**System Implementations**

**Recommended System Requirements**

Processors: Intel® Core™ i3 processor 4300M at 2.60 GHz.

Disk space: 4 to 8 GB.

Operating systems: Windows® 10, MACOS, and UBUNTU.

Python Versions: 3.X.X or Higher.

**Minimum System Requirements**

Processors: Intel Atom® processor or Intel® Core™ i3 processor.

Disk space: 1 GB.

Operating systems: Windows 7 or later, MACOS, and UBUNTU.

Python Versions: 2.7.X, 3.9.X.

**ACKNOWLEDGEMENT**TTT

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

We would like to express my deep and sincere gratitude to my subject teacher, Mr. Amit Udiwal, for giving me the opportunity to do research and providing invaluable guidance throughout this research. His dynamism, vision, sincerity and motivation have deeply inspired me. He has taught me the methodology to carry out the research and to present the research works as clearly as and honour to work and study under his guidance. We are very much thankful to our Sr. Jasmin for giving valuable time and moral support to develop this software. We would like to take opportunity to extend my sincere thanks and gratitude to our parents for being a source of inspiration and providing time and freedom to develop this software project. We also feel indebted to my friends for the valuable suggestions during the project work.

Ayush Sankla

[Roll No.

Class XII

**CERTIFICATE**

This is to certify that the project on ‘Medical Billing System’ is a work done by Ayush Sankla fulfilment of CBSE’S AISSCE EXAMINATION 2020¢and has been carried out under my direct supervision and guidance. This report or a similar report on the topic has not been submitted for any other examination and does not form any other examination and does not form any other course undergone by the candidate.

Name:

Ayush Sankla[Roll No.

………………….

Signature of Teacher / Guide

Name: Mr. Amit Udiwal

Designation:

………………. ….………………

**REFERENCE**

The order to work on this project on ‘Medical Billing System’ the following books & literature are referred by me during the various phrases of department of the project.

• http://www.python.org/.

• http://www.itsourcecode.org/.

• http://www.wikipedia.org/.

• Informatics Practices for Class XII

- By Sumita Arora

• Together with informatics practices.

Other than the above mentioned books, the suggestions and supervision of my teacher and my class experience also helped me to develop this software project.

**Introduction**

This **Medical Billing System**has an easy to use interface and is built with respected customer service to allow stock maintenance, database access and electronic documentation for billing purposes. The program also encourages consumer computing discounts, frequent revenue, and identifies measures to avoid possible loss of revenue.

This system platform is best for educational purposes or can be used to simplify the manual maintenance process of records and cash flows by a wide range of retail and wholesale outlets. It’s more versatile and saves you time and money. The Medical Store Management System measures and shows the overall daily sales and also tells the items about the expiry of the coming week.

**Objective and**

**Scope of The Project**

The main objective of the Python Project on Medical Billing System is to manage the details of Company, Medicines. Inventory.

Stocks, Medical Shop. It manages all the information about Company. Sells, Medical Shop, Company. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Company, Medicines, Sells, Inventory. It tracks all the details about the Inventory, Stocks,Medical Shop

***Functions:***

• Manage the information of Inventory

• Shows the information and description of the Customer, Supplier

• To increase efficiency of managing the Customer, Inventory

**Medical Billing System**

from tkinter import \*

import time

import sqlite3

import random

import tempfil

import win32api

import win32print

f = ''

flag = ''

flags = ''

login = sqlite3.connect("admin.db")

l = login.cursor()

c = sqlite3.connect("medicine.db")

cur = c.cursor()

columns = ('Sl No', 'Name', 'Type', 'Quantity Left', 'Cost', 'Purpose', 'Expiry Date', 'Rack location', 'Manufacture')

def open\_win():

global apt, flag

flag = 'apt'

apt = Tk()

apt.title("Interface")

Label(apt, text="EVANZ MEDICAL STORE COMPANY").grid(row=0, column=0, columnspan=3)

Label(apt, text='\*' \* 80).grid(row=1, column=0, columnspan=3)

