

**A**

**MINOR PROJECT**

**ON**

**“Student Result Management System”**

**(IN PYTHON)**

**SUBMITTED TO**

**BHARATI VIDYAPEETH INSTITUTE OF MANAGEMENT, KOLHAPUR.**

**FOR THE PARTIAL FULLFILLMENT OF**

**MASTERS OF COMPUTER APPLICATION**

**(MCA- I, SEM-II)**

**BY**

**Ms. Reshma D. Motwani**

**Ms. Mayuri B. Patel**

**UNDER GUIDANCE OF**

**Prof. Prashant G. Tandale**

**THROUGH**

**THE DIRECTOR**

**BHARATI VIDYAPEETH INSTITUTE OF MANAGEMENT**

**KOLHAPUR**

**2020-2021**



**BHARATI VIDYAPEETH DEEMED TO BE UNIVERSITY INSTITUTE OF MANAGEMENT, KOLHAPUR.**

# DIRECTOR’S RECOMMENDATION

To,

The Registrar,

Bharati Vidyapeeth University, Kolhapur.

SUB: Request for approval of MCA-I (Sem- II) project report.

Sir,

I recommend the project report entitled **“Student Result Management System”** prepared by **Ms. Reshma D. Motwani** and **Ms**. **Mayuri B. Patel** under the guidance of **Prof. Prashant G. Tandale** as per practical fulfillment of university requirements for the award of partial fulfillment of Masters of Computer Application (MCA - I) degree of Bharati Vidyapeeth Institute of Management, Kolhapur.

Thanking You. Dr. Ravindra A. Marathe

Place: Kolhapur (**DIRECTOR)**

|  |  |
| --- | --- |
| Date: |  |
|  |  |



**BHARATI VIDYAPEETH UNIVERSITY INSTITUTE OF MANAGEMENT KOLHAPUR.**

# GUIDE’S CERTIFICATE

This is to certify that **Ms. Reshma D. Motwani** and **Ms. Mayuri B Patel** under the guidance of **Prof. Prashant G. Tandale** have satisfactorily completed the project work on “**Student Result Management System**” of partial fulfillment of Master of Computer Application (MCA) degree of Bharati Vidyapeeth Institute of Management, Kolhapur during the academic year 2020-21.

To the best of my knowledge & belief the matter presented by them is original & not copied for any source. Also this report has not been submitted earlier for the award of partial fulfillment of any Degree or Diploma of Bharati Vidyapeeth University or any other University**.**

Place: Kolhapur **Prof. Prashant G. Tandale**

Date: (Project Guide)



**BHARATI VIDYAPEETH UNIVERSITY INSTITUTE OF MANAGEMENT KOLHAPUR.**

# DECLARATION

We undersigned hereby declare that the project entitled “**Student Result Management System”** developed in **PYTHON** as per prescribed by Bharati Vidyapeeth Institute of Management Kolhapur, during the academic year 2020-21 under the guidance of **Prof**. **Prashant G. Tandale.** The project presented is developed by us independently and is not duplication from any other source. We understand that any such copying is liable to be punishable in any way by the University authorities deem it.

The work is hereby submitted to Bharati Vidyapeeth Deemed to be university for the practical fulfillment of **MCA**-**I** (**Sem**-**II**).

Place:Kolhapur

Date:

**Ms. Reshma D. Motwani Ms. Mayuri B. Patel**



**BHARATI VIDYAPEETH**

**DEEMED TO BE UNIVERSITY INSTITUTE OF MANAGEMENT, KOLHAPUR.**

# CERTIFICATE

This to certify that

**Ms. Reshma D. Motwani** and **Ms. Mayuri B. Patel** bearing the **Bharati Vidyapeeth Deemed to be University, Institute of Management**, have successfully completed the project entitled **“Student Result Management System”** as part of the course curriculum in our organization.

They have completed the project using visual studio code editor technologies under the guidance and supervision of Prof. P. G. Tandale. They have completed the assigned project well within the time frame.

**Prof. P.G. Tandale Dr. K.M. Alaskar**

**(Co-ordinator) (HOD)**



**BHARATI VIDYAPEETH UNIVERSITY INSTITUTE OF MANAGEMENT KOLHAPUR.**

# ACKNOWLEDGEMENT

We have great pleasure while submitting this Minor Project report entitled **“Student Result Management System”** in partial fulfilment of MCA –I (Sem II) Minor Project. While submitting this minor project I take the opportunity to thank to those directly & indirectly related to minor project work, without their active co-operation & guidance it would have become extremely difficult to complete this task on time. I would like to express my highest regards to our Guide **Prof**. **Prashant G. Tandale** for this enormous help.

**Ms. Reshma D. Motwani Ms. Mayuri B. Patel**

INDEX

|  |  |
| --- | --- |
| SR. NO. | TITLE |
| 1. | ABSTRACT |
| 2. | INTRODUCTION OF PROJECT |
| 3. | SYSTEM ANALYSIS |
| 4. | FEASIBILITY ANALYISIS |
| 5. | SYSTEM REQUIREMENT |
| 6. | PROJECT DESCRIPTION |
| 7. | SCOPE OF PROJECT |
| 8. | SOURCE CODE |
| 9. | OUTPUT |
| 10. | CONCLUSION |
| 11. | BIBLIOGRAPHY |

1. **ABSTRACT**

Technology in today’s world has reached to extent that it can be used to do various tasks in day-to-day life easily with less effort and time. World today has realized importance of education in one’s life which has led to revolution in field of education. But, still most of the institutions handle the results of the students manually which takes lots of efforts and time. All the paper work is now handled online with the help of database and software. Software’s are now used for storing huge amount of data and hence allowing the users to search for a particular record in no time. Using software’s not only saves time but it also reduces human efforts for searching any record in papers. Software’s also have functionalities for performing certain tasks based on the data given to it which makes it more useful and thus increases the importance of a particular software.

Here we have developed a **“Student Result Management System”** where we can fill the student data and prepare results accordingly. The system provides the admin to handle a huge amount of student’s data, course data, student results, etc. and can perform tasks such as displaying a student’s record and its result. It also has functionality of updating a student’s information or deleting a particular student’s information.

1. **INTRODUCTION**

**Student Result Management System** is developed to computerize the major manual operations which were earlier being done on paper. It is mainly developed to analyze the result and keep track of students. The software can be used for maintaining the student details, searching or editing a particular student and to also display the results of the students. It is a simple project and easy to use for users as it has a user-friendly environment. This project is easy to operate and can be easily understood by the users.

Student result management system is created using python language which helps in reducing the efforts taken by human beings. It is created in such a way that serves an authority to the administrator in managing all the activities on a single platform. We are able to see the individual candidate’s results separately. Here in this project, we have tried to develop a simple software which will handle all the student data.

1. **SYSTEM ANALYSIS:**

1. **Existing System**

Existing process of getting information for the student is done manually. All the wanted information need to be provided by the administration. If the students want to know the results of the particular subject, they have to wait for longer period of time as the work is done manually. Considering the scenario that the institution or organization has more number of students and courses available, it becomes difficult to handle the results of the students manually.

1. **Proposed System**

We have successfully proposed the **“Student Result Management System”** for

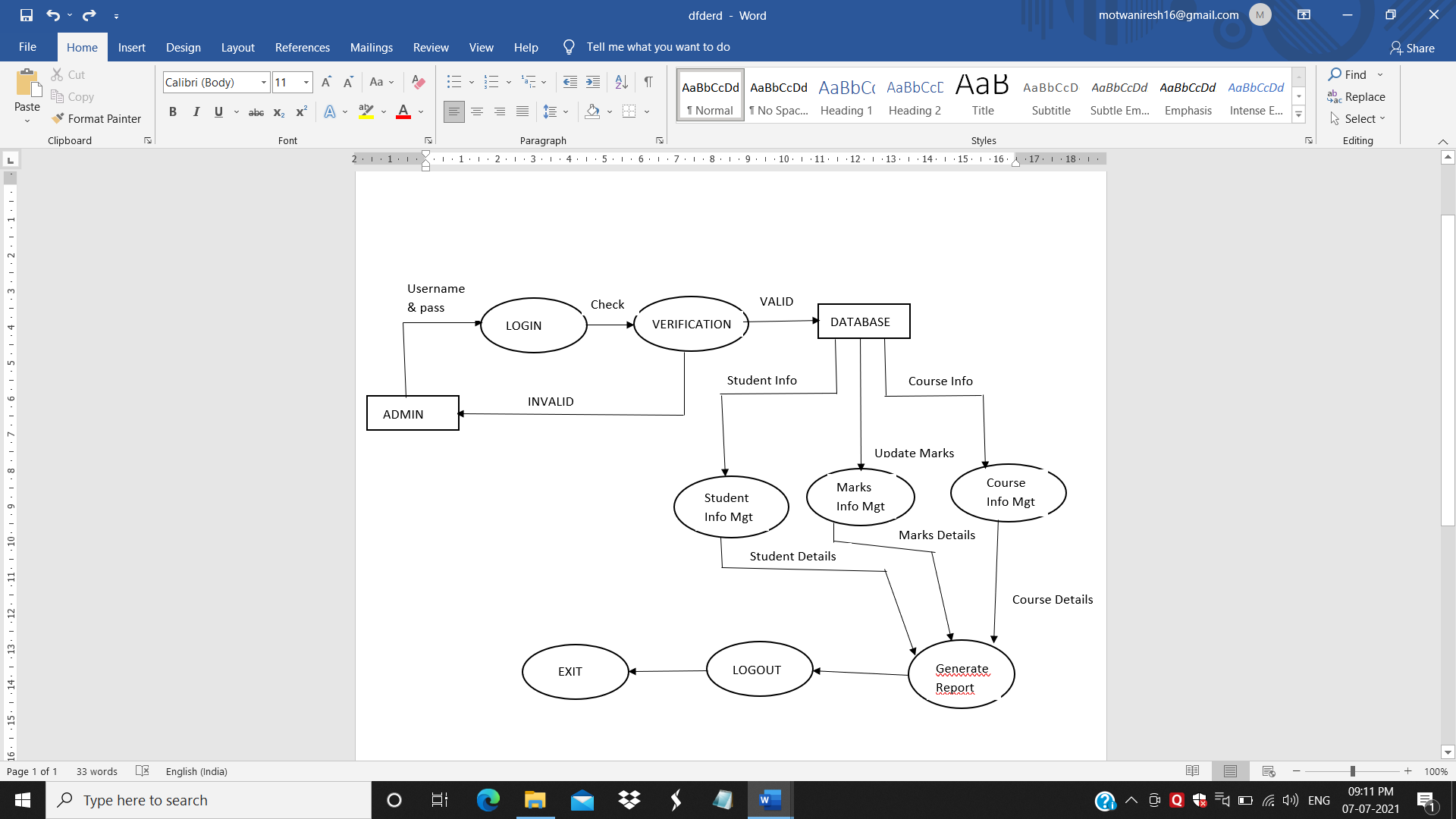
Replacing the manual work of the administration. By this application student can easily access the modules like student results and courses information required to student. It is basically used to store student’s data and their results respectively. It has additional functionalities to edit, delete and update courses as well as student’s information.

1. **Objective of the System**

The main objective behind this project that is **“Student Result Management System”** is to make the manual work computerized and other objectives of the project include:

* To provide ease to the user.
* To save time by minimizing efforts taken by user.
* To keep the student’s data in a computerized manner.
* To get the records of students or any student related data easily.
* To generate the results of the students.

**Data Flow Diagram :**



1. **Feasibility Analysis:**

**Steps in Feasibility Study:**

Feasibility study is carried out in the following steps:

Form a project team and appoint a project leader (Systems Analyst).

Start preliminary investigation through different fact-finding techniques.

Prepare the systems flowcharts of the current system.

Identify and describe the deficiencies in the current system.

Determine objectives of the proposed system.

Prepare the systems flowchart of the proposed system.

Identify and enumerate the existing computer systems along with their technical specifications.

Determine the cost and benefits of the proposed system.

Identify the responsible users and determine the operational feasibility.

