COMPUTER SCIENCE INVESTIGATORY PROJECT

2022-2023

HOLISTIC STUDENT DEVELOPMENT SYSTEM

Name: ADISHREE GUPTA

Class: 12

Board Roll Number:

Certificate

This is to certify that Adishree Gupta of Class XII of HAL Public School have successfully completed their project in **Computer Science** for the All-India Senior Secondary School Certificate Examination prescribed by the CBSE in the year 2022-23.

Date

Name **Roll Number**

ADISHREE GUPTA

Signature of External Examiner

Signature of Internal Examiner

Principal

ACKNOWLEDGEMENT

The success of a project depends upon the persistent efforts of the team projecting it and the sustained support received from a few others who are equally responsible for their precious appreciation of such endeavours. Our strength is all due to our honourable Principal **Smt.** ***Seema Gupta*** who has been an unending source of inspiration and support towards accomplishment of this project.

We would like to express our deepest sense of gratitude to our computer teacher **Ms. Savareena Ilango,** without whom we could not have successfully completed this project.

We would also like to thank all my friends who helped us to create such a project.

Our personal gratitude is extended towards our parents, who have been a constant source of engagement and support in the success of this project. Last but not the least we want to thank Almighty for enlightening, strengthening, and guiding us in the completion of this project.

INDEX

SYNOPSIS

Holistic Student Development System is a software to help the school authorities. In the current system all the activities manual and involve costly purchase of registers and paper. This is time consuming and costly. Our solution deals with the various activities related to the students.

There are mainly 3 modules in this software. User Registration, User Authentication module and Student Module. The Software can be used to register school administrators. The Administrator has the power to add new user and can edit and delete a user. A student details and can be added/ updated or deleted based on school requirements.

Validations have been in built on the screens to prevent erroneous data feed and promote correct user authentication for system access.

HARDWARE & SOFTWARE SPECIFICATIONS

HARDWARE

Processor: 4GZ or more

RAM : 8GB or more

VDU: LCD

SOFTWARE

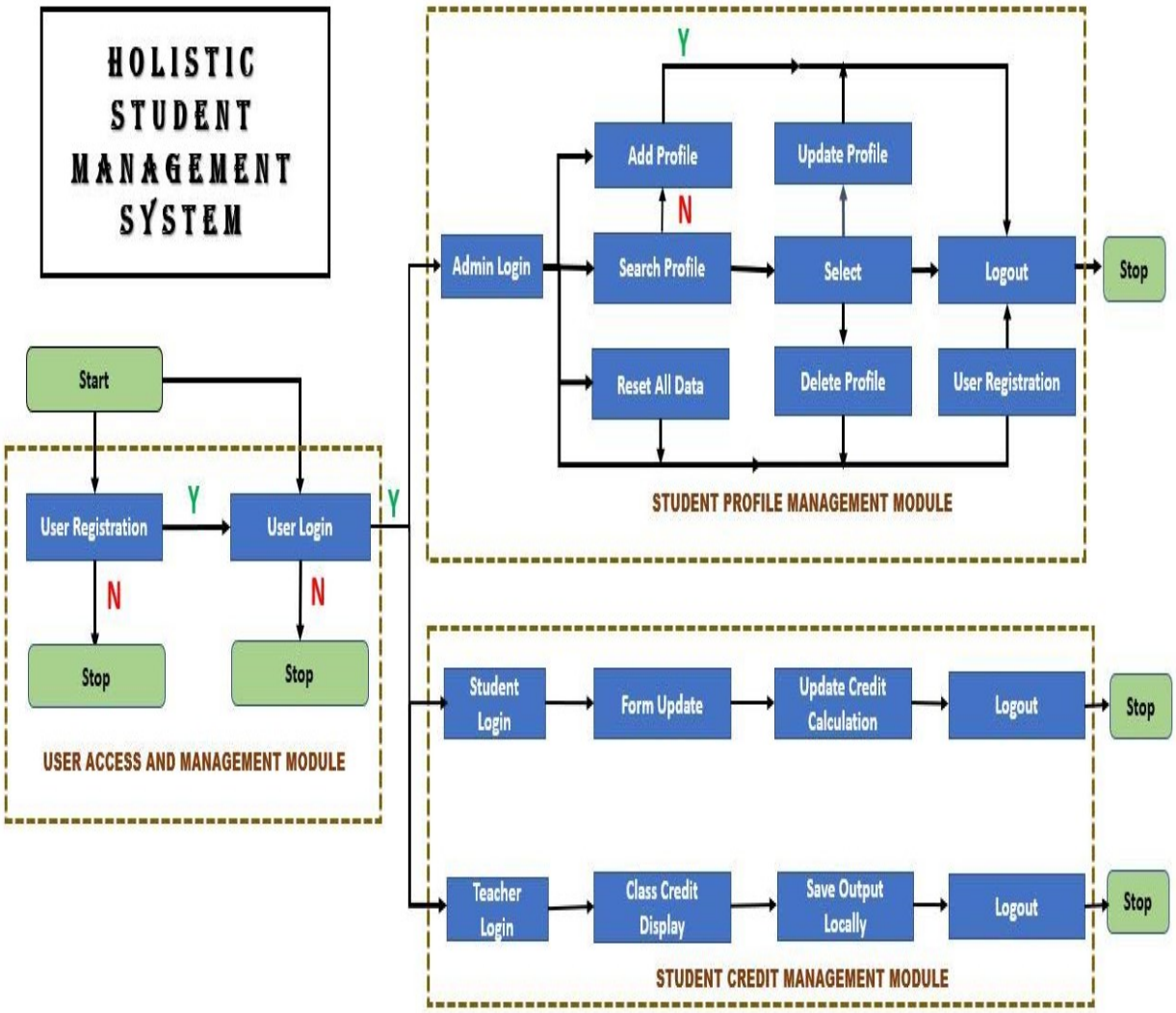
Operating System: Windows 10

Python: 3.10 64 bits

8.0.29 MySQL Community Server – GPL

MySQL Workbench 8.0 CE

FLOW CHART


HOLISTIC STUDENT MANAGEMENT SYSTEM
Start
User Registration
Y
User Login
Y
N
N
Stop
Stop
USER ACCESS AND MANAGEMENT MODULE
Y
Add Profile
Update Profile
N
Admin Login
Search Profile
Select
Logout
Stop
Reset All Data
Delete Profile
User Registration
STUDENT PROFILE MANAGEMENT MODULE
Student Login
Form Update
Update Credit Calculation
Logout
Stop
Teacher Login
Class Credit Display
Save Output Locally
Logout
Stop
STUDENT CREDIT MANAGEMENT MODULE

SYSTEM DESIGN

- The application has three screens: a) User Access & Management b) Student Profile Management Module c) Student Credit Management Module
- The user registration page is the home/landing page for the application
- The Admin Registration page provides option to both register and login. A user can click login and go to the Admin Login Page
- This page asks the user to fill up the contact number and a challenge question.
- The combination of the above will be requested offline outside the application by the system administrator in case the user and / or password is lost by an application user.
- The Admin Login screen is a simple login screen which on successful entry will open up the Student Profile Manager
- The Student Profile Manager helps to add, update, delete, search student profiles. A scroll bar is provided in the tree view output below in case the number of entries exceed fifteen
- Once the work is completed, the user can logout from the screen.
- The Student Registration page provides option to both register and login to the Credit Management System
- On logging in they can connect to their specific class and then fill up the credit questionnaire and earn extra points
- The Teacher Registration page provides option to both register and login to the Credit Management System
- On logging in they can connect to their specific class and then review the points of the students by choosing a specific class
- From the report display they can save the output in a local system

DATA DICTIONARY

LIBRARY MODULES AND THEIR PURPOSES

S.No	LIBRARY MODULES	PURPOSE
1	Tkinter()	Used to create simple GUI apps
2	Tkinter Treeview	display data in both tabular and hierarchical structures
3	tkinter.messagebox	Display the message boxes in the python applications
4	Image Tk	Support to create and modify Tkinter BitmapImage and PhotoImage objects from PIL images
5	Mysql.connector	Enables python programs to access my SQL databases
6	subprocess	To run new applications or programs through Python Code by creating new processes. It also helps to obtain the input/output/error pipes as well as the exit codes of various commands
S.No	Package	PURPOSE
1	Python Imaging Library	support for opening, manipulating, and saving many different images file formats.

FUNCTIONS AND THEIR PURPOSES

S.No	FUNCTIONS	PURPOSE
1	configure(background=")	Configure window background color
2	geometry("")	set dimensions of the Tkinter window display
3		
4		
5		

