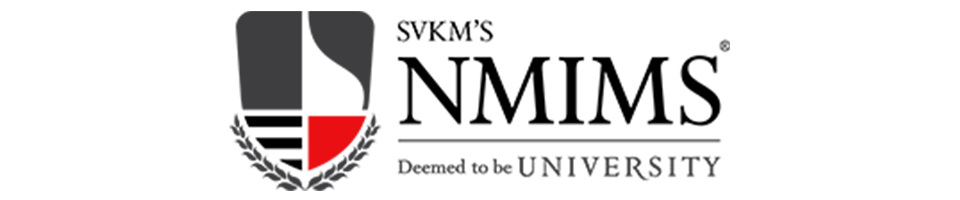


|  |
| --- |
| Student-Teacher College Interface |
|  |
| Python Programming  MBA.Tech IT  Contributors:  Pratyush Landekar I039 Nikhil Joshi I030 |



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IntroductionAbout We have made a Complete Student-College interface making use of multiple python programs and file dependencies. It has complete different Applications that run based on the role of the user (Student and Teacher) along with an admin console.  Our program has the following key features:   * Complete **Graphical User Interface** * **Login** Window for both teacher and student * **Signup** Window for both teacher and student * Complete Student and teacher **database storage** and **handling** * **Databases** for Login Credential, User Data, Student Marks, etc * User **Activity Logs** * **Error** and **Exception Handling** * Custom Libraries made by us * **Chatroom** For Users of the program * Custom **AI Assistant Chatbot** running on **GPT-3.5 Turbo** Model  Libraries Used:  * openai * tkinter * pandas * datetime * subprocess * threading * webbrowser * gradio * matplotlib * requests * json * csv  Names of Python Programs part of the project:  * Python Ultimate Student interface.py (Starter Program, login interface) * Signup\_stu.py (student signup window) * Signup\_tea.py (teacher signup window) * App\_Stu.py (Student end Application with Marks viewing, Noticeboard, Student Handbook, Chatroom, AI Assistant Chatbot) * App\_Tea.py (Teacher end Application with Marks updating, Noticeboard updating, Chatroom access) * Chatroom.py (tkinter based chatroom with local chat history) * API Test.py (AI chat bot gradio GUI and hosting) * pop.py (Custom Library with prompt processing and response retrieval for the chatbot) * pop.py (Custom Library for gpa, grade, and percentage caculation) * adminconsole.py (Gives the administrator access to all the databases, logs and history)  Databases:  * names.csv (Student Data like roll no, name and marks)  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Roll No. | Name | Last Name | LADE | QSP | EOB | PEM |  * passwords.csv (Student roll no and password storage)  |  |  | | --- | --- | | Roll No. | Password |  * names-t.csv (Teacher Data like sapid and name)  |  |  | | --- | --- | | Sapid | Name |  * passwords-t.csv (Teacher sapid and password storage)  |  |  | | --- | --- | | Sapid | Password |  * logs.csv (User activity logbook for activities like login, logout, marks update, noticeboard update)  |  |  |  |  | | --- | --- | --- | --- | | Time | User | Role | Activity |  * Temp.txt (Temporary Storage file for passing the user login credential) * chat\_history.txt (Local Storage for the chats in chatroom)  Scope Our program can be expanded to form a full fledged Student and teacher interface with several additional features to make it a viable application. However, the program has the potential to be expanded into a more comprehensive application with additional features. For instance, the interface could include a messaging system to allow students and teachers to communicate with each other. The interface could also include tools to assist with class scheduling, grading, and attendance tracking.  Another potential area for expansion is the use of machine learning algorithms to automate tasks like grading and provide personalized recommendations to students based on their learning styles and progress. Additionally, the interface could incorporate multimedia features, such as videos and interactive simulations, to enhance the learning experience.  Overall, the program has the potential to be expanded into a more robust and feature-rich Student and Teacher interface that can meet the diverse needs of both students and teachers.   Frontend Overview Python Ultimate Student interface.py:      Signup\_stu.py    Signup\_tea.py    App\_Stu.py        App\_Tea.py          Chatroom.py    API Test.py    adminconsole.py |

# Code Explained (Implementation)

Python Ultimate Student interface.py:

#-----------------------------------------------------------------------------------------------------------------

from tkinter import \*

from tkinter import messagebox

import ast

import datetime

#\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-Login Database\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

import pandas as pd

# Define file names

password\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/passwords.csv"

name\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/names.csv"

# Load data from files

passwords\_df = pd.read\_csv(password\_file, index\_col="Roll No.")

names\_df = pd.read\_csv(name\_file, index\_col="Roll No.")

# Define file names

password\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/passwords -t.csv"

name\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/names -t.csv"

# Load data from files

passwords\_dft = pd.read\_csv(password\_file, index\_col="Sapid")

names\_dft = pd.read\_csv(name\_file, index\_col="Sapid")

#\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-Login Logic\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

def teacherlog():

    Sapid = sapin.get()

    password = pw.get()

    # Check if password is correct

    if password != str(passwords\_dft.loc[Sapid, "Password"]):

        print("Invalid Password")

        messagebox.showerror("Invalid", "Invalid password entered")

        return

    # If login successful, print student name-----------------------------

    print("Login Successful. Welcome, ", names\_dft.loc[Sapid, "Name"])

    messagebox.showinfo("Success!", "Login successful")

    f=open('C:/Users/DELL/Desktop/PPS Code projects/Final project/Temp.txt','w+')

    f.write(Sapid)

    f.close()

    appt()

def login():

    roll\_no = sapin.get()

    password = pw.get()

    print(roll\_no,password)

    # Check if roll no. exists in passwords\_df

    if roll\_no=="admin" and password=="admin123":

        root.withdraw()

        subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/App/adminconsole.py"])

    elif roll\_no in passwords\_dft.index:

        teacherlog()

    # Check if roll no. exists in passwords\_df

    elif roll\_no not in passwords\_df.index and roll\_no not in passwords\_dft.index:

        print("Invalid Roll No.")

        messagebox.showerror("Invalid", "Invalid SAP ID entered")

        return

    # Check if password is correct

    elif password != str(passwords\_df.loc[roll\_no, "Password"]):

        print("Invalid Password")

        messagebox.showerror("Invalid", "Invalid password entered")

        return

    # If login successful, print student name-----------------------------

    print("Login Successful. Welcome, ", names\_df.loc[roll\_no, "Name"])

    messagebox.showinfo("Success!", "Login successful")

    f=open('C:/Users/DELL/Desktop/PPS Code projects/Final project/Temp.txt','w+')

    f.write(roll\_no)

    f.close()

    app()

#-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-Signup Query\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

import subprocess

def run\_script():

    root.withdraw()

    # Replace "other\_script.py" with the filename of the other Python script

    subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/Signup window/Signup\_stu.py"])

This is our login logic where we will retrieve the sapid and the password the user is

entering. The entered the sapid is being compared with the student database where it will match the entered sapid with the password and then only proceed. If it does not exist in the student data base but exists in the teachers data base the teacherlog() function which will authenticate the password and sapid in the teacher data base.

If it does not match will either of them it will display invalid credentials.

On successful login the entered sapid is stored in the temp.txt file. According to the data base to which the credentials are matched the respective applications will open(App\_Stu.py or App\_Tea.py )

#\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-Admin Authenticate\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

def run\_scriptt():

    root.withdraw()

    # Replace "other\_script.py" with the filename of the other Python script

    subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/Signup window/Signup\_tea.py"])

def login\_prompt():

    # Create the login window

    login\_window = Tk()

    login\_window.title("Admin Authorization")

    login\_window.geometry("+600+400")

    login\_window.config(bg="white")

    login\_window. resizable(False,False)

    h=Label(login\_window,text='Authenticate',fg='#d2232a',bg='white',font=('Bahnschrift',19,'bold'))

    h.grid(row=0, column=0, padx=0, pady=10)

    def on\_entry(e):

        # Move focus to next Entry widget

        e.widget.tk\_focusNext().focus()

    def on\_focus(event):

           username\_entry = event.widget

           if username\_entry.get() == 'Username':

            username\_entry.delete(0, 'end')

    def on\_focus\_o(event):

        password\_entry=event.widget

        if username\_entry.get() =="" :

         username\_entry.insert(0, 'Username')

         username\_entry.config(show="")

    # Create the username and password labels and entry fields

    username\_entry = Entry(login\_window,width=25, fg="black", border=0, bg="white", font=('Bahnschrift',10))

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

    username\_entry.bind("<FocusIn>", on\_focus)

    username\_entry.bind("<FocusOut>", on\_focus\_o)

    username\_entry.bind("<Return>", on\_entry)

    username\_entry.insert(0,'Username')

    Frame(login\_window,width=200,height=2,bg='black').grid(row=2, column=0, padx=5, pady=0)

    def on\_enter(e):

        # Execute function when Return key is pressed in the last Entry widget

        print("Function executed!")

        submit\_login()

    def onto\_focus(event):

           pw = event.widget

           if pw.get() == 'Password':

            pw.delete(0, 'end')

            pw.config(show='\*')

    def out\_focus(event):

        pw=event.widget

        if pw.get() =="" :

         pw.insert(0, 'Password')

         pw.config(show="")

    Frame(login\_window,width=200,height=2,bg='white').grid(row=3, column=0, padx=5, pady=10)

    password\_entry = Entry(login\_window, width=25, fg="black", border=0, bg="white", font=('Bahnschrift',10))

    password\_entry.grid(row=4, column=0, padx=0, pady=0)

    password\_entry.insert(0,'Password')

    password\_entry.bind("<FocusIn>", onto\_focus)

    password\_entry.bind("<FocusOut>", out\_focus)

    password\_entry.bind("<Return>", on\_enter)

    Frame(login\_window,width=200,height=2,bg='black').grid(row=5, column=0, padx=5, pady=0)

    # Create the submit button

    def submit\_login():

        username = username\_entry.get()

        password = password\_entry.get()

        if username == "admin" and password == "admin123":

            print("Login successful!")

            messagebox.showinfo("Authorized", "Authentication Successful")

            login\_window.destroy()

            run\_scriptt()

        else:

            messagebox.showerror("Unauthorized", "Authentication Failed")

            print("Authentication Failed")

            return

    submit\_button = Button(login\_window,width=29,pady=7,text='Authorize',bg='#d2232a',fg='white',border=0,cursor='hand2', command=submit\_login)

    submit\_button.grid(row=6, column=0, columnspan=2, padx=5, pady=10)

    # Function to change button color on hover

    def on\_hover(event):

        submit\_button['background'] = '#9f1a1f'

        submit\_button['foreground'] = 'white'

        submit\_button['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hover(event):

        submit\_button['background'] = '#d2232a'

        submit\_button['foreground'] = 'white'

        submit\_button['font'] = 'Bahnschrift 9'

    # Bind hover events to button

    submit\_button.bind('<Enter>', on\_hover)

    submit\_button.bind('<Leave>', off\_hover)

