ABSTRACT

The Student Performance Analyzer is a Python-based web application designed to address the challenges of manual grading and lack of visual insights in traditional report cards. Built using Streamlit, Pandas, and Matplotlib, this application automates the calculation of student grades, percentages, and performance metrics. It provides an interactive interface for entering student marks and generates instant, professional report cards accompanied by graphical performance analysis. This project demonstrates the practical application of Python in automating educational processes and decision-making.

DECLARATION

We the undersigned students of 1st semester, Department of Artificial Intelligence & Data Science, BMS College of Engineering, declare that the Introduction to Python Programming mini project work entitled "Student Performance Analyzer", is a bonafide work of us and our project is neither a copy nor by any means a modification of any other engineering project.

We also declare that this project was not entitled for submission to any other university in the past and shall remain the only submission made and will not be submitted by us to any other university in the future.

Name	**USN**	**Signature**
Lochan Gowda T M	1BM25AD056-T	_____
Maanas Bhandari	1BM25AD047-T	_____
Tanish Loya	1BM25AD048-T	_____

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our effort with success.

We are grateful to our institution, BMS College of Engineering, with its ideals and inspirations for having provided us with the facilities, which has made this project a success.

We earnestly thank Dr. Bheemsha Arya, Principal, BMS College of Engineering, for facilitating academic excellence in the college and providing us with the congenial environment to work in, that helped us in completing this project.

We wish to extend our profound thanks to Dr. Indiramma M, Prof. and Head, Department of Artificial Intelligence & Data Science, BMSCE, for giving us the consent to carry out this project.

We would like to express our sincere thanks to our faculty in charge Prof. Pramoda R, Assistant Professor, Department of Artificial Intelligence & Data Science, BMSCE, for his valuable guidance at every stage, which helped us in the successful completion of the project.

We would like to thank all the teaching and non-teaching staff for their valuable advice and support.

We would like to express our sincere thanks to our parents and friends for their support.

Lochan Gowda T M
Maanas Bhandari
Tanish Loya

TABLE OF CONTENTS

Prefatory Pages:

ABSTRACT i

DECLARATION ii

ACKNOWLEDGEMENT iii

LIST OF TABLES iv

LIST OF FIGURES v

Chapters:

1. Introduction 1

2. Literature Survey 2

3. Requirement Analysis and Specification 3

4. Design 4

5. Implementation 6

6. Testing and Validation 8

7. Results and Discussion 9

8. Conclusion 11

LIST OF TABLES

Table No	Description	Page No.
Table 6.1	Test Cases and Results for Grading System	8

LIST OF FIGURES

Figure No	Description	Page No.
Figure 4.1	System Architecture Diagram	6
Figure 7.1	Application Input Sidebar	9
Figure 7.2	Generated Report Output	10
Figure 7.3	Performance Visualization Chart	10

CHAPTER 1: INTRODUCTION

1.1 Overview

The Student Performance Analyzer is a web-based application developed to assist educational institutions and students in automating the grading process. By leveraging Python's computational power and Streamlit's interactive capabilities, the project transforms raw marks data into meaningful insights.

1.2 Problem Statement

In traditional manual systems, calculating totals, percentages, and assigning grades for multiple students is time-consuming and prone to human error. Additionally, static paper reports fail to provide a visual comparative analysis of a student's performance across different subjects.

1.3 Project Objectives

- **Automation:** To replace manual arithmetic calculations with instant, error-free automated processing.
- **Visualization:** To provide graphical representations (bar charts) that help identifying academic strengths and weaknesses instantly.
- **Accessibility:** To create a user-friendly interface that requires no technical training to use.

CHAPTER 2: LITERATURE SURVEY

2.1 Existing Systems

Traditional methods involve:

1. **Manual Calculation:** Teachers use calculators to sum marks and determine percentages.
2. **Excel Spreadsheets:** While better than manual calculation, they often require complex formula setups and lack interactive visualizations for individual students.

2.2 Proposed System

Our proposed Student Performance Analyzer improves upon these by:

- Providing a **dedicated GUI** specifically for grading.
- **Instantly generating charts** without manual configuration.
- **Assigning grades automatically** based on pre-defined logic, eliminating lookup errors.

