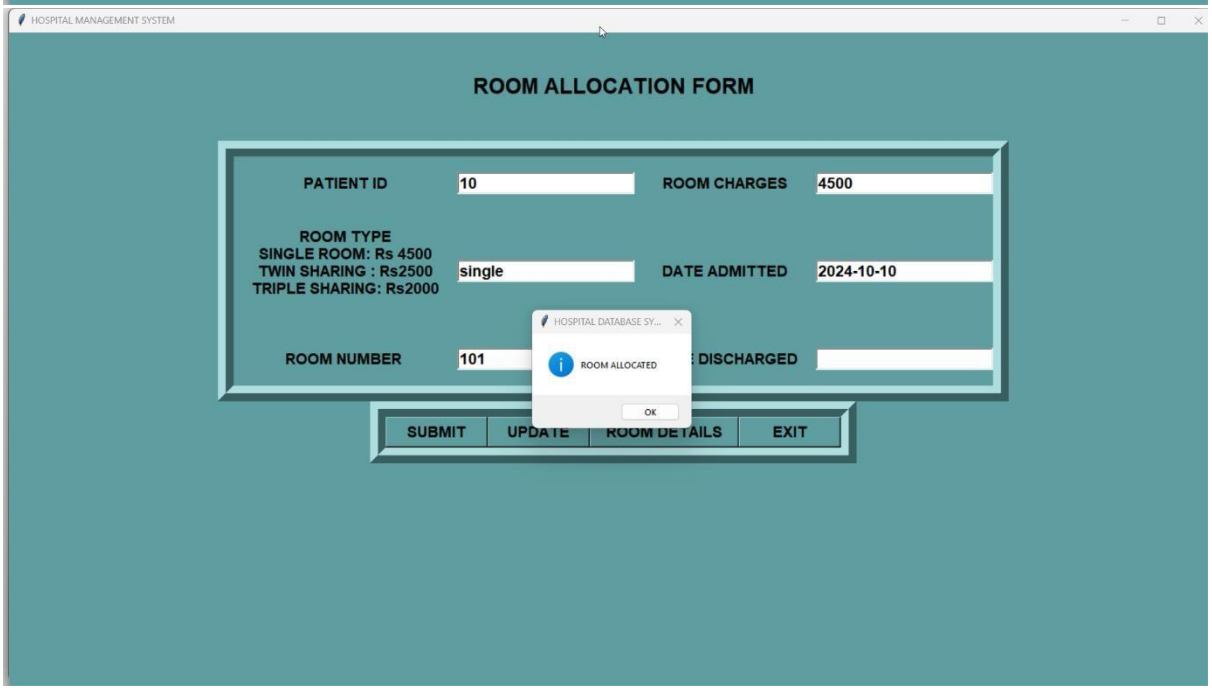

HOSPITAL MANAGEMENT SYSTEM
PATIENT REGISTRATION FORM
PATIENT NAME
PATIENT SEX
DOB (YYYY-MM-DD)
BLOOD GROUP
CONTACT NUMBER
0
ALTERNATE CONTACT
0
EMAIL
CONSULTING TEAM / DOCTOR
CONDITION
ADDRESS
SUBMIT
UPDATE
DELETE
SEARCH
EXIT

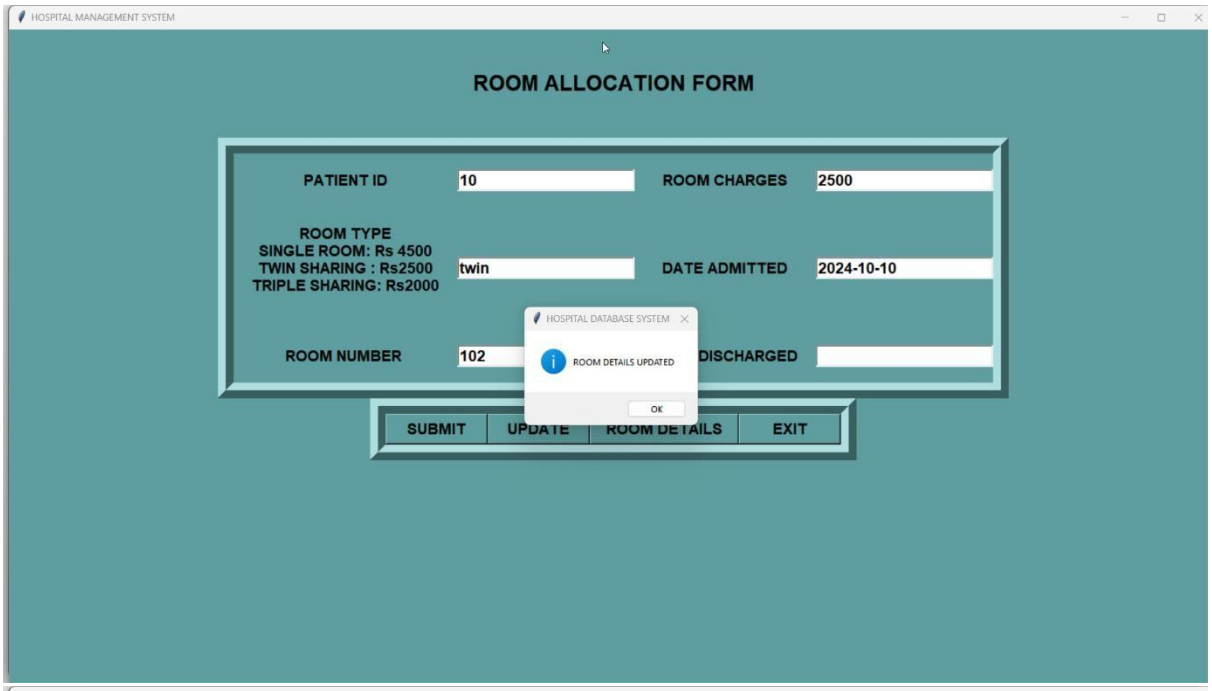

HOSPITAL MANAGEMENT SYSTEM
PATIENT REGISTRATION FORM
PATIENT NAME
maala
PATIENT SEX
female
DOB (YYYY-MM-DD)
2000-04-04
BLOOD GROUP
O+ve
CONTACT NUMBER
9876767655
ALTERNATE CONTACT
9845738573
EMAIL
maala@gmail.com
CONSULTING TEAM / DOCTOR
Dr.Vidhu
Heart Attack
Chennai
HOSPITAL DATABASE SYSTEM
Details inserted into database successfully
OK
SUBMIT
UPDATE
DELETE
SEARCH
EXIT


Hospital Management System
RECORDS MENU
1. PATIENT RECORD
2. ROOM RECORD
3. EMPLOYEE RECORD
4. PATIENT BILL RECORD
5. MEDICINE RECORD
6. OUTPATIENT RECORD
7. TREATMENT RECORD
8. CONSULT RECORD
9. EXIT

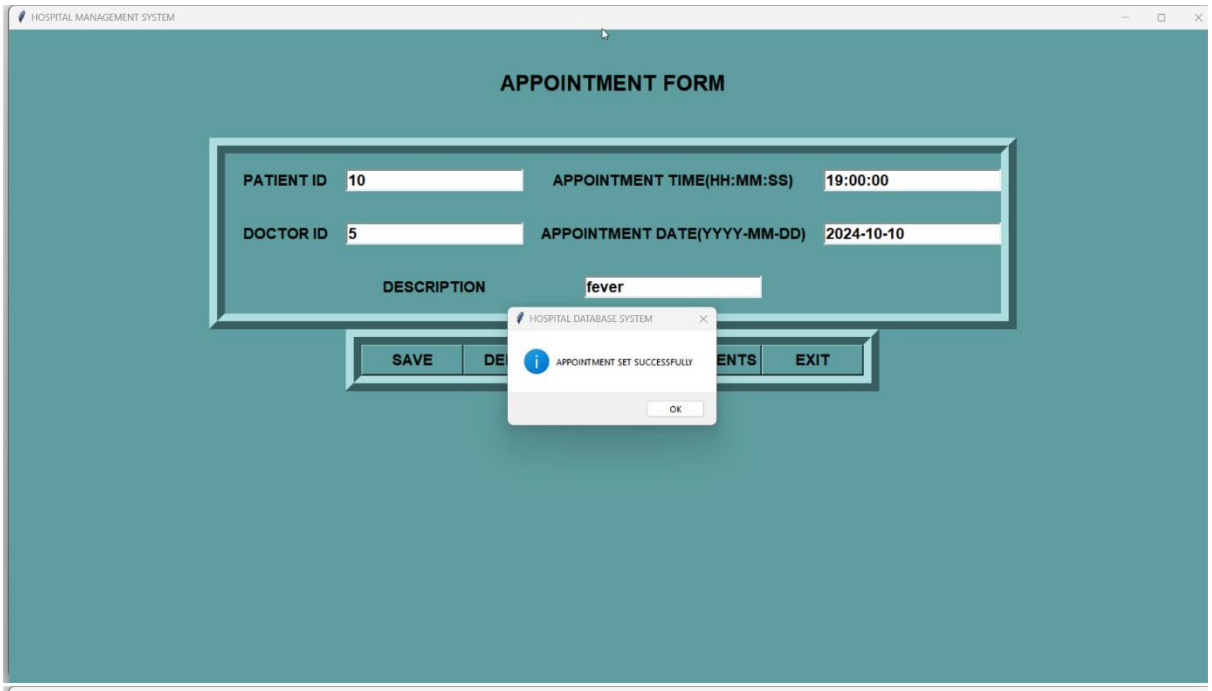

HOSPITAL MANAGEMENT SYSTEM
SEARCH WINDOW
ENTER PATIENT ID TO SEARCH
10
PATIENT ID
10
ADDRESS
Chennai
PATIENT NAME
maala
CONSULTING TEAM / DOCTOR
Dr.Vidhu
SEX
female
EMAIL
maala@gmail.com
DOB (YYYY-MM-DD)
O+ve
CONDITION
Heart Attack
BLOOD GROUP
2000-04-04
CONTACT NO.
9876767655
SEARCH
CLEAR
EXIT


CARE & CURE NURSING
TREATMENT RECORDS
Search By
Patient ID
4
Search
Exit
T_no
Patient_id
Doc_id
Treatment
Treatment_code
T_cost
T_date
9
4
None
Routine Checkup
RC003
100
2024-05-21 14:33:04