SOURCE CODE

```
#*************************************************************************
#'*                    FLOWER BOX                                         *
#'*************************************************************************
#'*   FILENAME:        User_registration.IPYNB                            *
#'*   AUTHOR:          ADISHREE GUPTA                                     *
#'*   COMPUTER NUMBER:  IN-00003291                                       *
#'*   DATE:            09/10/2022                                         *
#'*   PURPOSE:    THE BELOW APPLICATION WILL BE USED TO MANAGE            *
#'*            USER REGISTRATION FOR ALL USER PROFILES                    *
#'*************************************************************************

from tkinter import *
from tkinter import ttk
from tkinter.messagebox import *
import mysql.connector as sq

def login_window():
    import User_login

def clear():

    entrycontact.delete(0, END)
    entrypassword.delete(0, END)
    entryconfirmpassword.delete(0, END)
    entryusername.delete(0, END)
    entryfirstname.delete(0, END)
    entrylastname.delete(0, END)
    entryanswer.delete(0, END)
    check.set(0)

def register():
    if entryfirstname.get() == '' or entrylastname.get() == '' or entrycontact.get() == '' or entryusername.get() == ''
or \
            entrypassword.get() == '' or entryconfirmpassword.get() == '' or entryanswer.get() == '':
        showerror('Error', "All Fields Are Required", parent=root)
    elif len(entrycontact.get()) != 10:
        showerror('Error', "Phone number must be of 10 digits", parent=root)

    elif len(entrypassword.get()) != 8:
        showerror('Error', "Password must be alphanumeric and must have 8 characters", parent=root)

    elif entrypassword.get() != entryconfirmpassword.get():
        showerror('Error', "Password Mismatch", parent=root)

    elif check.get() == 0:
        showerror('Error', "Please Agree To Our Terms & Conditions", parent=root)

    else:
```

```
    try:
        con = sq.connect(host='localhost',user='root',password='HPSdb2018',
database='credit_management_system')
        cur = con.cursor()
        code='insert into user(f_name,l_name,contact,username,date_of_birth,password)
values(%s,%s,%s,%s,%s,%s)'
        values=(entryfirstname.get(),
entrylastname.get(),entrycontact.get(),entryusername.get(),entryanswer.get(), entrypassword.get(),)
        cur.execute(code,values)
        con.commit()
        con.close()
        showinfo('Success', "Registration Successful", parent=root)
        clear()

    except Exception as e:
        showerror('Error', f"Error due to: {e}", parent=root)


root = Tk()
root.configure(bg="Navy Blue")
root.title('USER REGISTRATION')

titleLabel = Label(root, text='Registration Form', font=('Arial', 40, 'bold '),bg='honeydew2',
            fg='navy blue', )
titleLabel.place(x=250, y=50)

firstnameLabel = Label(root, text='First Name', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
                fg='black', )
firstnameLabel.place(x=225, y=140)
entryfirstname = Entry(root, font=('Bookman Old Style', 16), bg='white')
entryfirstname.place(x=200, y=175, width=250, height=24)

lastnameLabel = Label(root, text='Last Name', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
                fg='black', )
lastnameLabel.place(x=600, y=140)
entrylastname = Entry(root, font=('Bookman Old Style', 16), bg='white')
entrylastname.place(x=550, y=175, width=250,height=24)

contactLabel = Label(root, text='Contact Number', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
            fg='black', )
contactLabel.place(x=225, y=230)
entrycontact = Entry(root, font=('Bookman Old Style', 16), bg='white')
entrycontact.place(x=200, y=265, width=250,height=24)

DOBLabel = Label(root, text='Date of Birth', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
                fg='black', )
DOBLabel.place(x=600, y=230)

entryanswer = Entry(root, font=('Bookman Old Style', 16), bg='white')
entryanswer.place(x=550, y=265, width=250,height=24)

UsernameLabel = Label(root, text='Username', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
            fg='black', )

UsernameLabel.place(x=225, y=330)
entryusername = Entry(root, font=('Bookman Old Style', 16), bg='white')
```

```
entryusername.place(x=550, y=330, width=250,height=24)

passwordLabel = Label(root, text='Password', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
                  fg='black', )
passwordLabel.place(x=225, y=390)
entrypassword = Entry(root,show="*", font=('Bookman Old Style', 16), bg='white')
entrypassword.place(x=200, y=430, width=250,height=24)

confirmpasswordLabel = Label(root, text='Confirm Password', font=('Bookman Old Style', 16, 'bold'),
                       bg='wheat2',
                       fg='black', )
confirmpasswordLabel.place(x=575, y=390)
entryconfirmpassword = Entry(root,show="*", font=('Bookman Old Style', 16), bg='white')
entryconfirmpassword.place(x=550, y=430, width=250,height=24)

check = IntVar()
checkButton = Checkbutton(root, text='I Agree All The Terms & Conditions', variable=check, onvalue=1,
                     offvalue=0, font=('times new roman', 10,), bg='white')
checkButton.place(x=215, y=490)

registerbutton = Button(root,text="Register",font=('Bookman Old Style', 18, 'bold'), bd=0, cursor='hand2',
fg='white', bg='grey', activebackground='white'
                    , activeforeground='white', command=register)
registerbutton.place(x=680, y=480)

loginbutton1 = Button(root,text="Login", font=('Bookman Old Style', 18, 'bold',), bd=0, cursor='hand2',
fg='white', bg='grey', activebackground='gold',
                  activeforeground='gold', command=login_window)
loginbutton1.place(x=550, y=480)

root.mainloop()


#*************************************************************************
#'*                    FLOWER BOX                                         *
#'*************************************************************************
#'*   FILENAME:        User_login.IPYNB                                  *
#'*   AUTHOR:          ADISHREE GUPTA                                    *
#'*   COMPUTER NUMBER:  IN-00003291                                      *
#'*   DATE:            09/10/2022                                        *
#'*   PURPOSE:    THE BELOW APPLICATION WILL BE USED TO MANAGE           *
#'*                    LOGIN FOR ALL USER PROFILES                       *
#'*************************************************************************

from tkinter import *
from tkinter import ttk
from tkinter import *
from tkinter import ttk
from tkinter.messagebox import *
import mysql.connector as sq

def register_window():
   import User_registration
def interface():
   import three_levels
def back():
```

```
    import User_registration
def signin():
    if userentry.get() == '' or passentry.get() == '':
        showerror('Error', 'All Fields Are Required')

    else:
        try:
            con = sq.connect(host='localhost',user='root',password='HPSdb2018',
database='credit_management_system')
            cur = con.cursor()
            cur.execute('select * from user where username=%s and password=%s', (userentry.get(),
passentry.get()))
            row = cur.fetchone()
            if row == None:
                showerror('error', 'Invalid username or Password')
                root.destroy()
            else:
                showinfo('Authenticated', 'Welcome')
                interface()
            con.close()
        except Exception as e:
            showerror('Error', f"Error due to: {e}", parent=root)

def forgot_password():

    def authentication():

        if user_entry.get()=="  " or DOB_entry.get()=="  " or newpass_entry=="  ":
            showerror('Error', 'All Fields Are Required',parent=window)
        else:
            con = sq.connect(host='localhost',user='root',password='HPSdb2018',
database='credit_management_system')
            cur = con.cursor()
            query='select * from user where username=%s and date_of_birth=%s'
            cur.execute(query,(user_entry.get(),DOB_entry.get()))
            row=cur.fetchone()
            if row==None:
                message.showerror('Error','Incorrect username and date of birth',parent=window)
            else:
                query='update user set password=%s where username=%s and date_of_birth=%s'
                cur.execute(query,(newpass_entry.get(),user_entry.get(),DOB_entry.get()))
                con.commit()
                con.close()
                showinfo('Success','Password is reset,please login with new password',parent=window)
                window.destroy()

    window=Toplevel()
    window.title("Forgot password")

    window.configure(bg="navy blue")

    heading_label = Label(window,text="Password Manager",font=('Arial','50' , 'bold'),bg='navy
blue',fg='white')
    heading_label.place(x=200,y=60)

    userlabel=Label(window,text=" Username   ",font=('Arial','20' , 'bold'),bg='navy blue',fg='white')
```

```
    userlabel.place(x=200,y=200)

    user_entry=Entry(window,width=25,fg='black',font=('Arial','20','italic'),bd=0)
    user_entry.place(x=520,y=200)

    DOBlabel=Label(window,text="Date of Birth",font=('Arial','20' , 'bold'),bg='navy blue',fg='white')
    DOBlabel.place(x=200,y=250)

    DOB_entry=Entry(window,width=25,fg='black',font=('Arial','20','italic'),bd=0)
    DOB_entry.place(x=520,y=250)

    newpasslabel=Label(window,text="New Password",font=('Arial','20' , 'bold'),bg='navy      blue',fg='white')
    newpasslabel.place(x=200,y=300)

    newpass_entry=Entry(window,width=25,fg='black',font=('Arial','20','italic'),bd=0)
    newpass_entry.place(x=520,y=300)

    resetbutton = Button(window,text=' RESET ',font=('arial', 15, 'bold'),fg='white',bg='grey',cursor='hand2',
            activebackground='gray20', activeforeground='white', command=authentication)
    resetbutton.place(x=800, y=380)

    window.mainloop()

root = Tk()
root.configure(background="navy blue")
root.title('USER LOGIN')

frame = Frame(root, bg='white', width=560, height=320)
frame.place(x=50, y=140)

label = Label(root, text="USER LOGIN", font=('Arial', 20),bg="grey",fg='white')
label.grid(row=0, column=0, columnspan=8, rowspan=2, padx=50, pady=105)

mailLabel = Label(frame, text='Username', font=('arial', 22, 'bold'), bg='white', fg='black')
mailLabel.place(x=200, y=32)
userentry = Entry(frame, font=('arial', 22,), bg='white', fg='black')
userentry.place(x=110, y=70)

passLabel = Label(frame, text='Password', font=('arial', 22, 'bold'), bg='white', fg='black')
passLabel.place(x=200, y=120)
passentry = Entry(frame,show="*",font=('arial', 22,), bg='white', fg='black')
passentry.place(x=110, y=160)

regbutton = Button(frame,text="New User ? Click here to register", bd=0, cursor='hand2', bg='gold',
activebackground='gold',
            activeforeground='gold', command=register_window)
regbutton.place(x=300, y=230)

forgotbutton = Button(frame,text="Forgot Password", bd=0, cursor='hand2', bg='gold',
activebackground='gold',
            activeforeground='gold', command=forgot_password)
forgotbutton.place(x=125, y=230)

loginbutton2 = Button(frame, text='Login', font=('arial', 15, 'bold'), fg='white', bg='grey', cursor='hand2',
            activebackground='gray20', activeforeground='white', command=signin)
loginbutton2.place(x=125, y=280)
```

```
backbutton=Button(frame,text=' Back ',font=('arial', 15, 'bold'),fg='white',bg='grey',cursor='hand2',
            activebackground='gray20', activeforeground='white', command=back)
backbutton.place(x=315,y=280)

root.mainloop()

#*************************************************************************
#'*                  FLOWER BOX                                           *
#'*************************************************************************
#'*  FILENAME:    THREE_LEVELS.IPYNB                                      *
#'*  AUTHOR:       ADISHREE GUPTA                                         *
#'*  COMPUTER NUMBER:  IN-00003291                                        *
#'*  DATE:          09/10/2022                                            *
#'*  PURPOSE:     THE BELOW PROGRAM WILL BE USED TO MANAGE                *
#'*               USER PROFILES ROLES AS ADMIN,TEACHER & STUDENT          *
#'*************************************************************************

from tkinter import *
from tkinter import ttk
from tkinter import *
from tkinter import ttk
from tkinter.messagebox import *
import mysql.connector as sq

root = Tk()
root.configure(background="navy blue")
root.title('THREE LEVELS')
label1 = Label(root, text="Holistic Student Development System", font=('Bookman Old Style',
30,"bold"),bg="wheat2",fg="navy blue")
label1.grid(row=0, column=0, columnspan=8, rowspan=2, padx=150, pady=55)

def admin():

   import Admin_login

adminbutton1 = Button(root, text='ADMIN', font=('Bookman Old Style', 25, 'bold'), fg='white', bg='grey',
            activebackground='orange', activeforeground='white',command=admin)
adminbutton1.place(x=200, y=200)

def students():

   import Student_login

studentbutton2 = Button(root, text='STUDENTS', font=('Bookman Old Style', 25, 'bold'), fg='white', bg='grey',
            activebackground='white', activeforeground='red',command=students)
studentbutton2.place(x=380, y=200)

def teachers():

   import Teacher_login

teacherbutton3 = Button(root, text='TEACHERS', font=('Bookman Old Style', 25, 'bold'), fg='white', bg='grey',