    # Run the login window

    login\_window.mainloop()

These two functions open the signup windows for student and teacher signup on being called. The student signup opens by withdrawing the current window and open a sub process to run Signup\_stu.py while the other functions conducts admin authentication which require admin user name and password and after successful admin authentication runs Signup\_tea.py

def open\_window():

    frme=Frame(root,width=350,height=320,bg="white")

    frme.place(x=560,y=180)

    # Create label and buttons

    label = Label(frme,text='Are you a',fg='#d2232a',bg='white',font=('Bahnschrift',13,'bold'))

    label.pack(padx=10, pady=5)

    a1=Button(frme,width=29,pady=7,text='Teacher',bg='#d2232a',fg='white',border=0,cursor='hand2',command=login\_prompt)

    a1.pack(padx=5,pady=10)

    # Function to change button color on hover

    def on\_hover(event):

        a1['background'] = '#9f1a1f'

        a1['foreground'] = 'white'

        a1['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hover(event):

        a1['background'] = '#d2232a'

        a1['foreground'] = 'white'

        a1['font'] = 'Bahnschrift 9'

    # Bind hover events to button

    a1.bind('<Enter>', on\_hover)

    a1.bind('<Leave>', off\_hover)

    label2 = Label(frame, text="",bg="#5a5d5d",font=('Bahnschrift',1))

    label2.pack(padx=0, pady=0)

    label = Label(frme, text="OR",bg="white")

    label.pack(padx=10, pady=5)

    #Close query window---------------------------------

    def close\_window():

        root.withdraw()

    a2=Button(frme,width=29,pady=7,text='Student',bg='#d2232a',fg='white',border=0,cursor='hand2',command=run\_script)

    a2.config(command=lambda: (close\_window(), run\_script()))

    a2.pack(padx=5,pady=10)

    # Function to change button color on hover

    def on\_hover(event):

        a2['background'] = '#9f1a1f'

        a2['foreground'] = 'white'

        a2['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hover(event):

        a2['background'] = '#d2232a'

        a2['foreground'] = 'white'

        a2['font'] = 'Bahnschrift 9'

    # Bind hover events to button

    a2.bind('<Enter>', on\_hover)

    a2.bind('<Leave>', off\_hover)

This is a subframe that contains the buttons for calling the signup window functions also calls the admin the authentication window

#-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-App\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

def app():

    root.withdraw()

    # Replace "other\_script.py" with the filename of the other Python script

    subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/App/App\_Stu.py"])

def appt():

    root.withdraw()

    # Replace "other\_script.py" with the filename of the other Python script

    subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/App/App\_Tea.py"])

#\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-Login window UI\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

root=Tk()

root. title('Login')

root.geometry('925x500+300+200')

root.configure(bg="#fff")

root. resizable(False,False)

#Image-----------------------------------

img=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/bbg.png')

Label(root,image=img,bg='white').place(x=0,y=0)

#Login Frame-----------------------------

frame=Frame(root,width=350,height=320,bg="white")

frame.place(x=520,y=100)

#Header Text-----------------------------

head=Label(frame,text='Sign In',fg='#d2232a',bg='white',font=('Bahnschrift',23,'bold'))

head.place(x=125,y=10 )

#SapID-----------------------------------

def on\_enter(e):

    # Move focus to next Entry widget

    e.widget.tk\_focusNext().focus()

def on\_focus\_in\_sap(event):

       sapin = event.widget

       if sapin.get() == 'SapID':

        sapin.delete(0, 'end')

def on\_focus\_out\_sap(event):

    pw=event.widget

    if sapin.get() =="" :

     sapin.insert(0, 'SapID')

     sapin.config(show="")

sapin=Entry(frame, width=25, fg="black", border=0, bg="white", font=('Bahnschrift',12))

sapin.bind("<FocusIn>", on\_focus\_in\_sap)

sapin.bind("<FocusOut>", on\_focus\_out\_sap)

sapin.bind("<Return>", on\_enter)

sapin.place(x=30,y=85)

sapin.insert(0,'SapID')

Frame(frame,width=295,height=2,bg='black').place(x=25,y=112)

#Password--------------------------------

def on\_submit(e):

    # Execute function when Return key is pressed in the last Entry widget

    print("Function executed!")

    login()

def on\_focus\_in(event):

       pw = event.widget

       if pw.get() == 'Password':

        pw.delete(0, 'end')

        pw.config(show='\*')

def on\_focus\_out(event):

    pw=event.widget

    if pw.get() =="" :

     pw.insert(0, 'Password')

     pw.config(show="")

pw=Entry(frame, width=25, fg="black", border=0, bg="white", font=('Bahnschrift',12))

pw.bind("<FocusIn>", on\_focus\_in)

pw.bind("<FocusOut>", on\_focus\_out)

pw.bind("<Return>", on\_submit)

pw.place(x=30,y=155)

pw.insert(0,'Password')

Frame(frame,width=295,height=2,bg='black').place(x=25,y=182)

#Buton-----------------------------------

ab=Button(frame,width=39,pady=7,text='Sign in',bg='#d2232a',fg='white',border=0,cursor='hand2',command=login)

ab.place(x=35,y=209)

# Function to change button color on hover

def on\_hover(event):

    ab['background'] = '#9f1a1f'

    ab['foreground'] = 'white'

    ab['font'] = 'Bahnschrift 9 underline'

# Function to change button color when not hovering

def off\_hover(event):

    ab['background'] = '#d2232a'

    ab['foreground'] = 'white'

    ab['font'] = 'Bahnschrift 9'

# Bind hover events to button

ab.bind('<Enter>', on\_hover)

ab.bind('<Leave>', off\_hover)

#Signup----------------------------------

# Create the hyperlink text

# Create the hyperlink-like label

def on\_enter(e):

    text.config(font=("Bahnschrift Underline", 10, "underline"), fg="#85161a")

def on\_leave(e):

    text.config(font=("Bahnschrift Underline", 10), fg="#d2232a")

text = Label(frame, text="Signup for a new account", font=("Bahnschrift Underline", 10), height=1, cursor="hand2", fg="#d2232a",bg="white")

text.place(x=25, y=260)

# Bind the hyperlink text to open a new window when clicked

text.bind("<Enter>", on\_enter)

text.bind("<Leave>", on\_leave)

text.bind("<Button-1>", lambda event: open\_window())

root.mainloop()

This is our landing page on executing the program it contains fields for entering sapid and password also submit button. It also contains a label that give option to sign up for new account based on the roles, it also contains event driven functions for the UI widgets

Signup \_stu.py:

from tkinter import \*

from tkinter import messagebox

import ast

#\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-Login Database\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

import pandas as pd

# Define file names

password\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/passwords.csv"

name\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/names.csv"

# Load data from files

passwords\_df = pd.read\_csv(password\_file, index\_col="Roll No.")

names\_df = pd.read\_csv(name\_file, index\_col="Roll No.")

#\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-Signup logic\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

import subprocess

def run\_script():

    root.withdraw()

    # Replace "other\_script.py" with the filename of the other Python script

    subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/Python Ultimate Student interface.py"])

This runs the original final window after the signup process is done while withdrawing

Current window.

def signup():

    # Get user input

    name = n.get()

    lastname = ln.get()

    roll\_no = sapin.get()

    password = pw.get()

    # Check if roll\_no already exists in the file

    if roll\_no in passwords\_df.index:

        messagebox.showerror("Invalid", "Account for this SAP ID already exists")

        print("Roll No. already exists. Please try again.")

        return

    # Add new student to dataframes

    passwords\_df.loc[roll\_no] = password

    names\_df.loc[roll\_no] = [name,lastname,0,0,0,0]

    # Save data to files

    passwords\_df.to\_csv(password\_file)

    names\_df.to\_csv(name\_file)

    print("Signup Successful. Welcome, ", name," ",lastname)

    messagebox.showinfo("Account Created", "Now head onto login page to proceed")

    run\_script()

The function here is used for signup, it retrieves the data entered in the entry fields.

it checks if the roll no entered in the field already exists in the data base if it does it will return an error saying “ACCOUNT ALREADY EXISTS”

If it does not, then it creates a new data field in the two databases namesnew.csv and passwordsnew.csv consisting of the entered data it sets the default marks of new student accounts as zero to avoid discrepancies.

root=Tk()

root. title('Login')

root.geometry('925x500+300+200')

root.configure(bg="#fff")

root. resizable(False,False)

#Image-----------------------------------

img=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/Signup window/bgg.png')

Label(root,image=img,bg='white').place(x=0,y=0)

#Login Frame-----------------------------

frame=Frame(root,width=350,height=350,bg="white")

frame.place(x=520,y=80)

#Header Text-----------------------------

head=Label(frame,text='Sign Up',fg='#d2232a',bg='white',font=('Bahnschrift',23,'bold'))

head.place(x=125,y=10 )

#Name-------------------------------------

def on\_enter(e):

    # Move focus to next Entry widget

    e.widget.tk\_focusNext().focus()

def on\_focus\_in\_sap(event):

       n = event.widget

       if n.get() == 'First Name':

        n.delete(0, 'end')

def on\_focus\_out\_sap(event):

    ln=event.widget

    if n.get() =="" :

     n.insert(0, 'First Name')

     n.config(show="")

n=Entry(frame, width=25, fg="black", border=0, bg="white", font=('Bahnschrift',12))

n.bind("<FocusIn>", on\_focus\_in\_sap)

n.bind("<FocusOut>", on\_focus\_out\_sap)

n.bind("<Return>", on\_enter)

n.place(x=30,y=83)

n.insert(0,'First Name')

Frame(frame,width=295,height=2,bg='black').place(x=25,y=110)

#Last Name-------------------------------------

def on\_enter(e):

    # Move focus to next Entry widget

    e.widget.tk\_focusNext().focus()

def on\_focus\_in\_l(event):

       ln = event.widget

       if ln.get() == 'Last Name':

        ln.delete(0, 'end')

def on\_focus\_out\_l(event):

    sapin=event.widget

    if ln.get() =="" :

     ln.insert(0, 'Last Name')

     ln.config(show="")

ln=Entry(frame, width=25, fg="black", border=0, bg="white", font=('Bahnschrift',12))

ln.bind("<FocusIn>", on\_focus\_in\_l)

ln.bind("<FocusOut>", on\_focus\_out\_l)

ln.bind("<Return>", on\_enter)

ln.place(x=30,y=130)

ln.insert(0,'Last Name')

Frame(frame,width=295,height=2,bg='black').place(x=25,y=157)

#SapID-----------------------------------

def on\_enter(e):

    # Move focus to next Entry widget

    e.widget.tk\_focusNext().focus()

def on\_focus\_in\_sap(event):

       sapin = event.widget

       if sapin.get() == 'SapID':

        sapin.delete(0, 'end')

def on\_focus\_out\_sap(event):

    pw=event.widget

    if sapin.get() =="" :

     sapin.insert(0, 'SapID')

     sapin.config(show="")

sapin=Entry(frame, width=25, fg="black", border=0, bg="white", font=('Bahnschrift',12))

sapin.bind("<FocusIn>", on\_focus\_in\_sap)

sapin.bind("<FocusOut>", on\_focus\_out\_sap)

sapin.bind("<Return>", on\_enter)

sapin.place(x=30,y=175)

sapin.insert(0,'SapID')

