# To use or import Tkinter

from tkinter import \*

from tkinter import ttk

# To display the date and time

import time

# To connect to Database

from sqlite3 import \*

# To show the messages

from tkinter import messagebox

# For Mail Functionality

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

import smtplib

# For Validations

import re

from hashlib import sha256

# For Reports

from fpdf import FPDF

import pandas as pd

#for the Font

from tkinter.font import Font

#---------------------------------------------------All Login Pages Code Start--------------------------------------------------------------------#

#---------------------------------------------------Main/User Login Page--------------------------------------------------------------------------#

class FoodManagementSystem:

def main(fms):

try:

fms.scr.destroy()

fms.scr=Tk()

except:

try:

fms.scr=Tk()

except:

pass

fms.scr.title("FOOD MANAGEMENT SYSTEM")

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

# Logo picture

fms.loginf1=Frame(fms.scr, bg="#ffffff")

fms.loginf1.pack(fill=BOTH,expand=YES)

fms.logo=PhotoImage(file="header.png")

fms.logo\_img = fms.logo.subsample(2,2)

fms.logo\_banner = Label(fms.loginf1, image=fms.logo\_img)

fms.logo\_banner.place(x=0,y=0)

fms.logo\_banner.pack(side=LEFT)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.loginf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.tim.pack(pady=45, anchor=N)

# Buttons

buttons\_frame = Frame(fms.loginf1, bg="#ffffff")

buttons\_frame.pack(pady=0)

fms.home=Button(buttons\_frame,text="Home", command=fms.main, bg="#0b1335",cursor="hand2",fg="white",font=("cooper black",14))

fms.home.grid(row=0, column=0, padx=10)

fms.adlog=Button(buttons\_frame,text="Admin Login",command=fms.Adminlogin,cursor="hand2", bg="#0b1335",fg="white",font=("cooper black",14))

fms.adlog.grid(row=0, column=1, padx=10)

fms.abt=Button(buttons\_frame,text="Manager Login", command=fms.Managerlogin, bg="#0b1335",cursor="hand2", fg="white",font=("cooper black",14))

fms.abt.grid(row=0, column=2, padx=10)

fms.loginf2=Frame(fms.scr,height=1080,width=1920)

fms.loginf2.pack(fill=BOTH,expand= YES)

fms.c=Canvas(fms.loginf2,height=1080,width=1920)

fms.c.pack()

fms.fmain=PhotoImage(file="main.png")

fms.c.create\_image(650,309,image=fms.fmain)

fms.c.create\_rectangle(350,100,1020,475,fill="#d3ede6",outline="white",width=6)

fms.log=Label(fms.loginf2,text="USER LOGIN",fg="white",bg="#0b1335",width=27,font=("cooper black",27))

fms.log.place(x=357,y=110)

fms.lab1=Label(fms.loginf2,text="User Name",bg="#d3ede6",font=("cooper black",22))

fms.lab1.place(x=360,y=200)

fms.user=Entry(fms.loginf2,bg="white",font=("cooper black",22),bd=5)

fms.user.place(x=650,y=200)

fms.lab2=Label(fms.loginf2,text="Password",bg="#d3ede6",font=("cooper black",22))

fms.lab2.place(x=360,y=270)

fms.pasd=Entry(fms.loginf2,bg="white",font=("cooper black",22),bd=5, show="\*")

fms.pasd.place(x=650,y=270)

fms.cl=Button(fms.loginf2,text="Clear",cursor="hand2",command=lambda:clear(fms),fg="white",bg="#0b1335",font=("cooper black",16),bd=4)

fms.cl.place(x=360,y=330)

fms.lg=Button(fms.loginf2,text="Login",cursor="hand2",fg="white",bg="#0b1335",command=fms.userdatabase,font=("cooper black",16),bd=4)

fms.lg.place(x=600,y=330)

forgot\_password\_button = Button(fms.loginf2, text="Forgot Password", cursor="hand2", fg="white", bg="#0b1335", command=fms.userforgot\_password, font=("cooper black", 16, 'bold'), bd=5)

forgot\_password\_button.place(x=775, y=330)

fms.rg=Button(fms.loginf2,text="Click Here to Register",command=fms.Register,fg="white",cursor="hand2",bg="#f39c12",font=("cooper black",16),bd=6)

fms.rg.place(x=525,y=400)

def clear(fms):

fms.user.delete(0,END)

fms.pasd.delete(0,END)

fms.scr.mainloop()

# Function to fetch manager data from the SQLite database

def fetch\_user\_data(fms):

try:

# Connect to the SQLite database

conn = connect('fms.db') # Replace 'your\_database.db' with your actual database file path

cursor = conn.cursor()

# Execute the query to fetch data from the 'managers' table

cursor.execute("SELECT user\_name, email FROM users")

# Fetch all rows from the result set

rows = cursor.fetchall()

# Close the cursor and connection

cursor.close()

conn.close()

# Convert the fetched rows into a list of dictionaries

manager\_data = [{'user\_name': row[0], 'email': row[1]} for row in rows]

return manager\_data

except Error as e:

print("Error fetching user data:", e)

return []

def userforgot\_password(fms):

user\_data = fms.fetch\_user\_data()

if not user\_data:

messagebox.showerror("Error", "No user data found. Please check your database connection.")

return

forgot\_password\_window = Toplevel(fms.loginf2)

forgot\_password\_window.title("Update Password")

window\_width = 600 # Width of the window

window\_height = 275 # Height of the window

screen\_width = forgot\_password\_window.winfo\_screenwidth()

screen\_height = forgot\_password\_window.winfo\_screenheight()

x\_coordinate = (screen\_width / 2) - (window\_width / 2)

y\_coordinate = (screen\_height / 2) - (window\_height / 2)

forgot\_password\_window.geometry(f"{window\_width}x{window\_height}+{int(x\_coordinate)}+{int(y\_coordinate)}")

cooper\_black\_font = Font(family="Cooper Black", size=12)

def is\_valid\_email(email):

email\_regex = r'^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

return re.match(email\_regex, email)

def update\_user\_password():

new\_username = new\_username\_entry.get()

new\_email = new\_email\_entry.get()

new\_password = new\_password\_entry.get()

confirm\_password = confirm\_password\_entry.get()

if not new\_username or not new\_email or not new\_password or not confirm\_password:

messagebox.showerror("Error", "All fields are required.", parent=forgot\_password\_window)

return

username\_exists = any(user['user\_name'] == new\_username for user in user\_data)

email\_exists = any(user['email'] == new\_email for user in user\_data)

if not is\_valid\_email(new\_email):

messagebox.showerror("Error", "Invalid email format.", parent=forgot\_password\_window)

return

if not username\_exists or not email\_exists:

messagebox.showerror("Error", "Username or email does not exist.", parent=forgot\_password\_window)

return

if not new\_password or not confirm\_password:

messagebox.showerror("Error", "Please enter new password and confirm password.", parent=forgot\_password\_window)

return

if new\_password != confirm\_password:

messagebox.showerror("Error", "Passwords do not match.", parent=forgot\_password\_window)

return

try:

conn = connect('fms.db')

cursor = conn.cursor()

cursor.execute("UPDATE users SET password = ? WHERE user\_name = ? AND email = ?", (new\_password, new\_username, new\_email))

conn.commit()

cursor.close()

conn.close()

messagebox.showinfo("Success", "Your password has been updated.", parent=forgot\_password\_window)

forgot\_password\_window.destroy() # Close window only when update is successful

except Error as e:

messagebox.showerror("Error", "Error updating password: " + str(e), parent=forgot\_password\_window)

label\_entry\_frame = Frame(forgot\_password\_window)

label\_entry\_frame.pack(pady=10)

new\_username\_label = Label(label\_entry\_frame, text="User Name:", font=cooper\_black\_font)

new\_username\_label.grid(row=0, column=0, padx=(10, 5), pady=10, sticky="e")

new\_username\_entry = Entry(label\_entry\_frame, font=cooper\_black\_font)

new\_username\_entry.grid(row=0, column=1, padx=(0, 10), pady=10, sticky="w")

new\_email\_label = Label(label\_entry\_frame, text="Email:", font=cooper\_black\_font)

new\_email\_label.grid(row=1, column=0, padx=(10, 5), pady=10, sticky="e")

new\_email\_entry = Entry(label\_entry\_frame, font=cooper\_black\_font)

new\_email\_entry.grid(row=1, column=1, padx=(0, 10), pady=10, sticky="w")

new\_password\_label = Label(label\_entry\_frame, text="New Password:", font=cooper\_black\_font)

new\_password\_label.grid(row=2, column=0, padx=(10, 5), pady=10, sticky="e")

new\_password\_entry = Entry(label\_entry\_frame, show="\*", font=cooper\_black\_font)

new\_password\_entry.grid(row=2, column=1, padx=(0, 10), pady=10, sticky="w")

confirm\_password\_label = Label(label\_entry\_frame, text="Confirm Password:", font=cooper\_black\_font)

confirm\_password\_label.grid(row=3, column=0, padx=(10, 5), pady=10, sticky="e")

confirm\_password\_entry = Entry(label\_entry\_frame, show="\*", font=cooper\_black\_font)

confirm\_password\_entry.grid(row=3, column=1, padx=(0, 10), pady=10, sticky="w")

update\_password\_button = Button(forgot\_password\_window, text="Update Password", fg="white", cursor="hand2", bg="#0b1335", command=update\_user\_password, bd=5, font=("cooper black", 16, 'bold'))

update\_password\_button.pack(pady=10)

#---------------------------------------------------Admin Login Page------------------------------------------------------------------------------#

def Adminlogin(fms):

fms.scr.destroy()

fms.scr=Tk()

style = ttk.Style()

style.configure('Red.TLabel', foreground='red')

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.adminf1=Frame(fms.scr, bg="#ffffff")

fms.adminf1.pack(fill=BOTH,expand=YES)

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img = fms.logo.subsample(2,2)

fms.logo\_banner = Label(fms.adminf1, image=fms.logo\_img)

fms.logo\_banner.place(x=0,y=0)

fms.logo\_banner.pack(side=LEFT)

fms.home=Button(fms.adminf1,text="Home", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.home.place(x=1075,y=100)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.adminf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.tim.place(x=1000,y=50)

fms.adminf2=Frame(fms.scr,height=1080,width=1920)

fms.adminf2.pack(fill=BOTH,expand= YES)

fms.c=Canvas(fms.adminf2,height=1080,width=1920)

fms.c.pack()

fms.fmain=PhotoImage(file="main.png")

fms.c.create\_image(650,309,image=fms.fmain)

fms.c.create\_rectangle(350,100,1020,425,fill="#d3ede6",outline="white",width=6)

fms.log=Label(fms.adminf2,text="ADMIN LOGIN",fg="white",bg="#0b1335",width=27,font=("cooper black",27))

fms.log.place(x=357,y=110)

fms.lab1=Label(fms.adminf2,text="User Name",bg="#d3ede6",font=("cooper black",22))

fms.lab1.place(x=360,y=200)

fms.usera=Entry(fms.adminf2,bg="white",font=("cooper black",22),bd=5)

fms.usera.place(x=650,y=200)

fms.lab2=Label(fms.adminf2,text="Password",bg="#d3ede6",font=("cooper black",22))

fms.lab2.place(x=360,y=270)

fms.pasda=Entry(fms.adminf2,bg="white",font=("cooper black",22),bd=5, show="\*")

fms.pasda.place(x=650,y=270)

fms.cl=Button(fms.adminf2,text="Back",cursor="hand2",fg="white",bg="#0b1335",command=fms.main,font=("cooper black",16,'bold'),bd=5)

fms.cl.place(x=360,y=350)

fms.lg=Button(fms.adminf2,text="Login",cursor="hand2",fg="white",bg="#0b1335",command=fms.admindatabase,font=("cooper black",16,'bold'),bd=5)

fms.lg.place(x=650,y=350)

fms.rg=Button(fms.adminf2,text="Clear",fg="white",cursor="hand2",bg="#0b1335",command=lambda:clear(fms),bd=5,font=("cooper black",16,'bold'))

fms.rg.place(x=900,y=350)

def clear(fms):

fms.usera.delete(0,END)

fms.pasda.delete(0,END)

fms.scr.mainloop()

#---------------------------------------------------Manager Login Page----------------------------------------------------------------------------#

def Managerlogin(fms):

fms.scr.destroy()

fms.scr=Tk()

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.managerf1=Frame(fms.scr, bg="#ffffff")

fms.managerf1.pack(fill=BOTH,expand=YES)

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img = fms.logo.subsample(2,2)

fms.logo\_banner = Label(fms.managerf1, image=fms.logo\_img)

fms.logo\_banner.place(x=0,y=0)

fms.logo\_banner.pack(side=LEFT)

fms.home=Button(fms.managerf1,text="Home", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.home.place(x=1075,y=100)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.managerf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.tim.place(x=1000,y=50)

fms.managerf2=Frame(fms.scr,height=1080,width=1920)

fms.managerf2.pack(fill=BOTH,expand= YES)

fms.c=Canvas(fms.managerf2,height=1080,width=1920)

fms.c.pack()

fms.fmain=PhotoImage(file="main.png")

fms.c.create\_image(650,309,image=fms.fmain)

fms.c.create\_rectangle(350,100,1020,425,fill="#d3ede6",outline="white",width=6)

fms.log=Label(fms.managerf2,text="MANAGER LOGIN",fg="white",bg="#0b1335",width=27,font=("cooper black",27))

fms.log.place(x=357,y=110)

fms.lab1=Label(fms.managerf2,text="User Name",bg="#d3ede6",font=("cooper black",22))

fms.lab1.place(x=360,y=200)

fms.usera=Entry(fms.managerf2,bg="white",font=("cooper black",22),bd=5)

fms.usera.place(x=650,y=200)

fms.lab2=Label(fms.managerf2,text="Password",bg="#d3ede6",font=("cooper black",22))

fms.lab2.place(x=360,y=270)

fms.pasda=Entry(fms.managerf2,bg="white",font=("cooper black",22),bd=5, show="\*")

fms.pasda.place(x=650,y=270)

fms.cl = Button(fms.managerf2, text="Back", cursor="hand2", fg="white", bg="#0b1335", command=fms.main, font=("cooper black", 16, 'bold'), bd=5)

fms.cl.place(x=360, y=350)

forgot\_password\_button = Button(fms.managerf2, text="Forgot Password", cursor="hand2", fg="white", bg="#0b1335", command=fms.forgot\_password, font=("cooper black", 16, 'bold'), bd=5)

forgot\_password\_button.place(x=465, y=350)

fms.lg = Button(fms.managerf2, text="Login", cursor="hand2", fg="white", bg="#0b1335", command=fms.managerdatabase, font=("cooper black", 16, 'bold'), bd=5)

fms.lg.place(x=710, y=350)

fms.rg = Button(fms.managerf2, text="Clear", fg="white", cursor="hand2", bg="#0b1335", command=lambda: clear(fms), bd=5, font=("cooper black", 16, 'bold'))

fms.rg.place(x=910, y=350)

def clear(fms):

fms.usera.delete(0,END)

fms.pasda.delete(0,END)

fms.scr.mainloop()

# Function to fetch manager data from the SQLite database

def fetch\_manager\_data(fms):

try:

# Connect to the SQLite database

conn = connect('fms.db') # Replace 'your\_database.db' with your actual database file path

cursor = conn.cursor()

# Execute the query to fetch data from the 'managers' table

cursor.execute("SELECT user\_name, email FROM managers")

# Fetch all rows from the result set

rows = cursor.fetchall()

# Close the cursor and connection

cursor.close()

conn.close()

# Convert the fetched rows into a list of dictionaries

manager\_data = [{'user\_name': row[0], 'email': row[1]} for row in rows]

return manager\_data

except Error as e:

print("Error fetching manager data:", e)

return []

def forgot\_password(fms):

manager\_data = fms.fetch\_manager\_data()

if not manager\_data:

messagebox.showerror("Error", "No manager data found. Please check your database connection.")