            activebackground='green', activeforeground='white',command=teachers)
teacherbutton3.place(x=630, y=200)
```

```
def back():

    import User_login

backbutton=Button(root,text=' BACK ',font=('arial', 15, 'bold'),fg='white',bg='grey',cursor='hand2',
                activebackground='gray20', activeforeground='white', command=back)
backbutton.place(x=460,y=450)

root.mainloop()

#*************************************************************************
#'*                    FLOWER BOX                                         *
#'*************************************************************************
#'*   FILENAME:     Admin_register.IPYNB                                 *
#'*   AUTHOR:       ADISHREE GUPTA                                       *
#'*   COMPUTER NUMBER:  IN-00003291                                      *
#'*   DATE:             09/10/2022                                       *
#'*   PURPOSE:      THE BELOW APPLICATION WILL BE USED TO MANAGE         *
#'*                 ADMIN REGISTRATION.SIMILAR PY FILES FOR STUDENT AND  *
#'*                 TEACHER ARE ALSO CREATED FOR DUAL AUTHENTICATION     *
#'*************************************************************************
from tkinter import *
from tkinter import ttk
from tkinter.messagebox import *
import mysql.connector as sq

def login_window():
    import Admin_login

def clear():

    entrycontact.delete(0, END)
    entrypassword.delete(0, END)
    entryconfirmpassword.delete(0, END)
    entryusername.delete(0, END)
    entryfirstname.delete(0, END)
    entrylastname.delete(0, END)
    entryanswer.delete(0, END)
    check.set(0)

def register():
    if entryfirstname.get() == '' or entrylastname.get() == '' or entrycontact.get() == '' or entryusername.get() == ''
or \
          entrypassword.get() == '' or entryconfirmpassword.get() == '' or entryanswer.get() == '':
        showerror('Error', "All Fields Are Required", parent=root)
    elif len(entrycontact.get()) != 10:
        showerror('Error', "Phone number must be of 10 digits", parent=root)

    elif len(entrypassword.get()) != 8:
        showerror('Error', "Password must be alphanumeric and must have 8 characters", parent=root)

    elif entrypassword.get() != entryconfirmpassword.get():
        showerror('Error', "Password Mismatch", parent=root)

    else:
        try:
```

```
        con = sq.connect(host='localhost',user='root',password='HPSdb2018',
database='credit_management_system')
        cur = con.cursor()
        code='insert into admin(f_name,l_name,contact,username,date_of_birth,password)
values(%s,%s,%s,%s,%s,%s)'
        values=(entryfirstname.get(),
entrylastname.get(),entrycontact.get(),entryusername.get(),entryanswer.get(), entrypassword.get(),)
        cur.execute(code,values)
        con.commit()
        con.close()
        showinfo('Success', "Registration Successful", parent=root)
        clear()

    except Exception as e:
        showerror('Error', f"Error due to: {e}", parent=root)


root = Tk()
root.configure(bg="Navy Blue")
root.title('ADMIN REGISTRATION')

titleLabel = Label(root, text='Registration Form', font=('Arial', 40, 'bold '),bg='honeydew2',
              fg='navy blue', )
titleLabel.place(x=250, y=50)

firstnameLabel = Label(root, text='First Name', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
                fg='black', )
firstnameLabel.place(x=225, y=140)
entryfirstname = Entry(root, font=('Bookman Old Style', 16), bg='white')
entryfirstname.place(x=200, y=175, width=250, height=24)

lastnameLabel = Label(root, text='Last Name', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
                fg='black', )
lastnameLabel.place(x=600, y=140)
entrylastname = Entry(root, font=('Bookman Old Style', 16), bg='white')
entrylastname.place(x=550, y=175, width=250,height=24)

contactLabel = Label(root, text='Contact Number', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
              fg='black', )
contactLabel.place(x=225, y=230)
entrycontact = Entry(root, font=('Bookman Old Style', 16), bg='white')
entrycontact.place(x=200, y=265, width=250,height=24)

DOBLabel = Label(root, text='Date of Birth', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
                fg='black', )
DOBLabel.place(x=600, y=230)

entryanswer = Entry(root, font=('Bookman Old Style', 16), bg='white')
entryanswer.place(x=550, y=265, width=250,height=24)

UsernameLabel = Label(root, text='Username', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
              fg='black', )

UsernameLabel.place(x=225, y=330)
entryusername = Entry(root, font=('Bookman Old Style', 16), bg='white')
entryusername.place(x=550, y=330, width=250,height=24)
```

```
passwordLabel = Label(root, text='Password', font=('Bookman Old Style', 16, 'bold'), bg='wheat2',
                  fg='black', )
passwordLabel.place(x=225, y=390)
entrypassword = Entry(root,show="*", font=('Bookman Old Style', 16), bg='white')
entrypassword.place(x=200, y=430, width=250,height=24)

confirmpasswordLabel = Label(root, text='Confirm Password', font=('Bookman Old Style', 16, 'bold'),
                       bg='wheat2',
                       fg='black', )
confirmpasswordLabel.place(x=575, y=390)
entryconfirmpassword = Entry(root,show="*", font=('Bookman Old Style', 16), bg='white')
entryconfirmpassword.place(x=550, y=430, width=250,height=24)

registerbutton = Button(root,text="Register",font=('Bookman Old Style', 18, 'bold'), bd=0, cursor='hand2',
fg='white', bg='grey', activebackground='white'
                     , activeforeground='white', command=register)
registerbutton.place(x=600, y=480)

loginbutton1 = Button(root,text="Login", font=('Bookman Old Style', 18, 'bold',), bd=0, cursor='hand2',
fg='white', bg='grey', activebackground='gold',
                  activeforeground='gold', command=login_window)
loginbutton1.place(x=270, y=480)

root.mainloop()


#*************************************************************************
#'*                  FLOWER BOX                                           *
#'*************************************************************************
#'*  FILENAME:    Admin_login.IPYNB                                       *
#'*  AUTHOR:      ADISHREE GUPTA                                          *
#'*  COMPUTER NUMBER:  IN-00003291                                        *
#'*  DATE:            09/10/2022                                          *
#'*  PURPOSE:     THE BELOW PROGRAM WILL BE USED TO MANAGE            *
#'*                 ADMIN LOGIN.SIMILAR PY FILES FOR STUDENT AND         *
#'*                 TEACHER ARE ALSO CREATED                              *
#'*************************************************************************


from tkinter import *
from tkinter import ttk
from tkinter import *
from tkinter import ttk
from tkinter.messagebox import *
import mysql.connector as sq

def register_window():
   import Admin_register
def interface():
   import Data_manager
def back():
   import three_levels
def signin():
   if userentry.get() == '' or passentry.get() == '':
      showerror('Error', 'All Fields Are Required')
```

```
    else:
        try:
            con = sq.connect(host='localhost',user='root',password='HPSdb2018',
database='credit_management_system')
            cur = con.cursor()
            cur.execute('select * from admin where username=%s and password=%s', (userentry.get(),
passentry.get()))
            row = cur.fetchone()
            if row == None:
                showerror('error', 'Invalid username or Password')
                root.destroy()
            else:
                showinfo('Authenticated', 'Welcome')
                interface()
            con.close()
        except Exception as e:
            showerror('Error', f"Error due to: {e}", parent=root)

def forgot_password():

    def authentication():

        if user_entry.get()==" " or DOB_entry.get()==" " or newpass_entry==" ":
            showerror('Error', 'All Fields Are Required',parent=window)
        else:
            con = sq.connect(host='localhost',user='root',password='HPSdb2018',
database='credit_management_system')
            cur = con.cursor()
            query='select * from admin where username=%s and date_of_birth=%s'
            cur.execute(query,(user_entry.get(),DOB_entry.get()))
            row=cur.fetchone()
            if row==None:
                message.showerror('Error','Incorrect username and date of birth',parent=window)
            else:
                query='update admin set password=%s where username=%s and date_of_birth=%s'
                cur.execute(query,(newpass_entry.get(),user_entry.get(),DOB_entry.get()))
                con.commit()
                con.close()
                showinfo('Success','Password is reset,please login with new password',parent=window)
                window.destroy()

    window=Toplevel()
    window.title("Forgot password")

    window.configure(bg="navy blue")

    heading_label = Label(window,text="Password Manager",font=('Arial','50' , 'bold'),bg='navy
blue',fg='white')
    heading_label.place(x=200,y=60)

    userlabel=Label(window,text=" Username   ",font=('Arial','20' , 'bold'),bg='navy blue',fg='white')
    userlabel.place(x=200,y=200)

    user_entry=Entry(window,width=25,fg='black',font=('Arial','20','italic'),bd=0)
    user_entry.place(x=520,y=200)
```

```
    DOBlabel=Label(window,text="Date of Birth",font=('Arial','20' , 'bold'),bg='navy blue',fg='white')
    DOBlabel.place(x=200,y=250)

    DOB_entry=Entry(window,width=25,fg='black',font=('Arial','20','italic'),bd=0)
    DOB_entry.place(x=520,y=250)

    newpasslabel=Label(window,text="New Password",font=('Arial','20' , 'bold'),bg='navy blue',fg='white')
    newpasslabel.place(x=200,y=300)

    newpass_entry=Entry(window,width=25,fg='black',font=('Arial','20','italic'),bd=0)
    newpass_entry.place(x=520,y=300)

    resetbutton = Button(window,text=' RESET ',font=('arial', 15, 'bold'),fg='white',bg='grey',cursor='hand2',
                activebackground='gray20', activeforeground='white', command=authentication)
    resetbutton.place(x=800, y=380)

    window.mainloop()

root = Tk()
root.configure(background="navy blue")
root.title('ADMIN LOGIN')

frame = Frame(root, bg='white', width=560, height=320)
frame.place(x=50, y=140)

label = Label(root, text="ADMIN LOGIN", font=('Arial', 20),bg="grey",fg='white')
label.grid(row=0, column=0, columnspan=8, rowspan=2, padx=50, pady=105)

mailLabel = Label(frame, text='Username', font=('arial', 22, 'bold'), bg='white', fg='black')
mailLabel.place(x=200, y=32)
userentry = Entry(frame, font=('arial', 22,), bg='white', fg='black')
userentry.place(x=110, y=70)

passLabel = Label(frame, text='Password', font=('arial', 22, 'bold'), bg='white', fg='black')
passLabel.place(x=200, y=120)
passentry = Entry(frame,show="*",font=('arial', 22,), bg='white', fg='black')
passentry.place(x=110, y=160)

regbutton = Button(frame,text="Register As Admin ?", bd=0, cursor='hand2', bg='gold',
activebackground='gold',
                activeforeground='gold', command=register_window)
regbutton.place(x=300, y=230)

forgotbutton = Button(frame,text="Forgot Password", bd=0, cursor='hand2', bg='gold',
activebackground='gold',
                activeforeground='gold', command=forgot_password)
forgotbutton.place(x=125, y=230)

loginbutton2 = Button(frame, text='Login', font=('arial', 15, 'bold'), fg='white', bg='grey', cursor='hand2',
                activebackground='gray20', activeforeground='white', command=signin)
loginbutton2.place(x=125, y=280)

backbutton=Button(frame,text=' Back ',font=('arial', 15, 'bold'),fg='white',bg='grey',cursor='hand2',
            activebackground='gray20', activeforeground='white', command=back)
backbutton.place(x=315,y=280)
```

```
root.mainloop()


#*************************************************************************
#'*                    FLOWER BOX                                         *
#'*************************************************************************
#'*   FILENAME:     Data_manager.IPYNB                                   *
#'*   AUTHOR:       ADISHREE GUPTA                                       *
#'*   COMPUTER NUMBER:  IN-00003291                                      *
#'*   DATE:            09/10/2022                                        *
#'*   PURPOSE:      THE BELOW PROGRAM WILL BE USED BY ADMIN TO        *
#'*                 MANAGE STUDENT PROFILES.ADD/ SELECT AND              *
#'*                 UPDATE/ DELETE/RESET ALL DATA AND LOGOUT FROM        *
#'*                 THE SCREEN                                           *
#'************************************************************************


import tkinter as tk
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from tkinter.messagebox import *
import mysql.connector as sq


def connection():
    conn = sq.connect(
        host='localhost',
        user='root',
        password='HPSdb2018',
        db='world',
    )
    return conn

def refreshTable():
    for data in my_tree.get_children():
        my_tree.delete(data)

    for array in read():
        my_tree.insert(parent='', index='end', iid=array, text="", values=(array), tag="orow")

    my_tree.tag_configure('orow', background='white', font=('Times New Roman', 12))
    my_tree.grid(row=8, column=0, columnspan=5, rowspan=11, padx=10, pady=20)

root = Tk()
root.geometry('900x600+50+50')
root.configure(bg="black")
root.title("ADMINSTRATOR")
