CBS1011-Programming in Python
Employee Attendance System Project

Bachelor of Technology

in

Computer Science Engineering and Business Systems

by

KHUSHAL AGRAWAL

23BBS0043

Under the guidance of Prof. / Dr.

S. SUDHA

VIT, Vellore.

April, 2024

<u>DECLARATION</u>

I hereby declare that the Project entitled "Employee Attendance System" submitted by me, for the course *CBS1011 Programming in Python* of <mark>Bachelor of Technology in Computer Science Engineering and Business Systems</mark> to VIT is a record of bonafide work carried out by me under the supervision of Dr. S. Sudha.

I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for any other course in this institute or any other institute or university.

Place   : Vellore

Date   : 21.04.24

KHUSHAL AGRAWAL

Signature of the Candidate

CERTIFICATE

This is to certify that the project entitled "Employee Attendance System" submitted by **KHUSHAL AGRAWAL 23BBS0043**, **SCOPE**, VIT, for the course of *Programming in Python*, is a record of bonafide work carried out by him / her under my supervision during the period, 01.12.2023 to 03.05.2024, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for award of marks for any other course in this institute or any other institute or university.

Place   : Vellore

Date   : 21.04.24   **Signature of the Guide**

CONTENTS

1. INTRODUCTION

1.1 THEORETICAL BACKGROUND:

Employee Attendance Management System (EAMS) is a crucial component of organizational management, ensuring the efficient tracking and monitoring of employees' work hours, absences, and attendance patterns. This system plays a vital role in maintaining workforce productivity, compliance with labor regulations, and payroll accuracy.

In today's dynamic work environment, where remote work and flexible schedules are increasingly common, having a robust attendance management system is essential for businesses of all sizes. Traditional manual methods of attendance tracking, such as paper timesheets or Excel spreadsheets, are prone to errors, time-consuming, and lack real-time visibility. As a result, organizations are turning to automated attendance management systems to streamline processes and improve efficiency.

Implementing an employee attendance project requires careful planning, customization to fit the organization's unique requirements, and thorough training for employees and administrators. Additionally, data security and privacy considerations are paramount when handling sensitive employee information.

Overall, an effective employee attendance project enhances operational efficiency, promotes accountability, and fosters a positive work culture, ultimately contributing to the organization's success and growth.

1.2 MOTIVATION:

We are excited to propose a collaborative endeavor aimed at initiating a Python project within our esteemed group. As we navigate the ever-evolving landscape of technology, harnessing the power of Python, a versatile and widely adopted programming language, presents an exceptional opportunity for us to synergize our collective expertise and drive impactful change within our organization.

Python's popularity and versatility make it an ideal choice for a wide range of applications, from web development and data analysis to artificial intelligence and automation.

Embarking on a Python project as a cohesive and collaborative team aligns with our shared goals of driving innovation, fostering teamwork, and delivering exceptional results for our organization. I am confident that by working together, we can leverage the power of Python to achieve remarkable success and make a lasting impact on our organization's journey towards digital transformation.

1.3 OBJECTIVE(S) OF THE PROPOSED WORK:

The objectives of an Employee Attendance Management System (EAMS) are multifaceted, aiming to streamline processes, enhance productivity, ensure compliance, and foster a positive work environment.

1. **Time Tracking**: The system records employees' clock-in and clock-out times, either through physical devices like biometric scanners, RFID cards, or through software-based solutions accessible via computers or mobile devices. This accurate time tracking helps in calculating total work hours and overtime, ensuring fair compensation for employees.

2. **Absence Managemen**t: Employees can request leaves or report absences through the system, which then undergoes approval by managers or HR personnel. This streamlines the leave approval process, reduces paperwork, and ensures compliance with company policies and labor laws.

3**. Real-time Monitoring**: Managers and administrators have access to real-time attendance data, allowing them to quickly identify attendance issues, such as late arrivals or unauthorized absences. This enables timely intervention and corrective actions to maintain productivity levels.

4. **Reporting and Analysis**: The system generates comprehensive reports and analytics on attendance trends, employee punctuality, absenteeism rates, and other relevant metrics. These insights help management make informed decisions regarding workforce planning, scheduling optimization, and performance evaluation.

5. **Integration with Payroll**: Seamless integration with payroll systems automates the calculation of salaries and benefits based on employees' attendance records, reducing manual errors and ensuring accurate and timely payroll processing.

2. PROPOSED SYSTEM ANALYSIS AND DESIGN

2.1 H/W Requirements(details about Application Specific Hardware)

To embark on a basic Python project, minimal hardware specifications suffice. A standard laptop or desktop computer with a modern processor (e.g., Intel Core i5 -7700 or AMD Ryzen 5) and at least 8GB of RAM is adequate. Additionally, ensure ample storage space, preferably 20GB SSD, to accommodate project files and dependencies. These modest hardware requirements enable smooth development and execution of Python scripts and applications, making Python accessible to a wide range of users with varying computing setups. There is no need for a dedicated GPU for executing this code.

2.2 S/W Requirements(details about Application Specific Software)

Python 3.12: Python is a versatile and high-level programming language known for its simplicity and readability, making it ideal for beginners and professionals alike. With a rich ecosystem of libraries and frameworks, Python facilitates rapid development across various domains, including web development, data analysis, machine learning, and automation. Its dynamic typing and interpreted nature contribute to its flexibility and ease of use, making it a popular choice for diverse projects and industries.

JAVA JRE: Java Runtime Environment (JRE) is a crucial component of the Java platform, enabling the execution of Java applications on various devices and operating systems. It includes the Java Virtual Machine (JVM), runtime libraries, and other components necessary to run Java bytecode. JRE is essential for running Java-based software, from desktop applications to web applets, ensuring compatibility and portability across different environments.

Microsoft Visual Studio Code: Visual Studio Code, often abbreviated as VS Code, is a versatile and popular code editor used by developers worldwide. With its intuitive user interface, extensive plugin ecosystem, and robust features, VS Code streamlines the coding experience. Whether you're working on web development, data science, or cloud applications, VS Code provides a seamless environment for writing, debugging, and collaborating on code.

2.3 System Modules

In Python, a system module typically refers to a library or package that provides functionality related to interacting with the operating system, managing files and directories, executing system commands, and accessing system resources.

We have used a library known as "tkinter" which allows python to create a UI UX Design. Tkinter is a built-in Python library used for creating GUI (Graphical User Interface) applications. It provides a set of tools and widgets that allow developers to create windows, buttons, labels, textboxes, menus, and more, making it easier to build interactive applications with a graphical interface.

The datetime module in Python provides classes and functions for working with dates and times. It allows developers to manipulate dates, times, and time intervals easily. With datetime, you can create, format, and perform arithmetic operations on dates and times, making it essential for applications involving scheduling, data analysis, and more. Whether parsing strings into datetime objects or calculating time differences, the datetime module simplifies complex temporal operations in Python.

