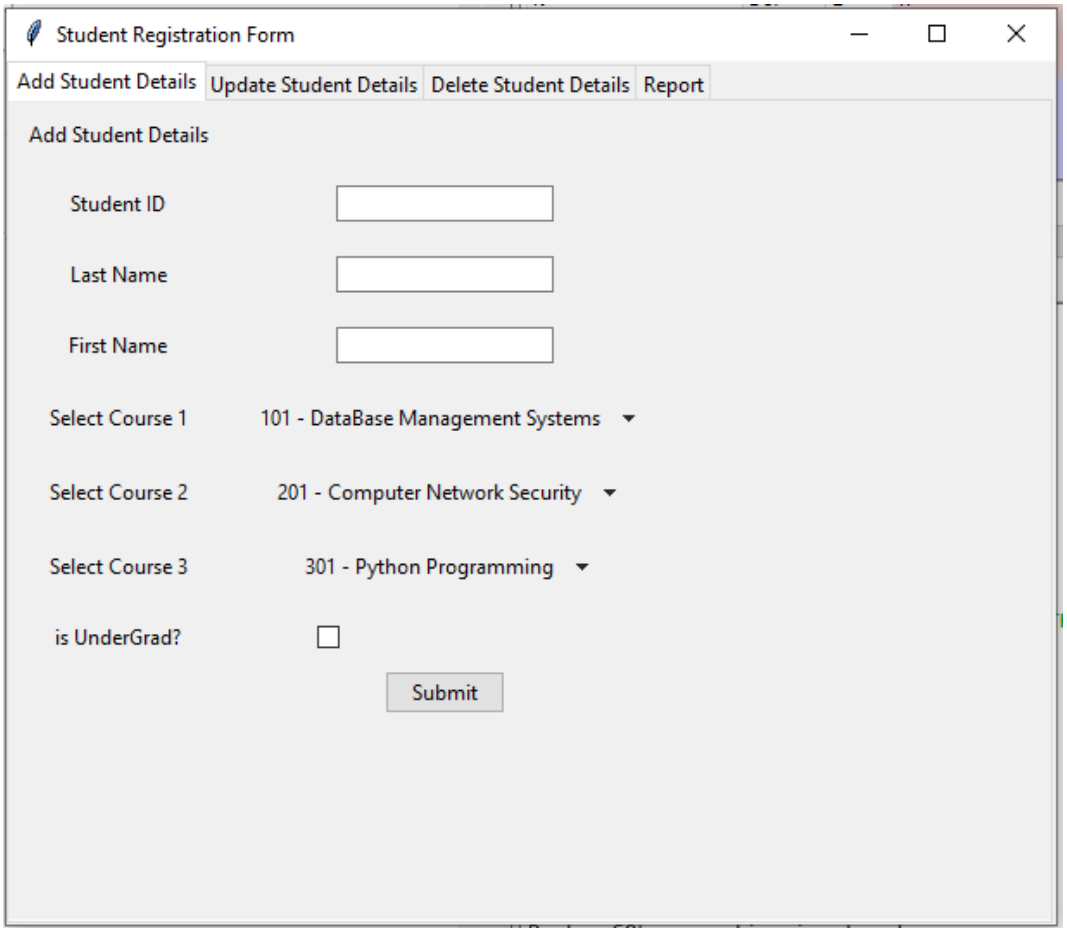
STUDENT REGISTRATION GUI APPLICATION

By Lakshmi Chaitanya Kakarla (865540)

Overview of the project:

In this project, I have developed a python application which will be useful for the academic staff and lecturers to monitor the student registration process. It also helps in finding the most registered course for a particular semester by which they can increase the class strength of that particular class or can introduce additional sections for that particular course. My application involves creating a graphical user interface which interacts with the PostgreSQL database for storing the student registration records. The Student Registration GUI application has four tabs namely Add Student Details, Update Student Details, Delete student Details and Report.

Add Student Details: This tab has the following fields that are essential for the registration process to add the details into the database.



Student ID – Data entry field for student ID

Last Name – Data entry field for last name of the student

First Name – Data entry field for the first name of the student

Select Course 1 – A dropdown menu to choose one course from the available three courses

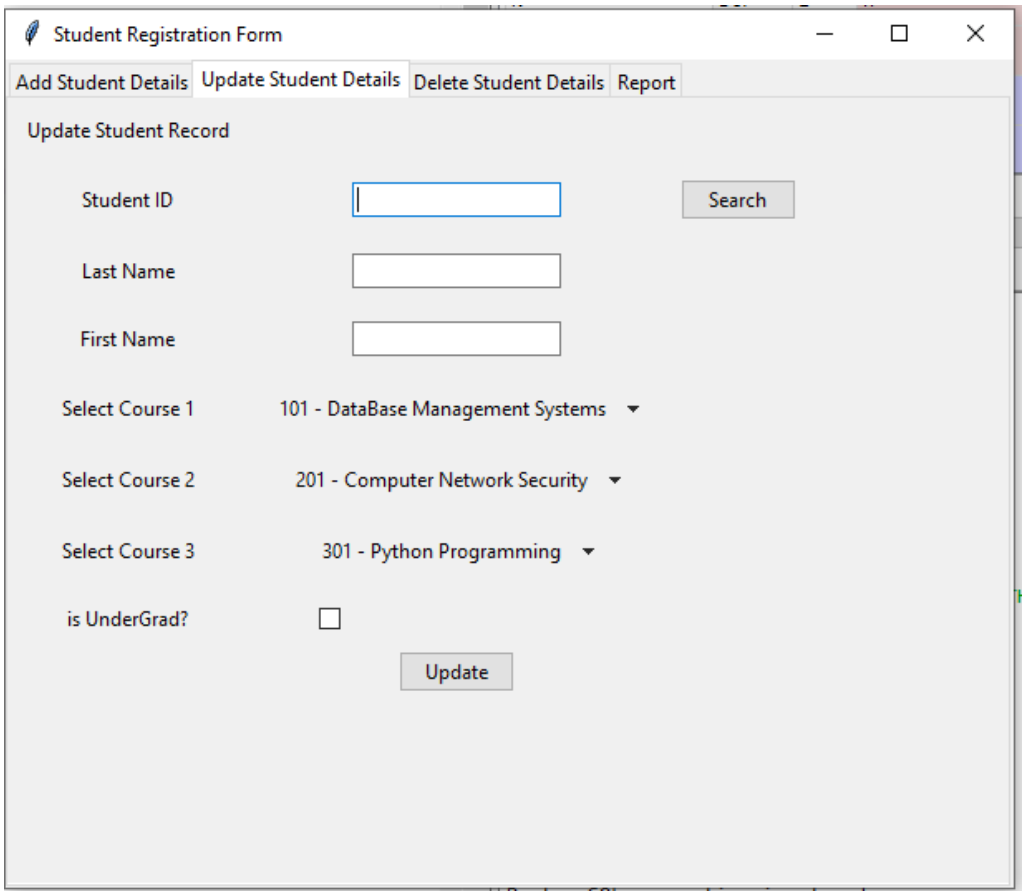
Select Course 2 – A dropdown menu to choose one course from the available three courses

Select Course 3 – A dropdown menu to choose one course from the available three courses

is UnderGrad? – A checkbox to identify whether a student is under graduate or graduate

Submit – A button to submit the above details

Update Student Details: This tab has the following fields that are essential to check whether the user entered student ID is already available in the database or not. If a particular student ID already exists in the database, then it will allow the user to update their details. The details can be updated for Last Name, First Name, Select Course1, Select Course 2, Select Course 3 and is UnderGrad fields. If the user entered Student ID does not exist in the database, then it will ask the user to go to the Add Student Details tab and insert the details.



Student ID – Data entry field for student ID

Search – A button to search the entered student ID in the database records

Last Name – Data entry field to update the last name of the student

First Name – Data entry field to update the first name of the student

Select Course 1 – A dropdown menu to update one course from the available three courses

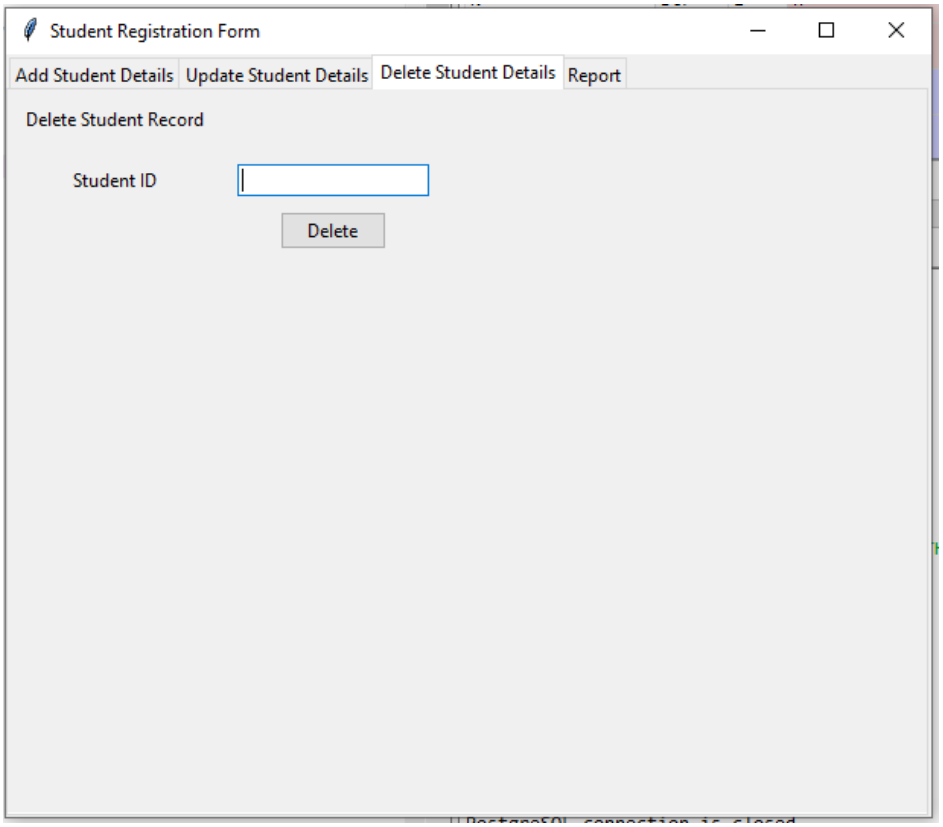
Select Course 2 – A dropdown menu to update one course from the available three courses

