#ENGINEERING EXPLORATION PROJECT BATCH 5 TKINTER APPLICATION\_ RAPPEL(LIBRARY BOOK REMINDER)\_\_\_\_\_Backend

#Import the required modules

import sqlite3

import cv2

from pyzbar.pyzbar import decode

import time

import smtplib

#Declaration of global variables

id\_3 = 1

bc=0

bn=''

an=''

sn=''

rn=0

em=''

adn=0

br=''

pn=0

s\_y=2021

s\_m=7

s\_d=3

b=''

a=0

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Scan Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def scan():

    cap= cv2.VideoCapture(0)

    cap.set(3, 600)        #To set the size of the video capture window

    cap.set(4, 600)

    camera = True

    while camera == True:

        success, frame=cap.read()

        key=cv2.waitKey(1)

        flag=0

        if key%256 == 27:

           break

        for code in decode(frame):

           global a

           a = code.data.decode('utf-8')         #To decode the barcode and store the information in the variable

           flag=1

           time.sleep(3)

        if flag == 1:

           camera=False

           break

        cv2.imshow('Testing-code-scan',frame)

        cv2.waitKey(10)

    cap.release()

    cv2.destroyAllWindows

    return a

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*To display the information on the info box\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*s

def call\_func(ar):

        global b,s\_d,s\_m,s\_y

        if ar=='update':

            b="UPDATED  BOOK  DETAILS  SUCCESSFULLY!"

        if ar=='add':

            b="ADDED BOOK  DETAILS  SUCCESSFULLY!"

        if ar=='delete':

            b="DELETED BOOK  DETAILS  SUCCESSFULLY!"

        if ar=='saved':

            b="DETAILS  RECORDED  SUCCESSFULLY!"

        if ar=='updated':

            b="UPDATED  STUDENT  DETAILS  SUCCESSFULLY!"

        if ar=='added':

            b="ADDED STUDENT  DETAILS  SUCCESSFULLY!"

        if ar=='deleted':

            b="DELETED  STUDENT  DETAILS  SUCCESSFULLY!"

        if ar=='sent':

            b="EMAIL  SENT  SUCCESSFULLY!"

        if ar=='date':

            b="TODAY's DATE  :  " + str(s\_d)+"/"+str(s\_m) +"/"+str(s\_y)

        if ar=='delete\_e':

            b="DETAILS  DELETED  SUCCESSFULLY!"

def Info\_func():

    global b

    return b

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*This is for the issued details database\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Fuction to establish a connection between code and the database and create a table if not exists

def connect\_3():

    conn = sqlite3.connect("send\_emails.db")

    cur = conn.cursor()

    cur.execute("CREATE TABLE IF NOT EXISTS send\_email (id\_3 INTEGER PRIMARY KEY, studentname text , rollnumber integer , email text ,admnum integer , branch text , phonenum integer, bookcode integer, bookname text, author text , s\_date integer , s\_month integer , s\_year integer)")

    conn.commit()

    conn.close()

#Function to view the details present in the database

def view\_3():

    conn = sqlite3.connect("send\_emails.db")

    cur = conn.cursor()

    cur.execute("SELECT \* FROM send\_email")

    rows=cur.fetchall()

    conn.close()

    return rows

#Funtion to insert new details in the database

def insert\_3(studentname,rollnumber,email,admnum,branch,phonenum,bookcode, bookname, author,s\_d,s\_m,s\_y):

    conn = sqlite3.connect("send\_emails.db")

    cur = conn.cursor()

    cur.execute("INSERT INTO send\_email VALUES (NULL,?,?,?,?,?,?,?,?,?,?,?,?)",(studentname, rollnumber ,email,admnum,branch,phonenum,bookcode, bookname, author,s\_d,s\_m,s\_y))

    #view\_3()

    #call\_func('add\_e')

    conn.commit()

    conn.close()

#Function to delete the details in the datbase

def delete\_3(id\_a):

    conn = sqlite3.connect("send\_emails.db")

    cur = conn.cursor()

    cur.execute("DELETE FROM send\_email WHERE id\_3=?",(id\_a,))

    conn.commit()

    conn.close()

    view\_3()

    print("Details deleted !!")

    call\_func('delete\_e')

#This is to search for a particular member in the data baseS

def search\_3(s\_d='',s\_m='',s\_y=''):

    conn = sqlite3.connect("send\_emails.db")

    cur = conn.cursor()

    cur.execute("SELECT \* FROM send\_email WHERE s\_date=? AND s\_month=? AND s\_year=? ",(s\_d,s\_m,s\_y))

    rows=cur.fetchall()

    conn.close()

    return rows

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*This is for the books database\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Fuction to establish a connection between code and the database and create a table if not exists

def connect():

    conn = sqlite3.connect("books.db")

    cur = conn.cursor()

    cur.execute("CREATE TABLE IF NOT EXISTS book (id INTEGER PRIMARY KEY, bookcode integer, bookname text, author text)")

    conn.commit()

    conn.close()