3. CODE IMPLEMENTATION AND RESULTS

3.1 IMPLEMENTATION CODE

```python
import tkinter
from tkinter import messagebox
from tkinter import ttk
import time
import datetime
from tkinter import *
from datetime import datetime
import copy
admin_id = "Dr.Sudha"
admin_pass = "1679"
def admin_login():
    if(admin_username_entry.get()==admin_id):
        if(admin_password_entry.get() == admin_pass):
            window.destroy()
        else:
            windowtre = tkinter.Tk()
            windowtre.title("INVALID LOGIN")
            windowtre.geometry('900x440')
            windowtre.configure(bg='#333333')
            saved_label = tkinter.Label(windowtre, text="INVALID PASSWORD,
TRY AGAIN", bg='#333333', fg="#2ff900", font=("Arial", 30))
            saved_label.grid(row=0, column=0, columnspan=2, sticky="news",
pady=40)
            windowtre.mainloop()
            exit()
    else:
            windowtre = tkinter.Tk()
```

```python
            windowtre.title("INVALID ID")
            windowtre.geometry('900x440')
            windowtre.configure(bg='#333333')
            saved_label = tkinter.Label(windowtre, text="INVALID ID, try
running the code again", bg='#333333', fg="#2ff900", font=("Arial", 30))
            saved_label.grid(row=0, column=0, columnspan=2, sticky="news",
pady=40)

            windowtre.mainloop()
            exit()


window = tkinter.Tk()
window.title("Admin authentication")
window.geometry('800x600')
window.configure(bg='#333333')
frame = tkinter.Frame(bg='#333333')
admin_login_label = tkinter.Label(
    frame, text="Admin Login", bg='#333333', fg="#1388e9", font=("Arial",
30))
admin_username_label = tkinter.Label(
    frame, text="Admin Username", bg='#333333', fg="#FFFFFF", font=("Arial",
16))
admin_username_entry = tkinter.Entry(frame, font=("Arial", 16))


admin_password_entry = tkinter.Entry(frame, show="*", font=("Arial", 16))
admin_password_label = tkinter.Label(
    frame, text="Admin password", bg='#333333', fg="#FFFFFF", font=("Arial",
16))
login_button = tkinter.Button(
    frame, text="Admin Login", bg="#1388e9", fg="#FFFFFF", font=("Arial",
16), command = admin_login)


# Placing widgets on the screen
admin_login_label.grid(row=0, column=0, columnspan=2, sticky="news", pady=40)
admin_username_label.grid(row=1, column=0)
admin_password_label.grid(row=2, column=0)

admin_username_entry.grid(row=1, column=1, pady=20)
admin_password_entry.grid(row=2, column=1, pady=20)
login_button.grid(row=4, column=3, columnspan=2, pady=30)

frame.pack()
window.mainloop()
################################################################################
######################################################
firstname = []
lastname  = []
gender = []
age =[]
typeemployee =[]
empolyee_ID = []
```

```python
passw = []
count = 0
counter = 3
personal_leave = 20
medical_leave = 12
leavedays = 0
def enter_data():
        global status
        global timein
        global timeout
        global time_worked
        global venue_code_in
        global venue_code_out
        global overtime
        global latepunch
        global earlypunch
        global leavedays
        global firstname
        global lastname
        global gender
        global age
        global typeemployee
        global empolyee_ID
        global passw
        global count
        global counter
        global personal_leave
        global medical_leave
        firstname.append( first_name_entry.get())
        lastname.append(last_name_entry.get())
        if firstname and lastname:
            gender.append( Gender_combobox.get() )
            age.append(age_spinbox.get())
            typeemployee.append( typemp_combobox.get())
            empolyee_ID.append(EMPID_entry.get())
            passw.append(pass_entry.get())
            count = count + 1
            first_name_entry.delete(0, 'end')
            last_name_entry.delete(0, 'end')
            Gender_combobox.set('')
            age_spinbox.delete(0, 'end')
            typemp_combobox.set('')
            EMPID_entry.delete(0, 'end')
            pass_entry.delete(0, 'end')
            if count==counter:
                window.destroy()
            windowy = tkinter.Tk()
            windowy.title("Data saved")
            windowy.geometry('700x250')
            windowy.configure(bg='#333333')
            saved_label = tkinter.Label(windowy, text="Data for employee
saved successfully", bg='#333333', fg="#2ff900", font=("Arial", 30))
```

```python
            saved_label.grid(row=0, column=0, columnspan=2, sticky="news",
pady=40)
            windowy.mainloop()


        else:
            messagebox.showinfo(title="Error", message="First name and last
name are required.")

status = [0]*counter
timein = [0]*counter
timeout= [0]*counter
timein_hour = [0]*counter
timein_seconds= [0]*counter
timein_minute= [0]*counter
time_worked= [0]*counter
venue_code_in = [0]*counter
venue_code_out = [0]*counter
overtime  = [0]*counter
latepunch = [0]*counter
earlypunch = [0] *counter
atten = [0] * counter
temp = [0]*counter
tempover=[0]*counter
timein_day =[0]*counter
timein_month =[0]*counter
timein_year=[0]*counter
timeout_hour=[0]*counter
timeout_seconds=[0]*counter
timeout_minute=[0]*counter
timeout_day=[0]*counter
timeout_month=[0]*counter
timeout_year=[0]*counter

window = tkinter.Tk()
window.title("Data Entry Form")

frame = tkinter.Frame(window)
frame.pack()

# Saving User Info
user_info_frame =tkinter.LabelFrame(frame, text="User Information")
user_info_frame.grid(row= 0, column=0, padx=20, pady=10)

first_name_label = tkinter.Label(user_info_frame, text="First Name")
first_name_label.grid(row=0, column=0)
last_name_label = tkinter.Label(user_info_frame, text="Last Name")
last_name_label.grid(row=0, column=1)

first_name_entry = tkinter.Entry(user_info_frame)
last_name_entry = tkinter.Entry(user_info_frame)
first_name_entry.grid(row=1, column=0)
last_name_entry.grid(row=1, column=1)
```

```python
Gender_label = tkinter.Label(user_info_frame, text="Gender")
Gender_combobox = ttk.Combobox(user_info_frame, values=["Male", "Female",])
Gender_label.grid(row=0, column=2)
Gender_combobox.grid(row=1, column=2)

age_label = tkinter.Label(user_info_frame, text="Age")
age_spinbox = tkinter.Spinbox(user_info_frame, from_=18, to=110)
age_label.grid(row=2, column=0)
age_spinbox.grid(row=3, column=0)

typemp_label = tkinter.Label(user_info_frame, text="Type of employee")
typemp_combobox = ttk.Combobox(user_info_frame, values=["Permanent" ,
"Intern"])
typemp_label.grid(row=2, column=1)
typemp_combobox.grid(row=3, column=1)

for widget in user_info_frame.winfo_children():
    widget.grid_configure(padx=10, pady=5)


# Saving Course Info
courses_frame = tkinter.LabelFrame(frame)
courses_frame.grid(row=1, column=0, sticky="news", padx=20, pady=10)

EMPID_label = tkinter.Label(courses_frame, text= "Employee ID")
EMPID_entry= tkinter.Entry(courses_frame)
EMPID_label.grid(row=0, column=1)
EMPID_entry.grid(row=1, column=1)

pass_label = tkinter.Label(courses_frame, text= "Password")
pass_entry= tkinter.Entry(courses_frame)
pass_label.grid(row=0, column=2)
pass_entry.grid(row=1, column=2)

for widget in courses_frame.winfo_children():
    widget.grid_configure(padx=10, pady=5)


# Button
button = tkinter.Button(frame, text="Enter data", command= enter_data)
button.grid(row=3, column=0, sticky="news", padx=20, pady=10)

window.mainloop()

###The Original display age that will be on at all times (default)
window_login = tkinter.Tk()
window_login.title("Login form")
window_login.geometry('800x500')
window_login.configure(bg='#333333')
frame = tkinter.Frame(bg='#333333')
```

```python
def login():
    global status
    global timein
    global timeout
    global time_worked
    global venue_code_in
    global venue_code_out
    global overtime
    global latepunch
    global earlypunch
    global leavedays
    global firstname
    global lastname
    global gender
    global age
    global typeemployee
    global empolyee_ID
    global passw
    global count
    global counter
    global personal_leave
    global medical_leave
    global timein_hour
    global timein_seconds
    global timein_minute
    global timein_day
    global timein_month
    global timein_year
    global timeout_hour
    global timeout_seconds
    global timeout_minute
    global timeout_day
    global timeout_month
    global timeout_year
    un = username_entry.get()
    pas= password_entry.get()
    if un in empolyee_ID:
        inde = empolyee_ID.index(un)
        if (passw[inde] == pas):
            if(status[inde] == 0):
                venue_code_in[inde] = venue_entry.get()
                atten[inde] = "Present"
                timein_temp = datetime.now()
                timein[inde] = datetime.now()
                status[inde] = 1
                timein_hour[inde] = int(timein_temp.hour)
                timein_minute[inde] = int(timein_temp.minute)
                timein_seconds[inde] =  int(timein_temp.second)
                timein_day[inde] = int(timein_temp.day)
                timein_month[inde] = int(timein_temp.month)
                timein_year[inde] = int(timein_temp.year)
                if(int(timein_hour[inde])>9):
                    latepunch[inde] = "late"
```

```python
            elif(status[inde] == 1):
                venue_code_out[inde] = venue_entry.get()
                timeout_temp = datetime.now()
                timeout[inde] = datetime.now()
                timeout_hour[inde] = int(timeout_temp.hour)
                timeout_minute[inde] = int(timeout_temp.minute)
                timeout_seconds[inde] =  int(timeout_temp.second)
