```
########################   TAYYABA FATIIMA     ################################
#                     CISP 71                              #
########################  CRUD APPLICATION USING sqllite3 ####################


#importing the modules
from tkinter import *                    #import the core components required for building interface
from tkinter import messagebox           # components for showing alerts(errors,warnings
information,question messages)
import tkinter.ttk as ttk
from tkcalendar import Calendar, DateEntry    # components for datentry through calendar
from PIL import ImageTk,Image             # component for importing  images from PIL modules
import sqlite3                           # importing sqlite3


from fabricdatabase import Database          #importing database from database file


#declaring the path for future use
path = 'C:/Users/Tayyaba Fatima/Desktop/fabricproject/'


#creating a root window
root = tk = Tk()
# craeting a title for window
root.title('THE FABRIC WORLD')
#setting the background color for my window
root.config(bg = 'ivory3')
#setting the size of the window
```

```
root.geometry('600x500')
# setting the window icon
root.iconbitmap(path + "color.ico")



#creating a label for our application
lbltitle = Label(root, text = 'FABRIC WHOLESALE RECORDS', font = ('Times New Roman',16,))



my_img = ImageTk.PhotoImage(Image.open(path + "fabric2.png"))
my_label = Label(image = my_img)
my_label.place(x = 0 ,y = 0)

lbltitle = Label(root, text = 'FABRIC WHOLESALE RECORDS', font = ('Times New Roman',16,))



#craetion of SQL table


 # Create a database or connection
conn = sqlite3.connect(path + "Fabrics.db")
try:                            #using exception handling methods to handle error,as
   c = conn.cursor()                #if  the wrong data like instead of integer someone input the str then error willl be handeled
   c.execute(
   ''' CREATE TABLE IF NOT EXISTS Fabric
      (LOTId integer PRIMARY KEY,
       Fabrictype text,
       fabriclength float,
       orderarrive text,
```

```
        dispatched text ,
        Totalpayment integer,
        shipped text)''')
    print('table succesfully created')

except:
    print('table already exist'),
    conn.rollback()
conn.close()


#{{{{{{{{{{{{{{{{{{{{{{{{{{{[   FUNCTIONS   ]}}}}}}}}}}}}}}}}}}}}}}}}}


#defining a method to exit the window with warning icon on the messagebox

def exit_rec():
    msgbox = messagebox.askquestion(
        "Exit Apllication",
        "Are you sure you want to exit the records?",
        icon="error",
    )
    if msgbox == "yes":
        root.destroy()


#defining a method to clear the entries

def clear_rec():
```

```
    FabricIdEN.delete(0,END)              #clear the lotId field
    FabriclentghEN.delete(0,END)          #clear fabriclength field
    orderarriveEN.delete(0,END)           #clear the orderarrive date field
    dispatchedEN.delete(0,END)            #clear the dispatched date field
    TotalpaymentEN.delete(0,END)          #clear the Totalpayment field
    clicked.set(FABoptlist[0])            #clear and set the fabrictype field by setting it to default information
    selected.set(shippedlist[0])          #clear and set the the shipping details field by setting it to default
information


    return()


#defining a function to add the records or save the record in the treeview


def add_rec():
     # Create a database or connection
    conn =  sqlite3.connect(path + "Fabrics.db")
    # create cursor, send orders to database
    c = conn.cursor()
    #insert into table Fabric
    c.execute(" insert into  Fabric values(?,?,?,?,?,?,?)",

(int(FabricIdEN.get()),clicked.get(),float(FabriclentghEN.get()),orderarriveEN.get(),dispatchedEN.get(),int
(TotalpaymentEN.get()),selected.get()))
    try:
        conn.commit()
        print('One record added succesfully')
    except:
        print('error in adding')
        conn.rollback()
```

```
    conn.close()
    clear_rec()
    display_rec()

#defining the function to search from database.