Select Course 3 – A dropdown menu to update one course from the available three courses

is UnderGrad? – A checkbox to update whether a student is under graduate or graduate

Update– A button to update the above details

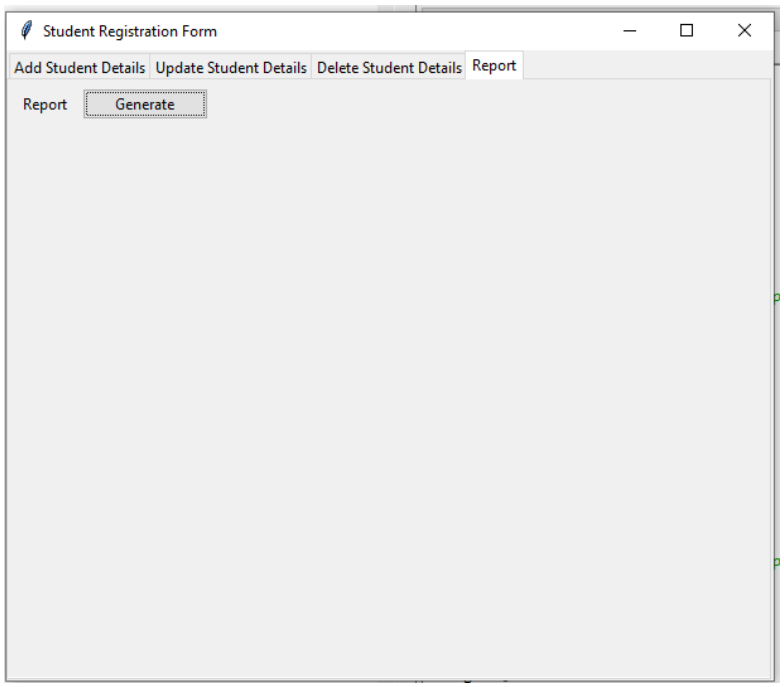
Delete Student Details: This tab has the following fields that are essential for the deletion of the student details in the database. If the user wants to delete a student details permanently from the database, then this tab will be useful. It allows the user to delete the student details who might have graduated or left the college.



Student ID – Data entry field for student ID

Delete – A delete button to delete the student records from the database using Student ID

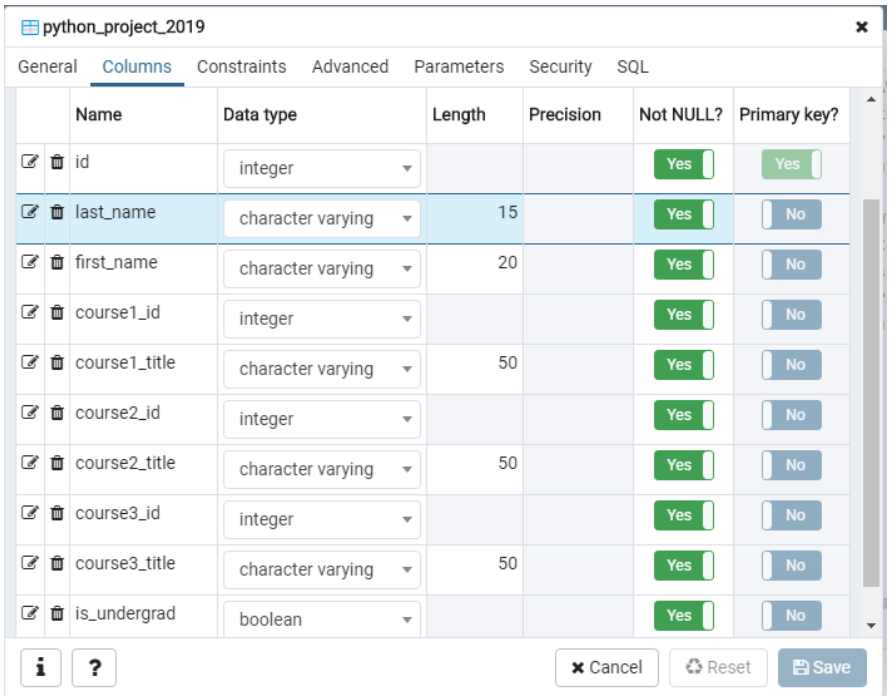
Report: This tab has the following fields that are essential for the generation of the student registration details report for each of the 9 available courses.



Generate – A button when clicked generates a report with the count of students for each of the nine available courses for a particular semester.

PostgreSQL Database Setup:

PostgreSQL database table: Using PostgreSQL PgAdmin tool I have created a database table called python_project_2019 with the below shown columns and constraints in the image.



Name	Data type	Length	Precision	Not NULL?	Primary key?
id	integer			Yes	Yes
last_name	character varying	15		Yes	No
first_name	character varying	20		Yes	No
course1_id	integer			Yes	No
course1_title	character varying	50		Yes	No
course2_id	integer			Yes	No
course2_title	character varying	50		Yes	No
course3_id	integer			Yes	No
course3_title	character varying	50		Yes	No
is_undergrad	boolean			Yes	No

I have inserted some sample data with 40 students already registered for various courses into the database. Below are the two screenshots of the sample data.

	id [PK] integer	last_name character varying (15)	first_name character varying (20)	course1_id integer	course1_title character varying (50)	course2_id integer	course2_title character varying (50)	course3_id integer	course3_title character varying (50)	is_undergrad boolean
1	103	Casey	Harriet	101	DataBase Management Syst...	201	Computer Network Security	301	Python Programming	true
2	122	Logan	Janet	102	Introduction to Data Analytics	202	Microcomputer Applications	302	R Programming	false
3	123	Hagen	Greg	103	Data Mining	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	true
4	139	Carroll	Pat	103	Data Mining	202	Microcomputer Applications	302	R Programming	true
5	148	Wolf	Bee	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	true
6	167	Krumple	Scott	102	Introduction to Data Analytics	202	Microcomputer Applications	303	AWS and Cloud Computing	false
7	171	Harvey	Elliot	101	DataBase Management Syst...	203	Introduction to Decision Sup...	302	R Programming	true
8	181	Zygote	Carrie	102	Introduction to Data Analytics	201	Computer Network Security	303	AWS and Cloud Computing	true
9	194	Loftus	Abner	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	true
10	251	Grainger	John	102	Introduction to Data Analytics	201	Computer Network Security	302	R Programming	true
11	321	Jones	Garrett	103	Data Mining	201	Computer Network Security	302	R Programming	false
12	156	Davis	John	102	Introduction to Data Analytics	202	Microcomputer Applications	303	AWS and Cloud Computing	false
13	1020	Carney	Logan	101	DataBase Management Syst...	203	Introduction to Decision Sup...	302	R Programming	true
14	149	Krepps	Owen	103	Data Mining	201	Computer Network Security	301	Python Programming	false
15	1183	Collins	Nathan	101	DataBase Management Syst...	203	Introduction to Decision Sup...	302	R Programming	true
16	601	Devlin	Sharron	101	DataBase Management Syst...	201	Computer Network Security	301	Python Programming	true
17	218	Scott	Travis	102	Introduction to Data Analytics	201	Computer Network Security	302	R Programming	false
18	324	David	Ashley	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	true
19	113	Roberts	Juliet	102	Introduction to Data Analytics	201	Computer Network Security	303	AWS and Cloud Computing	false
20	199	Cameron	Austin	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	true
21	1203	Stone	Racheal	101	DataBase Management Syst...	203	Introduction to Decision Sup...	302	R Programming	true
22	176	Pollard	Jessica	103	Data Mining	202	Microcomputer Applications	301	Python Programming	false
23	1021	Carney	Ashley	101	DataBase Management Syst...	203	Introduction to Decision Sup...	302	R Programming	true
24	2861	Bates	Katherine	103	Data Mining	201	Computer Network Security	301	Python Programming	true
25	391	Blinn	Ciara	102	Introduction to Data Analytics	202	Microcomputer Applications	303	AWS and Cloud Computing	false
26	351	Evans	Laraun	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	false
27	1107	Witt	Jonah	103	Data Mining	203	Introduction to Decision Sup...	302	R Programming	true
28	1012	Tate	Josh	101	DataBase Management Syst...	201	Computer Network Security	303	AWS and Cloud Computing	true
29	234	Galderisi	Matt	102	Introduction to Data Analytics	202	Microcomputer Applications	302	R Programming	false
30	1421	Miley	Lydia	101	DataBase Management Syst...	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
31	105	Collins	Hannah	103	Data Mining	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
32	174	Kimberly	Nick	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	true
33	1528	Nolan	Sean	101	DataBase Management Syst...	202	Microcomputer Applications	302	R Programming	true
34	2317	Barone	Mai	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false
35	219	Wiggin	Micheal	102	Introduction to Data Analytics	201	Computer Network Security	302	R Programming	false
36	1453	Barone	Steven	103	Data Mining	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
37	705	Devlin	Shannon	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false
38	874	Tea	Dalton	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	true
39	1322	Gramz	Luke	101	DataBase Management Syst...	202	Microcomputer Applications	302	R Programming	true
40	652	Auth	Dylan	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false

<u>Python Application Project Files</u>: The application project has three python(.py) files namely main.py, database.py and student.py.

Main.py – This is the main file which consists of graphical user interface elements for performing student registration crud (Create, Read, Update and Delete) operations and generating a bar plot report. This file imports database.py and student.py files for the student registration operations.