return

forgot\_password\_window = Toplevel(fms.managerf2)

forgot\_password\_window.title("Update Password")

window\_width = 600 # Width of the window

window\_height = 275 # Height of the window

screen\_width = forgot\_password\_window.winfo\_screenwidth()

screen\_height = forgot\_password\_window.winfo\_screenheight()

x\_coordinate = (screen\_width / 2) - (window\_width / 2)

y\_coordinate = (screen\_height / 2) - (window\_height / 2)

forgot\_password\_window.geometry(f"{window\_width}x{window\_height}+{int(x\_coordinate)}+{int(y\_coordinate)}")

cooper\_black\_font = Font(family="Cooper Black", size=12)

def is\_valid\_email(email):

email\_regex = r'^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

return re.match(email\_regex, email)

def update\_password():

new\_username = new\_username\_entry.get()

new\_email = new\_email\_entry.get()

new\_password = new\_password\_entry.get()

confirm\_password = confirm\_password\_entry.get()

# Check if any of the fields are empty

if not new\_username or not new\_email or not new\_password or not confirm\_password:

messagebox.showerror("Error", "All fields are required.", parent=forgot\_password\_window)

return

username\_exists = any(manager['user\_name'] == new\_username for manager in manager\_data)

email\_exists = any(manager['email'] == new\_email for manager in manager\_data)

if not is\_valid\_email(new\_email):

messagebox.showerror("Error", "Invalid email format.", parent=forgot\_password\_window)

return

if not username\_exists or not email\_exists:

messagebox.showerror("Error", "Username or email does not exist.", parent=forgot\_password\_window)

return

if not new\_password or not confirm\_password:

messagebox.showerror("Error", "Please enter new password and confirm password.", parent=forgot\_password\_window)

return

if new\_password != confirm\_password:

messagebox.showerror("Error", "Passwords do not match.", parent=forgot\_password\_window)

return

try:

conn = connect('fms.db')

cursor = conn.cursor()

cursor.execute("UPDATE managers SET password = ? WHERE user\_name = ? AND email = ?", (new\_password, new\_username, new\_email))

conn.commit()

cursor.close()

conn.close()

messagebox.showinfo("Success", "Your password has been updated.", parent=forgot\_password\_window)

forgot\_password\_window.destroy() # Close window only when update is successful

except Error as e:

messagebox.showerror("Error", "Error updating password: " + str(e), parent=forgot\_password\_window)

label\_entry\_frame = Frame(forgot\_password\_window)

label\_entry\_frame.pack(pady=10)

new\_username\_label = Label(label\_entry\_frame, text="User Name:", font=cooper\_black\_font)

new\_username\_label.grid(row=0, column=0, padx=(10, 5), pady=10, sticky="e")

new\_username\_entry = Entry(label\_entry\_frame, font=cooper\_black\_font)

new\_username\_entry.grid(row=0, column=1, padx=(0, 10), pady=10, sticky="w")

new\_email\_label = Label(label\_entry\_frame, text="Email:", font=cooper\_black\_font)

new\_email\_label.grid(row=1, column=0, padx=(10, 5), pady=10, sticky="e")

new\_email\_entry = Entry(label\_entry\_frame, font=cooper\_black\_font)

new\_email\_entry.grid(row=1, column=1, padx=(0, 10), pady=10, sticky="w")

new\_password\_label = Label(label\_entry\_frame, text="New Password:", font=cooper\_black\_font)

new\_password\_label.grid(row=2, column=0, padx=(10, 5), pady=10, sticky="e")

new\_password\_entry = Entry(label\_entry\_frame, show="\*", font=cooper\_black\_font)

new\_password\_entry.grid(row=2, column=1, padx=(0, 10), pady=10, sticky="w")

confirm\_password\_label = Label(label\_entry\_frame, text="Confirm Password:", font=cooper\_black\_font)

confirm\_password\_label.grid(row=3, column=0, padx=(10, 5), pady=10, sticky="e")

confirm\_password\_entry = Entry(label\_entry\_frame, show="\*", font=cooper\_black\_font)

confirm\_password\_entry.grid(row=3, column=1, padx=(0, 10), pady=10, sticky="w")

update\_password\_button = Button(forgot\_password\_window, text="Update Password", fg="white", cursor="hand2", bg="#0b1335", command=update\_password, bd=5, font=("cooper black", 16, 'bold'))

update\_password\_button.pack(pady=10)

# update\_button = Button(forgot\_password\_window, text="Update Password", command=update\_password, font=cooper\_black\_font, bg="#0b1335", fg="#fff", padx=10, pady=5)

# update\_button.pack(pady=10)

#---------------------------------------------------All Login Pages Code End----------------------------------------------------------------------#

#---------------------------------------------------User Registration Page Start------------------------------------------------------------------#

def Register(fms):

fms.scr.destroy()

fms.scr=Tk()

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

#fms.scr.resizable(False, False)

fms.regf1=Frame(fms.scr, bg="#ffffff")

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img = fms.logo.subsample(2,2)

fms.logo\_banner = Label(fms.regf1, image=fms.logo\_img)

fms.logo\_banner.place(x=0,y=0)

fms.logo\_banner.pack(side=LEFT)

fms.home=Button(fms.regf1,text="Home",command=fms.main,bg="#0b1335",cursor="hand2",fg="white",font=("cooper black",16))

fms.home.place(x=800,y=100)

fms.adlog=Button(fms.regf1,text="Admin Login",command=fms.Adminlogin,cursor="hand2",bg="#0b1335",fg="white",font=("cooper black",16))

fms.adlog.place(x=950,y=100)

fms.abt=Button(fms.regf1,text="Manager Login",command=fms.Managerlogin,bg="#0b1335",cursor="hand2",fg="white",font=("cooper black",16))

fms.abt.place(x=1210,y=100)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.regf1,text=fms.localtime,fg="white",font=("cooper black",16),bg="#0b1335")

fms.tim.place(x=925,y=50)

fms.regf1.pack(fill=BOTH,expand=1)

fms.regf2=Frame(fms.scr,height=1080,width=1920)

fms.c=Canvas(fms.regf2,height=1080,width=1920)

fms.c.pack()

fms.fmain = PhotoImage(file="main.png")

fms.c.create\_image(650,300,image=fms.fmain)

fms.c.create\_rectangle(180,110,1150,435,fill="#d3ede6",outline="white",width=6)

fms.log=Label(fms.regf2,text="REGISTRATION",fg="white",bg="#0b1335",width=20,font=("cooper black",27))

fms.log.place(x=480,y=120)

fms.lab1=Label(fms.regf2,text="First Name",bg="#d3ede6",font=("cooper black",18))

fms.lab1.place(x=190,y=200)

fms.first=Entry(fms.regf2,bg="white",width=15,font=("cooper black",18),bd=5)

fms.first.place(x=430,y=200)

fms.lab2=Label(fms.regf2,text="Last Name",bg="#d3ede6",font=("cooper black",18))

fms.lab2.place(x=730,y=200)

fms.last=Entry(fms.regf2,bg="white",width=15,font=("cooper black",18),bd=5)

fms.last.place(x=920,y=200)

fms.lab3=Label(fms.regf2,text="User Name",bg="#d3ede6",font=("cooper black",18))

fms.lab3.place(x=190,y=250)

fms.usern=Entry(fms.regf2,bg="white",width=15,font=("cooper black",18),bd=5)

fms.usern.place(x=430,y=250)

fms.lab4=Label(fms.regf2,text="Password",bg="#d3ede6",font=("cooper black",18))

fms.lab4.place(x=730,y=250)

fms.passd=Entry(fms.regf2,bg="white",width=15,font=("cooper black",18),bd=5, show="\*")

fms.passd.place(x=920,y=250)

fms.lab5 = Label(fms.regf2, text="Email", bg="#d3ede6", font=("cooper black", 18))

fms.lab5.place(x=190, y=300)

fms.email=Entry(fms.regf2,bg="white",width=15,font=("cooper black",18),bd=5)

fms.email.place(x=430, y=300)

fms.lab6 = Label(fms.regf2, text="Mobile No.", bg="#d3ede6", font=("cooper black", 18))

fms.lab6.place(x=730, y=300)

fms.mob = Entry(fms.regf2, bg="white", width=15, font=("cooper black", 18), bd=5)

fms.mob.place(x=920, y=300)

fms.bc=Button(fms.regf2,text="Back",cursor="hand2",command=fms.main,fg="white",bg="#0b1335",font=("cooper black",18),bd=5)

fms.bc.place(x=370,y=370)

fms.rg=Button(fms.regf2,text="Register",cursor="hand2",fg="white",bg="#0b1335",command=fms.userReg,font=("cooper black",18),bd=5)

fms.rg.place(x=610,y=370)

fms.cl=Button(fms.regf2,text="Clear",cursor="hand2",fg="white",bg="#0b1335",command=lambda:clear(fms),font=("cooper black",18),bd=5)

fms.cl.place(x=910,y=370)

fms.regf2.pack(fill=BOTH,expand=1)

def clear(fms):

fms.first.delete(0,END)

fms.last.delete(0,END)

fms.usern.delete(0,END)

fms.passd.delete(0,END)

fms.email.delete(0,END)

fms.mob.delete(0,END)

fms.scr.mainloop()

#---------------------------------------------------User Registration Page End--------------------------------------------------------------------#

#---------------------------------------------------Admin Dashboard Code Start -------------------------------------------------------------------#

#---------------------------------------------------Admin All Managers Page-----------------------------------------------------------------------#

def Viewmanagers(fms):

fms.scr.destroy()

fms.scr = Tk()

style = ttk.Style()

style.configure('Red.TLabel', foreground='red')

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.managerdashf1= Frame(fms.scr, bg="#ffffff")

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img1 = fms.logo.subsample(2,2)

fms.logo\_banner1 = Label(fms.managerdashf1, image=fms.logo\_img1)

fms.logo\_banner1.pack(side=LEFT)

fms.managerlabel= Label(text="Weclome to Admin Dashboard!!", bg="#0b1335", fg="white",font=("cooper black",14))

fms.managerlabel.place(x=1000,y=50)

fms.logout=Button(fms.managerdashf1,text="LOGOUT", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.logout.place(x=1075,y=100)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.managerdashf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.managerdashf1.pack(fill=BOTH)

fms.sidebar = Frame(fms.scr, bg="#d3ede6")

fms.sidebar.pack(side=LEFT, fill=Y)

# Add buttons to the sidebar

fms.viewmanagers= Button(fms.sidebar,text="VIEW MANAGERS", command=fms.Viewmanagers, bg="#f39c12",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewmanagers.grid(row=3, column=0, ipadx=11, padx=25, pady=25)

fms.viewusers= Button(fms.sidebar,text="VIEW USERS", command=fms.Viewusers, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewusers.grid(row=5, column=0, ipadx=36, padx=25, pady=25)

fms.viewdonations= Button(fms.sidebar,text="VIEW DONATIONS", command=fms.Viewdonations, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewdonations.grid(row=6, column=0, ipadx=8, padx=25, pady=25)

fms.viewreports = Button(fms.sidebar,text="REPORTS", command=fms.Viewreports, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewreports.grid(row=7, column=0, ipadx=53, padx=25, pady=25)

def show\_custom\_messages(message):

# Create a new top-level window

message\_dialog = Toplevel()

message\_dialog.title("Message")

# Set window size and position it in the center of the screen

window\_width = 350

window\_height = 150

screen\_width = message\_dialog.winfo\_screenwidth()

screen\_height = message\_dialog.winfo\_screenheight()

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

message\_dialog.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Add an icon label in #32CD32 color

style = ttk.Style()

style.configure("Error.TLabel", foreground="#32CD32")

error\_icon = ttk.Label(message\_dialog, text="\u24D8", style="Error.TLabel", font=("Cooper Black", 24))

error\_icon.pack(pady=10)

label = Label(message\_dialog, text=message, fg="#32CD32", font=("Cooper Black", 12))

label.pack(padx=10, pady=0)

# Add an "OK" button to close the dialog

ok\_button = Button(message\_dialog, text="OK", command=message\_dialog.destroy, width=10, font=("Cooper Black", 12))

ok\_button.pack(pady=10)

def show\_custom\_error(message):

# Create a new top-level window

error\_dialog = Toplevel()

error\_dialog.title("Error Message")

# Set window size and position it in the center of the screen

window\_width = 650

window\_height = 150

screen\_width = error\_dialog.winfo\_screenwidth()

screen\_height = error\_dialog.winfo\_screenheight()

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

error\_dialog.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Add an icon label in red color

style = ttk.Style()

style.configure("Error.TLabel", foreground="red")

error\_icon = ttk.Label(error\_dialog, text="\u26A0", style="Error.TLabel", font=("Cooper Black", 24))

error\_icon.pack(pady=10)

label = Label(error\_dialog, text=message, fg="red", font=("Cooper Black", 12))

label.pack(padx=10, pady=0)

# Add an "OK" button to close the dialog

ok\_button = Button(error\_dialog, text="OK", command=error\_dialog.destroy, width=10, font=("Cooper Black", 12))

ok\_button.pack(pady=10)

# Function to load data into the Treeview

def managers\_data():

# Clear the existing data in the Treeview

tree.delete(\*tree.get\_children())

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

# Retrieve the managers data from the database

cursor.execute("SELECT manager\_id, first\_name, last\_name, user\_name, department, email FROM managers")

managers = cursor.fetchall()

# Insert the data into the Treeview

for manager in managers:

tree.insert("", "end", values=(manager[0], manager[1], manager[2], manager[3], manager[4], manager[5]), tags=("my\_font"))

# Close the connection

conn.close()

# Function to edit a manager's details

def edit\_managers():

# Get the selected item from the Treeview

selected\_item = tree.focus()

if not selected\_item:

show\_custom\_error("Please select manager to edit")

return

#Get the values of the selected record

values = tree.item(selected\_item, "values")

manager\_id = values[0]

def submit():

# Get values from entry widgets

first\_name = first\_name\_entry.get()

last\_name = last\_name\_entry.get()

user\_name = user\_name\_entry.get()

department = selected\_department.get()

email = email\_entry.get()

# Validate fields

if not (first\_name and last\_name and user\_name and department and email):

show\_custom\_error("All fields are required")

return

# Validate email format

if not re.match(r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$', email):

show\_custom\_error("Invalid email format")

return

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

try:

# Check if the user\_name or email already exists for other managers

cursor.execute("SELECT COUNT(\*) FROM managers WHERE (user\_name=? OR email=?) AND manager\_id != ?", (user\_name, email, manager\_id))

existing\_managers\_count = cursor.fetchone()[0]

if existing\_managers\_count > 0:

# User\_name or email already exists for other managers, show error message

show\_custom\_error("User name or email already exists. Please choose a different one.")

else:

# Proceed with updating the manager details

# Print the SQL query to verify correctness

print(f"Executing EDIT query: EDIT FROM managers WHERE manager\_id = {manager\_id}")

# Update manager in the managers table

cursor.execute("UPDATE managers SET first\_name=?, last\_name=?, user\_name=?, department=?, email=? WHERE manager\_id=?",

(first\_name, last\_name, user\_name, department, email, manager\_id))

# Commit the changes

conn.commit()

# Show success message

show\_custom\_messages("Manager details updated successfully")

# Refresh the Treeview

managers\_data()

except Exception as e:

# Show error message if update fails

show\_custom\_error(f"Failed to update the manager: {str(e)}")

finally:

# Close the connection

conn.close()

# Close the edit window

edit\_window.destroy()

# Create a Toplevel window for editing a manager

edit\_window = Toplevel()

edit\_window.title("Edit Manager")

# Calculate the center coordinates of the screen

screen\_width = edit\_window.winfo\_screenwidth()

screen\_height = edit\_window.winfo\_screenheight()

window\_width = 700

window\_height = 225

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

# Set the geometry of the window to be centered and larger

edit\_window.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Create labels and entry widgets for each field

Label(edit\_window, text="First Name:", font=("Cooper Black", 12)).grid(row=0, column=0, padx=10, pady=10, sticky="e")

first\_name\_entry = Entry(edit\_window, font=("Cooper Black", 12))

first\_name\_entry.grid(row=0, column=1, padx=10, pady=10)

Label(edit\_window, text="Last Name:", font=("Cooper Black", 12)).grid(row=1, column=0, padx=10, pady=10, sticky="e")

last\_name\_entry = Entry(edit\_window, font=("Cooper Black", 12))

last\_name\_entry.grid(row=1, column=1, padx=10, pady=10)

