# Contents

# SCHOOL INFORMATION MANAGEMENT SYSTEM

## ABOUT PYTHON

Python is a popular general-purpose programming language that can be used for a wide variety of applications. It includes high-level data structures, dynamic typing, dynamic binding, and many more features that make it as useful for complex application development as it is for scripting or "glue code" that connects components together. It can also be extended to make system calls to almost all operating systems and to run code written in C or C++. Due to its ubiquity and ability to run on nearly every system architecture, Python is a universal language found in a variety of different applications.

First developed in the late 1980s by Guido van Rossum, Python has advanced as an open-source programming language by managing public discussion through Python Enhancement Proposals (PEPs). In 2018, van Rossum stepped down as the language's Benevolent Dictator For Life (BDFL), and, as officially outlined in PEP 13, a steering council was put in place to serve as the leadership of the language.

The Python Software Foundation (PSF) is a 501(c)(3) non-profit corporation that holds the intellectual property rights behind the Python programming language. This includes Python version 2.1 and later, PyPI, the CPython reference implementation, and infrastructure to maintain the language. The PSF also provides grants for software craftship and runs multiple PyCon conferences a year.

Python is currently on its third major version and is regularly updated.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

# OVERVIEW OF THE PROJECT

In this project, we are making a School Management System using PYTHON, covering the scope of all functions performed and managed by an Educational Institution management, also introducing some new features. The basic idea is to make and maintain a database of all the data in an educational institution to maximum accuracy, efficiency and precision.

Some key highlights of the project are as follows:

- Student data collection and management
- Attendance report
- Course Management system
- User dashboard feature with checklists for priority and organizing tasks
- Tuition fee and other educational and non-educational fee management and alerting
- Score and grade Management
- Notification of previously done, ongoing and upcoming events at the institution
- Time Table and planner management system
- Three categories are taken into account:
    1. The first category is of the students who can merely access the data but do not have the authorization to edit or delete it.
    2. The second category is of the teachers who can edit or delete data but cannot create it.
    3. The third category is of the admins who can create, edit and delete data.
    4. A suitable login feature is added to determine which category the user belongs to.
- A very promising feature of this project is its flexibility. In case of any emergency, the timetable can be changed to ensure minimum chaos and optimal usage of student time.

# PYTHON LIBRARIES USED

The main theme of our project is around data management and proper storage along with user interface.

Students can easily use our dashboard system to check their schedules, attendance, reports, school fees, events, etc.

It will be an online updated system; the users must have a stable network for proper functioning of the database servers.

We will be using python libraires for accessing inbuilt modules and functions for fast development of our project. Some of the main libraries which we will be using are:

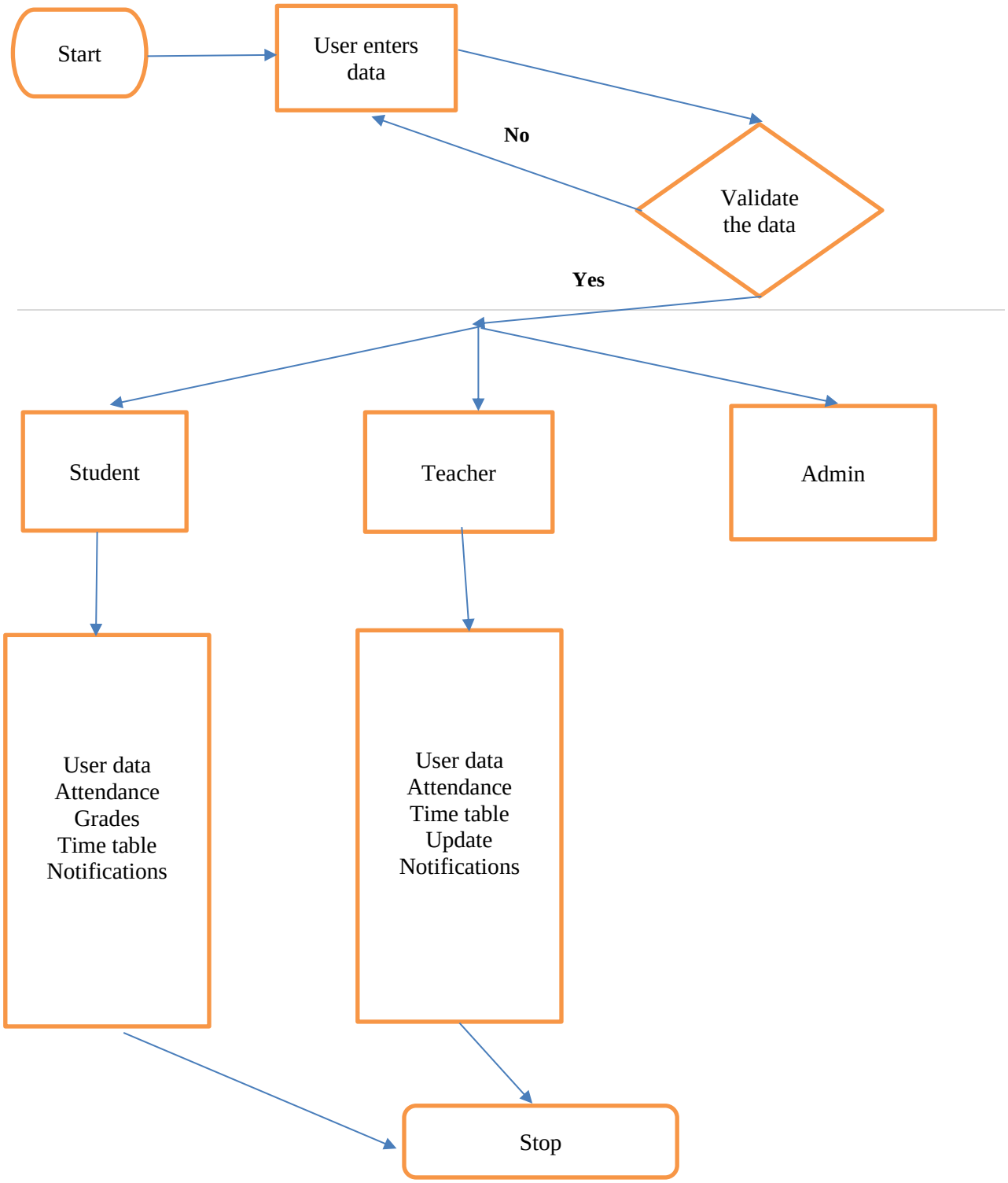
- **Tkinter:** It is a standard python library used for designing user interface. It is the fastest and simplest in terms of design and uses only python. We will be using tkinter for designing our user forms, reports and dashboard.
- **Pillow:** The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing too
- **Pandas:** It is a data analysis library used specifically for data manipulation and operation. It is mostly used because of its feature to display data in data frames and other ordered structure which makes it easier to analyze. It can also be used for tables and time series.
- **Matplotlib:** It is used for simple mathematical plotting.
- **Mysql.connector:** It is a standard python library used for integration of python with MySQL database. It allows data fetching, insertion, and update from python to SQL sides.

We will be using code editors, VS code, python and online IDEs for our project. Our project design will be divided into independent modules which will allow easy debugging and updating.

We will be creating a python extension file in script mode which could be run by a python interpreter.

Additionally, we will also create a fully executable windows application converted from the python file which could be run by any windows user without any python interpreter.

