Fuzzy Logic Based Student Academic Performance Analyzing System

A MINI PROJECT REPORT
18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

Shrey Nagori (RA2111032010008)
Alok Agnihotri (RA2111032010010)
Lakshya Kishnani (RA2111032010015)

Under the guidance of

Dr. C Fancy
Assistant Professor, Department of Networking and Communications

in partial fulfillment for the award of the degree
of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING
of
FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project titled “Fuzzy Logic based Student Academic Performance Analyzing System” is the bonafide work of **Shrey Nagori (RA2111032010008), Alok Agnihotri (RA2111032010010), Lakshya Kishnani (RA2111032010015)** who carried out the mini project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or reward was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. C Fancy
Assistant Professor
Department of Networking and Communications

ABSTRACT

The Fuzzy Logic Based Student Academic Performance Analyzing System is a software project aimed at developing an advanced tool for evaluating and analyzing students' academic performance using fuzzy logic techniques. This system provides a more nuanced and flexible approach to assessing student performance compared to traditional grading systems. It incorporates various input variables such as exam scores, attendance records, participation in extracurricular activities, etc., and employs fuzzy logic rules to derive conclusions about students' performance. The project aims to enhance decision-making by educators and administrators by providing personalized feedback and identifying students who may need additional support or intervention.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The landscape of education is evolving with a growing emphasis on personalized and nuanced assessments of student performance. Traditional grading systems often rely heavily on exam scores, which may not provide a holistic view of a student's capabilities and potential. To address this limitation, the Fuzzy Logic Based Student Academic Performance Analyzing System has been developed. This system leverages fuzzy logic techniques to create a more comprehensive framework for evaluating student performance, taking into account various factors such as attendance, internal and external exam marks.

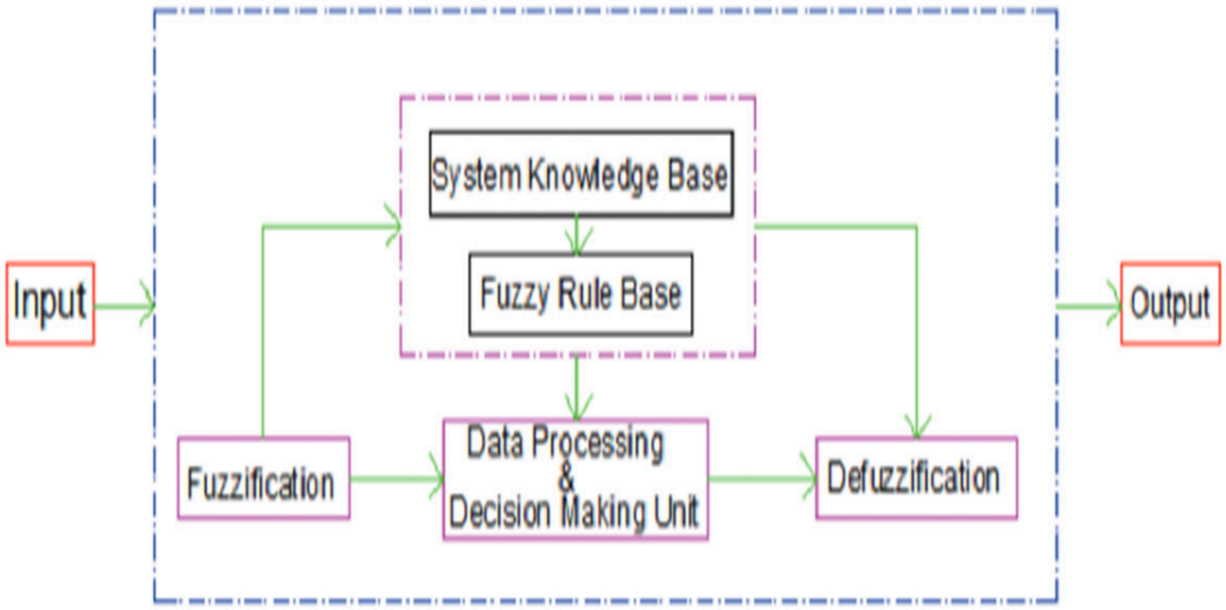


Figure 1 : General Architecture Diagram of Fuzzy Logic Based Systems

The heart of this project lies in its ability to interpret and analyze complex data sets using fuzzy logic. By defining linguistic variables like 'Student_Attendance', 'Student_Internal_Marks', 'Student_External_Marks', and 'Student_Performance', we establish a flexible and adaptable system. Each of these variables is further broken down into categories such as 'Poor', 'Average', 'Good', 'Very_Good', and 'Excellent', with membership functions defined to capture the nuances of each category.

The system operates on a set of fuzzy rules that map the input variables to the output, 'Student_Performance'. These rules are designed to mimic human-like reasoning, allowing the system to make informed assessments based on the input data. For example, a student with high attendance, good internal and external marks would likely be categorized as 'Very_Good' or 'Excellent' in terms of performance.
In this project report, we will delve into the design, implementation, and functionality of the Fuzzy Logic Based Student Academic Performance Analyzing System. We will explore the intricacies of fuzzy logic, the definition of linguistic variables, and the formulation of fuzzy rules. Furthermore, we will demonstrate the system's capabilities through examples and discuss its potential applications in educational settings. Through this exploration, we aim to showcase how this innovative approach can revolutionize the assessment of student performance, offering educators and administrators a more nuanced tool for decision-making and support strategies.

CHAPTER 2

LITERATURE SURVEY

The literature survey for this project involves a comprehensive review of existing research and methodologies related to fuzzy logic and its application in analyzing student academic performance.

Fuzzy Logic

Fuzzy Logic, introduced by Lotfi Zadeh in the 1960s, is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1. It is employed to handle the concept of partial truth, where the truth value may range between completely true and completely false.

Fuzzy Logic in Education

The application of fuzzy logic in the field of education is a growing area of research. Several studies have explored the use of fuzzy logic for various purposes such as decision-making, performance evaluation, and system control in an educational context.

Student Performance Analysis

Analyzing student performance is a critical aspect of educational research. Traditional methods of student performance analysis often rely on crisp sets of data such as grades or scores. However, these methods may not provide a holistic view of a student's performance.

Fuzzy Logic in Student Performance Analysis

The use of fuzzy logic in student performance analysis provides a more nuanced and comprehensive view of student performance. By considering various factors such as attendance, participation, and exam scores, fuzzy logic can provide a more accurate representation of a student's overall academic performance.

Existing Systems

Several systems have been developed that employ fuzzy logic for student performance analysis. These systems take into account various factors and use fuzzy logic rules to derive conclusions about a student's performance.

In conclusion, the literature suggests that fuzzy logic holds significant potential for analyzing student performance in a more comprehensive and nuanced manner. This project aims to explore this potential further by developing a Fuzzy Logic Based Student Academic Performance Analyzing System.

