

**Project Documentation:**  
**“ GPA Calculator and Result Analysis using Python”**

- **Harsh Bang, CSE (Data Science) – STME NMIMS HYD**  
[harsh.bang027@nmims.edu.in](mailto:harsh.bang027@nmims.edu.in)  
[harshbang10@gmail.com](mailto:harshbang10@gmail.com)

## **Table of Contents**

1. Title: GPA Calculator and Result Analysis using Python

2. Abstract

3. Chapter 1: Introduction

- 1.1 Background and Motivation
- 1.2 Problem Statement
- 1.3 Objectives
- 1.4 Scope

4. Chapter 2: Methodology

- 2.1 Data Collection and Analysis
- 2.2 Calculation of GPA and Grade Points
- 2.3 Data Visualization
- 2.4 Result Analysis

5. Chapter 3: Technology

- 3.1 Programming Language: Python
- 3.2 Libraries and Modules Used

6. Chapter 4: Result and Discussion

- 4.1 Passing/Failing Percentage (Subject wise)
- 4.2 Overall Passing Percentage
- 4.3 Average Marks Scored by Students
- 4.4 Failing Count Analysis

7. Chapter 5: Project Source Code

- 5.1 Introduction to the Source Code

- 5.2 Explanation of Key Code Segments

## 8. Chapter 6: Readme Files

- 6.1 Purpose of Readme Files
- 6.2 Explanation of Readme Content

## 9. Chapter 7: User Manual

- 7.1 Installation Instructions
- 7.2 Using the GPA Calculator
- 7.3 Interpreting the Results

## 10. Chapter 8: Conclusion and Future Scope

- 8.1 Summary of Achievements
- 8.2 Limitations and Challenges
- 8.3 Future Enhancements and Extensions

# **GPA Calculator and Result Analysis using Python**

## ***Abstract***

The GPA Calculator and Result Analysis project is aimed at automating the calculation of students' Grade Point Averages (GPA) and conducting a comprehensive analysis of their academic performance. Leveraging the power of Python programming and data analysis libraries, the project streamlines the process of data collection, processing, and visualization to provide accurate insights into students' performance across subjects. This documentation outlines the project's methodology, technology stack, obtained results, details of the source code, readme files, user manual, and potential directions for future enhancements.

## ***Chapter 1: Introduction***

### **1.1 Background and Motivation**

The GPA Calculator and Result Analysis project originated during the first semester Term End Examination. As the term end exams approached, manually predicting GPAs turned into a time-consuming ordeal, pushing the need for a smarter solution. To tackle this challenge, an initial version was created using C++, offering a glimpse into automation possibilities. However, realizing the potential for a more comprehensive tool, the project shifted to Python, aiming to create a user-friendly interface adaptable to various semesters.

The motivation behind this transition was to build an accessible tool that automates GPA calculations and provides result insights. The goal was to empower students to make informed academic choices and enhance their planning. This evolution transformed the project from a personal tool into the GPA Calculator and Result Analysis, serving the broader student community by simplifying GPA prediction and result analysis. The project encapsulates a journey from a personal solution to a versatile tool aimed at assisting students within the School of Technology Management and Engineering.

### **1.2 Problem Statement**

Manual GPA calculation and result analysis can be time-consuming, error-prone, and subject to inconsistencies. This project aims to eliminate these issues by automating the calculation of GPA and conducting a thorough analysis of student performance.

### **1.3 Objectives**

- Develop an efficient GPA calculator that considers both Internal Continuous Assessment (ICA) and Term End Examination (TEE) marks.

- Analyze passing and failing percentages for each subject, enabling a detailed examination of academic performance.
- Calculate the overall passing percentage, providing an overview of students' collective success.
- Visualize the results using graphs and charts to present data in an accessible manner.
- Create a user-friendly interface that simplifies data input and result interpretation for users.

