TITLE:ONLINE EXAMINATION SYSTEM

Name: SPOORTHI V (PES2UG23CS594)

Name: SRIVIDHYA M (PES2UG23CS602)

Table of contents

2.PROBLEM STATEMENT:

The goal of this project is to design and develop an Online Examination System that provides an efficient, secure, and user-friendly platform for conducting examinations through the web. The traditional process of conducting exams involves manual tasks like question paper setting, distribution, supervision, evaluation, and result generation — all of which are time-consuming and error-prone.

This project aims to digitize and automate the entire examination process, allowing students to register, take exams, and view their results online. It enables administrators to manage exams, subjects, questions, and exam centers effectively, ensuring transparency, security, and ease of use.

3.USER REQUIREMENT SPECIFICATION:

1. Purpose of the Project:

The purpose of the Online Examination System is to provide an efficient, secure, and user-friendly platform for conducting examinations through the web. Traditional examinations involve manual processes such as paper setting, distribution, supervision, evaluation, and result generation, which are time-consuming and prone to human errors. This project aims to digitize and automate these tasks, enabling students to register, attend exams, and receive results seamlessly. The system ensures transparency in evaluation, reduces administrative overhead, and allows administrators to manage exams, subjects, and questions effectively. It also benefits students by providing instant access to results, grades, and certificates, thereby making the examination process more reliable and convenient.

2. Scope of the Project:

The scope of the Online Examination System includes the complete management of examinations starting from student registration to result generation. The system will support multiple users including students, administrators, and exam centers. Students will be able to register, attend exams online, and view their results. Administrators will have the ability to create and manage exams, subjects, questions, and monitor exam activities. The system will also maintain exam center details where examinations are held, and it will generate certificates and grades for students. This system can be extended to support multiple educational institutions, different subjects, and various exam types. It will ensure secure login for all users, accurate data storage using a relational database, and efficient result computation.

3. Detailed Description:

The Online Examination System is a web-based application designed to conduct and manage examinations. The major entities involved in the system are:

• Student: Registers in the system with details such as Student_ID, Name, Qualification, and Address. Each student may have multiple contact numbers (stored in Student_Contact). A student can register for and attend multiple exams.

• Admin: Manages the overall examination process. Admins are identified by Admin_ID and have details such as Admin_name and Admin_roll. They are responsible for creating exams, managing subjects, adding questions, and assigning exams to websites and exam centers.

• Website: The examination is conducted via a website, identified by Website_ID. Each website has a name contact details, and is linked to an administrator.

• Examination: Defined by Exam_ID, it includes Exam_name , Exam_subject, number of questions, and Website_ID. An exam may be conducted at different exam centers.

• Exam Center: A physical or virtual location where the exam is held. It is identified by Center_no and Center_name, and linked to a specific exam.

• Subject: Contains Subject_code and Subject_name. Each subject is linked with multiple questions and may be part of multiple examinations.

• Questions: Each question has a Question_ID, text of the Question, and the Correct_answer. Questions are linked to subjects and assigned to examinations.

• Results: After a student attends an exam, results are generated. A result entry includes Certificate_no, Grade_Obtained, Exam_ID, and Student_ID.

The system ensures:

• Students can register and attend exams online.

• Admins can create exams, manage subjects, and assign questions.

• Exam centers are maintained for managing locations.

• Automatic evaluation of student answers and generation of results.

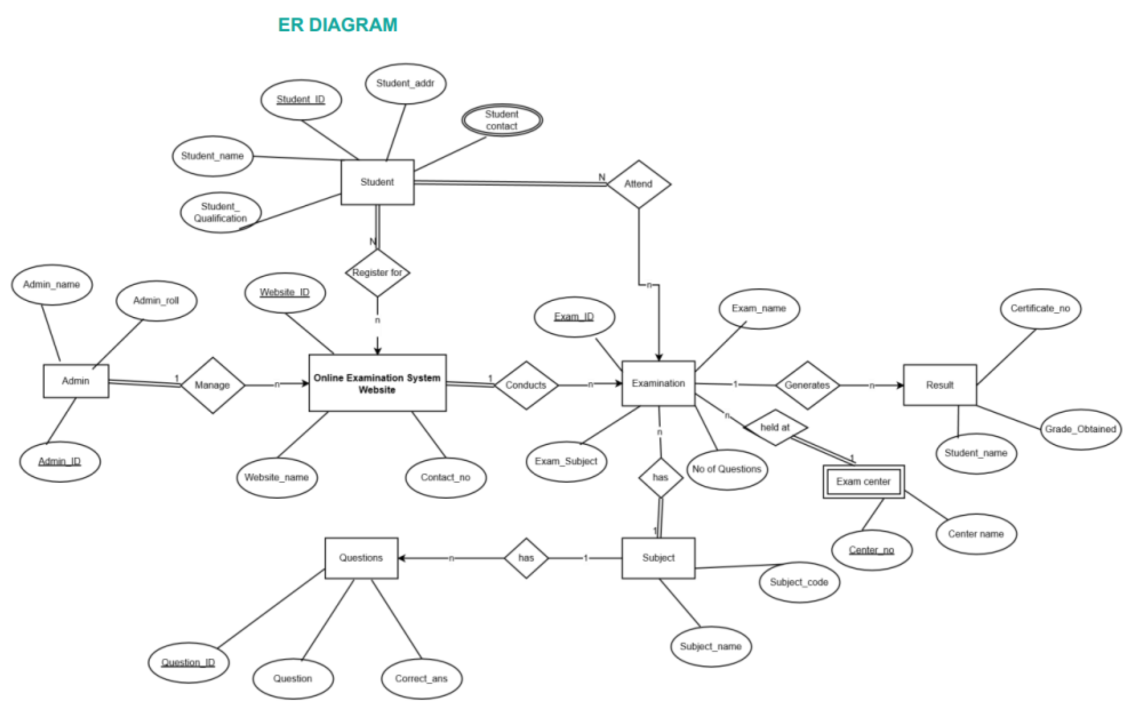
• Issuance of certificates and grades to students.

• This project provides an integrated solution for managing online examinations in a structured, secure, and efficient manner.