my_tree = ttk.Treeview(root)

ph1 = tk.StringVar()
ph2 = tk.StringVar()
ph3 = tk.StringVar()
ph4 = tk.StringVar()
```

```
ph5 = tk.StringVar()
ph6 = tk.StringVar()


def setph(word,num):
    if num ==1:
        ph1.set(word)
    if num ==2:
        ph2.set(word)
    if num ==3:
        ph3.set(word)
    if num ==4:
        ph4.set(word)
    if num ==5:
        ph5.set(word)
    if num ==6:
        ph6.set(word)

def read():
    conn = connection()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM profile")
    results = cursor.fetchall()
    conn.commit()
    conn.close()
    return results

def add():
    profileid = str(profileidEntry.get())
    name = str(nameEntry.get())
    email = str(emailEntry.get())
    address = str(addressEntry.get())
    phone = str(phoneEntry.get())
    Class = str(classEntry.get())
    decision = messagebox.askquestion("Add", "Add entered data?")
    if (profileid == "" or profileid == " ") or (name == "" or name == " ") or (email == "" or email == " ") or
(address == "" or address == " ") or (phone == "" or phone == " ") or (Class=="" or Class==" "):
        messagebox.showinfo("Error", "Please fill up the blank entry")
        return
    elif len(phone)!=10:
        messagebox.showinfo("Error", "Mobile Number must be of 10 digits")
    else:
        try:
            conn = connection()
            cursor = conn.cursor()
            cursor.execute("INSERT INTO profile VALUES
('"+profileid+"','"+name+"','"+email+"','"+address+"','"+phone+"','"+Class+"') ")
            conn.commit()
            conn.close()
        except:
            messagebox.showinfo("Error", "Profile ID already exist")
            return

    refreshTable()
```

```
def reset():
    decision = messagebox.askquestion("Warning!!", "Delete all data?")
    if decision != "yes":
        return
    else:
        try:
            conn = connection()
            cursor = conn.cursor()
            cursor.execute("DELETE FROM profile")
            conn.commit()
            conn.close()
        except:
            messagebox.showinfo("Error", "Sorry an error occured")
            return

        refreshTable()

def delete():
    decision = messagebox.askquestion("Warning!!", "Delete the selected data?")
    if decision != "yes":
        return
    else:
        selected_item = my_tree.selection()[0]
        deleteData = str(my_tree.item(selected_item)['values'][0])
        try:
            conn = connection()
            cursor = conn.cursor()
            cursor.execute("DELETE FROM profile WHERE Profile_ID='"+str(deleteData)+"'")
            conn.commit()
            conn.close()
        except:
            messagebox.showinfo("Error", "Sorry an error occured")
            return

        refreshTable()

def select():
    try:
        selected_item = my_tree.selection()[0]
        profileid = str(my_tree.item(selected_item)['values'][0])
        name = str(my_tree.item(selected_item)['values'][1])
        email = str(my_tree.item(selected_item)['values'][2])
        address = str(my_tree.item(selected_item)['values'][3])
        phone = str(my_tree.item(selected_item)['values'][4])
        Class = str(my_tree.item(selected_item)['values'][5])

        setph(profileid,1)
        setph(name,2)
        setph(email,3)
        setph(address,4)
        setph(phone,5)
        setph(Class,6)

    except:
        messagebox.showinfo("Error", "Please select a data row")
```

```
def search():
    profileid = str(profileidEntry.get())
    name = str(nameEntry.get())
    email = str(emailEntry.get())
    address = str(addressEntry.get())
    phone = str(phoneEntry.get())
    Class = str(classEntry.get())

    conn = connection()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM profile WHERE Profile_ID='"+
    profileid+"' or NAME='"+
    name+"' or Email_Id='"+
    email+"' or ADDRESS='"+
    address+"' or PHONE='"+
    phone+"' or CLASS='"+
    Class+"' ")

    try:
        result = cursor.fetchall()

        for num in range(0,5):
            setph(result[0][num],(num+1))

        conn.commit()
        conn.close()
    except:
        messagebox.showinfo("Error", "No data found")

def update():
    decision = messagebox.askquestion("UPDATE", "Update selected data?")
    selectedprofileid = ""

    try:
        selected_item = my_tree.selection()[0]
        selectedprofileid = str(my_tree.item(selected_item)['values'][0])
    except:
        messagebox.showinfo("Error", "Please select a data row")

    profileid = str(profileidEntry.get())
    name = str(nameEntry.get())
    email = str(emailEntry.get())
    address = str(addressEntry.get())
    phone = str(phoneEntry.get())
    Class = str(classEntry.get())

    if (profileid == "" or profileid == " ") or (name == "" or name == " ") or (email == "" or email == " ") or
(address == "" or address == " ") or (phone == "" or phone == " ") or (Class=="" or Class==" "):
        messagebox.showinfo("Error", "Please fill up the blank entry")
        return
    else:
        try:
            conn = connection()
            cursor = conn.cursor()
            cursor.execute("UPDATE profile SET Profile_ID='"+
            profileid+"', NAME='"+
```

```
            name+"', Email_Id='"+
            email+"', ADDRESS='"+
            address+"', PHONE='"+
            phone+"', CLASS='"+
            Class+"' WHERE Profile_ID='"+
            selectedprofileid+"' ")
            conn.commit()
            conn.close()
        except:
            messagebox.showinfo("Error", "Profile ID already exist")
            return

    refreshTable()

def logout():
    root.destroy()
    import Admin_login

label = Label(root, text="STUDENT MANAGEMENT SYSTEM", font=('Algerian', 25),bg="white")
label.grid(row=0, column=0, columnspan=8, rowspan=2, padx=10, pady=10)

profileidLabel = Label(root, text="PROFILE ID", font=('Arial', 15),bg="white")
nameLabel = Label(root, text="NAME", font=('Arial', 15),bg="white")
emailLabel = Label(root, text="EMAIL", font=('Arial', 15),bg="white")
addressLabel = Label(root, text="ADDRESS", font=('Arial', 15),bg="white")
phoneLabel = Label(root, text="PHONE", font=('Arial', 15),bg="white")
classLabel = Label(root, text="CLASS", font=('Arial', 15),bg="white")

profileidLabel.grid(row=2, column=0, columnspan=1, padx=12, pady=2)
nameLabel.grid(row=3, column=0, columnspan=1, padx=12, pady=2)
emailLabel.grid(row=4, column=0, columnspan=1, padx=12, pady=2)
addressLabel.grid(row=5, column=0, columnspan=1, padx=12, pady=2)
phoneLabel.grid(row=6, column=0, columnspan=1, padx=12, pady=2)
classLabel.grid(row=7, column=0, columnspan=1, padx=12, pady=2)

profileidEntry = Entry(root, width=55, bd=3, font=('Arial', 15), textvariable = ph1)
nameEntry = Entry(root, width=55, bd=3, font=('Arial', 15), textvariable = ph2)
emailEntry = Entry(root, width=55, bd=3, font=('Arial', 15), textvariable = ph3)
addressEntry = Entry(root, width=55, bd=3, font=('Arial', 15), textvariable = ph4)
phoneEntry = Entry(root, width=55, bd=3, font=('Arial', 15), textvariable = ph5)
classEntry = Entry(root, width=55, bd=3, font=('Arial', 15), textvariable = ph6)

profileidEntry.grid(row=2, column=1, columnspan=1, padx=1, pady=0)
nameEntry.grid(row=3, column=1, columnspan=1, padx=1, pady=0)
emailEntry.grid(row=4, column=1, columnspan=1, padx=2, pady=0)
addressEntry.grid(row=5, column=1, columnspan=1, padx=3, pady=0)
phoneEntry.grid(row=6, column=1, columnspan=1, padx=3, pady=0)
classEntry.grid(row=7, column=1, columnspan=1, padx=3, pady=0)

addBtn = Button(
    root, text="Add", padx=25, pady=10, width=5,
    bd=5, font=('lucida', 13), bg="orange", command=add)
updateBtn = Button(
    root, text="Update", padx=25, pady=10, width=5,
    bd=5, font=('lucida', 13), bg="blue", command=update)
deleteBtn = Button(
```

```
    root, text="Delete", padx=25, pady=10, width=5,
    bd=5, font=('lucida', 13), bg="green", command=delete)
searchBtn = Button(
    root, text="Search", padx=25, pady=10, width=5,
    bd=5, font=('lucida', 13), bg="yellow", command=search)
resetBtn = Button(
    root, text="Reset All", padx=25, pady=10, width=5,
    bd=5, font=('lucida', 13), bg="orange", command=reset)
selectBtn = Button(
    root, text="Select", padx=25, pady=10, width=5,
    bd=5, font=('lucida', 13), bg="white", command=select)
logoutBtn = Button(
    root, text="Log Out", padx=25, pady=10, width=5,
    bd=5, font=('lucida', 13), bg="deep pink", command=logout)

addBtn.grid(row=0, column=5, columnspan=1, rowspan=3)
updateBtn.grid(row=3, column=5, columnspan=1, rowspan=1)
deleteBtn.grid(row=4, column=5, columnspan=1, rowspan=1)
searchBtn.grid(row=5, column=5, columnspan=1, rowspan=1)
resetBtn.grid(row=6, column=5, columnspan=1, rowspan=1)
selectBtn.grid(row=7, column=5, columnspan=1, rowspan=2)
logoutBtn.grid(row=9, column=5, columnspan=1, rowspan=1)

style = ttk.Style()
style.configure("Treeview.Heading", font=('Arial Bold', 15))

scrollbary=Scrollbar(root,orient=VERTICAL)
scrollbary.configure(command=my_tree.yview)
my_tree.configure(yscrollcommand=scrollbary.set)
scrollbary.place(relx=0.8524,rely=0.68,width=22,height=176)
my_tree.configure(selectmode="extended")


my_tree['columns'] = ("PROFILE ID","NAME","EMAIL ID","ADDRESS","PHONE","CLASS")
my_tree.column("#0", width=0, stretch=NO)
my_tree.column("PROFILE ID", anchor=CENTER, width=170)
my_tree.column("NAME", anchor=CENTER, width=150)
my_tree.column("EMAIL ID", anchor=CENTER, width=150)
my_tree.column("ADDRESS", anchor=CENTER, width=165)
my_tree.column("PHONE", anchor=CENTER, width=150)
my_tree.column("CLASS", anchor=CENTER, width=160)

my_tree.heading("PROFILE ID", text="PROFILE ID", anchor=CENTER)
my_tree.heading("NAME", text="NAME", anchor=CENTER)
my_tree.heading("EMAIL ID", text="EMAIL ID", anchor=CENTER)
my_tree.heading("ADDRESS", text="ADDRESS", anchor=CENTER)
my_tree.heading("PHONE", text="PHONE", anchor=CENTER)
my_tree.heading("CLASS", text="CLASS", anchor=CENTER)

refreshTable()

root.mainloop()


#*************************************************************************
#'*                    FLOWER BOX                              *
```

```
#'*************************************************************************
#'*   FILENAME:     class9_credits.IPYNB                          *
#'*   AUTHOR:       ADISHREE GUPTA                                *
#'*   COMPUTER NUMBER:  IN-00003291                               *
#'*   DATE:            09/11/2022                                 *
#'*   PURPOSE:      THE BELOW APPLICATION WILL BE USED TO MANAGE        *
#'*               CREDIT QUESTIONAIRE AND ENTRY.SIMILAR PY FILES FOR    *
#'*               OTHER CLASSES ARE ALSO CREATED                    *
#'*************************************************************************



from tkinter import*
from tkinter import ttk
from tkinter import messagebox
from tkinter.messagebox import *
import mysql.connector as sq


def connection():
    conn = sq.connect(
        host='localhost',
        user='root',
        password='HPSdb2018',
        db='world',
    )
    return conn

def back():
    import STUDENT_DETAILS

root=Tk()
root.title("CREDIT BOOSTER")
root.geometry('900x600+50+50')
root.configure(bg="navy blue")
frame1=Frame(root,bg="white")
frame1.place(x=70,y=30,width=960,height=500)
title=Label(frame1,text="Questionnaire" , font=("time new roman",20,"bold"),bg= "white" ,fg
="green").place(x=360,y=10)
name=Label(frame1,text="Enter your registered name ",font=("time new roman",10,"bold"),bg= "white" ,fg
="gray").place(x=30,y=70)
entryname = Entry(frame1, font=('Bookman Old Style', 16), bg='white')
entryname.place(x=700, y=70, width=230, height=24)
que1=Label(frame1,text="Did you secure NTSE scholarship last year?" , font=("time new
roman",10,"bold"),bg= "white" ,fg ="gray").place(x=30,y=100)
que1 = ttk.Combobox(frame1,font=("time new roman",10),state = 'readonly',justify=CENTER)
que1['values'] = ("Select","yes","no")
que1.place(x=700,y=100,width=230)
que1.current(0)
que2=Label(frame1,text="Were you a state level awardee in the international science/maths/computer
olympiads last year?" , font=("time new roman",10,"bold"),bg= "white" ,fg ="gray").place(x=30,y=130)
que2 = ttk.Combobox(frame1,font=("time new roman",10),state = 'readonly',justify=CENTER)
que2['values'] = ("Select","yes","no")
que2.place(x=700,y=130,width=230)
que2.current(0)
```

```
que3=Label(frame1,text="Were you a national awardee in any of the CBSE competitions last year?" ,
font=("time new roman",10,"bold"),bg= "white" ,fg ="gray").place(x=30,y=160)
que3 = ttk.Combobox(frame1,font=("time new roman",10),state = 'readonly',justify=CENTER)
que3['values'] = ("Select","yes","no")
que3.place(x=700,y=160,width=230)
que3.current(0)
que4=Label(frame1,text="Were you national awardee in any of the IIT/Government hackathons organized  last
year?" , font=("time new roman",10,"bold"),bg= "white" ,fg ="gray").place(x=30,y=190)
que4 = ttk.Combobox(frame1,font=("time new roman",10),state = 'readonly',justify=CENTER)
que4['values'] = ("Select","yes","no")
que4.place(x=700,y=190,width=230)
que4.current(0)
que5=Label(frame1,text="Have you been an awardee for any of the Azadi ka Amrit Mahotsav competitions ?" ,
font=("time new roman",10,"bold"),bg= "white" ,fg ="gray").place(x=30,y=220)
que5 = ttk.Combobox(frame1,font=("time new roman",10),state = 'readonly',justify=CENTER)
que5['values'] = ("Select","yes","no")
que5.place(x=700,y=220,width=230)
que5.current(0)