CHAPTER 3: REQUIREMENT ANALYSIS AND SPECIFICATION

3.1 Hardware Requirements

To run this application efficiently, the following hardware specifications are recommended:

- **Processor:** Intel Core i3 (5th Gen) or higher / AMD Ryzen 3 or equivalent.
- **RAM:** Minimum 4GB (8GB recommended for smoother performance).
- **Storage:** At least 500MB of free disk space.
- **Display:** 1366x768 minimum resolution.

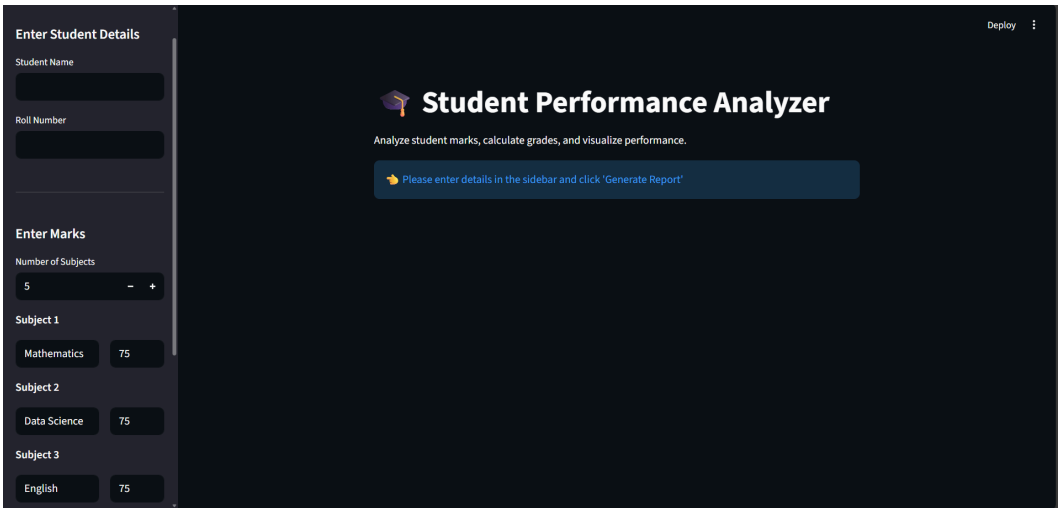
3.2 Software Requirements

- **Operating System:** Windows 10/11, macOS, or Linux distributions (Ubuntu/Fedora).
- **Programming Language:** Python 3.7 or later versions.
- **Development Environment:** Visual Studio Code (VS Code) or PyCharm.
- **Key Libraries:**
    - Streamlit: For the web application framework.
    - Pandas: For data manipulation and analysis.
    - Matplotlib: For data visualization and plotting.

CHAPTER 4: DESIGN

4.1 System Architecture

The system follows a modular architecture where inputs flow through a processing layer before being rendered as outputs.



- **Input Layer:** Collects Student Name, Roll No, and Subject Marks via the Sidebar.
- **Logic Layer:** Performs summation, percentage calculation, and grade logic application.
- **Presentation Layer:** Displays Metrics, Tables, and Charts to the user.

4.2 Data Flow

1. **User Input:** User enters marks for N subjects.
2. **Processing:**
    - Total = Sum(Marks)
    - Percentage = (Total / Max_Marks) * 100
    - Grade = Function(Percentage)
3. **Visualization:** Matplotlib receives subject names and marks to plot bars.
4. **Output:** Final Report Card is rendered on the screen.

CHAPTER 5: IMPLEMENTATION

5.1 Technology Stack

The project is implemented using Python, chosen for its simplicity and powerful libraries.

5.1.1 Streamlit (Frontend)

Streamlit is used to build the web interface. Functions like st.sidebar.text_input and st.metric allow for rapid UI development without HTML/CSS.

5.1.2 Pandas (Data Handling)

Pandas is used to create a DataFrame from the input marks. This allows us to easily display the data in a neat tabular format using st.dataframe.

5.1.3 Matplotlib (Visualization)

We use matplotlib.pyplot to generate the bar chart. It takes the subject names as the X-axis and marks as the Y-axis, applying distinct colors to each bar for better readability.

