ITI Examination System

Database Project

Presented By:

Abdullah Osama Abdelhafeez

Eslam Elsayed Mohammed

Omar Saad Ibrahim

Mariam Mohammed Saied Elkomi

Mohammed Masoud Raafat

Supervised By:

Eng : Ramy Nagi

1. Project Overview

The ITI Examination System is designed to efficiently manage students, courses, instructors, and exams. It facilitates exam generation, question management, student responses, and report generation. This system aims to streamline the examination process and improve accuracy in grading.

1.1 Objectives

- Automate the examination process for institutions.
- Ensure accurate storage and retrieval of student performance data.
- Provide detailed reports on student and instructor activities.
- Enhance security and integrity in examination management.

1.2 Technologies Used

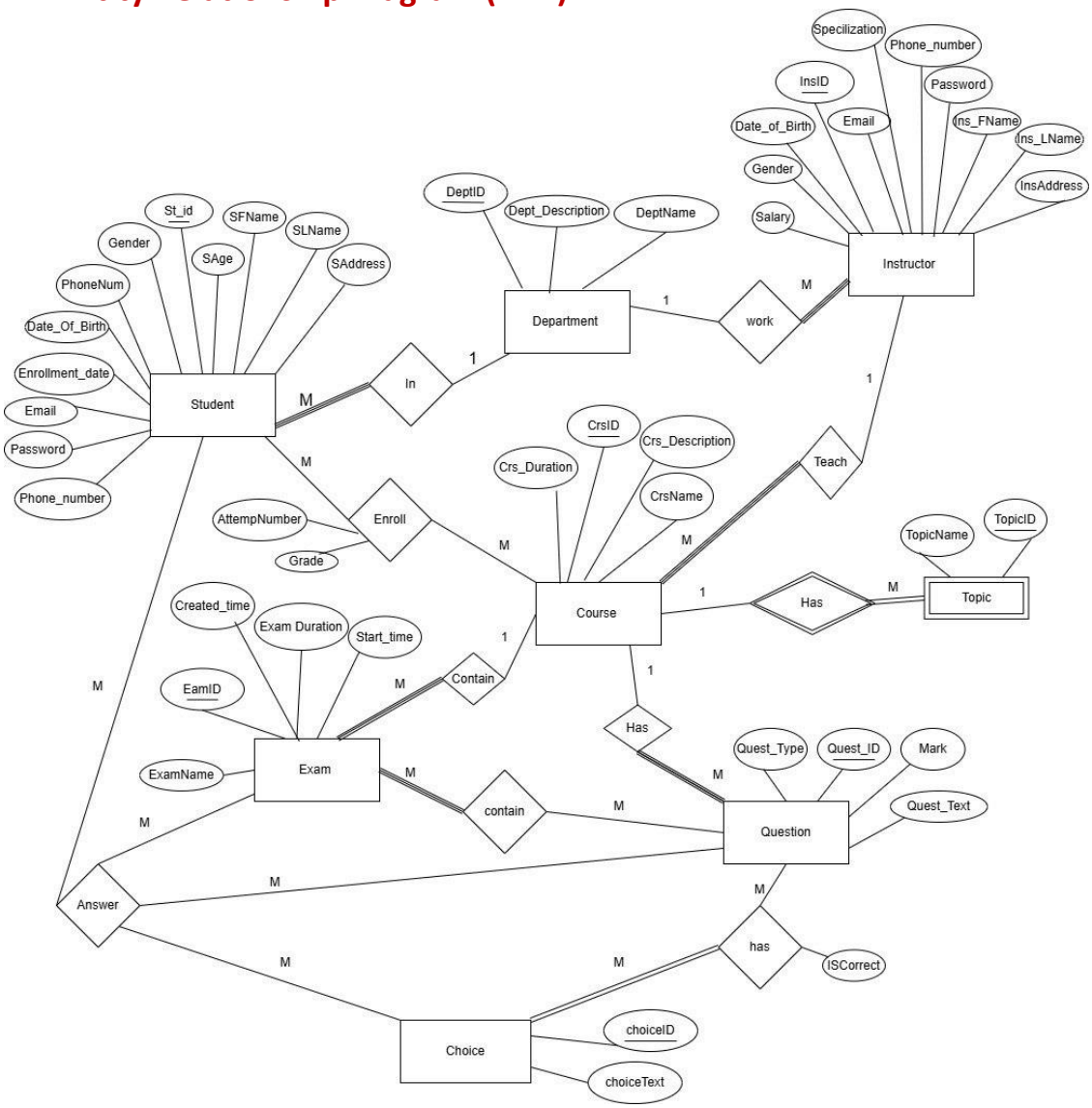
- **Database Management System**: Microsoft SQL Server
- **Backend**: SQL Stored Procedures
- **Frontend**: Web-based UI (optional integration)
- **Programming Languages**: SQL, C# (if applicable)
- **Reporting Tool**: SQL Reports

