MEDICAL SHOP MANAGEMENT

Project Report
Submitted in Partial fulfilment
of the Degree of
Bachelors of Computer Applications

Supervisor's Name:	**Submitted By:**
Dr. Shivani Vats	**Name: Vansh Tanwar**
	Enrolment No.: 120920076
	Semester-VI

Jagannath University
Bahadurgarh (NCR)
(2020-23)

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teachers **Dr. Shivani Vats , Mr. Mohit Mathur, Ms. Manisha Tripathi** who gave me this golden opportunity to do this wonderful project on the topic **Medical Shop Management**, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them.

Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame. Without their support and suggestion, this project would not have been completed.

PROJECT CERTIFICATE

This is to certify that the project report entitled **Medical Shop Management** submitted to **Jagannath University Bahadurgarh** in partial fulfilment of the requirement for the award of the degree of **Bachelor of Computer Applications (BCA)**, is an original work carried out by **Vansh Tanwa**r **Enrolment No. : 120920076** under the guidance of **Dr. Shivani Vats .**

The matter embodied in this project is a genuine work done by the student and has not been submitted whether to this University or to any other University/Institute for the fulfilment of the requirement of any course of study.

Name of Student: - Vansh Tanwar	**Name of the Guide:** Dr. Shivani Vats
Signature of the Student -	**Signature of the Guide -**
Enrolment No.: - 120920076	**Date: -** 01/05/2023

INDEX

S No.	Name	Page No.
1.	Introduction	1
2.	Objectives	2
3.	Tools/Environment	3
4.	Analysis Document	4-5
5.	Test Case and Validations	6
6.	Program Code	7-26
7.	Input/Output	27-29
8.	Limitations	30
9.	Future Application of the project	31
10.	Bibliography	

BIBLIOGRAPHY

- https://www.python.org/ (for downloading libraries)
- https://stackoverflow.com/
- https://www.geeksforgeeks.org/python-gui-tkinter/?ref=gcse
- https://www.sqlite.org/download.html
- https://sqlitebrowser.org/dl/

INTRODUCTION

What is Medical Shop Management Project ?

The Medical Shop Management Project is a computer-based program for managing, monitoring, and recording medical store activities. Through automated features, it helps to increase the efficiency of medical stores. It also aids in the resolution of challenges with manual pharmacy management.

The Medical Shop Management, often known as the pharmacy information system, is a system that organizes and manages the drug usage process within pharmacies by storing data and enabling functionality.

A Medical Shop Management in Python with Graphical User Interface (GUI) with a SQLite3 database connectivity in python using Tkinter. A user can add, update, delete, search medicine details.

Why Medical Shop Management is Important?

A Medical Shop Management can also help you keep track of your inventory. Prescriptions must be exact and supplied in precise amounts, according to Pharmacy Management software. This can also improve the quality and satisfaction scores. You can also appropriately control or manage the expiration of drugs.

OBJECTIVES

- It keeps track of all information pertaining to Medicines, Companies, and Medicines.
- The Medical Shop Management ‘s major goal is to keep track of Medicines, Stocks, Inventory, Pharmacy.

TOOLS, ENVIRONMENT AND INFORMATION

Project Title:	**Medical Shop Management Project in Python**
Abstract :	Medical Shop Management Project in Python is a system that stores data and enables functionality that organizes and maintains the medication use process within pharmacies.
Project Type:	Desktop Application
Technology :	Visual Studio Code
Language:	Python
Database :	SQLite3

HARDWARE USED:

- MacBook Air
- Windows PC

SOFTWARE USED:

- Visual Studio Code
- Python
- DB Browser for SQLite

LIBRARY USED:

- Tkinter → (from tkinter import *)
- Python Imaging Library (PIL) → (from PIL import Image, ImageTk)
- Sqlite3 → (import sqlite3)

FILES USED:

- med.py
- pharmacy.db

ANALYSIS DOCUMENT

Data Dictionary:

Name	Size	Data Type
REF_NO	5	INTEGER NOT NULL
COMPANY_NAME	30	Text
TYPE_OF_MED	30	Text
MED_NAME	30	Text
LOT_NO	5	Text
ISSUE_DT	10	Text
EXP_DT	10	Text
USES	30	Text
SIDE_EFFECT	30	Text
PRECAUTION	30	Text
DOSAGE	30	Text
PRICE		NUMERIC
QUANTITY		INTEGER
PRIMARY KEY ("REF_NO" AUTOINCREMENT)		

Table 4.1: Data Dictionary for table 'Information'

Data flow diagrams (DFDs):

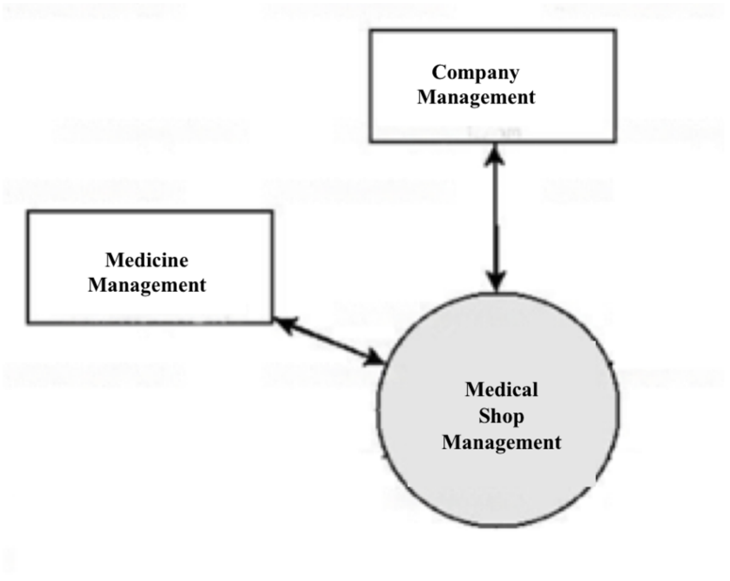


Fig 4.1: 0-Level DFD

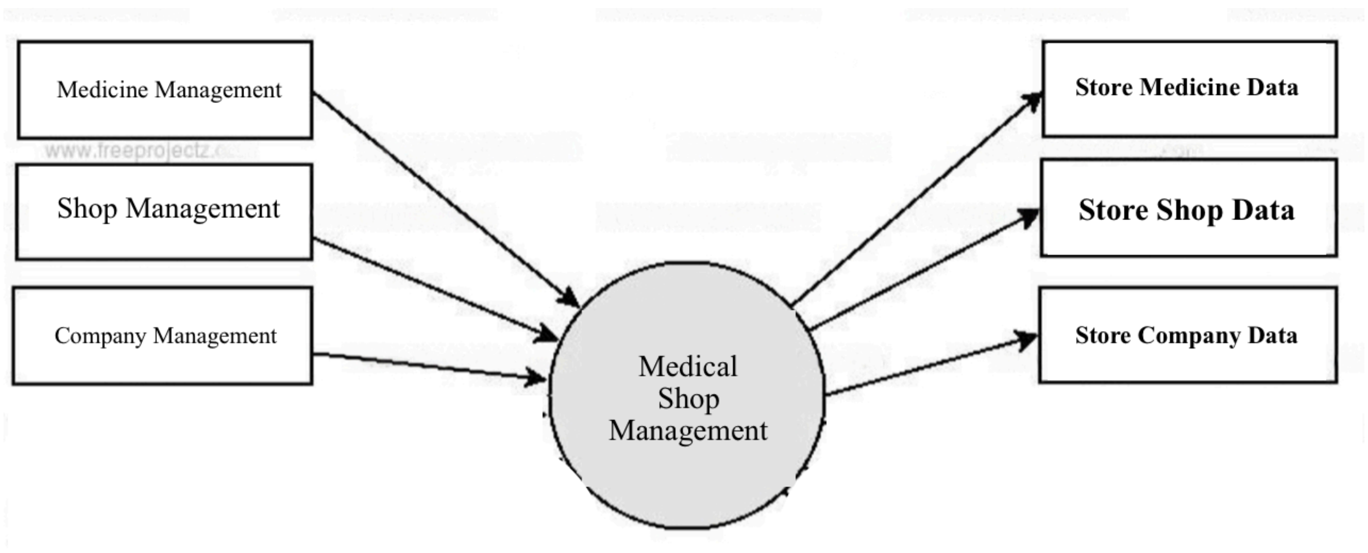


Fig 4.2:1-Level DFD

Entity Relationship diagrams (ERD):

