Information Technology Institute
Full-Stack .Net Track

Examination System

Instructor:
Eng. **Sara Salah**

Presented by:
Omar Ramadan Mahmoud
Qassem Shaban Ali
Mohammed Haggagg Mayhoub
Mustafa Hafez Mustafa
Ramez Noshy Tawfeek

ABSTRACT

1. **Project Overview**

   The Examination System is a comprehensive database-driven solution designed to manage student examinations, instructors, courses, and results. It facilitates automated exam scheduling, student attempt tracking, and result storage.

2. **System Objectives**

   - Efficiently manage students, instructors, and courses.
   - Automate exam scheduling and prevent scheduling conflicts.
   - Enable online exam attempts with question storage.
   - Store and retrieve student exam results for analysis.
   - Secure user authentication and role-based access.
   - Optimize database performance with indexing and queries.

3. **Functional Requirements**

   3.1. User Management Students

   - Instructors, and admins should have unique accounts.
   - Role-based authentication and authorization.

   3.2. Student Management Add

   - Update, and delete student information.
   - Assign students to intakes, branches, and tracks.

   3.3. Instructor Management Add

   - Update, and remove instructors.
   - Assign instructors to multiple courses.

   3.4. Course Management

   - Create and modify course details.
   - Define course prerequisites and exam structures.

3.5. Exam Management

- Create exams with start/end times.
- Define exam types (normal/corrective exams).
- Assign questions with varying difficulty levels.
- Prevent overlapping exams using automated scheduling.

3.6. Question & Answer Management

- Store multiple-choice, true/false, and textual questions.
- Define correct answers and scoring mechanisms.

3.7. Student Exam Attempts

- Allow students to attempt exams online.
- Store each student's responses securely.

3.8. Exam Results Processing

- Calculate scores and store exam results.
- Generate detailed reports on student performance.

4. Non-Functional Requirements.

1. Performance

- Optimize queries using indexing for fast data retrieval.
- Implement efficient stored procedures and triggers.

4.2. Security

- Encrypt passwords using a secure hashing algorithm.
- Restrict access to sensitive data based on user roles.

4.3. Scalability

- Design the database to support an increasing number of users.
- Enable automatic backups for data recovery.

4.4. Reliability & Maintainability

- Ensure system availability with scheduled database backups.
- Provide detailed logging for debugging and maintenance.

5. Database Schema Overview

- Students: Stores student details (ID, Name, Email, Track, and Intake).
- Instructors: Stores instructor details and assigned courses.
- Courses: Stores course descriptions, min/max passing grades.
- Exams: Stores exam schedules, types, and linked courses.
- Questions: Stores different types of questions and answers.
- Exam Attempts: Tracks student attempts with timestamps.
- Exam Results: Stores student answers and scores.
- Accounts: Manages login credentials and user roles.

6. Database Optimization Strategies

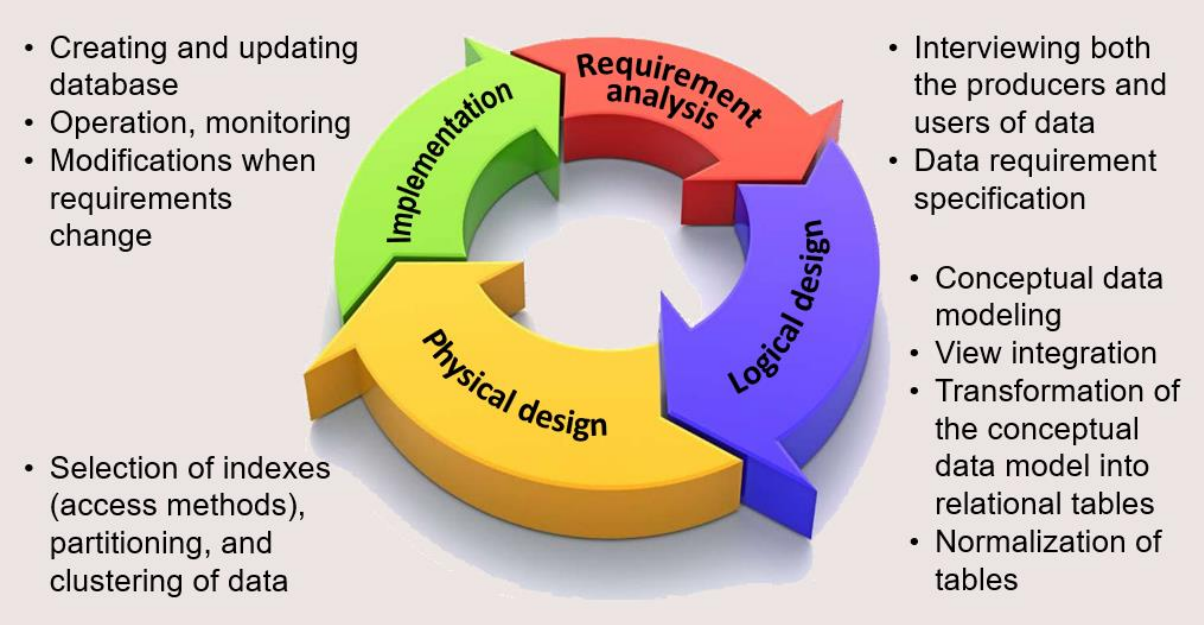
- Indexes: Implement indexing on frequently queried columns (e.g., Exam Date, Question Course, and Exam Results).
- Stored Procedures: Automate repetitive tasks (e.g., adding exams, generating reports).
- Triggers: Enforce business rules (e.g., prevent overlapping exams).
- Views: Simplify complex queries (e.g., viewing student results).