Prepare the feasibility study report.

Give the oral presentation of the feasibility study.

**Types of Feasibility**

1. **Technical Feasibility:**

During this study, the analyst identifies the existing computer systems of the concerned department and determines whether these technical resources are sufficient for the proposed system or not. If they are not sufficient, the analyst suggests the configuration of the computer systems that are required. The analyst generally pursues two or three different configurations which satisfy the key technical requirements but which represent different costs. During technical feasibility study, financial resources and budget is also considered. The main objective of technical feasibility is to determine whether the project is technically feasible or not, provided it is economically feasible.

1. **Economic Feasibility:**

Economic Feasibility the most important study that determines the cost and benefits of the proposed system and compares with the budget. The cost of the project should not outweigh the budget. The cost of the project includes the cost of hardware, software, development and implementation. Cost/benefit analysis is the common method to determine the benefits that are expected from the proposed system and compare them with the costs expected to spend on development of the system. c) Operational Feasibility:

When it is found that the project is both economic and technical feasible, the next step is to determine whether it is operationally feasible or not. During operational feasibility study, it is operationally feasible or not, during operational feasibility study, it is determined whether the system will operate in the way that user wants or not. Operational feasibility depends upon human resources for the development and implementation of the system. It is considered whether the qualified or experienced manpower is available for development and implementation of the system or not.

User involvement is more required in determining the operational feasibility.

## System Requirements:

**Hardware Requirements:**

* Pentium-IV(Processor)
* 256 MB Ram
* 512 KB Cache Memory
* Hard disk 10 GB
* Microsoft Compatible 101 or more Key Board
* Mouse

**Software Requirements:**

* **Operating System:** Windows

* **Front-end Language:**  PYTHON
* **Back-end Language:** SQLITE3

1. **PROJECT DESCRIPTION:**

* **TOOLS AND METHODOLOGY:**

**Python:**

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Python's design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its notable use of [significant indentation](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) as well as its [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects.

Python is [dynamically-typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigms), including [structured](https://en.wikipedia.org/wiki/Structured_programming) (particularly, [procedural](https://en.wikipedia.org/wiki/Procedural_programming)), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) and [functional programming](https://en.wikipedia.org/wiki/Functional_programming). Python is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

**SQLite:**

SQLite3 can be integrated with Python using sqlite3 module. Which was written by Gerhard Haring. It provides an SQL interface compliant with the DB-API2.0 specification described by PEP 249. You do not need to install this module separately because it is shipped by default along with Python version 2.5.x onwards.

To use sqlite3 module, you must first create a connection object that represents the database and then optionally you can create a cursor object, which will help you in executing all the SQL statement.

SQLite in general is a server-less database that you can use within almost all programming language including Python. Server-less means there is no need to install a separate server to work with SQLite so you can connect directly with the database.

SQLite is a lightweight database that can provide a relation database management system with zero-configuration because there is no need to configure or set up anything to use it.

* **STUDENT RESULT MANAGEMENT SYSTEM:**

The “Student Result Management System” Project in python with Source Code is a simple project developed using python and mysqlite. This project makes a convenient way for students to view their results online, without having them to wait for a long time in the institution and makes it easier to store information about students and courses. This system has functionalities of generating the results and viewing them easily. Also it has other functionalities of deleting, updating and storing data. This student result management system can be used by the institutions to make their work flow easy.

1. **Scope of Project:**

In this modern world, computer becomes more and more popular and important to our society. Computerized systems are being used everywhere and they are very useful and helpful to our daily life. Computerized systems can be used for scheduling jobs, storing the huge amount of data for complex calculations and also used for generating certain kind of reports.

Since the computerized systems are being used everywhere today. The software has a great scope in our day to day working. The software can be implemented in any educational institution provided that they can modify the system as per their requirements. Also other educational organisations can use the system to store the data of students.

BACKEND MODULES:

* **Dashboard**
* **Course**
* **Result**
* **Report**
* **Login and Logout**

FRONT END MODULES:

* **Home Page**
* **Course Details**
* **Student Details**
* **Generate Results**
* Report (View Results)

1. **Source Code:**
   1. **Login.py**

from tkinter import\*

from PIL import Image,ImageTk,ImageDraw

from datetime import\*

import time

from math import\*

import sqlite3

from tkinter import messagebox

import os

class Login\_window:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("GUI Analog Clock")

self.root.geometry("1350x700+0+0")

self.root.config(bg="#021e2f")

#==============Background colors==============

left\_lbl=Label(self.root,bg="#08A3D2",bd=0)

left\_lbl.place(x=0,y=0,relheight=1,width=600)

right\_lbl=Label(self.root,bg="#031F3C",bd=0)

right\_lbl.place(x=600,y=0,relheight=1,relwidth=1)

#===============Frames=============

login\_frame=Frame(self.root,bg="white")

login\_frame.place(x=250,y=100,width=800,height=500)

title=Label(login\_frame,text="LOGIN HERE",font=("times new roman",30,"bold"),bg="white",fg="#08A3D2").place(x=250,y=50)

email=Label(login\_frame,text="EMAIL ADDRESS",font=("times new roman",18,"bold"),bg="white",fg="gray").place(x=250,y=150)

self.txt\_email=Entry(login\_frame,font=("times new roman",15),bg="lightgray")

self.txt\_email.place(x=250,y=180,width=350,height=35)

pass\_=Label(login\_frame,text="PASSWORD",font=("times new roman",18,"bold"),bg="white",fg="gray").place(x=250,y=250)

self.txt\_pass\_=Entry(login\_frame,font=("times new roman",15),bg="lightgray")

self.txt\_pass\_.place(x=250,y=280,width=350,height=35)

btn\_reg=Button(login\_frame,text="Register New Account?",font=("times new roman",14),bg="white",bd=0,fg="#B00857",cursor="hand2",command=self.register\_window).place(x=250,y=320)

btn\_login=Button(login\_frame,text="Login",font=("times new roman",20,"bold"),fg="white",bg="#B00857",cursor="hand2",command=self.login).place(x=250,y=380,width=180,height=40)

#============Clock===============

self.lbl=Label(self.root,text="\nStudent Result Management System",font=("Book Antiqua",15,"bold"),fg="white",compound=BOTTOM,bg="#081923",bd=0)

self.lbl.place(x=90,y=120,height=450,width=350)

self.working()

def register\_window(self):

self.root.destroy()

import register

def login(self):

if self.txt\_email.get()=="" or self.txt\_pass\_.get()=="":

messagebox.showerror("Error","All Fields Are Required",parent=self.root)

else:

try:

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

cur.execute("select \* from employee where email=? and password=?",(self.txt\_email.get(),self.txt\_pass\_.get()))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Invalid USERNAME & PASSWORD",parent=self.root)

else:

messagebox.showinfo("Success",f"Welcome {self.txt\_email.get()}",parent=self.root)

self.root.destroy()

os.system("python dashboard.py")

con.close()

except Exception as es:

messagebox.showerror("Error",f"Error due to : {str(es)}",parent=self.root)

def clock\_image(self,hr,min\_,sec\_):

clock=Image.new("RGB",(400,400),(8,25,35))

draw=ImageDraw.Draw(clock)

#====For clock image

bg=Image.open("images/c.png")

bg=bg.resize((300,300),Image.ANTIALIAS)

clock.paste(bg,(50,50))

#formula to rotate the AntiClock

#angle\_in\_radians = angle in degrees \* math.pi / 180

#line\_length = 180

#center\_x= 250

#center\_y= 250

#end\_x = cnter\_x + line\_length \* math.cos(angle\_in\_radius)

#======Hour Line Image=====

origin=200,200

draw.line((origin,200+50\*sin(radians(hr)),200-50\*cos(radians(hr))),fill="#DF005E",width=4)

#=======Min Line Image=======

draw.line((origin,200+80\*sin(radians(min\_)),200-80\*cos(radians(min\_))),fill="white",width=3)

#=======SEC Line Image=====

draw.line((origin,200+100\*sin(radians(sec\_)),200-100\*cos(radians(sec\_))),fill="yellow",width=2)

draw.ellipse((195,195,210,210),fill="#1AD5D5")

clock.save("images/clock\_new.png")

def working(self):

h=datetime.now().time().hour

m=datetime.now().time().minute

s=datetime.now().time().second

hr=(h/12)\*360

min\_=(m/60)\*360

sec\_=(s/60)\*360

self.clock\_image(hr,min\_,sec\_)

self.img=ImageTk.PhotoImage(file="images/clock\_new.png")

self.lbl.config(image=self.img)

self.lbl.after(200,self.working)

root=Tk()

obj=Login\_window(root)

root.mainloop()

* 1. **Register.py**

from tkinter import\*

from tkinter import ttk,messagebox

from PIL import Image,ImageTk

import sqlite3

import os

class Register:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Registration Window")

self.root.geometry("1350x700+0+0")

self.root.config(bg="white")

#===========Bg Image=============

self.bg=ImageTk.PhotoImage(file="images/b2.jpg")

bg=Label(self.root,image=self.bg).place(x=250,y=0,relwidth=1,relheight=1)

#=======Left Image===========

self.left=ImageTk.PhotoImage(file="images/side.png")

left=Label(self.root,image=self.left).place(x=80,y=100,width=400,height=500)

#=============Register Frame================

frame1=Frame(self.root,bg="white")

frame1.place(x=480,y=100,width=700,height=500)

title=Label(frame1,text="REGISTER HERE",font=("times new roman",20,"bold"),bg="white",fg="green").place(x=50,y=30)

#================row1============

f\_name=Label(frame1,text="First Name",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=100)

self.txt\_f\_name=Entry(frame1,font=("times new roman",15),bg="lightgray")

self.txt\_f\_name.place(x=50,y=130,width=250)

l\_name=Label(frame1,text="Last Name",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=370,y=100)

self.txt\_l\_name=Entry(frame1,font=("times new roman",15),bg="lightgray")

self.txt\_l\_name.place(x=370,y=130,width=250)

#------------row2-------------

contact=Label(frame1,text="Contact No.",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=170)

self.txt\_contact=Entry(frame1,font=("times new roman",15),bg="lightgray")

self.txt\_contact.place(x=50,y=200,width=250)

email=Label(frame1,text="Email",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=370,y=170)

self.txt\_email=Entry(frame1,font=("times new roman",15),bg="lightgray")

self.txt\_email.place(x=370,y=200,width=250)

#---------------Row3-------------

question=Label(frame1,text="Security Question",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=240)

self.cmb\_quest=ttk.Combobox(frame1,font=("times new roman",13),state='readonly',justify=CENTER)

self.cmb\_quest['values']=("Select","Your First Pet Name","Your Birth Place","Your Bestfriend")

self.cmb\_quest.place(x=50,y=270,width=250)

self.cmb\_quest.current(0)

answer=Label(frame1,text="Answer",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=370,y=240)

self.txt\_answer=Entry(frame1,font=("times new roman",15),bg="lightgray")

self.txt\_answer.place(x=370,y=270,width=250)

#==============row4==============

password=Label(frame1,text="Password",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=50,y=310)

self.txt\_password=Entry(frame1,font=("times new roman",15),bg="lightgray")

self.txt\_password.place(x=50,y=340,width=250)

cpassword=Label(frame1,text="Confirm Password",font=("times new roman",15,"bold"),bg="white",fg="gray").place(x=370,y=310)

self.txt\_cpassword=Entry(frame1,font=("times new roman",15),bg="lightgray")

self.txt\_cpassword.place(x=370,y=340,width=250)