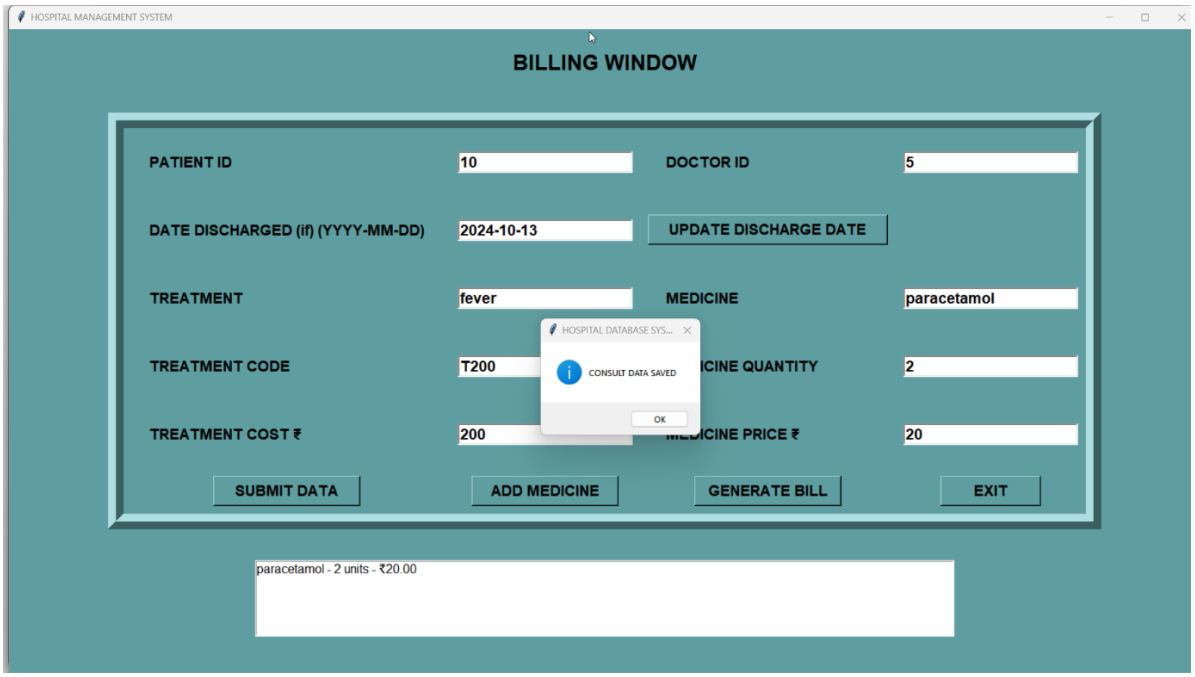

HOSPITAL MANAGEMENT SYSTEM
ROOM ALLOCATION FORM
PATIENT ID
10
ROOM CHARGES
4500
ROOM TYPE
SINGLE ROOM: Rs 4500
TWIN SHARING : Rs2500
TRIPLE SHARING: Rs2000
single
DATE ADMITTED
2024-10-10
ROOM NUMBER
101
DISCHARGED
HOSPITAL DATABASE SY...
ROOM ALLOCATED
OK
SUBMIT
UPDATE
ROOM DETAILS
EXIT


HOSPITAL MANAGEMENT SYSTEM
ROOM ALLOCATION FORM
PATIENT ID
10
ROOM CHARGES
2500
ROOM TYPE
SINGLE ROOM: Rs 4500
TWIN SHARING : Rs2500
TRIPLE SHARING: Rs2000
twin
DATE ADMITTED
2024-10-10
HOSPITAL DATABASE SYSTEM
ROOM DETAILS UPDATED
OK
ROOM NUMBER
102
DISCHARGED
SUBMIT
UPDATE
ROOM DETAILS
EXIT


HOSPITAL MANAGEMENT SYSTEM
EMPLOYEE REGISTRATION FORM
EMPLOYEE NAME
SALARY
0
SEX
EXPERIENCE
AGE
0
CONTACT NUMBER
EMPLOYEE DESIGNATION [DOCTOR, NURSE, RECEPTIONIST]
EMAIL
SUBMIT
UPDATE
DELETE
SEARCH
EXIT


HOSPITAL MANAGEMENT SYSTEM
APPOINTMENT FORM
PATIENT ID
10
APPOINTMENT TIME(HH:MM:SS)
19:00:00
DOCTOR ID
5
APPOINTMENT DATE(YYYY-MM-DD)
2024-10-10
DESCRIPTION
fever
HOSPITAL DATABASE SYSTEM
APPOINTMENT SET SUCCESSFULLY
OK
SAVE
EXIT


HOSPITAL MANAGEMENT SYSTEM
SEARCH APPOINTMENT WINDOW
PATIENT ID
PATIENT NAME
APPOINTMENT NO
DOCTOR ID
DOCTOR NAME
APPOINTMENT DATE
APPOINTMENT TIME
10
maala
10
5
Michael Brown
2024-10-10
19:00:00


HOSPITAL MANAGEMENT SYSTEM
BILLING WINDOW
PATIENT ID
10
DOCTOR ID
5
DATE DISCHARGED (if) (YYYY-MM-DD)
2024-10-13
UPDATE DISCHARGE DATE
TREATMENT
fever
MEDICINE
paracetamol
TREATMENT CODE
T200
ICINE QUANTITY
2
TREATMENT COST ₹
200
MEDICINE PRICE ₹
20
HOSPITAL DATABASE SYS...
CONSULT DATA SAVED
OK
SUBMIT DATA
ADD MEDICINE
GENERATE BILL
EXIT
paracetamol - 2 units - ₹20.00

FRONT-END AND BACK-END CODES

FRONT-END PROGRAMMING CODE – PYTHON TKINTER

LOGIN.PY

```
from tkinter import *
import tkinter.messagebox
from menu import Menu
from menu_reception import Menu_r

class MainWindow:
    def __init__(self, master):
        self.master = master
        self.master.title("CARE & CURE NURSING")
        self.master.geometry("1600x1000+0+0")
        self.master.config(bg="cadet blue")
        self.frame = Frame(self.master,bg="cadet blue")
        self.frame.pack()

        self.Username = StringVar()
        self.Password = StringVar()

        self.create_widgets()

    def create_widgets(self):
        lbl_title = Label(self.frame, text="CARE & CURE NURSING",
font="Helvetica 20 bold", bg="powder blue", fg="black")
        lbl_title.grid(row=0, column=0, columnspan=2, pady=40)

        login_frame1 = Frame(self.frame, width=400, height=80, relief="ridge",
bg="cadet blue", bd=20)
        login_frame1.grid(row=1, column=0)
        login_frame2 = Frame(self.frame, width=400, height=80, relief="ridge",
bg="cadet blue", bd=20)
        login_frame2.grid(row=2, column=0)

        lbl_username = Label(login_frame1, text="Username", font="Helvetica 14
bold", bg="cadet blue", bd=22)
        lbl_username.grid(row=0, column=0)
        entry_username = Entry(login_frame1, font="Helvetica 14 bold",
textvariable=self.Username, bd=2)
        entry_username.grid(row=0, column=1)
```

```
        lbl_password = Label(login_frame1, text="Password", font="Helvetica 14
bold", bg="cadet blue", bd=22)
        lbl_password.grid(row=1, column=0)
        entry_password = Entry(login_frame1, font="Helvetica 14 bold",
show="*", textvariable=self.Password, bd=2)
        entry_password.grid(row=1, column=1)

        btn_login = Button(login_frame2, text="Login", font="Helvetica 10
bold", width=10, bg="powder blue", command=self.login_system)
        btn_login.grid(row=3, column=0)
        btn_exit = Button(login_frame2, text="Exit", font="Helvetica 10 bold",
width=10, bg="powder blue", command=self.exit)
        btn_exit.grid(row=3, column=1)

    def login_system(self):
        username = self.Username.get()
        password = self.Password.get()
        if username == 'admin' and password == '1234':
            self.open_menu()
        elif username == 'reception' and password == '4321':
            self.open_menu_r()
        else:
            tkinter.messagebox.askretrycancel("CARE & CURE NURSING", "PLEASE
ENTER VALID USERNAME AND PASSWORD")

    def open_menu(self):
        self.new_window = Toplevel(self.master)
        self.app = Menu(self.new_window,self.master)
        self.master.withdraw()

    def open_menu_r(self):
        self.new_window = Toplevel(self.master)
        self.app = Menu_r(self.new_window,self.master)
        self.master.withdraw()


    def exit(self):
        self.master.destroy()

def main():
    root = Tk()
    app = MainWindow(root)
    root.mainloop()

if __name__== "__main__":
    main()
```