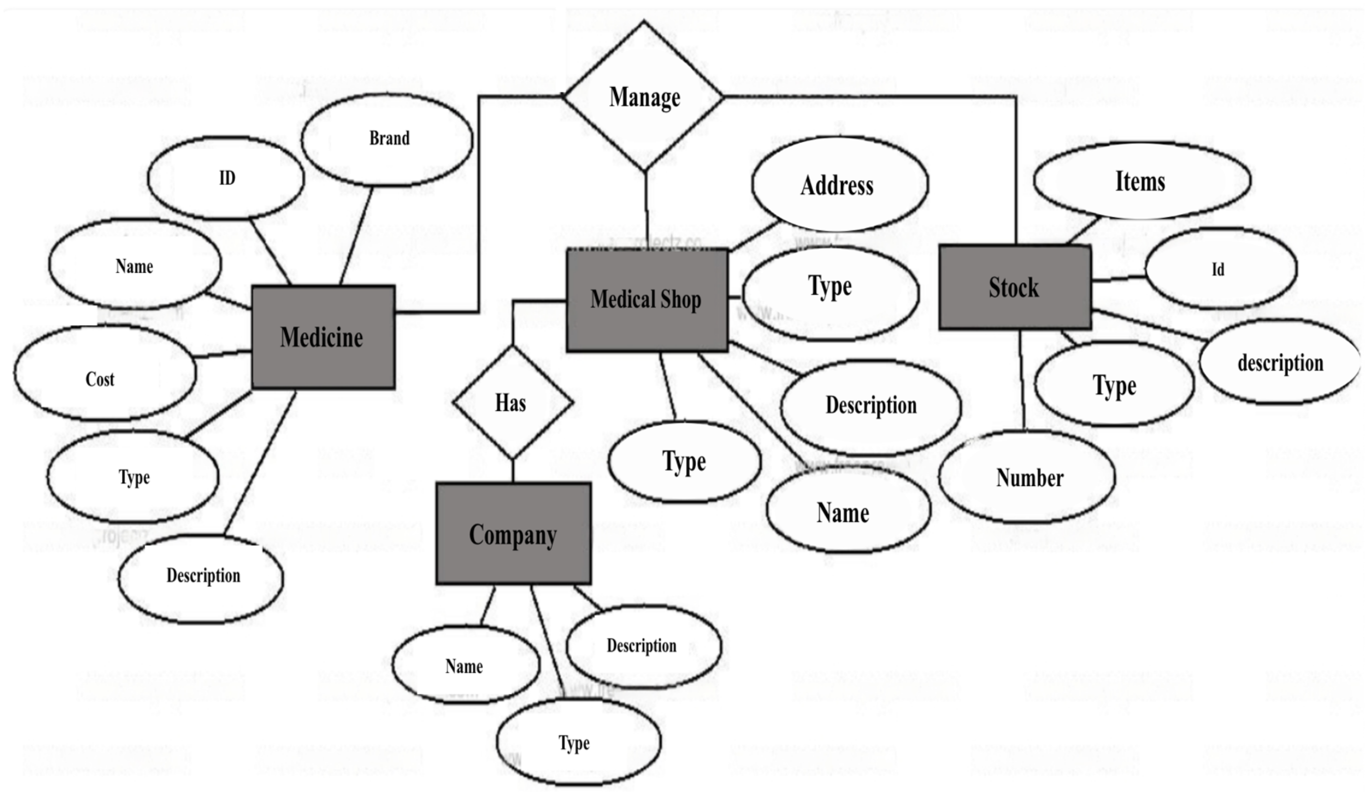


Fig 4.3: Entity Relationship diagrams (ERD)

TEST CASE AND VALIDATIONS

Test case	1
Description	Check whether the enter input format is correct or not
Testing Steps	Reference Number, Lot number, Tablet Quantity: Check that the enter value is Integer

Table 5.1

Test case	2
Description	Check whether the enter input format is correct or not
Testing Steps	Price : Check that the enter value is Float (for price)

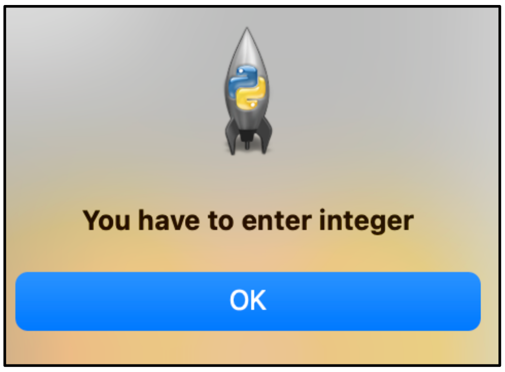
Table 5.2

Test case Scenario	**Test data**	**Output (Pass/Fail)**
Valid data as input for Ref No., Lot No, Tablet Quantity:	101	Pass
Invalid data as input for Ref No., Lot No, Tablet Quantity:	Abc123	Fail

Table 5.3

Test case Scenario	**Test data**	**Output (Pass/Fail)**
Valid data as input for Price	100.85 100	Pass
Invalid data as input for Price	Abc123	Fail