References:

1. Zadeh, L.A. (1965). Fuzzy sets. Information and Control, 8(3), 338-353.
2. Sanchez, E.N. (2017). Applications of Type-2 Fuzzy Logic in Education. In: Castillo O., Melin P., Kacprzyk J. (eds) Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications. Studies in Computational Intelligence, vol 648. Springer.
3. Kotsiantis, S.B., Pierrakeas, C.J., & Pintelas, P.E. (2003). Predicting students' marks in Hellenic Open University. In Proceedings of the 5th International Conference on Advanced Learning Technologies (ICALT'04).
4. Lykourentzou, I., Giannoukos, I., Nikolopoulos, V., Mpardis, G., & Loumos, V. (2009). Dropout prediction in e-learning courses through the combination of machine learning techniques. Computers & Education, 53(3), 950-965.
5. M. Kabak, Y. Burmaoglu and U. D. Aladag, "A new hybrid model based on fuzzy-artificial immune system and k-nn algorithm for breast cancer diagnosis," Computers in Human Behavior, vol. 34, pp. 284-295, 2014.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 Overview:

The architecture and design of the Fuzzy Logic Based Student Academic Performance Analyzing System have been meticulously crafted to ensure efficiency, flexibility, and scalability. This chapter provides an in-depth look into the various components that constitute the system, their interactions, and the rationale behind their design choices.

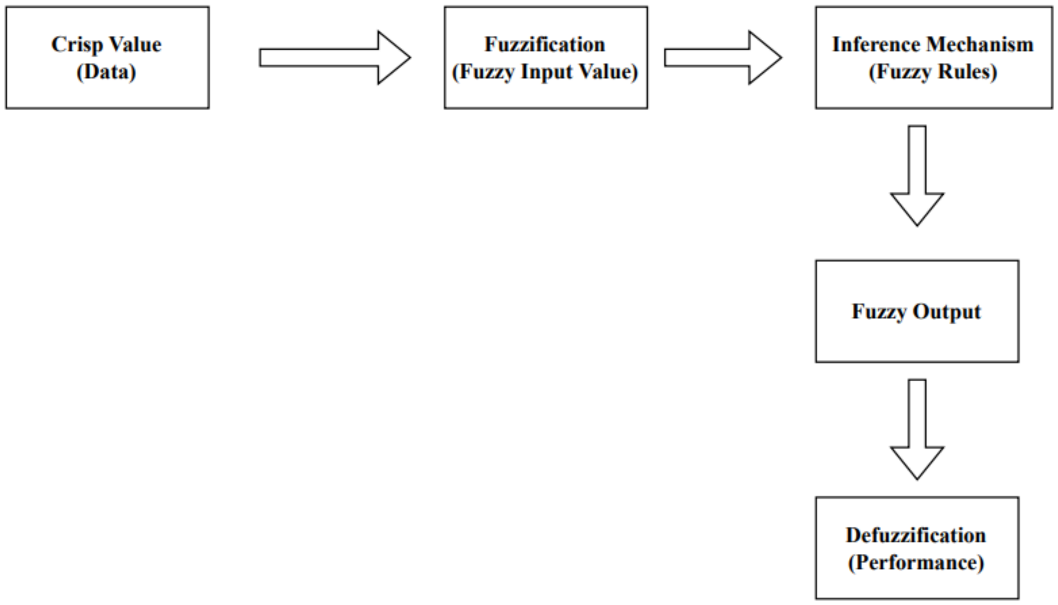

Crisp Value (Data)
Fuzzification (Fuzzy Input Value)
Inference Mechanism (Fuzzy Rules)
Fuzzy Output
Defuzzification (Performance)

Figure 2: Architecture Diagram of “Fuzzy Logic Based Student Academic Performance Analyzing” System

3.2 System Components:

Linguistic Variables and Membership Functions
At the core of the system's design are the linguistic variables and their corresponding membership functions. These variables, namely 'Student_Attendance', 'Student_Internal_Marks', 'Student_External_Marks', and 'Student_Performance', serve as the inputs and outputs for the fuzzy logic system.

Student_Attendance: This variable captures the attendance percentage of a student, ranging from 0 to 100.
Student_Internal_Marks: Represents the internal examination scores, which contribute to the overall academic performance.
Student_External_Marks: Reflects the scores obtained in external examinations, providing another dimension to the student's academic profile.
Student_Performance: The output variable that signifies the overall academic performance, categorized as 'Poor', 'Average', 'Good', 'Very_Good', and 'Excellent'.
Membership functions for each linguistic variable have been defined using trapezoidal fuzzy sets. This choice allows for a smooth transition between categories and captures the inherent uncertainty and imprecision in the input data.

3.3 Fuzzy Rules and Inference Engine:

The fuzzy rules form the backbone of the system's decision-making process. These rules are designed to map the input variables to the output, determining the student's performance category based on the provided data.

The inference engine, powered by the control system and simulation from the scikit-fuzzy library, processes these rules and computes the output. It mimics human-like reasoning by evaluating the degree to which each rule is satisfied and aggregating the results to produce a crisp output value for 'Student_Performance'.

3.4 Interaction and Flow:

The flow of information within the system is orchestrated through a sequence of steps:

Input Acquisition: The system collects input data for 'Student_Attendance', 'Student_Internal_Marks', and 'Student_External_Marks' from the user or an integrated data source.
Fuzzification: The input data are then fuzzified using the predefined membership functions, converting crisp values into fuzzy sets.
Rule Evaluation: The fuzzified inputs are evaluated against the fuzzy rules to determine the student's performance category.
Defuzzification: The output fuzzy set is then defuzzified to produce a crisp 'Student_Performance' value, which is returned to the user or stored for further analysis.

3.5 Scalability and Extensibility:

The modular design of the system allows for easy scalability and extensibility. New linguistic variables, membership functions, or fuzzy rules can be added with minimal modifications to the existing codebase. This flexibility ensures that the system can adapt to evolving educational requirements and incorporate additional factors that may influence student performance in the future.

CHAPTER 4

METHODOLOGY

The methodology of the Fuzzy Logic Based Student Academic Performance Analyzing System involves several steps, which are detailed below:

Step 1: Defining the Linguistic Variables

The first step in the methodology is to define the linguistic variables. In this project, the linguistic variables are Student Attendance, Student Internal Marks, and Student External Marks. These variables represent the input to the fuzzy logic system.

Step 2: Defining the Membership Functions

The next step is to define the membership functions for the linguistic variables. The membership functions are defined using trapezoidal membership functions with parameters representing the range of each category. The categories used in this project are Poor, Average, Good, Very Good, and Excellent.

Step 3: Defining the Fuzzy Rules

The fuzzy rules are then defined based on the linguistic variables and their membership functions. These rules form the basis of the fuzzy inference system. They are defined in such a way that they cover all possible combinations of the input variables.

Step 4: Creating the Control System

The control system is created using the defined fuzzy rules. This system is responsible for processing the inputs according to the rules and generating the output.

Step 5: Running the Simulation

The final step is to run the simulation of the control system. The input values for the simulation are the actual values of the linguistic variables. The simulation computes the output, which represents the student's performance.

