Menofia
Branch

Students Exams Management System

Team 1

Done By

Hend Hany Elkadeem

Mahmoud Mohamed Elframawi

Mahmoud Ahmed Mohamed Enany

Abdelrahman Salah Amin

Ahmed Mamdouh Abd El-Ghany

Abstract

This project is a web-based Examination Management System designed to manage the complete exam process, from exam creation by instructors to exam submission and evaluation by students. The system follows a role-based structure that ensures clear separation of responsibilities and secure handling of academic data.

The system supports three main user roles: **Admin**, **Instructor**, and **Student**. Students can log in to view their assigned courses, take online exams, submit answers, and review their grades. Instructors are able to create exams, and add questions to the system. Administrators have full control over the platform, including managing student and instructor accounts.

The front-end is developed using **React**, providing a responsive and interactive user interface. The back-end is implemented using **Node.js**, which handles business logic, authentication, and communication with the database through secure APIs. **Microsoft SQL Server** is used to store and manage system data efficiently.

Contents

Database

System Requirements

1. The system shall store and manage **branch data**, including branch ID and branch name.

2. The system shall store and manage **department data**, including department ID, name, and location.

3. The system shall enforce that **each branch must be associated with one or more departments**, while a department may be associated with zero or more branches.

4. The system shall store and manage **instructor data**, including ID, name, email, password, hire date, degree, and salary.

5. The system shall enforce that **each instructor must belong to at least one department**, and each department must have one or more instructors.

6. The system shall support assigning **one instructor as a manager for each department**, while an instructor may manage zero or one department.

7. The system shall store and manage **student data**, including ID, name, email, password, and address.

8. The system shall enforce that **each student must be assigned to exactly one department**, while a department may have zero or more students.

9. The system shall store and manage **course data**, including course ID, name, duration, and multi-valued topics.

10. The system shall enforce that **each course belongs to exactly one department**, while each department must offer one or more courses.

11. The system shall support a **many-to-many relationship between students and courses**, where each student must be enrolled in one or more courses, and each course may have zero or more students.

12. The system shall store additional data for student–course enrollment, including **grade and assignment date**.

13. The system shall support a **many-to-many relationship between instructors and courses**, ensuring that each course must be taught by one or more instructors and each instructor may teach multiple courses.

14. The system shall store and manage **question data**, including question ID, question text, question type, default score, correct choice, and difficulty level.

15. The system shall allow each question to be associated with **one or more courses**, while a course may have zero or more questions.

16. The system shall store and manage **exam data**, including exam ID, total score, and duration.

17. The system shall enforce that **each exam must include multiple questions**, while a question may appear in zero or more exams.

18. The system shall support a **many-to-many relationship between students and exams**, where each student may take one or more exams and each exam must be taken by multiple students.

19. The system shall store **score per question** for each student in an exam.

20. The system shall store and manage **choice data**, including choice ID and choice text.

21. The system shall enforce that **each choice belongs to exactly one question**, and each question must have one or more choices.

22. The system shall store and manage **university data**, including university ID and name.

23. The system shall enforce that **each student must belong to exactly one university**, while a university may have many students.

24. The system shall store and manage **major data**, including major ID and name.

25. The system shall enforce that **each student must belong to exactly one major**, while a major may have many students.