#-----------------Terms\_------------\

self.var\_chk=IntVar()

chk=Checkbutton(frame1,text="I Agree The Terms & Conditions",variable=self.var\_chk,onvalue=1,offvalue=0,bg="white",font=("times new roman",12)).place(x=50,y=380)

self.btn\_img=ImageTk.PhotoImage(file="images/register.png")

btn\_register=Button(frame1,image=self.btn\_img,bd=0,cursor="hand2",command=self.register\_data).place(x=50,y=420)

btn\_login=Button(self.root,text="Sign In",font=("times new roman",20),bd=0,cursor="hand2",command=self.login\_window).place(x=200,y=460,width=180)

def login\_window(self):

self.root.destroy()

os.system("python login.py")

def clear(self):

self.txt\_f\_name.delete(0,END)

self.txt\_l\_name.delete(0,END)

self.txt\_contact.delete(0,END)

self.txt\_email.delete(0,END)

self.txt\_answer.delete(0,END)

self.txt\_password.delete(0,END)

self.txt\_cpassword.delete(0,END)

self.cmb\_quest.current(0)

def register\_data(self):

if self.txt\_f\_name.get()=="" or self.txt\_contact.get()=="" or self.txt\_email.get()=="" or self.cmb\_quest.get()=="Select" or self.txt\_answer.get()=="" or self.txt\_password.get()=="" or self.txt\_cpassword.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 self.var\_chk.get()==0:

messagebox.showerror("Error","Please agree our terms and conditions!",parent=self.root)

else:

try:

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

cur.execute("select \* from employee where email=?",(self.txt\_email.get(),))

row=cur.fetchone()

#print(row)

if row!=None:

messagebox.showerror("Error","User Alrready Exists, Please Try With Another Email!",parent=self.root)

else:

cur.execute("insert into employee(f\_name,l\_name,contact,email,question,answer,password) values(?,?,?,?,?,?,?)",

(self.txt\_f\_name.get(),

self.txt\_l\_name.get(),

self.txt\_contact.get(),

self.txt\_email.get(),

self.cmb\_quest.get(),

self.txt\_answer.get(),

self.txt\_password.get()

))

con.commit()

con.close()

messagebox.showinfo("Success","Register Successful!",parent=self.root)

self.clear()

self.login\_window()

except Exception as es:

messagebox.showerror("Error",f"Error due to: {str(es)}",parent=self.root)

root=Tk()

obj=Register(root)

root.mainloop()

**3. Create\_db.py**

import sqlite3

def create\_db():

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS course(cid INTEGER PRIMARY KEY AUTOINCREMENT,name text,duration text,charges text,description text)")

con.commit()

cur.execute("CREATE TABLE IF NOT EXISTS student(roll INTEGER PRIMARY KEY AUTOINCREMENT,name text,email text,gender text,dob text,contact text,admission text,course text,state text,city text,pin text,address text)")

con.commit()

cur.execute("CREATE TABLE IF NOT EXISTS result(rid INTEGER PRIMARY KEY AUTOINCREMENT,roll text,name text,course text,marks\_ob text,full\_marks text,per text)")

con.commit()

cur.execute("CREATE TABLE IF NOT EXISTS employee(eid INTEGER PRIMARY KEY AUTOINCREMENT,f\_name text,l\_name text,contact text,email text,question text,answer text,password text)")

con.commit()

con.close()

create\_db()

**4. Dashboard.py**

from tkinter import\*

from PIL import Image,ImageTk,ImageDraw

from datetime import \*

import time

from math import \*

import sqlite3

from course import CourseClass

from student import studentClass

from result import resultClass

from report import reportClass

from tkinter import messagebox,ttk

import os

class RMS:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Student Result Management System")

self.root.geometry("1350x700+0+0")

self.root.config(bg="white")

#====icons===

self.logo\_dash=ImageTk.PhotoImage(file="images/logo\_p.png")

#====title===

title=Label(self.root,text="Student Result Management System",padx=10,compound=LEFT,image=self.logo\_dash,font=("goudy old style",20,"bold"),bg="#033054",fg="white").place(x=0,y=0,relwidth=1,height=50)

#=====Menu===

M\_Frame=LabelFrame(self.root,text="Menus",font=("times new roman",15),bg="white")

M\_Frame.place(x=10,y=70,width=1340,height=80)

btn\_course=Button(M\_Frame,text="Course",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add\_course).place(x=20,y=5,width=200,height=40)

btn\_student=Button(M\_Frame,text="Student",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add\_student).place(x=240,y=5,width=200,height=40)

btn\_result=Button(M\_Frame,text="Result",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add\_result).place(x=460,y=5,width=200,height=40)

btn\_view=Button(M\_Frame,text="View Student Results",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.add\_report).place(x=680,y=5,width=200,height=40)

btn\_logout=Button(M\_Frame,text="Logout",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.logout).place(x=900,y=5,width=200,height=40)

btn\_exit=Button(M\_Frame,text="Exit",font=("goudy old style",15,"bold"),bg="#0b5377",fg="white",cursor="hand2",command=self.exit\_).place(x=1120,y=5,width=200,height=40)

#======content window=====

self.bg\_img=Image.open("images/bg.png")

self.bg\_img=self.bg\_img.resize((920,350),Image.ANTIALIAS)

self.bg\_img=ImageTk.PhotoImage(self.bg\_img)

self.lbl\_bg=Label(self.root,image=self.bg\_img).place(x=400,y=180,width=920,height=350)

#======update labels====

self.lbl\_course=Label(self.root,text="Total Courses\n[ 0 ]",font=("goudy old style",20),bd=10,relief=RIDGE,bg="#e43b06",fg="white")

self.lbl\_course.place(x=400,y=530,width=300,height=100)

self.lbl\_student=Label(self.root,text="Total Students\n[ 0 ]",font=("goudy old style",20),bd=10,relief=RIDGE,bg="#0676ad",fg="white")

self.lbl\_student.place(x=710,y=530,width=300,height=100)

self.lbl\_result=Label(self.root,text="Total Results\n[ 0 ]",font=("goudy old style",20),bd=10,relief=RIDGE,bg="#038074",fg="white")

self.lbl\_result.place(x=1020,y=530,width=300,height=100)

#============Clock===============

self.lbl=Label(self.root,text="\nStudent Result Management System",font=("Book Antiqua",15,"bold"),fg="white",compound=BOTTOM,bg="#081923",bd=0)

self.lbl.place(x=10,y=180,height=450,width=350)

self.working()

#====footer===

footer=Label(self.root,text="SRMS-Student Result Management System\nContact Us For Any Technical Issue: 987xxxxx00",font=("goudy old style",12),bg="#262626",fg="white").pack(side=BOTTOM,fill=X)

self.update\_details()

#===============================================================================================================================

def update\_details(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select \* from course")

cr=cur.fetchall()

self.lbl\_course.config(text=f"Total Courses\n[{str(len(cr))}]")

cur.execute("select \* from student")

sr=cur.fetchall()

self.lbl\_student.config(text=f"Total Students\n[{str(len(sr))}]")

cur.execute("select \* from result")

rr=cur.fetchall()

self.lbl\_result.config(text=f"Total Results\n[{str(len(rr))}]")

self.lbl\_course.after(200,self.update\_details)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def working(self):

h=datetime.now().time().hour

m=datetime.now().time().minute

s=datetime.now().time().second

hr=(h/12)\*360

min\_=(m/60)\*360

sec\_=(s/60)\*360

self.clock\_image(hr,min\_,sec\_)

self.img=ImageTk.PhotoImage(file="images/clock\_new.png")

self.lbl.config(image=self.img)

self.lbl.after(200,self.working)

def clock\_image(self,hr,min\_,sec\_):

clock=Image.new("RGB",(400,400),(8,25,35))

draw=ImageDraw.Draw(clock)

#====For clock image

bg=Image.open("images/c.png")

bg=bg.resize((300,300),Image.ANTIALIAS)

clock.paste(bg,(50,50))

#formula to rotate the AntiClock

#angle\_in\_radians = angle in degrees \* math.pi / 180

#line\_length = 180

#center\_x= 250

#center\_y= 250

#end\_x = cnter\_x + line\_length \* math.cos(angle\_in\_radius)

#======Hour Line Image=====

origin=200,200

draw.line((origin,200+50\*sin(radians(hr)),200-50\*cos(radians(hr))),fill="#DF005E",width=4)

#=======Min Line Image=======

draw.line((origin,200+80\*sin(radians(min\_)),200-80\*cos(radians(min\_))),fill="white",width=3)

#=======SEC Line Image=====

draw.line((origin,200+100\*sin(radians(sec\_)),200-100\*cos(radians(sec\_))),fill="yellow",width=2)

draw.ellipse((195,195,210,210),fill="#1AD5D5")

clock.save("images/clock\_new.png")

def add\_course(self):

self.new\_win=Toplevel(self.root)

self.new\_obj=CourseClass(self.new\_win)

def add\_student(self):

self.new\_win=Toplevel(self.root)

self.new\_obj=studentClass(self.new\_win)

def add\_result(self):

self.new\_win=Toplevel(self.root)

self.new\_obj=resultClass(self.new\_win)

def add\_report(self):

self.new\_win=Toplevel(self.root)

self.new\_obj=reportClass(self.new\_win)

def logout(self):

op=messagebox.askyesno("Confirm","Do You Really Want To Logout?",parent=self.root)

if op==True:

self.root.destroy()

os.system("python login.py")

def exit\_(self):

op=messagebox.askyesno("Confirm","Do You Really Want To Exit?",parent=self.root)

if op==True:

self.root.destroy()

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

obj=RMS(root)

root.mainloop()

1. **Course.py**

from tkinter import\*

from PIL import Image,ImageTk

from tkinter import ttk,messagebox

import sqlite35.

class CourseClass:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Student Result Management System")

self.root.geometry("1200x480+80+170")

self.root.config(bg="white")

self.root.focus\_force()

#====title===

title=Label(self.root,text="Manage Course Details",font=("goudy old style",20,"bold"),bg="#033054",fg="white").place(x=10,y=15,width=1180,height=35)

#===========Variables=====

self.var\_course=StringVar()

self.var\_duration=StringVar()

self.var\_charges=StringVar()

#============widgets========

lbl\_courseName=Label(self.root,text="Course Name",font=("goudy old style",15,'bold'),bg='white').place(x=10,y=60)

lbl\_duration=Label(self.root,text="Duration",font=("goudy old style",15,'bold'),bg='white').place(x=10,y=100)

lbl\_charges=Label(self.root,text="Charges",font=("goudy old style",15,'bold'),bg='white').place(x=10,y=140)

lbl\_description=Label(self.root,text="Description",font=("goudy old style",15,'bold'),bg='white').place(x=10,y=180)

#=========entry fields========

self.txt\_courseName=Entry(self.root,textvariable=self.var\_course,font=("goudy old style",15,'bold'),bg='lightyellow')

self.txt\_courseName.place(x=150,y=60,width=200)

txt\_duration=Entry(self.root,textvariable=self.var\_duration,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=150,y=100,width=200)

txt\_charges=Entry(self.root,textvariable=self.var\_charges,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=150,y=140,width=200)

self.txt\_description=Text(self.root,font=("goudy old style",15,'bold'),bg='lightyellow')

self.txt\_description.place(x=150,y=180,width=500,height=130)

#========Buttons======================

self.btn\_add=Button(self.root,text='Save',font=("goudy old style",15,'bold'),bg='#2196f3',fg="white",cursor="hand2",command=self.add)

self.btn\_add.place(x=150,y=400,width=110,height=40)

self.btn\_update=Button(self.root,text='Update',font=("goudy old style",15,'bold'),bg='#4caf50',fg="white",cursor="hand2",command=self.update)

self.btn\_update.place(x=270,y=400,width=110,height=40)

self.btn\_delete=Button(self.root,text='Delete',font=("goudy old style",15,'bold'),bg='#f44336',fg="white",cursor="hand2",command=self.delete)

self.btn\_delete.place(x=390,y=400,width=110,height=40)

self.btn\_clear=Button(self.root,text='Clear',font=("goudy old style",15,'bold'),bg='#607d8b',fg="white",cursor="hand2",command=self.clear)

self.btn\_clear.place(x=510,y=400,width=110,height=40)

#=================Search Panel=======

self.var\_search=StringVar()

lbl\_search\_courseName=Label(self.root,text="Course Name",font=("goudy old style",15,'bold'),bg="white").place(x=720,y=60)

txt\_search\_courseName=Entry(self.root,textvariable=self.var\_search,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=870,y=60,width=180)

btn\_search=Button(self.root,text='Search',font=("goudy old style",15,'bold'),bg='#03a9f4',fg="white",cursor="hand2",command=self.search).place(x=1070,y=60,width=120,height=28)