Table 5.4



You have to enter integer

Fig.: 5.1

PROGRAM CODE

```
#Libraries
from tkinter import *
from PIL import Image, ImageTk
import random
from tkinter import ttk, messagebox
import sqlite3
from tkinter import messagebox as ms

class Pharmacy:
  def __init__(self, root):
    self.root = root
    self.root.title("Medical Shop Management ")
    self.root.geometry("1350x800+0+0")
    self.root.resizable(False, False)
    self.root.iconbitmap("image/doc.ico")

    ##### ADD_MED VARIABLE ######
    self.ref_variable = StringVar()
    self.addmed_variable = StringVar()

    ########## MEDICINE DEPARTMENT VARIABLE #######
    self.refno_var = StringVar()
    self.companyname_var = StringVar()
    self.typemed_var = StringVar()
    self.medicine_var = StringVar()
    self.lotno_var = StringVar()
    self.issuedt_var = StringVar()
    self.expdt_var = StringVar()
    self.uses_var = StringVar()
    self.sideeffect_var = StringVar()
    self.warning_var = StringVar()
    self.dosage_var = StringVar()
    self.price_var = StringVar()
    self.quantity_var = StringVar()
```

```
        self.search_by = StringVar()
        self.search_txt = StringVar()

        ######## title animation #########
        self.txt = "Medical Shop Management "
        self.count = 0
        self.text = ""
        # self.color = ["black"]
        self.heading = Label(self.root, text=self.txt, font=(
            "times new roman", 30, "bold"), bg='grey', fg="blue", bd=9, relief=RIDGE)
        self.heading.pack(side=TOP, fill=X)
        self.slider()
        self.heading_color()

        ######### pharmacy logo label #######
        img1 = Image.open(r"image/new.png")
        # img1 = img1.resize((70, 45), Image.ANTIALIAS)
        img1 = img1.resize((70, 45), Image.LANCZOS)
        self.photoimg1 = ImageTk.PhotoImage(img1)
        b1 = Button(self.root, image=self.photoimg1,borderwidth=0, bg='white')
        b1.place(x=15, y=8)

        ###### Top Frame #####
        topframe = Frame(self.root, bg='white', bd=10, relief=RIDGE, padx=20)
        topframe.place(x=0, y=62, width=1350, height=400)

        ########  down button frame #######
        down_buttonframe = Frame(
            self.root, bg='white', bd=10, relief=RIDGE, padx=20)
        down_buttonframe.place(x=0, y=462, width=1350, height=60)

            ###### all buttons ######
        add_button = Button(down_buttonframe, text="Add Medicine",
command=self.addmedicine, font=(
    "arial", 12, "bold"), width=14, fg="black", bg="white", bd=3, relief=RIDGE,
activebackground="white", activeforeground="white")
        add_button.grid(row=0, column=0)

        update_button = Button(down_buttonframe, command=self.update_new, text="Update",
font=(
            "arial", 13, "bold"), width=14, fg="black", bg="white", bd=3, relief=RIDGE,
activebackground="white", activeforeground="white")

        update_button.grid(row=0, column=1)

        delete_button = Button(down_buttonframe, command=self.Delete_medinfo, text="Delete",
font=("arial", 13, "bold"), width=13,
                               fg="black", bg="white", bd=3, relief=RIDGE, activebackground="white",
activeforeground="white")
        delete_button.grid(row=0, column=2)

        reset_button = Button(down_buttonframe, text="Reset", command=self.clear_new,
font=("arial", 13, "bold"), width=12,
                              fg="black", bg="white", bd=3, relief=RIDGE, activebackground="white",
activeforeground="white")
        reset_button.grid(row=0, column=3)

        exit_button = Button(down_buttonframe, command=self.root.quit, text="Exit", font=(
            "arial", 13, "bold"), width=10, fg="black", bg="white", bd=3, relief=RIDGE,
activebackground="white", activeforeground="white")
        exit_button.grid(row=0, column=4)

        search_by = Label(down_buttonframe, text="Search By", font=(
            "arial", 15, "bold"), fg="black", bg="grey", bd=3, padx=3)
        search_by.grid(row=0, column=5, sticky=W)

        self.search_combo = ttk.Combobox(down_buttonframe, width=12, font=(
            "arial", 13, "bold"), state="readonly", textvariable=self.search_by)
        self.search_combo["values"] = ("Select Options", "Ref No.")
        self.search_combo.grid(row=0, column=6)
        self.search_combo.current(0)
        entry_button = Entry(down_buttonframe, font=("arial", 15, "bold"), fg="black",
                    bg="grey", bd=3, width=12, relief=RIDGE, textvariable=self.search_txt)
        entry_button.grid(row=0, column=7)

        search_button = Button(down_buttonframe, text="Search", font=("arial", 13, "bold"),
width=10, fg="black", bg="white",
                     bd=3, relief=RIDGE, activebackground="white", activeforeground="white",
command=self.search_data)



        search_button.grid(row=0, column=8)

        show_button = Button(down_buttonframe, text="Show All", font=("arial", 13, "bold"),
fg="black", bg="white",
                    width=10, bd=3, relief=RIDGE, activebackground="white",
activeforeground="white", command=self.fetch_new)
        show_button.grid(row=0, column=9)

        ######## left small frame #######
        left_smallframe = LabelFrame(topframe, bg='grey', bd=10, relief=RIDGE,padx=20,
text="Medicine Information", font=("arial", 13, "bold"), fg="black")
        left_smallframe.place(x=0, y=5, width=820, height=350)

          #### labeling & entry box #########

        # 1

        ref_label = Label(left_smallframe, text="Reference No. :", padx=2, pady=4, font=(
            "times new roman", 13, "bold"), bg="grey")
        ref_label.grid(row=0, column=0, sticky=W)

        self.ref_entry = Entry(left_smallframe, textvariable=self.refno_var, width=24, font=("times
new roman", 13, "bold"), fg="black", bg="white")
        self.ref_entry.grid(row=0, column=1)

        # 2
        company_label = Label(left_smallframe, text="Company Name  :", padx=2, pady=4,
font=(
            "times new roman", 13, "bold"), bg="grey")
        company_label.grid(row=1, column=0)

        self.company_entry = Entry(left_smallframe, textvariable=self.companyname_var,
width=24, font=(
            "times new roman", 13, "bold"), fg="black", bg="white")
        self.company_entry.grid(row=1, column=1)

        # 3
        type_label = Label(left_smallframe, text="Type Of Medicine :", padx=2, pady=4, font=(
            "times new roman", 13, "bold"), bg="grey")

        type_label.grid(row=2, column=0, sticky=W)

        self.type_combo = ttk.Combobox(left_smallframe, textvariable=self.typemed_var,
width=22, font=(
            "times new roman", 13, "bold"), state="readonly")
        self.type_combo["values"] = (
            " Select  ", "Tablet", "Capsule", "Injection", "Ayurvedic", "Drops", "Inhales")
        self.type_combo.grid(row=2, column=1)
        self.type_combo.current(0)

        # 4

        medname_label = Label(left_smallframe, text="Medicine Name :", padx=2, pady=4,
font=(
            "times new roman", 13, "bold"), bg="grey")
        medname_label.grid(row=3, column=0, sticky=W)

        conn = sqlite3.connect(database=r'pharmacy.db')
        my_cursor = conn.cursor()
        my_cursor.execute("Select Med_name from pharma")
        row_02 = my_cursor.fetchall()
        self.medname_combo = ttk.Combobox(left_smallframe, textvariable=self.medicine_var,
width=22, font=(
            "times new roman", 13, "bold"), state="")
        self.medname_combo["values"] = ("Select",row_02)
        self.medname_combo.grid(row=3, column=1)
        self.medname_combo.current(0)

        # 5

        lot_label = Label(left_smallframe, text=" Lot No. :", padx=2, pady=4, font=(
            "times new roman", 13, "bold"), bg="grey")
        lot_label.grid(row=4, column=0)

        self.lot_entry = Entry(left_smallframe, textvariable=self.lotno_var, width=24, font=(
            "times new roman", 13, "bold"), fg="black", bg="white")
        self.lot_entry.grid(row=4, column=1)




        # 6

        issue_label = Label(left_smallframe, text=" Issue Date :", padx=2, pady=4, font=(
            "times new roman", 13, "bold"), bg="grey")
        issue_label.grid(row=5, column=0)

        self.issue_entry = Entry(left_smallframe, textvariable=self.issuedt_var, width=24, font=(
            "times new roman", 13, "bold"), fg="black", bg="white")
        self.issue_entry.grid(row=5, column=1)

        # 7

        exp_label = Label(left_smallframe, text=" Expiry Date :", padx=2, pady=4, font=(
            "times new roman", 13, "bold"), bg="grey")
        exp_label.grid(row=6, column=0)
        self.exp_entry = Entry(left_smallframe, textvariable=self.expdt_var, width=24, font=(
            "times new roman", 13, "bold"), fg="black", bg="white")
        self.exp_entry.grid(row=6, column=1)

        # 8

        use_label = Label(left_smallframe, text=" Uses :", padx=2, pady=4, font=(
            "times new roman", 13, "bold"), bg="grey")
        use_label.grid(row=7, column=0)

        self.use_entry = Entry(left_smallframe, textvariable=self.uses_var, width=24, font=(
            "times new roman", 13, "bold"), fg="black", bg="white")
        self.use_entry.grid(row=7, column=1)

        # 9

        sideeffect_label = Label(left_smallframe, text=" Side Effect :", padx=2, pady=4, font=(
            "times new roman", 13, "bold"), bg="grey")
        sideeffect_label.grid(row=8, column=0)

        self.sideeffect_entry = Entry(left_smallframe, textvariable=self.sideeffect_var, width=24,
font=(
            "times new roman", 13, "bold"), fg="black", bg="white")

        self.sideeffect_entry.grid(row=8, column=1)

        # 10

        warn_label = Label(left_smallframe, text=" Prec & warning:", padx=2, pady=4, font=(
            "times new roman", 13, "bold"), bg="grey")
        warn_label.grid(row=9, column=0)

        self.warn_entry = Entry(left_smallframe, textvariable=self.warning_var, width=24, font=(
            "times new roman", 13, "bold"), fg="black", bg="white")
        self.warn_entry.grid(row=9, column=1)
```