Database.py – This file consists of all database operational methods like database connection, database disconnection, database record query, database record insertion, database record update, database record delete.

Student.py – This file consists of student registration class with the properties studentId, lastName, firstName, course1Id, course1Title, course2Id, course2Title, course3Id, course3Title, isUnderGrad and a helper method toString()

Python Application Project Code:

Main.py code

```
#The Tkinter module is the standard Python interface to the Tk GUI toolkit.

#It imports objects in Tinkter into the current namespace and renames it locally as 'tk' to save you typing long.

import tkinter as tk

#to import tabs functionality in the GUI

from tkinter import ttk

#to import all form elements from the tabular tkinter like checkbox buttons, option menus, entry fields.

from tkinter import *

#importing dataframe from pandas

from pandas import DataFrame

#importing matplot library pyplot for plotting the graphs

import matplotlib.pyplot as plt

#importing figurecanvas for tkinter

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

#For displaying the y-axis tick locators

from matplotlib.ticker import MaxNLocator


# importing database module from database.py

import database

#importing student registration class from student.py

from student import StudentRegistration
```

```
# intializing the window
window = tk.Tk()
# Adding the title to the window
window.title("Student Registration Form")
# configuring size of the window
window.geometry('850x500')
#Create Tab Control (intialization of tab functionality inside a window using notebook)
TAB_CONTROL = ttk.Notebook(window)
# Creating a Tab1
TAB1 = ttk.Frame(TAB_CONTROL)
# Adding the Tab1 title
TAB_CONTROL.add(TAB1, text='Add Student Details')
#Creating a Tab2
TAB2 = ttk.Frame(TAB_CONTROL)
# Adding the Tab2 title
TAB_CONTROL.add(TAB2, text='Update Student Details')

# Creating a Tab3
TAB3 = ttk.Frame(TAB_CONTROL)
# Adding the Tab3 title
TAB_CONTROL.add(TAB3, text='Delete Student Details')

#Creating a Tab4
TAB4 = ttk.Frame(TAB_CONTROL)
# Adding the Tab4 title
TAB_CONTROL.add(TAB4, text='Report')
# Pack to make the tabs visible on the window
TAB_CONTROL.pack(expand=1, fill="both")
```

#Add Student Details code

```
#adding Tab1 Heading

ttk.Label(TAB1, text="Add Student Details").grid(column=0, row=0, padx=10, pady=10)


# defining string variables for the user input entries and status label

studentIdVar=StringVar()

lNameVar = StringVar()

fNameVar = StringVar()

isUnderGradCheckVar = IntVar()

statusVar_insert= StringVar()


#creating a label and user entry field for Student ID

ttk.Label(TAB1, text = "Student ID").grid(column=0, row=1, padx=10, pady=10) # this is placed in 0 1

ttk.Entry(TAB1, textvariable=studentIdVar).grid(column=1, row=1, padx=10, pady=10) # this is placed in 1 1


#creating a label and user entry field for Last Name

ttk.Label(TAB1, text = "Last Name").grid(column=0, row=2, padx=10, pady=10) # this is placed in 0 2

ttk.Entry(TAB1, textvariable=lNameVar).grid(column=1, row=2, padx=10, pady=10) # this is placed in 1 2


#creating a label and user entry field for First Name

ttk.Label(TAB1, text = "First Name").grid(column=0, row=3, padx=10, pady=10) # this is placed in 0 3

ttk.Entry(TAB1, textvariable=fNameVar).grid(column=1, row=3, padx=10, pady=10) # this is placed in 1 3
```

```
#creating array for courseGroupList1
courseGroupList1 = [
"101 - DataBase Management Systems",
"102 - Introduction to Data Analytics",
"103 - Data Mining"
]

#Initializing a variable called courseGroup1 for holding the selected choice from the optionmenu
courseGroup1 = tk.StringVar(TAB1)
#setting default value in the optionmenu
courseGroup1.set(courseGroupList1[0])
# creating a label for optionmenu
ttk.Label(TAB1, text="Select Course 1").grid(column=0, row=4, padx=10, pady=10)
#Initializing optionmenu for the courseGroup1
ttk.OptionMenu(TAB1,courseGroup1, courseGroupList1[0], *courseGroupList1).grid(column=1,
row=4, padx=10, pady=10)

#creating array for courseGroupList2
courseGroupList2 = [
"201 - Computer Network Security",
"202 - Microcomputer Applications",
"203 - Introduction to Decision Support Systems"
]

#Initializing a variable called courseGroup2 for holding the selected choice from the optionmenu
courseGroup2 = tk.StringVar(TAB1)
#setting default value in the optionmenu
courseGroup2.set(courseGroupList2[0])
# creating a label for optionmenu
```

```
ttk.Label(TAB1, text="Select Course 2").grid(column=0, row=5, padx=10, pady=10)
#Initializing optionmenu for the courseGroup2
ttk.OptionMenu(TAB1,courseGroup2, courseGroupList2[0], *courseGroupList2).grid(column=1, row=5, padx=10, pady=10)

#creating array for courseGroupList3
courseGroupList3 = [
"301 - Python Programming",
"302 - R Programming",
"303 - AWS and Cloud Computing"
]

#Initializing a variable called courseGroup3 for holding the selected choice from the optionmenu
courseGroup3 = tk.StringVar(TAB1)
#setting default value in the optionmenu
courseGroup3.set(courseGroupList3[0])
# creating a label for optionmenu
ttk.Label(TAB1, text="Select Course 3").grid(column=0, row=6, padx=10, pady=10)
#Initializing optionmenu for the courseGroup3
ttk.OptionMenu(TAB1,courseGroup3, courseGroupList3[0], *courseGroupList3).grid(column=1, row=6, padx=10, pady=10)


#creating a label and user check button field for is UnderGrad field
ttk.Label(TAB1, text="is UnderGrad?").grid(column=0, row=7, padx=10, pady=10)
#For defaulting check box unchecked (to make it checked pass value=1 in IntVar())
# 'Checkbutton' is used to create the check buttons
ttk.Checkbutton(TAB1, text = "", variable = isUnderGradCheckVar).grid(columnspan=2, row=7)
```

```
# This method is to submit user entered values in the Postgres database using a StudentRegistration class
def submitCallback():
    #creating an object called registrationObj for the StudentRegistration class
    registrationObj = StudentRegistration()
    #Assigning user entered studentID value to the StudentID of the registrationObj
    registrationObj.studentId = studentIdVar.get()
    #Assigning user entered lastName value to the lastName of the registrationObj
    registrationObj.lastName = lNameVar.get()
    #Assigning user entered firstName value to the firstName of the registrationObj
    registrationObj.firstName = fNameVar.get()
    #Splitting courseGroup1 user selected option and Assigning course1Id & course1Title value to the course1Id and course1Title of the registrationObj
    registrationObj.course1Id = courseGroup1.get().split('-')[0]
    registrationObj.course1Title = courseGroup1.get().split('-')[1].lstrip()
    #Splitting courseGroup2 user selected option and Assigning course2Id & course2Title value to the course2Id and course2Title of the registrationObj
    registrationObj.course2Id = courseGroup2.get().split('-')[0]
    registrationObj.course2Title = courseGroup2.get().split('-')[1].lstrip()
    #Splitting courseGroup3 user selected option and Assigning course3Id & course3Title value to the course3Id and course3Title of the registrationObj
    registrationObj.course3Id = courseGroup3.get().split('-')[0]
    registrationObj.course3Title = courseGroup3.get().split('-')[1].lstrip()
    #If the user checks the check box assign 'true' value to isUnderGrad of the registrationObj and if not checked assign it to 'false'
    if(isUnderGradCheckVar.get() == 1):
        registrationObj.isUnderGrad = 'true'
    else:
        registrationObj.isUnderGrad = 'false'
```

```
    print(registrationObj.toString())


    #Creating a database connection and storing it in a dbConnection variable

    dbConnection = database.connection();

    #quering the database records with the studentID and storing query results in the 'records' variable

    records = database.recordQuery(dbConnection, "id", registrationObj.studentId)

    #If the records length is greater than zero it gives a status alert saying that the studentID already registered

    if(len(records) > 0):

        print("Student Id: "+registrationObj.studentId+" already registered, to update registration please use update tab")

        statusVar_insert.set("Student Id: "+registrationObj.studentId+" already registered, to update registration please use update tab")

    # If the above condition is not met, we are inserting the record in the database by using the above dbConnection variable and registrationObj

    else:

        status = database.recordInsertion(dbConnection, registrationObj)

        #If the record insertion db operation returns a successful status, a status alert message will display a success message if not it displays error message

        if(status == True):

            statusVar_insert.set("Successfully added student id: "+registrationObj.studentId)

        else:

            statusVar_insert.set("Error in adding details of student id: "+registrationObj.studentId)

    # Once all the above db operations are completed, we are closing the database connection

    database.disConnection(dbConnection)

# submit button

ttk.Button(TAB1, text="Submit", width=10, command=submitCallback).grid(column=1, row=10)

#label for displaying status messages

ttk.Label(TAB1, textvariable=statusVar_insert).grid(column=1, row=11, ipadx=10)
```

```
# ----------------------------------------------------------------------------------------------------------------
---------------------------------
```