4.List of Software/Tools/Programming languages used

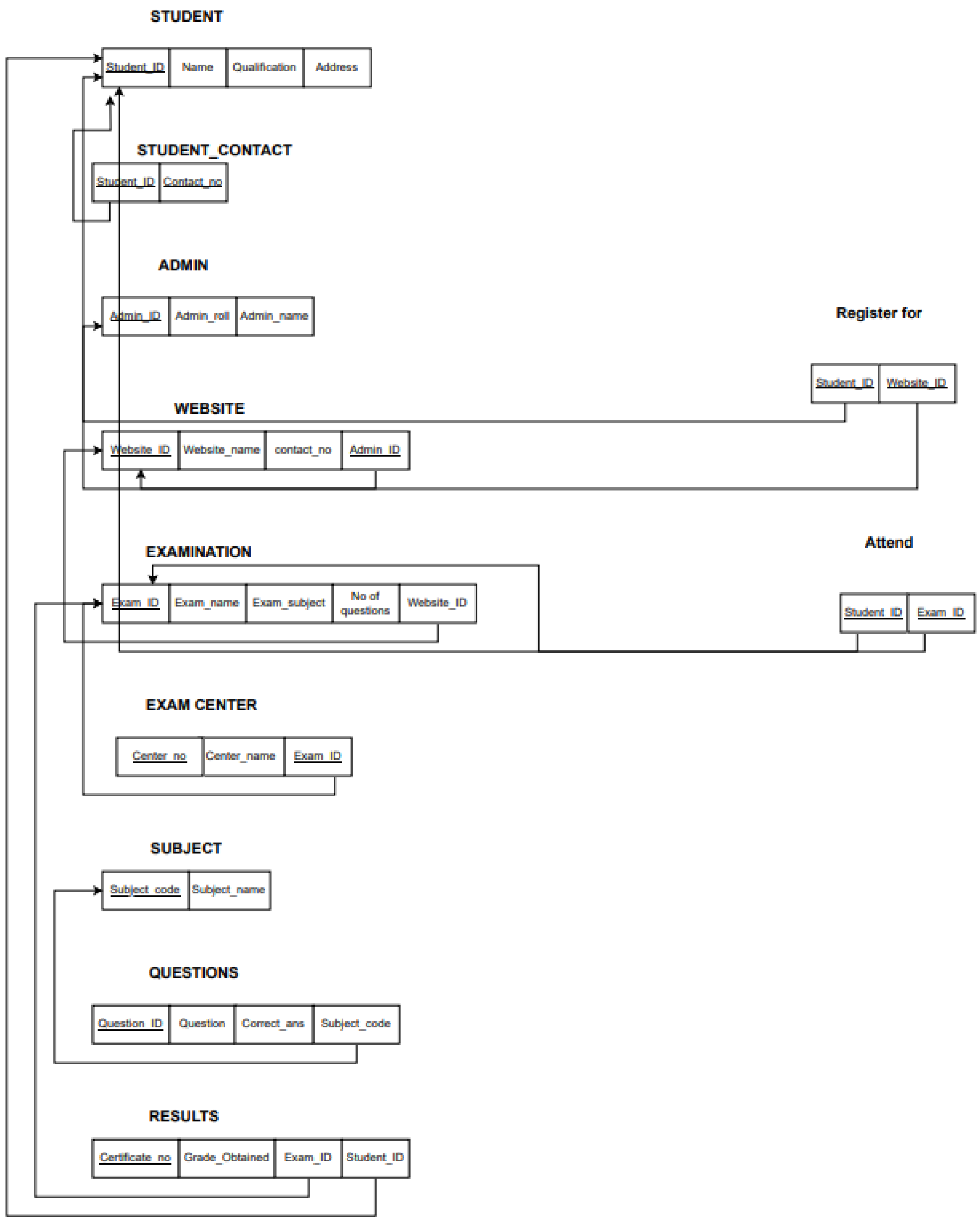
Category	Name / Description
Frontend Interface	**Tkinter** (Python GUI library for creating forms and result display)
Backend Language	**Python 3**
Database Management System (DBMS)	**MySQL** (used to store students, exams, questions, results, etc.)
Database Connectivity	**MySQL Connector for Python**
Stored Programs	**MySQL Procedures, Functions, and Triggers** (for automation and validation)
IDE / Code Editor	**Visual Studio Code / IDLE (Python)**
Server Environment	**MySQL Server 8.0**
Operating System	**Windows 10 / 11**
Additional Tools	**Command Prompt / MySQL CLI** (for executing SQL scripts)

5.ER Diagram



ONLINE EXAMINATION SYSTEM

6.Relational Schema:



7.DDL Commands

-- STUDENT TABLE

```
Create database Examination_system;
use Examination_system;

CREATE TABLE Student (
   Student_ID INT PRIMARY KEY,
   Name VARCHAR(100) NOT NULL,
   Qualification VARCHAR(50),
   Address VARCHAR(200)
);
```

-- STUDENT CONTACT TABLE

```
CREATE TABLE Student_Contact (
   Student_ID INT,
   Contact_No VARCHAR(15),
   PRIMARY KEY (Student_ID, Contact_No),
   FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID) ON DELETE CASCADE
);
```

-- ADMIN TABLE

```sql
CREATE TABLE ADMIN (
        Admin_ID INT PRIMARY KEY,
  Admin_Name VARCHAR(100) NOT NULL,
  Admin_Role VARCHAR(50) NOT NULL
);

-- WEBSITE TABLE
CREATE TABLE Website (
        Website_ID INT PRIMARY KEY,
  Website_Name VARCHAR(100) NOT NULL,
  Contact_Details VARCHAR(100),
  Admin_ID INT,
  FOREIGN KEY (Admin_ID) REFERENCES Admin(Admin_ID)
);


-- SUBJECT TABLE
CREATE TABLE Subject (
  Subject_Code INT PRIMARY KEY,
  Subject_Name VARCHAR(100) NOT NULL
);

-- EXAMINATION TABLE
CREATE TABLE Examination (
  Exam_ID INT PRIMARY KEY,
  Exam_Name VARCHAR(100) NOT NULL,
  Exam_Subject INT,
```

```sql
    Num_Questions INT,
    Website_ID INT,
    FOREIGN KEY (Exam_Subject) REFERENCES Subject(Subject_Code),
    FOREIGN KEY (Website_ID) REFERENCES Website(Website_ID)
);
```

