TITLE OF THE PROJECT

Submitted by

Name of the Students: Aritra Ghosal
Enrolment number: 12022002018036
Section: F
Class Roll Number: 28
Stream: C.S.B.S
Subject: Programming for Problem Solving
Subject Code: IVC101
Department: Basic Science and Humanities

Under the supervision of
Name of the teachers

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 recommed that the project prepared under our supervision by Aritra Ghosal, entitled Title of the Project 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

Python is a versatile and easy to use language often used in data manipulation. What separates Python from all other languages is its large number of use cases. Whereas Javascript is used for the web, C for systems, R for data, Python can be used for all three and many more. The following project demonstrates a model system run using mainly python.

1.1 Objective

This project attempts to model a small scale database management system utilized by an academic institution. The objective of this project is to learn and demonstrate several python programming concepts including:

- Using python code from other files
- Importing and using third party modules
- Reading and writing text files
- Managing CSV data
- Plotting data
- Building a basic user interface
- Utilizing concepts of Object Oriented Programming

This project also demonstrates general programming concepts such as ER diagrams.

1.2 Organization of the Project

```
.
├── code
│   ├── batch.py
│   ├── course.py
│   ├── department.py
│   ├── examination.py
│   ├── main.py
│   └── student.py
├── databases
│   ├── batch.csv
│   ├── course.csv
│   ├── department.csv
│   └── student.csv
├── fonts
│   ├── CONSOLAB.TTF
│   ├── timesnewroman-bold.ttf
│   ├── timesnewroman-bolditalic.ttf
│   ├── timesnewroman-italic.ttf
│   └── timesnewroman-regular.ttf
├── images
│   ├── … .2 instructions
│   ├── IEM logo.png
│   ├── Project Details.docx
│   └── Project Report Template.docx
├── latex
│   ├── er-diagram.sty
│   ├── er.tex
│   ├── project.tex
│   └── template.tex
├── Makefile
└── outputs
    ├── screenshots
    └── …
```

The **code** directory contains all the python code that is being executed at runtime. **batch.py** is a module that exports functions that operate on a batch. Likewise, **course.py** is a module that exports functions that operate on courses in the database. Same for **department.py**, which is a module that exports functions that operate on a department. **examination.py** exports the **Examination** class that represents an examination being held by the institution. **main.py** is a file with executive permissions which imports all of the above and runs a simple graphical user interface to demonstrate the modules.

The **databases** directory contains all the data in CSV format.

The **fonts** directory contains the fonts required to compile this document.

The **images** directory contains all of the images required tomake the interface.

The **instructions** directory contains all of the raw material to given to build this project.

The **latex** directory contains all of the LaTeX code used to build the project report (this file). **template.tex** sets the default values necessary for the project report. **project.tex** contains the code that is compiled into the project report. It contains sources the outputs and diagrams along with the python code to include in the project report.
er.tex contains the er diagram for the database and **er-diagram.sty** is a third party library used to draw the er diagram. **output.tex** is an automatically generated file which sources all of the plots into the final report.

The **Makefile** contains the build system for the entire project. It specifies the dependencies for each component and runs the commands to create each component. The **Makefile** also contains code that generates the databases and fills them with random data modelling the system as closely as possible. This is the centre point of the entire project, it determines the order and execution of everything else in the project.

The **outputs** directory contains all of the output generated by the python code at runtime. The **screenshots** folder contains the screenshots of the Graphic User Interface.

2 Database Descriptions

Each student in the **student.csv** database has a unique ID, along with a name and a class roll number. Each student is associated with a single batch.
Each batch in **batch.csv** is assigned a unique ID. They also have name and a department they fall under. Each batch has a list of courses and a list of students who appear for the courses.
Each course in **course.csv** has an ID, subject name and a storage of marks obtained by each student appearing for the course.
Each department in **department.csv** has an ID, name and list of batches that worked under that department.

2.1 Database Samples

batch.csv

Batch ID	Batch Name	Department Name	List of Courses	List of Students
CSE00	CSE 2000-2004	CSE	…	…
CSE02	CSE 2002-2006	CSE	…	…
CSE03	CSE 2003-2007	CSE	…	…
CSE04	CSE 2004-2008	CSE	…	…
CSE11	CSE 2011-2015	CSE	…	…

CSE20	CSE 2020-2024	CSE	…	…
CSE21	CSE 2021-2025	CSE	…	…
CSE92	CSE 1992-1996	CSE	…	…
CSE95	CSE 1995-1999	CSE	…	…
CSE98	CSE 1998-2002	CSE	…	…
CSE99	CSE 1999-2003	CSE	…	…
ECE01	ECE 2001-2005	ECE	…	…
ECE03	ECE 2003-2007	ECE	…	…
ECE05	ECE 2005-2009	ECE	…	…
ECE06	ECE 2006-2010	ECE	…	…
ECE11	ECE 2011-2015	ECE	…	…
ECE12	ECE 2012-2016	ECE	…	…
ECE13	ECE 2013-2017	ECE	…	…
ECE14	ECE 2014-2018	ECE	…	…
ECE17	ECE 2017-2021	ECE	…	…
ECE18	ECE 2018-2022	ECE	…	…
ECE20	ECE 2020-2024	ECE	…	…
ECE94	ECE 1994-1998	ECE	…	…
ECE96	ECE 1996-2000	ECE	…	…
ECE97	ECE 1997-2001	ECE	…	…
ECE98	ECE 1998-2002	ECE	…	…
ECE99	ECE 1999-2003	ECE	…	…
IT06	IT 2006-2010	IT	…	…
IT07	IT 2007-2011	IT	…	…
IT08	IT 2008-2012	IT	…	…
IT09	IT 2009-2013	IT	…	…
IT13	IT 2013-2017	IT	…	…
IT14	IT 2014-2018	IT	…	…
IT15	IT 2015-2019	IT	…	…
IT16	IT 2016-2020	IT	…	…
IT18	IT 2018-2022	IT	…	…
IT19	IT 2019-2023	IT	…	…
IT22	IT 2022-2026	IT	…	…
IT89	IT 1989-1993	IT	…	…
IT91	IT 1991-1995	IT	…	…
IT92	IT 1992-1996	IT	…	…
ME03	ME 2003-2007	ME	…	…

ME04	ME 2004-2008	ME	…	…
ME05	ME 2005-2009	ME	…	…
ME08	ME 2008-2012	ME	…	…
ME09	ME 2009-2013	ME	…	…
ME12	ME 2012-2016	ME	…	…
ME15	ME 2015-2019	ME	…	…
ME19	ME 2019-2023	ME	…	…
ME21	ME 2021-2025	ME	…	…
ME22	ME 2022-2026	ME	…	…
ME90	ME 1990-1994	ME	…	…
ME91	ME 1991-1995	ME	…	…
ME92	ME 1992-1996	ME	…	…
ME93	ME 1993-1997	ME	…	…
ME94	ME 1994-1998	ME	…	…
ME98	ME 1998-2002	ME	…	…
CSE15	CSE 2015-2019	CSE	…	…

`course.csv`

Course ID	Course Name	Marks Obtained
CSE00	CSE 2000-2004	CSE
CSE02	CSE 2002-2006	CSE
CSE03	CSE 2003-2007	CSE
CSE04	CSE 2004-2008	CSE
CSE11	CSE 2011-2015	CSE
CSE20	CSE 2020-2024	CSE
CSE21	CSE 2021-2025	CSE
CSE92	CSE 1992-1996	CSE
CSE95	CSE 1995-1999	CSE
CSE98	CSE 1998-2002	CSE
CSE99	CSE 1999-2003	CSE
ECE01	ECE 2001-2005	ECE
ECE03	ECE 2003-2007	ECE
ECE05	ECE 2005-2009	ECE
ECE06	ECE 2006-2010	ECE
ECE11	ECE 2011-2015	ECE
ECE12	ECE 2012-2016	ECE

ECE13	ECE 2013-2017	ECE
ECE14	ECE 2014-2018	ECE
ECE17	ECE 2017-2021	ECE
ECE18	ECE 2018-2022	ECE
ECE20	ECE 2020-2024	ECE
ECE94	ECE 1994-1998	ECE
ECE96	ECE 1996-2000	ECE
ECE97	ECE 1997-2001	ECE
ECE98	ECE 1998-2002	ECE
ECE99	ECE 1999-2003	ECE
IT06	IT 2006-2010	IT
IT07	IT 2007-2011	IT
IT08	IT 2008-2012	IT
IT09	IT 2009-2013	IT
IT13	IT 2013-2017	IT
IT14	IT 2014-2018	IT
IT15	IT 2015-2019	IT
IT16	IT 2016-2020	IT
IT18	IT 2018-2022	IT
IT19	IT 2019-2023	IT
IT22	IT 2022-2026	IT
IT89	IT 1989-1993	IT
IT91	IT 1991-1995	IT
IT92	IT 1992-1996	IT
ME03	ME 2003-2007	ME
ME04	ME 2004-2008	ME
ME05	ME 2005-2009	ME
ME08	ME 2008-2012	ME
ME09	ME 2009-2013	ME
ME12	ME 2012-2016	ME
ME15	ME 2015-2019	ME
ME19	ME 2019-2023	ME
ME21	ME 2021-2025	ME
ME22	ME 2022-2026	ME
ME90	ME 1990-1994	ME
ME91	ME 1991-1995	ME
ME92	ME 1992-1996	ME

ME93	ME 1993-1997	ME
ME94	ME 1994-1998	ME
ME98	ME 1998-2002	ME
CSE15	CSE 2015-2019	CSE

`department.csv`

Department ID	Department Name	List of Batches
CSE	Computer Science and Engineering	…
ECE	Electronics and Communication Engineering	…
IT	Information Technology	…
BA	Business Administration	…

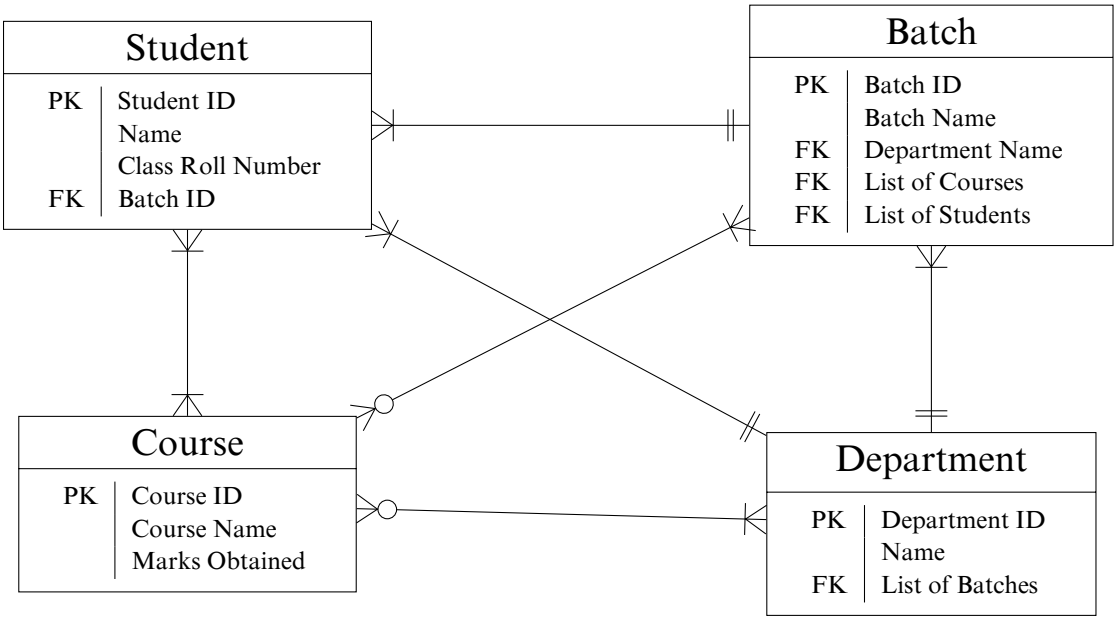
`student.csv`

Student ID	Name	Class Roll No	Batch ID
ME1578	Onkar Kuruvilla	H-69	ME15
ME0542	Rhea Wable	H-43	ME05
ME0308	Alisha Krishnan	G-94	ME03
ME2244	Anay Solanki	A-31	ME22
CSE2021	Indranil Kakar	F-79	CSE20
ME0498	Lakshit Dugar	C-86	ME04
ECE0645	Hridaan Deol	C-18	ECE06
CSE2198	Hrishita Bhargava	F-93	CSE21
IT9244	Saanvi Dua	E-77	IT92
ECE9898	Kimaya Wason	B-05	ECE98
ME9842	Mehul Seshadri	H-12	ME98
CSE1173	Kaira Mandal	E-42	CSE11
ME9294	Dhanuk Garg	A-54	ME92
IT0935	Jivika Baria	B-33	IT09
ECE0520	Urvi Wagle	D-54	ECE05
ME0540	Vardaniya Hayer	C-48	ME05
CSE0331	Rohan Dara	E-11	CSE03
CSE9251	Sara Shah	G-53	CSE92
ECE1131	Jayant Khosla	D-80	ECE11
ECE0339	Yuvraj Subramaniam	E-13	ECE03
IT0800	Zain Tak	C-42	IT08