PATIENT_FORM.PY

```
from tkinter import *
import tkinter.messagebox
from tkinter import ttk
from tkinter import font
import mysql.connector

conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Lights@123",
    database="HMS"
)

#root = Tk()
cursor = conn.cursor()
print("DATABASE CONNECTION SUCCESSFUL")


# PATIENT FORM
class Patient:
    def __init__(self, master,main_window):
        self.master = master
        self.main_window = main_window
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x1000+0+0")
        self.master.config(bg="cadet blue")
        self.frame = Frame(self.master, bg="cadet blue")
        self.frame.pack()

        # =============ATTRIBUTES===========

        self.pat_name = StringVar()
        self.pat_dob = StringVar()
        self.pat_address = StringVar()
        self.pat_sex = StringVar()
        self.pat_BG = StringVar()
        self.pat_email = StringVar()
        self.pat_contact = IntVar()
        self.pat_contactalt = IntVar()
        self.pat_CT = StringVar()
        self.pat_C = StringVar()


        # ===============TITLE==========
        self.lblTitle = Label(self.frame, text="PATIENT REGISTRATION FORM",
font="Helvetica 20 bold", bg="cadet blue")
        self.lblTitle.grid(row=0, column=0, columnspan=2, pady=50)
```

```
        # ==============FRAME==========
        self.LoginFrame = Frame(self.frame, width=400, height=80,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame.grid(row=1, column=0)

        self.LoginFrame2 = Frame(self.frame, width=400, height=80,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame2.grid(row=2, column=0)
        # ===========LABELS=============
        self.lblpatid = Label(self.LoginFrame, text="PATIENT NAME",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblpatid.grid(row=0, column=0)
        self.lblpatid  = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.pat_name)
        self.lblpatid.grid(row=0, column=1)

        self.lblPatname = Label(self.LoginFrame, text="PATIENT SEX",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblPatname.grid(row=1, column=0)
        self.lblPatname  = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.pat_sex)
        self.lblPatname.grid(row=1, column=1)

        self.lblsex = Label(self.LoginFrame, text="DOB (YYYY-MM-DD)",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblsex.grid(row=2, column=0)
        self.lblsex  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2,
textvariable=self.pat_dob)
        self.lblsex.grid(row=2, column=1)

        self.lblDOB = Label(self.LoginFrame, text="BLOOD GROUP",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblDOB.grid(row=3, column=0)
        self.lblDOB  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2,
textvariable=self.pat_BG)
        self.lblDOB.grid(row=3, column=1)

        self.lblbgrp = Label(self.LoginFrame, text="CONTACT NUMBER",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblbgrp.grid(row=4, column=0)
        self.lblbgrp  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2,
textvariable=self.pat_contact)
        self.lblbgrp.grid(row=4, column=1)

        self.lblCon = Label(self.LoginFrame, text="ALTERNATE CONTACT",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblCon.grid(row=0, column=2)
```

```
        self.lblCon  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2,
textvariable=self.pat_contactalt)
        self.lblCon.grid(row=0, column=3)

        self.lblAlt = Label(self.LoginFrame, text="EMAIL", font="Helvetica 14
bold", bg="cadet blue", bd=22)
        self.lblAlt.grid(row=1, column=2)
        self.lblAlt  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2,
textvariable=self.pat_email)
        self.lblAlt.grid(row=1, column=3)

        self.lbleid = Label(self.LoginFrame, text="CONSULTING TEAM / DOCTOR",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lbleid.grid(row=2, column=2)
        self.lbleid  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2,
textvariable=self.pat_CT)
        self.lbleid.grid(row=2, column=3)

        self.lbldoc = Label(self.LoginFrame, text="CONDITION", font="Helvetica
14 bold", bg="cadet blue", bd=22)
        self.lbldoc.grid(row=3, column=2)
        self.lbldoc  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2,
textvariable=self.pat_C)
        self.lbldoc.grid(row=3, column=3)

        self.lbladd = Label(self.LoginFrame, text="ADDRESS", font="Helvetica
14 bold", bg="cadet blue", bd=22)
        self.lbladd.grid(row=4, column=2)
        self.lbladd  = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2,
textvariable=self.pat_address)
        self.lbladd.grid(row=4, column=3)

        self.button2 = Button(self.LoginFrame2, text="SUBMIT", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.INSERT_PAT)
        self.button2.grid(row=3, column=1)

        self.button3 = Button(self.LoginFrame2, text="UPDATE", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.UPDATE)
        self.button3.grid(row=3, column=2)

        self.button4 = Button(self.LoginFrame2, text="DELETE", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.D_DISPLAY)
        self.button4.grid(row=3, column=3)

        self.button5 = Button(self.LoginFrame2, text="SEARCH", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.S_DISPLAY)
        self.button5.grid(row=3, column=4)
```

```
        self.button6 = Button(self.LoginFrame2, text="EXIT", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.Exit)
        self.button6.grid(row=3, column=5)



    def clear(self):
        self.lblpatid.delete(0, 'end')
        self.lblPatname.delete(0, 'end')
        self.lblsex.delete(0, 'end')
        self.lblDOB.delete(0, 'end')
        self.lblbgrp.delete(0, 'end')
        self.lblCon.delete(0, 'end')
        self.lblAlt.delete(0, 'end')
        self.lbleid.delete(0, 'end')
        self.lbldoc.delete(0, 'end')
        self.lbladd.delete(0, 'end')

        # Clear the corresponding fields in the database table
        #query = "DELETE FROM your_table_name"
        #cursor.execute(query)







    #INSERT DATA IN PATIENT FORM
    def INSERT_PAT(self):
        try:
            conn = mysql.connector.connect(
                host="localhost",
                user="root",
                password="Lights@123",
                database="HMS"
            )

            p11 = self.pat_C.get()
            p2 = self.pat_name.get()
            p3 = self.pat_sex.get()
            p4 = self.pat_BG.get()
            p5 = self.pat_dob.get()
            p6 =  self.pat_contact.get()
            p7 = self.pat_contactalt.get()
            p8 =  self.pat_address.get()
            p9 = self.pat_CT.get()
            p10 = self.pat_email.get()

            cursor = conn.cursor()
```

```
            # Check if patient already exists
            cursor.execute('SELECT * FROM PATIENT WHERE NAME = %s AND EMAIL =
%s', (p2, p10))
            p = cursor.fetchall()
            x = len(p)

            if x != 0:
                tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM",
"Patient already exists")
            else:
                # Insert patient details
                cursor.execute('INSERT INTO PATIENT (NAME, SEX, BLOOD_GROUP,
DOB, ADDRESS, CONSULT_TEAM, EMAIL, `CONDITION`) VALUES (%s, %s, %s, %s, %s,
%s, %s, %s)',
                               (p2, p3, p4, p5, p8, p9, p10, p11))

                # Get the newly inserted patient_id
                cursor.execute('SELECT PATIENT_ID FROM PATIENT WHERE NAME = %s
AND EMAIL = %s', (p2, p10))
                p1 = cursor.fetchone()[0]

                # Insert contact numbers
                cursor.execute('INSERT INTO CONTACT_NO (PATIENT_ID, CONTACTNO,
ALT_CONTACT) VALUES (%s, %s, %s)', (p1, p6, p7,))

                tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM",
"Details inserted into database successfully")

            self.clear()
            conn.commit()
        except mysql.connector.Error as err:
            tkinter.messagebox.showerror("Database Error", f"Error: {err}")




    def Exit(self):
        self.master.destroy()
        self.main_window.deiconify()

    def D_DISPLAY(self):
        self.master.withdraw()
        self.newWindow = Toplevel(self.master)
        self.app = DMenu(self.newWindow,self.master)

    def S_DISPLAY(self):
```

```
        self.master.withdraw()
        self.newWindow = Toplevel(self.master)
        self.app = SMenu(self.newWindow,self.master)

    def UPDATE(self):
        self.master.withdraw()
        self.newWindow = Toplevel(self.master)
        self.app = U_PAT(self.newWindow,self.master)
```

APPOINTMENT_FORM.PY

```
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
from tkinter import font
import mysql.connector
from mysql.connector import Error
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Lights@123",
    database = "HMS"
)
#root = Tk()
cursor=conn.cursor()
print("DATABASE CONNECTION SUCCESSFUL")
class Appointment:
    def __init__(self,master,main_window):
        self.master = master
        self.main_window = main_window
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x1000+0+0")
        self.master.config(bg="cadet blue")
        self.frame = Frame(self.master,bg="cadet blue")
        self.frame.pack()

        #=============ATTRIBUTES===========

        self.pat_ID=IntVar()
        self.emp_ID=StringVar()
        self.ap_no=StringVar()
        self.ap_time=StringVar()
        self.ap_date=StringVar()
        self.des=StringVar()

        #===============TITLE==========
```

```
        self.lblTitle = Label(self.frame,text = "APPOINTMENT FORM",
font="Helvetica 20 bold",bg="cadet blue")
        self.lblTitle.grid(row =0 ,column = 0,columnspan=2,pady=50)
        #==============FRAME==========
        self.LoginFrame =
Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
        self.LoginFrame.grid(row=1,column=0)

        self.LoginFrame2 =
Frame(self.frame,width=400,height=80,relief="ridge",bg="cadet blue",bd=20)
        self.LoginFrame2.grid(row=2,column=0)
        #===========LABELS=============
        self.lblpid = Label(self.LoginFrame,text="PATIENT ID",font="Helvetica
14 bold",bg="cadet blue",bd=22)
        self.lblpid.grid(row=0,column=0)
        self.lblpid  = Entry(self.LoginFrame,font="Helvetica 14
bold",bd=2,textvariable= self.pat_ID)
        self.lblpid.grid(row=0,column=1)

        self.lbldid = Label(self.LoginFrame,text="DOCTOR ID",font="Helvetica
14 bold",bg="cadet blue",bd=22)
        self.lbldid.grid(row=1,column=0)
        self.lbldid  = Entry(self.LoginFrame,font="Helvetica 14
bold",bd=2,textvariable=self.emp_ID )
        self.lbldid.grid(row=1,column=1)

        self.lblapt = Label(self.LoginFrame,text="APPOINTMENT
TIME(HH:MM:SS)",font="Helvetica 14 bold",bg="cadet blue",bd=22)
        self.lblapt.grid(row=0,column=2)
        self.lblapt  = Entry(self.LoginFrame,font="Helvetica 14
bold",bd=2,textvariable=self.ap_time )
        self.lblapt.grid(row=0,column=3)

        self.lblapd = Label(self.LoginFrame,text="APPOINTMENT DATE(YYYY-MM-
DD)",font="Helvetica 14 bold",bg="cadet blue",bd=22)
        self.lblapd.grid(row=1,column=2)
        self.lblapd  = Entry(self.LoginFrame,font="Helvetica 14
bold",bd=2,textvariable= self.ap_date)
        self.lblapd.grid(row=1,column=3)

        self.lbldes = Label(self.LoginFrame,text="DESCRIPTION",font="Helvetica
14 bold",bg="cadet blue",bd=22)
        self.lbldes.grid(row=2,column=1)
        self.lbldes  = Entry(self.LoginFrame,font="Helvetica 14
bold",bd=2,textvariable=self.des)
        self.lbldes.grid(row=2,column=2)
```

```
        self.button2 = Button(self.LoginFrame2, text="SAVE",width
=10,font="Helvetica 14 bold",bg="cadet blue",command = self.INSERT_AP)
        self.button2.grid(row=3,column=1)

        self.button3 = Button(self.LoginFrame2, text="DELETE",width
=10,font="Helvetica 14 bold",bg="cadet blue",command= self.DE_AP_DISPLAY)
        self.button3.grid(row=3,column=2)



        self.button3 = Button(self.LoginFrame2, text="SEARCH
APPOINTMENTS",width =20,font="Helvetica 14 bold",bg="cadet blue",command=
self.S_AP_DISPLAY)
        self.button3.grid(row=3,column=3)

        self.button6 = Button(self.LoginFrame2, text="EXIT",width
=10,font="Helvetica 14 bold",bg="cadet blue",command = self.Exit)
        self.button6.grid(row=3,column=4)

    def Exit(self):
        self.master.destroy()
        self.main_window.deiconify()

    def INSERT_AP(self):
        global e1, e2, e3, e4, e5, e6

        e2 = self.emp_ID.get()
        e3 = self.pat_ID.get()
        e4 = self.ap_time.get()
        e5 = self.ap_date.get()
        e6 = self.des.get()

        try:
            conn = mysql.connector.connect(
                host="localhost",
                user="root",
                password="Lights@123",
                database="HMS"
            )
            cursor = conn.cursor()

            cursor.execute(
                "SELECT AP_NO FROM appointment WHERE AP_DATE = %s AND
PATIENT_ID = %s AND EMP_ID = %s AND AP_TIME = %s",
                (e5, e3, e2, e4)
            )
            e1 = cursor.fetchall()

            if e1:  # Check if the appointment exists
```

```
                Tk.messagebox.showerror("HOSPITAL DATABASE SYSTEM",
"APPOINTMENT ALREADY EXISTS")
            else:
                cursor.execute(
                    "INSERT INTO appointment (PATIENT_ID, EMP_ID, AP_DATE,
AP_TIME, DESCRIPTION) VALUES (%s, %s, %s, %s, %s)",
                    (e3, e2, e5, e4, e6)
                )
                messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "APPOINTMENT
SET SUCCESSFULLY")

            conn.commit()

        except Error as e:
            messagebox.showerror("HOSPITAL DATABASE SYSTEM", f"Error: {e}")

        finally:
            if conn.is_connected():
                cursor.close()
                conn.close()

    def DE_AP_DISPLAY(self):
        self.newWindow = Toplevel(self.master)
        self.app = DEL_AP(self.newWindow)

    def S_AP_DISPLAY(self):
        self.newWindow = Toplevel(self.master)
        self.app = SEA_AP(self.newWindow)
```

