A

Case Study

On

“Patient Care Manager”

Course Code- 24CAH-606

MASTERS IN COMPUTER APPLICATION

Submitted By

Name- Anand Raj
UID-24MCA20362
Branch- MCA
Section-6(A)

Submitted To

Ms. Palwinder Kaur Mangat
Assistant Professor

INDEX

Introduction:

In the digital age, effective and efficient management of healthcare data is crucial for delivering quality patient care. The "Hospital Management System" project is designed as a comprehensive application that streamlines various administrative and medical processes within a healthcare facility. Built using Python with a Tkinter GUI and integrated with a MySQL database, this project provides a user-friendly interface for managing patient records, staff details, appointments, billing, and inventory.

The choice of MySQL as the database management system enables robust, scalable, and secure data storage and retrieval, essential for managing the large volume of data typical in hospital operations. This system allows healthcare administrators to efficiently handle day-to-day tasks, reduce paperwork, minimize errors, and ultimately improve the quality of care provided to patients. Through a carefully structured database schema, it also ensures data integrity and enables quick access to patient and hospital records.

This case study explores the development and implementation of the Hospital Management System, focusing on the functionality provided by the MySQL database, the seamless integration with Python through MySQL connectors, and the role of Tkinter in delivering an intuitive user experience. The study will highlight the system's database structure, data flow, and the interaction between the application and MySQL, showcasing how this integration can be used as a reliable solution for hospital management tasks.

Problem Definition:

This code for the hospital management system using Tkinter and MySQL provides a functional interface for managing patient data and prescriptions. Here are some highlights and suggestions:

Highlights:

1. Tkinter Structure: Frames and Labels are well-organized, giving structure to patient information, prescription, and buttons.

2. MySQL Integration: Functions like `iprescriptionData`, `update_data`, and `idelete` handle database operations for inserting, updating, and deleting records.

3. Error Handling: `try-except` blocks handle database errors, showing messages to guide the user.

4. Prescription Area: `iPrectioption` method formats and displays prescription details, making the app more interactive.

Suggested Improvements:

- **Enhanced Exception Handling:**
  - Consider separating connection and cursor setup in each database-related function with `finally` blocks to ensure they close properly even if an error occurs.

- **Improving `fetch_data` Method:**
  - Implement sorting or filtering to enhance data retrieval, especially if dealing with large datasets.

- **Side Effects and Driving Information in `iPrectioption`:**
  - Ensure `SideEffect`, `FurtherInformation`, `DrivingUsingMachine` variables are included in `self.txtPrescription.insert()` calls if needed.

- **Enhanced SQL Security:**

- Although parameterized queries are used, ensure MySQL connection credentials are securely managed, preferably with environmental variables.

Objective:

- **Combo box Dropdown Values**:
  - Ensure the `values` field in the comboboxes is correctly updated if you want to expand the list of medications or other dropdown options dynamically from a database.

- **Button Command Typos**:
  - The method for generating the prescription text, `iPrectioption`, has a typo in both the function name and the button command. You might want to rename it to `generate_prescription` for clarity and change the button command to `command=self.generate_prescription`.
    .

- **Database Connection Error Handling**:
  - Ensure error handling on the database connection, with specific error messages for failed connections or SQL errors. The code does this in several places, but it could be standardized across all database interactions.

- **Data Fetching Logic**:
  - In `fetch_data`, add logic to handle cases when there are no records in the database, so the Treeview isn't unnecessarily refreshed.

- **Optimization in `get_cursor`**:
  - You can add a check to make sure a row is selected before accessing `self.hospital_table.focus()` to avoid unexpected behavior.

Solution Approach:

This is a well-structured Hospital Management System GUI using Tkinter, with database operations performed via MySQL. Here are a few minor fixes and improvements you may consider to make your application smoother:

- **Typo Correction in Label and Button Names:**
  - Change `iPrectioption` to `iPrescription` for clarity and consistency.
  - Fix the typo in `lblDrivingMachine`'s text parameter from `"Further Information:"` to `"Driving Using Machine:"`.

- **Duplicate Label Removal:**

- The label `Further Information` is added twice in `DataframeLeft`. Replace the duplicate with `Driving Using Machine` as the placeholder label to improve field organization.

- **Scrollbar Configuration for Treeview:**
  - `scroll_x.config(command=self.hospital_table.xview)`
    `scroll_y.config(command=self.hospital_table.yview)`. .

