```python
# This project belongs to Mohd Tabish Khan 20231010115006 and Vimal Pandey 202310101150005

import tkinter as tk

from tkinter import ttk

# from tkmacosx import Button

from tkinter import Button

from tkinter import messagebox

import firebase_admin

from firebase_admin import credentials, firestore

import requests


# Fetching Firestore credentials and initializing the app


cred = credentials.Certificate("serviceAccountKey.json")

firebase_admin.initialize_app(cred)

db = firestore.client()

counter = 0

api_key = "AIzaSyADdFExEyGBDmD-ulDp8GFqTCUj-92UZOs"


# Global variables for pagination

last_expense = None

expense_counter = 1  # Counter for serial number

# Data Initialize for the Treeview

data = []


# login

def login():

    root = tk.Tk()


    def loginFunction():

        email = entry_username.get()

        password = entry_password.get()


        url = f"https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key={api_key}"


        payload = {

            "email": email,

            "password": password,

            "returnSecureToken": True

        }

        if not email or not password:

            messagebox.showerror("Fill The Given Details","Make Sure Each Required Details Are Filled")

            return

        loading_window = tk.Toplevel(root)

        loading_window.title("Loading...")

        loading_window.geometry("200x100")

        loading_label = tk.Label(

        loading_window, text="Logging ...", font=("Arial", 12))

        loading_label.pack(pady=20)
```

```
        root.update()
        try:
            response = requests.post(url, json=payload)
            data = response.json()
            if "error" in data:
                messagebox.showerror("Login Failed", data["error"]["message"])
            else:
                root.destroy()
                ExpenseManager( data["localId"],data["idToken"])
        except Exception as e:
            messagebox.showerror("Error", "An error occurred")
        loading_window.destroy()

    def navigateToSignUp():
        root.destroy()
        SignUp()
    root.title("Login Page")
    root.geometry("720x500")
    root.resizable(False, False)
    # Set the background color to black
    root.configure(bg="black")

    # Set the text color to green
    text_color = "green"

    label_username = tk.Label(root, text="Email :", fg=text_color, bg="black", font=("Arial", 28))
    label_username.place(x=50, y=100)

    entry_username = tk.Entry(root, fg=text_color, bg="black", font=("Arial", 28))
    entry_username.place(x=250, y=100)

    label_password = tk.Label(root, text="Password:", fg=text_color, bg="black", font=("Arial", 28))
    label_password.place(x=50, y=200)

    entry_password = tk.Entry(root, show="*", fg=text_color, bg="black", font=("Arial", 28))
    entry_password.place(x=250, y=200)

    login_button = Button(root, text="Login",command=loginFunction,  fg="black", bg=text_color, font=("Arial", 28))
    login_button.place(x=300, y=300)

    no_account_btn = Button(root, text="Don't have an account?",fg=text_color, bg="black", command=navigateToSignUp, font=("Arial", 28))
    no_account_btn.place(x=150, y=400)

    root.mainloop()

# signup
def SignUp():
    root = tk.Tk()
```

```
# function


def SignUpFunction():
    email = entry_username.get()
    password = entry_password.get()
    if (not email or not password):
        messagebox.showerror("Fill The Given Details","Make Sure Each Required Details Are Filled")
        return
    url = f"https://identitytoolkit.googleapis.com/v1/accounts:signUp?key={api_key}"
    payload = {
        "email":email,
        "password":password,
        "returnSecureToken":True
    }
    loading_window = tk.Toplevel(root)
    loading_window.title("Loading...")
