Student Performance Report

Submitted by

Name of the Students: Arjab Sengupta
Enrollment Number: 12022002013023
Section: K
Class Roll Number: 32
Stream: EEE
Subject: Programming for Problem Solving with Python
Subject Code: IVC101
Department: Basic Science and Humanities

Under the supervision of
Dr. Swarnendu Ghosh
Mrs. Sumana Sinha

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 **Arjab Sengupta,** entitled Student Performance Report accepted in partial fulfilment of the requirements for the degree of partial fulfilment of the first semester.

Head of the Department Basic Sciences and Humanities IEM, Kolkata	Project Supervisor

1 Introduction

This project allows users to keep track of students' performance by creating a database to keep a record of their marks and grades.

1.1 Objective

This project allows users to keep track of students' performance by creating a database to keep a record of their marks and grades. Organization of the Project
For the successful running of the project, the following modules were imported:
OS
CSV
subprocess
time
matplotlib

2 Programs

Provide the python programs of the various modules.

1) functions used throughout the code

```
def loading_screen():
    for i in range(10):
        sys.stdout.write("\rLoading" + "." * i)
        sys.stdout.flush()
        time.sleep(0.5)
    sys.stdout.write("\rLoading complete!")



def createfile(name, lst):
    with open(f"{path}/{name}", "a", newline="") as f:
        script = csv.writer(f)
        script.writerow(lst)
        print(f"{name} file has been UPDATED")
```

```
def percent(num):
    if (
        stream.lower() == "cse"
        or stream.lower() == "cseai"
        or stream.lower() == "cseaiml"
        or stream.lower() == "cseiotcsbs"
    ):
        num = ((num * 100) / 600)
    elif stream.lower() == "it" or stream.lower() == "ece" or
stream.lower() == "me":
        num = ((num * 100) / 500)
    return num


def grade(num):
    if num >= 90:
        return "Outstanding Performance... You have passed the exam with
grade A."
    elif num < 90 and num >= 80:
        return "Excellent Performance... You have passed the exam with
grade B."
    elif num < 80 and num >= 70:
        return "Good Performance... You have passed the exam with grade
C."
    elif num < 70 and num >= 60:
        return "Your performance is average... Work hard... You have
passed the exam with grade D."
    elif num < 60 and num >= 50:
        return "Your performance is below average... There is massive
scope of improvement... You have barely passed the exam with grade E."
    else:
        return "Extremely poor performance... You have Failed the Exam
and got F."


def count(lst):
    num = 0
```

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


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


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


def choice(stream):
    if (
        stream.lower() == "cse"
        or stream.lower() == "cseai"
        or stream.lower() == "cseaiml"
        or stream.lower() == "cseiotcsbs"
    ):
        return "C001:C002:C003:C004:C005:C006"
```

```
    elif stream.lower() == "it" or stream.lower() == "ece" or
stream.lower() == "me":
        return "C002:C003:C004:C005:C006"


def get_batch():
    with open(f"C:/PythonProgrammingProject_main-folder/Batch.csv", "r")
as f:
        reader = csv.reader(f)
        rows = [row for row in reader]
        column = []
        for i in range(len(rows)):
            if i == 0:
                pass
            else:
                column += [rows[i][0]]
    return column


def remove(string):
    with open(
        f"C:/PythonProgrammingProject_main-folder/Student.csv", "r+",
newline=""
    ) as f:
        script = csv.reader(f)
        rows = [row for row in script]
        for i in rows:
            if i[0] == string:
                rows[rows.index(i)] = ["", "", "", ""]
            else:
                pass
        f.seek(0)
        f.truncate()
        writer = csv.writer(f)
        writer.writerows(rows)


def course_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
    with open(f"{path}/Course.csv", "r") as f:
        script = csv.reader(f)
        rows = [row for row in script]
        req = []
        for i in range(len(rows)):
            if i == 0:
                pass
            else:
                req += [rows[i][2]]
        lst = [
            ["Python", (req[0].split("-"))[0:-1]],
            ["Math", (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]],
        ]

        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(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}

        for j in lst:
            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

        for text in leg.get_texts():
            text.set_color("White")

        plt.show()


def batch_graph(arg):
    with open(f"{path}/Batch.csv", "r") as f:
        reader = csv.reader(f)
        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}/Course.csv", "r") as f:
        reader = csv.reader(f)
        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)):
        if lst[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": grade_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 = []
    for i in range(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"
    # 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), 0.60, 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,
    )
    for text in leg.get_texts():
        text.set_color("white")