Label(apt, text='-' \* 80).grid(row=3, column=0, columnspan=3)

Label(apt, text="Stock Maintenance", bg='green', fg='white').grid(row=2, column=0)

Button(apt, text='New V.C.', width=25, bg='green', fg='white', command=val\_cus).grid(row=4, column=0)

Button(apt, text='Add product to Stock', bg='green', fg='white', width=25, command=stock).grid(row=5, column=0)

Button(apt, text='Delete product from Stock', bg='red', fg='white', width=25, command=delete\_stock).grid(row=6,

column=0)

Label(apt, text="Access Database", bg='blue', fg='white').grid(row=2, column=1)

Button(apt, text='Modify', width=15, bg='blue', fg='white', command=modify).grid(row=4, column=1)

Button(apt, text='Search', width=15, bg='blue', fg='white', command=search).grid(row=5, column=1)

Button(apt, text='Expiry Check', bg='red', fg='white', width=15, command=exp\_date).grid(row=6, column=1)

Label(apt, text="Handle Cash Flows", bg='skyblue', fg='black').grid(row=2, column=2)

Button(apt, text="Check Today's Revenue", bg='skyblue', fg='black', width=20, command=show\_rev).grid(row=5,

column=2)

Button(apt, text='Billing', width=20, bg='skyblue', fg='black', command=billing).grid(row=4, column=2)

Button(apt, text='Logout', bg='red', fg='white', width=20, command=again).grid(row=6, column=2)

apt.mainloop()

def delete\_stock():

global cur, c, flag, lb1, d

apt.destroy()

flag = 'd'

d = Tk()

d.title("Delete a product from Stock")

Label(d, text='Enter Product to delete:').grid(row=0, column=0)

Label(d, text='', width=30, bg='white').grid(row=0, column=1)

Label(d, text='Product').grid(row=2, column=0)

Label(d, text='Qty. Exp.dt. Cost ').grid(row=2, column=1)

ren()

b = Button(d, width=20, text='Delete', bg='red', fg='white', command=delt).grid(row=0, column=3)

b = Button(d, width=20, text='Main Menu', bg='green', fg='white', command=main\_menu).grid(row=5, column=3)

d.mainloop()

def ren():

global lb1, d, cur, c

def onvsb(\*args):

lb1.yview(\*args)

lb2.yview(\*args)

def onmousewheel():

lb1.ywiew = ('scroll', event.delta, 'units')

lb2.ywiew = ('scroll', event.delta, 'units')

return 'break'

cx = 0

vsb = Scrollbar(orient='vertical', command=onvsb)

lb1 = Listbox(d, width=25, yscrollcommand=vsb.set)

lb2 = Listbox(d, width=30, yscrollcommand=vsb.set)

vsb.grid(row=3, column=2, sticky=N + S)

lb1.grid(row=3, column=0)

lb2.grid(row=3, column=1)

lb1.bind('<MouseWheel>', onmousewheel)

lb2.bind('<MouseWheel>', onmousewheel)

cur.execute("select \*from med")

for i in cur:

cx += 1

s1 = [str(i[0]), str(i[1])]

s2 = [str(i[3]), str(i[6]), str(i[4])]

lb1.insert(cx, '. '.join(s1))

lb2.insert(cx, ' '.join(s2))

c.commit()

lb1.bind('<<ListboxSelect>>', sel\_del)

def sel\_del(e):

global lb1, d, cur, c, p, sl2

p = lb1.curselection()

print(p)

x = 0

sl2 = ''

cur.execute("select \* from med")

for i in cur:

print(x, p[0])

if x == int(p[0]):

sl2 = i[0]

break

x += 1

c.commit()

print(sl2)

Label(d, text=' ', bg='white', width=20).grid(row=0, column=1)

cur.execute('Select \* from med')

for i in cur:

if i[0] == sl2:

Label(d, text=i[0] + '. ' + i[1], bg='white').grid(row=0, column=1)

c.commit()

def delt():

global p, c, cur, d

cur.execute("delete from med where sl\_no=?", (sl2,))

c.commit()

ren()

def modify():

global cur, c, accept, flag, att, up, n, name\_, apt, st, col, col\_n

col = ('', '', 'type', 'qty\_left', 'cost', 'purpose', 'expdt', 'loc', 'mfg')

col\_n = ('', '', 'Type', 'Quantity Left', 'Cost', 'Purpose', 'Expiry Date', 'Rack location', 'Manufacture')

flag = 'st'

name\_ = ''

apt.destroy()

n = []

cur.execute("select \* from med")

for i in cur:

n.append(i[1])

c.commit()

st = Tk()

st.title('MODIFY')

Label(st, text='-' \* 48 + ' MODIFY DATABASE ' + '-' \* 48).grid(row=0, column=0, columnspan=6)

def onvsb(\*args):

name\_.yview(\*args)

def onmousewheel():

name\_.ywiew = ('scroll', event.delta, 'units')

return 'break'

cx = 0

vsb = Scrollbar(orient='vertical', command=onvsb)

vsb.grid(row=1, column=3, sticky=N + S)

name\_ = Listbox(st, width=43, yscrollcommand=vsb.set)

cur.execute("select \*from med")

for i in cur:

cx += 1

name\_.insert(cx, (str(i[0]) + '. ' + str(i[1])))

name\_.grid(row=1, column=1, columnspan=2)

c.commit()

name\_.bind('<MouseWheel>', onmousewheel)

name\_.bind('<<ListboxSelect>>', sel\_mn)

Label(st, text='Enter Medicine Name: ').grid(row=1, column=0)

Label(st, text='Enter changed Value of: ').grid(row=2, column=0)

att = Spinbox(st, values=col\_n)

att.grid(row=2, column=1)

up = Entry(st)

up.grid(row=2, column=2)

Button(st, width=10, text='Submit', bg='green', fg='white', command=save\_mod).grid(row=2, column=4)

Button(st, width=10, text='Reset', bg='red', fg='white', command=res).grid(row=2, column=5)

Button(st, width=10, text='Show data', bg='blue', fg='white', command=show\_val).grid(row=1, column=4)

Label(st, text='-' \* 120).grid(row=3, column=0, columnspan=6)

Button(st, width=10, text='Main Menu', bg='green', fg='white', command=main\_menu).grid(row=5, column=5)

st.mainloop()

def res():

global st, up

up = Entry(st)

up.grid(row=2, column=2)

Label(st, width=20, text=' ').grid(row=5, column=i)

def sel\_mn(e):

global n, name\_, name\_mn, sl, c, cur

name\_mn = ''

p = name\_.curselection()

print(p)

x = 0

sl = ''

cur.execute("select \* from med")

for i in cur:

print(x, p[0])

if x == int(p[0]):

sl = i[0]

break

x += 1

c.commit()

print(sl)

name\_nm = n[int(sl)]

print(name\_nm)

def show\_val():

global st, name\_mn, att, cur, c, col, col\_n, sl

for i in range(3):

Label(st, width=20, text=' ').grid(row=5, column=i)

cur.execute("select \* from med")

for i in cur:

for j in range(9):

if att.get() == col\_n[j] and sl == i[0]:

Label(st, text=str(i[0])).grid(row=5, column=0)

Label(st, text=str(i[1])).grid(row=5, column=1)

Label(st, text=str(i[j])).grid(row=5, column=2)

c.commit()

def save\_mod(): # save modified data

global cur, c, att, name\_mn, st, up, col\_n, sl

for i in range(9):

if att.get() == col\_n[i]:

a = col[i]

sql = "update med set '%s' = '%s' where sl\_no = '%s'" % (a, up.get(), sl)

cur.execute(sql)

c.commit()

Label(st, text='Updated!').grid(row=5, column=4)

def stock():

global cur, c, columns, accept, flag, sto, apt

apt.destroy()

flag = 'sto'

accept = [''] \* 10

sto = Tk()

sto.title('STOCK ENTRY')

Label(sto, text='ENTER NEW PRODUCT DATA TO THE STOCK').grid(row=0, column=0, columnspan=2)