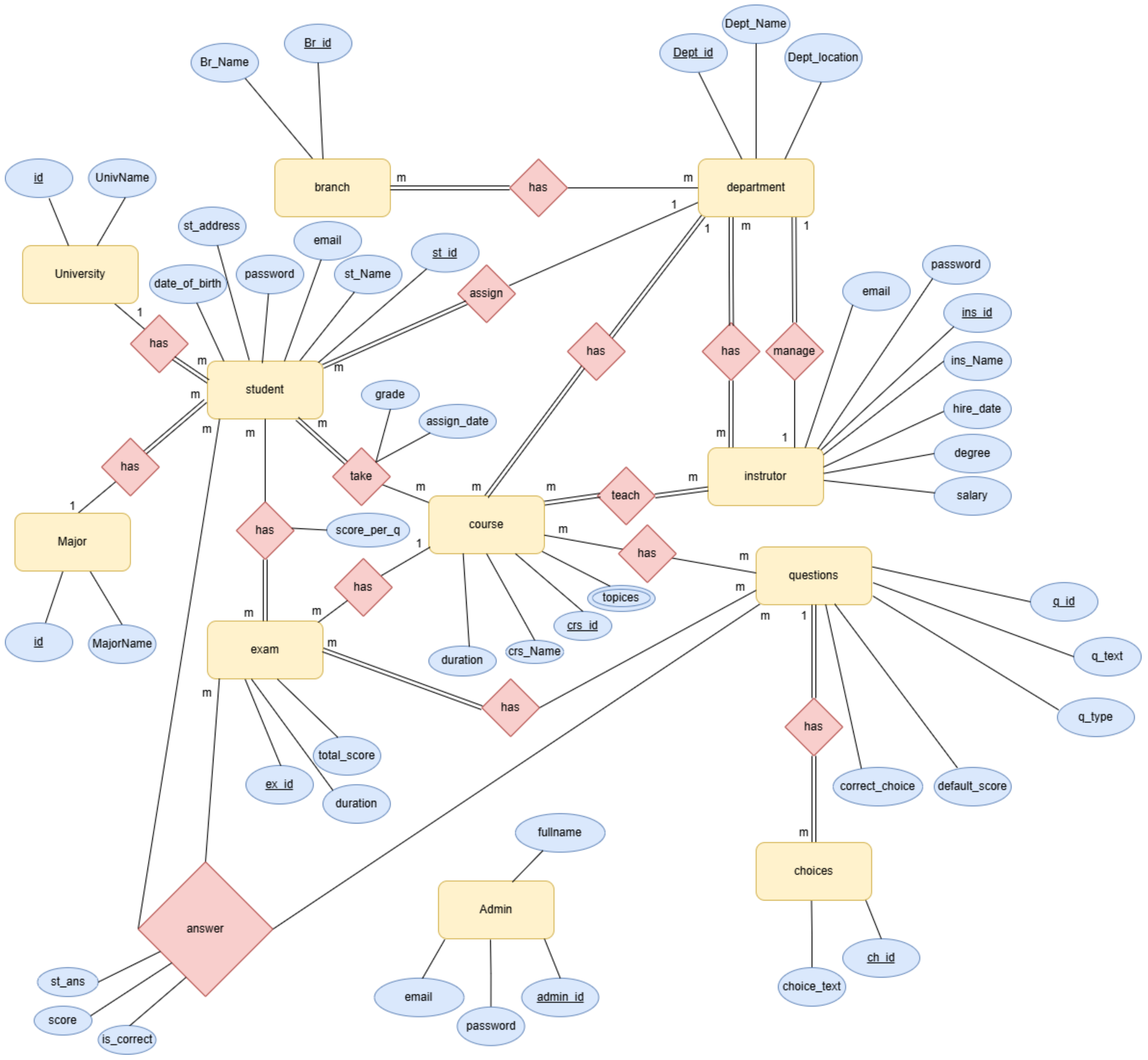
26. The system shall store and manage **student answers** for exams through an answer relationship that combines student, exam, and question.