BILLING_FORM.PY

```
import mysql.connector
from mysql.connector import Error
from tkinter import *
from tkinter import ttk, messagebox

class Billing:
    def __init__(self, master, main_window):
        self.master = master
        self.main_window = main_window
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x1000+0+0")
        self.master.config(bg="cadet blue")
        self.frame = Frame(self.master, bg="cadet blue")
        self.frame.pack()

        self.initialize_variables()
```

```
        self.create_widgets()

    def initialize_variables(self):
        self.P_id = IntVar()
        self.dd = StringVar()
        self.treat_1 = StringVar()
        self.treat_2 = StringVar()
        self.cost_t = IntVar()
        self.med = StringVar()
        self.med_q = IntVar()
        self.price = DoubleVar()
        self.D_id = IntVar()  # New variable for DOC_ID

    def create_widgets(self):
        self.lblTitle = Label(self.frame, text="BILLING WINDOW",
font="Helvetica 20 bold", bg="cadet blue")
        self.lblTitle.grid(row=0, column=0, columnspan=4, pady=25)

        self.LoginFrame = Frame(self.frame, width=800, height=400,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame.grid(row=1, column=0, padx=50, pady=20)

        self.lblpid = Label(self.LoginFrame, text="PATIENT ID",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblpid.grid(row=0, column=0, padx=10, pady=10, sticky='w')

        self.lblpid_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.P_id)
        self.lblpid_entry.grid(row=0, column=1, padx=10, pady=10)

        self.lbldocid = Label(self.LoginFrame, text="DOCTOR ID",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lbldocid.grid(row=0, column=2, padx=10, pady=10, sticky='w')

        self.lbldocid_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.D_id)
        self.lbldocid_entry.grid(row=0, column=3, padx=10, pady=10)

        self.lbldid = Label(self.LoginFrame, text="DATE DISCHARGED (if) (YYYY-
MM-DD)", font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lbldid.grid(row=1, column=0, padx=10, pady=10, sticky='w')

        self.lbldid_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.dd)
        self.lbldid_entry.grid(row=1, column=1, padx=10, pady=10)

        self.button2 = Button(self.LoginFrame, text="UPDATE DISCHARGE DATE",
width=25, font="Helvetica 14 bold", bg="cadet blue", command=self.UPDATE_DATE)
```

```
        self.button2.grid(row=1, column=2, padx=10, pady=10)

        self.lbltreat = Label(self.LoginFrame, text="TREATMENT",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lbltreat.grid(row=2, column=0, padx=10, pady=10, sticky='w')

        self.lbltreat_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.treat_1)
        self.lbltreat_entry.grid(row=2, column=1, padx=10, pady=10)

        self.lblcode_t1 = Label(self.LoginFrame, text="TREATMENT CODE",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblcode_t1.grid(row=3, column=0, padx=10, pady=10, sticky='w')

        self.lblcode_t1_entry = Entry(self.LoginFrame, font="Helvetica 14
bold", bd=2, textvariable=self.treat_2)
        self.lblcode_t1_entry.grid(row=3, column=1, padx=10, pady=10)

        self.lblap = Label(self.LoginFrame, text="TREATMENT COST ₹",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblap.grid(row=4, column=0, padx=10, pady=10, sticky='w')

        self.lblap_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.cost_t)
        self.lblap_entry.grid(row=4, column=1, padx=10, pady=10)

        self.lblmed = Label(self.LoginFrame, text="MEDICINE", font="Helvetica
14 bold", bg="cadet blue", bd=22)
        self.lblmed.grid(row=2, column=2, padx=10, pady=10, sticky='w')

        self.lblmed_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.med)
        self.lblmed_entry.grid(row=2, column=3, padx=10, pady=10)

        self.med_t1 = Label(self.LoginFrame, text="MEDICINE QUANTITY",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.med_t1.grid(row=3, column=2, padx=10, pady=10, sticky='w')

        self.med_t1_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.med_q)
        self.med_t1_entry.grid(row=3, column=3, padx=10, pady=10)

        self.lblapd = Label(self.LoginFrame, text="MEDICINE PRICE ₹",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblapd.grid(row=4, column=2, padx=10, pady=10, sticky='w')

        self.lblapd_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.price)
```

```
        self.lblapd_entry.grid(row=4, column=3, padx=10, pady=10)

        self.button3 = Button(self.LoginFrame, text="SUBMIT DATA", width=15,
font="Helvetica 14 bold", bg="cadet blue", command=self.UPDATE_DATA)
        self.button3.grid(row=5, column=0, padx=10, pady=10)

        self.button4 = Button(self.LoginFrame, text="ADD MEDICINE", width=15,
font="Helvetica 14 bold", bg="cadet blue", command=self.ADD_MEDICINE)
        self.button4.grid(row=5, column=1, padx=10, pady=10)

        self.button5 = Button(self.LoginFrame, text="GENERATE BILL", width=15,
font="Helvetica 14 bold", bg="cadet blue", command=self.GEN_BILL)
        self.button5.grid(row=5, column=2, padx=10, pady=10)

        self.button6 = Button(self.LoginFrame, text="EXIT", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.Exit)
        self.button6.grid(row=5, column=3, padx=10, pady=10)

        self.listbox = Listbox(self.frame, width=100, height=5,
font=("Helvetica", 12))
        self.listbox.grid(row=2, column=0, columnspan=4, padx=50, pady=20)

    def Exit(self):
        self.master.destroy()
        self.main_window.deiconify()

    def db_connect(self):
        try:
            conn = mysql.connector.connect(
                host="localhost",
                user="root",
                password="Lights@123",
                database="HMS"
            )
            return conn
        except mysql.connector.Error as e:
            messagebox.showerror("Database Error", f"Error connecting to MySQL
database:\n{e}")
            return None

    def UPDATE_DATE(self):
        b1 = self.P_id.get()
        b2 = self.dd.get()
        conn = self.db_connect()
        if conn:
            try:
                cursor = conn.cursor()
```

```
                cursor.execute("UPDATE ROOM SET DATE_DISCHARGED=%s WHERE
PATIENT_ID=%s", (b2, b1))
                messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "DISCHARGE
DATE UPDATED")
                conn.commit()
                cursor.close()
            except mysql.connector.Error as e:
                messagebox.showerror("Database Error", f"MySQL Error: {e}")
            finally:
                conn.close()

    def UPDATE_DATA(self):
        b1 = self.P_id.get()
        b2 = self.D_id.get()
        b3 = self.treat_1.get()
        b4 = self.treat_2.get()
        b5 = self.cost_t.get()

        conn = self.db_connect()
        if conn:
            c1 = conn.cursor()
            try:
                c1.execute("SELECT * FROM CONSULT WHERE PATIENT_ID=%s AND
EMP_ID = %s", (b1,b2))
                p = c1.fetchall()
                if len(p) == 0:
                    c1.execute("INSERT INTO CONSULT (PATIENT_ID, EMP_ID)
VALUES (%s, %s)",
                               (b1, b2))
                    conn.commit()
                    messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "CONSULT
DATA SAVED")

                c1.execute("INSERT INTO TREATMENT (PATIENT_ID, DOC_ID,
TREATMENT, TREATMENT_CODE, T_COST) VALUES (%s, %s, %s, %s, %s)",
                           (b1, b2, b3, b4, b5))
                conn.commit()

                # Insert medicines into MEDICINE table
                medicine_list = self.listbox.get(0, END)
                for item in medicine_list:
                    parts = item.split(" - ")
                    med_name = parts[0]
                    quantity = int(parts[1].split()[0])
                    price = float(parts[2][1:])

                    try:
                        # Insert into MEDICINE table
```

```
                        c1.execute("INSERT INTO MEDICINE (PATIENT_ID,
MEDICINE_NAME, M_QTY, M_COST) VALUES (%s, %s, %s, %s)",
                                   (b1, med_name, quantity, price))
                        conn.commit()
                    except mysql.connector.Error as err:
                        messagebox.showerror("HOSPITAL DATABASE SYSTEM",
f"MySQL Error: {err}")

                messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "BILLING DATA
SAVED")
            except mysql.connector.Error as e:
                messagebox.showerror("Database Error", f"MySQL Error: {e}")
            finally:
                c1.close()
                conn.close()

    def ADD_MEDICINE(self):
        price = 0
        med_name = self.med.get()
        med_quantity = self.med_q.get()
        med_price = self.price.get()
        price += med_price
        if med_name and med_quantity and med_price:
            item = f"{med_name} - {med_quantity} units - ₹{med_price:.2f}"
            self.listbox.insert(END, item)
        else:
            messagebox.showerror("Error", "Please enter all fields for
medicine.")

    def GEN_BILL(self):
        b1 = self.P_id.get()
        conn = self.db_connect()
        if conn:
            try:
                rate = 0
                cursor = conn.cursor()

                # Fetch room rate for the patient

                cursor.execute("SELECT RATE FROM ROOM WHERE PATIENT_ID = %s
AND PAY_STATUS = 'NO'", (b1,))
                room_data = cursor.fetchone()
                if room_data:
                    rate = room_data[0]

                # Calculate total amount including treatment and medicine
costs
```

```
                cursor.execute("SELECT T_COST FROM TREATMENT WHERE PATIENT_ID
= %s AND DATE(T_DATE) = CURRENT_DATE ORDER BY T_DATE DESC LIMIT 1", (b1,))
                treatment_cost_data = cursor.fetchone()
                treatment_cost = treatment_cost_data[0] if
treatment_cost_data[0] else 0

                cursor.execute("SELECT (M_COST * M_QTY) FROM MEDICINE WHERE
PATIENT_ID = %s AND DATE(M_DATE) = CURRENT_DATE ORDER BY M_DATE DESC LIMIT 1
", (b1,))
                medicine_cost_data = cursor.fetchone()
                medicine_cost = medicine_cost_data[0] if medicine_cost_data[0]
else 0

                total_amount = treatment_cost + rate + medicine_cost

                cursor.execute("INSERT INTO BILL (PATIENT_ID, BILL) VALUES
(%s, %s)", (b1, total_amount))

                # Display total amount in listbox
                self.listbox.delete(0, END)
                self.listbox.insert(END, f"Total Amount: ₹{total_amount:.2f}")

                if rate != 0:
                    cursor.execute("UPDATE ROOM SET PAY_STATUS = 'YES' WHERE
PATIENT_ID = %s", (b1,))
                    messagebox.showinfo("HOSPITAL MANAGEMENT", "TOTAL AMOUNT
INCLUDES ROOM CHARGES")

                conn.commit()
            except mysql.connector.Error as e:
                messagebox.showerror("Database Error", f"MySQL Error: {e}")
            finally:
                cursor.close()
                conn.close()
```