7. Backup & Recovery Strategy

- Daily backups scheduled using SQL Agent.
- Backup stored at C:\Backup\ExaminationSystem.bak.
- Recovery mechanism to restore data from the latest backup.

IMPLEMENTATION LIFE CYCLE

The implementation life cycle is an iterative process needed to ensure the system meets the requirements that outline the phases a system goes through, from entity relationship diagrams to reports and interactive dashboards.



1. Requirements Analysis

- Identify business needs and objectives.
- Gather information about the system's requirements.
- Define what data needs to be stored and accessed.

2. Conceptual Design

- Create an **Entity-Relationship Diagram (ERD)**.
- Define **entities, relationships, and attributes**.

3. Logical Design

- Convert the ERD into a relational schema.
- Normalize the database to avoid redundancy.

.

4. Physical Design

- Define **indexes, constraints, and storage** structures.
- Optimize database performance.

5. Implementation

- Convert the logical design into an actual database.
- Load data into tables.
- Set up security measures.

6. Testing & Evaluation

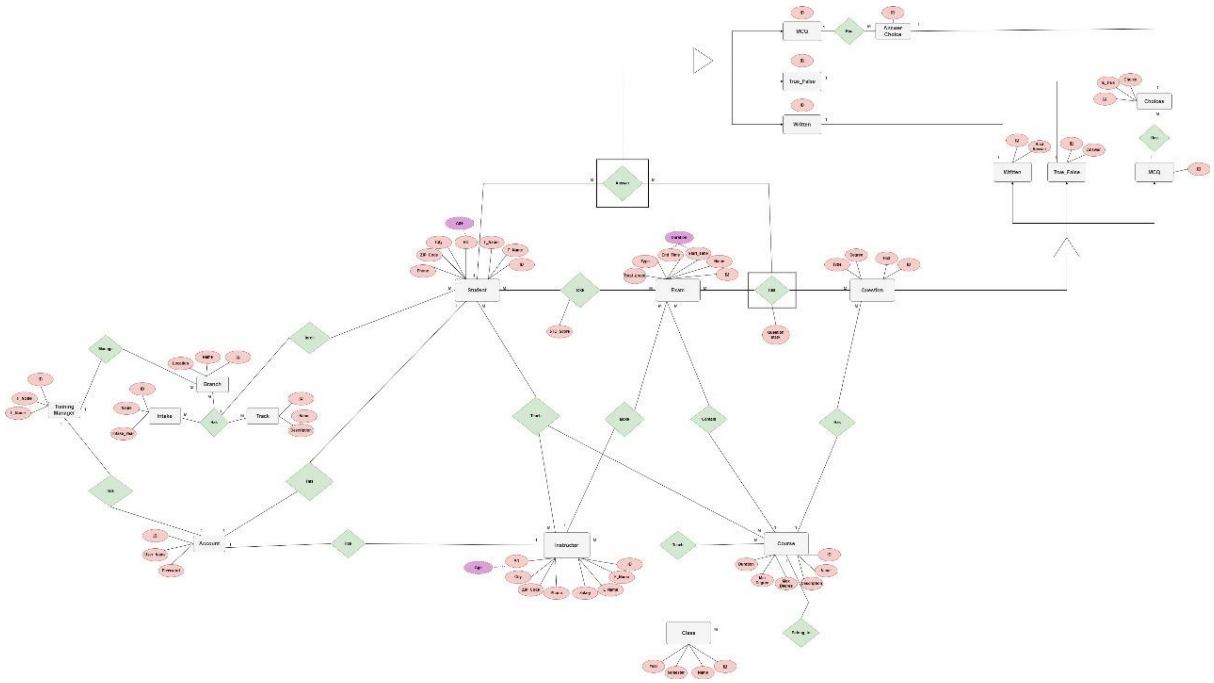
- Run test cases to validate data integrity.
- Optimize query performance.

