CERTIFICATE

This is to certify that [SOUVIK MAHANTA] a bonafide student of Class XII-[J] has successfully completed the project titled "[SCHOOL ADMISSION SYSTEM]" in the Computer Lab during the Year 2021 – 2022 for the A.I.S.S.C.E Computer Science Practical Term-2 Examination -2022.

It is further certified that this project is the individual work of the Candidate.

_________________ _________________

External Examiner Internal Examiner

DATED: __________

SEAL : __________

ACKNOWLEDGEMENT

I gratefully acknowledge my sincere thanks to our Computer Science Teacher Mr. Sudarshan Kumar Manna for his remarkable, valuable guidance and supervision throughout the project work. I'm also utmost indebted to all my batch mates for their encouragement, help, suggestion and readily helpful service in the successful completion of the project.

I wish to express my deep gratitude and sincere thanks to the Principal, Mrs. Joyoti Chaudhuri, Delhi Public School, School, Ruby Park for her encouragement and for all the facilities that she provided for this project work.

[SOUVIK MAHANTA]

INDEX

HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS:

1. CPU: Intel or AMD processor with 64-bit support; *Recommended:* 2.8 GHz or faster processor.
2. GPU: nVidia GeForce GTX 1050 or equivalent; *Recommended:* nVidia GeForce GTX 1660 or Quadro T1000
3. Disk Storage: 256 GB of free disk space
4. Monitor Resolution: 1280 x 800; *Recommended:* 1920 x 1080
5. Internet: Internet connection required for software activation
6. Ram: 8GB or higher

SOFTWARE REQUIREMENTS:

Operating System:

Microsoft Windows 10 / Mac Os

Programming IDE:

1)Python 3.10.0

2)MySQL x86 64-bit – version 8.0

OBJECTIVES OF THE PROJECT

Objective of the Project:

The main goal of the system is to automate the process carried out in the organization with improved performance and reduced work load.

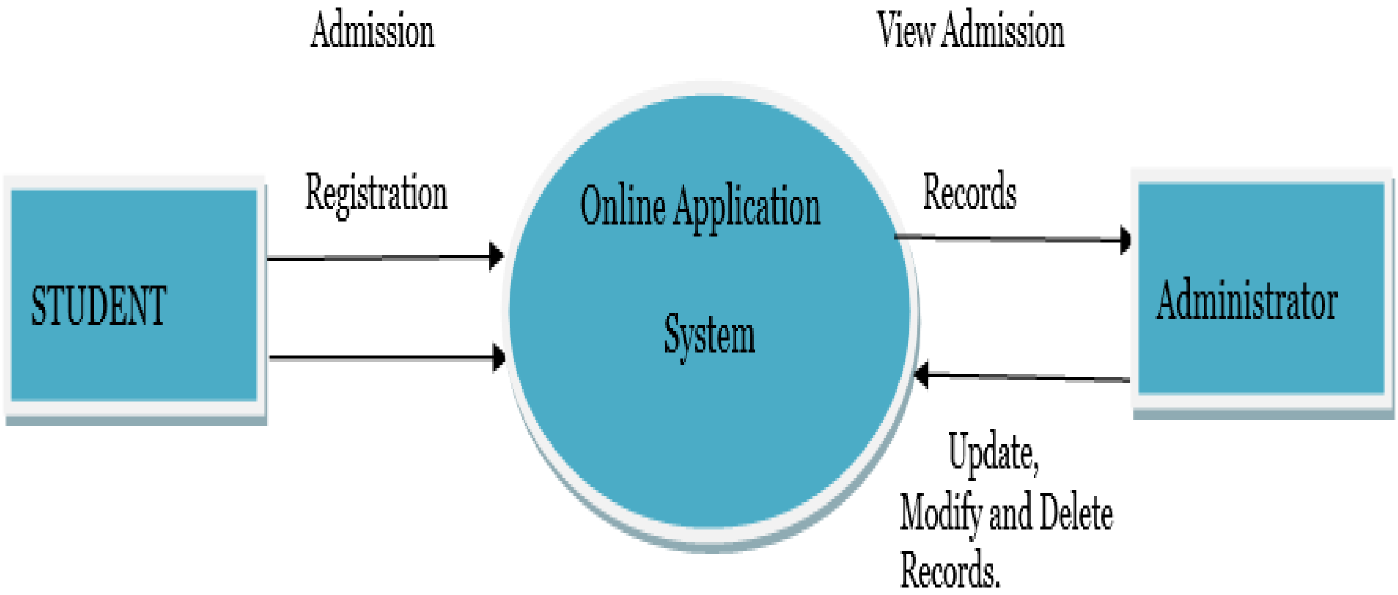
Features of the Proposed System:

School Admission System is aimed at developing an online admission application for a school. This is an online system that can be accessed outside as well. Our system has two type of accessing modes, administrator and student/parent. This entire system is managed by an administrator. Students are required to register themselves with unique username and password and fill out some necessary details. All the details entered by the students will be stored in a database. The administrator has got the access to the database. He / She has a comprehensive set of tools which helps him/her to make necessary changes such as updating, modification and deletion of records.

Advantages of the proposed system:

Today in school student details are entered manually and it is a tedious task. There are chances for more manual errors. As the strength of the students are increasing at a tremendous speed, manual maintenance of student admission is very difficult. Hence, the need for online admission is inevitable. Maintaining backup is very easy. Here almost all the works are computerized and can be done within a few clicks.

PROJECT DESIGN



THE EDUCATIONAL BASED MANAGEMENT SYSTEM HAS 2 PANELS:

1.) Student / Parents

2.) Administrator

INPUTS OF THE PROPOSED SYSTEM:

Input:

1) **Student Panel:**

   Students have to register themselves by filling

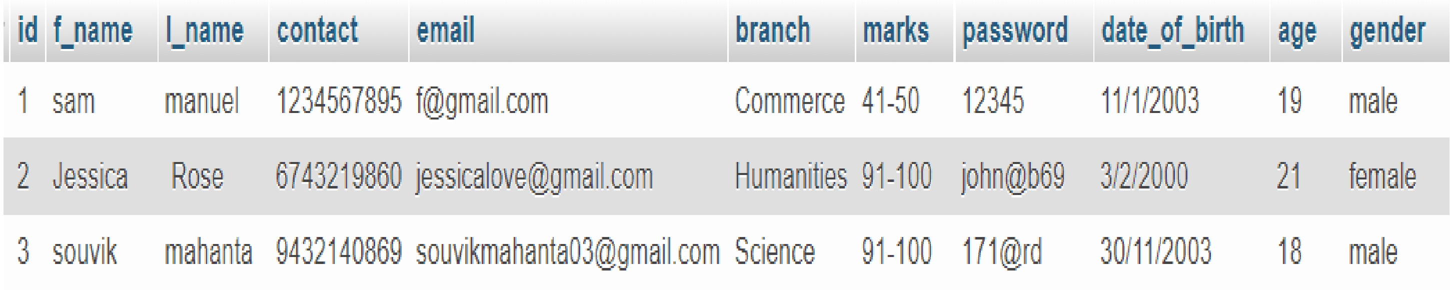
   out some necessary details.

2) **Administrator Panel**

   The administrator has to login using the correct username and password to view the student details.

DATABASE DESIGN

TABLE STRUCTURE :



id	f_name	l_name	contact	email	branch	marks	password	date_of_birth	age	gender
1	sam	manuel	1234567895	f@gmail.com	Commerce	41-50	12345	11/1/2003	19	male
2	Jessica	Rose	6743219860	jessicalove@gmail.com	Humanities	91-100	john@b69	3/2/2000	21	female
3	souvik	mahanta	9432140869	souvikmahanta03@gmail.com	Science	91-100	171@rd	30/11/2003	18	male

MODULE DOCUMENTATION

Tkinter DESCRIPTION :

Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. This Python framework provides an interface to the Tk toolkit and works as a thin object-oriented layer on top of Tk. The Tk toolkit is a cross-platform collection of ‘graphical control elements’, as known as widgets, for building application interfaces.

FUNCTIONS :

1. **Ttk :**

   This module provides classes to allow using Tk themed widget set. Its basic idea is to separate, to the extent possible, the code implementing a widget's behavior from the code implementing its appearance. Widget class bindings are primarily responsible for maintaining the widget state and invoking callbacks, all aspects of the widgets appearance lies at Themes.

2. **Messagebox :**

   tk common message boxes this module provides an interface to the native message boxes available in Tk 4.2 and newer. showerror(title=None, message=None, **options) : Show an error message showinfo(title=None, message=None, **options) : Show an info message showwarning(title=None, message=None, **options) : Show a warning message.

3. **ImageTk :**

   The ImageTk module contains support to create and modify Tkinter BitmapImage and PhotoImage objects from PIL images.