- **Error Handling in `get_cursor`:**
  - Check if a row is selected before attempting to retrieve values to prevent errors when the Treeview is empty or a selection isn't made.

- **Close Database Connections:**
  - To prevent potential memory leaks, add a `finally` block to close `conn` in `iprescriptionData`, `update_data`, and `idelete` functions if an error is raised.

Data Preparation:

Code for the Hospital Management System interface looks quite comprehensive. However, a few adjustments and enhancements can be made to improve its functionality and readability. Here's an outline of modifications and points of attention for completion:

- **SQL Database Schema Matching:**
  - Ensure that the table `hospital1` in your MySQL database matches the columns you defined in the code. Specifically, the table should have columns corresponding to `Name_of_tablets`, `ref`, `Dose`, `Numbersoftablets`, etc.

- **Error Handling:**
  - For database connections and cursor operations, wrapping them in `try-except` blocks is ideal, which you have partially implemented. This will catch any exceptions and help diagnose issues if the database connection fails.

- **Method Name Corrections:**
  - The method `iPrectioption` should be renamed to `generate_prescription` or a similar meaningful name.

    - Similarly, `idelete` could be `delete_record`.

- **Scrollbar Linking:**
  - The scrollbars need to be properly linked to the `hospital_table` widget. Replace:

- **Database Field Consistency:**

- Confirm that the field names in your SQL queries match exactly with the ones in your MySQL table (e.g., `Refrence_No`, `Name_of_tablets`).

- **Side Effects and Further Information:**
  - You haven't fully utilized `SideEffect` and `FurtherInformation` in `iprescriptionData`. You may want to add these in both the `fetch_data` and `update_data` methods as well.

- **Additional Functions:**
  - You could add a `search_data` function to retrieve specific patient records based on input criteria, such as `Patient ID` or `Reference Number`.

Result:

- **Label Duplication in `DataframeLeft`:**
  - The labels "Further Information" and "DrivingUsingMachine" are both set in the same position (`row=0, column=2`), so they overlap. Update one of these labels and entries to another row or column.

- **Scrollbar Configuration for TreeView (`hospital_table`):**
  - When setting up the scrollbars, the line `scroll_x=ttk.Scrollbar(command=self.hospital_table.xview)` should be `scroll_x.config(command=self.hospital_table.xview)`. The same applies to `scroll_y`.

- **Error Handling for Database Connection:**
  - Wrap the database connection sections (`conn = mysql.connector.connect(...)`) in a `try-except` block to handle cases where the database connection fails. This will prevent the program from crashing and display an appropriate error message.

- **Closing Database Connections:**
  - You should ensure `conn.close()` is called in all database operations, especially in `fetch_data()` and `idelete()` functions, to avoid potential database connection issues over time.

- **Event Binding for Buttons:**
  - Make sure the functions `self.iPrectioption`, `self.clear_fields`, and `self.iExit` are defined, as these are invoked in button commands.

Code:

```
from tkinter import *
from tkinter import ttk
import random
import time
```

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

class Hospital:
    def __init__(self, root):
        self.root = root
        self.root.title("Hospital Management System")
        self.root.geometry("1540x800+0+0")

        self.Name_of_tablets=StringVar()
        self.ref=StringVar()
        self.Dose=StringVar()
        self.Numbersoftablets=StringVar()
        self.Lot=StringVar()
        self.issuedate=StringVar()
        self.expdate=StringVar()
        self.dailydose=StringVar()
        self.SideEffect=StringVar()
        self.BloodPressure=StringVar()
        self.FurtherInformation=StringVar()
        self.StorageAdvice=StringVar()
        self.DrivingUsingMachine=StringVar()
        self.HowToUseMedication=StringVar()
        self.PatientId=StringVar()
        self.nhsNumber=StringVar()
        self.patientname=StringVar()
        self.DateOfBirth=StringVar()
        self.patientaddress=StringVar()

        lbltitle = Label(self.root, bd=20, relief=RIDGE, text="HOSPITAL MANAGEMENT
SYSTEM", fg="red", bg="white", font=("times new roman", 50, "bold"))
        lbltitle.pack(side=TOP, fill=X)

        # ==========data frame===========

        Dataframe = Frame(self.root, bd=20, relief=RIDGE)
        Dataframe.place(x=0, y=130, width=1530, height=400)