7. Maintenance & Monitoring

- Regularly update the database.
- Perform backups and performance tuning.

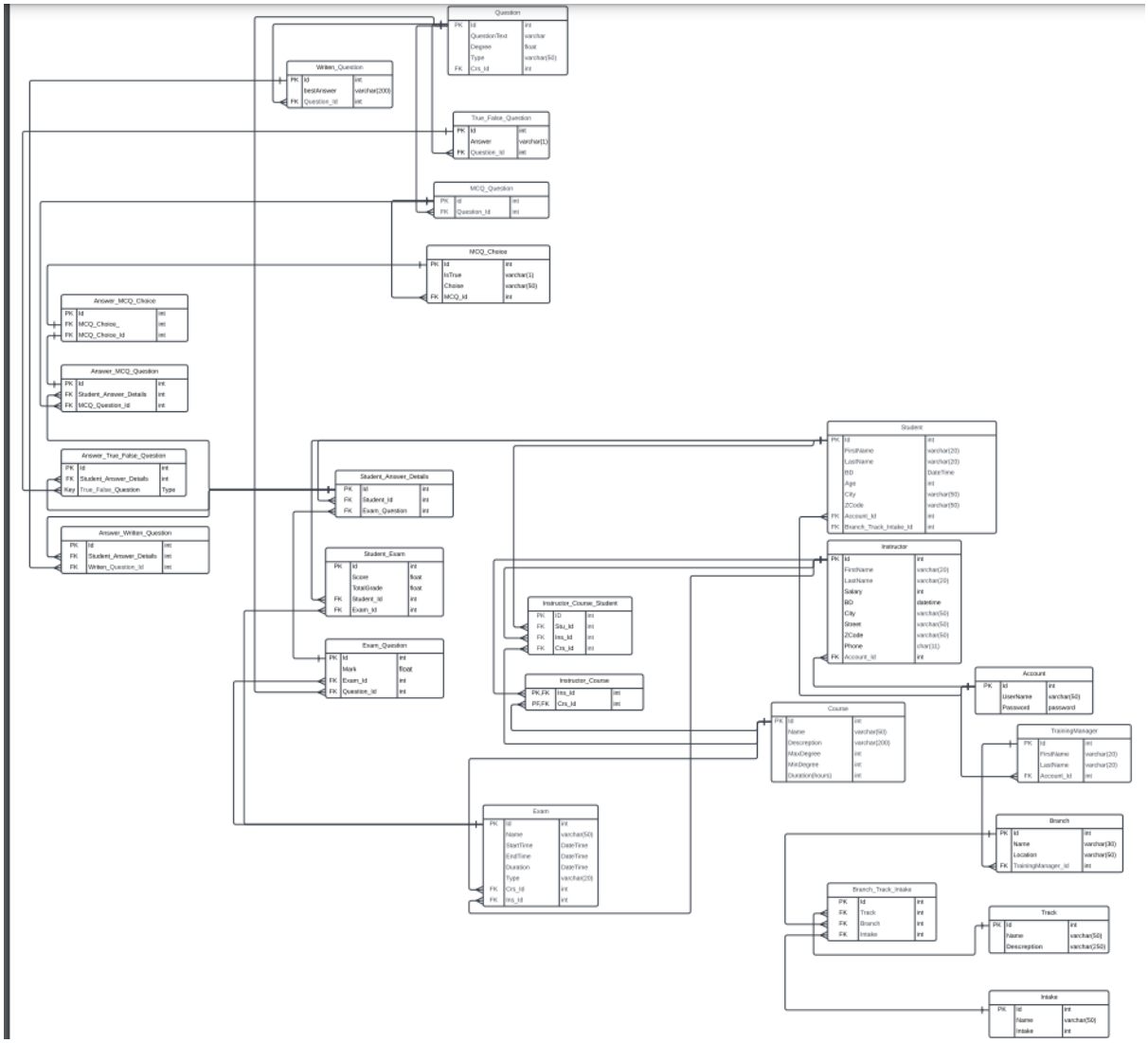
DATABASE DESIGN (ERD)

Examination system ERD includes entities such as Intake, department, students, questions, exams, courses, etc. and their relationships to each other.