## **1.4 Scope**

The project is specifically tailored for students at the School of Technology Management and Engineering. It encompasses the development of a GPA calculator, in-depth analysis of passing/failing percentages, and data visualization to facilitate a comprehensive understanding of academic performance. However, it is important to note that the project's scope does not extend to the integration of advanced predictive models through machine learning techniques.

## ***Chapter 2: Methodology***

### **2.1 Data Collection and Analysis**

The project collects data through user input, including ICA and TEE marks for each subject. The data is then processed, analyzed, and visualized to calculate GPA, grade points, and passing/failing percentages.

### **2.2 Calculation of GPA and Grade Points**

The project employs a predefined grading system to assign grade points based on the final marks obtained. GPA is calculated by summing the products of grade points and subject credits and dividing by the total credits.

### **2.3 Data Visualization**

Passing and failing percentages are visually presented using pie charts, providing a quick overview of student performance trends. Furthermore, the average marks scored by students in each subject are visually communicated through line graphs to facilitate effective comparison.

## **2.4 Result Analysis**

The project's analysis includes subject-wise passing/failing percentages, the calculation of the overall passing percentage, and the assessment of failing counts within different subject categories.

## ***Chapter 3: Technology***

### **3.1 Programming Language: Python**

Python is selected as the primary programming language due to its simplicity, versatility, and extensive library support, making it ideal for data analysis and visualization tasks.

### **3.2 Libraries and Modules Used**

- **OS:** os is primarily used for directory navigation, enabling efficient access to data files.
- **Numpy:** Numpy is utilized for efficient numerical computations and handling arrays.
- **Pandas:** Pandas is employed for data manipulation, processing, and analysis.
- **Matplotlib:** Matplotlib is used for creating graphs and charts to visualize the analyzed data.
- **CSV:** The CSV module is utilized for reading and writing data to CSV files.
- **Datetime:** The Datetime module aids in capturing and managing date and time information.

## ***Chapter 4: Result and Discussion***

### **4.1 Passing/Failing Percentage (Subject wise)**

The project analyzes and presents passing and failing percentages for each subject, helping to identify areas of strength and weakness.

### **4.2 Overall Passing Percentage**

The overall passing percentage is calculated, providing an overarching view of students' academic success.

### **4.3 Average Marks Scored by Students**

The project calculates and visualizes the average marks scored by students in each subject, facilitating comparison and assessment of subject difficulty.

#### 4.4 Failing Count Analysis

The project analyzes the number of students failing in one subject, two subjects, and more than three subjects, allowing for a comprehensive understanding of academic challenges.