27. The system shall store answer attributes including **student answer, score, and correctness status**.

28. The system shall store and manage **admin data**, including admin ID, full name, email, and password.

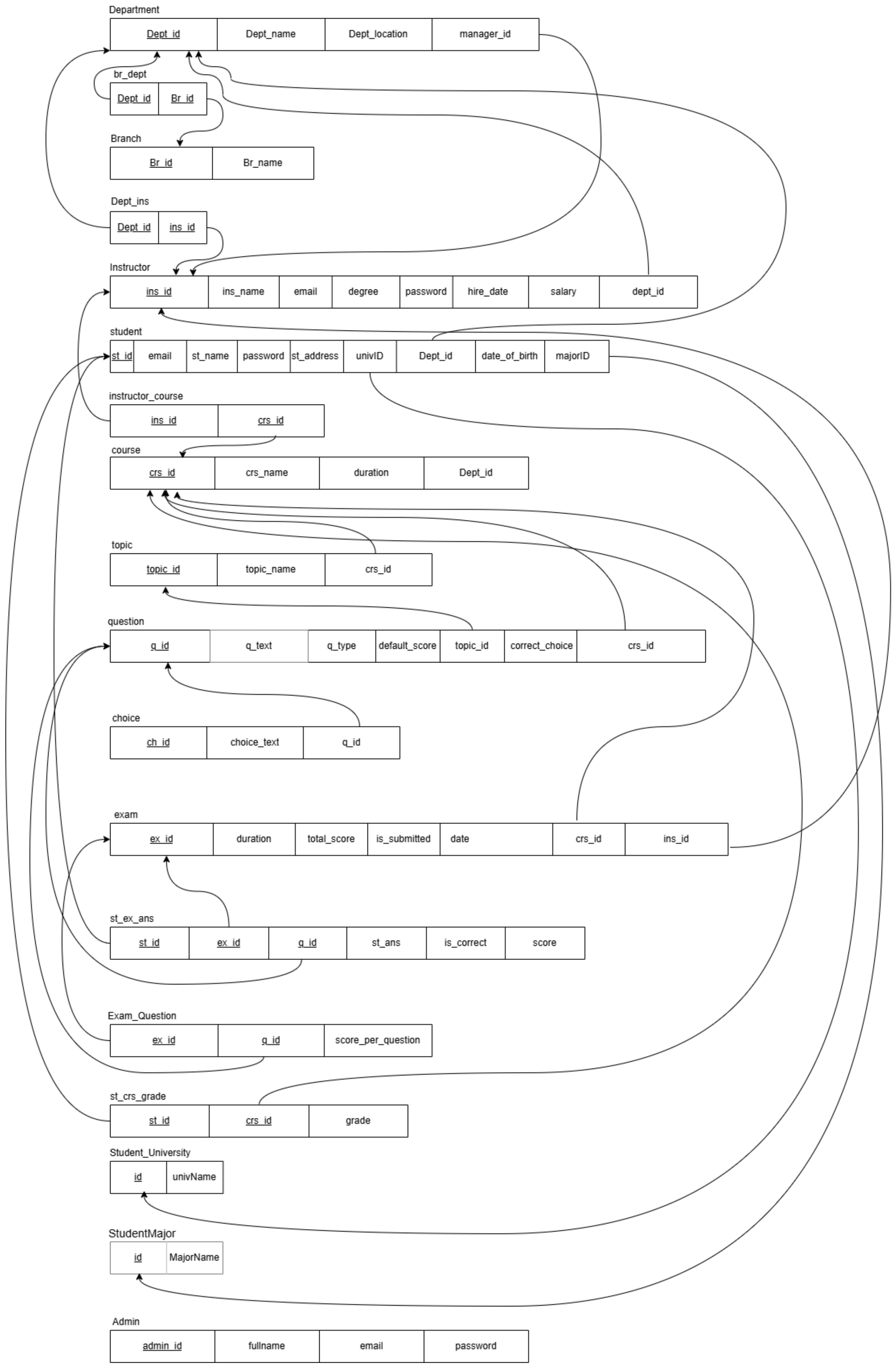
29. The system shall allow administrators to **manage system data** and oversee system operations.

ER Diagram



Br_Name
Br_id
branch
has
m
m
department
Dept_id
Dept_Name
Dept_location
1
1
m
1
id
UnivName
University
1
has
m
st_address
email
password
st_Name
st_id
date_of_birth
assign
m
student
has
has
manage
m
1
m
m
m
m
grade
assign_date
email
password
ins_id
ins_Name
hire_date
degree
salary
instrutor
has
1
Major
id
MajorName
take
m
m
m
course
teach
m
has
score_per_q
1
m
has
m
has
m
questions
topices
crs_id
crs_Name
duration
q_id
q_text
q_type
m
m
exam
m
has
m
1
has
total_score
ex_id
duration
correct_choice
default_score
m
choices
ch_id
choice_text
fullname
Admin
email
password
admin_id
answer
st_ans
score
is_correct

Mapping


Department
Dept_id | Dept_name | Dept_location | manager_id
br_dept
Dept_id | Br_id
Branch
Br_id | Br_name
Dept_ins
Dept_id | ins_id
Instructor
ins_id | ins_name | email | degree | password | hire_date | salary | dept_id
student
st_id | email | st_name | password | st_address | univID | Dept_id | date_of_birth | majorID
instructor_course
ins_id | crs_id
course
crs_id | crs_name | duration | Dept_id
topic
topic_id | topic_name | crs_id
question
q_id | q_text | q_type | default_score | topic_id | correct_choice | crs_id
choice
ch_id | choice_text | q_id
exam
ex_id | duration | total_score | is_submitted | date | crs_id | ins_id
st_ex_ans
st_id | ex_id | q_id | st_ans | is_correct | score
Exam_Question
ex_id | q_id | score_per_question
st_crs_grade
st_id | crs_id | grade
Student_University
id | univName
StudentMajor
id | MajorName
Admin
admin_id | fullname | email | password