#==============Content==========

self.C\_Frame=Frame(self.root,bd=2,relief=RIDGE)

self.C\_Frame.place(x=720,y=100,width=470,height=340)

scrolly=Scrollbar(self.C\_Frame,orient=VERTICAL)

scrollx=Scrollbar(self.C\_Frame,orient=HORIZONTAL)

self.CourseTable=ttk.Treeview(self.C\_Frame,columns=("cid","name","duration","charges","description"),xscrollcommand=scrollx.set,yscrollcommand=scrolly.set)

scrollx.pack(side=BOTTOM,fill=X)

scrolly.pack(side=RIGHT,fill=Y)

scrollx.config(command=self.CourseTable.xview)

scrolly.config(command=self.CourseTable.yview)

self.CourseTable.heading("cid",text="Course ID")

self.CourseTable.heading("name",text="Name")

self.CourseTable.heading("duration",text="Duration")

self.CourseTable.heading("charges",text="Charges")

self.CourseTable.heading("description",text="Description")

self.CourseTable["show"]='headings'

self.CourseTable.column("cid",width=100)

self.CourseTable.column("name",width=100)

self.CourseTable.column("duration",width=100)

self.CourseTable.column("charges",width=100)

self.CourseTable.column("description",width=150)

self.CourseTable.pack(fill=BOTH,expand=1)

self.CourseTable.bind("<ButtonRelease-1>",self.get\_data)

self.show()

#=====================================================================

def clear(self):

self.show()

self.var\_course.set("")

self.var\_duration.set("")

self.var\_charges.set("")

self.var\_search.set("")

self.txt\_description.delete('1.0',END)

self.txt\_courseName.config(state=NORMAL)

def delete(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_course.get()=="":

messagebox.showerror("Error","Course Name is required",parent=self.root)

else:

cur.execute("select \* from course where name=?",(self.var\_course.get(),))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Please Select Course From The List First!",parent=self.root)

else:

op=messagebox.askyesno("Confirm","Do you really want to delete?",parent=self.root)

if op==True:

cur.execute("delete from course where name=?",(self.var\_course.get(),))

con.commit()

messagebox.showinfo("Delete","Course Deleted Successfully!",parent=self.root)

self.clear()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def get\_data(self,ev):

self.txt\_courseName.config(state='readonly')

self.txt\_courseName

r=self.CourseTable.focus()

content=self.CourseTable.item(r)

row=content["values"]

self.var\_course.set(row[1])

self.var\_duration.set(row[2])

self.var\_charges.set(row[3])

#self.var\_description.set(row[4])

self.txt\_description.delete('1.0',END)

self.txt\_description.insert(END,row[4])

def add(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_course.get()=="":

messagebox.showerror("Error","Course Name is required",parent=self.root)

else:

cur.execute("select \* from course where name=?",(self.var\_course.get(),))

row=cur.fetchone()

if row!=None:

messagebox.showerror("Error","Course Name already exists",parent=self.root)

else:

cur.execute("insert into course (name,duration,charges,description) values(?,?,?,?)",(

self.var\_course.get(),

self.var\_duration.get(),

self.var\_charges.get(),

self.txt\_description.get("1.0",END)

))

con.commit()

messagebox.showinfo("Success","Course Added Successfully!",parent=self.root)

self.show()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def update(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_course.get()=="":

messagebox.showerror("Error","Course Name is required",parent=self.root)

else:

cur.execute("select \* from course where name=?",(self.var\_course.get(),))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Select Course from list",parent=self.root)

else:

cur.execute("update course set duration=?,charges=?,description=? where name=?",(

self.var\_duration.get(),

self.var\_charges.get(),

self.txt\_description.get("1.0",END),

self.var\_course.get()

))

con.commit()

messagebox.showinfo("Success","Course Updated Successfully!",parent=self.root)

self.show()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def show(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select \* from course")

rows=cur.fetchall()

self.CourseTable.delete(\*self.CourseTable.get\_children())

for row in rows:

self.CourseTable.insert('',END,values=row)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def search(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute(f"select \* from course where name LIKE '%{self.var\_search.get()}%'")

rows=cur.fetchall()

self.CourseTable.delete(\*self.CourseTable.get\_children())

for row in rows:

self.CourseTable.insert('',END,values=row)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

obj=CourseClass(root)

root.mainloop()

1. **Student.py**

from tkinter import\*

from PIL import Image,ImageTk

from tkinter import ttk,messagebox

import sqlite3

class studentClass:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Student Result Management System")

self.root.geometry("1200x480+80+170")

self.root.config(bg="white")

self.root.focus\_force()

#====title===

title=Label(self.root,text="Manage Student Details",font=("goudy old style",20,"bold"),bg="#033054",fg="white").place(x=10,y=15,width=1180,height=35)

#===========Variables=====

self.var\_roll=StringVar()

self.var\_name=StringVar()

self.var\_email=StringVar()

self.var\_gender=StringVar()

self.var\_dob=StringVar()

self.var\_contact=StringVar()

self.var\_course=StringVar()

self.var\_a\_date=StringVar()

self.var\_state=StringVar()

self.var\_city=StringVar()

self.var\_pin=StringVar()

#============widgets========

#==============Column 1=================

lbl\_roll=Label(self.root,text="Roll No.",font=("goudy old style",15,'bold'),bg='white').place(x=10,y=60)

lbl\_Name=Label(self.root,text="Name",font=("goudy old style",15,'bold'),bg='white').place(x=10,y=100)

lbl\_Email=Label(self.root,text="Email",font=("goudy old style",15,'bold'),bg='white').place(x=10,y=140)

lbl\_gender=Label(self.root,text="Gender",font=("goudy old style",15,'bold'),bg='white').place(x=10,y=180)

lbl\_state=Label(self.root,text="State",font=("goudy old style",15,'bold'),bg='white').place(x=10,y=220)

txt\_state=Entry(self.root,textvariable=self.var\_state,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=150,y=220,width=150)

lbl\_city=Label(self.root,text="City",font=("goudy old style",15,'bold'),bg='white').place(x=310,y=220)

txt\_city=Entry(self.root,textvariable=self.var\_city,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=380,y=220,width=100)

lbl\_pin=Label(self.root,text="Pin",font=("goudy old style",15,'bold'),bg='white').place(x=500,y=220)

txt\_pin=Entry(self.root,textvariable=self.var\_pin,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=560,y=220,width=120)

lbl\_address=Label(self.root,text="Address",font=("goudy old style",15,'bold'),bg='white').place(x=10,y=260)

#=========entry fields========

self.txt\_roll=Entry(self.root,textvariable=self.var\_roll,font=("goudy old style",15,'bold'),bg='lightyellow')

self.txt\_roll.place(x=150,y=60,width=200)

txt\_name=Entry(self.root,textvariable=self.var\_name,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=150,y=100,width=200)

txt\_email=Entry(self.root,textvariable=self.var\_email,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=150,y=140,width=200)

self.txt\_gender=ttk.Combobox(self.root,textvariable=self.var\_gender,values=("Select","Male","Female","Other"),font=("goudy old style",15,'bold'),state='readonly',justify=CENTER)

self.txt\_gender.place(x=150,y=180,width=200)

self.txt\_gender.current(0)

txt\_dob=Entry(self.root,textvariable=self.var\_dob,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=480,y=60,width=200)

#================Column 2============================

lbl\_dob=Label(self.root,text="D. O. B",font=("goudy old style",15,'bold'),bg='white').place(x=360,y=60)

lbl\_contact=Label(self.root,text="Contact",font=("goudy old style",15,'bold'),bg='white').place(x=360,y=100)

lbl\_admission=Label(self.root,text="Admission",font=("goudy old style",15,'bold'),bg='white').place(x=360,y=140)

lbl\_course=Label(self.root,text="Course",font=("goudy old style",15,'bold'),bg='white').place(x=360,y=180)

#=========entry fields========

self.course\_list=[]

#function\_call to update the list

self.fetch\_course()

txt\_dob=Entry(self.root,textvariable=self.var\_dob,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=480,y=60,width=200)

txt\_contact=Entry(self.root,textvariable=self.var\_contact,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=480,y=100,width=200)

txt\_admission=Entry(self.root,textvariable=self.var\_a\_date,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=480,y=140,width=200)

self.txt\_course=ttk.Combobox(self.root,textvariable=self.var\_course,values=self.course\_list,font=("goudy old style",15,'bold'),state='readonly',justify=CENTER)

self.txt\_course.place(x=480,y=180,width=200)

self.txt\_course.set("Select")

#==============Text Address================

self.txt\_address=Text(self.root,font=("goudy old style",15,'bold'),bg='lightyellow')

self.txt\_address.place(x=150,y=260,width=540,height=100)

#========Buttons======================

self.btn\_add=Button(self.root,text='Save',font=("goudy old style",15,'bold'),bg='#2196f3',fg="white",cursor="hand2",command=self.add)

self.btn\_add.place(x=150,y=400,width=110,height=40)

self.btn\_update=Button(self.root,text='Update',font=("goudy old style",15,'bold'),bg='#4caf50',fg="white",cursor="hand2",command=self.update)

self.btn\_update.place(x=270,y=400,width=110,height=40)

self.btn\_delete=Button(self.root,text='Delete',font=("goudy old style",15,'bold'),bg='#f44336',fg="white",cursor="hand2",command=self.delete)

self.btn\_delete.place(x=390,y=400,width=110,height=40)

self.btn\_clear=Button(self.root,text='Clear',font=("goudy old style",15,'bold'),bg='#607d8b',fg="white",cursor="hand2",command=self.clear)

self.btn\_clear.place(x=510,y=400,width=110,height=40)

#=================Search Panel=======

self.var\_search=StringVar()

lbl\_search\_roll=Label(self.root,text="Roll No.",font=("goudy old style",15,'bold'),bg="white").place(x=720,y=60)

txt\_search\_roll=Entry(self.root,textvariable=self.var\_search,font=("goudy old style",15,'bold'),bg='lightyellow').place(x=870,y=60,width=180)

btn\_search=Button(self.root,text='Search',font=("goudy old style",15,'bold'),bg='#03a9f4',fg="white",cursor="hand2",command=self.search).place(x=1070,y=60,width=120,height=28)

#==============Content==========

self.C\_Frame=Frame(self.root,bd=2,relief=RIDGE)

self.C\_Frame.place(x=720,y=100,width=470,height=340)

scrolly=Scrollbar(self.C\_Frame,orient=VERTICAL)

scrollx=Scrollbar(self.C\_Frame,orient=HORIZONTAL)

self.CourseTable=ttk.Treeview(self.C\_Frame,columns=("roll","name","email","gender","dob","contact","admission","course","state","city","pin","address"),xscrollcommand=scrollx.set,yscrollcommand=scrolly.set)

scrollx.pack(side=BOTTOM,fill=X)

scrolly.pack(side=RIGHT,fill=Y)

scrollx.config(command=self.CourseTable.xview)

scrolly.config(command=self.CourseTable.yview)

self.CourseTable.heading("roll",text="Roll No.")

self.CourseTable.heading("name",text="Name")

self.CourseTable.heading("email",text="Email")

self.CourseTable.heading("gender",text="Gender")

self.CourseTable.heading("dob",text="D. O. B")

self.CourseTable.heading("contact",text="Contact")

self.CourseTable.heading("admission",text="Admission")

self.CourseTable.heading("course",text="Course")

self.CourseTable.heading("state",text="State")

self.CourseTable.heading("city",text="City")

self.CourseTable.heading("pin",text="PIN")

self.CourseTable.heading("address",text="Address")

self.CourseTable["show"]='headings'

self.CourseTable.column("roll",width=100)

self.CourseTable.column("name",width=100)

self.CourseTable.column("email",width=100)

self.CourseTable.column("gender",width=100)

self.CourseTable.column("dob",width=150)

self.CourseTable.column("contact",width=150)

self.CourseTable.column("admission",width=150)

self.CourseTable.column("course",width=150)

self.CourseTable.column("state",width=150)

self.CourseTable.column("city",width=150)

self.CourseTable.column("pin",width=150)

self.CourseTable.column("address",width=150)

self.CourseTable.pack(fill=BOTH,expand=1)

self.CourseTable.bind("<ButtonRelease-1>",self.get\_data)

self.show()