### *Chapter 5: Project Source Code*

```
import os
import numpy as np
import pandas as pd
from csv import writer
from datetime import datetime as dt
import matplotlib.pyplot as plt

os.chdir(r'E:\NM\Sem 3\gpa_project')

# initialising empty lists
ICA, TEE, TotalM, rounded_TM, gp_multiplied, credit_multiplied = [], [], [], [], [], []
ICA_P_F, TEE_P_F, TM_P_F, final = [], [], [], [] # P <- Pass, F <- Fail

# to calculate grade point student got by passing final marks scored in each subject
# and appending to an empty list 'gp_multiplied'
def grade_point(t):
    if ( T>=85 and T<=100):
        gp_multiplied.append(4.00)
    elif ( T>=81 and T<=84.99 ):
        gp_multiplied.append(3.75)
    elif ( T>=77 and T<=80.99 ):
        gp_multiplied.append(3.50)
    elif ( T>=73 and T<=76.99 ):
        gp_multiplied.append(3.25)
    elif ( T>=69 and T<=72.99 ):
        gp_multiplied.append(3.00)
    elif ( T>=65 and T<=68.99 ):
        gp_multiplied.append(2.75)
    elif ( T>=61 and T<=64.99 ):
        gp_multiplied.append(2.50)
    elif ( T>=57 and T<=60.99 ):
        gp_multiplied.append(2.25)
    elif ( T>=50 and T<=56.99 ):
        gp_multiplied.append(2.00)
```

```

elif ( T<=40 and T<=49.99 ):
    gp_multiplied.append(1.15)
else:
    gp_multiplied.append(0)

# "round-half-to-even" rounding method
def round_TM(x):
    if x >= 0:
        return int(x + 0.5)
    else:
        return int(x - 0.5)

def comment(gpa):
    if (gpa > 4):
        print('ErRoR')
    elif (gpa >= 3.5 and gpa <= 4):
        print('Hurray, you have done EXCELLENT')
    elif (gpa >= 3 and gpa <= 3.49 ):
        print('You have done GOOD, can IMPROVE')
    elif (gpa >= 2 and gpa<= 2.99):
        print('You can IMPROVE')
    else:
        print('You have to work HARDER')

sum = 0

print('***** STME GPA CALCULATOR *****')

name = input('Enter Name: ')
sap_id = int(input('Enter SAP ID: '))
now=dt.now()

data = [now,name,sap_id] # to gather information and save in list (later CSV file)

print('\nSelect any one')
print('1. Predict GPA')
print('2. Result Analysis (Data Visualization)')
#print('3. Prediction model')
op = int(input('action from above (int):')) # taking users input for above options

if op==1:

    sm=int(input('\nSelect Semester (1. Sem1 || 2. Sem2): '))

    if sm==1:

```



```

subject=['Calculus','Programming for Problem Solving',
        'Basic Electrical and Electronics Engineering','Elements of Biology',
        'Engineering Graphics and Design','English Communication',
        'Professional Ethics','Constitution of India',
        'Design Thinking']
sem1_subs_credit=[4,4,3,3,3,1,1,0,0]
sem1_subs_credit_Total = 19

print("\nNOTE: \nEnter Internal Continuous Assignment Marks(ICA out of 50)")
print(' & Term End Examination Marks(TEE out of 100): ')

for i in range(0,5):
    print(f"\nMarks in {subject[i]}:")

    icaM = int(input('ICA: '))
    if 0 <= icaM <= 50:
        ICA.append(icaM) # storing ica marks in 'ICA' list
    else:
        print('Entered marks out of range\nEnter ICA in the range of 50')
        icaM = int(input('Valid ICA: '))
        if 0 <= icaM <= 50:
            ICA.append(icaM) # storing the correct ICA marks in 'ICA' list
        else:
            print('Invalid ICA marks')

    teeM = int(input('TEE: '))
    if 0 <= teeM <= 100:
        TEE.append(teeM)# storing term end marks in 'TEE' list
    else:
        print('Entered marks out of range\nEnter TEE in range of 100')
        teeM = int(input('TEE: '))
        if 0 <= teeM <= 100:
            TEE.append(teeM) # storing the correct term end marks in 'TEE' list
        else:
            print('Invalid TEE marks')

for i in range(0,5):
    TEE[i]=TEE[i]/2
for i in range(5,9):
    TEE.append(0)

for j in range(5,9):
    print(f"\nMarks in {subject[j]}:")
    icaM = int(input('ICA: '))

```

```

if 0<= icaM <= 50:
    ICA.append(icaM)
else:
    print('Entered marks out of range\nEnter ICA in the range of 50')
    icaM = int(input('ICA: '))
    if 0<= icaM <= 50:
        ICA.append(icaM)
    else:
        print('Invalid ICA marks')

for j in range(5,9):
    ICA[j]= ICA[j]*2

for k in range(0,9):
    TotalM.append(ICA[k] + TEE[k])

#round TM
for d in TotalM:
    D = round_TM(d) # Function call
    rounded_TM.append(D)

for l in range(0,9):
    data.append(rounded_TM[l]) # Storing marks in list

for i in range(5,9):
    TEE[i]=22
    ICA[i]= ICA[i]/2

for i in range(0,9):
    if (ICA[i] >= 20):
        ICA_P_F.append('P')
    elif (ICA[i] < 20):
        ICA_P_F.append('F')

for i in range(0,9):
    if (TEE[i] >= 20):
        TEE_P_F.append('P')
    elif (TEE[i] < 20):
        TEE_P_F.append('F')

for i in range(0,9):
    if (TotalM[i] >= 50):
        TM_P_F.append('P')
    elif (TotalM[i] < 50):
        TM_P_F.append('F')