Tables

The system contians 18 tables, we will move through them step by step

Admin

```
CREATE TABLE Admin
(
    admin_id   INT IDENTITY(1,1) PRIMARY KEY,
    email      NVARCHAR(100) NOT NULL UNIQUE,
    password   NVARCHAR(255) NOT NULL,
    full_name  NVARCHAR(100) NULL
);
```

br_dept

```
CREATE TABLE br_dept
(
    Dept_id INT NOT NULL,
    Br_id   INT NOT NULL,
    PRIMARY KEY (Dept_id, Br_id),
    FOREIGN KEY (Dept_id) REFERENCES Department(Dept_id),
    FOREIGN KEY (Br_id)   REFERENCES Branch(Br_id)
);
```

Branch

```
CREATE TABLE Branch
(
    Br_id   INT IDENTITY(1,1) PRIMARY KEY,
    Br_name NVARCHAR(50) NOT NULL
);
```

Choice

```
CREATE TABLE Choice
(
    ch_id        INT IDENTITY(1,1) PRIMARY KEY,
    choice_text  NVARCHAR(MAX) NOT NULL,
    q_id         INT NOT NULL,
    FOREIGN KEY (q_id) REFERENCES Question(q_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

Course

```
CREATE TABLE Course
(
    crs_id     INT PRIMARY KEY,
    crs_name   NVARCHAR(50) NULL,
    duration   INT NULL,
    Dept_id    INT NOT NULL,
    FOREIGN KEY (Dept_id) REFERENCES Department(Dept_id)
);
```

Dep_Ins

```
CREATE TABLE Dep_Ins
(
    Dept_id INT NOT NULL,
    ins_id  INT NOT NULL,
    PRIMARY KEY (Dept_id, ins_id),
    FOREIGN KEY (Dept_id) REFERENCES Department(Dept_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (ins_id) REFERENCES Instructor(ins_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

Department

```
CREATE TABLE Department
(
    Dept_id        INT PRIMARY KEY,
    Dept_name      NVARCHAR(50) NOT NULL,
    Dept_location NVARCHAR(50) NOT NULL,
    manager_id     INT NULL,
    FOREIGN KEY (manager_id) REFERENCES Instructor(ins_id));
```

Exam

```
CREATE TABLE Exam
(
    ex_id         INT IDENTITY(1,1) PRIMARY KEY,
    duration      INT NOT NULL,
    total_score   INT NOT NULL,
    date          DATETIME DEFAULT GETDATE(),
    crs_id        INT NOT NULL,
    ins_id        INT NOT NULL,
    is_submitted BIT DEFAULT 0,
    FOREIGN KEY (crs_id) REFERENCES Course(crs_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (ins_id) REFERENCES Instructor(ins_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

Exam_Question

```
CREATE TABLE Exam_Question
(
    ex_ID  INT NOT NULL,
    q_ID   INT NOT NULL,
    degree INT NOT NULL,
    PRIMARY KEY (ex_ID, q_ID),
    FOREIGN KEY (ex_ID) REFERENCES Exam(ex_id),
    FOREIGN KEY (q_ID)  REFERENCES Question(q_id)
);
```

Instructor

```
CREATE TABLE Instructor
(
    ins_id        INT IDENTITY(1,1) PRIMARY KEY,
    ins_name      VARCHAR(50) NOT NULL,
    degree        VARCHAR(20) NULL,
    hire_date     DATE DEFAULT GETDATE(),
    salary        FLOAT NOT NULL,
    Dept_id       INT NULL,
    ins_email    NVARCHAR(100) NULL,
    ins_password NVARCHAR(100) NULL,
    FOREIGN KEY (Dept_id) REFERENCES Department(Dept_id)
);

Instructor_course
CREATE TABLE Instructor_Course
(
    instructor_ID INT NOT NULL,
    course_ID     INT NOT NULL,
    PRIMARY KEY (instructor_ID, course_ID),
    FOREIGN KEY (instructor_ID) REFERENCES Instructor(ins_id),
    FOREIGN KEY (course_ID)     REFERENCES Course(crs_id)
);
```

Question

```
CREATE TABLE Question
(
    q_id           INT IDENTITY(1,1) PRIMARY KEY,
    q_text         NVARCHAR(MAX) NOT NULL,
    q_type         NVARCHAR(20) NOT NULL,
    default_score  FLOAT NOT NULL,
    correct_choice NVARCHAR(MAX) NOT NULL,
    crs_id         INT NULL,
    top_id         INT NOT NULL,
    FOREIGN KEY (crs_id) REFERENCES Course(crs_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (top_id) REFERENCES Topic(topic_id)
);
```

st_crs_grade

