STUDENT EXAMINATION PORTAL

Submitted by

Name of the Students: AMBARISH SENGUPTA
Enrolment Number: 12022002002011
Section: A
Class Roll Number: 10
Stream: CSE
Subject: Programming for Problem Solving with Python
Subject Code: IVC101
Department: Basic Science and Humanities

Under the supervision of
Dr. INDRAJIT DEY

Academic Year: 2022-26

PROJECT REPORT SUBMITTED 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 **AMBARISH SENGUPTA ,** entitled STUDENT EXAMINATION PORTAL be accepted in partial fulfillment of the requirements for the degree of partial fulfillment of the first semester.

_____ _____

Head of the Department
Basic Sciences and Humanities
IEM, Kolkata

Project Supervisor

1 Introduction

In this “Student Examination Portal” project we can create a student database and link that database to the databases of Batches, Courses and Departments and generate an examination report card.

1.1 Objective

Create a student’s database of his/her batch, course, department, examination details and generate a report card.

1.2 Organization of the Project

First we execute the moduleselection.py file there we can find a menu selection with four options:

1. Create a student database.
2. View Batch database.
3. View Course database.
4. View Department database.

If we select the first option it will execute the stdmanagement.py file placed in the same directory and create a student database step by step. It will store its data in the Student.csv, Batch.csv, Course.csv, Department.csv simultaneously. It will also show the report card of the student, plot graph, pie chart and histogram grade wise.

If we select the second option it will allow us to see the detailed Batch database stored till now.

If we select the second option it will allow us to see the detailed Course database stored till now.

If we select the second option it will allow us to see the detailed Department database stored till now.

2 Database Descriptions

The Student.csv database contains the name and ID of the students in a particular batch and department.

The Batch.csv database contains Batch Id, Batch name, Department name and list of courses and students enrolled in the department.

The Course.csv database contains the course ID of each course and marks of each student enrolled in the particular course.

The Department.csv database contains details of each department.

2.1 Database Samples

Student ID	Name	Class Roll Number	Batch ID
ECE2145	Abhirup Kundu	45	ECE21
CSE2275	Tiyasha Paul	75	CSE22
CSE2250	Indrava Chowdhury	50	CSE22
CSE2287	Farhan Rastogi	87	CSE22
CSE2274	Tista Mukherjee	74	CSE22
CSE2261	Devsatyam Ray	61	CSE22
ECE2185	Suvadra Roy Chowdhury	85	ECE21

Student.csv

Batch ID	Batch Name	Department Name	List of Courses	List of Students
ECE21	ECE2021-25	ECE	C002:C003:C004:C005:C006	ECE2145:ECE2185
CSE22	CSE2022-26	CSE	C001:C002:C003:C004:C005:C006	CSE2275:CSE2250:CSE2287:CSE2274:CSE2261

Batch.csv

Course ID	Course Name	Marks Obtained
C001	Python Programming	ECE2145:78-CSE2275:98-CSE2250:88-CSE2287:80-CSE2274:88-CSE2261:74-ECE2185:75-
C002	Math	ECE2145:89-CSE2275:77-CSE2250:98-CSE2287:79-CSE2274:86-CSE2261:86-ECE2185:88-
C003	Physics	ECE2145:77-CSE2275:54-CSE2250:99-CSE2287:88-CSE2274:97-CSE2261:77-ECE2185:94-
C004	Chemistry	ECE2145:88-CSE2275:84-CSE2250:78-CSE2287:75-CSE2274:67-CSE2261:79-ECE2185:52-
C005	Biology	ECE2145:96-CSE2275:89-CSE2250:65-CSE2287:98-CSE2274:99-CSE2261:72-ECE2185:77-
C006	English	ECE2145:100-CSE2275:63-CSE2250:99-CSE2287:96-CSE2274:97-CSE2261:88-ECE2185:76-