4. **mysql.connector :**

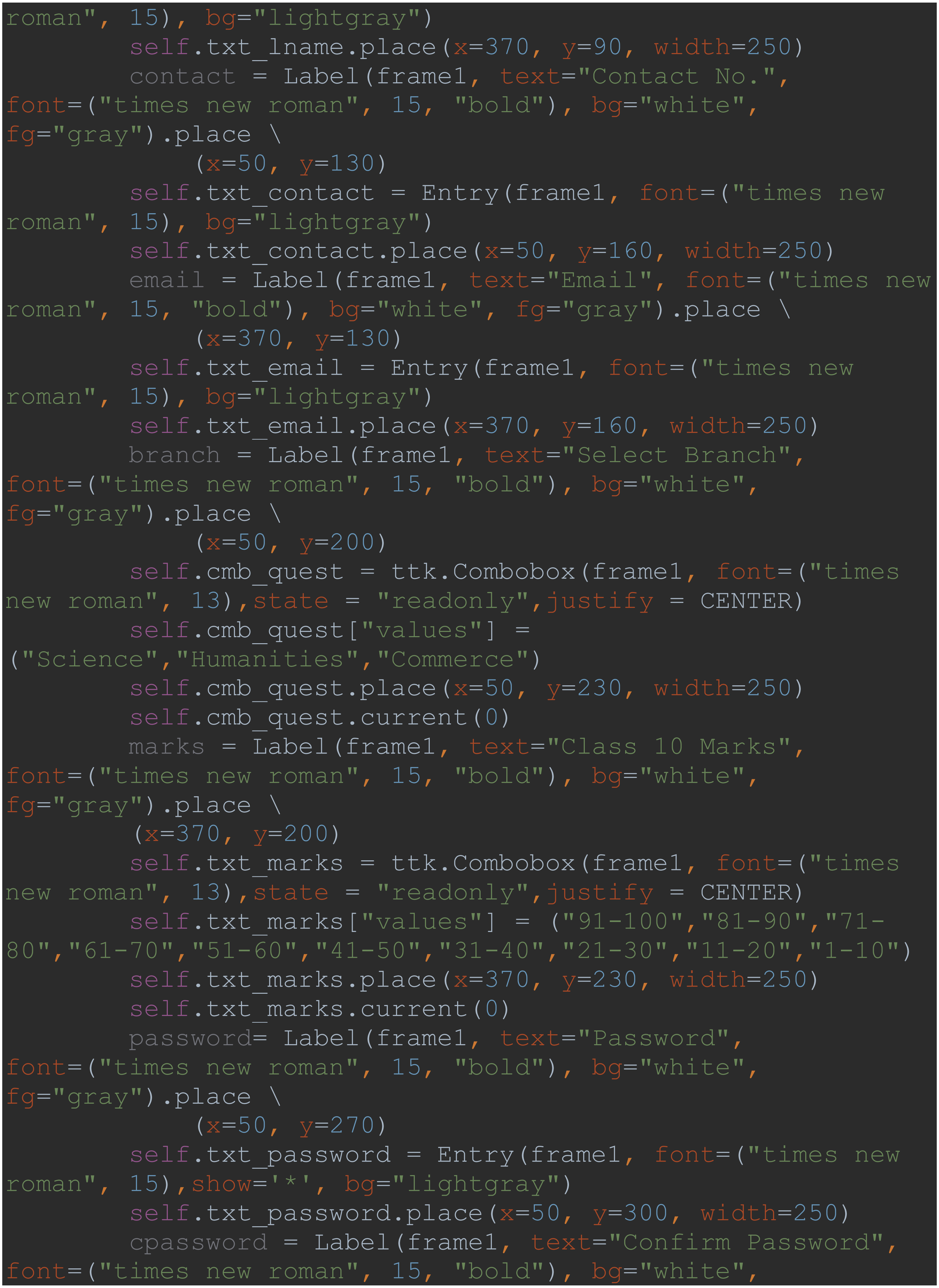
   MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2. 0 (PEP 249).

SOURCE CODE FOR STUDENT PANEL

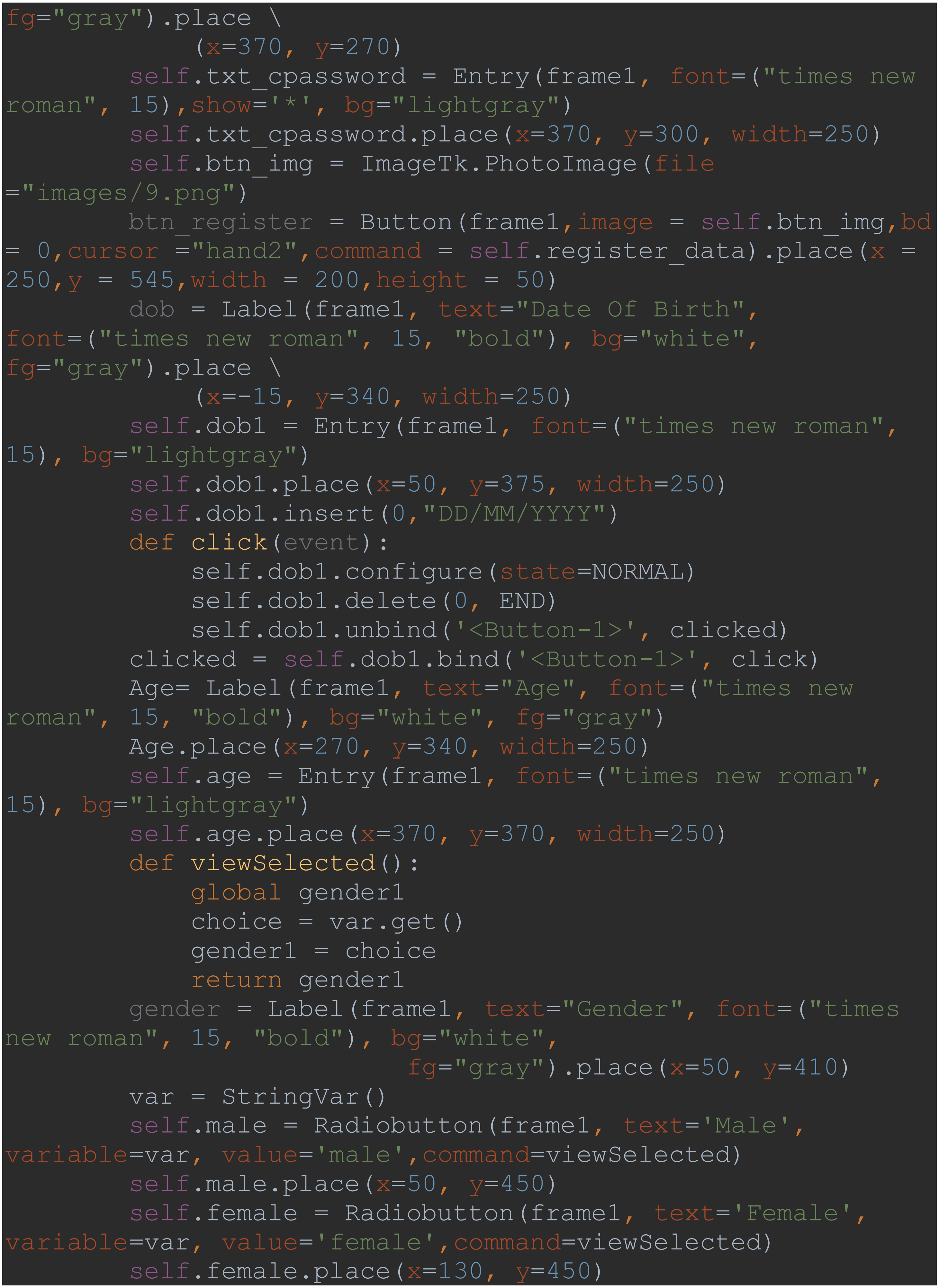
File Name: Student.py

Code:

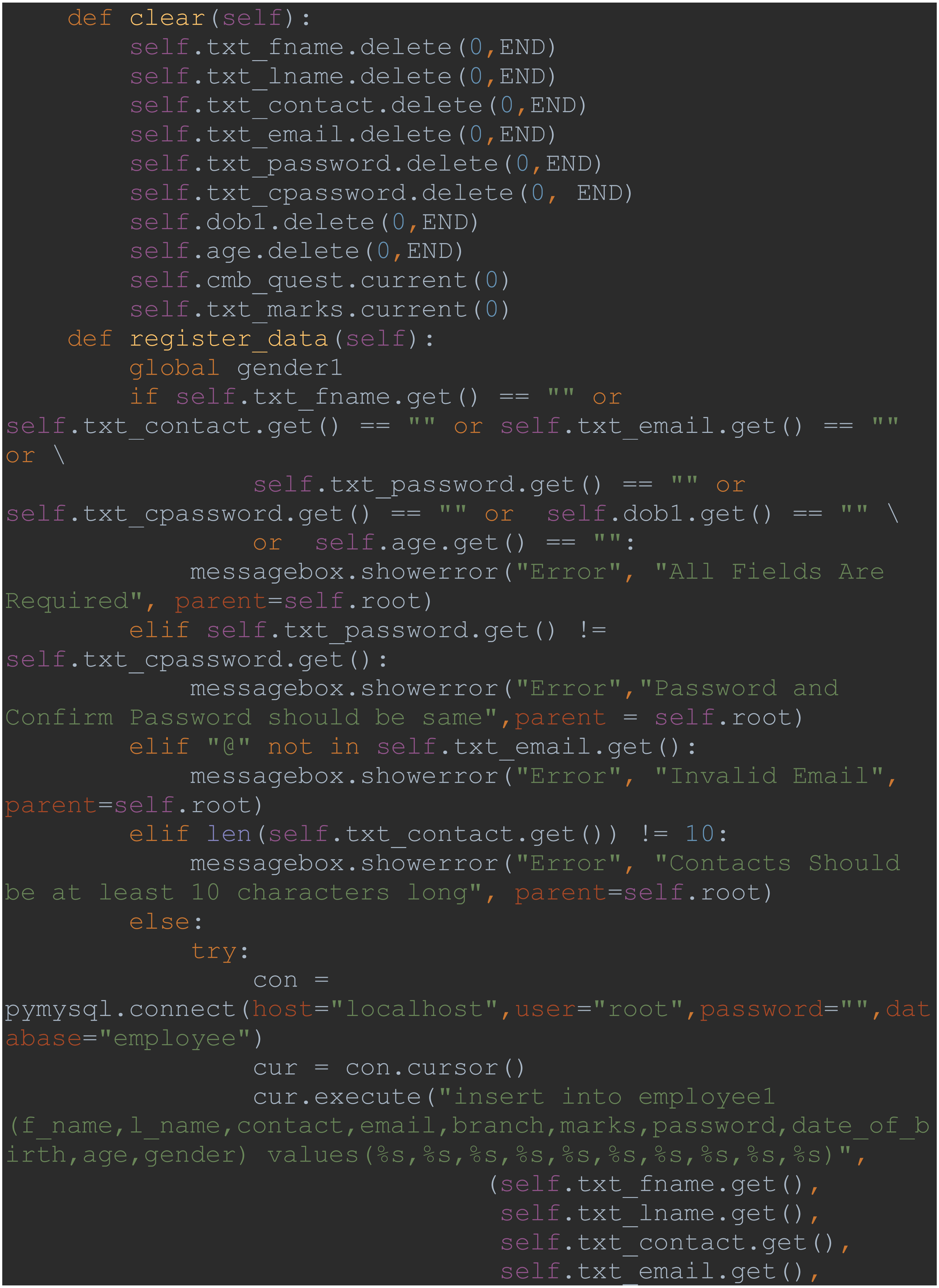
```
from tkinter import *
from PIL import Image, ImageTk
from tkinter import ttk,messagebox
import pymysql
gender1 = "null"
class Register:
    def __init__(self, root):
        self.root = root
        self.root.title("Registration Window")
        self.root.geometry("1600x900+0+0")
        self.root.config(bg="white")
        self.bg = ImageTk.PhotoImage(file="images/3.jpg")
        bg = Label(self.root, image=self.bg).place(x=0, y=0,
width=1600, height=900)
        self.left = ImageTk.PhotoImage(file="images/5.jpeg")
        left = Label(self.root, image=self.left,bg =
"black").place(x=50, y=100, width=600, height=600)
        frame1 = Frame(self.root, bg="white")
        frame1.place(x=650, y=100, width=700, height=600)
        title = Label(frame1, text="REGISTER HERE",
font=("times new roman", 20, "bold"), bg="white",
fg="green").place \
            (x=45, y=10)
        f_name = Label(frame1, text="First Name",
font=("times new roman", 15, "bold"), bg="white",
fg="gray").place \
            (x=50, y=60)
        self.txt_fname = Entry(frame1, font=("times new
roman", 15), bg="lightgrey")
        self.txt_fname.place(x=50, y=90, width=250)
        l_name = Label(frame1, text="Last Name",
font=("times new roman", 15, "bold"), bg="white",
fg="gray").place \
            (x=370, y=60)
        self.txt_lname = Entry(frame1, font=("times new
```



```
roman", 15), bg="lightgray")
        self.txt_lname.place(x=370, y=90, width=250)
        contact = Label(frame1, text="Contact No.",
font=("times new roman", 15, "bold"), bg="white",
fg="gray").place \
            (x=50, y=130)
        self.txt_contact = Entry(frame1, font=("times new
roman", 15), bg="lightgray")
        self.txt_contact.place(x=50, y=160, width=250)
        email = Label(frame1, text="Email", font=("times new
roman", 15, "bold"), bg="white", fg="gray").place \
            (x=370, y=130)
        self.txt_email = Entry(frame1, font=("times new
roman", 15), bg="lightgray")
        self.txt_email.place(x=370, y=160, width=250)
        branch = Label(frame1, text="Select Branch",
font=("times new roman", 15, "bold"), bg="white",
fg="gray").place \
            (x=50, y=200)
        self.cmb_quest = ttk.Combobox(frame1, font=("times
new roman", 13),state = "readonly",justify = CENTER)
        self.cmb_quest["values"] =
("Science","Humanities","Commerce")
        self.cmb_quest.place(x=50, y=230, width=250)
        self.cmb_quest.current(0)
        marks = Label(frame1, text="Class 10 Marks",
font=("times new roman", 15, "bold"), bg="white",
fg="gray").place \
        (x=370, y=200)
        self.txt_marks = ttk.Combobox(frame1, font=("times
new roman", 13),state = "readonly",justify = CENTER)
        self.txt_marks["values"] = ("91-100","81-90","71-
80","61-70","51-60","41-50","31-40","21-30","11-20","1-10")
        self.txt_marks.place(x=370, y=230, width=250)
        self.txt_marks.current(0)
        password= Label(frame1, text="Password",
font=("times new roman", 15, "bold"), bg="white",
fg="gray").place \
            (x=50, y=270)
        self.txt_password = Entry(frame1, font=("times new
roman", 15),show='*', bg="lightgray")
        self.txt_password.place(x=50, y=300, width=250)
        cpassword = Label(frame1, text="Confirm Password",
font=("times new roman", 15, "bold"), bg="white",
```



```
fg="gray").place \
            (x=370, y=270)
        self.txt_cpassword = Entry(frame1, font=("times new
roman", 15),show='*', bg="lightgray")
        self.txt_cpassword.place(x=370, y=300, width=250)
        self.btn_img = ImageTk.PhotoImage(file
="images/9.png")
        btn_register = Button(frame1,image = self.btn_img,bd
= 0,cursor ="hand2",command = self.register_data).place(x =
250,y = 545,width = 200,height = 50)
        dob = Label(frame1, text="Date Of Birth",
font=("times new roman", 15, "bold"), bg="white",
fg="gray").place \
            (x=-15, y=340, width=250)
        self.dob1 = Entry(frame1, font=("times new roman",
15), bg="lightgray")
        self.dob1.place(x=50, y=375, width=250)
        self.dob1.insert(0,"DD/MM/YYYY")
        def click(event):
            self.dob1.configure(state=NORMAL)
            self.dob1.delete(0, END)
            self.dob1.unbind('<Button-1>', clicked)
        clicked = self.dob1.bind('<Button-1>', click)
        Age= Label(frame1, text="Age", font=("times new
roman", 15, "bold"), bg="white", fg="gray")
        Age.place(x=270, y=340, width=250)
        self.age = Entry(frame1, font=("times new roman",
15), bg="lightgray")
        self.age.place(x=370, y=370, width=250)
        def viewSelected():
            global gender1
            choice = var.get()
            gender1 = choice
            return gender1
        gender = Label(frame1, text="Gender", font=("times
new roman", 15, "bold"), bg="white",
                          fg="gray").place(x=50, y=410)
        var = StringVar()
        self.male = Radiobutton(frame1, text='Male',
variable=var, value='male',command=viewSelected)
        self.male.place(x=50, y=450)
        self.female = Radiobutton(frame1, text='Female',
variable=var, value='female',command=viewSelected)
        self.female.place(x=130, y=450)
```