Frame(frame,width=295,height=2,bg='black').place(x=25,y=202)

#Password--------------------------------

def on\_submit(e):

    # Execute function when Return key is pressed in the last Entry widget

    print("Function executed!")

    signup()

def on\_focus\_in(event):

       pw = event.widget

       if pw.get() == 'Password':

        pw.delete(0, 'end')

        pw.config(show='\*')

def on\_focus\_out(event):

    pw=event.widget

    if pw.get() =="" :

     pw.insert(0, 'Password')

     pw.config(show="")

pw=Entry(frame, width=25, fg="black", border=0, bg="white", font=('Bahnschrift',12))

pw.bind("<FocusIn>", on\_focus\_in)

pw.bind("<FocusOut>", on\_focus\_out)

pw.bind("<Return>", on\_submit)

pw.place(x=30,y=225)

pw.insert(0,'Password')

Frame(frame,width=295,height=2,bg='black').place(x=25,y=252)

#Buton-----------------------------------

ab=Button(frame,width=39,pady=7,text='Sign up',bg='#d2232a',fg='white',border=0,cursor='hand2',command=signup)

ab.place(x=35,y=279)

# Function to change button color on hover

def on\_hover(event):

    ab['background'] = '#9f1a1f'

    ab['foreground'] = 'white'

    ab['font'] = 'Bahnschrift 9 underline'

# Function to change button color when not hovering

def off\_hover(event):

    ab['background'] = '#d2232a'

    ab['foreground'] = 'white'

    ab['font'] = 'Bahnschrift 9'

# Bind hover events to button

ab.bind('<Enter>', on\_hover)

ab.bind('<Leave>', off\_hover)

root.mainloop()

subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/App/adminconsole.py"])

This code snippet is contains all the Gui elements of the signup window along with the various event driven functions for each element.

Signup\_Tea.py:

from tkinter import \*

from tkinter import messagebox

import ast

#\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-Login Database\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

import pandas as pd

# Define file names

password\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/passwords -t.csv"

name\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/names -t.csv"

# Load data from files

passwords\_df = pd.read\_csv(password\_file, index\_col="Sapid")

names\_df = pd.read\_csv(name\_file, index\_col="Sapid")

#\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-Signup logic\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

import subprocess

def run\_script():

    root.withdraw()

    # Replace "other\_script.py" with the filename of the other Python script

    subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/Python Ultimate Student interface.py"])

This runs the original Sign in window after the signup process is done while withdrawing current window.

def signup():

    # Get user input

    name = n.get()

    Sapid = sapin.get()

    password = pw.get()

    # Check if roll\_no already exists in the file

    if Sapid in passwords\_df.index:

        messagebox.showerror("Invalid", "Account for this SAP ID already exists")

        print("Roll No. already exists. Please try again.")

        return

    # Add new student to dataframes

    passwords\_df.loc[Sapid] = password

    names\_df.loc[Sapid] = name

    # Save data to files

    passwords\_df.to\_csv(password\_file)

    names\_df.to\_csv(name\_file)

    print("Signup Successful. Welcome, ", name)

    messagebox.showinfo("Account Created", "Now head onto login page to proceed")

    run\_script()

The function here is used for signup, it retrieves the data entered in the entry fields.

it checks if the roll no entered in the field already exists in the data base if it does it will return an error saying “ACCOUNT ALREADY EXISTS”

If it does not, then it creates a new data field in the two databases names-t.csv and passwords-t.csv consisting of the entered data.

root=Tk()

root. title('Teacher Sign Up')

root.geometry('925x500+300+200')

root.configure(bg="#fff")

root. resizable(False,False)

#Image-----------------------------------

img=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/bgt.png')

Label(root,image=img,bg='white').place(x=0,y=0)

#Login Frame-----------------------------

frame=Frame(root,width=350,height=350,bg="white")

frame.place(x=520,y=80)

#Header Text-----------------------------

head=Label(frame,text='Sign Up',fg='#23aaf2',bg='white',font=('Bahnschrift',23,'bold'))

head.place(x=125,y=10 )

#Name-------------------------------------

def on\_enter(e):

    # Move focus to next Entry widget

    e.widget.tk\_focusNext().focus()

def on\_focus\_in\_sap(event):

       n = event.widget

       if n.get() == 'Name':

        n.delete(0, 'end')

def on\_focus\_out\_sap(event):

    sapin=event.widget

    if n.get() =="" :

     n.insert(0, 'Name')

     n.config(show="")

n=Entry(frame, width=25, fg="black", border=0, bg="white", font=('Bahnschrift',12))

n.bind("<FocusIn>", on\_focus\_in\_sap)

n.bind("<FocusOut>", on\_focus\_out\_sap)

n.bind("<Return>", on\_enter)

n.place(x=30,y=85)

n.insert(0,'Name')

Frame(frame,width=295,height=2,bg='black').place(x=25,y=112)

#SapID-----------------------------------

def on\_enter(e):

    # Move focus to next Entry widget

    e.widget.tk\_focusNext().focus()

def on\_focus\_in\_sap(event):

       sapin = event.widget

       if sapin.get() == 'SapID':

        sapin.delete(0, 'end')

def on\_focus\_out\_sap(event):

    pw=event.widget

    if sapin.get() =="" :

     sapin.insert(0, 'SapID')

     sapin.config(show="")

sapin=Entry(frame, width=25, fg="black", border=0, bg="white", font=('Bahnschrift',12))

sapin.bind("<FocusIn>", on\_focus\_in\_sap)

sapin.bind("<FocusOut>", on\_focus\_out\_sap)

sapin.bind("<Return>", on\_enter)

sapin.place(x=30,y=155)

sapin.insert(0,'SapID')

Frame(frame,width=295,height=2,bg='black').place(x=25,y=182)

#Password--------------------------------

def on\_submit(e):

    # Execute function when Return key is pressed in the last Entry widget

    print("Function executed!")

    signup()

def on\_focus\_in(event):

       pw = event.widget

       if pw.get() == 'Password':

        pw.delete(0, 'end')

        pw.config(show='\*')

def on\_focus\_out(event):

    pw=event.widget

    if pw.get() =="" :

     pw.insert(0, 'Password')

     pw.config(show="")

pw=Entry(frame, width=25, fg="black", border=0, bg="white", font=('Bahnschrift',12))

pw.bind("<FocusIn>", on\_focus\_in)

pw.bind("<FocusOut>", on\_focus\_out)

pw.bind("<Return>", on\_submit)

pw.place(x=30,y=225)

pw.insert(0,'Password')

Frame(frame,width=295,height=2,bg='black').place(x=25,y=252)

#Buton-----------------------------------

ab=Button(frame,width=39,pady=7,text='Sign up',bg='#23aaf2',fg='white',border=0,cursor='hand2',command=signup)

ab.place(x=35,y=279)

# Function to change button color on hover

def on\_hover(event):

    ab['background'] = '#1b86bf'

    ab['foreground'] = 'white'

    ab['font'] = 'Bahnschrift 9 underline'

# Function to change button color when not hovering

def off\_hover(event):

    ab['background'] = '#23aaf2'

    ab['foreground'] = 'white'

    ab['font'] = 'Bahnschrift 9'

# Bind hover events to button

ab.bind('<Enter>', on\_hover)

ab.bind('<Leave>', off\_hover)

root.mainloop()

subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/App/adminconsole.py"])

This code snippet is contains all the Gui elements of the signup window along with the various event driven functions for each element. Also closing this window leads back to signin window

App\_Stu.py

from tkinter import \*

import pandas as pd

import matplotlib.pyplot as plt

import subprocess

import webbrowser

from pop import calculate\_gpa

from pop import grade

from pop import percent

from tkinter import messagebox

def plot():

    data = pd.read\_csv("C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/namesnew.csv")

    with open("C:/Users/DELL/Desktop/PPS Code projects/Final project/Temp.txt", "r") as file1:

        roll\_no=file1.read()

    student\_data = data[data["Roll No."] == roll\_no][["LADE","QSP","EOB","PEM"]]

    ax = student\_data.plot(kind="bar")

    n=names\_df.loc[roll\_no,"Name"]

    l=names\_df.loc[roll\_no,"Last Name"]

    # Set the title and axis labels

    ax.set\_title(f"{n} {l} Marks")

    ax.set\_xlabel("Subject")

    ax.set\_ylabel("Marks")

    ax.set\_ylim([0, 50])

    # Display the graph

    plt.show()

#Home frame------------------------------------------------------------------------------------------------------

def create\_home\_frame(master, go\_to\_next\_frame\_func,go\_chat\_func,im1,im2,plot):

    home\_frame = Frame(master, width=925,height=425)

    home\_frame.place(x=0,y=75)

    home\_frame.config(bg="white")

    def on\_hover(event):

        home\_button['background'] = '#9f1a1f'

        home\_button['foreground'] = 'white'

        home\_button['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hover(event):

        home\_button['background'] = '#d2232a'

        home\_button['foreground'] = 'white'

        home\_button['font'] = 'Bahnschrift 9'

    def butt():

      go\_to\_next\_frame\_func()

      plot()

    home\_button = Button(home\_frame,width=20,pady=7, text="See Your report",bg='#d2232a',fg='white',border=0,cursor='hand2', command=butt)

    home\_button.place(x=160, y=150)

    home\_button.bind('<Enter>', on\_hover)

    home\_button.bind('<Leave>', off\_hover)

    Label(home\_frame,image=im1,bg="white").place(x=180,y=25)

    def on\_hover2(event):

        home\_button2['background'] = '#9f1a1f'

        home\_button2['foreground'] = 'white'

        home\_button2['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hover2(event):

        home\_button2['background'] = '#d2232a'

        home\_button2['foreground'] = 'white'

        home\_button2['font'] = 'Bahnschrift 9'

    home\_button2 = Button(home\_frame,width=20,pady=7, text="Chat with Mentors",bg='#d2232a',fg='white',border=0,cursor='hand2', command=go\_chat\_func)

    home\_button2.place(x=390, y=150)

    home\_button2.bind('<Enter>', on\_hover2)

    home\_button2.bind('<Leave>', off\_hover2)

    Label(home\_frame,image=im2,bg="white").place(x=410,y=25)

    #notes

    def noteslink():

        webbrowser.open\_new("https://drive.google.com/drive/u/1/folders/1Z9lZ-tZjlKEKiKGoHGt6QAx07W5cj1KA")

    def on\_hover3(event):

        notes['background'] = '#9f1a1f'

        notes['foreground'] = 'white'

        notes['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hover3(event):

        notes['background'] = '#d2232a'

        notes['foreground'] = 'white'

        notes['font'] = 'Bahnschrift 9'

    notes = Button(home\_frame,width=20,pady=7, text="Lecture Notes",bg='#d2232a',fg='white',border=0,cursor='hand2', command=noteslink)

    notes.place(x=620, y=150)

    notes.bind('<Enter>', on\_hover3)

    notes.bind('<Leave>', off\_hover3)

