ITI Examination System

Project’s team members:

1. Nour Ahmed Shalbi
2. Nourhan Gamil Eltalawy
3. Mohamed Khaled Mohamed
4. Hisham Essam Abdelfatah Mohamed
5. Mohamed Salah Hussein

INDEX

Chapter 1

Introduction

Chapter 1: Introduction

In this chapter, we introduce the proposed ITI Examination Management System and cover key aspects, including project objectives, and the scope of implementation. This system is designed as an innovative educational technology solution aimed at streamlining assessment processes, improving fairness and efficiency, and supporting instructors, students, and administrators through automated exam creation, secure delivery, real-time monitoring, and instant result generation.

1.1 Introduction

The ITI Examination System is an innovative desktop-based examination management platform designed to enhance the assessment process at the Information Technology Institute (ITI). The system interacts directly with a SQL Server database to manage question banks, generate dynamic exams, automate correction processes, and store examination data securely.

ITI Examination System aims to modernize traditional examination workflows by providing a reliable and efficient solution for exam creation, scheduling, evaluation, and reporting. By integrating automated grading mechanisms and structured data management, the system ensures accuracy, fairness, and consistency in assessments. In addition, Power BI integration enables the generation of detailed analytical reports that support informed academic and administrative decision-making.

1.2 Problem Statement

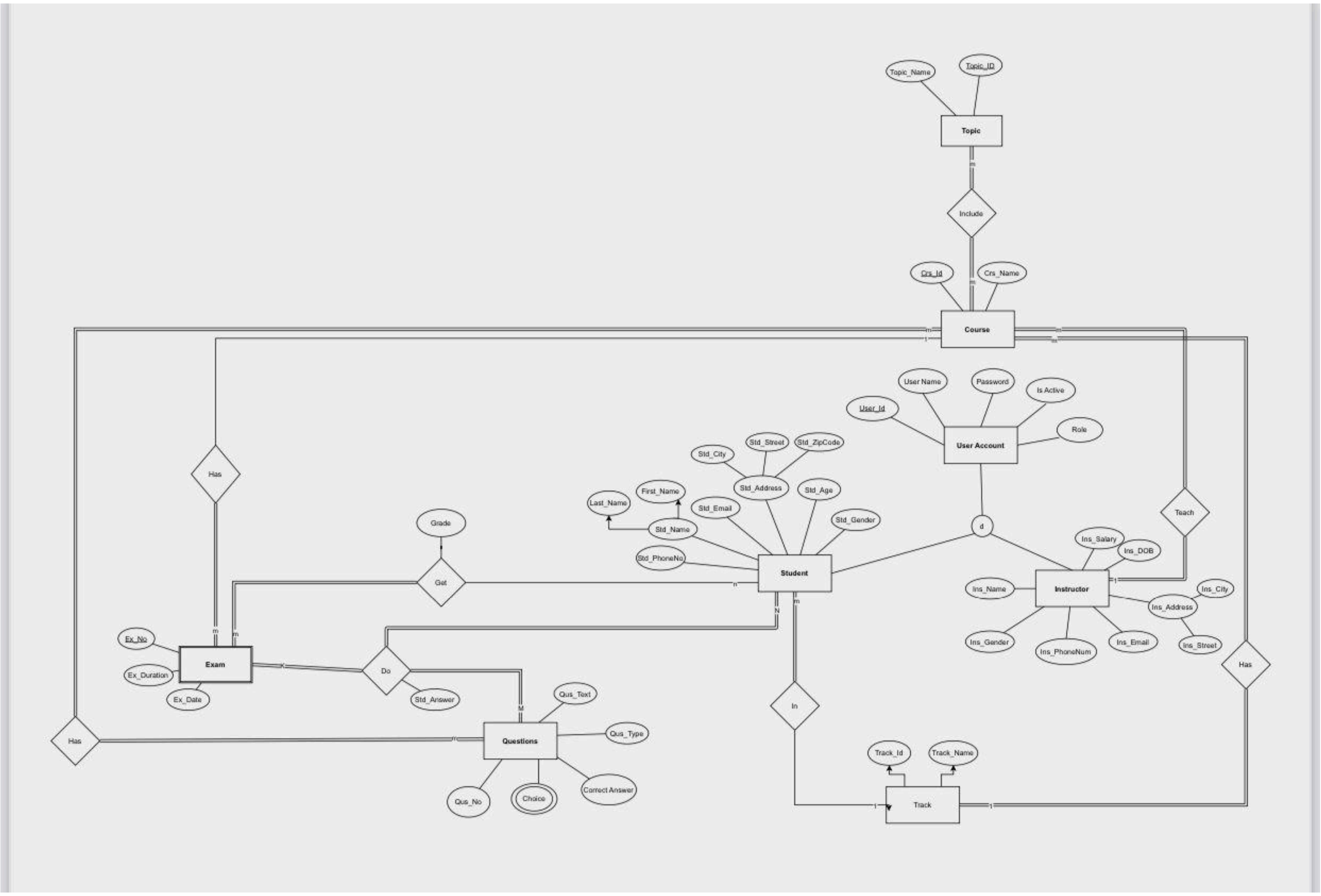
At ITI, the need for a centralized and automated examination system has become essential to manage large numbers of students, ensure exam integrity, and provide timely feedback. The absence of dynamic exam generation, automated correction, and advanced reporting tools limits the ability to evaluate student performance effectively.
The ITI Examination System addresses these challenges by offering a desktop-based solution that automates exams, correction, and reporting through SQL Server and PowerBI.

Chapter 2

System Analysis And Stored Procedures

Chapter 2: System Analysis Stored Procedures Details

2.1 ERD(Entity RealtionShip Diagram)



2.2 Mapping

```
User_Account(
  User_Id PK,
  User_Name,
  Password,
  Role,        -- e.g. 'student' or 'instructor'
  Is_Active
)
Student(
  User_Id PK, FK -> User_Account(User_Id),
  First_Name,
  Last_Name,
  Std_PhoneNo,
  Std_Email,
  Std_Gender,
  Std_Age,
  Std_City,
  Std_Street,
  Std_ZipCode,
  Track_Id FK -> Track(Track_Id)
)
Instructor(
  User_Id PK, FK -> User_Account(User_Id),
  Ins_Name,
  Ins_Gender,
  Ins_PhoneNum,
  Ins_Email,
  Ins_Salary,
  Ins_DOB,
  Ins_City,
  Ins_Street
)
Track(
  Track_Id PK,
  Track_Name
)
Course(
  Crs_Id PK,
  Crs_Name,
  Instructor_UserId FK -> Instructor(User_Id)   -- the instructor who teaches the course
)
```

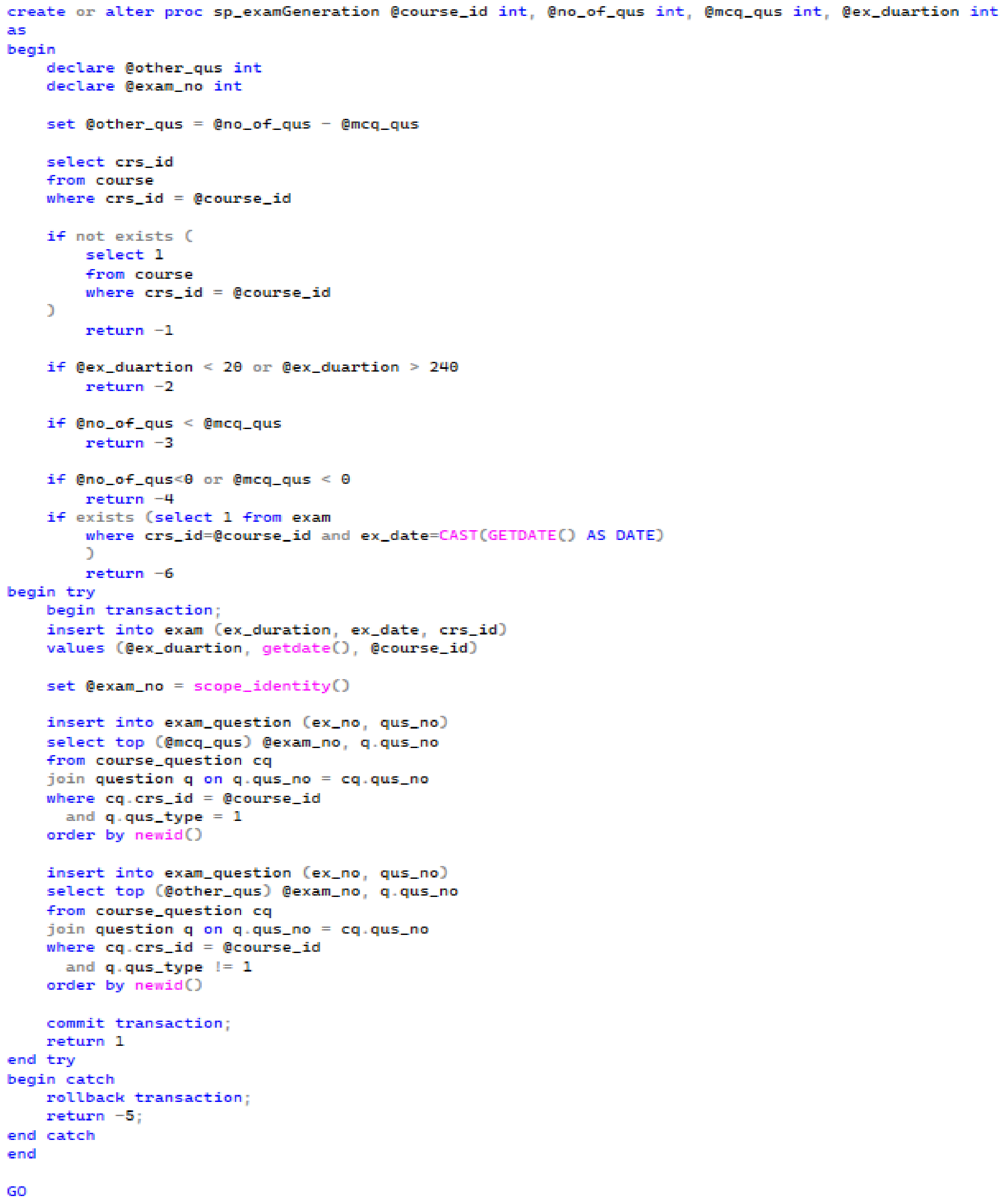
```
Track_Course(
  Track_Id PK, FK -> Track(Track_Id),
  Crs_Id  PK, FK -> Course(Crs_Id)
)
Topic(
  Topic_Id PK,
  Topic_Name
)

Crs_Topic(
  Crs_Id  PK, FK -> Course(Crs_Id),
  Topic_Id PK, FK -> Topic(Topic_Id)
)
Student_Course(
  User_Id PK, FK -> Student(User_Id),   -- student (from User_Account)
  Crs_Id  PK, FK -> Course(Crs_Id),
  Attend_Status                         -- e.g. 'enrolled', 'dropped', 'completed'
)
Exam(
  Ex_No PK,
  Ex_Duration,
  Ex_Date,
  Crs_Id FK -> Course(Crs_Id)
)
Question(
  Qus_No PK,
  Qus_Text,
  Qus_Type,         -- e.g. 'MCQ', 'TF', 'Essay'
  Correct_Answer    -- (for MCQ may be a choice id or canonical text)
)
Choice(
  Choice_Id PK,
  Qus_No FK -> Question(Qus_No),
  Choice_Text
)
Exam_Question(
  Ex_No PK, FK -> Exam(Ex_No),
  Qus_No PK, FK -> Question(Qus_No)
)
Student_Exam_Grade(
  Ex_No PK, FK -> Exam(Ex_No),
  User_Id PK, FK -> Student(User_Id),
  Grade
)
```

Student_Exam_Ans(

```
ID PK,                  -- surrogate key for each answer record
Ex_No  FK -> Exam(Ex_No),
User_Id FK -> Student(User_Id),
Qus_No  FK -> Question(Qus_No),
Choice_Id FK -> Choice(Choice_Id),
Answer_Text
)
```

2.3 Stored Procedure

2.3.1 Exam Generation



```
create or alter proc sp_examGeneration @course_id int, @no_of_qus int, @mcq_qus int, @ex_duartion int
as
begin
    declare @other_qus int
    declare @exam_no int

    set @other_qus = @no_of_qus - @mcq_qus

    select crs_id
    from course
    where crs_id = @course_id

    if not exists (
        select 1
        from course
        where crs_id = @course_id
    )
        return -1

    if @ex_duartion < 20 or @ex_duartion > 240
        return -2

    if @no_of_qus < @mcq_qus
        return -3

    if @no_of_qus<0 or @mcq_qus < 0
        return -4
    if exists (select 1 from exam
        where crs_id=@course_id and ex_date=CAST(GETDATE() AS DATE)
        )
        return -6
begin try
    begin transaction;
    insert into exam (ex_duration, ex_date, crs_id)
    values (@ex_duartion, getdate(), @course_id)

    set @exam_no = scope_identity()

    insert into exam_question (ex_no, qus_no)
    select top (@mcq_qus) @exam_no, q.qus_no
    from course_question cq
    join question q on q.qus_no = cq.qus_no
    where cq.crs_id = @course_id
      and q.qus_type = 1
    order by newid()

    insert into exam_question (ex_no, qus_no)
    select top (@other_qus) @exam_no, q.qus_no
    from course_question cq
    join question q on q.qus_no = cq.qus_no
    where cq.crs_id = @course_id
      and q.qus_type != 1
    order by newid()

    commit transaction;
    return 1
end try
begin catch
    rollback transaction;
    return -5;
end catch
end

GO
```

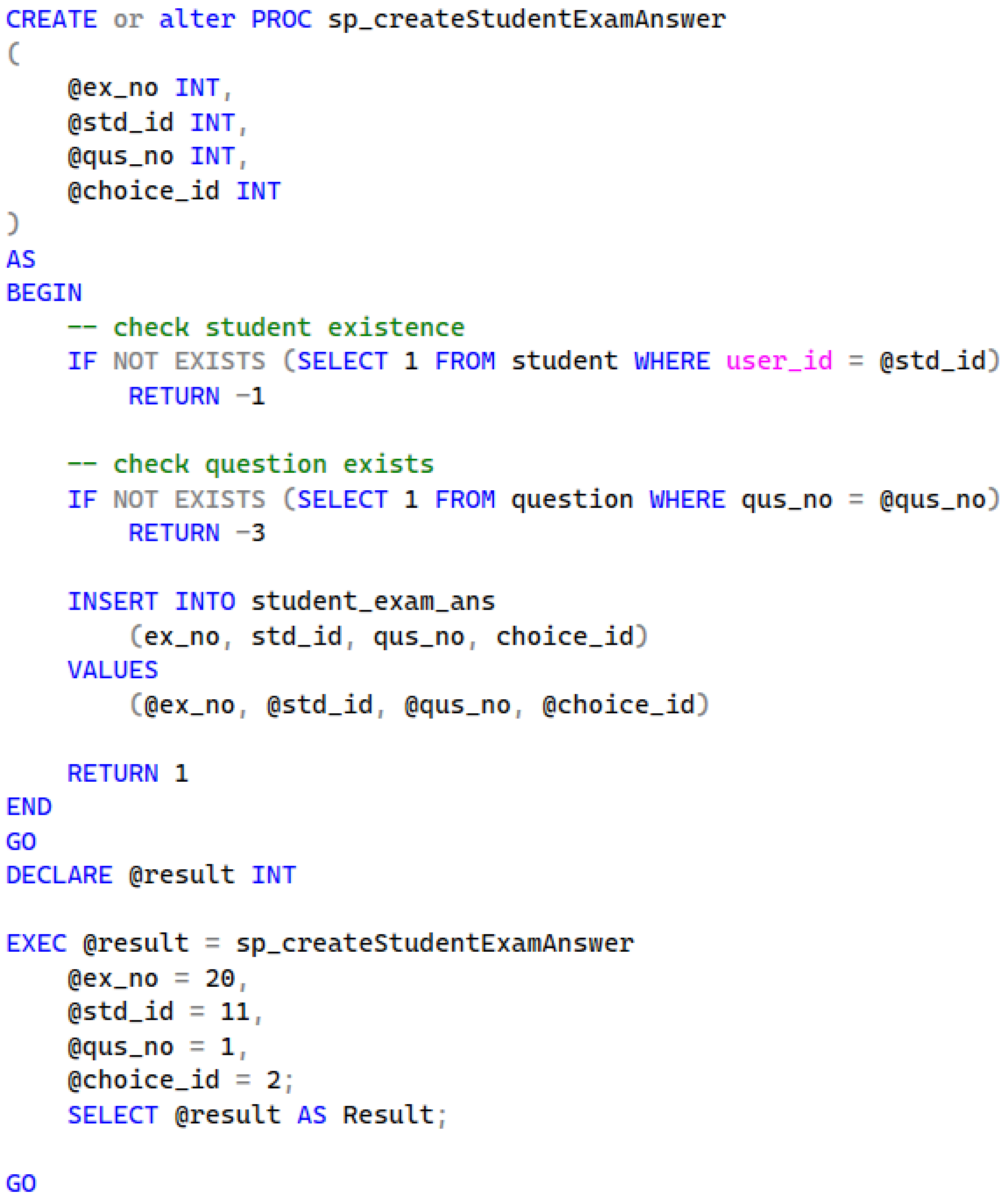
This stored procedure spExamGeneration generates a new exam for a given course automatically. It takes the following parameters:

- @course_id (INT) – The ID of the course for which the exam is being created
- @no_of_qus (INT) – Total number of questions in the exam
- @mcq (INT) – Number of MCQ (multiple-choice) questions
- @exduration (INT) – Exam duration in minutes

What the stored procedure does (step-by-step logic):

1. **Input Validation & Early Returns**
    - Calculates number of non-MCQ questions: @other_qus = @no_of_qus - @mcq
    - Returns error codes if conditions are not met:
        - -1 → Course does not exist
        - -2 → Duration is less than 20 min or more than 240 min
        - -3 → Total questions < number of MCQs
        - -4 → Total questions or MCQs are zero or negative
2. **Duplicate Exam Check**
    - Checks if an exam already exists for this course on the current date
    - If yes → returns -6 (prevents duplicate exams on the same day)
3. **Exam Creation (Main Logic – inside TRY block)**
    - Starts a transaction
    - Inserts a new row into exam table:
        - Duration, current date/time, course ID
    - Captures the newly created ex_no using SCOPE_IDENTITY()
4. **Question Selection & Assignment**
    - Inserts **MCQ questions** into exam_question:
        - Selects top @mcq questions of type = 1 (MCQ)
        - From course_question + question for the given course
        - Ordered randomly (NEWID())
    - Inserts **non-MCQ (other) questions** into exam_question:
        - Selects top @other_qus questions of type ≠ 1
        - Same random ordering
5. **Finalization**
    - Commits the transaction if everything succeeds → returns 1 (success)
    - If any error occurs → catches it, rolls back the transaction, returns -5 (failure)