2. Database Schema

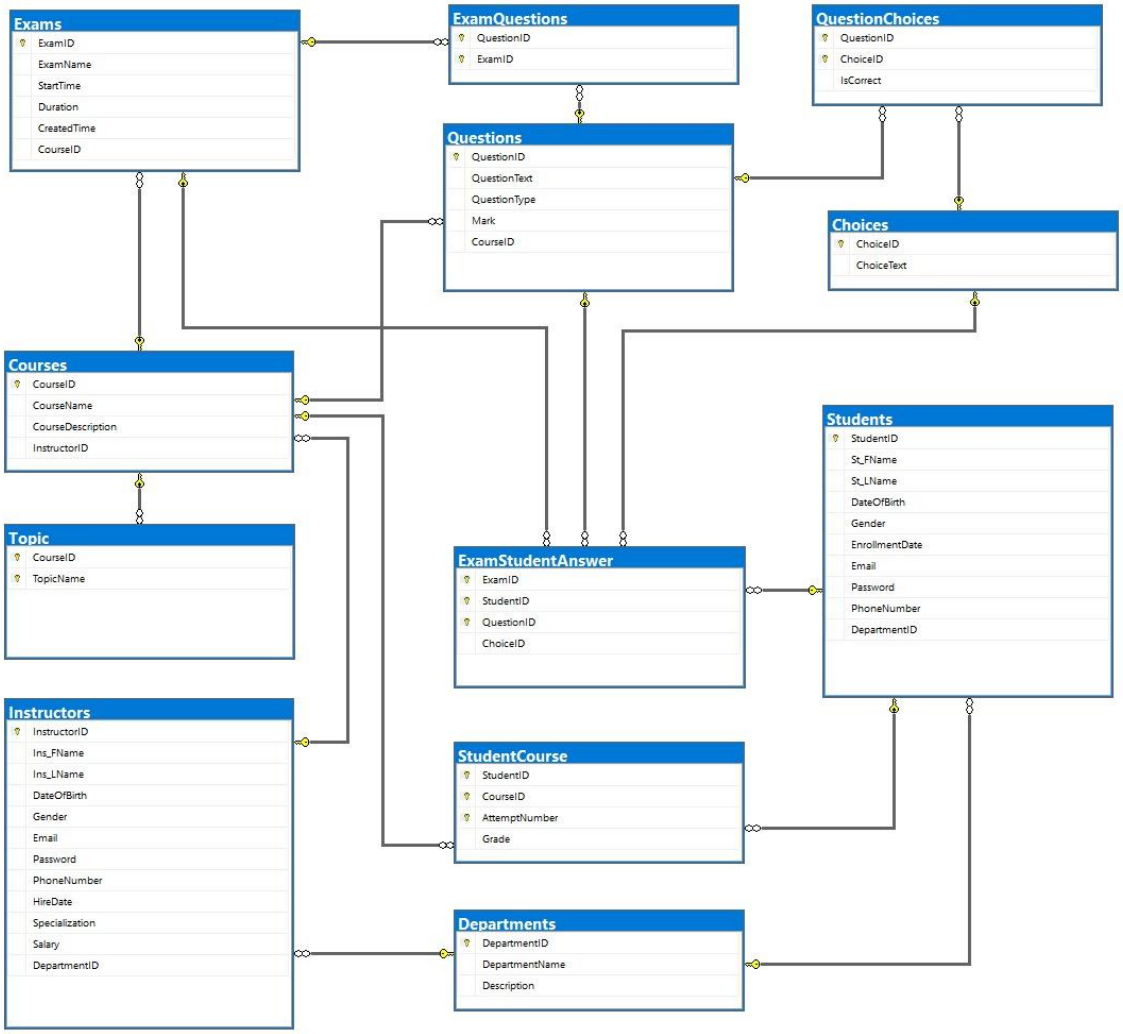
The database consists of multiple tables, including:

- **Student**: Stores student details.
- **Instructor**: Stores instructor details.
- **Course**: Manages available courses.
- **Exam**: Contains exam details.
- **Questions**: Stores exam questions.
- **Choices**: Holds multiple-choice options.
- **Student Answers**: Tracks student responses.
- **Department**: Manages department information.
- **Exam Questions**: Links exams with questions.

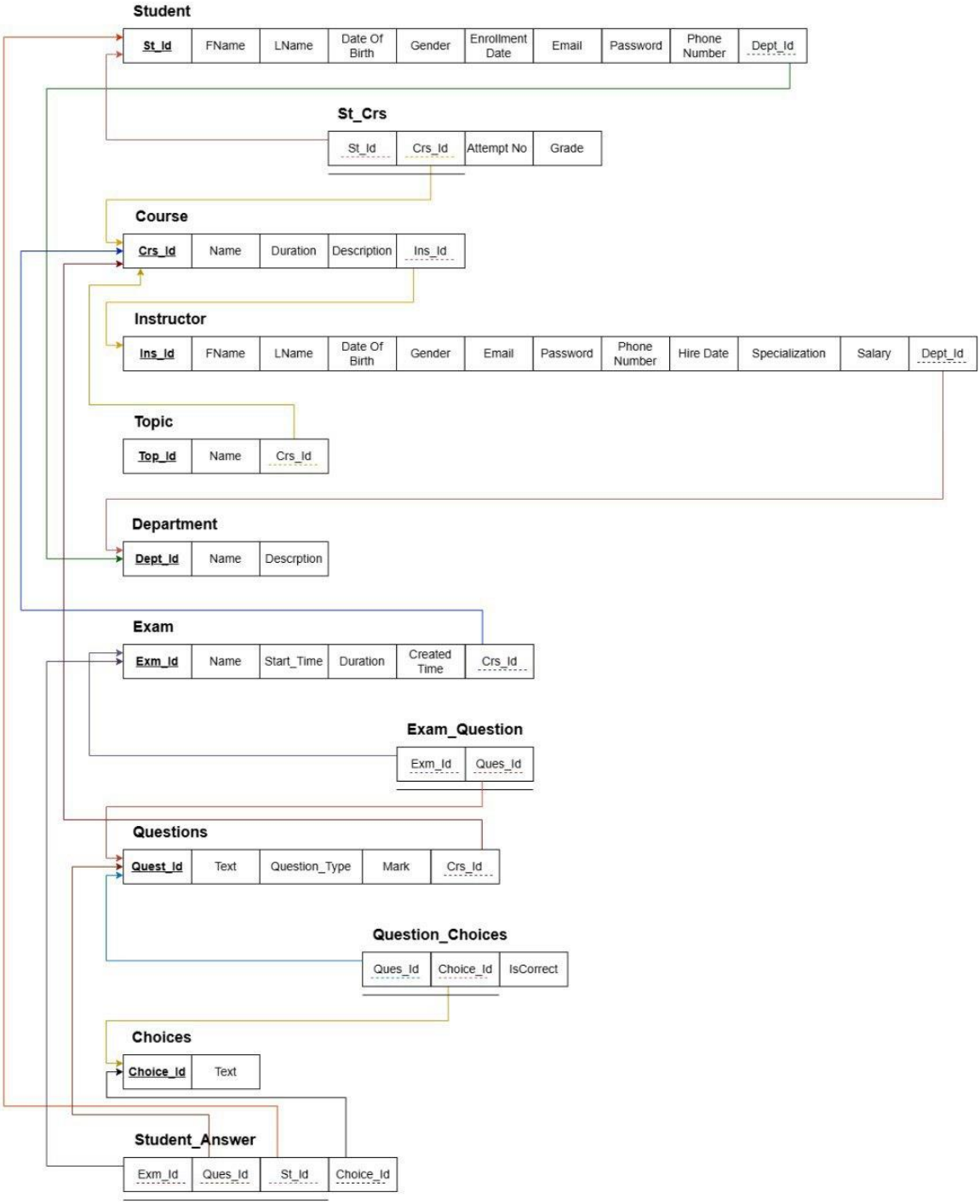
2.1 Entity Relationship Diagram (ERD)