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


def department_graph():
    need = {}
    with open(f"{path}/Batch.csv", "r") as f:
        reader = csv.reader(f)
        batch = [batch[0] for batch in reader]
        batch = batch[1:]
    for arg in batch:
        avg = 0
        with open(f"{path}/Batch.csv", "r") as f:
            reader = csv.reader(f)
            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}/Course.csv", "r") as f:
            reader = csv.reader(f)
            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)):
        if lst[i][0][:3] == "CSE":
            lst[i][1] = (lst[i][1] * 100) / 600
        else:
            lst[i][1] = (lst[i][1] * 100) / 500
    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())
```

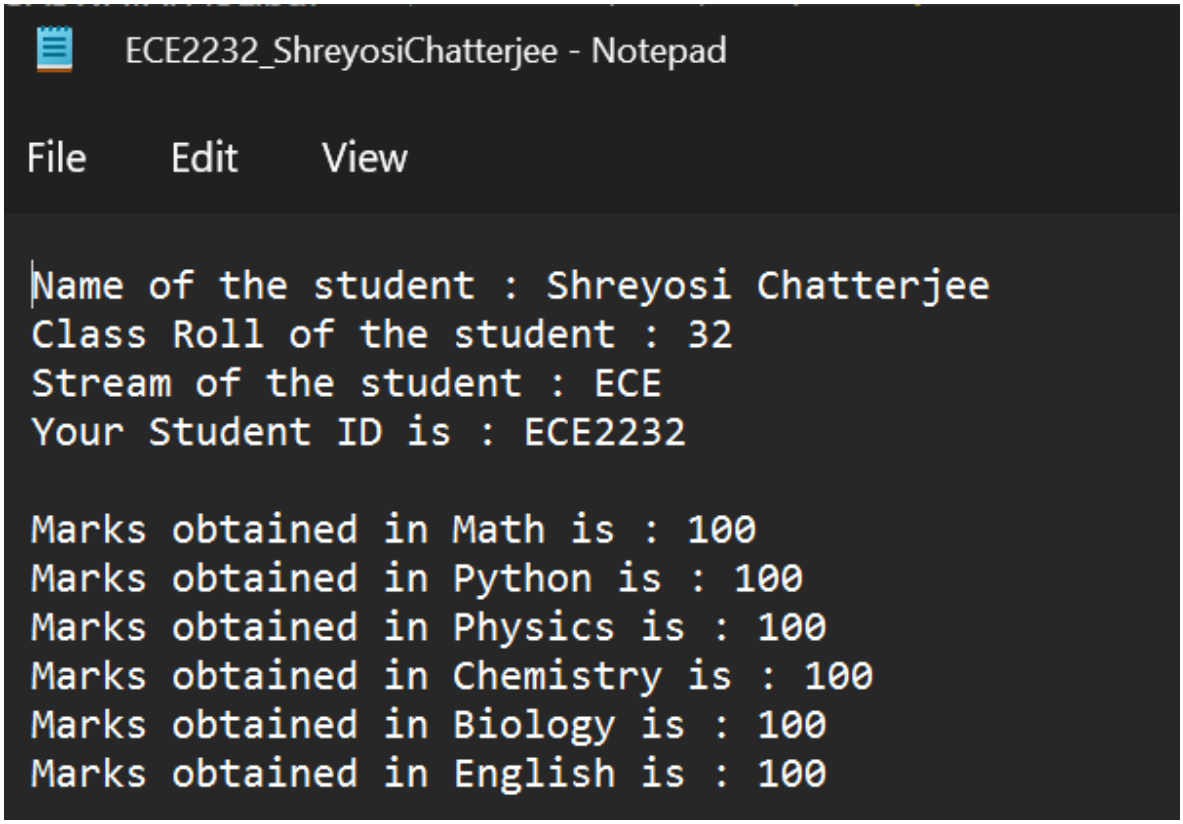
```
    color_lst = ["#C70039", "#9BB1F2", "#FFC300", "#FF5733", "#DAAFB1",
"#86B7C8"]
    fig, ax = plt.subplots()
    ax.set_facecolor("Black")
    fig.set_facecolor("Black")
    ax.set_xlabel("X axis", color="white")
    ax.set_ylabel("Y axis", color="white")
    ax.spines["bottom"].set_color("white")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

3 Outputs

The sample outputs demonstrate the functionalities of programs.


ECE2232_ShreyosiChatterjee - Notepad
File
Edit
View
Name of the student : Shreyosi Chatterjee
Class Roll of the student : 32
Stream of the student : ECE
Your Student ID is : ECE2232
Marks obtained in Math is : 100
Marks obtained in Python is : 100
Marks obtained in Physics is : 100
Marks obtained in Chemistry is : 100
Marks obtained in Biology is : 100
Marks obtained in English is : 100