2.3.2 Add Student Exam Answer



```
CREATE or alter PROC sp_createStudentExamAnswer
(
    @ex_no INT,
    @std_id INT,
    @qus_no INT,
    @choice_id INT
)
AS
BEGIN
    -- check student existence
    IF NOT EXISTS (SELECT 1 FROM student WHERE user_id = @std_id)
        RETURN -1

    -- check question exists
    IF NOT EXISTS (SELECT 1 FROM question WHERE qus_no = @qus_no)
        RETURN -3

    INSERT INTO student_exam_ans
        (ex_no, std_id, qus_no, choice_id)
    VALUES
        (@ex_no, @std_id, @qus_no, @choice_id)

    RETURN 1
END
GO
DECLARE @result INT

EXEC @result = sp_createStudentExamAnswer
    @ex_no = 20,
    @std_id = 11,
    @qus_no = 1,
    @choice_id = 2;
    SELECT @result AS Result;

GO
```

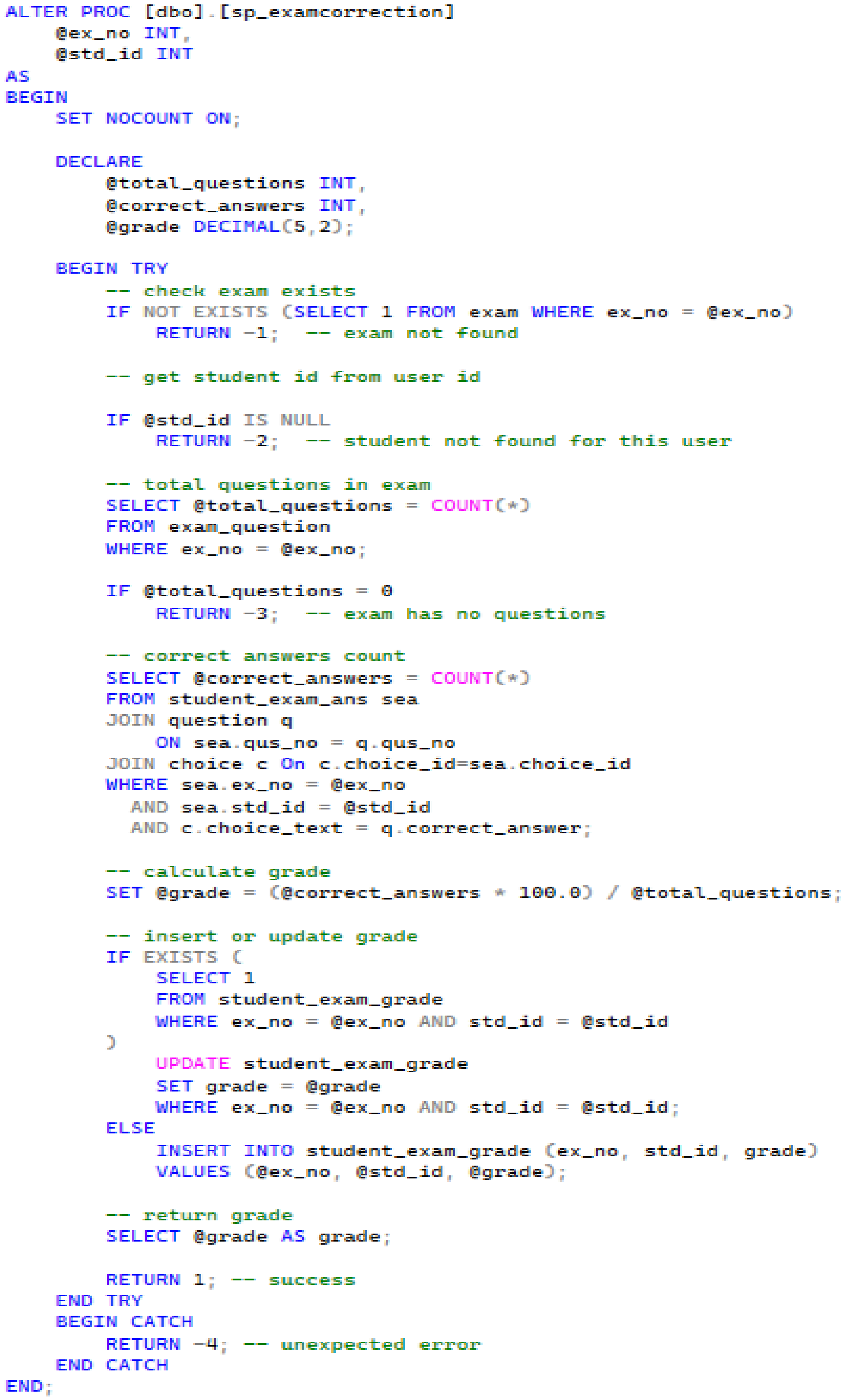
This stored procedure sp_createStudentExamAnswer records a student's answer for a specific question in an exam. It takes the following parameters:

- **@ex_no (INT) – Exam number**
- **@std_id (INT) – Student ID (from the student table)**
- **@qus_no (INT) – Question number**
- **@choice_id (INT) – The ID of the chosen answer (for MCQ questions)**

What the stored procedure does (step-by-step logic):

1. **Validation Checks**
    - **Checks if the student exists in the student table (using user_id = @std_id) → If not found → returns -1**
    - **Checks if the question exists in the question table → If not found → returns -3**
2. **Record the Answer**
    - **Inserts a new row into the student_exam_ans table with:**
        - **ex_no**
        - **std_id**
        - **qus_no**
        - **choice_id**
3. **Result**
    - **If the insert succeeds → returns 1 (success)**

2.3.3 Exam Correction



```
ALTER PROC [dbo].[sp_examcorrection]
    @ex_no INT,
    @std_id INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE
        @total_questions INT,
        @correct_answers INT,
        @grade DECIMAL(5,2);

    BEGIN TRY
        -- check exam exists
        IF NOT EXISTS (SELECT 1 FROM exam WHERE ex_no = @ex_no)
            RETURN -1;  -- exam not found

        -- get student id from user id

        IF @std_id IS NULL
            RETURN -2;  -- student not found for this user

        -- total questions in exam
        SELECT @total_questions = COUNT(*)
        FROM exam_question
        WHERE ex_no = @ex_no;

        IF @total_questions = 0
            RETURN -3;  -- exam has no questions

        -- correct answers count
        SELECT @correct_answers = COUNT(*)
        FROM student_exam_ans sea
        JOIN question q
            ON sea.qus_no = q.qus_no
        JOIN choice c On c.choice_id=sea.choice_id
        WHERE sea.ex_no = @ex_no
          AND sea.std_id = @std_id
          AND c.choice_text = q.correct_answer;

        -- calculate grade
        SET @grade = (@correct_answers * 100.0) / @total_questions;

        -- insert or update grade
        IF EXISTS (
            SELECT 1
            FROM student_exam_grade
            WHERE ex_no = @ex_no AND std_id = @std_id
        )
            UPDATE student_exam_grade
            SET grade = @grade
            WHERE ex_no = @ex_no AND std_id = @std_id;
        ELSE
            INSERT INTO student_exam_grade (ex_no, std_id, grade)
            VALUES (@ex_no, @std_id, @grade);

        -- return grade
        SELECT @grade AS grade;

        RETURN 1; -- success
    END TRY
    BEGIN CATCH
        RETURN -4; -- unexpected error
    END CATCH
END;
```

This stored procedure [dbo].[sp_examcorrection] automatically corrects a student's exam and calculates their grade. It takes the following parameters:

- @ex_no (INT) – Exam number
- @std_id (INT) – Student ID

What the stored procedure does (step-by-step logic):

1. **Validation Checks**
    - Checks if the exam exists in the exam table → If not found → returns -1
    - Checks if the student ID is provided (not null) → If null → returns -2
    - Counts total questions in the exam (exam_question table) → If zero questions → returns -3
2. **Grade Calculation**
    - Counts the number of **correct answers** by joining:
        - student_exam_ans (student's submitted answers)
        - question (to get correct answer)
        - choice (to match student's choice with correct one) → Only counts cases where student's chosen choice_text matches the question's correct_answer
3. **Grade Computation**
    - Calculates grade as: (correct_answers * 100.0) / total_questions → Result stored as decimal with 2 places (e.g., 85.50)
4. **Save or Update Grade**
    - Checks if a grade already exists in student_exam_grade for this exam + student
        - If exists → **updates** the existing record with the new grade
        - If not exists → **inserts** a new record with exam number, student ID, and calculated grade
5. **Result**
    - Returns the final calculated grade (as a SELECT statement)
    - On success → returns the grade value
    - On error → returns -4 (unexpected error during execution)

Chapter 3

System Design

Chapter 3: System Design

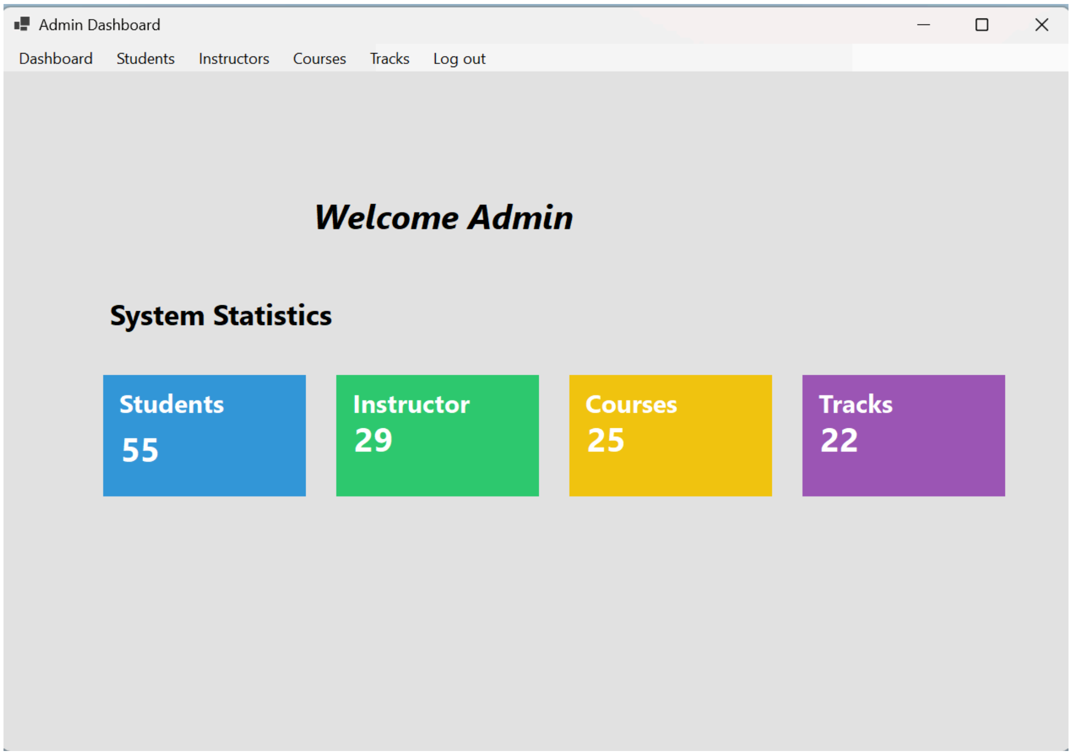
3.1 Application

3.1.1 Admin DashBoard

1️⃣Login Screen

Admin enters:

  - **Username**
  - **Password**
- Click **Login**
- ✔ If credentials are correct → go to **Admin Dashboard**
- ❌ If wrong → show error message



Welcome Admin

System Statistics

Students
55

Instructor
29

Courses
25

Tracks
22

2️⃣AdminDashboard

1- App Data

2- Top Menu:

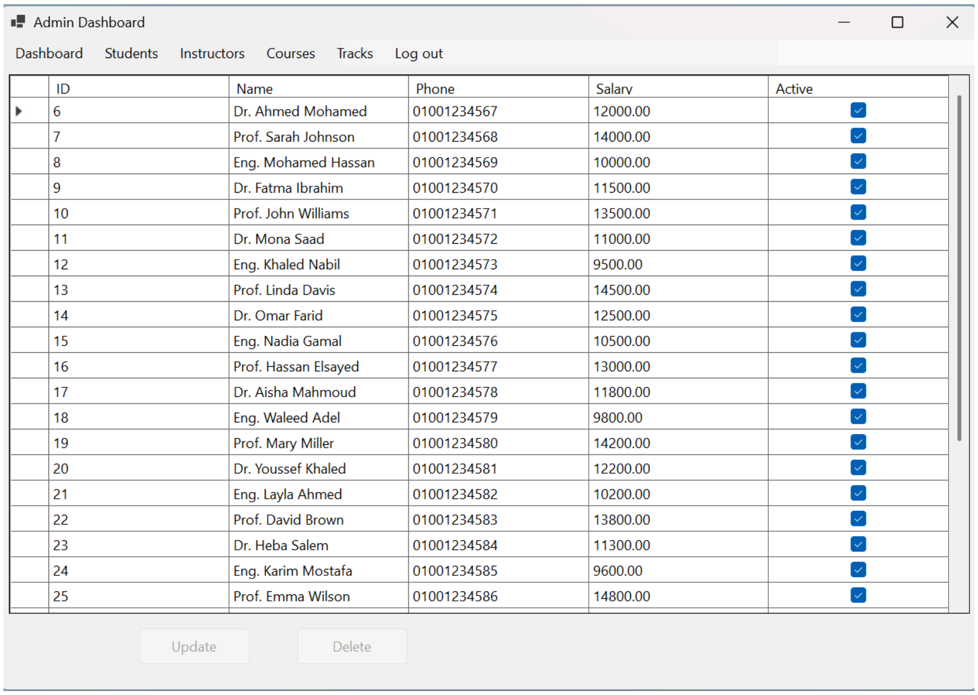
- **Dashboard**
- **Students**
- **Courses**
- **Instructors**
- **Track**
- **Logout**

3 Student Screen

1-Show all Student

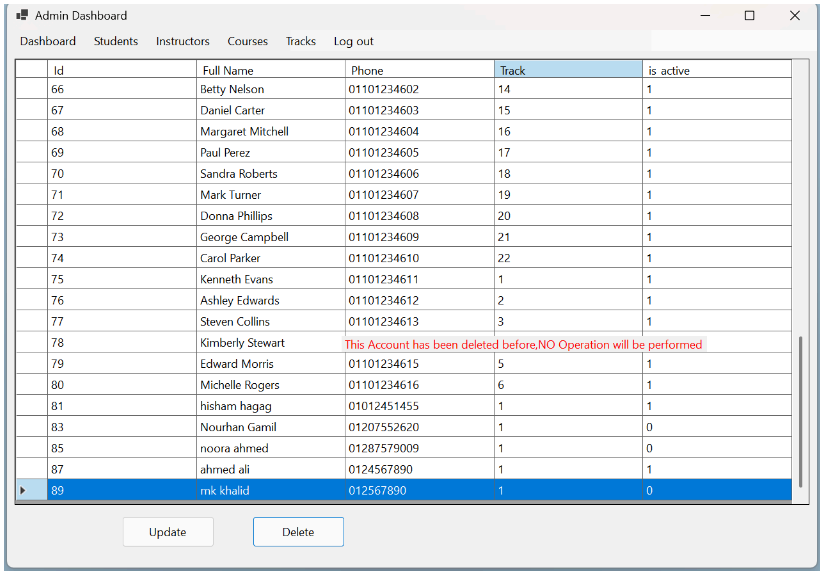
2-Add Student

3-Delete by make is active =0



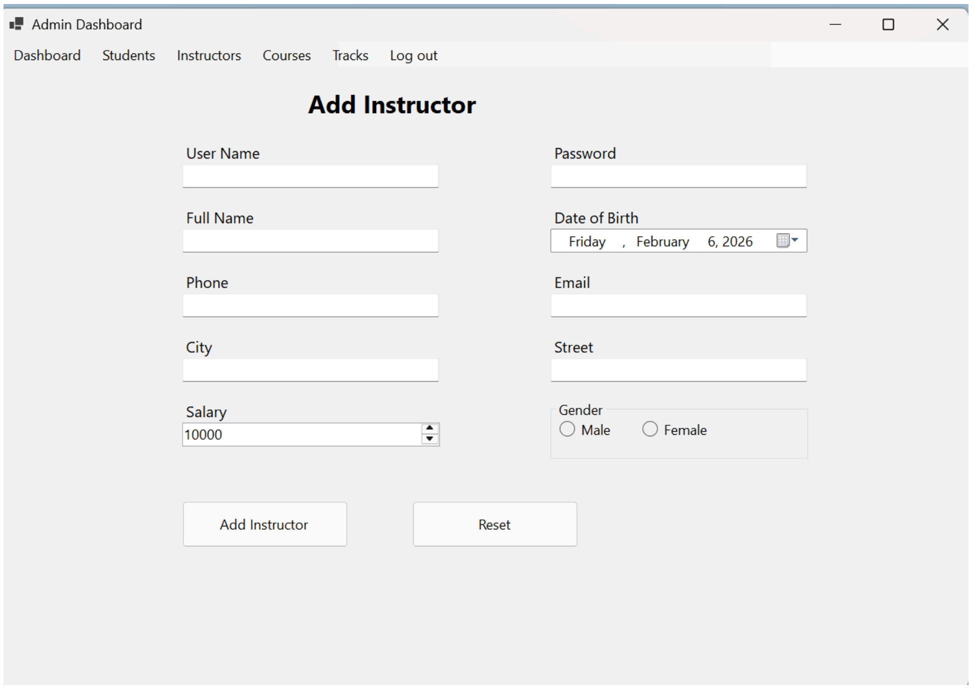
ID	Name	Phone	Salary	Active
6	Dr. Ahmed Mohamed	01001234567	12000.00	☑
7	Prof. Sarah Johnson	01001234568	14000.00	☑
8	Eng. Mohamed Hassan	01001234569	10000.00	☑
9	Dr. Fatma Ibrahim	01001234570	11500.00	☑
10	Prof. John Williams	01001234571	13500.00	☑
11	Dr. Mona Saad	01001234572	11000.00	☑
12	Eng. Khaled Nabil	01001234573	9500.00	☑
13	Prof. Linda Davis	01001234574	14500.00	☑
14	Dr. Omar Farid	01001234575	12500.00	☑
15	Eng. Nadia Gamal	01001234576	10500.00	☑
16	Prof. Hassan Elsayed	01001234577	13000.00	☑
17	Dr. Aisha Mahmoud	01001234578	11800.00	☑
18	Eng. Waleed Adel	01001234579	9800.00	☑
19	Prof. Mary Miller	01001234580	14200.00	☑
20	Dr. Youssef Khaled	01001234581	12200.00	☑
21	Eng. Layla Ahmed	01001234582	10200.00	☑
22	Prof. David Brown	01001234583	13800.00	☑
23	Dr. Heba Salem	01001234584	11300.00	☑
24	Eng. Karim Mostafa	01001234585	9600.00	☑
25	Prof. Emma Wilson	01001234586	14800.00	☑