2.2 Database Diagram



2.2 Database Mapping



Stored Procedures

1. **Student Table**
   - SP_AddNewStudent
   - SP_DeleteStudent
   - SP_UpdateStudent
   - SP_SelectStudent
   - SP_SelectAllStudentsWithDepartment

2. **Instructors Table**
   - SP_AddNewInstructor
   - SP_DeleteInstructor
   - SP_UpdateInstructor
   - SP_SelectInstructor
   - SP_SelectAllInstructorsWithDepartment

3. **Departments Table**
   - SP_AddNewDepartment
   - SP_DeleteDepartment
   - SP_UpdateDepartment
   - SP_SelectDepartment
   - SP_SelectAllDepartments

4. **Courses Table**
   - SP_AddNewCourse
   - SP_DeleteCourse
   - SP_UpdateCourse
   - SP_SelectCourse
   - SP_SelectAllCoursesWithInstructorAndDepartment
   -

5. **Topic Table**
   - SP_AddNewTopic
   - SP_DeleteTopic
   - SP_UpdateTopic
   - SP_SelectTopic

6. **Questions Table**
   - SP_AddNewQuestion
   - SP_DeleteQuestion
   - SP_UpdateQuestion
   - SP_SelectQuestion
7. **Choices Table**
   - SP_AddNewChoice
   - SP_DeleteChoice
   - SP_UpdateChoice
   - SP_SelectChoice
   - SP_GetChoicesByQuestionId
8. **StudentCourse Table**
   - SP_EnrollStudentInCourse
   - SP_DeleteStudentCourse
   - SP_UpdateStudentCourse
   - SP_SelectStudentCourse
9. **QuestionChoices Table**
   - SP_DeleteQuestionChoice
   - SP_UpdateQuestionChoice
   - SP_SelectQuestionChoice
10. **Exams Table**
    - SP_CreateExam
    - SP_DeleteExam
    - SP_UpdateExam
    - SP_SelectExam
11. **ExamStudentAnswer Table**
    - SP_DeleteExamStudentAnswer
    - SP_SelectExamStudentAnswer

12. ExamQuestions Table

- SP_DeleteExamQuestion
- SP_SelectExamQuestion

13. Additional Procedures

- **SP_CheckExamEligibility**

```
-- SP_CheckExamEligibility
CREATE OR ALTER PROCEDURE SP_CheckExamEligibility
    @StudentId INT,
    @CourseId INT
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM StudentCourse WHERE StudentID = @StudentId AND CourseID = @CourseId)
    BEGIN
        SELECT 'Student is not enrolled in this course.' AS Result;
        RETURN;
    END

    DECLARE @AttemptNumber INT, @Grade NVARCHAR(2);
    SELECT
        @AttemptNumber = AttemptNumber,
        @Grade = Grade
    FROM StudentCourse
    WHERE StudentID = @StudentId AND CourseID = @CourseId;

    IF @AttemptNumber = 2
    BEGIN
        SELECT 'Student has exhausted all attempts for this course.' AS Result;
    END
    ELSE IF @AttemptNumber = 1 AND @Grade IN ('A', 'B', 'C', 'D')
    BEGIN
        SELECT 'Student has already achieved a passing grade. Cannot take the exam again.' AS Result;
    END
    ELSE
    BEGIN
        SELECT 'Student is eligible to take the exam.' AS Result;
    END
END;
```