DEPARTMENT-FORM.PY

```
from tkinter import *
import tkinter.messagebox
import mysql.connector

conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Lights@123",
    database="HMS"
```

```
)

cursor = conn.cursor()
print("DATABASE CONNECTION SUCCESSFUL")

class dept:
    def __init__(self, master,main_window):
        self.master = master
        self.main_window = main_window
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x1000+0+0")
        self.master.config(bg="cadet blue")
        self.frame = Frame(self.master, bg="cadet blue")
        self.frame.pack()

        # Attributes
        #self.dep_ID = StringVar()
        self.dep_name = StringVar()
        self.dep_head = StringVar()
        self.dep_fund = StringVar()

        # Title
        self.lblTitle = Label(self.frame, text="DEPARTMENT REGISTRATION FORM",
font="Helvetica 20 bold", bg="cadet blue")
        self.lblTitle.grid(row=0, column=0, columnspan=2, pady=50)

        # Frame 1
        self.LoginFrame = Frame(self.frame, width=400, height=80,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame.grid(row=1, column=0)

        # Frame 2
        self.LoginFrame2 = Frame(self.frame, width=400, height=80,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame2.grid(row=2, column=0)

        # Labels and Entries
        #self.lbldepid = Label(self.LoginFrame, text="DEPARTMENT ID",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        #self.lbldepid.grid(row=0, column=0)
        #self.lbldpid = Entry(self.LoginFrame, font="Helvetica 14 bold", bd=2,
textvariable=self.dep_ID)
        #self.lbldpid.grid(row=0, column=1)

        self.lbldepname = Label(self.LoginFrame, text="DEPARTMENT NAME",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lbldepname.grid(row=1, column=0)
```

```
        self.lbldpname = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.dep_name)
        self.lbldpname.grid(row=1, column=1)

        self.lbldephead = Label(self.LoginFrame, text="DEPARTMENT HEAD",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lbldephead.grid(row=2, column=0)
        self.lbldphead = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.dep_head)
        self.lbldphead.grid(row=2, column=1)

        self.lbldepfund = Label(self.LoginFrame, text="DEPARTMENT FUND",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lbldepfund.grid(row=3, column=0)
        self.lbldpfund = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.dep_fund)
        self.lbldpfund.grid(row=3, column=1)

        # Buttons
        self.button2 = Button(self.LoginFrame2, text="SAVE", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.INSERT_DEP)
        self.button2.grid(row=3, column=1)

        self.button3 = Button(self.LoginFrame2, text="DELETE", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.DELETE_DEP_DISPLAY)
        self.button3.grid(row=3, column=2)

        self.button4 = Button(self.LoginFrame2, text="UPDATE", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.UPDATE_DEP_DISPLAY)
        self.button4.grid(row=3, column=3)

        self.button6 = Button(self.LoginFrame2, text="EXIT", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.Exit)
        self.button6.grid(row=3, column=4)

    def Exit(self):
        self.master.destroy()
        self.main_window.deiconify()

    def INSERT_DEP(self):
        global d1, d2, d3, d4


        d2 = self.dep_name.get().capitalize()
        d4 = self.dep_head.get()
        d3 = self.dep_fund.get()

        conn = mysql.connector.connect(
```

```
            host="localhost",
            user="root",
            password="Lights@123",
            database="HMS"
        )
        cursor = conn.cursor()

        cursor.execute("SELECT * FROM department WHERE DEPARTMENT_NAME = %s",
(d2,))
        p = cursor.fetchall()
        x = len(p)

        if x != 0:
            tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM",
"DEPARTMENT ID ALREADY EXISTS")
        else:
            cursor.execute("INSERT INTO department
(DEPARTMENT_NAME,DEPARTMENT_FUND,DEPARTMENT_HEAD) VALUES (%s, %s, %s)", (d2,
d3, d4))
            tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM",
"DEPARTMENT DATA ADDED")

        conn.commit()

    def DELETE_DEP_DISPLAY(self):
        self.newWindow = Toplevel(self.master)
        self.app = D_DEP(self.newWindow)

    def UPDATE_DEP_DISPLAY(self):
        self.newWindow = Toplevel(self.master)
        self.app = U_DEP(self.newWindow)
```

EMPLOYEE_FORM.PY

```
from sqlite3 import Cursor
import tkinter as tk
from tkinter import messagebox
import mysql.connector

conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Lights@123",
    database="HMS"
)

class Employee:
```

```
    def __init__(self, master,main_window):
        self.master = master
        self.main_window = main_window
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x1000+0+0")
        self.master.config(bg="cadet blue")
        self.frame = tk.Frame(self.master, bg="cadet blue")
        self.frame.pack()

        # Attributes
        self.emp_name = tk.StringVar()
        self.emp_sex = tk.StringVar()
        self.emp_age = tk.IntVar()
        self.emp_type = tk.StringVar()
        self.emp_salary = tk.IntVar()
        self.emp_exp = tk.StringVar()
        self.emp_email = tk.StringVar()
        self.emp_phno = tk.StringVar()  # Use StringVar to validate length

        # Title
        self.lblTitle = tk.Label(self.frame, text="EMPLOYEE REGISTRATION
FORM", font="Helvetica 20 bold", bg="cadet blue")
        self.lblTitle.grid(row=0, column=0, columnspan=2, pady=50)

        # Frame
        self.LoginFrame = tk.Frame(self.frame, width=400, height=80,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame.grid(row=1, column=0)
        self.LoginFrame2 = tk.Frame(self.frame, width=400, height=80,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame2.grid(row=2, column=0)

        # Labels and Entries
        self.lblempname = tk.Label(self.LoginFrame, text="EMPLOYEE NAME",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblempname.grid(row=0, column=0)
        self.lblempname_entry = tk.Entry(self.LoginFrame, font="Helvetica 14
bold", bd=2, textvariable=self.emp_name)
        self.lblempname_entry.grid(row=0, column=1)

        self.lblsex = tk.Label(self.LoginFrame, text="SEX", font="Helvetica 14
bold", bg="cadet blue", bd=22)
        self.lblsex.grid(row=1, column=0)
        self.lblsex_entry = tk.Entry(self.LoginFrame, font="Helvetica 14
bold", bd=2, textvariable=self.emp_sex)
        self.lblsex_entry.grid(row=1, column=1)
```

```
        self.lblage = tk.Label(self.LoginFrame, text="AGE", font="Helvetica 14
bold", bg="cadet blue", bd=22)
        self.lblage.grid(row=2, column=0)
        self.lblage_entry = tk.Entry(self.LoginFrame, font="Helvetica 14
bold", bd=2, textvariable=self.emp_age)
        self.lblage_entry.grid(row=2, column=1)

        self.lbltype = tk.Label(self.LoginFrame, text="EMPLOYEE DESIGNATION
[DOCTOR, NURSE, RECEPTIONIST]", font="Helvetica 14 bold", bg="cadet blue",
bd=22)
        self.lbltype.grid(row=3, column=0)
        self.lbltype_entry = tk.Entry(self.LoginFrame, font="Helvetica 14
bold", bd=2, textvariable=self.emp_type)
        self.lbltype_entry.grid(row=3, column=1)

        self.lblsalary = tk.Label(self.LoginFrame, text="SALARY",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblsalary.grid(row=0, column=2)
        self.lblsalary_entry = tk.Entry(self.LoginFrame, font="Helvetica 14
bold", bd=2, textvariable=self.emp_salary)
        self.lblsalary_entry.grid(row=0, column=3)

        self.lblexp = tk.Label(self.LoginFrame, text="EXPERIENCE",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblexp.grid(row=1, column=2)
        self.lblexp_entry = tk.Entry(self.LoginFrame, font="Helvetica 14
bold", bd=2, textvariable=self.emp_exp)
        self.lblexp_entry.grid(row=1, column=3)

        self.lblphno = tk.Label(self.LoginFrame, text="CONTACT NUMBER",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblphno.grid(row=2, column=2)
        self.lblphno_entry = tk.Entry(self.LoginFrame, font="Helvetica 14
bold", bd=2, textvariable=self.emp_phno)
        self.lblphno_entry.grid(row=2, column=3)

        self.lblemail = tk.Label(self.LoginFrame, text="EMAIL",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblemail.grid(row=3, column=2)
        self.lblemail_entry = tk.Entry(self.LoginFrame, font="Helvetica 14
bold", bd=2, textvariable=self.emp_email)
        self.lblemail_entry.grid(row=3, column=3)

        self.button2 = tk.Button(self.LoginFrame2, text="SUBMIT", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=lambda:
self.INSERT_EMP(self.get_emp_data()))
        self.button2.grid(row=0, column=1)
```

```
        self.button3 = tk.Button(self.LoginFrame2, text="UPDATE", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=lambda:
self.UPDATE_EMP(self.get_emp_data()))
        self.button3.grid(row=0, column=2)

        self.button4 = tk.Button(self.LoginFrame2, text="DELETE", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.DE_DISPLAY)
        self.button4.grid(row=0, column=3)

        self.button5 = tk.Button(self.LoginFrame2, text="SEARCH", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.S_DISPLAY)
        self.button5.grid(row=0, column=4)

        self.button6 = tk.Button(self.LoginFrame2, text="EXIT", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.Exit)
        self.button6.grid(row=0, column=5)

    def Exit(self):
        self.master.destroy()
        self.main_window.deiconify()

    def clear(self):
        self.lblempname_entry.delete(0, 'end')
        self.lblage_entry.delete(0, 'end')
        self.lblsex_entry.delete(0, 'end')
        self.lblemail_entry.delete(0, 'end')
        self.lblphno_entry.delete(0, 'end')
        self.lblsalary_entry.delete(0, 'end')
        self.lblexp_entry.delete(0, 'end')
        self.lbltype_entry.delete(0, 'end')

        # Clear the corresponding fields in the database table
        #query = "DELETE FROM your_table_name"
        #cursor.execute(query)
        conn.commit()


    def get_emp_data(self):
        return {
            "name": self.emp_name.get(),
            "sex": self.emp_sex.get(),
            "age": self.emp_age.get(),
            "type": self.emp_type.get(),
            "salary": self.emp_salary.get(),
            "exp": self.emp_exp.get(),
            "email": self.emp_email.get(),
            "phno": self.emp_phno.get()
        }
    def check_designation(self):
```

```
        if self.emp_type.get() in ["doctor", "nurse"]:
            self.newWindow = tk.Toplevel(self.master)
            self.app = Department(self.newWindow, self.get_emp_data())

    def DE_DISPLAY(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = D_EMP(self.newWindow)

    def S_DISPLAY(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = S_EMP(self.newWindow)

    def INSERT_EMP(self, emp_data):
        try:
            cursor = conn.cursor()

            insert_query = """
                INSERT INTO employee (EMP_NAME, SEX, AGE, DESIG, SAL, EXP,
EMAIL, PHONE)
                VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
            """
            cursor.execute(insert_query, (emp_data['name'], emp_data['sex'],
emp_data['age'], emp_data['type'],
                                          emp_data['salary'], emp_data['exp'],
emp_data['email'], emp_data['phno']))
            conn.commit()

            messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "EMPLOYEE DATA
ADDED")
            self.check_designation()
            self.clear()

        except ValueError as ve:
            messagebox.showerror("Invalid Input", str(ve))
        except mysql.connector.Error as err:
            messagebox.showerror("Database Error", f"Error: {err}")

    def UPDATE_EMP(self, emp_data):
        self.newWindow = tk.Toplevel(self.master)
        self.app = U_EMP(self.newWindow)
```