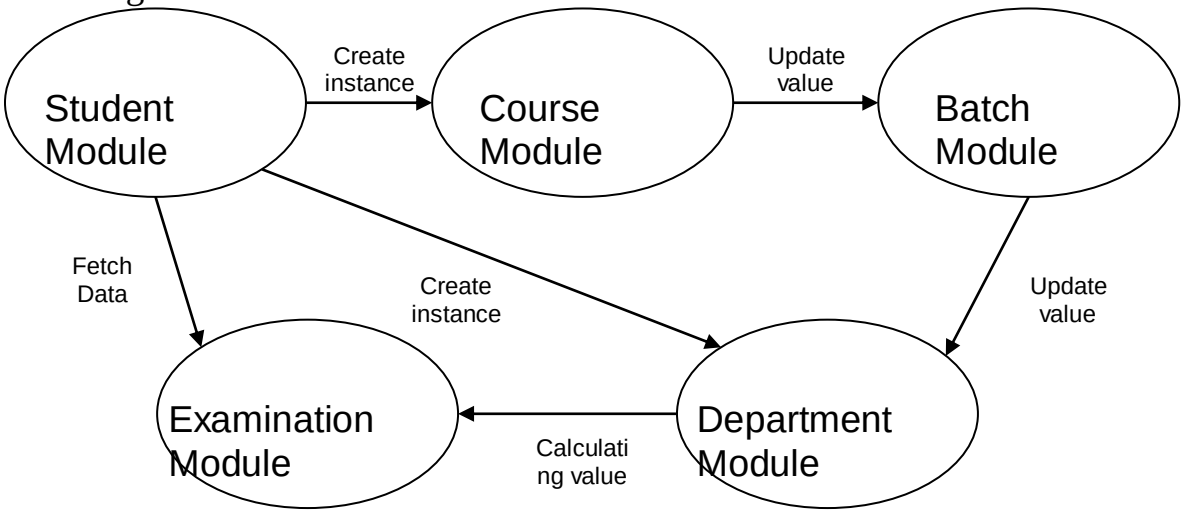
Course.csv

Department ID	Department Name
CSE	Computer Sience and Engineering
CSEAI	Computer Sience and Engineering and Artificial Intelligence
CSEAIML	Computer Sience and Engineering and Artificial Intelligence and Machine Learning
CSEIOTCSBS	Computer Sience and Engineering and Internet of Things and Business Studies
IT	Information Technology
ECE	Electrical and Communications Engineering
ME	Mechanical Engineering

Department.csv

3. Data Flow and E-R Diagrams

Demonstrate the dependency of all the python modules written using data flow diagrams



4. Programs

Provide the python programs of the various modules.