```

```

for i in range(0,9):
    if (ICA_P_F[i] == 'P' and TEE_P_F[i] == 'P' and TM_P_F[i] == 'P'):
        final.append('Pass')
    else:
        final.append('Fail')

dt = [final, ICA_P_F, TEE_P_F, TM_P_F]

for T in rounded_TM:
    grade_point(T) #function call

for c in range(0,9):
    credit_multiplied.append(sem1_subs_credit[c] * gp_multiplied[c])

for s in credit_multiplied:
    sum+=s

gpa = sum / sem1_subs_credit_Total
data.append(gpa)

for i in range(4):
    data.append(dt[i])

with open('sem1_gpa.csv', 'a') as f_object:
    writer(f_object).writerow(data)
    f_object.close()

print("\n ````````````````````")
print("GPA=",gpa)
comment(gpa)

elif sm==2:
    subject=['Linear Algebra and Differential Equations',
            'Physics','Digital Circuits and Computer Architecture',
            'Principles of Economics and Management','Python for data analysis',
            'Environmental Science','Digital Manufacturing Laboratory',
            'Electrical and Electronics Workshop','Critical Thinking ']

ICA, TEE, TotalM, gp_multiplied, credit_multiplied = [], [], [], [], []

sem2_subs_credit=[4,4,4,3,2,2,1,1,0]
sem2_subs_credit_Total = 21

print("\nEnter Internal Continous Assignment Marks(ICA out of 50)")

```

```

print(' & Term End Examination Marks(TEE out of 100): ')

for i in range(0,5):
    print(f'\nMarks in {subject[i]}:')

    icaM = int(input('ICA: '))
    if 0 <= icaM <= 50:
        ICA.append(icaM)
    else:
        print('Entered marks out of range\nEnter ICA in the range of 50')
        icaM = int(input('Valid ICA: '))
        if 0 <= icaM <= 50:
            ICA.append(icaM)
        else:
            print('Invalid ICA marks')

    teeM = int(input('TEE: '))
    if 0 <= teeM <= 100:
        TEE.append(teeM)
    else:
        print('Entered marks out of range\nEnter TEE in the range of 100')
        teeM = int(input('TEE: '))
        if 0 <= teeM <= 100:
            TEE.append(teeM)
        else:
            print('Invalid TEE marks')

for i in range(0,5):
    TEE[i]=TEE[i]/2
for i in range(5,9):
    TEE.append(0)

for j in range(5,9):
    print(f'\nMarks in {subject[j]}:')
    icaM = int(input('ICA: '))
    if 0<= icaM <= 50:
        ICA.append(icaM)
    else:
        print('Entered marks out of range\nEnter ICA in the range of 50')
        icaM = int(input('ICA: '))
        if 0<= icaM <= 50:
            ICA.append(icaM)
        else:
            print('Invalid ICA marks')

