Students' Report

Submitted by

Name of the student: Ayantika Mondal

Enrolment Number: 12022002011044

Section: I

Class Roll Number: 46

Stream: Electrical Engineering

Subject: Programming for Problem Solving with Python

Subject Code: IVC101

Department: Basic Science and Humanities

Under the Supervision of

Academic Year: 2022-26

PROJECT REPORT SUBMITED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE FIRST SEMESTER

DEPARTMENT OF BASIC SCIENCE AND HUMANITIES

INSTITUTE OF ENGINEERING AND MANAGEMENT, KOLKATA

CERTIFICATE OF RECOMMENDATION

We hereby recommend that the project prepared under our supervision by Ayantika Mondal, entitled **Students' Record** be accepted in partial fulfilment of the requirements for the degree of partial fulfilment of the first semester.

_____ _____

Head of the Department

Basic Science and Humanities

IEM, Kolkata

Project Supervisor

1. Introduction

There are multiple instances where the teachers are overwhelmed by the huge number of students and when they must keep track of their marks during the exam time. So, a program has been created in python keeping this problem in our mind to help the teachers to overcome this problem. This program aims to help the teachers in doing their work. The databases that are created here helps the teachers to access the data that is being already organized by the program created here into separate files which are in-turn related to each other via well-defined data relationships in order to ease the process of viewing the data, the program

creates several spreadsheets (in *.csv format) and displays different graphs in multiple forms for easier understanding at a glance.

1.1Objective

This program carefully sorts the students into various branches as per their courses and academic year. This program aims too efficiently: -

i)sort the students into their individual batches

ii)enrol them into their specific courses

iii)While keeping them under the supervision of their core department.

1.1 Organization of the Project

This program fetches the following data from the user: -

i) Name of the student,
ii) batch year in which they are studying,
iii) stream in which they are studying,
iv) class roll no. of the student.

After fetching the data from the user, the program provides the user with a detailed overview of the: i) Student details such as - Student Name, Student ID, Batch ID, and Class Roll No.

ii) Batch details such as - Batch ID, Department name, Batch name, Course list, Student list.

iii) Department details such as- Department ID, Department name, Batch list.

IV)Course details such as- Course ID, Course name, Marks obtained by the students

2. Database Descriptions ·

Student Database: It contains the Student Name, Student ID, Batch ID, and Class Roll No. It contains the basic information related to a student. It contains data in VARCHAR format. The student ID is the Primary Key.

Batch Database: It contains the various batches, their ids, the courses which are offered under those batches and the list of students who are under the batch.It consists of Batch ID , Department name, Batch name, Course list, Student list. It contains data in VARCHAR format. Batch ID is the primary key here. ·

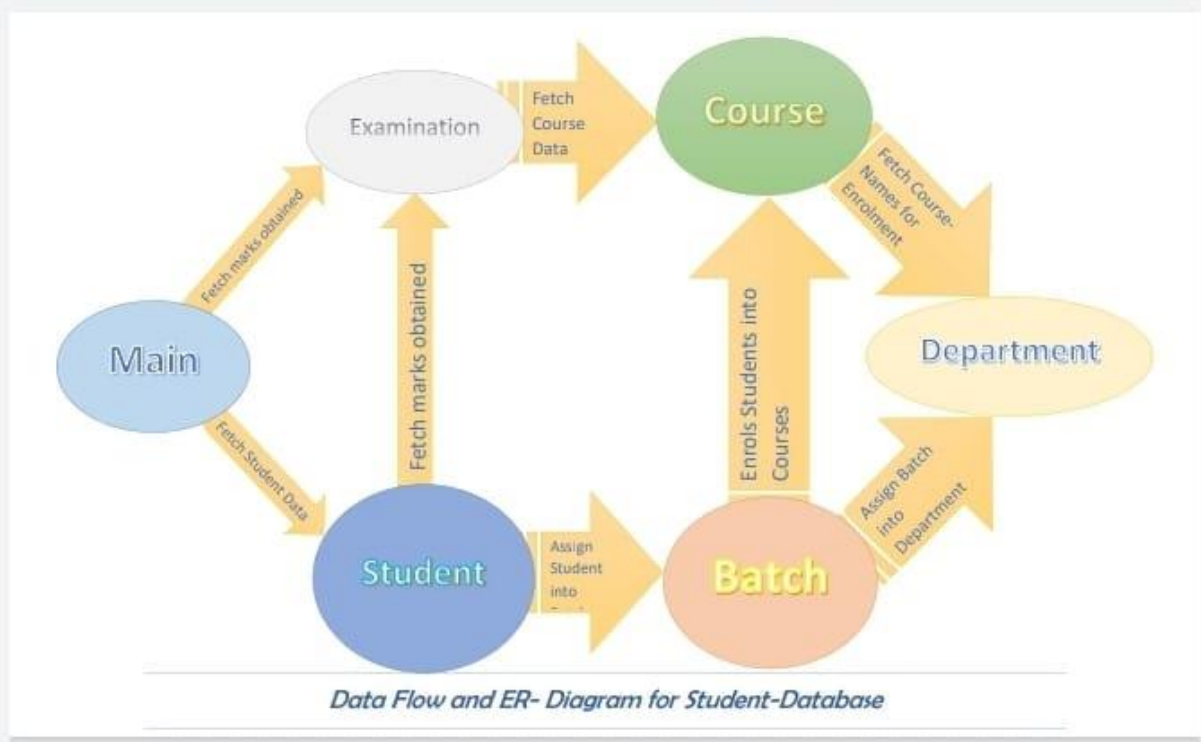
Department Database: It contains Department ID, Department name, Batch list. It contains data in VARCHAR format. Department ID is the primary key here. ·

