SIMPLE GRADE MANAGER

DSA PROJECT

PRATIK BANSAL
B.TECH CSE (CORE)
REG-NO. 10323210293
YEAR: 2023-27

Introduction

A Student Grade Checking System Made Using Python And MySQL

- A Python application integrated with MySQL for managing student grades.
- Allows users to add, modify, delete, show, and search student records.

Features

- **Add Student**
  - Input student details (Roll No, Name, Gender, Marks for 5 Subjects).
  - Calculate total marks, percentage, and grade.
- **Modify Student Data**
  - Update marks for existing students.
  - Recalculate total marks, percentage, and grade.
- **Delete Student**
  - Remove student records based on Roll No.
- **Show All Students**
  - Display all student records in the database.
- **Search for Student**
  - Search and display details for a specific Roll No.

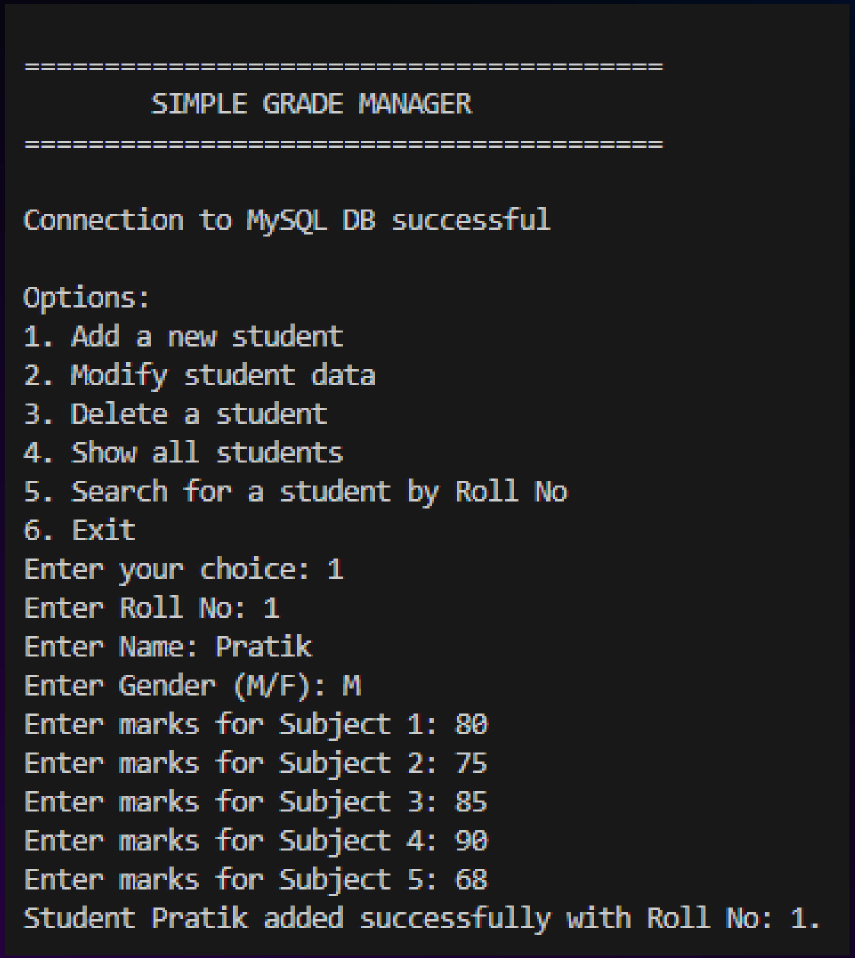
Functionality - Adding Students

- **Process:**
  - User inputs Roll No, Name, Gender, and marks for five subjects.
  - System calculates total marks, percentage, and grade.
  - Record is added to the database.
- **Code Snippet:**

```python
# Function to add a new student
def add_student(connection, roll_no, name, gender, sub1, sub2, sub3, sub4, sub5):
    total_marks, percentage, grade = calculate_results(sub1, sub2, sub3, sub4, sub5)

    query = """
    INSERT INTO STUDENTS (Roll_No, Name, Gender, Sub1, Sub2, Sub3, Sub4, Sub5, Total_Marks, Percentage, Grade)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
    """
    values = (roll_no, name, gender, sub1, sub2, sub3, sub4, sub5, total_marks, percentage, grade)
    execute_query(connection, query, values)
    print(f"Student {name} added successfully with Roll No: {roll_no}.")
```