```

```

for j in range(5,9):
    ICA[j]= ICA[j]*2

for k in range(0,9):
    TotalM.append(ICA[k] + TEE[k])
#round TM
for d in TotalM:
    D = round_TM(d) # Function call
    rounded_TM.append(D)

for l in range(0,9):
    data.append(rounded_TM[l]) # Storing marks in File

for i in range(5,9):
    TEE[i]=22
    ICA[i]= ICA[i]/2

for i in range(0,9):
    if (ICA[i] >= 20):
        ICA_P_F.append('P')
    elif (ICA[i] < 20):
        ICA_P_F.append('F')

for i in range(0,9):
    if (TEE[i] >= 20):
        TEE_P_F.append('P')
    elif (TEE[i] < 20):
        TEE_P_F.append('F')

for i in range(0,9):
    if (TotalM[i] >= 50):
        TM_P_F.append('P')
    elif (TotalM[i] < 50):
        TM_P_F.append('F')

for i in range(0,9):
    if (ICA_P_F[i] == 'P' and TEE_P_F[i] == 'P' and TM_P_F[i] == 'P'):
        final.append('Pass')
    else:
        final.append('Fail')

dt = [final, ICA_P_F, TEE_P_F, TM_P_F]

for T in rounded_TM:
    grade_point(T) #function call

```

```

for c in range(0,9):
    credit_multiplied.append(sem2_subs_credit[c] * gp_multiplied[c])

for s in credit_multiplied:
    sum+=s

gpa = sum / sem2_subs_credit_Total
data.append(gpa)

for i in range(4):
    data.append(dt[i])

with open('sem2_gpa.csv', 'a') as f_object:
    writer(f_object).writerow(data)
    f_object.close()

print("\n``````````")
print("GPA=",gpa)
comment(gpa)

if op ==2:
    sm=int(input("\nSelect Semester (1. Sem1 || 2. Sem2): "))

    # Set options to display all columns and rows
    pd.set_option('display.max_columns', None)
    pd.set_option('display.max_rows', None)
    pd.set_option('display.width', None)
    pd.set_option('display.max_colwidth', None)

    if sm==1:
        dt = pd.read_csv('sem1_gpa.csv')
        subject=['CALCULUS','C++','BEEE','BIOLOGY','EGD','ENGLISH',
                'ETHICS','CONSTITUTION','DESIGN THINKING']

    elif sm==2:
        dt = pd.read_csv('sem2_gpa.csv')
        subject=['LADE','Physics','DCCA','PEM','Python',
                'EVS','DML','EEWS','Critical Thinking ']

    print(f'\nData of Registered students (SEM {sm}): ')
    columns_drop = ['ICA_Pass_Fail','TEE_Pass_Fail','TotalM_Pass_Fail']
    dt.drop(columns_drop, axis=1, inplace=True)
    print(dt,'\n') # complete data of CSV

```

```

df = pd.DataFrame(dt)
columns_to_drop = ['Date & Time', 'Name', 'SAP ID', 'GPA']
df.drop(columns_to_drop, axis=1, inplace=True)
# Cleaned data of CSV

print('    ~ ~ ~ Analysis: ~ ~ ~')

num_rows = df.shape[0] # students count
print("\n1. Number of students REGISTERED:", num_rows)

print("\n2. Passing/Failing Percentage (Subjectwise): ")
pass_counts = [0] * len(subject)
fail_counts = [0] * len(subject)

sub_pf = dt['Final_Pass_Fail']

# Calculate pass and fail counts for each subject
for row in sub_pf:
    if isinstance(row, str):
        pass_fail_list = eval(row) # Convert string representation to actual list
        if len(pass_fail_list) == len(subject):
            for idx, status in enumerate(pass_fail_list):
                if status == 'Pass':
                    pass_counts[idx] += 1
                elif status == 'Fail':
                    fail_counts[idx] += 1

# Calculate the total number of students
total_students = len(sub_pf)

# Calculate passing and failing percentages for each subject
passing_percentages = [(count / total_students) * 100 for count in pass_counts]
failing_percentages = [(count / total_students) * 100 for count in fail_counts]

# Create a DataFrame to display the results
result_df = pd.DataFrame({
    'Subjects': subject,
    'Passing Percentage': passing_percentages,
    'Failing Percentage': failing_percentages
})
print(result_df)

pass_count=[]
for i in range(len(sub_pf)):
    count = 0