Course Database: It contains the course names which fall under all the departments along with their ids and the marks received by all the students in the respective courses. It consists of Course ID, Course name, Marks obtained by the students. It contains data in VARCHAR format. Course ID is the primary key here. ·

2.1 Database Samples:

Provides samples of the database that are created or used. You may use screenshots.

3. Data Flow and E-R Diagrams:



Programs:

1. **All import functions used in the program**

```
#All imports
import os
import csv
import subprocess
import time
import sys
try:
    import matplotlib.pyplot as plt
except:
    subprocess.run(['pip', 'install', 'matplotlib'])
    import matplotlib.pyplot as plt
```

```
    path='C:/ProgrammingProject2022_Database'
    print('->'*15," Welcome to Student Database !!!!! ",'->'*15)
```

2. All sub-functions repeatedly used in the program

```
#All the Functions used Throughout the code
```

2)i) Loading function

```
#A simple loading function
def loading_screen():
    for i in range(6):
        sys.stdout.write("\r Loading" + "." * 3)
        sys.stdout.flush()
        time.sleep(0.3)
    sys.stdout.write("\r Loading complete !!!")
```

ii) file creation

```
#file-creation
def createfile(name,lst):
    with open(f'{path}/{name}','a',newline='')as x:
        script= csv.writer(x)
        script.writerow(lst)
        print(f" The Directory : {name} has been duly Updated
!!!!! ")
```

iii) marks% calculation

```
#marks % calculation
def percent(num):
    if stream.lower()=='cse' or stream.lower()=='cseai' or
stream.lower()=='cseaiml' or stream.lower()=='cseiot' or
stream.lower()=='csbs':
        num=(num*100)//600
    elif stream.lower()=='it' or stream.lower()=='ece' or
stream.lower()=='me':
        num=(num*100)//600
    return num
```

iv) gradation according to marks obtained

```
# Gradation according to marks obtained in total
def grade(num):
    if num>=90:
        return("Your Performance has been Outstanding ...\n Grade
obtained is :- A.")
    elif num>=80 and num<90:
```

```
        return("Your Performance has been Excellent...\n Grade
Obtained is :- B.")
    elif num>=70 and num<80:
        return("Your Performance has been Very Good...\n Grade
obtained is :- C.")
    elif num>=60 and num<70:
        return("Your performance is Good...\n Grade obtained is :-
D.")
    elif num>=50 and num<60:
        return("Your performance is Average and just have been
Passed...\nGrade Obtained is :- E.")
    else:
        return("Your performance has been extremely Poor and have
been Failed...\n Grade Obtained is :- F.")
```

v) counter function

```
#counter
def count(lst):
    num=0
    for i in lst:
        if str(type(i))=="<class 'int'>":
            num+=1
        else:
            pass
    return num
```

vi) adder function

```
#adder
def add(lst):
    plus=0
    for i in lst:
        try:
            plus+=i
        except:
            pass
    return plus
```

vii) duplicate record check function

```
#Function for duplication check
def duplicate(file,attr,pos=0):
    with open(f'{path}/{file}','r') as f:
        reader = csv.reader(f)
        dup_lst=[]
        for i in reader:
```

```
            dup_lst+=[i[pos]]
    if attr in dup_lst:
        return True
    else:
        return False
```