Label(edit\_window, text="User Name:", font=("Cooper Black", 12)).grid(row=0, column=2, padx=10, pady=10, sticky="e")

user\_name\_entry = Entry(edit\_window, font=("Cooper Black", 12))

user\_name\_entry.grid(row=0, column=3, padx=10, pady=10)

# user\_name\_entry.insert(0, values[3]) # Insert the user\_name from the selected record

Label(edit\_window, text="Email:", font=("Cooper Black", 12)).grid(row=3, column=0, padx=10, pady=10, sticky="e")

email\_entry = Entry(edit\_window, font=("Cooper Black", 12))

email\_entry.grid(row=3, column=1, padx=10, pady=10)

# email\_entry.insert(0, values[5]) # Insert the email from the selected record

Label(edit\_window, text="Department:", font=("Cooper Black", 12)).grid(row=1, column=2, padx=10, pady=10, sticky="e")

department\_options = ["One Central", "Eatry", "Merill", "Dine and Connect", "Catering"]

selected\_department = StringVar(edit\_window)

selected\_department.set(department\_options[0])

department\_dropdown = OptionMenu(edit\_window, selected\_department, \*department\_options)

department\_dropdown.config(font=("Cooper Black", 12))

department\_dropdown.grid(row=1, column=3, padx=10, pady=10, sticky="w")

# Populate the entry widgets with the selected manager's data

first\_name\_entry.insert(0, values[1])

last\_name\_entry.insert(0, values[2])

user\_name\_entry.insert(0, values[3])

selected\_department.set(values[4])

email\_entry.insert(0, values[5])

# Create a Submit button

submit\_button = Button(edit\_window, text="Submit", command=submit, font=("Cooper Black", 12))

submit\_button.grid(row=4, columnspan=4, pady=10)

# Main loop for the edit window

edit\_window.mainloop()

# Function to delete a manager

def delete\_managers():

# Get the selected item from the Treeview

selected\_item = tree.focus()

if not selected\_item:

show\_custom\_error("Please select manager to delete")

return

# Fetch the manager\_id from the dictionary

values = tree.item(selected\_item, "values")

manager\_id = values[0]

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

try:

# Check if the manager has any donations associated with them

cursor.execute("SELECT COUNT(\*) FROM donation\_details WHERE manager\_id = ?", (manager\_id,))

donation\_count = cursor.fetchone()[0]

if donation\_count > 0:

# Manager has donations associated with them, show error message

show\_custom\_error("Cannot delete manager. Manager has donations associated with them.")

else:

# Proceed with deleting the manager

# Print the SQL query to verify correctness

print(f"Executing DELETE query: DELETE FROM managers WHERE manager\_id = {manager\_id}")

# Delete the record from the managers table

cursor.execute("DELETE FROM managers WHERE manager\_id = ?", (manager\_id,))

# Commit the changes

conn.commit()

# Show success message

show\_custom\_messages("Manager deleted successfully")

# Refresh the Treeview

managers\_data()

except Exception as e:

# Show error message if deletion fails

show\_custom\_error(f"Failed to delete manager: {str(e)}")

finally:

# Close the connection

conn.close()

def create\_managers\_table\_if\_not\_exists():

conn = connect('fms.db') # Replace 'your\_database.db' with your actual database name

cursor = conn.cursor()

# SQL query to create the table if it does not exist

create\_table\_query = '''

CREATE TABLE IF NOT EXISTS managers (

id INTEGER PRIMARY KEY AUTOINCREMENT,

first\_name TEXT NOT NULL,

last\_name TEXT NOT NULL,

user\_name TEXT NOT NULL UNIQUE,

password TEXT NOT NULL,

email TEXT NOT NULL,

department TEXT NOT NULL

);

'''

cursor.execute(create\_table\_query)

conn.commit()

conn.close()

# Function to add a new manager

def add\_managers():

def submit():

# Get values from entry widgets

user\_name = user\_name\_entry.get()

password = password\_entry.get()

email = email\_entry.get()

first\_name = first\_name\_entry.get()

last\_name = last\_name\_entry.get()

department = selected\_department.get()

# Validate fields

if not (user\_name and password and email and first\_name and last\_name and department):

# Example usage

show\_custom\_error("All fields are required")

return

# Validate email format

if not re.match(r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$', email):

show\_custom\_error("Invalid email format")

return

try:

create\_managers\_table\_if\_not\_exists() # Check and create table if not exists

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

# Insert new manager into the managers table

cursor.execute("INSERT INTO managers (first\_name, last\_name, user\_name, password, email, department) VALUES (?, ?, ?, ?, ?, ?)",

(first\_name, last\_name, user\_name, password, email, department))

# Commit the changes

conn.commit()

# Show success message

show\_custom\_messages("Manager added successfully")

# Refresh the Treeview

managers\_data()

except Exception as e:

# Show error message if insertion fails

show\_custom\_error("Failed to add manager: {str(e)}")

finally:

# Close the connection

conn.close()

# Close the add window

add\_window.destroy()

# Create a Toplevel window for adding a new manager

add\_window = Toplevel()

add\_window.title("Add Manager")

# Calculate the center coordinates of the screen

screen\_width = add\_window.winfo\_screenwidth()

screen\_height = add\_window.winfo\_screenheight()

window\_width = 750

window\_height = 225

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

# Set the geometry of the window to be centered and larger

add\_window.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Create labels and entry widgets for each field

Label(add\_window, text="First Name:", font=("Cooper Black", 12)).grid(row=0, column=0, padx=10, pady=10, sticky="e")

first\_name\_entry = Entry(add\_window, font=("Cooper Black", 12))

first\_name\_entry.grid(row=0, column=1, padx=10, pady=10)

Label(add\_window, text="Last Name:", font=("Cooper Black", 12)).grid(row=1, column=0, padx=10, pady=10, sticky="e")

last\_name\_entry = Entry(add\_window, font=("Cooper Black", 12))

last\_name\_entry.grid(row=1, column=1, padx=10, pady=10)

Label(add\_window, text="User Name:", font=("Cooper Black", 12)).grid(row=2, column=0, padx=10, pady=10, sticky="e")

user\_name\_entry = Entry(add\_window, font=("Cooper Black", 12))

user\_name\_entry.grid(row=2, column=1, padx=10, pady=10)

Label(add\_window, text="Password:", font=("Cooper Black", 12)).grid(row=0, column=2, padx=10, pady=10, sticky="e")

password\_entry = Entry(add\_window, font=("Cooper Black", 12))

password\_entry.grid(row=0, column=3, padx=10, pady=10)

Label(add\_window, text="Email:", font=("Cooper Black", 12)).grid(row=2, column=2, padx=10, pady=10, sticky="e")

email\_entry = Entry(add\_window, font=("Cooper Black", 12))

email\_entry.grid(row=2, column=3, padx=10, pady=10)

Label(add\_window, text="Department:", font=("Cooper Black", 12)).grid(row=1, column=2, padx=10, pady=10, sticky="e")

department\_options = ["One Central", "Eatry", "Merill", "Dine and Connect", "Catering"]

selected\_department = StringVar(add\_window)

selected\_department.set(department\_options[0])

department\_dropdown = OptionMenu(add\_window, selected\_department, \*department\_options)

department\_dropdown.config(font=("Cooper Black", 12))

department\_dropdown.grid(row=1, column=3, padx=10, pady=10, sticky="w")

# Create a Submit button

submit\_button = Button(add\_window, text="Submit", command=submit, font=("Cooper Black", 12))

submit\_button.grid(row=4, columnspan=4, pady=10)

# Main loop for the add window

add\_window.mainloop()

# Create the main content frame

content\_frame = Frame(fms.scr, bg="lightgray")

content\_frame.pack(side=LEFT, fill=BOTH, expand=True)

# Create a label for the title

title\_label = Label(content\_frame, text="VIEW MANAGERS", font=("Cooper Black", 16), bg="#0b1335", fg="white", pady=10)

title\_label.pack(fill=X)

title\_label.pack(pady=(0, 85))

# Create treeview

tree = ttk.Treeview(content\_frame, columns=("Manager ID", "First Name", "Last Name", "User Name", "Department", "Email"), show="headings", padding=(0, 5))

# Set the font for the headings

tree.heading("Manager ID", text="Manager ID", anchor=CENTER)

tree.heading("First Name", text="First Name", anchor=CENTER)

tree.heading("Last Name", text="Last Name", anchor=CENTER)

tree.heading("User Name", text="User Name", anchor=CENTER)

tree.heading("Department", text="Department", anchor=CENTER)

tree.heading("Email", text="Email", anchor=CENTER)

# Configure Treeview style

style = ttk.Style()

style.configure("Custom.Treeview", font=("Cooper Black", 12))

style.configure("Treeview.Row", padding=(0, 5))

# Set the background color for the headers

style.configure("Treeview.Heading", background="gray")

# Define a tag named "my\_font" with the desired font

tree.tag\_configure("my\_font", font=("Cooper Black", 12))

tree.pack()

# Connect to the SQLite3 database

conn = connect('fms.db')

cursor = conn.cursor()

# Fetch and display data

cursor.execute("SELECT manager\_id, first\_name, last\_name, user\_name, department, email FROM managers")

managers = cursor.fetchall()

for manager in managers:

tree.insert("", END, values=(manager[0], manager[1], manager[2], manager[3], manager[4], manager[5]))

# Apply the "my\_font" tag to all items in the treeview

for manager in tree.get\_children():

tree.item(manager, tags=("my\_font",))

conn.close()

# Buttons for CRUD operations

add\_button = Button(content\_frame, text="ADD MANAGER", command=add\_managers, bg="#f39c12", fg="white", font=("Cooper Black", 12))

add\_button.pack(side=LEFT, padx=35, pady=10)

edit\_button = Button(content\_frame, text="EDIT MANAGER", command=edit\_managers, bg="#3498db", fg="white", font=("Cooper Black", 12))

edit\_button.pack(side=LEFT, padx=325, pady=10)

delete\_button = Button(content\_frame, text="DELETE MANAGER", command=delete\_managers, bg="#e74c3c", fg="white", font=("Cooper Black", 12))

delete\_button.pack(side=LEFT, padx=35, pady=10)

fms.scr.mainloop()

#---------------------------------------------------Admin All Users Page--------------------------------------------------------------------------#

def Viewusers(fms):

fms.scr.destroy()

fms.scr = Tk()

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.managerdashf1= Frame(fms.scr, bg="#ffffff")

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img1 = fms.logo.subsample(2,2)

fms.logo\_banner1 = Label(fms.managerdashf1, image=fms.logo\_img1)

fms.logo\_banner1.pack(side=LEFT)

fms.managerlabel= Label(text="Weclome to Admin Dashboard!!", bg="#0b1335", fg="white",font=("cooper black",14))

fms.managerlabel.place(x=1000,y=50)

fms.logout=Button(fms.managerdashf1,text="LOGOUT", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.logout.place(x=1075,y=100)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.managerdashf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.managerdashf1.pack(fill=BOTH)

fms.sidebar = Frame(fms.scr, bg="#d3ede6")

fms.sidebar.pack(side=LEFT, fill=Y)

# Add buttons to the sidebar

fms.viewmanagers= Button(fms.sidebar,text="VIEW MANAGERS", command=fms.Viewmanagers, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewmanagers.grid(row=3, column=0, ipadx=11, padx=25, pady=25)

fms.viewusers= Button(fms.sidebar,text="VIEW USERS", command=fms.Viewusers, bg="#f39c12",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewusers.grid(row=5, column=0, ipadx=36, padx=25, pady=25)

fms.viewdonations= Button(fms.sidebar,text="VIEW DONATIONS", command=fms.Viewdonations, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewdonations.grid(row=6, column=0, ipadx=8, padx=25, pady=25)

fms.viewreports = Button(fms.sidebar,text="REPORTS", command=fms.Viewreports, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewreports.grid(row=7, column=0, ipadx=53, padx=25, pady=25)

# Create the main content frame

content\_frame = Frame(fms.scr, bg="lightgray")

content\_frame.pack(side=LEFT, fill=BOTH, expand=True)

# Create a label for the title

title\_label = Label(content\_frame, text="VIEW USERS", font=("Cooper Black", 16), bg="#0b1335", fg="white", pady=10)

title\_label.pack(fill=X)

title\_label.pack(pady=(0, 85))

# Create treeview

tree = ttk.Treeview(content\_frame, columns=("User ID", "First Name", "Last Name", "User Name", "Email", "Mobile No"), show="headings", padding=(0, 5))

# Set the font for the headings

tree.heading("User ID", text="User ID", anchor=CENTER)

tree.heading("First Name", text="First Name", anchor=CENTER)

tree.heading("Last Name", text="Last Name", anchor=CENTER)

tree.heading("User Name", text="User Name", anchor=CENTER)

tree.heading("Email", text="Email", anchor=CENTER)

tree.heading("Mobile No", text="Mobile No", anchor=CENTER)

# Configure Treeview style

style = ttk.Style()

style.configure("Custom.Treeview", font=("Cooper Black", 12))

style.configure("Treeview.Row", padding=(0, 5))

# Set the background color for the headers

style.configure("Treeview.Heading", background="gray")

# Define a tag named "my\_font" with the desired font

tree.tag\_configure("my\_font", font=("Cooper Black", 12))

tree.pack()

# Connect to the SQLite3 database

conn = connect('fms.db')

cursor = conn.cursor()

# Fetch and display data

cursor.execute("SELECT user\_id, first\_name, last\_name, user\_name, email, mobile\_no FROM users")

users = cursor.fetchall()

for user in users:

tree.insert("", END, values=(user[0], user[1], user[2], user[3], user[4], user[5]))

# Apply the "my\_font" tag to all items in the treeview

for user in tree.get\_children():

tree.item(user, tags=("my\_font",))

conn.close()

fms.scr.mainloop()

#---------------------------------------------------Admin All Donations Page----------------------------------------------------------------------#

def Viewdonations(fms):

fms.scr.destroy()

fms.scr = Tk()

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.managerdashf1= Frame(fms.scr, bg="#ffffff")

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img1 = fms.logo.subsample(2,2)

fms.logo\_banner1 = Label(fms.managerdashf1, image=fms.logo\_img1)

fms.logo\_banner1.pack(side=LEFT)

fms.managerlabel= Label(text="Weclome to Admin Dashboard!!", bg="#0b1335", fg="white",font=("cooper black",14))

fms.managerlabel.place(x=1000,y=50)

fms.logout=Button(fms.managerdashf1,text="LOGOUT", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.logout.place(x=1075,y=100)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.managerdashf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.managerdashf1.pack(fill=BOTH)

fms.sidebar = Frame(fms.scr, bg="#d3ede6")

fms.sidebar.pack(side=LEFT, fill=Y)

# Add buttons to the sidebar

fms.viewmanagers= Button(fms.sidebar,text="VIEW MANAGERS", command=fms.Viewmanagers, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewmanagers.grid(row=3, column=0, ipadx=11, padx=25, pady=25)

fms.viewusers= Button(fms.sidebar,text="VIEW USERS", command=fms.Viewusers, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewusers.grid(row=5, column=0, ipadx=36, padx=25, pady=25)

fms.viewdonations= Button(fms.sidebar,text="VIEW DONATIONS", command=fms.Viewdonations, bg="#f39c12",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewdonations.grid(row=6, column=0, ipadx=8, padx=25, pady=25)

fms.viewreports = Button(fms.sidebar,text="REPORTS", command=fms.Viewreports, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewreports.grid(row=7, column=0, ipadx=53, padx=25, pady=25)

# Create the main content frame

content\_frame = Frame(fms.scr, bg="lightgray")

content\_frame.pack(side=LEFT, fill=BOTH, expand=True)

# Create a label for the title

title\_label = Label(content\_frame, text="VIEW DONATIONS", font=("Cooper Black", 16), bg="#0b1335", fg="white", pady=10)

title\_label.pack(fill=X)

title\_label.pack(pady=(0, 85))

# Create treeview

tree = ttk.Treeview(content\_frame, columns=("Donation ID", "Item Name", "Item Type", "Calories", "Amount(Lbs)", "Servings", "Added By", "Status", "Accepted By"), show="headings", padding=(0, 5))