Rule:we cant delta student which is active column =0



Id	Full Name	Phone	Track	is active
66	Betty Nelson	01101234602	14	1
67	Daniel Carter	01101234603	15	1
68	Margaret Mitchell	01101234604	16	1
69	Paul Perez	01101234605	17	1
70	Sandra Roberts	01101234606	18	1
71	Mark Turner	01101234607	19	1
72	Donna Phillips	01101234608	20	1
73	George Campbell	01101234609	21	1
74	Carol Parker	01101234610	22	1
75	Kenneth Evans	01101234611	1	1
76	Ashley Edwards	01101234612	2	1
77	Steven Collins	01101234613	3	1
78	Kimberly Stewart	This Account has been deleted before,NO Operation will be performed		
79	Edward Morris	01101234615	5	1
80	Michelle Rogers	01101234616	6	1
81	hisham hagag	01012451455	1	1
83	Nourhan Gamil	01207552620	1	0
85	noora ahmed	01287579009	1	0
87	ahmed ali	0124567890	1	1
89	mk khalid	012567890	1	0

Adding Student Screen



Add Instructor

User Name

Password

Full Name

Date of Birth: Friday , February 6, 2026

Phone

Email

City

Street

Salary: 10000

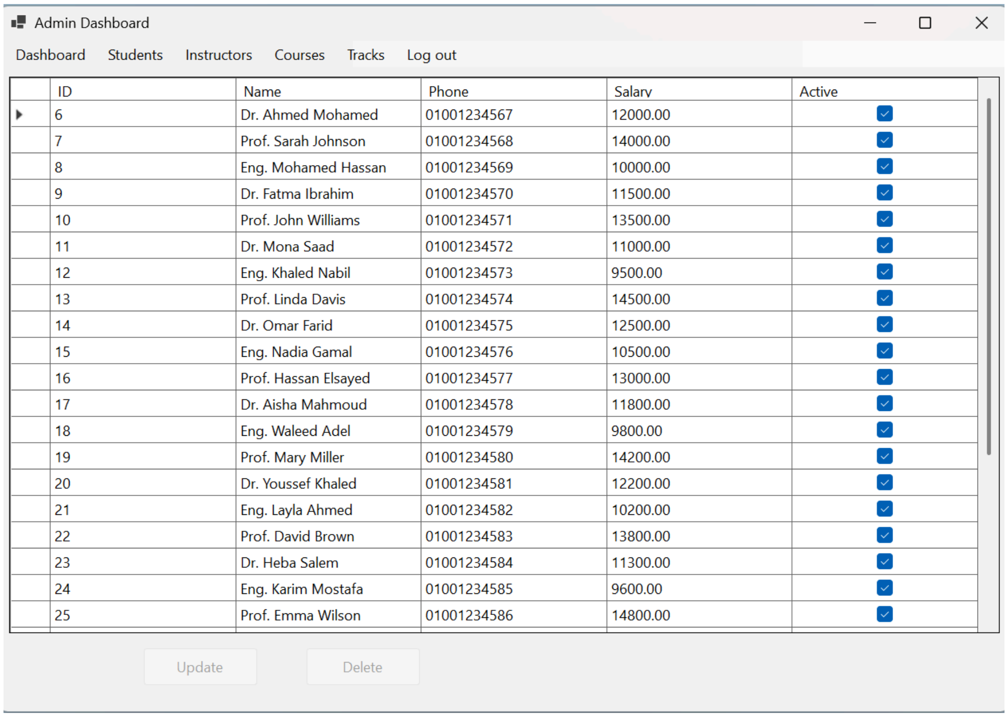
Gender
○ Male ○ Female

3 Instructor Screen

1-Show all Instructor

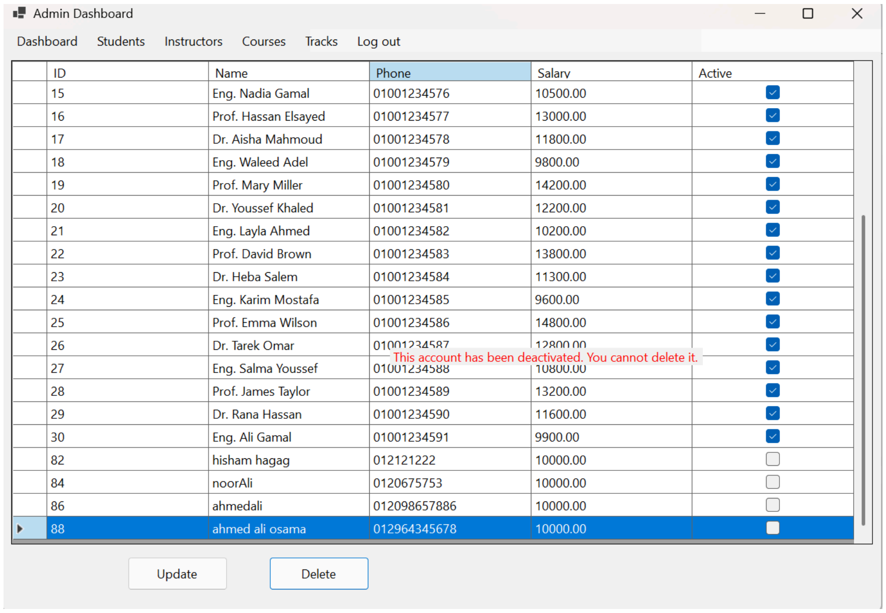
2-Add Instructor

3-Delete by make is active =0



ID	Name	Phone	Salary	Active
6	Dr. Ahmed Mohamed	01001234567	12000.00	☑
7	Prof. Sarah Johnson	01001234568	14000.00	☑
8	Eng. Mohamed Hassan	01001234569	10000.00	☑
9	Dr. Fatma Ibrahim	01001234570	11500.00	☑
10	Prof. John Williams	01001234571	13500.00	☑
11	Dr. Mona Saad	01001234572	11000.00	☑
12	Eng. Khaled Nabil	01001234573	9500.00	☑
13	Prof. Linda Davis	01001234574	14500.00	☑
14	Dr. Omar Farid	01001234575	12500.00	☑
15	Eng. Nadia Gamal	01001234576	10500.00	☑
16	Prof. Hassan Elsayed	01001234577	13000.00	☑
17	Dr. Aisha Mahmoud	01001234578	11800.00	☑
18	Eng. Waleed Adel	01001234579	9800.00	☑
19	Prof. Mary Miller	01001234580	14200.00	☑
20	Dr. Youssef Khaled	01001234581	12200.00	☑
21	Eng. Layla Ahmed	01001234582	10200.00	☑
22	Prof. David Brown	01001234583	13800.00	☑
23	Dr. Heba Salem	01001234584	11300.00	☑
24	Eng. Karim Mostafa	01001234585	9600.00	☑
25	Prof. Emma Wilson	01001234586	14800.00	☑

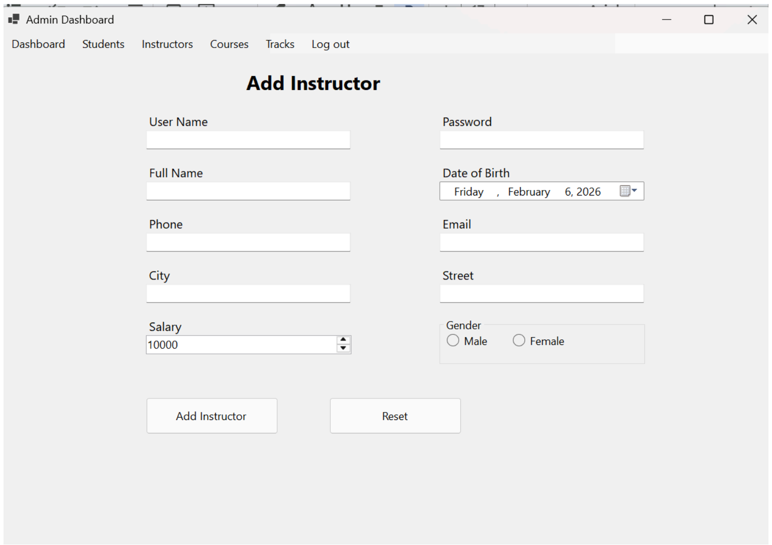
Rule:we cant delta Instructor which is active column =0



ID	Name	Phone	Salary	Active
15	Eng. Nadia Gamal	01001234576	10500.00	☑
16	Prof. Hassan Elsayed	01001234577	13000.00	☑
17	Dr. Aisha Mahmoud	01001234578	11800.00	☑
18	Eng. Waleed Adel	01001234579	9800.00	☑
19	Prof. Mary Miller	01001234580	14200.00	☑
20	Dr. Youssef Khaled	01001234581	12200.00	☑
21	Eng. Layla Ahmed	01001234582	10200.00	☑
22	Prof. David Brown	01001234583	13800.00	☑
23	Dr. Heba Salem	01001234584	11300.00	☑
24	Eng. Karim Mostafa	01001234585	9600.00	☑
25	Prof. Emma Wilson	01001234586	14800.00	☑
26	Dr. Tarek Omar	01001234587	12800.00	☑
27	Eng. Salma Youssef	01001234588	10800.00	☑
28	Prof. James Taylor	01001234589	13200.00	☑
29	Dr. Rana Hassan	01001234590	11600.00	☑
30	Eng. Ali Gamal	01001234591	9900.00	☑
82	hisham hagag	012121222	10000.00	□
84	noorAli	0120675753	10000.00	□
86	ahmedali	012098657886	10000.00	□
88	ahmed ali osama	012964345678	10000.00	□

This account has been deactivated. You cannot delete it.

Adding Instructor Screen



Add Instructor

User Name

Password

Full Name

Date of Birth
Friday , February 6, 2026

Phone

Email

City

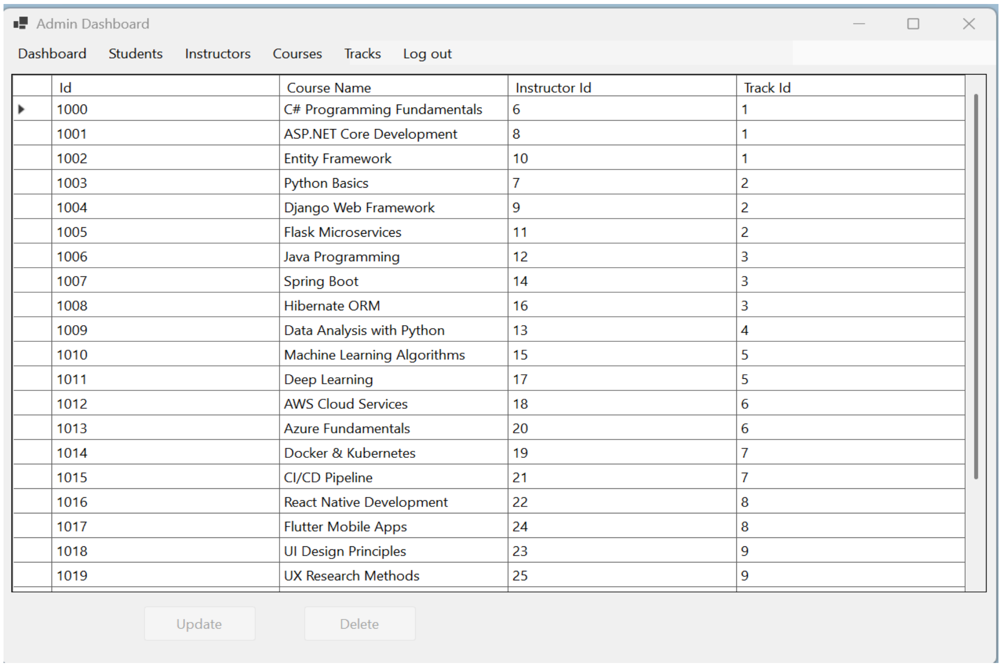
Street

Salary
10000

Gender
○ Male ○ Female

Course Screen

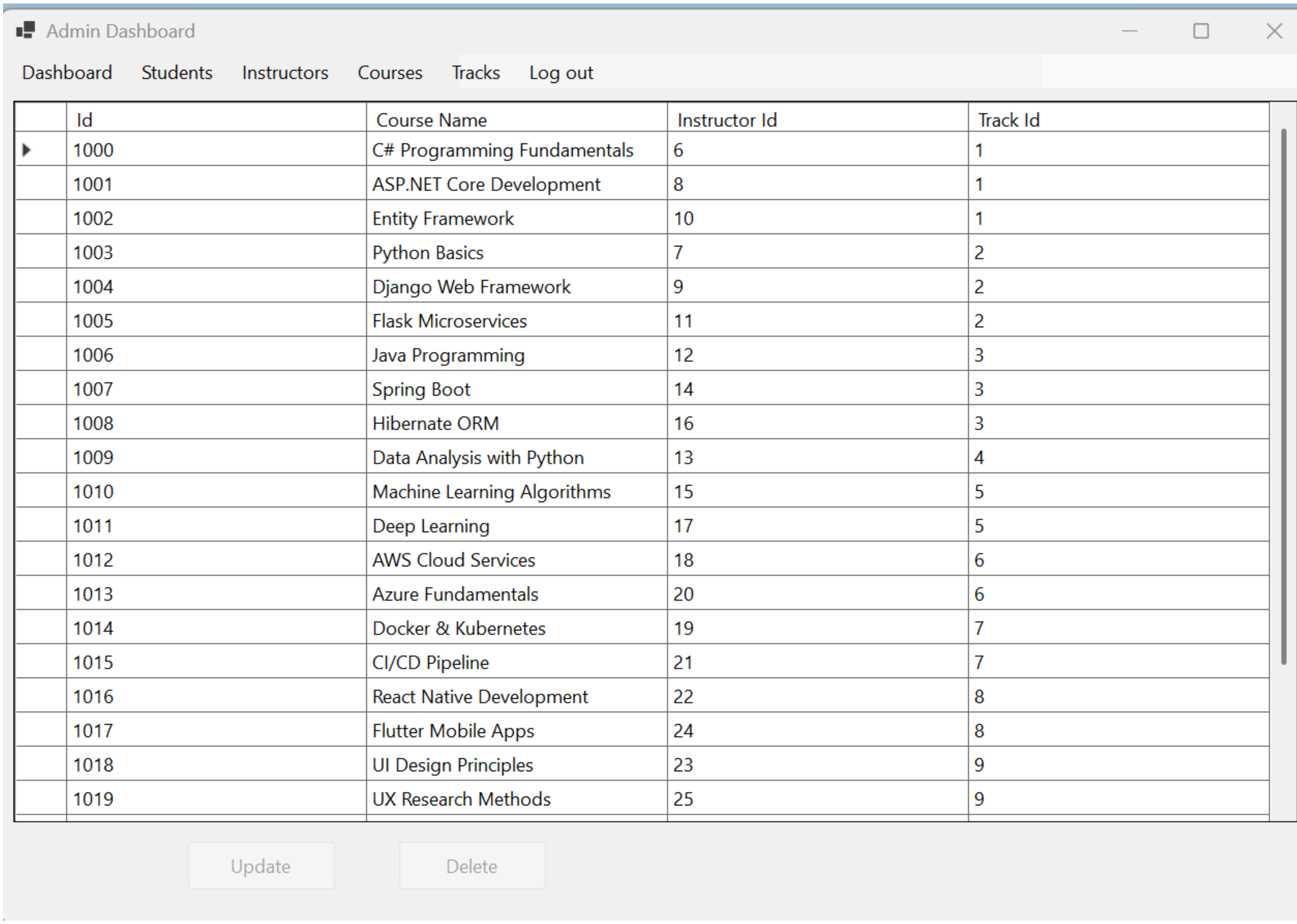
1-Show All Courses
2- Have The Auth To Add ,Update,Delete Courses



Id	Course Name	Instructor Id	Track Id
1000	C# Programming Fundamentals	6	1
1001	ASP.NET Core Development	8	1
1002	Entity Framework	10	1
1003	Python Basics	7	2
1004	Django Web Framework	9	2
1005	Flask Microservices	11	2
1006	Java Programming	12	3
1007	Spring Boot	14	3
1008	Hibernate ORM	16	3
1009	Data Analysis with Python	13	4
1010	Machine Learning Algorithms	15	5
1011	Deep Learning	17	5
1012	AWS Cloud Services	18	6
1013	Azure Fundamentals	20	6
1014	Docker & Kubernetes	19	7
1015	CI/CD Pipeline	21	7
1016	React Native Development	22	8
1017	Flutter Mobile Apps	24	8
1018	UI Design Principles	23	9
1019	UX Research Methods	25	9