    loading_window.geometry("200x100")
    loading_label = tk.Label(
    loading_window, text="Logging ...", font=("Arial", 12))
    loading_label.pack(pady=20)
    root.update()
    try:
        response = requests.post(url,data=payload);
        if (response.status_code == 200):
            data = response.json();
            root.destroy()
            ExpenseManager(data["localId"],data["refreshToken"])
        else:
            messagebox.showerror("Something Went Wrong","User already found")

    except Exception as e:
        messagebox.showerror("Something Went Wrong","User already found")
    loading_window.destroy()




def navigateToSignIn():
    root.destroy()
    login()


root.title("Signup Page")
root.geometry("720x500")
root.resizable(False, False)
# Set the background color to black
root.configure(bg="black")


# Set the text color to green
text_color = "green"
```

```
    label_username = tk.Label(root, text="Email:",
                              fg=text_color, bg="black", font=("Arial", 28))
    label_username.place(x=50, y=100)

    entry_username = tk.Entry(root, fg=text_color, bg="black", font=("Arial", 28))
    entry_username.place(x=250, y=100)

    label_password = tk.Label(root, text="Password:",
                              fg=text_color, bg="black", font=("Arial", 28))
    label_password.place(x=50, y=200)

    entry_password = tk.Entry(root, show="*", fg=text_color, bg="black", font=("Arial", 28))
    entry_password.place(x=250, y=200)

    signup_button = Button(root, text="Signup", fg="black", bg=text_color, font=("Arial", 28),command=SignUpFunction)
    signup_button.place(x=300, y=300)

    already_have_account_btn = Button(
        root, text="Already have an account?", fg=text_color, bg="black", command=navigateToSignIn, font=("Arial", 28))
    already_have_account_btn.place(x=150, y=400)

    root.mainloop()



# Expense
def ExpenseManager(uid, token):

    root = tk.Tk()
    root.title("Expense Tracker")
    root.geometry("720x500")
    root.config(bg="black")
    root.resizable(False, False)



    def logOut():
        global data
        root.destroy()
        data = []
        login()

    def addExpense(payload):
        try:
            doc_ref = db.collection("user").document(uid).collection("expenses").add(payload)
            return {"status": 1, "resp": doc_ref}
        except Exception as e:
            return {"status": 1, "message": e}
    # Function to retrieve next 10 expenses and update the Tkinter Treeview widget

    def load_more_expenses():
```

```
    global last_expense, expense_counter, counter

    loading_window = tk.Toplevel(root)

    loading_window.title("Loading...")

    loading_window.geometry("200x100")

    loading_label = tk.Label(

    loading_window, text="Fetching Expense...", font=("Arial", 12))

    loading_label.pack(pady=20)

    len_expenses = 0

    root.update()

    # Check if last_expense is None, set it to an empty string in that case

    if last_expense is None:

        last_expense = ""



    expenses = db.collection("user").document(uid).collection(

        "expenses").order_by("date").start_after(last_expense).limit(10).stream()

    for expense in expenses:

        len_expenses += 1

        counter += 1



        data.append({

            "date": expense.get('date'),

            "spent_on": expense.get('spent_on'),

            "amount": expense.get('amount'),

            "key": expense.id

        })

        update_treeview()

        last_expense = expense

        expense_counter += 1



    # Check if there are more expenses, if not, hide the Load More button

    if len_expenses == 0:

        load_More.destroy()



        messagebox.showinfo("Alert", "No more expenses to load.")

    loading_window.destroy()

# Function to fetch the initial 10 expenses and update the Tkinter Treeview widget



def show_initial_expenses():

    global last_expense, expense_counter, counter

    loading_window = tk.Toplevel(root)

    loading_window.title("Loading...")

    loading_window.geometry("200x100")

    loading_label = tk.Label(

    loading_window, text="Fetching Expense...", font=("Arial", 12))

    loading_label.pack(pady=20)

    len_expenses = 0

    root.update()

    expenses = db.collection("user").document(uid).collection(
```

```
        "expenses").order_by("date").limit(10).stream()
    for expense in expenses:
        counter += 1
        data.append({
            "date": expense.get('date'),
            "spent_on": expense.get('spent_on'),
            "amount": expense.get('amount'),
            "key": expense.id
        })
        last_expense = expense  # Update last_expense here
        expense_counter += 1
    update_treeview()
    loading_window.destroy()
# Function to update the Tkinter Treeview with the data


def update_treeview():
    for item in table.get_children():
        table.delete(item)
    # Clear existing items in the Treeview
    if not data:
        # If there are no expenses, add a row with a message
        table.insert("", "end", values=(
            "No Expense Found", "", "", "", ""))
        return




# Insert data into the Treeview with sequential serial number
    for i, item in enumerate(data, start=1):
        table.insert("", "end", values=(
            i, item["date"], item["amount"], item["spent_on"],item["key"]))
# Create a Treeview widget for the table
table = ttk.Treeview(root, columns=(
    "No", "Date", "Spent On", "Amount", "Key"), height=7,)
table.heading("#1", text="No")
table.heading("#2", text="Date")
table.heading("#3", text="Spent On")
table.heading("#4", text="Amount")
table.heading("#5", text="Key")


def add_expense():
    # Get data from input fields
    date_value = date_entry.get()
    spent_on_value = spent_entry.get()
    amount_value = amount_entry.get()
    # Validate if all fields are filled
    if not (date_value and spent_on_value and amount_value):
        # You can show an error message or handle this case in another way
        messagebox.showerror("Error", "Please fill all the fields")
```

```
        return


    global counter

    loading_window = tk.Toplevel(root)

    loading_window.title("Loading...")