    Label(home\_frame,image=im5,bg="white").place(x=640,y=25)

    #here is time table

    def ttlink():

        webbrowser.open\_new("https://cdn.discordapp.com/attachments/1026538138203594793/1100841880075378768/9e4ad7d4-c31d-4b70-b41d-929aa9d0b4d6.png")

    def on\_hover3(event):

        tt['background'] = '#9f1a1f'

        tt['foreground'] = 'white'

        tt['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hover3(event):

        tt['background'] = '#d2232a'

        tt['foreground'] = 'white'

        tt['font'] = 'Bahnschrift 9'

    tt = Button(home\_frame,width=20,pady=7, text="Class Time Table",bg='#d2232a',fg='white',border=0,cursor='hand2', command=ttlink)

    tt.place(x=270, y=350)

    tt.bind('<Enter>', on\_hover3)

    tt.bind('<Leave>', off\_hover3)

    Label(home\_frame,image=im6,bg="white").place(x=290,y=225)

    #here is handbook

    def hblink():

        webbrowser.open\_new("https://svkmmumbai-my.sharepoint.com/personal/kasturi\_shirodkar\_nmims\_edu/\_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fkasturi%5Fshirodkar%5Fnmims%5Fedu%2FDocuments%2FKasturi%20%2D%20MPSTME%2FStudent%20Information%20Handout%20A%2EY%2E%202022%20%2D23&ga=1")

    def on\_hover3(event):

        hb['background'] = '#9f1a1f'

        hb['foreground'] = 'white'

        hb['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hover3(event):

        hb['background'] = '#d2232a'

        hb['foreground'] = 'white'

        hb['font'] = 'Bahnschrift 9'

    hb = Button(home\_frame,width=20,pady=7, text="Student Handbook",bg='#d2232a',fg='white',border=0,cursor='hand2', command=hblink)

    hb.place(x=510, y=350)

    hb.bind('<Enter>', on\_hover3)

    hb.bind('<Leave>', off\_hover3)

    Label(home\_frame,image=im7,bg="white").place(x=530,y=225)

    return home\_frame

This function creates a home frame containing all the navigational buttons to secondary frames and subframes like Results frame and Communications frame along with buttons for opening timetable, notes, and student handbook

#second\_frame------------------------------------------------------------------------

def create\_second\_frame(master, go\_back\_func,marks):

    second\_frame = Frame(master, width=925,height=425)

    second\_frame.config(bg="white")

    def on\_hover(event):

        back\_button['background'] = '#9f1a1f'

        back\_button['foreground'] = 'white'

        back\_button['font'] = 'Bahnschrift 12 bold'

    # Function to change button color when not hovering

    def off\_hover(event):

        back\_button['background'] = '#d2232a'

        back\_button['foreground'] = 'white'

        back\_button['font'] = 'Bahnschrift 12'

    back\_button = Button(second\_frame,padx=3,pady=1, text="<",font='Bahnschrift 12',bg='#d2232a',fg='white',border=0,cursor='hand2', command=go\_back\_func)

    back\_button.place(x=1, y=1)

    back\_button.bind('<Enter>', on\_hover)

    back\_button.bind('<Leave>', off\_hover)

    Label(second\_frame,text="Your Report Card ",fg='#d2232a',bg='white',font=('Bahnschrift',23,'bold')).place(x=350,y=40)

    Label(second\_frame,text="LADE: "+str(marks[0]),bg='white',font=('Bahnschrift',15) ).place(x=350,y=80)

    Label(second\_frame,text="QSP: "+str(marks[1]),bg='white',font=('Bahnschrift',15) ).place(x=350,y=120)

    Label(second\_frame,text="EOB: "+str(marks[2]),bg='white',font=('Bahnschrift',15) ).place(x=350,y=160)

    Label(second\_frame,text="PEM: "+str(marks[3]),bg='white',font=('Bahnschrift',15)).place(x=350,y=200)

    perc=percent(marks[0],marks[1],marks[2],marks[3])

    Label(second\_frame,text="Percentage: "+perc,bg='white',font=('Bahnschrift',15)).place(x=350,y=240)

    gpa=calculate\_gpa(marks[0],marks[1],marks[2],marks[3])

    g=str(gpa)

    Label(second\_frame,text="Your GPA: "+g,bg='white',font=('Bahnschrift',15)).place(x=350,y=280)

    grd= grade(gpa)

    gg= Label(second\_frame,text="Grade: "+grd,bg='white',font=('Bahnschrift',15))

    gg.place(x=500,y=280)

    if grd == "F" or grd == "D-" or grd == "D" or grd == "D+":

        gg['fg'] = '#d2232a'

    else:

        gg['fg'] = '#04e448'

    return second\_frame

def go\_back(second\_frame, home\_frame):

    second\_frame.place\_forget()

    home\_frame.place(x=0,y=75)

def go\_to\_next\_frame(home\_frame, second\_frame):

    home\_frame.place\_forget()

    second\_frame.place(x=0,y=75)

This function creates the second frame which takes the home frame as an argument to implement the back button which takes you back to the home screen. The second frame reads the roll\_no of the user from the temp.txt and retrieves the marks of the student from the namesnew.csv file

#frame3------------------------------------------------------------------------------

def create\_frame3(master, go\_back\_home\_func):

    frame3 = Frame(master, width=925,height=425)

    frame3.config(bg="white")

    def on\_hover(event):

        back\_button['background'] = '#9f1a1f'

        back\_button['foreground'] = 'white'

        back\_button['font'] = 'Bahnschrift 12 bold'

    # Function to change button color when not hovering

    def off\_hover(event):

        back\_button['background'] = '#d2232a'

        back\_button['foreground'] = 'white'

        back\_button['font'] = 'Bahnschrift 12'

    back\_button = Button(frame3,padx=3,pady=1, text="<",font='Bahnschrift 12',bg='#d2232a',fg='white',border=0,cursor='hand2', command=go\_back\_home\_func)

    back\_button.place(x=1, y=1)

    back\_button.bind('<Enter>', on\_hover)

    back\_button.bind('<Leave>', off\_hover)

    Label(frame3,image=im3,bg="white").place(x=250,y=100)

    def on\_hoverl(event):

        launche['background'] = '#9f1a1f'

        launche['foreground'] = 'white'

        launche['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hoverl(event):

        launche['background'] = '#d2232a'

        launche['foreground'] = 'white'

        launche['font'] = 'Bahnschrift 9'

    def on\_hover2(event):

        start['background'] = '#9f1a1f'

        start['foreground'] = 'white'

        start['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hover2(event):

        start['background'] = '#d2232a'

        start['foreground'] = 'white'

        start['font'] = 'Bahnschrift 9'

    def startchat():

        webbrowser.open\_new("http://127.0.0.1:7860/")

    start = Button(frame3,width=20,pady=7, text="Start Chatting",bg='#d2232a',fg='white',border=0,cursor='hand2', command=startchat)

    start.bind('<Enter>', on\_hover2)

    start.bind('<Leave>', off\_hover2)

    def host():

        import time

        launche['text']="Hosting..."

        subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/Chat/API Test.py"])

        time.sleep(10)

        launche.place\_forget()

        start.place(x=230, y=225)

    launche = Button(frame3,width=20,pady=7, text="Chat with Mentors",bg='#d2232a',fg='white',border=0,cursor='hand2', command=host)

    launche.place(x=230, y=225)

    launche.bind('<Enter>', on\_hoverl)

    launche.bind('<Leave>', off\_hoverl)

    Label(frame3,image=im4,bg="white").place(x=585,y=100)

    def chatter():

        subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/App/Chatroom.py"])

    def on\_hoverl(event):

        talk['background'] = '#9f1a1f'

        talk['foreground'] = 'white'

        talk['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hoverl(event):

        talk['background'] = '#d2232a'

        talk['foreground'] = 'white'

        talk['font'] = 'Bahnschrift 9'

    talk = Button(frame3,width=20,pady=7, text="Chat with others",bg='#d2232a',fg='white',border=0,cursor='hand2', command=chatter)

    talk.place(x=580, y=225)

    talk.bind('<Enter>', on\_hoverl)

    talk.bind('<Leave>', off\_hoverl)

    return frame3

def go\_back\_home(frame3, home\_frame):

    frame3.place\_forget()

    home\_frame.place(x=0,y=75)

def go\_chat(home\_frame, frame3):

    home\_frame.place\_forget()

    frame3.place(x=0,y=75)

The code defines a function create\_frame3() that creates a tkinter frame with buttons and labels for a chat application. The buttons have hover effects, changing their color and font when the user hovers over them. There are three buttons: "Back" to return to the home screen, "Start Chatting" to start a chat session with a mentor, and "Chat with others" to enter a public chat room. The function also defines two other functions go\_back\_home() and go\_chat() that hide and show the home screen and chat frame, respectively.Clicking the chat with mentors button starts a subprocess that runs API Test.py that creates and hosts our gradio application for our chatbot. The chat with others button starts a subprocess and runs our chatroom.py program

#Database load----------------------------------------------

# Define file names

password\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/passwordsnew.csv"

name\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/namesnew.csv"

# Load data from files

passwords\_df = pd.read\_csv(password\_file, index\_col="Roll No.")

names\_df = pd.read\_csv(name\_file, index\_col="Roll No.")

with open("C:/Users/DELL/Desktop/PPS Code projects/Final project/Temp.txt", "r") as file1:

    roll\_no=file1.read()

marks=(names\_df.loc[roll\_no,"LADE"],names\_df.loc[roll\_no,"QSP"],names\_df.loc[roll\_no,"EOB"],names\_df.loc[roll\_no,"PEM"])

#Top bar Content------------------------------

import datetime

# Get the current time

now = datetime.datetime.now()

# Determine the greeting based on the time of day

if now.hour < 12:

    greeting = "Good morning, "

elif now.hour < 18:

    greeting = "Good afternoon, "

else:

    greeting = "Good evening, "

# Print the greeting

print(greeting)

app=Tk()

app.title("MPSTME Student Interface")

app.geometry('925x500+300+200')

app.resizable(False,False)

hi=Label(app,text=greeting+names\_df.loc[roll\_no, "Name"]+' '+names\_df.loc[roll\_no, "Last Name"],font=('Bahnschrift',20,'bold'))

hi.place(x=300,y=20)

img=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/log.png')

Label(app,image=img).place(x=25,y=10)

im1=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/report.png')

im2=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/speech.png')

im3=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/question.png')

im4=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/stuchat.png')

im5=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/books.png')

im6=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/timetable.png')

im7=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/information.png')

def create\_marquee(window, text, width=400):