ECE9839	Advik Anne	D-11	ECE98
IT9213	Onkar Mangal	B-53	IT92
ECE0587	Saira Choudhry	H-64	ECE05
ECE9743	Badal Badami	A-18	ECE97
IT9125	Aarush Lanka	D-26	IT91
ME0933	Ryan Vala	A-69	ME09
ME9479	Adira Mangal	F-66	ME94
ME2131	Samar Agrawal	G-69	ME21
ECE1819	Romil Chowdhury	C-82	ECE18
IT0611	Kiaan Sangha	F-58	IT06
CSE9545	Anika Sabharwal	G-80	CSE95
IT8901	Shalv Guha	C-09	IT89
IT1469	Shamik Dayal	A-66	IT14
ECE1263	Suhana Ghose	A-14	ECE12
ME1241	Charvi Sule	E-75	ME12
ECE9712	Ira Chanda	F-64	ECE97
CSE9940	Siya Upadhyay	C-37	CSE99
IT0932	Aarna Aggarwal	C-80	IT09
IT0755	Rasha Balay	D-95	IT07
ECE1426	Riaan Iyengar	E-08	ECE14
CSE0437	Aayush Singh	A-08	CSE04
IT1818	Yuvraj Bera	B-85	IT18
ME0939	Misha Jani	H-71	ME09
ME0825	Aarush Divan	G-17	ME08
CSE9944	Shayak Hans	B-18	CSE99
ME1500	Oorja Tandon	D-70	ME15
CSE0018	Adah Seth	G-10	CSE00
ECE1707	Samaira Yohannan	A-33	ECE17
ECE0320	Aradhya Wadhwa	H-04	ECE03
IT1670	Sara Swaminathan	H-63	IT16
ECE0569	Siya Tak	A-98	ECE05
ECE0173	Vihaan Ahluwalia	D-12	ECE01
ECE2069	Vardaniya Chander	G-11	ECE20
IT1311	Mishti Comar	C-77	IT13
ME9161	Aarna Sha	C-65	ME91
CSE9810	Shaan Sabharwal	D-79	CSE98
IT9161	Mamooty Hayer	D-21	IT91

IT2220	Rhea Bhattacharyya	G-18	IT22
ME0588	Raunak Chaudry	B-72	ME05
IT1959	Lakshit Sandhu	B-30	IT19
ME0563	Taran Sarraf	H-11	ME05
IT9157	Adah Khurana	D-36	IT91
ME9053	Myra Ratta	B-32	ME90
ECE9634	Damini Kata	B-48	ECE96
ECE9454	Vritika Bhardwaj	D-98	ECE94
ME1508	Mamooty Grewal	A-04	ME15
ECE0348	Nayantara Kanda	E-26	ECE03
IT9174	Shlok Sura	G-66	IT91
CSE9290	Neelofar Ganesh	B-59	CSE92
ME1969	Ivan Wali	F-45	ME19
ME9357	Anvi Warrior	A-80	ME93
CSE0295	Renee Goda	F-82	CSE02
ME1953	Oorja Choudhury	A-79	ME19
ECE9994	Krish Amble	B-81	ECE99
ECE1345	Vanya Subramaniam	E-98	ECE13
IT1607	Neelofar Buch	F-41	IT16
ME0479	Emir Dhawan	E-56	ME04
IT1545	Vivaan Sibal	E-94	IT15
CSE0045	Kartik Joshi	B-22	CSE00