que6=Label(frame1,text="Have you participated in any interhouse team events in your school last year?" ,
font=("time new roman",10,"bold"),bg= "white" ,fg ="gray").place(x=30,y=250)
que6 = ttk.Combobox(frame1,font=("time new roman",10),state = 'readonly',justify=CENTER)
que6['values'] = ("Select","yes","no")
que6.place(x=700,y=250,width=230)
que6.current(0)
que7=Label(frame1,text="Are you currently a part of any awareness campaigns being  led in your
neighborhood ?" , font=("time new roman",10,"bold"),bg= "white" ,fg ="gray").place(x=30,y=280)
que7 = ttk.Combobox(frame1,font=("time new roman",10),state = 'readonly',justify=CENTER)
que7['values'] = ("Select","yes","no")
que7.place(x=700,y=280,width=230)
que7.current(0)
que8=Label(frame1,text="Have you celebrated  Har Ghar Tiranga  this year in your house?" , font=("time new
roman",10,"bold"),bg= "white" ,fg ="gray").place(x=30,y=310)
que8 = ttk.Combobox(frame1,font=("time new roman",10),state = 'readonly',justify=CENTER)
que8['values'] = ("Select","yes","no")
que8.place(x=700,y=310,width=230)
que8.current(0)
que9=Label(frame1,text="Have you won any inter school sports tournaments this year?" , font=("time new
roman",10,"bold"),bg= "white" ,fg ="gray").place(x=30,y=340)
que9 = ttk.Combobox(frame1,font=("time new roman",10),state = 'readonly',justify=CENTER)
que9['values'] = ("Select","yes","no")
que9.place(x=700,y=340,width=230)
que9.current(0)
que10=Label(frame1,text="Have contributed towards teaching poor children this year?" , font=("time new
roman",10,"bold"),bg= "white" ,fg ="gray").place(x=30,y=370)
que10 = ttk.Combobox(frame1,font=("time new roman",10),state = 'readonly',justify=CENTER)
que10['values'] = ("Select","yes","no")
que10.place(x=700,y=370,width=230)
que10.current(0)

def submit_ans():

    result=0
    if que1.get() =="yes":
        result+=100
    if que2.get() =="yes":
        result+=80
```

```
    if que3.get() =="yes":
      result+=50
    if que4.get() =="yes":
      result+=50
    if que5.get() =="yes":
      result+=80
    if que6.get() =="yes":
      result+=50
    if que7.get() =="yes":
      result+=80
    if que8.get() =="yes":
      result+=50
    if que9.get() =="yes":
      result+=80
    if que10.get() =="yes":
      result+=100
    messagebox.showinfo("Bonus",str(result),parent=root)

    try:
          con = sq.connect(host='localhost',user='root',password='HPSdb2018', database='world')
          cur = con.cursor()
          code='insert into student_9_credits(name,credits) values(%s,%s)'
          values=(entryname.get(),str(result),)
          cur.execute(code,values)
          con.commit()
          con.close()
          showinfo('Success', "Submitted Successfully", parent=root)

    except Exception as e:
          showerror('Error', f"Error due to: {e}", parent=root)

submit=Button(frame1,text ="Submit",font=("time new
roman",15),bd=0,cursor="hand2",command=submit_ans).place(x=30,y=440)

backbutton=Button(frame1,text=' Back ',font=('arial', 15, 'bold'),fg='white',bg='grey',cursor='hand2',
              activebackground='gray20', activeforeground='white', command=back)
backbutton.place(x=850,y=440)

root.mainloop()

*************************************************************************
#'*                    FLOWER BOX                                         *
#'*************************************************************************
#'*   FILENAME:    class_9_data.IPYNB                                    *
#'*   AUTHOR:      ADISHREE GUPTA                                        *
#'*   COMPUTER NUMBER:  IN-00003291                                      *
#'*   DATE:            09/10/2022                                        *
#'*   PURPOSE:     THE BELOW PROGRAM WILL BE USED TO EXPORT CREDIT     *
#'*   DATA LOCALLY FOR FURTHER PROCESSING BY THE USER ON THEIR SYSTEM    *
#'*   SIMILAR PROGRAM FILES ARE CREATED FOR OTHER CLASSES                *
#'*************************************************************************


import mysql.connector as sq
import tkinter as tk
from tkinter import ttk
```

```
from tkinter import *
from tkinter.messagebox import *

def back():
    import Teacher_credentials

my_w = tk.Tk()
my_w.geometry('900x600+50+50')
my_w.configure(bg="navy blue")
my_w.geometry("400x250")

Label1=Label(my_w,text="Your Class Report",font=("Arial",40,"bold"),bg="wheat2",fg="navy blue")
Label1.place(x=280,y=0)

connect=sq.connect(host="localhost",user="root",passwd="HPSdb2018",database="world")
conn=connect.cursor()
conn.execute("select r.name as NAME,r.age as AGE,r.class as CLASS,k.credits as CREDITS from
class_9_details r, student_9_credits k where r.name=k.name group by class order by credits")
results = conn.fetchall()
tree=ttk.Treeview(my_w)
tree['show']='headings'

tree["columns"]=("NAME","AGE","CLASS","CREDITS")

tree.column("NAME",width=100,minwidth=10,anchor=tk.CENTER)
tree.column("AGE",width=100,minwidth=10,anchor=tk.CENTER)
tree.column("CLASS",width=100,minwidth=10,anchor=tk.CENTER)
tree.column("CREDITS",width=100,minwidth=10,anchor=tk.CENTER)

tree.heading("NAME",text="NAME",anchor=tk.CENTER)
tree.heading("AGE",text="AGE",anchor=tk.CENTER)
tree.heading("CLASS",text="CLASS",anchor=tk.CENTER)
tree.heading("CREDITS",text="CREDITS",anchor=tk.CENTER)

i=0
for student in results:
    tree.insert(",i,text="",values=(student[0],student[1],student[2],student[3]))
    i=i+1

scrollbary=ttk.Scrollbar(my_w,orient='vertical')
scrollbary.configure(command=tree.yview)
tree.configure(yscrollcommand=scrollbary.set)
scrollbary.place(relx=0.509,rely=0.145,width=22,height=225)

tree.grid(padx=300,pady=100)

def save():
    import csv
    f=open("F:\\HSDS_ADISHREE_CLASS_12\\student_9.csv","w")
    teachers_redpen=csv.writer(f)
    teachers_redpen.writerow(['NAME','AGE','CLASS','CREDITS'])
    for row in results:
        teachers_redpen.writerow(row)
    showinfo('Success', "Saved Successfully", parent=my_w)
```

```
savebutton=Button(my_w,text="SAVE",font=("arial",35,"bold"),bg="green",fg="white",command=save).place
(x=300,y=350)

backbutton=Button(my_w,text=' BACK ',font=('arial', 35, 'bold'),fg='white',bg='grey',cursor='hand2',
                activebackground='gray20', activeforeground='white', command=back)
backbutton.place(x=520,y=350)

my_w.mainloop()

#*************************************************************************
#'*                    FLOWER BOX                                         *
#'*************************************************************************
#'*   FILENAME:     STUDENT_DETAILS.IPYNB                                 *
#'*   AUTHOR:       ADISHREE GUPTA                                        *
#'*   COMPUTER NUMBER:  IN-00003291                                       *
#'*   DATE:             19/11/2022                                        *
#'*   PURPOSE:      THE BELOW PROGRAM WILL BE USED TO MANAGE              *
#'*              STUDENT CREDENTIALS FOR LOGGING INTO CREDIT MANAGEMENT SYSTEM*
#'*************************************************************************


from tkinter import *
from tkinter import ttk
from tkinter.messagebox import *
import mysql.connector as sq

root = Tk()
root.configure(bg="navy blue")
root.title('STUDENT CREDENTIALS')

def back():

    import Student_login

titleLabel = Label(root, text='STUDENT CREDENTIALS', font=('Bookman Old Style', 40, 'bold '),bg='black',
                fg='white', )
titleLabel.place(x=200, y=50)

nameLabel = Label(root, text='REGISTERED NAME', font=('Bookman Old Style', 16, 'bold'), bg='black',
                  fg='white', )
nameLabel.place(x=200, y=180)
entryname = Entry(root, font=('Bookman Old Style', 16), bg='white')
entryname.place(x=500, y=180, width=250, height=24)

AgeLabel = Label(root, text='AGE', font=('Bookman Old Style', 16, 'bold'), bg='black',
                 fg='white', )
AgeLabel.place(x=200, y=250)
entryage = Entry(root, font=('Bookman Old Style', 16), bg='white')
entryage.place(x=500, y=250, width=250,height=24)

def class9():
    try:
        con = sq.connect(host='localhost',user='root',password='HPSdb2018', database='world')
        cur = con.cursor()
        code='insert into class_9_details(name,age,class) values(%s,%s,%s)'
        values=(entryname.get(),str(entryage.get()),'9')
```

```
        cur.execute(code,values)
        con.commit()
        con.close()

    except Exception as e:
        showerror('Error', f"Error due to: {e}", parent=root)

    import class_9_credits

class9button = Button(root,text="CLASS 9",font=('Arial', 18, 'bold'), bd=0, cursor='hand2', fg='white',
bg='grey', activebackground='white'
                , activeforeground='white', command=class9)
class9button.place(x=200, y=300)

def class10():
    try:
        con = sq.connect(host='localhost',user='root',password='HPSdb2018', database='world')
        cur = con.cursor()
        code='insert into class_10_details(name,age,class) values(%s,%s,%s)'
        values=(entryname.get(),str(entryage.get()),'10')
        cur.execute(code,values)
        con.commit()
        con.close()

    except Exception as e:
        showerror('Error', f"Error due to: {e}", parent=root)

    import class_10_credits

class10button = Button(root,text="CLASS 10",font=('Arial', 18, 'bold'), bd=0, cursor='hand2', fg='white',
bg='grey', activebackground='white'
                , activeforeground='white', command=class10)
class10button.place(x=620, y=300)

def class11():
    try:
        con = sq.connect(host='localhost',user='root',password='HPSdb2018', database='world')
        cur = con.cursor()
        code='insert into class_11_details(name,age,class) values(%s,%s,%s)'
        values=(entryname.get(),str(entryage.get()),'11')
        cur.execute(code,values)
        con.commit()
        con.close()

    except Exception as e:
        showerror('Error', f"Error due to: {e}", parent=root)

    import class_11_credits

class11button = Button(root,text="CLASS 11",font=('Arial', 18, 'bold'), bd=0, cursor='hand2', fg='white',
bg='grey', activebackground='white'
                , activeforeground='white', command=class11)
class11button.place(x=200, y=400)

def class12():
```

```
    try:
        con = sq.connect(host='localhost',user='root',password='HPSdb2018', database='world')
        cur = con.cursor()
        code='insert into class_12_details(name,age,class) values(%s,%s,%s)'
        values=(entryname.get(),str(entryage.get()),'12')
        cur.execute(code,values)
        con.commit()
        con.close()

    except Exception as e:
        showerror('Error', f"Error due to: {e}", parent=root)

    import class_12_credits

class12button = Button(root,text="CLASS 12",font=('Arial', 18, 'bold'), bd=0, cursor='hand2', fg='white',
bg='grey', activebackground='white'
                , activeforeground='white', command=class12)
class12button.place(x=620, y=400)

backbutton=Button(root,text=' BACK ',font=('arial', 15, 'bold'),fg='white',bg='grey',cursor='hand2',
            activebackground='gray20', activeforeground='white', command=back)
backbutton.place(x=460,y=500)
root.mainloop()

#*************************************************************************
#'*                FLOWER BOX                                             *
#'*************************************************************************
#'*  FILENAME:    TEACHER_CREDENTIALS.IPYNB                              *
#'*  AUTHOR:      ADISHREE GUPTA                                         *
#'*  COMPUTER NUMBER:  IN-00003291                                       *
#'*  DATE:            09/10/2022                                         *
#'*  PURPOSE:     THE BELOW PROGRAM WILL BE USED TO MANAGE               *
#'*            TEACHER CREDENTIALS FOR LOGGING INTO CREDIT MANAGEMENT SYSTEM*
#'*************************************************************************

from tkinter import *
from tkinter import ttk
from tkinter.messagebox import *

root = Tk()
root.configure(bg="navy blue")
root.title('CHOICE OF CLASS')

titleLabel = Label(root, text='CHOOSE YOUR CLASS', font=('Arial', 40, 'bold '),bg='black',
            fg='white', )
titleLabel.place(x=200, y=50)

def class9():
    import class_9_data

def class10():
    import class_10_data

def class11():
    import class_11_data
```

```
def class12():
    import class_12_data

def back():
    import Teacher_login

class9button = Button(root,text="CLASS 9",font=('Arial', 30, 'bold'), bd=0, cursor='hand2', fg='white',
bg='grey', activebackground='white'
                    , activeforeground='white', command=class9)
class9button.place(x=200, y=200)

class10button = Button(root,text="CLASS 10",font=('Arial', 30, 'bold'), bd=0, cursor='hand2', fg='white',
bg='grey', activebackground='white'
                    , activeforeground='white', command=class10)
class10button.place(x=570, y=200)

class11button = Button(root,text="CLASS 11",font=('Arial', 30, 'bold'), bd=0, cursor='hand2', fg='white',
bg='grey', activebackground='white'
                    , activeforeground='white', command=class11)
class11button.place(x=200, y=330)

class12button = Button(root,text="CLASS 12",font=('Arial', 30, 'bold'), bd=0, cursor='hand2', fg='white',
bg='grey', activebackground='white'
                    , activeforeground='white', command=class12)
class12button.place(x=570, y=330)

backbutton=Button(root,text=' BACK ',font=('arial', 15, 'bold'),fg='white',bg='grey',cursor='hand2',
            activebackground='gray20', activeforeground='white', command=back)
backbutton.place(x=460,y=500)

root.mainloop()
```