    # Define the marquee label

    marquee\_label = Label(window, text=text,fg="#d2232a", font=('Bahnschrift', 8))

    marquee\_label.pack(side=LEFT)

    # Define the marquee scrolling function

    def scroll\_marquee():

        # Get the current position of the marquee label

        x, y = marquee\_label.winfo\_x(), marquee\_label.winfo\_y()

        # Check if the label is out of bounds

        if x < -marquee\_label.winfo\_width():

            x = window.winfo\_width()

        # Move the label to the left

        marquee\_label.place(x=x-width, y=57)

        marquee\_label.after(120, scroll\_marquee)

    # Start scrolling the marquee

    scroll\_marquee()

with open("C:/Users/DELL/Desktop/PPS Code projects/Final project/App/Mar.txt", "r") as file1:

        txt=file1.read()

# Create the marquee

create\_marquee(app,txt, 30)

home\_frame = create\_home\_frame(app, lambda: go\_to\_next\_frame(home\_frame, second\_frame),lambda:go\_chat(home\_frame, frame3),im1,im2,lambda:plot())

second\_frame = create\_second\_frame(app, lambda: go\_back(second\_frame, home\_frame),marks)

frame3= create\_frame3(app,lambda:go\_back\_home(frame3, home\_frame))

logs=open('C:/Users/DELL/Desktop/PPS Code projects/Final project/his.txt','a+')

now=datetime.datetime.now()

print(now)

logentry=str(now)+" Student: "+str(roll\_no)+" has logged in\n"

print(logentry)

logs.write(logentry)

logs.close()

time=str(now)

role="Student"

activity="logged in"

logfile='C:/Users/DELL/Desktop/PPS Code projects/Final project/logs.csv'

log\_df=pd.read\_csv(logfile,index\_col="Time")

log\_df.loc[time]=[roll\_no,role,activity]

log\_df.to\_csv(logfile)

def on\_hover2(event):

    logout['background'] = '#9f1a1f'

    logout['foreground'] = 'white'

    logout['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

def off\_hover2(event):

    logout['background'] = '#d2232a'

    logout['foreground'] = 'white'

    logout['font'] = 'Bahnschrift 9'

def loglogout():

    time=str(now)

    role="Student"

    activity="logged out"

    log\_df.loc[time]=[roll\_no,role,activity]

    log\_df.to\_csv(logfile)

    messagebox.showinfo("Logged out","Logged out successfully")

    app.withdraw()

    subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/Python Ultimate Student interface.py"])

logout = Button(app,width=10,pady=2, text="Logout",bg='#d2232a',fg='white',border=0,cursor='hand2', command=loglogout)

logout.place(x=860, y=20)

logout.bind('<Enter>', on\_hover2)

logout.bind('<Leave>', off\_hover2)

app.mainloop()

This code defines a function create\_marquee which creates a scrolling marquee label in a GUI window. The function takes three arguments: window which is the GUI window in which the marquee label will be displayed, text which is the text that will be displayed in the marquee label, and width which is the width of the marquee label.

Inside the function, the marquee\_label is defined as a Label object with the given text and some specific formatting. The label is then packed to the left of the window.

The scroll\_marquee function is defined to scroll the marquee\_label horizontally from right to left by changing its x-coordinate. The after method is used to repeatedly call the scroll\_marquee function after a delay of 120ms, creating the scrolling effect.

The create\_marquee function is then called with the window, text, and width arguments to create the marquee label.

The code also includes some additional functionality for logging in and logging out a student, and a button to initiate the logout process.

pop.py (Calculator library)

def calculate\_gpa(lade\_marks, qsp\_marks, eob\_marks, pem\_marks):

    lade\_grade = "F" if lade\_marks < 20 else "D" if lade\_marks < 35 else "C" if lade\_marks < 45 else "B" if lade\_marks < 50 else "A"

    qsp\_grade = "F" if qsp\_marks < 20 else "D" if qsp\_marks < 35 else "C" if qsp\_marks < 45 else "B" if qsp\_marks < 50 else "A"

    eob\_grade = "F" if eob\_marks < 20 else "D" if eob\_marks < 35 else "C" if eob\_marks < 45 else "B" if eob\_marks < 50 else "A"

    pem\_grade = "F" if pem\_marks < 20 else "D" if pem\_marks < 35 else "C" if pem\_marks < 45 else "B" if pem\_marks < 50 else "A"

    lade\_points = 0 if lade\_grade == "F" else 1 if lade\_grade == "D" else 2 if lade\_grade == "C" else 3 if lade\_grade == "B" else 4

    qsp\_points = 0 if qsp\_grade == "F" else 1 if qsp\_grade == "D" else 2 if qsp\_grade == "C" else 3 if qsp\_grade == "B" else 4

    eob\_points = 0 if eob\_grade == "F" else 1 if eob\_grade == "D" else 2 if eob\_grade == "C" else 3 if eob\_grade == "B" else 4

    pem\_points = 0 if pem\_grade == "F" else 1 if pem\_grade == "D" else 2 if pem\_grade == "C" else 3 if pem\_grade == "B" else 4

    total\_credits = 4 + 3 + 3 + 3

    total\_points = lade\_points \* 4 + qsp\_points \* 3 + eob\_points \* 3 + pem\_points \* 3

    gpa = total\_points / total\_credits

    approxgpa=round(gpa,2)

    return approxgpa

def percent(lade\_marks, qsp\_marks, eob\_marks, pem\_marks):

    total = lade\_marks + qsp\_marks + eob\_marks + pem\_marks

    perc= str((total/200)\*100)

    pc= "%"

    percentage=str(perc+pc)

    return percentage

def grade(gpa):

    grade = "F" if gpa<=1.49 else "D" if gpa<=1.99 else "C-" if gpa<=2.24 else "C" if gpa<=2.49  else "C+" if gpa<=2.74 else "B-" if gpa<=2.99 else "B" if gpa<=3.24  else "B+" if gpa<=3.49 else "A-" if gpa<=3.74 else "A" if gpa<=3.9  else "A+"

    return grade

This library provides three functions:

calculate\_gpa(lade\_marks, qsp\_marks, eob\_marks, pem\_marks): This function takes in four arguments, which are the marks obtained by a student in four different subjects. It calculates the grade obtained by the student in each subject, based on a predefined grading system (F, D, C, B, A), and assigns a corresponding point value (0, 1, 2, 3, 4) to each grade. It then calculates the student's GPA (Grade Point Average) by weighting the point values by the credits assigned to each subject. The function returns the GPA rounded to two decimal places.

percent(lade\_marks, qsp\_marks, eob\_marks, pem\_marks): This function takes in the same four arguments as the previous function, and calculates the percentage of total marks obtained by the student across all four subjects. The function returns the percentage as a string with a "%" sign appended.

grade(gpa): This function takes in a single argument, which is the GPA calculated by the previous function. It assigns a letter grade to the GPA based on a predefined grading system (A+, A, A-, B+, B, B-, C+, C, C-, D, F). The function returns the letter grade.

API Test.py

import os

import openai

import gradio as gr

from pop import getresponsegpt

openai.api\_key = "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"

start\_sequence = "\nAI:"

restart\_sequence = "\nHuman: "

prompt = "The following is a conversation with an AI assistant for the students of MPSTME. The assistant is helpful, creative, clever, and very friendly and helps the students of MPSTME.\n\nHuman: Hello, who are you?\nAI: I am Chatbot built to Assist Students of MPSTME in their subject doubts. How can I help you today?\nHuman: "

def openai\_create(prompt):

    response=getresponsegpt(prompt)

    return response

def chatgpt\_clone(input, history):

    history = history or []

    s = list(sum(history, ()))

    s.append(input)

    inp = ' '.join(s)

    output = openai\_create(inp)

    history.append((input, output))

    return history, history

block = gr.Blocks(theme=gr.themes.Soft(font=[gr.themes.GoogleFont("Bahnschrift")]))

with block:

    gr.Markdown("""<head> <title>MPSTME Chatbot</title>

    <img src="https://th.bing.com/th/id/R.7c0f694e69f77954e20f846a71b987ad?rik=r%2bDu860FJIOysA&riu=http%3a%2f%2fengineering.nmims.edu%2fimages%2fmpstme\_main-logo-horizontal.jpg&ehk=Tp1EF7L%2bfbpBY8FxtoMuu1saiMxxd11Yedr382WlNbU%3d&risl=&pid=ImgRaw&r=0">

    <h1><center>NMIMS Mentor Bot</center></h1></head>

    <h3>Hello I am Chatbot built to Assist Students in their subject doubts</h3>

    """)

    chatbot = gr.Chatbot()

    message = gr.Textbox(placeholder="Ask a doubt")

    state = gr.State()

    submit = gr.Button("SEND",variant="primary")

    submit.click(chatgpt\_clone, inputs=[message, state], outputs=[chatbot, state])

block.launch(debug = True)

This code imports the necessary libraries/modules for building a conversational chatbot using OpenAI's GPT-3 model. It sets the OpenAI API key, defines a start and restart sequence for the conversation, and creates a prompt for initiating the conversation.

The openai\_create function uses the GPT-3 model to generate a response to a given prompt. The chatgpt\_clone function takes user input and conversation history as input, concatenates them into a single prompt, calls openai\_create to generate a response, and updates the conversation history.

The code then creates a Gradio interface with a text input for the user to ask a question and a button to submit the question. When the user clicks the submit button, the chatgpt\_clone function is called to generate a response based on the user input and conversation history. The response is then displayed in the chatbot interface along with the updated conversation history.

pop.py (Prompt processing and response retrieval)

import requests,json

def getresponsegpt(question):

    urlgpt3\_5="https://api.openai.com/v1/chat/completions"

    headers\_dict=dict()

    headers\_dict['Content-Type']='application/json'

    headers\_dict['Authorization']= 'Bearer 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'

    data\_dict=dict()

    data\_dict["model"] = "gpt-3.5-turbo"

    data\_dict["messages"] =[{

            "role": "user",

            "content": question

            }]

    data\_dict=json.dumps(data\_dict)

    response = requests.post(url=urlgpt3\_5,headers=headers\_dict,data=data\_dict)

    response=response.json()

    required\_ans=response['choices'][0]['message']['content']

    return required\_ans

This code imports the necessary libraries/modules for building a conversational chatbot using OpenAI's GPT-3 model. It sets the OpenAI API key, defines a start and restart sequence for the conversation, and creates a prompt for initiating the conversation.

The openai\_create function uses the GPT-3 model to generate a response to a given prompt. The chatgpt\_clone function takes user input and conversation history as input, concatenates them into a single prompt, calls openai\_create to generate a response, and updates the conversation history.

The code then creates a Gradio interface with a text input for the user to ask a question and a button to submit the question. When the user clicks the submit button, the chatgpt\_clone function is called to generate a response based on the user input and conversation history. The response is then displayed in the chatbot interface along with the updated conversation history.