#=====================================================================

def clear(self):

self.show()

self.var\_roll.set(""),

self.var\_name.set(""),

self.var\_email.set(""),

self.var\_gender.set("Select"),

self.var\_dob.set(""),

self.var\_contact.set(""),

self.var\_a\_date.set(""),

self.var\_course.set("Select"),

self.var\_state.set(""),

self.var\_city.set(""),

self.var\_pin.set(""),

self.txt\_address.delete("1.0",END)

self.txt\_roll.config(state=NORMAL)

self.var\_search.set("")

def delete(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_roll.get()=="":

messagebox.showerror("Error","Roll No is required",parent=self.root)

else:

cur.execute("select \* from student where roll=?",(self.var\_roll.get(),))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Please Select Student From The List First!",parent=self.root)

else:

op=messagebox.askyesno("Confirm","Do you really want to delete?",parent=self.root)

if op==True:

cur.execute("delete from student where roll=?",(self.var\_roll.get(),))

con.commit()

messagebox.showinfo("Delete","Student Deleted Successfully!",parent=self.root)

self.clear()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def get\_data(self,ev):

self.txt\_roll.config(state='readonly')

r=self.CourseTable.focus()

content=self.CourseTable.item(r)

row=content["values"]

#print(row)

self.var\_roll.set(row[0]),

self.var\_name.set(row[1]),

self.var\_email.set(row[2]),

self.var\_gender.set(row[3]),

self.var\_dob.set(row[4]),

self.var\_contact.set(row[5]),

self.var\_a\_date.set(row[6]),

self.var\_course.set(row[7]),

self.var\_state.set(row[8]),

self.var\_city.set(row[9]),

self.var\_pin.set(row[10]),

self.txt\_address.delete("1.0",END)

self.txt\_address.insert(END,row[11])

def add(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_roll.get()=="":

messagebox.showerror("Error","Roll No is required",parent=self.root)

else:

cur.execute("select \* from student where roll=?",(self.var\_roll.get(),))

row=cur.fetchone()

if row!=None:

messagebox.showerror("Error","Roll No already exists",parent=self.root)

else:

cur.execute("insert into student (roll,name,email,gender,dob,contact,admission,course,state,city,pin,address) values(?,?,?,?,?,?,?,?,?,?,?,?)",(

self.var\_roll.get(),

self.var\_name.get(),

self.var\_email.get(),

self.var\_gender.get(),

self.var\_dob.get(),

self.var\_contact.get(),

self.var\_a\_date.get(),

self.var\_course.get(),

self.var\_state.get(),

self.var\_city.get(),

self.var\_pin.get(),

self.txt\_address.get("1.0",END)

))

con.commit()

messagebox.showinfo("Success","Student Added Successfully!",parent=self.root)

self.show()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def update(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_roll.get()=="":

messagebox.showerror("Error","Roll No. is required",parent=self.root)

else:

cur.execute("select \* from student where roll=?",(self.var\_roll.get(),))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Select Student from list",parent=self.root)

else:

cur.execute("update student set name=?,email=?,gender=?,dob=?,contact=?,admission=?,course=?,state=?,city=?,pin=?,address=? where roll=?",(

self.var\_name.get(),

self.var\_email.get(),

self.var\_gender.get(),

self.var\_dob.get(),

self.var\_contact.get(),

self.var\_a\_date.get(),

self.var\_course.get(),

self.var\_state.get(),

self.var\_city.get(),

self.var\_pin.get(),

self.txt\_address.get("1.0",END),

self.var\_roll.get()

))

con.commit()

messagebox.showinfo("Success","Student Updated Successfully!",parent=self.root)

self.show()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def show(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select \* from student")

rows=cur.fetchall()

self.CourseTable.delete(\*self.CourseTable.get\_children())

for row in rows:

self.CourseTable.insert('',END,values=row)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def fetch\_course(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select name from course")

rows=cur.fetchall()

if len(rows)>0:

for row in rows:

self.course\_list.append(row[0])

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def search(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select \* from student where roll=?",(self.var\_search.get(),))

row=cur.fetchone()

if row!=None:

self.CourseTable.delete(\*self.CourseTable.get\_children())

self.CourseTable.insert('',END,values=row)

else:

messagebox.showerror("Error","No record found!",parent=self.root)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

obj=studentClass(root)

root.mainloop()

1. **Result.py**

from tkinter import\*

from PIL import Image,ImageTk

from tkinter import ttk,messagebox

import sqlite3

class resultClass:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Student Result Management System")

self.root.geometry("1200x480+80+170")

self.root.config(bg="white")

self.root.focus\_force()

#====title===

title=Label(self.root,text="Add Student Results",font=("goudy old style",20,"bold"),bg="orange",fg="#262626").place(x=10,y=15,width=1180,height=50)

#============Widgets===============

#=============Variables============

self.var\_roll=StringVar()

self.var\_name=StringVar()

self.var\_course=StringVar()

self.var\_marks=StringVar()

self.var\_full\_marks=StringVar()

self.roll\_list=[]

self.fetch\_roll()

lbl\_select=Label(self.root,text="Select Student",font=("goudy old style",20,"bold"),bg="white").place(x=50,y=100)

lbl\_name=Label(self.root,text="Name",font=("goudy old style",20,"bold"),bg="white").place(x=50,y=160)

lbl\_course=Label(self.root,text="Course",font=("goudy old style",20,"bold"),bg="white").place(x=50,y=220)

lbl\_marks\_ob=Label(self.root,text="Marks Obtained",font=("goudy old style",20,"bold"),bg="white").place(x=50,y=280)

lbl\_full\_marks=Label(self.root,text="Full Marks",font=("goudy old style",20,"bold"),bg="white").place(x=50,y=340)

self.txt\_student=ttk.Combobox(self.root,textvariable=self.var\_roll,values=self.roll\_list,font=("goudy old style",15,'bold'),state='readonly',justify=CENTER)

self.txt\_student.place(x=280,y=100,width=200)

self.txt\_student.set("Select")

btn\_search=Button(self.root,text='Search',font=("goudy old style",15,'bold'),bg='#03a9f4',fg="white",cursor="hand2",command=self.search).place(x=500,y=100,width=100,height=28)

txt\_name=Entry(self.root,textvariable=self.var\_name,font=("goudy old style",20,'bold'),bg='lightyellow',state='readonly').place(x=280,y=160,width=320)

txt\_course=Entry(self.root,textvariable=self.var\_course,font=("goudy old style",20,'bold'),bg='lightyellow',state='readonly').place(x=280,y=220,width=320)

txt\_marks=Entry(self.root,textvariable=self.var\_marks,font=("goudy old style",20,'bold'),bg='lightyellow').place(x=280,y=280,width=320)

txt\_full\_marks=Entry(self.root,textvariable=self.var\_full\_marks,font=("goudy old style",20,'bold'),bg='lightyellow').place(x=280,y=340,width=320)

#===========Buttons=========================

btn\_add=Button(self.root,text="Submit",font=("times new roman",15),bg="lightgreen",activebackground="lightgreen",cursor="hand2",command=self.add).place(x=300,y=420,width=120,height=35)

btn\_clear=Button(self.root,text="Clear",font=("times new roman",15),bg="lightgray",activebackground="lightgray",cursor="hand2",command=self.clear).place(x=430,y=420,width=120,height=35)

#===============Image===============

self.bg\_img=Image.open("images/result.jpg")

self.bg\_img=self.bg\_img.resize((500,300),Image.ANTIALIAS)

self.bg\_img=ImageTk.PhotoImage(self.bg\_img)

self.lbl\_bg=Label(self.root,image=self.bg\_img).place(x=650,y=100)

#======================================================

def fetch\_roll(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select s.roll from student s LEFT JOIN result r ON s.roll == r.roll WHERE r.roll IS NULL")

rows=cur.fetchall()

if len(rows)>0:

for row in rows:

self.roll\_list.append(row[0])

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def search(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

cur.execute("select name,course from student where roll=?",(self.var\_roll.get(),))

row=cur.fetchone()

if row!=None:

self.var\_name.set(row[0])

self.var\_course.set(row[1])

else:

messagebox.showerror("Error","No record found!",parent=self.root)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def add(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_name.get()=="":

messagebox.showerror("Error","Please First Search Student Record",parent=self.root)

else:

cur.execute("select \* from result where roll=? and course=?",(self.var\_roll.get(),self.var\_course.get()))

row=cur.fetchone()

if row!=None:

messagebox.showerror("Error","Result already present!",parent=self.root)

else:

per=(int(self.var\_marks.get())\*100)/int(self.var\_full\_marks.get())

cur.execute("insert into result (roll,name,course,marks\_ob,full\_marks,per) values(?,?,?,?,?,?)",(

self.var\_roll.get(),

self.var\_name.get(),

self.var\_course.get(),

self.var\_marks.get(),

self.var\_full\_marks.get(),

str(per)

))

con.commit()

messagebox.showinfo("Success","Result Added Successfully!",parent=self.root)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def clear(self):

self.var\_roll.set("Select")

self.var\_name.set("")

self.var\_course.set("")

self.var\_marks.set("")

self.var\_full\_marks.set("")

if \_\_name\_\_=="\_\_main\_\_":

root=Tk()

obj=resultClass(root)

root.mainloop()

1. **Report.py**

from tkinter import\*

from PIL import Image,ImageTk

from tkinter import ttk,messagebox

import sqlite3

class reportClass:

def \_\_init\_\_(self,root):

self.root=root

self.root.title("Student Result Management System")

self.root.geometry("1200x480+80+170")

self.root.config(bg="white")

self.root.focus\_force()

#====title===

title=Label(self.root,text="View Student Results",font=("goudy old style",20,"bold"),bg="orange",fg="#262626").place(x=10,y=15,width=1180,height=50)

#========================search============

self.var\_search=StringVar()

self.var\_id=""

lbl\_search=Label(self.root,text="Search By Roll No.",font=("goudy old style",20,"bold"),bg="white").place(x=280,y=100)

txt\_search=Entry(self.root,textvariable=self.var\_search,font=("goudy old style",20),bg="lightyellow").place(x=520,y=100,width=150)

btn\_search=Button(self.root,text='Search',font=("goudy old style",15,'bold'),bg='#03a9f4',fg="white",cursor="hand2",command=self.search).place(x=680,y=100,width=100,height=35)

btn\_clear=Button(self.root,text='Clear',font=("goudy old style",15,'bold'),bg='gray',fg="white",cursor="hand2",command=self.clear).place(x=800,y=100,width=100,height=35)

#====================Result\_labels============================

lbl\_roll=Label(self.root,text="Roll No.",font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE).place(x=150,y=230,width=150,height=50)

lbl\_name=Label(self.root,text="Name",font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE).place(x=300,y=230,width=150,height=50)

lbl\_course=Label(self.root,text="Course",font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE).place(x=450,y=230,width=150,height=50)

lbl\_marks=Label(self.root,text="Marks Obtained",font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE).place(x=600,y=230,width=150,height=50)

lbl\_full=Label(self.root,text="Total Marks",font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE).place(x=750,y=230,width=150,height=50)

lbl\_per=Label(self.root,text="Percentage",font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE).place(x=900,y=230,width=150,height=50)

self.roll=Label(self.root,font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE)

self.roll.place(x=150,y=280,width=150,height=50)

self.name=Label(self.root,font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE)

self.name.place(x=300,y=280,width=150,height=50)

self.course=Label(self.root,font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE)

self.course.place(x=450,y=280,width=150,height=50)

self.marks=Label(self.root,font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE)

self.marks.place(x=600,y=280,width=150,height=50)

self.full=Label(self.root,font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE)

self.full.place(x=750,y=280,width=150,height=50)

self.per=Label(self.root,font=("goudy old style",15,"bold"),bg="white",bd=2,relief=GROOVE)

self.per.place(x=900,y=280,width=150,height=50)