#Update Student Details code

```
#adding Tab2 Heading
ttk.Label(TAB2, text="Update Student Record").grid(column=0, row=0, padx=10, pady=10)


# defining string variables for the user input entries and status label
studentIdVar_update=StringVar()
lNameVar_update = StringVar()
fNameVar_update = StringVar()
isUnderGradCheckVar_update = IntVar()
statusVar_update= StringVar()


#searchCallback method will search for the studentID entered by the user in the database records
and populates user entry fields if the record exists
def searchCallback():
    #creating a registration object of StudentRegistration
    registrationObj = StudentRegistration()
    #Storing studentID value entered by the user in the studentID of the registrationObj
    registrationObj.studentId = studentIdVar_update.get()
    #creating a database connection and storing in a db connection variable
    dbConnection = database.connection();
    #quering the database records with the studentID and storing query results in the 'records'
variable
    records = database.recordQuery(dbConnection, "id", registrationObj.studentId)


    #If the records length is 1 then the record data will be asssigned to the registrationObj
    if(len(records) > 0 and len(records) < 2):
        record = records[0]
```

```
#assigning record [0] which is a studentId value to studentId of the registrationObj
registrationObj.studentId = str(record[0])
#assigning record [1] which is a lastName value to lastName of the registrationObj
registrationObj.lastName = str(record[1])
#assigning record [2] which is a firstName value to firstName of the registrationObj
registrationObj.firstName = str(record[2])
#assigning record [3] which is a course1Id value to course1Id of the registrationObj
registrationObj.course1Id = str(record[3])
#assigning record[4] which is a course1Title value to course1Title of the registrationObj
registrationObj.course1Title = str(record[4])
#assigning record[5] which is a course2Id value to course2Id of the registrationObj
registrationObj.course2Id = str(record[5])
#assigning record[6] which is a course2Title value to course2Title of the registrationObj
registrationObj.course2Title = str(record[6])
#assigning record[7] which is a course3Id value to course3Id of the registrationObj
registrationObj.course3Id = str(record[7])
#assigning record[8] which is a course3Title value to course3Title of the registrationObj
registrationObj.course3Title = str(record[8])
#assigning record[9] which is a isUnderGrad boolean value to isUnderGrad of the registrationObj
registrationObj.isUnderGrad = str(record[9])

# Assigning above all collected record values i.e., registartionObj to the user entry fields
lNameVar_update.set(registrationObj.lastName)
fNameVar_update.set(registrationObj.firstName)

#identifying the courseId and assigning the respective courseGroup value to the optionmenu
if(registrationObj.course1Id == "101"):
    courseGroup1_update.set(courseGroupList1[0])
```

```
    elif(registrationObj.course1Id == "102"):
        courseGroup1_update.set(courseGroupList1[1])
    else:
        courseGroup1_update.set(courseGroupList1[2])

    if(registrationObj.course2Id == "201"):
        courseGroup2_update.set(courseGroupList2[0])
    elif(registrationObj.course2Id == "202"):
        courseGroup2_update.set(courseGroupList2[1])
    else:
        courseGroup2_update.set(courseGroupList2[2])

    if(registrationObj.course3Id == "301"):
        courseGroup3_update.set(courseGroupList3[0])
    elif(registrationObj.course3Id == "302"):
        courseGroup3_update.set(courseGroupList3[1])
    else:
        courseGroup3_update.set(courseGroupList3[2])

    #identifying isUnderGrad value and setting respective values 1 or 0 to the checkbox
    if(registrationObj.isUnderGrad == "true" or registrationObj.isUnderGrad == "True"):
        isUnderGradCheckVar_update.set(1)
    else:
        isUnderGradCheckVar_update.set(0)

#If the above condition is not met, a status message studentId not registered will be displayed
else:
    print("Student Id: "+registrationObj.studentId+" not registered, to enroll please use Add tab")
```

```
    statusVar_update.set("Student Id: "+registrationObj.studentId+" not registered, to enroll please use Add tab")
  # closing the database connection
  database.disConnection(dbConnection);

#creating a label and user entry field for Student ID
ttk.Label(TAB2, text = "Student ID").grid(column=0, row=1, padx=10, pady=10) # this is placed in 0 1
ttk.Entry(TAB2, textvariable=studentIdVar_update).grid(column=1, row=1, padx=10, pady=10) # this is placed in 1 1
#creating a button for search callback
ttk.Button(TAB2, text="Search", width=10, command=searchCallback).grid(column=2, row=1, padx=10, pady=10)

#creating a label and user entry field for Last Name
ttk.Label(TAB2, text = "Last Name").grid(column=0, row=2, padx=10, pady=10) # this is placed in 0 2
ttk.Entry(TAB2, textvariable=lNameVar_update).grid(column=1, row=2, padx=10, pady=10) # this is placed in 1 2

#creating a label and user entry field for First Name
ttk.Label(TAB2, text = "First Name").grid(column=0, row=3, padx=10, pady=10) # this is placed in 0 3
ttk.Entry(TAB2, textvariable=fNameVar_update).grid(column=1, row=3, padx=10, pady=10) # this is placed in 1 3
#creating array for courseGroupList1
courseGroupList1 = [
"101 - DataBase Management Systems",
"102 - Introduction to Data Analytics",
"103 - Data Mining"
]
```

```
#Initializing a variable called courseGroup1 for holding the selected choice from the optionmenu
courseGroup1_update = tk.StringVar(TAB2)
#setting default value in the optionmenu
courseGroup1_update.set(courseGroupList1[0])
# creating a label for optionmenu
ttk.Label(TAB2, text="Select Course 1").grid(column=0, row=4, padx=10, pady=10)
#Initializing optionmenu for the courseGroup1
ttk.OptionMenu(TAB2, courseGroup1_update, courseGroupList1[0], *courseGroupList1).grid(column=1, row=4, padx=10, pady=10)

#creating array for courseGroupList2
courseGroupList2 = [
"201 - Computer Network Security",
"202 - Microcomputer Applications",
"203 - Introduction to Decision Support Systems"
]

#Initializing a variable called courseGroup2 for holding the selected choice from the optionmenu
courseGroup2_update = tk.StringVar(TAB2)
#setting default value in the optionmenu
courseGroup2_update.set(courseGroupList2[0])
# creating a label for optionmenu
ttk.Label(TAB2, text="Select Course 2").grid(column=0, row=5, padx=10, pady=10)
#Initializing optionmenu for the courseGroup2
ttk.OptionMenu(TAB2, courseGroup2_update, courseGroupList2[0], *courseGroupList2).grid(column=1, row=5, padx=10, pady=10)

#creating array for courseGroupList3
courseGroupList3 = [
```

```
"301 - Python Programming",
"302 - R Programming",
"303 - AWS and Cloud Computing"
]

#Initializing a variable called courseGroup3 for holding the selected choice from the optionmenu
courseGroup3_update = tk.StringVar(TAB2)
#setting default value in the optionmenu
courseGroup3_update.set(courseGroupList3[0])
# creating a label for optionmenu
ttk.Label(TAB2, text="Select Course 3").grid(column=0, row=6, padx=10, pady=10)
#Initializing optionmenu for the courseGroup3
ttk.OptionMenu(TAB2, courseGroup3_update, courseGroupList3[0], *courseGroupList3).grid(column=1, row=6, padx=10, pady=10)

#creating a label and user check button field for is UnderGrad field
ttk.Label(TAB2, text="is UnderGrad?").grid(column=0, row=7, padx=10, pady=10)
#For defaulting check box unchecked (to make it checked pass value=1 in IntVar())
# 'Checkbutton' is used to create the check buttons
ttk.Checkbutton(TAB2, text = "", variable = isUnderGradCheckVar_update).grid(columnspan=2, row=7)


# This method is to submit user entered values in the Postgres database using a StudentRegistration class
def updateCallback():
    #creating an object called registrationObj for the StudentRegistration class
    registrationObj = StudentRegistration()
    #Assigning user entered studentID value to the StudentID of the registrationObj
```

```
registrationObj.studentId = studentIdVar_update.get()
#Assigning user entered lastName value to the lastName of the registrationObj
registrationObj.lastName = lNameVar_update.get()
#Assigning user entered firstName value to the firstName of the registrationObj
registrationObj.firstName = fNameVar_update.get()
#Splitting courseGroup1 user selected option and Assigning course1Id & course1Title value to the course1Id and course1Title of the registrationObj
registrationObj.course1Id = courseGroup1_update.get().split('-')[0]
registrationObj.course1Title = courseGroup1_update.get().split('-')[1].lstrip()
#Splitting courseGroup2 user selected option and Assigning course2Id & course2Title value to the course2Id and course2Title of the registrationObj
registrationObj.course2Id = courseGroup2_update.get().split('-')[0]
registrationObj.course2Title = courseGroup2_update.get().split('-')[1].lstrip()
#Splitting courseGroup3 user selected option and Assigning course3Id & course3Title value to the course3Id and course3Title of the registrationObj
registrationObj.course3Id = courseGroup3_update.get().split('-')[0]
registrationObj.course3Title = courseGroup3_update.get().split('-')[1].lstrip()
#If the user checks the check box assign 'true' value to isUnderGrad of the registrationObj and if not checked assign it to 'false'
if(isUnderGradCheckVar_update.get() == 1):
    registrationObj.isUnderGrad = 'true'
else:
    registrationObj.isUnderGrad = 'false'


#Creating a database connection and storing it in a dbConnection variable
dbConnection = database.connection();
#quering the database records with the studentID and storing query results in the 'records' variable
records = database.recordQuery(dbConnection, "id", registrationObj.studentId)
if(len(records) > 0):
```

```
    # If the above condition is met, we are updating the record in the database by using the above dbConnection variable and registrationObj
    status = database.recordUpdate(dbConnection, registrationObj)
    #If the record updation db operation returns a successful status, a status alert message will display a success message if not it displays error message
    if(status == True):
        statusVar_update.set("Successfully updated student id: "+registrationObj.studentId)
    else:
        statusVar_update.set("Error in updating details of student id: "+registrationObj.studentId)
  #if the above if condition is not met, a status message called "StudentID not registered" will be displayed
  else:
    print("Student Id: "+registrationObj.studentId+" not registered, to register please use Add tab")
    statusVar_update.set("Student Id: "+registrationObj.studentId+" not registered, to register please use Add tab")
  # Once all the above db operations are completed, we are closing the database connection
  database.disConnection(dbConnection);


# update button
ttk.Button(TAB2, text="Update", width=10, command=updateCallback).grid(column=1, row=10)
# status label
ttk.Label(TAB2, textvariable=statusVar_update).grid(column=1, row=11, ipadx=10)
```

```
# ----------------------------------------------------------------------------------------------------------------
---------------------------------
```