Chatroom.py

import tkinter as tk

import datetime

class ChatroomWindow:

    def \_\_init\_\_(self):

        self.root = tk.Tk()

        self.root.config(bg="#0b141a")

        self.root.title("Chatroom")

        # Create chat history text box

        self.chat\_history = tk.Text(self.root, state="disabled",font=('Bahnschrift',12),bg="#0b141a",fg="White")

        self.chat\_history.pack(side="top", fill="both", expand=True)

        # Create input box

        self.input\_frame = tk.Frame(self.root)

        self.input\_frame.pack(side="bottom", fill="x")

        self.input\_box = tk.Entry(self.input\_frame, font=('Bahnschrift', 12))

        self.input\_box.pack(side="left", fill="x", expand=True)

        self.input\_box.bind("<Return>", self.send)

        def on\_hover(event):

                self.button['background'] = '#0b8e32'

                self.button['foreground'] = 'white'

            # Function to change button color when not hovering

        def off\_hover(event):

                self.button['background'] = '#04e448'

                self.button['foreground'] = 'white'

        self.button = tk.Button(self.input\_frame, text="↪",font=('Bahnschrift', 12), bg="#04e448", fg="white",borderwidth=0, relief="solid", width=3, height=1,cursor='hand2',command=self.send)

        self.button.pack(side="left")

        self.button.bind("<Enter>",on\_hover)

        self.button.bind("<Leave>",off\_hover)

        self.chat\_log = []  # initialize chat log list

        # Load chat history from file

        try:

            with open("C:/Users/DELL/Desktop/PPS Code projects/Final project/App/chat\_history.txt", "r") as file:

                self.chat\_log = file.readlines()

                self.chat\_history.configure(state="normal")

                for line in self.chat\_log:

                    self.chat\_history.insert("end", line)

                self.chat\_history.configure(state="disabled")

        except FileNotFoundError:

            print("wtf")

            pass

    def send(self, event=None):

        # Get message from input box

        message = self.input\_box.get()

        # Create timestamp

        timestamp = datetime.datetime.now().strftime("[%d-%m %H:%M]")

        with open("C:/Users/DELL/Desktop/PPS Code projects/Final project/Temp.txt", "r") as file1:

            roll\_no=file1.read()

        # Add message to chat history

        self.chat\_history.configure(state="normal")

        self.chat\_history.insert("end", f"\n{timestamp}\n{roll\_no}: {message}")

        self.chat\_history.configure(state="disabled")

        def write\_to\_file(msg):

            with open("C:/Users/DELL/Desktop/PPS Code projects/Final project/App/chat\_history.txt", "a") as f:

                f.write(timestamp+"\n"+roll\_no+": "+ msg + "\n")

        write\_to\_file(message)

        # Clear input box

        self.input\_box.delete(0, "end")

    def run(self):

        self.root.mainloop()

chatroom = ChatroomWindow()

chatroom.run()

This code creates a simple chatroom window using the tkinter library. The code defines a class called ChatroomWindow which has an initializer method (\_\_init\_\_) and a run method.

In the initializer method, the code creates the main window for the chatroom using the Tk class from the tkinter library. The window is given a title and a background color. It also creates a text box to display the chat history, an input box for the user to type messages, and a button to send the message.

The try and except block loads the chat history from a file if it exists, and displays it in the chat history text box.

The send method is called when the user types a message and clicks the send button. It gets the message from the input box, creates a timestamp using the datetime library, and writes the message to the chat history text box along with the timestamp and the user's name. It also writes the message to a file.

The run method is called to start the chatroom application and runs the main loop of the tkinter window, which allows the window to respond to user events.

App\_Tea.py

from tkinter import \*

from tkinter import messagebox

import ast

import subprocess

from pop import calculate\_gpa

from pop import percent

from pop import grade

#\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-Login Database\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_-\_

import pandas as pd

#Home frame------------------------------------------------------------------------------------------------------

def create\_home\_frame(master, go\_to\_next\_frame\_func,go\_chat\_func,im1,im2):

    home\_frame = Frame(master, width=925,height=425)

    home\_frame.place(x=0,y=75)

    home\_frame.config(bg="white")

    def on\_hover(event):

        home\_button['background'] = '#9f1a1f'

        home\_button['foreground'] = 'white'

        home\_button['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hover(event):

        home\_button['background'] = '#d2232a'

        home\_button['foreground'] = 'white'

        home\_button['font'] = 'Bahnschrift 9'

    def butt():

      go\_to\_next\_frame\_func()

    home\_button = Button(home\_frame,width=20,pady=7, text="View Sudent Data",bg='#d2232a',fg='white',border=0,cursor='hand2', command=butt)

    home\_button.place(x=230, y=225)

    home\_button.bind('<Enter>', on\_hover)

    home\_button.bind('<Leave>', off\_hover)

    def on\_hover2(event):

        home\_button2['background'] = '#9f1a1f'

        home\_button2['foreground'] = 'white'

        home\_button2['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hover2(event):

        home\_button2['background'] = '#d2232a'

        home\_button2['foreground'] = 'white'

        home\_button2['font'] = 'Bahnschrift 9'

    home\_button2 = Button(home\_frame,width=20,pady=7, text="Chat with Mentors",bg='#d2232a',fg='white',border=0,cursor='hand2', command=go\_chat\_func)

    home\_button2.place(x=570, y=225)

    home\_button2.bind('<Enter>', on\_hover2)

    home\_button2.bind('<Leave>', off\_hover2)

    Label(home\_frame,image=im1,bg="white").place(x=250,y=100)

    Label(home\_frame,image=im2,bg="white").place(x=600,y=100)

    def openchat():

        subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/App/Chatroom.py"])

    def on\_hover(event):

        chatr['image'] = chatroom2

    # Function to change button color when not hovering

    def off\_hover(event):

        chatr['image'] = chatroom1

    def butt():

      go\_to\_next\_frame\_func()

    chatr = Button(home\_frame,border=0,bg="white",image=chatroom1,cursor='hand2', command=openchat)

    chatr.place(x=850, y=350)

    chatr.bind('<Enter>', on\_hover)

    chatr.bind('<Leave>', off\_hover)

    return home\_frame

This is a Python function that creates a frame for a graphical user interface (GUI) application. The function takes in several parameters including the master widget, functions for navigating to the next frame and for chatting, as well as two image objects for display on the frame.

The function first creates a new frame with a specific width and height and places it at a specific location within the master widget. It sets the background color of the frame to white using the config() method.

The function then defines two functions (on\_hover and off\_hover) that will be used to change the appearance of the two button widgets when the user hovers over them. These functions modify the color and font properties of the buttons using various methods such as config().

The function then defines a third function (butt) that is called when one of the buttons is clicked. This function executes the go\_to\_next\_frame\_func function.

The function then creates two button widgets (home\_button and home\_button2) with specific properties such as size, text, color, font, and command to execute when clicked. These buttons are placed at specific locations within the frame using the place() method. The bind() method is also used to bind the on\_hover and off\_hover functions to these buttons, so that their appearance changes when the user hovers over them.

The function also creates two image widgets (im1 and im2) that are displayed on the frame using the place() method.

Finally, the function creates a third button widget (chatr) with a specific image, size, and position. This button is used to open a chatroom application when clicked, and the openchat() function is executed. Like the other buttons, the on\_hover and off\_hover functions are bound to this button using the bind() method.

The function returns the created home\_frame widget.

#second\_frame------------------------------------------------------------------------

def create\_second\_frame(master, go\_back\_func):

    second\_frame = Frame(master, width=925,height=425)

    second\_frame.config(bg="white")

    name\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/namesnew.csv"

    df = pd.read\_csv(name\_file, index\_col="Roll No.")

    def on\_hover(event):

        back\_button['background'] = '#9f1a1f'

        back\_button['foreground'] = 'white'

        back\_button['font'] = 'Bahnschrift 15 bold'

    # Function to change button color when not hovering

    def off\_hover(event):

        back\_button['background'] = '#d2232a'

        back\_button['foreground'] = 'white'

        back\_button['font'] = 'Bahnschrift 15'

    def clear():

        Name['text'] = ' '

        sapidi['text'] = ' '

        subject['text'] = ' '

        score['text'] = ' '

        lade['text'] = ' '

        ladesc['text'] = ' '

        qsp['text'] = ' '

        qspsc['text'] = ' '

        eob['text'] = ' '

        eobsc['text'] = ' '

        pem['text'] = ' '

        pemsc['text'] = ' '

        perc['text'] = ' '

        gpa['text'] = ' '

        grd['text'] = ' '

        editor.place\_forget()

        ledit.place\_forget()

        qedit.place\_forget()

        eedit.place\_forget()

        pedit.place\_forget()

        savebutton.place\_forget()

    def back():

        go\_back\_func()

        clear()

    Name=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    Name.place(x=250,y=100)

    sapidi=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    sapidi.place(x=500,y=100)

    subject=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    subject.place(x=300,y=140)

    score=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    score.place(x=550,y=140)

    lade=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    lade.place(x=300,y=180)

    ladesc=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    ladesc.place(x=560,y=180)

    qsp=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    qsp.place(x=300,y=200)

    qspsc=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    qspsc.place(x=560,y=200)

    eob=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    eob.place(x=300,y=220)

    eobsc=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    eobsc.place(x=560,y=220)

    pem=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    pem.place(x=300,y=240)

    pemsc=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    pemsc.place(x=560,y=240)

    perc=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    perc.place(x=300,y=260)

    gpa=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    gpa.place(x=300,y=280)

    grd=Label(second\_frame,text=" ",font=('Bahnschrift',12),bg="White")

    grd.place(x=380,y=280)

    back\_button = Button(second\_frame,width=3,pady=1, text="<",font='Bahnschrift 15',bg='#d2232a',fg='white',border=0,cursor='hand2', command=back)

    back\_button.place(x=1, y=1)

    back\_button.bind('<Enter>', on\_hover)

    back\_button.bind('<Leave>', off\_hover)

    ledit=Entry(second\_frame, width=2, fg="black", border=1, bg="white", font=('Bahnschrift',12))

    qedit=Entry(second\_frame, width=2, fg="black", border=1, bg="white", font=('Bahnschrift',12))

    eedit=Entry(second\_frame, width=2, fg="black", border=1, bg="white", font=('Bahnschrift',12))

    pedit=Entry(second\_frame, width=2, fg="black", border=1, bg="white", font=('Bahnschrift',12))

    def flower():

        ledit.lower()

        qedit.lower()

        eedit.lower()

        pedit.lower()

    def flift():

        ledit.lift()

        qedit.lift()

        eedit.lift()

        pedit.lift()

    def on\_enter(e):

        # Move focus to next Entry widget

        search()

    def on\_focus\_in\_sap(event):

           sapin = event.widget

           if sapin.get() == 'SapID':

            sapin.delete(0, 'end')

    def on\_focus\_out\_sap(event):

        pw=event.widget

        if sapin.get() =="" :

         sapin.insert(0, 'SapID')

         sapin.config(show="")

    sapin=Entry(second\_frame, width=32, fg="black", border=1, bg="white", font=('Bahnschrift',12))

    sapin.bind("<FocusIn>", on\_focus\_in\_sap)

    sapin.bind("<FocusOut>", on\_focus\_out\_sap)

    sapin.bind("<Return>", on\_enter)

    sapin.place(x=400,y=65)

    sapin.insert(0,'SapID')