viii) stream choice an course assigner function

```
#Stream choice and course assigner
def choice(stream):
    if stream.lower()=='cse' or stream.lower()=='cseai' or
stream.lower()=='cseaiml' or stream.lower()=='cseiot' or
stream.lower()=='csbs':
        return ("C001:C002:C004:C005:C006:C007")
    elif stream.lower()=='it' or stream.lower()=='ece' or
stream.lower()=='me':
        return ("C001:C003:C004:C005:C006:C007")
```

ix) function for the assignment of batch

```
#batch assigner
def get_batch():
    with
open(f'C:/ProgrammingProject2022_Database/batchrecords.csv','r')
as x:
        reader=csv.reader(x)
        rows=[row for row in reader]
        column=[]
        for i in range(len(rows)):
            if i==0:
                pass
            else:
                column+=[rows[i][0]]
    return column
```

x) function for removing a particular student data from the directory

```
#Removal of a particular student from whole directory
def remove(string):
    with
open(f'C:/ProgrammingProject2022_Database/studentrecords.csv','r+'
,newline='') as x:
        script=csv.reader(x)
        rows=[row for row in script]
        for i in rows:
            if i[0]==string:
                rows[rows.index(i)]=['','','','']
            else:
                pass
        x.seek(0)
        x.truncate()
        writer=csv.writer(x)
```

```
        writer.writerows(rows)
```

3) Creation of Course graph

```
#creation of graph for course
def course_graph():
    color_lst=['#6666FF','#FF8000','#00FFFF','#1A7DE1','#FFFF00','
#FF007F']
    fig, ax = plt.subplots()
    legend_properties = {'weight':'heavy'}
    ax.set_facecolor("Blue")
    ax.tick_params(axis="both", colors="white")
    fig.set_facecolor("#B2FF66")
    ax.set_xlabel('Grades--------->', color="#FF8000")
    ax.set_ylabel('No. of Students--------->', color="#FF8000")
    ax.spines["bottom"].set_color("black")
    ax.spines["left"].set_color("black")
    ax.xaxis.label.set_weight("heavy")
    ax.yaxis.label.set_weight("heavy")
    count=0
    with open(f'{path}/courserecords.csv','r')as x:
        script= csv.reader(x)
        rows=[row for row in script]
        req=[]
        for i in range(len(rows)):
            if i==0:
                pass
            else:
                req+=[rows[i][2]]
        lst=[['Mathematics',(req[0].split('-'))[0:-1]],
             ['Physics',(req[1].split('-'))[0:-1]],
             ['Chemistry',(req[2].split('-'))[0:-1]],
             ['Electrical',(req[3].split('-'))[0:-1]],
             ['Mechanics',(req[4].split('-'))[0:-1]],
             ['PythonProgramming',(req[5].split('-'))[0:-1]],
             ['Biology',(req[6].split('-'))[0:-1]]]

        for i in range(len(lst)):
            for j in range(len(lst[i][1])):
                try:
                    lst[i][1][j]=grade(int((lst[i][1][j].split(':'
))[-1]))[-2]
                except:
                    lst[i][1][j]=''

        for k in range(7):
            a=lst[k][1].count('A')
```

```
            b=lst[k][1].count('B')
            c=lst[k][1].count('C')
            d=lst[k][1].count('D')
            e=lst[k][1].count('E')
            f=lst[k][1].count('F')
            lst[k][1]={'A':a,'B':b,'C':c,'D':d,'E':e,'F':f}

        for j in lst:
            x=list(j[1].keys())
            y=list(j[1].values())
            ax.plot(x, y,marker=",",color=color_lst[count-
1],label=j[0],linewidth=5)
            leg=plt.legend(fontsize=12,loc="upper right",
facecolor="Violet",edgecolor="Blue",prop=legend_properties)
            count+=1

        for text in leg.get_texts():
            text.set_color('Brown')

        plt.show()
```