# Set the font for the headings

tree.heading("Donation ID", text="Donation ID", anchor=CENTER)

tree.heading("Item Name", text="Item Name", anchor=CENTER)

tree.heading("Item Type", text="Item Type", anchor=CENTER)

tree.heading("Calories", text="Calories", anchor=CENTER)

tree.heading("Amount(Lbs)", text="Amount(Lbs)", anchor=CENTER)

tree.heading("Servings", text="Servings", anchor=CENTER)

tree.heading("Added By", text="Added By", anchor=CENTER)

tree.heading("Status", text="Status", anchor=CENTER)

tree.heading("Accepted By", text="Accepted By", anchor=CENTER)

# Configure Treeview style

style = ttk.Style()

style.configure("Custom.Treeview", font=("Cooper Black", 12))

style.configure("Treeview.Row", padding=(0, 5))

# Set the background color for the headers

style.configure("Treeview.Heading", background="gray")

# Define a tag named "my\_font" with the desired font

tree.tag\_configure("my\_font", font=("Cooper Black", 12))

# Set the column widths

tree.column("Donation ID", width=100)

tree.column("Item Name", width=170)

tree.column("Item Type", width=135)

tree.column("Calories", width=135)

tree.column("Amount(Lbs)", width=135)

tree.column("Servings", width=135)

tree.column("Added By", width=135)

tree.column("Status", width=135)

tree.column("Accepted By", width=135)

tree.pack()

# Connect to the SQLite3 database

conn = connect('fms.db')

cursor = conn.cursor()

# Fetch and display data

cursor.execute("SELECT donation\_id, item\_name, item\_type, calories, amount\_lb, servings, manager\_id, donation\_status, user\_id FROM donation\_details")

donations = cursor.fetchall()

for donation in donations:

# Fetch the manager's user\_name from the managers table

cursor.execute("SELECT user\_name FROM managers WHERE manager\_id=?", (donation[6],))

manager\_result = cursor.fetchone()

manager\_name = manager\_result[0] if manager\_result else None # Assuming manager\_id is at index 7

# Fetch the user's user\_name from the users table

cursor.execute("SELECT user\_name FROM users WHERE user\_id=?", (donation[8],))

user\_result = cursor.fetchone()

user\_name = user\_result[0] if user\_result else None # Assuming user\_id is at index 9

tree.insert("", END, values=(donation[0], donation[1], donation[2], donation[3], donation[4], donation[5], manager\_name, donation[7], user\_name))

# Apply the "my\_font" tag to all items in the treeview

for donation in tree.get\_children():

tree.item(donation, tags=("my\_font",))

conn.close()

fms.scr.mainloop()

#---------------------------------------------------Admin Reports Page----------------------------------------------------------------------------#

def Viewreports(fms):

fms.scr.destroy()

fms.scr = Tk()

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.admindashf1= Frame(fms.scr, bg="#ffffff")

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img1 = fms.logo.subsample(2,2)

fms.logo\_banner1 = Label(fms.admindashf1, image=fms.logo\_img1)

fms.logo\_banner1.pack(side=LEFT)

fms.adminlabel= Label(text="Weclome to Admin Dashboard!!", bg="#0b1335", fg="white",font=("cooper black",14))

fms.adminlabel.place(x=1000,y=50)

fms.logout=Button(fms.admindashf1,text="LOGOUT", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.logout.place(x=1075,y=100)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.admindashf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.admindashf1.pack(fill=BOTH)

fms.sidebar = Frame(fms.scr, bg="#d3ede6")

fms.sidebar.pack(side=LEFT, fill=Y)

# Add buttons to the sidebar

fms.viewmanagers= Button(fms.sidebar,text="VIEW MANAGERS", command=fms.Viewmanagers, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewmanagers.grid(row=3, column=0, ipadx=11, padx=25, pady=25)

fms.viewusers= Button(fms.sidebar,text="VIEW USERS", command=fms.Viewusers, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewusers.grid(row=5, column=0, ipadx=36, padx=25, pady=25)

fms.viewdonations= Button(fms.sidebar,text="VIEW DONATIONS", command=fms.Viewdonations, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewdonations.grid(row=6, column=0, ipadx=8, padx=25, pady=25)

fms.viewreports = Button(fms.sidebar,text="REPORTS", command=fms.Viewreports, bg="#f39c12",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewreports.grid(row=7, column=0, ipadx=53, padx=25, pady=25)

def show\_custom\_messages(message):

# Create a new top-level window

message\_dialog = Toplevel()

message\_dialog.title("Message")

# Set window size and position it in the center of the screen

window\_width = 450

window\_height = 150

screen\_width = message\_dialog.winfo\_screenwidth()

screen\_height = message\_dialog.winfo\_screenheight()

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

message\_dialog.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Add an icon label in #32CD32 color

style = ttk.Style()

style.configure("Error.TLabel", foreground="#32CD32")

error\_icon = ttk.Label(message\_dialog, text="\u24D8", style="Error.TLabel", font=("Cooper Black", 24))

error\_icon.pack(pady=10)

label = Label(message\_dialog, text=message, fg="#32CD32", font=("Cooper Black", 12))

label.pack(padx=10, pady=0)

# Add an "OK" button to close the dialog

ok\_button = Button(message\_dialog, text="OK", command=message\_dialog.destroy, width=10, font=("Cooper Black", 12))

ok\_button.pack(pady=10)

def export\_to\_pdf(data):

pdf = FPDF()

pdf.add\_page()

pdf.set\_font("Arial", size=8)

# Add table headers

headers = ["Donation ID", "Item Name", "Item Type", "Calories", "Amount (lb)", "Servings", "Donation Status"]

col\_width = 24.5

row\_height = 5

for header in headers:

pdf.cell(col\_width, row\_height, header, 1, 0, 'C')

pdf.ln()

# Add table data

for item in data:

# Extract item values and add them to the PDF

item\_values = tree.item(item, "values")

for value in item\_values:

pdf.cell(col\_width, row\_height, str(value), 1, 0, 'C')

pdf.ln()

# Specify the file path for the PDF

pdf\_file\_path = "C:/Users/navee/Downloads/git-prjcts/fms/donations\_report.pdf"

pdf.output(pdf\_file\_path)

# Show custom message after successful export

show\_custom\_messages("Donations PDF file exported successfully.")

def export\_to\_excel(data):

values = []

for item in data:

item\_values = tree.item(item, "values")

values.append([item\_values[i] for i in range(len(item\_values))]) # Convert tuple to list

# Define column names

columns = ["Donation ID", "Item Name", "Item Type", "Calories", "Amount (lb)", "Servings", "Donation Status"]

# Create DataFrame

df = pd.DataFrame(values, columns=columns)

# Specify the full path where you want to save the file

file\_path = "C:/Users/navee/Downloads/git-prjcts/fms/donations\_report.xlsx"

# Export to Excel

try:

df.to\_excel(file\_path, index=False)

print("Excel file exported successfully.")

# Show custom message after successful export

show\_custom\_messages("Donations Excel file exported successfully.")

except Exception as e:

print(f"Error exporting Excel file: {e}")

def fetch\_and\_display\_data():

conn = connect('fms.db')

cursor = conn.cursor()

cursor.execute("SELECT donation\_id, item\_name, item\_type, calories, amount\_lb, servings, donation\_status FROM donation\_details")

donations = cursor.fetchall()

for donation in donations:

tree.insert("", END, values=(donation[0], donation[1], donation[2], donation[3], donation[4], donation[5], donation[6]))

for donation in tree.get\_children():

tree.item(donation, tags=("my\_font",))

conn.close()

# Create the main content frame

# fms = Tk()

# fms.scr = Toplevel()

# Create the main content frame

content\_frame = Frame(fms.scr, bg="lightgray")

content\_frame.pack(side=LEFT, fill=BOTH, expand=True)

# Create a label for the title

title\_label = Label(content\_frame, text="REPORTS", font=("Cooper Black", 16), bg="#0b1335", fg="white", pady=10)

title\_label.pack(fill=X)

title\_label.pack(pady=(0, 85))

# Your Treeview widget declaration goes here

tree = ttk.Treeview()

# Define tree columns

tree["columns"] = ("Donation ID", "Item Name", "Item Type", "Calories", "Amount (lb)", "Servings", "Donation Status")

# Set column headings

for column in tree["columns"]:

tree.heading(column, text=column)

# Fetch and display data

fetch\_and\_display\_data()

# Create PDF export button

pdf\_button = Button(content\_frame, text="EXPORT DONATIONS to PDF", command=lambda: export\_to\_pdf(tree.get\_children()), bg="#f39c12", fg="white", font=("Cooper Black", 14))

pdf\_button.pack(side=LEFT, padx=(275, 25), pady=0, anchor="center")

# Create Excel export button

excel\_button = Button(content\_frame, text="EXPORT DONATIONS to EXCEL", command=lambda: export\_to\_excel(tree.get\_children()), bg="#3498db", fg="white", font=("Cooper Black", 14))

excel\_button.pack(side=LEFT, padx=(25, 100), pady=0, anchor="center")

fms.scr.mainloop()

#---------------------------------------------------Admin Dashboard Code End----------------------------------------------------------------------#

#---------------------------------------------------Manager Dashboard Code Start------------------------------------------------------------------#

#---------------------------------------------------Manager Main Page-----------------------------------------------------------------------------#

def Managerdetails(fms, manager\_id):

fms.scr.destroy()

fms.scr = Tk()

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.managerdashf1= Frame(fms.scr, bg="#ffffff")

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img1 = fms.logo.subsample(2,2)

fms.logo\_banner1 = Label(fms.managerdashf1, image=fms.logo\_img1)

fms.logo\_banner1.pack(side=LEFT)

fms.managerlabel= Label(text="Weclome to Manager Dashboard!!", bg="#0b1335", fg="white",font=("cooper black",14))

fms.managerlabel.place(x=1000,y=50)

fms.logout=Button(fms.managerdashf1,text="LOGOUT", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.logout.place(x=1075,y=100)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.managerdashf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.managerdashf1.pack(fill=BOTH)

fms.sidebar = Frame(fms.scr, bg="#d3ede6")

fms.sidebar.pack(side=LEFT, fill=Y)

fms.profilepage = Button(fms.sidebar,text="VIEW PROFILE", command=lambda:fms.Managerdetails(manager\_id), bg="#f39c12",cursor="hand2", fg="white", font=("cooper black",14))

fms.profilepage.grid(row=4, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewdonations= Button(fms.sidebar,text="VIEW DONATIONS", command=lambda:fms.Mviewdonations(manager\_id), bg="#0b1335",cursor="hand2", fg="white", font=("cooper black",14))

fms.viewdonations.grid(row=8, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewaccepteddonations= Button(fms.sidebar,text="ACCEPTED DONATIONS", command=lambda:fms.Mview\_accepted\_donations(manager\_id), bg="#0b1335",cursor="hand2", fg="white", font=("cooper black",14))

fms.viewaccepteddonations.grid(row=12, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

def show\_custom\_messages(message):

# Create a new top-level window

message\_dialog = Toplevel()

message\_dialog.title("Message")

# Set window size and position it in the center of the screen

window\_width = 350

window\_height = 150

screen\_width = message\_dialog.winfo\_screenwidth()

screen\_height = message\_dialog.winfo\_screenheight()

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

message\_dialog.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Add an icon label in #32CD32 color

style = ttk.Style()

style.configure("Error.TLabel", foreground="#32CD32")

error\_icon = ttk.Label(message\_dialog, text="\u24D8", style="Error.TLabel", font=("Cooper Black", 24))

error\_icon.pack(pady=10)

label = Label(message\_dialog, text=message, fg="#32CD32", font=("Cooper Black", 12))

label.pack(padx=10, pady=0)

# Add an "OK" button to close the dialog

ok\_button = Button(message\_dialog, text="OK", command=message\_dialog.destroy, width=10, font=("Cooper Black", 12))

ok\_button.pack(pady=10)

def show\_custom\_error(message):

# Create a new top-level window

error\_dialog = Toplevel()

error\_dialog.title("Error Message")

# Set window size and position it in the center of the screen

window\_width = 650

window\_height = 150

screen\_width = error\_dialog.winfo\_screenwidth()

screen\_height = error\_dialog.winfo\_screenheight()

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

error\_dialog.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Add an icon label in red color

style = ttk.Style()

style.configure("Error.TLabel", foreground="red")

error\_icon = ttk.Label(error\_dialog, text="\u26A0", style="Error.TLabel", font=("Cooper Black", 24))

error\_icon.pack(pady=10)

label = Label(error\_dialog, text=message, fg="red", font=("Cooper Black", 12))

label.pack(padx=10, pady=0)

# Add an "OK" button to close the dialog

ok\_button = Button(error\_dialog, text="OK", command=error\_dialog.destroy, width=10, font=("Cooper Black", 12))

ok\_button.pack(pady=10)

# Function to load data into the Treeview

def manager\_data():

# Clear the existing data in the Treeview

tree.delete(\*tree.get\_children())

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

# Retrieve the manager's data from the database

cursor.execute("SELECT manager\_id, first\_name, last\_name, user\_name, department, email FROM managers WHERE manager\_id = ?", (manager\_id,))

manager = cursor.fetchone()

if manager:

# Insert the data into the Treeview

tree.insert("", END, values=(manager[0], manager[1], manager[2], manager[3], manager[4], manager[5]), tags=("my\_font"))

# Close the connection

conn.close()

# Function to edit a manager's details

def edit\_manager():

# Get the selected item from the Treeview

selected\_item = tree.focus()

if not selected\_item:

show\_custom\_error("Please select manager to edit")

return

# Get the values of the selected record

values = tree.item(selected\_item, "values")

manager\_id = values[0]

def submit():

# Get values from entry widgets

first\_name = first\_name\_entry.get()

last\_name = last\_name\_entry.get()

user\_name = user\_name\_entry.get()

department = selected\_department.get()

email = email\_entry.get()

# Validate fields

if not (first\_name and last\_name and user\_name and department and email):

show\_custom\_error("All fields are required")

return

# Validate email format

if not re.match(r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$', email):

show\_custom\_error("Invalid email format")

return

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

try:

# Check if the user\_name or email already exists for other managers

cursor.execute("SELECT COUNT(\*) FROM managers WHERE (user\_name=? OR email=?) AND manager\_id != ?", (user\_name, email, manager\_id))

existing\_managers\_count = cursor.fetchone()[0]

if existing\_managers\_count > 0:

# User\_name or email already exists for other managers, show error message

show\_custom\_error("User name or email already exists. Please choose a different one.")

else:

# Proceed with updating the manager details

# Print the SQL query to verify correctness

print(f"Executing EDIT query: EDIT FROM managers WHERE manager\_id = {manager\_id}")

# Update manager in the managers table

cursor.execute("UPDATE managers SET first\_name=?, last\_name=?, user\_name=?, department=?, email=? WHERE manager\_id=?",

(first\_name, last\_name, user\_name, department, email, manager\_id))

# Commit the changes

conn.commit()

# Show success message

show\_custom\_messages("Manager details updated successfully")

# Refresh the Treeview

manager\_data()

except Exception as e:

# Show error message if update fails

show\_custom\_error(f"Failed to update the manager details: {str(e)}")

finally:

# Close the connection

conn.close()

# Close the edit window

edit\_window.destroy()

# Create a Toplevel window for editing a manager

edit\_window = Toplevel()

edit\_window.title("Edit Profile")

# Calculate the center coordinates of the screen

screen\_width = edit\_window.winfo\_screenwidth()

screen\_height = edit\_window.winfo\_screenheight()

window\_width = 700

window\_height = 225

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

# Set the geometry of the window to be centered and larger

edit\_window.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Create labels and entry widgets for each field

Label(edit\_window, text="First Name:", font=("Cooper Black", 12)).grid(row=0, column=0, padx=10, pady=10, sticky="e")

first\_name\_entry = Entry(edit\_window, font=("Cooper Black", 12))

first\_name\_entry.grid(row=0, column=1, padx=10, pady=10)

Label(edit\_window, text="Last Name:", font=("Cooper Black", 12)).grid(row=1, column=0, padx=10, pady=10, sticky="e")

last\_name\_entry = Entry(edit\_window, font=("Cooper Black", 12))

last\_name\_entry.grid(row=1, column=1, padx=10, pady=10)