1) rootDir/stdmanagement.py

```
importos
importcsv
importsubprocess
importtime
importsys
print("                                    *****STUDENT MANAGEMENT
SYSTEM*****                                   ")
try:
importmatplotlib.pyplotasplt
except:
subprocess.run(['pip', 'install', 'matplotlib'])
importmatplotlib.pyplotasplt

path='C:/StudentManagement_main-folder'
```

```
defpercent(num):
ifstream.lower()=='cse'orstream.lower()=='cseai'orstream.lower()=='cseaim
l'orstream.lower()=='cseiotcsbs':
num=(num*100)//600
elifstream.lower()=='it'orstream.lower()=='ece'orstream.lower()=='me':
num=(num*100)//500
returnnum
defcreatefile(name,lst):
withopen(f'{path}/{name}','a',newline='')asf:
script= csv.writer(f)
script.writerow(lst)
print(f"{name} file has been SAVED!!")


defcount(lst):
num=0
foriinlst:
ifstr(type(i))=="<class 'int'>":
num+=1
else:
pass
returnnum



defgrade(num):
ifnum>=90:
return("You have passed the exam with grade A.")
elifnum<90andnum>=80:
return("You have passed the exam with grade B.")
elifnum<80andnum>=70:
return("You have passed the exam with grade C.")
elifnum<70andnum>=60:
return("You have passed the exam with grade D.")
elifnum<60andnum>=50:
return("You have passed the exam with grade E.")
else:
```

```
return("You have Failed the Exam with grade F.")



defadd(lst):
plus=0
foriinlst:
try:
plus+=i
except:
pass
returnplus

defduplicate(file,attr,pos=0):
withopen(f'{path}/{file}','r') asf:
reader = csv.reader(f)
dup_lst=[]
foriinreader:
dup_lst+=[i[pos]]
ifattrindup_lst:
returnTrue
else:
returnFalse

defchoice(stream):
ifstream.lower()=='cse'orstream.lower()=='cseai'orstream.lower()=='cseaim
l'orstream.lower()=='cseiotcsbs':
return ("C001:C002:C003:C004:C005:C006")
elifstream.lower()=='it'orstream.lower()=='ece'orstream.lower()=='me':
return ("C002:C003:C004:C005:C006")

defget_batch():
withopen(f'C:/StudentManagementSystem_main-folder/Batch.csv','r') asf:
reader=csv.reader(f)
rows=[rowforrowinreader]
column=[]
foriinrange(len(rows)):
ifi==0:
pass
```

```
else:
column+=[rows[i][0]]
returncolumn

defremove(string):
withopen(f'C:/StudentManagementSystem_main-
folder/Student.csv','r+',newline='') asf:
script=csv.reader(f)
rows=[rowforrowinscript]
foriinrows:
ifi[0]==string:
rows[rows.index(i)]=['','','','']
else:
pass
f.seek(0)
f.truncate()
writer=csv.writer(f)
writer.writerows(rows)



defcourse_graph():
color_lst=['#C70039','#9BB1F2','#FFC300','#FF5733','#DAAFB1','#86B7C8']
fig, ax = plt.subplots()
legend_properties = {'weight':'heavy'}
ax.set_facecolor("Black")
ax.tick_params(axis="both", colors="white")
fig.set_facecolor("Black")
ax.set_xlabel('Grades--------->', color="white")
ax.set_ylabel('No. of Students--------->', color="white")
ax.spines["bottom"].set_color("white")
ax.spines["left"].set_color("white")
ax.xaxis.label.set_weight("heavy")
ax.yaxis.label.set_weight("heavy")
count=0
withopen(f'{path}/Course.csv','r')asf:
script= csv.reader(f)
rows=[rowforrowinscript]
req=[]
```

```
foriinrange(len(rows)):
ifi==0:
pass
else:
req+=[rows[i][2]]
lst=[['Problem Solving with Python',(req[0].split('-'))[0:-1]],
             ['Mathematics',(req[1].split('-'))[0:-1]],
             ['Physics',(req[2].split('-'))[0:-1]],
             ['Chemistry',(req[3].split('-'))[0:-1]],
             ['Biology',(req[4].split('-'))[0:-1]],
             ['English',(req[5].split('-'))[0:-1]]]

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

forkinrange(6):
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}

forjinlst:
x=list(j[1].keys())
y=list(j[1].values())
ax.plot(x, y,marker=",",color=color_lst[count],label=j[0],linewidth=3)
leg=plt.legend(fontsize=10,loc="upper right",
facecolor="Black",edgecolor="Black",prop=legend_properties)
count+=1

fortextinleg.get_texts():
text.set_color('White')
```

```
plt.show()

defbatch_graph(arg):
withopen(f'{path}/Batch.csv','r') asf:
reader=csv.reader(f)
req=''
rows=[rowforrowinreader]
foriinrange(len(rows)):
ifarg==rows[i][0]:
req=rows[i][4]
break
req_lst=req.split(':')
withopen(f'{path}/Course.csv','r') asf:
reader=csv.reader(f)
rows=[rowforrowinreader]
column=[]
foriinrange(len(rows)):
ifi==0:
pass
else:
column+=[rows[i][2]]
new_column=[]
forjinrange(len(column)):
new_column+=(column[j].split('-'))[0:-1]
new_req_lst=[]
temp=[]
foriinreq_lst:
forjinrange(len(new_column)):
ifiinnew_column[j]:
temp+=[(new_column[j].split(':'))[-1]]
new_req_lst+=[[[i]]+[temp]]
temp=[]
lst=[]
temp=0
grade_lst=[]
foriinrange(len(new_req_lst)):
forjinrange(6):
try:
temp+=int(new_req_lst[i][1][j])
```

```
except:
pass
lst+=[new_req_lst[i][0]+[temp]]
temp=0
foriinrange(len(lst)):
iflst[i][0][:3]=='CSE':
grade_lst+=[grade((lst[i][1]*100)//600)[-2]]
lst[i][1]=grade((lst[i][1]*100)//600)[-2]
else:
grade_lst+=[grade((lst[i][1]*100)//500)[-2]]
lst[i][1]=grade((lst[i][1]*100)//500)[-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':gra
de_lst.count('F')}

labels = list(grade_no_lst.keys())
sizes = list(grade_no_lst.values())
color_lst=['#C70039','#9BB1F2','#FFC300','#FF5733','#DAAFB1','#86B7C8']
explode = (0.01,0.1,0.02,0.05,0.03,0.1)
new_labels=[]
foriinrange(len(labels)):
new_labels+=[f'{labels[i]} : {str(sizes[i])}']

fig,ax = plt.subplots()
ax.set_facecolor("Black")
fig.set_facecolor("Black")
plt.rcParams['font.weight'] = 'heavy'


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),0.60,fc='black')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

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