4) Creation of batch graph

```
#creation of graph for different Batch
def batch_graph(arg):
    with open(f'{path}/batchrecords.csv','r') as x:
        reader=csv.reader(x)
        req=''
        rows=[row for row in reader]
        for i in range(len(rows)):
            if arg==rows[i][0]:
                req=rows[i][4]
                break
    req_lst=req.split(':')
    with open(f'{path}/courserecords.csv','r') as x:
        reader=csv.reader(x)
        rows=[row for row in reader]
        column=[]
        for i in range(len(rows)):
            if i==0:
                pass
            else:
                column+=[rows[i][2]]
        new_column=[]
        for j in range(len(column)):
            new_column+=(column[j].split('-'))[0:-1]
    new_req_lst=[]
    temp=[]
```

```
    for i in req_lst:
        for j in range(len(new_column)):
            if i in new_column[j]:
                temp+=[(new_column[j].split(':'))[-1]]
        new_req_lst+=[[[i]]+[temp]]
        temp=[]
    lst=[]
    temp=0
    grade_lst=[]
    for i in range(len(new_req_lst)):
        for j in range(7):
            try:
                temp+=int(new_req_lst[i][1][j])
            except:
                pass
        lst+=[new_req_lst[i][0]+[temp]]
        temp=0
    for i in range(len(lst)):
            grade_lst+=[grade(((lst[i][1]*100)//600)+10)[-2]]
            lst[i][1]=grade(((lst[i][1]*100)//600)+10)[-2]

    grade_no_lst={'A':grade_lst.count('A'),'B':grade_lst.count('B'
),'C':grade_lst.count('C'),'D':grade_lst.count('D'),'E':grade_lst.
count('E'),'F':grade_lst.count('F')}

    labels = list(grade_no_lst.keys())
    sizes = list(grade_no_lst.values())
    color_lst=['#F5DF53','#55EFBC','#895CC3','#1A7DE1','#A9F513','
#724706']
    explode = (0.01,0.1,0.02,0.05,0.03,0.1)
    new_labels=[]
    for i in range(len(labels)):
        new_labels+=[f'{labels[i]} : {str(sizes[i])}']

    fig,ax = plt.subplots()
    ax.set_facecolor("White")
    fig.set_facecolor("orange")
    plt.rcParams['font.weight'] = 'heavy'
    #plt.rcParams['font.size'] = '1'

    patches, texts=ax.pie(sizes, labels=new_labels,
colors=color_lst,explode=explode,shadow=True,startangle= -
90,textprops={'fontsize': 0})

    centre_circle = plt.Circle((0,0),0.8,fc='blue')
    fig = plt.gcf()
    fig.gca().add_artist(centre_circle)

    legend_properties = {'weight':'heavy'}
```

```
    leg=plt.legend(fontsize=12,loc="center",
facecolor="#FFCCCC",edgecolor="#990000",prop=legend_properties)
    for text in leg.get_texts():
        text.set_color('#FF3399')

    plt.title('Grades vs No. of Students in a
Batch',color='#3333FF',weight='heavy')
    plt.axis('equal')
    plt.show()
```

5) Creation of department graph

```
#creation of graph for different departments
def department_graph():
    need={}
    with open(f'{path}/batchrecords.csv','r') as x:
        reader=csv.reader(x)
        batch=[batch[0] for batch in reader]
        batch=batch[1:]
    for arg in batch:
        avg=0
        with open(f'{path}/batchrecords.csv','r') as x:
            reader=csv.reader(x)
            req=''
            rows=[row for row in reader]
            for i in range(len(rows)):
                if arg==rows[i][0]:
                    req=rows[i][4]
                    break
        req_lst=req.split(':')
        with open(f'{path}/courserecords.csv','r') as x:
            reader=csv.reader(x)
            rows=[row for row in reader]
            column=[]
            for i in range(len(rows)):
                if i==0:
                    pass
                else:
                    column+=[rows[i][2]]
            new_column=[]
            for j in range(len(column)):
                new_column+=(column[j].split('-'))[0:-1]
        new_req_lst=[]
        temp=[]
        for i in req_lst:
            for j in range(len(new_column)):
                if i in new_column[j]:
```

```
                    temp+=[(new_column[j].split(':'))[-1]]
            new_req_lst+=[[[i]]+[temp]]
            temp=[]
        lst=[]
        temp=0
        grade_lst=[]
        for i in range(len(new_req_lst)):
            for j in range(6):
                try:
                    temp+=int(new_req_lst[i][1][j])
                except:
                    pass
            lst+=[new_req_lst[i][0]+[temp]]
            temp=0

        for i in range(len(lst)):
                lst[i][1]=(lst[i][1]*100)/600

        for i in range(len(lst)):
            avg+=lst[i][1]
        avg=int(avg//len(lst))
        need[arg]=avg

    xdata = list(need.keys())
    ydata = list(need.values())
    color_lst=['#F5DF53','#55EFBC','#895CC3','#1A7DE1','#A9F513','
#724706']
    fig,ax = plt.subplots()
    ax.set_facecolor("White")
    fig.set_facecolor("#F29B18")
    ax.set_xlabel("X axis", color="black")
    ax.set_ylabel("Y axis", color="black")
    ax.spines["bottom"].set_color("#C0BBBB")
    ax.spines["left"].set_color("#F3E3E3")
    ax.spines['bottom'].set_linewidth(5)
    ax.spines['left'].set_linewidth(5)
    ax.xaxis.label.set_weight("heavy")
    ax.yaxis.label.set_weight("heavy")
    ax.tick_params(axis='x', labelcolor='#99FF33',
labelsize=10,color='#994C00',width=4)
    ax.tick_params(axis='y', labelcolor='#FFFF00',
labelsize=10,color='#994C00',width=4)

    plt.barh(xdata,ydata,color=color_lst,height=0.6,align='center'
)

    plt.title('Histogram depicting Average-score of Students in
each Batch',color='#990000',pad=20,fontweight='bold')
```

```
    plt.xlabel('Average->->->->->->->->->->->->->->->->-
>',labelpad=20)
    plt.ylabel('Batch->->->->->->->->->->->->->->->->->',
labelpad=20)
    plt.show()
```