Label(edit\_window, text="User Name:", font=("Cooper Black", 12)).grid(row=0, column=2, padx=10, pady=10, sticky="e")

user\_name\_entry = Entry(edit\_window, font=("Cooper Black", 12))

user\_name\_entry.grid(row=0, column=3, padx=10, pady=10)

# user\_name\_entry.insert(0, values[3]) # Insert the user\_name from the selected record

Label(edit\_window, text="Email:", font=("Cooper Black", 12)).grid(row=3, column=0, padx=10, pady=10, sticky="e")

email\_entry = Entry(edit\_window, font=("Cooper Black", 12))

email\_entry.grid(row=3, column=1, padx=10, pady=10)

# email\_entry.insert(0, values[5]) # Insert the email from the selected record

Label(edit\_window, text="Department:", font=("Cooper Black", 12)).grid(row=1, column=2, padx=10, pady=10, sticky="e")

department\_options = ["One Central", "Eatry", "Merill", "Dine and Connect", "Catering"]

selected\_department = StringVar(edit\_window)

selected\_department.set(department\_options[0])

department\_dropdown = OptionMenu(edit\_window, selected\_department, \*department\_options)

department\_dropdown.config(font=("Cooper Black", 12))

department\_dropdown.grid(row=1, column=3, padx=10, pady=10, sticky="w")

# Populate the entry widgets with the selected manager's data

first\_name\_entry.insert(0, values[1])

last\_name\_entry.insert(0, values[2])

user\_name\_entry.insert(0, values[3])

selected\_department.set(values[4])

email\_entry.insert(0, values[5])

# Create a Submit button

submit\_button = Button(edit\_window, text="Submit", command=submit, font=("Cooper Black", 12))

submit\_button.grid(row=4, columnspan=4, pady=10)

# Main loop for the edit window

edit\_window.mainloop()

# Create the main content frame

content\_frame = Frame(fms.scr, bg="lightgray")

content\_frame.pack(side=LEFT, fill=BOTH, expand=True)

# Create a label for the title

title\_label = Label(content\_frame, text="VIEW PROFILE", font=("Cooper Black", 16), bg="#0b1335", fg="white", pady=10)

title\_label.pack(fill=X)

title\_label.pack(pady=(0, 85))

# Create treeview

tree = ttk.Treeview(content\_frame, columns=("Manager ID", "First Name", "Last Name", "User Name", "Department", "Email"), show="headings", padding=(0, 5))

# Set the font for the headings

tree.heading("Manager ID", text="Manager ID", anchor=CENTER)

tree.heading("First Name", text="First Name", anchor=CENTER)

tree.heading("Last Name", text="Last Name", anchor=CENTER)

tree.heading("User Name", text="User Name", anchor=CENTER)

tree.heading("Department", text="Department", anchor=CENTER)

tree.heading("Email", text="Email", anchor=CENTER)

# Configure Treeview style

style = ttk.Style()

style.configure("Custom.Treeview", font=("Cooper Black", 12))

style.configure("Treeview.Row", padding=(0, 5))

# Set the background color for the headers

style.configure("Treeview.Heading", background="gray")

# Define a tag named "my\_font" with the desired font

tree.tag\_configure("my\_font", font=("Cooper Black", 12))

tree.pack()

# Connect to the SQLite3 database

conn = connect('fms.db')

cursor = conn.cursor()

# Retrieve the manager's data from the database

cursor.execute("SELECT manager\_id, first\_name, last\_name, user\_name, department, email FROM managers WHERE manager\_id = ?", (manager\_id,))

manager = cursor.fetchone()

if manager:

# Insert the manager's data into the Treeview

tree.insert("", END, values=(manager[0], manager[1], manager[2], manager[3], manager[4], manager[5]))

# Apply the "my\_font" tag to all items in the treeview

for manager in tree.get\_children():

tree.item(manager, tags=("my\_font",))

edit\_button = Button(content\_frame, text="EDIT PROFILE", command=edit\_manager, bg="#3498db", fg="white", font=("Cooper Black", 12))

edit\_button.pack(side=LEFT, padx=500, pady=10)

fms.scr.mainloop()

#---------------------------------------------------Manager Pending Donations Page----------------------------------------------------------------#

def Mviewdonations(fms, manager\_id):

fms.scr.destroy()

fms.scr = Tk()

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.managerdashf1= Frame(fms.scr, bg="#ffffff")

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img1 = fms.logo.subsample(2,2)

fms.logo\_banner1 = Label(fms.managerdashf1, image=fms.logo\_img1)

fms.logo\_banner1.pack(side=LEFT)

fms.managerlabel= Label(text="Weclome to Manager Dashboard!!", bg="#0b1335", fg="white",font=("cooper black",14))

fms.managerlabel.place(x=1000,y=50)

fms.logout=Button(fms.managerdashf1,text="LOGOUT", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.logout.place(x=1075,y=100)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.managerdashf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.managerdashf1.pack(fill=BOTH)

fms.sidebar = Frame(fms.scr, bg="#d3ede6")

fms.sidebar.pack(side=LEFT, fill=Y)

# Add buttons to the sidebar

fms.profilepage = Button(fms.sidebar,text="VIEW PROFILE", command=lambda:fms.Managerdetails(manager\_id), bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.profilepage.grid(row=4, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewdonations= Button(fms.sidebar,text="VIEW DONATIONS", command=lambda:fms.Mviewdonations(manager\_id), bg="#f39c12",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewdonations.grid(row=8, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewaccepteddonations= Button(fms.sidebar,text="ACCEPTED DONATIONS", command=lambda:fms.Mview\_accepted\_donations(manager\_id), bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewaccepteddonations.grid(row=12, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

def show\_custom\_messages(message):

# Create a new top-level window

message\_dialog = Toplevel()

message\_dialog.title("Message")

# Set window size and position it in the center of the screen

window\_width = 350

window\_height = 150

screen\_width = message\_dialog.winfo\_screenwidth()

screen\_height = message\_dialog.winfo\_screenheight()

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

message\_dialog.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Add an icon label in #32CD32 color

style = ttk.Style()

style.configure("Error.TLabel", foreground="#32CD32")

error\_icon = ttk.Label(message\_dialog, text="\u24D8", style="Error.TLabel", font=("Cooper Black", 24))

error\_icon.pack(pady=10)

label = Label(message\_dialog, text=message, fg="#32CD32", font=("Cooper Black", 12))

label.pack(padx=10, pady=0)

# Add an "OK" button to close the dialog

ok\_button = Button(message\_dialog, text="OK", command=message\_dialog.destroy, width=10, font=("Cooper Black", 12))

ok\_button.pack(pady=10)

def show\_custom\_error(message):

# Create a new top-level window

error\_dialog = Toplevel()

error\_dialog.title("Error Message")

# Set window size and position it in the center of the screen

window\_width = 650

window\_height = 150

screen\_width = error\_dialog.winfo\_screenwidth()

screen\_height = error\_dialog.winfo\_screenheight()

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

error\_dialog.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Add an icon label in red color

style = ttk.Style()

style.configure("Error.TLabel", foreground="red")

error\_icon = ttk.Label(error\_dialog, text="\u26A0", style="Error.TLabel", font=("Cooper Black", 24))

error\_icon.pack(pady=10)

label = Label(error\_dialog, text=message, fg="red", font=("Cooper Black", 12))

label.pack(padx=10, pady=0)

# Add an "OK" button to close the dialog

ok\_button = Button(error\_dialog, text="OK", command=error\_dialog.destroy, width=10, font=("Cooper Black", 12))

ok\_button.pack(pady=10)

# Function to load data into the Treeview

def donation\_data():

#Clear the existing data in the Treeview

tree.delete(\*tree.get\_children())

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

# Retrieve the donation data from the database

cursor.execute("SELECT donation\_id, item\_name, item\_type, calories, amount\_lb, servings, manager\_id, donation\_status, user\_id FROM donation\_details WHERE manager\_id = ? and donation\_status = 'pending'", (manager\_id,))

donations = cursor.fetchall()

# Insert the data into the Treeview

for donation in donations:

# Fetch the manager's user\_name from the managers table

cursor.execute("SELECT user\_name FROM managers WHERE manager\_id=?", (donation[6],))

manager\_result = cursor.fetchone()

manager\_name = manager\_result[0] if manager\_result else None # Assuming manager\_id is at index 7

# Fetch the user's user\_name from the users table

cursor.execute("SELECT user\_name FROM users WHERE user\_id=?", (donation[8],))

user\_result = cursor.fetchone()

user\_name = user\_result[0] if user\_result else None # Assuming user\_id is at index 9

# Exclude the manager\_id (index 0) and donation\_status (index 7) from the display

tree.insert("", "end", values=(donation[0], donation[1], donation[2], donation[3], donation[4], donation[5], manager\_name, donation[7], user\_name), tags=("my\_font"))

# Close the connection

conn.close()

# Function to send an email notification

def send\_email(recipient, user\_name, item\_name, item\_type, calories, amount\_lb, servings):

# Validate recipient email address

if not re.match(r"[^@]+@[^@]+\.[^@]+", recipient):

print(f"Invalid email address: {recipient}")

return

sender\_email = "fms38865@gmail.com"

sender\_password = "ktee vlno ediy uiop"

subject = "Uploaded Food Donation Updated"

body = f"Hello {user\_name},\n\nA food donation that was previously uploaded has been updated with the following details:\nItem Name: {item\_name}\nItem Type: {item\_type}\nCalories: {calories}\nAmount (lb): {amount\_lb}\nServings: {servings}\n\nCheck it out!\n\nThank you,\nFood Management Team"

# Compose email

message = MIMEMultipart()

message['From'] = sender\_email

message['To'] = recipient

message['Subject'] = subject

message.attach(MIMEText(body, 'plain'))

# Send email

try:

server = smtplib.SMTP('smtp.gmail.com', 587)

server.starttls()

server.login(sender\_email, sender\_password)

server.sendmail(sender\_email, recipient, message.as\_string())

server.quit()

print("Email sent successfully")

except Exception as e:

print(f"Failed to send email: {str(e)}")

def edit\_donation():

# Get the selected item from the Treeview

selected\_item = tree.focus()

if not selected\_item:

show\_custom\_error("Please select a donation to edit")

return

# Get the values of the selected record

values = tree.item(selected\_item, "values")

donation\_id = values[0]

donation\_status = values[7] # Assuming donation\_status is at index 8

# Check if the donation has already been accepted

if donation\_status == "accepted":

show\_custom\_error("Donation already accepted. You cannot edit it.")

return

# Define a function to submit the edited donation details

def submit():

# Get values from entry widgets

# item\_no = item\_no\_entry.get()

item\_name = item\_name\_entry.get()

item\_type = item\_type\_entry.get()

calories = calories\_entry.get()

amount\_lb = amount\_lb\_entry.get()

servings = servings\_entry.get()

# manager\_id =

# Validate fields

if not (item\_name and item\_type and calories and amount\_lb and servings):

show\_custom\_error("All fields are required")

return

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

# Initialize a flag to track whether the update was successful

update\_successful = False

try:

# Validate fields

if not ( item\_name and item\_type and calories and amount\_lb and servings):

show\_custom\_error("All fields are required")

elif not (calories.isdigit() and amount\_lb.isdigit() and servings.isdigit()):

show\_custom\_error("Please enter only digits for Calories, Amount (lb), and Servings.")

else:

# Update the donation details if donation\_status is not "accepted"

if donation\_status != "accepted":

# Print the SQL query to verify correctness

print(f"Executing EDIT query: EDIT FROM donation\_details WHERE donation\_id = {donation\_id}")

# Update donation details in the donation\_details table

cursor.execute("UPDATE donation\_details SET item\_name=?, item\_type=?, calories=?, amount\_lb=?, servings=? WHERE donation\_id=?",

(item\_name, item\_type, calories, amount\_lb, servings, donation\_id))

# Commit the changes

conn.commit()

# Set update flag to True

update\_successful = True

# Show success message

show\_custom\_messages("Donation details updated successfully")

# Fetch user\_name and email from the users table

cursor.execute('SELECT user\_name, email FROM users')

user\_data = cursor.fetchall()

# Iterate through the result and send emails

for user\_name, email in user\_data:

send\_email(email, user\_name, item\_name, item\_type, calories, amount\_lb, servings)

# Refresh the Treeview

donation\_data()

else:

show\_custom\_error("Donation already accepted. You cannot edit it.")

except Exception as e:

# Show error message if update fails

show\_custom\_error(f"Failed to update the donation details: {str(e)}")

finally:

# Close the connection

conn.close()

# Close the edit window only if the update was successful

if update\_successful:

edit\_window.destroy()

# Create a Toplevel window for editing a manager

edit\_window = Toplevel()

edit\_window.title("Edit Donation")

# Calculate the center coordinates of the screen

screen\_width = edit\_window.winfo\_screenwidth()

screen\_height = edit\_window.winfo\_screenheight()

window\_width = 725

window\_height = 225

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

# Set the geometry of the window to be centered and larger

edit\_window.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Create labels and entry widgets for each field

# Label(edit\_window, text="Item No:", font=("Cooper Black", 12)).grid(row=0, column=0, padx=10, pady=10, sticky="e")

# item\_no\_entry = Entry(edit\_window, font=("Cooper Black", 12))

# item\_no\_entry.grid(row=0, column=1, padx=10, pady=10)

Label(edit\_window, text="Item Name:", font=("Cooper Black", 12)).grid(row=0, column=0, padx=10, pady=10, sticky="e")

item\_name\_entry = Entry(edit\_window, font=("Cooper Black", 12))

item\_name\_entry.grid(row=0, column=1, padx=10, pady=10)

Label(edit\_window, text="Item Type:", font=("Cooper Black", 12)).grid(row=1, column=0, padx=10, pady=10, sticky="e")

item\_type\_entry = Entry(edit\_window, font=("Cooper Black", 12))

item\_type\_entry.grid(row=1, column=1, padx=10, pady=10)

Label(edit\_window, text="Calories:", font=("Cooper Black", 12)).grid(row=1, column=2, padx=10, pady=10, sticky="e")

calories\_entry = Entry(edit\_window, font=("Cooper Black", 12))

calories\_entry.grid(row=1, column=3, padx=10, pady=10)

Label(edit\_window, text="Amount(Lb):", font=("Cooper Black", 12)).grid(row=0, column=2, padx=10, pady=10, sticky="e")

amount\_lb\_entry = Entry(edit\_window, font=("Cooper Black", 12))

amount\_lb\_entry.grid(row=0, column=3, padx=10, pady=10)

Label(edit\_window, text="Servings:", font=("Cooper Black", 12)).grid(row=2, column=0, padx=10, pady=10, sticky="e")

servings\_entry = Entry(edit\_window, font=("Cooper Black", 12))

servings\_entry.grid(row=2, column=1, padx=10, pady=10)

# Populate the entry widgets with the selected manager's data

# item\_no\_entry.insert(0, values[1])

item\_name\_entry.insert(0, values[1])

item\_type\_entry.insert(0, values[2])

calories\_entry.insert(0, values[3])

amount\_lb\_entry.insert(0, values[4])

servings\_entry.insert(0, values[5])

# Create a Submit button

submit\_button = Button(edit\_window, text="Submit", command=submit, font=("Cooper Black", 12))

submit\_button.grid(row=4, columnspan=4, pady=10)

# Main loop for the edit window

edit\_window.mainloop()

def delete\_donation():

# Get the selected item from the Treeview

selected\_item = tree.focus()

if not selected\_item:

show\_custom\_error("Please select a donation to delete.")

return

# Get the values of the selected record

values = tree.item(selected\_item, "values")

donation\_id = values[0]

donation\_status = values[7] # Assuming donation\_status is at index 8

# Check if the donation has already been accepted

if donation\_status == "accepted":

show\_custom\_error("Donation already accepted. You cannot delete it.")

return

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

try:

# Print the SQL query to verify correctness

print(f"Executing DELETE query: DELETE FROM donation\_details WHERE donation\_id = {donation\_id}")

# Delete the record from the donation\_details table

cursor.execute("DELETE FROM donation\_details WHERE donation\_id = ?", (donation\_id,))

# Commit the changes

conn.commit()

# Show success message

show\_custom\_messages("Donation deleted successfully")