```

```

if sub_pf[i].count('Pass') == 9:
    pass_count.append(1)

passing_percentage = (len(pass_count) / len(sub_pf)) * 100
print("\n3. Overall passing percentage:", passing_percentage, '%')

avg = np.mean(df) # pandas Series, of Average marks of each subjects
print(f"\n4. Average marks scored by all in each SEM {sm} Subjects:")
print(avg)

print("\n5. Failing Count: ")
students_failing_1_subject = 0
students_failing_2_subjects = 0
students_failing_more_than_3_subjects = 0

# Analyze students failing in 1 subject, 2 subjects, and more than 3 subjects
for row in sub_pf:
    fail_count = row.count('Fail')
    if fail_count == 1:
        students_failing_1_subject += 1
    elif fail_count == 2:
        students_failing_2_subjects += 1
    elif fail_count >= 3:
        students_failing_more_than_3_subjects += 1

print(f"Students failing in 1 subject: {students_failing_1_subject}")
print(f"Students failing in 2 subjects: {students_failing_2_subjects}")
print(f"Students failing in more than 3 subjects: {students_failing_more_than_3_subjects}")
print("\n")

# piechart of overall passing and failing percentage (plot 1)
y = np.array([passing_percentage, 100-passing_percentage])
mylabels = ['Passing percentage', 'Failing percentage']
myexplode = [0, 0.1]
plt.pie(y, labels = mylabels, explode = myexplode, startangle = 90, shadow = True)
plt.legend(title = "Overall:")
plt.show()

# linegraph of avg (plot 2)
avg.plot(marker='*', linestyle='--', markersize=10, markeredgecolor='g')
plt.xticks(rotation=45)
plt.xlabel('Subjects', fontsize=15)
plt.ylabel('Marks', fontsize=15)
plt.title('3. Average Marks In Respective Subjects', fontsize=15)
plt.grid()

```



```
plt.show()
```

```
# Reset display options to their defaults
```

```
pd.reset_option('display.max_columns')
```

```
pd.reset_option('display.max_rows')
```

```
pd.reset_option('display.width')
```

```
pd.reset_option('display.max_colwidth')
```

## Chapter 6: Readme Files

### 6.1 Purpose of Readme Files


The purpose of README files is to provide essential information and instructions about a project's setup, usage, and other relevant details for users and developers.

Semester I		
Sr. No	Course Name	Credit/s
1	Calculus	4
2	English Communication	1
3	Basic Electrical and Electronics Engineering	3
4	Programming for Problem Solving	4
5	Engineering Graphics and Design	3
6	Elements of Biology	3
7	Professional Ethics	1
8	Constitution of India	0
9	Design Thinking	0
	<b>Total</b>	<b>19</b>

Figure 1 Semester 1 subjects of CSE DS

Semester II		
Sr. No	Course Name	Credit/s
1	Linear Algebra and Differential Equations	4
2	Physics	4
3	Digital Circuits and Computer Architecture	4
4	Principles of Economics and Management	3
5	Environmental Science	2
6	Python for data analysis	2
7	Digital Manufacturing Laboratory	1
8	Electrical and Electronics Workshop	1
9	Critical Thinking	0
	<b>Total</b>	<b>21</b>


Figure 2 Semester 2 subjects of CSE DS



## Grading Matrix

Grade	Points	Class interval of marks (%)	
A+	4	100	85
A	3.75	84.99	81
A-	3.5	80.99	77
B+	3.25	76.99	73
B	3	72.99	69
B-	2.75	68.99	65
C+	2.5	64.99	61
C	2.25	60.99	57
C-	2	56.99	50
D	1.5	49.99	40
F	0	39	0