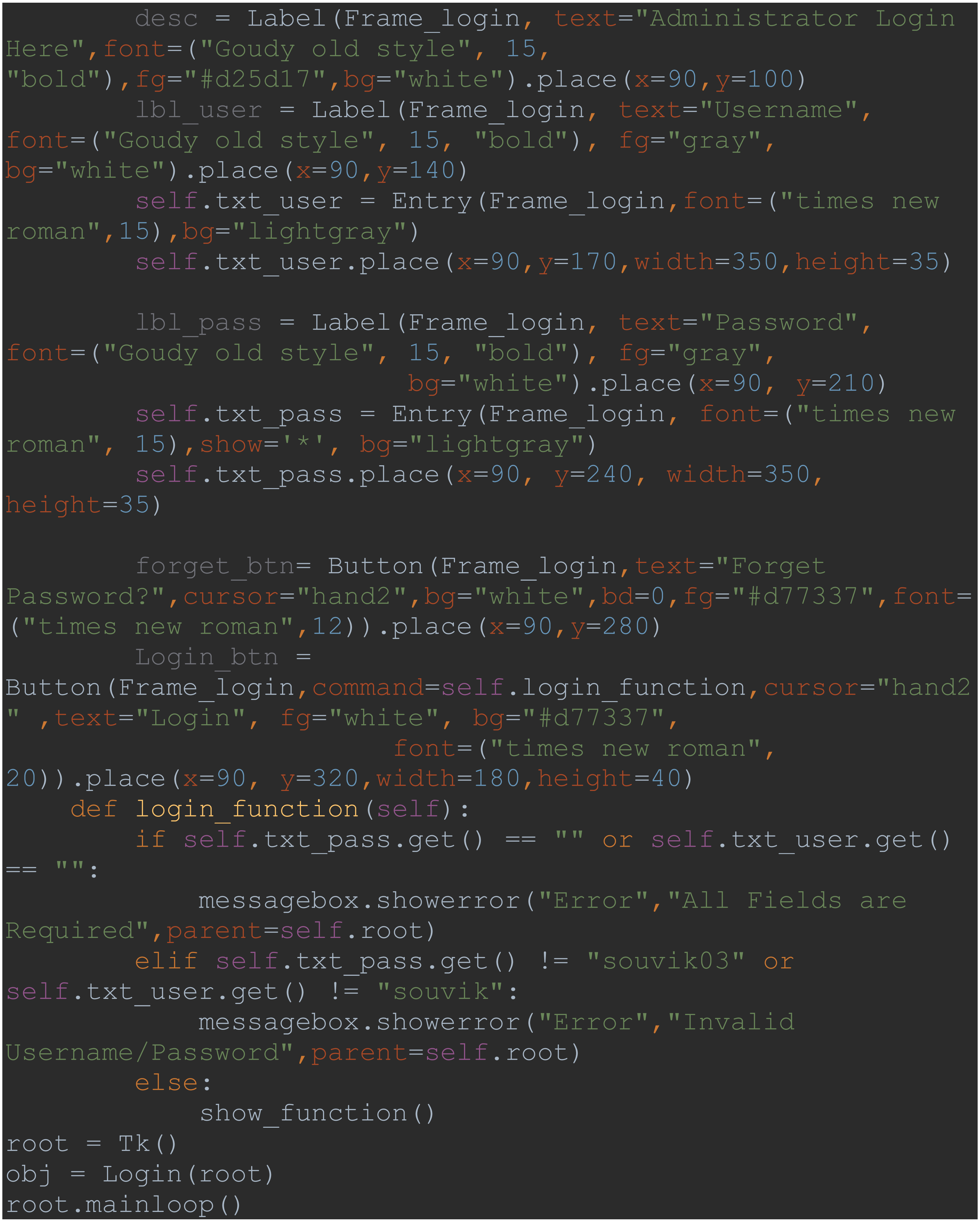
```
    def clear(self):
        self.txt_fname.delete(0,END)
        self.txt_lname.delete(0,END)
        self.txt_contact.delete(0,END)
        self.txt_email.delete(0,END)
        self.txt_password.delete(0,END)
        self.txt_cpassword.delete(0, END)
        self.dob1.delete(0,END)
        self.age.delete(0,END)
        self.cmb_quest.current(0)
        self.txt_marks.current(0)
    def register_data(self):
        global gender1
        if self.txt_fname.get() == "" or
self.txt_contact.get() == "" or self.txt_email.get() == ""
or \
                self.txt_password.get() == "" or
self.txt_cpassword.get() == "" or  self.dob1.get() == "" \
                or  self.age.get() == "":
            messagebox.showerror("Error", "All Fields Are
Required", parent=self.root)
        elif self.txt_password.get() !=
self.txt_cpassword.get():
            messagebox.showerror("Error","Password and
Confirm Password should be same",parent = self.root)
        elif "@" not in self.txt_email.get():
            messagebox.showerror("Error", "Invalid Email",
parent=self.root)
        elif len(self.txt_contact.get()) != 10:
            messagebox.showerror("Error", "Contacts Should
be at least 10 characters long", parent=self.root)
        else:
            try:
                con =
pymysql.connect(host="localhost",user="root",password="",dat
abase="employee")
                cur = con.cursor()
                cur.execute("insert into employee1
(f_name,l_name,contact,email,branch,marks,password,date_of_b
irth,age,gender) values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",
                               (self.txt_fname.get(),
                                self.txt_lname.get(),
                                self.txt_contact.get(),
                                self.txt_email.get(),
                                self.cmb_quest.get(),
                                self.txt_marks.get(),
                                self.txt_password.get(),
                                self.dob1.get(),
                                self.age.get(),
                                gender1
                            ))
                con.commit()
                con.close()
                messagebox.showinfo("Success","Registration
Successful",parent = self.root)
                self.clear()
            except Exception as es:
                messagebox.showerror("Error", f"Error Due
to: {str(es)}", parent=self.root)
root = Tk()
obj = Register(root)
root.mainloop()
```

SOURCE CODE FOR ADMIN PANEL

File Name: Admin.py

Code:

```
from tkinter import*
import mysql.connector
from PIL import ImageTk
import tkinter as tk
from tkinter import messagebox
def show_function():
    my_w = tk.Tk()
    my_w.geometry("400x250")
    my_connect = mysql.connector.connect(host="localhost",
user="root", password="", database="employee")
    my_conn = my_connect.cursor()
    my_conn.execute("SELECT * FROM employee1")
    i = 0
    for student in my_conn:
        for j in range(len(student)):
            e = Entry(my_w, width=25, fg='blue')
            e.grid(row=i, column=j)
            e.insert(END, student[j])
            f = Label(my_w, width=10, text=student[j],
borderwidth=2, relief='ridge', anchor="w")
        i = i + 1
    my_w.mainloop()
class Login:
    def __init__(self,root):
        self.root = root
        self.root.title("Login System")
        self.root.geometry("1600x900+0+0")
        self.bg =
ImageTk.PhotoImage(file="images/admin.jpg")
        self.bg_image = Label(self.root,image =
self.bg).place(x=0,y=0,relwidth=1, relheight=1)
        Frame_login = Frame(self.root,bg="white")
        Frame_login.place(x=150,y=150,height=500,width=500)
        title = Label(Frame_login,text="Login
Here",font=("Impact",35,"bold"),fg="#d77337",bg="white").pla
ce(x=90,y=30)
```