        DataframeLeft = LabelFrame(Dataframe, bd=10, relief=RIDGE, padx=20,
                                   font=("arial", 12, "bold"), text="Patient Information")
        DataframeLeft.place(x=0, y=5, width=980, height=350)

        DataframeRight = LabelFrame(Dataframe, bd=10, relief=RIDGE, padx=20,
                                    font=("arial", 12, "bold"), text="Prescription")
        DataframeRight.place(x=990, y=5, width=460, height=350)

        # ========buttons frame=========

        Buttonframe = Frame(self.root, bd=20, relief=RIDGE)
        Buttonframe.place(x=0, y=530, width=1530, height=70)

        # ===========details frame=========
```

```
        detailsframe = Frame(self.root, bd=20, relief=RIDGE)
        detailsframe.place(x=0, y=600, width=1530, height=190)

        # ==========DataframeLeft============

        lblNameTablet = Label(DataframeLeft, text="Names of Tablets", font=("times new
roman", 12, "bold"),padx=2,pady=6)
        lblNameTablet.grid(row=0, column=0)

        # Correct usage of ttk.Combobox and correction of "column"
        comNameTablet = ttk.Combobox(DataframeLeft,textvariable=self.Name_of_tablets,
font=("times new roman", 12, "bold"), width=33)
        comNameTablet["values"] = ("Nice", "Corona Vaccine", "Acetaminophen", "Adderall",
"Amlodipine", "Ativan")
        comNameTablet.current(0)
        comNameTablet.grid(row=0,column=1)


        lblref=Label(DataframeLeft,font=("times new roman",12,"bold"),text="Refrence
No:",padx=2)
        lblref.grid(row=1,column=0,sticky=W)
        txtref=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.ref,width=35)
        txtref.grid(row=1,column=1)


        lblDose=Label(DataframeLeft,font=("times new
roman",12,"bold"),text="Dose:",padx=2,pady=4)
        lblDose.grid(row=2,column=0,sticky=W)
        txtDose=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.Dose,width=35)
        txtDose.grid(row=2,column=1)


        lblNoOftablets=Label(DataframeLeft,font=("times new roman",12,"bold"),text="No Of
Tablets:",padx=2,pady=6)
        lblNoOftablets.grid(row=3,column=0,sticky=W)
        txtNoOftablets=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.Numbersof
tablets,width=35)
        txtNoOftablets.grid(row=3,column=1)


        lblLot=Label(DataframeLeft,font=("times new
roman",12,"bold"),text="Lot:",padx=2,pady=6)
        lblLot.grid(row=4,column=0,sticky=W)
        txtLot=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.Lot,width=35)
        txtLot.grid(row=4,column=1)


        lblissueDate=Label(DataframeLeft,font=("times new roman",12,"bold"),text="Issue
Date:",padx=2,pady=6)
        lblissueDate.grid(row=5,column=0,sticky=W)
        txtissueDate=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.issuedate,wid
th=35)
```

```
        txtissueDate.grid(row=5,column=1)


        lblExpDate=Label(DataframeLeft,font=("times new roman",12,"bold"),text="Exp
Date:",padx=2,pady=6)
        lblExpDate.grid(row=6,column=0,sticky=W)
        txtExpDate=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.expdate,width
=35)
        txtExpDate.grid(row=6,column=1)


        lblDailyDose=Label(DataframeLeft,font=("times new roman",12,"bold"),text="Daily
Dose:",padx=2,pady=4)
        lblDailyDose.grid(row=7,column=0,sticky=W)
        txtDailyDose=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.dailydose,wi
dth=35)
        txtDailyDose.grid(row=7,column=1)


        lblSideEffect=Label(DataframeLeft,font=("times new roman",12,"bold"),text="Side
Effect:",padx=2,pady=6)
        lblSideEffect.grid(row=8,column=0,sticky=W)
        txtSideEffect=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.SideEffect,
width=35)
        txtSideEffect.grid(row=8,column=1)


        lblFurtherInfo=Label(DataframeLeft,font=("times new roman",12,"bold"),text="Further
Information:",padx=2)
        lblFurtherInfo.grid(row=0,column=2,sticky=W)
        txtFurtherInfo=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.FurtherInfo
rmation,width=35)
        txtFurtherInfo.grid(row=0,column=3)


