import tkinter as tk

import mysql.connector

import json

import imaplib

import email

import traceback

from mysql.connector import Error

#pip install mysql

#pip3 install mysql-connector

class GuiOverlay:

def \_\_init\_\_(self, parent):

#formatting parent frame

parent.grid\_rowconfigure(1, weight = 1)

parent.grid\_columnconfigure(0, weight = 1)

#create main frames in parent frame

self.top\_frame = tk.Frame(parent, bg = 'cadetblue3', width = 450, height = 50, pady = 1)

self.center = tk.Frame(parent, bg = 'gray2', width = 50, height = 40, pady = 1)

btm\_frame = tk.Frame(parent, bg = 'white', width = 450, height = 45, pady = 1)

#display main frames in parent frame

self.top\_frame.grid(row = 0, sticky = 'ew')

self.center.grid(row = 1, sticky = 'nsew')

btm\_frame.grid(row = 2, sticky = 'ew')

#formatting top frame

self.top\_frame.grid\_rowconfigure(0, weight = 1)

self.top\_frame.grid\_columnconfigure(1, weight = 1)

#create widgets for top frame

bigTitle = tk.Label(self.top\_frame, text = '3S Co., Ltd.', fg = 'white', bg = 'cadetblue3')

bigTitle.config(font=('Tw Cen MT', 13, 'bold'))

self.pageTitle = tk.Label(self.top\_frame)

#display widgets in top frame

bigTitle.grid(row = 0, column = 0, columnspan = 3)

#formatting center frame

self.center.grid\_rowconfigure(0, weight=1)

self.center.grid\_columnconfigure(0, weight=1)

#create center widgets

self.cc = tk.Frame(self.center, bg = 'gray95', width = 100, height = 100)

#display center widgets

self.cc.grid(row = 0, column = 1, sticky = 'nsew')

#formatting bottom frame

btm\_frame.grid\_rowconfigure(0, weight = 1)

btm\_frame.grid\_columnconfigure(1, weight = 1)

#create widgets for bottom frame

exit = tk.Button(btm\_frame, text = 'Exit', width = 5, command = parent.destroy)

#display widgets in the bottom frame

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

#build all frames

self.build\_checkOrders()

#show home page

self.show\_page(self.pLOOK)

def add\_sep(self, r, c, widg):

'''

create empty label for formatting in grid

'''

tk.Label(widg).grid(row=r, column=c)

def update\_scrollregion(self, event):

'''

update scroll region

'''

self.cc.configure(scrollregion=self.cc.bbox("all"))

def FrameWidth(self, event):

'''

expand frame in canvas

'''

canvas\_width = event.width

self.cc.itemconfig(self.canvasFRM, width = canvas\_width)

def build\_checkOrders(self):

'''

build the frame to check the orders

'''

def read\_gmail():

'''

read the gmail for order details

'''

user = "3s.company3ss@gmail.com"

password = "3s11002299."

try:

mail = imaplib.IMAP4\_SSL("imap.gmail.com" , 993)

mail.login(user, password)

mail.select('inbox')

data = mail.search(None, 'ALL')

mail\_ids = data[1]

id\_list = mail\_ids[0].split()

first\_email\_id = int(id\_list[0])

latest\_email\_id = int(id\_list[-1])

#set var

orderDetails\_json = {}

orderDetails\_dict = {}

count = 0

#read gmail

for i in range(latest\_email\_id,first\_email\_id-1, -1):

data = mail.fetch(str(i), '(RFC822)' )

for response\_part in data:

arr = response\_part[0]

if isinstance(arr, tuple):

msg = email.message\_from\_string(str(arr[1],'utf-8'))

if msg['subject'] == 'ABC Interior Design & Decoration Agency | Order ':

#read body message

for part in msg.walk():

orderDetails\_json[count] = part.get\_payload()

count += 1

#set var

count = 0

#decoding json

for e in orderDetails\_json:

orderDetails\_dict[count] = json.loads(orderDetails\_json[e])

count += 1

#clear var

orderDetails\_json = {}

connection = mysql.connector.connect(host='localhost',

database='3s\_co\_ltd',

user='root',

password='')

try:

mycursor = connection.cursor()

#get the last no.

mycursor.execute("Select no from building\_material order by no desc limit 1")

latestNum = mycursor.fetchone()[0]

#insert new order details into the table

for e in orderDetails\_dict:

for num in orderDetails\_dict[e]:

if int(num) > int(latestNum):

mycursor.execute("Insert into building\_material(no,case\_id,created\_dt,item,order\_id,price,quantity,status,total\_price) values (%s,%s,%s,%s,%s,%s,%s,%s,%s)", (num, orderDetails\_dict[e][num]['case\_id'], orderDetails\_dict[e][num]['created\_dt'], orderDetails\_dict[e][num]['item'], orderDetails\_dict[e][num]['order\_id'], orderDetails\_dict[e][num]['price'], orderDetails\_dict[e][num]['quantity'], orderDetails\_dict[e][num]['status'], orderDetails\_dict[e][num]['total\_price']))

connection.commit()