#======================Button Delete===================

btn\_delete=Button(self.root,text='Delete',font=("goudy old style",15,'bold'),bg='red',fg="white",cursor="hand2",command=self.delete).place(x=500,y=350,width=150,height=35)

#=================================================================================================================

def search(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_search.get()=="":

messagebox.showerror("Error","Roll No. should be required.",parent=self.root)

else:

cur.execute("select \* from result where roll=?",(self.var\_search.get(),))

row=cur.fetchone()

if row!=None:

self.var\_id=row[0]

self.roll.config(text=row[1])

self.name.config(text=row[2])

self.course.config(text=row[3])

self.marks.config(text=row[4])

self.full.config(text=row[5])

self.per.config(text=row[6])

else:

messagebox.showerror("Error","No record found!",parent=self.root)

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

def clear(self):

self.var\_id=""

self.roll.config(text="")

self.name.config(text="")

self.course.config(text="")

self.marks.config(text="")

self.full.config(text="")

self.per.config(text="")

self.var\_search.set("")

def delete(self):

con=sqlite3.connect(database="srms.db")

cur=con.cursor()

try:

if self.var\_id=="":

messagebox.showerror("Error","Search Student Result First",parent=self.root)

else:

cur.execute("select \* from result where rid=?",(self.var\_id,))

row=cur.fetchone()

if row==None:

messagebox.showerror("Error","Invalid Student Result!",parent=self.root)

else:

op=messagebox.askyesno("Confirm","Do you really want to delete?",parent=self.root)

if op==True:

cur.execute("delete from result where rid=?",(self.var\_id,))

con.commit()

messagebox.showinfo("Delete","Result Deleted Successfully!",parent=self.root)

self.clear()

except Exception as ex:

messagebox.showerror("Error",f"Error due to {str(ex)}")

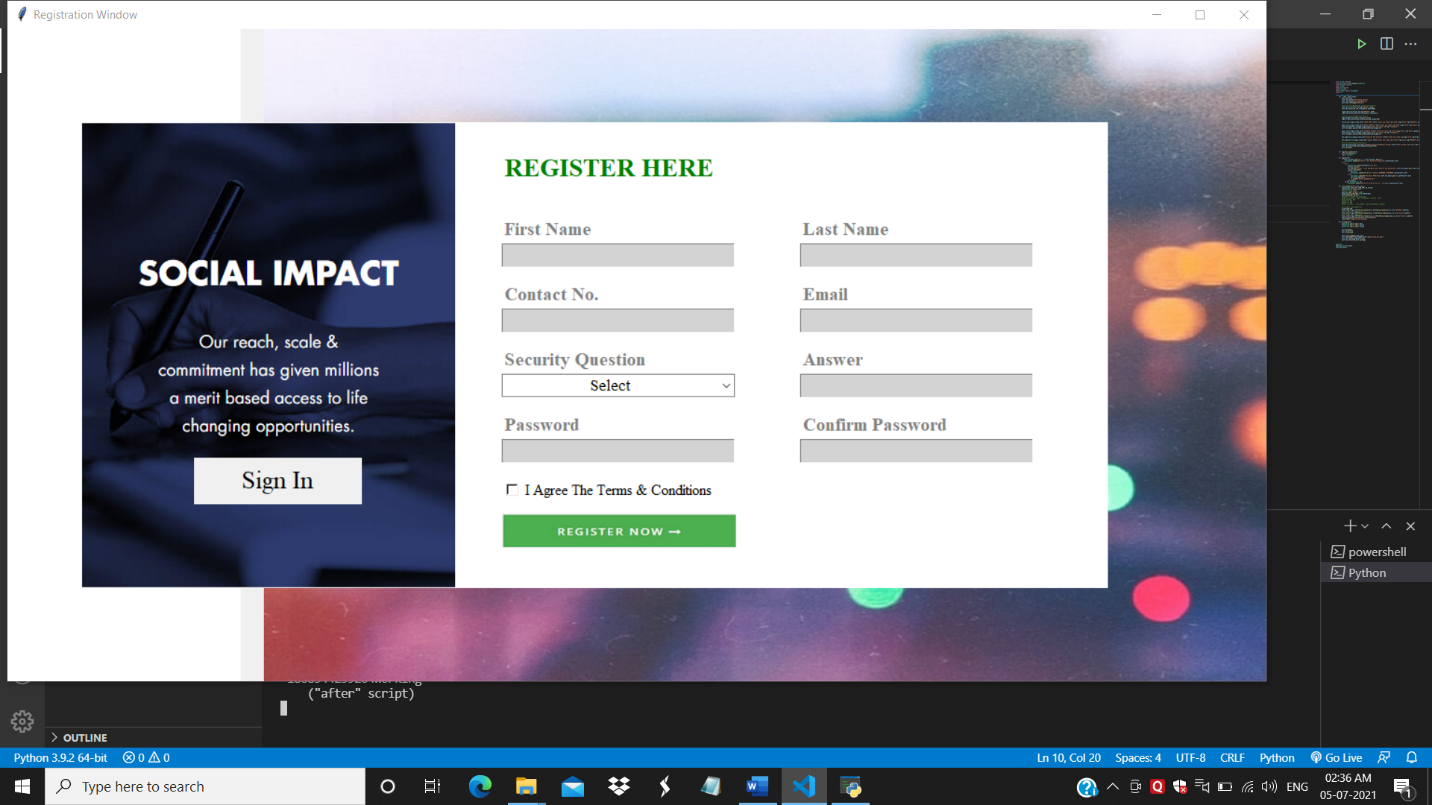
if \_\_name\_\_=="\_\_main\_\_":

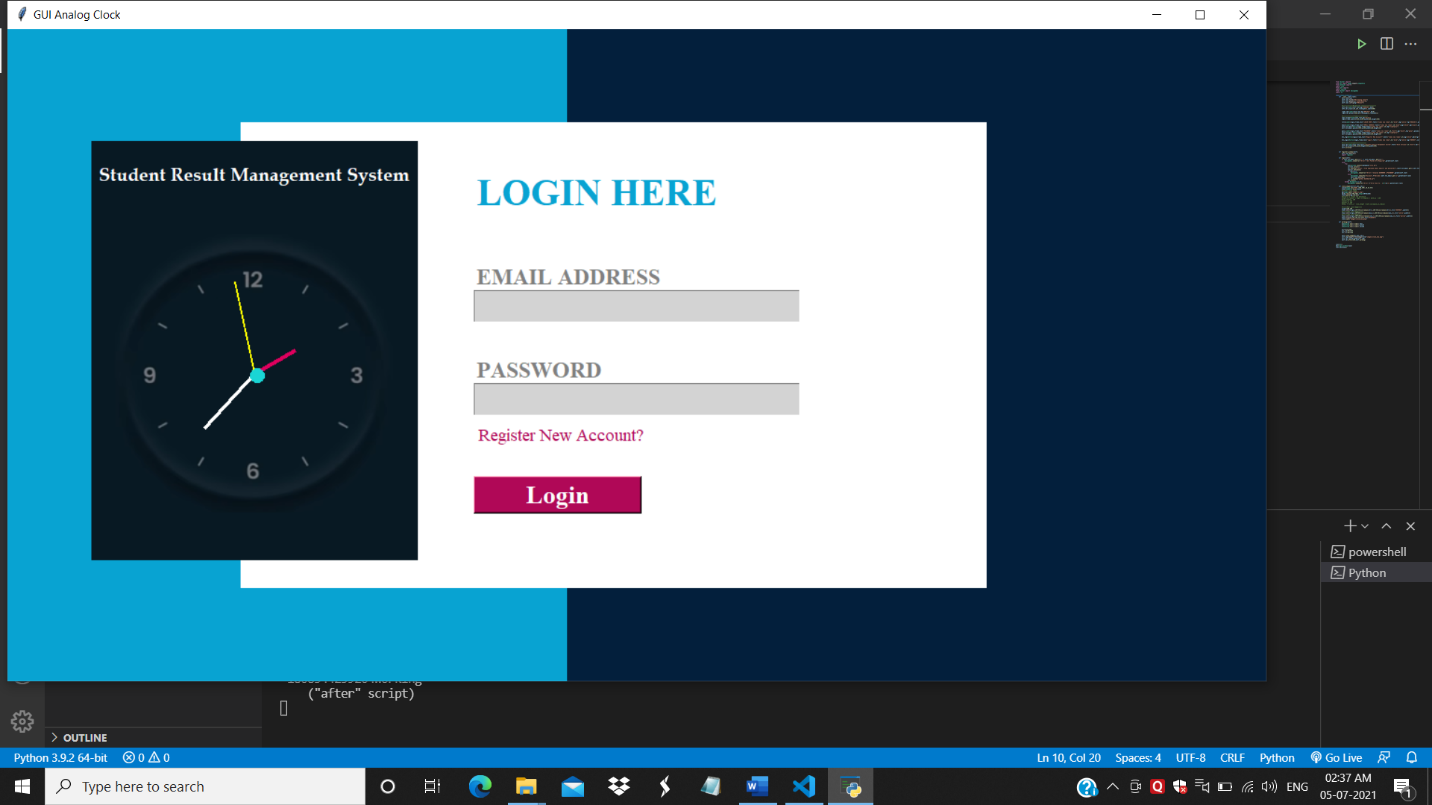
root=Tk()

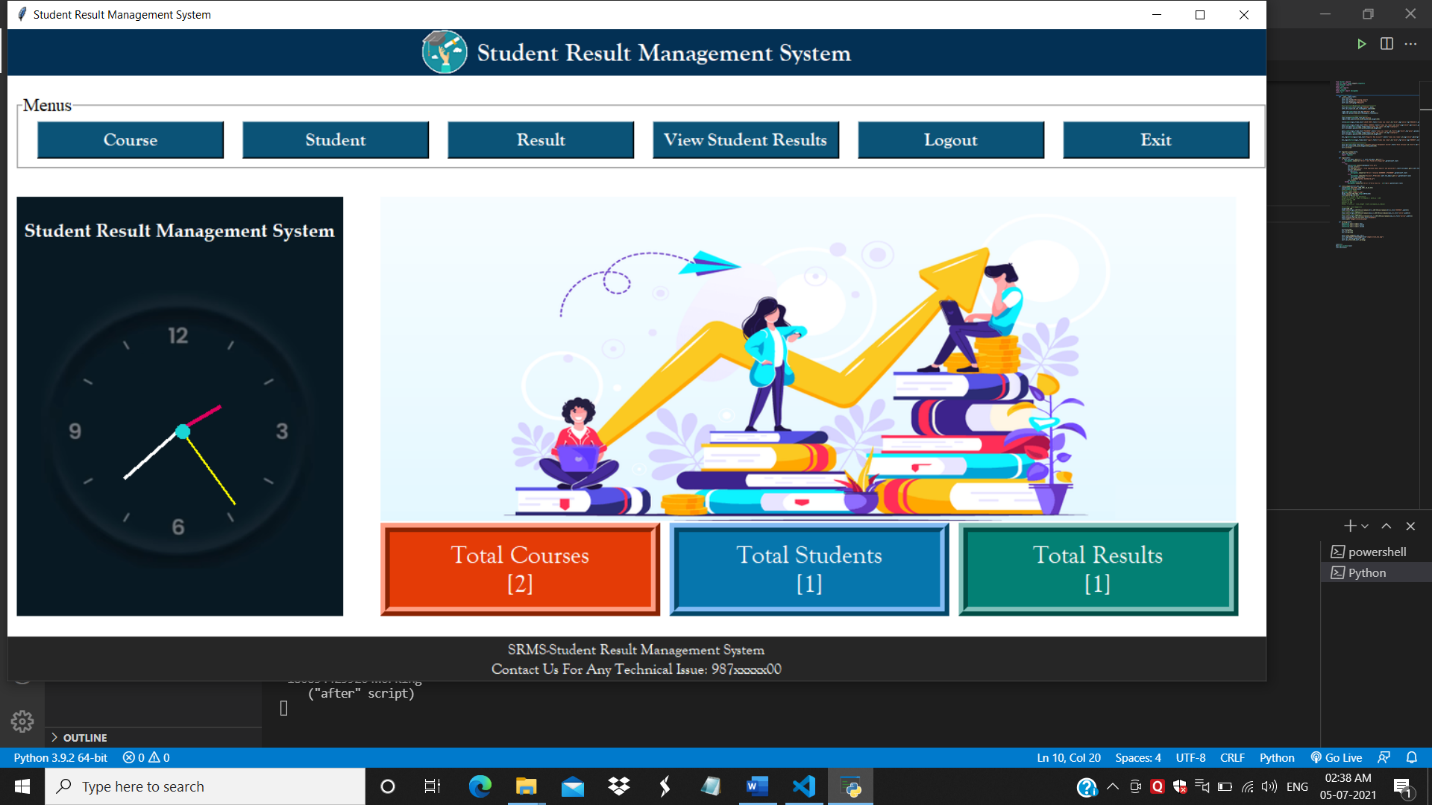
obj=reportClass(root)