3 E-R Diagram



4 Programs

main.py

```
#!/bin/python3
from tkinter import
 ↪  Tk,Frame,Button,Label,Entry,StringVar,DoubleVar,END
from tkinter.ttk import Notebook
from tkinter.scrolledtext import ScrolledText
from PIL.Image import open as image_open
from PIL.ImageTk import PhotoImage
from texttable import Texttable
from matplotlib.pyplot import show
from re import compile
from functools import partial
#import modules
from student import
 ↪  create_student,update_student,remove_student,report
from course import create_course,course_performance,course_statistics
from batch import
 ↪  create_batch,students,courses,batch_performance,batch_statistics
from department import
 ↪  create_department,batches,batch_averages,department_statistics
from examination import Examination
#---
interface=Tk()
interface.title('Menu')
interface.resizable(False,False)
home_image=PhotoImage(image_open('images/home.png').resize((15,15)))
plus_image=PhotoImage(image_open('images/plus.ico').resize((15,15)))
minus_image=PhotoImage(image_open('images/minus.ico').resize((15,15)))
#button functions
```

```
def retrieve(var):
    val=var.get()
    if val=='':raise Exception('Empty Entry')
    var.set('')
    return val
def add_marks():
    global course_row
    course_marks_input.append((
        roll_no:=StringVar(),
        student:=StringVar(),
        marks:=DoubleVar()
    ))
    marks_entries.extend((
        roll_entry:=Entry(course_creation,textvariable=roll_no),
        name_entry:=Entry(course_creation,textvariable=student),
        marks_entry:=Entry(course_creation,textvariable=marks)
    ))
    roll_entry.grid(column=0,row=course_row)
    name_entry.grid(column=1,row=course_row)
    marks_entry.grid(column=2,row=course_row)
    course_row+=1
    marks_add.grid_forget();marks_add.grid(column=1,row=course_row,st
     ↪  icky='e')
    marks_remove.grid_forget();marks_remove.grid(column=2,row=course_
     ↪  row,sticky='w')
    course_create.grid_forget();course_create.grid(columnspan=3,row=c
     ↪  ourse_row+1,sticky='nsew')
def add_courses():
    global students_row,courses_row
    batch_courses_input.append(course_id:=StringVar())
    course_entries.append(course_entry:=Entry(batch_creation,textvari
     ↪  able=course_id))
    course_entry.grid(column=0,row=courses_row)
    courses_row+=1;students_row+=1
    courses_add.grid_forget();courses_add.grid(column=0,row=courses_r
     ↪  ow,sticky='e')
    courses_remove.grid_forget();courses_remove.grid(column=1,row=cou
     ↪  rses_row,sticky='w')
    students_heading.grid_forget();students_heading.grid(columnspan=2
     ↪  ,row=courses_row+1,sticky='nsew')
    for i,student_entry in enumerate(student_entries,courses_row+2):
        student_entry.grid_forget()
        student_entry.grid(column=0,row=i)
    students_add.grid_forget();students_add.grid(column=0,row=student
     ↪  s_row,sticky='e')
    students_remove.grid_forget();students_remove.grid(column=1,row=s
     ↪  tudents_row,sticky='w')
    batch_create.grid_forget();batch_create.grid(columnspan=2,row=stu
     ↪  dents_row+1,sticky='nsew')
def add_students():
    global students_row
    batch_students_input.append(student_id:=StringVar())
    student_entries.append(student_entry:=Entry(batch_creation,textva
     ↪  riable=student_id))
```

```
    student_entry.grid(column=0,row=students_row)
    students_row+=1
    students_add.grid_forget();students_add.grid(column=0,row=student
     ↪  s_row,sticky='e')
    students_remove.grid_forget();students_remove.grid(column=1,row=s
     ↪  tudents_row,sticky='w')
    batch_create.grid_forget();batch_create.grid(columnspan=2,row=stu
     ↪  dents_row+1,sticky='nsew')
def add_batches():
    global batches_row
    department_batch_input.append(batch_id:=StringVar())
    batch_entries.append(batch_entry:=Entry(department_creation,textv
     ↪  ariable=batch_id))
    batch_entry.grid(columnspan=2,row=batches_row)
    batches_row+=1
    batch_add.grid_forget();batch_add.grid(column=0,row=batches_row,s
     ↪  ticky='e')
    batch_remove.grid_forget();batch_remove.grid(column=1,row=batches
     ↪  _row,sticky='w')
    department_create.grid_forget();department_create.grid(columnspan
     ↪  =2,row=batches_row+1,sticky='nsew')
def add_batches_for_exam():
    global exam_batch_row
    exam_batch_input.append(batch_id:=StringVar())
    exam_batch_entries.append(batch_entry:=Entry(hold_exam,textvariab
     ↪  le=batch_id))
    batch_entry.grid(columnspan=2,row=exam_batch_row)
    exam_batch_row+=1
    exam_batches_add.grid_forget();exam_batches_add.grid(column=0,row
     ↪  =exam_batch_row,sticky='e')
    exam_batches_remove.grid_forget();exam_batches_remove.grid(column
     ↪  =1,row=exam_batch_row,sticky='w')
    start_exam.grid_forget();start_exam.grid(columnspan=2,row=exam_ba
     ↪  tch_row+1,sticky='nsew')
course_id_pattern=compile('^C0[0-9]{2}$')
batch_id_pattern=compile('^[A-Z]+[0-9]{2}$')
batch_name_pattern=compile('^[A-Z]+ (19|20)[0-9]{2}-(19|20)[0-9]{2}$')
def course_resolve(func):
    course=retrieve(course_name)
    if course_id_pattern.search(course)==None:
        return func(course_name=course)
    else:
        return func(course_id=course)
def clean_entries(entry_list):
    for i in entry_list:i.grid_forget()
    entry_list.clear()
def view_course_performance():
    sheet=Texttable()
    sheet.set_cols_align(('l','l','r'))
    sheet.add_row(['Class Roll','Name','Marks'])
    for roll,name,marks in course_resolve(course_performance):sheet.a
     ↪  dd_row([roll,name,marks])
    show_performance.configure(state='normal')
    show_performance.delete(0.0,END)
```

```
    show_performance.insert(0.0,sheet.draw())
    show_performance.configure(state='disabled')
def batch_resolve(func):
    batch=retrieve(batch_name)
    if batch_id_pattern.search(batch)!=None:
        return func(batch_id=batch)
    elif batch_name_pattern.search(batch)!=None:
        return func(batch_name=batch)
    else:
        raise Error('Not a valid batch name')
def view_batch_list(func,widget):
    widget.configure(state='normal')
    widget.delete(0.0,END)
    widget.insert(0.0,'\n'.join(batch_resolve(func)))
    widget.configure(state='disabled')
def view_student_average():
    sheet=Texttable()
    sheet.set_cols_align(['l','l','r'])
    sheet.add_row(['Student','Name','Percentage'])
    for roll,name,percentage in batch_resolve(batch_performance):
        sheet.add_row([roll,name,f'{percentage:.2f}%'])
    show_student_average.configure(state='normal')
    show_student_average.delete(0.0,END)
    show_student_average.insert(0.0,sheet.draw())
    show_student_average.configure(state='disabled')
def view_department_batches():
    show_batches.configure(state='normal')
    show_batches.delete(0.0,END)
    show_batches.insert(0.0,'\n'.join(batches(department_id=retrieve(
    ↪  department_id))))
    show_batches.configure(state='disabled')
def view_department_performance():
    sheet=Texttable()
    sheet.set_cols_align(['l','r'])
    sheet.add_row(['Batch','Average'])
    for batch,percentage in
    ↪  batch_averages(department_id=retrieve(department_id)):
        sheet.add_row([batch,f'{percentage:.2f}%'])
    show_averages.configure(state='normal')
    show_averages.delete(0.0,END)
    show_averages.insert(0.0,sheet.draw())
    show_averages.configure(state='disabled')
def exam():
    def get_marks(papers):
        for paper in papers:
            batch_label.configure(text=paper.batch)
            student_label.configure(text=paper.roll_no)
            course_label.configure(text=f'Marks in {paper.course}')
            accept_marks.wait_variable(marks_input)
            yield paper,float(marks_input.get())
    def view_student_performance():
        chart=Texttable()
        chart.set_cols_align(('l','r'))
        chart.add_row(['Student ID','Average'])
```

```
            for student,average in
             ↪  examination.student_performance.items():
                chart.add_row([student,average])
            show_exam_performance.configure(state='normal')
            show_exam_performance.delete(0.0,END)
            show_exam_performance.insert(0.0,chart.draw())
            show_exam_performance.configure(state='disabled')
        exam_title=f'{retrieve(exam_name)} Exam'
        examination=Examination(exam_title,*[i.get() for i in
         ↪  exam_batch_input])
        interface.title(exam_title)
        exam_batch_input.clear()
        clean_entries(exam_batch_entries)
        hold_exam.pack_forget()
        provide_marks.pack()
        examination.enter_marks(get_marks(examination.take_exam()))
        #After exam
        post_exam=Frame(interface)
        Button(post_exam,image=home_image,command=lambda:(post_exam.pack_
         ↪  forget(),menu.pack())).grid(row=0)
        Button(post_exam,text='View Exam
         ↪  Results',command=view_student_performance).grid(row=1)
        Button(post_exam,text='View Exam Scatter Plot',command=lambda:(ex
         ↪  amination.statistics(),show())).grid(row=2)
        (show_exam_performance:=ScrolledText(post_exam)).grid(row=3);show
         ↪  _exam_performance.configure(state='disabled')
        provide_marks.pack_forget()
        post_exam.pack()
        interface.title('Results')
    #menu
    menu=Notebook(interface)
    menu.add(student_tab:=Frame(menu),text='Student')
    menu.add(course_tab:=Frame(menu),text='Course')
    menu.add(batch_tab:=Frame(menu),text='Batch')
    menu.add(department_tab:=Frame(menu),text='Department')
    menu.add(exam_tab:=Frame(menu),text='Exam')
        #student
    Button(student_tab,text='Create Student',command=lambda:(menu.pack_fo
        rget(),student_creation.pack(),interface.title('Create
     ↪  Student'))).grid(column=0,row=0,sticky='nsew')
    Button(student_tab,text='Update Student',command=lambda:(menu.pack_fo
        rget(),student_update.pack(),interface.title('Update
     ↪  Student'))).grid(column=1,row=0,sticky='nsew')
    Button(student_tab,text='Remove Student',command=lambda:(menu.pack_fo
        rget(),student_removal.pack(),interface.title('Remove
     ↪  Student'))).grid(column=0,row=1,sticky='nsew')
    Button(student_tab,text='Generate Report',command=lambda:(menu.pack_f
        orget(),report_generation.pack(),interface.title('Generate
     ↪  Report'))).grid(column=1,row=1,sticky='nsew')
        #course
    Button(course_tab,text='Create Course',command=lambda:(menu.pack_forg
        et(),course_creation.pack(),interface.title('Create
     ↪  Course'))).grid(column=0,row=0,sticky='nsew')
```

```
Button(course_tab,text='View Course Performance',command=lambda:(menu
    .pack_forget(),course_perform.pack(),interface.title('Course
 ↪  Performance'))).grid(column=1,row=0,sticky='nsew')
Button(course_tab,text='Course Statistics',command=lambda:(menu.pack_
    forget(),course_statisticize.pack(),interface.title('Course
 ↪  Statistics'))).grid(columnspan=2,row=1,sticky='nsew')
    #batch
Button(batch_tab,text='Create Batch',command=lambda:(menu.pack_forget
    (),batch_creation.pack(),interface.title('Create
 ↪  Batch'))).grid(column=0,row=0,sticky='nsew')
Button(batch_tab,text='View Batch Students',command=lambda:(menu.pack
    _forget(),batch_students.pack(),interface.title('Batch
 ↪  Students'))).grid(column=1,row=0,sticky='nsew')
Button(batch_tab,text='View Batch Courses',command=lambda:(menu.pack_
    forget(),batch_courses.pack(),interface.title('Batch
 ↪  Courses'))).grid(column=0,row=1,sticky='nsew')
Button(batch_tab,text='View Batch Performance',command=lambda:(menu.p
    ack_forget(),batch_performed.pack(),interface.title('Batch
 ↪  Performance'))).grid(column=1,row=1,sticky='nsew')
Button(batch_tab,text='Batch Pie Chart',command=lambda:(menu.pack_for
    get(),batch_statisticize.pack(),interface.title('Batch Pie
 ↪  Chart'))).grid(columnspan=2,row=2,sticky='nsew')
    #department
Button(department_tab,text='Create Department',command=lambda:(menu.p
    ack_forget(),department_creation.pack(),interface.title('Create
 ↪  Department'))).grid(column=0,row=0,sticky='nsew')
Button(department_tab,text='View Department
    Batches',command=lambda:(menu.pack_forget(),department_batches.pa
    ck(),interface.title('Department
 ↪↪  Batches'))).grid(column=1,row=0,sticky='nsew')
Button(department_tab,text='View Department
    Performance',command=lambda:(menu.pack_forget(),department_perfor
    mance.pack(),interface.title('Department
 ↪↪  Performance'))).grid(column=0,row=1,sticky='nsew')
Button(department_tab,text='Department Line
    Plot',command=lambda:(menu.pack_forget(),department_statisticize.
    pack(),interface.title('Department Line
 ↪↪  Plot'))).grid(column=1,row=1,sticky='nsew')
    #examination
Button(exam_tab,text='Take Exam',command=lambda:(menu.pack_forget(),h
    old_exam.pack(),interface.title('Hold
 ↪  Exam'))).grid(columnspan=2,sticky='nsew')
#entry variables
student_id=StringVar();student_name=StringVar();roll_no=StringVar();b
 ↪  atch_id=StringVar()
course_id=StringVar();course_name=StringVar()
batch_id=StringVar();batch_name=StringVar();
department_id=StringVar();department_name=StringVar()
exam_name=StringVar();marks_input=DoubleVar()
#create Student
student_creation=Frame(interface)
```

```
Button(student_creation,image=home_image,command=lambda:(student_crea
    tion.pack_forget(),menu.pack(),interface.title('Menu'))).grid(col
 ↪  umn=0,row=0,sticky='e')
Label(student_creation,text='Student
 ↪  ID:').grid(column=0,row=1,sticky='e')
Entry(student_creation,textvariable=student_id).grid(column=1,row=1,s
 ↪  tick='w')
Label(student_creation,text='Student
 ↪  Name:').grid(column=0,row=2,sticky='e')
Entry(student_creation,textvariable=student_name).grid(column=1,row=2
 ↪  ,stick='w')
Label(student_creation,text='Class Roll
 ↪  Number').grid(column=0,row=3,sticky='e')
Entry(student_creation,textvariable=roll_no).grid(column=1,row=3,stic
 ↪  k='w')
Label(student_creation,text='Batch
 ↪  ID:').grid(column=0,row=4,sticky='e')
Entry(student_creation,textvariable=batch_id).grid(column=1,row=4,sti
 ↪  ck='w')
Button(student_creation,text='Create',command=lambda:
    create_student(
        student_id=retrieve(student_id),
        name=retrieve(student_name),
        class_roll_no=retrieve(roll_no),
        batch=retrieve(batch_id)
    )
).grid(row=5,columnspan=2,sticky='nsew')
#update Student
student_update=Frame(interface)
Button(student_update,image=home_image,command=lambda:(student_update
    .pack_forget(),menu.pack(),interface.title('Menu'))).grid(column=
 ↪  0,row=0,sticky='e')
Label(student_update,text='Student
 ↪  ID:').grid(column=0,row=1,sticky='e')
Entry(student_update,textvariable=student_id).grid(column=1,row=1,sti
 ↪  ck='w')
Label(student_update,text='Student
 ↪  Name:').grid(column=0,row=2,sticky='e')
Entry(student_update,textvariable=student_name).grid(column=1,row=2,s
 ↪  tick='w')
Label(student_update,text='Class Roll
 ↪  Number').grid(column=0,row=3,sticky='e')
Entry(student_update,textvariable=roll_no).grid(column=1,row=3,stick=
 ↪  'w')
Button(student_update,text='Update',command=lambda:
    update_student(
        student_id=retrieve(student_id),
        name=retrieve(student_name),
        class_roll_no=retrieve(roll_no)
    )
).grid(row=4,columnspan=2,sticky='nsew')
#remove Student
student_removal=Frame(interface)
```

```
Button(student_removal,image=home_image,command=lambda:(student_removal.pack_forget(),menu.pack(),interface.title('Menu'))).grid(column=0,row=0,sticky='e')
Label(student_removal,text='Student ID:').grid(column=0,row=1,sticky='e')
Entry(student_removal,textvariable=student_id).grid(column=1,row=1,stick='w')
Button(student_removal,text='Remove',command=lambda:remove_student(retrieve(student_id))).grid(row=2,columnspan=2,sticky='nsew')
#report Student
report_generation=Frame(interface)
Button(report_generation,image=home_image,command=lambda:(report_generation.pack_forget(),menu.pack(),interface.title('Menu'))).grid(column=0,row=0,sticky='e')
Label(report_generation,text='Student ID:').grid(column=0,row=1,sticky='e')
Entry(report_generation,textvariable=student_id).grid(column=1,row=1,stick='w')
Button(report_generation,text='Generate Report',command=lambda:
    report(retrieve(student_id))
).grid(row=2,columnspan=2,sticky='nsew')
#create Course
course_creation=Frame(interface)
course_marks_input=[]
marks_entries=[]
Button(course_creation,image=home_image,command=lambda:(course_creation.pack_forget(),menu.pack(),interface.title('Menu'))).grid(column=0,row=0,sticky='e')
Label(course_creation,text='Course ID:').grid(column=0,columnspan=2,row=1,sticky='e')
Entry(course_creation,textvariable=course_id).grid(column=2,row=1,stick='w')
Label(course_creation,text='Course Name').grid(column=0,columnspan=2,row=2,sticky='e')
Entry(course_creation,textvariable=course_name).grid(column=2,row=2,sticky='w')
Label(course_creation,text='Class Roll No').grid(column=0,row=3,sticky='nsew')
Label(course_creation,text='Student Name').grid(column=1,row=3,sticky='nsew')
Label(course_creation,text='Marks').grid(column=2,row=3,sticky='nsew')
course_row=4
(marks_add:=Button(course_creation,image=plus_image,command=add_marks)).grid(column=1,row=4,sticky='e')
(marks_remove:=Button(course_creation,image=minus_image,command=lambda:(course_marks_input.pop(),marks_entries.pop().grid_forget(),marks_entries.pop().grid_forget()))).grid(column=2,row=4,sticky='w')
(course_create:=Button(course_creation,text='Create',command=lambda:(
    create_course(
        course_id=retrieve(course_id),
        course_name=retrieve(course_name),
        marks=[{
            'roll number':retrieve(roll_val),
```

```
            'name':retrieve(name_val),
            'marks':retrieve(marks_val)
        } for roll_val,name_val,marks_val in course_marks_input]
    ),
    [entry.grid_forget() for entry in marks_entries],
    marks_entries.clear(),
    course_marks_input.clear()
))).grid(columnspan=3,row=5,sticky='nsew')
#performance Course
course_perform=Frame(interface)
Button(course_perform,image=home_image,command=lambda:(course_perform
    .pack_forget(),menu.pack(),interface.title('Menu'))).grid(column=
 ↪  0,row=0,sticky='e')
Label(course_perform,text='Course:').grid(column=0,row=1,sticky='e')
Entry(course_perform,textvariable=course_name).grid(column=1,row=1,st
 ↪  icky='w')
Button(course_perform,text='View',command=view_course_performance).gr
 ↪  id(columnspan=2,row=2,sticky='nsew')
show_performance=ScrolledText(course_perform,width=60,height=10);show
    _performance.grid(columnspan=2,row=3,sticky='nsew');show_performa
 ↪  nce.configure(state='disabled')
#statisticize course
course_statisticize=Frame(interface)
Button(course_statisticize,image=home_image,command=lambda:(course_st
    atisticize.pack_forget(),menu.pack(),interface.title('Menu'))).gr
 ↪  id(column=0,row=0,sticky='e')
Label(course_statisticize,text='Course:').grid(column=0,row=1,sticky=
 ↪  'e')
Entry(course_statisticize,textvariable=course_name).grid(column=1,row
 ↪  =1,sticky='w')
Button(course_statisticize,text='View',command=lambda:(course_resolve
    (course_statistics),show())).grid(columnspan=2,row=2,sticky='nsew
 ↪  ')
#create Batch
batch_creation=Frame(interface)
courses_row=5;students_row=7
batch_courses_input=[];batch_students_input=[]
course_entries=[];student_entries=[]
Button(batch_creation,image=home_image,command=lambda:(batch_creation
    .pack_forget(),menu.pack(),interface.title('Menu'))).grid(column=
 ↪  0,row=0,sticky='e')
Label(batch_creation,text='Batch ID:').grid(column=0,row=1,sticky='e')
Entry(batch_creation,textvariable=batch_id).grid(column=1,row=1,stick
 ↪  y='w')
Label(batch_creation,text='Batch
 ↪  Name:').grid(column=0,row=2,sticky='e')
Entry(batch_creation,textvariable=batch_name).grid(column=1,row=2,sti
 ↪  cky='w')
Label(batch_creation,text='Department
 ↪  ID:').grid(column=0,row=3,sticky='e')
Entry(batch_creation,textvariable=department_name).grid(column=1,row=
 ↪  3,sticky='w')
```

```
Label(batch_creation,text='Courses').grid(columnspan=2,row=4,sticky='nsew')
(courses_add:=Button(batch_creation,image=plus_image,command=add_courses)).grid(column=0,row=5,sticky='e')
(courses_remove:=Button(batch_creation,image=minus_image,command=lambda:(course_entries.pop().grid_forget(),batch_courses_input.pop()))).grid(column=1,row=5,sticky='w')
(students_heading:=Label(batch_creation,text='Students')).grid(columnspan=2,row=6,sticky='nsew')
(students_add:=Button(batch_creation,image=plus_image,command=add_students)).grid(column=0,row=7,sticky='e')
(students_remove:=Button(batch_creation,image=minus_image,command=lambda:(student_entries.pop().grid_forget(),batch_students_input.pop()))).grid(column=1,row=7,sticky='w')
(batch_create:=Button(batch_creation,text='Create',command=lambda:(
    create_batch(
        batch_id=retrieve(batch_id),
        batch_name=retrieve(batch_name),
        department_name=retrieve(department_name),
        courses=[retrieve(i) for i in batch_courses_input],
        students=[retrieve(i) for i in batch_students_input]
    ),
    batch_courses_input.clear(),
    batch_students_input.clear(),
    clean_entries(course_entries),
    clean_entries(student_entries)
))).grid(columnspan=2,row=8,sticky='nsew')
#batch students
batch_students=Frame(interface)
Button(batch_students,image=home_image,command=lambda:(batch_students.pack_forget(),menu.pack(),interface.title('Menu'))).grid(column=0,row=0,sticky='e')
Label(batch_students,text='Batch:').grid(column=0,row=1,sticky='e')
Entry(batch_students,textvariable=batch_name).grid(column=1,row=1,sticky='w')
(show_students:=ScrolledText(batch_students,width=10,height=3)).grid(columnspan=2,row=3,sticky='nsew');show_students.configure(state='disabled')
Button(batch_students,text='View',command=partial(view_batch_list,students,show_students)).grid(columnspan=2,row=2,sticky='nsew')
#batch courses
batch_courses=Frame(interface)
Button(batch_courses,image=home_image,command=lambda:(batch_courses.pack_forget(),menu.pack(),interface.title('Menu'))).grid(column=0,row=0,sticky='e')
Label(batch_courses,text='Batch:').grid(column=0,row=1,sticky='e')
Entry(batch_courses,textvariable=batch_name).grid(column=1,row=1,sticky='w')
(show_courses:=ScrolledText(batch_courses,width=10,height=5)).grid(columnspan=2,row=3,sticky='nsew');show_courses.configure(state='disabled')
Button(batch_courses,text='View',command=partial(view_batch_list,courses,show_courses)).grid(columnspan=2,row=2,sticky='nsew')
```

```
#batch performance
batch_performed=Frame(interface)
Button(batch_performed,image=home_image,command=lambda:(batch_perform
    ed.pack_forget(),menu.pack(),interface.title('Menu'))).grid(colum
 ↪  n=0,row=0,sticky='e')
Label(batch_performed,text='Batch:').grid(column=0,row=1,sticky='e')
Entry(batch_performed,textvariable=batch_name).grid(column=1,row=1,st
 ↪  icky='w')
Button(batch_performed,text='View',command=view_student_average).grid
 ↪  (columnspan=2,row=2,sticky='nsew')
(show_student_average:=ScrolledText(batch_performed,width=50,height=1
    0)).grid(columnspan=2,row=3,sticky='nsew');show_student_average.c
 ↪  onfigure(state='disabled')
#batch statisticize
batch_statisticize=Frame(interface)
Button(batch_statisticize,image=home_image,command=lambda:(batch_stat
    isticize.pack_forget(),menu.pack(),interface.title('Menu'))).grid
 ↪  (column=0,row=0,sticky='e')
Label(batch_statisticize,text='Batch:').grid(column=0,row=1,sticky='e
 ↪  ')
Entry(batch_statisticize,textvariable=batch_name).grid(column=1,row=1
 ↪  ,sticky='w')
Button(batch_statisticize,text='View',command=lambda:(batch_resolve(b
 ↪  atch_statistics),show())).grid(columnspan=2,row=2,sticky='nsew')
#create Department
department_creation=Frame(interface)
batch_entries=[]
department_batch_input=[]
Button(department_creation,image=home_image,command=lambda:(departmen
    t_creation.pack_forget(),menu.pack(),interface.title('Menu'))).gr
 ↪  id(column=0,row=0,sticky='e')
Label(department_creation,text='Department
 ↪  ID:').grid(column=0,row=1,sticky='e')
Entry(department_creation,textvariable=department_id).grid(column=1,r
 ↪  ow=1,sticky='w')
Label(department_creation,text='Department
 ↪  Name:').grid(column=0,row=2,sticky='e')
Entry(department_creation,textvariable=department_name).grid(column=1
 ↪  ,row=2,sticky='w')
Label(department_creation,text='Batches').grid(columnspan=2,row=3,sti
 ↪  cky='nsew')
batches_row=4
(batch_add:=Button(department_creation,image=plus_image,command=add_b
 ↪  atches)).grid(column=0,row=4,sticky='e')
(batch_remove:=Button(department_creation,image=minus_image,command=l
    ambda:(batch_entries.pop().grid_forget(),department_batch_input.p
 ↪  op()))).grid(column=1,row=4,sticky='w')
(department_create:=Button(department_creation,text='Create',command=
 ↪  lambda:(
    create_department(
        department_id=retrieve(department_id),
        department_name=retrieve(department_name),
```

```
        batches=[retrieve(i) for i in department_batch_input]
    ),
    department_batch_input.clear(),
    clean_entries(batch_entries)
))).grid(columnspan=2,row=5,sticky='nsew')
#department batches
department_batches=Frame(interface)
Button(department_batches,image=home_image,command=lambda:(department
    _batches.pack_forget(),menu.pack(),interface.title('Menu'))).grid
 ↪  (column=0,row=0,sticky='e')
Label(department_batches,text='Department
 ↪  ID:').grid(column=0,row=1,sticky='e')
Entry(department_batches,textvariable=department_id).grid(column=1,ro
 ↪  w=1,sticky='w')
Button(department_batches,text='View',command=view_department_batches
 ↪  ).grid(columnspan=2,row=2,sticky='nsew')
(show_batches:=ScrolledText(department_batches,width=10,height=8)).gr
    id(columnspan=2,row=3,sticky='nsew');show_batches.configure(state
 ↪  ='disabled')
#department performance
department_performance=Frame(interface)
Button(department_performance,image=home_image,command=lambda:(depart
    ment_performance.pack_forget(),menu.pack(),interface.title('Menu'
 ↪  ))).grid(column=0,row=0,sticky='e')
Label(department_performance,text='Department
 ↪  ID:').grid(column=0,row=1,sticky='e')
Entry(department_performance,textvariable=department_id).grid(column=
 ↪  1,row=1,sticky='w')
Button(department_performance,text='View',command=view_department_per
 ↪  formance).grid(columnspan=2,row=2,sticky='nsew')
(show_averages:=ScrolledText(department_performance,width=10,height=1
    0)).grid(columnspan=2,row=3,sticky='nsew');show_averages.configur
 ↪  e(state='disabled')
#department statisticize
department_statisticize=Frame(interface)
Button(department_statisticize,image=home_image,command=lambda:(depar
    tment_statisticize.pack_forget(),menu.pack(),interface.title('Men
 ↪  u'))).grid(column=0,row=0,sticky='e')
Label(department_statisticize,text='Department
 ↪  ID:').grid(column=0,row=1,sticky='e')
Entry(department_statisticize,textvariable=department_id).grid(column
 ↪  =1,row=1,sticky='w')
Button(department_statisticize,text='View',command=lambda:(
    department_statistics(department_id=retrieve(department_id)),
    show()
)).grid(columnspan=2,row=2,sticky='nsew')
#hold exam
exam_batch_input=[]
exam_batch_entries=[]
hold_exam=Frame(interface)
Button(hold_exam,image=home_image,command=lambda:(hold_exam.pack_forg
    et(),menu.pack(),interface.title('Menu'))).grid(column=0,row=0,st
 ↪  icky='e')
```

```
Label(hold_exam,text='Examination:').grid(column=0,row=1,sticky='e')
Entry(hold_exam,textvariable=exam_name).grid(column=1,row=1,sticky='w
 ↪  ')
Label(hold_exam,text='Batches').grid(columnspan=2,row=2,sticky='nsew')
exam_batch_row=3
(exam_batches_add:=Button(hold_exam,image=plus_image,command=add_batc
 ↪  hes_for_exam)).grid(column=0,row=3,sticky='e')
(exam_batches_remove:=Button(hold_exam,image=minus_image,command=lamb
    da:(exam_batch_input.pop(),exam_batch_entries.pop().grid_forget()
 ↪  ))).grid(column=1,row=3,sticky='w')
(start_exam:=Button(hold_exam,text='Start
 ↪  Exam',command=exam)).grid(columnspan=2,row=4,sticky='nsew')
#take exam
provide_marks=Frame(interface)
Label(provide_marks,text='Batch: ').grid(column=0,row=0)
Label(provide_marks,text='Student: ').grid(column=0,row=1)
(batch_label:=Label(provide_marks)).grid(column=1,row=0)
(student_label:=Label(provide_marks)).grid(column=1,row=1)
(course_label:=Label(provide_marks)).grid(column=0,row=2)
(exam_marks_entry:=Entry(provide_marks)).grid(column=1,row=2)
(accept_marks:=Button(provide_marks,text='Enter',command=lambda:(mark
    s_input.set(exam_marks_entry.get()),exam_marks_entry.delete(0,END
 ↪  )))).grid(columnspan=2,row=3)
#---
menu.pack()
interface.mainloop()
```