6) Directory and sub folder creation

```
 #Directory and subfolder creation
try:
    os.makedirs(f'{path}/StudentReportCards')
    message=True
except:
    message=False

while message:
    #Batch-file creation
    createfile('batchrecords.csv',['Batch_ID','Batch_Name','Depart
ment_Name','Course_List','Student_List'])
    #Course-file creation
    createfile('courserecords.csv',['Course_ID','Course_Name','Obt
ained_Marks'])
    with open(f'{path}/courserecords.csv','a',newline='')as x:
        script= csv.writer(x)
        script.writerow(['C001','Mathematics'])
        script.writerow(['C002','Physics'])
        script.writerow(['C003','Chemistry'])
        script.writerow(['C004','Electrical'])
        script.writerow(['C005','Mechanics'])
        script.writerow(['C006','PythonProgramming'])
        script.writerow(['C007','Biology'])
    createfile('departmentrecords.csv',['Department_ID','Departmen
t_Name','Batch_List'])
    with open(f'{path}/departmentrecords.csv','a',newline='')as x:
        script= csv.writer(x)
        script.writerow(['CSE','Computer Science & Engineering'])
        script.writerow(['ECE','Electronics & Communications
Engineering'])
        script.writerow(['CSEAI','Computer Science & Engineering
(Artificial Intelligence)'])
        script.writerow(['CSEAIML','Computer Science & Engineering
(Artificial Intelligence and Machine Learning)'])
        script.writerow(['CSEIOT','Computer Science & Engineering
(Internet of Things)'])
        script.writerow(['CSBS','Computer Science & Business
Studies'])
        script.writerow(['IT','Information Technology'])
        script.writerow(['ME','Mechanical Engineering'])
```

```
    createfile('studentrecords.csv',['Student_ID','Student_Name','
Class_Roll-No.','Batch_ID'])
    createfile('examresults.csv',['Course_Name','Student_ID','Obta
ined_Marks'])
    break
```

7) Main-screen directions for know-how to use

```
#Main-screen directions for know-how to use
print('\n',
      ' #1 Computer Science & Engineering
:                                                    CSE\n',
      ' #2 Electronics & Communications Engineering
:                                              ECE\n',
      ' #3 Computer Science & Engineering and Artificial
Intelligence :                           CSEAI\n',
      ' #4 Computer Science & Engineering and Artificial
Intelligence and Machine Learning :   CSEAIML\n',
      ' #5 Computer Science & Engineering and Internet of Things
and Business Studies :         CSEIOT\n',
      ' #6 Computer Science & Business Studies
:                                                CSBS\n',
      ' #7 Information Technology
:
IT\n',
      ' #8 Mechanical Engineering
:
ME\n')
print("\n!!!!!! Stream Names to be written in short form as
mentioned above and in CAPITAL LETTERS only !!!!!\n\n")
print()
student_no=int(input("No. of students whose data are to be taken
input :- "))
print()
print('->'*35)
for i in range(student_no):
    name=input("Name of the Student : ")
    batch=input("Batch-year (e.g. 2019-23) : ")
    stream=input("Stream (e.g. CSE,ECE,CSEAI) : ")
    roll=input("Class Roll-Number : ")
```

8) ID creation

```
#id-creations
    batch_id=stream+batch[2:4]
    student_id=batch_id+roll
    batch_name=stream+batch
```