# Refresh the Treeview

donation\_data()

except Exception as e:

# Show error message if deletion fails

show\_custom\_error(f"Failed to delete donation: {str(e)}")

finally:

# Close the connection

conn.close()

# Function to send an email notification

def send\_mail(recipient, user\_name, item\_name, item\_type, calories, amount\_lb, servings):

# Validate recipient email address

if not re.match(r"[^@]+@[^@]+\.[^@]+", recipient):

print(f"Invalid email address: {recipient}")

return

# Email configuration

sender\_email = "fms38865@gmail.com"

sender\_password = "ktee vlno ediy uiop"

subject = "New Food Donation Uploaded"

body = f"Hello {user\_name},\n\nA new food donation has been added with the following details:\nItem Name: {item\_name}\nItem Type: {item\_type}\nCalories: {calories}\nAmount (lb): {amount\_lb}\nServings: {servings}\n\nCheck it out!\n\nThank you,\nFood Management Team"

# Compose email

message = MIMEMultipart()

message['From'] = sender\_email

message['To'] = recipient

message['Subject'] = subject

message.attach(MIMEText(body, 'plain'))

# Send email

try:

server = smtplib.SMTP('smtp.gmail.com', 587)

server.starttls()

server.login(sender\_email, sender\_password)

server.sendmail(sender\_email, recipient, message.as\_string())

server.quit()

print("Email sent successfully")

except Exception as e:

print(f"Failed to send email: {str(e)}")

def create\_donation\_details\_table\_if\_not\_exists():

conn = connect('fms.db')

cursor = conn.cursor()

# SQL query to create the table if it does not exist

create\_table\_query = '''

CREATE TABLE IF NOT EXISTS donation\_details (

donation\_id INTEGER PRIMARY KEY AUTOINCREMENT,

item\_name TEXT NOT NULL,

item\_type TEXT NOT NULL,

calories INTEGER NOT NULL,

amount\_lb INTEGER NOT NULL,

servings INTEGER NOT NULL,

manager\_id INTEGER NOT NULL,

FOREIGN KEY (manager\_id) REFERENCES managers(id)

);

'''

cursor.execute(create\_table\_query)

conn.commit()

conn.close()

# Function to add a new manager

def add\_donation():

def submit():

# Get values from entry widgets

# item\_no = item\_no\_entry.get()

item\_name = item\_name\_entry.get()

item\_type = item\_type\_entry.get()

calories = calories\_entry.get()

amount\_lb = amount\_lb\_entry.get()

servings = servings\_entry.get()

# Initialize a flag to track insertion success

insertion\_successful = False

# Validate fields

if not ( item\_name and item\_type and calories and amount\_lb and servings):

show\_custom\_error("All fields are required")

else:

try:

create\_donation\_details\_table\_if\_not\_exists() # Check and create table if not exists

conn = connect('fms.db')

cursor = conn.cursor()

# Validate inputs

if not ( calories.isdigit() and amount\_lb.isdigit() and servings.isdigit()):

show\_custom\_error("Please enter only digits for Calories, Amount (lb), and Servings.")

else:

# Insert new manager into the managers table

cursor.execute("INSERT INTO donation\_details (item\_name, item\_type, calories, amount\_lb, servings, manager\_id) VALUES (?, ?, ?, ?, ?, ?)",

(item\_name, item\_type, calories, amount\_lb, servings, manager\_id))

# Commit the changes

conn.commit()

# Set insertion flag to True

insertion\_successful = True

# Show success message

show\_custom\_messages("Donation added successfully")

# Fetch user\_name and email from the users table

cursor.execute('SELECT user\_name, email FROM users')

user\_data = cursor.fetchall()

# Iterate through the result and send emails

for user\_name, email in user\_data:

send\_mail(email, user\_name, item\_name, item\_type, calories, amount\_lb, servings)

# Refresh the Treeview

donation\_data()

except Exception as e:

# Show error message if insertion fails

show\_custom\_error(f"Failed to add donation: {e}")

finally:

# Close the connection

conn.close()

# Close the add window only if the insertion was successful

if insertion\_successful:

add\_window.destroy()

# Create a Toplevel window for adding a new donation

add\_window = Toplevel()

add\_window.title("Add Donation")

# Calculate the center coordinates of the screen

screen\_width = add\_window.winfo\_screenwidth()

screen\_height = add\_window.winfo\_screenheight()

window\_width = 750

window\_height = 225

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

# Set the geometry of the window to be centered and larger

add\_window.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Create labels and entry widgets for each field

# Label(add\_window, text="Item No:", font=("Cooper Black", 12)).grid(row=0, column=0, padx=10, pady=10, sticky="e")

# item\_no\_entry = Entry(add\_window, font=("Cooper Black", 12))

# item\_no\_entry.grid(row=0, column=1, padx=10, pady=10)

Label(add\_window, text="Item Name:", font=("Cooper Black", 12)).grid(row=0, column=0, padx=10, pady=10, sticky="e")

item\_name\_entry = Entry(add\_window, font=("Cooper Black", 12))

item\_name\_entry.grid(row=0, column=1, padx=10, pady=10)

Label(add\_window, text="Amount(Lb):", font=("Cooper Black", 12)).grid(row=0, column=2, padx=10, pady=10, sticky="e")

amount\_lb\_entry = Entry(add\_window, font=("Cooper Black", 12))

amount\_lb\_entry.grid(row=0, column=3, padx=10, pady=10)

Label(add\_window, text="Item Type:", font=("Cooper Black", 12)).grid(row=1, column=0, padx=10, pady=10, sticky="e")

item\_type\_entry = Entry(add\_window, font=("Cooper Black", 12))

item\_type\_entry.grid(row=1, column=1, padx=10, pady=10)

Label(add\_window, text="Calories:", font=("Cooper Black", 12)).grid(row=1, column=2, padx=10, pady=10, sticky="e")

calories\_entry = Entry(add\_window, font=("Cooper Black", 12))

calories\_entry.grid(row=1, column=3, padx=10, pady=10)

Label(add\_window, text="Servings:", font=("Cooper Black", 12)).grid(row=2, column=0, padx=10, pady=10, sticky="e")

servings\_entry = Entry(add\_window, font=("Cooper Black", 12))

servings\_entry.grid(row=2, column=1, padx=10, pady=10)

# Create a Submit button

submit\_button = Button(add\_window, text="Submit", command=submit, font=("Cooper Black", 12))

submit\_button.grid(row=3, columnspan=4, pady=10)

# Main loop for the add window

add\_window.mainloop()

# Create the main content frame

content\_frame = Frame(fms.scr, bg="lightgray")

content\_frame.pack(side=LEFT, fill=BOTH, expand=True)

# Create a label for the title

title\_label = Label(content\_frame, text="VIEW DONATIONS", font=("Cooper Black", 16), bg="#0b1335", fg="white", pady=10)

title\_label.pack(fill=X)

title\_label.pack(pady=(0, 85))

# Create treeview

tree = ttk.Treeview(content\_frame, columns=("Donation ID", "Item Name", "Item Type", "Calories", "Amount(Lbs)", "Servings", "Added By", "Status", "Accepted By"), show="headings", padding=(0, 5))

# Set the font for the headings

tree.heading("Donation ID", text="Donation ID", anchor=CENTER)

tree.heading("Item Name", text="Item Name", anchor=CENTER)

tree.heading("Item Type", text="Item Type", anchor=CENTER)

tree.heading("Calories", text="Calories", anchor=CENTER)

tree.heading("Amount(Lbs)", text="Amount(Lbs)", anchor=CENTER)

tree.heading("Servings", text="Servings", anchor=CENTER)

tree.heading("Added By", text="Added By", anchor=CENTER)

tree.heading("Status", text="Status", anchor=CENTER)

tree.heading("Accepted By", text="Accepted By", anchor=CENTER)

# Configure Treeview style

style = ttk.Style()

style.configure("Custom.Treeview", font=("Cooper Black", 12))

style.configure("Treeview.Row", padding=(0, 5))

# Set the background color for the headers

style.configure("Treeview.Heading", background="gray")

# Define a tag named "my\_font" with the desired font

tree.tag\_configure("my\_font", font=("Cooper Black", 12))

# Set the column widths

tree.column("Donation ID", width=100)

tree.column("Item Name", width=170)

tree.column("Item Type", width=135)

tree.column("Calories", width=135)

tree.column("Amount(Lbs)", width=135)

tree.column("Servings", width=135)

tree.column("Added By", width=135)

tree.column("Status", width=135)

tree.column("Accepted By", width=135)

tree.pack()

# Connect to the SQLite3 database

conn = connect('fms.db')

cursor = conn.cursor()

# Fetch and display data

cursor.execute("SELECT donation\_id, item\_name, item\_type, calories, amount\_lb, servings, manager\_id, donation\_status, user\_id FROM donation\_details WHERE manager\_id = ? and donation\_status = 'pending'", (manager\_id,))

donations = cursor.fetchall()

for donation in donations:

# Fetch the manager's user\_name from the managers table

cursor.execute("SELECT user\_name FROM managers WHERE manager\_id=?", (donation[6],))

manager\_result = cursor.fetchone()

manager\_name = manager\_result[0] if manager\_result else None # Assuming manager\_id is at index 6

# Fetch the user's user\_name from the users table

cursor.execute("SELECT user\_name FROM users WHERE user\_id=?", (donation[8],))

user\_result = cursor.fetchone()

user\_name = user\_result[0] if user\_result else None # Assuming user\_id is at index 8

tree.insert("", END, values=(donation[0], donation[1], donation[2], donation[3], donation[4], donation[5], manager\_name, donation[7], user\_name))

# Apply the "my\_font" tag to all items in the treeview

for donation in tree.get\_children():

tree.item(donation, tags=("my\_font",))

# Buttons for CRUD operations

add\_button = Button(content\_frame, text="ADD DONATION", command=add\_donation, bg="#f39c12", fg="white", font=("Cooper Black", 12))

add\_button.pack(side=LEFT, padx=30, pady=10)

edit\_button = Button(content\_frame, text="EDIT DONATION", command=edit\_donation, bg="#3498db", fg="white", font=("Cooper Black", 12))

edit\_button.pack(side=LEFT, padx=320, pady=10)

delete\_button = Button(content\_frame, text="DELETE DONATION", command=delete\_donation, bg="#e74c3c", fg="white", font=("Cooper Black", 12))

delete\_button.pack(side=LEFT, padx=30, pady=10)

conn.close()

fms.scr.mainloop()

#---------------------------------------------------Manager Accepted Donations Page---------------------------------------------------------------#

def Mview\_accepted\_donations(fms, manager\_id):

fms.scr.destroy()

fms.scr = Tk()

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.managerdashf1= Frame(fms.scr, bg="#ffffff")

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img1 = fms.logo.subsample(2,2)

fms.logo\_banner1 = Label(fms.managerdashf1, image=fms.logo\_img1)

fms.logo\_banner1.pack(side=LEFT)

fms.managerlabel= Label(text="Weclome to Manager Dashboard!!", bg="#0b1335", fg="white",font=("cooper black",14))

fms.managerlabel.place(x=1000,y=50)

fms.logout=Button(fms.managerdashf1,text="LOGOUT", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.logout.place(x=1075,y=100)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.managerdashf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.managerdashf1.pack(fill=BOTH)

fms.sidebar = Frame(fms.scr, bg="#d3ede6")

fms.sidebar.pack(side=LEFT, fill=Y)

# Add buttons to the sidebar

fms.profilepage = Button(fms.sidebar,text="VIEW PROFILE", command=lambda:fms.Managerdetails(manager\_id), bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.profilepage.grid(row=4, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewdonations= Button(fms.sidebar,text="VIEW DONATIONS", command=lambda:fms.Mviewdonations(manager\_id), bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewdonations.grid(row=8, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewaccepteddonations= Button(fms.sidebar,text="ACCEPTED DONATIONS", command=lambda:fms.Mview\_accepted\_donations(manager\_id), bg="#f39c12",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewaccepteddonations.grid(row=12, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

# Create the main content frame

content\_frame = Frame(fms.scr, bg="lightgray")

content\_frame.pack(side=LEFT, fill=BOTH, expand=True)

# Create a label for the title

title\_label = Label(content\_frame, text="VIEW ACCEPTED DONATIONS", font=("Cooper Black", 16), bg="#0b1335", fg="white", pady=10)

title\_label.pack(fill=X)

title\_label.pack(pady=(0, 85))

# Create treeview

tree = ttk.Treeview(content\_frame, columns=("Donation ID", "Item Name", "Item Type", "Calories", "Amount(Lbs)", "Servings", "Added By", "Status", "Accepted By"), show="headings", padding=(0, 5))

# Set the font for the headings

tree.heading("Donation ID", text="Donation ID", anchor=CENTER)

tree.heading("Item Name", text="Item Name", anchor=CENTER)

tree.heading("Item Type", text="Item Type", anchor=CENTER)

tree.heading("Calories", text="Calories", anchor=CENTER)

tree.heading("Amount(Lbs)", text="Amount(Lbs)", anchor=CENTER)

tree.heading("Servings", text="Servings", anchor=CENTER)

tree.heading("Added By", text="Added By", anchor=CENTER)

tree.heading("Status", text="Status", anchor=CENTER)

tree.heading("Accepted By", text="Accepted By", anchor=CENTER)

# Configure Treeview style

style = ttk.Style()

style.configure("Custom.Treeview", font=("Cooper Black", 12))

style.configure("Treeview.Row", padding=(0, 5))

# Set the background color for the headers

style.configure("Treeview.Heading", background="gray")

# Define a tag named "my\_font" with the desired font

tree.tag\_configure("my\_font", font=("Cooper Black", 12))

# Set the column widths

tree.column("Donation ID", width=100)

tree.column("Item Name", width=170)

tree.column("Item Type", width=135)

tree.column("Calories", width=135)

tree.column("Amount(Lbs)", width=135)

tree.column("Servings", width=135)

tree.column("Added By", width=135)

tree.column("Status", width=135)

tree.column("Accepted By", width=135)

tree.pack()

# Connect to the SQLite3 database

conn = connect('fms.db')

cursor = conn.cursor()

# Fetch and display data

cursor.execute("SELECT donation\_id, item\_name, item\_type, calories, amount\_lb, servings, manager\_id, donation\_status, user\_id FROM donation\_details WHERE manager\_id = ? and donation\_status = 'accepted'", (manager\_id,))

donations = cursor.fetchall()

for donation in donations:

# Fetch the manager's user\_name from the managers table

cursor.execute("SELECT user\_name FROM managers WHERE manager\_id=?", (donation[6],))

manager\_result = cursor.fetchone()

manager\_name = manager\_result[0] if manager\_result else None # Assuming manager\_id is at index 6

# Fetch the user's user\_name from the users table

cursor.execute("SELECT user\_name FROM users WHERE user\_id=?", (donation[8],))

user\_result = cursor.fetchone()

user\_name = user\_result[0] if user\_result else None # Assuming user\_id is at index 8

tree.insert("", END, values=(donation[0], donation[1], donation[2], donation[3], donation[4], donation[5], manager\_name, donation[7], user\_name))

# Apply the "my\_font" tag to all items in the treeview

for donation in tree.get\_children():

tree.item(donation, tags=("my\_font",))

# Buttons for CRUD operations

# add\_button = Button(content\_frame, text="ADD DONATION", command=add\_donation, bg="#f39c12", fg="white", font=("Cooper Black", 12))

# add\_button.pack(side=LEFT, padx=30, pady=10)

# edit\_button = Button(content\_frame, text="EDIT DONATION", command=edit\_donation, bg="#3498db", fg="white", font=("Cooper Black", 12))

# edit\_button.pack(side=LEFT, padx=320, pady=10)

# delete\_button = Button(content\_frame, text="DELETE DONATION", command=delete\_donation, bg="#e74c3c", fg="white", font=("Cooper Black", 12))

# delete\_button.pack(side=LEFT, padx=30, pady=10)

conn.close()

fms.scr.mainloop()

#---------------------------------------------------Manager Dashboard Code End--------------------------------------------------------------------#

#---------------------------------------------------User Dashboard Code Start---------------------------------------------------------------------#