-- EXAM CENTER TABLE

```sql
CREATE TABLE Exam_Center (
    Center_No INT PRIMARY KEY,
    Center_Name VARCHAR(100) NOT NULL,
    Exam_ID INT,
    FOREIGN KEY (Exam_ID) REFERENCES Examination(Exam_ID)
);
```

-- QUESTION TABLE

```sql
CREATE TABLE Question (
    Question_ID INT PRIMARY KEY,
    Question_Text VARCHAR(500) NOT NULL,
    Correct_Answer VARCHAR(100) NOT NULL,
    Subject_Code INT,
    Exam_ID INT,
    FOREIGN KEY (Subject_Code) REFERENCES Subject(Subject_Code),
    FOREIGN KEY (Exam_ID) REFERENCES Examination(Exam_ID)
);
```

-- RESULT TABLE

```sql
CREATE TABLE Result (
```

```
    Certificate_No INT PRIMARY KEY,
    Grade_Obtained CHAR(2),
    Exam_ID INT,
    Student_ID INT,
    FOREIGN KEY (Exam_ID) REFERENCES Examination(Exam_ID),
    FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID)
);
```

-- NEW RELATIONSHIP TABLE: STUDENT-EXAM REGISTRATION

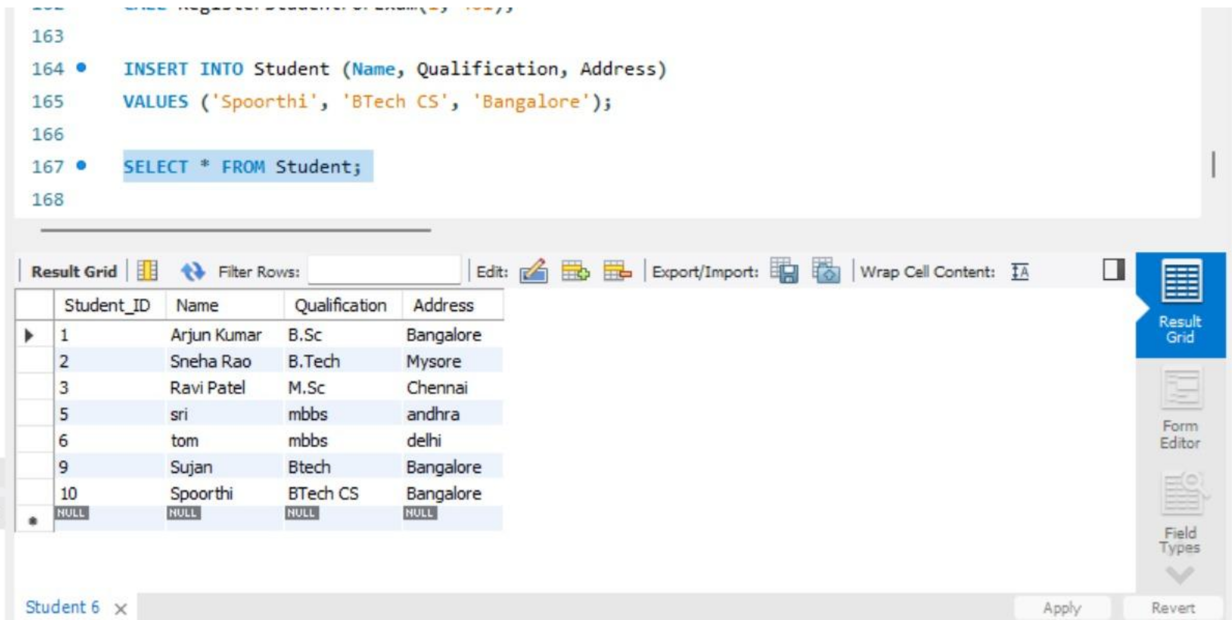
```
CREATE TABLE Student_Exam (
    Student_ID INT,
    Exam_ID INT,
    Registration_Date DATE DEFAULT (CURRENT_DATE),
    PRIMARY KEY (Student_ID, Exam_ID),
    FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID),
    FOREIGN KEY (Exam_ID) REFERENCES Examination(Exam_ID)
);
```

-- ATTEND TABLE

```
CREATE TABLE Attend (
    Student_ID INT,
    Exam_ID INT,
    Attendance_Date DATE DEFAULT (CURRENT_DATE),
    PRIMARY KEY (Student_ID, Exam_ID),
    FOREIGN KEY (Student_ID) REFERENCES Student(Student_ID) ON DELETE CASCADE,
    FOREIGN KEY (Exam_ID) REFERENCES Examination(Exam_ID) ON DELETE CASCADE
);
```

8. CRUD operation Screenshots

CREATE



READ

```sql
SELECT
    s.Name AS Student_Name,
    e.Subject AS Exam_Subject,
    r.Score,
    r.Grade_Obtained
FROM Result r
JOIN Student s ON r.Student_ID = s.Student_ID
JOIN Exam e ON r.Exam_ID = e.Exam_ID;
```

Student_Name	Exam_Subject	Score	Grade_Obtained
Arjun Kumar	Mathematics	92	A+
Sneha Rao	Physics	81	A
Ravi Patel	Chemistry	68	C
Sneha Rao	Mathematics	37	F
Arjun Kumar	Mathematics	98	A+
Sneha Rao	Mathematics	88	A
Sneha Rao	Mathematics	93	A+