9) Student duplicate record check

```
#Student duplicate record check
```

```
    if duplicate('studentrecords.csv',student_id,0):
        print("Record of the student with Student_id :
",student_id," is already present in the Student-directory")
        print(f"Report card for the student can be found here :
{path}/StudentReportCards/{student_id}_{name}.txt")
    else:
        print("The subjects are
[Mathematics,Physics,Chemistry,Electrical,Mechanics,PythonProgramm
ing,Biology]")
        print("Subjects marks are to be entered in the above
mentioned order in a list type and",
              "\n if you dont have a particular subject put '0'
marks there \n e.g., for a CSE student [100,0,98,75,67,85,74])")
        print('\nEach Subject carries 100 marks !!!\n')
        print()
        marks_lst=eval(input("\nMarks list : "))
        total_marks=add(marks_lst)
        print()
```

10) Text file creation for student report card

```
 #text file creation for student report card
        with
open(f"{path}/StudentReportCards/{student_id}_{''.join(name.split(
))}.txt",'w') as x:

            x.writelines([f' Student Name : {name} \n',
                          f' Class Roll-number: {roll} \n',
                          f' Stream : {stream} \n',
                          f' Student ID : {student_id}\n',
                          '\n\n Marks obtained in the Following
subjects are\n ','->-'*15,
                          f'\n Mathematics :- {marks_lst[0]} \n'])
            if stream.lower()=='cse' or stream.lower()=='cseai' or
stream.lower()=='cseaiml' or stream.lower()=='cseiot' or
stream.lower()=='csbs':
                        x.writelines([f' Physics :- {marks_lst[1]}
\n'])
            elif stream.lower()=='it' or stream.lower()=='ece' or
stream.lower()=='me':
                        x.writelines([f' Chemistry :-
{marks_lst[2]} \n'])
            x.writelines([f' Electrical :- {marks_lst[3]} \n',
                          f' Mechanics :- {marks_lst[4]} \n'
                          f' PythonProgramming :- {marks_lst[5]}
\n',
                          f' Biology :- {marks_lst[6]} \n'])
            x.write('\n')
```

```
            x.write(f'Total marks out of 600 is : {total_marks}
and Percentage scored is : {percent(total_marks)}%\n')
            x.write(grade(total_marks/(count(marks_lst)-1)))
            x.write('\n')
        createfile('studentrecords.csv',[student_id,name,roll,batc
h_id])
        print(f"You can find your report card here :
{path}/StudentReportCards/{student_id}_{''.join(name.split())}.txt
")
        openpath=f"{path}/StudentReportCards/{student_id}_{''.join
(name.split())}.txt"
        subprocess.run(['start',openpath], shell=True)
```

11) Removal of student details from directory

```
#removal of student details from directory

        ask=input("Is this Student to be removed from database now
? (Y/N) : ")

        if ask.lower()=='n':
            if duplicate('batchrecords.csv',batch_id,0):
                with
open(f'{path}/batchrecords.csv','r+',newline='') as x:
                    script=csv.reader(x)
                    rows=[row for row in script]
                    for i in rows:
                        if batch_id==i[0]:
                            rows[rows.index(i)][4]+=f':{student_id
}'
                    x.seek(0)
                    x.truncate()
                    writer=csv.writer(x)
                    writer.writerows(rows)

                print("The Directory : batchrecords.csv has been
updated !!!")
            else:
                createfile('batchrecords.csv',[batch_id,batch_name
,stream,choice(stream),student_id])

            with open(f'{path}/courserecords.csv','r+',newline='')
as x:
                script=csv.reader(x)
                rows=[row for row in script]
                for i in range(len(rows)):
                    if i==0:
                        pass
                    else:
```

```
                        try:
                            rows[i][2]+=f'{student_id}:{marks_lst[
i-1]}-'
                        except:
                            rows[i].append(f'{student_id}:{marks_l
st[i-1]}-')
                x.seek(0)
                x.truncate()
                writer=csv.writer(x)
                writer.writerows(rows)
        else:
            remove(student_id)
            subprocess.call("TASKKILL /F /IM notepad.exe",
shell=True)
            os.remove(openpath)
            print('Records of this Student have been successfully
removed from the Student-directory')
    print('->'*35)
    print()
```

12) Department record updation

```
try:
    with open(f'{path}/departmentrecords.csv','r+',newline='') as
x:
        script=csv.reader(x)
        rows=[row for row in script]
        lst=get_batch()
        for i in lst:
            for j in rows:
                if i[0:-2]==j[0]:
                    try:
                        if i in j[2]:
                            pass
                        else:
                            rows[rows.index(j)][2]+=f'{i}:'
                    except:
                        rows[rows.index(j)].append(f'{i}:')
                    break
        x.seek(0)
        x.truncate()
        writer=csv.writer(x)
        writer.writerows(rows)

except:
    print("No new record to be added in departmentrecords.csv")
```