Functionality - Adding Students

- Screenshot:



```
========================================
        SIMPLE GRADE MANAGER
========================================

Connection to MySQL DB successful

Options:
1. Add a new student
2. Modify student data
3. Delete a student
4. Show all students
5. Search for a student by Roll No
6. Exit
Enter your choice: 1
Enter Roll No: 1
Enter Name: Pratik
Enter Gender (M/F): M
Enter marks for Subject 1: 80
Enter marks for Subject 2: 75
Enter marks for Subject 3: 85
Enter marks for Subject 4: 90
Enter marks for Subject 5: 68
Student Pratik added successfully with Roll No: 1.
```

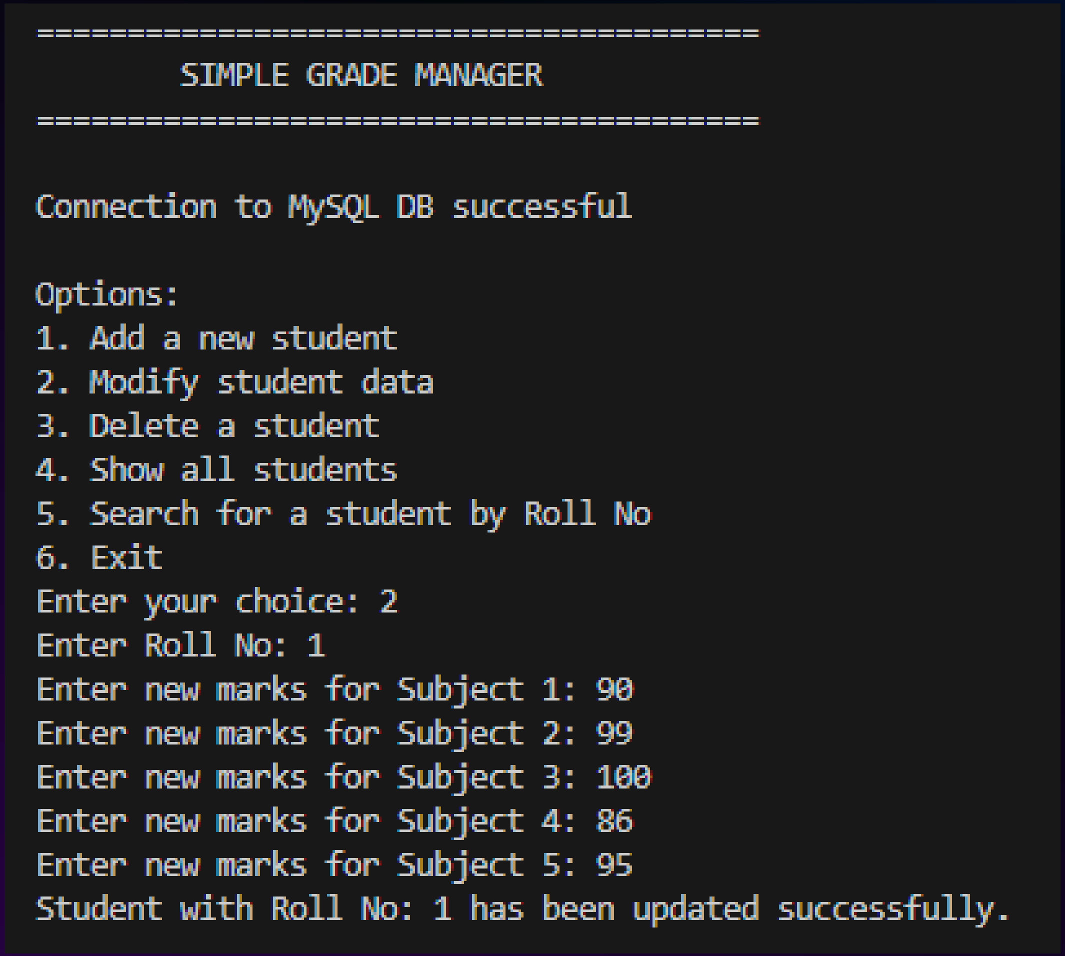
Functionality - Modifying Students

- **Process:**
  - User provides Roll No and new marks for subjects.
  - System updates the record with new marks and recalculates total, percentage, and grade.
- **Code Snippet:**

```python
# Function to modify a student's data
def modify_student(connection, roll_no, sub1, sub2, sub3, sub4, sub5):
    total_marks, percentage, grade = calculate_results(sub1, sub2, sub3, sub4, sub5)

    query = """
    UPDATE STUDENTS
    SET Sub1 = %s, Sub2 = %s, Sub3 = %s, Sub4 = %s, Sub5 = %s, Total_Marks = %s, Percentage = %s, Grade = %s
    WHERE Roll_No = %s
    """