#Delete Student Details code

```
#adding Tab3 Heading
ttk.Label(TAB3, text="Delete Student Record").grid(column=0, row=0, padx=10, pady=10)


# defining string variables for the user input entries and status label
studentIdVar_delete = StringVar()
statusVar_delete = StringVar()
#creating a label and user entry field for Student ID
ttk.Label(TAB3, text = "Student ID").grid(column=0, row=1, padx=10, pady=10) # this is placed
in 0 1
ttk.Entry(TAB3, textvariable=studentIdVar_delete).grid(column=1, row=1, padx=10, pady=10) #
this is placed in 1 1


#deleteCallback method will be used to delete user entered studentId record from the database
def deleteCallback():
    #creating a registration object of StudentRegistration
    registrationObj = StudentRegistration()
    #Storing studentID value entered by the user in the studentID of the registrationObj
    registrationObj.studentId = studentIdVar_delete.get()
    #creating a database connection and storing in a db connection variable
    dbConnection = database.connection();
    #quering the database records with the studentID and storing query results in the 'records'
variable
    records = database.recordQuery(dbConnection, "id", registrationObj.studentId)


    if(len(records) > 0):
        # If the above condition is met, we are deleting the record in the database by using the above
dbConnection variable and registrationObj
```

```
        status = database.recordDelete(dbConnection, registrationObj)
```

#If the record deletion db operation returns a successful status, a status alert message will display a success message if not it displays error message

```
        if(status == True):
            statusVar_delete.set("Successfully deleted student id: "+registrationObj.studentId)
        else:
            statusVar_delete.set("Error in deleting details of student id: "+registrationObj.studentId)
```

#if the above if condition is not met, a status message called "Invalid Student id" will be displayed

```
    else:
        statusVar_delete.set("Invalid Student id: " + registrationObj.studentId)
```

 # Once all the above db operations are completed, we are closing the database connection

```
    database.disConnection(dbConnection);


# delete button
ttk.Button(TAB3, text="Delete", width=10, command=deleteCallback).grid(column=1, row=2)
# status label
ttk.Label(TAB3, textvariable=statusVar_delete).grid(column=1, row=11, ipadx=10)
# ----------------------------------------------------------------------------------------------------------------------------------
```

#Report Generation code

#adding Tab4 Heading

```
ttk.Label(TAB4, text="Report").grid(column=0, row=0, padx=10, pady=10)
```

#generateReport method will get the student records from database and generates the report based on courseID registration

```
def generateReport():
    #creating the data for dataframe for the plot with courses on x-axis and students enrolled on y-axis
    Data = {'Courses': ['101','102','103','201','202','203','301','302','303'],
```

```
        'Students Enrolled': [0,0,0,0,0,0,0,0,0]}

    #creating a database connection and storing in a dbConnection variable
    dbConnection = database.connection();
    #Replacing default values in the student enrolled data frame with the database query result of course_id 101
    Data['Students Enrolled'][0] = len(database.recordQuery(dbConnection, "course1_id", '101'))
    #Replacing default values in the student enrolled data frame with the database query result of course_id 102
    Data['Students Enrolled'][1] = len(database.recordQuery(dbConnection, "course1_id", '102'))
    #Replacing default values in the student enrolled data frame with the database query result of course_id 103
    Data['Students Enrolled'][2] = len(database.recordQuery(dbConnection, "course1_id", '103'))
    #Replacing default values in the student enrolled data frame with the database query result of course_id 201
    Data['Students Enrolled'][3] = len(database.recordQuery(dbConnection, "course2_id", '201'))
    #Replacing default values in the student enrolled data frame with the database query result of course_id 202
    Data['Students Enrolled'][4] = len(database.recordQuery(dbConnection, "course2_id", '202'))
    #Replacing default values in the student enrolled data frame with the database query result of course_id 203
    Data['Students Enrolled'][5] = len(database.recordQuery(dbConnection, "course2_id", '203'))
    #Replacing default values in the student enrolled data frame with the database query result of course_id 301
    Data['Students Enrolled'][6] = len(database.recordQuery(dbConnection, "course3_id", '301'))
    #Replacing default values in the student enrolled data frame with the database query result of course_id 302
    Data['Students Enrolled'][7] = len(database.recordQuery(dbConnection, "course3_id", '302'))
    #Replacing default values in the student enrolled data frame with the database query result of course_id 303
    Data['Students Enrolled'][8] = len(database.recordQuery(dbConnection, "course3_id", '303'))
```

```
# Once all the above db operations are completed, we are closing the database connection
database.disConnection(dbConnection)

#defining the dataframe with data and data columns
df = DataFrame(Data, columns= ['Courses', 'Students Enrolled'])
#Grouping by course count value for each courseId
df = df[['Courses', 'Students Enrolled']].groupby('Courses').sum()

#defining the size of the figure
figure = plt.Figure(figsize=(6,5), dpi=80)
#defining the orientation of the plot
ax1 = figure.add_subplot(111)
#For displaying the y-axis tick locators
ax1.yaxis.set_major_locator(MaxNLocator(integer=True))
#creating a bar graph in the Tab4
bar = FigureCanvasTkAgg(figure, master=TAB4)
#creating a widget for the above bar graph
bar.get_tk_widget().grid(row=1, column=0)
#to display the bar plot
df.plot(kind='bar', legend=True, ax=ax1)
#adding the title for the bar plot
ax1.set_title('Courses Vs. Students Enrolled')
#Displaying the detailed course list to explain the bar graph
coursesList.set("101 - DataBase Management Systems,\n102 - Introduction to Data Analytics,\n103 - Data Mining,\n201 - Computer Network Security,\n202 - Microcomputer Applications,\n203 - Introduction to Decision Support Systems,\n301 - Python Programming,\n302 - R Programming,\n303 - AWS and Cloud Computing")

#creating a string variable for storing the course list
```

```
coursesList=StringVar()
#displaying the course list data in the label
ttk.Label(TAB4, textvariable=coursesList).grid(column=1, row=1, ipadx=10)
#Generate button
ttk.Button(TAB4, text="Generate", width=15, command=generateReport).grid(column=1, row=0)

#Calling Main()
# Start GUI
window.mainloop()
```

database.py code

```
#Psycopg2 is the most popular PostgreSQL database adapter for the Python programming language
import psycopg2

#method for opening a connection with postgres database
def connection():
    try:
        #opening a connection by passing postgres database configurations
        connection = psycopg2.connect(user="postgres",
                                      password="rmu2019",
                                      host="127.0.0.1",
                                      port="5432",
                                      database="postgres")
        #returns the connection
        return connection
    #if the above connection is not successful, an expection will be thrown
    except (Exception, psycopg2.Error) as error :
```

```
        print ("Error while establishing a PostgreSQL connection: ", error)

#this method takes dbConnection as an argument and closes that particular dbConnection
def disConnection(dbConnection):
    #closing database connection.
    if(dbConnection):
        #If the dbConnection exists closing the database connection
        dbConnection.close()
        print("PostgreSQL connection is closed")

#Allows Python code to execute PostgreSQL command in a database session.
# Cursors are created by the connection.cursor() method: they are bound to the connection for the entire lifetime
#and all the commands are executed in the context of the database session wrapped by the connection.

#This method takes dbConnection queryId, queryValue as arguments and queries the records
def recordQuery(dbConnection, queryId, queryValue):
    try:
        #opening a cursor from a dbConnection
        cursor = dbConnection.cursor()
        #creating a query string by using the queryId and queryValue from the arguments
        queryString = "SELECT * FROM public.python_project_2019 WHERE " + queryId + " = " + queryValue
        #executing the above query string in the cursor
        cursor.execute(queryString)
        #fetching the result from the cursor
        records = cursor.fetchall()
        #returns the records
```

```
        return records

    #If the above cursor query executions is not successful an exception will be thrown
    except (Exception, psycopg2.Error) as error :
        print ("Error while fetching data from PostgreSQL", error)

    finally:
        if(cursor):
            #if the cursor is open clsoing the cursor
            cursor.close()
            print("PostgreSQL cursor is closed")

#This method takes dbConnection and registrationObj as arguments and inserts the record in the database
def recordInsertion(dbConnection, registrationObj):
    try:
        #opening a cursor from a dbConnection
        cursor = dbConnection.cursor()
        #executing the insert query string with the registration object in the cursor
        cursor.execute("INSERT INTO public.python_project_2019 (id, last_name, first_name,course1_id,course1_title,course2_id,course2_title,course3_id,course3_title,is_undergrad)VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)",(registrationObj.studentId,registrationObj.lastName,registrationObj.firstName,registrationObj.course1Id,registrationObj.course1Title,registrationObj.course2Id,registrationObj.course2Title,registrationObj.course3Id,registrationObj.course3Title,registrationObj.isUnderGrad))
        #committing the above insert operation
        dbConnection.commit()
        #assigning status to true
        status = True
```

```
    #If the cursor insert query execution is not successful an exception will be thrown
    except (Exception, psycopg2.Error) as error :
        print ("Error while fetching data from PostgreSQL", error)
        #assigning status to false
        status =  False

    finally:
        if(cursor):
            #if the cursor is open closing the cursor
            cursor.close()
            #returns the status
            return status
            print("PostgreSQL cursor is closed")

#This method takes dbConnection and registrationObj as arguments and updates the record in the database
def recordUpdate(dbConnection, registrationObj):
    try:
        #opening a cursor from a dbConnection
        cursor = dbConnection.cursor()
        #executing the update query string with the registration object in the cursor
        cursor.execute("UPDATE public.python_project_2019 SET last_name = %s,first_name = %s,course1_id = %s,course1_title = %s,course2_id = %s,course2_title = %s, course3_id = %s, course3_title = %s,is_undergrad = %s WHERE id = %s", (registrationObj.lastName,registrationObj.firstName,registrationObj.course1Id,registrationObj.course1Title,registrationObj.course2Id,registrationObj.course2Title,registrationObj.course3Id,registrationObj.course3Title,registrationObj.isUnderGrad, registrationObj.studentId))
        #committing the above update operation
        dbConnection.commit()
        #assigning status to true
```

```
        status = True

    #If the cursor update query execution is not successful an exception will be thrown
    except (Exception, psycopg2.Error) as error :
        print ("Error while fetching data from PostgreSQL", error)
        #assigning status to false
        status =  False

    finally:
        if(cursor):
            #if the cursor is open clsoing the cursor
            cursor.close()
            #returns the status
            return status
            print("PostgreSQL cursor is closed")

#This method takes dbConnection and registrationObj as arguments and deletes the record in the database
def recordDelete(dbConnection, registrationObj):
    try:
        #opening a cursor from a dbConnection
        cursor = dbConnection.cursor()
        #creating a query string by using the studentId of registrationObj from the arguments
        queryString = "DELETE FROM public.python_project_2019 WHERE id = " + registrationObj.studentId
        #executing the above query string in the cursor
        cursor.execute(queryString)
        #committing the above delete operation
        dbConnection.commit()
```

```
        #assigning status to true
        status = True

    #If the cursor update query execution is not successful an exception will be thrown
    except (Exception, psycopg2.Error) as error :
        print ("Error while fetching data from PostgreSQL", error)
        #assigning status to false
        status =  False

    finally:
        if(cursor):
            #if the cursor is open clsoing the cursor
            cursor.close()
            #returns the status
            return status
            print("PostgreSQL cursor is closed")
```