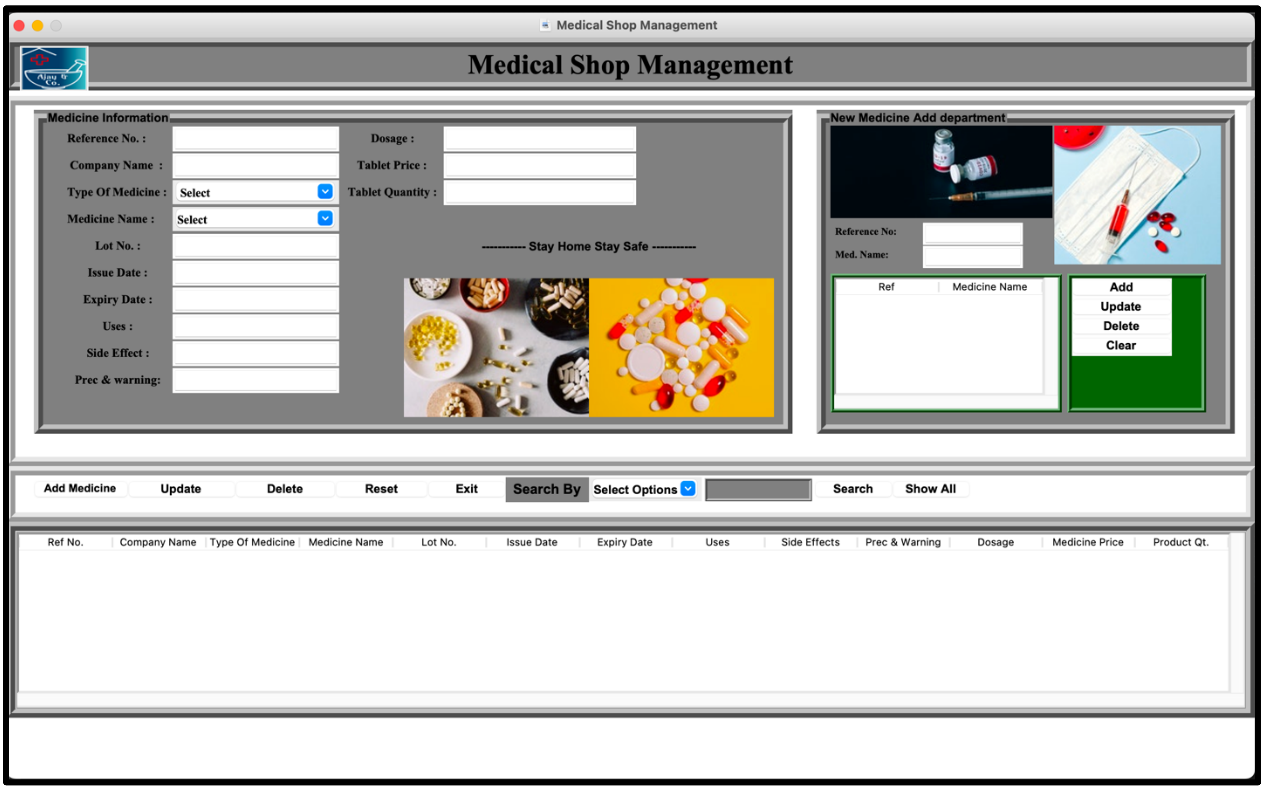
```
# 11

dosage_label = Label(left_smallframe, text=" Dosage :", padx=2, pady=4, font=(
    "times new roman", 13, "bold"), bg="grey")
dosage_label.grid(row=0, column=2)

self.dosage_entry = Entry(left_smallframe, textvariable=self.dosage_var, width=28, font=(
    "times new roman", 13, "bold"), fg="black", bg="white")
self.dosage_entry.grid(row=0, column=3)

# 12

price_label = Label(left_smallframe, text=" Tablet Price :", padx=2, pady=4, font=(
    "times new roman", 13, "bold"), bg="grey")
price_label.grid(row=1, column=2)

self.price_entry = Entry(left_smallframe, textvariable=self.price_var, width=28, font=(
    "times new roman", 13, "bold"), fg="black", bg="white")
self.price_entry.grid(row=1, column=3)

# 13

qt_label = Label(left_smallframe, text=" Tablet Quantity :", padx=2, pady=4, font=(
    "times new roman", 13, "bold"), bg="grey")
qt_label.grid(row=2, column=2)



self.qt_entry = Entry(left_smallframe, textvariable=self.quantity_var, width=28, font=(
    "times new roman", 13, "bold"), fg="black", bg="white")
self.qt_entry.grid(row=2, column=3)

    ######## image in left small frame #####
# image 1
self.bg = ImageTk.PhotoImage(file=r"image/med.jpg")
lbl_bg = Label(left_smallframe, image=self.bg)
```

```
lbl_bg.place(x=370, y=165, width=200, height=150)
# image 2
self.bgg = ImageTk.PhotoImage(file=r"image/medi.jpg")
lbl_bgg = Label(left_smallframe, image=self.bgg)
lbl_bgg.place(x=570, y=165, width=200, height=150)

# save life label
save_bgg = Label(left_smallframe, text="----------- Stay Home Stay Safe -----------",
                 font=("arial", 13, "bold"), bg='grey', fg="black")
save_bgg.place(x=370, y=120, width=400)

############ right frame #########
right_frame = LabelFrame(topframe, bg='grey', bd=10, relief=RIDGE, padx=5,text="New
Medicine Add department", font=("arial", 13, "bold"), fg="black")
right_frame.place(x=846, y=5, width=452, height=350)

 # image & label

# image 1
self.bg1 = ImageTk.PhotoImage(file=r"image/co.jpg")
lbl_bg1 = Label(right_frame, image=self.bg1)
lbl_bg1.place(x=0, y=0, width=240, height=100)
# image 2
self.bg2 = ImageTk.PhotoImage(file=r"image/inject.jpg")
lbl_bg2 = Label(right_frame, image=self.bg2)
lbl_bg2.place(x=242, y=0, width=180, height=150)

#### label & entry in right frame ####
# 1


no_label = Label(right_frame, text="Reference No:", font=(
   "times new roman", 11, "bold"), bg="grey")
no_label.place(x=0, y=105)

self.no_entry = Entry(right_frame, textvariable=self.ref_variable, width=16, font=(
   "times new roman", 11, "bold"), bg="white",fg="black")
self.no_entry.place(x=100, y=105)
# 2
med_label = Label(right_frame, text="Med. Name:", font=(
   "times new roman", 11, "bold"), bg="grey")
med_label.place(x=0, y=130)

self.med_entry = Entry(right_frame, textvariable=self.addmed_variable, width=16, font=(
   "times new roman", 11, "bold"), bg="white",fg="black")
self.med_entry.place(x=100, y=130)

#### in right frame small frame #####

newframe = Frame(right_frame, bg='darkgreen', bd=5, relief=RIDGE)
newframe.place(x=256, y=160, width=150, height=150)

 ###### button in this frame ###
add_button = Button(newframe, text="Add", font=("arial", 13, "bold"), width=13,
fg="black", bg="white",
                  bd=3, command=self.AddMed, relief=RIDGE, activebackground="white",
activeforeground="white")
add_button.grid(row=0, column=0)

updatenew_button = Button(newframe, text="Update", font=("arial", 13, "bold"),
width=13, fg="black", bg="white",
                     bd=3, command=self.Update_med, relief=RIDGE,
activebackground="white", activeforeground="white")
updatenew_button.grid(row=1, column=0)

delnew_button = Button(newframe, text="Delete", font=("arial", 13, "bold"), width=13,
fg="black", bg="white",
                   bd=3, relief=RIDGE, activebackground="white", activeforeground="white",
command=self.Delete_med)
delnew_button.grid(row=2, column=0)

clr_button = Button(newframe, text="Clear", command=self.clear_med, font=("arial", 13,
"bold"), width=13,
                  fg="black", bg="white", bd=3, relief=RIDGE, activebackground="white",
activeforeground="white")
clr_button.grid(row=3, column=0)

##### scrollbar frame in right frame ####
side_frame = Frame(right_frame, bd=4, relief=RIDGE, bg="dark green")
side_frame.place(x=0, y=160, width=250, height=150)

### scrollbar code ###

sc_x = ttk.Scrollbar(side_frame, orient=HORIZONTAL)
sc_y = ttk.Scrollbar(side_frame, orient=VERTICAL)
self.medicine_table = ttk.Treeview(side_frame, column=(
   "ref", "medname"), xscrollcommand=sc_x.set, yscrollcommand=sc_y.set)

sc_x.pack(side=BOTTOM, fill=X)
sc_y.pack(side=RIGHT, fill=Y)

sc_x.config(command=self.medicine_table.xview)
sc_y.config(command=self.medicine_table.yview)

self.medicine_table.heading("ref", text="Ref")
self.medicine_table.heading("medname", text="Medicine Name")

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

self.medicine_table.column("ref", width=100)
self.medicine_table.column("medname", width=100)

self.medicine_table.bind("<ButtonRelease-1>", self.medget_cursor)
self.fetch_datamed()

######### down frame #######
down_frame = Frame(self.root, bg='grey', bd=10, relief=RIDGE)
down_frame.place(x=0, y=522, width=1350, height=212)
########## scrollbar in down frame ########
scroll_frame = Frame(down_frame, bd=2, relief=RIDGE, bg="white")
scroll_frame.place(x=0, y=0, width=1330, height=192)

##### scrollbar code #####
scroll_x = ttk.Scrollbar(scroll_frame, orient=HORIZONTAL)
scroll_y = ttk.Scrollbar(scroll_frame, orient=VERTICAL)
self.info_table = ttk.Treeview(scroll_frame, column=("ref no", "comp name", "type",
"medi name", "lot no", "issue", "exp",
                                      "uses", "side effect", "warning", "dosage", "price", "product"),
xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)

scroll_x.pack(side=BOTTOM, fill=X)
scroll_y.pack(side=RIGHT, fill=Y)

scroll_x.config(command=self.info_table.xview)
scroll_y.config(command=self.info_table.yview)

self.info_table.heading("ref no", text="Ref No.")
self.info_table.heading("comp name", text="Company Name")
self.info_table.heading("type", text="Type Of Medicine")
self.info_table.heading("medi name", text="Medicine Name")
self.info_table.heading("lot no", text="Lot No.")
self.info_table.heading("issue", text="Issue Date")
self.info_table.heading("exp", text="Expiry Date")
self.info_table.heading("uses", text="Uses")
self.info_table.heading("side effect", text="Side Effects")
self.info_table.heading("warning", text="Prec & Warning")
self.info_table.heading("dosage", text="Dosage")
self.info_table.heading("price", text="Medicine Price")
self.info_table.heading("product", text="Product Qt.")

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