```
leg=plt.legend(fontsize=10,loc="center",
facecolor="Black",edgecolor="Black",prop=legend_properties)
fortextinleg.get_texts():
text.set_color('white')

plt.title('Overall Grades vs No. of
Students',color='White',weight='heavy')
plt.axis('equal')
plt.show()

defdepartment_graph():
need={}
withopen(f'{path}/Batch.csv','r') asf:
reader=csv.reader(f)
batch=[batch[0] forbatchinreader]
batch=batch[1:]
forarginbatch:
avg=0
withopen(f'{path}/Batch.csv','r') asf:
reader=csv.reader(f)
req=''
rows=[rowforrowinreader]
foriinrange(len(rows)):
ifarg==rows[i][0]:
req=rows[i][4]
break
req_lst=req.split(':')
withopen(f'{path}/Course.csv','r') asf:
reader=csv.reader(f)
rows=[rowforrowinreader]
column=[]
foriinrange(len(rows)):
ifi==0:
pass
else:
column+=[rows[i][2]]
new_column=[]
forjinrange(len(column)):
new_column+=(column[j].split('-'))[0:-1]
```

```
new_req_lst=[]
temp=[]
foriinreq_lst:
forjinrange(len(new_column)):
ifiinnew_column[j]:
temp+=[(new_column[j].split(':'))[-1]]
new_req_lst+=[[[i]]+[temp]]
temp=[]
lst=[]
temp=0
grade_lst=[]
foriinrange(len(new_req_lst)):
forjinrange(6):
try:
temp+=int(new_req_lst[i][1][j])
except:
pass
lst+=[new_req_lst[i][0]+[temp]]
temp=0
foriinrange(len(lst)):
iflst[i][0][:3]=='CSE':
lst[i][1]=(lst[i][1]*100)/600
else:
lst[i][1]=(lst[i][1]*100)/500
foriinrange(len(lst)):
avg+=lst[i][1]
avg=int(avg//len(lst))
need[arg]=avg

xdata = list(need.keys())
ydata = list(need.values())
color_lst=['#C70039','#9BB1F2','#FFC300','#FF5733','#DAAFB1','#86B7C8']
fig,ax = plt.subplots()
ax.set_facecolor("Black")
fig.set_facecolor("Black")
ax.set_xlabel("X axis", color="white")
ax.set_ylabel("Y axis", color="white")
ax.spines["bottom"].set_color("white")
ax.spines["left"].set_color("white")
```

```
ax.spines['bottom'].set_linewidth(2)
ax.spines['left'].set_linewidth(2)
ax.xaxis.label.set_weight("heavy")
ax.yaxis.label.set_weight("heavy")
ax.tick_params(axis='x', labelcolor='white',
labelsize=10,color='white',width=2)
ax.tick_params(axis='y', labelcolor='white',
labelsize=10,color='white',width=2)

plt.barh(xdata,ydata,color=color_lst,height=0.3,align='center')

plt.title('Histogram of Average of Students vs
Batch',color='white',pad=17,fontweight='bold')
plt.xlabel('Average----------------->')
plt.ylabel('Batch----------------->', labelpad=15)
plt.show()

defloading_screen():
foriinrange(10):
sys.stdout.write("\rLoading" + "_" * i)
sys.stdout.flush()
time.sleep(0.3)
sys.stdout.write("\rLoading completed!")