# APPLICATION FLOWCHART



# SOURCE CODE

## Main.py

```
import tkinter as tk
from PIL import Image,ImageTk
import student_user_interface as sui
import teacher_user_interface as tui
import datafile as df

#database coding work here
dataname,datapassword,datauserId=df.update_username_password()
#print(dataname,datapassword,datauserId)

#global variable
entered_data=[]

store_username="empty"
store_password="empty"
store_val="empty"

#valid check functions
def valid_credentials(user_login):
    #print("checking the wrapped data",user_login)
    My_username,My_userpassword,My_userId=user_login

    if(My_username in dataname):
        n=dataname.index(My_username)
        if(My_username==dataname[n] and My_userpassword==datapassword[n] and
My_userId==datauserId[n]):
            l = list(user_login)
            l=l.copy()

            global entered_data
            entered_data=tuple(l)
            return True
    else:
        return False

#First login frame
my_login_window=tk.Tk()
my_login_window.title("Login Window")
```

```
#getting the width and height of screen
scr_width= my_login_window.winfo_screenwidth()
scr_height= my_login_window.winfo_screenheight()
my_login_window.geometry("%dx%d" % (scr_width,scr_height))
#maximum and minimum size of screen window
my_login_window.maxsize(scr_width,scr_height)
my_login_window.minsize(int(scr_width/2),int(scr_height/2))
#background login image
my_image1= Image.open("login_bg.png")
Myimage_w,Myimage_h=my_image1.size
bg_ratio=Myimage_w/Myimage_h

bg_image= my_image1.resize((int(3*bg_ratio*scr_height/4),int(3*scr_height/4)),
Image.ANTIALIAS)
scr_bg= ImageTk.PhotoImage(bg_image)

#image canvas
login_bg=tk.Canvas(my_login_window,bg="#88cffa",width=int(3/4*scr_width))
login_bg.pack(fill="both",expand=True)
login_bg.create_image(int(scr_width/2),int(scr_height/2.05),image=scr_bg,anchor="center")

#text
login_bg.create_text(int(scr_width/2),int(scr_height/4),font=("Helvetica",int(scr_width/40)),t
ext="Login Page")
login_bg.create_text(int(scr_width/2),int(scr_height/6),font=("Helvetica",int(scr_width/35),"
underline"),text="Bangalore International Academy")

#login credentials
username_val=tk.StringVar(value="")
password_val=tk.StringVar(value="")
username_label=tk.Label(my_login_window,font=("Arial",int(scr_width/64)),text='Name')
password_label=tk.Label(my_login_window,font=("Arial",int(scr_width/64)),text='Passwor
d')
username=tk.Entry(my_login_window,width=int(scr_width/55),font=("default",int(scr_widt
h/65)),textvariable=username_val)
password=tk.Entry(my_login_window,width=int(scr_width/55),font=("default",int(scr_widt
h/65)),textvariable=password_val)

display_username_label=login_bg.create_window(int(scr_width/5),int(scr_height*2/5),wind
ow=username_label,anchor="center")
display_username=login_bg.create_window(int(scr_width*4/10),int(scr_height*2/5),window
```

```
=username,anchor="center")
display_password_label=login_bg.create_window(int(scr_width/5),int(scr_height*1/2),wind
ow=password_label,anchor="center")
display_password=login_bg.create_window(int(scr_width*4/10),int(scr_height*1/2),window
=password,anchor="center")

#single option buttons
btn_val=tk.StringVar(value="Student")
student_box=tk.Radiobutton(my_login_window,font=("default",int(scr_width/65)),text='Stu
dent',value='Student',variable=btn_val)
Teacher_box=tk.Radiobutton(my_login_window,font=("default",int(scr_width/65)),text='Te
acher',value='Teacher',variable=btn_val)
Admin_box=tk.Radiobutton(my_login_window,font=("default",int(scr_width/65)),text='Ad
min',value='Admin',variable=btn_val)
display_student_box=login_bg.create_window(int(scr_width*3/10),int(scr_height*4/7),wind
ow=student_box,anchor="center")
display_Teacher_box=login_bg.create_window(int(scr_width*4/10),int(scr_height*4/7),win
dow=Teacher_box,anchor="center")
display_Admin_box=login_bg.create_window(int(scr_width*5/10),int(scr_height*4/7),wind
ow=Admin_box,anchor="center")

#enter button
def submit():
    global store_username,store_password,store_val
    store_username=username_val.get()
    store_password=password_val.get()
    store_val=btn_val.get()
    t=(store_username,store_password,store_val)

    if(valid_credentials(t)==True):
        my_login_window.destroy()
    else:
        my_Text=login_bg.create_text(int(scr_width/2),int(scr_height/3),text="Incorrect
userdata",font=("default",int(scr_width/67),"underline"),fill="red")
        my_login_window.after(3000,login_bg.delete, my_Text)

login_btn=tk.Button(my_login_window,font=("Helvetica",int(scr_width/65),"underline"),tex
t='LOGIN',bd='5',command=lambda: [submit()])
display_login_btn=login_bg.create_window(int(scr_width*1/2),int(scr_height*5/7),window=
login_btn,anchor="center")
```

```
my_login_window.mainloop()

#print("final value",entered_data)
if(entered_data!=[]):
    if(entered_data[2]=="Student"):
        sui.student_data(entered_data[0])
        sui.Student_UI()
    elif(entered_data[2]=="Teacher"):
        tui.teacher_data(entered_data[0])
        tui.Teacher_UI()
    elif(entered_data[2]=="Admin"):
        pass
else:
    print("ERROR")
```

## datafile.py

```
import mysql.connector as sqltor
mydatabase=sqltor.connect(host="localhost",user="user",password="mypass",database="cla
ss12")
mycursor=mydatabase.cursor()
def update_username_password():
    name=[]
    password=[]
    userId=[]
    mycursor.execute("select Name,Password from studentrecord")
    data=mycursor.fetchall()
    for i in data:
        name.append(i[0])
        password.append(i[1])
        userId.append("Student")
    mycursor.execute("select Name,Password from teacher")
    data=mycursor.fetchall()
    for i in data:
        name.append(i[0])
        password.append(i[1])
        userId.append("Teacher")
    mycursor.execute("select Name,Password from admins")
    data=mycursor.fetchall()
    for i in data:
        name.append(i[0])
        password.append(i[1])
```

```
        userId.append("Admin")
    dataname=tuple(name)
    datapassword=tuple(password)
    datauserId=tuple(userId)
    return (dataname,datapassword,datauserId)
def update_studentdata(name):
    st_data={}
    mycursor.execute("select
Name,Rollno,section,gender,English,Physics,Mathematics,Chemistry,Computer,attendence
from studentrecord")
    data=mycursor.fetchall()
    for i in data:
        if(i[0]==name):
            st_data["rollno"]=i[1]
            st_data["sec"]=i[2]
            if(i[3]=="M" or i[3]=="m"):
                st_data["gender"]="Male"
            else:
                st_data["gender"]="Female"
            st_data["attend"]=i[9]
            st_data["marks"]={"English":i[4],"Physics":i[5],"Math":i[6],"Chemistry":i[7],"Comp
uter":i[8]}
    return st_data
def update_teacherdata(name):
    t_data={}
    mycursor.execute("select Name,Rollno,Subject,gender,Salary,Working_days from
teacher")
    data=mycursor.fetchall()
    for i in data:
        if(i[0]==name):
            t_data["rollno"]=i[1]
            t_data["subject"]=i[2]
            if(i[3]=="M" or i[3]=="m"):
                t_data["gender"]="Male"
            else:
                t_data["gender"]="Female"
            t_data["salary"]=i[4]
            t_data["attend"]=i[5]
    return t_data
def update_timetable():
    ttable_data={"Monday":[],"Tuesday":[],"Wednesday":[],"Thursday":[],"Friday":[]}
```

```
    mycursor.execute("select Monday,Tuesday,Wednesday,Thursday,Friday from
12d_timetable")
    data=mycursor.fetchall()
    for i in data:
        ttable_data["Monday"].append(i[0])
        ttable_data["Tuesday"].append(i[1])
        ttable_data["Wednesday"].append(i[2])
        ttable_data["Thursday"].append(i[3])
        ttable_data["Friday"].append(i[4])
    return ttable_data

def update_student_notifications(rollno):
    snotif=[]
    mycursor.execute("select StudentRollno,message from student_notification")
    data=mycursor.fetchall()
    for i in data:
        if(i[0]==rollno):
            snotif.append(i[1])
    return snotif
def update_class_notifications(mystate,mymessage=""):
    if(mystate==0):
        cnotif=[]
        mycursor.execute("select message from 12d_notification")
        data=mycursor.fetchall()
        for i in data:
            cnotif.append(i[0])
        return cnotif
    elif(mystate==1 and mymessage!=""):
        mycursor.execute("select message from 12d_notification")
        data=mycursor.fetchall()
        n=len(data)
        mycursor.execute("insert into 12d_notification values(%s,%s)",(str(n+1),mymessage))
        mydatabase.commit()
#stand alone testing
#update_username_password()
#update_studentdata()
```