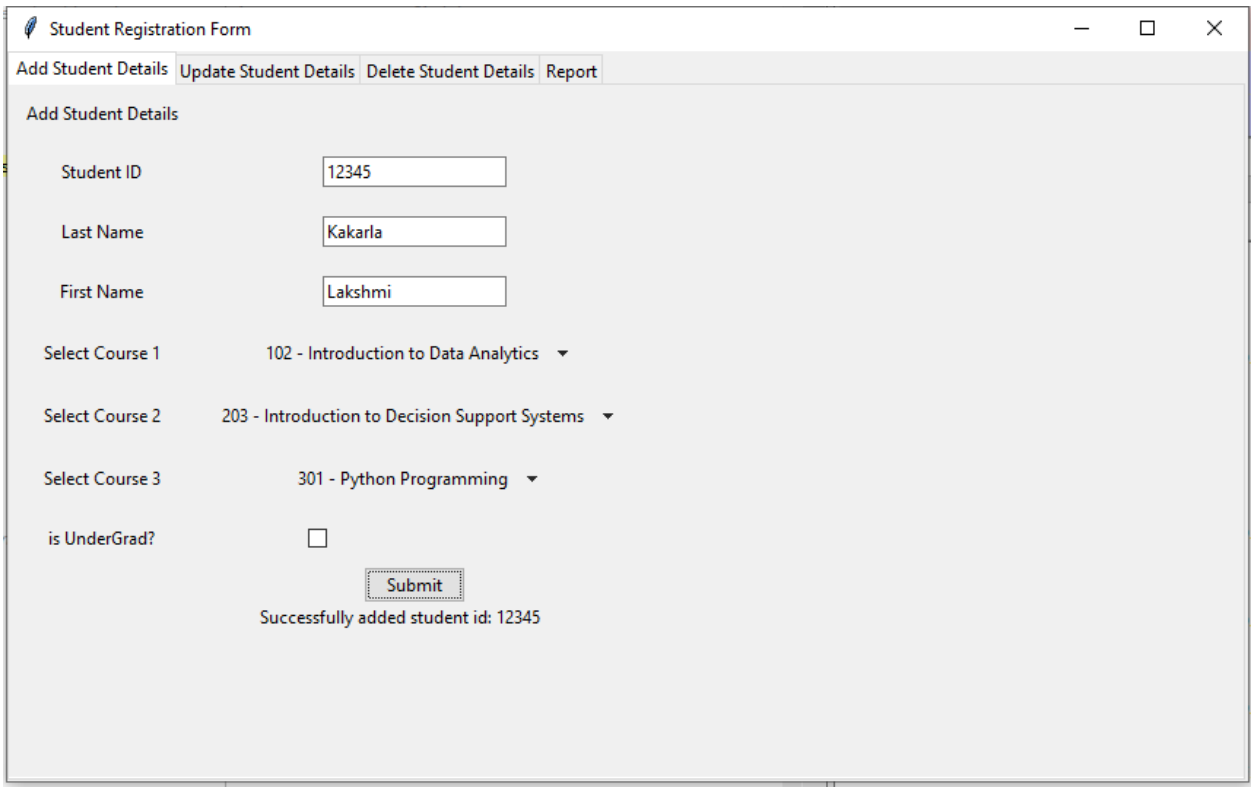
student.py code

```
# defining the properties of a student registration class
class StudentRegistration:
    studentId = str()
    lastName = str()
    firstname = str()
    course1Id = str()
    course1Title = str()
    course2Id = str()
    course2Title = str()
    course3Id = str()
    course3Title = str()
```

```
    isUnderGrad = str()

    #helper method to display all the student details in the string format

    def toString(self):

        return ("Student ID: "+self.studentId+", Last Name: "+self.lastName+", First Name: "+self.firstName+", Course1 ID: "+self.course1Id+", Course1 Title: "+self.course1Title+

                ", Course2 ID: "+self.course2Id+", Course2 Title: "+self.course2Title+", Course3 ID: "+self.course3Id+", Course3 Title: "+self.course3Title+", IsUnderGrad: "+self.isUnderGrad)
```

RESULTS:

1) Inserting a sample data to the database using the GUI



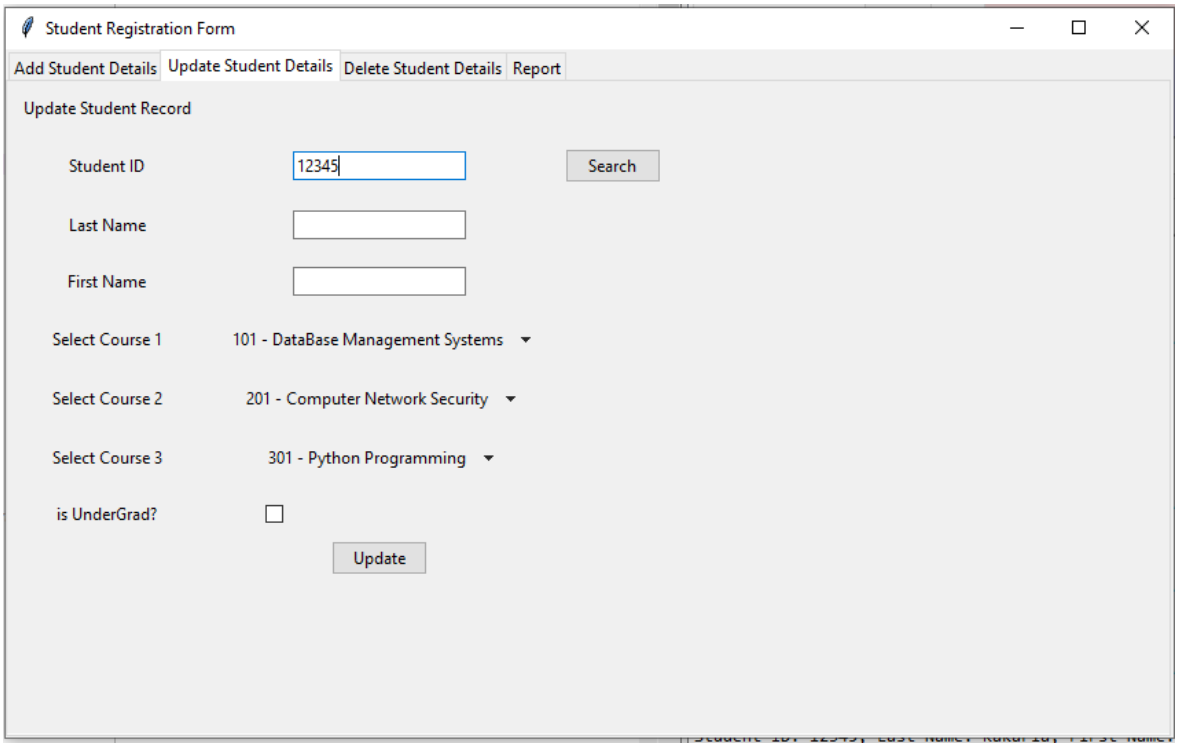
2) Output reflecting in the database as 41st student record

30	1421	Miley	Lydia	101	DataBase Management Syst...	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
31	105	Collins	Hannah	103	Data Mining	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
32	174	Kimberly	Nick	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	true
33	1528	Nolan	Sean	101	DataBase Management Syst...	202	Microcomputer Applications	302	R Programming	true
34	2317	Barone	Mai	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false
35	219	Wiggin	Micheal	102	Introduction to Data Analytics	201	Computer Network Security	302	R Programming	false
36	1453	Barone	Steven	103	Data Mining	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
37	705	Devlin	Shannon	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false
38	874	Tea	Dalton	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	true
39	1322	Gramz	Luke	101	DataBase Management Syst...	202	Microcomputer Applications	302	R Programming	true
40	652	Auth	Dylan	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false
41	12345	Kakarla	Lakshmi	102	Introduction to Data Analytics	203	Introduction to Decision Sup...	301	Python Programming	false