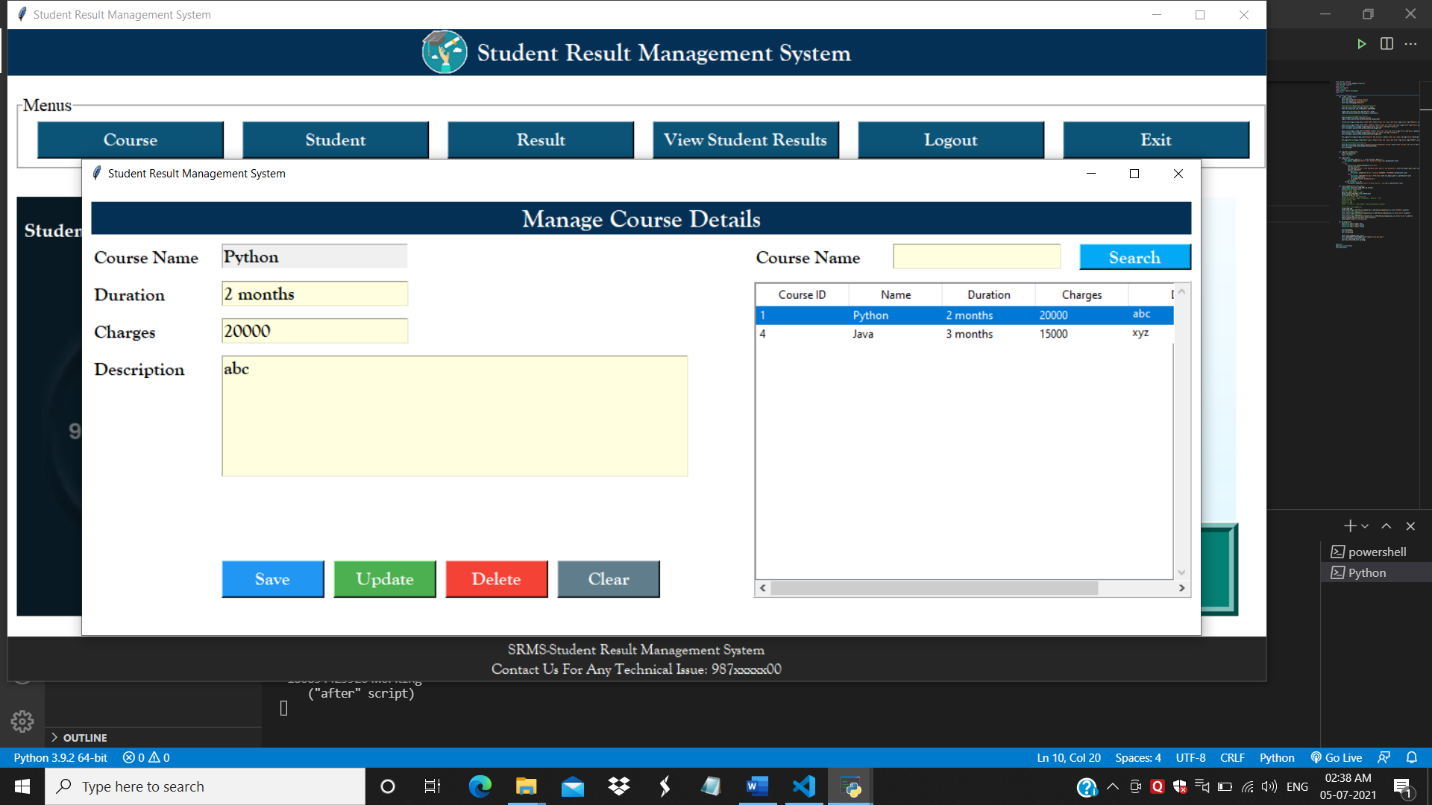
root.mainloop()

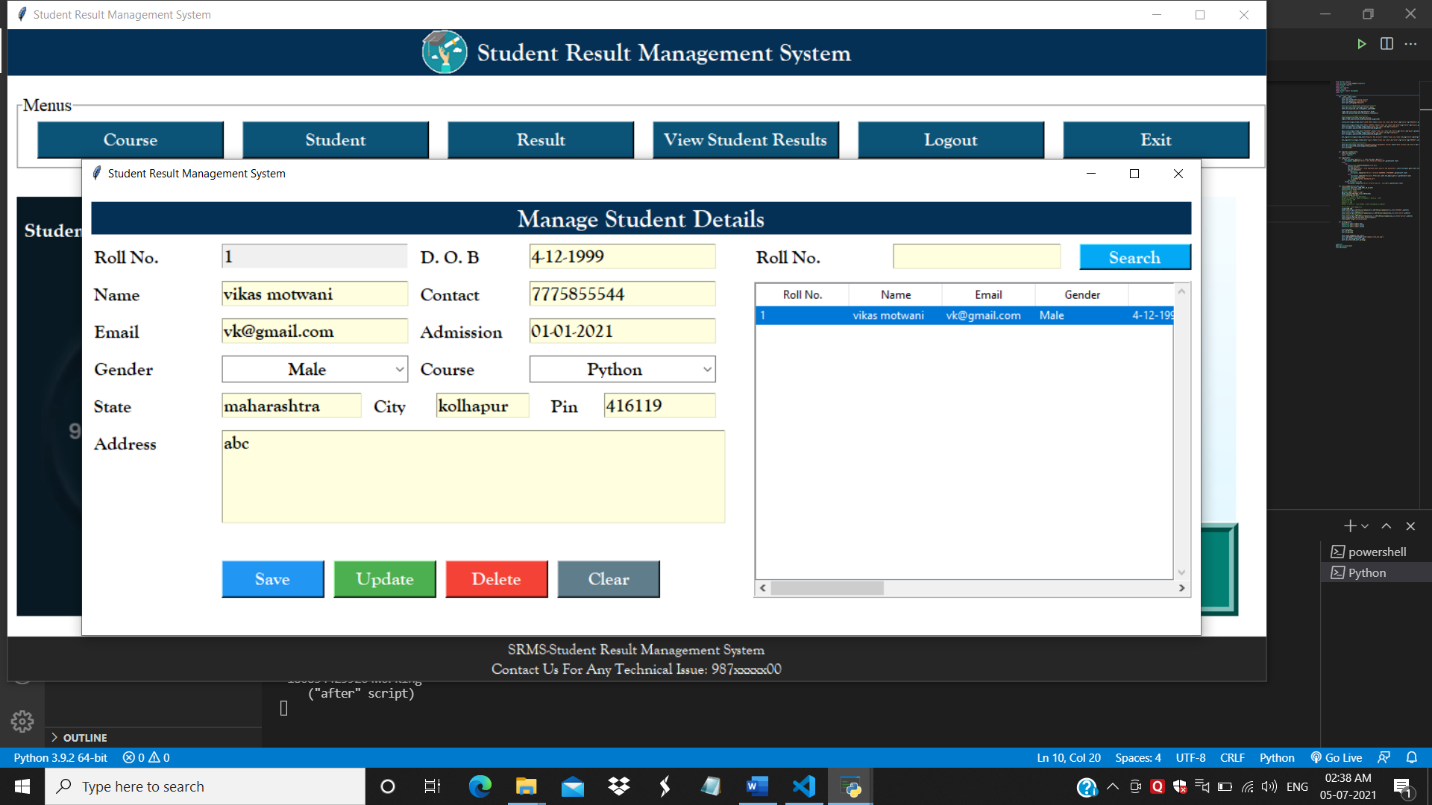
# OUTPUT SCREEN:

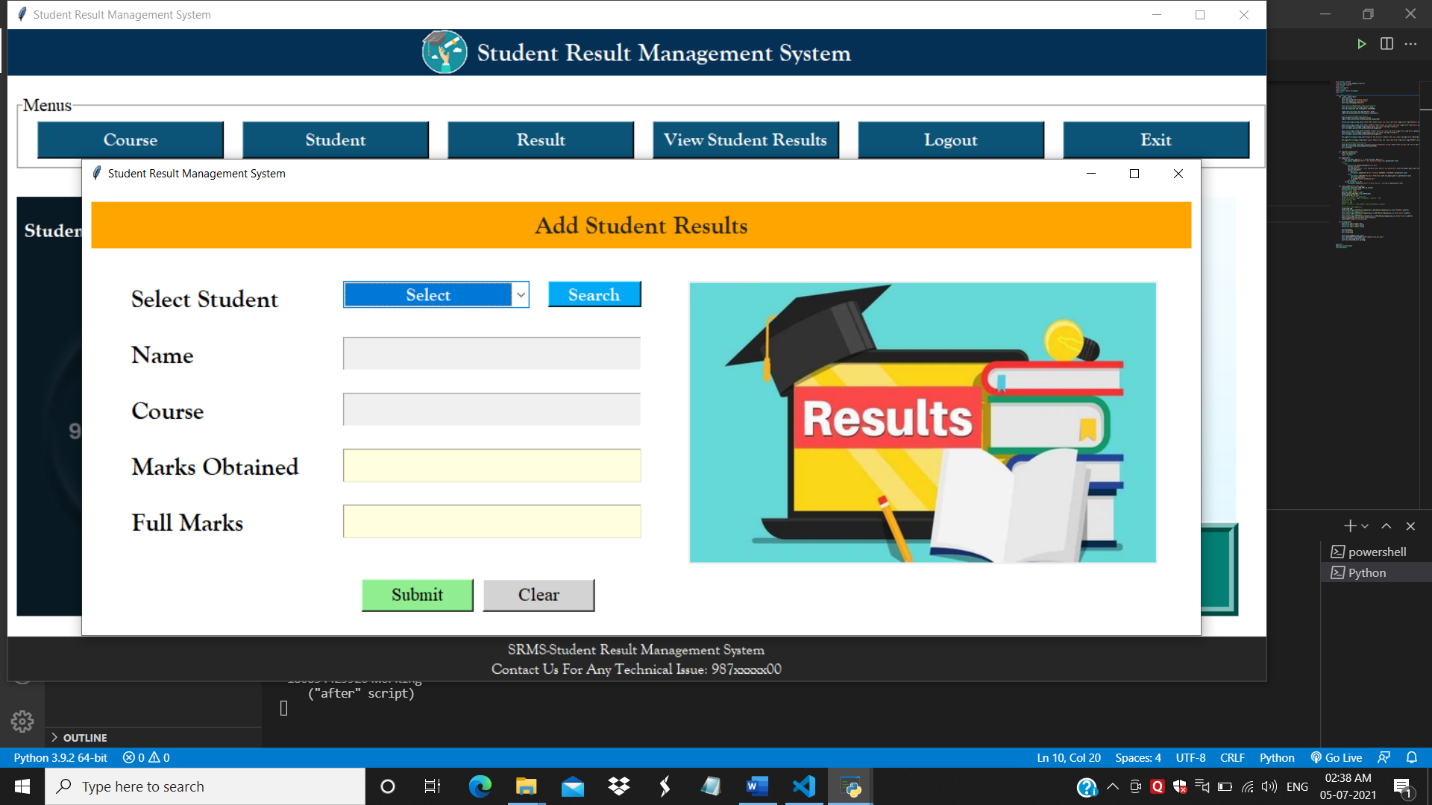


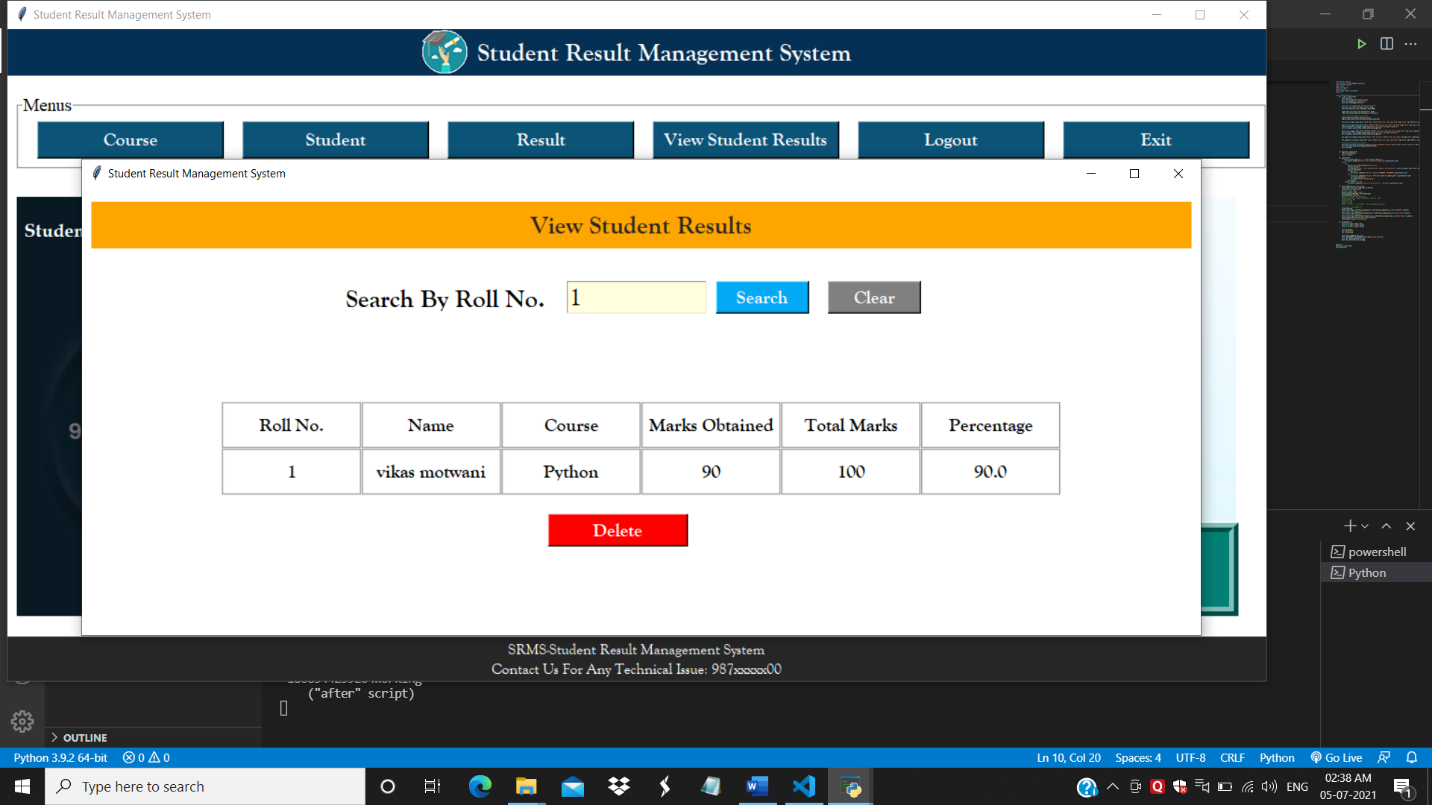


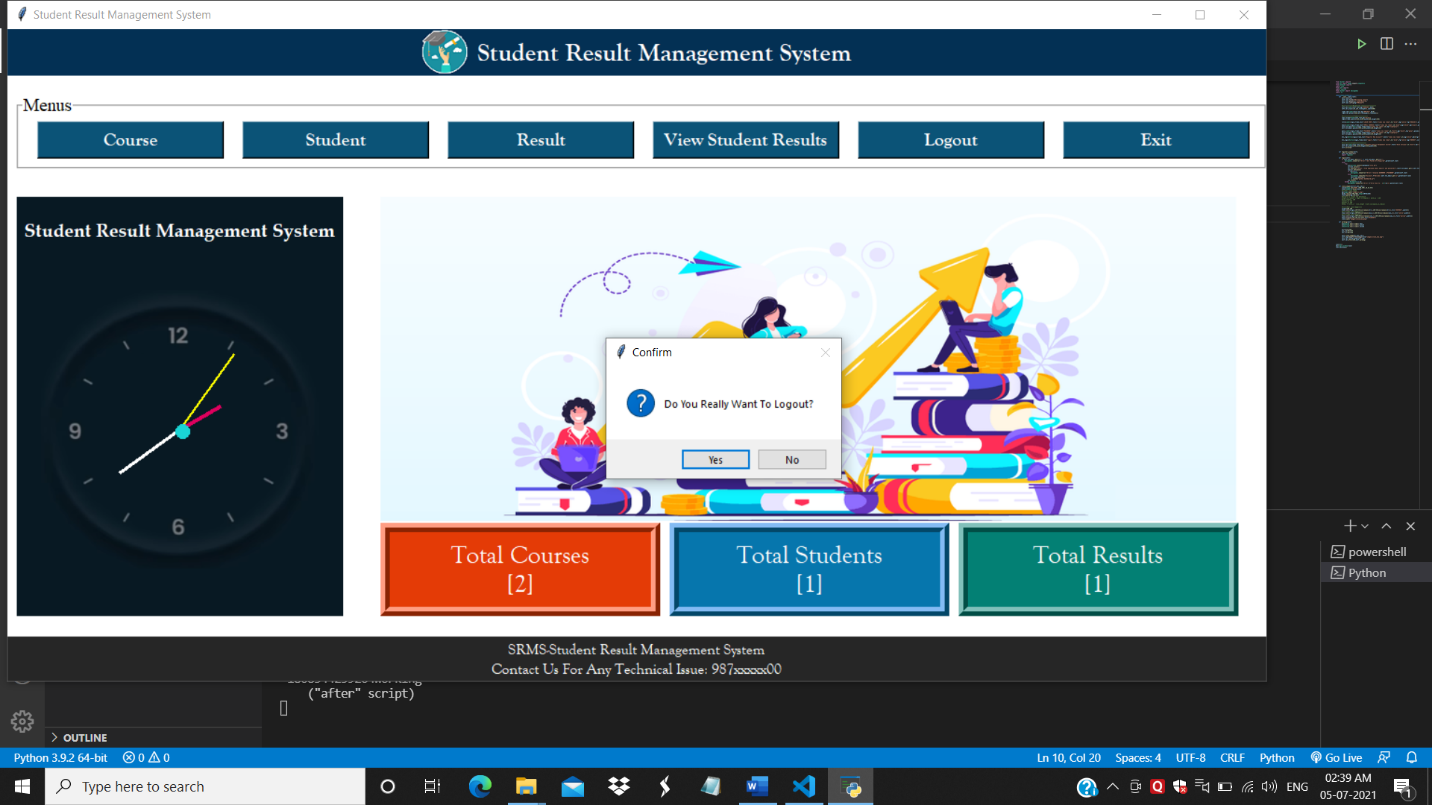


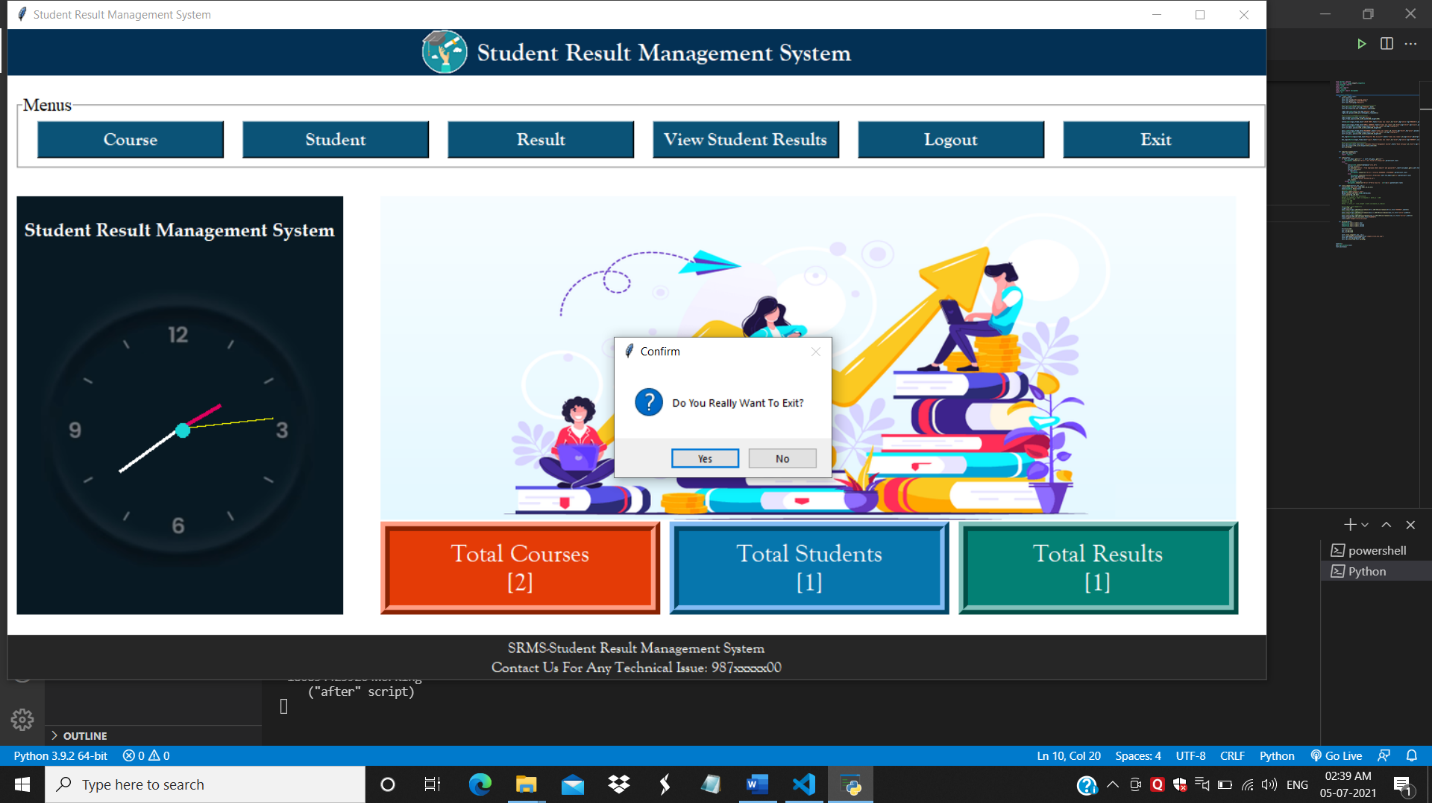












1. **CONCLUSION:**

On completion of the project we conclude the following conclusions:

Implementing the computerized system that stores the information about student and courses offered by the institution makes it easier and saves a lot of efforts as it is not done manually.

This minor project is a simple one but helps to solve lot of complex problem like maintaining the records of students but it also has the functionalities of deleting, updating and also displaying the results of the students.

1. **BIBLIOGRAPHY:**

The following websites were referred during the analysis and execution phase of the project :

<https://wordpress.org>

<https://www.sqlite.com>

<https://www.gnu.org/copyleft/gpl.html>

<https://es.wikipedia.org>

<https://w3schools.com>