## Student _User_interface.py

```
import tkinter as tk
from PIL import Image,ImageTk
import datafile as df
```

```
from pandas import DataFrame
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

#global variables
Student_class,Student_sec,Student_name= "empty","empty","empty"
Student_gender,Student_roll="empty","empty"
current_working_days=75
attended_working_days="empty"

Student_marks={"English":0,"Physics":0,"Mathematics":0,"Chemistry":0,"Computer":0}
Student_timetable={"Time":["empty"],"Monday":["empty"],"Tuesday":["empty"],"Wednesd
ay":["empty"],"Thursday":["empty"],"Friday":["empty"],"Saturday":["Holiday"],"Sunday":["
Holiday"]}
Class_notifications=["empty"]
Student_notifications=["empty"]
events=[]

def student_data(student_name):
    #Database code working here
    global
Student_name,Class_notifications,Student_notifications,Student_marks,attended_working_d
ays
    global Student_class,Student_sec,Student_gender,Student_roll,Student_timetable,events

    student_userdata=df.update_studentdata(student_name)
    Student_name=student_name
    Student_class=12
    Student_sec=student_userdata["sec"]
    Student_gender=student_userdata["gender"]
    Student_roll=student_userdata["rollno"]
    attended_working_days=student_userdata["attend"]
    time_table=df.update_timetable()
    Student_timetable={"Time":["9:00-10:00","10:00-11:00","11:00-11:30","11:30-
12:30","12:30-1:30","1:30-2:30"],}
    for i in time_table:
        Student_timetable[i]=time_table[i]
    Student_timetable["Saturday"]=["Holiday"]
    Student_timetable["Sunday"]=["Holiday"]
    Student_marks=student_userdata["marks"]
    Student_notifications=df.update_student_notifications(Student_roll)
```

```
    Class_notifications=df.update_class_notifications(0)
    events=["Computer Project on 21st"]

def refresh_window():
    Student_UI()

#student user interface function
def Student_UI():

    #Student window
    Student_window=tk.Tk()
    Student_window.title("Student Dashboard")
    #getting the width and height of screen
    scr_width= Student_window.winfo_screenwidth()
    scr_height= Student_window.winfo_screenheight()
    Student_window.geometry("%dx%d" % (scr_width,scr_height))
    #maximum and minimum size of screen window
    Student_window.maxsize(int(scr_width),int(scr_height))
    Student_window.minsize(int(scr_width/2),int(scr_height/2))

    #Partial frame for canvas and scrollbar
Student_frame=tk.Frame(Student_window,width=scr_width/2,height=scr_height/2)
    Student_frame.pack(fill="both",expand="true")
    #option menu
    menu_frame=tk.Frame(Student_frame)
    menu_frame.pack(side="top",fill="x")
    m=tk.Menu(menu_frame)
    Student_window.config(menu=m)
    def close_window():
        Student_window.destroy()

    submenu=tk.Menu(m)
    m.add_cascade(label='Options',menu=submenu)

submenu.add_command(label='Refresh',command=lambda:[close_window(),refresh_windo
w()])
    submenu.add_command(label='Exit', command=close_window)
    submenu.add_separator()

    #main canvas
```

```
Student_canvas=tk.Canvas(Student_frame,bg='#C2E5D3',width=scr_width/3,height=scr_hei
ght/3,scrollregion=(0,0,scr_width,scr_height))
    #vertical scrollbar
    vbar=tk.Scrollbar(Student_frame,orient="vertical")
    vbar.pack(side="right",fill="y")
    vbar.config(command=Student_canvas.yview)
    Student_canvas.config(width=scr_width/3,height=scr_height/3)
    Student_canvas.config(yscrollcommand=vbar.set)
    Student_canvas.pack(side="left",expand=True,fill="both")

    #student images
    my_image1= Image.open("boy_bg.png")
    Myimage1_w,Myimage1_h=my_image1.size
    boyi_ratio=Myimage1_w/Myimage1_h
boy_image=my_image1.resize((int(boyi_ratio*scr_height*2/10),int(scr_height*2/10)),
Image.ANTIALIAS)
    boy_bg= ImageTk.PhotoImage(boy_image)

    my_image2= Image.open("girl_bg.png")
    Myimage2_w,Myimage2_h=my_image2.size
    girli_ratio=Myimage2_w/Myimage2_h
girl_image=my_image2.resize((int(girli_ratio*scr_height*2/10),int(scr_height*2/10)),
Image.ANTIALIAS)
    girl_bg= ImageTk.PhotoImage(girl_image)

    #background image
    my_image3=Image.open("type1_bg.png")
    Myimage3_w,Myimage3_h=my_image3.size
    sui_ratio=Myimage3_w/Myimage3_h
sui_image=my_image3.resize((int(sui_ratio*scr_height*14/15),int(scr_height*14/15)),
Image.ANTIALIAS)
    sui_bg= ImageTk.PhotoImage(sui_image)


Student_canvas.create_image(int(scr_width/2),int(scr_height/2),image=sui_bg,anchor="cent
er")

    #nameholder display
    my_image4=Image.open("nameholder_bg.png")
    Myimage4_w,Myimage4_h=my_image4.size
    holder_ratio=Myimage4_w/Myimage4_h*1.25
```

```
holder_image=my_image4.resize((int(holder_ratio*scr_height*1/2.25),int(scr_height*1/2.25
)), Image.ANTIALIAS)
    holder_bg= ImageTk.PhotoImage(holder_image)


Student_canvas.create_image(int(scr_width/2),int(scr_height/3.5),image=holder_bg,anchor=
"center")

    #Basic info display
Student_canvas.create_text(int(scr_width/2),int(scr_height/4),font=("Helvetica",int(scr_widt
h/50)),text="Welcome "+Student_name+",")
Student_canvas.create_text(int(scr_width/1.95),int(scr_height/3),font=("Helvetica",int(scr_w
idth/60)),text="Class: "+str(Student_class)+" Section: "+Student_sec+" Roll no:
"+str(Student_roll))

    if(Student_gender=="Male"):

Student_canvas.create_image(int(scr_width/3.15),int(scr_height/3.25),image=boy_bg,anchor
="center")
    elif(Student_gender=="Female"):

Student_canvas.create_image(int(scr_width/3.15),int(scr_height/3.25),image=girl_bg,anchor
="center")
    else:

Student_canvas.create_text(int(scr_width/3.15),int(scr_height/3.25),font=("Helvetica",int(scr
_width/75)),text="no data",anchor="center")
    def data_window():
        userdata_window=tk.Toplevel(Student_window)
        userdata_window.title("User data")
        userdata_window.geometry("%dx%d" % (scr_width*4/5,scr_height/3))

        #maximum and minimum size of screen window
        userdata_window.maxsize(int(scr_width*4/5),int(scr_height/3))
        userdata_window.minsize(int(scr_width/2),int(scr_height/3))

        #other label
l=tk.Label(userdata_window,text="Data",font=("Helvetica",int(scr_width/90)))
        l.pack()
```

```
        #scrollbar
        s=tk.Scrollbar(userdata_window)
        s.pack(side="right",fill="y")

        #text
t=tk.Text(userdata_window,wrap="word",yscrollcommand=s.set,font=("Helvetica",int(scr_
width/85)),bg="#FFFDD0")
        t.pack(side="left",fill="both",expand="true")

        #display
d={"name":Student_name,"class":Student_class,"section":Student_sec,"roll
no":Student_roll,"gender":Student_gender}
        #display loop
        for i in d:
            t.insert('end',i+": "+str(d[i])+"\n")
        s.config(command=t.yview)
        t.config(state='disabled')
        #buttons

b=tk.Button(userdata_window,text="EXIT",font=int(scr_width/50),command=lambda:[user
data_window.destroy()])
        b.pack(side="bottom")
        userdata_window.mainloop()
Student_user_btn=tk.Button(Student_frame,font=("Helvetica",int(scr_width/85),"underline")
,text='User data',bd='3',command=lambda:[data_window()])

Student_canvas.create_window(int(scr_width*2/3),int(scr_height/2.45),window=Student_us
er_btn,anchor="center")

    #widget image
    timetable_image= tk.PhotoImage(file='timetable_bg.png')
    attendence_image= tk.PhotoImage(file='attendence_bg.png')
    grades_image= tk.PhotoImage(file='grades_bg.png')
    events_image= tk.PhotoImage(file='events_bg.png')

    timetable_bg= timetable_image.subsample(4,4)
    attendence_bg= attendence_image.subsample(4,4)
    grades_bg= grades_image.subsample(4,4)
    events_bg= events_image.subsample(3,3)

    #widget functions
```

```
    def show_mytimetable():
        userdata_window=tk.Toplevel(Student_window)
        userdata_window.title("Schedule")
        userdata_window.geometry("%dx%d" % (scr_width*3/4,scr_height/2))

        #maximum and minimum size of screen window
        userdata_window.resizable(0,0)

        #other label
        l=tk.Label(userdata_window,text="Time table",font=("Helvetica",int(scr_width/90)))
        l.pack()
        s=tk.Scrollbar(userdata_window)
        s.pack(side="right",fill="y")

t=tk.Text(userdata_window,wrap="word",yscrollcommand=s.set,font=("Helvetica",int(scr_
width/85)),bg="#FFFDD0")
        t.pack(side="left",fill="both",expand="true")

        #display
        for i in (Student_timetable):
            t.insert("end",i+": ")
            for j in ((Student_timetable[i])):
                t.insert('end',j+" | ")
            t.insert('end',"\n")
        t.config(state='disabled')

        s.config(command=t.yview)

b=tk.Button(userdata_window,text="EXIT",font=int(scr_width/50),command=lambda:[user
data_window.destroy()])
        b.pack(side="bottom")
        userdata_window.mainloop()

    def show_myevents():
        userdata_window=tk.Toplevel(Student_window)
        userdata_window.title("Events")
        userdata_window.geometry("%dx%d" % (scr_width*3/4,scr_height/3))
        #maximum and minimum size of screen window
        userdata_window.resizable(0,0)
        #other label
        l=tk.Label(userdata_window,text="School
```

```
Events",font=("Helvetica",int(scr_width/90)))
        l.pack()
        s=tk.Scrollbar(userdata_window)
        s.pack(side="right",fill="y")

t=tk.Text(userdata_window,wrap="word",yscrollcommand=s.set,font=("Helvetica",int(scr_
width/85)),bg="#FFFDD0")
        t.pack(side="left",fill="both",expand="true")

        #display
        t.insert('end',"Events:"+"\n")
        for i in events:
            t.insert('end',str(i)+"\n")
        t.config(state='disabled')
        s.config(command=t.yview)
b=tk.Button(userdata_window,text="EXIT",font=int(scr_width/50),command=lambda:[user
data_window.destroy()])
        b.pack(side="bottom")
        userdata_window.mainloop()

    def show_myattendence():
        #database and pandas needed
        userdata_window=tk.Toplevel(Student_window)
        userdata_window.title("Attendence Card")
        userdata_window.geometry("%dx%d" % (scr_width*7/10,scr_height*9/10))

        #maximum and minimum size of screen window
        userdata_window.resizable(0,0)

        #other label
        l=tk.Label(userdata_window,text="Attendence",font=("Helvetica",int(scr_width/90)))
        l.pack(side='top')
        #marks data
        data1={"Columns":["attended","total
days"],"attended":[attended_working_days,current_working_days]}
        dataf1=DataFrame(data1,columns=["Columns","attended"])
        #plot graph
        graph=plt.figure(figsize=(6,8),dpi=100)
        ax1=graph.add_subplot(111)
        bargraph=FigureCanvasTkAgg(graph,userdata_window)
        bargraph.get_tk_widget().pack(side="left",fill='y')
```

```
    df1=dataf1[['Columns','attended']].groupby('Columns').sum()
    df1.plot(kind="bar",legend=True,ax=ax1)
    ax1.set_title("Attendence")

    #marks digits display

l1=tk.Text(userdata_window,height=int(scr_height/3),width=int(scr_width/10),font=("Helve
tica",int(scr_width/85)))
    l1.pack(side='right',fill='y')
    l1.insert('end','Name: '+str(Student_name)+"\n")
    l1.insert('end','Total working days: '+str(current_working_days)+"\n")
    l1.insert('end','Total days attended: '+str(attended_working_days)+"\n")
    l1.config(state='disabled')
    userdata_window.mainloop()

  def show_mygrades():
    #database and pandas needed
    userdata_window=tk.Toplevel(Student_window)
    userdata_window.title("Marks Report")
    userdata_window.geometry("%dx%d" % (scr_width*7/10,scr_height*9/10))

    #maximum and minimum size of screen window
    userdata_window.resizable(0,0)

    #other label
l=tk.Label(userdata_window,text="Grades",font=("Helvetica",int(scr_width/90)))
    l.pack(side='top')

    #marks data
    mysubject=list(Student_marks.keys())
    mymarks=list(Student_marks.values())
    data1={"Subjects":mysubject,"Marks":mymarks}
    dataf1=DataFrame(data1,columns=["Subjects","Marks"])

    #plot graph
    graph=plt.figure(figsize=(6,8),dpi=100)
    ax1=graph.add_subplot(111)
    bargraph=FigureCanvasTkAgg(graph,userdata_window)
    bargraph.get_tk_widget().pack(side="left",fill='y')
    df1=dataf1[['Subjects','Marks']].groupby('Subjects').sum()
    df1.plot(kind="bar",legend=True,ax=ax1)
```

```
        ax1.set_title("Marks Report")

        #marks digits display

l1=tk.Text(userdata_window,height=int(scr_height/3),width=int(scr_width/10),font=("Helve
tica",int(scr_width/85)))
        l1.pack(side='right',fill='y')
        l1.insert('end','Name: '+str(Student_name)+"\n\nMarks obtained:\n")
        for i in Student_marks:
            l1.insert('end',str(i)+": "+str(Student_marks[i])+"\n")
        l1.config(state='disabled')
        userdata_window.mainloop()

    #option widgets

sui_timetable_btn=tk.Button(Student_frame,font=("Helvetica",int(scr_width/60),"underline"
),text='Time
table',image=timetable_bg,compound="left",bd='5',command=lambda:[show_mytimetable()]
)

sui_Attendence_btn=tk.Button(Student_frame,font=("Helvetica",int(scr_width/60),"underlin
e"),text='Attendence',image=attendence_bg,compound="left",bd='5',command=lambda:[sho
w_myattendence()])

sui_grades_btn=tk.Button(Student_frame,font=("Helvetica",int(scr_width/60),"underline"),te
xt='Grade and
score',image=grades_bg,compound="right",bd='5',command=lambda:[show_mygrades()])

sui_events_btn=tk.Button(Student_frame,font=("Helvetica",int(scr_width/60),"underline"),te
xt='Events',image=events_bg,compound="right",bd='5',command=lambda:[show_myevents(
)])
Student_canvas.create_window(int(scr_width*4/11),int(scr_height*7/11),window=sui_timet
able_btn,anchor="center")
Student_canvas.create_window(int(scr_width*4/11),int(scr_height*9/11),window=sui_Atten
dence_btn,anchor="center")
Student_canvas.create_window(int(scr_width*7/11),int(scr_height*7/11),window=sui_grade
s_btn,anchor="center")
Student_canvas.create_window(int(scr_width*7/11),int(scr_height*9/11),window=sui_event
s_btn,anchor="center")

    #notifications panel
```

```
    my_notificationframe=tk.Frame(Student_canvas)
Student_canvas.create_window(int(scr_width/8),int(scr_height/2),height=int(scr_height*5/6)
,width=int(scr_width/5.5),window=my_notificationframe,anchor="center")

    #notifcation box
    mylabel=tk.Label(my_notificationframe,font=("Helvetica",int(scr_width/75),"underline
italic"),text="Notifications")
mytextbox=tk.Text(my_notificationframe,wrap="word",height=int(scr_height*5/6),width=in
t(scr_width/6),font=("Helvetica",int(scr_width/85)),bg="#D1FFEA")

    mylabel.pack()
    mytextbox.pack(expand="true",fill="both")

    #display notifications
    mytextbox.insert('end',"Class notifications"+"\n")
    for i in range(0,len(Class_notifications)):
        mytextbox.insert('end',"*"+Class_notifications[i])
        mytextbox.insert('end',"\n")
    mytextbox.insert('end','\nStudent notifications'+'\n')
    for i in range(0,len(Student_notifications)):
        mytextbox.insert("end","-"+Student_notifications[i])
        mytextbox.insert('end',"\n")
    mytextbox.config(state='disabled')

    Student_window.mainloop()

#Stand alone testing
#student_data("Anshurup Gupta")
#Student_UI()
```