    values = (sub1, sub2, sub3, sub4, sub5, total_marks, percentage, grade, roll_no)
    execute_query(connection, query, values)
    print(f"Student with Roll No: {roll_no} has been updated successfully.")
```

Functionality - Modifying Students

- **Screenshot:**



```
========================================
        SIMPLE GRADE MANAGER
========================================

Connection to MySQL DB successful

Options:
1. Add a new student
2. Modify student data
3. Delete a student
4. Show all students
5. Search for a student by Roll No
6. Exit
Enter your choice: 2
Enter Roll No: 1
Enter new marks for Subject 1: 90
Enter new marks for Subject 2: 99
Enter new marks for Subject 3: 100
Enter new marks for Subject 4: 86
Enter new marks for Subject 5: 95
Student with Roll No: 1 has been updated successfully.
```

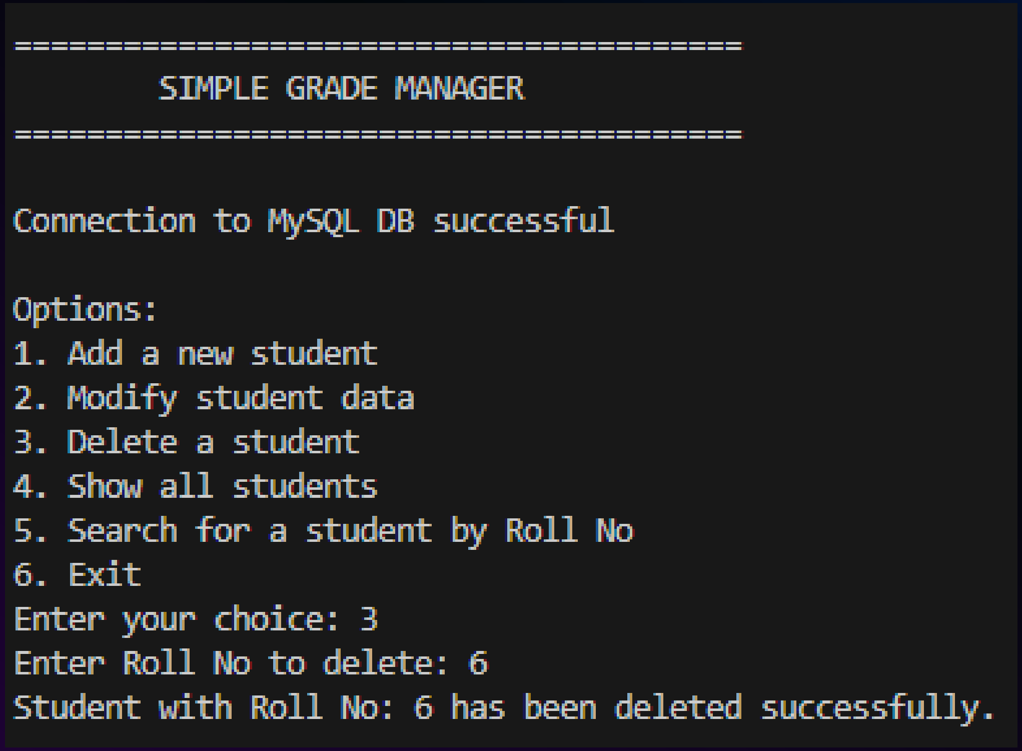
Functionality - Deleting Students

- **Process:**
  - User inputs Roll No of the student to be deleted.
  - Record is removed from the database.
- **Code Snippet:**

```python
# Function to delete a student record
def delete_student(connection, roll_no):
    query = "DELETE FROM STUDENTS WHERE Roll_No = %s"
    values = (roll_no,)
    execute_query(connection, query, values)
    print(f"Student with Roll No: {roll_no} has been deleted successfully.")
```