        lblDrivingMachine=Label(DataframeLeft,font=("times new
roman",12,"bold"),text="Further Information:",padx=2)
        lblDrivingMachine.grid(row=0,column=2,sticky=W)
        txtDrivingMachine=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.Drivin
gUsingMachine,width=35)
        txtDrivingMachine.grid(row=0,column=3)


        lblBloodPressure=Label(DataframeLeft,font=("times new roman",12,"bold"),text="Blood
Pressure:",padx=2,pady=6)
        lblBloodPressure.grid(row=1,column=2,sticky=W)
        txtBloodPressure=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.BloodPr
essure,width=35)
        txtBloodPressure.grid(row=1,column=3)


        lblStorage=Label(DataframeLeft,font=("times new roman",12,"bold"),text="Storage
Advice:",padx=2,pady=6)
```

```
        lblStorage.grid(row=2,column=2,sticky=W)
        txtStorage=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.StorageAdvice,
width=35)
        txtStorage.grid(row=2,column=3)


        lblMedicine=Label(DataframeLeft,font=("times new
roman",12,"bold"),text="Medication:",padx=2,pady=6)
        lblMedicine.grid(row=3,column=2,sticky=W)
        txtMedicine=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.HowToUseM
edication,width=35)
        txtMedicine.grid(row=3,column=3)


        lblPatientId=Label(DataframeLeft,font=("times new
roman",12,"bold"),text="PatientId:",padx=2,pady=6)
        lblPatientId.grid(row=4,column=2,sticky=W)
        txtPatientId=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.PatientId,widt
h=35)
        txtPatientId.grid(row=4,column=3)


        lblNhsNumber=Label(DataframeLeft,font=("times new roman",12,"bold"),text="NHS
Number:",padx=2,pady=6)
        lblNhsNumber.grid(row=5,column=2,sticky=W)
        txtNhsNumber=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.nhsNumbe
r,width=35)
        txtNhsNumber.grid(row=5,column=3)


        lblPatientName=Label(DataframeLeft,font=("times new roman",12,"bold"),text="Patient
Name:",padx=2,pady=6)
        lblPatientName.grid(row=6,column=2,sticky=W)
        txtPatientName=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.patientna
me,width=35)
        txtPatientName.grid(row=6,column=3)

        lblDateOfBirth=Label(DataframeLeft,font=("times new roman",12,"bold"),text="Date Of
Birth:",padx=2,pady=6)
        lblDateOfBirth.grid(row=7,column=2,sticky=W)
        txtDateOfBirth=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.DateOfBir
th,width=35)
        txtDateOfBirth.grid(row=7,column=3)

        lblPatientAddress=Label(DataframeLeft,font=("times new
roman",12,"bold"),text="Patient Address:",padx=2,pady=6)
        lblPatientAddress.grid(row=8,column=2,sticky=W)
        txtPatientAddress=Entry(DataframeLeft,font=("arial",13,"bold"),textvariable=self.patienta
ddress,width=35)
        txtPatientAddress.grid(row=8,column=3)

    # ==========DataFrameRight=======
```

```
        self.txtPrescription=Text(DataframeRight,font=("times new
roman",12,"bold"),width=55,height=16,padx=2,pady=6)
        self.txtPrescription.grid(row=0,column=0)

       #===========Buttons==================
        btnPrescription=Button(Buttonframe,command=self.iPrectioption,text="Prescription",bg=
"green",fg="white",font=("times new roman",12,"bold"),width=27,height=1,padx=2,pady=6)
        btnPrescription.grid(row=0,column=0)

        btnPrescriptionData=Button(Buttonframe,text="Prescription
Data",bg="green",fg="white",font=("times new
roman",12,"bold"),width=26,height=1,padx=2,pady=6,command=self.iprescriptionData)
        btnPrescriptionData.grid(row=0,column=1)

        btnUpdate=Button(Buttonframe,command=self.update_data,text="Update",bg="green",fg
="white",font=("times new roman",12,"bold"),width=26,height=1,padx=2,pady=6)
        btnUpdate.grid(row=0,column=2)

        btnDelete=Button(Buttonframe,command=self.idelete,text="Delete",bg="green",fg="whit
e",font=("times new roman",12,"bold"),width=27,height=1,padx=2,pady=6)
        btnDelete.grid(row=0,column=3)

        btnClear=Button(Buttonframe,command=self.clear_fields,text="Clear",bg="green",fg="w
hite",font=("times new roman",12,"bold"),width=26,height=1,padx=2,pady=6)
        btnClear.grid(row=0,column=4)

        btnExit=Button(Buttonframe,command=self.iExit,text="Exit",bg="green",fg="white",font
=("times new roman",12,"bold"),width=26,height=1,padx=2,pady=6)
        btnExit.grid(row=0,column=5)