Result 14 Read Only

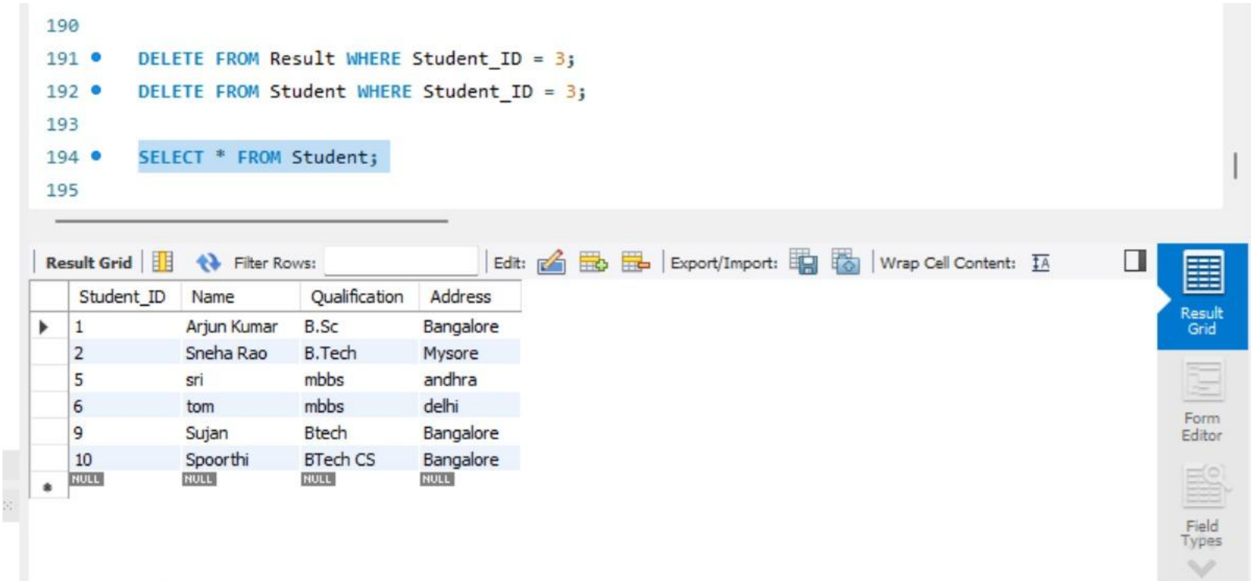
UPDATE

```sql
UPDATE Result
SET Score = 90
WHERE Student_ID = 1 AND Exam_ID = 1;

SELECT * FROM Result WHERE Student_ID = 1;
```

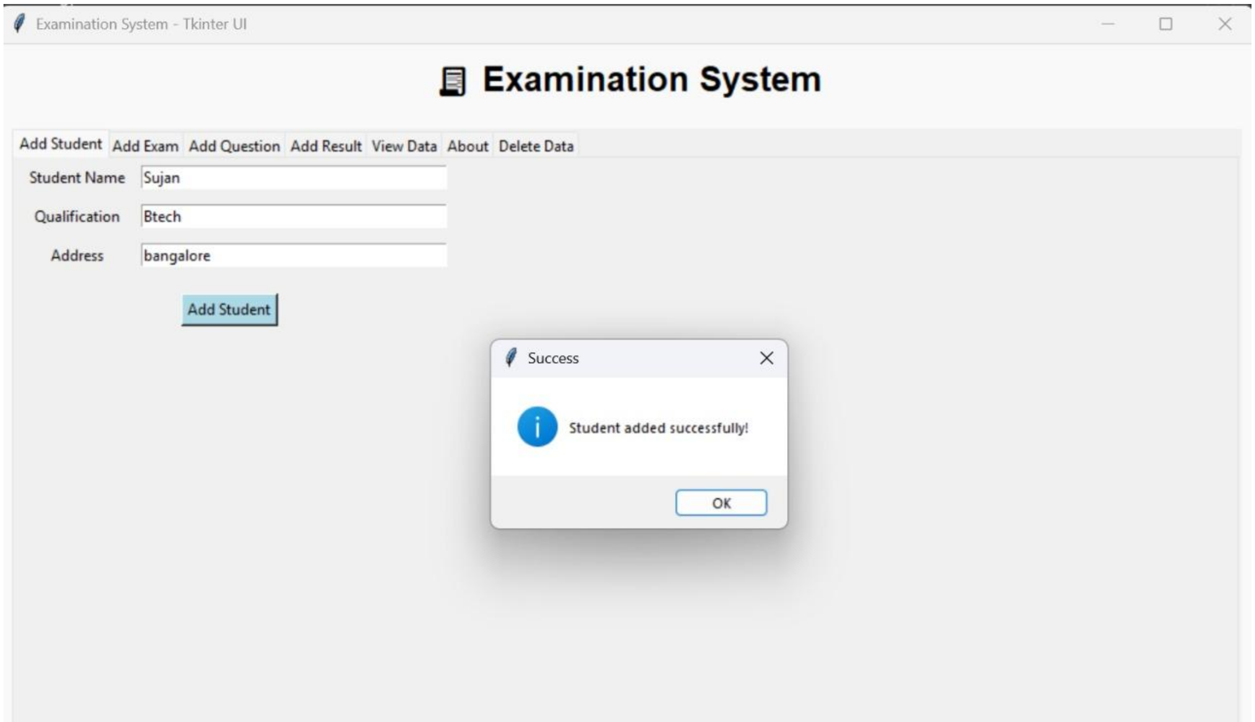
Result_ID	Certificate_No	Score	Grade_Obtained	Exam_ID	Student_ID
1	501	90	A+	1	1