Functionality - Deleting Students

- Screenshot:



```
========================================
        SIMPLE GRADE MANAGER
========================================

Connection to MySQL DB successful

Options:
1. Add a new student
2. Modify student data
3. Delete a student
4. Show all students
5. Search for a student by Roll No
6. Exit
Enter your choice: 3
Enter Roll No to delete: 6
Student with Roll No: 6 has been deleted successfully.
```

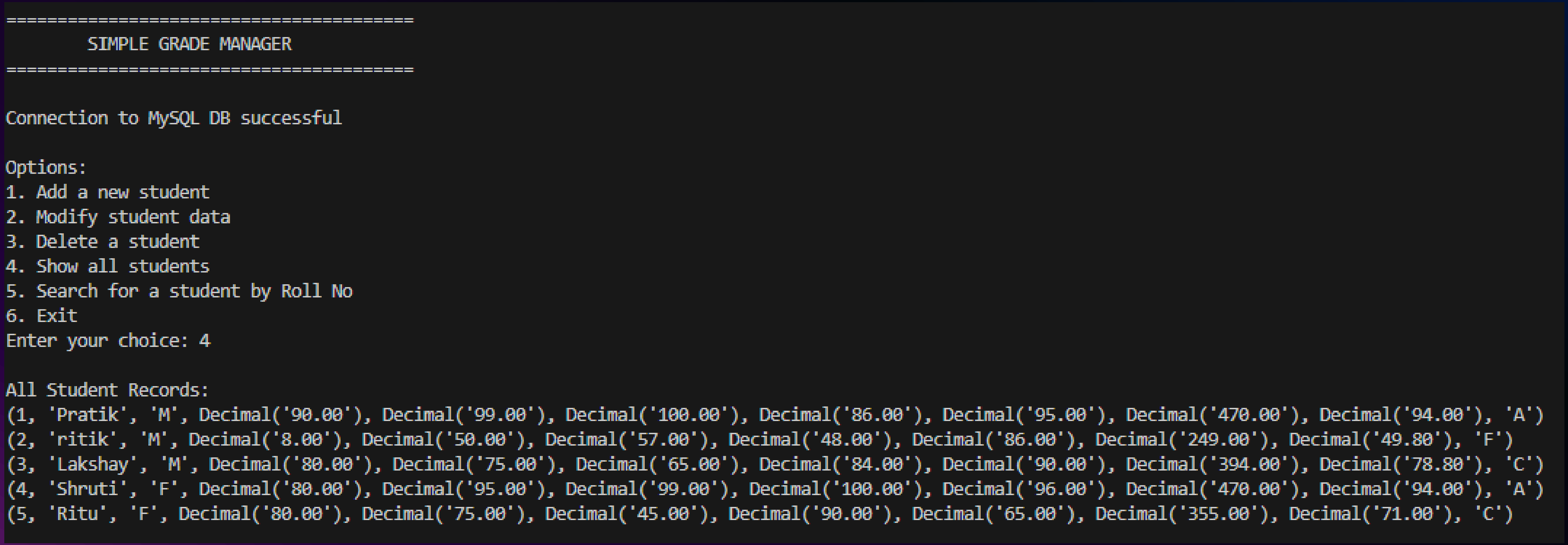
Functionality - Showing All Students

- **Process:**
  - Retrieve and display all student records.
- **Code Snippet:**

```python
# Function to show all students
def show_students(connection):
    query = "SELECT * FROM STUDENTS"
    results = fetch_query(connection, query)

    if results:
        print("\nAll Student Records:")
        for record in results:
            print(record)
    else:
        print("No records found.")
```