#---------------------------------------------------User Main Page--------------------------------------------------------------------------------#

def Userdetails(fms, user\_id):

fms.scr.destroy()

fms.scr = Tk()

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.managerdashf1= Frame(fms.scr, bg="#ffffff")

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img1 = fms.logo.subsample(2,2)

fms.logo\_banner1 = Label(fms.managerdashf1, image=fms.logo\_img1)

fms.logo\_banner1.pack(side=LEFT)

fms.managerlabel= Label(text="Weclome to User Dashboard!!", bg="#0b1335", fg="white",font=("cooper black",14))

fms.managerlabel.place(x=1000,y=50)

fms.logout=Button(fms.managerdashf1,text="LOGOUT", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.logout.place(x=1075,y=100)

# code to display the local time

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.managerdashf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.managerdashf1.pack(fill=BOTH)

fms.sidebar = Frame(fms.scr, bg="#d3ede6")

fms.sidebar.pack(side=LEFT, fill=Y)

# Add buttons to the sidebar

fms.profile = Button(fms.sidebar,text="VIEW PROFILE", command=lambda:fms.Userdetails(user\_id), bg="#f39c12",cursor="hand2",

fg="white",font=("cooper black",14))

fms.profile.grid(row=3, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewdonations= Button(fms.sidebar,text="PENDING DONATIONS", command=lambda:fms.Userdonations(user\_id), bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewdonations.grid(row=4, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewaccepteddonations= Button(fms.sidebar,text="ACCEPTED DONATIONS", command=lambda:fms.User\_accepted\_donations(user\_id), bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewaccepteddonations.grid(row=8, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

def show\_custom\_messages(message):

# Create a new top-level window

message\_dialog = Toplevel()

message\_dialog.title("Message")

# Set window size and position it in the center of the screen

window\_width = 350

window\_height = 150

screen\_width = message\_dialog.winfo\_screenwidth()

screen\_height = message\_dialog.winfo\_screenheight()

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

message\_dialog.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Add an icon label in #32CD32 color

style = ttk.Style()

style.configure("Error.TLabel", foreground="#32CD32")

error\_icon = ttk.Label(message\_dialog, text="\u24D8", style="Error.TLabel", font=("Cooper Black", 24))

error\_icon.pack(pady=10)

label = Label(message\_dialog, text=message, fg="#32CD32", font=("Cooper Black", 12))

label.pack(padx=10, pady=0)

# Add an "OK" button to close the dialog

ok\_button = Button(message\_dialog, text="OK", command=message\_dialog.destroy, width=10, font=("Cooper Black", 12))

ok\_button.pack(pady=10)

def show\_custom\_error(message):

# Create a new top-level window

error\_dialog = Toplevel()

error\_dialog.title("Error Message")

# Set window size and position it in the center of the screen

window\_width = 600

window\_height = 150

screen\_width = error\_dialog.winfo\_screenwidth()

screen\_height = error\_dialog.winfo\_screenheight()

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

error\_dialog.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Set window icon

# error\_dialog.iconbitmap(r"house.ico")

# Add an icon label in red color

style = ttk.Style()

style.configure("Error.TLabel", foreground="red")

error\_icon = ttk.Label(error\_dialog, text="\u26A0", style="Error.TLabel", font=("Cooper Black", 24))

error\_icon.pack(pady=10)

label = Label(error\_dialog, text=message, fg="red", font=("Cooper Black", 12))

label.pack(padx=10, pady=0)

# Add an "OK" button to close the dialog

ok\_button = Button(error\_dialog, text="OK", command=error\_dialog.destroy, width=10, font=("Cooper Black", 12))

ok\_button.pack(pady=10)

# Function to load data into the Treeview

def user\_data():

# Clear the existing data in the Treeview

tree.delete(\*tree.get\_children())

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

# Retrieve the manager's data from the database

cursor.execute("SELECT user\_id, first\_name, last\_name, user\_name, email, mobile\_no FROM users WHERE user\_id = ?", (user\_id,))

user = cursor.fetchone()

if user:

# Insert the data into the Treeview

tree.insert("", END, values=(user[0], user[1], user[2], user[3], user[4], user[5]))

# Apply the "my\_font" tag to all items in the Treeview

for item in tree.get\_children():

tree.item(item, tags=("my\_font",))

# Close the connection

conn.close()

# Function to edit a manager's details

def edit\_user():

# Get the selected item from the Treeview

selected\_item = tree.focus()

if not selected\_item:

show\_custom\_error("Please select user to edit")

return

# Get the values of the selected record

values = tree.item(selected\_item, "values")

user\_id = values[0]

def submit():

# Get values from entry widgets

first\_name = first\_name\_entry.get()

last\_name = last\_name\_entry.get()

user\_name = user\_name\_entry.get()

email = email\_entry.get()

mobile\_no = mobile\_no\_entry.get()

# Validate fields

if not (first\_name and last\_name and user\_name and email and mobile\_no):

show\_custom\_error("All fields are required")

return

# Validate email format

if not re.match(r'^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$', email):

show\_custom\_error("Invalid email format")

return

# Validate mobile number format (10 digits only)

if not re.match(r'^\d{10}$', mobile\_no):

show\_custom\_error("Invalid mobile number format. It should consist of 10 digits only.")

return

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

try:

# Check if the user\_name or email already exists for other users

cursor.execute("SELECT COUNT(\*) FROM users WHERE (user\_name=? OR email=?) AND user\_id != ?", (user\_name, email, user\_id))

existing\_users\_count = cursor.fetchone()[0]

if existing\_users\_count > 0:

# User\_name or email already exists for other users, show error message

show\_custom\_error("User name or email already exists. Please choose a different one.")

else:

# Proceed with updating the user details

# Print the SQL query to verify correctness

print(f"Executing EDIT query: EDIT FROM users WHERE user\_id = {user\_id}")

# Update user in the users table

cursor.execute("UPDATE users SET first\_name=?, last\_name=?, user\_name=?, email=?, mobile\_no=? WHERE user\_id=?",

(first\_name, last\_name, user\_name, email, mobile\_no, user\_id))

# Commit the changes

conn.commit()

# Show success message

show\_custom\_messages("User details updated successfully")

# Refresh the Treeview

user\_data()

# Close the edit window

edit\_window.destroy()

except Exception as e:

# Show error message if update fails

show\_custom\_error(f"Failed to update user details: {str(e)}")

finally:

# Close the connection

conn.close()

# Create a Toplevel window for editing a user

edit\_window = Toplevel()

edit\_window.title("Edit Profile")

# Calculate the center coordinates of the screen

screen\_width = edit\_window.winfo\_screenwidth()

screen\_height = edit\_window.winfo\_screenheight()

window\_width = 700

window\_height = 225

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

# Set the geometry of the window to be centered and larger

edit\_window.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Create labels and entry widgets for each field

Label(edit\_window, text="First Name:", font=("Cooper Black", 12)).grid(row=0, column=0, padx=10, pady=10, sticky="e")

first\_name\_entry = Entry(edit\_window, font=("Cooper Black", 12))

first\_name\_entry.grid(row=0, column=1, padx=10, pady=10)

Label(edit\_window, text="Last Name:", font=("Cooper Black", 12)).grid(row=1, column=0, padx=10, pady=10, sticky="e")

last\_name\_entry = Entry(edit\_window, font=("Cooper Black", 12))

last\_name\_entry.grid(row=1, column=1, padx=10, pady=10)

Label(edit\_window, text="User Name:", font=("Cooper Black", 12)).grid(row=0, column=2, padx=10, pady=10, sticky="e")

user\_name\_entry = Entry(edit\_window, font=("Cooper Black", 12))

user\_name\_entry.grid(row=0, column=3, padx=10, pady=10)

Label(edit\_window, text="Email:", font=("Cooper Black", 12)).grid(row=3, column=0, padx=10, pady=10, sticky="e")

email\_entry = Entry(edit\_window, font=("Cooper Black", 12))

email\_entry.grid(row=3, column=1, padx=10, pady=10)

Label(edit\_window, text="Mobile No:", font=("Cooper Black", 12)).grid(row=1, column=2, padx=10, pady=10, sticky="e")

mobile\_no\_entry = Entry(edit\_window, font=("Cooper Black", 12))

mobile\_no\_entry.grid(row=1, column=3, padx=10, pady=10)

# Populate the entry widgets with the selected manager's data

first\_name\_entry.insert(0, values[1])

last\_name\_entry.insert(0, values[2])

user\_name\_entry.insert(0, values[3])

email\_entry.insert(0, values[4])

mobile\_no\_entry.insert(0, values[5])

# Create a Submit button

submit\_button = Button(edit\_window, text="Submit", command=submit, font=("Cooper Black", 12))

submit\_button.grid(row=4, columnspan=4, pady=10)

# Main loop for the edit window

edit\_window.mainloop()

# Create the main content frame

content\_frame = Frame(fms.scr, bg="lightgray")

content\_frame.pack(side=LEFT, fill=BOTH, expand=True)

# Create a label for the title

title\_label = Label(content\_frame, text="VIEW PROFILE", font=("Cooper Black", 16), bg="#0b1335", fg="white", pady=10)

title\_label.pack(fill=X)

title\_label.pack(pady=(0, 85))

# Create treeview

tree = ttk.Treeview(content\_frame, columns=("User ID", "First Name", "Last Name", "User Name", "Email", "Mobile No"), show="headings", padding=(0, 5))

# Set the font for the headings

tree.heading("User ID", text="User ID", anchor=CENTER)

tree.heading("First Name", text="First Name", anchor=CENTER)

tree.heading("Last Name", text="Last Name", anchor=CENTER)

tree.heading("User Name", text="User Name", anchor=CENTER)

tree.heading("Email", text="Email", anchor=CENTER)

tree.heading("Mobile No", text="Mobile No", anchor=CENTER)

# Configure Treeview style

style = ttk.Style()

style.configure("Custom.Treeview", font=("Cooper Black", 12))

style.configure("Treeview.Row", padding=(0, 5))

# Set the background color for the headers

style.configure("Treeview.Heading", background="gray")

# Define a tag named "my\_font" with the desired font

tree.tag\_configure("my\_font", font=("Cooper Black", 12))

tree.pack()

# Connect to the SQLite3 database

conn = connect('fms.db')

cursor = conn.cursor()

# Retrieve the user's data from the database

cursor.execute("SELECT user\_id, first\_name, last\_name, user\_name, email, mobile\_no FROM users WHERE user\_id = ?", (user\_id,))

user = cursor.fetchone()

if user:

# Insert the manager's data into the Treeview

tree.insert("", END, values=(user[0], user[1], user[2], user[3], user[4], user[5]))

# Apply the "my\_font" tag to all items in the treeview

for user in tree.get\_children():

tree.item(user, tags=("my\_font",))

edit\_button = Button(content\_frame, text="EDIT PROFILE", command=edit\_user, bg="#3498db", fg="white", font=("Cooper Black", 12))

edit\_button.pack(side=LEFT, padx=550, pady=10)

fms.scr.mainloop()

#---------------------------------------------------User Pending Donations Page-------------------------------------------------------------------#

def Userdonations(fms, user\_id):

fms.scr.destroy()

fms.scr = Tk()

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.managerdashf1= Frame(fms.scr, bg="#ffffff")

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img1 = fms.logo.subsample(2,2)

fms.logo\_banner1 = Label(fms.managerdashf1, image=fms.logo\_img1)

fms.logo\_banner1.pack(side=LEFT)

fms.managerlabel= Label(text="Weclome to User Dashboard!!", bg="#0b1335", fg="white",font=("cooper black",14))

fms.managerlabel.place(x=1000,y=50)

fms.logout=Button(fms.managerdashf1,text="LOGOUT", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.logout.place(x=1075,y=100)

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.managerdashf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.managerdashf1.pack(fill=BOTH)

fms.sidebar = Frame(fms.scr, bg="#d3ede6")

fms.sidebar.pack(side=LEFT, fill=Y)

# Add buttons to the sidebar

fms.profile = Button(fms.sidebar,text="VIEW PROFILE", command=lambda:fms.Userdetails(user\_id), bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.profile.grid(row=3, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewdonations= Button(fms.sidebar,text="PENDING DONATIONS", command=lambda:fms.Userdonations(user\_id), bg="#f39c12",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewdonations.grid(row=4, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewaccepteddonations= Button(fms.sidebar,text="ACCEPTED DONATIONS", command=lambda:fms.User\_accepted\_donations(user\_id), bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewaccepteddonations.grid(row=8, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

def show\_custom\_messages(message):

# Create a new top-level window

message\_dialog = Toplevel()

message\_dialog.title("Message")

# Set window size and position it in the center of the screen

window\_width = 350

window\_height = 150

screen\_width = message\_dialog.winfo\_screenwidth()

screen\_height = message\_dialog.winfo\_screenheight()

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

message\_dialog.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Add an icon label in #32CD32 color

style = ttk.Style()

style.configure("Error.TLabel", foreground="#32CD32")

error\_icon = ttk.Label(message\_dialog, text="\u24D8", style="Error.TLabel", font=("Cooper Black", 24))

error\_icon.pack(pady=10)

label = Label(message\_dialog, text=message, fg="#32CD32", font=("Cooper Black", 12))

label.pack(padx=10, pady=0)

# Add an "OK" button to close the dialog

ok\_button = Button(message\_dialog, text="OK", command=message\_dialog.destroy, width=10, font=("Cooper Black", 12))

ok\_button.pack(pady=10)

def show\_custom\_error(message):

# Create a new top-level window

error\_dialog = Toplevel()

error\_dialog.title("Error Message")

# Set window size and position it in the center of the screen

window\_width = 350

window\_height = 150

screen\_width = error\_dialog.winfo\_screenwidth()

screen\_height = error\_dialog.winfo\_screenheight()

x = (screen\_width - window\_width) // 2

y = (screen\_height - window\_height) // 2

error\_dialog.geometry(f"{window\_width}x{window\_height}+{x}+{y}")

# Set window icon

# error\_dialog.iconbitmap(r"house.ico")

# Add an icon label in red color

style = ttk.Style()

style.configure("Error.TLabel", foreground="red")

error\_icon = ttk.Label(error\_dialog, text="\u26A0", style="Error.TLabel", font=("Cooper Black", 24))

error\_icon.pack(pady=10)

label = Label(error\_dialog, text=message, fg="red", font=("Cooper Black", 12))

label.pack(padx=10, pady=0)

# Add an "OK" button to close the dialog

ok\_button = Button(error\_dialog, text="OK", command=error\_dialog.destroy, width=10, font=("Cooper Black", 12))

ok\_button.pack(pady=10)

# Function to load data into the Treeview

def load\_donations():

# Clear the existing data in the Treeview

tree.delete(\*tree.get\_children())

# Connect to the SQLite3 database

conn = connect('fms.db')

cursor = conn.cursor()

try:

# Fetch and display pending donations

cursor.execute("SELECT donation\_id, item\_name, item\_type, calories, amount\_lb, servings, manager\_id, donation\_status FROM donation\_details WHERE donation\_status = 'pending'")

pending\_donations = cursor.fetchall()

for donation in pending\_donations:

# Fetch the manager's user\_name from the managers table

cursor.execute("SELECT user\_name FROM managers WHERE manager\_id=?", (donation[6],))

manager\_name = cursor.fetchone()[0] # Assuming manager\_id is at index 7

# Set the status color based on the donation status

status\_color = 'red' # Assuming pending donations are marked in red

# Insert a checkbox in the first column for selecting the donation

tree.insert("", END, values=(donation[0], donation[1], donation[2], donation[3], donation[4], donation[5], manager\_name, donation[7]), tags=("my\_font", status\_color))

# Apply the "my\_font" tag to all items in the treeview

for donation in tree.get\_children():

tree.item(donation, tags=("my\_font", status\_color))

except Exception as e:

# Show error message if retrieval fails

show\_custom\_error(f"Failed to load donations: {str(e)}")

finally:

# Close the connection

conn.close()

# code to accept the donation

def accept\_donation(user\_id):

# Get the selected item from the Treeview

selected\_item = tree.focus()

if not selected\_item:

show\_custom\_error("Please select a donation to accept.")

return

# Get the donation\_id of the selected donation

donation\_id = tree.item(selected\_item, "values")[0]