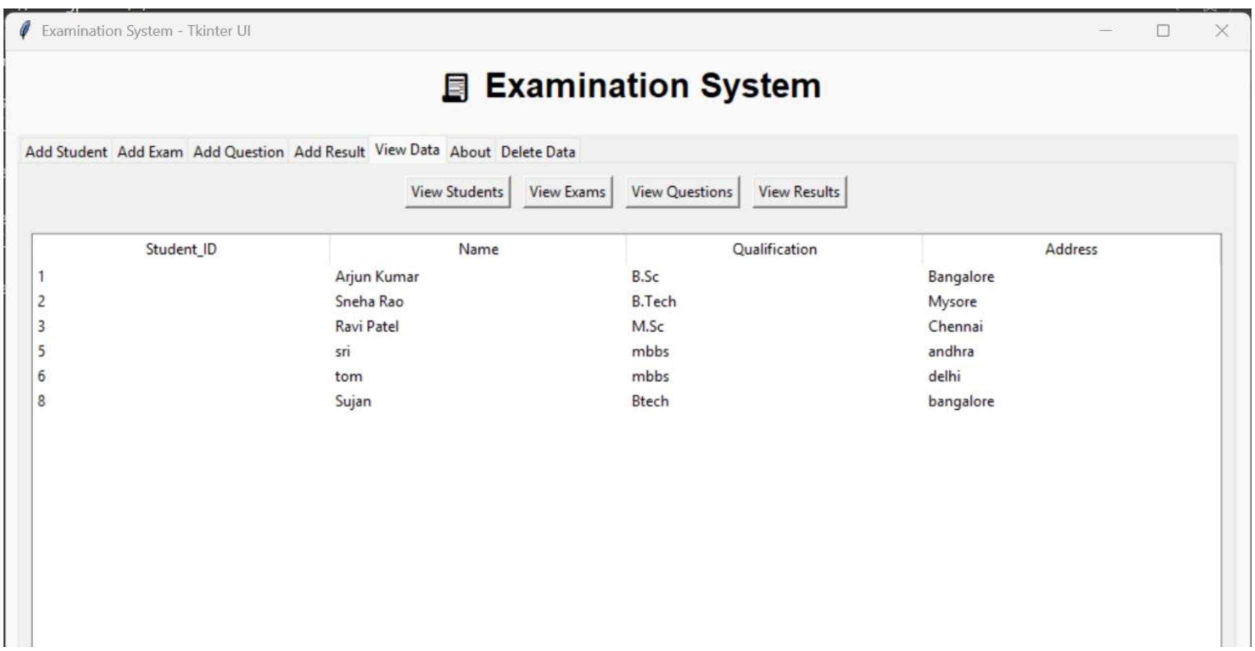
DELETE



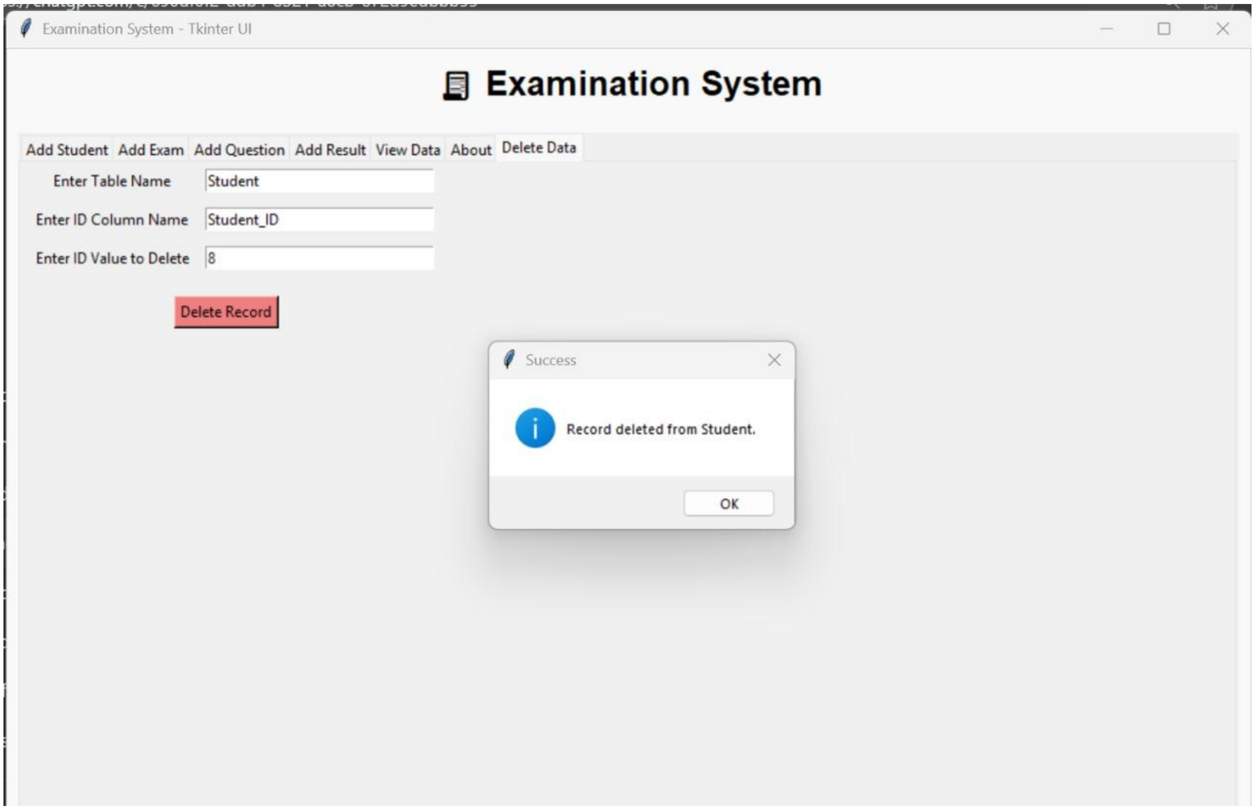
9.List of functionalities/features of the application and its associated screenshots using front end