Functionality - Showing All Students

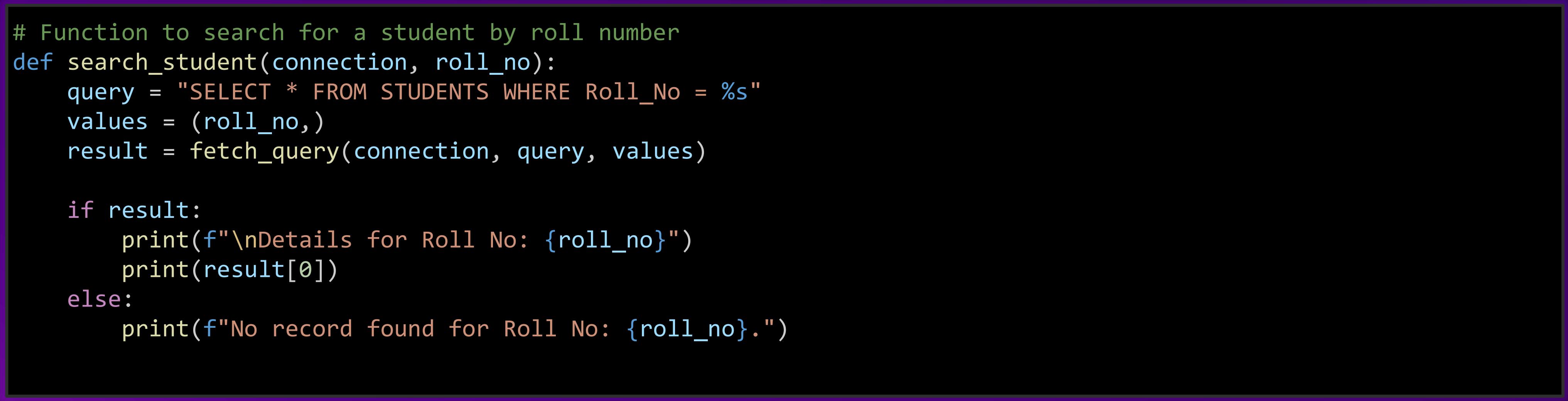
- Screenshot:



```
========================================
        SIMPLE GRADE MANAGER
========================================

Connection to MySQL DB successful

Options:
1. Add a new student
2. Modify student data
3. Delete a student
4. Show all students
5. Search for a student by Roll No
6. Exit
Enter your choice: 4

All Student Records:
(1, 'Pratik', 'M', Decimal('90.00'), Decimal('99.00'), Decimal('100.00'), Decimal('86.00'), Decimal('95.00'), Decimal('470.00'), Decimal('94.00'), 'A')
(2, 'ritik', 'M', Decimal('8.00'), Decimal('50.00'), Decimal('57.00'), Decimal('48.00'), Decimal('86.00'), Decimal('249.00'), Decimal('49.80'), 'F')
(3, 'Lakshay', 'M', Decimal('80.00'), Decimal('75.00'), Decimal('65.00'), Decimal('84.00'), Decimal('90.00'), Decimal('394.00'), Decimal('78.80'), 'C')
(4, 'Shruti', 'F', Decimal('80.00'), Decimal('95.00'), Decimal('99.00'), Decimal('100.00'), Decimal('96.00'), Decimal('470.00'), Decimal('94.00'), 'A')
(5, 'Ritu', 'F', Decimal('80.00'), Decimal('75.00'), Decimal('45.00'), Decimal('90.00'), Decimal('65.00'), Decimal('355.00'), Decimal('71.00'), 'C')
```