```
    self.info_table.column("ref no", width=100)
    self.info_table.column("comp name", width=100)
    self.info_table.column("type", width=100)

    self.info_table.column("medi name", width=100)
    self.info_table.column("lot no", width=100)
    self.info_table.column("issue", width=100)
    self.info_table.column("exp", width=100)
    self.info_table.column("uses", width=100)
    self.info_table.column("side effect", width=100)
    self.info_table.column("warning", width=100)
    self.info_table.column("dosage", width=100)
    self.info_table.column("price", width=100)
    self.info_table.column("product", width=100)

    self.info_table.bind("<ButtonRelease-1>", self.get_cursor)

    self.fetch_new()

####### MEDICINE ADD FUNCTIONALITY  DECLARATION #######

def AddMed(self):

    if self.ref_variable.get() == "" or self.addmed_variable.get() == "":
        messagebox.showerror("Error", "All fields are required")

    else:
        conn = sqlite3.connect(database=r'pharmacy.db')
        my_cursor = conn.cursor()
        my_cursor.execute("Insert into pharma(Ref_no,Med_name) values(?,?)", (
            self.ref_variable.get(),
            self.addmed_variable.get(),))

        conn.commit()
        self.fetch_datamed()
        conn.close()

        messagebox.showinfo("Success", "MEDICINE ADDED")

def fetch_datamed(self):
    conn = sqlite3.connect(database=r'pharmacy.db')
    my_cursor = conn.cursor()

    my_cursor.execute("select * from pharma")
    rows = my_cursor.fetchall()

    if len(rows) != 0:
        self.medicine_table.delete(*self.medicine_table.get_children())

        for i in rows:
            self.medicine_table.insert("", END, values=i)

        conn.commit()
        conn.close()

###### for show data on click #####

def medget_cursor(self, event=""):
    cursor_row = self.medicine_table.focus()
    content = self.medicine_table.item(cursor_row)
    row = content["values"]
    self.ref_variable.set(row[0])
    self.addmed_variable.set(row[1])

def Update_med(self):

    if self.ref_variable.get() == "" or self.addmed_variable.get()=="":

        messagebox.showerror("Error", "Ref No. and med name is required")
    else:
        try:
            conn = sqlite3.connect(database=r'pharmacy.db')
            my_cursor = conn.cursor()

            my_cursor.execute("Update pharma set Med_name=? where Ref_no=?", (
                                                                self.addmed_variable.get(),
                                                                self.ref_variable.get(),
                                                                ))

            conn.commit()
            messagebox.showinfo("Update", "Successfully Updated", parent=self.root)

            self.fetch_datamed()
            conn.close()
        except Exception as e:
            messagebox.showerror("Error",f"Error due to:{str(e)}",parent=self.root)

    def Delete_medinfo(self):
        if self.refno_var.get()=="":
            messagebox.showerror("Error","Ref no is required",parent=self.root)
        else:

            try:
                conn=sqlite3.connect(database=r'pharmacy.db')
                my_cursor=conn.cursor()

                my_cursor.execute("Delete from Information where REF_NO=?
",(self.refno_var.get(),))
                conn.commit()
                messagebox.showinfo("Delete","Successfully Deleted",parent=self.root)
                self.fetch_new()
            except Exception as e:
                messagebox.showerror("Error",f"Error due to:{str(e)}",parent=self.root)

    def Delete_med(self):
        if self.ref_variable.get()=="":
            messagebox.showerror("Error","Ref no is required",parent=self.root)
        else:


            try:
                conn=sqlite3.connect(database=r'pharmacy.db')
                my_cursor=conn.cursor()


                my_cursor.execute("Delete from pharma where Ref_no=? ",(self.ref_variable.get(),))
                conn.commit()
                messagebox.showinfo("Delete","Successfully Deleted",parent=self.root)
                self.fetch_datamed()
            except Exception as e:
                messagebox.showerror("Error",f"Error due to:{str(e)}",parent=self.root)




    def clear_med(self):
        self.ref_variable.set("")
        self.addmed_variable.set("")