- **SP_AddStudentAnswers**

```sql
-- SP_AddStudentAnswers
CREATE OR ALTER PROCEDURE SP_AddStudentAnswers
    @ExamId INT,
    @StudentId INT,
    @Answers StudentAnswersType READONLY
AS
BEGIN
    DECLARE @CourseId INT;
    SELECT @CourseId = CourseID FROM Exams WHERE ExamID = @ExamId;

    IF NOT EXISTS (SELECT 1 FROM StudentCourse WHERE StudentID = @StudentId AND CourseID = @CourseId)
    BEGIN
        SELECT 'Student is not enrolled in this course.' AS Result;
        RETURN;
    END

    INSERT INTO ExamStudentAnswer (ExamID, StudentID, QuestionID, ChoiceID)
    SELECT @ExamId, @StudentId, QuestionId, CorrectChoiceId
    FROM @Answers A
    WHERE NOT EXISTS (
        SELECT 1
        FROM ExamStudentAnswer ESA
        WHERE ESA.ExamID = @ExamId
          AND ESA.StudentID = @StudentId
          AND ESA.QuestionID = A.QuestionId
    );

    SELECT 'Answers added successfully.' AS Result;
END;
```

- **SP_StartExam**

```sql
-- SP_StartExam
CREATE OR ALTER PROCEDURE SP_StartExam
    @StudentId INT,
    @CourseId INT
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM StudentCourse WHERE StudentID = @StudentId AND CourseID = @CourseId)
    BEGIN
        SELECT 'Student is not enrolled in this course.' AS Result;
        RETURN;
    END

    UPDATE StudentCourse
    SET AttemptNumber = AttemptNumber + 1
    WHERE StudentID = @StudentId AND CourseID = @CourseId;

    SELECT 'Exam started successfully. AttemptNumber has been incremented.' AS Result;
END;

CREATE TYPE StudentAnswersType AS TABLE (
    QuestionId INT,
    CorrectChoiceId INT
);
```

- **SP_CalculateGrade**

```sql
-- SP_CalculateGrade
CREATE OR ALTER PROCEDURE SP_CalculateGrade
    @ExamId INT,
    @StudentId INT
AS
BEGIN
    DECLARE @TotalMarks FLOAT, @StudentMarks FLOAT;


    SELECT @TotalMarks = SUM(Mark)
    FROM Questions Q
    JOIN ExamQuestions EQ ON Q.QuestionID = EQ.QuestionID
    WHERE EQ.ExamID = @ExamId;


    SELECT @StudentMarks = SUM(Q.Mark)
    FROM Questions Q
    JOIN ExamStudentAnswer ESA ON Q.QuestionID = ESA.QuestionID
    JOIN QuestionChoices QC ON Q.QuestionID = QC.QuestionID AND ESA.ChoiceID = QC.ChoiceID
    WHERE ESA.ExamID = @ExamId AND ESA.StudentID = @StudentId AND QC.IsCorrect = 1;


    DECLARE @Percentage FLOAT = (@StudentMarks / @TotalMarks) * 100;

    DECLARE @Grade NVARCHAR(2);
    SET @Grade = CASE
        WHEN @Percentage >= 90 THEN 'A'
        WHEN @Percentage >= 80 THEN 'B'
        WHEN @Percentage >= 70 THEN 'C'
        WHEN @Percentage >= 60 THEN 'D'
        ELSE 'F'
    END;

    DECLARE @CourseId INT;
    SELECT @CourseId = CourseID FROM Exams WHERE ExamID = @ExamId;

    UPDATE StudentCourse
    SET Grade = @Grade
    WHERE StudentID = @StudentId AND CourseID = @CourseId;

    SELECT
        'Exam completed successfully.' AS Result,
        @Percentage AS Percentage,
        @Grade AS Grade;
END;
```

