Student Result Management System

SUBMITTED BY:-

Name:-Tanishq Gupta

Roll Number:-25BAI11026

INTRODUCTION

Background Student assessment and result declaration are important parts of any educational institution. In many school With the increasing number of students and subjects, managing results manually becomes difficult.

Search in Need for Automation An automated Student Result Management System helps to: -

Store student details and marks in a structured way.

Automatically calculate total marks and percentage.

Assign grades and pass/fail status using predefined rules.

Display individual as well as class-wise results quickly.

Reduce manual workload of teachers and minimize calculation errors. Because of these advantages, it is useful to design a simple computer-based system for result management.

Aim of the Project The aim of this project is to develop a simple Student Result Management System using Python. The system

PROBLEM STATEMENT

1.Existing System In the existing manual system, student marks are usually written in notebooks, registers or maintain any mistake is found later, it is difficult to correct and update all related records.

2. Limitations of Existing System The major limitations of the manual or semi-manual system are: - High chances of calculation mistakes. - Time-consuming process for entering and updating marks. - Difficult to search and retrieve old records quickly. - No standard way to generate reports, such as toppers list or list of failed students. - Physical records can be damaged or lost.

3. The proposed Student Result Management System is a Python-based application that will: - Store student details such as roll number, name and class. - Accept marks for a fixed set of subjects (for example: Maths, Science, English, Computer and Hindi). - Validate the marks entered by the user. - Automatically calculate total marks, percentage and grade. - Decide whether the student has passed or failed based on predefined criteria. - Display individual student results and also show a list of all students with their results. This system reduces manual work, provides faster result generation and ensures more accurate calculation

FUNCTIONAL REQUIREMENTS

1. ADD NEW STUDENT THE SYSTEM SHOULD ALLOW THE USER TO ADD A NEW STUDENT BY ENTERING DETAILS LIKE ROLL NUMBER, NAME AND CLASS SECTION.

2. ENTER OR UPDATE MARKS THE SYSTEM SHOULD ALLOW THE USER TO ENTER OR UPDATE MARKS FOR EACH STUDENT. MARKS WILL BE ENTERED

3. CALCULATE RESULT AFTER MARKS ARE ENTERED, THE SYSTEM SHOULD CALCULATE THE TOTAL MARKS AND PERCENTAGE OF THE STUDENT

4. VIEW INDIVIDUAL STUDENT RESULT THE SYSTEM SHOULD ALLOW THE USER TO SEARCH FOR A STUDENT USING THE ROLL NUMBER AND VIEW THAT STUDENT

5. VIEW ALL STUDENTS' RESULTS THE SYSTEM SHOULD DISPLAY A LIST OF ALL STUDENTS WITH IMPORTANT INFORMATION LIKE ROLL NUMBER, NAME.

6. DELETE STUDENT RECORD (OPTIONAL) THE REPORT MAY DESCRIBE AN OPTIONAL FEATURE TO DELETE A STUDENT RECORD, BUT IN THE CURRENT IMPLEMEN

NON-FUNCTIONAL REQUIREMENTS

NON-FUNCTIONAL REQUIREMENTS DESCRIBE HOW THE SYSTEM SHOULD BEHAVE.

1. USABILITY THE SYSTEM SHOULD HAVE A SIMPLE AND CLEAR MENU-DRIVEN INTERFACE. MESSAGES AND PROMPTS SHOULD BE EASY TO UNDERSTAND.

2. RELIABILITY THE SYSTEM SHOULD PERFORM CALCULATIONS CORRECTLY EVERY TIME. THE RESULT FOR A GIVEN SET OF MARKS MUST ALWAYS BE INTEGER TYPE

3. PERFORMANCE THE SYSTEM SHOULD RESPOND QUICKLY TO USER COMMANDS. ADDING STUDENTS, ENTERING MARKS AND GENERATING RESULT

4. MAINTAINABILITY THE CODE OF THE SYSTEM SHOULD BE MODULAR AND WELL-ORGANIZED. DIFFERENT TASKS SUCH AS ADDING STUDENTS, ENTERING ROLL NO.