                timeout_day[inde] = int(timeout_temp.day)
                timeout_month[inde] = int(timeout_temp.month)
                timeout_year[inde] = int(timeout_temp.year)
                if(int(timeout_hour[inde]) <17):
                    earlypunch[inde] = "early"
                time_worked[inde] = int(time_worked[inde]) +
(int(timeout_temp.hour)*3600 - (int(timein_hour[inde])*3600)) +
(int(timeout_temp.minute*60) - (int(timein_minute[inde])*60)) +
(int(timeout_temp.second) - int(timein_seconds[inde]))
                if(int(time_worked[inde])>(8*3600)):
                    overtime[inde] = int(overtime[inde]) +
int(time_worked[inde]-(8*3600))
                status[inde] = 0

            username_entry.delete(0, 'end')
            venue_entry.delete(0, 'end')
            password_entry.delete(0, 'end')
            window = tkinter.Tk()
            window.title("Logged in successfully")
            window.geometry('800x500')
            window.configure(bg='#333333')
            frame = tkinter.Frame(window)
            frame.pack()
            def logout():
                window.destroy()

            def update_data():
                global status
                global timein
                global timeout
                global time_worked
                global venue_code_in
                global venue_code_out
                global overtime
                global latepunch
                global earlypunch
                global leavedays
                global firstname
                global lastname
                global gender
                global age
                global typeemployee
                global empolyee_ID
                global passw
```

```python
                    global count
                    global counter
                    global personal_leave
                    global medical_leave
                    windows = tkinter.Tk()
                    windows.title("Data update form")
                    framed = tkinter.Frame(windows)
                    framed.pack()
                    def update_data_final():
                        global status
                        global timein
                        global timeout
                        global time_worked
                        global venue_code_in
                        global venue_code_out
                        global overtime
                        global latepunch
                        global earlypunch
                        global leavedays
                        global firstname
                        global lastname
                        global gender
                        global age
                        global typeemployee
                        global empolyee_ID
                        global passw
                        global count
                        global counter
                        global personal_leave
                        global medical_leave
                        firstname[inde]= (first_name_entry.get())
                        lastname[inde] = (last_name_entry.get())

                        if firstname and lastname:
                            age[inde] = (age_spinbox.get())
                            passw[inde] = (pass_entry.get())
                            windows.destroy()
                            window = tkinter.Tk()
                            window.title("Data updated successfully")
                            window.geometry('900x440')
                            window.configure(bg='#333333')
                            saved_label = tkinter.Label(window, text="Data
for employee saved successfully", bg='#333333', fg="#2ff900", font=("Arial",
30))
                            saved_label.grid(row=0, column=0, columnspan=2,
sticky="news", pady=40)
                            window.mainloop()

                    # Saving User Info
                    user_info_frame =tkinter.LabelFrame(framed, text="User
Information update")
                    user_info_frame.grid(row= 0, column=0, padx=20, pady=10)
```

```python
                        first_name_label = tkinter.Label(user_info_frame,
text="First Name")
                        first_name_label.grid(row=0, column=0)
                        last_name_label = tkinter.Label(user_info_frame,
text="Last Name")
                        last_name_label.grid(row=0, column=1)

                        first_name_entry = tkinter.Entry(user_info_frame)
                        last_name_entry = tkinter.Entry(user_info_frame)
                        first_name_entry.grid(row=1, column=0)
                        last_name_entry.grid(row=1, column=1)


                        age_label = tkinter.Label(user_info_frame, text="Age")
                        age_spinbox = tkinter.Spinbox(user_info_frame, from_=18,
to=110)
                        age_label.grid(row=2, column=0)
                        age_spinbox.grid(row=3, column=0)



                        for widget in user_info_frame.winfo_children():
                            widget.grid_configure(padx=10, pady=5)


                        # Saving Course Info
                        courses_frame = tkinter.LabelFrame(framed)
                        courses_frame.grid(row=1, column=0, sticky="news",
padx=20, pady=10)


                        pass_label = tkinter.Label(courses_frame, text=
"Password")
                        pass_entry= tkinter.Entry(courses_frame)
                        pass_label.grid(row=0, column=2)
                        pass_entry.grid(row=1, column=2)

                        for widget in courses_frame.winfo_children():
                            widget.grid_configure(padx=10, pady=5)


                        # Button
                        button = tkinter.Button(framed, text="Enter Updated
data", command= update_data_final)
                        button.grid(row=3, column=0, sticky="news", padx=20,
pady=10)

                        window.mainloop()

            def leaves():
                global status
                global timein
                global timeout
                global time_worked
                global venue_code_in
```

```python
            global venue_code_out
            global overtime
            global latepunch
            global earlypunch
            global leavedays
            global firstname
            global lastname
            global gender
            global age
            global typeemployee
            global empolyee_ID
            global passw
            global count
            global counter
            global personal_leave
            global medical_leave
            windowed = tkinter.Tk()
            windowed.title("Leaves")
            windowed.geometry('900x440')
            windowed.configure(bg='#333333')
            frameds = tkinter.Frame(windowed)
            frameds.pack()

            leaves_label = tkinter.Label(frameds, text="Enter the number
of days you will be on leave", bg='#333333', fg="#2ff900", font=("Arial",
30))
            leaves_label.grid(row=0, column=0, columnspan=2,
sticky="news", pady=40)

            lday_label = tkinter.Label(frameds, text="No.of days ")
            lday_spinbox = tkinter.Spinbox(frameds, from_=0, to=20)
            lday_label.grid(row=2, column=0)
            lday_spinbox.grid(row=3, column=0)

            TOL_label = tkinter.Label(frameds, text="Type of leave")
            TOL_combobox = ttk.Combobox(frameds, values=["Personal" ,
"Medical"])
            TOL_label.grid(row=2, column=1)
            TOL_combobox.grid(row=3, column=1)


            def leaves_lol():
                global status
                global timein
                global timeout
                global time_worked
                global venue_code_in
                global venue_code_out
                global overtime
                global latepunch
                global earlypunch
                global leavedays
```

```python
                            global firstname
                            global lastname
                            global gender
                            global age
                            global typeemployee
                            global empolyee_ID
                            global passw
                            global count
                            global counter
                            global personal_leave
                            global medical_leave
                            global personal_leave
                            global medical_leave
                            global leavedays
                            if(TOL_combobox.get() == "Personal" and
personal_leave>0):
                                if(int(lday_spinbox.get())<=personal_leave):
                                    personal_leave = personal_leave -
int(lday_spinbox.get())
                                    leavedays = int(lday_spinbox.get())
                                    window = tkinter.Tk()
                                    window.title("PL granted")
                                    window.geometry('900x440')
                                    window.configure(bg='#333333')
                                    saved_label = tkinter.Label(window,
text="Personal Leave granted", bg='#333333', fg="#2ff900", font=("Arial",
30))
                                    saved_label.pack()
                                    window.mainloop()
                                else:
                                    window = tkinter.Tk()
                                    window.title("PL not granted")