Figure 3 Grade Point Matrix



## Computation of GPA and CGPA (1)

Name of the subject	Credit value	Grades as per result	Grade Point	Calculation (Credit x Grade point)	TOTAL
<b>SEM I subject</b>					
Financial Statement Analysis	4	C-	2	2 x 4	8
Effective communication	4	B-	2.75	4 X 2.75	11
Fundamentals of Business Mathematics	4	A+	4	4 x 4	16
				<b>TOTAL</b>	<b>35</b>
<b>SEM II Subjects</b>					
India Social Political and Economic Sys.	4	C-	2	4 x 2	8
Principles of Business Economics	4	C-	2	4 X 2	8
				<b>TOTAL</b>	<b>16</b>
$\text{GPA Trim I} = \frac{\text{Sum (grade point * Credits)}}{\text{Sum (Total Credits)}} = \frac{35}{12} = 2.91$					
$\text{GPA Trim II} = \frac{\text{Sum (grade point * Credits)}}{\text{Sum (Total Credits)}} = \frac{16}{8} = 2.00$					
$\text{CGPA SEM II} = \frac{\text{Sum (Trim I CG + Trim II CG)}}{\text{Sum (Trim I credits + Trim II Credits)}} = \frac{51}{20} = 2.55$					

Figure 4 Final GPA calculation formula

## 6.2 Explanation of Readme Content

In Figure 1 (Image 1), you will find detailed information about the subjects in Semester 1, including their respective credit values. This serves as a quick reference guide for users to understand the composition of Semester 1's curriculum.

Figure 2 (Image 2) presents the subjects in Semester 2 along with their corresponding credits and the total credits for the semester. This information assists users in comprehending the structure of Semester 2.

Figure 3 (Image 3) provides a comprehensive grading matrix that outlines the relationship between grades, grade points, and the class intervals of marks. It offers a clear understanding of how the final grade is determined based on students' performance.

Lastly, Figure 4 (Image 4) illustrates the step-by-step process of calculating GPA and CGPA for each subject. This guide is a valuable resource for students who wish to calculate their GPA manually and understand the underlying calculations.

## *Chapter 7: User Manual*

### 7.1 Installation Instructions

Use the GPA Calculator and Result Analysis tool, follow these installation instructions:

- i. **Clone the Repository:** Start by cloning the project's GitHub repository to your local machine. You can do this by running the following command in your terminal or command prompt:

```
git clone [!!!repository URL!!!]
```

- ii. **Python Environment:** Ensure that you have Python installed on your computer. This project was developed using Python. Preferred IDE – Spyder.
- iii. **Change Working Directory:** The 'OS' library in Python is used to change your working directory to the location where you cloned the repository. Here is the syntax for changing the working directory:

```
# in line no: 8  
os.chdir('path_to_project_directory')
```

- iv. **CSV Files:** Place the necessary CSV files in the project directory. The tool expects two CSV files with specific names: 'sem1\_gpa.csv' and 'sem2\_gpa.csv'. If you use different file names, make sure to update the file names within the code where they are accessed (line no: 195, 337; 319, 342 respectively).

- v. **Review Code Instructions:** Carefully review the code for this project to understand how it works and what it does. You can find comments and explanations within the code to guide you.

## 7.2 Using the GPA Calculator

### Input Data:

To use the GPA Calculator, ensure you have met the installation requirements mentioned above.

Run the Python script associated with this project.

Follow the on-screen prompts and input the requested information, such as your name, SAP ID, and semester choice (Semester 1 or Semester 2).

To predict GPA, give ICA (out of 50) and TEE exam marks (out of 100) for each subject.

### 7.3 Interpreting the Results:

The tool will display various results and analyses based on the provided data. Here is a brief explanation of what you will see:

- **Passing/Failing Percentage (Subject wise):** This section displays the passing and failing percentages for each subject in the selected semester.
- **Overall Passing Percentage:** The overall passing percentage for the selected semester is shown.
- **Average Marks:** You will see the average marks scored by all students in each subject for the chosen semester.
- **Failing Count:** The tool analyzes the number of students is failing in one subject, two subjects, or more than three subjects.