MENU_RECEPTION.PY

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import mysql.connector
```

```python
# Import your forms for Patient, Room, Employee, Appointment, and Billing
from patient_form import Patient
from room_form import Room
from appointment_form import Appointment
from billing_form import Billing
from record_win import Record



# Establish database connection
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Lights@123",
    database="HMS"
)

print("DATABASE CONNECTION SUCCESSFUL")

#root=Tk()

class Menu_r:
    def __init__(self, master,main_window):
        self.master = master
        self.main_window = main_window
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x1000+0+0")
        self.master.config(bg="cadet blue")
        self.frame = tk.Frame(self.master, bg="cadet blue")
        self.frame.pack()

        self.lblTitle = tk.Label(self.frame, text="MAIN MENU",
font=("Helvetica", 20, "bold"), bg="cadet blue")
        self.lblTitle.grid(row=0, column=0, columnspan=2, pady=50)

        self.LoginFrame = tk.Frame(self.frame, width=400, height=80,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame.grid(row=1, column=0)

        buttons = [
            ("1. PATIENT REGISTRATION", self.Patient_Reg),
            ("2. ROOM ALLOCATION", self.Room_Allocation),
            ("3. BOOK APPOINTMENT", self.Appointment_Form),
            ("4. PATIENT BILL", self.Billing_Form),
            ("5. VIEW RECORDS", self.View_Records),
            ("6. EXIT", self.Exit)
        ]
```

```
        for i, (text, command) in enumerate(buttons):
            button = tk.Button(self.LoginFrame, text=text, width=30,
font=("Helvetica", 14, "bold"), bg="cadet blue", command=command)
            button.grid(row=i, column=0, pady=10)

    def Exit(self):
        self.master.destroy()
        self.main_window.deiconify()

    def Patient_Reg(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = Patient(self.newWindow,self.master)
        self.master.withdraw()

    def Room_Allocation(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = Room(self.newWindow,self.master)

    def Appointment_Form(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = Appointment(self.newWindow,self.master)
        self.master.withdraw()

    def Billing_Form(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = Billing(self.newWindow,self.master)
        self.master.withdraw()

    def View_Records(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = Record(self.newWindow,self.master)
        self.master.withdraw()
```

MENU.PY

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import mysql.connector

# Import your forms for Patient, Room, Employee, Appointment, and Billing
from patient_form import Patient
from room_form import Room
from employee_form import Employee
from appointment_form import Appointment
from billing_form import Billing
```

```
from record_win import Record
from department_form import dept

# Establish database connection
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Lights@123",
    database="HMS"
)

print("DATABASE CONNECTION SUCCESSFUL")

#root=Tk()

class Menu:
    def __init__(self, master,main_window):
        self.master = master
        self.main_window = main_window
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x1000+0+0")
        self.master.config(bg="cadet blue")
        self.frame = tk.Frame(self.master, bg="cadet blue")
        self.frame.pack()

        self.lblTitle = tk.Label(self.frame, text="MAIN MENU",
font=("Helvetica", 20, "bold"), bg="cadet blue")
        self.lblTitle.grid(row=0, column=0, columnspan=2, pady=50)

        self.LoginFrame = tk.Frame(self.frame, width=400, height=80,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame.grid(row=1, column=0)

        buttons = [
            ("1. PATIENT REGISTRATION", self.Patient_Reg),
            ("2. ROOM ALLOCATION", self.Room_Allocation),
            ("3. EMPLOYEE REGISTRATION", self.Employee_Reg),
            ("4. BOOK APPOINTMENT", self.Appointment_Form),
            ("5. PATIENT BILL", self.Billing_Form),
            ("6. VIEW RECORDS", self.View_Records),
            ("7. DEPARTMENT REGISTRATION", self.Department_Reg),
            ("8. EXIT", self.Exit)
        ]

        for i, (text, command) in enumerate(buttons):
            button = tk.Button(self.LoginFrame, text=text, width=30,
font=("Helvetica", 14, "bold"), bg="cadet blue", command=command)
            button.grid(row=i, column=0, pady=10)
```

```
    def Exit(self):
        self.master.destroy()
        self.main_window.deiconify()

    def Patient_Reg(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = Patient(self.newWindow,self.master)
        self.master.withdraw()

    def Room_Allocation(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = Room(self.newWindow,self.master)
        self.master.withdraw()

    def Employee_Reg(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = Employee(self.newWindow,self.master)
        self.master.withdraw()

    def Appointment_Form(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = Appointment(self.newWindow,self.master)
        self.master.withdraw()

    def Billing_Form(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = Billing(self.newWindow,self.master)
        self.master.withdraw()

    def View_Records(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = Record(self.newWindow,self.master)
        self.master.withdraw()

    def Department_Reg(self):
        self.newWindow = tk.Toplevel(self.master)
        self.app = dept(self.newWindow,self.master)
        self.master.withdraw()
```

RECORD_WIN.PY

```
import tkinter as tk
from tkinter import ttk, messagebox
import mysql.connector
```

```
# Establish database connection
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Lights@123",
    database="HMS"
)

cursor = conn.cursor()
print("DATABASE CONNECTION SUCCESSFUL")

class Record:
    def __init__(self, master, main_window):
        self.master = master
        self.main_window = main_window
        self.master.title("Hospital Management System")
        self.master.geometry("1600x1000+0+0")
        self.master.config(bg="cadet blue")

        self.frame = tk.Frame(self.master, bg="cadet blue")
        self.frame.pack()

        self.lblTitle = tk.Label(self.frame, text="RECORDS MENU",
font=("Helvetica", 20, "bold"), bg="cadet blue")
        self.lblTitle.grid(row=0, column=0, columnspan=2, pady=50)

        self.LoginFrame = tk.Frame(self.frame, width=400, height=80,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame.grid(row=1, column=0)

        buttons = [
            ("1. PATIENT RECORD", self.open_patient_window),
            ("2. ROOM RECORD", self.open_room_window),
            ("3. EMPLOYEE RECORD", self.open_employee_window),
            ("4. PATIENT BILL RECORD", self.open_bill_window),
            ("5. MEDICINE RECORD", self.open_medicine_window),
            ("6. OUTPATIENT RECORD", self.open_outpatient_window),
            ("7. TREATMENT RECORD", self.open_treatment_window),
            ("8. CONSULT RECORD", self.open_consult_window),
            ("9. EXIT", self.Exit)
        ]

        for i, (text, command) in enumerate(buttons):
            button = tk.Button(self.LoginFrame, text=text, width=30,
font=("Helvetica", 14, "bold"), bg="cadet blue", command=command)
            button.grid(row=i, column=0, pady=10)

    def Exit(self):
```

```
        self.master.destroy()
        self.main_window.deiconify()

    def open_patient_window(self):
        self.master.withdraw()
        PatientWindow(self.master)

    def open_room_window(self):
        self.master.withdraw()
        RoomWindow(self.master)

    def open_employee_window(self):
        self.master.withdraw()
        EmployeeWindow(self.master)

    def open_bill_window(self):
        self.master.withdraw()
        BillWindow(self.master)

    def open_medicine_window(self):
        self.master.withdraw()
        MedicineWindow(self.master)

    def open_outpatient_window(self):
        self.master.withdraw()
        OutpatientWindow(self.master)

    def open_treatment_window(self):
        self.master.withdraw()
        TreatmentWindow(self.master)

    def open_consult_window(self):
        self.master.withdraw()
        ConsultWindow(self.master)
```

ROOM_FORM.PY

```
from tkinter import *
import tkinter.messagebox
from tkinter import ttk
from tkinter import font
import mysql.connector

conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Lights@123",
```

```
    database="HMS"
)
cursor = conn.cursor()

class Room:
    def __init__(self, master, main_window):
        self.master = master
        self.main_window = main_window
        self.master.title("HOSPITAL MANAGEMENT SYSTEM")
        self.master.geometry("1600x1000+0+0")
        self.master.config(bg="cadet blue")
        self.frame = Frame(self.master, bg="cadet blue")
        self.frame.pack()

        # ATTRIBUTES
        self.P_id = IntVar()
        self.room_t = StringVar()
        self.room_no = StringVar()
        self.rate = IntVar()
        self.da = StringVar()
        self.dd = StringVar()

        # TITLE
        self.lblTitle = Label(self.frame, text="ROOM ALLOCATION FORM",
font="Helvetica 20 bold", bg="cadet blue")
        self.lblTitle.grid(row=0, column=0, columnspan=2, pady=50)

        # FRAME
        self.LoginFrame = Frame(self.frame, width=400, height=80,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame.grid(row=1, column=0)

        self.LoginFrame2 = Frame(self.frame, width=400, height=80,
relief="ridge", bg="cadet blue", bd=20)
        self.LoginFrame2.grid(row=2, column=0)

        # LABELS
        self.lblpatid = Label(self.LoginFrame, text="PATIENT ID",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblpatid.grid(row=0, column=0)
        self.lblpatid_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.P_id)
        self.lblpatid_entry.grid(row=0, column=1)

        self.room_t1 = Label(self.LoginFrame, text="ROOM TYPE\nSINGLE ROOM: Rs
4500\nTWIN SHARING : Rs2500\nTRIPLE SHARING: Rs2000\n",
                             font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.room_t1.grid(row=1, column=0)
```

```
        self.room_t1_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.room_t)
        self.room_t1_entry.grid(row=1, column=1)

        self.room_no1 = Label(self.LoginFrame, text="ROOM NUMBER ",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.room_no1.grid(row=2, column=0)
        self.room_no1_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.room_no)
        self.room_no1_entry.grid(row=2, column=1)

        self.lblrate = Label(self.LoginFrame, text="ROOM CHARGES",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblrate.grid(row=0, column=2)
        self.lblrate_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.rate)
        self.lblrate_entry.grid(row=0, column=3)

        self.lblda = Label(self.LoginFrame, text="DATE ADMITTED",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lblda.grid(row=1, column=2)
        self.lblda_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.da)
        self.lblda_entry.grid(row=1, column=3)

        self.lbldd = Label(self.LoginFrame, text="DATE DISCHARGED",
font="Helvetica 14 bold", bg="cadet blue", bd=22)
        self.lbldd.grid(row=2, column=2)
        self.lbldd_entry = Entry(self.LoginFrame, font="Helvetica 14 bold",
bd=2, textvariable=self.dd)
        self.lbldd_entry.grid(row=2, column=3)

        self.button2 = Button(self.LoginFrame2, text="SUBMIT", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.INSERT_ROOM)
        self.button2.grid(row=3, column=1)

        self.button3 = Button(self.LoginFrame2, text="UPDATE", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.UPDATE_ROOM)
        self.button3.grid(row=3, column=2)

        self.button5 = Button(self.LoginFrame2, text="ROOM DETAILS", width=15,
font="Helvetica 14 bold", bg="cadet blue", command=self.SEARCH_ROOM)
        self.button5.grid(row=3, column=4)

        self.button6 = Button(self.LoginFrame2, text="EXIT", width=10,
font="Helvetica 14 bold", bg="cadet blue", command=self.Exit)
        self.button6.grid(row=3, column=5)
```

```
    def clear(self):
        self.lblpatid_entry.delete(0, 'end')
        self.room_t1_entry.delete(0, 'end')
        self.room_no1_entry.delete(0, 'end')
        self.lblrate_entry.delete(0, 'end')
        self.lblda_entry.delete(0, 'end')
        self.lbldd_entry.delete(0, 'end')

    def INSERT_ROOM(self):
        global r1, r2, r3, r4, r5, r6, conn
        try:
            r1 = self.P_id.get()
            r2 = self.room_t.get()
            r3 = self.room_no.get()
            r4 = self.rate.get()
            r5 = self.da.get()
            r6 = self.dd.get()
            r7 = 'NO'

            if r6 == '':
                r6 = r5

            cursor.execute("SELECT * FROM ROOM WHERE ROOM_NO = %s", (r3,))
            p = cursor.fetchall()

            if p:
                tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM",
"ROOM_NO IS CURRENTLY OCCUPIED")
            else:
                cursor.execute('INSERT INTO ROOM VALUES (%s, %s, %s, %s, %s,
%s, %s)', (r1, r3, r2, r4, r5, r6, r7))
                tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "ROOM
ALLOCATED")
                self.clear()
                conn.commit()

        except mysql.connector.Error as err:
            tkinter.messagebox.showerror("MySQL Error", f"Error: {err}")

    def SEARCH_ROOM(self):
        try:
            self.newWindow = Toplevel(self.master)
            self.app = S_Room(self.newWindow)
        except mysql.connector.Error as err:
            tkinter.messagebox.showerror("MySQL Error", f"Error: {err}")

    def Exit(self):
        try:
```

```
            self.master.destroy()
            self.main_window.deiconify()
        except mysql.connector.Error as err:
            tkinter.messagebox.showerror("MySQL Error", f"Error: {err}")

    def UPDATE_ROOM(self):
        global r1, r2, r3, r4, r5, r6, conn
        try:
            r1 = self.P_id.get()
            r2 = self.room_t.get()
            r3 = self.room_no.get()
            r4 = self.rate.get()
            r5 = self.da.get()
            r6 = self.dd.get()

            if r6 == '':
                r6 = r5

            cursor.execute("SELECT * FROM ROOM WHERE PATIENT_ID = %s ", (r1,))
            p = cursor.fetchall()

            if not p:
                tkinter.messagebox.showerror("HOSPITAL DATABASE SYSTEM",
"PATIENT IS NOT ALLOCATED A ROOM")
            else:
                cursor.execute('UPDATE ROOM SET ROOM_NO=%s, ROOM_TYPE=%s,
RATE=%s, DATE_ADMITTED=%s, DATE_DISCHARGED=%s WHERE PATIENT_ID=%s', (r3, r2,
r4, r5, r6, r1,))
                tkinter.messagebox.showinfo("HOSPITAL DATABASE SYSTEM", "ROOM
DETAILS UPDATED")
                self.clear()
                conn.commit()

        except mysql.connector.Error as err:
            tkinter.messagebox.showerror("MySQL Error", f"Error: {err}")
```