Track Screen

1-Show All Tracks
2- Have The Auth To Add ,Update,Delete Track



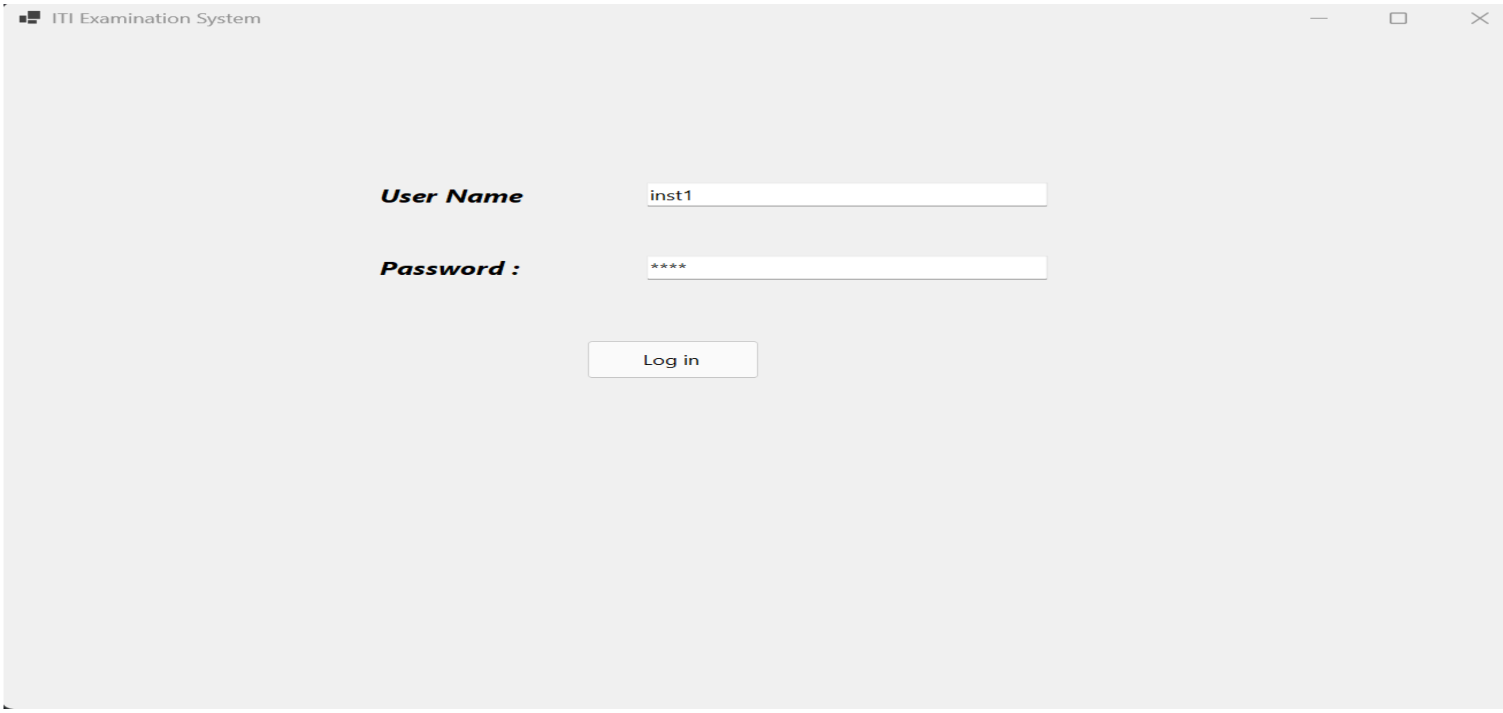
Id	Course Name	Instructor Id	Track Id
1000	C# Programming Fundamentals	6	1
1001	ASP.NET Core Development	8	1
1002	Entity Framework	10	1
1003	Python Basics	7	2
1004	Django Web Framework	9	2
1005	Flask Microservices	11	2
1006	Java Programming	12	3
1007	Spring Boot	14	3
1008	Hibernate ORM	16	3
1009	Data Analysis with Python	13	4
1010	Machine Learning Algorithms	15	5
1011	Deep Learning	17	5
1012	AWS Cloud Services	18	6
1013	Azure Fundamentals	20	6
1014	Docker & Kubernetes	19	7
1015	CI/CD Pipeline	21	7
1016	React Native Development	22	8
1017	Flutter Mobile Apps	24	8
1018	UI Design Principles	23	9
1019	UX Research Methods	25	9

3.1.2 Instructor Dashboard

1️⃣ Login Screen

Instructor enters:

- ○ **Username**
- ○ **Password**
- Click **Login**
- ✔ If credentials are correct → go to **Instructor Dashboard**
- ❌ If wrong → show error message



ITI Examination System

User Name inst1

Password : ****

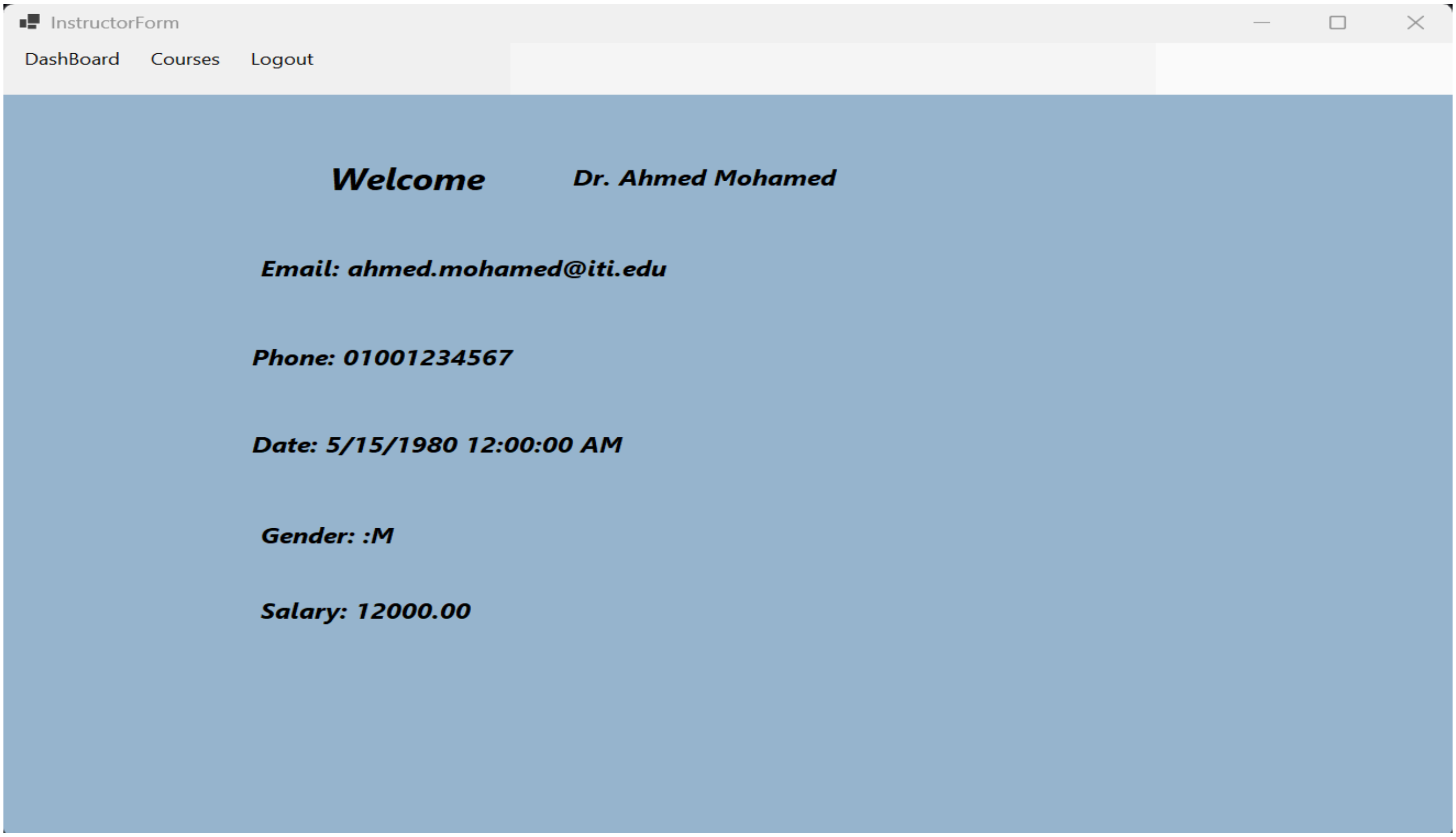
2️⃣Instructor Dashboard

1- Instructor Data
2- Top Menu:

- **Dashboard**
- **Courses**
- **Logout**

From here instructor can:

- View assigned courses
- Generate exams
- View exams
- View student grades



Welcome **Dr. Ahmed Mohamed**

Email: ahmed.mohamed@iti.edu

Phone: 01001234567

Date: 5/15/1980 12:00:00 AM

Gender: :M

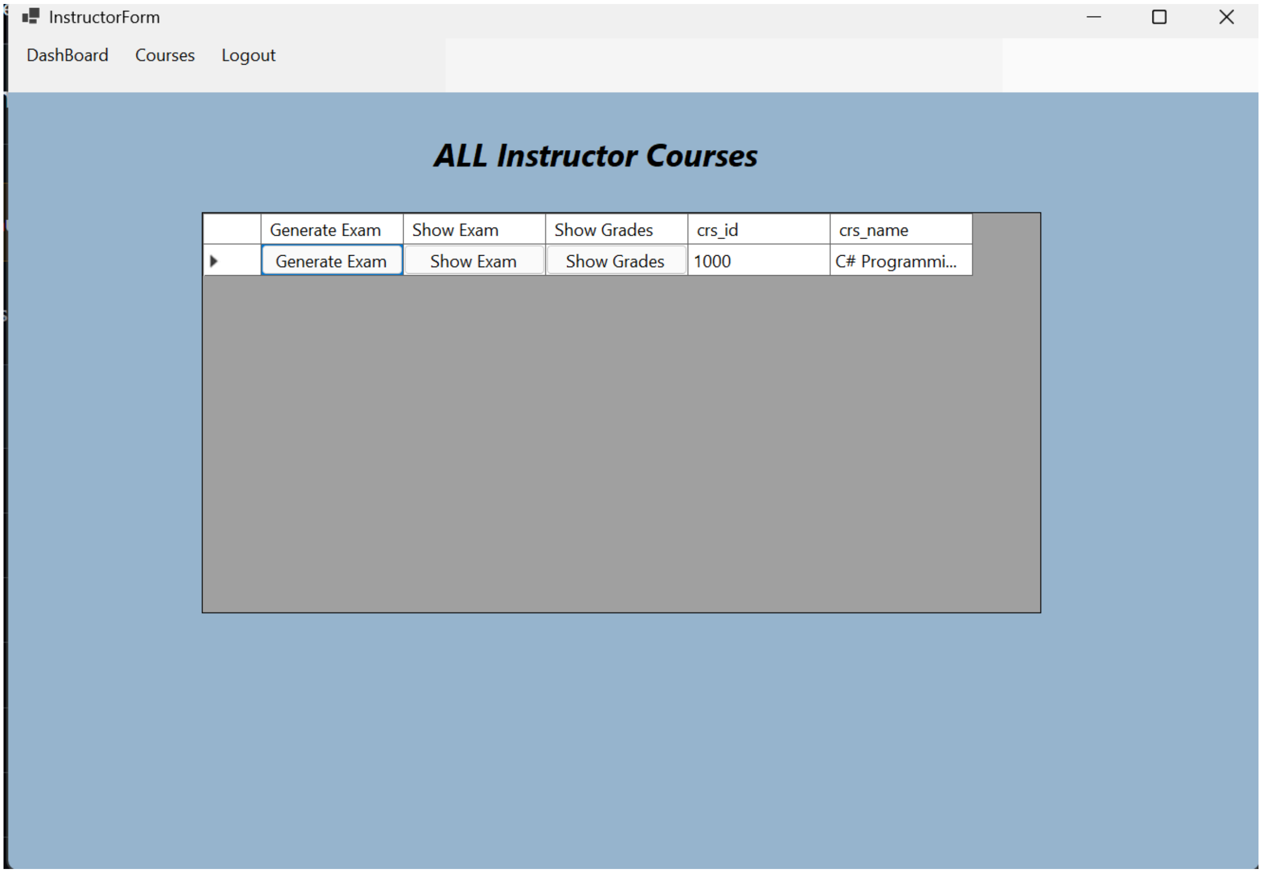
Salary: 12000.00

3️⃣ Courses Screen (All Instructor Courses)

Displayed as a table:

- Course ID
- Course Name
- Actions:
  - **Generate Exam**
  - **Show Exam**
  - **Show Grades**

➡ Instructor selects a course and clicks one action



	Generate Exam	Show Exam	Show Grades	crs_id	crs_name
▶	Generate Exam	Show Exam	Show Grades	1000	C# Programmi...

4️⃣ Generate Exam

Instructor enters:

- **Course ID**
- **Number of MCQ**
- **Number of True/False**
- **Exam Duration**

Actions:

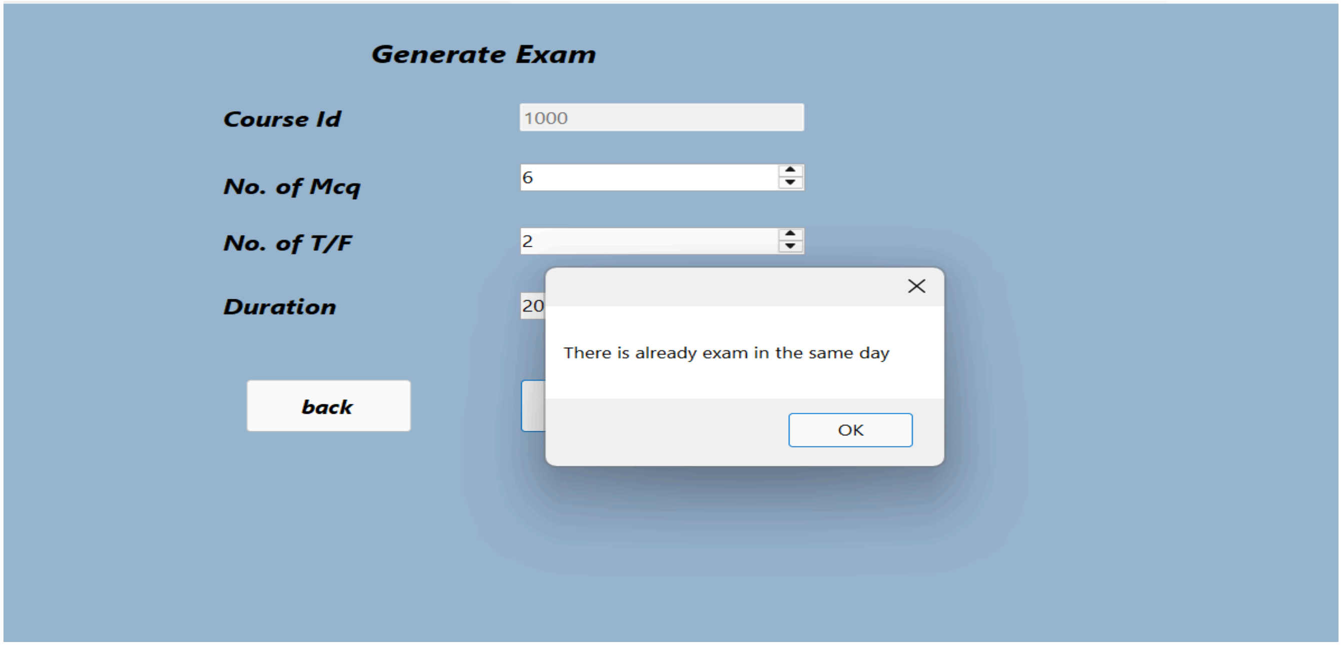
- Click **Generate**

Validation:

- ❌ If exam already exists on same day
→ Show message:
"There is already exam in the same day"
- ✔ Otherwise → Exam created successfully

Buttons:

- **Generate**
- **Back**



Generate Exam

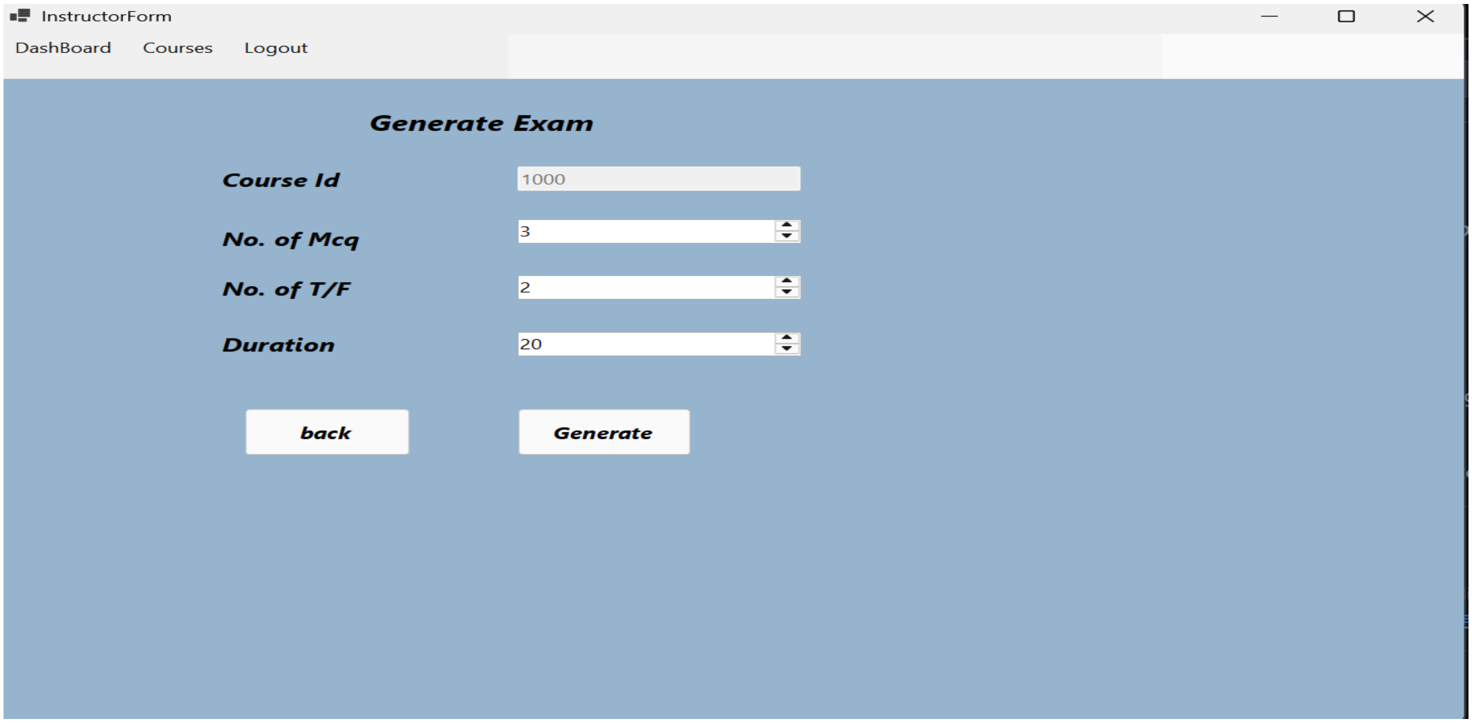
Course Id: 1000

No. of Mcq: 6

No. of T/F: 2

Duration: 20

There is already exam in the same day



Generate Exam

Course Id 1000

No. of Mcq 3

No. of T/F 2

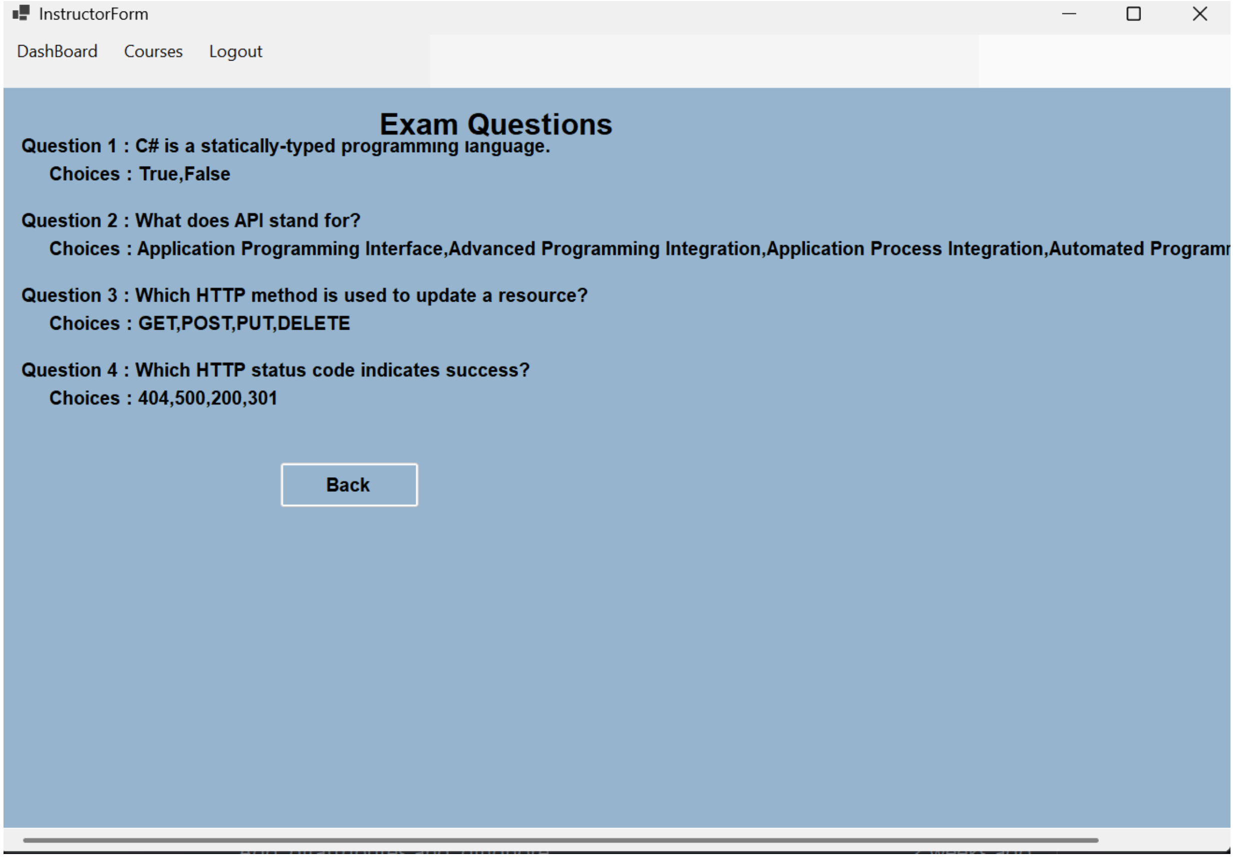
Duration 20

5️⃣ Show Exam

- Displays:

  ○ Exam questions
  ○ MCQ & T/F
- Instructor can:

Review questions
- Click **Back** to return



Exam Questions

Question 1 : C# is a statically-typed programming language.
Choices : True,False

Question 2 : What does API stand for?
Choices : Application Programming Interface,Advanced Programming Integration,Application Process Integration,Automated Programr

Question 3 : Which HTTP method is used to update a resource?
Choices : GET,POST,PUT,DELETE

Question 4 : Which HTTP status code indicates success?
Choices : 404,500,200,301

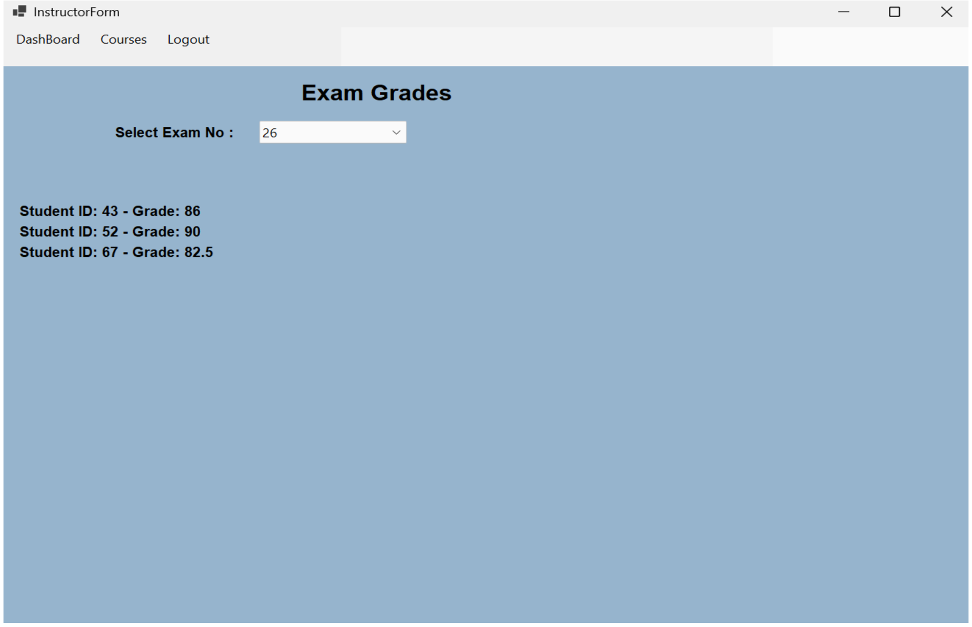
6️⃣ Show Grades

- Instructor selects:

  - **Exam Number**
- System displays:
  - Student ID
  - Grade

Example:

- Student ID: 43 → Grade: 86
- Student ID: 52 → Grade: 90
- Student ID: 67 → Grade: 82.5



Exam Grades

Select Exam No : 26

Student ID: 43 - Grade: 86
Student ID: 52 - Grade: 90
Student ID: 67 - Grade: 82.5

7️⃣Logout

- Instructor clicks **Logout**
- System returns to **Login Screen**

3.1.3 Student Dashboard

1️⃣Login Screen

Student enters:

○ **Username**
○ **Password**

- Click **Login**
- ✔ Valid credentials → **Student Dashboard**
- ❌ Invalid credentials → Error message

2️⃣ Student Dashboard

Top Menu:

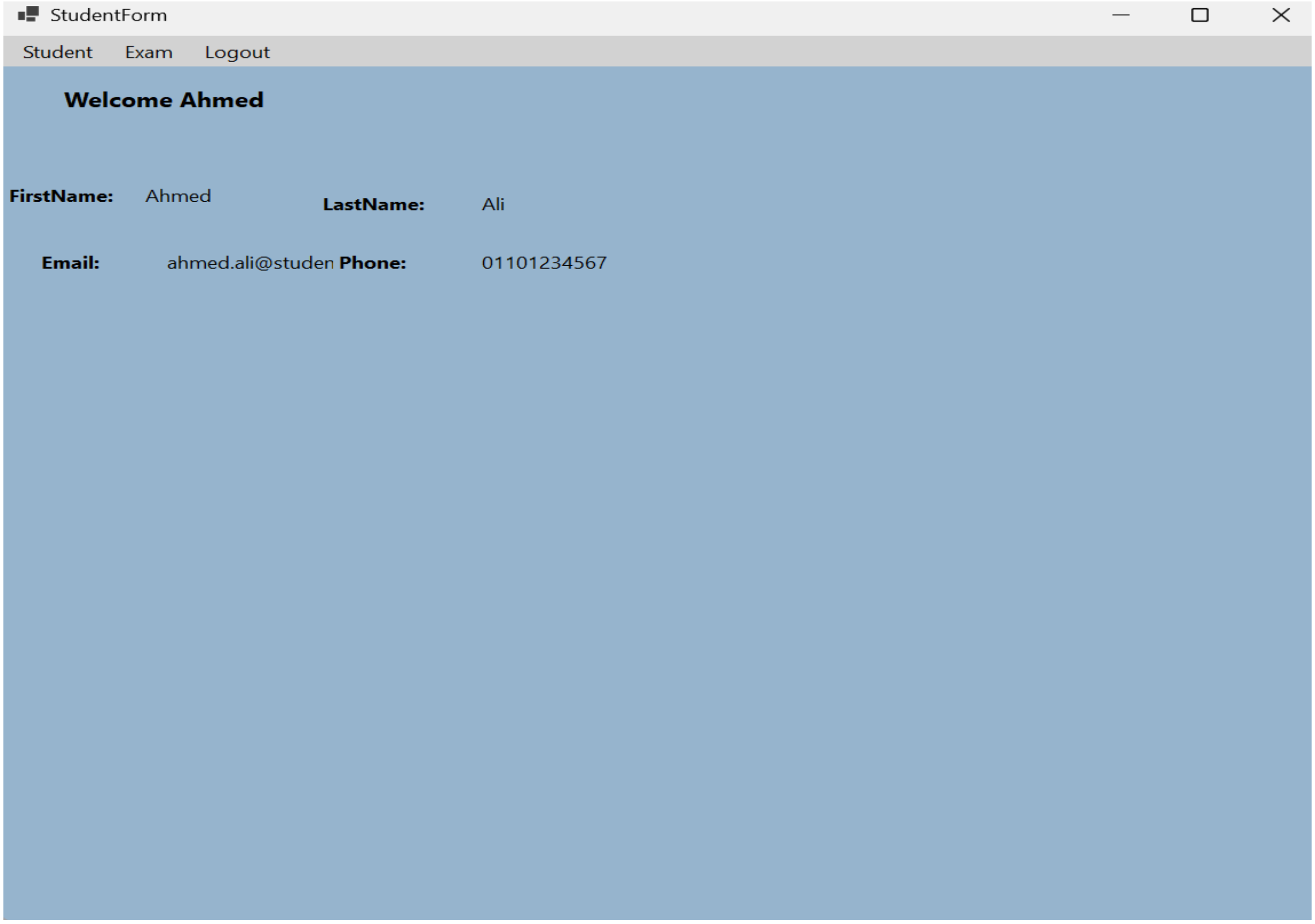
- **Student - - - > [INFO , Course]**
- **Exam**
- **Logout**

From here student can:

- View available courses and their Details
- Take exams

From here Exam can:

Show Answer



Welcome Ahmed

FirstName: Ahmed **LastName:** Ali

Email: ahmed.ali@studen **Phone:** 01101234567

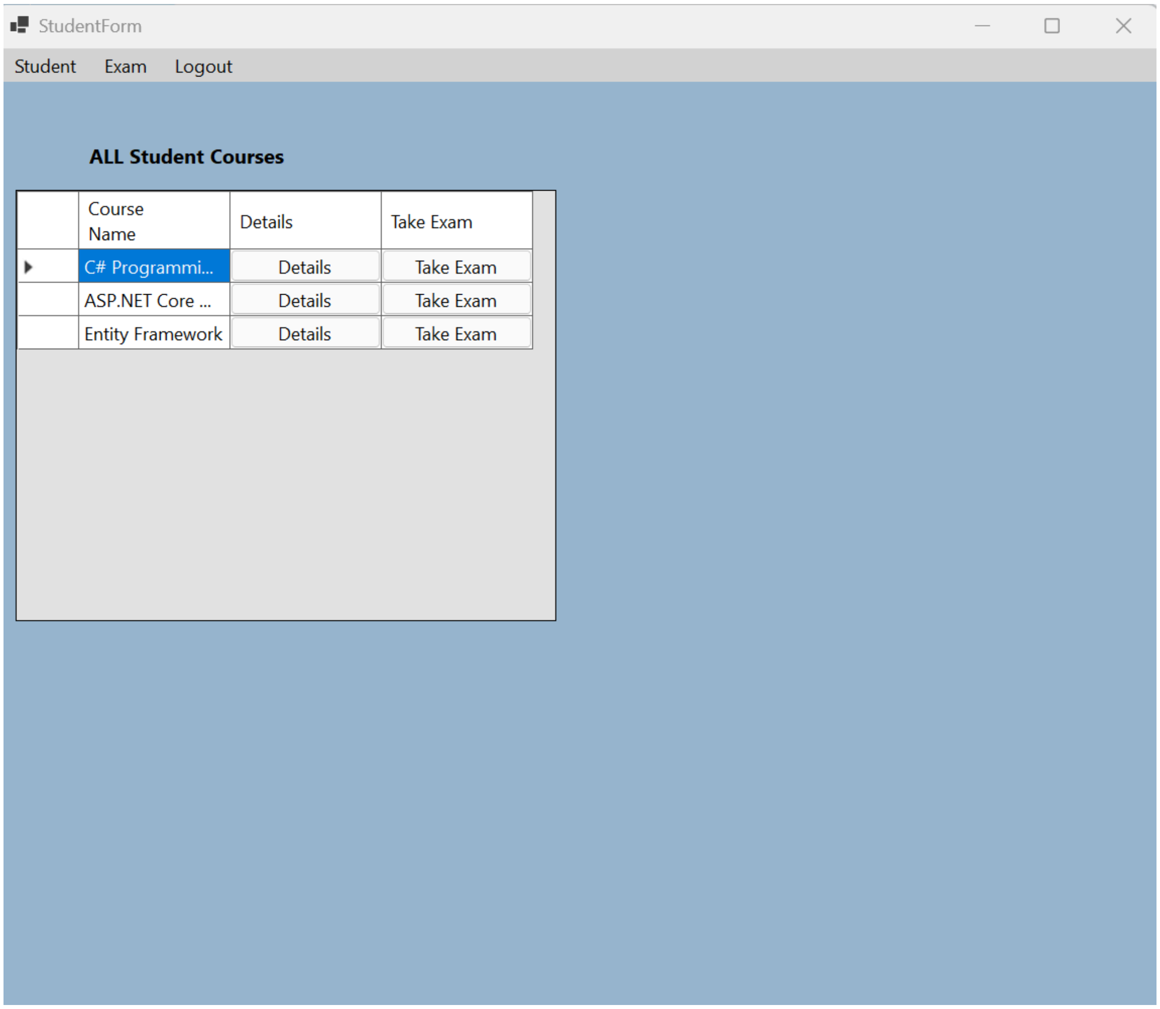
4️⃣ Take Exam

- Exam timer starts
- Student answers:

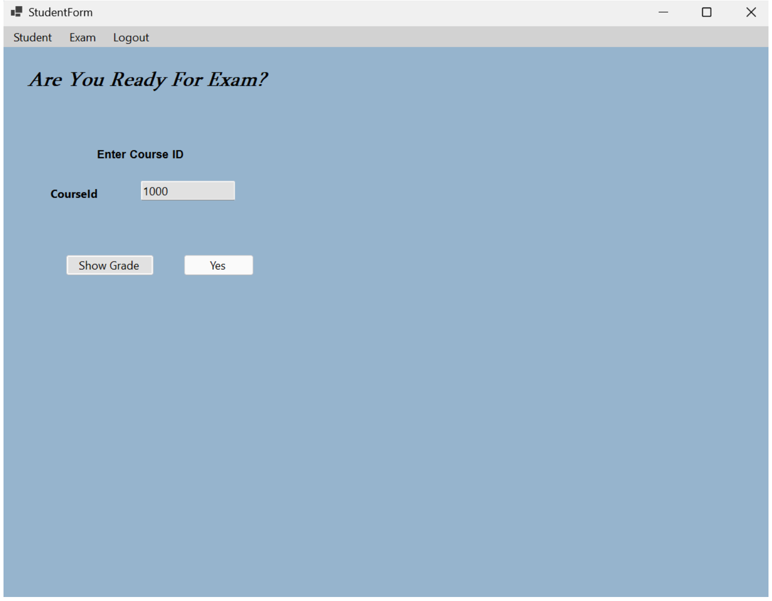
  - **MCQ questions**
  - **True / False questions**

Controls:

- Next
- Submit



	Course Name	Details	Take Exam
▶	C# Programmi...	Details	Take Exam
	ASP.NET Core ...	Details	Take Exam
	Entity Framework	Details	Take Exam

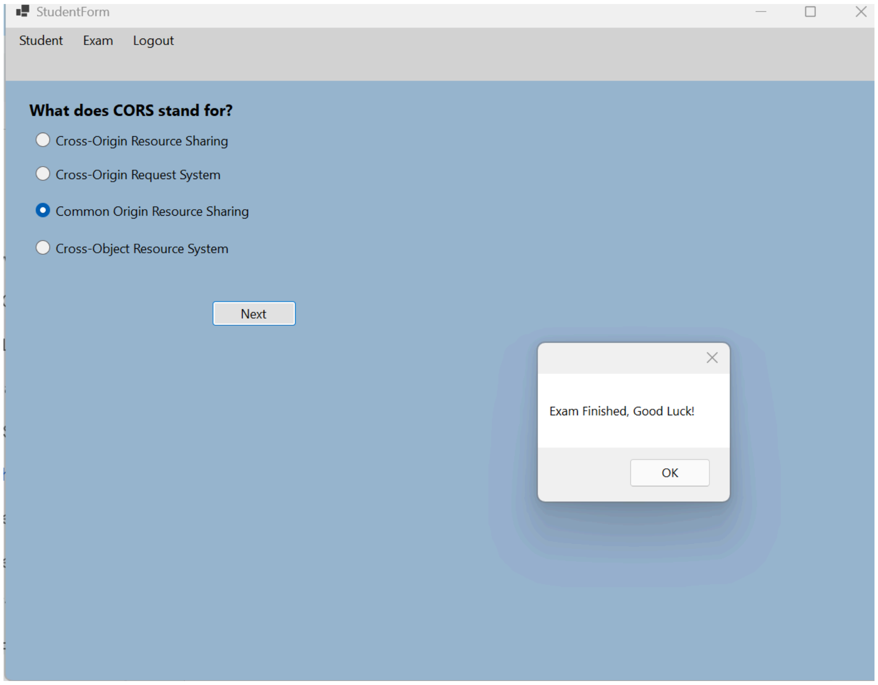


Are You Ready For Exam?

Enter Course ID

CourseId 1000

After Taking Exam This is Last Q in Exam - - - - > Submit Exam



What does CORS stand for?