```
CREATE TABLE st_crs_grade
(
    st_id   INT NOT NULL,
    crs_id  INT NOT NULL,
    grade   INT NULL,
    PRIMARY KEY (st_id, crs_id),
    FOREIGN KEY (st_id)  REFERENCES Student(st_id),
    FOREIGN KEY (crs_id) REFERENCES Course(crs_id)
);
```

Student

```
CREATE TABLE Student
(
    st_id       INT IDENTITY(1,1) PRIMARY KEY,
    st_name     NVARCHAR(50) NOT NULL,
    st_address  NVARCHAR(100) NULL,
    Dept_id     INT NOT NULL,
    DateOfBirth DATE NULL,
    Univ_id     INT NULL,
    Major_id    INT NULL,
    st_email    NVARCHAR(100) NULL,
    st_password NVARCHAR(100) NULL,
    FOREIGN KEY (Dept_id)  REFERENCES Department(Dept_id) ON
UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (Major_id) REFERENCES StudentMajor(MajorID),
    FOREIGN KEY (Univ_id)  REFERENCES
Student_University(Univ_id)
);
```

Student_Exam_Answer

```
CREATE TABLE Student_Exam_Answer
(
    student_ID     INT NOT NULL,
    exam_ID        INT NOT NULL,
    question_ID    INT NOT NULL,
    student_Answer NVARCHAR(250) NOT NULL,
    isCorrect      BIT NULL,
    score          INT NULL,
    PRIMARY KEY (student_ID, exam_ID, question_ID),
    FOREIGN KEY (student_ID)  REFERENCES Student(st_id),
    FOREIGN KEY (exam_ID)     REFERENCES Exam(ex_id),
    FOREIGN KEY (question_ID) REFERENCES Question(q_id)
);
```

Student_University

```
CREATE TABLE Student_University
(
    Univ_id   INT IDENTITY(1,1) PRIMARY KEY,
    univ_name NVARCHAR(50) NOT NULL
);
```

StudentMajor

```
CREATE TABLE StudentMajor
(
    MajorID   INT IDENTITY(1,1) PRIMARY KEY,
    MajorName NVARCHAR(50) NOT NULL
);
```

Topic

```
CREATE TABLE Topic
(
    topic_id   INT PRIMARY KEY,
    topic_name NVARCHAR(50) NULL,
    crs_id     INT NOT NULL,
    FOREIGN KEY (crs_id) REFERENCES Course(crs_id)
);
```

Stored Procedures

Exam generation

```
CREATE OR ALTER PROC sp_generate_exam
@ex_id INT,
@mcq_cnt INT,
@tf_cnt INT,
@mode NVARCHAR(20)
AS
BEGIN
    DECLARE @easy_pct FLOAT, @med_pct FLOAT, @hard_pct FLOAT;

    IF @mode = 'hard'
    BEGIN
        SET @easy_pct = 0.1;
        SET @med_pct = 0.3;
        SET @hard_pct = 0.6;
    END
    ELSE IF @mode = 'easy'
    BEGIN
        SET @easy_pct = 0.6;
        SET @med_pct = 0.3;
        SET @hard_pct = 0.1;
    END
    ELSE
    BEGIN
        SET @easy_pct = 0.2;
        SET @med_pct = 0.6;
        SET @hard_pct = 0.2;
    END

    DECLARE @crs_id INT;
    SELECT @crs_id = crs_id FROM exam WHERE ex_id = @ex_id;

    INSERT INTO exam_question (ex_id, q_id, degree)
    SELECT TOP (CAST(CEILING(@mcq_cnt * @easy_pct) AS INT))
        @ex_id, q_id, default_score
    FROM question
    WHERE crs_id = @crs_id AND q_type = 'mcq' AND default_score
BETWEEN 1 AND 2
```

```
    ORDER BY NEWID();

    INSERT INTO exam_question (ex_id, q_id, degree)
    SELECT TOP (CAST(CEILING(@mcq_cnt * @med_pct) AS INT))
        @ex_id, q_id, default_score
    FROM question
    WHERE crs_id = @crs_id AND q_type = 'mcq' AND default_score
BETWEEN 3 AND 5
    ORDER BY NEWID();