       # ==============Table===============

        scroll_x=ttk.Scrollbar(detailsframe,orient=HORIZONTAL)
        scroll_y=ttk.Scrollbar(detailsframe,orient=VERTICAL)
        self.hospital_table=ttk.Treeview(detailsframe,column=("Nameoftablets","ref","dose","noo
ftablets","lot","issuedate","expdate","dailydose","storage","nhsnumber","pname","dob","addres
s"),xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)
        scroll_x.pack(side=BOTTOM,fill=X)
        scroll_y.pack(side=RIGHT,fill=Y)

        scroll_x=ttk.Scrollbar(command=self.hospital_table.xview)
        scroll_y=ttk.Scrollbar(command=self.hospital_table.yview)

        self.hospital_table.heading("Nameoftablets",text="Name Of Tablets")
        self.hospital_table.heading("ref",text="Refrence Number")
        self.hospital_table.heading("dose",text="Dose")
        self.hospital_table.heading("nooftablets",text="No Of table")
        self.hospital_table.heading("lot",text="Lot")
        self.hospital_table.heading("issuedate",text="Issue Date")
        self.hospital_table.heading("expdate",text="Exp Date")
        self.hospital_table.heading("dailydose",text="Daily Dose")
        self.hospital_table.heading("storage",text="Storage")
        self.hospital_table.heading("nhsnumber",text="NHS number")
```

```
        self.hospital_table.heading("pname",text="Patient Name")
        self.hospital_table.heading("dob",text="DOB")
        self.hospital_table.heading("address",text="Address")

        self.hospital_table["show"]="headings"
        self.hospital_table.pack(fill=BOTH,expand=1)

        self.hospital_table.column("Nameoftablets",width=100)
        self.hospital_table.column("ref",width=100)
        self.hospital_table.column("dose",width=100)
        self.hospital_table.column("nooftablets",width=100)
        self.hospital_table.column("lot",width=100)
        self.hospital_table.column("issuedate",width=100)
        self.hospital_table.column("expdate",width=100)
        self.hospital_table.column("dailydose",width=100)
        self.hospital_table.column("storage",width=100)
        self.hospital_table.column("nhsnumber",width=100)
        self.hospital_table.column("pname",width=100)
        self.hospital_table.column("dob",width=100)
        self.hospital_table.column("address",width=100)

        self.hospital_table.pack(fill=BOTH,expand=1)
        self.hospital_table.bind("<ButtonRelease-1>",self.get_cursor)
        self.fetch_data()


    #==========Functionality============

    def iprescriptionData(self):
        if self.Name_of_tablets.get()=="" or self.ref.get()=="":
            messagebox.showerror("Error","All fields are requirerd")
        else:
            try:
                conn=mysql.connector.connect(host="localhost",username="root",password="Mysql
@123#",database="mydata")
                my_cursor=conn.cursor()
                my_cursor.execute("INSERT INTO hospital1
VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",(
                    self.Name_of_tablets.get(),
                    self.ref.get(),
                    self.Dose.get(),
                    self.Numbersoftablets.get(),
                    self.Lot.get(),
                    self.issuedate.get(),
                    self.expdate.get(),
                    self.dailydose.get(),
                    self.StorageAdvice.get(),
                    self.nhsNumber.get(),
                    self.patientname.get(),
                    self.DateOfBirth.get(),
                    self.patientaddress.get()