13) Viewing of the graphs

```
#Creation of the Graphs...
print()
print(" Provide necessary details Below to view Graph for various
Batches(in %)")
batch=input(" Batch-year (e.g. 2019-23) : ")
stream=input(" Stream (e.g. CSE,ECE,CSEAI) : ")
print('\n\n Please Close the dialog-box showing the Graph after
viewing to continue this Program !!!','\n Batch performance Graph
is Loading...','->'*15)
batch_id=stream+batch[2:4]

with open(f'{path}/batchrecords.csv','r') as x:
    reader=csv.reader(x)
    batch=[batch[0] for batch in reader]
    batch=batch[1:]

while True:
    if batch_id in batch:
        batch_graph(batch_id)
        break
    else:
        print(f'No Batch with Batch ID : {batch_id} exists in the
directory')
        ask=input("Do you want to View the Three Graphs ? (y/n) :
")
        if ask.lower()=='y':
            batch=input("Batch-year (e.g. 2019-23) : ")
            stream=input("Stream (e.g. CSE,ECE,CSEAI) : ")
            batch_id=stream+batch[2:4]
            continue
        else:
            print('Okay !!!')
            break
print()
print('\n\n Please Close the dialog-box showing the Graph after
viewing to continue this Program !!!','Course graph is
Loading...','->'*15)
print()
loading_screen()
course_graph()
print()
print()
print('\n\n Please Close the dialog-box showing the Graph after
viewing to continue this Program !!!',"Department wise average
graph is Loading...",'->'*15)
loading_screen()
department_graph()
print()
```

```
print()
```

16) Viewing of all the record files

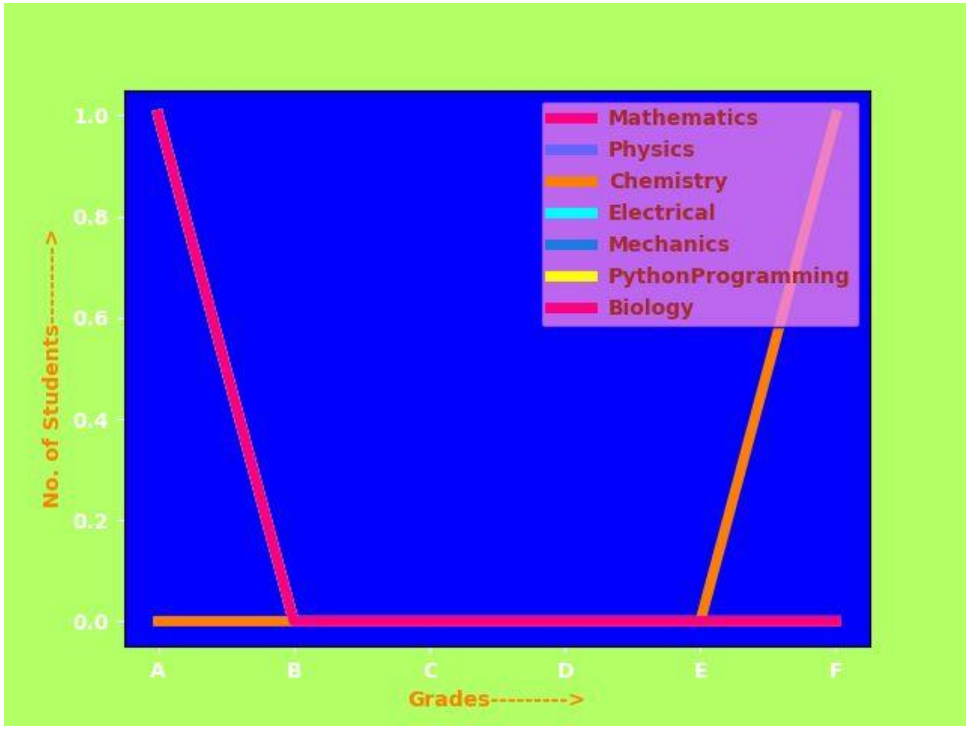
```
#viewing of all the record files
while True:
    ask2=input("Do you want to View all the Records (CSV format) ?
(y/n) : ")
    if ask2.lower()=='y':
            loading_screen()
            openpath=f"{path}/batchrecords.csv"
            subprocess.run(['start',openpath], shell=True)
            print("\n ")
            print("\n ")
            loading_screen()
            openpath=f"{path}/courserecords.csv"
            subprocess.run(['start',openpath], shell=True)
            print("\n ")
            print("\n ")
            loading_screen()
            openpath=f"{path}/departmentrecords.csv"
            subprocess.run(['start',openpath], shell=True)
            print("\n ")
            print("\n ")
            loading_screen()
            openpath=f"{path}/studentrecords.csv"
            subprocess.run(['start',openpath], shell=True)
            print("\n ")
            print("\n ")
            loading_screen()
            openpath=f"{path}/examresults.csv"
            subprocess.run(['start',openpath], shell=True)
            print("\n Viewing all Record files is completed now
Program will be closed ... Thank You !!!")
            break
    else:
        break

print("\n All File operations are completed now Program will be
closed ... Thank You !!!")
last=input("Press Enter to exit !!!")
subprocess.call("TASKKILL /F /IM notepad.exe", shell=True)
```