○ Cross-Origin Resource Sharing

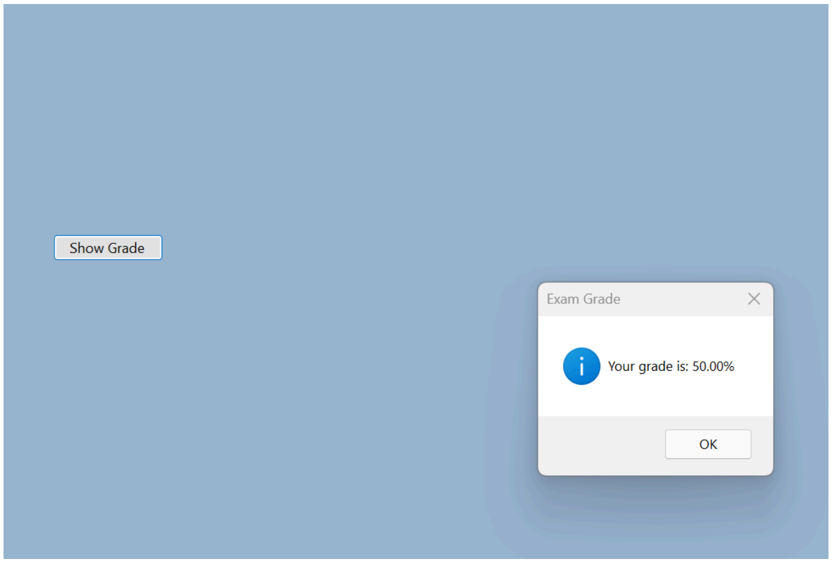
○ Cross-Origin Request System

● Common Origin Resource Sharing

○ Cross-Object Resource System

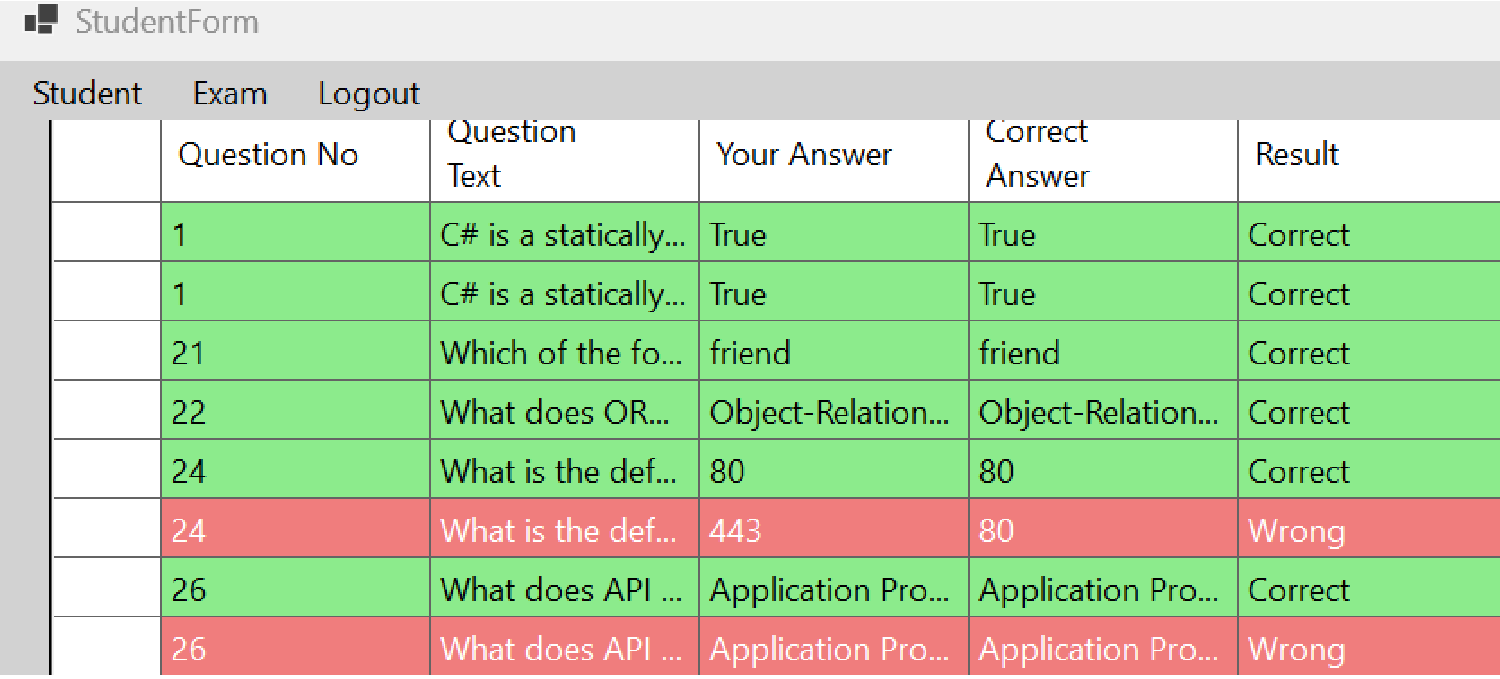
Exam Finished, Good Luck!

After Submitting Exam Click - - - > ShowGrade



Your grade is: 50.00%

By Clicking on **ShowAnswer** From Menu we can check answers



	Question No	Question Text	Your Answer	Correct Answer	Result
	1	C# is a statically...	True	True	Correct
	1	C# is a statically...	True	True	Correct
	21	Which of the fo...	friend	friend	Correct
	22	What does OR...	Object-Relation...	Object-Relation...	Correct
	24	What is the def...	80	80	Correct
	24	What is the def...	443	80	Wrong
	26	What does API ...	Application Pro...	Application Pro...	Correct
	26	What does API ...	Application Pro...	Application Pro...	Wrong

7 Logout

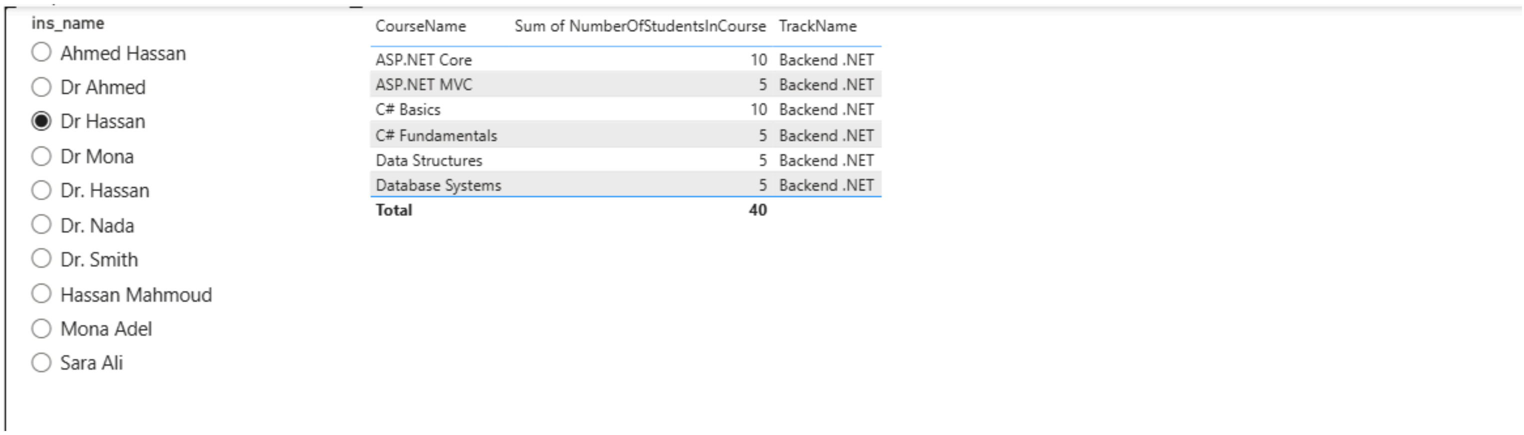
- Student clicks **Logout**
- Return to **Login**

Chapter 4
Results and Reports

Chapter 4: Results And Reports

4.1 Reports

4.1.1 Instructor Courses Report



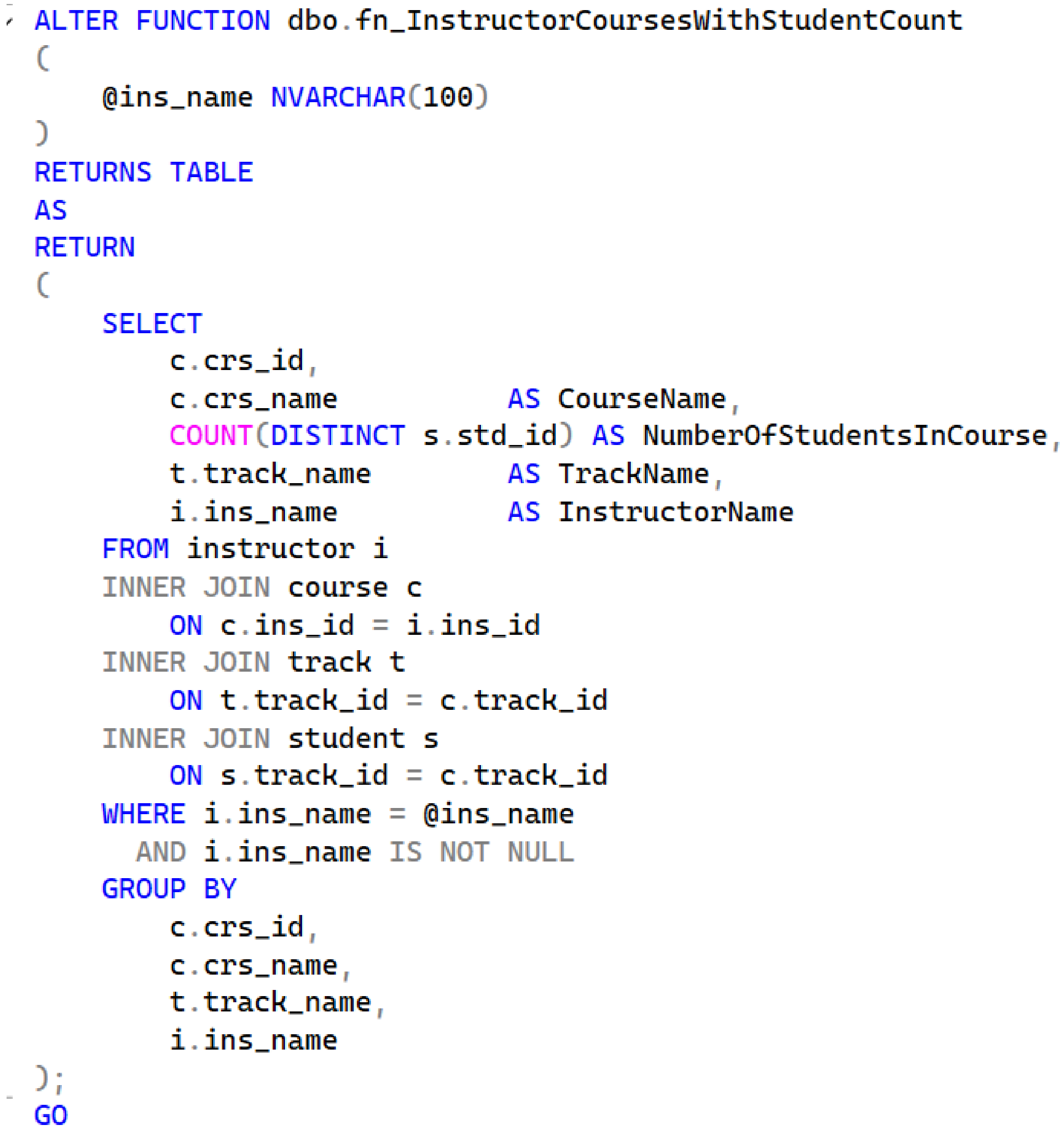
ins_name

- ○ Ahmed Hassan
- ○ Dr Ahmed
- ● Dr Hassan
- ○ Dr Mona
- ○ Dr. Hassan
- ○ Dr. Nada
- ○ Dr. Smith
- ○ Hassan Mahmoud
- ○ Mona Adel
- ○ Sara Ali

CourseName	Sum of NumberOfStudentsInCourse	TrackName
ASP.NET Core	10	Backend .NET
ASP.NET MVC	5	Backend .NET
C# Basics	10	Backend .NET
C# Fundamentals	5	Backend .NET
Data Structures	5	Backend .NET
Database Systems	5	Backend .NET
Total	**40**	

Overview: When you select an instructor name, the report displays all courses taught by that instructor along with the number of enrolled students. The table updates automatically based on the selected instructor.

Implementation Explanation : I created a slicer using the ins_name (instructor name) field, allowing users to select a single instructor. A table visual was then added to display CourseName, Sum of NumberOfStudentsInCourse, and TrackName, which automatically filters to show only the courses taught by the selected instructor. This interactive setup provides a focused overview of each instructor's courses and student enrollment with one click.

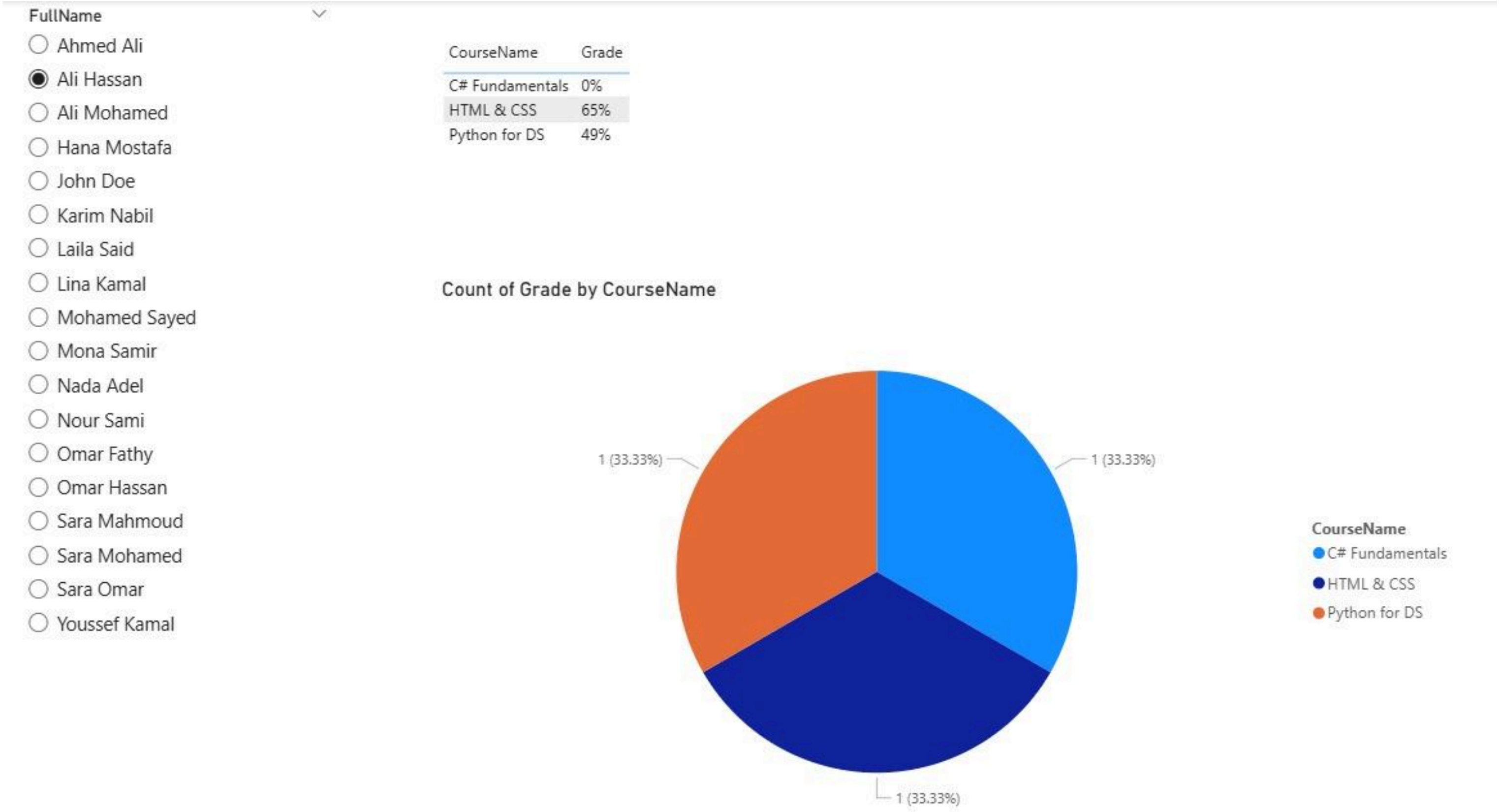


```sql
ALTER FUNCTION dbo.fn_InstructorCoursesWithStudentCount
(
    @ins_name NVARCHAR(100)
)
RETURNS TABLE
AS
RETURN
(
    SELECT
        c.crs_id,
        c.crs_name          AS CourseName,
        COUNT(DISTINCT s.std_id) AS NumberOfStudentsInCourse,
        t.track_name        AS TrackName,
        i.ins_name          AS InstructorName
    FROM instructor i
    INNER JOIN course c
        ON c.ins_id = i.ins_id
    INNER JOIN track t
        ON t.track_id = c.track_id
    INNER JOIN student s
        ON s.track_id = c.track_id
    WHERE i.ins_name = @ins_name
      AND i.ins_name IS NOT NULL
    GROUP BY
        c.crs_id,
        c.crs_name,
        t.track_name,
        i.ins_name
);
GO
```

This table-valued function fn_InstructorCoursesWithStudentCount takes an instructor name (@ins_name) and returns a table listing:

- Each course taught by that instructor
- The course name
- The number of **distinct students** enrolled in that course (based on students in the same track)
- The track name
- The instructor name