def searchdb_rows(keyword):
     # Create a database or connection
    conn = sqlite3.connect(path + "Fabrics.db")
    # create cursor, send orders to database
    c = conn.cursor()
    c.execute(
        "SELECT * FROM Fabric WHERE LOTId LIKE ?",
        ("%" + keyword + "%",),
    )
    rows = c.fetchall()
    return rows


# defining Function to search database with keywords

def searchdb():
    for a in tvfabric.get_children():
        tvfabric.delete(a)
    count = 0
    for row in searchdb_rows(searchdbEn.get()):
        LOTId = row[0],
        Fabrictype = row[1]
```

```
        fabriclength = row[2]
        orderarrive = row[3]
        dispatched = row[4]
        Totalpayment = row[5]
        shipped = row[6]
        tvfabric.insert(
            "",
            "end",
            text="LOTId",
            values=(LOTId,Fabrictype,fabriclength,orderarrive,dispatched,Totalpayment,shipped),
        )
    count += 1



#defining a function to display the record in the treeview.


def display_rec():
    for row in tvfabric.get_children():
        tvfabric.delete(row)
    # Create a database or connection
    conn = sqlite3.connect(path + "Fabrics.db")
    # create cursor, send orders to database
    c = conn.cursor()
    # Insert into table
    c.execute("SELECT *, oid FROM Fabric")
    records = c.fetchall()
    # Print result by looping 1 by 1
    for row in records:
        LOTId = row[0],
```

```
    Fabrictype = row[1]
    fabriclength = row[2]
    orderarrive = row[3]
    dispatched = row[4]
    Totalpayment = row[5]
    shipped = row[6]
    tvfabric.insert(
        "",
        "end",
        text=id,
        values=(
            LOTId,
            Fabrictype,
            fabriclength,
            orderarrive,
            dispatched,
            Totalpayment,
            shipped,
            id,
        ),
    )
# Commite changes
conn.commit()
# Close connection
conn.close()



return()
```

```
#defining the function to show the selected record

def show_selected_rec(event):

    #clearing rec by calling clear_rec  function
    clear_rec()
    for selection in tvfabric.selection():
        item = tvfabric.item(selection)
        global id
        # grab the values from SQL
        LOTId,Fabrictype,fabriclength,orderarrived,dispatched,Totalpayment,shipped = item[
            "values"
        ][0:7]
        # Inserting back to each entry
        FabricIdEN.insert(0, LOTId)
        FabriclentghEN.insert(0, fabriclength)
        orderarriveEN.insert(0, orderarrived)
        dispatchedEN.insert(0, dispatched)
        TotalpaymentEN.insert(0, Totalpayment)
        clicked.set(Fabrictype)
        selected.set(shipped)
    return id


 #defining the function to update the record

def update_rec():
     # Create a database or connection
    conn = sqlite3.connect(path + "Fabrics.db")
```

```
    qry = " update Fabric set  Fabrictype=?, fabriclength=?, orderarrive=?, dispatched=?, Totalpayment=?, shipped=? where LOTId = ?"
    try:
        # create cursor, send orders to database
        c = conn.cursor()
        c.execute(qry, (
                    clicked.get(),
                    float(FabriclentghEN.get()),
                    orderarriveEN.get(),
                    dispatchedEN.get(),
                    int(TotalpaymentEN.get()),
                    selected.get(),
                    int(FabricIdEN.get())
                    ),
                )
        #commit function
        conn.commit()
        print("RECORD updated successfully")
    except:
        print("Record updated unsuccessfully")
        conn.rollback()
    conn.close()
    #clear the record in the entries by calling clear_rec function
    clear_rec()
    #display the record in treeview by calling display_rec function
    display_rec()


#defining the function to delete the record
```

```
def delete_rec():

    # Create a database or connection
    conn = sqlite3.connect(path + "Fabrics.db")
    qry = "DELETE from Fabric where LOTId = ?;"
    try:
        # create cursor, send orders to database
        c = conn.cursor()
        c.execute(qry, (FabricIdEN.get(),))
        conn.commit()
        # print("Stock deleted successfully")
        print("RECORD deleted successfully")

    except:
        # print("Error in delete operation")
        print("Record deleted unsuccessfully")
        conn.rollback()
    #close the connection
    conn.close()
    #clear record in the entries by calling clear_rec function
    clear_rec()
    #display the record in treeview by calling display_rec function
    display_rec()

#define the function for Popup notification for delete

def deletePop():
```

```
   message = messagebox.askquestion("Delete Confirmation", "Are you sure you want to delete record?")