## Teacher_user_interface.py

```
import tkinter as tk
from PIL import Image,ImageTk
import datafile as df
from pandas import DataFrame
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

#global variables
Teacher_name= "empty"
Teacher_roll="empty"
```

```
Teacher_subject="empty"
Teacher_gender,Teacher_salary="empty",0
current_working_days=75
attended_working_days="empty"
Teacher_timetable={"Time":["empty"],"Monday":["empty"],"Tuesday":["empty"],"Wednesd
ay":["empty"],"Thursday":["empty"],"Friday":["empty"],"Saturday":["Holiday"],"Sunday":["
Holiday"]}
Class_notifications=["empty"]
events=[]

def teacher_data(teacher_name):
    #Database code working here
    global Teacher_name,Class_notifications,Teacher_salary,attended_working_days
    global Teacher_roll,Teacher_timetable,events,Teacher_subject,Teacher_gender

    teacher_userdata=df.update_teacherdata(teacher_name)
    Teacher_name=teacher_name
    Teacher_roll=teacher_userdata["rollno"]
    Teacher_subject=teacher_userdata["subject"]
    Teacher_gender=teacher_userdata["gender"]
    Teacher_salary=teacher_userdata["salary"]
    attended_working_days=teacher_userdata["attend"]
    time_table=df.update_timetable()
    Teacher_timetable={"Time":["9:00-10:00","10:00-11:00","11:00-11:30","11:30-
12:30","12:30-1:30","1:30-2:30"],}
    for i in time_table:
        Teacher_timetable[i]=time_table[i]
    Teacher_timetable["Saturday"]=["Holiday"]
    Teacher_timetable["Sunday"]=["Holiday"]
    Class_notifications=df.update_class_notifications(0)
    events=["Computer Project on 21st"]

def refresh_window():
    Teacher_UI()

#Teacher user interface function
def Teacher_UI():

    #Student window
    Teacher_window=tk.Tk()
    Teacher_window.title("Teacher Dashboard")
```

```
    #getting the width and height of screen
    scr_width= Teacher_window.winfo_screenwidth()
    scr_height= Teacher_window.winfo_screenheight()
    Teacher_window.geometry("%dx%d" % (scr_width,scr_height))
    #maximum and minimum size of screen window
    Teacher_window.maxsize(int(scr_width),int(scr_height))
    Teacher_window.minsize(int(scr_width/2),int(scr_height/2))
    #Partial frame for canvas and scrollbar
Teacher_frame=tk.Frame(Teacher_window,width=scr_width/2,height=scr_height/2)
    Teacher_frame.pack(fill="both",expand="true")

    #option menu
    menu_frame=tk.Frame(Teacher_frame)
    menu_frame.pack(side="top",fill="x")
    m=tk.Menu(menu_frame)
    Teacher_window.config(menu=m)

    def close_window():
        Teacher_window.destroy()
    submenu=tk.Menu(m)
    m.add_cascade(label='Options',menu=submenu)

submenu.add_command(label='Refresh',command=lambda:[close_window(),refresh_windo
w()])
    submenu.add_command(label='Exit', command=close_window)
    submenu.add_separator()

    #main canvas

Teacher_canvas=tk.Canvas(Teacher_frame,bg='#C2E5D3',width=scr_width/3,height=scr_he
ight/3,scrollregion=(0,0,scr_width,scr_height))

    #vertical scrollbar
    vbar=tk.Scrollbar(Teacher_frame,orient="vertical")
    vbar.pack(side="right",fill="y")
    vbar.config(command=Teacher_canvas.yview)
    Teacher_canvas.config(width=scr_width/3,height=scr_height/3)
    Teacher_canvas.config(yscrollcommand=vbar.set)
    Teacher_canvas.pack(side="left",expand=True,fill="both")
    #Subject images
```

```
    math_image= Image.open("math_bg.png")
    math_image_w,math_image_h=math_image.size
    mathi_ratio=math_image_w/math_image_h

    math_img= math_image.resize((int(mathi_ratio*scr_height*2/10),int(scr_height*2/10)),
Image.ANTIALIAS)
    math_bg= ImageTk.PhotoImage(math_img)
    comp_image= Image.open("comp_bg.png")
    comp_image_w,comp_image_h=comp_image.size
    compi_ratio=comp_image_w/comp_image_h

    comp_img= comp_image.resize((int(compi_ratio*scr_height*2/10),int(scr_height*2/10)),
Image.ANTIALIAS)
    comp_bg= ImageTk.PhotoImage(comp_img)
    eng_image= Image.open("eng_bg.png")
    eng_image_w,eng_image_h=eng_image.size
    engi_ratio=eng_image_w/eng_image_h
    eng_img= eng_image.resize((int(engi_ratio*scr_height*2/10),int(scr_height*2/10)),
Image.ANTIALIAS)
    eng_bg= ImageTk.PhotoImage(eng_img)
    chem_image= Image.open("chem_bg.png")
    chem_image_w,chem_image_h=chem_image.size
    chemi_ratio=chem_image_w/chem_image_h
    chem_img= chem_image.resize((int(chemi_ratio*scr_height*2/10),int(scr_height*2/10)),
Image.ANTIALIAS)
    chem_bg= ImageTk.PhotoImage(chem_img)
    phy_image= Image.open("phy_bg.png")
    phy_image_w,phy_image_h=phy_image.size
    phyi_ratio=phy_image_w/phy_image_h
    phy_img= phy_image.resize((int(phyi_ratio*scr_height*2/10),int(scr_height*2/10)),
Image.ANTIALIAS)
    phy_bg= ImageTk.PhotoImage(phy_img)

    #background image
    my_image3=Image.open("type1_bg.png")
    Myimage3_w,Myimage3_h=my_image3.size
    tui_ratio=Myimage3_w/Myimage3_h
tui_image=my_image3.resize((int(tui_ratio*scr_height*14/15),int(scr_height*14/15)),
Image.ANTIALIAS)
    tui_bg= ImageTk.PhotoImage(tui_image)
Teacher_canvas.create_image(int(scr_width/2),int(scr_height/2),image=tui_bg,anchor="cent
```

```
er")
    #nameholder display
    my_image4=Image.open("nameholder_bg.png")
    Myimage4_w,Myimage4_h=my_image4.size
    holder_ratio=Myimage4_w/Myimage4_h*1.25

holder_image=my_image4.resize((int(holder_ratio*scr_height*1/2.25),int(scr_height*1/2.25
)), Image.ANTIALIAS)
    holder_bg= ImageTk.PhotoImage(holder_image)

Teacher_canvas.create_image(int(scr_width/2),int(scr_height/3.5),image=holder_bg,anchor=
"center")
    #Basic info display
Teacher_canvas.create_text(int(scr_width/2),int(scr_height/4),font=("Helvetica",int(scr_widt
h/50)),text="Welcome "+Teacher_name+",")

Teacher_canvas.create_text(int(scr_width/1.95),int(scr_height/3),font=("Helvetica",int(scr_
width/60)),text=" Subject:"+Teacher_subject+"  Roll no:"+str(Teacher_roll))
    imager=phy_bg
    if(Teacher_subject=="Mathematics"):
        imager=math_bg
    elif(Teacher_subject=="Computer"):
        imager=comp_bg
    elif(Teacher_subject=="English"):
        imager=eng_bg
    elif(Teacher_subject=="Chemistry"):
        imager=chem_bg
    elif(Teacher_subject=="Physics"):
        imager=phy_bg

Teacher_canvas.create_image(int(scr_width/3.5),int(scr_height/3.25),image=imager,anchor=
"center")

    def data_window():
        userdata_window=tk.Toplevel(Teacher_window)
        userdata_window.title("User data")
        userdata_window.geometry("%dx%d" % (scr_width*4/5,scr_height/3))

        #maximum and minimum size of screen window
        userdata_window.maxsize(int(scr_width*4/5),int(scr_height/3))
        userdata_window.minsize(int(scr_width/2),int(scr_height/3))
```

```
        #other label
l=tk.Label(userdata_window,text="Data",font=("Helvetica",int(scr_width/90)))
        l.pack()
        #scrollbar
        s=tk.Scrollbar(userdata_window)
        s.pack(side="right",fill="y")
        #text
t=tk.Text(userdata_window,wrap="word",yscrollcommand=s.set,font=("Helvetica",int(scr_
width/85)),bg="#FFFDD0")
        t.pack(side="left",fill="both",expand="true")

        #display
        d={"name":Teacher_name,"subject":Teacher_subject,"roll
no":Teacher_roll,"gender":Teacher_gender,"Salary":Teacher_salary}
        #display loop
        for i in d:
            t.insert('end',i+": "+str(d[i])+"\n")
        s.config(command=t.yview)
        t.config(state='disabled')
        #buttons
b=tk.Button(userdata_window,text="EXIT",font=int(scr_width/50),command=lambda:[user
data_window.destroy()])
        b.pack(side="bottom")
        userdata_window.mainloop()

Teacher_user_btn=tk.Button(Teacher_frame,font=("Helvetica",int(scr_width/85),"underline"
),text='Teacher data',bd='3',command=lambda:[data_window()])

Teacher_canvas.create_window(int(scr_width*2/3),int(scr_height/2.45),window=Teacher_us
er_btn,anchor="center")

    #widget image
    timetable_image= tk.PhotoImage(file='timetable_bg.png')
    attendence_image= tk.PhotoImage(file='attendence_bg.png')
    events_image= tk.PhotoImage(file='events_bg.png')
    timetable_bg= timetable_image.subsample(4,4)
    attendence_bg= attendence_image.subsample(4,4)
    events_bg= events_image.subsample(3,3)

    #widget functions
```

```
    def show_mytimetable():
        userdata_window=tk.Toplevel(Teacher_window)
        userdata_window.title("Schedule")
        userdata_window.geometry("%dx%d" % (scr_width*3/4,scr_height/2))

        #maximum and minimum size of screen window
        userdata_window.resizable(0,0)

        #other label
        l=tk.Label(userdata_window,text="Time table",font=("Helvetica",int(scr_width/90)))
        l.pack()
        s=tk.Scrollbar(userdata_window)
        s.pack(side="right",fill="y")

t=tk.Text(userdata_window,wrap="word",yscrollcommand=s.set,font=("Helvetica",int(scr_
width/85)),bg="#FFFDD0")
        t.pack(side="left",fill="both",expand="true")

        #display
        for i in (Teacher_timetable):
            t.insert("end",i+": ")
            for j in ((Teacher_timetable[i])):
                t.insert('end',j+" | ")
            t.insert('end',"\n")
        t.config(state='disabled')

        s.config(command=t.yview)

b=tk.Button(userdata_window,text="EXIT",font=int(scr_width/50),command=lambda:[user
data_window.destroy()])
        b.pack(side="bottom")
        userdata_window.mainloop()

    def show_myattendence():
        #database and pandas needed
        userdata_window=tk.Toplevel(Teacher_window)
        userdata_window.title("Attendence Card")
        userdata_window.geometry("%dx%d" % (scr_width*7/10,scr_height*9/10))

        #maximum and minimum size of screen window
        userdata_window.resizable(0,0)
```