DATABASE DESIGN (MAPPING)

Mapping in the examination system refers to the process of creating a correspondence or association between different entities or attributes in the system needed to implement in the SQL Server Management Studio.



IMPLEMENTATION

1. Requirement: The system must have a well-structured relational database to store students, instructors, courses, exams, and results.

```
CREATE DATABASE ExaminationSystem;
GO
USE ExaminationSystem;
GO
```

2. User Roles and Authentication Requirement: The system must support three roles: Student, Instructor, and Admin.

```
CREATE TABLE Accounts (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    Username NVARCHAR(200) UNIQUE NOT NULL,
    Password NVARCHAR(20) NOT NULL
);
```

3. **The `Instructor Course Student`** table ensures that each student is enrolled in a specific course under a specific instructor and within a particular class.

```
Create table Instructor_Course_Student
(
    ClassId int not null,
    StdId   int not null,
    CrsId   int not null,
    InsId   int not null,
    EnrollmentDate date default getdate(),
    Evaluation int check (Evaluation between 0 and 100),
    constraint PK_Ins_Crs_Std primary key (CrsId, ClassId, StdId),
    constraint FK_Student foreign key (StdId) references Student(Id) on delete No Action,
    constraint FK_Course_Student foreign key (CrsId) references Course(Id) on delete No Action,
    constraint FK_Class_Student foreign key (ClassId) references Class(Id) on delete No Action,
    constraint FK_Instructor_Student foreign key (InsId) references Instructor(Id) on delete No Action,

    constraint UQ_Course_Class unique (CrsId, ClassId, InsId)
) on NextFG
```

4. Exam and Question Management Requirement: The system must allow different types of questions (MCQ, True/False, Open-End).

```
Create table Question
(
    Id int primary key identity,
    Body nvarchar(100),
    Degree float ,
    QuestionType nvarchar(50),
    CourseId int foreign key references Course(Id)
) on QuestionFG
```

5. Exam Scheduling and Constraints Requirement: Exams must not overlap for the same course.

```
-- Trigger: Prevent Exam Overlapping
CREATE TRIGGER trg_PreventExamOverlap
ON Exam
AFTER INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM inserted i
        JOIN Exam e ON i.CourseID = e.CourseID
        WHERE i.StartTime BETWEEN e.StartTime AND e.EndTime
    )
    BEGIN
        PRINT 'Exam timings are overlapping!';
        ROLLBACK;
    END;
END;
```

6. Exam Submission and Grading Requirement: Students must submit their answers, and the system must calculate scores.

```
Create Table Student_Answer_Details
(
    Id int primary key identity ,
    ManualScore int ,
    StudentId int foreign key references Student(Id),
    Exam_QuestionId int foreign key references Exam_Question(Id)
) on ResultsFG
```

7. Performance Optimization Requirement: Indexing must be used to speed up queries.

```
-- Indexing for Optimization
CREATE NONCLUSTERED INDEX IDX_ExamDate ON Exams (StartTime);
CREATE NONCLUSTERED INDEX IDX_QuestionCourse ON Questions (CourseID);
CREATE NONCLUSTERED INDEX IDX_ExamCourse ON Exams (CourseID);
CREATE NONCLUSTERED INDEX IDX_InstructorCourses ON InstructorCourses (InstructorID, CourseID);
```

8. Reporting and Analytics Requirement: A view should be created to analyse student results.

```
-- View: Get Exam Results
CREATE VIEW StudentExamResults_Ranking AS
SELECT
    e.ExamID, s.StudentID, s.Name, e.CourseID, c.Name AS CourseName,
    er.Score, er.StudentAnswer
FROM ExamResults er
JOIN ExamAttempts ea ON er.AttemptID = ea.AttemptID
JOIN Exams e ON ea.ExamID = e.ExamID
JOIN Students s ON ea.StudentID = s.StudentID
JOIN Courses c ON e.CourseID = c.CourseID;
```