                                    window.geometry('900x440')
                                    window.configure(bg='#333333')
                                    var = IntVar()
                                    var.set(personal_leave)
                                    saved_label = tkinter.Label(window, text="Not
enough leaves, try again", bg='#333333', fg="#2ff900", font=("Arial", 30))
                                    bruh_label = tkinter.Label(window, text=
"Personal leaves available:", bg='#333333', fg="#2ff900", font=("Arial", 30))
                                    dude_label = tkinter.Label(window, textvariable =
var, bg='#333333', fg="#2ff900", font=("Arial", 30))
                                    saved_label.pack()
                                    bruh_label.pack()
                                    dude_label.pack()
                                    window.mainloop()

                            if(TOL_combobox.get() == "Medical" and medical_leave>0):
                                if(int(lday_spinbox.get())<=medical_leave):
                                    personal_leave = medical_leave -
int(lday_spinbox.get())
                                    leavedays = int(lday_spinbox.get())
                                    window = tkinter.Tk()
```

```python
                            window.title("ML granted")
                            window.geometry('900x440')
                            window.configure(bg='#333333')
                            saved_label = tkinter.Label(window, text="Your
Medical leave is granted, take care!", bg='#333333', fg="#2ff900",
font=("Arial", 30))
                            saved_label.pack()
                            window.mainloop()
                        else:
                            window = tkinter.Tk()
                            window.title("PL not granted")
                            window.geometry('900x440')
                            window.configure(bg='#333333')
                            var = IntVar()
                            var.set(medical_leave)
                            saved_label = tkinter.Label(window, text="Not
enough leaves available for this year", bg='#333333', fg="#2ff900",
font=("Arial", 30))
                            bruh_label = tkinter.Label(window, text= "Medical
leaves will be deducted from next year", bg='#333333', fg="#2ff900",
font=("Arial", 30))
                            lolmao_label = tkinter.Label(window, text= "Next
Year's Medical leaves:", bg='#333333', fg="#2ff900", font=("Arial", 30))
                            dude_label = tkinter.Label(window, textvariable =
12+(var-int(lday_spinbox.get()))), bg='#333333', fg="#2ff900", font=("Arial",
30))
                            medical_leave = 12+(var-int(lday_spinbox.get()))
                            saved_label.pack()
                            lolmao_label.pack()
                            bruh_label.pack()
                            dude_label.pack()
                            window.mainloop()

                apply_button = tkinter.Button(frameds, text="Apply leave",
command= leaves_lol)
                apply_button.grid(row=4, column=1, sticky="news", padx=20,
pady=10)
                windowed.mainloop()

            leaves_button = tkinter.Button(frame, text="Apply for leaves",
bg="#1388e9", fg="#FFFFFF", font=("Arial", 16), command = leaves)
            personal_button = tkinter.Button(frame, text="Update personal
information", bg="#1388e9", fg="#FFFFFF", font=("Arial", 16), command =
update_data)
            signout_button = tkinter.Button(frame, text="LOGOUT",
bg="#1388e9", fg="#FFFFFF", font=("Arial", 16), command = logout)

            personal_button.grid(row=0, column=1, pady=40, sticky="news")
            signout_button.grid(row=1, column=2, pady=40, sticky="news")
            leaves_button.grid(row=0, column=2, columnspan=2, pady=40,
sticky="news")
            window.mainloop()
```

```python
        else:
            window = tkinter.Tk()
            window.title("Wrong password, try again")
            window.geometry('900x440')
            window.configure(bg='#333333')
            saved_label = tkinter.Label(window, text="Error in
authentication", bg='#333333', fg="#2ff900", font=("Arial", 30))
            saved_label.grid(row=0, column=0, columnspan=2, sticky="news",
pady=40)
            window.mainloop()
    else:
        window = tkinter.Tk()
        window.title("Employee ID authentication error")
        window.geometry('900x440')
        window.configure(bg='#333333')
        saved_label = tkinter.Label(window, text="Employee ID not found",
bg='#333333', fg="#2ff900", font=("Arial", 30))
        saved_label.grid(row=0, column=0, columnspan=2, sticky="news",
pady=40)
        window.mainloop()

def endatten_button():
    global status
    global timein
    global timeout
    global time_worked
    global venue_code_in
    global venue_code_out
    global overtime
    global latepunch
    global earlypunch
    global leavedays
    global firstname
    global lastname
    global gender
    global age
    global typeemployee
    global empolyee_ID
    global passw
    global count
    global counter
    global personal_leave
    global medical_leave
    with open("C:/Users/khush/OneDrive/Desktop/python/pythonprojectfile.txt"
,'a') as f:
        for i in range(0,count):
            if (atten[i]==0):
                if(int(leavedays) == 0):
                    f.write(firstname[i] + lastname[i] + " was absent for
today. Employee ID: " + empolyee_ID[i] +"\n")
                else:
                    leavedays = int(int(leavedays) - 1)
            else:
```

```python
                f.write("The punch in time for "+ firstname[i] +" "+
lastname[i] +" was " )
                f.write(str(timein_hour[i]) + ":" + str(timein_minute[i])
+":" + str(timein_seconds[i])+ " " +str(timein_day[i])+ "/"+ str(
timein_month[i]) +"/" + str(timein_year[i]))
                f.write("\n")
                f.write("The punch out time for "+ firstname[i] +" "+
lastname[i] +" was ")
                f.write(str(timeout_hour[i]) + ":" + str(timeout_minute[i])
+":" + str(timeout_seconds[i]) + " "+ str(timeout_day[i]) + "/"+
str(timeout_month[i])+ "/" + str(timeout_year[i] ))
                f.write("\n")
                f.write("The venue code in for " + firstname[i] +" "+
lastname[i] +" was " + str(venue_code_in[i]) +"\n")
                f.write("The venue code out for " + firstname[i] +" "+
lastname[i] +" was " + str(venue_code_out[i]) +"\n")
                f.write("The overtime for " + firstname[i] + " " +lastname[i]
+" is " + str(overtime[i])+"\n")
                if(latepunch[i] =="late"):
                    f.write(firstname[i] + lastname[i] +" was " +
str(latepunch[i])+"\n")
                if(earlypunch[i] == "early"):
                    f.write(firstname[i] + lastname[i] +" punched out " +
str(earlypunch[i]) +"\n")
                latepunch[i] = 0
                atten[i] = 0
                earlypunch[i] = 0
                temp[i]= i
        for i in range(0,counter-1):
            tempover[i] = int(overtime[i])
        for i in range(0,counter-1):
            for j in range(0,counter-i-1):
                if(tempover[j]>tempover[j+1]):
                    temptemp = tempover[j]
                    tempover[j] = tempover[j+1]
                    tempover[j+1] = temptemp
                    temptemptemp = temp[j]
                    tempover[j] = temp[j+1]
                    temp[j+1] = temptemptemp
        f.write("The members arranged in order of most hardworking based on
overtime is:\n")
        for i in range(0,counter):
            f.write(firstname[i] +" " + lastname[i] +"\n")

    window = tkinter.Tk()
    window.title("End Attendance")
    window.geometry('900x440')
    window.configure(bg='#333333')
    saved_label = tkinter.Label(window, text="The attendance for the day is
over", bg='#333333', fg="#2ff900", font=("Arial", 30))
    saved_label.pack()
    window_login.destroy()
    window.mainloop()
```

```python
login_label = tkinter.Label(
    frame, text="Login", bg='#333333', fg="#1388e9", font=("Arial", 30))

username_label = tkinter.Label(
    frame, text="Username", bg='#333333', fg="#FFFFFF", font=("Arial", 16))
username_entry = tkinter.Entry(frame, font=("Arial", 16))
endatten_label = tkinter.Label(frame, text="End attendance for the day",
bg='#333333', fg="#FFFFFF", font=("Arial", 16))

venue_label = tkinter.Label(
    frame, text="Venue Code", bg='#333333', fg="#FFFFFF", font=("Arial", 16))
venue_entry = tkinter.Entry(frame, font=("Arial", 16))

password_entry = tkinter.Entry(frame, show="*", font=("Arial", 16))
password_label = tkinter.Label(
    frame, text="Password", bg='#333333', fg="#FFFFFF", font=("Arial", 16))
login_button = tkinter.Button(
    frame, text="Login", bg="#1388e9", fg="#FFFFFF", font=("Arial", 16),
command = login)
endatten = tkinter.Button(
    frame, text="End attendance", bg="#1388e9", fg="#FFFFFF", font=("Arial",
16), command = endatten_button)

# Placing widgets on the screen
login_label.grid(row=0, column=0, columnspan=2, sticky="news", pady=40)
username_label.grid(row=1, column=0)
password_label.grid(row=2, column=0)
venue_label.grid(row = 3, column = 0)
endatten_label.grid(row = 4, column = 0)


username_entry.grid(row=1, column=1, pady=20)
password_entry.grid(row=2, column=1, pady=20)
venue_entry.grid(row = 3, column = 1, pady=20)

endatten.grid(row=5, column=0,columnspan=2, pady=30)
login_button.grid(row=4, column=3, columnspan=2, pady=30)

frame.pack()
window.mainloop()
```