MY SQL SETTINGS

```
MySQL 8.0 Command Line Client
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database credit_management_system ;
Query OK, 1 row affected (0.01 sec)

mysql> Use credit_management_system;
Database changed
mysql> create table user
    -> (f_name varchar(30),
    -> l_name varchar(30),
    -> contact varchar(10),
    -> username varchar(30),
    -> date_of_birth varchar(11),
    -> password varchar(30));
Query OK, 0 rows affected (0.02 sec)

mysql> describe user;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| f_name        | varchar(30) | YES  |     | NULL    |       |
| l_name        | varchar(30) | YES  |     | NULL    |       |
| contact       | varchar(10) | YES  |     | NULL    |       |
| username      | varchar(30) | YES  |     | NULL    |       |
| date_of_birth | varchar(11) | YES  |     | NULL    |       |
| password      | varchar(30) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

MySQL 8.0 Command Line Client

```
mysql> create table admin
    -> (f_name varchar(30),
    -> l_name varchar(30),
    -> contact varchar(10),
    -> username varchar(30),
    -> date_of_birth varchar(11),
    -> password varchar(30));
Query OK, 0 rows affected (0.02 sec)

mysql> describe admin;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| f_name        | varchar(30) | YES  |     | NULL    |       |
| l_name        | varchar(30) | YES  |     | NULL    |       |
| contact       | varchar(10) | YES  |     | NULL    |       |
| username      | varchar(30) | YES  |     | NULL    |       |
| date_of_birth | varchar(11) | YES  |     | NULL    |       |
| password      | varchar(30) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