4. Reports

1. Report that returns the students information according to Department No parameter.

Student ID	Full Name	Date Of Birth	Gender	Enrollment Date	Phone Number	Email
1	Eslam Elsayed	1/1/2000 12:00:00 AM	Male	9/1/2023 12:00:00 AM	111-111-1111	eslam.elsayed@student.edu
2	Abdalla Osama	2/2/2000 12:00:00 AM	Male	9/1/2023 12:00:00 AM	222-222-2222	abdalla.osama@student.edu
3	Mariam Elkomi	3/3/2000 12:00:00 AM	Female	9/1/2023 12:00:00 AM	333-333-3333	mariam.elkomi@student.edu
4	Omar Saad	4/4/2000 12:00:00 AM	Male	9/1/2023 12:00:00 AM	444-444-4444	omar.saad@student.edu
5	Mohamed Masoud	5/5/2000 12:00:00 AM	Male	9/1/2023 12:00:00 AM	555-555-5555	mohamed.masoud@student.edu

2. Report that takes the student ID and returns the grades of the student in all courses.

3. Report that takes the instructor ID and returns the name of the courses that he teaches and the number of student per course

Instructor ID	Course Name	Student Count
1	C# OOP	5
1	Entity Framework	5

4. Report that takes course ID and returns its topics

Course ID	Topic Name
2	API Development
2	Authentication
2	MVC Pattern
2	Razor Pages

Course ID	Topic Name
3	Code First
3	Database First
3	LINQ
3	Migrations

Course ID	Topic Name
1	Abstraction
1	Encapsulation
1	Inheritance
1	Intro
1	Polymorphism

5. Report that takes exam number and returns Questions in it

5	Which keyword is used to prevent a class from being inherited?	MCQ	2
5	Which method is called when an object is created in C#?	MCQ	2
5	What is the purpose of the "abstract" keyword in C#?	MCQ	2

Exam ID	Question Text	Question Type	Mark
5	C# is a strongly typed language.	TF	1
5	Abstraction is the process of hiding implementation details.	TF	1
5	Encapsulation is the process of wrapping data and methods into a single unit.	TF	1
5	An abstract class cannot be instantiated.	TF	1
5	Interfaces can contain method implementations.	TF	1
5	C# supports multiple inheritance.	TF	1
5	Which keyword is used to prevent a class from being inherited?	MCQ	2
5	Which method is called when an object is created in C#?	MCQ	2

6. Report that takes exam number and the student ID then returns the Questions in this exam with the student answers.

Exam ID	Question Text	Question Type	Student Answer	Mark
5	C# is a strongly typed language.	TF	True	1
5	Abstraction is the process of hiding implementation details.	TF	True	1
5	Encapsulation is the process of wrapping data and methods into a single unit.	TF	True	1
5	A class can inherit from multiple classes in C#.	TF	False	1
5	An abstract class cannot be instantiated.	TF	True	1
5	Interfaces can contain method implementations.	TF	False	1
5	C# supports multiple inheritance.	TF	False	1
5	Which keyword is used to prevent a class from being inherited?	MCQ	sealed	2
5	Which method is called when an object is created in C#?	MCQ	Constructor	2
5	What is the purpose of the "abstract" keyword in C#?	MCQ	To define a class that cannot be instantiated	2

5. Setup & Usage Instructions

5.1 Database Setup

1. Install Microsoft SQL Server.
2. Execute the provided SQL scripts to create tables and stored procedures.
3. Insert initial data (departments, instructors, courses, students, etc.).

5.2 System Usage

1. **Exam Creation**: Use SP_CreateExam to generate new exams.
2. **Adding Questions**: Use SP_AddNewQuestion to insert questions.
3. **Student Enrollment**: Enroll students using SP_EnrollStudentInCourse.
4. **Exam Execution**: Students can start exams using SP_StartExam.
5. **Grading**: Calculate student grades using SP_CalculateGrade.

5.3 Security & Maintenance

- Regular database backups should be performed.
- User roles and permissions should be managed carefully to prevent unauthorized access.
- Stored procedures should be optimized for performance.

6. Conclusion

The ITI Examination System offers an efficient and secure way to manage exams, student records, and instructor activities. The system ensures quick data retrieval, scalability, and enhanced exam security by leveraging stored procedures. Future enhancements may include an intuitive web interface and AI-based analytics for performance evaluation.

7. Future Enhancements

- Web-based frontend for better accessibility.
- AI-driven exam analytics for performance evaluation.
- Integration with Learning Management Systems (LMS).
- Mobile-friendly examination interface for students.