#Function to view the details present in the database

def view():

    conn = sqlite3.connect("books.db")

    cur = conn.cursor()

    cur.execute("SELECT \* FROM book")

    rows=cur.fetchall()

    conn.close()

    return rows

#Funtion to insert new details in the database

def insert(bookcode, bookname, author):

    conn = sqlite3.connect("books.db")

    cur = conn.cursor()

    cur.execute("INSERT INTO book VALUES (NULL,?,?,?)",(bookcode, bookname, author))

    view()

    call\_func('add')

    conn.commit()

    conn.close()

#This is to search for a particular member in the data baseS

def search(bookcode="",bookname="",author=""):

    conn = sqlite3.connect("books.db")

    cur = conn.cursor()

    cur.execute("SELECT \* FROM book WHERE bookcode=? OR bookname=? OR author=?",(bookcode,bookname,author))

    rows=cur.fetchall()

    conn.close()

    return rows

#Function to delete the details in the database

def delete(id):

    conn = sqlite3.connect("books.db")

    cur = conn.cursor()

    cur.execute("DELETE FROM book WHERE id=?",(id,))

    view()

    call\_func('delete')

    conn.commit()

    conn.close()

#Function to update the details in the database

def update(id,bookcode,bookname,author):

    conn = sqlite3.connect("books.db")

    cur = conn.cursor()

    cur.execute("UPDATE book SET bookcode=?, bookname=?, author=? WHERE id=?",(bookcode,bookname,author,id))

    conn.commit()

    call\_func('update')

    conn.close()

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*This is for the students database\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Fuction to establish a connection between code and the database and create a table if not exists

def connect\_1():

    conn = sqlite3.connect("students.db")

    cur = conn.cursor()

    cur.execute("CREATE TABLE IF NOT EXISTS student (id\_1 INTEGER PRIMARY KEY, studentname text,rollno integer, email text, addmissionno integer, branch text, phonenumber integer)")

    conn.commit()

    conn.close()

#This is to search for a particular member in the data baseS

def find(studentname, rollno, email, addmissionno, branch, phonenumber):

    conn = sqlite3.connect("students.db")

    cur = conn.cursor()

    cur.execute("SELECT \* FROM student WHERE studentname=? OR rollno=? OR email=? OR addmissionno=? OR branch=? OR phonenumber=?",( studentname, rollno, email, addmissionno, branch, phonenumber))

    rows=cur.fetchall()

    conn.close()

    print("rows")

    print(rows)

    return rows

#Funtion to insert new details in the database

def add(studentname, rollno, email, addmissionno, branch, phonenumber):

    conn = sqlite3.connect("students.db")

    cur = conn.cursor()

    cur.execute("INSERT INTO student VALUES (NULL,?,?,?,?,?,?)",(studentname, rollno, email, addmissionno, branch, phonenumber))

    view()

    call\_func('added')

    conn.commit()

    conn.close()

#Function to view the details present in the database

def view\_1():

    conn = sqlite3.connect("students.db")

    cur = conn.cursor()

    cur.execute("SELECT \* FROM student")

    rows=cur.fetchall()

    conn.close()

    return rows

#Function to delete the details in the database

def erase(id\_1):

    conn = sqlite3.connect("students.db")

    cur = conn.cursor()

    cur.execute("DELETE FROM student WHERE id\_1=?",(id\_1,))

    view\_1()

    call\_func('deleted')

    conn.commit()

    conn.close()

#Function to update the details in the database

def update\_s(id\_1,studentname, rollno, email, addmissionno, branch, phonenumber):

    conn = sqlite3.connect("students.db")

    cur = conn.cursor()

    cur.execute("UPDATE student SET studentname=?, rollno=?, email=?, addmissionno=?,branch=?,phonenumber=? WHERE id\_1=?",(studentname, rollno, email, addmissionno, branch, phonenumber,id\_1))

    conn.commit()

    call\_func('updated')

    conn.close()

#Function to save the details in the database

def save\_date(date,month,year):

    global s\_d,s\_m,s\_y

    s\_d=date

    s\_m=month

    s\_y=year

    call\_func('date')

    print("DATE:"+s\_d +"-"+s\_m+"-"+ s\_y)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*To save all the student and book details and date of issue of the book\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def save\_details(id\_s,student\_name,roll\_no,e\_mail,addmission\_no,branch,phone\_number,book\_code,book\_name,author\_name,d,m,y):

     global bc,bn,an,sn,rn,em,adn,br,pn,s\_d,s\_m,s\_y,id\_3

     tuple3=(id\_s,student\_name,roll\_no,e\_mail,addmission\_no,branch,phone\_number,book\_code,book\_name,author\_name,d,m,y)

     id\_3 = tuple3[0]

     sn=tuple3[1]

     rn=tuple3[2]

     em=tuple3[3]

     adn=tuple3[4]

     br=tuple3[5]

     pn=tuple3[6]

     bc=tuple3[7]