    loading_window.geometry("200x100")

    loading_label = tk.Label(

    loading_window, text="Adding Expense...", font=("Arial", 12))

    loading_label.pack(pady=20)


    # Update the main window during the loading process

    root.update()


    # Create a payload tuple

    expense_payload = {

        "date": date_value,

        "spent_on": spent_on_value,

        "amount": amount_value

    }


    # Call the addExpense function with the payload

    resp = addExpense(expense_payload)

    # Close the loading window after the operation completes

    loading_window.destroy()


    if resp.get("status") == 1:


        timestamp, doc_ref = resp.get("resp")

        expense_payload["key"] = doc_ref.id


        data.append(expense_payload)

        counter += 1


        update_treeview()


    # Clear input fields after adding an expense

    date_entry.delete(0, tk.END)

    spent_entry.delete(0, tk.END)

    amount_entry.delete(0, tk.END)

# Add an event to the Treeview to capture double-clicks

table.bind("<Double-1>", lambda event: on_treeview_double_click(event))

# Function to handle double-click event on the Treeview


def on_treeview_double_click(event):

    item = table.selection()[0]  # Get the selected item


    # Get the values of the selected item

    item_values = table.item(item, "values")
```

```
if (not item_values[0] or not item_values[1] or not item_values[2] or not item_values[3] or not item_values[4]):
    return
# Open a new window for updating the expense
update_window = tk.Toplevel(root)
update_window.title("Update Expense")
update_window.geometry("300x200")

# Create labels and entry widgets for updating fields
date_label = tk.Label(update_window, text="Date", font=("Arial", 12))
date_label.grid(row=0, column=0, padx=10, pady=10)
date_entry = tk.Entry(update_window, font=("Arial", 12))
date_entry.grid(row=0, column=1, padx=10, pady=10)
date_entry.insert(0, item_values[1])

spent_on_label = tk.Label(
update_window, text="Spent On", font=("Arial", 12))
spent_on_label.grid(row=1, column=0, padx=10, pady=10)
spent_on_entry = tk.Entry(update_window, font=("Arial", 12))
spent_on_entry.grid(row=1, column=1, padx=10, pady=10)
spent_on_entry.insert(0, item_values[2])

amount_label = tk.Label(
    update_window, text="Amount", font=("Arial", 12))
amount_label.grid(row=2, column=0, padx=10, pady=10)
amount_entry = tk.Entry(update_window, font=("Arial", 12))
amount_entry.grid(row=2, column=1, padx=10, pady=10)
amount_entry.insert(0, item_values[3])

# Function to update the expense
def update_expense():
    new_date = date_entry.get()
    new_spent_on = spent_on_entry.get()
    new_amount = amount_entry.get()

    # Validate if all fields are filled
    if not (new_date and new_spent_on and new_amount):
        messagebox.showerror("Error", "Please fill all the fields")
        return

    # Update the data in the data list
    updated_data = {
        "date": new_date,
        "spent_on": new_spent_on,
        "amount": new_amount,
        "key": item_values[4]  # The key of the expense
    }

    try:
```

```
            # Attempt to update the expense in Firestore


            db.collection("user").document(uid).collection("expenses").document(item_values[4]).update({
                "date": new_date,
                "spent_on": new_spent_on,
                "amount": new_amount
            })
            # If the update in Firestore is successful, update the data in the data list
            data[table.index(item)] = updated_data
            update_treeview()  # Refresh the Treeview widget
            # Close the update window
            update_window.destroy()
        except Exception as e:
            # If an error occurs during the Firestore update, show an alert message
            messagebox.showerror(
                "Error", f"Failed to update expense: {str(e)}")
    def delete_expense():
        pass
    # Button to trigger the update
    update_button = tk.Button(update_window, text="Update", command=update_expense)
    update_button.grid(row=3, column=0, columnspan=2, pady=10)


#function for delete
def on_treeview_click(event,delete=False):
    # Get the selected item
    item = table.selection()[0]


    # Get the data from the item