Label(sto, text='-' \* 50).grid(row=1, column=0, columnspan=2)

for i in range(1, len(columns)):

Label(sto, width=15, text=' ' \* (14 - len(str(columns[i]))) + str(columns[i]) + ':').grid(row=i + 2, column=0)

accept[i] = Entry(sto)

accept[i].grid(row=i + 2, column=1)

Button(sto, width=15, text='Submit', bg='blue', fg='white', command=submit).grid(row=12, column=1)

Label(sto, text='-' \* 165).grid(row=13, column=0, columnspan=7)

Button(sto, width=15, text='Reset', bg='red', fg='white', command=reset).grid(row=12, column=0)

Button(sto, width=15, text='Refresh stock', bg='skyblue', fg='black', command=ref).grid(row=12, column=4)

for i in range(1, 6):

Label(sto, text=columns[i]).grid(row=14, column=i - 1)

Label(sto, text='Exp Rack Manufacturer ').grid(row=14, column=5)

Button(sto, width=10, text='Main Menu', bg='green', fg='white', command=main\_menu).grid(row=12, column=5)

ref()

sto.mainloop()

def ref():

global sto, c, cur

def onvsb(\*args):

lb1.yview(\*args)

lb2.yview(\*args)

lb3.yview(\*args)

lb4.yview(\*args)

lb5.yview(\*args)

lb6.yview(\*args)

def onmousewheel():

lb1.ywiew = ('scroll', event.delta, 'units')

lb2.ywiew = ('scroll', event.delta, 'units')

lb3.ywiew = ('scroll', event.delta, 'units')

lb4.ywiew = ('scroll', event.delta, 'units')

lb5.ywiew = ('scroll', event.delta, 'units')

lb6.ywiew = ('scroll', event.delta, 'units')

return 'break'

cx = 0

vsb = Scrollbar(orient='vertical', command=onvsb)

lb1 = Listbox(sto, yscrollcommand=vsb.set)

lb2 = Listbox(sto, yscrollcommand=vsb.set)

lb3 = Listbox(sto, yscrollcommand=vsb.set, width=10)

lb4 = Listbox(sto, yscrollcommand=vsb.set, width=7)

lb5 = Listbox(sto, yscrollcommand=vsb.set, width=25)

lb6 = Listbox(sto, yscrollcommand=vsb.set, width=37)

vsb.grid(row=15, column=6, sticky=N + S)

lb1.grid(row=15, column=0)

lb2.grid(row=15, column=1)

lb3.grid(row=15, column=2)

lb4.grid(row=15, column=3)

lb5.grid(row=15, column=4)

lb6.grid(row=15, column=5)

lb1.bind('<MouseWheel>', onmousewheel)

lb2.bind('<MouseWheel>', onmousewheel)

lb3.bind('<MouseWheel>', onmousewheel)

lb4.bind('<MouseWheel>', onmousewheel)

lb5.bind('<MouseWheel>', onmousewheel)

lb6.bind('<MouseWheel>', onmousewheel)

cur.execute("select \*from med")

for i in cur:

cx += 1

seq = (str(i[0]), str(i[1]))

lb1.insert(cx, '. '.join(seq))

lb2.insert(cx, i[2])

lb3.insert(cx, i[3])

lb4.insert(cx, i[4])

lb5.insert(cx, i[5])

lb6.insert(cx, i[6] + ' ' + i[7] + ' ' + i[8])

c.commit()

def reset():

global sto, accept

for i in range(1, len(columns)):

Label(sto, width=15, text=' ' \* (14 - len(str(columns[i]))) + str(columns[i]) + ':').grid(row=i + 2, column=0)

accept[i] = Entry(sto)

accept[i].grid(row=i + 2, column=1)

exp = Tk()

exp.title('EXPIRY CHECK')

Label(exp, text='Today : ' + str(now[2]) + '/' + str(now[1]) + '/' + str(now[0])).grid(row=0, column=0,

columnspan=3)

Label(exp, text='Selling Expired Medicines and Drugs is Illegal').grid(row=1, column=0, columnspan=3)