3.2 CODE EXPLANATION WITH SCREENSHOTS

Our code focuses on Attendance Management of employees in VIT.

It consists of several different functions and datatypes which help in maintaining the necessary attributes required in order to manage and maintain attendance.

1. Admin Login Section:

- The code begins with an admin login interface created using Tkinter, a Python GUI library.
- Admin credentials are hardcoded (**admin_id** and **admin_pass**).
- The **admin_login()** function is called when the admin attempts to log in.
- If the entered credentials match the predefined ones, the admin is granted access to the main application.
- If the credentials are incorrect, appropriate error messages are displayed, prompting the admin to try again.

2. Employee Data Entry Form:

- After successful admin login, the program displays a data entry form for entering employee details.
- Various fields such as first name, last name, gender, age, type of employee, employee ID, and password are provided.
- Once all the required fields are filled, the entered data is stored in separate lists (**firstname**, **lastname**, etc.).
- Error handling is implemented to ensure that essential information like first name and last name are provided before data submission.
- Upon reaching a certain count of entries (defined by **counter**), the window is closed.

3. User Login Section:

- After data entry, the program presents a login interface for employees.
- Employees are required to enter their employee ID, password, and venue code.
- Upon successful login, employees can mark their attendance, apply for leaves, update personal information, or log out.
- Error messages are displayed for incorrect employee IDs or passwords.

4. Attendance Marking and Actions:

- When an employee logs in successfully, they can mark their attendance by entering a venue code.

- The system records the time of login as the time in, and upon logout, records the time out.
- Overtime calculation is done by comparing the time worked against a standard 8-hour workday.
- Late or early punch-ins are flagged and handled accordingly.
- Employees can apply for leaves, update personal information, or log out using dedicated buttons.

5. End Attendance Button:

- This button, accessible only to the admin, allows for ending the attendance for the day.
- Upon clicking, it saves attendance details to a file, including venue codes, overtime, late punch-ins, etc.
- A confirmation message is displayed upon successful completion.

6. Overall Structure and File Handling:

- The code is structured into logical sections, with functions for each specific task, promoting readability and modularity.
- The Model-View-Controller (MVC) pattern is followed, separating data management, user interface, and application logic.
- File handling is integrated to store attendance details in a text file for future reference.

In essence, the code constitutes a comprehensive attendance management system, facilitating admin authentication, employee data entry, attendance marking, leave application, and data updating functionalities. It utilizes Tkinter for GUI, implements error handling, and ensures data integrity through file handling.