#return error message when sql failed to run

except Error as e:

print("Error reading data from MySQL table", e)

connection.close()

except Exception as e:

traceback.print\_exc()

print(str(e))

read\_gmail()

#create canvas

self.cc.destroy()

self.cc = tk.Canvas(self.center, bg="white", bd=0, highlightthickness=0)

self.cc.grid(sticky="news")

#create check orders frame

self.pLOOK = tk.Frame(self.cc, bg = 'gray95', width = 100, height = 100)

self.pLOOK.grid(row = 0, column = 0, sticky = 'nsew')

self.canvasFRM = self.cc.create\_window(0, 0, window=self.pLOOK, anchor='nw')

self.cc.bind('<Configure>', self.FrameWidth)

#create scroll bars

self.scrollbarV=tk.Scrollbar(self.center, orient="vertical", command=self.cc.yview)

self.scrollbarV.grid(row=0, column=1, sticky = 'nse')

self.scrollbarH=tk.Scrollbar(self.center, orient="horizontal", command=self.cc.xview)

self.scrollbarH.grid(row=1, column=0, sticky = 'nswe')

#config scroll bars

self.cc.config(yscrollcommand=self.scrollbarV.set)

self.cc.config(xscrollcommand=self.scrollbarH.set)

#fillin the bottom right gp302 area

uselessArea = tk.Frame(self.center, bg = 'whitesmoke')

uselessArea.grid(row=1, column=1, sticky = 'nswe')

#formatting the frame

for e in range(10):

self.pLOOK.grid\_columnconfigure(e, weight = 1)

self.cc.grid\_columnconfigure(e, weight = 1)

#create data titles

num = tk.Label(self.pLOOK, text='No.')

caseid = tk.Label(self.pLOOK, text='Case ID')

orderid = tk.Label(self.pLOOK, text='Order ID')

item = tk.Label(self.pLOOK, text='Item')

quantity = tk.Label(self.pLOOK, text='Quantity')

price = tk.Label(self.pLOOK, text='Price')

totalPrice = tk.Label(self.pLOOK, text='Total price')

status = tk.Label(self.pLOOK, text='Status')

since = tk.Label(self.pLOOK, text='Since')

#display data titles

row = 0

num.grid(row = row, column = 0)

caseid.grid(row = row, column = 1)

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

item.grid(row = row, column = 3)

quantity.grid(row = row, column = 4)

price.grid(row = row, column = 5)

totalPrice.grid(row = row, column = 6)

status.grid(row = row, column = 7)

since.grid(row = row, column = 8)

connection = mysql.connector.connect(host='localhost',

database='3s\_co\_ltd',

user='root',

password='')

try:

#set var

results = []

mycursor = connection.cursor()

mycursor.execute("SELECT \* FROM building\_material ORDER BY order\_id desc, status <> 'pending', status <> 'preparing', status <> 'delivering'")

results = mycursor.fetchall()

counter = 1

for row in results:

#no.

numL = tk.Label(self.pLOOK, text = counter)

numL.grid(row = counter, column = 0)

#case id

caseidL = tk.Label(self.pLOOK, text = row[1])

caseidL.grid(row = counter, column = 1)

#order id

orderidL = tk.Label(self.pLOOK, text = row[2])

orderidL.grid(row = counter, column = 2)

#item

itemL = tk.Label(self.pLOOK, text = row[3])

itemL.grid(row = counter, column = 3)

#quantity

quantityL = tk.Label(self.pLOOK, text = row[4])

quantityL.grid(row = counter, column = 4)

#price

priceL = tk.Label(self.pLOOK, text = row[5])

priceL.grid(row = counter, column = 5)

#total price

totalPriceL = tk.Label(self.pLOOK, text = row[6])

totalPriceL.grid(row = counter, column = 6)

#status

statusL = tk.Label(self.pLOOK, text = row[7])

statusL.grid(row = counter, column = 7)

#since

sinceL = tk.Label(self.pLOOK, text = row[8])

sinceL.grid(row = counter, column = 8)

#formatting on sale frame

counter += 1

#return error message when sql failed to run

except Error as e:

print("Error reading data from MySQL table", e)

connection.close()

'''

update scroll region

'''

self.pLOOK.bind("<Configure>", self.update\_scrollregion)

def show\_page(self, page):

'''

show called frame

'''

self.pageTitle.destroy()

if page == self.pLOOK:

self.pageTitle = tk.Label(self.top\_frame, text = 'Check orders', bg = 'cadetblue3')

self.pageTitle.config(font=('Courier', 10, 'bold'))

self.pageTitle.grid(row = 1, column = 0, padx = 100)

page.lift()

def main():

root = tk.Tk()

root.geometry('1000x565')

root.title('3S Co., Ltd.')

GuiOverlay(root)

root.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

main()