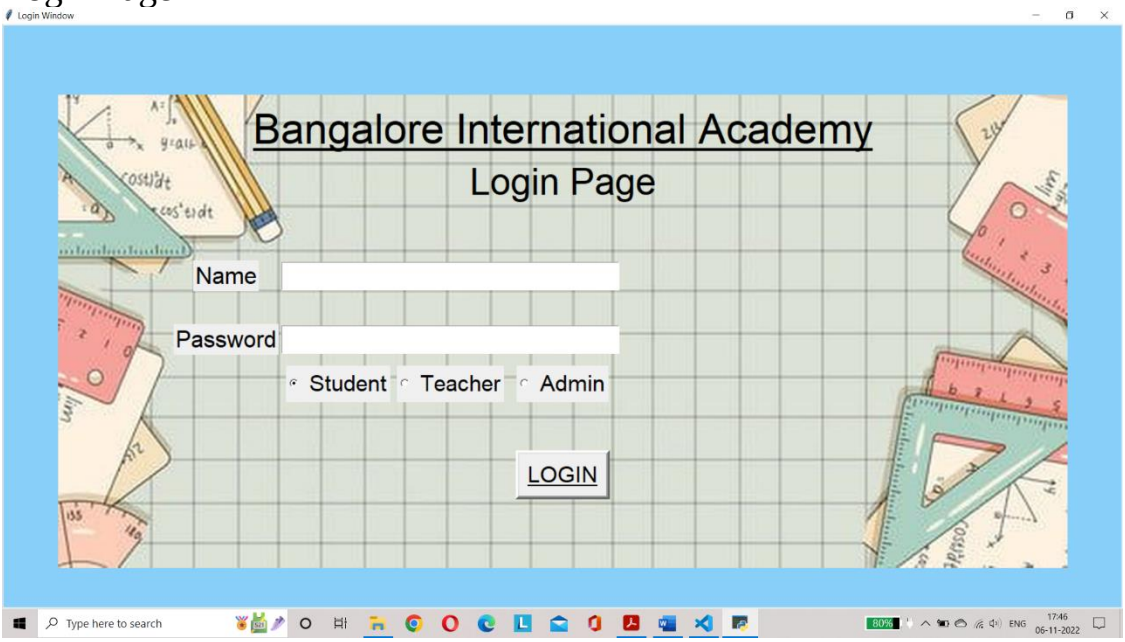
```
        #other label
l=tk.Label(userdata_window,text="Attendence",font=("Helvetica",int(scr_width/90)))
        l.pack(side='top')
        #marks data
        data1={"Columns":["attended","total
days"],"attended":[attended_working_days,current_working_days]}
        dataf1=DataFrame(data1,columns=["Columns","attended"])

        #plot graph
        graph=plt.figure(figsize=(6,8),dpi=100)
        ax1=graph.add_subplot(111)
        bargraph=FigureCanvasTkAgg(graph,userdata_window)
        bargraph.get_tk_widget().pack(side="left",fill='y')
        df1=dataf1[['Columns','attended']].groupby('Columns').sum()
        df1.plot(kind="bar",legend=True,ax=ax1)
        ax1.set_title("Attendence")

        #marks digits display
l1=tk.Text(userdata_window,height=int(scr_height/3),width=int(scr_width/10),font=("Helve
tica",int(scr_width/85)))
        l1.pack(side='right',fill='y')
        l1.insert('end','Name: '+str(Teacher_name)+"\n")
        l1.insert('end','Total working days: '+str(current_working_days)+"\n")
        l1.insert('end','Total days attended: '+str(attended_working_days)+"\n")
        l1.config(state='disabled')
        userdata_window.mainloop()

    def quick_update(m):
        global Class_notifications
        df.update_class_notifications(1,m)
        Class_notifications=df.update_class_notifications(0)

    def show_myevents():
        userdata_window=tk.Toplevel(Teacher_window)
        userdata_window.title("Class notifications")
        userdata_window.geometry("%dx%d" % (scr_width*3/4,scr_height*4/5))
        #maximum and minimum size of screen window
        userdata_window.resizable(0,0)

        #other label
```

```
        frame1=tk.Frame(userdata_window)
        frame1.grid(row=0,column=0)
        l=tk.Label(frame1,text="Class notifications",font=("Helvetica",int(scr_width/90)))
        l.grid(row=0,column=0)

        mymessage=tk.StringVar("")
        frame2=tk.Frame(userdata_window)
        frame2.grid(row=4,column=0)

b=tk.Button(frame2,text="Add",font=int(scr_width/50),command=lambda:[quick_update(m
ymessage.get()),userdata_window.destroy(),show_myevents()])
        b.pack()
        frame3=tk.Frame(userdata_window)
        t=tk.Text(frame3,wrap="word",font=("Helvetica",int(scr_width/100)),bg="#FFFDD0")
        t.grid(row=0,column=0,sticky="N")
        for i in Class_notifications:
            t.insert("end",i+"\n")
        t.config(state='disabled')
        frame3.grid(row=2,column=0)
        frame4=tk.Frame(userdata_window)
        frame4.grid(row=3,column=0)

        mymessage_label=tk.Label(frame4,font=("Arial",int(scr_width/70)),text='message')

mymessage_entry=tk.Entry(frame4,width=int(scr_width/60),font=("default",int(scr_width/7
5)),textvariable=mymessage)
        mymessage_label.grid(row=0,column=0)
        mymessage_entry.grid(row=0,column=1)
        userdata_window.mainloop()

    #option widgets
tui_timetable_btn=tk.Button(Teacher_frame,font=("Helvetica",int(scr_width/60),"underline"
),text='Time
table',image=timetable_bg,compound="left",bd='5',command=lambda:[show_mytimetable()]
)

tui_Attendence_btn=tk.Button(Teacher_frame,font=("Helvetica",int(scr_width/60),"underlin
e"),text='Working
days',image=attendence_bg,compound="right",bd='5',command=lambda:[show_myattenden
ce()])
```