❖ admin_login()

The **admin_login()** function checks the entered admin username and password against predefined values. If both match, it closes the admin login window. If the password is incorrect, it displays an error message. If the username is incorrect, it also displays an error message. Each error message is shown in a new Tkinter window with specific attributes. After displaying the error message, the function exits the program. In summary, **admin_login()** facilitates admin authentication and error handling for incorrect credentials, providing a streamlined user experience.
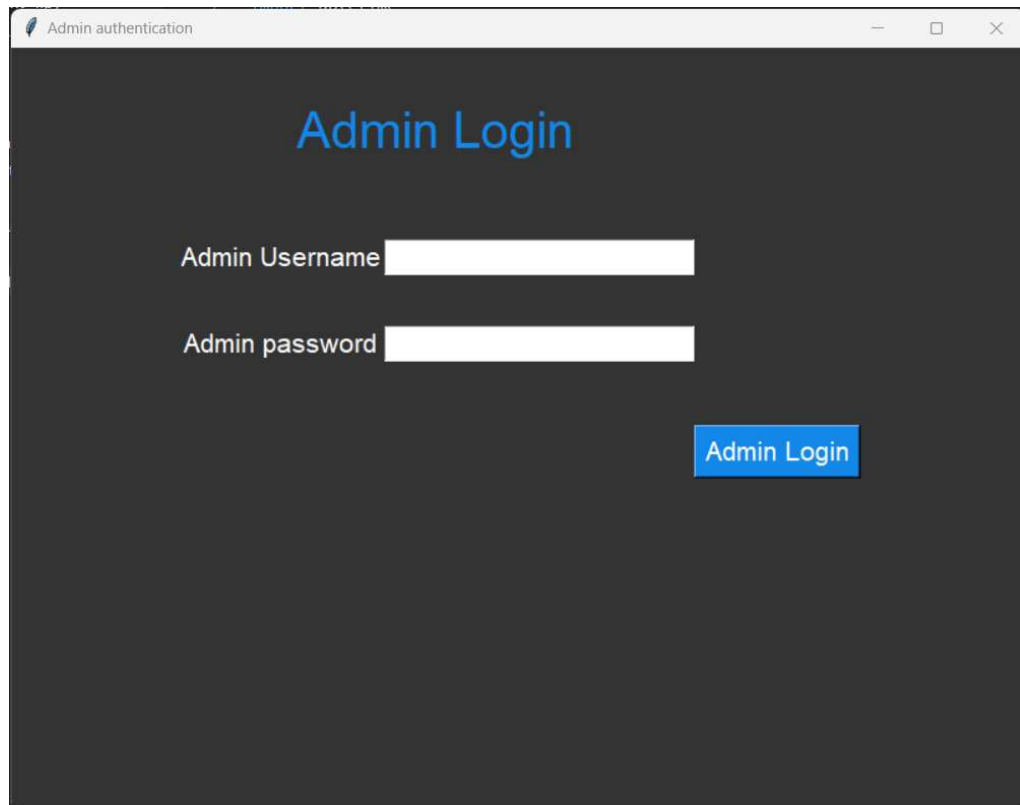
When incorrect credentials are put:
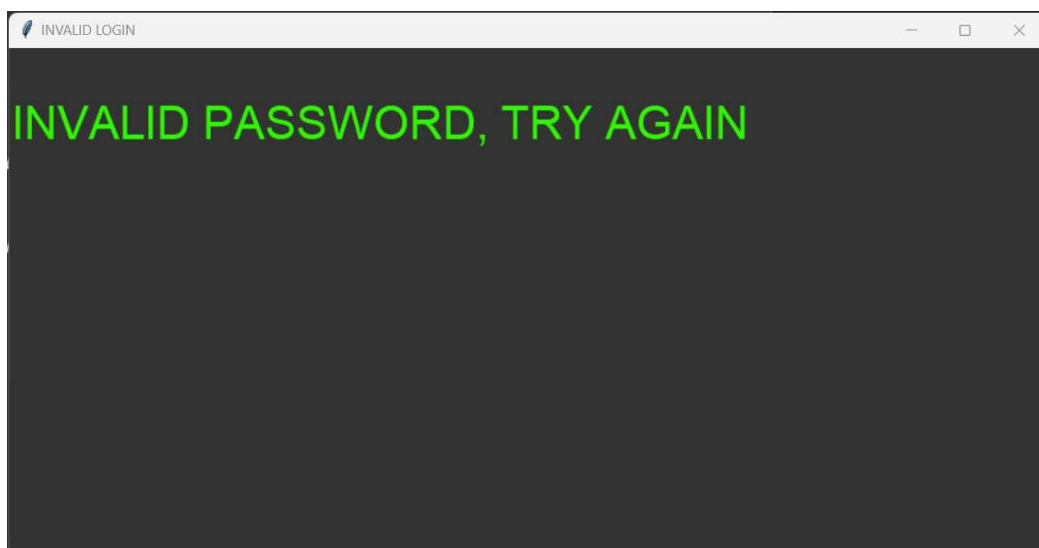
❖ enter_data()

The Python code defines a function, enter_data(), tasked with capturing and storing employee information. It utilizes global variables to manage employee data, counters, and leave details. Extracting data from Tkinter GUI input fields, including first name, last name, gender, age, employee type, ID, and password, the function appends this information to corresponding lists. Upon successful data entry, the counter increments, tracking the number of employees entered. An error message is displayed if required fields remain unfilled. Once data for the specified number of employees is entered, the Tkinter window closes, marking completion. Additionally, a confirmation message appears in a separate Tkinter window, affirming successful data saving. This function streamlines the employee data entry process through a user-friendly GUI interface

❖ Saving User Information and GUI

The provided Python code segment is responsible for creating a section within the GUI labeled "User Information" to input and display user details. It first defines a label frame titled "User Information" within the main frame, ensuring proper organization and visual separation of this section. Inside this label frame, input fields are provided for the user's first name and last name, each accompanied by respective labels for clarity. Additionally, there are dropdown menus for selecting the user's gender and type of employment, with predefined options such as "Male/Female" and "Permanent/Intern," respectively. Furthermore, a spinbox widget is included to allow users to specify their age within a predefined range.