BACKEND PROGRAMMING CODE – MYSQL + PYTHON

DATABASE.PY

```
import mysql.connector

# Connect to MySQL database
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Lights@123",
    database="HMS"
)

print("DATABASE CONNECTION SUCCESSFUL")

c = conn.cursor()

# Drop existing tables if they exist
c.execute("DROP TABLE IF EXISTS APPOINTMENT")
c.execute("DROP TABLE IF EXISTS ROOM")
c.execute("DROP TABLE IF EXISTS MEDICINE")
c.execute("DROP TABLE IF EXISTS TREATMENT")
c.execute("DROP TABLE IF EXISTS EMPLOYEE")
c.execute("DROP TABLE IF EXISTS CONTACT_NO")
c.execute("DROP TABLE IF EXISTS PATIENT")
c.execute("DROP TABLE IF EXISTS OUTPATIENT")
c.execute("DROP VIEW IF EXISTS EMPLOYEE_V")
c.execute("DROP TABLE IF EXISTS WORKS")
c.execute("DROP TABLE IF EXISTS DEPARTMENT")
c.execute("DROP TABLE IF EXISTS CONSULT")

# Create tables
c.execute("""
    CREATE TABLE PATIENT (
        PATIENT_ID INT PRIMARY KEY AUTO_INCREMENT,
        NAME VARCHAR(20) NOT NULL,
        SEX VARCHAR(10) NOT NULL,
        BLOOD_GROUP VARCHAR(5) NOT NULL,
        DOB DATE NOT NULL,
        ADDRESS VARCHAR(100) NOT NULL,
        CONSULT_TEAM VARCHAR(50) NOT NULL,
        EMAIL VARCHAR(50) NOT NULL,
        `CONDITION` VARCHAR(30) NOT NULL
    )
""")
```

```
print("PATIENT TABLE CREATED SUCCESSFULLY")

c.execute("""
    CREATE TABLE CONTACT_NO (
        PATIENT_ID INT PRIMARY KEY,
        CONTACTNO BIGINT NOT NULL,
        ALT_CONTACT BIGINT,
        FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE
CASCADE
    )
""")
print("CONTACT_NO TABLE CREATED SUCCESSFULLY")

c.execute("""
    CREATE TABLE EMPLOYEE (
        EMP_ID INT PRIMARY KEY AUTO_INCREMENT,
        EMP_NAME VARCHAR(20) NOT NULL,
        SEX VARCHAR(10) NOT NULL,
        AGE INT NOT NULL,
        DESIG VARCHAR(20) NOT NULL,
        SAL INT NOT NULL,
        EXP VARCHAR(100) NOT NULL,
        EMAIL VARCHAR(40) NOT NULL UNIQUE,
        PHONE BIGINT
    )
""")
print("EMPLOYEE TABLE CREATED SUCCESSFULLY")

c.execute("""
CREATE VIEW EMPLOYEE_V AS
SELECT EMP_ID, EMP_NAME, SEX, AGE, DESIG, EXP, EMAIL, PHONE
FROM EMPLOYEE
WHERE EMP_ID <> 1;
""")

c.execute("""INSERT INTO EMPLOYEE VALUES (1, 'ADMIN', 'UNIDENT', 50,
'PRESIDENT', 1000000, 'FOUNDER', 'CAREANDCURE@GMAIL.COM', 0000000000)""")

c.execute("""
CREATE TABLE IF NOT EXISTS DEPARTMENT (
    DEPT_ID INT AUTO_INCREMENT PRIMARY KEY,
    DEPARTMENT_NAME VARCHAR(255) NOT NULL UNIQUE,
    DEPARTMENT_FUND INT,
    DEPARTMENT_HEAD INT,
    FOREIGN KEY (DEPARTMENT_HEAD) REFERENCES EMPLOYEE (EMP_ID) ON DELETE SET
NULL
)
""")
```

```
print("DEPT TABLE CREATED SUCCESSFULLY")

# Initialize common departments
departments = [
    ('Management', 1000000, 1),
    ('HR', 500000, 1),
    ('Cardiology', 2000000, 1),
    ('Neurology', 2000000, 1),
    ('Orthopedics', 2000000, 1),
    ('Pediatrics', 2000000, 1)
]

c.executemany("INSERT INTO DEPARTMENT (DEPARTMENT_NAME, DEPARTMENT_FUND,
DEPARTMENT_HEAD) VALUES (%s, %s, %s)", departments)

c.execute("""
CREATE TABLE IF NOT EXISTS WORKS (
    DEPT_ID INT,
    EMP_ID INT,
    ROLE VARCHAR(20),
    SHIFT_DETAILS VARCHAR(30),
    FOREIGN KEY (DEPT_ID) REFERENCES DEPARTMENT (DEPT_ID) ON DELETE SET NULL,
    FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID) ON DELETE CASCADE
)
""")
print("WORKS TABLE CREATED SUCCESSFULLY")


c.execute("""
    CREATE TABLE TREATMENT (
    T_NO INT AUTO_INCREMENT PRIMARY KEY,
    PATIENT_ID INT,
    DOC_ID INT,
    TREATMENT VARCHAR(100) NOT NULL,
    TREATMENT_CODE VARCHAR(30) NOT NULL,
    T_COST INT NOT NULL,
    T_DATE DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (DOC_ID) REFERENCES EMPLOYEE (EMP_ID) ON DELETE SET NULL,
    FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE CASCADE
)
""")
print("TREATMENT TABLE CREATED SUCCESSFULLY")

c.execute("""
CREATE TABLE IF NOT EXISTS CONSULT (
    PATIENT_ID INT,
    EMP_ID INT,
    FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE
CASCADE,
```

```
    FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID) ON DELETE CASCADE,
    PRIMARY KEY(PATIENT_ID,EMP_ID)
)
""")
print("CONSULT TABLE CREATED SUCCESSFULLY")

c.execute("""
    CREATE TABLE MEDICINE (
        M_NO INT AUTO_INCREMENT PRIMARY KEY,
        PATIENT_ID INT,
        MEDICINE_NAME VARCHAR(100) NOT NULL,
        M_COST INT NOT NULL,
        M_QTY INT NOT NULL,
        M_DATE DATETIME DEFAULT CURRENT_TIMESTAMP,
        FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE
CASCADE
    )
""")
print("MEDICINE TABLE CREATED SUCCESSFULLY")

c.execute("""
    CREATE TABLE ROOM (
    PATIENT_ID INT NOT NULL,
    ROOM_NO VARCHAR(20) PRIMARY KEY,
    ROOM_TYPE VARCHAR(10) NOT NULL,
    RATE  INT  NOT  NULL,
    DATE_ADMITTED DATE,
    DATE_DISCHARGED  DATE,
    PAY_STATUS ENUM('YES', 'NO'),
    FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE CASCADE
)
""")
print("ROOM TABLE CREATED SUCCESSFULLY")

c.execute("""
    CREATE TABLE BILL (
        PATIENT_ID INT NOT NULL,
        BILL INT,
        FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE
CASCADE
    )
""")

c.execute("""
    CREATE TABLE APPOINTMENT (
        PATIENT_ID INT NOT NULL,
        EMP_ID INT NOT NULL,
        AP_NO INT  AUTO_INCREMENT PRIMARY KEY,
```

```
        AP_TIME TIME,
        AP_DATE DATE,
        DESCRIPTION VARCHAR(100),
        FOREIGN KEY (PATIENT_ID) REFERENCES PATIENT (PATIENT_ID) ON DELETE
CASCADE,
        FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID)
    )
""")
print("APPOINTMENT TABLE CREATED SUCCESSFULLY")

# Create OUTPATIENT table to log discharged patients
c.execute("""
    CREATE TABLE OUTPATIENT (
        PATIENT_ID INT PRIMARY KEY,
        NAME VARCHAR(20) NOT NULL,
        SEX VARCHAR(10) NOT NULL,
        BLOOD_GROUP VARCHAR(5) NOT NULL,
        DOB DATE NOT NULL,
        ADDRESS VARCHAR(100) NOT NULL,
        CONSULT_TEAM VARCHAR(50) NOT NULL,
        EMAIL VARCHAR(50) NOT NULL,
        `CONDITION` VARCHAR(30) NOT NULL,
        DATE_DISCHARGED DATE NOT NULL
    )
""")
print("OUTPATIENT TABLE CREATED SUCCESSFULLY")

# Trigger to check room allocation before insert
c.execute("""
    CREATE TRIGGER check_room_allocation_before_insert
BEFORE INSERT ON ROOM
FOR EACH ROW
BEGIN
    DECLARE room_count INT;
    DECLARE patient_count INT;

    -- Check if the room is already allocated to another patient
    SELECT COUNT(*) INTO room_count FROM ROOM WHERE ROOM_NO = NEW.ROOM_NO;
    IF room_count > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error: The room is already
allocated to another patient.';
    END IF;

    -- Check if the patient is already allocated a room
    SELECT COUNT(*) INTO patient_count FROM ROOM WHERE PATIENT_ID =
NEW.PATIENT_ID;
    IF patient_count > 0 THEN
```

```
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error: The patient is
already allocated a room.';
    END IF;
END;
    """)

    # Trigger to check room allocation before update
c.execute("""
    CREATE TRIGGER check_room_allocation_before_update
BEFORE UPDATE ON ROOM
FOR EACH ROW
BEGIN
    DECLARE room_count INT;

    -- Only check if ROOM_NO is being updated
    IF NEW.ROOM_NO != OLD.ROOM_NO THEN
        -- Check if the new room number is already allocated to another
patient
        SELECT COUNT(*) INTO room_count FROM ROOM WHERE ROOM_NO = NEW.ROOM_NO
AND PATIENT_ID != NEW.PATIENT_ID;
        IF room_count > 0 THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Error: The room is
already allocated to another patient.';
        END IF;
    END IF;
END
    """)