1.ADD STUDENT

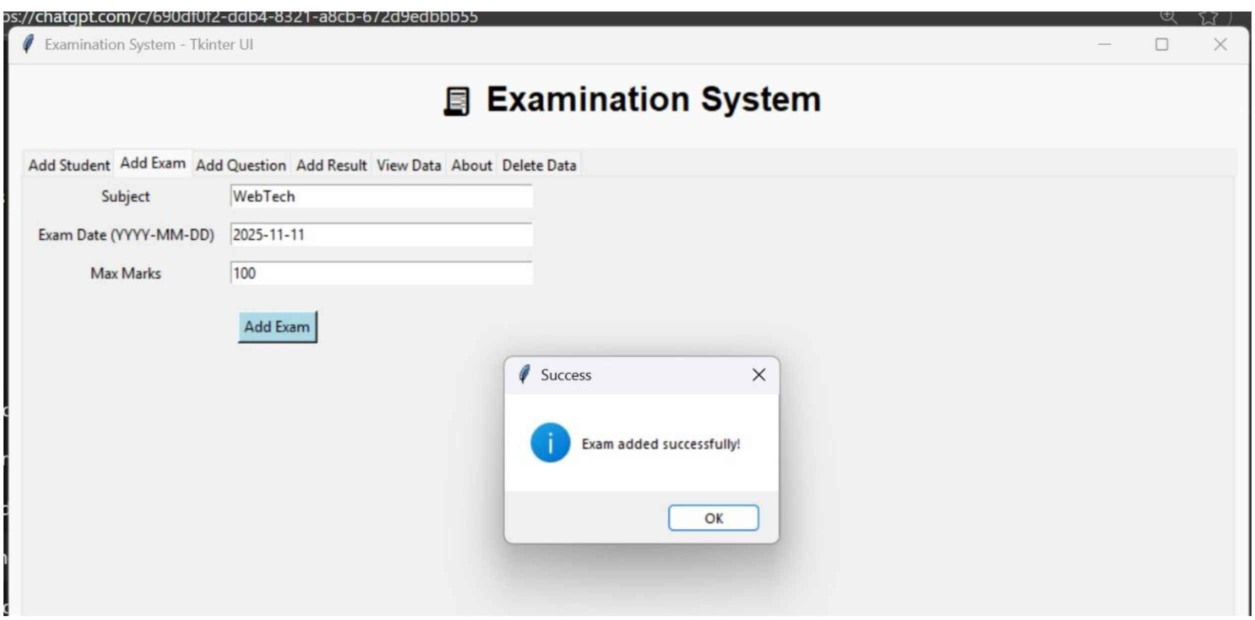

Examination System - Tkinter UI
Examination System
Add Student
Add Exam
Add Question
Add Result
View Data
About
Delete Data
Student Name
Sujan
Qualification
Btech
Address
bangalore
Add Student
Success
Student added successfully!
OK

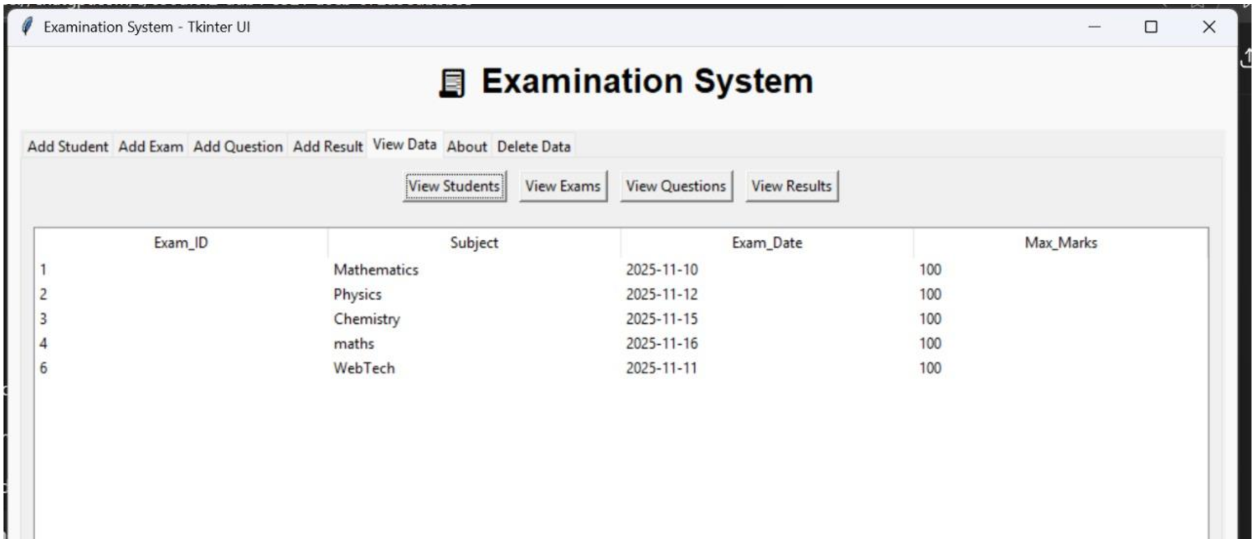

Examination System - Tkinter UI
Examination System
Add Student
Add Exam
Add Question
Add Result
View Data
About
Delete Data
View Students
View Exams
View Questions
View Results
Student_ID
Name
Qualification
Address
1 Arjun Kumar B.Sc Bangalore
2 Sneha Rao B.Tech Mysore
3 Ravi Patel M.Sc Chennai
5 sri mbbs andhra
6 tom mbbs delhi
8 Sujan Btech bangalore