# Connect to the database

conn = connect('fms.db')

cursor = conn.cursor()

try:

# Check if the donation is already accepted

donation\_status = tree.item(selected\_item, "values")[7] # Assuming status is at index 7

if donation\_status == 'accepted':

show\_custom\_messages("Donation already accepted")

else:

# Update the donation\_status to 'accepted' in the donation\_details table

cursor.execute("UPDATE donation\_details SET donation\_status = 'accepted', user\_id = ? WHERE donation\_id = ?", (user\_id, donation\_id,))

conn.commit()

# Get manager details

cursor.execute("SELECT managers.user\_name, managers.email FROM managers JOIN donation\_details ON managers.manager\_id = donation\_details.manager\_id WHERE donation\_details.donation\_id = ?", (donation\_id,))

manager\_details = cursor.fetchone()

manager\_name, manager\_email = manager\_details

# Get user\_name and donation details

cursor.execute("SELECT users.user\_name, donation\_details.\* FROM users JOIN donation\_details ON users.user\_id = donation\_details.user\_id WHERE donation\_id = ?", (donation\_id,))

user\_details = cursor.fetchone()

user\_name = user\_details[0]

donation\_id, item\_name, item\_type, calories, amount\_lb, servings = user\_details[1:7]

# Compose email

sender\_email = "fms38865@gmail.com"

sender\_password = "ktee vlno ediy uiop"

subject = "Donation Accepted"

message = f"Hello {manager\_name},\n\nYour food donation has been accepted by {user\_name} with the following details:\n\nItem Name: {item\_name}\nItem Type: {item\_type}\nCalories: {calories}\nAmount (lb): {amount\_lb}\nServings: {servings}\n\nThank you,\nFood Management Team"

# Send email to the manager

msg = MIMEMultipart()

msg['From'] = sender\_email

msg['To'] = manager\_email

msg['Subject'] = subject

msg.attach(MIMEText(message, 'plain'))

server = smtplib.SMTP('smtp.gmail.com', 587)

server.starttls()

server.login(sender\_email, sender\_password)

server.sendmail(sender\_email, manager\_email, msg.as\_string())

server.quit()

# Show success message

show\_custom\_messages("Donation accepted successfully")

# Refresh the Treeview

load\_donations()

except Exception as e:

# Show error message if the update fails

show\_custom\_error(f"Failed to accept donation: {str(e)}")

finally:

# Close the connection

conn.close()

# Create the main content frame

content\_frame = Frame(fms.scr, bg="lightgray")

content\_frame.pack(side=LEFT, fill=BOTH, expand=True)

# Create a label for the title

title\_label = Label(content\_frame, text="VIEW PENDING DONATIONS", font=("Cooper Black", 16), bg="#0b1335", fg="white", pady=10)

title\_label.pack(fill=X)

title\_label.pack(pady=(0, 85))

# Create treeview

tree = ttk.Treeview(content\_frame, columns=("Donation ID", "Item Name", "Item Type", "Calories", "Amount(Lbs)", "Servings", "Added By", "Status"), show="headings", padding=(0, 5))

# Set the font for the headings

tree.heading("Donation ID", text="Donation ID", anchor=CENTER)

tree.heading("Item Name", text="Item Name", anchor=CENTER)

tree.heading("Item Type", text="Item Type", anchor=CENTER)

tree.heading("Calories", text="Calories", anchor=CENTER)

tree.heading("Amount(Lbs)", text="Amount(Lbs)", anchor=CENTER)

tree.heading("Servings", text="Servings", anchor=CENTER)

tree.heading("Added By", text="Added By", anchor=CENTER)

tree.heading("Status", text="Status", anchor=CENTER)

# Configure Treeview style

style = ttk.Style()

style.configure("Custom.Treeview", font=("Cooper Black", 12))

style.configure("Treeview.Row", padding=(0, 5))

# Set the background color for the headers

style.configure("Treeview.Heading", background="gray")

# Define a tag named "my\_font" with the desired font

tree.tag\_configure("my\_font", font=("Cooper Black", 12))

# Set the column widths

tree.column("Donation ID", width=100)

tree.column("Item Name", width=200)

tree.column("Item Type", width=150)

tree.column("Calories", width=150)

tree.column("Amount(Lbs)", width=150)

tree.column("Servings", width=150)

tree.column("Added By", width=150)

tree.column("Status", width=150)

tree.pack()

# Connect to the SQLite3 database

conn = connect('fms.db')

cursor = conn.cursor()

# Fetch and display pending donations

cursor.execute("SELECT donation\_id, item\_name, item\_type, calories, amount\_lb, servings, manager\_id, donation\_status, user\_id FROM donation\_details WHERE donation\_status = 'pending'")

pending\_donations = cursor.fetchall()

for donation in pending\_donations:

# Fetch the manager's user\_name from the managers table

cursor.execute("SELECT user\_name FROM managers WHERE manager\_id=?", (donation[6],))

manager\_name = cursor.fetchone()[0]

# Set the status color based on the donation status

status\_color = 'red' # Assuming pending donations are marked in red

# Insert the data into the Treeview with the status color

tree.insert("", END, values=(donation[0], donation[1], donation[2], donation[3], donation[4], donation[5], manager\_name, donation[7]), tags=("my\_font", status\_color))

# Apply the "my\_font" tag to all items in the treeview

for donation in tree.get\_children():

tree.item(donation, tags=("my\_font", status\_color))

# Button to accept the selected donation

accept\_button = Button(content\_frame, text="ACCEPT DONATION", command=lambda: accept\_donation(user\_id), bg="#2ecc71", fg="white", font=("Cooper Black", 12))

accept\_button.pack(side=LEFT, padx=500, pady=10)

conn.close()

fms.scr.mainloop()

#---------------------------------------------------User Accepted Donations Page------------------------------------------------------------------#

def User\_accepted\_donations(fms, user\_id):

fms.scr.destroy()

fms.scr = Tk()

# getting screen width and height of display

width= fms.scr.winfo\_screenwidth()

height= fms.scr.winfo\_screenheight()

#setting tkinter window size

fms.scr.geometry("%dx%d" % (width, height))

fms.scr.title("FOOD MANAGEMENT SYSTEM")

fms.managerdashf1= Frame(fms.scr, bg="#ffffff")

fms.logo = PhotoImage(file="logo.png")

fms.logo\_img1 = fms.logo.subsample(2,2)

fms.logo\_banner1 = Label(fms.managerdashf1, image=fms.logo\_img1)

fms.logo\_banner1.pack(side=LEFT)

fms.managerlabel= Label(text="Weclome to User Dashboard!!", bg="#0b1335", fg="white",font=("cooper black",14))

fms.managerlabel.place(x=1000,y=50)

fms.logout=Button(fms.managerdashf1,text="LOGOUT", command=fms.main, bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.logout.place(x=1075,y=100)

fms.localtime=time.asctime(time.localtime(time.time()))

fms.tim=Label(fms.managerdashf1,text=fms.localtime,fg="white",font=("cooper black",14),bg="#0b1335")

fms.managerdashf1.pack(fill=BOTH)

fms.sidebar = Frame(fms.scr, bg="#d3ede6")

fms.sidebar.pack(side=LEFT, fill=Y)

# Add buttons to the sidebar

fms.profile = Button(fms.sidebar,text="VIEW PROFILE", command=lambda:fms.Userdetails(user\_id), bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.profile.grid(row=3, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewdonations= Button(fms.sidebar,text="PENDING DONATIONS", command=lambda:fms.Userdonations(user\_id), bg="#0b1335",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewdonations.grid(row=4, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

fms.viewaccepteddonations= Button(fms.sidebar,text="ACCEPTED DONATIONS", command=lambda:fms.User\_accepted\_donations(user\_id), bg="#f39c12",cursor="hand2",

fg="white",font=("cooper black",14))

fms.viewaccepteddonations.grid(row=8, column=0, ipadx=0, padx=15, pady=15, sticky="ew")

# Create the main content frame

content\_frame = Frame(fms.scr, bg="lightgray")

content\_frame.pack(side=LEFT, fill=BOTH, expand=True)

# Create a label for the title

title\_label = Label(content\_frame, text="VIEW ACCEPTED DONATIONS", font=("Cooper Black", 16), bg="#0b1335", fg="white", pady=10)

title\_label.pack(fill=X)

title\_label.pack(pady=(0, 85))

# Create treeview

tree = ttk.Treeview(content\_frame, columns=("Donation ID", "Item Name", "Item Type", "Calories", "Amount(Lbs)", "Servings", "Added By", "Status"), show="headings", padding=(0, 5))

# Set the font for the headings

tree.heading("Donation ID", text="Donation ID", anchor=CENTER)

tree.heading("Item Name", text="Item Name", anchor=CENTER)

tree.heading("Item Type", text="Item Type", anchor=CENTER)

tree.heading("Calories", text="Calories", anchor=CENTER)

tree.heading("Amount(Lbs)", text="Amount(Lbs)", anchor=CENTER)

tree.heading("Servings", text="Servings", anchor=CENTER)

tree.heading("Added By", text="Added By", anchor=CENTER)

tree.heading("Status", text="Status", anchor=CENTER)

# Configure Treeview style

style = ttk.Style()

style.configure("Custom.Treeview", font=("Cooper Black", 12))

style.configure("Treeview.Row", padding=(0, 5))

# Set the background color for the headers

style.configure("Treeview.Heading", background="gray")

# Define a tag named "my\_font" with the desired font

tree.tag\_configure("my\_font", font=("Cooper Black", 12))

# Set the column widths

tree.column("Donation ID", width=100)

tree.column("Item Name", width=200)

tree.column("Item Type", width=150)

tree.column("Calories", width=150)

tree.column("Amount(Lbs)", width=150)

tree.column("Servings", width=150)

tree.column("Added By", width=150)

tree.column("Status", width=150)

tree.pack()

# Connect to the SQLite3 database

conn = connect('fms.db')

cursor = conn.cursor()

# Fetch and display pending donations

cursor.execute("SELECT donation\_id, item\_name, item\_type, calories, amount\_lb, servings, manager\_id, donation\_status, user\_id FROM donation\_details WHERE donation\_status = 'accepted'AND user\_id = ?", (user\_id,))

pending\_donations = cursor.fetchall()

for donation in pending\_donations:

# Fetch the manager's user\_name from the managers table

cursor.execute("SELECT user\_name FROM managers WHERE manager\_id=?", (donation[6],))

manager\_name = cursor.fetchone()[0]

# Set the status color based on the donation status

status\_color = 'red' # Assuming pending donations are marked in red

# Insert the data into the Treeview with the status color

tree.insert("", END, values=(donation[0], donation[1], donation[2], donation[3], donation[4], donation[5], manager\_name, donation[7]), tags=("my\_font", status\_color))

# Apply the "my\_font" tag to all items in the treeview

for donation in tree.get\_children():

tree.item(donation, tags=("my\_font", status\_color))

# Button to accept the selected donation

# accept\_button = Button(content\_frame, text="ACCEPT DONATION", command=lambda: accept\_donation(user\_id), bg="#2ecc71", fg="white", font=("Cooper Black", 12))

# accept\_button.pack(side=LEFT, padx=500, pady=10)

conn.close()

fms.scr.mainloop()

#---------------------------------------------------User Dashboard Code End-----------------------------------------------------------------------#

#---------------------------------------------------Database Connections Code Start---------------------------------------------------------------#

def validate\_email(fms, email):

# Regular expression pattern for email validation

pattern = r'^[a-zA-Z0-9\_.+-]\*@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]\*$'

return re.match(pattern, email) is not None

def validate\_mobile\_number(fms, mobile\_no):

return mobile\_no.isdigit() and len(mobile\_no) == 10

def userReg(fms):

def resultreg():

first\_name = fms.first.get()

last\_name = fms.last.get()

user\_name = fms.usern.get()

password = fms.passd.get()

email = fms.email.get()

mobile\_no = fms.mob.get()

return first\_name, last\_name, user\_name, password, email, mobile\_no

fms.credreg = resultreg()

fms.con = connect("fms.db")

fms.cur = fms.con.cursor()

try:

fms.cur.execute('''CREATE TABLE IF NOT EXISTS users (

user\_id INTEGER PRIMARY KEY AUTOINCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

user\_name VARCHAR(50) NOT NULL,

password VARCHAR(50) NOT NULL,

email VARCHAR(50),

mobile\_no VARCHAR(50) NOT NULL

)''')

except Exception as e:

print(e)

# Check if the username and email already exist

fms.cur.execute("SELECT count(\*) FROM users WHERE user\_name=? OR email=?", (fms.credreg[2], fms.credreg[4]))

count = fms.cur.fetchone()[0]

if count > 0:

messagebox.showerror("Error", "Username or email already exists.", icon='error')

elif "" in fms.credreg[:6]:

messagebox.showerror("Error", "Empty entry is not allowed", icon='error')

elif not fms.validate\_email(fms.credreg[4]):

messagebox.showerror("Error", "Invalid email address", icon='error')

elif not fms.validate\_mobile\_number(fms.credreg[5]):

messagebox.showerror("Error", "Invalid mobile number format. It should consist of 10 digits only.", icon='error')

else:

try:

fms.cur.execute("INSERT INTO users (first\_name, last\_name, user\_name, password, email, mobile\_no) VALUES (?, ?, ?, ?, ?, ?)",

(fms.credreg[0], fms.credreg[1], fms.credreg[2], fms.credreg[3], fms.credreg[4], fms.credreg[5]))

fms.con.commit()

messagebox.showinfo("Success", "You have successfully registered", icon="info")

fms.main()

fms.scr.destroy() # Destroy the registration window

except Exception as e:

print(e)

def userlog(fms):

fms.loguser=fms.user.get()

fms.logpass=fms.pasd.get()

return fms.loguser,fms.logpass

def userdatabase(fms):

fms.credlog=fms.userlog()

fms.con=connect("fms.db")

fms.cur=fms.con.cursor()

x=fms.cur.execute("select count(\*), user\_id from users where user\_name=%r and password=%r"%(fms.credlog[0],fms.credlog[1]))

count, user\_id = list(x)[0]

if count == 0:

if fms.credlog[0]=="" or fms.credlog[1]=="":

messagebox.showerror("Error", "Empty entry is not allowed", icon='error')

else:

messagebox.showerror("Error", "You are not registered yet", icon='error')

else:

messagebox.showinfo("Success", "You have successfully logged in", icon="info")

# Assuming you have user\_id here

fms.Userdetails(user\_id)

# fms.Userdonations(user\_id)

def resultadmin(fms):

fms.loguser=fms.usera.get()

fms.logpass=fms.pasda.get()

return fms.loguser,fms.logpass

def admindatabase(fms):

fms.credadm=fms.resultadmin()

fms.con=connect("fms.db")

fms.cur=fms.con.cursor()

x=fms.cur.execute("select count(\*) from admin where user\_name=%r and password=%r"%(fms.credadm[0],fms.credadm[1]))

if list(x)[0][0]==0:

if fms.credadm[0]=="" or fms.credadm[1]=="":

messagebox.showerror("Error", "Empty entry is not allowed", icon='error')

else:

messagebox.showerror("Error", "Invalid credentials!", icon='error')

else:

messagebox.showinfo("Success", "You have successfully logged in", icon="info")

fms.Viewmanagers()

def resultmanager(fms):

fms.loguser=fms.usera.get()

fms.logpass=fms.pasda.get()

return fms.loguser,fms.logpass

def managerdatabase(fms):

#code for checking credentials

fms.credmanager = fms.resultmanager()

fms.con = connect("fms.db")

fms.cur = fms.con.cursor()

x = fms.cur.execute("select count(\*), manager\_id from managers where user\_name=%r and password=%r" % (fms.credmanager[0], fms.credmanager[1]))

count, manager\_id = list(x)[0]

if count == 0:

if fms.credmanager[0] == "" or fms.credmanager[1] == "":

messagebox.showerror("Error", "Empty entry is not allowed", icon='error')

else:

messagebox.showerror("Error", "Please contact your administrator to register your account.", icon='error')

else:

messagebox.showinfo("Success", "You have successfully logged in", icon="info")

fms.Managerdetails(manager\_id)

#---------------------------------------------------Database Connections Code End-----------------------------------------------------------------#

x = FoodManagementSystem()

x.main()