#Creation of Folder and all the Modules recquired...
try:
os.makedirs(f'{path}/ReportCards')
message=True
except:
message=False

whilemessage:
createfile('Batch.csv',['Batch ID','BatchName','DepartmentName','List of
Courses','List of Students'])
createfile('Course.csv',['Course ID','CourseName','Marks Obtained'])
withopen(f'{path}/Course.csv','a',newline='')asf:
script= csv.writer(f)
script.writerow(['C001','Python Programming'])
```

```
script.writerow(['C002','Math'])
script.writerow(['C003','Physics'])
script.writerow(['C004','Chemistry'])
script.writerow(['C005','Biology'])
script.writerow(['C006','English'])
createfile('Department.csv',['Department ID','DepartmentName','List of
Batches'])
withopen(f'{path}/Department.csv','a',newline='')asf:
script= csv.writer(f)
script.writerow(['CSE','Computer Science and Engineering'])
script.writerow(['CSEAI','Computer Science and Engineering and Artificial
Intelligence'])
script.writerow(['CSEAIML','Computer Science and Engineering and
Artificial Intelligence and Machine Learning'])
script.writerow(['CSEIOTCSBS','Computer Science and Engineering and
Internet of Things and Business Studies'])
script.writerow(['IT','Information Technology'])
script.writerow(['ECE','Electrical and Communications Engineering'])
script.writerow(['ME','Mechanical Engineering'])
createfile('Student.csv',['Student ID','Name','Class Roll Number','Batch
ID'])
createfile('Examination.csv',['Course Name','StudentID','Marks'])
break

print('\n','Computer Science and Engineering : CSE','\n',
'Computer Science and Engineering and Artificial Intelligence :
CSEAI','\n',
'Computer Science and Engineering and Artificial Intelligence and Machine
Learning : CSEAIML','\n',
'Computer Science and Engineering and Internet of Things and Business
Studies : CSEIOTCSBS','\n',
'Information Technology : IT','\n',
'Electrical and Communications Engineering : ECE','\n',
'Mechanical Engineering : ME','\n')
print("Please write all the stream name in short form as mentioned above
and in capital letters only!!!")
print()
```

```
student_no=int(input("Enter the no. of students whose data you want to
input : "))
print()
print('-'*50)
foriinrange(student_no):
name=input("Enter Student's Name : ")
batch=input("Enter batch (e.g. 2022-26) : ")
stream=input("Enter stream (e.g. CSE) : ")
roll=input("Enter Class Roll Number : ")

batch_id=stream+batch[2:4]
student_id=batch_id+roll
batch_name=stream+batch

ifduplicate('Student.csv',student_id,0):
print("the student is already present in the directory")
print(f"You can find your report card here :
{path}/ReportCards/{student_id}_{name}.txt")
else:
print()
print("The subjects are [Problem Solving with
Python,Math,Physics,Chemistry,Biology,English]")
print('please enter the subjects marks in the above mentioned order in a
list type and if you dont have a particular subject write there "null"
(e.g. [100,100,"null",75,69,85])')
print('Each Subject is ot of 100 marks')
print()
marks_lst=eval(input("Enter the Marks list : "))
total_marks=add(marks_lst)
print()

withopen(f"{path}/ReportCards/{student_id}_{''.join(name.split())}.txt",'
w') asf:

f.writelines([f'Name of the student : {name}\n',
f'Class Roll of the student : {roll}\n',
f'Stream of the student : {stream}\n',
f'Your Student ID is : {student_id}\n',
'\n',
```