### CSV File Columns:

*Date & Time:* The date and time when the data was recorded.

*Name:* The name of the student.

*SAP ID:* The student's SAP ID.

*Marks in Subjects:* Next columns represent the individual subjects Final Marks for the semester.

*GPA:* The calculated Grade Point Average for the semester.

*Final\_Pass\_Fail*: Displays 'Pass' or 'Fail' for each subject if a student passes in ICA, TEE, and Final Marks.

*ICA\_Pass\_Fail*: Indicates whether the student passed or failed the Internal Continuous Assessment (ICA) for each subject. It will display 'Pass' if  $ICA > 20$ , otherwise 'Fail'.

*TEE\_Pass\_Fail*: Indicates whether the student passed or failed the Term End Examination (TEE) for each subject. It will display 'Pass' if  $TEE > 40$ , otherwise 'Fail'.

*TotalM\_Pass\_Fail*: Indicates whether the student passed or failed based on the total marks ( $ICA + TEE$ )  $> 50$  for each subject. It will display 'Pass' if the condition is met, otherwise 'Fail'.

## **PLOTS:**

With the analysis of that semester data, we plot a pie chart and a line graph.

- **Pie Chart**: Illustrating the overall passing and failing percentages.
- **Line Graph**: Displays the average marks obtained in respective subjects.

Please note that this tool is for educational purposes only and should be used responsibly.

## ***Chapter 8: Conclusion and Future Scope***

### **8.1 Summary of Achievements**

In the development of the GPA Calculator and Result Analysis project, several significant achievements have been realized:

**Automation of GPA Calculations**: We successfully automated the complex and time-consuming process of GPA calculations. This not only saves time but also reduces the chances of manual errors.

**Comprehensive Result Analysis**: Our project provides comprehensive result analysis, including passing/failing percentages, average marks, and detailed subject-wise data visualization.

**User-Friendly Interface**: The project offers an intuitive user interface, making it accessible for students to calculate their GPA effortlessly.

**Educational Insights**: It provides educational insights by visualizing academic performance, helping students identify areas for improvement.

**Contribution to Education**: This project contributes to educational institutions by simplifying the evaluation process and offering valuable insights into student performance.

## 8.2 Limitations and Challenges

While we have achieved significant milestones in this project, it is essential to acknowledge the limitations and challenges we encountered:

**Dependence on Data Entry:** The accuracy of GPA calculations relies on the accurate entry of ICA and TEE marks by students. Any discrepancies in data entry can affect the results.

**Data Availability:** The project requires access to student data, which might be subject to privacy and authorization constraints.

**Initial Setup:** Users need to clone the project's GitHub repository and ensure Python is installed in their environment. Additionally, they should correctly set the working directory and maintain the naming convention for CSV files.

**Single-Semester Analysis:** The current version of the project supports GPA calculations and result analysis for one semester at a time. Expanding it to accommodate multiple semesters is a potential future enhancement.

## 8.3 Future Enhancements and Extensions

The GPA Calculator and Result Analysis project hold promise for future enhancements and extensions:

**Multi-Semester Support:** Extend the project to handle data from multiple semesters, enabling students to track their academic progress across their entire course duration.

**Machine Learning Integration:** Incorporate machine learning algorithms for predictive analytics, allowing students to forecast their future academic performance based on past data.

**Cloud Deployment:** Move the project to a cloud-based platform for greater accessibility and scalability, ensuring students can use it from anywhere.

**Web Application:** Transform the project into a web application with an intuitive graphical user interface, eliminating the need for command-line interactions.

**Collaborative Features:** Enable students to collaborate on result analysis and share insights with peers, fostering a sense of community and mutual support.

The GPA Calculator and Result Analysis project have the potential to evolve into a comprehensive educational tool, benefiting both students and educational institutions. As technology advances and user needs change, we anticipate further innovations and enhancements in the realm of academic performance analysis.