To ensure consistent spacing and alignment of the interface elements, a loop iterates over all widgets within the "User Information" label frame, configuring them with appropriate padding values. Overall, this code segment contributes to the creation of a structured and user-friendly interface for capturing essential user information within the larger admin authentication application.
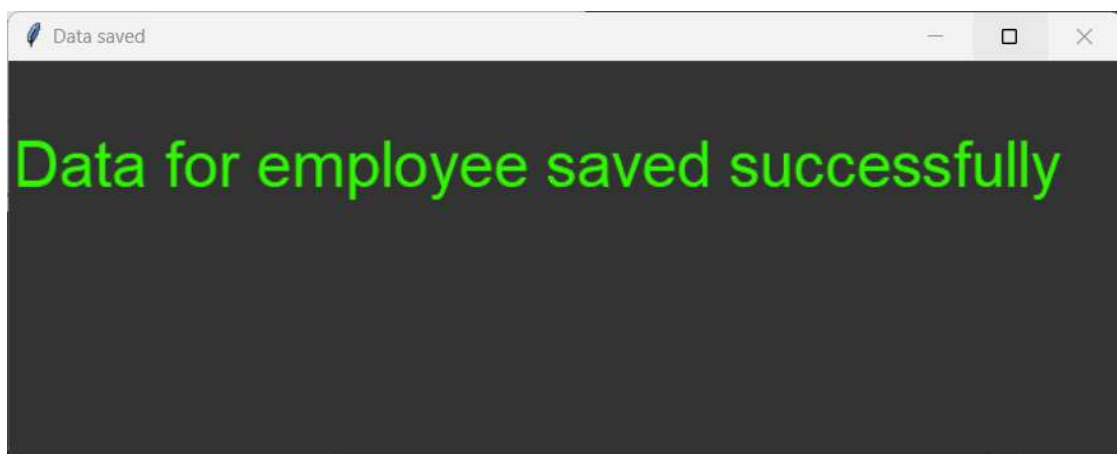
The provided Python code segment contributes to the creation of an interactive graphical user interface (GUI) for a login form within a larger application. It establishes a section labeled "Course Info" where users can input their employee ID and password. This section is contained within a labeled frame for visual organization. The user interface elements, including labels and entry fields for employee ID and password, are configured within this frame. Additionally, consistent spacing and alignment of widgets are ensured through a loop that iterates over all widgets within the frame and configures them with appropriate padding values.

Furthermore, a button labeled "Enter data" is created outside the "Course Info" frame, allowing users to submit their information. This button is associated with a command enter_data, likely triggering a function to process the entered data.

The code also initializes another GUI window titled "Login form" with default dimensions and a dark background color. This window is intended to display the login form to users.

Overall, this code segment contributes to the development of a user-friendly interface for entering employee information and initiating data submission processes within the application.

❖ Login

The **login** function is a crucial part of an employee management system, particularly for handling user authentication and various related tasks. Let's break down the code and its functionalities in more detail:

1. **Global Variables:**
   - The function begins by declaring a series of global variables that store important information about employees, such as their names, genders, ages, employee IDs, passwords, attendance status, leave entitlements, and more.

2. **User Input Retrieval:**
   - The function retrieves user inputs from entry fields in the graphical user interface (GUI). These inputs include the username, password, and venue code (likely representing the location where the user is logging in).

3. **Authentication Process:**
   - The function checks whether the entered username exists in the list of employee IDs (**empolyee_ID**). If the username is found, it verifies whether the corresponding password matches the input.
   - If the authentication is successful, the function proceeds to handle the user's login or logout process based on their current status (**status[inde]**). If the user is not logged in (**status[inde] == 0**), the function records the login time, updates attendance status, and checks for late punch-in. If the user is already logged in (**status[inde] == 1**), the function records the logout time, calculates total work hours, checks for early punch-out, and calculates overtime if necessary.

   User Interface Interaction:
   - After processing the login or logout actions, the function clears the input fields in the GUI and displays a confirmation message in a new window to inform the user about the success or failure of their login attempt.
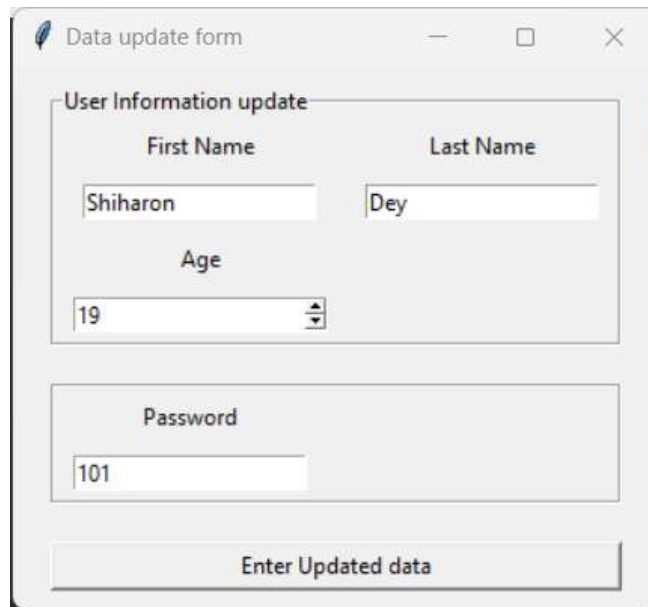
❖ Update_data()

This code snippet defines a function update_data responsible for updating employee information in the system. It initializes a tkinter window (windows) with a title "Data update form" and creates a frame (framed) within it. Inside framed, it sets up input fields for user

information (first name, last name, age) and password.

A nested function update_data_final handles finalizing the data update, retrieving updated employee information from GUI elements, modifying global variables accordingly, and displaying a confirmation message.

The function organizes user and password input fields within labeled frames (user_info_frame and courses_frame), configures their grid layout, and adds a button labeled "Enter Updated data" to trigger the update process.

Overall, it provides a structured GUI for securely updating employee information.
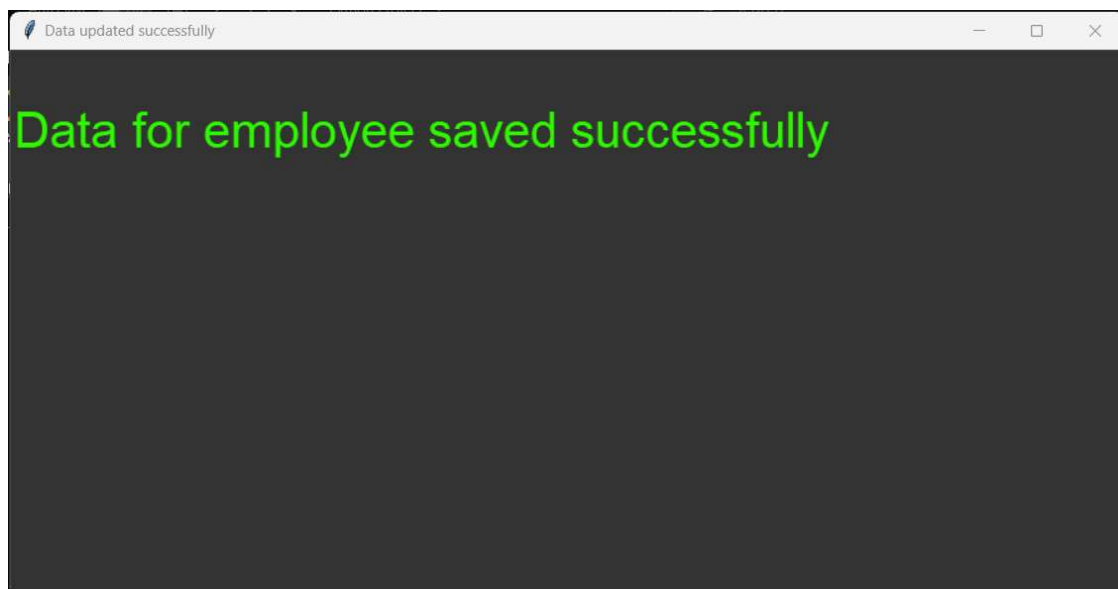
❖ Leaves()

This code defines a function named **leaves()** responsible for handling employee leave applications. It begins by creating a new tkinter window (windowed) titled "Leaves" with specific dimensions and background color. Within this window, a frame (frameds) is created to organize the GUI elements.