Label(exp, text='-' \* 80).grid(row=2, column=0, columnspan=3)

s = Spinbox(exp, values=n)

s.grid(row=3, column=0)

Button(exp, text='Check Expiry date', bg='red', fg='white', command=s\_exp).grid(row=3, column=1)

Label(exp, text='-' \* 80).grid(row=4, column=0, columnspan=3)

if flags == 'apt1':

Button(exp, text='Main Menu', bg='green', fg='white', command=main\_cus).grid(row=5, column=2)

else:

Button(exp, width=20, text='Check Products expiring', bg='red', fg='white', command=exp\_dt).grid(row=5,

column=0)

Button(exp, text='Main Menu', bg='green', fg='white', command=main\_menu).grid(row=5, column=2)

exp.mainloop()

def s\_exp():

global c, cur, s, exp, top

from datetime import date

now = time.localtime()

d1 = date(now[0], now[1], now[2])

cur.execute("select \* from med")

for i in cur:

if (i[1] == s.get()):

q = i[6]

d2 = date(int('20' + q[8:10]), int(q[3:5]), int(q[0:2]))

if d1 > d2:

Label(exp, text='EXPIRED! on ' + i[6]).grid(row=3, column=2)

top = Tk()

Label(top, text='EXPIRED!').pack()

else:

Label(exp, text=i[6]).grid(row=3, column=2)

c.commit()

def exp\_dt():

global c, cur, exp, top

x = 0

z = 1

from datetime import datetime, timedelta

N = 7

dt = datetime.now() + timedelta(days=N)

d = str(dt)

from datetime import date

now = time.localtime()

d1 = date(now[0], now[1], now[2])

d3 = date(int(d[0:4]), int(d[5:7]), int(d[8:10]))

Label(exp, text='S.No' + ' ' + 'Name' + ' Qty. ' + 'Exp\_date').grid(row=6, column=0, columnspan=2)

cur.execute("select \* from med")

for i in cur:

s = i[6]

d2 = date(int('20' + s[8:10]), int(s[3:5]), int(s[0:2]))

if d1 < d2 < d3:

Label(exp, text=str(z) + '. ' + str(i[1]) + ' ' + str(i[3]) + ' ' + str(i[6])).grid(row=x + 7,

column=0,

columnspan=2)

x += 1

z += 1

elif d1 > d2:

top = Tk()

Label(top, width=20, text=str(i[1]) + ' is EXPIRED!').pack()

c.commit()

def refresh():

global cur, c, st, lb1, lb2, vsb

def onvsb(\*args):

lb1.yview(\*args)

lb2.yview(\*args)

def onmousewheel():

lb1.ywiew = ('scroll', event.delta, 'units')

lb2.ywiew = ('scroll', event.delta, 'units')

return 'break'

cx = 0

vsb = Scrollbar(orient='vertical', command=onvsb)

lb1 = Listbox(st, width=25, yscrollcommand=vsb.set)

lb2 = Listbox(st, width=25, yscrollcommand=vsb.set)

vsb.grid(row=8, column=2, sticky=N + S)

lb1.grid(row=8, column=0)

lb2.grid(row=8, column=1)

lb1.bind('<MouseWheel>', onmousewheel)

lb2.bind('<MouseWheel>', onmousewheel)

cur.execute("select \*from med")

for i in cur:

cx += 1

lb1.insert(cx, str(i[0]) + '. ' + str(i[1]))

lb2.insert(cx, ' ' + str(i[7]) + ' ' + str(i[3]) + ' PHP ' + str(i[4]))

c.commit()

lb1.bind('<<ListboxSelect>>', select\_mn)

def select\_mn(e):

global st, lb1, n, p, nm, sl1

p = lb1.curselection()

x = 0

sl1 = ''

from datetime import date

now = time.localtime()

d1 = date(now[0], now[1], now[2])

cur.execute("select \* from med")

for i in cur:

if x == int(p[0]):

sl1 = int(i[0])

break

x += 1

c.commit()

print(sl1)

nm = n[x]

print(nm)

def append2bill():

global st, names, nm, qty, sl, cur, c, sl1

sl.append(sl1)

names.append(nm)

qty.append(qtys.get())

print(qty)

print(sl[len(sl) - 1], names[len(names) - 1], qty[len(qty) - 1])

def blue():

global st, c, cur, named, addd, t, vc\_id

cur.execute("select \* from cus")

for i in cur:

if vc\_id.get() != '' and int(vc\_id.get()) == i[2]:

named = i[0]

addd = i[1]