Functionality - Searching for Students

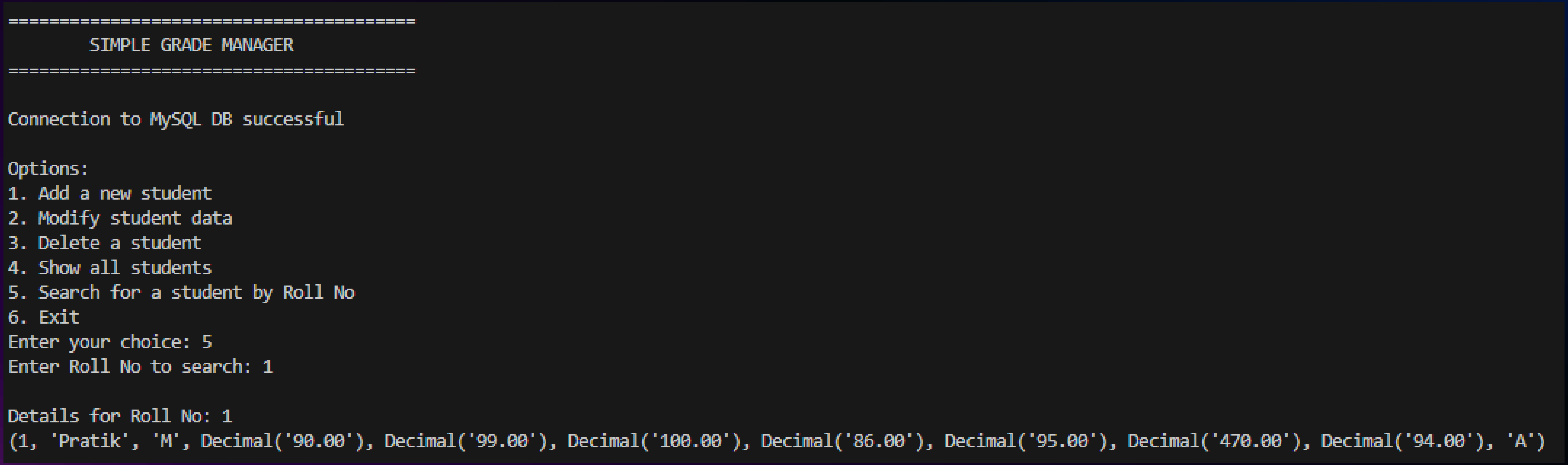
- **Process:**
  - User inputs Roll No to search.
  - Display details of the student with the given Roll No.
- **Code Snippet:**



```python
# Function to search for a student by roll number
def search_student(connection, roll_no):
    query = "SELECT * FROM STUDENTS WHERE Roll_No = %s"
    values = (roll_no,)
    result = fetch_query(connection, query, values)

    if result:
        print(f"\nDetails for Roll No: {roll_no}")
        print(result[0])
    else:
        print(f"No record found for Roll No: {roll_no}.")
```

Functionality - Searching for Students

- Screenshot:



```
========================================
        SIMPLE GRADE MANAGER
========================================

Connection to MySQL DB successful

Options:
1. Add a new student
2. Modify student data
3. Delete a student
4. Show all students
5. Search for a student by Roll No
6. Exit
Enter your choice: 5
Enter Roll No to search: 1

Details for Roll No: 1
(1, 'Pratik', 'M', Decimal('90.00'), Decimal('99.00'), Decimal('100.00'), Decimal('86.00'), Decimal('95.00'), Decimal('470.00'), Decimal('94.00'), 'A')
```

Advantages

- **Easy to Use:**
  - Simple interface for managing student data.
- **Real-Time Calculations:**
  - Automatic calculation of total marks, percentage, and grades.
- **Integration with MySQL:**
  - Reliable storage and management of student data.
- **Flexible Operations:**
  - Add, modify, delete, show, and search functionalities.

Disadvantages

- **Limited Functionality:**
  - Basic operations without advanced features like reporting or analytics.
- **Database Dependency:**
  - Requires MySQL setup and configuration.
- **No User Authentication:**
  - Does not include user management or security features.

Time Complexity

- **Database Operations:**
  - Generally involve querying or modifying the database, which can range from $O(1)$O(1) to $O(n)$O(n) depending on the specifics of the operation and database indexing.
- **In-Memory Operations:**
  - Functions like calculate_results are $O(1)$O(1) as they perform a fixed number of operations regardless of the input size.

Overall, the efficiency of the project largely depends on database indexing, query optimization, and the number of records managed.

Space Complexity

- **Database Operations:**
  - Generally involve space complexities of $O(1)$O(1) for individual operations (insert, update, delete) but $O(n)$O(n) for operations involving multiple records (fetch, show).
- **In-Memory Operations:**
  - Functions like calculate_results use constant space $O(1)$O(1).

The space complexity in your project primarily depends on how much data you are handling and the operations being performed. For basic CRUD operations with a manageable dataset, the space complexity is mostly $O(1)$O(1) or $O(n)$O(n) in terms of the number of records being processed or retrieved.

Conclusion

Summary:

- A straightforward tool for managing students grade.
- Useful for education institutes or individuals needing a simple grade management system.

Future Enhancements:

- Adding User Authorization.
- Generating Detailed Report.
- Implementing advanced grading criteria.

Bibliography

Books:

- Computer Science With Python – By Preeti Arora
- Basics of Python programming – By Dr. Neetu Goel, Dr. Sachin Gupta and Dr. Pooja Thakar

Websites:

- W3school
- Google

Thanks

PRATIK BANSAL
B.TECH CSE (CORE)
REG-NO. 10323210293
YEAR: 2023-27