MySQL 8.0 Command Line Client

```
mysql> create table student
    -> (f_name varchar(30),
    -> l_name varchar(30),
    -> contact varchar(10),
    -> username varchar(30),
    -> date_of_birth varchar(11),
    -> password varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> describe student;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| f_name        | varchar(30) | YES  |     | NULL    |       |
| l_name        | varchar(30) | YES  |     | NULL    |       |
| contact       | varchar(10) | YES  |     | NULL    |       |
| username      | varchar(30) | YES  |     | NULL    |       |
| date_of_birth | varchar(11) | YES  |     | NULL    |       |
| password      | varchar(30) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

MySQL 8.0 Command Line Client

```
mysql> create table teacher
    -> (f_name varchar(30),
    -> l_name varchar(30),
    -> contact varchar(10),
    -> username varchar(30),
    -> date_of_birth varchar(11),
    -> password varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> describe teacher;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| f_name        | varchar(30) | YES  |     | NULL    |       |
| l_name        | varchar(30) | YES  |     | NULL    |       |
| contact       | varchar(10) | YES  |     | NULL    |       |
| username      | varchar(30) | YES  |     | NULL    |       |
| date_of_birth | varchar(11) | YES  |     | NULL    |       |
| password      | varchar(30) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

MySQL 8.0 Command Line Client

```
mysql> use world;
Database changed
mysql> create table profile
    -> (Profile_ID varchar(100),
    -> NAME varchar(30),
    -> Email_Id varchar(30),
    -> ADDRESS varchar(50),
    -> PHONE varchar(11),
    -> CLASS varchar(2));
Query OK, 0 rows affected (0.04 sec)

mysql> describe profile;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| Profile_ID | varchar(100) | YES  |     | NULL    |       |
| NAME       | varchar(30)  | YES  |     | NULL    |       |
| Email_Id   | varchar(30)  | YES  |     | NULL    |       |
| ADDRESS    | varchar(50)  | YES  |     | NULL    |       |
| PHONE      | varchar(11)  | YES  |     | NULL    |       |
| CLASS      | varchar(2)   | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

MySQL 8.0 Command Line Client

```
mysql> create table student_9_credits
    -> (name varchar(30),
    -> credits varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> create table student_10_credits
    -> (name varchar(30),
    -> credits varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> describe student_9_credits;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(30) | YES  |     | NULL    |       |
| credits | varchar(30) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> describe student_10_credits;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(30) | YES  |     | NULL    |       |
| credits | varchar(30) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

MySQL 8.0 Command Line Client

```
mysql> create table student_11_credits
    -> (name varchar(30),
    -> credits varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> create table student_12_credits
    -> (name varchar(30),
    -> credits varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> describe student_11_credits;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(30) | YES  |     | NULL    |       |
| credits | varchar(30) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> describe student_12_credits;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(30) | YES  |     | NULL    |       |
| credits | varchar(30) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

MySQL 8.0 Command Line Client

```
mysql> create table class_9_details
    -> (name varchar(30),
    -> age varchar(30),
    -> class varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> create table class_10_details
    -> (name varchar(30),
    -> age varchar(30),
    -> class varchar(30));
Query OK, 0 rows affected (0.03 sec)
```

MySQL 8.0 Command Line Client

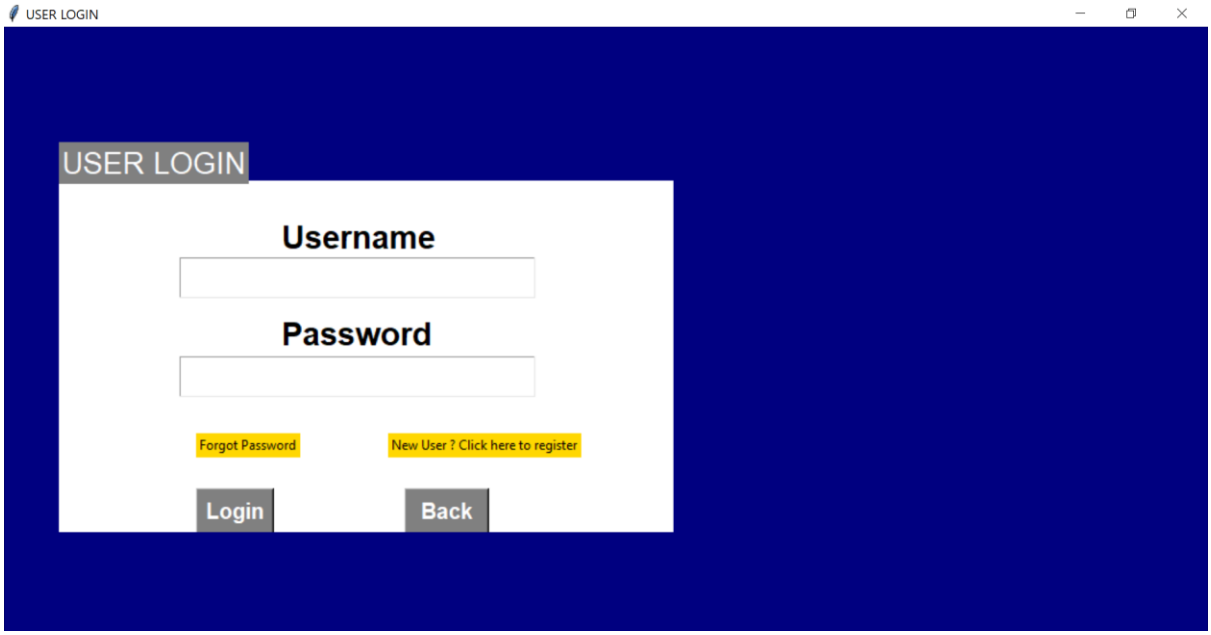
```
mysql> create table class_11_details
    -> (name varchar(30),
    -> age varchar(30),
    -> class varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> create table class_12_details
    -> (name varchar(30),
    -> age varchar(30),
    -> class varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> describe class_9_details;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(30) | YES  |     | NULL    |       |
| age   | varchar(30) | YES  |     | NULL    |       |
| class | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> describe class_10_details;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(30) | YES  |     | NULL    |       |
| age   | varchar(30) | YES  |     | NULL    |       |
| class | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> describe class_11_details;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(30) | YES  |     | NULL    |       |
| age   | varchar(30) | YES  |     | NULL    |       |
| class | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