Label(st, text=named, width=20).grid(row=1, column=1)

Label(st, text=addd, width=20).grid(row=2, column=1)

Label(st, text=i[2], width=20).grid(row=3, column=1)

Label(st, text='Valued Customer!').grid(row=4, column=1)

t = 1

break

c.commit()

def make\_bill():

global t, c, B, cur, st, names, qty, sl, named, addd, name1, add, det, vc\_id

price = [0.0] \* 10

q = 0

det = ['', '', '', '', '', '', '', '']

det[2] = str(sl)

for i in range(len(sl)):

print(sl[i], ' ', qty[i], ' ', names[i])

for k in range(len(sl)):

cur.execute("select \* from med where sl\_no=?", (sl[k],))

for i in cur:

price[k] = int(qty[k]) \* float(i[4])

print(qty[k], price[k])

cur.execute("update med set qty\_left=? where sl\_no=?", (int(i[3]) - int(qty[k]), sl[k]))

c.commit()

det[5] = str(random.randint(100, 999))

B = 'bill\_' + str(det[5]) + '.txt'

total = 0.00

for i in range(10):

if price[i] != '':

total += price[i] # totalling

m = '\n\n\n'

m += "===============================================\n"

m += " No :%s\n\n" % det[5]

m += " EVANZ MEDICAL STORE COMPANY\n"

m += " BINALBAGAN BRANCH, NEGROS OCCIDENTAL\n\n"

m += "-----------------------------------------------\n"

if t == 1:

m += "Name: %s\n" % named

m += "Address: %s\n" % addd

det[0] = named

det[1] = addd

cur.execute('select \* from cus')

for i in cur:

if i[0] == named:

det[7] = i[2]

else:

m += "Name: %s\n" % name1.get()

m += "Address: %s\n" % add.get()

det[0] = name1.get()

det[1] = add.get()

m += "-----------------------------------------------\n"

m += "Product Qty. Price\n"

m += "-----------------------------------------------\n"

for i in range(len(sl)):

if names[i] != 'nil':

s1 = ' '

s1 = (names[i]) + (s1 \* (27 - len(names[i]))) + s1 \* (3 - len(qty[i])) + qty[i] + s1 \* (

15 - len(str(price[i]))) + str(price[i]) + '\n'

m += s1

m += "\n-----------------------------------------------\n"

if t == 1:

ntotal = total \* 0.8

m += 'Total' + (' ' \* 25) + (' ' \* (15 - len(str(total)))) + str(total) + '\n'

m += "Valued customer Discount" + (' ' \* (20 - len(str(total - ntotal)))) + '-' + str(total - ntotal) + '\n'

m += "-----------------------------------------------\n"

m += 'Total' + (' ' \* 25) + (' ' \* (12 - len(str(ntotal)))) + 'PHP ' + str(ntotal) + '\n'

det[3] = str(ntotal)

else:

m += 'Total' + (' ' \* 25) + (' ' \* (12 - len(str(total)))) + 'PHP ' + str(total) + '\n'

det[3] = str(total)

m += "-----------------------------------------------\n\n"

m += "Dealer 's signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

m += "===============================================\n"

print(m)

p = time.localtime()

det[4] = str(p[2]) + '/' + str(p[1]) + '/' + str(p[0])

det[6] = m

bill = open(B, 'w')

bill.write(m)

bill.close()

cb = ('cus\_name', 'cus\_add', 'items', 'Total\_cost', 'bill\_dt', 'bill\_no', 'bill', 'val\_id')