5.2 Core Logic Snippet

The grading logic is implemented using standard conditional statements:

```
if percentage >= 90:
    grade = "O (Outstanding)"
elif percentage >= 80:
    grade = "A+ (Excellent)"
elif percentage >= 70:
    grade = "A (Very Good)"
elif percentage >= 60:
    grade = "B (Good)"
elif percentage >= 50:
    grade = "C (Pass)"
else:
    grade = "F (Fail)"
```

CHAPTER 6: TESTING AND VALIDATION

6.1 Testing Strategy

We performed manual testing by entering various combinations of marks to ensure the system handles all edge cases correctly.

6.2 Test Cases

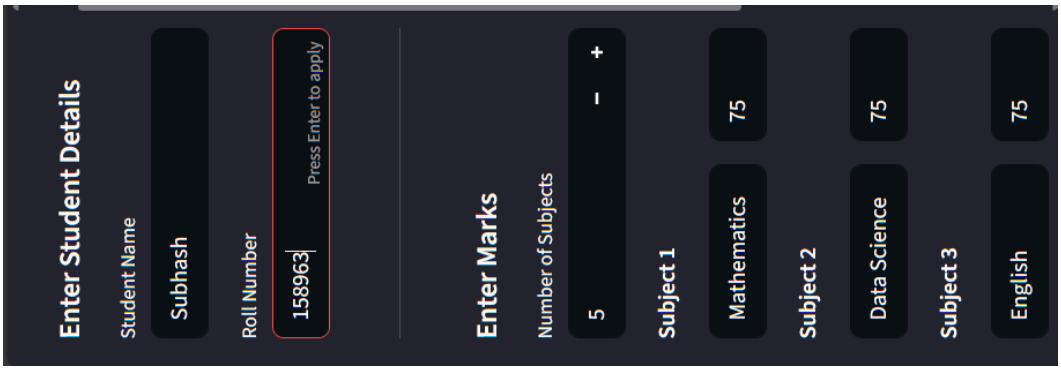
Table 6.1: Test Cases and Results for Grading System

Test Case ID	Scenario	Input Data	Expected Output	Actual Output	Status
TC-01	Max Possible Score	100 in all subjects	Grade: O, 100%	Grade: O, 100%	PASS
TC-02	Minimum Passing	50 in all subjects	Grade: C, 50%	Grade: C, 50%	PASS
TC-03	Fail Scenario	30 in all subjects	Grade: F, 30%	Grade: F, 30%	PASS
TC-04	Mixed Performance	90, 80, 70, 60, 50	Grade: A, 70%	Grade: A, 70%	PASS

CHAPTER 7: RESULTS AND DISCUSSION

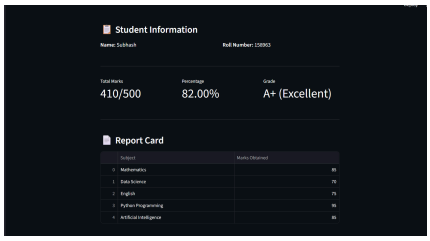
7.1 Input Interface

The application provides a clean sidebar for inputs. Users can dynamically select the number of subjects (1-10), and the form adjusts automatically.



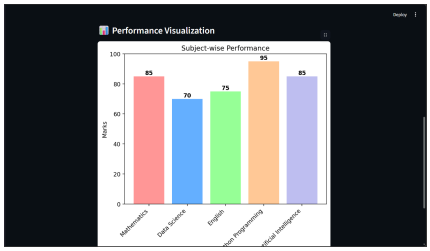
7.2 Report Generation

Upon clicking "Generate Report", the system displays the student's details, total marks, calculated percentage, and the final grade in a prominent metric style.



7.3 Visualization Output

A color-coded bar chart is generated to visually represent performance. This allows for instant comparison between subjects.



CHAPTER 8: CONCLUSION

8.1 Conclusion

The Student Performance Analyzer project successfully demonstrates how Python can be used to automate and simplify educational tasks. By creating a tool that combines calculation, reporting, and visualization, we have addressed the limitations of manual grading. The project meets all defined objectives and provides a robust foundation for further development.

8.2 Future Enhancements

- **Database Connectivity:** Integrating SQLite or MySQL to store student records permanently.
- **PDF Export:** Adding a feature to download the generated report as a PDF file.
- **Multi-User Support:** Implementing login functionality for different teachers.
- **Batch Processing:** Allow uploading CSV files with multiple students' marks.
- **Performance Analytics:** Generate class-level analytics and trends.

END OF REPORT