student.py

```
from csv import writer,reader
from texttable import Texttable
def create_student(**kwargs):
    batch_id=kwargs['batch']
    student_id=kwargs['student_id']
    with open('databases/student.csv','a') as csvfile:
        writer(csvfile).writerow([
            student_id,
            kwargs['name'],
            kwargs['class_roll_no'],
            batch_id
        ])
    rows=[]
    with open('databases/batch.csv','r') as csvfile:
        for row in reader(csvfile):
            if row[0]==batch_id:
                row[4]+=f':{student_id}'
            rows.append(row)
    with open('databases/batch.csv','w') as csvfile:
        db=writer(csvfile)
        for row in rows:
            db.writerow(row)
def update_student(**kwargs):#update by student id
    rows=[]
    EXIT_CODE=1
```

```
    with open('databases/student.csv','r') as csvfile:
        db=reader(csvfile)
        for row in db:
            if row[0]==kwargs['student_id']:
                EXIT_CODE=0
                rows.append([
                    row[0],
                    kwargs['name'] if 'name' in kwargs else row[1],
                    kwargs['class_roll_no'] if 'class_roll_no' in
                     ↪  kwargs else row[2],
                    kwargs['student_id'][:-2]
                ])
                break
            rows.append(row)
        for row in db:rows.append(row)#add remaining
    with open('databases/student.csv','w') as csvfile:#update file
        db=writer(csvfile)
        for row in rows:db.writerow(row)
    return EXIT_CODE
def remove_student(student_id):#remove by student id
    rows=[]
    EXIT_CODE=1
    with open('databases/student.csv','r') as csvfile:
        db=reader(csvfile)
        for row in db:
            if row[0]==student_id:#found
                batch_id=row[3]
                EXIT_CODE=0
                break
            rows.append(row)
        for row in db:rows.append(row)#add remaining
    with open('databases/student.csv','w') as csvfile:#update file
        db=writer(csvfile)
        for row in rows:db.writerow(row)
    if EXIT_CODE==1:return 1#student not found
    rows=[]
    empty_batch=False
    with open('databases/batch.csv','r') as csvfile:
        db=reader(csvfile)
        for row in db:
            if row[0]==batch_id:
                students=row[4].split(':')
                students.remove(student_id)
                courses=row[3].split(':')
                if len(students)==0:
                    empty_batch=True
                    department_name=row[2]
                else:
                    row[4]=':'.join(students)
                    rows.append(row)
                break
            rows.append(row)
        for row in db:rows.append(row)
    with open('databases/batch.csv','w') as csvfile:
        db=writer(csvfile)
        for row in rows:db.writerow(row)
    rows=[]
```

```
    with open('databases/course.csv','r') as csvfile:
        db=reader(csvfile)
        for row in db:
            if row[0] in courses:
                try:
                    marks=row[2]
                    a=marks.index(student_id)
                    b=marks.find('-',a)
                    row[2]=marks[:a-1]+marks[b:]
                except ValueError:#no marks given
                    continue
            rows.append(row)
    with open('databases/course.csv','w') as csvfile:
        db=writer(csvfile)
        for row in rows:db.writerow(row)
    if not empty_batch:return 0
    rows=[]
    with open('databases/department.csv','r') as csvfile:
        db=reader(csvfile)
        for row in db:
            if row[0]==department_name:
                batches=row[2].split(':')
                batches.remove(batch_id)
                row[2]=':'.join(batches)
                rows.append(row)
                break
            rows.append(row)
        for row in db:rows.append(row)
    with open('databases/department.csv','w') as csvfile:
        db=writer(csvfile)
        for row in rows:db.writerow(row)
def report(student_id):
    def grade(marks):
        if marks>=90:grade='A'
        elif marks>=80:grade='B'
        elif marks>=70:grade='C'
        elif marks>=60:grade='D'
        elif marks>=50:grade='E'
        else: return 'F','Failed'
        return (grade,'Passed')
    EXIT_CODE=1
    with open('databases/student.csv') as csvfile:
        db=reader(csvfile)
        for row in db:
            if row[0]==student_id:
                _,name,roll,batch_id=row
                EXIT_CODE=0
                break
    if EXIT_CODE==1:return 1
    with open('databases/batch.csv') as csvfile:
        db=reader(csvfile)
        for row in db:
            if row[0]==batch_id:
                exams=row[3].split(':')
                break
    marksheet=Texttable()
    marksheet.set_cols_align(('l','l','r','r','c','l'))
```

```
    marksheet.add_row(['Course','Course Id','Marks Obtained','Full
     ↪  Marks','Grade','Remarks'])
    total=0
    with open('databases/course.csv') as csvfile:
        db=reader(csvfile)
        for row in db:
            if row[0] in exams:
                performance=row[2]
                i=performance.index(student_id)
                a=performance.find(':',i)
                b=performance.find('-',i)
                marks=float(performance[a+1:b])
                total+=marks
                marksheet.add_row([
                    row[1],
                    row[0],
                    marks,
                    100,
                    *grade(marks)
                ])
    number=len(exams)
    marksheet.add_row(['Total','-',total,number*100,*grade(total/numb
     ↪  er)])
    with open(f'outputs/{student_id}-report_card.txt','w') as
     ↪  report:report.write(f'''
{name} ({roll})
{marksheet.draw()}
ID:{student_id}
Batch:{batch_id}
 ''')
    return EXIT_CODE
```