4.1.2 Student Course Grade Report



FullName

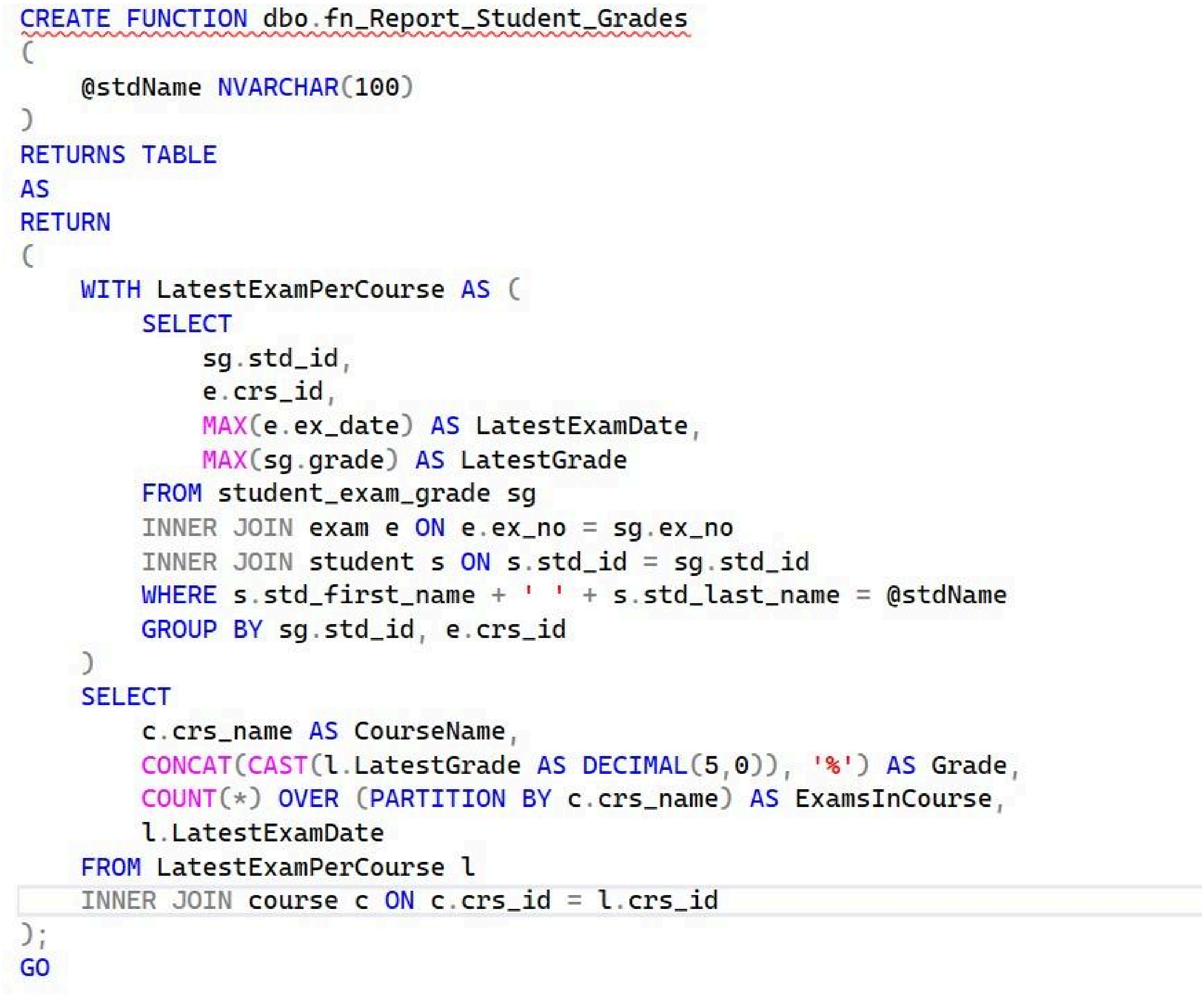
○ Ahmed Ali
● Ali Hassan
○ Ali Mohamed
○ Hana Mostafa
○ John Doe
○ Karim Nabil
○ Laila Said
○ Lina Kamal
○ Mohamed Sayed
○ Mona Samir
○ Nada Adel
○ Nour Sami
○ Omar Fathy
○ Omar Hassan
○ Sara Mahmoud
○ Sara Mohamed
○ Sara Omar
○ Youssef Kamal

CourseName	Grade
C# Fundamentals	0%
HTML & CSS	65%
Python for DS	49%

Count of Grade by CourseName

Overview: This section allows users to select a student and view their courses, grades, and grade distribution through an interactive chart.

Implementation Explanation: I added a slicer based on the FullName field to let users select a single student. A table visual was placed next to it showing the selected student's CourseName and corresponding Grade, which updates automatically to display only that student's enrolled courses and grades. Below the table, a pie chart visual titled "Count of Grade by CourseName" shows the distribution of courses (with their grades) for the selected student, making it easy to see at a glance how many courses the student has taken and their performance across them.

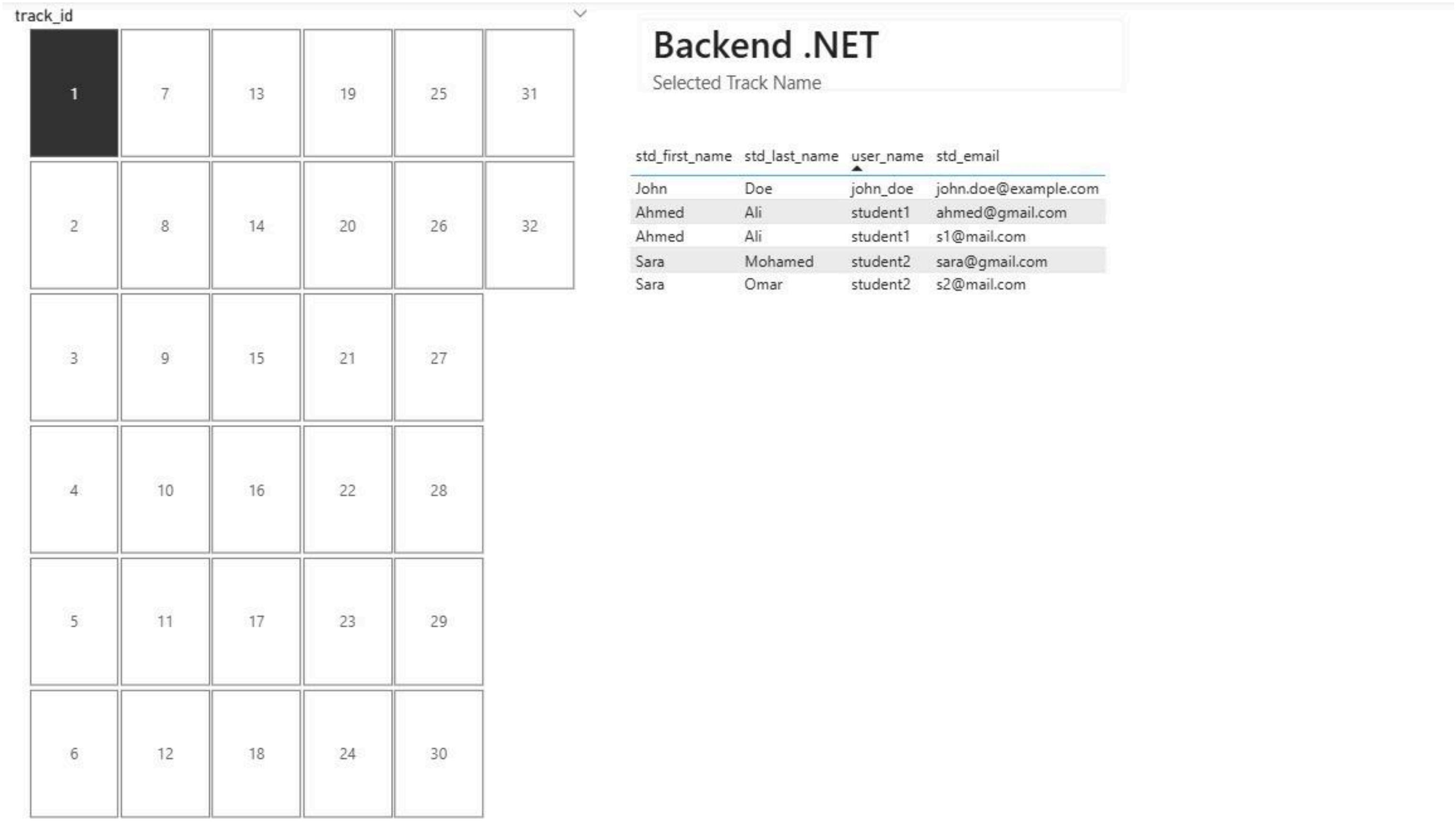


```
CREATE FUNCTION dbo.fn_Report_Student_Grades
(
    @stdName NVARCHAR(100)
)
RETURNS TABLE
AS
RETURN
(
    WITH LatestExamPerCourse AS (
        SELECT
            sg.std_id,
            e.crs_id,
            MAX(e.ex_date) AS LatestExamDate,
            MAX(sg.grade) AS LatestGrade
        FROM student_exam_grade sg
        INNER JOIN exam e ON e.ex_no = sg.ex_no
        INNER JOIN student s ON s.std_id = sg.std_id
        WHERE s.std_first_name + ' ' + s.std_last_name = @stdName
        GROUP BY sg.std_id, e.crs_id
    )
    SELECT
        c.crs_name AS CourseName,
        CONCAT(CAST(l.LatestGrade AS DECIMAL(5,0)), '%') AS Grade,
        COUNT(*) OVER (PARTITION BY c.crs_name) AS ExamsInCourse,
        l.LatestExamDate
    FROM LatestExamPerCourse l
    INNER JOIN course c ON c.crs_id = l.crs_id
);
GO
```

This Function fn_Report_Student_Grades takes a student Name (@stdName) and returns a report showing:

- For each course the student has taken: the course name
- The grade of the **most recent exam** in that course (with % sign)
- The date of that latest exam
- How many exams the student has taken in that course altogether

4.1.3 Track Student Report



track_id

1	7	13	19	25	31
2	8	14	20	26	32
3	9	15	21	27	
4	10	16	22	28	
5	11	17	23	29	
6	12	18	24	30	

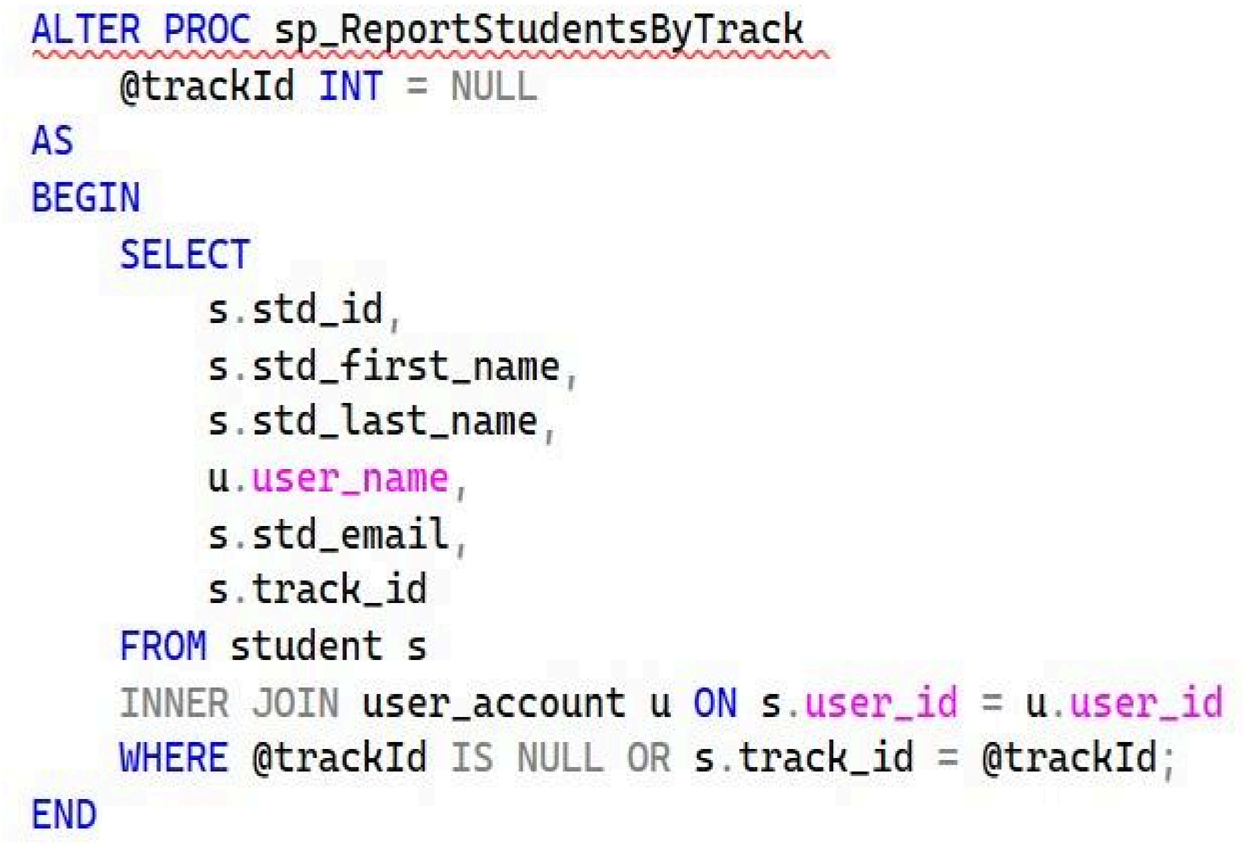
Backend .NET

Selected Track Name

std_first_name	std_last_name	user_name	std_email
John	Doe	john_doe	john.doe@example.com
Ahmed	Ali	student1	ahmed@gmail.com
Ahmed	Ali	student1	s1@mail.com
Sara	Mohamed	student2	sara@gmail.com
Sara	Omar	student2	s2@mail.com

Overview: This report shows the students enrolled in a selected track .When a track is selected from the slicer, the table updates automatically to display only the students enrolled in that track.

Implementation Explanation : I used a slicer (or single-select filter) on the TrackName field to allow users to choose one track . A Table visual displays the students organized in a grid layout by track_id, and student details (std_first_name, std_last_name, user_name, std_email) appearing in a separate table and finally a card that display the current track name then updates to list only the students enrolled in the selected track.

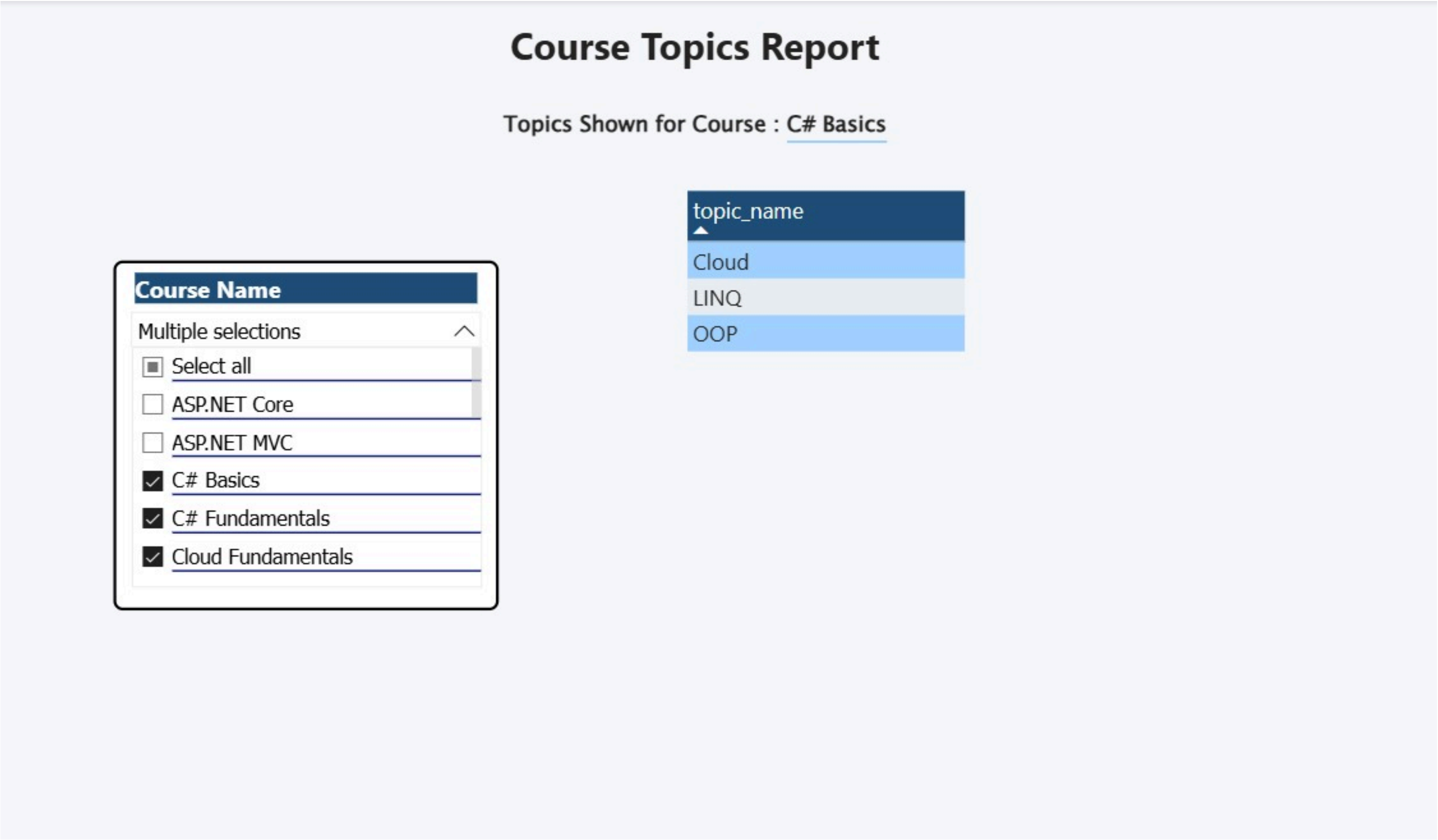


```sql
ALTER PROC sp_ReportStudentsByTrack
    @trackId INT = NULL
AS
BEGIN
    SELECT
        s.std_id,
        s.std_first_name,
        s.std_last_name,
        u.user_name,
        s.std_email,
        s.track_id
    FROM student s
    INNER JOIN user_account u ON s.user_id = u.user_id
    WHERE @trackId IS NULL OR s.track_id = @trackId;
END
```

This stored procedure sp_ReportStudentsByTrack takes an optional track ID parameter (@trackId INT = NULL) and returns:

- A list of all students (or only those in the specified track if @trackId is provided)
- For each student: std_id, std_first_name, std_last_name, user_name, std_email
- Student details are retrieved by joining the student and user_account tables
- When @trackId is NULL, it returns students from **all tracks**; otherwise, it filters to show only students enrolled in the given track

4.1.4 Course Topics Report



Course Topics Report

Topics Shown for Course : C# Basics

topic_name
Cloud
LINQ
OOP

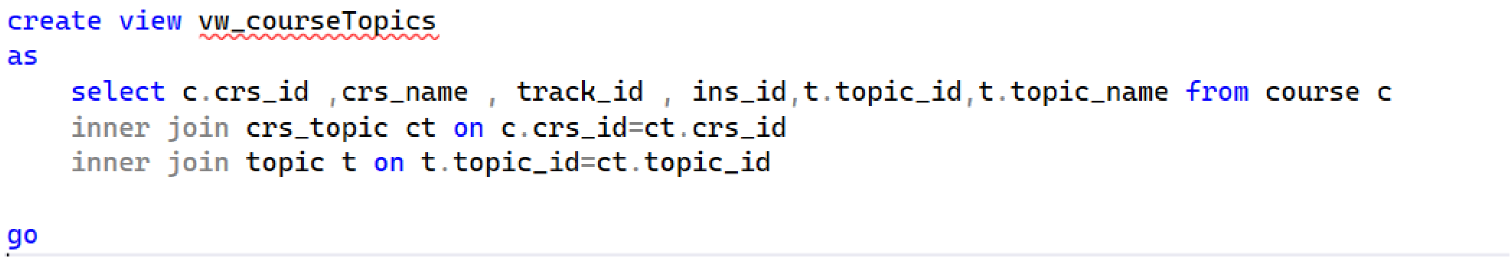
Course Name

Multiple selections

- ▣ Select all
- □ ASP.NET Core
- □ ASP.NET MVC
- ☑ C# Basics
- ☑ C# Fundamentals
- ☑ Cloud Fundamentals

Overview: This report displays the topics covered in one or more selected courses. When the user selects course(s) from the multi-select slicer (e.g., checking “C# Basics”), the list of topics automatically updates to show only the topics associated with the chosen course(s), providing a clear view of the curriculum content for the selected course(s).