2.DELETE

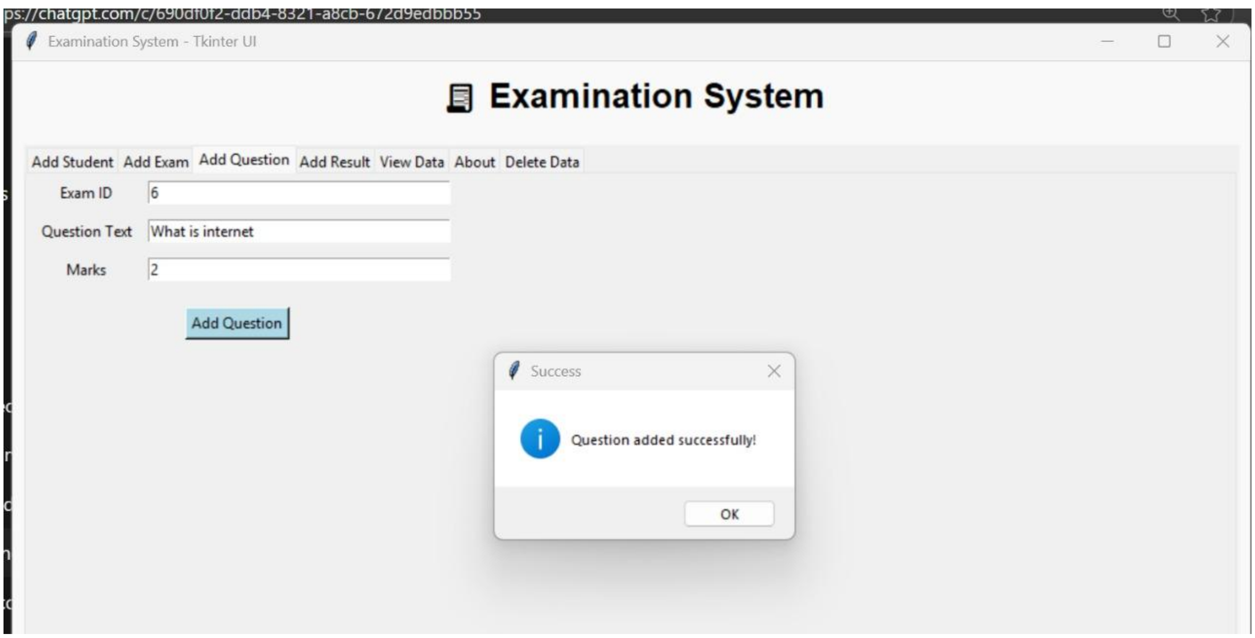


3.ADD EXAM

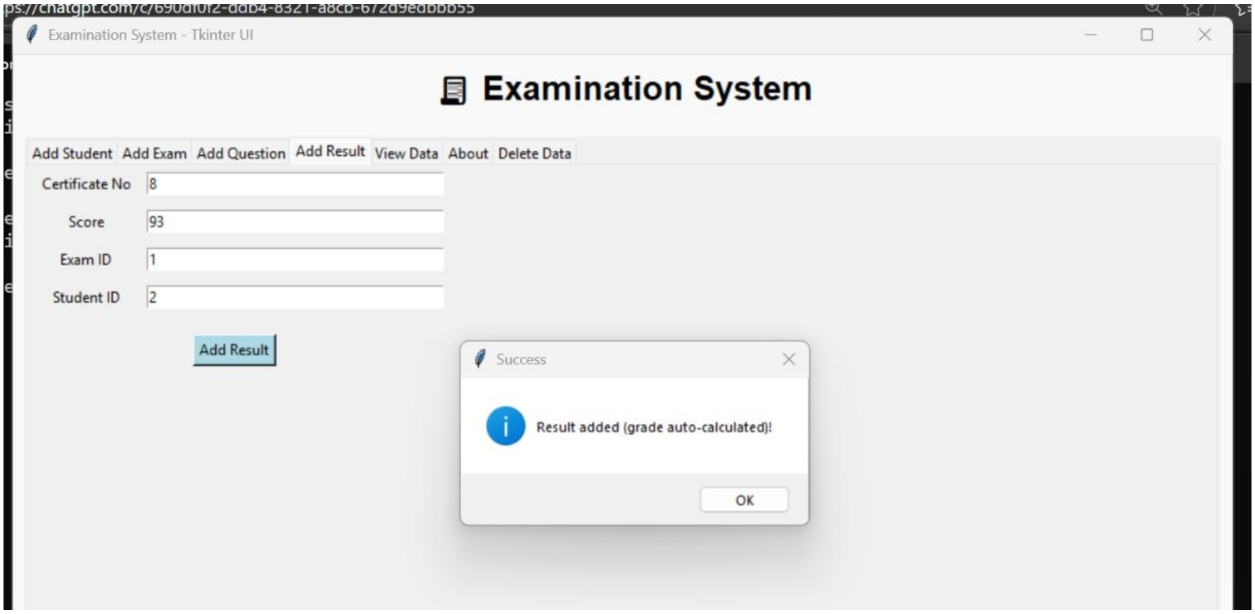


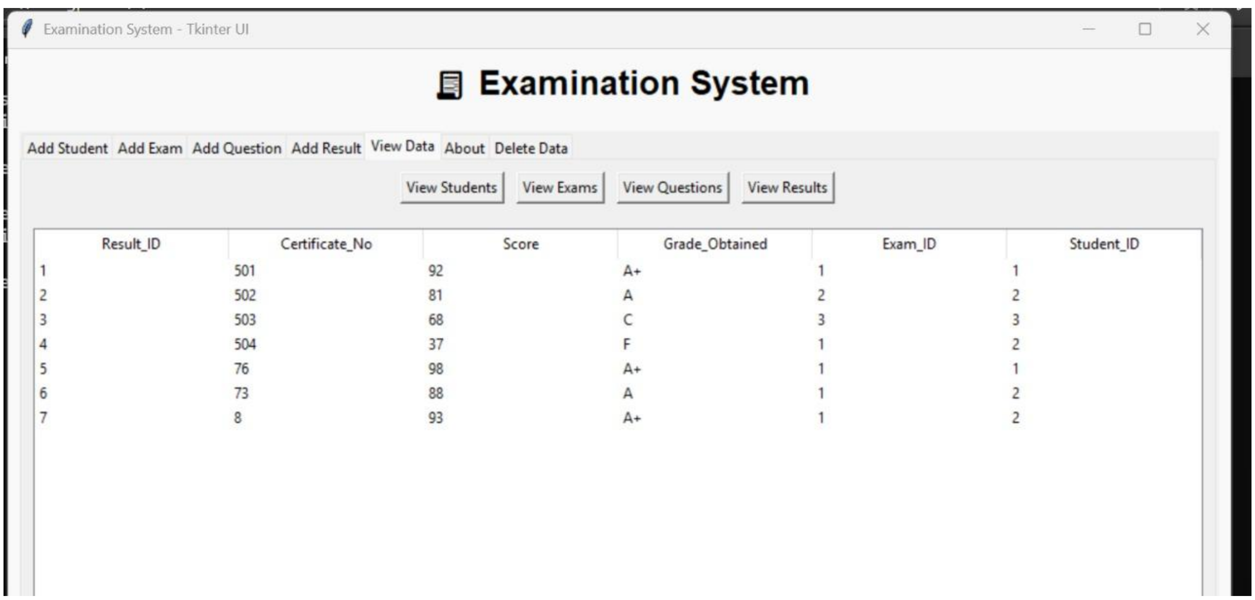


4.ADD QUESTION



5.ADD RESULT


Examination System - Tkinter UI
Examination System
Add Student
Add Exam
Add Question
Add Result
View Data
About
Delete Data
Certificate No
8
Score
93
Exam ID
1
Student ID
2
Add Result
Success
Result added (grade auto-calculated)!
OK