course.py

```
from csv import reader,writer
from collections import namedtuple
from matplotlib.pyplot import
 ↪  hist,title,xlabel,ylabel,xticks,xlim,style,close,savefig
Student=namedtuple("Student",('roll','name','marks'))
def _parse_args(argdict):
    wrong_arg=Exception('Either provide course_id or course_name')
    if len(argdict)>1:raise wrong_arg
    (param,val),=argdict.items()
    if param=='course_id':rown=0
    elif param=='course_name':rown=1
    else:raise wrong_arg
    return rown,val
def create_course(**kwargs):
    marks=''
    batches=set()
    course_id=kwargs['course_id']
    with open('databases/student.csv','r') as csvfile:
        db=reader(csvfile)
        for student_data in kwargs['marks']:
            roll=student_data['roll number']
```

```
            for row in db:
                if row[2]==roll:
                    student_id=row[0]
                    marks+=f"{student_id}:{student_data['marks']}-"
                    batches.add(student_id[0:-2])
                    csvfile.seek(0)
                    break
    with open('databases/course.csv','a') as csvfile:
        writer(csvfile).writerow([
            course_id,
            kwargs['course_name'],
            marks[:-1]#skip last '-'
        ])
    rows=[]
    with open('databases/batch.csv','r') as csvfile:
        for row in reader(csvfile):
            if row[0] in batches:
                row[3]+=':'+course_id
            rows.append(row)
    with open('databases/batch.csv','w') as csvfile:
        db=writer(csvfile)
        for row in rows:db.writerow(row)
def course_performance(**kwargs):
    rown,val=_parse_args(kwargs)
    marks=False
    with open('databases/course.csv','r') as csvfile:
        for row in reader(csvfile):
            if row[rown]==val and (perf:=row[2]):
                marks=perf.split('-')
                break
    if not marks:return -1
    with open('databases/student.csv','r') as csvfile:
        db=reader(csvfile)
        for perf in marks:
            student_id,mark=perf.split(':')
            for row in db:
                if row[0]==student_id:
                    yield Student(row[2],row[1],float(mark))
                    csvfile.seek(0)
                    break
def course_statistics(**kwargs):
    close()
    rown,val=_parse_args(kwargs)
    marks=False
    with open('databases/course.csv','r') as csvfile:
        for row in reader(csvfile):
            if row[rown]==val:
                performance=row[2]
                if performance=='':return -1
                marks=[float(i[i.index(':')+1:]) for i in
                ↪  performance.split('-')]
                break
    if not marks:return -1
    style.use('Solarize_Light2')
    hist(marks,bins=[0,50,60,70,80,90,100])
    title(val)
    xlabel('marks')
    ylabel('number of students')
```

```
    xticks([25,55,65,75,85,95],['F','E','D','C','B','A'])
    xlim(100,0)
    savefig(f'outputs/Course Statistics-{val}.pdf')
```

batch.py

```
from csv import reader,writer
from functools import partial
from collections import namedtuple
from matplotlib.pyplot import
 ↪  pie,title,style,xticks,yticks,close,savefig
Student=namedtuple("Student",('roll','name','percentage'))
def _parse_args(argdict):
    wrong_arg=Exception('Either provide batch_id or batch_name')
    if len(argdict)>1:raise wrong_arg
    (param,val),=argdict.items()
    if param=='batch_id':rown=0
    elif param=='batch_name':rown=1
    else:raise wrong_arg
    return rown,val
def _direct_list(col,**kwargs):
    rown,val=_parse_args(kwargs)
    with open('databases/batch.csv','r') as csvfile:
        for row in reader(csvfile):
            if row[rown]==val:
                return row[col].split(':')
    return -1
def create_batch(**kwargs):
    with open('databases/batch.csv','a') as csvfile:
        writer(csvfile).writerow([
            kwargs['batch_id'],
            kwargs['batch_name'],
            kwargs['department_name'],
            ':'.join(kwargs['courses']),
            ':'.join(kwargs['students'])
        ])
students=partial(_direct_list,4)
courses=partial(_direct_list,3)
def batch_performance(**kwargs):
    rown,val=_parse_args(kwargs)
    students=[];exams=[]
    with open('databases/batch.csv','r') as csvfile:
        for row in reader(csvfile):
            if row[rown]==val:
                students=row[4].split(':')
                exams=row[3].split(':')
                break
    if not students and not exams:return -1
    lexams=len(exams)
    with open('databases/student.csv','r') as
     ↪  studentcsv,open('databases/course.csv') as csvfile:
        courses=reader(csvfile)
        for row in reader(studentcsv):
            student_id=row[0]
            if student_id in students:
```

```
                total=0
                for course in courses:
                    if course[0] in exams:
                        marks=course[2]
                        i=marks.index(student_id)
                        a=marks.find(':',i)
                        b=marks.find('-',i)
                        total+=float(marks[a+1:b])
                csvfile.seek(0)
                yield Student(row[2],row[1],total/lexams)
def batch_statistics(**kwargs):
    close()
    slices,roll_numbers=[],[]
    for student in batch_performance(**kwargs):
        slices.append(student.percentage)
        roll_numbers.append(student.roll)
    name=tuple(kwargs.values())[0]
    title(name)
    xticks([],[])
    yticks([],[])
    style.use('Solarize_Light2')
    pie(slices,labels=roll_numbers,shadow=True,frame=True)
    savefig(f'outputs/Batch Statistics-{name}.pdf')
```

department.py

```
from csv import reader,writer
from collections import namedtuple
from matplotlib.pyplot import
 ↪  plot,xlabel,ylabel,style,title,close,savefig
Batch=namedtuple('Performance',('batch','average'))
def _parse_args(argdict):
    wrong_arg=Exception('Either provide department_id or
     ↪  department_name')
    if len(argdict)>1:raise wrong_arg
    (param,val),=argdict.items()
    if param=='department_id':rown=0
    elif param=='department_name':rown=1
    else:raise wrong_arg
    return rown,val
def create_department(**kwargs):
    with open('databases/department.csv','a') as db:
        writer(db).writerow([
            kwargs['department_id'],
            kwargs['department_name'],
            ':'.join(kwargs['batches'])
        ])
def batches(**kwargs):
    rown,val=_parse_args(kwargs)
    with open('databases/department.csv','r') as db:
        for row in reader(db):
            if row[rown]==val:
                return row[2].split(':')
    return -1
def batch_averages(**kwargs):
```

```
    with open('databases/batch.csv','r') as
     ↪  batch_csv,open('databases/course.csv','r') as course_csv:
        batch_db=reader(batch_csv)
        course_db=reader(course_csv)
        for batch in batches(**kwargs):
            total=0
            for row in batch_db:
                if row[0]==batch:
                    batch_csv.seek(0)
                    courses=row[3].split(':')
                    students=row[4].split(':')
                    batch_csv.seek(0)
                    break
            for course in courses:
                for row in course_db:
                    if row[0]==course:
                        performance=row[2]
                        for student in students:
                            i=performance.index(student)
                            a=performance.find(':',i)
                            b=performance.find('-',i)
                            total+=float(performance[a+1:b])
                        course_csv.seek(0)
                        break
            yield Batch(batch,total/(len(students)*len(courses)))
def department_statistics(**kwargs):
    close()
    def year(performance):
        a=float(performance.batch[-2:])
        if a>22:
            return 1900+a
        return 2000+a
    stat=list(batch_averages(**kwargs))
    stat.sort(key=year)
    style.use('Solarize_Light2')
    plot([p.average for p in stat],[p.batch for p in
     ↪  stat],linestyle='--')
    xlabel('Batch Average')
    ylabel('Batch')
    name=tuple(kwargs.values())[0]
    title(name)
    savefig(f'outputs/Department Statistics-{name}.pdf')
```

examination.py

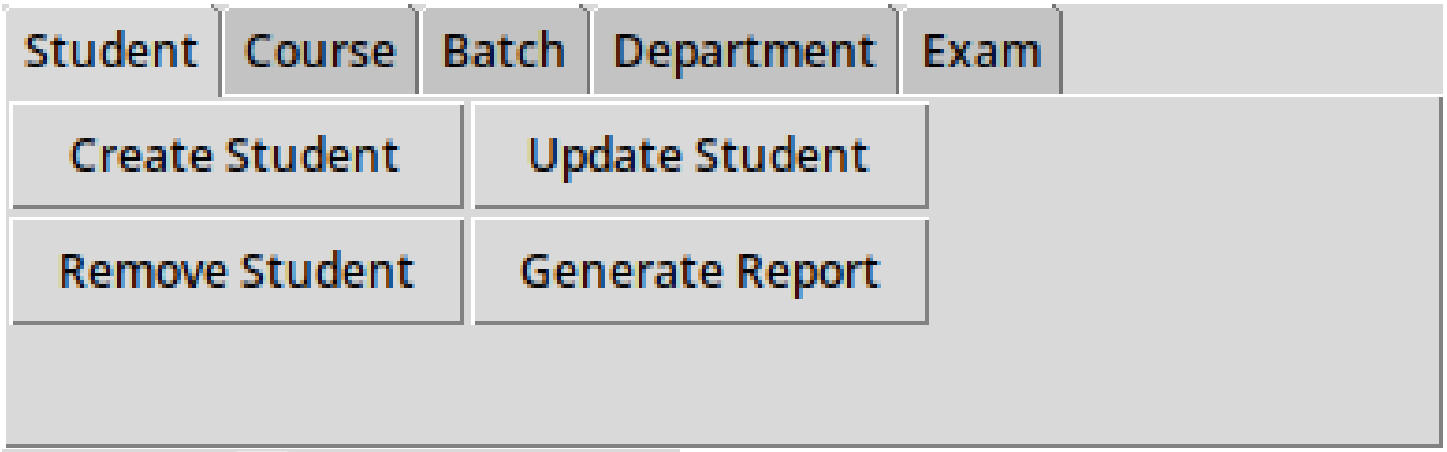
```
from csv import reader,writer
from numpy import nan,linspace
from collections import namedtuple
from matplotlib.pyplot import
 ↪  scatter,title,xlabel,ylabel,style,legend,close,savefig
from matplotlib.cm import Oranges as colormap #change to change
 ↪  colormap
Student=namedtuple('Performance',('student_id','average'))
Paper=namedtuple('Paper',('batch','student','roll_no','course','stude
 ↪  nts','courses'))
class Examination:
```