    Frame(second\_frame,width=295,height=2,bg='black').place(x=400,y=89)

    Label(second\_frame,text="Enter Student SapID to view details",font=('Bahnschrift',12),bg="white").place(x=120,y=65)

    def search():

        clear()

        flower()

        editbut()

        roll\_no=sapin.get()

        if roll\_no not in df.index:

            messagebox.showerror("Invalid", "Entered SapID does not exist")

            return

        else:

            Name['text'] = str('Name: ' + df.loc[roll\_no, "Name"]+" "+df.loc[roll\_no, "Last Name"])

            sapidi['text'] = "SapID: "+roll\_no

            subject['text'] = "Subject"

            score['text'] = "Score"

            lade['text'] = 'LADE'

            ladesc['text'] = df.loc[roll\_no, "LADE"]

            qsp['text'] = 'QSP'

            qspsc['text'] = df.loc[roll\_no, "QSP"]

            eob['text'] = 'EOB'

            eobsc['text'] = df.loc[roll\_no, "EOB"]

            pem['text'] = 'PEM'

            pemsc['text'] = df.loc[roll\_no, "PEM"]

            pc= percent(df.loc[roll\_no, "LADE"],df.loc[roll\_no, "QSP"],df.loc[roll\_no, "EOB"],df.loc[roll\_no, "PEM"])

            perc['text'] = 'Percentage: '+pc

            ggp= calculate\_gpa(df.loc[roll\_no, "LADE"],df.loc[roll\_no, "QSP"],df.loc[roll\_no, "EOB"],df.loc[roll\_no, "PEM"])

            gp= str(ggp)

            gpa['text'] = 'GPA: '+gp

            ggr= grade(ggp)

            grd['text'] = 'Grade: '+ggr

    def on\_hoverf(event):

        find['background'] = '#9f1a1f'

        find['foreground'] = 'white'

        find['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

    def off\_hoverf(event):

       find['background'] = '#d2232a'

       find['foreground'] = 'white'

       find['font'] = 'Bahnschrift 9'

    find = Button(second\_frame,width=8,pady=4, text="Search",fg="white",bg='#d2232a',border=0,cursor='hand2', command=search)

    find.place(x=700,y=65)

    find.bind('<Enter>', on\_hoverf)

    find.bind('<Leave>', off\_hoverf)

    def savechanges():

        roll\_no=sapin.get()

        try:

            lmarks=int(ledit.get())

            qmarks=int(qedit.get())

            emarks=int(eedit.get())

            pmarks=int(pedit.get())

            if lmarks>50 or qmarks>50 or emarks>50 or pmarks>50:

                messagebox.showerror("Invalid Marks", "Only Enter Marks out 50")

                return

            else:

                confirmed = messagebox.askyesno("Confirmation", "Are you sure you want to save changes?")

                if confirmed:

                    messagebox.showwarning("Warning","This will change the marks of students")

                    print(lmarks+qmarks+emarks+pmarks)

                    df.loc[roll\_no, "LADE"] = lmarks

                    df.loc[roll\_no, "QSP"] = qmarks

                    df.loc[roll\_no, "EOB"] = emarks

                    df.loc[roll\_no, "PEM"] = pmarks

                    df.to\_csv(name\_file)

                    messagebox.showinfo("Success","Changes Saved successfully")

                    time=str(now)

                    role="Teacher"

                    activity="Changed marks of Student "+roll\_no

                    log\_df=pd.read\_csv(logfile,index\_col="Time")

                    log\_df.loc[time]=[Sapid,role,activity]

                    log\_df.to\_csv(logfile)

                    flower()

                    savebutton.place\_forget()

                    editor.place(x=700,y=100)

                    ladesc['text'] = df.loc[roll\_no, "LADE"]

                    qspsc['text'] = df.loc[roll\_no, "QSP"]

                    eobsc['text'] = df.loc[roll\_no, "EOB"]

                    pemsc['text'] = df.loc[roll\_no, "PEM"]

                    pc= percent(df.loc[roll\_no, "LADE"],df.loc[roll\_no, "QSP"],df.loc[roll\_no, "EOB"],df.loc[roll\_no, "PEM"])

                    perc['text'] = 'Percentage: '+pc

                    ggp= calculate\_gpa(df.loc[roll\_no, "LADE"],df.loc[roll\_no, "QSP"],df.loc[roll\_no, "EOB"],df.loc[roll\_no, "PEM"])

                    gp= str(ggp)

                    gpa['text'] = 'GPA: '+gp

                    ggr= grade(ggp)

                    grd['text'] = 'Grade: '+ggr

                else:

                    return

        except ValueError:

            messagebox.showerror("Invalid", "Enter only Integers for Marks")

            return

        print("botton")

    savebutton= Button(second\_frame,width=12,pady=4, text="Save Changes",fg="white",bg='#d2232a',border=0,cursor='hand2', command=savechanges)

    def edit():

        flift()

        editor.place\_forget()

        roll\_no=sapin.get()

        ledit.place(x=560,y=180)

        ledit.delete(0,'end')

        ledit.insert(0,df.loc[roll\_no, "LADE"])

        qedit.place(x=560,y=200)

        qedit.delete(0,'end')

        qedit.insert(0,df.loc[roll\_no, "QSP"])

        eedit.place(x=560,y=220)

        eedit.delete(0,'end')

        eedit.insert(0,df.loc[roll\_no, "EOB"])

        pedit.place(x=560,y=240)

        pedit.delete(0,'end')

        pedit.insert(0,df.loc[roll\_no, "PEM"])

        def on\_hovere(event):

            savebutton['background'] = '#9f1a1f'

            savebutton['foreground'] = 'white'

            savebutton['font'] = 'Bahnschrift 9 underline'

        # Function to change button color when not hovering

        def off\_hovere(event):

           savebutton['background'] = '#d2232a'

           savebutton['foreground'] = 'white'

           savebutton['font'] = 'Bahnschrift 9'

        savebutton.place(x=700,y=100)

        savebutton.bind('<Enter>', on\_hovere)

        savebutton.bind('<Leave>', off\_hovere)

    editor = Button(second\_frame,width=8,pady=4, text="Edit",fg="white",bg='#d2232a',border=0,cursor='hand2', command=edit)

    def editbut():

        def on\_hovere(event):

            editor['background'] = '#9f1a1f'

            editor['foreground'] = 'white'

            editor['font'] = 'Bahnschrift 9 underline'

        # Function to change button color when not hovering

        def off\_hovere(event):

           editor['background'] = '#d2232a'

           editor['foreground'] = 'white'

           editor['font'] = 'Bahnschrift 9'

        editor.place(x=700,y=100)

        editor.bind('<Enter>', on\_hovere)

        editor.bind('<Leave>', off\_hovere)

    return second\_frame

This is a Python function that creates a frame with various labels, buttons, and Entry widgets.

The create\_second\_frame() function takes two arguments, master and go\_back\_func. master is the tkinter parent widget in which the new frame will be created, and go\_back\_func is a function to call when the "back" button is clicked.

The function begins by creating a new frame with a width of 925 and a height of 425 pixels. The frame is set to have a white background color. It then reads a CSV file containing student names and stores the data in a Pandas dataframe.

The function then defines two nested functions, on\_hover and off\_hover, that change the appearance of the "back" button when the mouse hovers over it. The clear() function is also defined, which clears the text of various labels and hides several Entry widgets.

Next, the function creates several Label widgets to display information about a student's grades. These labels include the student's name, SAP ID, subject, score, letter grade, letter grade description, quiz score percentage, end-of-term score percentage, project score percentage, percentage grade, GPA, and letter grade.

After creating the labels, the function creates a "back" button using the tkinter Button widget. The button is positioned in the top-left corner of the frame and bound to the on\_hover and off\_hover functions to change its appearance when the mouse hovers over it.

Four Entry widgets are also created for the user to input new grades. These widgets include Entry boxes for the letter grade, quiz score, end-of-term score, and project score. The function defines two additional functions, flower() and flift(), that lower and lift the Entry widgets, respectively.

Finally, the function defines two more functions, on\_enter() and on\_focus\_in\_sap(), that handle focus and text deletion in the Entry widgets.

def go\_back(second\_frame, home\_frame):

    second\_frame.place\_forget()

    home\_frame.place(x=0,y=75)

def go\_to\_next\_frame(home\_frame, second\_frame):

    home\_frame.place\_forget()

    second\_frame.place(x=0,y=75)

#frame3------------------------------------------------------------------------------

def create\_frame3(master, go\_back\_home\_func):

    frame3 = Frame(master, width=925,height=425)

    frame3.config(bg="white")

    def on\_hover(event):

        back\_button['background'] = '#9f1a1f'

        back\_button['foreground'] = 'white'

        back\_button['font'] = 'Bahnschrift 15 bold'

    # Function to change button color when not hovering

    def off\_hover(event):

        back\_button['background'] = '#d2232a'

        back\_button['foreground'] = 'white'

        back\_button['font'] = 'Bahnschrift 15'

    back\_button = Button(frame3,width=3,pady=1, text="<",font='Bahnschrift 15',bg='#d2232a',fg='white',border=0,cursor='hand2', command=go\_back\_home\_func)

    back\_button.place(x=1, y=1)

    back\_button.bind('<Enter>', on\_hover)

    back\_button.bind('<Leave>', off\_hover)

    with open("C:/Users/DELL/Desktop/PPS Code projects/Final project/App/Mar.txt", "r") as file1:

        txt=file1.read()

    Noticeb=Text(frame3,border=2,height=6, width=50,font='Bahnschrift 12')

    Noticeb.place(x=250,y=100)

    Noticeb.insert("1.0",txt)

    def savenb():

        confirmed = messagebox.askyesno("Confirmation", "Are you sure you want to save changes?")

        if confirmed:

            messagebox.showwarning("Warning","This will Update the Notice board with new message")

            nb=Noticeb.get("1.0", "end")

            time=str(now)

            role="Teacher"

            activity="Updated Noticeboard "

            log\_df=pd.read\_csv(logfile,index\_col="Time")

            log\_df.loc[time]=[Sapid,role,activity]

            log\_df.to\_csv(logfile)

            nbfile=open('C:/Users/DELL/Desktop/PPS Code projects/Final project/App/Mar.txt','w+')

            nbfile.write(nb)

            nbfile.close

            messagebox.showinfo("Success","Changes Saved successfully")

        else:

            return

    def on\_hover(event):

        saveit['background'] = '#9f1a1f'

        saveit['foreground'] = 'white'

        saveit['font'] = 'Bahnschrift 12 underline'

    # Function to change button color when not hovering

    def off\_hover(event):

        saveit['background'] = '#d2232a'

        saveit['foreground'] = 'white'

        saveit['font'] = 'Bahnschrift 12'

    saveit = Button(frame3,padx=3,pady=1, text="Save Message",font='Bahnschrift 12',bg='#d2232a',fg='white',border=0,cursor='hand2', command=savenb)

    saveit.place(x=325, y=250)

    saveit.bind('<Enter>', on\_hover)

    saveit.bind('<Leave>', off\_hover)

    return frame3

def go\_back\_home(frame3, home\_frame):

    frame3.place\_forget()

    home\_frame.place(x=0,y=75)

def go\_chat(home\_frame, frame3):

    home\_frame.place\_forget()

    frame3.place(x=0,y=75)

This code defines a function called create\_frame3 that takes two arguments: master and go\_back\_home\_func. The master argument is the parent widget of the Frame that will be created, and go\_back\_home\_func is a callback function that will be called when the user clicks the back button.