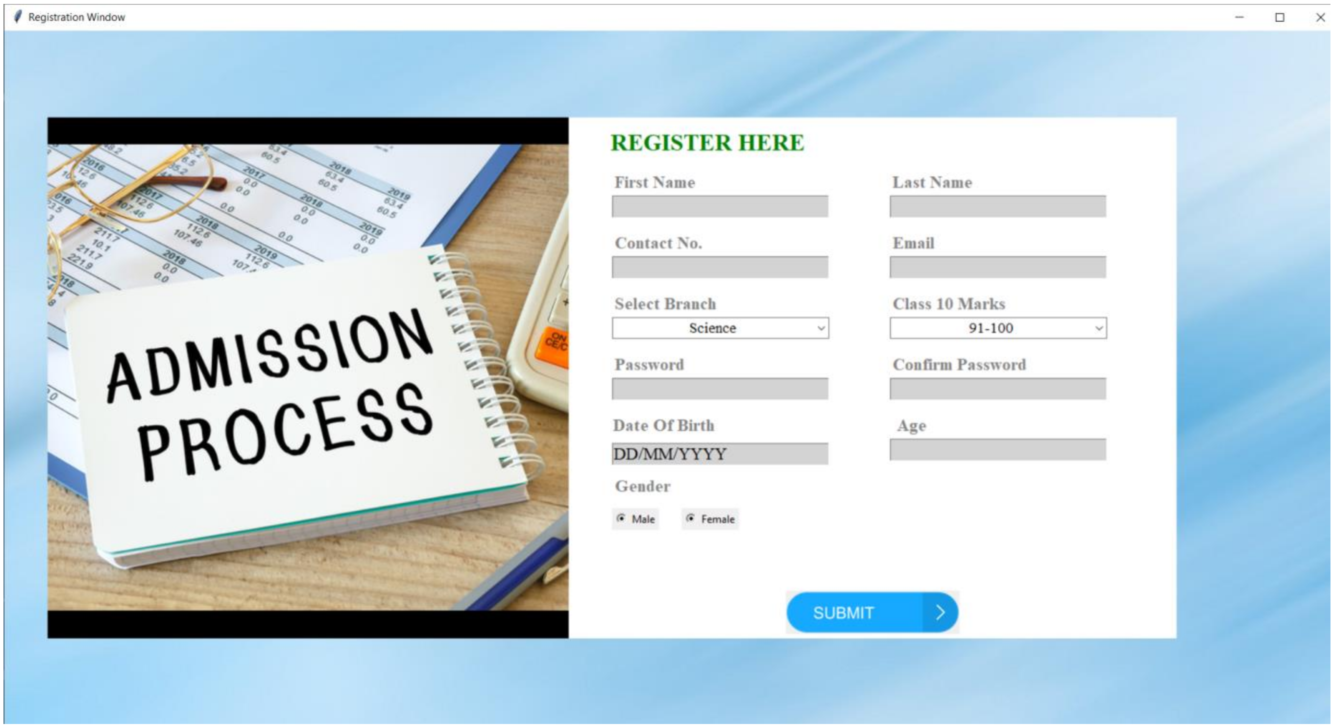


```
        desc = Label(Frame_login, text="Administrator Login
Here",font=("Goudy old style", 15,
"bold"),fg="#d25d17",bg="white").place(x=90,y=100)
        lbl_user = Label(Frame_login, text="Username",
font=("Goudy old style", 15, "bold"), fg="gray",
bg="white").place(x=90,y=140)
        self.txt_user = Entry(Frame_login,font=("times new
roman",15),bg="lightgray")
        self.txt_user.place(x=90,y=170,width=350,height=35)

        lbl_pass = Label(Frame_login, text="Password",
font=("Goudy old style", 15, "bold"), fg="gray",
                         bg="white").place(x=90, y=210)
        self.txt_pass = Entry(Frame_login, font=("times new
roman", 15),show='*', bg="lightgray")
        self.txt_pass.place(x=90, y=240, width=350,
height=35)

        forget_btn= Button(Frame_login,text="Forget
Password?",cursor="hand2",bg="white",bd=0,fg="#d77337",font=
("times new roman",12)).place(x=90,y=280)
        Login_btn =
Button(Frame_login,command=self.login_function,cursor="hand2
" ,text="Login", fg="white", bg="#d77337",
                        font=("times new roman",
20)).place(x=90, y=320,width=180,height=40)
    def login_function(self):
        if self.txt_pass.get() == "" or self.txt_user.get()
== "":
            messagebox.showerror("Error","All Fields are
Required",parent=self.root)
        elif self.txt_pass.get() != "souvik03" or
self.txt_user.get() != "souvik":
            messagebox.showerror("Error","Invalid
Username/Password",parent=self.root)
        else:
            show_function()
root = Tk()
obj = Login(root)
root.mainloop()
```

OUTPUT SCREEN

1) STUDENT PANEL:

On executing the program Student.py,
the following window is displayed to the user:

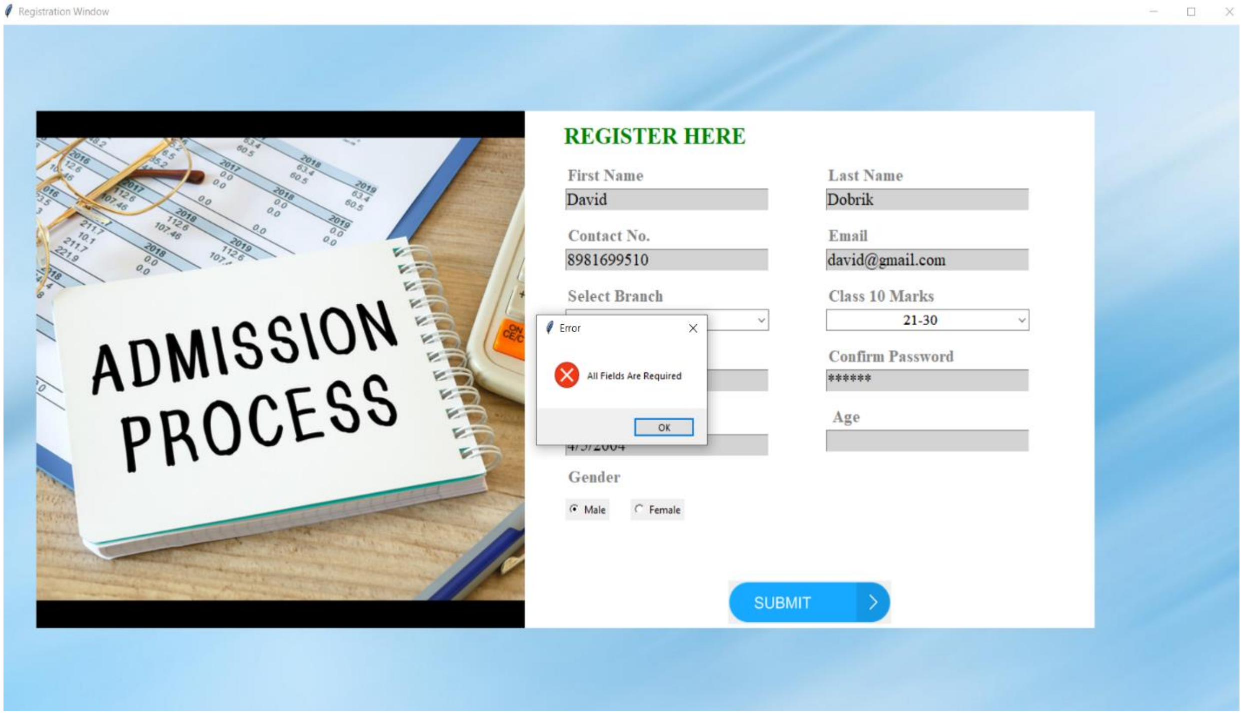


Select Branch: Science

Class 10 Marks: 91-100

Date Of Birth: DD/MM/YYYY

a) If any field is left empty while submitting a popup occurs stating that:



REGISTER HERE

First Name: David

Last Name: Dobrik

Contact No.: 8981699510

Email: david@gmail.com

Select Branch

Class 10 Marks: 21-30

Confirm Password: ******

Age

4/5/2004

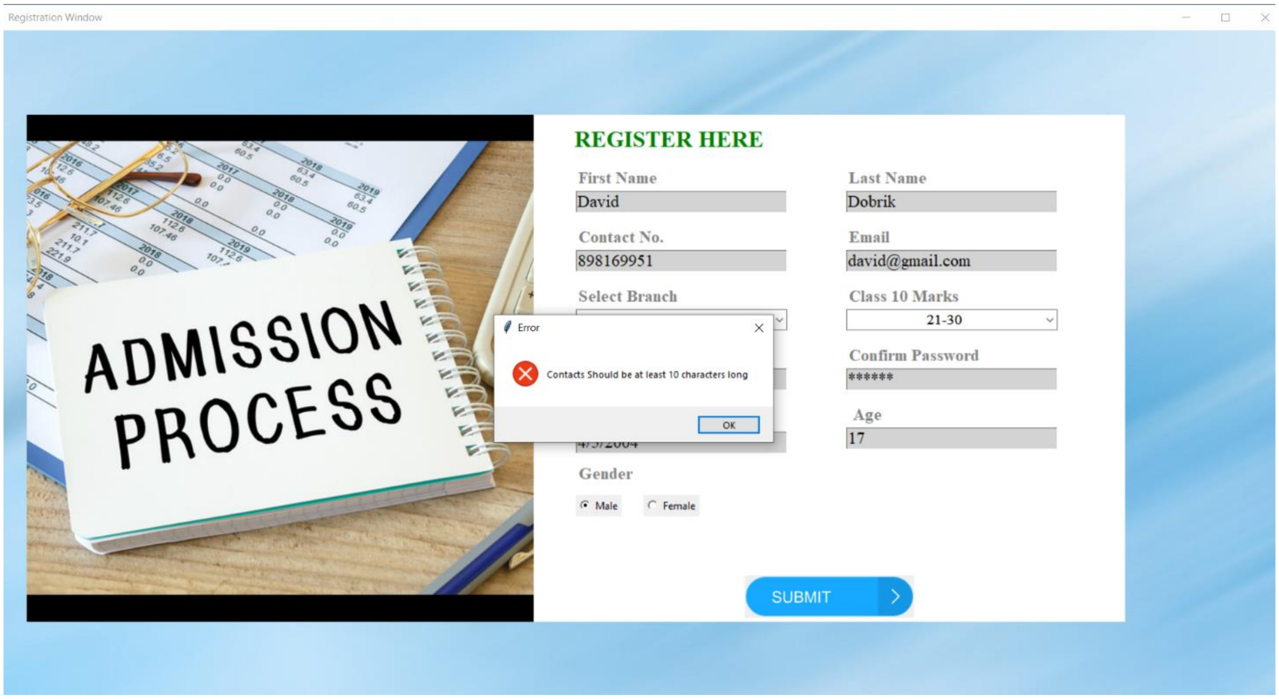
Gender: ● Male ○ Female

Error

All Fields Are Required

OK

b) If the contact is not 10 characters long:



REGISTER HERE

First Name: David

Last Name: Dobrik

Contact No.: 898169951

Email: david@gmail.com

Select Branch

Class 10 Marks: 21-30

Confirm Password: ******

Age: 17

4/5/2004

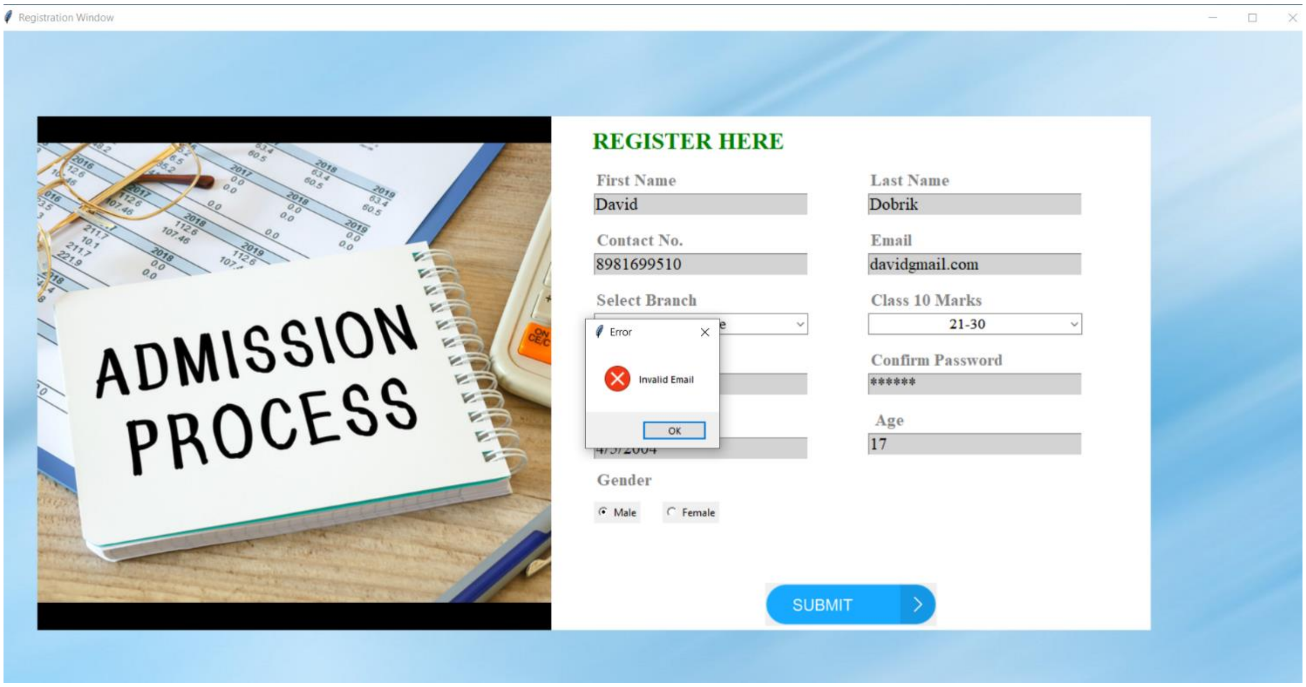
Gender: ● Male ○ Female

Error

Contacts Should be at least 10 characters long

OK

c) If the email is invalid:



Registration Window

REGISTER HERE

First Name: David

Last Name: Dobrik

Contact No.: 8981699510

Email: davidgmail.com

Select Branch

Class 10 Marks: 21-30

Confirm Password: ******

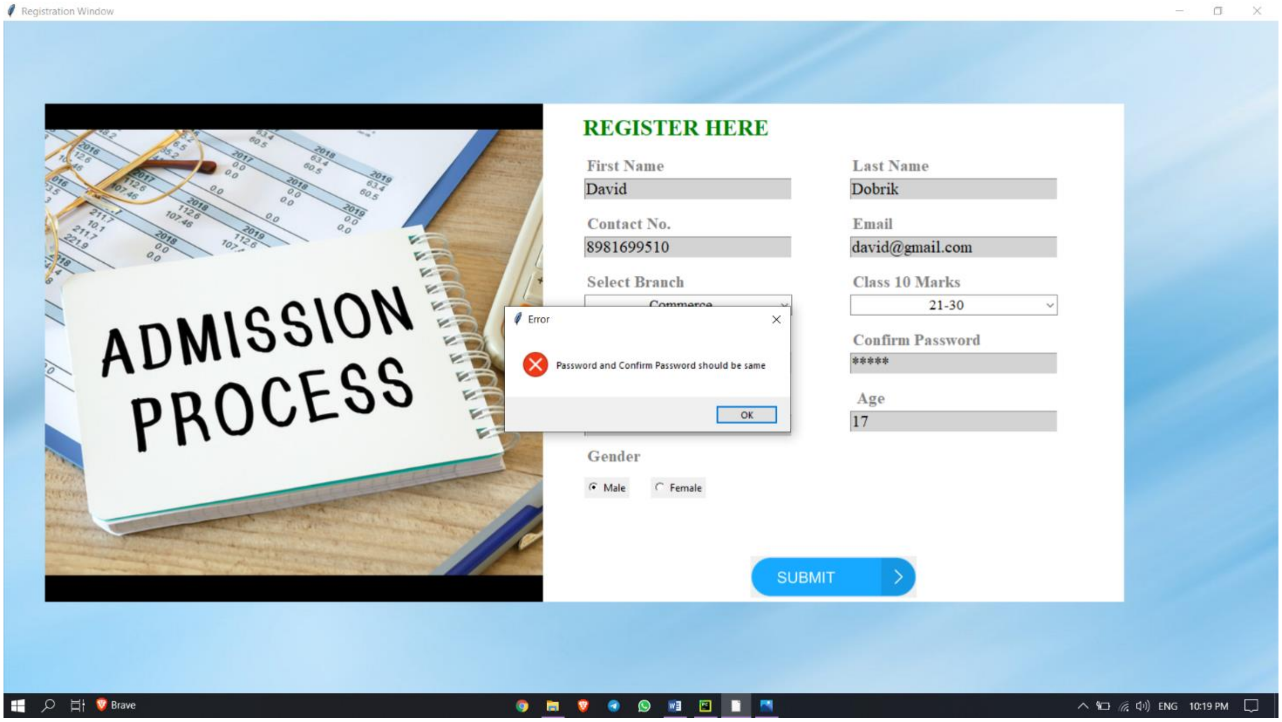
Age: 17

Gender: ● Male ○ Female

Error

Invalid Email

d) If the password and confirm password does not match:



Registration Window

REGISTER HERE

First Name: David

Last Name: Dobrik

Contact No.: 8981699510

Email: david@gmail.com

Select Branch: Commerce

Class 10 Marks: 21-30

Confirm Password: *****

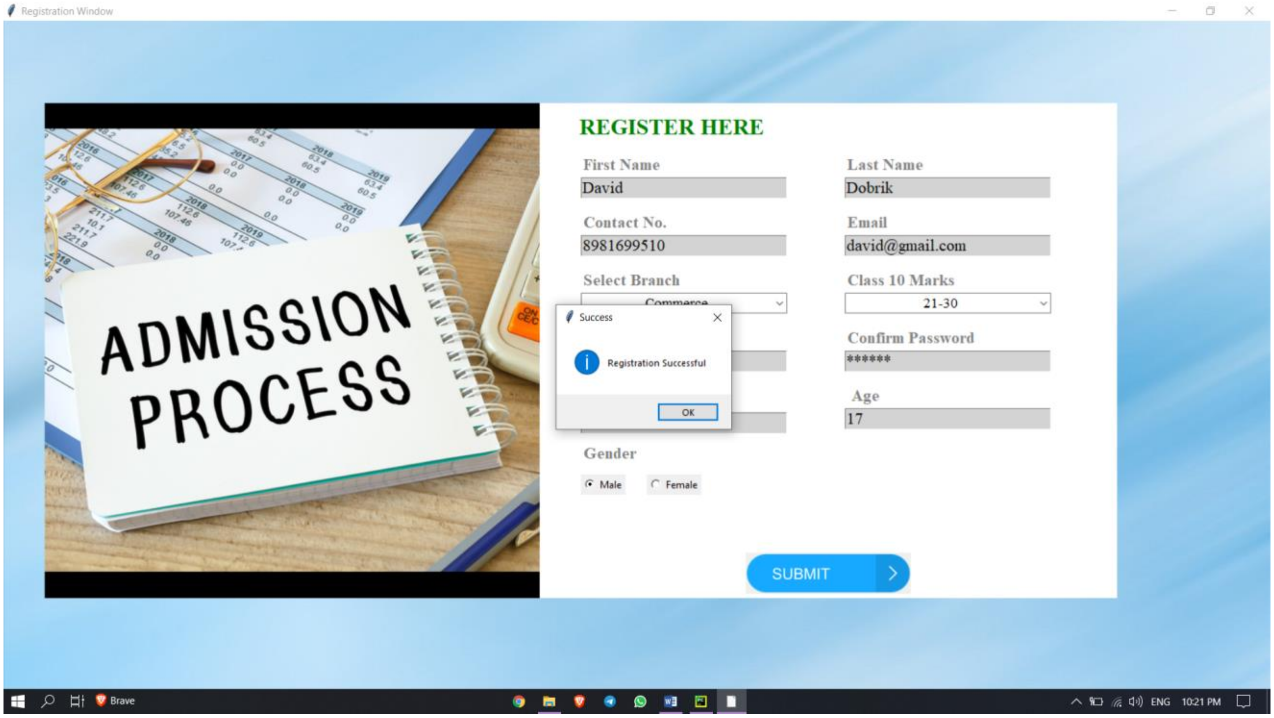
Age: 17

Gender: ● Male ○ Female

Error

Password and Confirm Password should be same

e) If the fields are correctly filled and on clicking the "Submit" button:



REGISTER HERE

First Name: David

Last Name: Dobrik

Contact No.: 8981699510

Email: david@gmail.com

Select Branch: Commerce

Class 10 Marks: 21-30

Confirm Password: ******

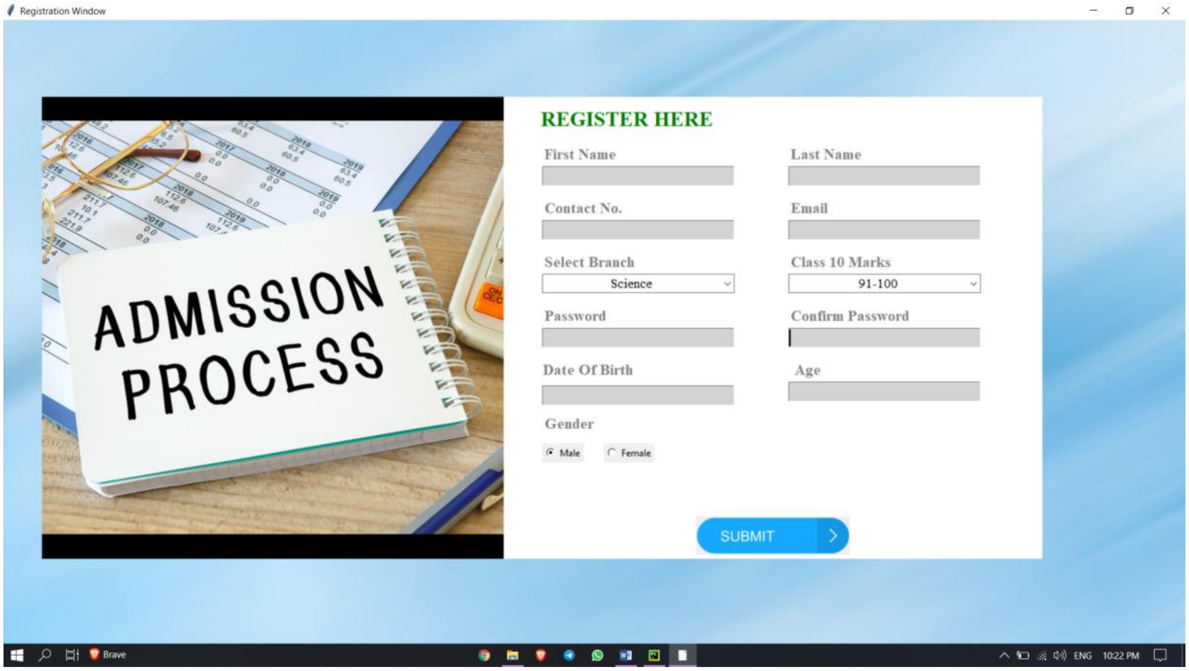
Age: 17

Gender: ● Male ○ Female

Success

Registration Successful

f) On clicking the "OK" button on the popup:



REGISTER HERE

First Name

Last Name

Contact No.

Email

Select Branch: Science

Class 10 Marks: 91-100

Password

Confirm Password

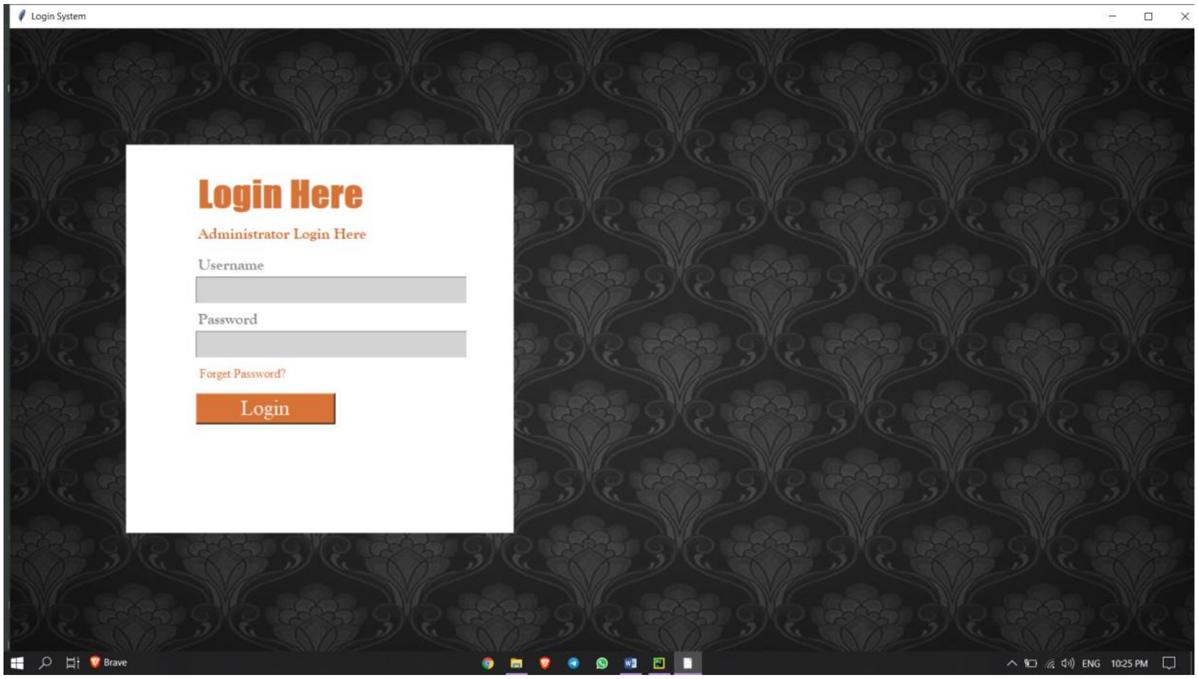
Date Of Birth

Age

Gender: ● Male ○ Female

2) ADMINISTRATOR PANEL:

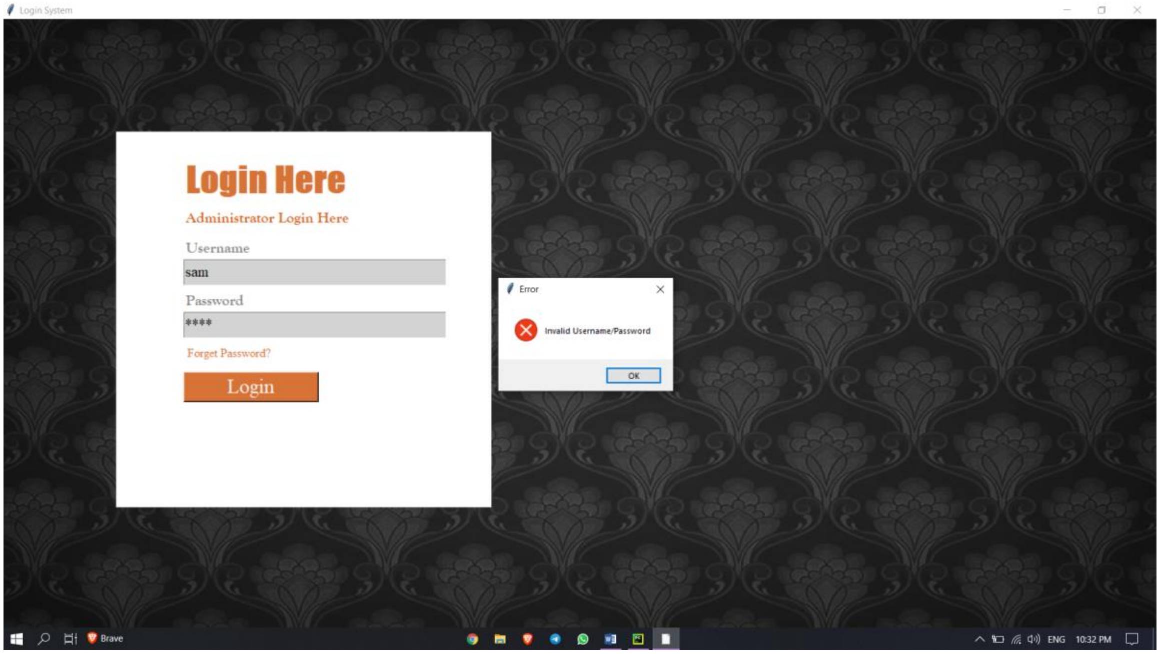
On executing the program Admin.py,
the following window is displayed to the admin:



Here the password and the username is pre-registered by the programmer.