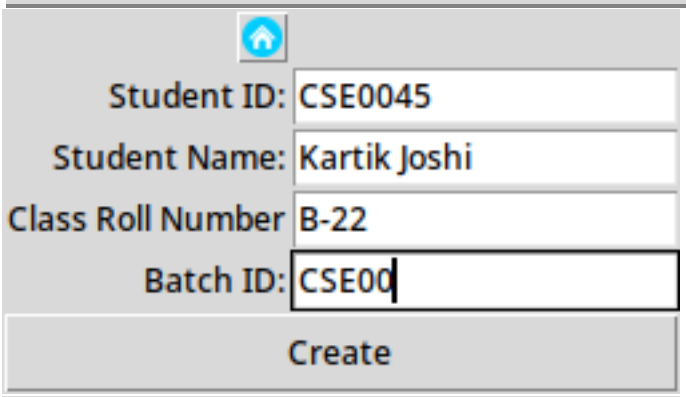
```
def __init__(self,exam_name,*batches):
    self.name=exam_name
    self.batches=batches
    self.course_name={}
    self.exam_data={}
    #remember data
    with open('databases/course.csv','r') as csvfile:
        csvfile.readline()
        for course_id,name,performance in reader(csvfile):
            self.exam_data[course_id]={} if performance=='' else
                dict((i.split(':') for i in
             ↪  performance.split('-')))
            self.course_name[course_id]=name
    self.student_performance={}
def take_exam(self):
    with open('databases/batch.csv','r') as
     ↪  batchcsv,open('databases/student.csv') as studentcsv:
        student_info=reader(studentcsv)
        for row in reader(batchcsv):
            batch_id=row[0]
            if batch_id in self.batches:
                courses=row[3].split(':')
                lcourses=len(courses)
                students=row[4].split(':')
                lstudents=len(students)
                for student in students:
                    total=0
                    for info in student_info:
                        if info[0]==student:#found student id
                            studentcsv.seek(0)
                            break
                    for course in courses:
                        yield Paper(batch_id,student,info[2],cour
                         ↪  se,lstudents,lcourses)
def enter_marks(self,exam):
    #get data
    plot_data={}
    for paper,marks in exam:
        try:
            self.student_performance[paper.student]+=marks/paper.
             ↪  courses
        except KeyError:
            self.student_performance[paper.student]=marks/paper.c
             ↪  ourses
        self.exam_data[paper.course][paper.student]=marks
        try:
            plot_data[paper.course][paper.batch]+=marks/paper.stu
             ↪  dents
        except KeyError:
            try:
                plot_data[paper.course][paper.batch]=marks/paper.
                 ↪  students
            except KeyError:
                plot_data[paper.course]={paper.batch:marks/paper.
                 ↪  students}
    #save data
    with open('databases/course.csv','w') as csvfile:
        db=writer(csvfile)
```

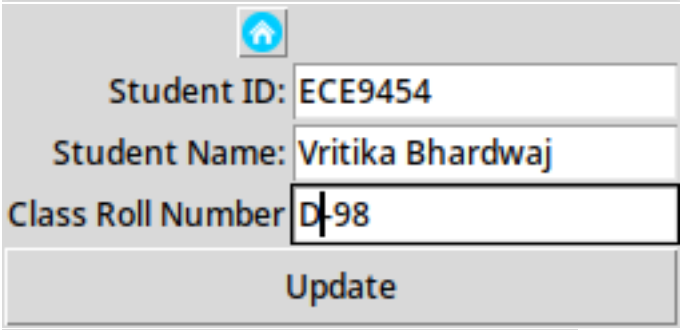
```
            db.writerow(['Course ID','Course Name','Marks Obtained'])
            for course in self.course_name:
                db.writerow([
                    course,
                    self.course_name[course],
                    '-'.join((f'{student}:{marks}' for student,marks
                     ↪  in self.exam_data[course].items()))
                ])
        #arrange data
        self.data=[]
        self.courses=[]
        for course,course_data in plot_data.items():
            batch_data=[]
            for batch in self.batches:
                try:
                    batch_data.append(course_data[batch])
                except KeyError:
                    batch_data.append(nan)
            self.courses.append(course)
            self.data.append(batch_data)
        self.courses,self.data=tuple(zip(*((x,y) for x,y in
         ↪  sorted(zip(self.courses,self.data)))))#sort data
    def statistics(self):
        close()
        style.use('Solarize_Light2')
        xlabel('Average Marks')
        ylabel('Batch')
        title(self.name)
        legend(
            (scatter(marks,self.batches,color=color,edgecolor='black'
                ) for marks,color in
                zip(self.data,colormap(linspace(0,1,len(self.data))))
             ↪  ),
            self.courses
        )
        savefig(f'outputs/{self.name} Exam.pdf')
```

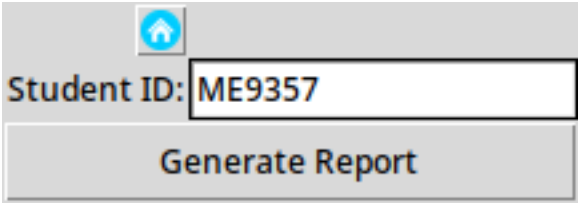
5 Outputs

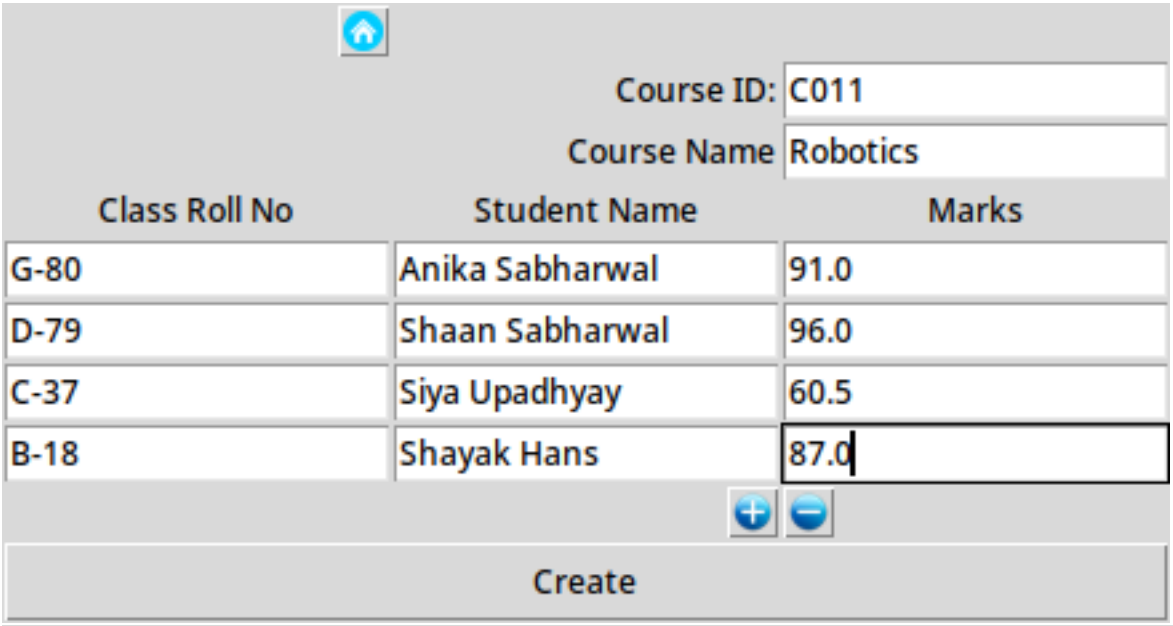
Graphic User Interface

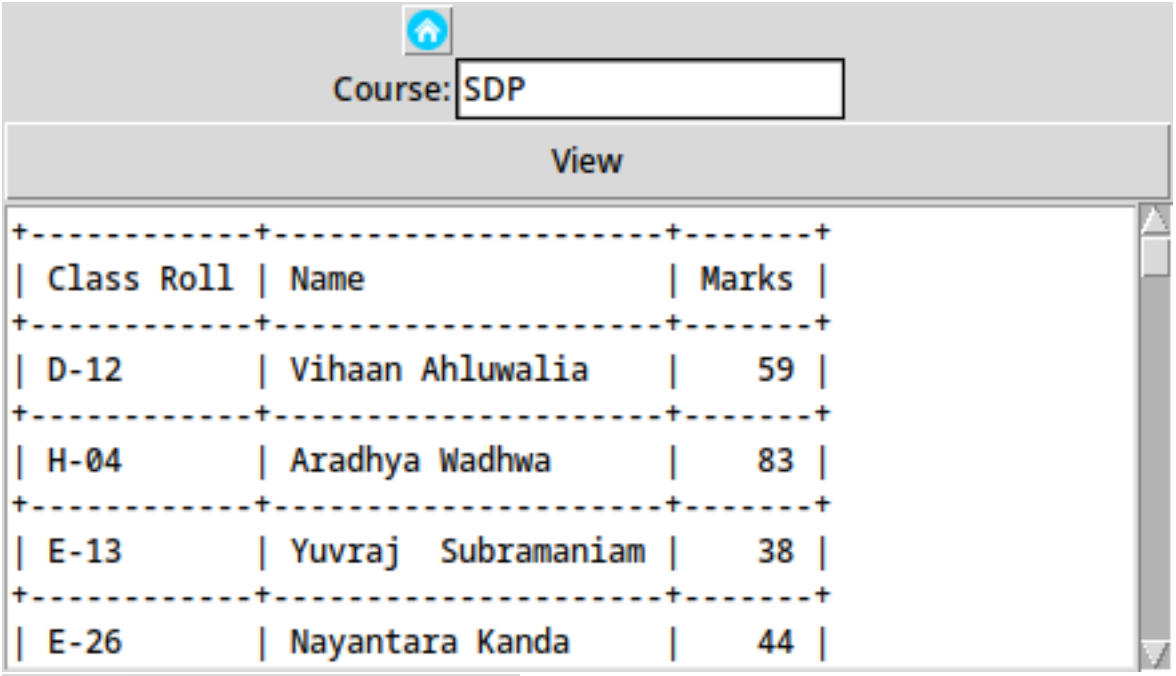


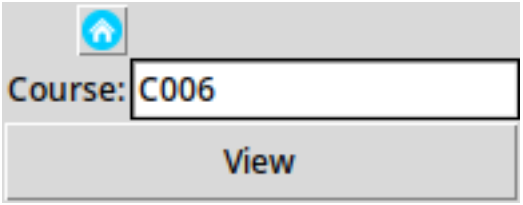


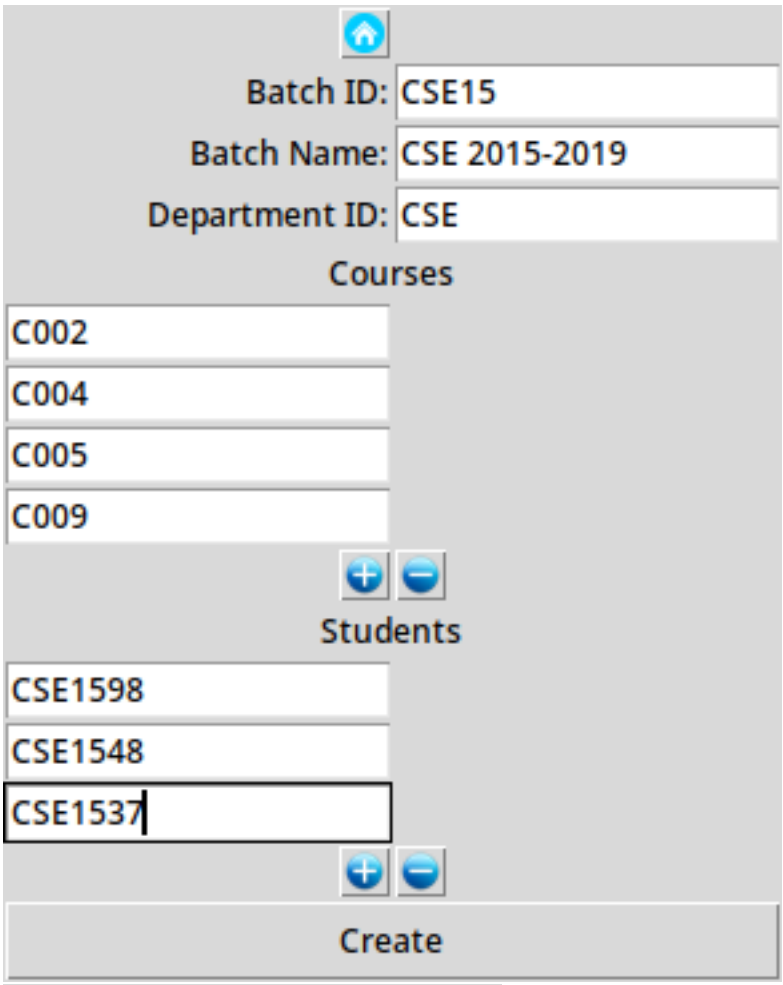


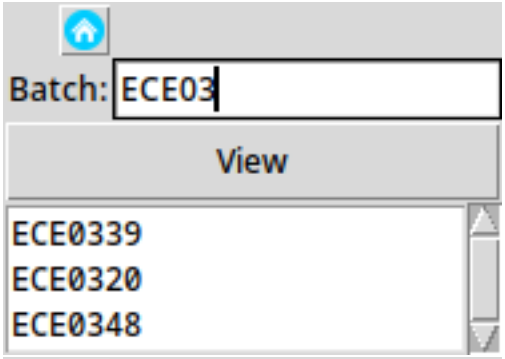


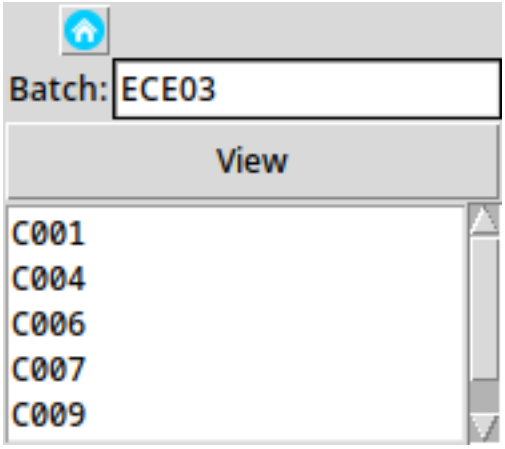

Course ID: C011
Course Name Robotics
Class Roll No
Student Name
Marks
G-80
Anika Sabharwal
91.0
D-79
Shaan Sabharwal
96.0
C-37
Siya Upadhyay
60.5
B-18
Shayak Hans
87.0
Create