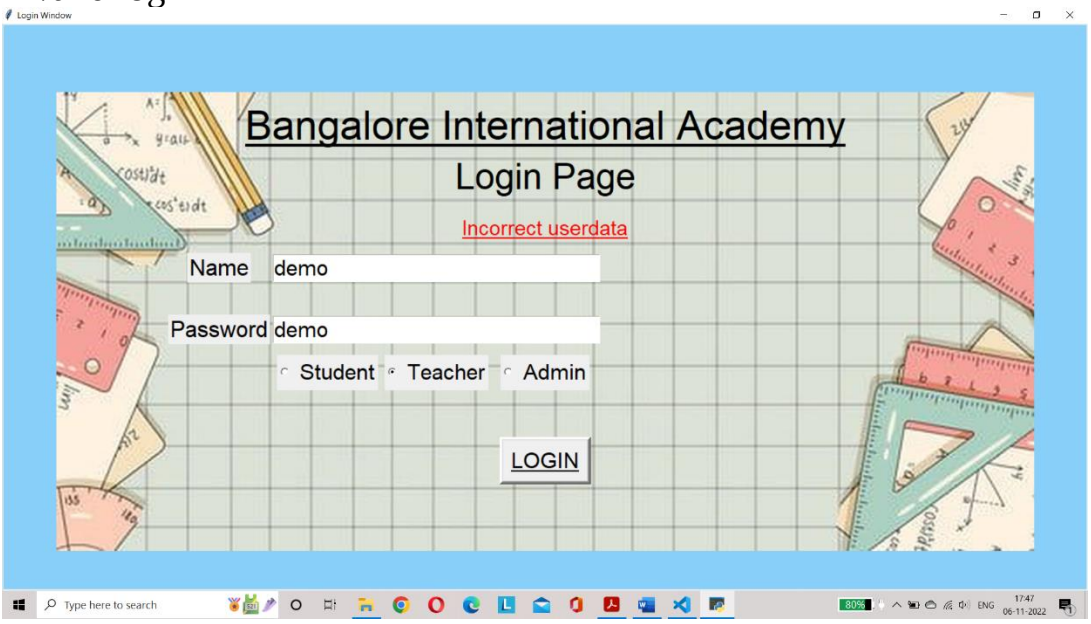
```
tui_events_btn=tk.Button(Teacher_frame,font=("Helvetica",int(scr_width/60),"underline"),text='Class
notifications',image=events_bg,compound="right",bd='5',command=lambda:[show_myevents()])


Teacher_canvas.create_window(int(scr_width*4/11),int(scr_height*7/11),window=tui_timetable_btn,anchor="center")
Teacher_canvas.create_window(int(scr_width*7/11),int(scr_height*7/11),window=tui_Attendence_btn,anchor="center")
Teacher_canvas.create_window(int(scr_width*4/11),int(scr_height*9/11),window=tui_events_btn,anchor="center")
Teacher_window.mainloop()

#Stand alone testing
#teacher_data("Kopal ma'am")
#Teacher_UI()
```