```
f'Marks obtained in Problem Solving with Python is : {marks_lst[0]}\n',
f'Marks obtained in Math is : {marks_lst[1]}\n',
f'Marks obtained in Physics is : {marks_lst[2]}\n',
f'Marks obtained in Chemistry is : {marks_lst[3]}\n',
f'Marks obtained in Biology is : {marks_lst[4]}\n',
f'Marks obtained in English is : {marks_lst[5]}\n'])

f.write('\n')
f.write(f'You have got {total_marks} in total with
{percent(total_marks)}%\n')
f.write(grade(total_marks/count(marks_lst)))
createfile('Student.csv',[student_id,name,roll,batch_id])
# print(f"You can find your report card here :
{path}/ReportCards/{student_id}_{''.join(name.split())}.txt")
openpath=f"{path}/ReportCards/{student_id}_{''.join(name.split())}.txt"
subprocess.run(['start',openpath], shell=True)

ask=input("Do you want to remove this student's data from database now is
the time (Y/N) : ")

ifask.lower()=='n':
ifduplicate('Batch.csv',batch_id,0):
withopen(f'{path}/Batch.csv','r+',newline='') asf:
script=csv.reader(f)
rows=[rowforrowinscript]
foriinrows:
ifbatch_id==i[0]:
rows[rows.index(i)][4]+=f':{student_id}'
f.seek(0)
f.truncate()
writer=csv.writer(f)
writer.writerows(rows)

print("Batch.csv has been updated")
else:
createfile('Batch.csv',[batch_id,batch_name,stream,choice(stream),student
_id])

withopen(f'{path}/Course.csv','r+',newline='') asf:
```

```
script=csv.reader(f)
rows=[rowforrowinscript]
foriinrange(len(rows)):
ifi==0:
pass
else:
try:
rows[i][2]+=f'{student_id}:{marks_lst[i-1]}-'
except:
rows[i].append(f'{student_id}:{marks_lst[i-1]}-')
f.seek(0)
f.truncate()
writer=csv.writer(f)
writer.writerows(rows)
else:
remove(student_id)
subprocess.call("TASKKILL /F /IM notepad.exe", shell=True)
os.remove(openpath)
print('Your details have been successfully removed from the directory')
print('-'*50)
print()

try:
withopen(f'{path}/Department.csv','r+',newline='') asf:
script=csv.reader(f)
rows=[rowforrowinscript]
lst=get_batch()
foriinlst:
forjinrows:
ifi[0:-2]==j[0]:
try:
ifiinj[2]:
pass
else:
rows[rows.index(j)][2]+=f'{i}:'
except:
rows[rows.index(j)].append(f'{i}:')
break
f.seek(0)
```

```
f.truncate()
writer=csv.writer(f)
writer.writerows(rows)

except:
print(student_no,"student's data has been successfully updated in
Department.csv")



#Creation of the Graphs...
print()
print("Give the details Below to see the Batchwise percent Graph")
batch=input("Which batch they are in (e.g. 2022-26) : ")
stream=input("Which Stream are they in (e.g. CSE) : ")
print('Please Close the Figure window after viewing to continue')
batch_id=stream+batch[2:4]

withopen(f'{path}/Batch.csv','r') asf:
reader=csv.reader(f)
batch=[batch[0] forbatchinreader]
batch=batch[1:]