9. Backup and Data Recovery Requirement: Daily backup of the database must be performed.

```
BACKUP DATABASE ExaminationSystem
TO DISK = 'C:\Backup\ExaminationSystem.bak'
WITH FORMAT, INIT, NAME = 'Daily Backup';
```

- Create Exam is a stored procedure created to get the student id with the exam number if the student id is not exist print an error message, @Name, @CourseID, @startTime,@endTime,@type,@totalGrade,@mcqCount,@trueFalseCount,@ writtenCount,@mcqMark.@trueFalseMark,@writtenMark,@username.@passw ord.

```
----------------------  Proc To Create Exam -------------------------------
go
CREATE OR ALTER PROCEDURE CreateExam
    @Name VARCHAR(50),
    @StartTime DATETIME,
    @EndTime DATETIME,
    @Type VARCHAR(50),
    @TotalGradeOfExam INT,
    @CourseId INT,
    @MCQCount INT,
    @TrueFalseCount INT,
    @WrittenCount INT,
    @MCQMark INT,
    @TrueFalseMark INT,
    @WrittenMark INT,
    @UserName VARCHAR(50),
    @Password VARCHAR(40)
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @AccId INT, @InsID INT, @ExamId INT;
    DECLARE @AvailableMCQ INT, @AvailableTrueFalse INT, @AvailableWritten INT;


    SELECT @AccId = Id FROM UserManagement.Account WHERE UserName = @UserName AND Password = @Password;
    IF (@AccId IS NULL)
    BEGIN
        PRINT 'User or password is not valid';
        RETURN;
    END;

    SELECT @InsID = Id FROM UserManagement.Instructor WHERE AccountId = @AccId;
    IF (@InsID IS NULL)
    BEGIN
        PRINT 'You are not an instructor';
        RETURN;
    END;
```

```sql
IF NOT EXISTS (SELECT 1 FROM AcademicRecords.Course WHERE Id = @CourseId AND InstructorId = @InsID)
BEGIN
    PRINT 'You are not the instructor of this course';
    RETURN;
END;

IF (@TotalGradeOfExam != (@MCQCount * @MCQMark + @TrueFalseCount * @TrueFalseMark + @WrittenCount * @WrittenMark))
BEGIN
    PRINT 'TotalGradeOfExam must equal the sum of question marks';
    RETURN;
END;

SELECT
    @AvailableMCQ = COUNT(*) FROM ExamManagement.Question WHERE QuestionType = 'Multiple Choice' AND CourseId = @CourseId;

SELECT
    @AvailableTrueFalse = COUNT(*) FROM ExamManagement.Question WHERE QuestionType = 'True/False' AND CourseId = @CourseId;

SELECT
    @AvailableWritten = COUNT(*) FROM ExamManagement.Question WHERE QuestionType = 'Open-Ended' AND CourseId = @CourseId;

IF (@MCQCount > @AvailableMCQ)
BEGIN
    PRINT 'Not enough Multiple Choice questions available';
    RETURN;
END;

IF (@TrueFalseCount > @AvailableTrueFalse)
BEGIN
    PRINT 'Not enough True/False questions available';
    RETURN;
END;

IF (@WrittenCount > @AvailableWritten)
BEGIN
    PRINT 'Not enough Written questions available';
    RETURN;
END;
```