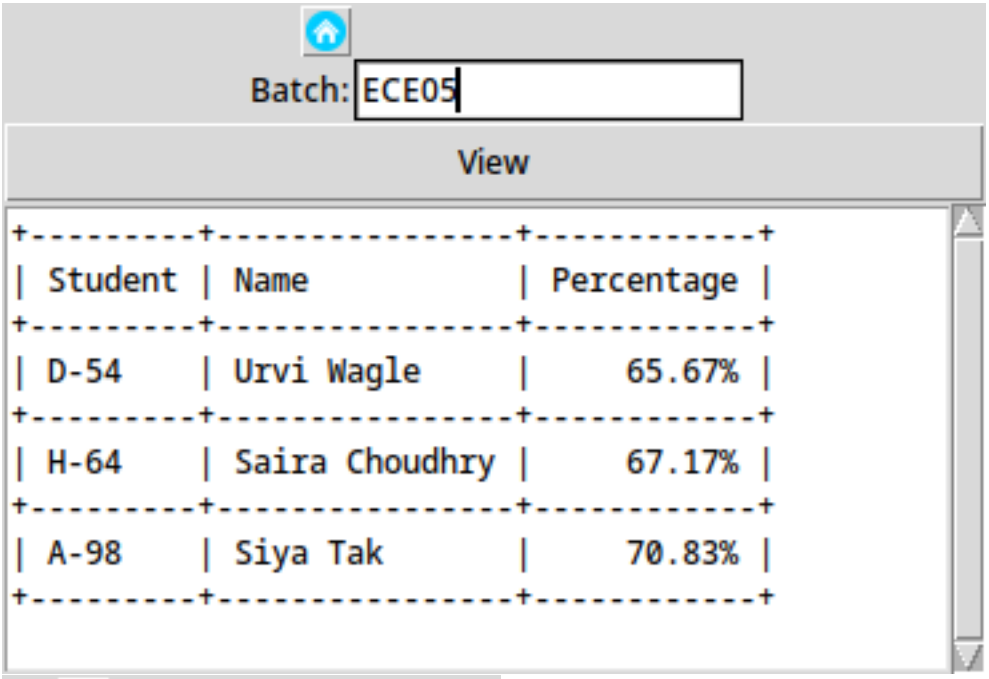

Course: SDP
View
+------------+----------------------+-------+
| Class Roll | Name | Marks |
+------------+----------------------+-------+
| D-12 | Vihaan Ahluwalia | 59 |
+------------+----------------------+-------+
| H-04 | Aradhya Wadhwa | 83 |
+------------+----------------------+-------+
| E-13 | Yuvraj Subramaniam | 38 |
+------------+----------------------+-------+
| E-26 | Nayantara Kanda | 44 |

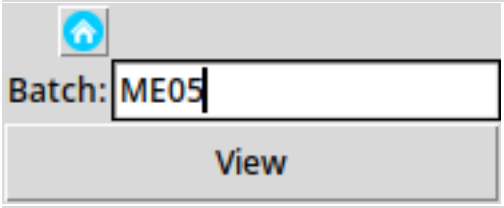

Course: C006
View

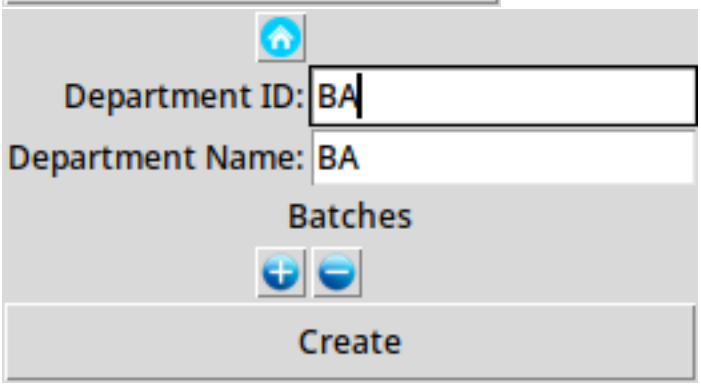

Batch ID: CSE15
Batch Name: CSE 2015-2019
Department ID: CSE
Courses
C002
C004
C005
C009
Students
CSE1598
CSE1548
CSE1537
Create

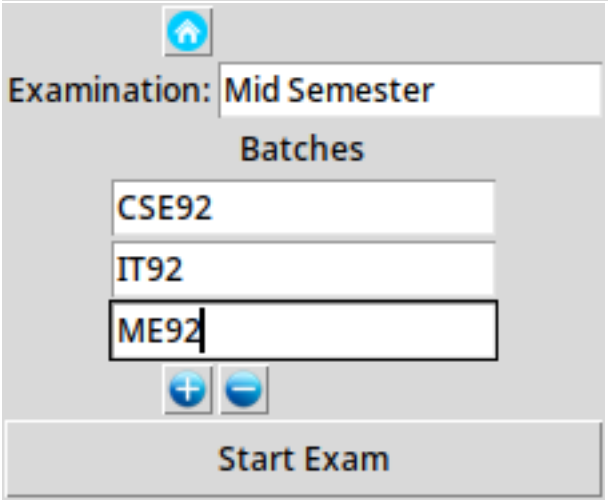

Batch: ECE03
View
ECE0339
ECE0320
ECE0348

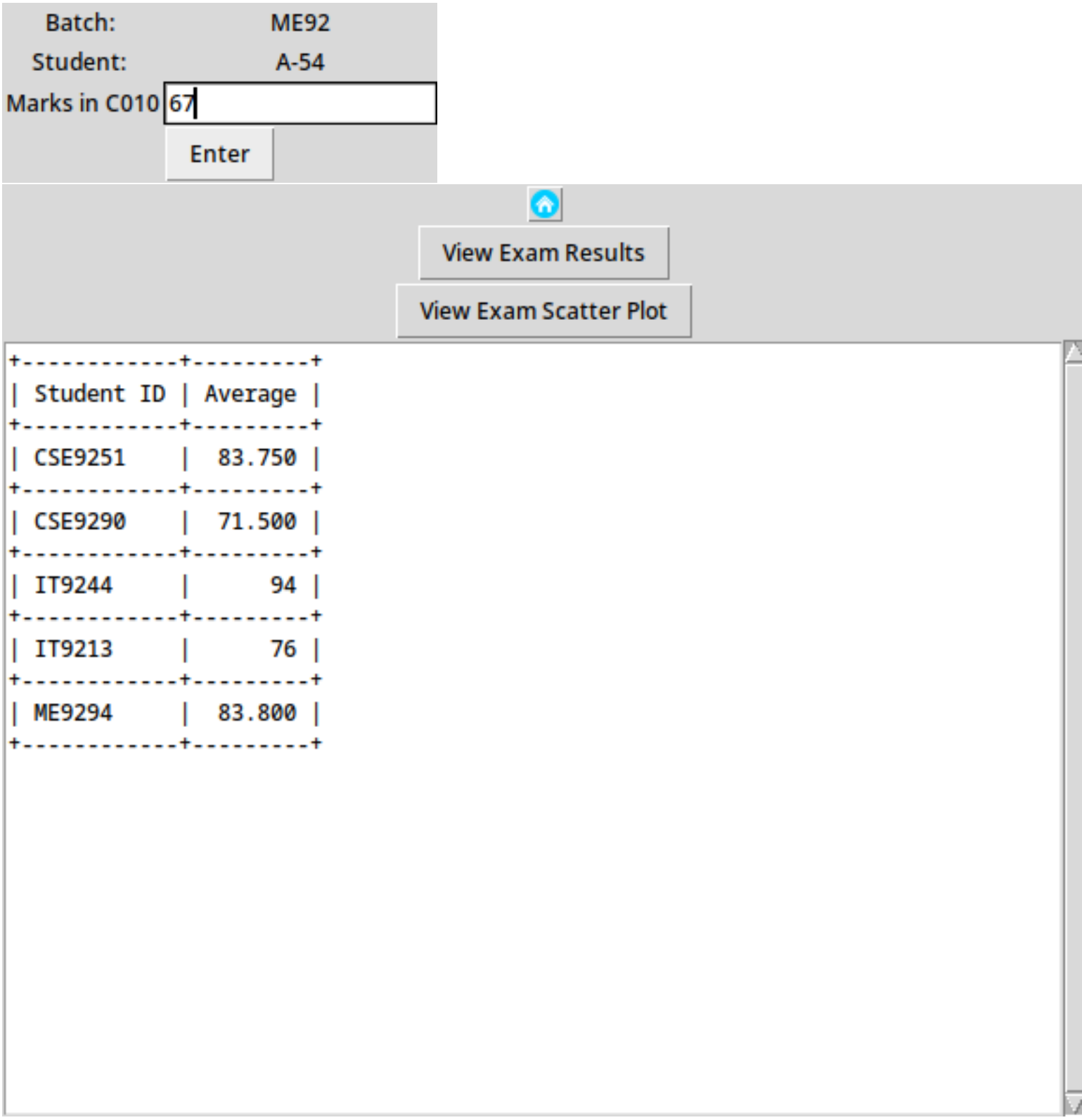

Batch: ECE03
View
C001
C004
C006
C007
C009


Batch: ECE05
View
+---------+----------------+------------+
| Student | Name | Percentage |
+---------+----------------+------------+
| D-54 | Urvi Wagle | 65.67% |
+---------+----------------+------------+
| H-64 | Saira Choudhry | 67.17% |
+---------+----------------+------------+
| A-98 | Siya Tak | 70.83% |
+---------+----------------+------------+


Batch: ME05
View


Department ID: BA
Department Name: BA
Batches
Create


Examination: Mid Semester
Batches
CSE92
IT92
ME92
Start Exam



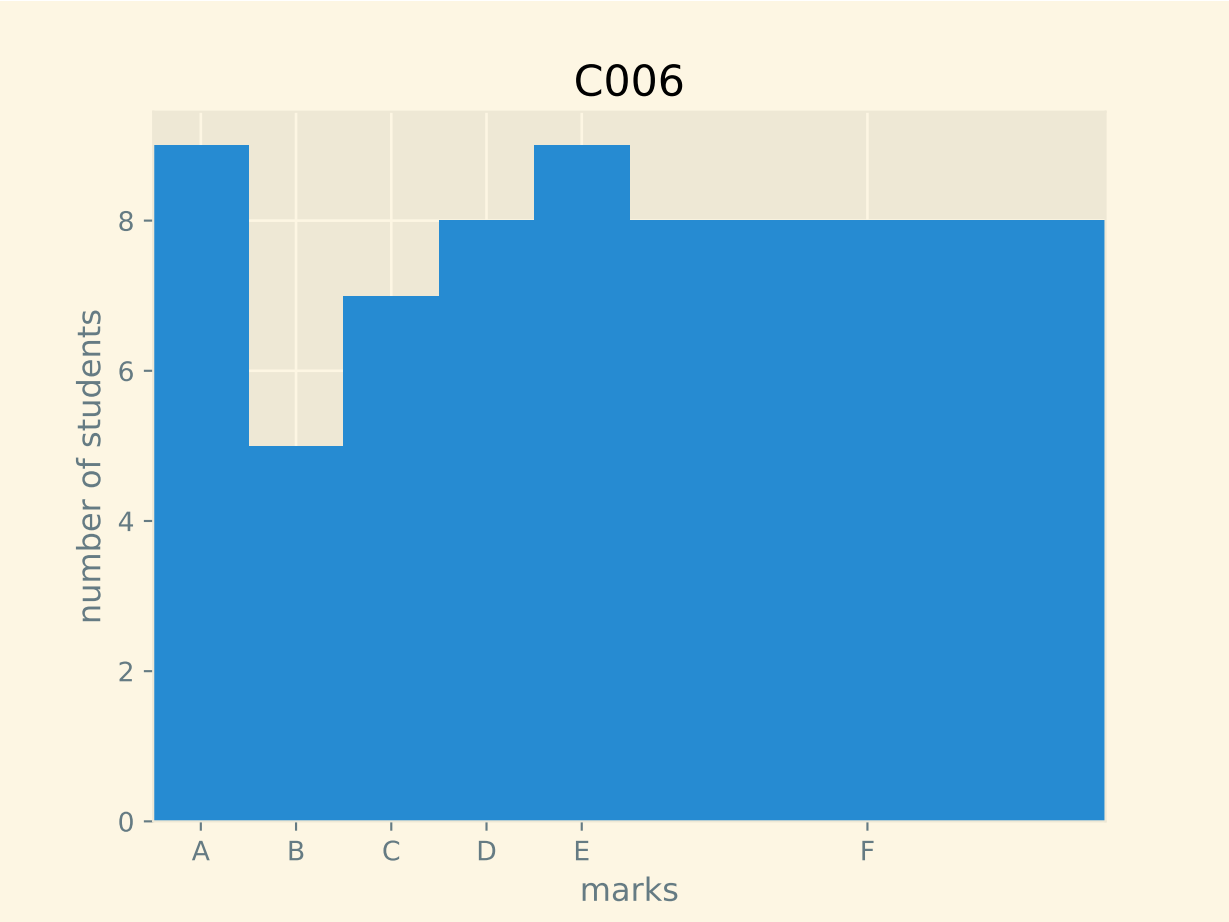
ME9357-report_card.txt

Anvi Warrior (A-80)