The frame contains:

A label (leaves_label) prompting the user to enter the number of days they intend to take as leave.

A spinbox (lday_spinbox) allowing the user to select the number of days from 0 to 20. Another label (TOL_label) indicating the type of leave, and a combobox (TOL_combobox) offering options for "Personal" or "Medical" leave.

A nested function leaves_lol() is defined to handle leave application logic based on the user's selection. It first checks if the selected type of leave is "Personal" and if the employee has enough personal leave days available (personal_leave). If so, it deducts the requested leave days from the personal leave balance and displays a confirmation message.

Similarly, if the selected leave type is "Medical" and the employee has sufficient medical leave days available (medical_leave), it deducts the requested leave days and updates the medical leave balance. Otherwise, it notifies the user about the unavailability of leave days for the current year and the deduction from next year's balance.

Finally, an "Apply leave" button (apply_button) is added to the frame to trigger the leave application process when clicked.

Overall, this function provides a user-friendly interface for employees to apply for leave, considering both personal and medical leave types, and manages the deduction of leave days from the available balance.
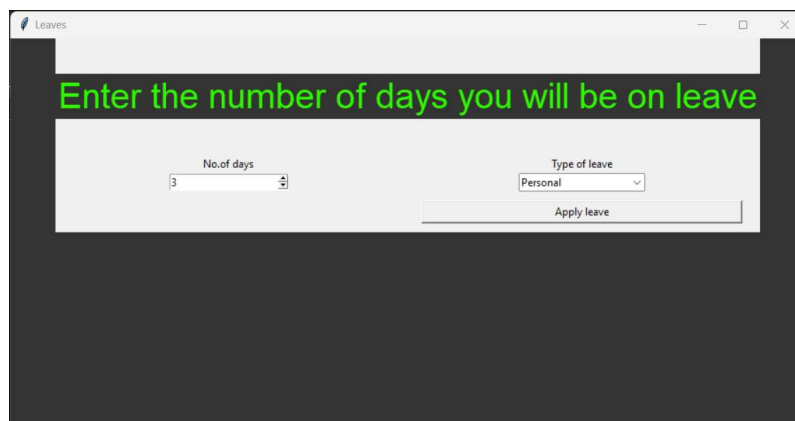
❖ Endatten_button()

The **endatten_button()** function serves to manage attendance records and document them into a text file. It starts by opening a file named "pythonprojectfile.txt" in append mode. Then, it iterates through each employee's attendance data. If an employee is marked as absent, it checks if there are any remaining leave days and records their absence if necessary. For employees with attendance, it documents their venue codes for both punching in and out, their overtime hours, and any instances of being late or leaving early. After recording attendance, it resets relevant variables. Additionally, the function sorts employees based on their overtime hours using a bubble sort algorithm, arranging them in descending order. Finally, it writes a summary in the text file listing the most hardworking employees based on overtime hours. Overall, this function efficiently manages attendance data and ensures accurate documentation for further analysis.
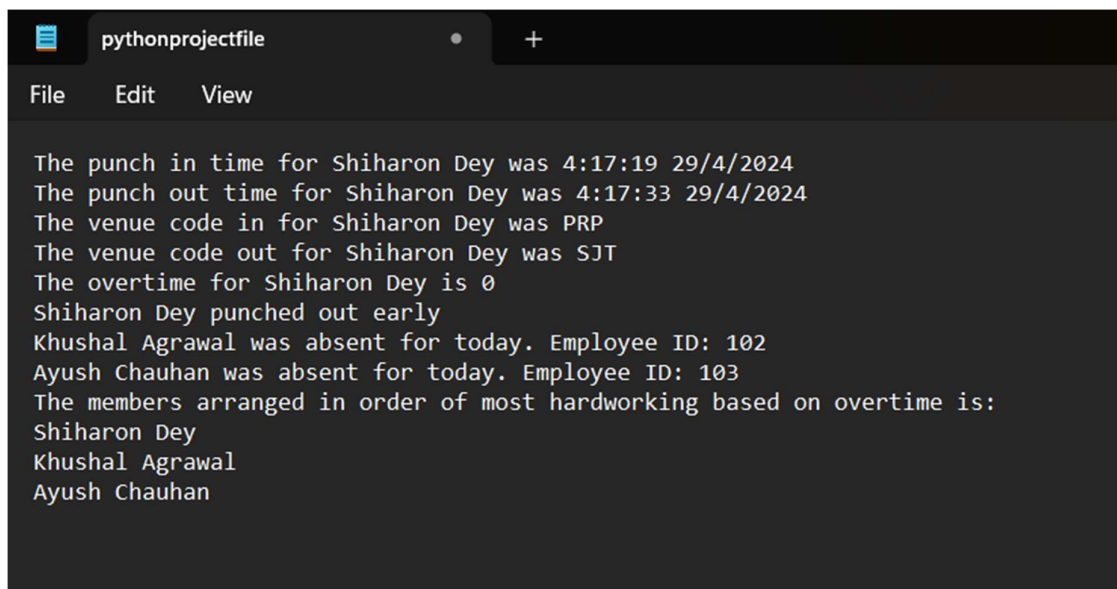
❖ **FILE**

```
The punch in time for Shiharon Dey was 4:17:19 29/4/2024
The punch out time for Shiharon Dey was 4:17:33 29/4/2024
The venue code in for Shiharon Dey was PRP
The venue code out for Shiharon Dey was SJT
The overtime for Shiharon Dey is 0
Shiharon Dey punched out early
Khushal Agrawal was absent for today. Employee ID: 102
Ayush Chauhan was absent for today. Employee ID: 103
The members arranged in order of most hardworking based on overtime is:
Shiharon Dey
Khushal Agrawal
Ayush Chauhan
```

4. REFERENCES

We extend our sincerest gratitude to our esteemed teacher for their invaluable guidance and unwavering support throughout the development of our Python project. Their expertise, patience, and encouragement have been instrumental in shaping our understanding and proficiency in Python programming. The following are some of the sources from the internet that helped us understand new and important things and made us more skilled and knowledgeable:

https://www.youtube.com/watch?v=MeMCBdnhvQs&list=PLs3IFJPw3G9IiHm9PEP1UaM
tuvACmxVMj&index=2

https://github.com/codefirstio/tkinter-data-entry/blob/main/excel-data-entry/main.py

https://github.com/codefirstio/tkinter-data-entry/tree/main/excel-data-entry

https://github.com/codefirstio/tkinter-url-shortener/blob/main/main.py

https://stackoverflow.com/

https://stackoverflow.com/questions/35829187/how-to-print-a-variable-in-a-tkinter-label-
using-python

https://stackabuse.com/writing-to-a-file-with-pythons-print-function/