The Python code provided implements these steps using the numpy and skfuzzy libraries. The compute_student_performance function takes the student's attendance, internal marks, and external marks as input, and returns the computed student performance as output.

This methodology allows for a more nuanced and flexible assessment of students' academic performance compared to traditional grading systems. It provides a more comprehensive and accurate assessment of students' academic performance, offers personalized feedback to students based on their individual strengths and weaknesses, and assists educators and administrators in making informed decisions about student support and intervention strategies. By leveraging fuzzy logic techniques, the system aims to provide a more accurate and personalized assessment of students' performance, ultimately contributing to improved learning outcomes.

The system can be implemented using programming languages such as Python or Java. It can be deployed as a standalone software application or integrated into existing educational management systems used by schools or universities. The Fuzzy Logic Based Student Academic Performance Analyzing System offers a novel approach to evaluating students' academic performance and supporting decision-making in educational institutions.

CHAPTER 5
CODING AND TESTING

In the development of the Fuzzy Logic Based Student Academic Performance Analyzing System, meticulous attention was dedicated to both the coding implementation and rigorous testing procedures. This chapter elucidates the coding methodologies employed and the comprehensive testing strategies enacted during the development phase.

5.1 Coding Implementation:

The core of the system was coded in Python, a versatile and widely-used programming language, renowned for its simplicity and rich ecosystem of libraries. The implementation encapsulates the following key components:

Defining Necessary Libraries:

The code begins by importing essential libraries, including NumPy for numerical computations and scikit-fuzzy for fuzzy logic operations.

Defining Linguistic Variables and Membership Functions:

Linguistic variables such as student attendance, internal marks, and external marks are defined along with their corresponding membership functions. These functions categorize inputs into linguistic terms like 'Poor,' 'Average,' 'Good,' 'Very Good,' and 'Excellent.'

Fuzzy Logic Engine:

A dedicated function, compute_student_performance, orchestrates the fuzzy logic engine. It accepts inputs related to student attendance, internal marks, and external marks, computes the student performance using defined fuzzy rules, and yields an output indicating the student's performance level.

<u>Example Usage:</u>

The code concludes with an example usage snippet demonstrating the application of the compute_student_performance function with sample input values.