## Sample Output

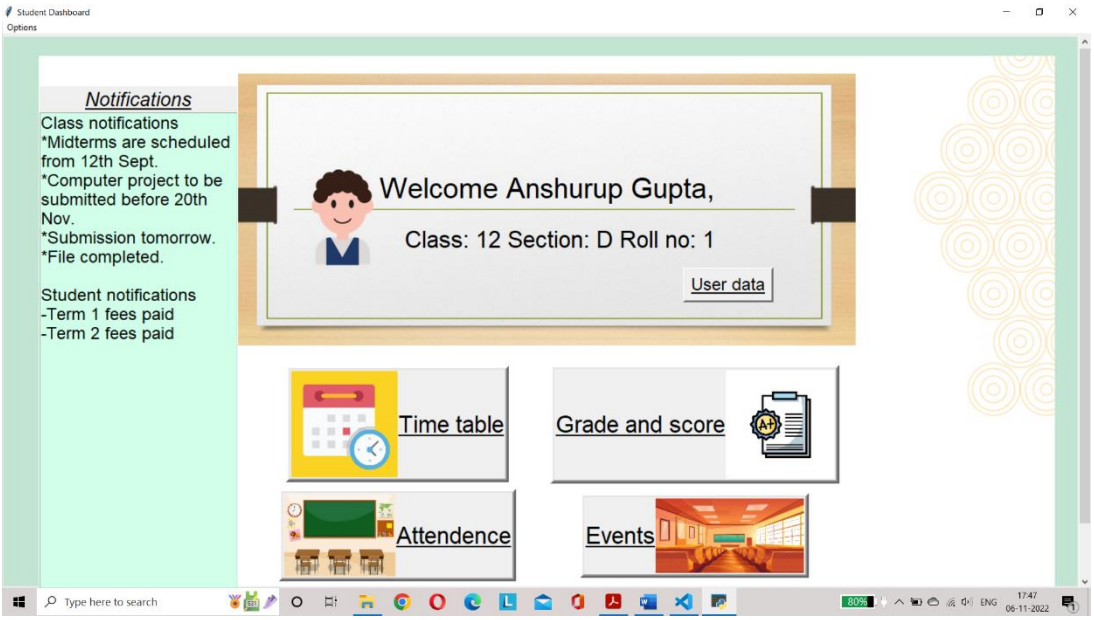
Login Page



Invalid login

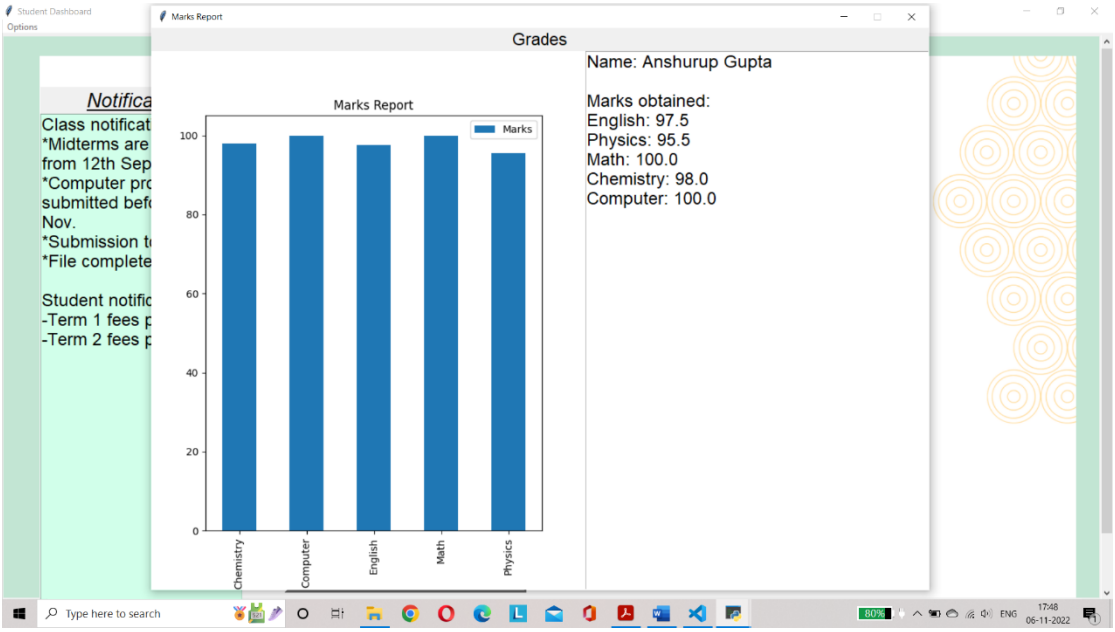



Student user interface

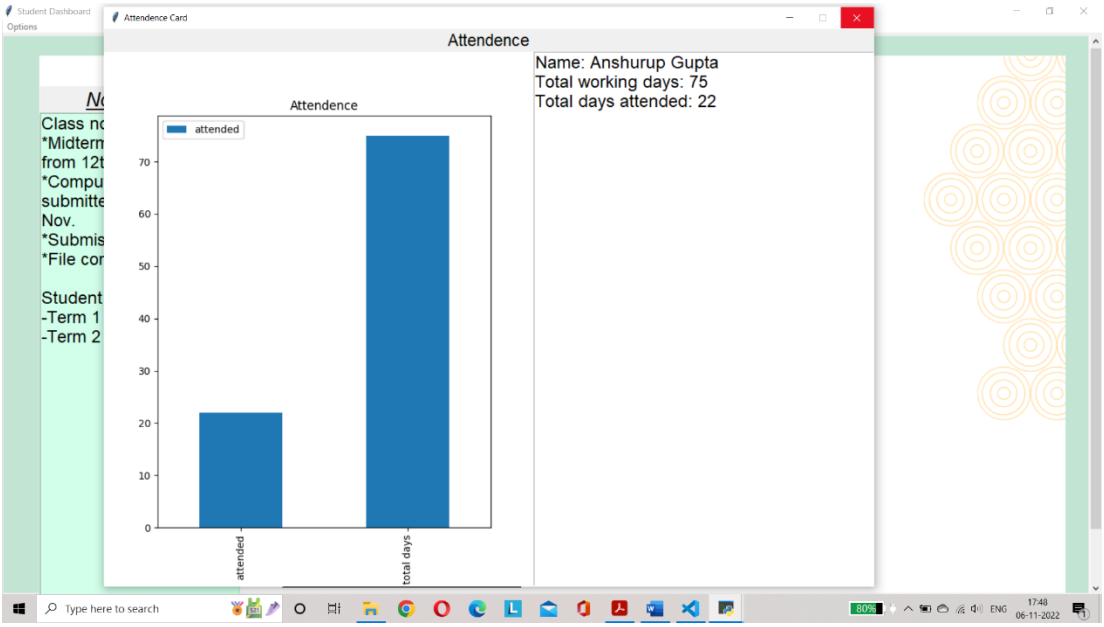


## Time table

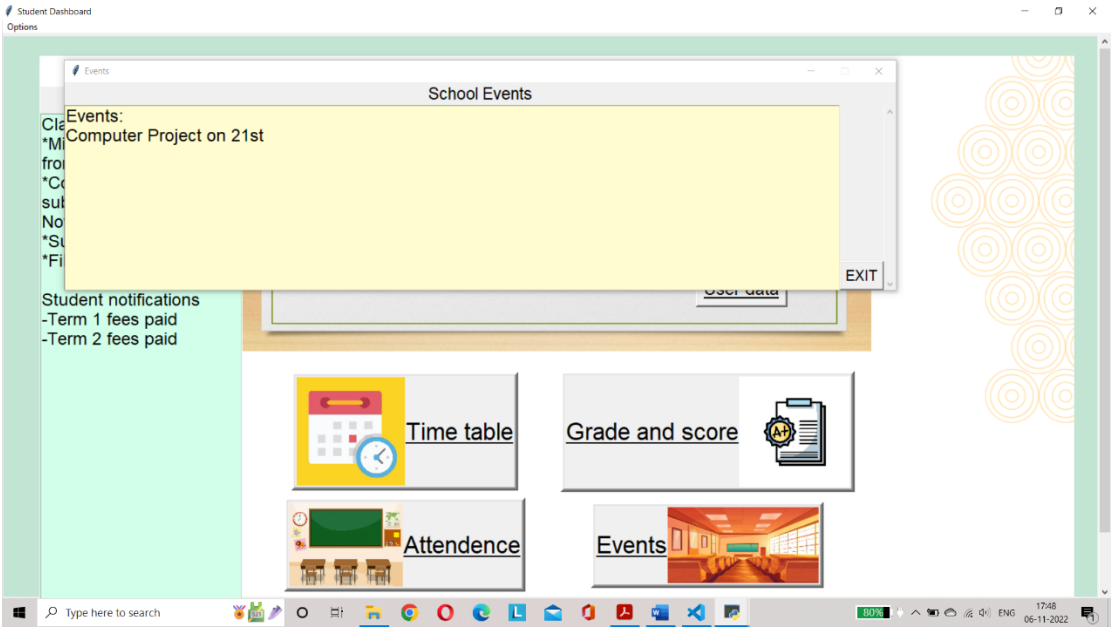

## Grades and Scores



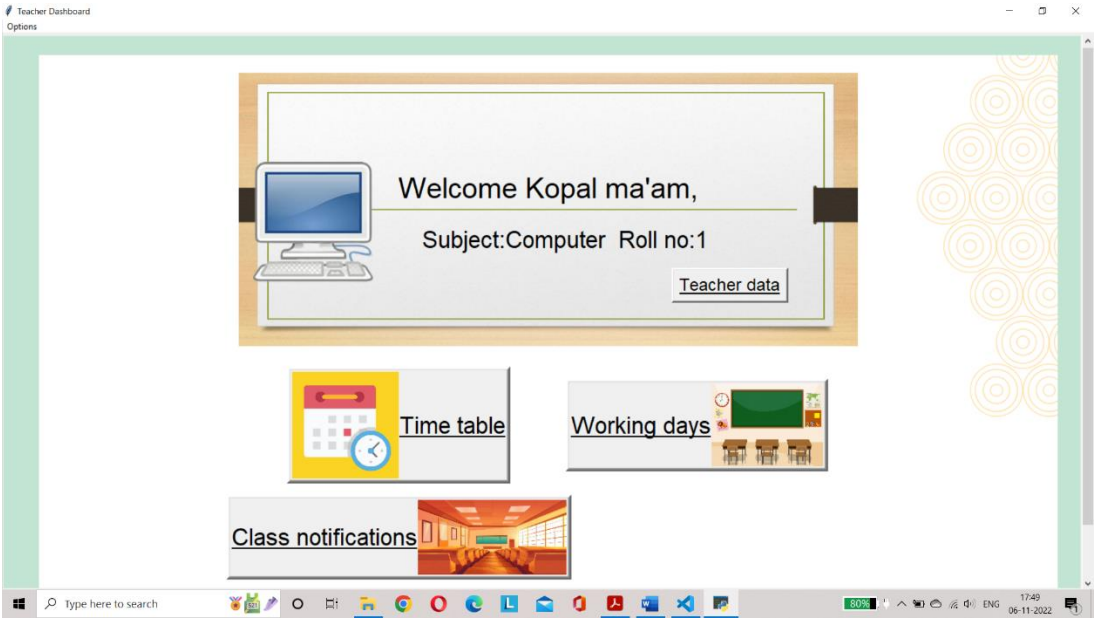
# Attendence



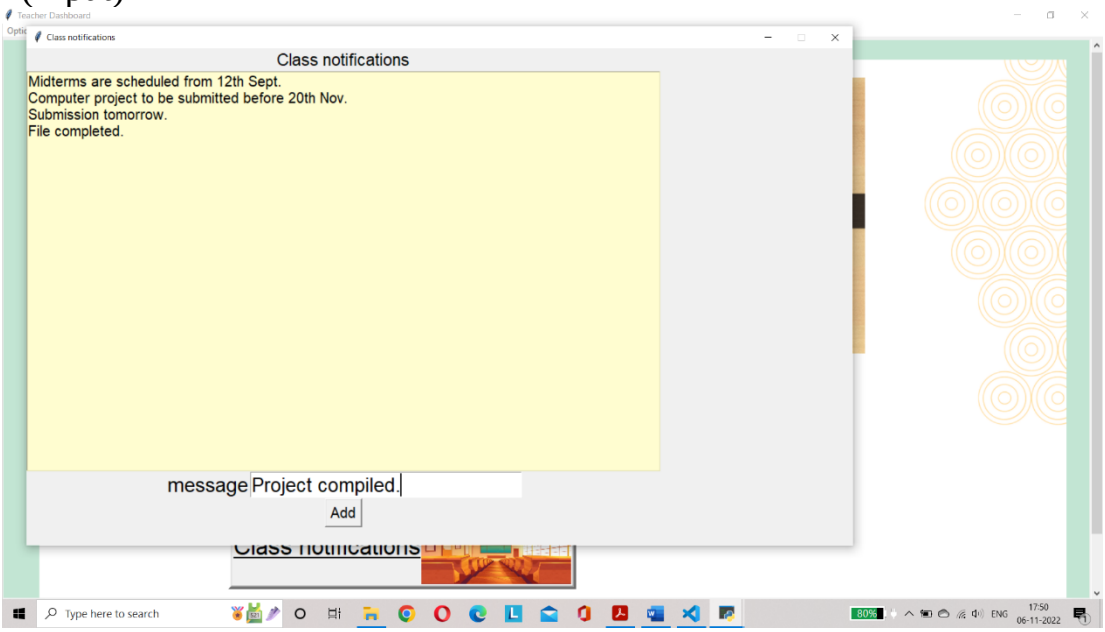

# Class events



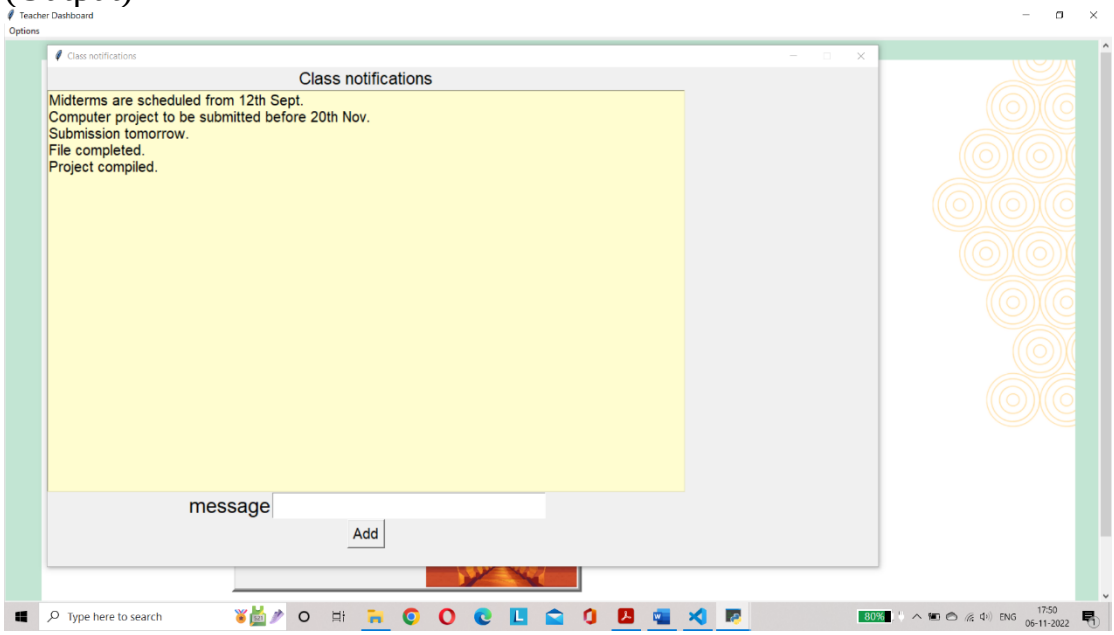
Teacher user interface




Class notifications
(Input)

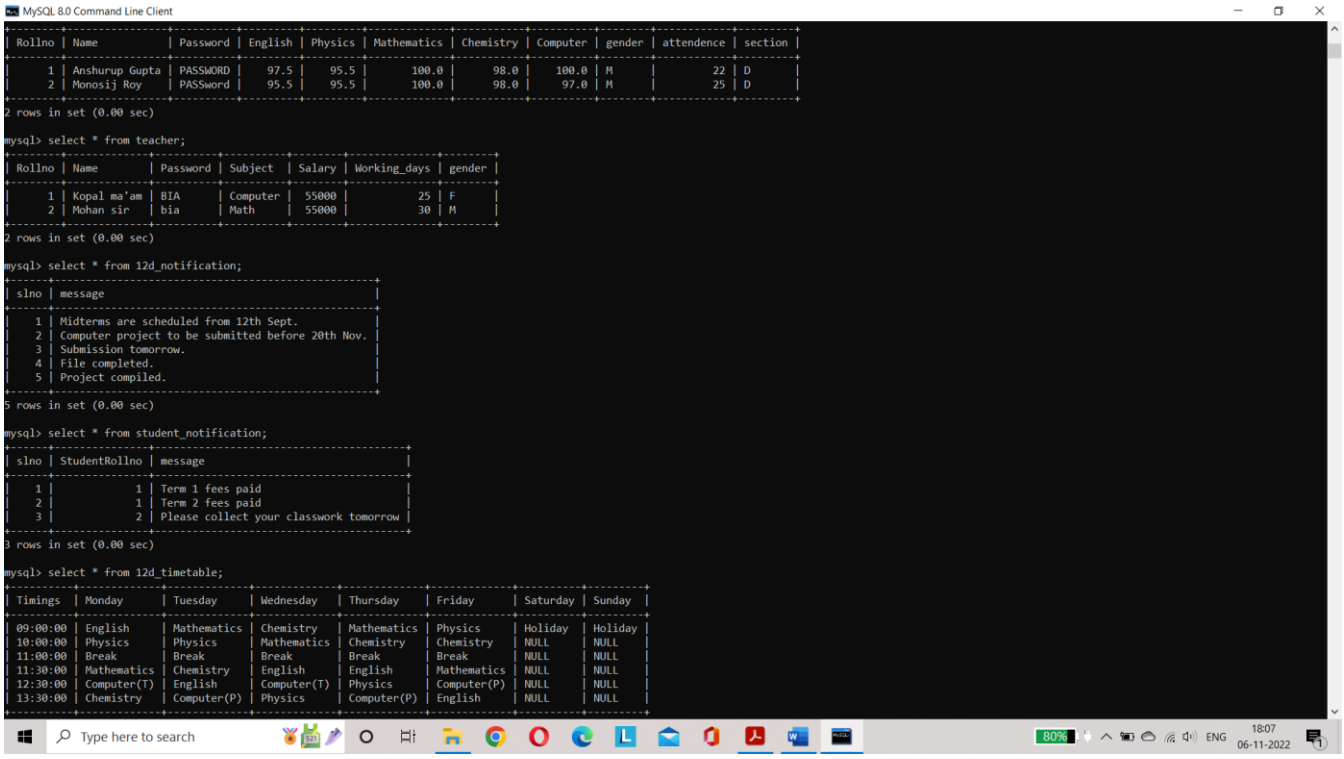


(Output)




## MySql databases:

Student record table
Teacher table
12d notification table
Student notification table
12d timetable



# Future Scope

Our project allows the students and teachers to access all their data in a simple and intuitive Gui platform. However, addition of more utilities and application to the project would allow more easy access to the data.

This project, though optimized at the current level, still has a lot of scope for improvement in the coming future.

Some of the ideas we would like to propose are as follows:

- Student behavior and participation analytics
- Integration with cloud-based database management
- Track student activity in class
- Integrate study group discussions among teacher and student or a group of students
- All feature for students to interact with each other according to section and class
- Application version for smart phone users
- File sharing features for notes and class assignments.
- Access to links with other systems such as exam portals, fee portals, etc.
- Display of regular tests along with major test results.
- Auto notifications and warning for respective student incase of low attendance, or low-test scores, or complaints.
- Better data security from SQL and python code sides.

# CONCLUSION

Python is an object-oriented, high-level language. It is indeed easy learn and use. Its inbuilt functions and libraries, and multiple other external libraries with which it can be integrated help all developers to easily solve a problem, and provide a suitable and necessary solution to it.

We have built a comprehensive School Management System application, which is not only easy to use, but is also flexible, which allows the user, be it Admins, Teachers, or Students to use it hassle free. Students can use it to see their timetable, attendance etc. and is also a necessary tool for managing school in these dire times.

# BIBLIOGRAPHY


- Computer book – Sumita Arora
- https://www.w3schools.com/python/default.asp
- https://opensource.com/resources/python
- https://www.geeksforgeeks.org/python-gui-tkinter/
- https://www.geeksforgeeks.org/python-pillow-a-fork-of-pil/
- https://stackoverflow.com