   if message == 'yes':
      delete_rec()



#***********************************[  LABEL,ENTRY,BUTTONS ]**************************

#creating label widgets for each entry that are LOTId,Fabrictype,fabriclength,orderarrive,dispatched ,totalpayment,shipping information

FabricIdlb = Label(root,text = 'LOT-ID',bg = 'snow',font = ('Helvetica' ,11 ,'bold' ))

fabrictypelb =Label(root,text = 'FABRIC-TYPE',bg = 'snow',font = ('Helvetica', 11, 'bold' ))

Fabriclengthlb = Label(root, text = ' FABRIC-LENGTH(CM).',bg = 'snow',font = ('Helvetica', 11, 'bold' ))

orderarrivelb = Label(root, text = ' ORDER-BOOKED.',bg = 'snow',font = ('Helvetica', 11 ,'bold' ))

dispatchedlb = Label(root, text = 'DISPATCH-DATE.',bg = 'snow',font = ('Helvetica', 11 ,'bold' ))

Totalpaymentlb = Label(root, text = 'TOTAL-COST($)',bg = 'snow',font = ('Helvetica', 11 ,'bold' ))

shippedboxlb = Label(root,text = 'SHIPPING INFO.',bg = 'snow',font = ('Helvetica', 11 ,'bold' ))

searchdbLb = Label(root, text="ENTER LOT-ID",bg = 'snow',font = ('Helvetica',13,'bold' ))



#list for Fabrictype option

FABoptlist = ["COTTON","GEORGETTE","SILK","CHIFFON","VELVET","DENIM","SATIN","POLYSTER","NYLON","LEATHER","CHENILLE"]

clicked = StringVar()

clicked.set(FABoptlist[0])

#list for shipping option
```

```
shippedlist = ['YES','NO','WAITING','PACKED','LOADED','ON THE WAY']

selected= StringVar()

selected.set(shippedlist[0])



#creating entry widgets for each entry that are LOTId,Fabrictype,fabriclength,orderarrive,dispatched
,totalpayment,shipping information



FabricIdEN = Entry(root,font=('ariel',11),width = 15,bg = 'mint cream')

#creating an optionmenu for fabrictype

fabrictypEN =OptionMenu(root,clicked, * FABoptlist)

FabriclentghEN = Entry(root,font=('ariel',11),width = 15,bg = 'mint cream')

#creating a datenetry

sel = StringVar()

orderarriveEN = DateEntry(root,selectmode = 'day', textvariable = sel ,width = 20)

#creating a dateEntry

cli = StringVar()

dispatchedEN = DateEntry(root, selectmode = 'day', textvariable = cli ,width = 20)

TotalpaymentEN = Entry(root, text = 'TOTAL-COST($)',bg = 'mint cream')

#creating an optionmenu for shipping details

shippedbox = OptionMenu(root,  selected,*shippedlist)

#creating as search entry box

searchdbEn = Entry(root,font=('ariel',11),width = 25,bg = 'mint cream')



#creating button widgets

addBT = Button(root, text = 'ADD-REC',width = 10, bg = 'light grey', fg = 'black',command = add_rec,font
= ('Comic Sans MS' ,11, 'bold'))

updateBT = Button(root , text = 'UPDATE ',width = 10, bg = 'light grey', fg =
'black',command=update_rec,font = ('Comic Sans MS' ,11, 'bold'))
```

```
displayBT = Button(root, text = 'DISPLAY ',width = 10, bg = 'light grey', fg =
'black',command=display_rec,font = ('Comic Sans MS' ,11, 'bold'))

deleteBT = Button(root,text = "DELETE",width = 10, bg = 'red', fg = 'black',command = deletePop,font =
('Comic Sans MS' ,11, 'bold' ))

clearBT = Button(root,text = 'CLEAR', width = 10,bg = 'light grey', fg = 'black',command = clear_rec,font =
('Comic Sans MS' ,11, 'bold' ))

exitBT = Button(root,text = 'EXIT',width = 10, bg = 'cyan', fg = 'black',command = exit_rec,font = ('Comic
Sans MS' ,11, 'bold' ))

searchdbBt = Button(root, text="SEARCH",width = 10, bg = 'light green',font = ('Comic Sans MS' ,11,