MySQL 8.0 Command Line Client

```
mysql> describe class_12_details;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(30) | YES  |     | NULL    |       |
| age   | varchar(30) | YES  |     | NULL    |       |
| class | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

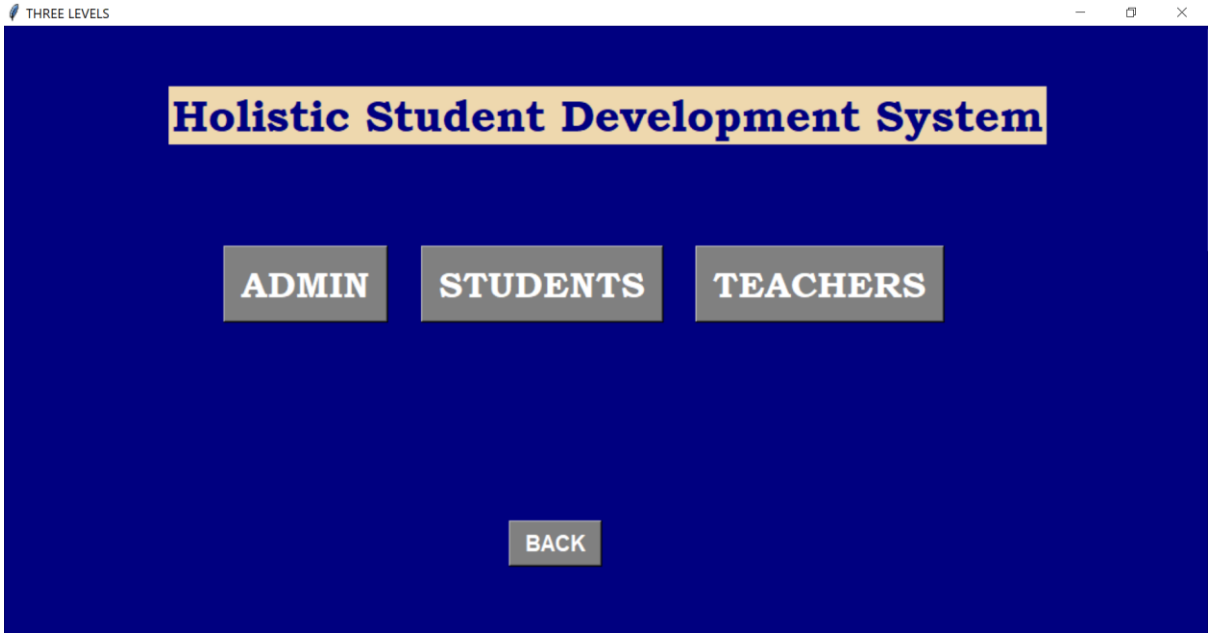
SCREEN FLOWS

USER REGISTRATION



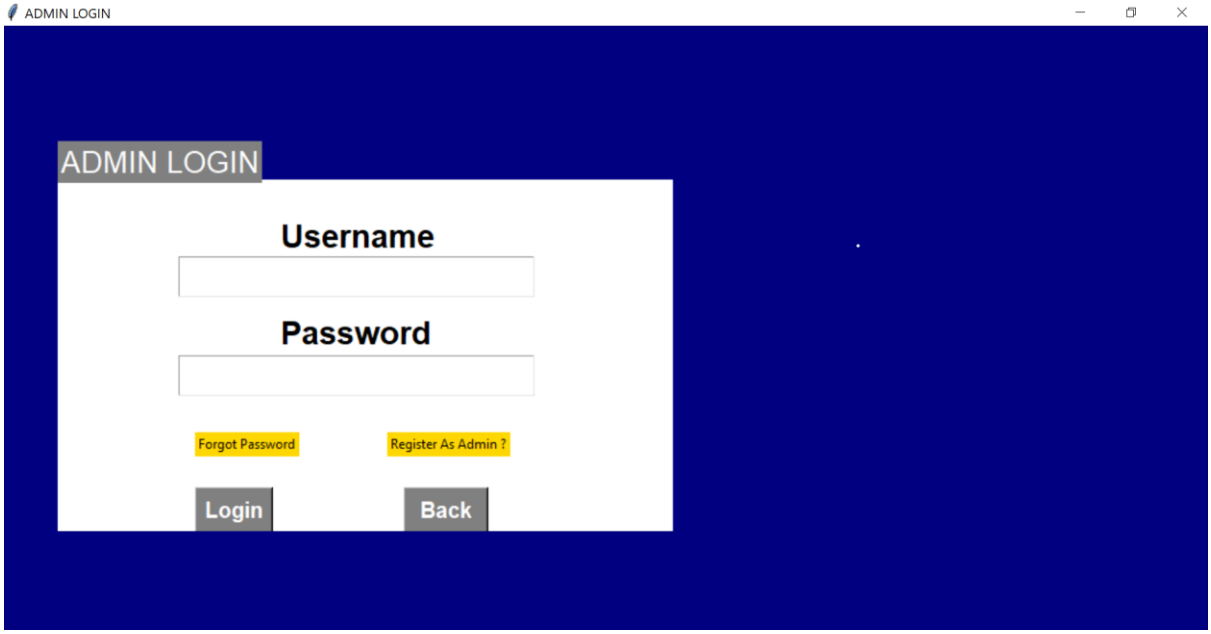
RESET PASSWORD

POST USER LOGIN

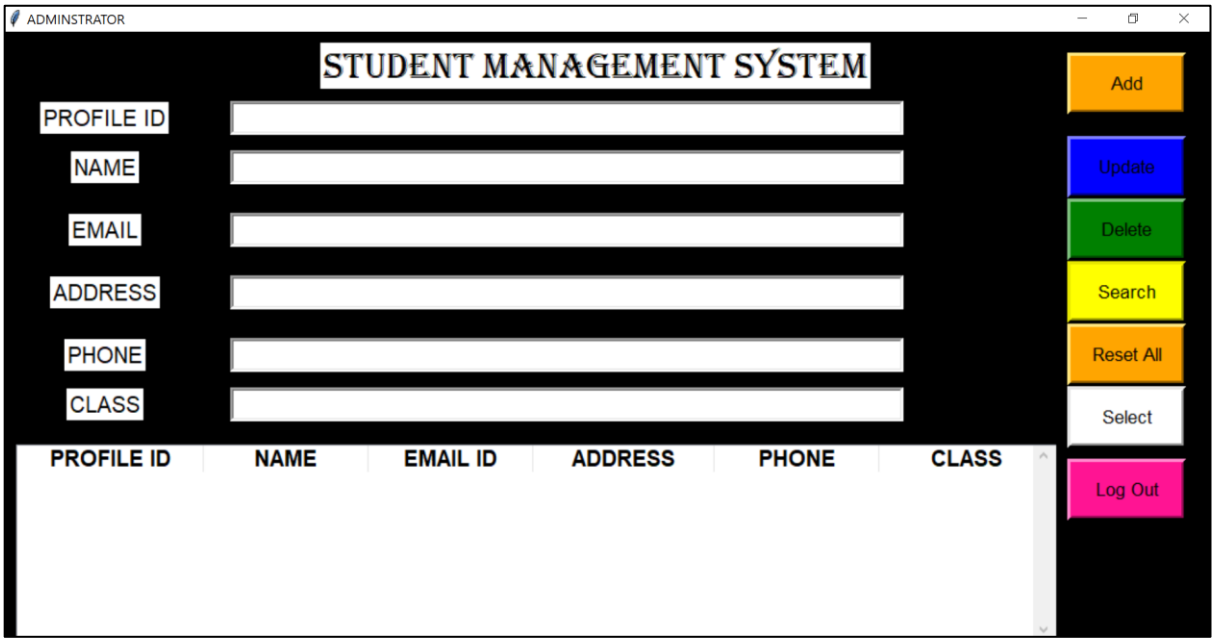


ADMIN REGISTRATION

ADMIN LOGIN

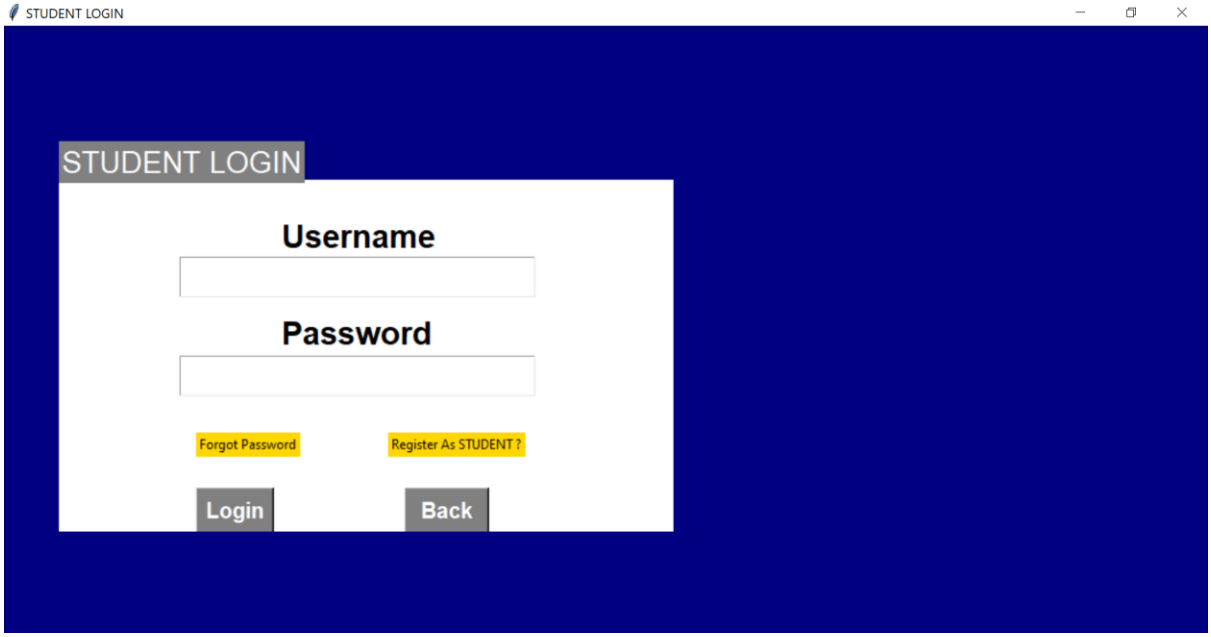


STUDENT PROFILE MANAGEMENT SYSTEM

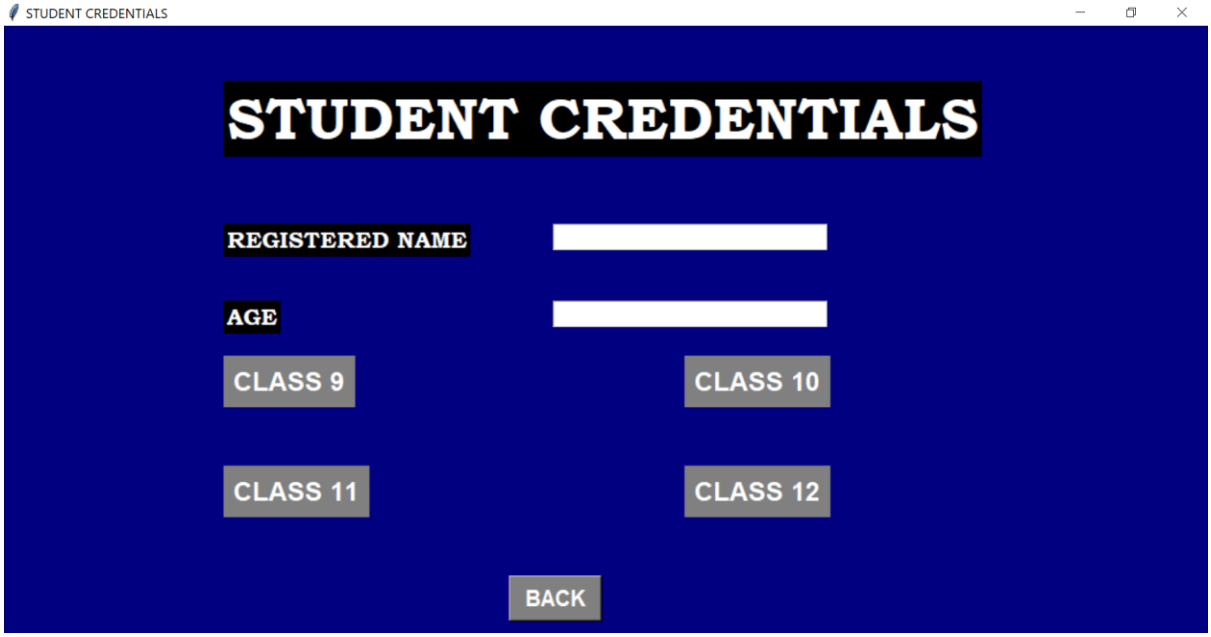


STUDENT REGISTRATION

STUDENT LOGIN

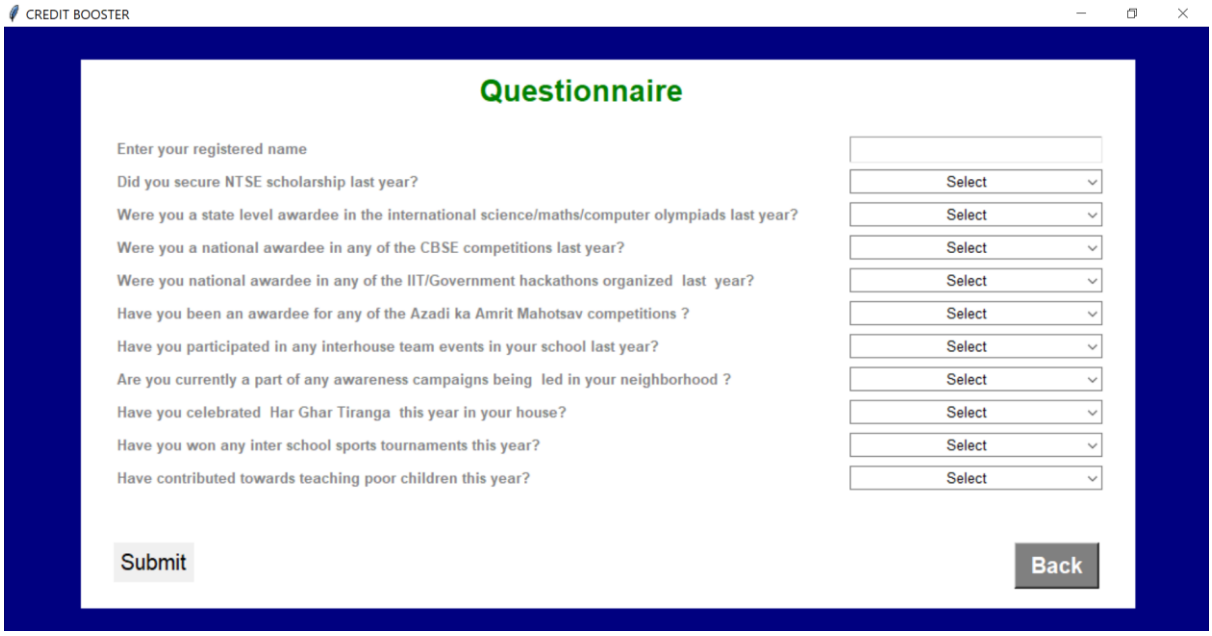


POST STUDENT LOGIN



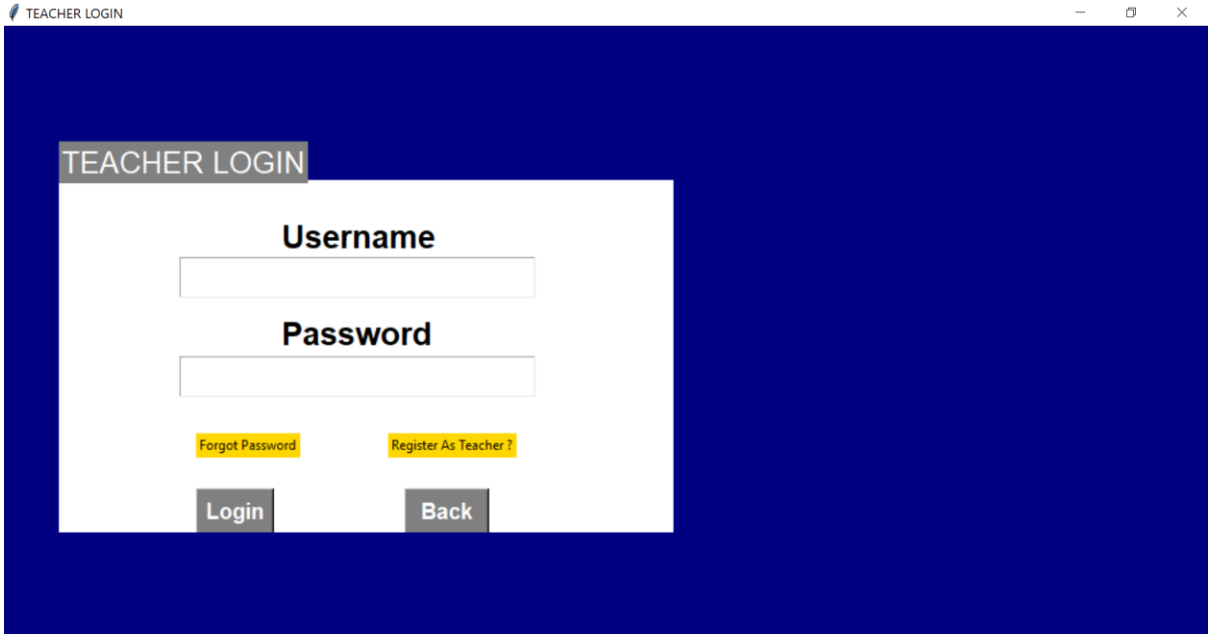
Student enters registered name, age and chooses specific class to fill up questionnaire

In the questionnaire the selections are in Yes and No format. A Yes has a score of 100. On pressing submit the total bonus score is displayed and the same submitted in the database. A student can choose to respond to some questions only



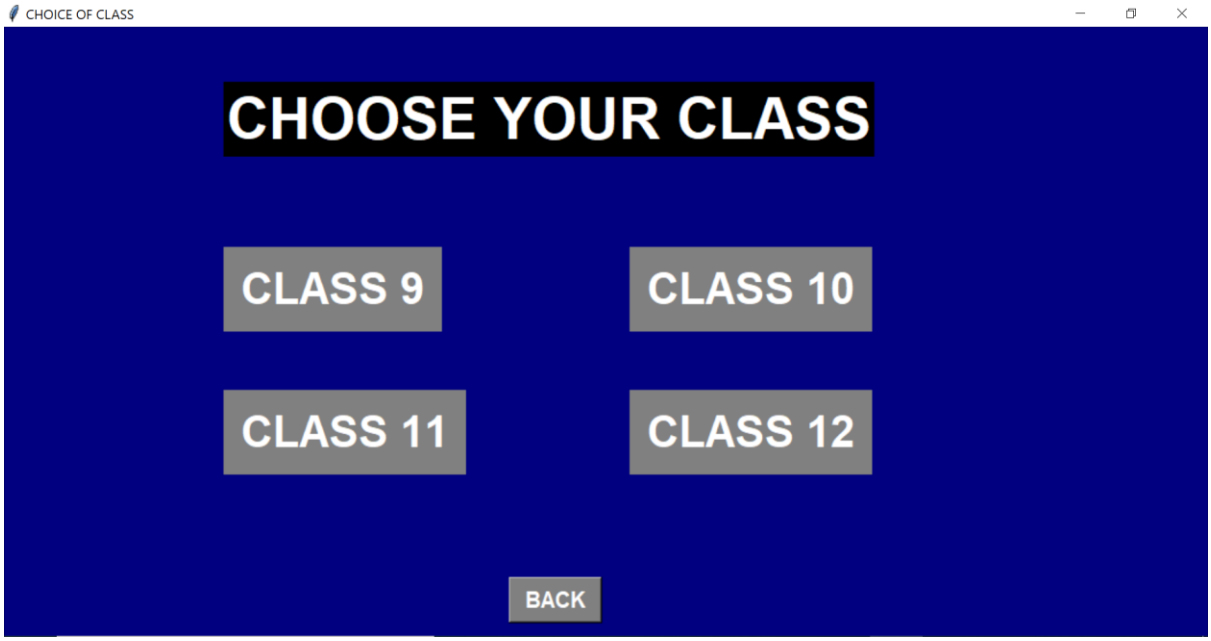
TEACHER REGISTRATION

TEACHER LOGIN



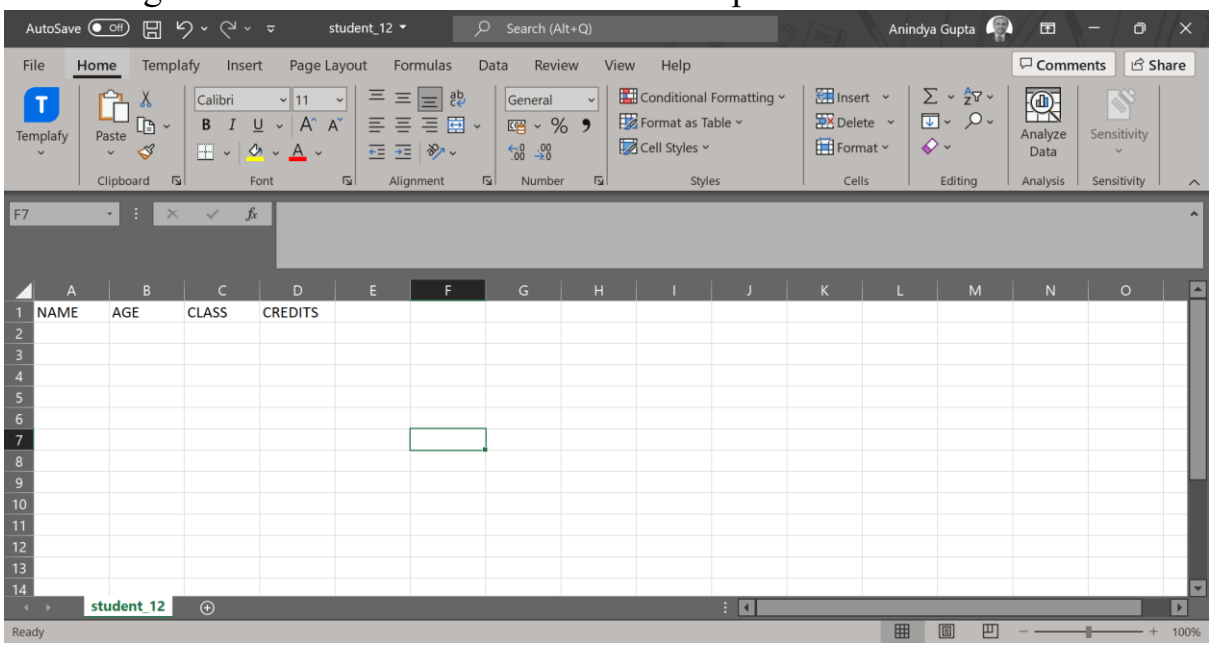
POST TEACHER LOGIN

Post teacher login, the class needs to be chosen to display the report



In the below display the teacher can check for the report and then choose to save the report locally by pressing SAVE.

The data gets saved as a csv file in the local computer in the below format



SCOPE FOR IMPROVEMENT

1. Incorporating comments in the code could improve on the code documentation.
2. Sizing of the Graphical User Interface can be further optimized
3. Many more Validations can be inserted for every screen
4. Login authentication can be further simplified
5. Multiple user profile defined based on separate roles like selective access to screens or fields can be created.
6. The solution can be extended to many more screens like student's performance management, attendance management, fees management, course management etc.
7. Leave and fund management approval workflows can be created in future iterations
8. Report and certificate printing can be activated in future iterations
9. GUI can be made more interactive.
10. A web-based GUI could be built to make this accessible from anywhere.

BIBLIOGRAPHY

1. COMPUTER SCIENCE WITH PYTHON BY SUMITA ARORA
2. HTTPS://DOCS.PYTHON.ORG/3/LIBRARY/TKINTER.HTML
3. HTTPS://DOCS.PYTHON.ORG/3/LIBRARY/TK.HTML
4. HTTPS://STACKOVERFLOW.COM/QUESTIONS/TAGGED/PYTHON
5. HTTPS://WWW.GEEKSFORGEEKS.ORG/PYTHON-PROGRAMMING-LANGUAGE/
6. HTTPS://WWW.PYTHONTUTORIAL.NET/TKINTER/TKINTER-TREEVIEW/