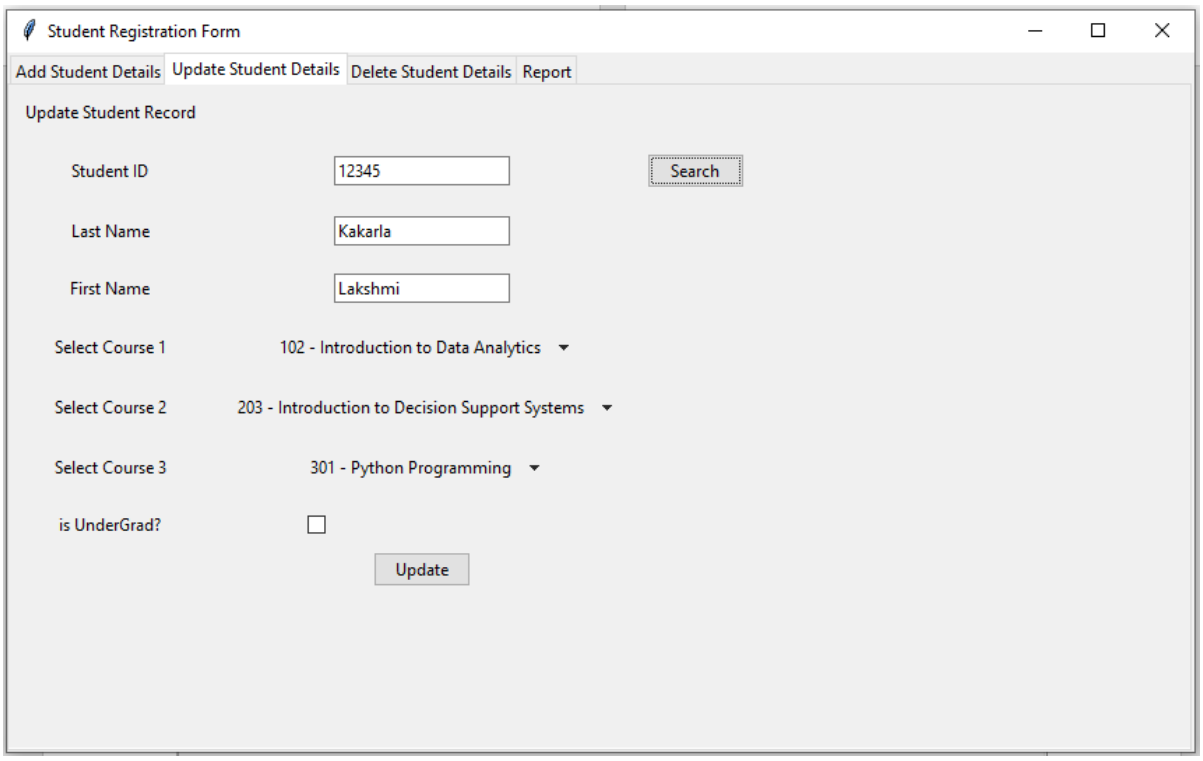
Successfully run. Total query runtime: 75 msec. 41 rows affected.

3) **Retrieving the student details using 'studentID' as search criteria**

a) **Entering Student Id value in the student ID entry field**

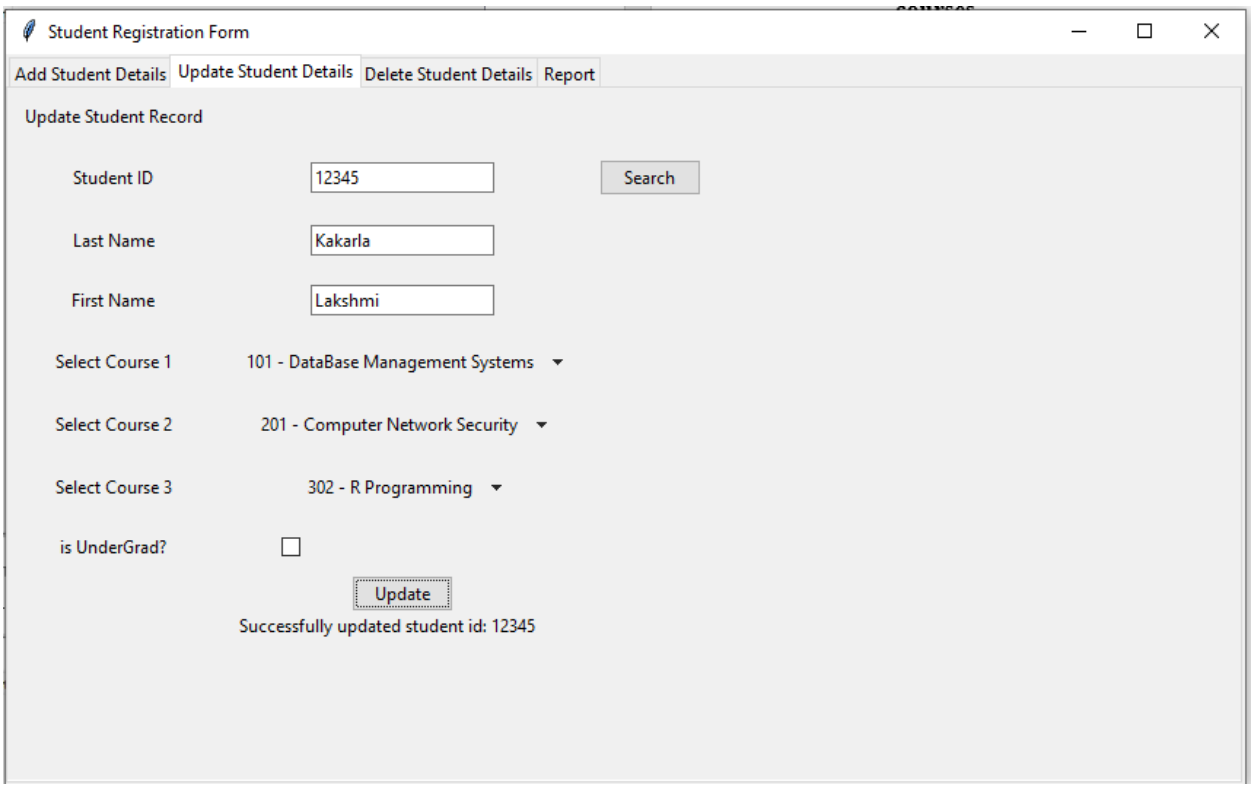


b) **Once entering the student ID value, click on the search button to retrieve the student registration details associated with the entered student ID**



4) **Updating the student Id: 12345 record details by changing the previous courses**

GUI

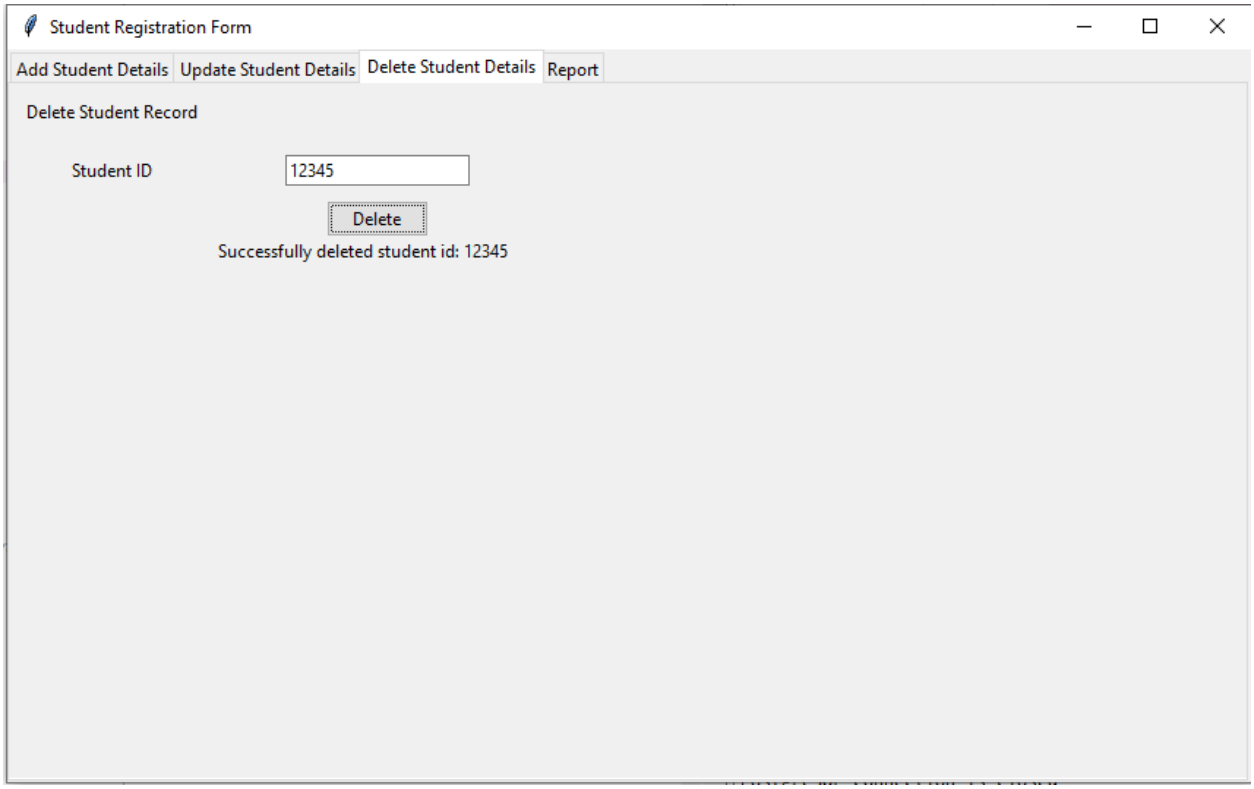


POSTGRESQL DATABASE

30	1421	Miley	Lydia	101	DataBase Management Syst...	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
31	105	Collins	Hannah	103	Data Mining	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
32	174	Kimberly	Nick	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	true
33	1528	Nolan	Sean	101	DataBase Management Syst...	202	Microcomputer Applications	302	R Programming	true
34	2317	Barone	Mai	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false
35	219	Wiggin	Micheal	102	Introduction to Data Analytics	201	Computer Network Security	302	R Programming	false
36	1453	Barone	Steven	103	Data Mining	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
37	705	Devlin	Shannon	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false
38	874	Tea	Dalton	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	true
39	1322	Gramz	Luke	101	DataBase Management Syst...	202	Microcomputer Applications	302	R Programming	true
40	652	Auth	Dylan	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false
41	12345	Kakarla	Lakshmi	101	DataBase Management Syst...	201	Computer Network Security	302	R Programming	false