                ))
```

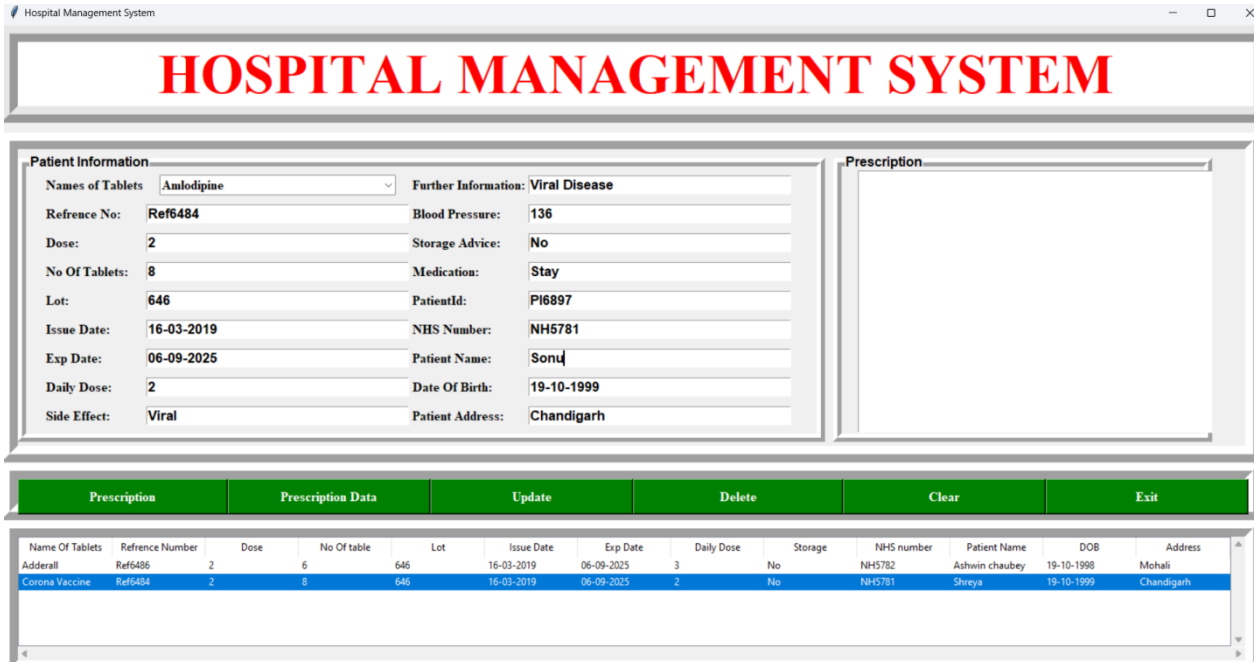
```
            conn.commit()
            self.fetch_data()
            conn.close()
            messagebox.showinfo("Success","Record has been inserted")
        except mysql.connector.Error as err:
            messagebox.showerror("Database error",f"error:{err}")


    def update_data(self):
        if self.ref.get() == "":
            messagebox.showerror("Error", "Refrence number is required to update a record")
            return

        try:
            conn = mysql.connector.connect(host="localhost", username="root",
password="Mysql@123#", database="mydata")
            my_cursor = conn.cursor()
            my_cursor.execute("""
                UPDATE hospital1 SET
                Name_of_tablets=%s, Dose=%s, Numbersoftablets=%s, Lot=%s, issuedate=%s,
expdate=%s, dailydose=%s,
                Storage=%s, NHSNumber=%s, patientname=%s, DOB=%s, patientaddress=%s
                WHERE Refrence_No=%s
                """, (
                self.Name_of_tablets.get(),
                self.Dose.get(),
                self.Numbersoftablets.get(),
                self.Lot.get(),
                self.issuedate.get(),
                self.expdate.get(),
                self.dailydose.get(),
                self.StorageAdvice.get(),
                self.nhsNumber.get(),
                self.patientname.get(),
                self.DateOfBirth.get(),
                self.patientaddress.get(),
                self.ref.get()
            ))

            conn.commit()
            self.fetch_data()
            conn.close()
            messagebox.showinfo("Success", "Record has been updated")

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



    def fetch_data(self):
        conn=mysql.connector.connect(host="localhost",username="root",password="Mysql@12
3#",database="mydata")
        my_cursor=conn.cursor()
```

```
        my_cursor.execute("select * from hospital1")
        rows=my_cursor.fetchall()
        if len(rows)!=0:
            self.hospital_table.delete(*self.hospital_table.get_children())
            for i in rows:
                self.hospital_table.insert("",END,values=i)
            conn.commit()
        conn.close()

    def get_cursor(self,event=""):
        cursor_row=self.hospital_table.focus()
        content=self.hospital_table.item(cursor_row)
        row=content["values"]
        self.Name_of_tablets.set(row[0])
        self.ref.set(row[1])
        self.Dose.set(row[2])
        self.Numbersoftablets.set(row[3])
        self.Lot.set(row[4])
        self.issuedate.set(row[5])
        self.expdate.set(row[6])
        self.dailydose.set(row[7])
        self.StorageAdvice.set(row[8])
        self.nhsNumber.set(row[9])
        self.patientname.set(row[10])
        self.DateOfBirth.set(row[11])
        self.patientaddress.set(row[12])