     bn=tuple3[8]

     an=tuple3[9]

     s\_d=tuple3[10]

     s\_m=tuple3[11]

     s\_y=tuple3[12]

     insert\_3(sn,rn,em,adn,br,pn,bc,bn,an,s\_d,s\_m,s\_y)

     call\_func('saved')

     view\_3()

def save\_img(roll\_no):

     roll = roll\_no

     return roll

std\_email=''

book\_code=1

book\_name=''

author\_name=''

std\_name=''

roll\_no=0

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*To send emails to the selected students\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def send\_mail(t\_d,t\_m,t\_y):

    global s\_d,s\_m,s\_y,std\_email,book\_code,book\_name,author\_name,std\_name,roll\_no

    t\_d=int(t\_d)

    t\_m=int(t\_m)

    t\_y=int(t\_y)

    #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Logic fo get the date 14 days back the present date\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    if t\_d<15:

        check\_month=t\_m-1

        if t\_m==1:

            check\_month=12

            check\_date=31+t\_d-14

            check\_year=t\_y-1

        else:

            #check\_month=t\_m

            check\_month=t\_m-1

            check\_year=t\_y

            leap=check\_year%4

            print("leap "+str(leap))

            if check\_month in (1,3,5,7,8,10,12):

                check\_date=31+t\_d-14

            elif check\_month == 2:

                if check\_year%4==0:

                    check\_date=29+t\_d-14

                else:

                    check\_date=28+t\_d-14

            else:

                check\_date=30+t\_d-14

        print(str(check\_date)+"-"+str(check\_month)+"-"+str(check\_year))

    else:

        check\_date=t\_d-14

        check\_month=t\_m

        check\_year=t\_y

        print(str(check\_date)+"-"+str(check\_month)+"-"+str(check\_year))

    tuple\_list=search\_3(check\_date,check\_month,check\_year)

    email\_tuple\_elements = []

    for a\_tuple in tuple\_list:           #To store required emails

        email\_tuple\_elements.append(a\_tuple[3])

    tuple\_length=len(email\_tuple\_elements)

    stdname\_tuple\_elements=[]

    for r\_tuple in tuple\_list:           #To store required student\_name

        stdname\_tuple\_elements.append(r\_tuple[1])

    tuple\_length=len(stdname\_tuple\_elements)

    bookcode\_tuple\_elements=[]

    for b\_tuple in tuple\_list:           #To store required book\_codes

        bookcode\_tuple\_elements.append(b\_tuple[7])

    tuple\_length=len(bookcode\_tuple\_elements)

    bookname\_tuple\_elements=[]

    for c\_tuple in tuple\_list:           #To store required book\_name

        bookname\_tuple\_elements.append(c\_tuple[8])

    tuple\_length=len(bookname\_tuple\_elements)

    id\_tuple\_elements=[]

    for d\_tuple in tuple\_list:           #To store required id

        id\_tuple\_elements.append(d\_tuple[0])

    tuple\_length=len(id\_tuple\_elements)

    authorname\_tuple\_elements=[]

    for e\_tuple in tuple\_list:           #To store required author\_name

        authorname\_tuple\_elements.append(e\_tuple[9])

    tuple\_length=len(authorname\_tuple\_elements)

    rollno\_tuple\_elements=[]

    for s\_tuple in tuple\_list:           #To store required roll\_no

        rollno\_tuple\_elements.append(s\_tuple[2])

    tuple\_length=len(rollno\_tuple\_elements)

    #To send the email to multiple required students

    for i in range(0,tuple\_length):

        std\_email=email\_tuple\_elements[i]

        book\_code=bookcode\_tuple\_elements[i]

        book\_name=bookname\_tuple\_elements[i]

        id\_tobe\_deleted=id\_tuple\_elements[i]

        author\_name=authorname\_tuple\_elements[i]

        std\_name=stdname\_tuple\_elements[i]

        roll\_no=rollno\_tuple\_elements[i]

        def send\_email():

            global std\_email,book\_code,book\_name,author\_name,std\_name,roll\_no

            str\_book\_code=str(book\_code)

            server = smtplib.SMTP\_SSL('smtp.gmail.com',465)            #To establish a connection to the server

            server.login("rohithgundaram@gmail.com","sonugundaram@gmail.com")

            message= "FROM LIBRARY DEPARTMENT OF CBIT.\n\nHi, "+std\_name +"\n\nBOOK DETAILS: \n\nBook name    : "+ str(book\_name)  +"\nBook code    : "+ str\_book\_code +"\nAuthor Name: "+author\_name +"\n\nPLEASE RETURN THE BOOK TO THE LIBRARY BY TOMORROW.\nELSE YOU WILL HAVE TO PAY THE FINE!!\nTHANK YOU."

            server.sendmail("rohithgundaram@gmail.com", std\_email ,message)

            server.quit()

        send\_email()

        delete\_3(str(id\_tobe\_deleted))

    call\_func('sent')

connect\_3()

connect\_1()

connect()