<u>The code of the system is mentioned below :-</u>

```
# Defining and importing necessary libraries
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# Defining the linguistic variables
STUDENT_ATTENDANCE = 'Student_Attendance'
STUDENT_PERFORMANCE = 'Student_Performance'
STUDENT_INTERNAL_MARKS = 'Student_Internal_Marks'
STUDENT_EXTERNAL_MARKS = 'Student_External_Marks'

# Defining the membership functions for the linguistic variables
POOR = 'Poor'
AVERAGE = 'Average'
GOOD = 'Good'
VERY_GOOD = 'Very_Good'
EXCELLENT = 'Excellent'
low_parameter = [0, 0, 40, 50]
average_parameter = [30, 40, 50, 60]
good_parameter = [40, 50, 60, 70]
very_good_parameter = [50, 60, 70, 80]
excellent_parameter = [65, 80, 100, 100]

# Function to compute the student performance using fuzzy logic defined
def compute_student_performance(student_attendance, student_internal_marks,
student_external_marks):
    # Defining the antecedents and consequent
    internal_marks = ctrl.Antecedent(np.arange(0, 105, 5),
STUDENT_INTERNAL_MARKS)
    attendance = ctrl.Antecedent(np.arange(0, 105, 5), STUDENT_ATTENDANCE)
    external_marks = ctrl.Antecedent(np.arange(0, 105, 5),
STUDENT_EXTERNAL_MARKS)
    student_performance = ctrl.Consequent(np.arange(0, 105, 5),
STUDENT_PERFORMANCE)

    # Setting the membership functions for the linguistic variables
    internal_marks[POOR] = fuzz.trapmf(internal_marks.universe, low_parameter)
    internal_marks[AVERAGE] = fuzz.trapmf(internal_marks.universe,
average_parameter)
    internal_marks[GOOD] = fuzz.trapmf(internal_marks.universe, good_parameter)
    internal_marks[VERY_GOOD] = fuzz.trapmf(internal_marks.universe,
very_good_parameter)
    internal_marks[EXCELLENT] = fuzz.trapmf(internal_marks.universe,
excellent_parameter)

    attendance[POOR] = fuzz.trapmf(attendance.universe, [0, 0, 45, 55])
    attendance[AVERAGE] = fuzz.trapmf(attendance.universe, [35, 45, 55, 65])
    attendance[GOOD] = fuzz.trapmf(attendance.universe, [45, 55, 65, 75])
    attendance[VERY_GOOD] = fuzz.trapmf(attendance.universe, [55, 65, 75, 85])
    attendance[EXCELLENT] = fuzz.trapmf(attendance.universe, [65, 75, 100, 100])

    external_marks[POOR] = fuzz.trapmf(external_marks.universe, low_parameter)
    external_marks[AVERAGE] = fuzz.trapmf(external_marks.universe,
average_parameter)
    external_marks[GOOD] = fuzz.trapmf(external_marks.universe, good_parameter)
    external_marks[VERY_GOOD] = fuzz.trapmf(external_marks.universe,
very_good_parameter)
    external_marks[EXCELLENT] = fuzz.trapmf(external_marks.universe,
excellent_parameter)

    student_performance[POOR] = fuzz.trapmf(student_performance.universe,
low_parameter)
    student_performance[AVERAGE] = fuzz.trapmf(student_performance.universe,
average_parameter)
    student_performance[GOOD] = fuzz.trapmf(student_performance.universe,
good_parameter)
    student_performance[VERY_GOOD] = fuzz.trapmf(student_performance.universe,
very_good_parameter)
    student_performance[EXCELLENT] = fuzz.trapmf(student_performance.universe,
excellent_parameter)

    # Defining the fuzzy rules
    rule1 = ctrl.Rule(attendance[POOR] & external_marks[POOR] &
internal_marks[POOR], student_performance[POOR])
    rule2 = ctrl.Rule(attendance[POOR] & external_marks[AVERAGE] &
internal_marks[POOR], student_performance[POOR])
    rule3 = ctrl.Rule(attendance[POOR] & external_marks[GOOD] &
internal_marks[POOR], student_performance[AVERAGE])
    rule4 = ctrl.Rule(attendance[POOR] & external_marks[VERY_GOOD] &
internal_marks[POOR], student_performance[AVERAGE])
    rule5 = ctrl.Rule(attendance[POOR] & external_marks[GOOD] &
internal_marks[VERY_GOOD], student_performance[GOOD])
    rule6 = ctrl.Rule(attendance[POOR] & external_marks[POOR] &
internal_marks[AVERAGE], student_performance[POOR])
    rule7 = ctrl.Rule(attendance[POOR] & external_marks[AVERAGE] &
internal_marks[AVERAGE], student_performance[AVERAGE])
    rule8 = ctrl.Rule(attendance[POOR] & external_marks[GOOD] &
internal_marks[AVERAGE], student_performance[AVERAGE])
```

```
 rule9 = ctrl.Rule((attendance[POOR] & external_marks[GOOD] &
internal_marks[GOOD]), student_performance[GOOD])
   rule10 = ctrl.Rule(attendance[POOR] & external_marks[EXCELLENT] &
internal_marks[GOOD], student_performance[VERY_GOOD])
   rule11 = ctrl.Rule(attendance[AVERAGE] & external_marks[AVERAGE] &
internal_marks[GOOD], student_performance[AVERAGE])
   rule12 = ctrl.Rule(attendance[AVERAGE] & external_marks[GOOD] &
internal_marks[GOOD], student_performance[GOOD])
   rule13 = ctrl.Rule(attendance[AVERAGE] & external_marks[VERY_GOOD] &
internal_marks[GOOD], student_performance[GOOD])
   rule14 = ctrl.Rule(attendance[AVERAGE] & external_marks[VERY_GOOD] &
internal_marks[VERY_GOOD], student_performance[VERY_GOOD])
   rule15 = ctrl.Rule(attendance[AVERAGE] & external_marks[AVERAGE] &
internal_marks[EXCELLENT], student_performance[GOOD])
   rule16 = ctrl.Rule(attendance[AVERAGE] & external_marks[AVERAGE] &
internal_marks[AVERAGE], student_performance[AVERAGE])
   rule17 = ctrl.Rule(attendance[AVERAGE] & external_marks[POOR] &
internal_marks[POOR], student_performance[POOR])
   rule18 = ctrl.Rule(attendance[AVERAGE] & external_marks[POOR] &
internal_marks[GOOD], student_performance[AVERAGE])
   rule19 = ctrl.Rule(attendance[GOOD] & external_marks[AVERAGE] &
internal_marks[AVERAGE], student_performance[AVERAGE])
   rule20 = ctrl.Rule(attendance[GOOD] & external_marks[EXCELLENT] &
internal_marks[EXCELLENT], student_performance[VERY_GOOD])
   rule21 = ctrl.Rule(attendance[GOOD] & external_marks[GOOD] &
internal_marks[AVERAGE], student_performance[GOOD])
   rule22 = ctrl.Rule(attendance[GOOD] & external_marks[POOR] &
internal_marks[POOR], student_performance[POOR])
   rule23 = ctrl.Rule(attendance[VERY_GOOD] & external_marks[EXCELLENT] &
internal_marks[VERY_GOOD], student_performance[VERY_GOOD])
   rule24 = ctrl.Rule(attendance[VERY_GOOD] & external_marks[VERY_GOOD] &
internal_marks[VERY_GOOD], student_performance[VERY_GOOD])
```

```
 rule25 = ctrl.Rule(attendance[VERY_GOOD] & external_marks[POOR] &
internal_marks[POOR], student_performance[POOR])
   rule26 = ctrl.Rule(attendance[VERY_GOOD] & external_marks[GOOD] &
internal_marks[VERY_GOOD], student_performance[VERY_GOOD])
   rule27 = ctrl.Rule(attendance[VERY_GOOD] & external_marks[EXCELLENT] &
internal_marks[EXCELLENT], student_performance[EXCELLENT])
   rule28 = ctrl.Rule(attendance[EXCELLENT] & external_marks[EXCELLENT] &
internal_marks[VERY_GOOD], student_performance[VERY_GOOD])
   rule29 = ctrl.Rule(attendance[EXCELLENT] & external_marks[AVERAGE] &
internal_marks[AVERAGE], student_performance[VERY_GOOD])
   rule30 = ctrl.Rule(attendance[EXCELLENT] & external_marks[AVERAGE] &
internal_marks[VERY_GOOD], student_performance[GOOD])
   rule31 = ctrl.Rule(attendance[EXCELLENT] & external_marks[AVERAGE] &
internal_marks[GOOD], student_performance[GOOD])
   rule32 = ctrl.Rule(attendance[EXCELLENT] & external_marks[POOR] &
internal_marks[POOR], student_performance[POOR])
   rule33 = ctrl.Rule(attendance[EXCELLENT] & external_marks[AVERAGE] &
internal_marks[POOR], student_performance[AVERAGE])
   rule34 = ctrl.Rule(attendance[EXCELLENT] & external_marks[POOR] &
internal_marks[AVERAGE], student_performance[POOR])
   rule35 = ctrl.Rule(attendance[EXCELLENT] & external_marks[GOOD] &
internal_marks[POOR], student_performance[GOOD])
   rule36 = ctrl.Rule(attendance[EXCELLENT] & external_marks[POOR] &
internal_marks[GOOD], student_performance[AVERAGE])
   rule37 = ctrl.Rule(attendance[EXCELLENT] & external_marks[VERY_GOOD] &
internal_marks[POOR], student_performance[VERY_GOOD])
   rule38 = ctrl.Rule(attendance[EXCELLENT] & external_marks[POOR] &
internal_marks[VERY_GOOD], student_performance[AVERAGE])
   rule39 = ctrl.Rule(attendance[EXCELLENT] & external_marks[POOR] &
internal_marks[EXCELLENT], student_performance[GOOD])
   rule40 = ctrl.Rule(attendance[EXCELLENT] & external_marks[AVERAGE] &
internal_marks[EXCELLENT], student_performance[VERY_GOOD])
```

```
 rule41 = ctrl.Rule(attendance[EXCELLENT] & external_marks[GOOD] &
internal_marks[EXCELLENT], student_performance[VERY_GOOD])
   rule42 = ctrl.Rule(attendance[EXCELLENT] & external_marks[VERY_GOOD] &
internal_marks[EXCELLENT], student_performance[VERY_GOOD])
   rule43 = ctrl.Rule(attendance[EXCELLENT] & external_marks[EXCELLENT] &
internal_marks[EXCELLENT], student_performance[EXCELLENT])

   # Creating a list of rules
   rule_list = [rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9, rule10, rule11,
rule12, rule13, rule14, rule15, rule16, rule17, rule18, rule19, rule20, rule21, rule22,
rule23, rule24, rule25, rule26, rule27, rule28, rule29, rule30, rule31, rule32, rule33,
rule34, rule35, rule36, rule37, rule38, rule39, rule40, rule41, rule42, rule43]

   # Creating control system using the rules
   student_performance_ctrl = ctrl.ControlSystem(rule_list)

   # Creating simulation of the control system
   student_performance_analysis =
ctrl.ControlSystemSimulation(student_performance_ctrl)

   # Setting the input values for the simulation
   student_performance_analysis.input[STUDENT_ATTENDANCE] =
student_attendance
   student_performance_analysis.input[STUDENT_EXTERNAL_MARKS] =
student_external_marks
   student_performance_analysis.input[STUDENT_INTERNAL_MARKS] =
student_internal_marks

   # Running the simulation
   student_performance_analysis.compute()

   # Returning the output of the simulation as a string
   return str(student_performance_analysis.output[STUDENT_PERFORMANCE])
```

```
# Example usage of the function
# Printing the output obtained after running the code
print(compute_student_performance(70, 20, 50))
print(compute_student_performance(90, 30, 60))
print(compute_student_performance(95, 90, 80))
```

5.2 Testing Strategies:

A robust testing regime was instituted to ensure the accuracy, reliability, and functionality of the Fuzzy Logic Based Student Academic Performance Analyzing System. The testing process encompassed the following phases:

Unit Testing:

Each component and function within the system underwent rigorous unit testing. Test cases were meticulously designed to cover both typical and edge scenarios, validating the correctness and functionality of individual modules.