    ######## MEDICINE DEPARTMENT FUNCTIONALITY #######
    def addmedicine(self):
        if self.refno_var.get() == "" or self.lotno_var.get() == "" or self.typemed_var.get() == "":
            messagebox.showerror("Error","All fields are required")
        else:
            conn=sqlite3.connect(database=r'pharmacy.db')
            new_cursor=conn.cursor()
            new_cursor.execute("Insert into
Information(REF_NO,COMPANY_NAME,TYPE_OF_MED,MED_NAME,LOT_NO,ISSUE_
DT,EXP_DT,USES,SIDE_EFFECT,PRECAUTION,DOSAGE,PRICE,QUANTITY) values(?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)",(


            self.refno_var.get(),
            self.companyname_var.get(),
            self.typemed_var.get(),
            self.medicine_var.get(),
            self.lotno_var.get(),
            self.issuedt_var.get(),
            self.expdt_var.get(),
            self.uses_var.get(),
            self.sideeffect_var.get(),
            self.warning_var.get(),
            self.dosage_var.get(),
            self.price_var.get(),
            self.quantity_var.get(),
            ))
            conn.commit()
            self.fetch_new()

            messagebox.showinfo("Success","Successfully added")

    def fetch_new(self):
        conn=sqlite3.connect(database=r'pharmacy.db')
        new_cursor=conn.cursor()
        new_cursor.execute("select * from Information")

        row=new_cursor.fetchall()

        if len(row)!=0:
            self.info_table.delete(*self.info_table.get_children())

            for i in row:
                self.info_table.insert("",END,values=i)

            conn.commit()



    def get_cursor(self,event=""):
        cursor_row=self.info_table.focus()
        content=self.info_table.item(cursor_row)
        row=content["values"]
        self.refno_var.set(row[0])
        self.companyname_var.set(row[1])
        self.typemed_var.set(row[2])
        self.medicine_var.set(row[3])
        self.lotno_var.set(row[4])
        self.issuedt_var.set(row[5])
        self.expdt_var.set(row[6])
        self.uses_var.set(row[7])
        self.sideeffect_var.set(row[8])
        self.warning_var.set(row[9])
        self.dosage_var.set(row[10])
        self.price_var.set(row[11])
        self.quantity_var.set(row[12])

    def update_new(self):

        if self.refno_var.get() == "" or self.lotno_var.get() == "" or self.typemed_var.get() == "":
            messagebox.showerror("Error","All fields are required")
        else:
            conn=sqlite3.connect(database=r'pharmacy.db')
            new_cursor=conn.cursor()

  new_cursor.execute("Update Information set
COMPANY_NAME=?,TYPE_OF_MED=?,MED_NAME=?,LOT_NO=?,ISSUE_DT=?,EXP_
DT=?,USES=?,SIDE_EFFECT=?,PRECAUTION=?,DOSAGE=?,PRICE=?,QUANTITY=?
where REF_NO=?",(

            self.companyname_var.get(),
            self.typemed_var.get(),
            self.medicine_var.get(),
            self.lotno_var.get(),
            self.issuedt_var.get(),
            self.expdt_var.get(),
            self.uses_var.get(),
            self.sideeffect_var.get(),
            self.warning_var.get(),
            self.dosage_var.get(),
            self.price_var.get(),
            self.quantity_var.get(),
            self.refno_var.get(),

            ))
            conn.commit()
            self.fetch_new()

            self.clear_new()
            messagebox.showinfo("Success","Successfully updated")

    def clear_new(self):
        self.refno_var.set("")
        self.companyname_var.set("")
        self.typemed_var.set("")
        self.medicine_var.set("")
        self.lotno_var.set("")
        self.issuedt_var.set("")
        self.expdt_var.set("")
        self.uses_var.set("")
        self.sideeffect_var.set("")
        self.warning_var.set("")
        self.dosage_var.set("")
        self.price_var.set("")
        self.quantity_var.set("")



    def search_data(self):

        conn=sqlite3.connect(database=r'pharmacy.db')
        new_cursor=conn.cursor()
        selected = self.search_combo.get()
        if selected == "Select Options":
            messagebox.showerror("Error","You have to choose an option")

        else:
```

```
      new_cursor.execute("Select * from Information where
REF_NO=?",(self.search_txt.get(),))
      row=new_cursor.fetchone()

      if len(row)!=0:
        self.info_table.delete(*self.info_table.get_children())

        for i in row:
          self.info_table.insert("",END,values=i)

        conn.commit()

  def slider(self):
    if self.count>=len(self.txt):
      self.count=-1
      self.text=""
      self.heading.config(text=self.text)
    else:
      self.text=self.text+self.txt[self.count]
      self.heading.config(text=self.text)
    self.count+=1
    self.heading.after(200,self.slider)

  def heading_color(self):
    self.heading.config(foreground='black')
    self.heading.after(100, self.heading_color)



if __name__ == '__main__':