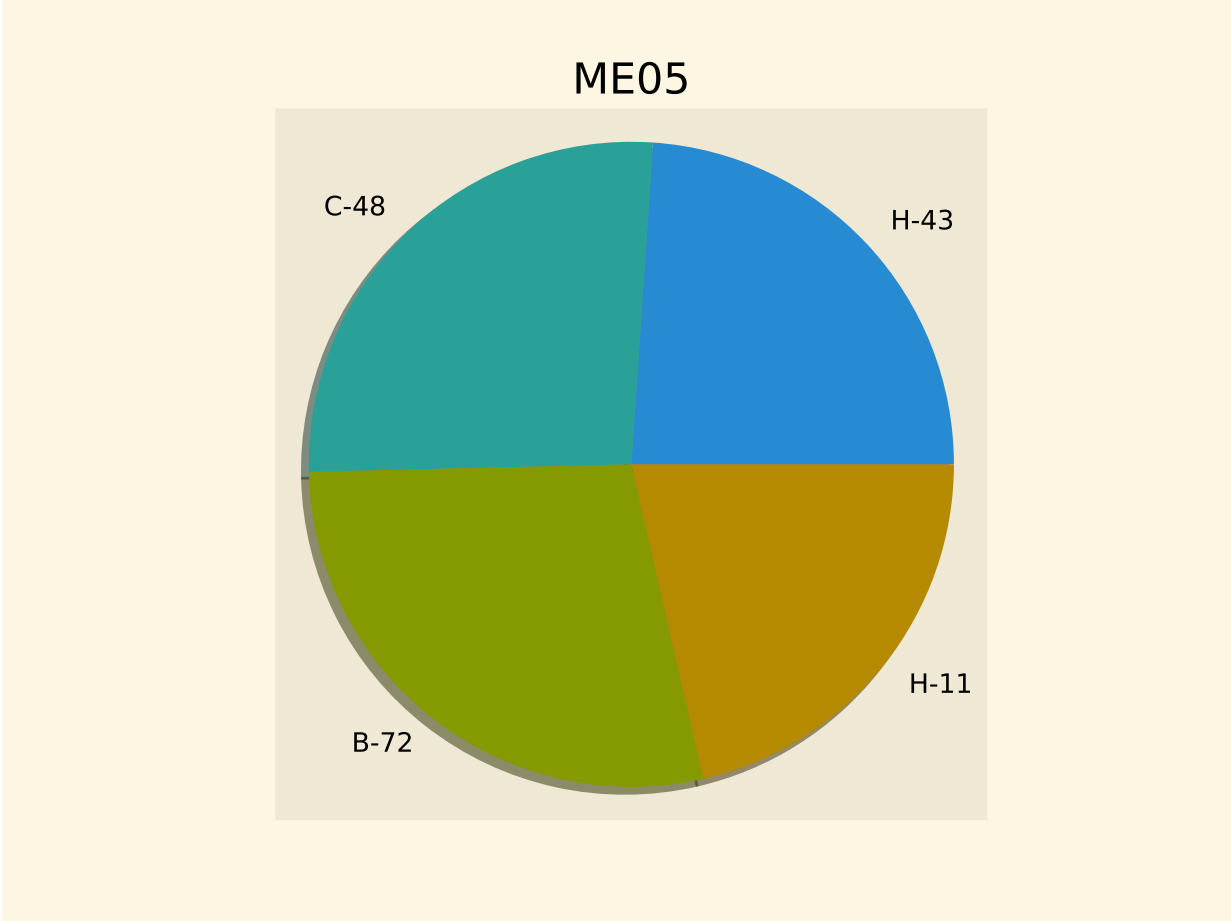
Course	Course Id	Marks Obtained	Full Marks	Grade	Remarks
Biology	C003	32	100	F	Failed
Electrical	C004	79	100	C	Passed
Python	C006	60	100	D	Passed
ESP	C009	62	100	D	Passed
SDP	C010	45	100	F	Failed
Total	-	278	500	E	Passed

ID:ME9357
Batch:ME93

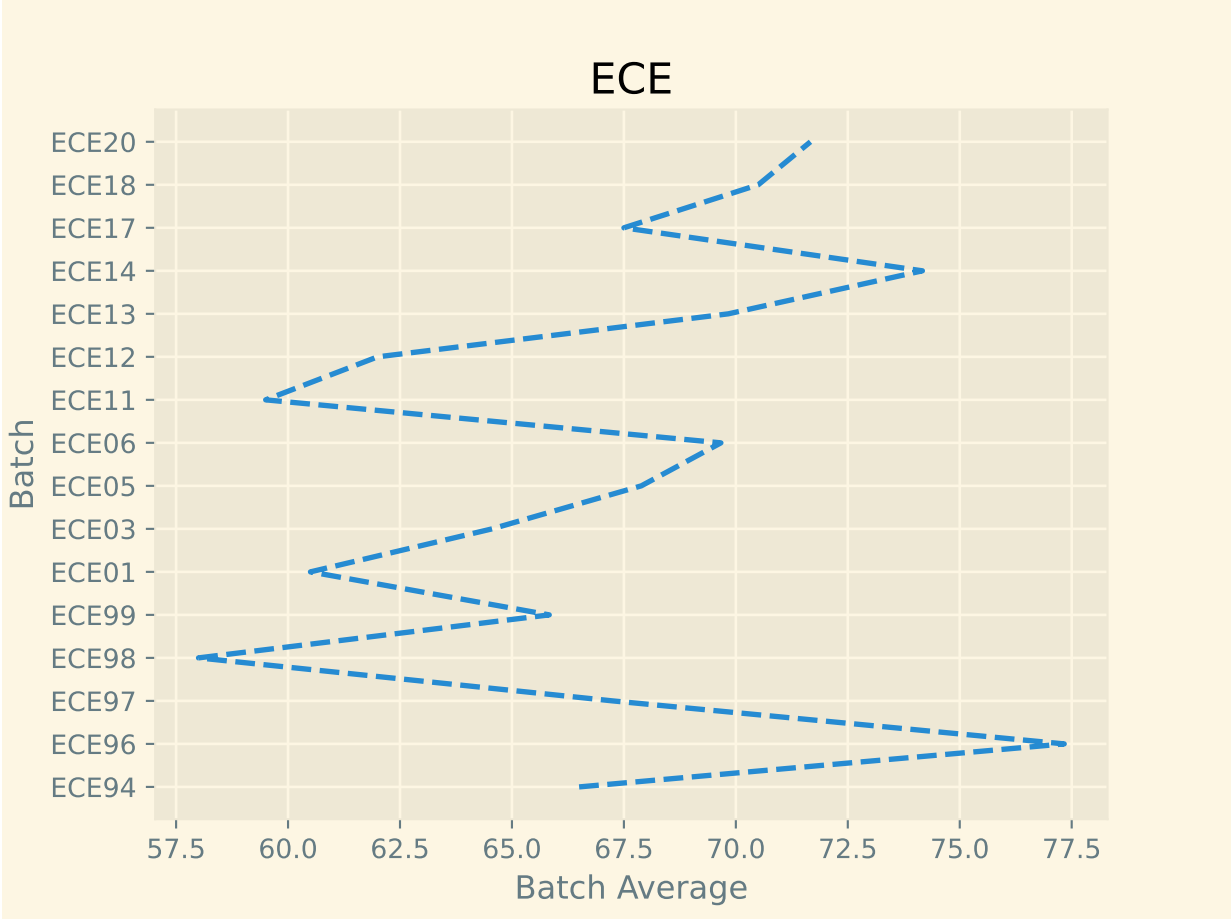
Course Statistics-C006.pdf

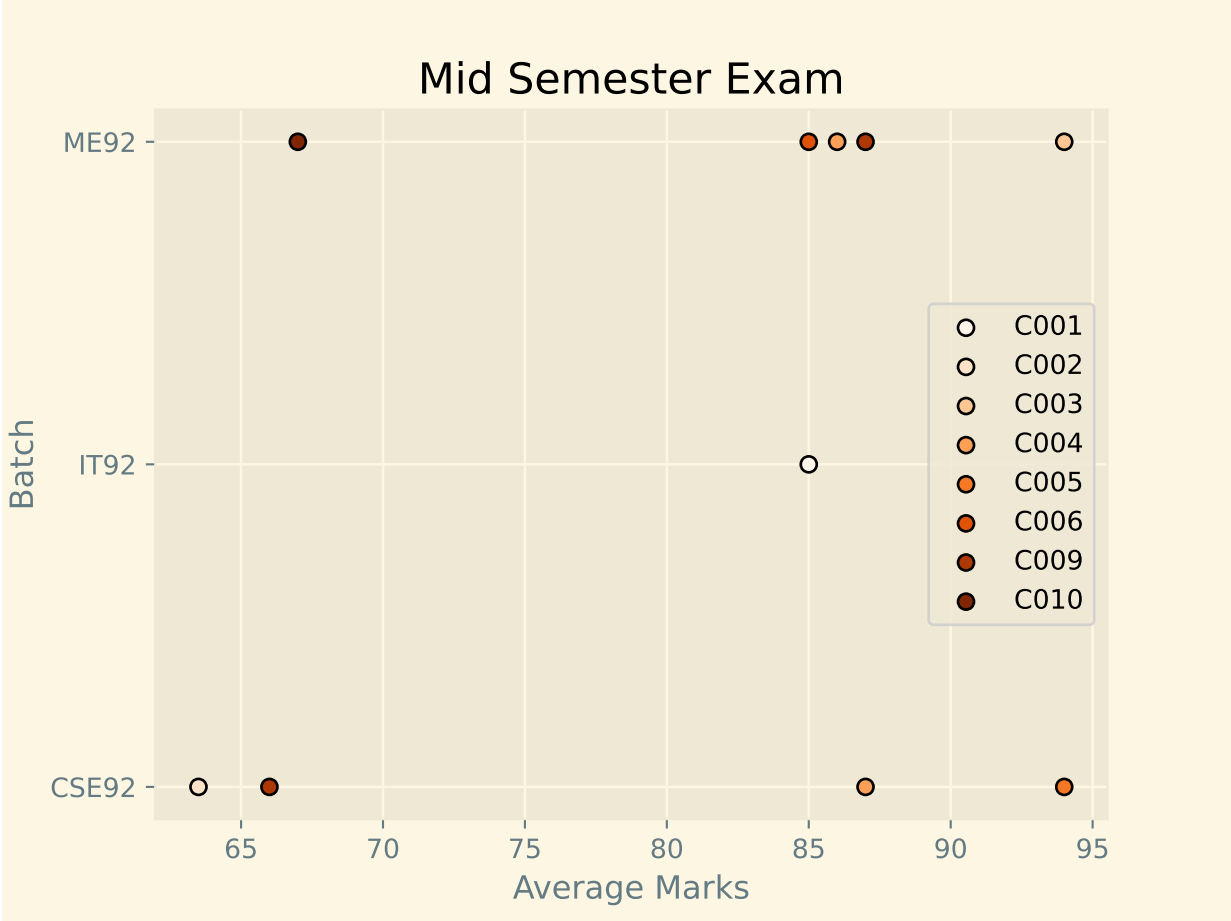


Batch Statistics-ME05.pdf



Department Statistics-ECE.pdf




Mid Semester Exam
ME92
IT92
CSE92
Batch
65
70
75
80
85
90
95
Average Marks
C001
C002
C003
C004
C005
C006
C009
C010