cur.execute('insert into bills values(?,?,?,?,?,?,?,?)',

(det[0], det[1], det[2], det[3], det[4], det[5], det[6], det[7]))

c.commit()

def print\_bill():

win32api.ShellExecute(0, "print", B, '/d:"%s"' % win32print.GetDefaultPrinter(), ".", 0)

def show\_rev():

global c, cur, flag, rev

apt.destroy()

cb = ('cus\_name', 'cus\_add', 'items', 'Total\_cost', 'bill\_dt', 'bill\_no', 'bill', 'val\_id')

flag = 'rev'

rev = Tk()

total = 0.0

today = str(time.localtime()[2]) + '/' + str(time.localtime()[1]) + '/' + str(time.localtime()[0])

Label(rev, text='Today: ' + today).grid(row=0, column=0)

cur.execute('select \* from bills')

for i in cur:

if i[4] == today:

total += float(i[3])

print(total)

Label(rev, width=22, text='Total revenue: PHP ' + str(total), bg='blue', fg='white').grid(row=1, column=0)

cx = 0

vsb = Scrollbar(orient='vertical')

lb1 = Listbox(rev, width=25, yscrollcommand=vsb.set)

vsb.grid(row=2, column=1, sticky=N + S)

lb1.grid(row=2, column=0)

vsb.config(command=lb1.yview)

cur.execute("select \* from bills")

for i in cur:

if i[4] == today:

cx += 1

lb1.insert(cx, 'Bill No.: ' + str(i[5]) + ' : PHP ' + str(i[3]))

c.commit()

Button(rev, text='Main Menu', bg='green', fg='white', command=main\_menu).grid(row=15, column=0)

rev.mainloop()

def search():

global c, cur, flag, st, mn, sym, flags

flag = 'st'

apt.destroy()

cur.execute("Select \* from med")

symp = ['nil']

med\_name = ['nil']

for i in cur:

symp.append(i[5])

med\_name.append(i[1])

st = Tk()

st.title('SEARCH')

Label(st, bg='green', fg='white', text=' SEARCH FOR MEDICINE ').grid(row=0, column=0, columnspan=3)

Label(st, text='~' \* 40).grid(row=1, column=0, columnspan=3)

Label(st, text='Symptom Name').grid(row=3, column=0)

sym = Spinbox(st, values=symp)

sym.grid(row=3, column=1)

Button(st, width=15, text='Search', bg='blue', fg='white', command=search\_med).grid(row=3, column=2)

Label(st, text='-' \* 70).grid(row=4, column=0, columnspan=3)

if flags == 'apt1':

Button(st, width=15, text='Main Menu', bg='green', fg='white', command=main\_cus).grid(row=6, column=2)

else:

Button(st, width=15, text='Main Menu', bg='green', fg='white', command=main\_menu).grid(row=6, column=2)

st.mainloop()

def search\_med():

global c, cur, st, sym, columns

cur.execute("select \* from med")

y = []

x = 0

for i in cur:

if i[5] == sym.get():

y.append(

str(i[0]) + '. ' + str(i[1]) + ' PHP ' + str(i[4]) + ' Rack : ' + str(i[7]) + ' Mfg : ' + str(

i[8]))

x = x + 1

top = Tk()

for i in range(len(y)):

Label(top, text=y[i]).grid(row=i, column=0)

Button(top, text='OK', command=top.destroy).grid(row=5, column=0)

c.commit()

top.mainloop()

def val\_cus():

global val, flag, dbt, name\_vc, add\_vc, cur, c, vc\_id

apt.destroy()

cur.execute("select \* from cus")

flag = 'val'

val = Tk()

Label(val, bg='blue', fg='white', text="\*\*\*\*ENTER VALUED CUSTOMER DETAILS\*\*\*\*").grid(row=0, column=0, columnspan=3)

Label(val, text="-" \* 60).grid(row=1, column=0, columnspan=3)

Label(val, text="Name: ").grid(row=2, column=0)

name\_vc = Entry(val)

name\_vc.grid(row=2, column=1)