Examination System - Tkinter UI
Examination System
Add Student
Add Exam
Add Question
Add Result
View Data
About
Delete Data
View Students
View Exams
View Questions
View Results
Result_ID Certificate_No Score Grade_Obtained Exam_ID Student_ID
1 501 92 A+ 1 1
2 502 81 A 2 2
3 503 68 C 3 3
4 504 37 F 1 2
5 76 98 A+ 1 1
6 73 88 A 1 2
7 8 93 A+ 1 2

10.Triggers, Procedures/Functions, Nested query, Join, Aggregate queries

1. TRIGGER

Trigger Name: trg_auto_grade

```
DELIMITER //

CREATE TRIGGER trg_auto_grade

BEFORE INSERT ON Result

FOR EACH ROW

BEGIN

   SET NEW.Grade_Obtained = CalculateGrade(NEW.Score);

END //

DELIMITER ;
```

2. FUNCTIONS

Function Name: CalculateGrade

```
DELIMITER //

CREATE FUNCTION CalculateGrade(score INT)

RETURNS VARCHAR(2)

DETERMINISTIC

BEGIN

   DECLARE grade VARCHAR(2);

   IF score >= 90 THEN
```

```
        SET grade = 'A+';
    ELSEIF score >= 80 THEN
        SET grade = 'A';
    ELSEIF score >= 70 THEN
        SET grade = 'B';
    ELSEIF score >= 60 THEN
        SET grade = 'C';
    ELSEIF score >= 40 THEN
        SET grade = 'D';
    ELSE
        SET grade = 'F';
    END IF;
    RETURN grade;
END //
DELIMITER ;
```

3. STORED PROCEDURES

(a) AddStudent

Adds a new student record to the database.

```
DELIMITER //
CREATE PROCEDURE AddStudent(
    IN p_Name VARCHAR(100),
```

```
    IN p_Qualification VARCHAR(50),
    IN p_Address VARCHAR(100)
)
BEGIN
    INSERT INTO Student (Name, Qualification, Address)
    VALUES (p_Name, p_Qualification, p_Address);
END //
DELIMITER ;
```

(b) **RegisterStudentForExam**

Registers a student for a given exam.

```
DELIMITER //
CREATE PROCEDURE RegisterStudentForExam(
    IN p_student_id INT,
    IN p_exam_id INT
)
BEGIN
    INSERT INTO Student_Exam (Student_ID, Exam_ID)
    VALUES (p_student_id, p_exam_id);
END //
DELIMITER ;
```

(c) AddQuestion

Adds a question for a specific subject and exam.

```
DELIMITER //

CREATE PROCEDURE AddQuestion(

    IN p_question_text VARCHAR(255),

    IN p_correct_answer VARCHAR(100),

    IN p_subject_code INT,

    IN p_exam_id INT

)

BEGIN

    INSERT INTO Question (Question_Text, Correct_Answer, Subject_Code, Exam_ID)

    VALUES (p_question_text, p_correct_answer, p_subject_code, p_exam_id);

END //

DELIMITER ;
```

4. NESTED QUERY (Subquery)

Example 1 – Find Students Who Scored Above Average in DBMS Final Exam:

```
SELECT Name

FROM Student

WHERE Student_ID IN (

    SELECT Student_ID

    FROM Result

    WHERE Exam_ID = 401
```

```sql
    AND Score > (SELECT AVG(Score) FROM Result WHERE Exam_ID = 401)
);
```

5. JOIN QUERIES

(a) INNER JOIN

```sql
SELECT s.Name, e.Exam_Name, r.Score, r.Grade_Obtained
FROM Result r
JOIN Student s ON r.Student_ID = s.Student_ID
JOIN Examination e ON r.Exam_ID = e.Exam_ID;
```

(b) LEFT JOIN

```sql
SELECT s.Name, COALESCE(e.Exam_Name, '—') AS Exam_Name,
       COALESCE(r.Score, '—') AS Score,
       COALESCE(r.Grade_Obtained, '—') AS Grade
FROM Student s
LEFT JOIN Result r ON s.Student_ID = r.Student_ID
LEFT JOIN Examination e ON r.Exam_ID = e.Exam_ID
ORDER BY s.Name;
```

6. AGGREGATE QUERIES

(a) COUNT()

```sql
SELECT e.Exam_Name, COUNT(se.Student_ID) AS No_of_Students
FROM Student_Exam se
```

JOIN Examination e ON se.Exam_ID = e.Exam_ID

GROUP BY e.Exam_Name;

(b) AVG()

SELECT e.Exam_Name, AVG(r.Score) AS Average_Score

FROM Result r

JOIN Examination e ON r.Exam_ID = e.Exam_ID

GROUP BY e.Exam_Name;

(c) **MAX() / MIN()**

SELECT e.Exam_Name, MAX(r.Score) AS Highest_Score, MIN(r.Score) AS Lowest_Score

FROM Result r

JOIN Examination e ON r.Exam_ID = e.Exam_ID

GROUP BY e.Exam_Name;

(d) **SUM()**

SELECT s.Name, SUM(r.Score) AS Total_Score

FROM Result r

JOIN Student s ON r.Student_ID = s.Student_ID

GROUP BY s.Name;

1. **Invoking Stored Procedure — AddStudent**

   CALL AddStudent('Arjun Rao', 'B.Tech', 'Bangalore');

   def add_student():

```
name = name_entry.get().strip()
qualification = qualification_entry.get().strip()
address = address_entry.get().strip()

cursor.callproc("AddStudent", [name, qualification, address])
conn.commit()
messagebox.showinfo("Success", " Student added successfully!")
```

2. **Invoking Stored Procedure — RegisterStudentForExam**

```
cursor.callproc("RegisterStudentForExam", [4, 403])
conn.commit()
```

3. **Invoking Stored Procedure — AddQuestion**

```
cursor.callproc("AddQuestion", ['What is a primary key?', 'A unique identifier for a record', 301, 403])
conn.commit()
```

4. **Invoking Function — CalculateGrade()**

```
SELECT CalculateGrade(86) AS Grade;
```

5. **Invoking Trigger — trg_auto_grade**

```
INSERT INTO Result (Certificate_No, Score, Exam_ID, Student_ID)
VALUES (705, 86, 403, 4);
```

12: Github repo link

https://github.com/s-p-o-o-r-t-h-i-v/DBMS-mini-project-ONLINE-EXAMINATION-SYSTEM