    item_values = table.item(item, 'values')
    if (not item_values[0] or not item_values[1] or not item_values[2] or not item_values[3] or not item_values[4]):
        return
    if delete:
    # Print the selected data
        try:
            db.collection("user").document(uid).collection("expenses").document(item_values[4]).delete()
            data.remove(data[table.index(item)])
            update_treeview()
        except Exception as e:
            messagebox.showerror("Error",f"Something Went Wrong Can't Detete The Expense {str(e)}")


    else:
        pass



# function for calculating
def oncalc():
    Calculator = tk.Toplevel()
    Calculator.geometry("390x650")
    Calculator.config(bg="black")
```

```
# calc = tk.Toplevel()
global expression
expression =""
Calculator.title("Calculator")


def show(val):
    global expression
    expression += val
    l2.config(text=expression)


def clear():
    global expression
    expression = ''
    l2.config(text=expression)


def cal():
    result = ''
    global expression
    if expression != '':
        try:
            result = str(eval(expression))
            expression = result
        except:
            messagebox.showwarning(
                'Invalid', 'Invalid Operators and number use -_-')
            result = 'error'
            expression = result
    l2.config(text=expression)
l2 = tk.Label(Calculator, text='', font=(
    'Helvetica', 30, 'bold',), pady="-5", padx="5")
l2.place(x=0, y=100)
l2.config(bg="black", fg="white")
Button(Calculator, text='C', width=7, height=3, font=(20), bg="#ffd700", bd=4, fg="black",
       command=clear, activebackground="black", activeforeground="white").place(x=0, y=150)
Button(Calculator, text='/', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
       command=lambda: show('/'), activebackground="black", activeforeground="white").place(x=100, y=150)
Button(Calculator, text='%', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
       command=lambda: show('%'), activebackground="black", activeforeground="white").place(x=200, y=150)
Button(Calculator, text='^', width=7, height=3, font=(20), bg='#282C35', bd=1, fg='white',
       command=lambda: show('**'), activebackground="black", activeforeground="white").place(x=300, y=150)
Button(Calculator, text='7', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
       command=lambda: show('7'), activebackground="black", activeforeground="white").place(x=0, y=250)
Button(Calculator, text='8', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
       command=lambda: show('8'), activebackground="black", activeforeground="white").place(x=100, y=250)
Button(Calculator, text='9', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
       command=lambda: show('9'), activebackground="black", activeforeground="white").place(x=200, y=250)
Button(Calculator, text='-', width=7, height=3, font=(20), bg='#282C35', bd=1, fg='white',
```

```
            command=lambda: show('-'), activebackground="black", activeforeground="white").place(x=300, y=250)
    Button(Calculator, text='4', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
            command=lambda: show('4'), activebackground="black", activeforeground="white").place(x=0, y=350)
    Button(Calculator, text='5', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
            command=lambda: show('5'), activebackground="black", activeforeground="white").place(x=100, y=350)
    Button(Calculator, text='6', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
            command=lambda: show('6'), activebackground="black", activeforeground="white").place(x=200, y=350)
    Button(Calculator, text='+', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
            command=lambda: show('+'), activebackground="black", activeforeground="white").place(x=300, y=350)
    Button(Calculator, text='1', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
            command=lambda: show('1'), activebackground="black", activeforeground="white").place(x=0, y=450)
    Button(Calculator, text='2', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
            command=lambda: show('2'), activebackground="black", activeforeground="white").place(x=100, y=450)
    Button(Calculator, text='3', width=7, height=3, font=(20), bg='#282C35', bd=4, fg='white',