Label(val, text="Address: ").grid(row=3, column=0)

add\_vc = Entry(val)

add\_vc.grid(row=3, column=1)

Label(val, text="Value Id: ").grid(row=4, column=0)

vc\_id = Entry(val)

vc\_id.grid(row=4, column=1)

Button(val, text='Submit', bg='blue', fg='white', command=val\_get).grid(row=5, column=1)

Button(val, text='Main Menu', bg='green', fg='white', command=main\_menu).grid(row=5, column=2)

Label(val, text='-' \* 60).grid(row=6, column=0, columnspan=3)

val.mainloop()

def val\_get():

global name\_vc, add\_vc, val, dbt, c, cur, apt, vc\_id

cur.execute("insert into cus values(?,?,?)", (name\_vc.get(), add\_vc.get(), vc\_id.get()))

l.execute("insert into log values(?,?)", (name\_vc.get(), vc\_id.get()))

cur.execute("select \* from cus")

for i in cur:

print(i[0], i[1], i[2])

c.commit()

login.commit()

def again():

global un, pwd, flag, root, apt

if flag == 'apt':

apt.destroy()

root = Tk()

root.geometry('300x150')

root.title('ITSOURCECODE COMPANY')

Label(root, text='EVANZ MEDICAL STORE COMPANY').grid(row=0, column=0, columnspan=5)

Label(root, text="BINALBAGAN BRANCH, NEGROS OCCIDENTAL").grid(row=1, column=0, columnspan=5)

Label(root, text='-------------------------------------------------------').grid(row=2, column=0, columnspan=5)

Label(root, text='Username').grid(row=3, column=0)

un = Entry(root, width=30)

un.grid(row=3, column=1)

Label(root, text='Password').grid(row=4, column=0)

pwd = Entry(root, width=30)

pwd.grid(row=4, column=1)

Button(root, width=6, bg='blue', fg='white', text='Enter', command=check).grid(row=5, column=0)

Button(root, width=6, bg='red', fg='white', text='Close', command=root.destroy).grid(row=5, column=1)

root.mainloop()

def check():

global un, pwd, login, l, root

u = un.get()

p = pwd.get()

l.execute("select \* from log")

for i in l:

if i[0] == u and i[1] == p and u == 'admin':

root.destroy()

open\_win()

elif i[0] == u and i[1] == p:

root.destroy()

open\_cus()

login.commit()

def main\_menu():

global sto, apt, flag, root, st, val, exp, st1, rev

if flag == 'sto':

sto.destroy()

if flag == 'rev':

rev.destroy()

elif flag == 'st':

st.destroy()

elif flag == 'st1':

st1.destroy()

elif flag == 'val':

val.destroy()

elif flag == 'exp':

exp.destroy()

elif flag == 'd':

d.destroy()

open\_win()

def main\_cus():

global st, flag, exp

if flag == 'exp':

exp.destroy()

elif flag == 'st':

st.destroy()

open\_cus()

def open\_cus():

global apt, flag, flags

flags = 'apt1'

apt = Tk()

apt.title("Interface")

Label(apt, text="\*\*\* EVANZ MEDICAL DRUG STORE \*\*\*", bg='blue', fg='white').grid(row=0, column=0)

Label(apt, text='\*' \* 40).grid(row=1, column=0)

Label(apt, text='\* WELCOME CUSTOMER SERVICES \*', bg='green', fg='white').grid(row=2, column=0)

Label(apt, text='-' \* 40).grid(row=3, column=0)

Label(apt, text='-' \* 40).grid(row=5, column=0)

Button(apt, text='Search', bg='blue', fg='white', width=15, command=search).grid(row=6, column=0)

Button(apt, text='Expiry Check', bg='red', fg='white', width=15, command=exp\_date).grid(row=7, column=0)

Label(apt, text='-' \* 40).grid(row=8, column=0)

Button(apt, text='Logout', bg='green', fg='white', command=again1).grid(row=9, column=0)

apt.mainloop()

def again1():

global flags

apt.destroy()

flags = ''

again()

again()