    INSERT INTO exam_question (ex_id, q_id, degree)
    SELECT TOP (CAST(CEILING(@mcq_cnt * @hard_pct) AS INT))
        @ex_id, q_id, default_score
    FROM question
    WHERE crs_id = @crs_id AND q_type = 'mcq' AND default_score
> 5
    ORDER BY NEWID();

    INSERT INTO exam_question (ex_id, q_id, degree)
    SELECT TOP (CAST(CEILING(@tf_cnt * @easy_pct) AS INT))
        @ex_id, q_id, default_score
    FROM question
    WHERE crs_id = @crs_id AND q_type = 'tf' AND default_score
BETWEEN 1 AND 2
    ORDER BY NEWID();

    INSERT INTO exam_question (ex_id, q_id, degree)
    SELECT TOP (CAST(CEILING(@tf_cnt * @med_pct) AS INT))
        @ex_id, q_id, default_score
    FROM question
    WHERE crs_id = @crs_id AND q_type = 'tf' AND default_score
BETWEEN 3 AND 5
    ORDER BY NEWID();

    INSERT INTO exam_question (ex_id, q_id, degree)
    SELECT TOP (CAST(CEILING(@tf_cnt * @hard_pct) AS INT))
        @ex_id, q_id, default_score
    FROM question
    WHERE crs_id = @crs_id AND q_type = 'tf' AND default_score>5
    ORDER BY NEWID();
END;
```

Overview:

sp_generate_exam is a stored procedure that automatically generates an exam by selecting random MCQ and True/False questions based on difficulty mode.

Parameters:

- @ex_id: Exam ID
- @mcq_cnt: Number of MCQ questions
- @tf_cnt: Number of True/False questions
- @mode: Difficulty mode (easy, normal, hard)

Difficulty Distribution:

Easy Mode -> 60% Easy, 30% Medium, 10% Hard

Normal Mode -> 20% Easy, 60% Medium, 20% Hard

Hard Mode -> 10% Easy, 30% Medium, 60% Hard

Difficulty Classification:

Easy -> default_score between 1 and 2

Medium -> default_score between 3 and 5

Hard -> default_score > 5

Tables Used:

- exam (to get course ID)
- question (source of questions)
- exam_question (generated exam questions)

Randomization:

Uses ORDER BY NEWID() to randomize question selection.

Example Usage:

EXEC sp_generate_exam 5, 20, 10, 'hard';

Notes:

- No duplicate check if executed multiple times.

- Ensure sufficient questions exist per difficulty level.

Exam Answers

```
CREATE PROCEDURE [dbo].[sp_ExamAnswers]
(
    @Student_ID     INT,
    @Exam_ID        INT,
    @Question_ID    INT,
    @Student_Answer NVARCHAR(250)
)
AS
BEGIN
    INSERT INTO Student_Exam_Answer
    VALUES
    (
        @Student_ID,
        @Exam_ID,
        @Question_ID,
        @Student_Answer,
        NULL,
        NULL
    );
END
```

Overview:

The sp_ExamAnswers stored procedure is used to store a student’s answer for a specific question in a specific exam. It inserts the student’s response into the Student_Exam_Answer table.

Input Parameters:

Parameter Name	Description
@Student_ID	Unique identifier of the student
@Exam_ID	Unique identifier of the exam
@Question_ID	Unique identifier of the question
@Student_Answer	The answer selected or entered by the student

Logic Description :

1. Receives the student, exam, and question identifiers.
2. Stores the student’s answer without validation or correction.
3. Leaves correction-related columns empty for later processing.

Exam Correction

```
CREATE PROCEDURE [dbo].[sp_ExamCorrection]
(
    @student_ID INT,
    @exam_ID    INT
)
AS
BEGIN
    UPDATE SEA
    SET
        isCorrect = CASE
                        WHEN SEA.student_Answer =
Q.correct_choice THEN 1
                        ELSE 0
                    END,
        score = CASE
                    WHEN SEA.student_Answer = Q.correct_choice
THEN EQ.degree
                    ELSE 0
                END
    FROM Student_Exam_Answer SEA
    INNER JOIN Question Q ON SEA.question_ID = Q.q_id
    INNER JOIN Exam_Question EQ ON EQ.q_ID = Q.q_id
    WHERE SEA.student_ID = @student_ID
      AND SEA.exam_ID = @exam_ID;

    -- Calculate total score and maximum score
    DECLARE @totalScore INT;
    DECLARE @maxScore   INT;