CalcGradeOfExamForStudent is a stored procedure is responsible for calculating the total score of a student for a given exam. It supports **multiple question types**, including: Multiple Choice Questions (MCQ)-True/False Questions-Open-Ended (Essay) Questions

```sql
GO
CREATE OR ALTER PROCEDURE CalcGradeOfExamForStudent
    @ExamId INT,
    @StudentId INT,
    @UserName VARCHAR(50),
    @Password VARCHAR(40)
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @InstructorId INT, @CourseId INT, @TotalScore INT = 0, @KeywordCount INT, @MatchedKeywords INT, @QuestionMark INT;
    DECLARE @Keywords VARCHAR(MAX), @StudentAnswer VARCHAR(MAX);
    DECLARE @Keyword NVARCHAR(100);
    DECLARE @KeywordTable TABLE (Keyword NVARCHAR(100));

    -- التحقق من صحة بيانات المعلم
    SELECT @InstructorId = I.Id
    FROM UserManagement.Instructor I
    INNER JOIN UserManagement.Account A ON I.AccountId = A.Id
    WHERE A.UserName = @UserName AND A.Password = @Password;


    IF @InstructorId IS NULL
    BEGIN
        PRINT 'Invalid instructor credentials';
        RETURN;
    END;

    -- التحقق من أن المعلم لديه صلاحية على الامتحان
    SELECT @CourseId = CourseId
    FROM ExamManagement.Exam
    WHERE Id = @ExamId;

     IF NOT EXISTS (SELECT 1 FROM AcademicRecords.Course WHERE Id = @CourseId AND InstructorId = @InstructorId)
    BEGIN
        PRINT 'You are not the instructor of this course';
        RETURN;
    END;
```

```sql
-- التحقق من أن الطالب مسجل في الامتحان
IF NOT EXISTS (SELECT 1 FROM ExamManagement.StudentListExam WHERE StudentId = @StudentId AND ExamId = @ExamId)
BEGIN
    PRINT 'Student is not registered for this exam';
    RETURN;
END;

-- التحقق من أن الطالب قدم إجابات
IF NOT EXISTS (SELECT 1 FROM ExamManagement.Student_Answer_Details WHERE StudentId = @StudentId)
BEGIN
    PRINT 'No answers found for this student';
    RETURN;
END;


-- حساب درجة الاختيار من متعدد
SELECT @TotalScore = @TotalScore + EQ.Mark
FROM ExamManagement.Exam_Question EQ
INNER JOIN ExamManagement.Question Q ON Q.Id = EQ.QuestionId
INNER JOIN ExamManagement.Student_Answer_Details SAD ON SAD.Exam_QuestionId = EQ.Id AND SAD.StudentId = @StudentId
INNER JOIN ExamManagement.Answer_MCQ AM ON AM.Student_Answer_DetailsId = SAD.Id -- ربط إجابة MCQ بالطالب
INNER JOIN ExamManagement.Answer_Choice AC ON AC.Answer_MCQId = AM.Id -- ربط الاختيار بإجابة الطالب
INNER JOIN ExamManagement.Choices C ON C.Id = AC.ChoiceId -- ربط الاختيار بالسؤال
WHERE EQ.ExamId = @ExamId
AND Q.QuestionType = 'Multiple Choice'
AND C.IsTrue = 1; -- التأكد من صحة الإجابة


-- حساب درجة الصح والخطأ
SELECT @TotalScore = @TotalScore + EQ.Mark
FROM ExamManagement.Exam_Question EQ
INNER JOIN ExamManagement.Question Q ON Q.Id = EQ.QuestionId
INNER JOIN ExamManagement.True_False TF ON Q.Id = TF.QuestionId
INNER JOIN ExamManagement.Student_Answer_Details SAD ON SAD.Exam_QuestionId = EQ.Id AND SAD.StudentId = @StudentId
INNER JOIN ExamManagement.Answer_True_False ATF ON ATF.Student_Answer_DetailsId = SAD.Id
WHERE EQ.ExamId = @ExamId AND Q.QuestionType = 'True/False' AND ATF.StudentanswerOfTrueFalse = TF.Answer;
```

Add Student to Exam is a stored procedure that responsible for **registering a student for an exam** while ensuring **data integrity and validation**. It prevents duplicate registrations and ensures that both the **exam and student exist** before adding the record.

```sql
  GO
CREATE OR ALTER PROCEDURE AddStudentsToExam
    @ExamId INT,
    @StudentId INT
AS
BEGIN
    SET NOCOUNT ON;

    IF NOT EXISTS (SELECT 1 FROM ExamManagement.Exam WHERE Id = @ExamId)
    BEGIN
        PRINT 'Exam does not exist';
        RETURN;
    END;

    IF NOT EXISTS (SELECT 1 FROM AcademicRecords.Student WHERE Id = @StudentId)
    BEGIN
        PRINT 'Student does not exist';
        RETURN;
    END;

    IF EXISTS (SELECT 1 FROM ExamManagement.StudentListExam WHERE ExamId = @ExamId AND StudentId = @StudentId)
    BEGIN
        PRINT 'Student is already registered for this exam';
        RETURN;
    END;

    INSERT INTO ExamManagement.StudentListExam (ExamId, StudentId)
    VALUES (@ExamId, @StudentId);

    PRINT 'Student added to the exam successfully';
END;
GO
```