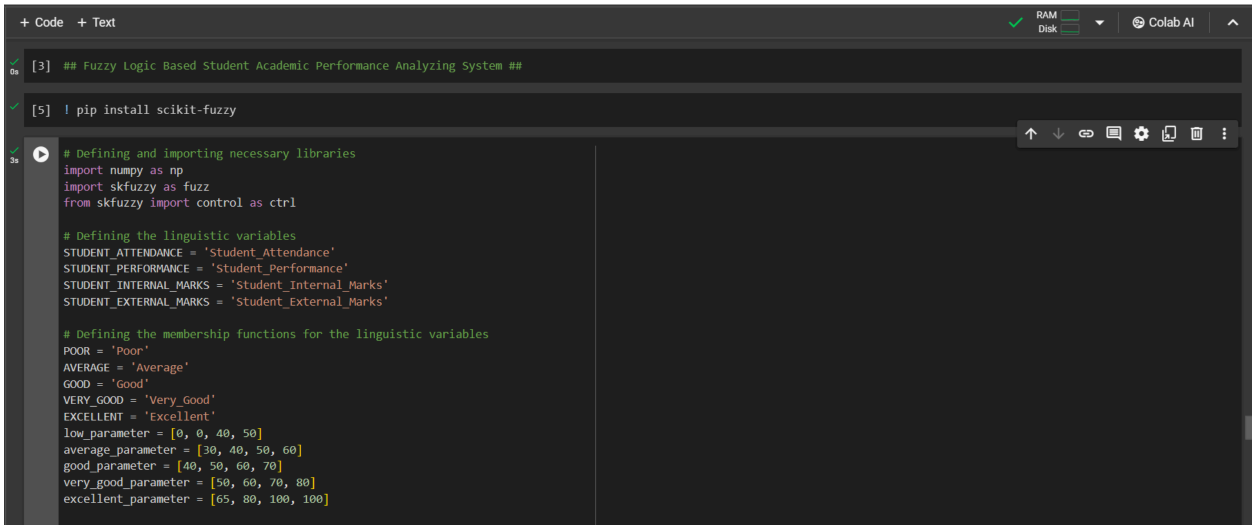
Integration Testing:

Once individual components passed unit testing, integration testing was conducted to evaluate the interactions and interoperability between different modules. Integration tests focused on ensuring seamless data flow and error handling mechanisms.

CHAPTER 6

SCREENSHOTS AND RESULT

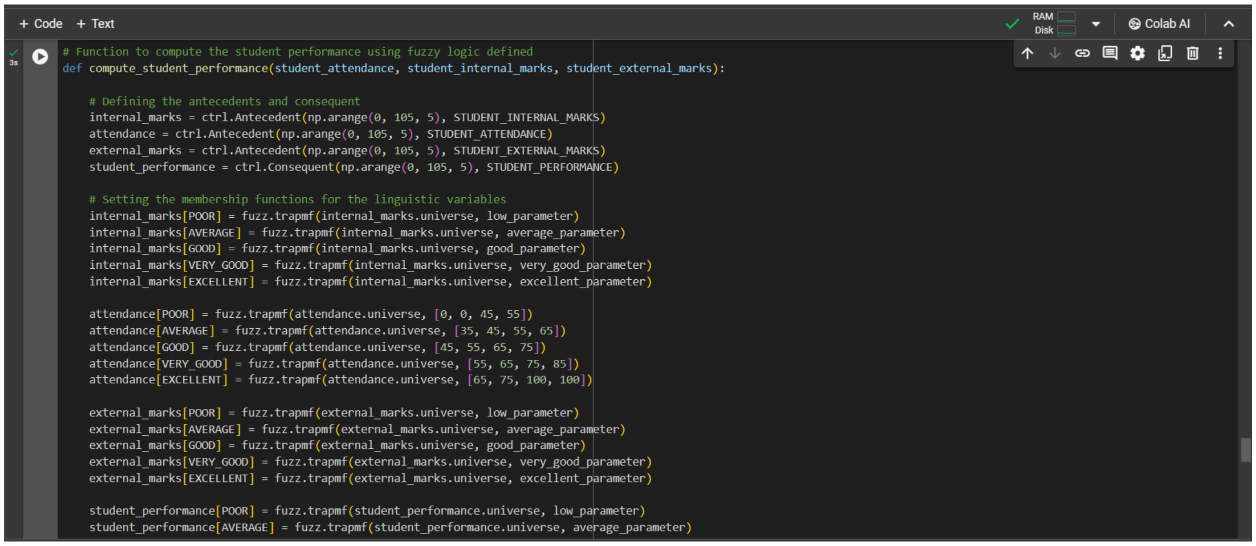
Screenshots of the running code and input used :-



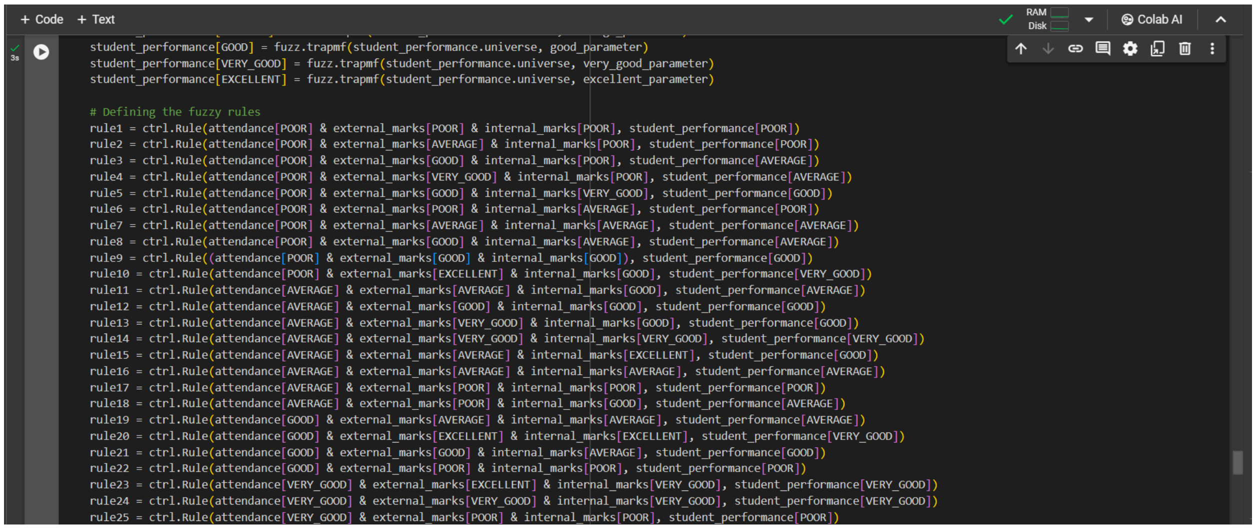
```
[3]  ## Fuzzy Logic Based Student Academic Performance Analyzing System ##
```

```
[5]  ! pip install scikit-fuzzy
```