  root=Tk()
  ph=Pharmacy(root)
  root.mainloop()
```

Input/Output



Medical Shop Management

Medical Shop Management

Medicine Information

Reference No. :	
Company Name :	
Type Of Medicine :	Select
Medicine Name :	Select
Lot No. :	
Issue Date :	
Expiry Date :	
Uses :	
Side Effect :	
Prec & warning:	
Dosage :	
Tablet Price :	
Tablet Quantity :	

---------- Stay Home Stay Safe ----------

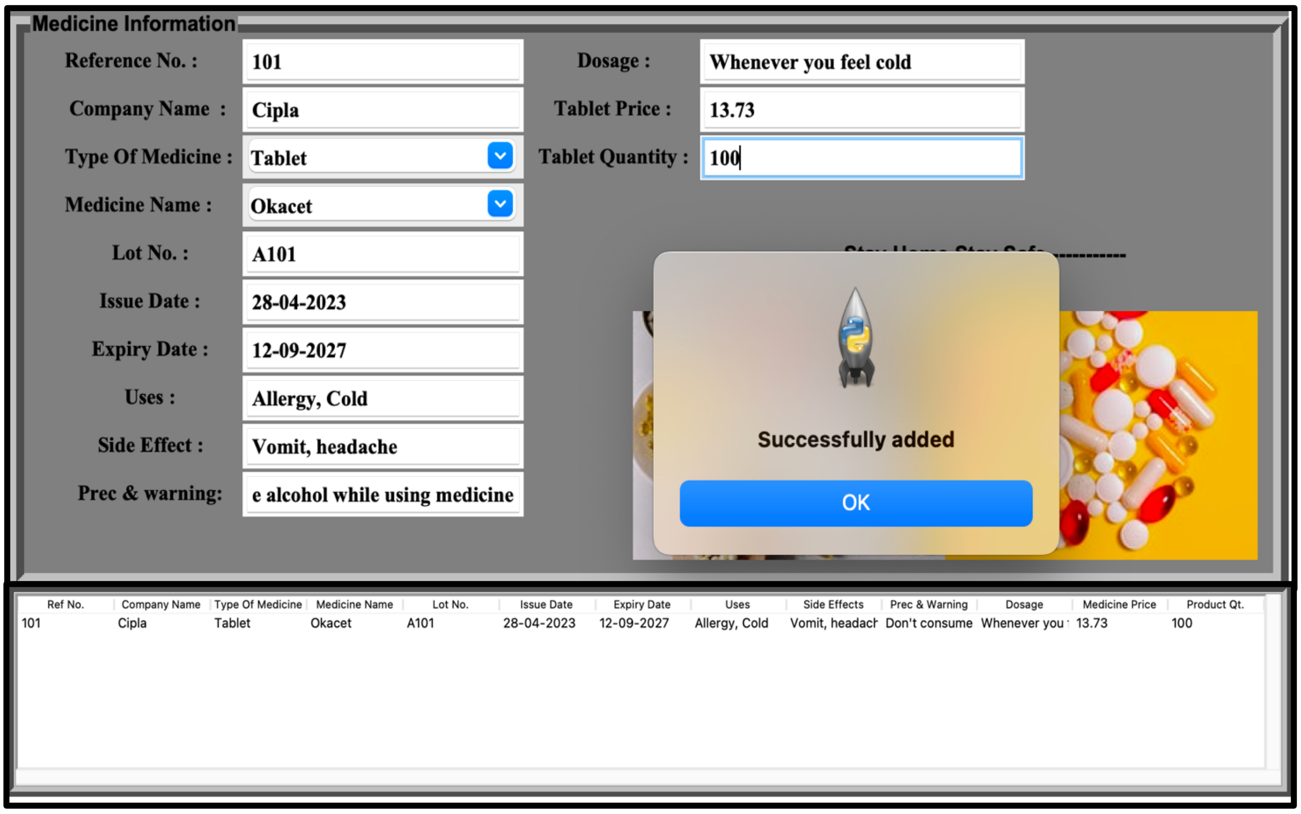
New Medicine Add department

Reference No:	
Med. Name:	

Ref	Medicine Name

Ref No.	Company Name	Type Of Medicine	Medicine Name	Lot No.	Issue Date	Expiry Date	Uses	Side Effects	Prec & Warning	Dosage	Medicine Price	Product Qt.

Fig 1: Homepage



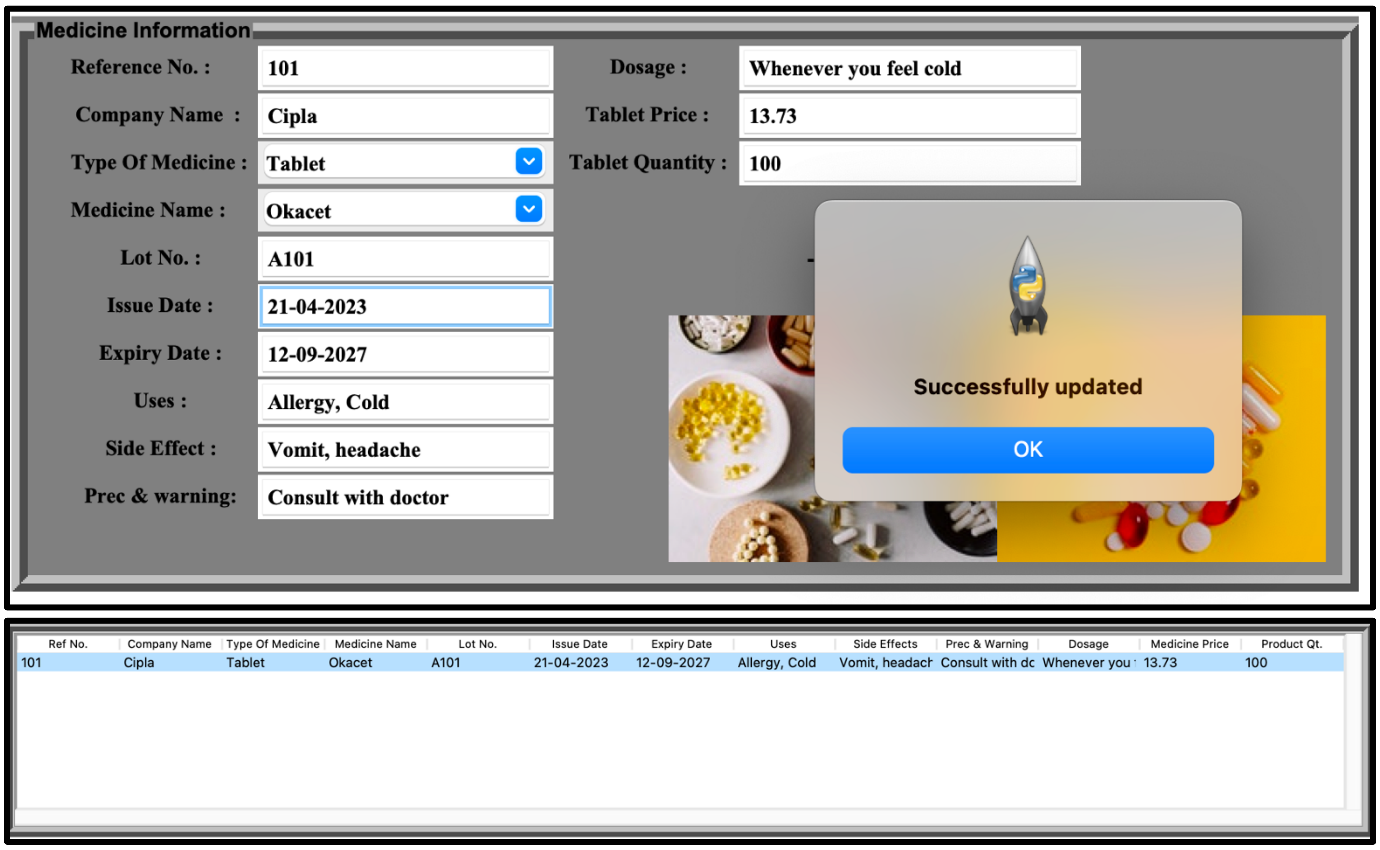
Medicine Information

Reference No. :	101
Company Name :	Cipla
Type Of Medicine :	Tablet
Medicine Name :	Okacet
Lot No. :	A101
Issue Date :	28-04-2023
Expiry Date :	12-09-2027
Uses :	Allergy, Cold
Side Effect :	Vomit, headache
Prec & warning:	e alcohol while using medicine
Dosage :	Whenever you feel cold
Tablet Price :	13.73
Tablet Quantity :	100

Successfully added

Ref No.	Company Name	Type Of Medicine	Medicine Name	Lot No.	Issue Date	Expiry Date	Uses	Side Effects	Prec & Warning	Dosage	Medicine Price	Product Qt.
101	Cipla	Tablet	Okacet	A101	28-04-2023	12-09-2027	Allergy, Cold	Vomit, headach	Don't consume	Whenever you	13.73	100

Fig 2: Add medicine:



Medicine Information

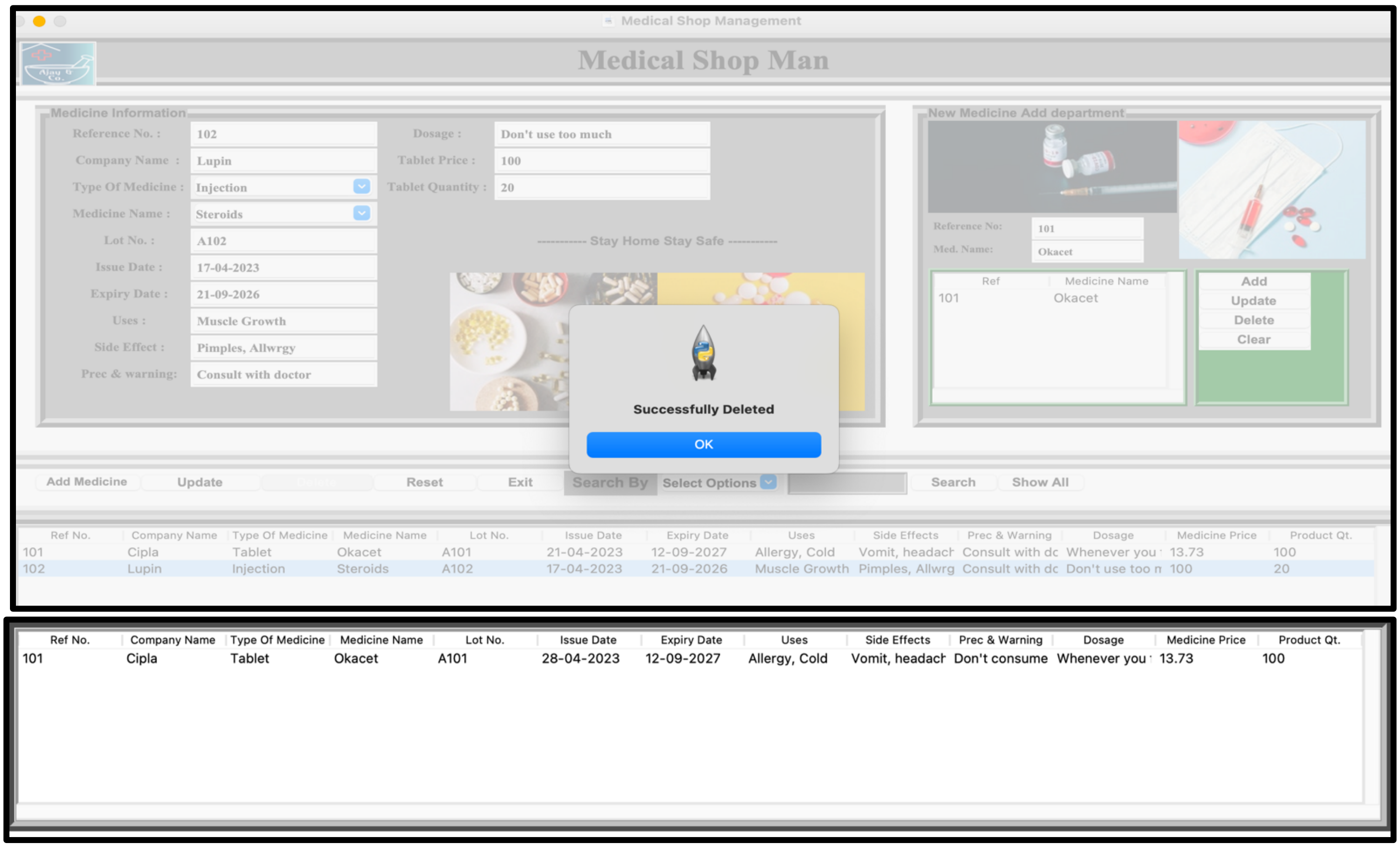
Reference No. :	101	Dosage :	Whenever you feel cold
Company Name :	Cipla	Tablet Price :	13.73
Type Of Medicine :	Tablet	Tablet Quantity :	100
Medicine Name :	Okacet		
Lot No. :	A101		
Issue Date :	21-04-2023		
Expiry Date :	12-09-2027		
Uses :	Allergy, Cold		
Side Effect :	Vomit, headache		
Prec & warning:	Consult with doctor		

Successfully updated

OK

Ref No.	Company Name	Type Of Medicine	Medicine Name	Lot No.	Issue Date	Expiry Date	Uses	Side Effects	Prec & Warning	Dosage	Medicine Price	Product Qt.