5. DATA INTEGRITY AND VALIDATION THE SYSTEM SHOULD VALIDATE THE DATA ENTERED BY THE USER. FOR EXAMPLE, MARKS SHOULD NOT BE NEGATIVE AND SHOULD BE INTEGER TYPE

SYSTEM ARCHITECTURE

1.ARCHITECTURE DESCRIPTION THE STUDENT RESULT MANAGEMENT SYSTEM FOLLOWS A SIMPLE LAYERED STRUCTURE:

- **INPUT LAYER:** THE USER INTERACTS WITH THE SYSTEM USING A CONSOLE-BASED MENU. THE USER ENTERS STUDENT DETA
- **PROCESSING LAYER:** THE PYTHON PROGRAM PROCESSES THE INPUT, VALIDATES DATA, PERFORMS CALCULATIONS.
- **DATA LAYER:** STUDENT RECORDS AND MARKS ARE STORED IN SUITABLE DATA STRUCTURES SUCH AS LISTS AND DICTIONARIESTHE USER SENDS INPUT TO THE APPLICATION. THE APPLICATION THEN USES ITS MODULES TO PROCESS THE DATA AND FINAL RESULT.

2 ARCHITECTURE DIAGRAM (DESCRIPTION) THE ARCHITECTURE CAN BE REPRESENTED BY THREE MAIN BLOCKS: USER → APPLICATION (STUDENT RESULT MANAGEMENT SYSTEM) → DATA STORAGE INSIDE THE APPLICATION BLOCK, WE CAN IMAGINE SMALLER COMPONENTS SUCH AS STUDENT MODULE, MARKS MODULE

DESIGN DIAGRAMS (DESCRIPTIONS)

1 USE CASE DIAGRAM (DESCRIPTION) ACTOR: TEACHER OR ADMIN

MAIN USE CASES:

- ADD STUDENT
- ENTER MARKS
- UPDATE MARKS
- VIEW STUDENT RESULT
- VIEW ALL RESULTS
- DELETE STUDENT (OPTIONAL, FUTURE ENHANCEMENT) THE TEACHER INTERACTS WITH EACH OF THESE USE CASES.

IMPLEMENTATION DETAILS

1. Programming Language and Tools The project is implemented in Python. Any standard Python environment can be used, such as IDLE, VS Code

2.Module and Function Structure The system is divided into several functions, for example:

- main_menu(): displays the main menu and handles user choices.
- add_student(): reads student details from the user and stores them.
- input_marks(): reads subject marks for a student and validates them. - calculate_result(): calculates total marks, percentage, grade and pass/fail status for a student.
- view_student_result(): shows the result of a single student.
- view_all_students(): prints a summary of all students with their results.

3. Algorithm for Result Calculation The algorithm for calculating the result is as follows:

1. Read marks of all subjects.
2. Calculate the total by summing all subject marks.
3. Calculate the percentage = total / number_of_subjects.
4. Based on the percentage, assign a grade:

CHALLENGES FACED

DURING THE DEVELOPMENT OF THE STUDENT RESULT MANAGEMENT SYSTEM, SEVERAL CHALLENGES WERE FACED:

- UNDERSTANDING HOW TO STORE AND MANAGE MULTIPLE STUDENT RECORDS USING SUITABLE DATA STRUCTURES.
- DESIGNING A USER-FRIENDLY MENU THAT ALLOWS THE USER TO PERFORM ALL OPERATIONS EASILY.
- IMPLEME NTING INPUT VALIDATION TO HANDLE INCORRECT OR UNEXPECTED INPUTS SUCH AS TEXT INSTEAD OF NUMBERS.
- DEBUGGING ERRORS THAT OCCURRED DURING DEVELOPMENT AND TESTING. THESE CHALLENGES WERE RESOLVED BY BREAKING THE PROBLEM INTO SMALLER PARTS, TESTING EACH PART SEPARATELY

LEARNINGS AND CONCLUSION

1 Learnings

Through this project, the following concepts were learned and practiced:

- Writing structured Python programs using functions and modules.
- Using lists and dictionaries to store and manage data.
- Designing simple software systems using requirements, architecture and design diagrams.
- Applying basic testing techniques to verify the correctness of a program.
- Understanding the importance of planning and step-by-step development.

FUTURE ENHANCEMENTS

The project can be improved in several ways in the future:

- Adding a login system for administrators and students.
- Developing a graphical user interface using libraries like Tkinter or creating a web-based front end.
- Storing data in a proper database system such as MySQL or PostgreSQL.
- Providing options to export results to PDF or Excel files.
- Sending result notifications to students through email or SMS.
- Integrating the result system with other modules such as attendance or fee management.

REFERENCES

The following resources were used while developing and documenting this project:

- Python official documentation.
- Class notes and materials provided by the faculty.
- Standard textbooks on Python programming and software engineering.
- Various online tutorials and articles on Python and basic software design.

The End