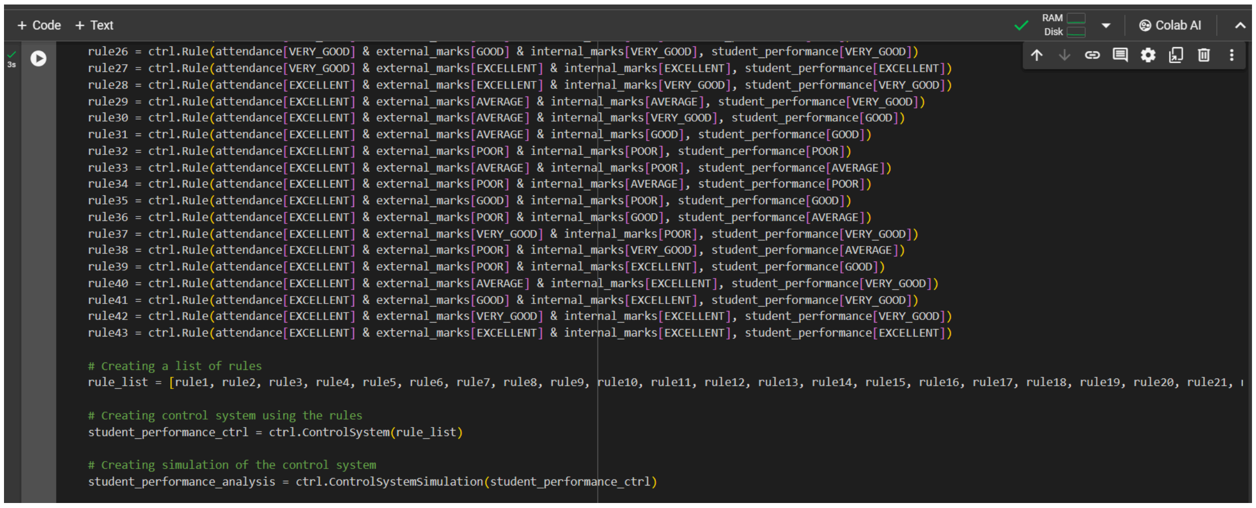
```
# Defining and importing necessary libraries
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# Defining the linguistic variables
STUDENT_ATTENDANCE = 'Student_Attendance'
STUDENT_PERFORMANCE = 'Student_Performance'
STUDENT_INTERNAL_MARKS = 'Student_Internal_Marks'
STUDENT_EXTERNAL_MARKS = 'Student_External_Marks'

# Defining the membership functions for the linguistic variables
POOR = 'Poor'
AVERAGE = 'Average'
GOOD = 'Good'
VERY_GOOD = 'Very_Good'
EXCELLENT = 'Excellent'
low_parameter = [0, 0, 40, 50]
average_parameter = [30, 40, 50, 60]
good_parameter = [40, 50, 60, 70]
very_good_parameter = [50, 60, 70, 80]
excellent_parameter = [65, 80, 100, 100]
```