    -- Assign total score obtained by the student
    SELECT @totalScore = SUM(score)
    FROM Student_Exam_Answer
    WHERE student_ID = @student_ID
      AND exam_ID = @exam_ID;

    -- Assign maximum possible score for the exam
    SELECT @maxScore = SUM(EQ.degree)
    FROM Exam_Question EQ
    INNER JOIN Question Q ON EQ.q_ID = Q.q_id
    WHERE EQ.ex_ID = @exam_ID;


    SELECT
        @student_ID AS Student_ID,
        @exam_ID AS Exam_ID,
        @totalScore AS Total_Score,
        @maxScore AS Full_Mark,
        CAST((@totalScore * 100.0 / @maxScore) AS DECIMAL(5, 2))
AS Percentage;
END
```

Overview :

The sp_ExamCorrection stored procedure is responsible for:

- Correcting the student’s exam automatically.
- Updating correctness and score per question.
- Calculating the total score and percentage result.

Input Parameters :

Parameter Name	**Data Type**	**Description**
@student_ID	int	Unique identifier of the student
@exam_ID	int	Unique identifier of the exam

Correction Logic (Question Level)

This part compares the student's answers with the correct answers stored in the Question table.

Actions Performed:

- Updates the isCorrect column:
  - 1 if the student answer matches the correct choice.
  - 0 otherwise.
- Updates the score column:
  - Assigns default_score if the answer is correct.
  - Assigns 0 if the answer is incorrect.

This is achieved using:

- CASE expressions for conditional logic.
- INNER JOIN between Student_Exam_Answer and Question tables.

Reports

Report 1

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

GetStudentsByDeptID

```
CREATE PROCEDURE GetStudentsByDeptID @dept_ID INT
AS
BEGIN
     SELECT S.st_id[ID], S.st_name[Student Name], S.st_address
[Government], D.Dept_name[Department],
     U.univ_name[University], M.MajorName[Major],
YEAR(GETDATE())- YEAR(DateOfBirth)[Age]
     FROM student S
     inner join student_university U
          ON S.Univ_id = U.Univ_id
          inner join StudentMajor M
          ON S.Major_id = M.MajorID
          inner join Department D
          ON D.Dept_id = S.Dept_id
          WHERE D.Dept_id = @dept_ID
END
```

Purpose: Retrieve all students who belong to a specific department.

Execution behavior:

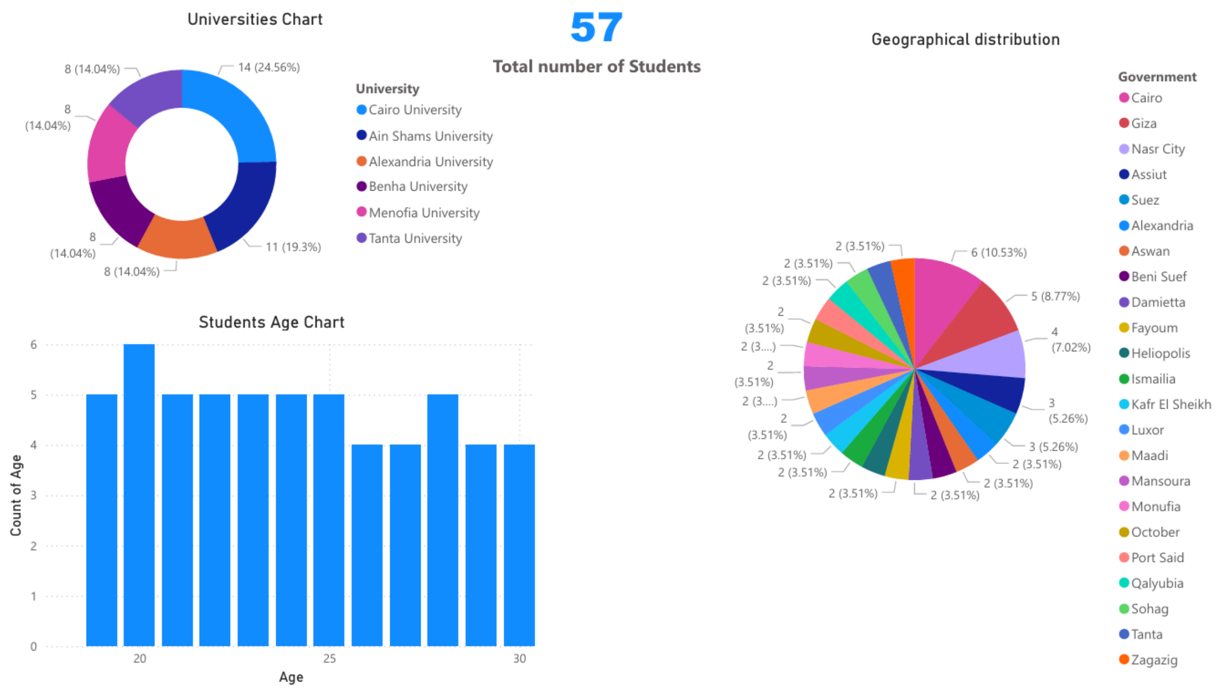
- The procedure takes a department ID.
- It looks in the student table for all students linked to that department.
- It joins with other tables (student_university, StudentMajor, Department) to get extra details like university name, major name, and department name.
- It also calculates the student's age by subtracting the year of birth from the current year.

Input:

- @dept_ID → the department ID you want to filter by.

Output: A result set with these columns:

- Student ID
- Student Name
- Address (Government)
- Department Name
- University Name
- Major Name
- Age (calculated from DateOfBirth).



Report 1 : Get Student By Department ID

Report 2

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

SP_Student_Grades