Implementation Explanation: I added a multi-select slicer using the Course Name field, allowing users to choose one or multiple courses (with “Select all” and individual checkboxes). A separate card visual displays the currently filtered course name(s) for context (e.g., “Topics Shown for Course: C# Basics”). A table or list visual then shows the topic_name field, which automatically filters to display only the topics linked to the selected course(s), making it easy to review the detailed topics of any chosen course in the program.



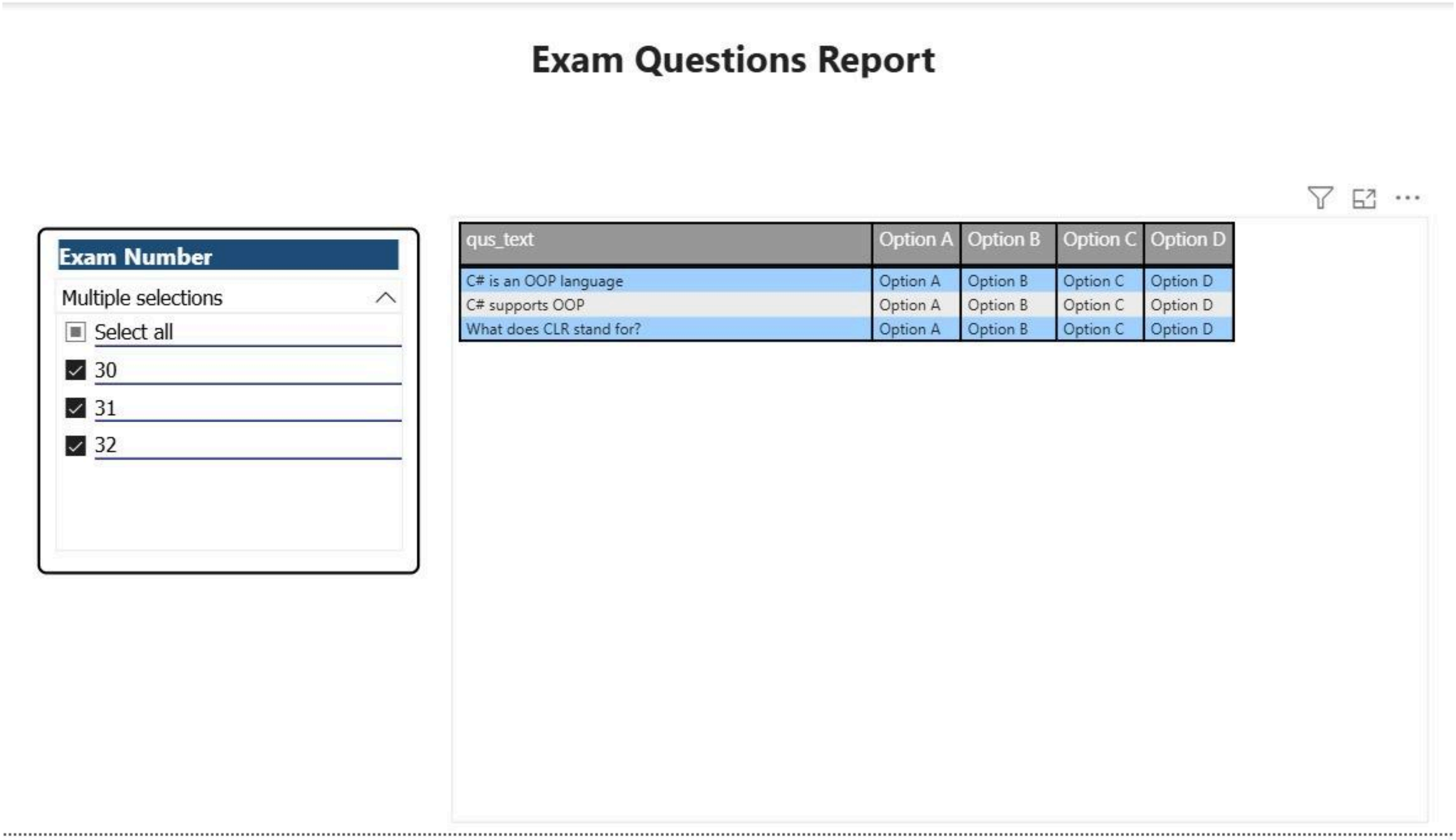
```
create view vw_courseTopics
as
    select c.crs_id ,crs_name , track_id , ins_id,t.topic_id,t.topic_name from course c
    inner join crs_topic ct on c.crs_id=ct.crs_id
    inner join topic t on t.topic_id=ct.topic_id

go
```

This SQL view named vw_courseTopics that joins four tables:

- course → to get course details (crs_id, crs_name, track_id, ins_id)
- crs_topic → the bridge table that links courses to topics
- topic → to retrieve the actual topic names (topic_id, topic_name)

The view uses INNER JOINs to combine these tables correctly, producing one row per course-topic relationship.

4.1.5 Exam Questions Report



Exam Questions Report

Exam Number

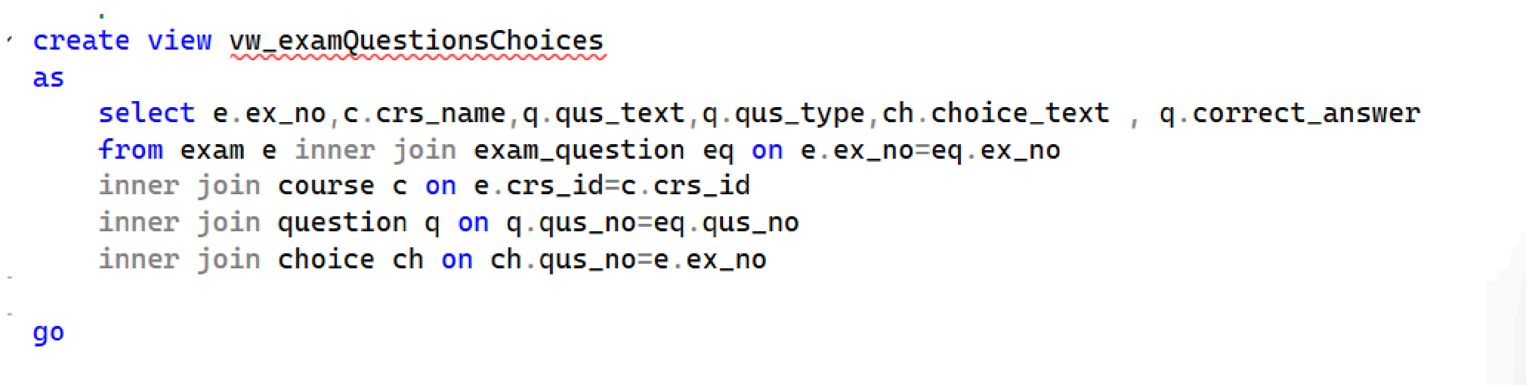
Multiple selections

- ☐ Select all
- ☑ 30
- ☑ 31
- ☑ 32

qus_text	Option A	Option B	Option C	Option D
C# is an OOP language	Option A	Option B	Option C	Option D
C# supports OOP	Option A	Option B	Option C	Option D
What does CLR stand for?	Option A	Option B	Option C	Option D

Overview: This report displays the questions (and their multiple-choice options) for one or more selected exams. When the user selects exam number(s) from the multi-select slicer (e.g., checking exams 30, 31, 32), the table automatically updates to show only the questions belonging to the chosen exam(s), along with the four answer options (A, B, C, D) for each question.

Implementation Explanation: I added a multi-select slicer using the Exam Number (ex_no) field, allowing users to choose one or multiple exams (with “Select all” and individual checkboxes). A table visual was created to display qus_text (question text) together with four calculated or pivoted columns: Option A, Option B, Option C, and Option D, which automatically filter to show only the questions and choices for the selected exam(s), making it simple to review complete exam content and answer options side by side.



```
create view vw_examQuestionsChoices
as
    select e.ex_no,c.crs_name,q.qus_text,q.qus_type,ch.choice_text , q.correct_answer
    from exam e inner join exam_question eq on e.ex_no=eq.ex_no
    inner join course c on e.crs_id=c.crs_id
    inner join question q on q.qus_no=eq.qus_no
    inner join choice ch on ch.qus_no=e.ex_no

go
```

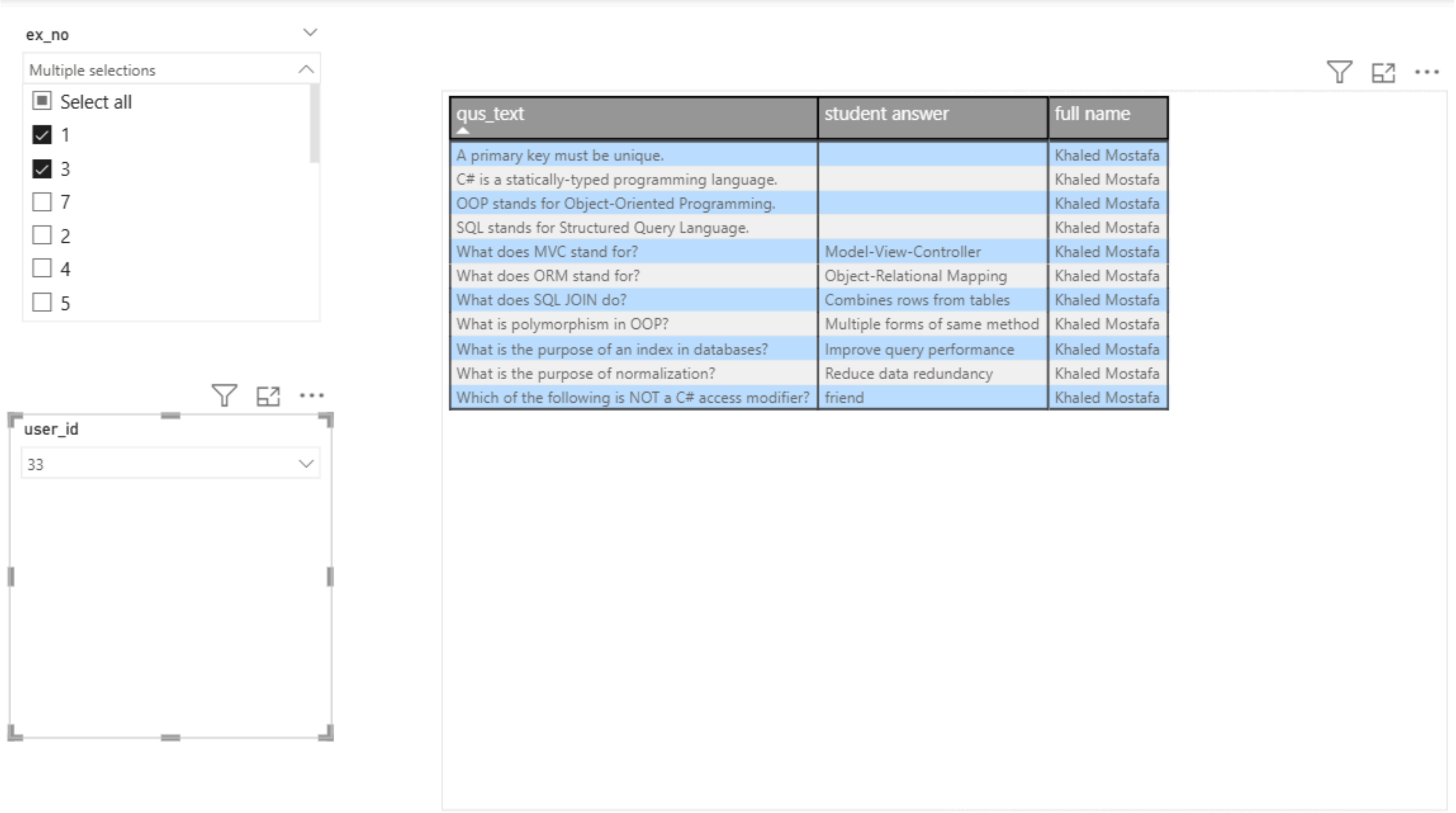
This view vw_examQuestionsChoices provides a denormalized, flattened structure that combines:

- Exam details (ex_no)
- The course name the exam belongs to (crs_name)
- Each question in the exam (qus_text, qus_type, correct_answer)
- All possible answer choices for that question (choice_text)

It joins the following tables to achieve this:

- exam → exam header
- exam_question → links exams to their questions
- course → to get the course name
- question → question text, type, and correct answer
- choice → the individual answer choices for each question

4.1.6 Student Exam Answer Report



ex_no

Multiple selections

- ▣ Select all
- ☑ 1
- ☑ 3
- □ 7
- □ 2
- □ 4
- □ 5

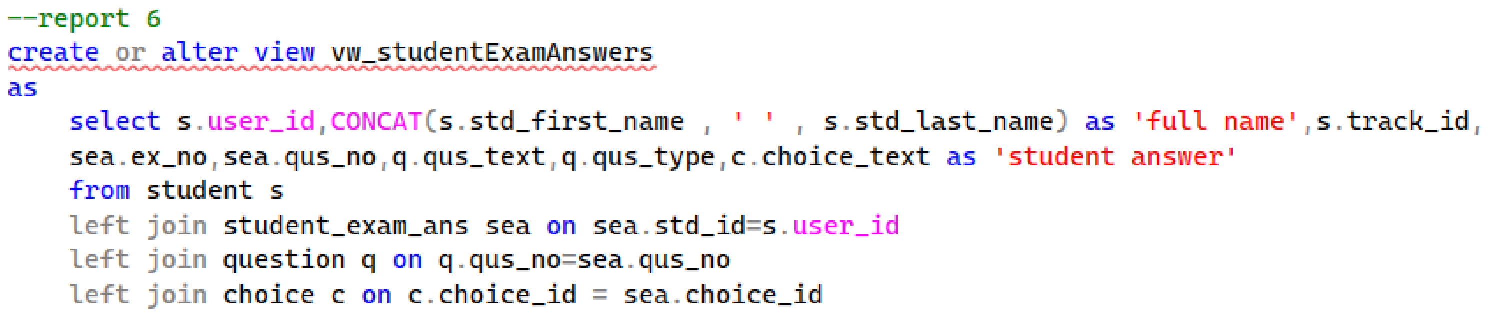
user_id

33

qus_text	student answer	full name
A primary key must be unique.		Khaled Mostafa
C# is a statically-typed programming language.		Khaled Mostafa
OOP stands for Object-Oriented Programming.		Khaled Mostafa
SQL stands for Structured Query Language.		Khaled Mostafa
What does MVC stand for?	Model-View-Controller	Khaled Mostafa
What does ORM stand for?	Object-Relational Mapping	Khaled Mostafa
What does SQL JOIN do?	Combines rows from tables	Khaled Mostafa
What is polymorphism in OOP?	Multiple forms of same method	Khaled Mostafa
What is the purpose of an index in databases?	Improve query performance	Khaled Mostafa
What is the purpose of normalization?	Reduce data redundancy	Khaled Mostafa
Which of the following is NOT a C# access modifier?	friend	Khaled Mostafa

Overview: This report shows the questions attempted by a selected student along with their submitted answers. When a student is chosen from the user_id slicer (or filter), the table automatically updates to display only the questions that student has answered, including the question text (qus_text), the student's selected answer (student answer), and the student's full name for context. It helps instructors or admins quickly review individual student responses across one or more exams.

Implementation Explanation: I created a slicer using the user_id field (with multiple selection possible), allowing users to filter by one or more students. A table visual was added to display the columns qus_text (question text), student answer (the choice text the student submitted), and full name (from the vw_studentExamAnswers view), which automatically filters to show only the answered questions for the selected student(s). Additional slicers/filters on ex_no (exam number) were included to narrow down to specific exams if needed, making it easy to inspect a student's performance or submitted answers in detail.



```
--report 6
create or alter view vw_studentExamAnswers
as
    select s.user_id,CONCAT(s.std_first_name , ' ' , s.std_last_name) as 'full name',s.track_id,
    sea.ex_no,sea.qus_no,q.qus_text,q.qus_type,c.choice_text as 'student answer'
    from student s
    left join student_exam_ans sea on sea.std_id=s.user_id
    left join question q on q.qus_no=sea.qus_no
    left join choice c on c.choice_id = sea.choice_id
```

This view vw_studentExamAnswers provides a denormalized, readable structure that combines:

- Student identification and personal details (user_id, full name, track_id)
- Exam and question context (ex_no, qus_no)
- The question text and type (qus_text, qus_type)
- The student's submitted answer (student answer – the choice text they selected)

It joins the following tables to achieve this:

- student → student personal information and full name construction
- student_exam_ans → student's submitted answers per exam/question
- question → question text and type
- choice → the text of the selected answer choice

4.2 Publishing to Power BI Service

1. **Prepared & Published**
   - Opened .pbix in Power BI Desktop
   - Verified data loads from local SQL Server (.\SQLEXPRESS, DB: ITIExaminationSystem)
   - Clicked **Home** → **Publish** → selected **My workspace**
2. **Fixed Cloud Connection Issue**
   - After publishing, data refresh failed (cloud can't reach local DB)
   - Installed **On-premises Data Gateway** on the same machine
   - Signed in with Power BI account → gateway registered successfully
3. **Created Gateway Connection**
   - In Power BI Service: **Settings** → **Manage gateways** → **New**
   - Settings:
     - Gateway: installed gateway
     - Name: ITI_Exam_SQLExpress
     - Server: localhost\SQLEXPRESS
     - Database: ITIExaminationSystem
     - Authentication: SQL Server Authentication
   - Saved and tested
4. **Mapped Semantic Model**
   - Went to **My workspace** → **Semantic model** → **Settings**
   - Under **Gateway and cloud connections** → selected the new gateway connection
   - Applied changes
5. **Verified Refresh**
   - Clicked **Refresh now** → succeeded
   - Report now shows live data in the browser
6. **Shared with Team**
   - In workspace → **Access**
   - Added teammates' emails → assigned **Viewer** or **Contributor** role
7. **PDF Export**
   - Manual: **File** → **Export** → **PDF** (full report)
   - Per-page button: Added Power Automate flow → **Export to file (PDF)** → linked to button on each page