5) Deleting the Student ID:12345 record from the database

GUI

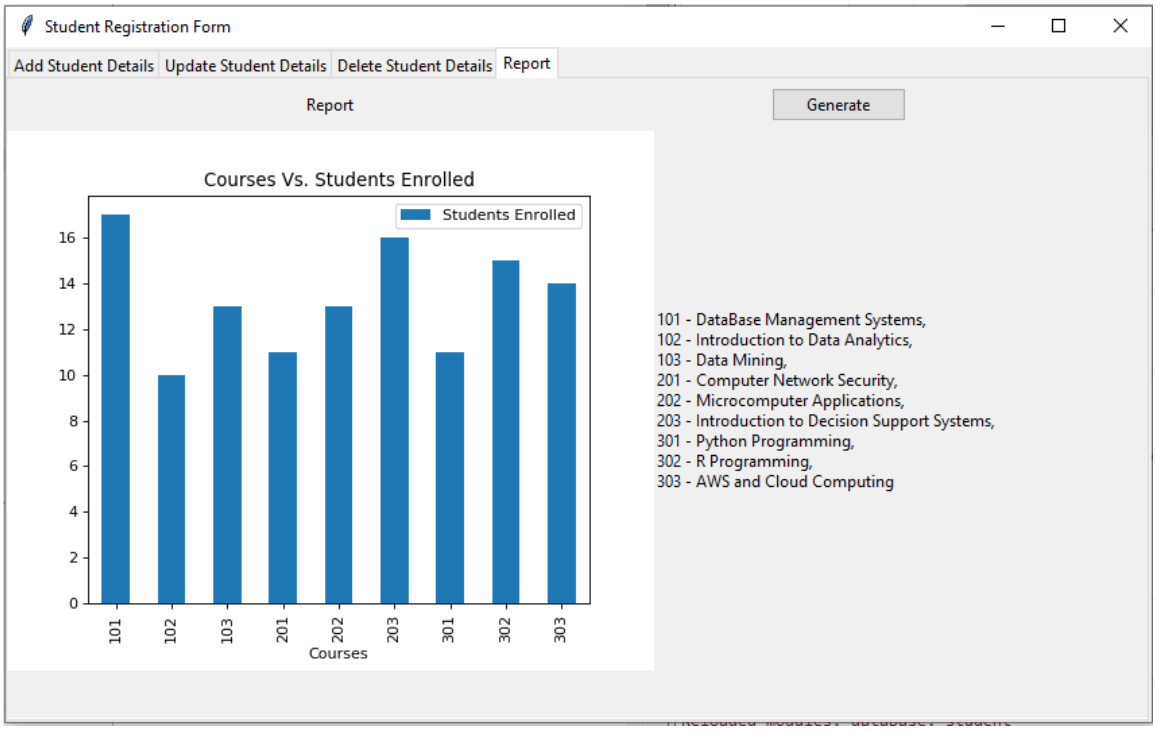


POSTGRESQL DATABASE

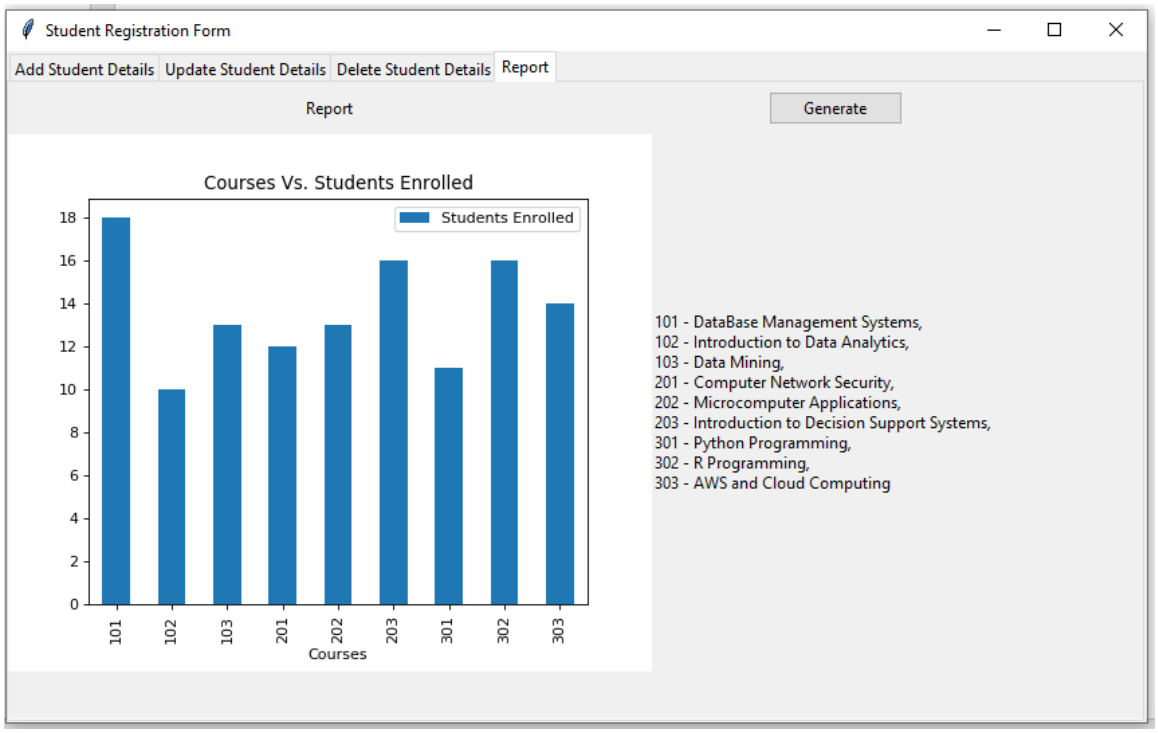
30	1421	Miley	Lydia	101	DataBase Management Syst...	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
31	105	Collins	Hannah	103	Data Mining	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
32	174	Kimberly	Nick	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	true
33	1528	Nolan	Sean	101	DataBase Management Syst...	202	Microcomputer Applications	302	R Programming	true
34	2317	Barone	Mai	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false
35	219	Wiggin	Micheal	102	Introduction to Data Analytics	201	Computer Network Security	302	R Programming	false
36	1453	Barone	Steven	103	Data Mining	203	Introduction to Decision Sup...	303	AWS and Cloud Computing	false
37	705	Devlin	Shannon	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false
38	874	Tea	Dalton	101	DataBase Management Syst...	203	Introduction to Decision Sup...	301	Python Programming	true
39	1322	Gramz	Luke	101	DataBase Management Syst...	202	Microcomputer Applications	302	R Programming	true
40	652	Auth	Dylan	103	Data Mining	202	Microcomputer Applications	303	AWS and Cloud Computing	false

6) Generating a report: Courses VS Students Enrolled

GUI Before adding student Id:12345



GUI After adding student Id:12345



ISSUES FACED DURING THE PROJECT DEVELOPMENT:

1) Psycopg2 module no found error

```
In [2]:

In [2]: runfile('C:/Users/Arjun/Desktop/PYTHON/project/studentRegistration/database.py', wdir='C:/Users/Arjun/Desktop/PYTHON/
project/studentRegistration')
Traceback (most recent call last):

  File "<ipython-input-2-856fc5d6009b>", line 1, in <module>
    runfile('C:/Users/Arjun/Desktop/PYTHON/project/studentRegistration/database.py', wdir='C:/Users/Arjun/Desktop/PYTHON/
project/studentRegistration')

  File "C:\Users\Arjun\Anaconda3\lib\site-packages\spyder_kernels\customize\spydercustomize.py", line 827, in runfile
    execfile(filename, namespace)

  File "C:\Users\Arjun\Anaconda3\lib\site-packages\spyder_kernels\customize\spydercustomize.py", line 110, in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "C:/Users/Arjun/Desktop/PYTHON/project/studentRegistration/database.py", line 2, in <module>
    import psycopg2

ModuleNotFoundError: No module named 'psycopg2'
```

Solution:

```
In [3]: pip install psycopg2
   ...:
Collecting psycopg2
  Downloading https://files.pythonhosted.org/packages/1a/85/853f11abfccfd581b099e5ae5f2dd807cc2919745b13d14e565022fd821c/
psycopg2-2.8.4-cp37-cp37m-win_amd64.whl (1.1MB)
Installing collected packages: psycopg2
Successfully installed psycopg2-2.8.4
Note: you may need to restart the kernel to use updated packages.
```

2) Error in generating plot in the tabular mode

```
In [22]: runfile('C:/Users/Arjun/Desktop/PYTHON/project/studentRegistration/main.py', wdir='C:/Users/Arjun/Desktop/PYTHON/
project/studentRegistration')
Reloaded modules: database, student
PostgreSQL cursor is closed
PostgreSQL cursor is closed
PostgreSQL cursor is closed
PostgreSQL cursor is closed
PostgreSQL cursor is closed
PostgreSQL cursor is closed
PostgreSQL cursor is closed
PostgreSQL cursor is closed
PostgreSQL cursor is closed
PostgreSQL connection is closed
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\Arjun\Anaconda3\lib\tkinter\__init__.py", line 1705, in __call__
    return self.func(*args)
  File "C:/Users/Arjun/Desktop/PYTHON/project/studentRegistration/main.py", line 473, in generateReport
    bar.get_tk_widget().grid(row=1, column=0)
NameError: name 'bar' is not defined
```

Solution:

Using *FigureCanvasTkAgg(figure, master=TAB4)*