The function creates a Frame widget with a width of 925 pixels and a height of 425 pixels, and sets its background color to white. It then defines two functions, on\_hover and off\_hover, which are used to change the color and font of the back button when the user hovers over it with the mouse.

The function then creates a back button with the text "<" and binds the on\_hover and off\_hover functions to it so that the button's color and font will change when the user hovers over it. The button is also given a callback function (go\_back\_home\_func) that will be called when the button is clicked.

The function then reads the contents of a file called "Mar.txt" and inserts the text into a Text widget called Noticeb. The text widget is placed on the Frame at position (250, 100).

The function then defines another function called savenb that will be called when the user clicks the "Save Message" button. This function first displays a confirmation dialog asking the user if they want to save changes. If the user clicks "Yes", the function retrieves the text from the Noticeb widget, logs the update activity into a CSV file called logfile, and saves the text to the file "Mar.txt". The function then displays a success message. If the user clicks "No", the function returns without doing anything.

The function then defines two more functions, on\_hover and off\_hover, which are used to change the color and font of the "Save Message" button when the user hovers over it with the mouse.

Finally, the function creates the "Save Message" button, binds the on\_hover and off\_hover functions to it, and places it on the Frame at position (325, 250).

The code also defines two more functions: go\_back\_home and go\_chat. The go\_back\_home function takes two arguments, frame3 and home\_frame, and is used to switch from the current frame3 to the home\_frame. The go\_chat function takes two arguments, home\_frame and frame3, and is used to switch from the home\_frame to the current frame3. After saving the text in the mar.txt it changes the scrolling text in the App\_Stu.py

#Database load----------------------------------------------

# Define file names

password\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/passwords -t.csv"

name\_file = "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/names -t.csv"

# Load data from files

passwords\_df = pd.read\_csv(password\_file, index\_col="Sapid")

names\_df = pd.read\_csv(name\_file, index\_col="Sapid")

with open("C:/Users/DELL/Desktop/PPS Code projects/Final project/Temp.txt", "r") as file1:

    Sapid=file1.read()

#Top bar Content------------------------------

import datetime

# Get the current time

now = datetime.datetime.now()

# Determine the greeting based on the time of day

if now.hour < 12:

    greeting = "Good morning, "

elif now.hour < 18:

    greeting = "Good afternoon, "

else:

    greeting = "Good evening, "

# Print the greeting

print(greeting)

app=Tk()

app.title("MPSTME Faculty Interface")

app.geometry('925x500+300+200')

app.resizable(False,False)

hi=Label(app,text=greeting+names\_df.loc[Sapid, "Name"],font=('Bahnschrift',20,'bold')) # Add '+names\_df.loc[Sapid, "Last Name"]

hi.place(x=300,y=20)

img=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/log.png')

Label(app,image=img).place(x=25,y=10)

im1=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/graduated.png')

im2=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/speech.png')

chatroom1=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/Chatoff.png')

chatroom2=PhotoImage(file='C:/Users/DELL/Desktop/PPS Code projects/Final project/App/chaton.png')

home\_frame = create\_home\_frame(app, lambda: go\_to\_next\_frame(home\_frame, second\_frame),lambda:go\_chat(home\_frame, frame3),im1,im2)

second\_frame = create\_second\_frame(app, lambda: go\_back(second\_frame, home\_frame))

frame3= create\_frame3(app,lambda:go\_back\_home(frame3, home\_frame))

logs=open('C:/Users/DELL/Desktop/PPS Code projects/Final project/his.txt','a+')

now=datetime.datetime.now()

logentry=str(now)+" Teacher: [ "+str(Sapid)+" ] has logged in\n"

print(logentry)

logs.write(logentry)

logs.close()

time=str(now)

role="Teacher"

activity="logged in"

logfile='C:/Users/DELL/Desktop/PPS Code projects/Final project/logs.csv'

log\_df=pd.read\_csv(logfile,index\_col="Time")

log\_df.loc[time]=[Sapid,role,activity]

log\_df.to\_csv(logfile)

def on\_hover2(event):

    logout['background'] = '#9f1a1f'

    logout['foreground'] = 'white'

    logout['font'] = 'Bahnschrift 9 underline'

    # Function to change button color when not hovering

def off\_hover2(event):

    logout['background'] = '#d2232a'

    logout['foreground'] = 'white'

    logout['font'] = 'Bahnschrift 9'

def loglogout():

    time=str(now)

    role="Student"

    activity="logged out"

    log\_df.loc[time]=[Sapid,role,activity]

    log\_df.to\_csv(logfile)

    messagebox.showinfo("Logged out","Logged out successfully")

    app.withdraw()

    subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/Python Ultimate Student interface.py"])

logout = Button(app,width=10,pady=2, text="Logout",bg='#d2232a',fg='white',border=0,cursor='hand2', command=loglogout)

logout.place(x=860, y=20)

logout.bind('<Enter>', on\_hover2)

logout.bind('<Leave>', off\_hover2)

app.mainloop()

adminconsole.py

import tkinter as tk

import subprocess

# Define the colors

bg\_color = "#36393F"

button\_bg\_off = "#5865f2"

button\_bg\_on = "#4752c4"

button\_fg = "white"

def studata():

    subprocess.call(['start', '', "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/namesnew.csv"], shell=True)

def studatapass():

    subprocess.call(['start', '', "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/passwordsnew.csv"], shell=True)

def teadata():

    subprocess.call(['start', '', "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/names -t.csv"], shell=True)

def teadatapass():

    subprocess.call(['start', '', "C:/Users/DELL/Desktop/PPS Code projects/Final project/Student interface/passwords -t.csv"], shell=True)

def useract():

    subprocess.call(['start', '', "C:/Users/DELL/Desktop/PPS Code projects/Final project/logs.csv"], shell=True)

def chistory():

    subprocess.call(['start', '', "C:/Users/DELL/Desktop/PPS Code projects/Final project/App/chat\_history.txt"], shell=True)

# Define the functions for button events

def on\_hover(event):

    event.widget.config(bg=button\_bg\_on)

def off\_hover(event):

    event.widget.config(bg=button\_bg\_off)

# Create the tkinter window

root = tk.Tk()

root.title("Admin Console")

root.config(bg=bg\_color)

root.geometry("+560+240")

frm=tk.Frame(root,bg="#2b2d31")

# Create the buttons

button1 = tk.Button(root,pady=5, text="Open Student Data", bg=button\_bg\_off, fg=button\_fg,border=0,cursor='hand2',command=studata)

button1.bind("<Enter>", on\_hover)

button1.bind("<Leave>", off\_hover)

button2 = tk.Button(root,pady=1, text="Open Student Login Data", bg=button\_bg\_off, fg=button\_fg,border=0,cursor='hand2',command=studatapass)

button2.bind("<Enter>", on\_hover)

button2.bind("<Leave>", off\_hover)

button3 = tk.Button(root,pady=5, text="Open Teacher Data", bg=button\_bg\_off, fg=button\_fg,border=0,cursor='hand2',command=teadata)

button3.bind("<Enter>", on\_hover)

button3.bind("<Leave>", off\_hover)

button4 = tk.Button(root,pady=5, text="Open Teacher Login Data", bg=button\_bg\_off, fg=button\_fg,border=0,cursor='hand2',command=teadatapass)

button4.bind("<Enter>", on\_hover)

button4.bind("<Leave>", off\_hover)

button5 = tk.Button(root,pady=5, text="Open User activity logs", bg=button\_bg\_off, fg=button\_fg,border=0,cursor='hand2',command=useract)

button5.bind("<Enter>", on\_hover)

button5.bind("<Leave>", off\_hover)

button6 = tk.Button(root,pady=5, text="Open Chatroom History", bg=button\_bg\_off, fg=button\_fg,border=0,cursor='hand2',command=chistory)

button6.bind("<Enter>", on\_hover)

button6.bind("<Leave>", off\_hover)

im=tk.PhotoImage(file="C:/Users/DELL/Desktop/PPS Code projects/Final project/App/Admin.png")

labl=tk.Label(image=im,bg="#36393F")

labl.grid(row=0, column=0, pady=10)

Head=tk.Label(text="Admin Console",bg="#36393F",fg="white", font=('Bauhaus 93',20))

Head.grid(row=0, column=1, pady=10)

# Display the buttons in the window

button1.grid(row=1, column=0, padx=10, pady=10,sticky=tk.N+tk.S+tk.E+tk.W)

button2.grid(row=1, column=1, padx=10,pady=10,sticky=tk.N+tk.S+tk.E+tk.W)

button3.grid(row=2, column=0, padx=10, pady=10,sticky=tk.N+tk.S+tk.E+tk.W)

button4.grid(row=2, column=1, padx=10,pady=10,sticky=tk.N+tk.S+tk.E+tk.W)

button5.grid(row=3, column=0, padx=10, pady=10,sticky=tk.N+tk.S+tk.E+tk.W)

button6.grid(row=3, column=1, padx=10,pady=10,sticky=tk.N+tk.S+tk.E+tk.W)

# Run the tkinter event loop

root.mainloop()

subprocess.Popen(["python", "C:/Users/DELL/Desktop/PPS Code projects/Final project/Python Ultimate Student interface.py"])

This code is a Python script that creates a graphical user interface (GUI) using the tkinter library. The GUI is an Admin Console that contains six buttons. Each button is associated with a function that opens a file when the button is clicked.

The files that can be opened include student data, student login data, teacher data, teacher login data, user activity logs, and chatroom history. The files are opened using the subprocess module, which runs system commands in a new process.

The code defines several variables for the colors used in the GUI, and creates functions for changing the button color when the mouse pointer hovers over the button.

The tkinter window is created using the Tk class, and the six buttons are created using the Button class. Each button is bound to the functions that open the corresponding file when clicked. The Label class is used to display an image, and the Label and grid methods are used to display the image and a heading on the GUI.

Finally, the mainloop() method of the Tk class is called to run the GUI, and after the GUI is closed, the subprocess.Popen() method is called to start another Python script called "Python Ultimate Student interface.py".