'bold' ),command=searchdb)


#placing the label widgets for each entry that are LOTId,Fabrictype,fabriclength,orderarrive,dispatched
,totalpayment,shipping information



lbltitle.place(x = 500,y =5)

FabricIdlb.place(x = 460 , y = 80)

fabrictypelb.place(x = 420 , y = 130)

Fabriclengthlb.place(x = 420, y = 175)

orderarrivelb.place(x = 420, y = 215)

dispatchedlb.place(x = 420 , y = 255)

Totalpaymentlb.place(x = 420 , y = 300)

shippedboxlb.place(x = 420, y = 340)

searchdbLb.place(x = 20 , y = 510)


# placing the entry widgets for each entry that are LOTId,Fabrictype,fabriclength,orderarrive,dispatched
,totalpayment,shipping information



FabricIdEN.place(x = 630, y = 80)

fabrictypEN.place(x = 630, y = 120)
```

```
FabriclentghEN.place(x = 630, y = 170)

orderarriveEN.place(x = 630, y = 215)

dispatchedEN.place(x = 630, y = 255)

TotalpaymentEN.place(x = 630, y =300)

shippedbox.place( x = 630, y = 340)

searchdbEn.place(x = 20, y = 550)



#placing the buttons on the root window


addBT.place(x = 860, y = 70)

updateBT.place(x = 860, y = 120)

displayBT.place(x = 860, y = 170)

clearBT.place ( x = 860, y = 220)

deleteBT.place ( x= 860, y = 270)

exitBT.place(x =860,y = 320)

searchdbBt.place(x = 20, y = 573)


##########################################[  TREEVIEW
]####################################


#creating a style for treeview to set the background ,foreground color


style = ttk.Style()

style.theme_use("clam")

style.configure('Treeview',background = 'azure2',fg='black',rowheight=20,fieldbackground = 'azure2')

style.map('Treeview',background=[('selected','maroon')])


#creating number of columns in the treeview
```

```
columns = ("#1","#2","#3","#4","#5","#6","#7")
tvfabric = ttk.Treeview(root,show = 'headings',height='12',columns=columns
                    )

#creating a treeview for each entry that are LOTId,Fabrictype,fabriclength,orderarrive,dispatched
,totalpayment,shipping information

tvfabric.heading('#1',text = 'LOT-ID',anchor = 'center')
tvfabric.column('#1',width = 60,anchor = 'center',stretch = True)

tvfabric.heading('#2',text = 'FAB-TYPE',anchor = 'center')
tvfabric.column('#2',width = 100 ,anchor = 'center',stretch = False)


tvfabric.heading('#3',text = 'FAB-LENGTH(CM)',anchor = 'center')
tvfabric.column('#3',width = 120 ,anchor = 'center',stretch = True)

tvfabric.heading('#4',text = 'ORDER-BOOKED',anchor = 'center')
tvfabric.column('#4',width = 120 ,anchor = 'center',stretch = True)


tvfabric.heading('#5',text = 'DISPATCHED',anchor = 'center')
tvfabric.column('#5',width = 100 ,anchor = 'center',stretch = True)


tvfabric.heading('#6',text = 'TOTALPAYMENT($)',anchor = 'center')
tvfabric.column('#6',width = 120 ,anchor = 'center',stretch = True)
```

```
tvfabric.heading('#7',text = 'SHIPPING-INFO',anchor = 'center')
tvfabric.column('#7',width = 120,anchor = 'center',stretch = True)



tvfabric.bind("<<TreeviewSelect>>",show_selected_rec)

tvfabric.place(x = 300, y =380 )

#creating Scrollbar vertically

vsb = ttk.Scrollbar(root, orient="vertical", command=tvfabric.yview)

#placing the scrollbar vertically
vsb.place(x=1030, y=383, height=263)
tvfabric.configure(yscrollcommand=vsb.set)

#displaying record when open an application
display_rec()

#mainloop
root.mainloop()
```