whileTrue:
ifbatch_idinbatch:
batch_graph(batch_id)
break
else:
print(f'details with {batch_id} this Batch ID is not in the directory')
ask=input("Do you want to continue (y/n) : ")
ifask.lower()=='y':
batch=input("Which batch they are in (e.g. 2022-26) : ")
stream=input("Which Stream are they in (e.g. CSE) : ")
batch_id=stream+batch[2:4]
continue
else:
print('OK')
break
print()
```

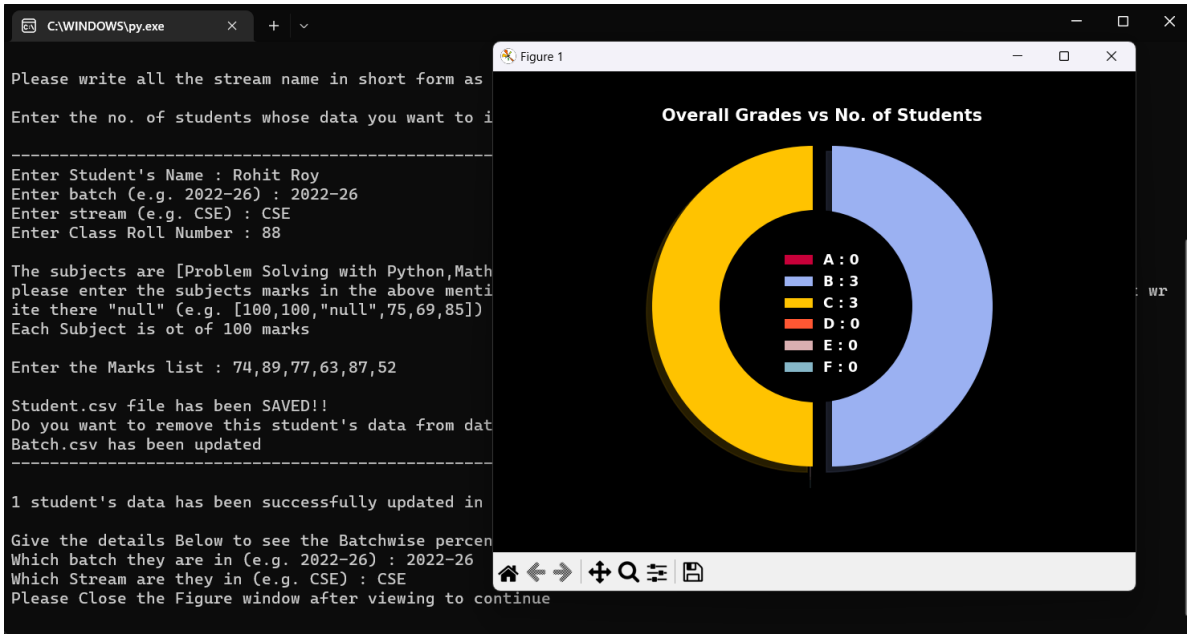
```
print("HERE'S SHOWING THE OVERALL COURSE GRAPH")
print('Please Close the Figure window after viewing to continue')
loading_screen()
course_graph()
print()
print()
print("HERE'S SHOWING The overall Department wise average graph ")
print('Please Close the Figure window after viewing to continue')
loading_screen()
department_graph()
print()
print()