# Create triggers for validation
c.execute("""
    CREATE TRIGGER check_patient_email
    BEFORE INSERT ON PATIENT
    FOR EACH ROW
    BEGIN
        IF NEW.EMAIL NOT LIKE '%_@__%.__%' THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Invalid email format for patient';
        END IF;
    END
""")

c.execute("""
    CREATE TRIGGER validate_employee_email
    BEFORE INSERT ON EMPLOYEE
    FOR EACH ROW
    BEGIN
        IF NEW.EMAIL NOT LIKE '%_@__%.__%' THEN
            SIGNAL SQLSTATE '45000'
```

```
            SET MESSAGE_TEXT = 'Invalid email format for employee';
        END IF;
    END
""")

c.execute("""
    CREATE TRIGGER prevent_delete_admin
    BEFORE DELETE ON EMPLOYEE
    FOR EACH ROW
    BEGIN
        IF OLD.EMP_ID = 1 THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot delete the
admin employee.';
        END IF;
    END;
""")

c.execute("""
    CREATE TRIGGER validate_employee_age
    BEFORE INSERT ON EMPLOYEE
    FOR EACH ROW
    BEGIN
        IF NEW.AGE < 18 OR NEW.AGE > 65 THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Employee age must be
between 18 and 65.';
        END IF;
    END;
""")

c.execute("""
     CREATE TRIGGER validate_patient_phone
        BEFORE INSERT ON CONTACT_NO
        FOR EACH ROW
        BEGIN
            IF CHAR_LENGTH(NEW.CONTACTNO) != 10 OR
(CHAR_LENGTH(NEW.ALT_CONTACT)!=10 AND NEW.ALT_CONTACT !=0)THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Phone number must
be 10 digits';
            END IF;
        END;
""")

# Create trigger to log discharged patients in OUTPATIENT table
c.execute("""
    CREATE TRIGGER log_discharge_patient
    AFTER DELETE ON PATIENT
    FOR EACH ROW
    BEGIN
```

```
        INSERT INTO OUTPATIENT (PATIENT_ID, NAME, SEX, BLOOD_GROUP, DOB,
ADDRESS, CONSULT_TEAM, EMAIL, `CONDITION`, DATE_DISCHARGED)
        VALUES (OLD.PATIENT_ID, OLD.NAME, OLD.SEX, OLD.BLOOD_GROUP, OLD.DOB,
OLD.ADDRESS, OLD.CONSULT_TEAM, OLD.EMAIL, OLD.`CONDITION`, CURDATE());
    END;
""")
print("TRIGGERS CREATED SUCCESSFULLY")

# Create procedures
c.execute("""
    CREATE PROCEDURE SearchPatients(IN search_by VARCHAR(50), IN search_term
VARCHAR(100))
BEGIN
    IF search_by = 'Patient ID' THEN
        SELECT * FROM patient WHERE patient_id LIKE CONCAT('%', search_term,
'%');
    ELSEIF search_by = 'Patient Name' THEN
        SELECT * FROM patient WHERE name LIKE CONCAT('%', search_term, '%');
    END IF;
END;
""")
print("PROCEDURE CREATED SUCCESSFULLY")

c.execute("""
    CREATE PROCEDURE SearchRooms(IN search_by VARCHAR(50), IN search_term
VARCHAR(100))
BEGIN
    IF search_by = 'Room Number' THEN
        SELECT * FROM room WHERE room_no LIKE CONCAT('%', search_term, '%');
    ELSEIF search_by = 'Patient ID' THEN
        SELECT * FROM room WHERE patient_id LIKE CONCAT('%', search_term,
'%');
    END IF;
END
""")
print("PROCEDURE CREATED SUCCESSFULLY")

c.execute("""
    CREATE PROCEDURE SearchBills(IN search_by VARCHAR(50), IN search_term
VARCHAR(100))
BEGIN
    IF search_by = 'Patient ID' THEN
        SELECT b.patient_id, b.bill
        FROM bill b
        JOIN patient p ON b.patient_id = p.patient_id
        WHERE p.patient_id LIKE CONCAT('%', search_term, '%');
    ELSEIF search_by = 'Patient Name' THEN
        SELECT b.patient_id, b.bill
```

```
        FROM bill b
        JOIN patient p ON b.patient_id = p.patient_id
        WHERE p.name LIKE CONCAT('%', search_term, '%');
    END IF;
END
""")
print("PROCEDURE CREATED SUCCESSFULLY")

c.execute("""
    CREATE PROCEDURE SearchMedicines(IN search_by VARCHAR(50), IN search_term
VARCHAR(100))
BEGIN
    IF search_by = 'Patient ID' THEN
        SELECT m.m_no, m.patient_id, m.medicine_name, m.m_cost, m.m_qty,
m.m_date
        FROM medicine m
        JOIN patient p ON m.patient_id = p.patient_id
        WHERE p.patient_id LIKE CONCAT('%', search_term, '%');
    ELSEIF search_by = 'Patient Name' THEN
        SELECT m.m_no, m.patient_id, m.medicine_name, m.m_cost, m.m_qty,
m.m_date
        FROM medicine m
        JOIN patient p ON m.patient_id = p.patient_id
        WHERE p.name LIKE CONCAT('%', search_term, '%');
    END IF;
END
""")
print("PROCEDURE CREATED SUCCESSFULLY")

c.execute("""
    CREATE PROCEDURE SearchOutpatients(IN search_by VARCHAR(50), IN
search_term VARCHAR(100))
BEGIN
    IF search_by = 'Patient ID' THEN
        SELECT * FROM outpatient WHERE patient_id LIKE CONCAT('%',
search_term, '%');
    ELSEIF search_by = 'Patient Name' THEN
        SELECT * FROM outpatient WHERE name LIKE CONCAT('%', search_term,
'%');
    END IF;
END
""")
print("PROCEDURE CREATED SUCCESSFULLY")

c.execute("""
    CREATE PROCEDURE SearchTreatments(IN search_by VARCHAR(50), IN search_term
VARCHAR(100))
BEGIN
```

```
    IF search_by = 'Patient ID' THEN
        SELECT t.t_no, t.patient_id, t.doc_id, t.treatment, t.treatment_code,
t.t_cost, t.t_date
        FROM treatment t
        JOIN patient p ON t.patient_id = p.patient_id
        WHERE p.patient_id LIKE CONCAT('%', search_term, '%');
    ELSEIF search_by = 'Patient Name' THEN
        SELECT t.t_no, t.patient_id, t.doc_id, t.treatment, t.treatment_code,
t.t_cost, t.t_date
        FROM treatment t
        JOIN patient p ON t.patient_id = p.patient_id
        WHERE p.name LIKE CONCAT('%', search_term, '%');
    ELSEIF search_by = 'Doctor ID' THEN
        SELECT t.t_no, t.patient_id, t.doc_id, t.treatment, t.treatment_code,
t.t_cost, t.t_date
        FROM treatment t
        JOIN employee e ON t.doc_id = e.emp_id
        WHERE e.emp_id LIKE CONCAT('%', search_term, '%');
    END IF;
END
""")
print("PROCEDURE CREATED SUCCESSFULLY")

c.execute("""
    CREATE PROCEDURE SearchConsults(IN search_by VARCHAR(50), IN search_term
VARCHAR(100))
BEGIN
    IF search_by = 'Patient ID' THEN
        SELECT c.patient_id, c.emp_id
        FROM consult c
        JOIN patient p ON c.patient_id = p.patient_id
        WHERE p.patient_id LIKE CONCAT('%', search_term, '%');
    ELSEIF search_by = 'Patient Name' THEN
        SELECT c.patient_id, c.emp_id
        FROM consult c
        JOIN patient p ON c.patient_id = p.patient_id
        WHERE p.name LIKE CONCAT('%', search_term, '%');
    ELSEIF search_by = 'Doctor ID' THEN
        SELECT c.patient_id, c.emp_id
        FROM consult c
        JOIN employee e ON c.emp_id = e.emp_id
        WHERE e.emp_id LIKE CONCAT('%', search_term, '%');
    END IF;
END
""")
print("PROCEDURE CREATED SUCCESSFULLY")

# Commit changes and close the connection
```

```
conn.commit()
conn.close()
print("DATABASE SETUP COMPLETED SUCCESSFULLY")
```

FUTURE SCOPE

The features that could not be added are array of objects and multi-threading. In the near future these concepts will be added in an appropriate way. The hospital management system can be further improved to take care of large volume of data and incorporate more functionalities.

CONCLUSION

From this Mini Project I learned the various Python concept like Inheritance, Exception Handling, Packages, basic Class and objects, Basic Object-Oriented Programming concepts, IO handling, GUI (Tkinter), Database Connectivity (MySQL) and String Functions. This helped me to strengthen the core Python and Database concepts.

REFERENCES

1. https://www.javatpoint.com
2. https://www.geeksforgeeks.org
3. https://stackoverflow.com/questions/22492118/payroll-program-for-pythonusingmultiple-functions-and-return-function
4. https://chat.openai.com
5. https://github.com
6. Python Essential Reference.
7. MySQL Essential Reference.