            command=lambda: show('3'), activebackground="black", activeforeground="white").place(x=200, y=450)
    Button(Calculator, text='=', width=7, height=8, font=(20), bg="#ffd700", bd=4, fg="black",
            command=cal, activebackground="black", activeforeground="white").place(x=300, y=453)
    Button(Calculator, text='0', width=18, height=3, font=(20), bg='#282C35', fg='white', command=lambda: show(
        '0'), activebackground="black", activeforeground="white").place(x=1, y=550)
    Button(Calculator, text='.', width=7, height=3, font=(20), bg='#282C35', fg='white', command=lambda: show(
        '.'), activebackground="black", activeforeground="white").place(x=200, y=550)


# Calculate the width of each column to make them evenly spaced
total_columns = 5
table_width = 720
column_width = table_width // total_columns
for i in range(total_columns):
    table.column(i,width=column_width, anchor="center")


# Hide the first column
table.column("#0", width=0, stretch=tk.NO)


style = ttk.Style()
style.configure("Treeview",
                background="black",
                fieldbackground="black",
                foreground="white",
                )


# Clear fun
def clear_input_fields():
    date_entry.delete(0, tk.END)
    spent_entry.delete(0, tk.END)
    amount_entry.delete(0, tk.END)




# Calculate the row height (in pixels)
row_height = 40
```

```
row1y = 340

row1x = 10

row2y = 390

row2x = 10

row3y = 1


# Create a custom style to adjust row height

style = ttk.Style()

style.configure("Treeview", rowheight=row_height)

table.pack()


line_frame = tk.Frame(root, width=720, height=2, bg="#00ff00")

line_frame.place(y=320, x=0)


# Create a label and Input Fields #tabish

date_label = tk.Label(root, text="Date", font=(

    "Arial", 18), fg="white", bg="black")

date_label.place(y=row1y, x=row1x)


date_entry = tk.Entry(root, font=("Arial", 18),

                      fg="white", bg="black", borderwidth=0.5)

date_entry.config(width=14)

date_entry.place(y=row1y, x=150)  # row1


spent_label = tk.Label(root, text="Spent On ", font=(

    "Arial", 18), fg="white", bg="black")

spent_label.place(y=row2y, x=row2x)


spent_entry = tk.Entry(root, font=("Arial", 18),

                       fg="white", bg="black", borderwidth=0.5)

spent_entry.config(width=14)

spent_entry.place(y=row2y, x=row1x+140)


amount_label = tk.Label(root, text="Amount ", font=(

    "Arial", 18), fg="white", bg="black")

amount_label.place(y=440, x=10)


amount_entry = tk.Entry(root, font=("Arial", 18),

                        fg="white", bg="black", borderwidth=0.5)

amount_entry.config(width=14)

amount_entry.place(y=440, x=row1x+140)


# Buttons

clear = Button(root, text="Clear", bg="black", fg="green",

               font=("Arial", 18), command=clear_input_fields)

clear.place(y=row1y, x=row1x+340)


add = Button(root, text="Add  ", bg="black", fg="green",
```

```
                font=("Arial", 18), command=add_expense)
    add.place(y=410, x=row1x+340)
    # message for no load more expenses
    load_more_msg = tk.Label(root, text="No Expense",
                             bg="black", font=("Arial", 17), fg="green")
    load_more_msg.place(y=345, x=row1x+430)

    load_More = Button(root, text="Load More", bg="black", fg="green", font=(
        "Arial", 18), command=load_more_expenses)
    load_More.place(y=row1y, x=row1x+430)

    delete = Button(root, text="Delete", bg="black",
                    fg="green", font=("Arial", 18),command=lambda:on_treeview_click(None,True))
    delete.place(y=row1y, x=row1x+580)

    calc_button = Button(root, text="Calculator", bg="black",
                         fg="green", font=("Arial", 18), command=oncalc)
    calc_button.place(y=410, x=440)

    logout_button = Button(root, text="Logout", bg="black",
                           fg="green", font=("Arial", 18), command=logOut)
    logout_button.place(y=410, x=row2x+580)
    table.bind('<ButtonRelease-1>', on_treeview_click)

    # Start the Tkinter main loop
    # Display initial 10 expenses
    show_initial_expenses()
    root.mainloop()



if __name__ == "__main__":
    login()
```

#ServicesAccountKey.json not found :- Please contact Vimal 8427796817 or Tabish - 9519493113