last=input("Press Enter to exit")
subprocess.call("TASKKILL /F /IM notepad.exe", shell=True)
```

1 Output:-

2 PRIMARY INTERFACE

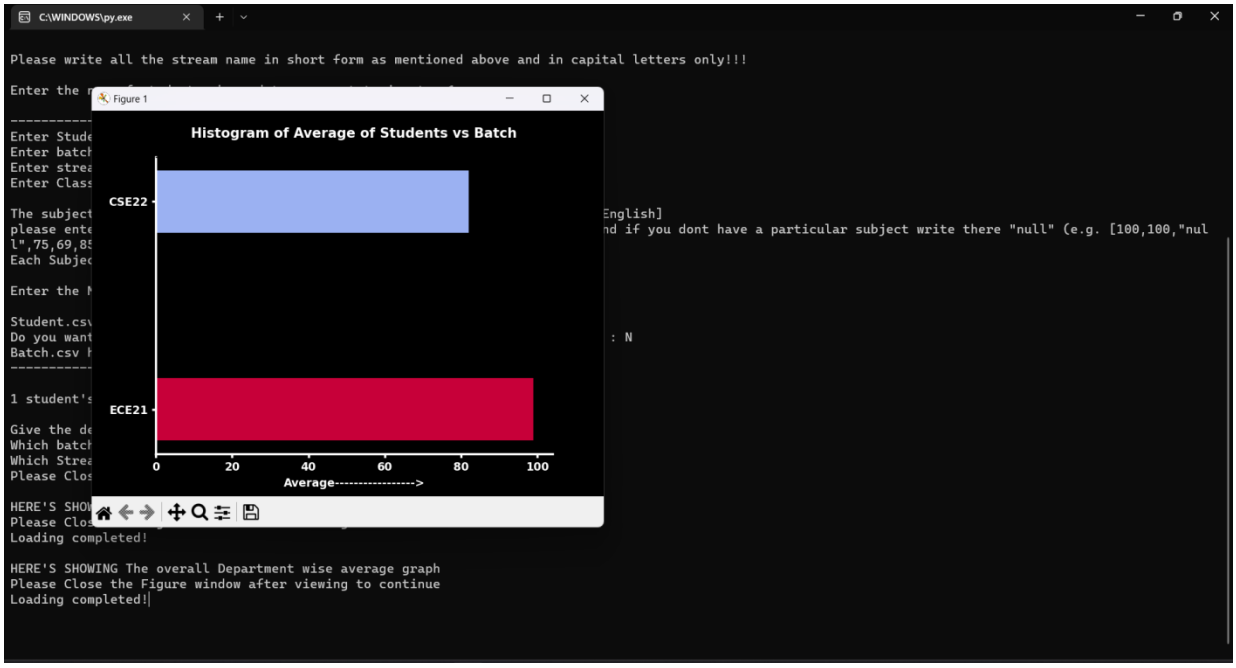
3

Pie-chart of grades of all students in Batch



Histogram of average of students vs batch

16



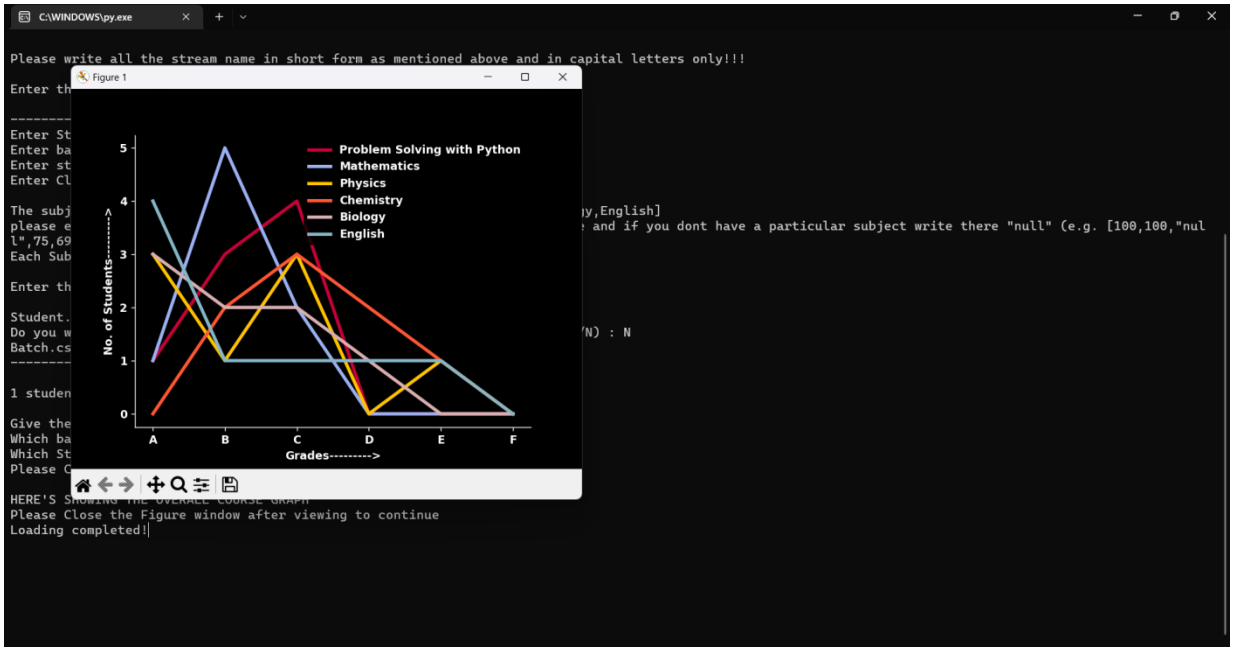
17

18

19

20 **Line plot of no. of students in a department vs. their grades**

21



22

23