Here,
Username: souvik
Password: souvik03

a) If the username or the password entered by the admin is incorrect:



Login System

Login Here

Administrator Login Here

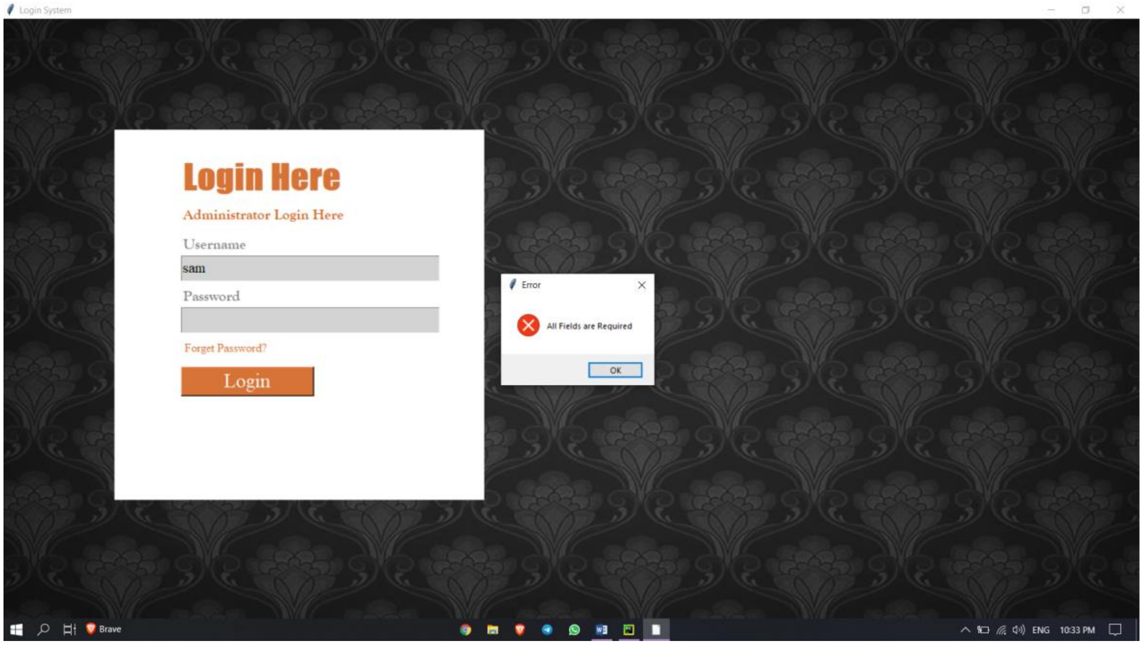
Username: sam

Password: ****

Error

Invalid Username/Password

b) If any of the fields are empty:



Login System

Login Here

Administrator Login Here

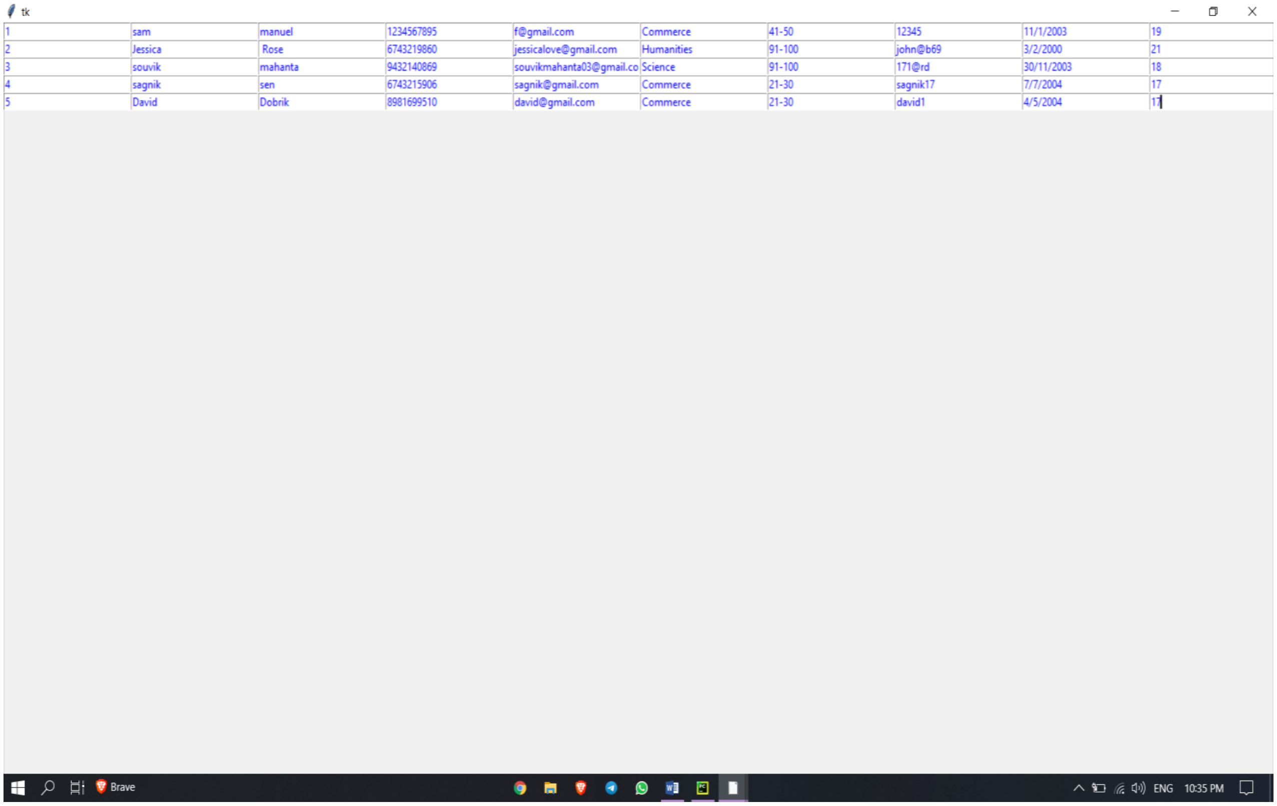
Username: sam

Password:

Error

All Fields are Required

c) If both the username and the password are correct, then on clicking the “Login” button the following window with all the student details are shown:



1	sam	manuel	1234567895	f@gmail.com	Commerce	41-50	12345	11/1/2003	19
2	Jessica	Rose	6743219860	jessicalove@gmail.com	Humanities	91-100	john@b69	3/2/2000	21
3	souvik	mahanta	9432140869	souvikmahanta03@gmail.co	Science	91-100	171@rd	30/11/2003	18
4	sagnik	sen	6743215906	sagnik@gmail.com	Commerce	21-30	sagnik17	7/7/2004	17
5	David	Dobrik	8981699510	david@gmail.com	Commerce	21-30	david1	4/5/2004	17

LIMITATIONS

1. **Online admission integration.**

2. **In the STUDENT panel the prototype takes the value once and once the value is stored the person can't go back to the previous window and change it.**

3. **On further modifications in the ADMIN panel, students can be directly shortlisted on the press of a few buttons. Roll numbers and sections can be allotted to students digitally. Finalizing the fees can be done digitally on the basis of allotted scholarships. Important messages or updates can be sent to the students directly in the form of sms, using platforms such as "Twilio" or by using "Twilio Helper Library".**

BIBLIOGRAPHY & REFERENCES

1. Computer Science for class XII by Sumita Arora

WEBSITE LINKS:

1. https://stackoverflow.com/
2. https://www.geeksforgeeks.org/