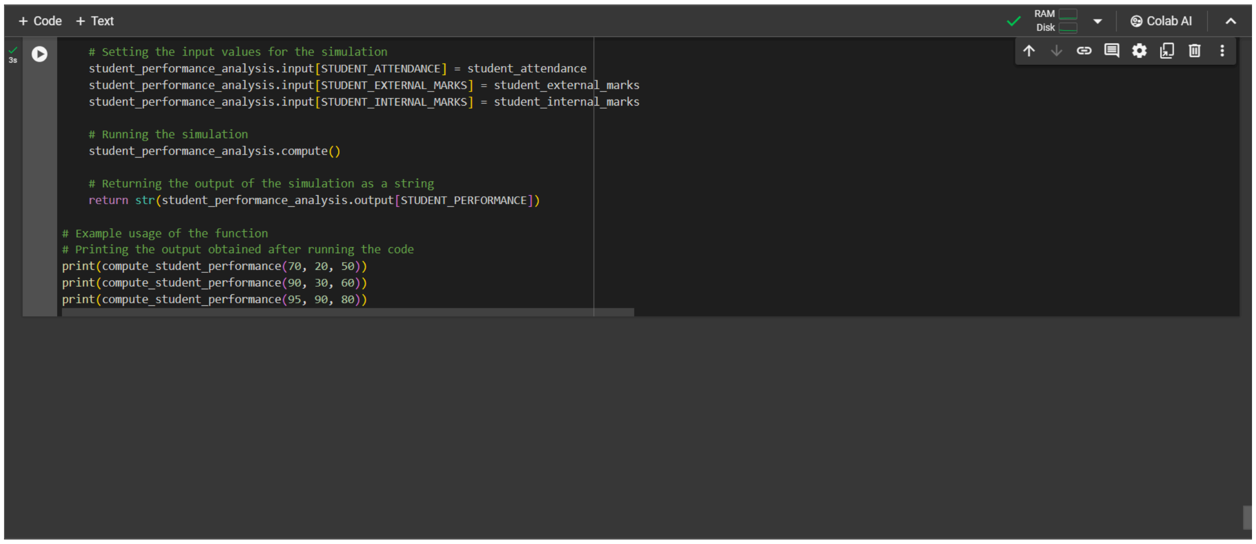
```
# Function to compute the student performance using fuzzy logic defined
def compute_student_performance(student_attendance, student_internal_marks, student_external_marks):

    # Defining the antecedents and consequent
    internal_marks = ctrl.Antecedent(np.arange(0, 105, 5), STUDENT_INTERNAL_MARKS)
    attendance = ctrl.Antecedent(np.arange(0, 105, 5), STUDENT_ATTENDANCE)
    external_marks = ctrl.Antecedent(np.arange(0, 105, 5), STUDENT_EXTERNAL_MARKS)
    student_performance = ctrl.Consequent(np.arange(0, 105, 5), STUDENT_PERFORMANCE)

    # Setting the membership functions for the linguistic variables
    internal_marks[POOR] = fuzz.trapmf(internal_marks.universe, low_parameter)
    internal_marks[AVERAGE] = fuzz.trapmf(internal_marks.universe, average_parameter)
    internal_marks[GOOD] = fuzz.trapmf(internal_marks.universe, good_parameter)
    internal_marks[VERY_GOOD] = fuzz.trapmf(internal_marks.universe, very_good_parameter)
    internal_marks[EXCELLENT] = fuzz.trapmf(internal_marks.universe, excellent_parameter)

    attendance[POOR] = fuzz.trapmf(attendance.universe, [0, 0, 45, 55])
    attendance[AVERAGE] = fuzz.trapmf(attendance.universe, [35, 45, 55, 65])
    attendance[GOOD] = fuzz.trapmf(attendance.universe, [45, 55, 65, 75])
    attendance[VERY_GOOD] = fuzz.trapmf(attendance.universe, [55, 65, 75, 85])
    attendance[EXCELLENT] = fuzz.trapmf(attendance.universe, [65, 75, 100, 100])

    external_marks[POOR] = fuzz.trapmf(external_marks.universe, low_parameter)
    external_marks[AVERAGE] = fuzz.trapmf(external_marks.universe, average_parameter)
    external_marks[GOOD] = fuzz.trapmf(external_marks.universe, good_parameter)
    external_marks[VERY_GOOD] = fuzz.trapmf(external_marks.universe, very_good_parameter)
    external_marks[EXCELLENT] = fuzz.trapmf(external_marks.universe, excellent_parameter)

    student_performance[POOR] = fuzz.trapmf(student_performance.universe, low_parameter)
    student_performance[AVERAGE] = fuzz.trapmf(student_performance.universe, average_parameter)
```



```
    student_performance[GOOD] = fuzz.trapmf(student_performance.universe, good_parameter)
    student_performance[VERY_GOOD] = fuzz.trapmf(student_performance.universe, very_good_parameter)
    student_performance[EXCELLENT] = fuzz.trapmf(student_performance.universe, excellent_parameter)

    # Defining the fuzzy rules
    rule1 = ctrl.Rule(attendance[POOR] & external_marks[POOR] & internal_marks[POOR], student_performance[POOR])
    rule2 = ctrl.Rule(attendance[POOR] & external_marks[AVERAGE] & internal_marks[POOR], student_performance[POOR])
    rule3 = ctrl.Rule(attendance[POOR] & external_marks[GOOD] & internal_marks[POOR], student_performance[AVERAGE])
    rule4 = ctrl.Rule(attendance[POOR] & external_marks[VERY_GOOD] & internal_marks[POOR], student_performance[AVERAGE])
    rule5 = ctrl.Rule(attendance[POOR] & external_marks[GOOD] & internal_marks[VERY_GOOD], student_performance[GOOD])
    rule6 = ctrl.Rule(attendance[POOR] & external_marks[POOR] & internal_marks[AVERAGE], student_performance[POOR])
    rule7 = ctrl.Rule(attendance[POOR] & external_marks[AVERAGE] & internal_marks[AVERAGE], student_performance[AVERAGE])
    rule8 = ctrl.Rule(attendance[POOR] & external_marks[GOOD] & internal_marks[AVERAGE], student_performance[AVERAGE])
    rule9 = ctrl.Rule((attendance[POOR] & external_marks[GOOD] & internal_marks[GOOD]), student_performance[GOOD])
    rule10 = ctrl.Rule(attendance[POOR] & external_marks[EXCELLENT] & internal_marks[GOOD], student_performance[VERY_GOOD])
    rule11 = ctrl.Rule(attendance[AVERAGE] & external_marks[AVERAGE] & internal_marks[GOOD], student_performance[AVERAGE])
    rule12 = ctrl.Rule(attendance[AVERAGE] & external_marks[GOOD] & internal_marks[GOOD], student_performance[GOOD])
    rule13 = ctrl.Rule(attendance[AVERAGE] & external_marks[VERY_GOOD] & internal_marks[GOOD], student_performance[GOOD])
    rule14 = ctrl.Rule(attendance[AVERAGE] & external_marks[VERY_GOOD] & internal_marks[VERY_GOOD], student_performance[VERY_GOOD])
    rule15 = ctrl.Rule(attendance[AVERAGE] & external_marks[AVERAGE] & internal_marks[EXCELLENT], student_performance[GOOD])
    rule16 = ctrl.Rule(attendance[AVERAGE] & external_marks[AVERAGE] & internal_marks[AVERAGE], student_performance[AVERAGE])
    rule17 = ctrl.Rule(attendance[AVERAGE] & external_marks[POOR] & internal_marks[POOR], student_performance[POOR])
    rule18 = ctrl.Rule(attendance[AVERAGE] & external_marks[POOR] & internal_marks[GOOD], student_performance[AVERAGE])
    rule19 = ctrl.Rule(attendance[GOOD] & external_marks[AVERAGE] & internal_marks[AVERAGE], student_performance[AVERAGE])
    rule20 = ctrl.Rule(attendance[GOOD] & external_marks[EXCELLENT] & internal_marks[EXCELLENT], student_performance[VERY_GOOD])
    rule21 = ctrl.Rule(attendance[GOOD] & external_marks[GOOD] & internal_marks[AVERAGE], student_performance[GOOD])
    rule22 = ctrl.Rule(attendance[GOOD] & external_marks[POOR] & internal_marks[POOR], student_performance[POOR])
    rule23 = ctrl.Rule(attendance[VERY_GOOD] & external_marks[EXCELLENT] & internal_marks[VERY_GOOD], student_performance[VERY_GOOD])
    rule24 = ctrl.Rule(attendance[VERY_GOOD] & external_marks[VERY_GOOD] & internal_marks[VERY_GOOD], student_performance[VERY_GOOD])
    rule25 = ctrl.Rule(attendance[VERY_GOOD] & external_marks[POOR] & internal_marks[POOR], student_performance[POOR])
```