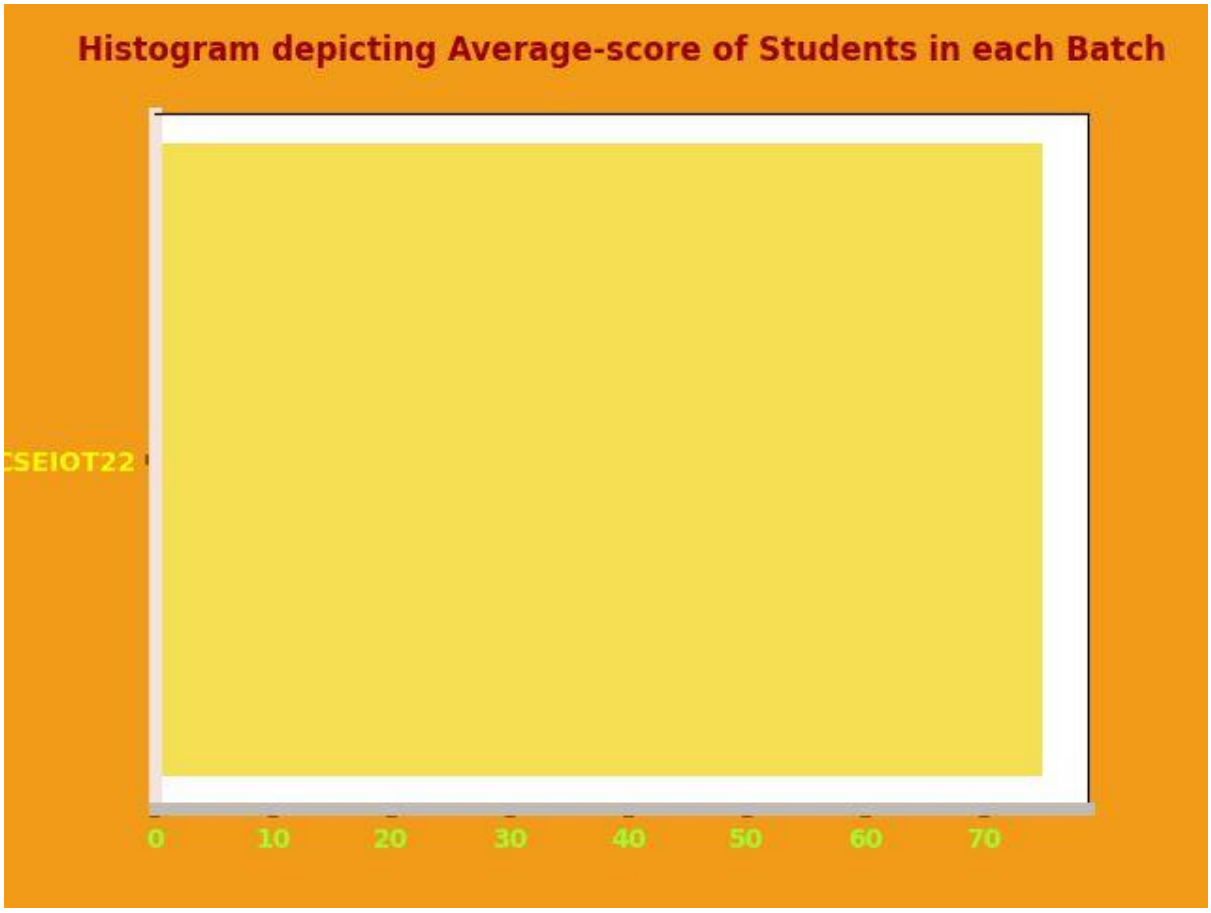
Outputs:

Pie chart for depicting the performance of students in each subject

Line graph depicting the performance of students in each subject



Histogram graph depicting the stream and batch wise performance of students


Histogram depicting Average-score of Students in each Batch
CSEIOT22
0
10
20
30
40
50
60
70