101	Cipla	Tablet	Okacet	A101	21-04-2023	12-09-2027	Allergy, Cold	Vomit, headach	Consult with dc	Whenever you	13.73	100

Fig 3: Update medicine:



Medicine Information

Reference No. :	102	Dosage :	Don't use too much
Company Name :	Lupin	Tablet Price :	100
Type Of Medicine :	Injection	Tablet Quantity :	20
Medicine Name :	Steroids		
Lot No. :	A102		
Issue Date :	17-04-2023		
Expiry Date :	21-09-2026		
Uses :	Muscle Growth		
Side Effect :	Pimples, Allwrgy		
Prec & warning:	Consult with doctor		

----------- Stay Home Stay Safe -----------

New Medicine Add department

Reference No:	101
Med. Name:	Okacet

Ref	Medicine Name
101	Okacet

Add
Update
Delete
Clear

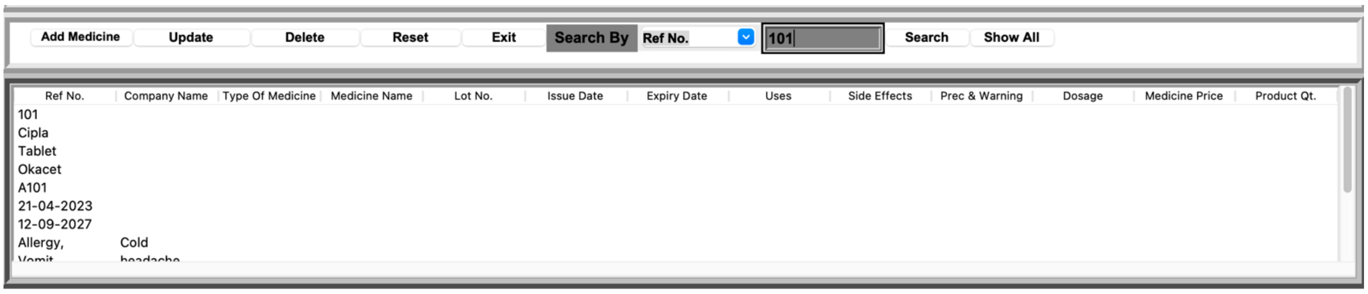
Successfully Deleted

OK

Ref No.	Company Name	Type Of Medicine	Medicine Name	Lot No.	Issue Date	Expiry Date	Uses	Side Effects	Prec & Warning	Dosage	Medicine Price	Product Qt.
101	Cipla	Tablet	Okacet	A101	21-04-2023	12-09-2027	Allergy, Cold	Vomit, headach	Consult with dc	Whenever you	13.73	100
102	Lupin	Injection	Steroids	A102	17-04-2023	21-09-2026	Muscle Growth	Pimples, Allwrg	Consult with dc	Don't use too n	100	20

Ref No.	Company Name	Type Of Medicine	Medicine Name	Lot No.	Issue Date	Expiry Date	Uses	Side Effects	Prec & Warning	Dosage	Medicine Price	Product Qt.
101	Cipla	Tablet	Okacet	A101	28-04-2023	12-09-2027	Allergy, Cold	Vomit, headach	Don't consume	Whenever you	13.73	100

Fig 4: Delete medicine:



```
Ref No.     Company Name  Type Of Medicine  Medicine Name  Lot No.  Issue Date  Expiry Date  Uses  Side Effects  Prec & Warning  Dosage  Medicine Price  Product Qt.
101
Cipla
Tablet
Okacet
A101
21-04-2023
12-09-2027
Allergy,    Cold
```

Fig 5: Search Medicine

LIMITATIONS

- Anyone can access the data of medicine.
- Anyone can add, update or delete the data.
- It can be resolve if there is username password.
- There is no feature of online order placing.

FUTURE APPLICATION OF THE PROJECT

- To keep the store updated with new stocks
- To check medicine expiry.
- To delete medicine of no use.