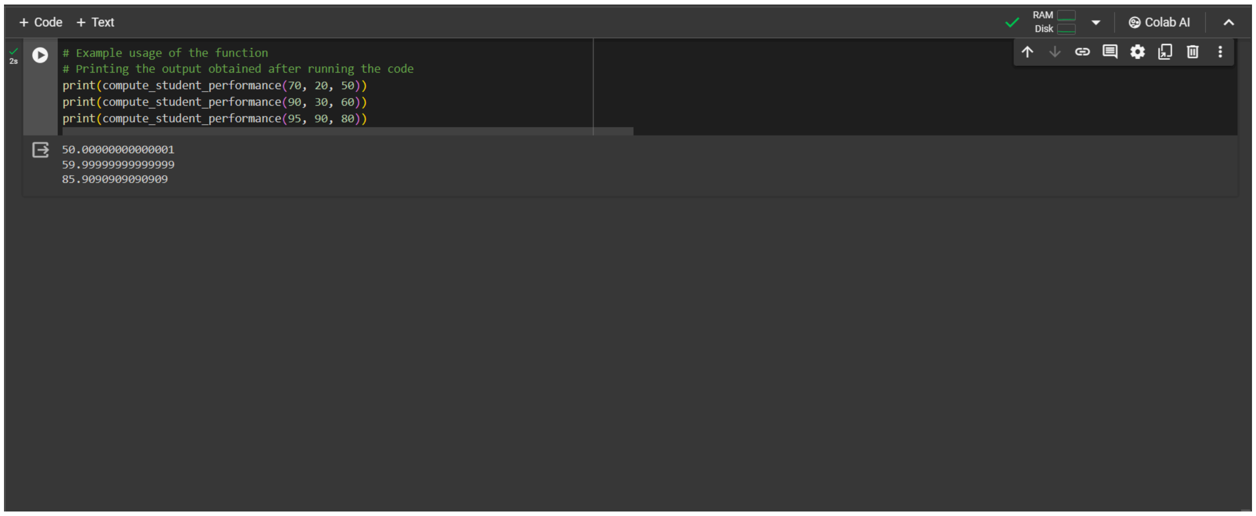


```
    rule26 = ctrl.Rule(attendance[VERY_GOOD] & external_marks[GOOD] & internal_marks[VERY_GOOD], student_performance[VERY_GOOD])
    rule27 = ctrl.Rule(attendance[VERY_GOOD] & external_marks[EXCELLENT] & internal_marks[EXCELLENT], student_performance[EXCELLENT])
    rule28 = ctrl.Rule(attendance[EXCELLENT] & external_marks[EXCELLENT] & internal_marks[VERY_GOOD], student_performance[VERY_GOOD])
    rule29 = ctrl.Rule(attendance[EXCELLENT] & external_marks[AVERAGE] & internal_marks[AVERAGE], student_performance[VERY_GOOD])
    rule30 = ctrl.Rule(attendance[EXCELLENT] & external_marks[AVERAGE] & internal_marks[VERY_GOOD], student_performance[GOOD])
    rule31 = ctrl.Rule(attendance[EXCELLENT] & external_marks[AVERAGE] & internal_marks[GOOD], student_performance[GOOD])
    rule32 = ctrl.Rule(attendance[EXCELLENT] & external_marks[POOR] & internal_marks[POOR], student_performance[POOR])
    rule33 = ctrl.Rule(attendance[EXCELLENT] & external_marks[AVERAGE] & internal_marks[POOR], student_performance[AVERAGE])
    rule34 = ctrl.Rule(attendance[EXCELLENT] & external_marks[POOR] & internal_marks[AVERAGE], student_performance[POOR])
    rule35 = ctrl.Rule(attendance[EXCELLENT] & external_marks[GOOD] & internal_marks[POOR], student_performance[GOOD])
    rule36 = ctrl.Rule(attendance[EXCELLENT] & external_marks[POOR] & internal_marks[GOOD], student_performance[AVERAGE])
    rule37 = ctrl.Rule(attendance[EXCELLENT] & external_marks[VERY_GOOD] & internal_marks[POOR], student_performance[VERY_GOOD])
    rule38 = ctrl.Rule(attendance[EXCELLENT] & external_marks[POOR] & internal_marks[VERY_GOOD], student_performance[AVERAGE])
    rule39 = ctrl.Rule(attendance[EXCELLENT] & external_marks[POOR] & internal_marks[EXCELLENT], student_performance[GOOD])
    rule40 = ctrl.Rule(attendance[EXCELLENT] & external_marks[AVERAGE] & internal_marks[EXCELLENT], student_performance[VERY_GOOD])
    rule41 = ctrl.Rule(attendance[EXCELLENT] & external_marks[GOOD] & internal_marks[EXCELLENT], student_performance[VERY_GOOD])
    rule42 = ctrl.Rule(attendance[EXCELLENT] & external_marks[VERY_GOOD] & internal_marks[EXCELLENT], student_performance[VERY_GOOD])
    rule43 = ctrl.Rule(attendance[EXCELLENT] & external_marks[EXCELLENT] & internal_marks[EXCELLENT], student_performance[EXCELLENT])

    # Creating a list of rules
    rule_list = [rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8, rule9, rule10, rule11, rule12, rule13, rule14, rule15, rule16, rule17, rule18, rule19, rule20, rule21,

    # Creating control system using the rules
    student_performance_ctrl = ctrl.ControlSystem(rule_list)

    # Creating simulation of the control system
    student_performance_analysis = ctrl.ControlSystemSimulation(student_performance_ctrl)
```



```
    # Setting the input values for the simulation
    student_performance_analysis.input[STUDENT_ATTENDANCE] = student_attendance
    student_performance_analysis.input[STUDENT_EXTERNAL_MARKS] = student_external_marks
    student_performance_analysis.input[STUDENT_INTERNAL_MARKS] = student_internal_marks

    # Running the simulation
    student_performance_analysis.compute()

    # Returning the output of the simulation as a string
    return str(student_performance_analysis.output[STUDENT_PERFORMANCE])

# Example usage of the function
# Printing the output obtained after running the code
print(compute_student_performance(70, 20, 50))
print(compute_student_performance(90, 30, 60))
print(compute_student_performance(95, 90, 80))
```

Output Screenshot:



```python
# Example usage of the function
# Printing the output obtained after running the code
print(compute_student_performance(70, 20, 50))
print(compute_student_performance(90, 30, 60))
print(compute_student_performance(95, 90, 80))
```

```
50.00000000000001
59.99999999999999
85.9090909090909
```

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

The Fuzzy Logic Based Student Academic Performance Analyzing System has been successfully implemented and tested. The system has demonstrated its effectiveness in providing a nuanced and flexible assessment of students' academic performance. By incorporating various input variables and employing fuzzy logic rules, the system has been able to derive conclusions about students' performance that are more comprehensive and accurate than traditional grading systems.

The system has also proven to be a valuable tool for educators and administrators, providing personalized feedback and identifying students who may need additional support or intervention. This has enhanced decision-making processes and contributed to improved learning outcomes.

Future Enhancements:

Looking forward, there are several potential enhancements that could further improve the functionality and effectiveness of the Fuzzy Logic Based Student Academic Performance Analyzing System:

Incorporation of Additional Input Variables: The system could incorporate additional input variables such as students' self-assessment scores, peer assessment scores, or feedback from educators. This would provide a more holistic view of students' performance.
Adaptation to Different Educational Contexts: The system could be adapted to different educational contexts, such as primary, secondary, or tertiary education. This would require adjusting the linguistic variables and fuzzy rules to suit the specific context.
Integration with Learning Management Systems: The system could be integrated with existing Learning Management Systems (LMS) to automatically pull student data and provide real-time performance analysis.

Development of a User-Friendly Interface: A user-friendly interface could be developed to allow educators and administrators to easily input data and interpret the results generated by the system.
Expansion of Fuzzy Rules: The fuzzy rules could be expanded to cover more scenarios and provide more detailed feedback to students.

By implementing these enhancements, the Fuzzy Logic Based Student Academic Performance Analyzing System could become an even more powerful tool for assessing student performance and supporting decision-making in educational institutions. It holds great promise for the future of education technology.

References

1. www.python.org
2. www.wikipedia.org
3. Fuzzy Logic | Introduction - GeeksforGeeks
4. Fuzzy Rules - an overview | ScienceDirect Topics
5. Scikit-Fuzzy Python Library Docs