    def iPrectioption(self):
        self.txtPrescription.insert(END,"Name of Tablets:\t\t\t" +self.Name_of_tablets.get() +
"\n")
        self.txtPrescription.insert(END,"Refrence Number:\t\t\t" +self.ref.get() + "\n")
        self.txtPrescription.insert(END,"Dose:\t\t\t" +self.Dose.get() + "\n")
        self.txtPrescription.insert(END,"Number of Tablets:\t\t\t" +self.Numbersoftablets.get() +
"\n")
        self.txtPrescription.insert(END,"Lot:\t\t\t" +self.Lot.get() + "\n")
        self.txtPrescription.insert(END,"Issue Date:\t\t\t" +self.issuedate.get() + "\n")
        self.txtPrescription.insert(END,"Exp date:\t\t\t" +self.expdate.get() + "\n")
        self.txtPrescription.insert(END,"dailyDose:\t\t\t" +self.dailydose.get() + "\n")
        self.txtPrescription.insert(END,"Side Efffect:\t\t\t" +self.SideEffect.get() + "\n")
        self.txtPrescription.insert(END,"Further Information:\t\t\t" +self.FurtherInformation.get()
+ "\n")
        self.txtPrescription.insert(END,"StorageAdvice:\t\t\t" +self.StorageAdvice.get() + "\n")
        self.txtPrescription.insert(END,"DrivingUsingMachine:\t\t\t"
+self.DrivingUsingMachine.get() + "\n")
        self.txtPrescription.insert(END,"PatientId:\t\t\t" +self.PatientId.get() + "\n")
        self.txtPrescription.insert(END,"NHSNumber:\t\t\t" +self.nhsNumber.get() + "\n")
        self.txtPrescription.insert(END,"PatientName:\t\t\t" +self.patientname.get() + "\n")
        self.txtPrescription.insert(END,"DateOfBirth:\t\t\t" +self.DateOfBirth.get() + "\n")
        self.txtPrescription.insert(END,"PatientAddress:\t\t\t" +self.patientaddress.get() + "\n")
```

```
    def idelete(self):
        if self.ref.get() == "":
            messagebox.showerror("Error", "Refrence number is required to delete a record")
            return

        try:
            conn = mysql.connector.connect(host="localhost", username="root",
password="Mysql@123#", database="mydata")
            my_cursor = conn.cursor()
            query = "DELETE FROM hospital1 WHERE Refrence_No=%s"
            value = (self.ref.get(),)

            my_cursor.execute(query, value)

            conn.commit()
            conn.close()

            self.fetch_data()
            messagebox.showinfo("Delete", "Record has been deleted successfully")

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


    # Add this method inside the Hospital class
    def clear_fields(self):
        self.Name_of_tablets.set("")
        self.ref.set("")
        self.Dose.set("")
        self.Numbersoftablets.set("")
        self.Lot.set("")
        self.issuedate.set("")
        self.expdate.set("")
        self.dailydose.set("")
        self.SideEffect.set("")
        self.BloodPressure.set("")
        self.FurtherInformation.set("")
        self.StorageAdvice.set("")
        self.DrivingUsingMachine.set("")
        self.HowToUseMedication.set("")
        self.PatientId.set("")
        self.nhsNumber.set("")
        self.patientname.set("")
        self.DateOfBirth.set("")
        self.patientaddress.set("")
        self.txtPrescription.delete(1.0, END)


    def iExit(self):
        iExit=messagebox.askyesno("Hospital Management System", "Are you sure you want to
exit?")
        if iExit>0:
            root.destroy()
```

```
        return


root = Tk()
ob = Hospital(root)
root.mainloop()
```

Output:



Conclusion:

- **Database Integration**:
    - Adding records to the MySQL database with iprescriptionData, updating records in update_data, deleting records in idelete, and fetching all records in fetch_data.
- **GUI Layouts**:
    - Frames for structuring the interface (Dataframe, Buttonframe, and detailsframe), and various widgets like labels, entries, and a Treeview table for displaying data.
- **Event Handling**:
    - Functions such as get_cursor to retrieve the selected row's data for further operations, and iPrectioption to display a formatted prescription in the txtPrescription widget.