```
CREATE PROCEDURE [dbo].[SP_Student_Grades]
(
    @St_id INT
)
AS
BEGIN
    SET NOCOUNT ON;

    SELECT
        c.crs_id,
        c.crs_name,
        sc.grade,
        CASE
            WHEN sc.grade >= 60 THEN 'Pass'
            ELSE 'Fail'
        END AS status
    FROM st_crs_grade sc
    INNER JOIN Course c
        ON sc.crs_id = c.crs_id
    WHERE sc.st_id = @St_id;
END
```

Purpose: Retrieve all grades for a specific student across all courses.

Execution behavior:

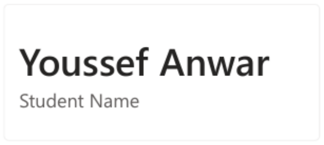
- When you run this procedure, the database looks into the st_crs_grade table (where student grades are stored).
- It matches the student ID you provide (@St_id).
- It joins with the Course table to get the course names.
- It also joins with the student table to get the student's name.
- Finally, it returns a list of all courses that student has taken, with the grade in each.

Input:

- @St_id → the student's ID (INT).

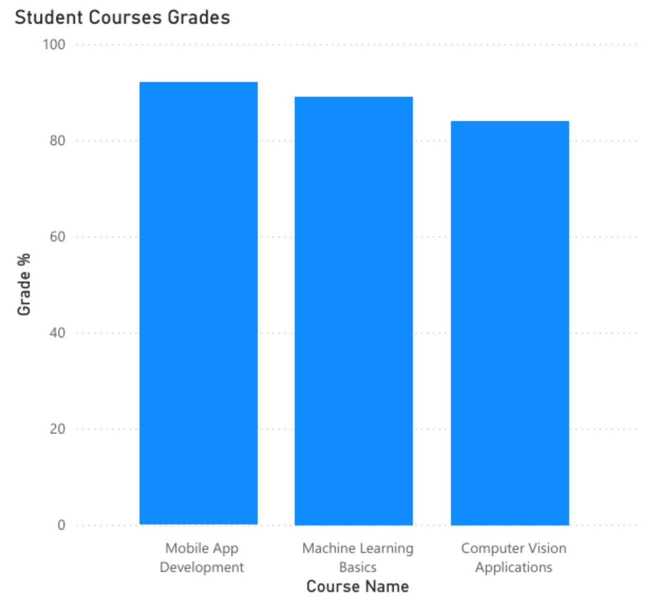
Output: A result set with four columns:

- Student ID
- Student Name
- Course Name
- Grade (percentage or score).



Youssef Anwar

Student Name



Reposrt 2 : Student Grade in all courses

Report 3

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

SP_InstructorCoursesReport

```
CREATE PROCEDURE [dbo].[SP_InstructorCoursesReport]
@InstructorID INT
AS
    SELECT
        I.ins_name [Instructor],c.crs_name AS CourseName,
        COUNT(scg.st_id) AS StudentCount
    FROM instructor_Course ic INNER JOIN Course c
        ON ic.course_ID = c.crs_id
    LEFT JOIN st_crs_grade scg
        ON c.crs_id = scg.crs_id
          inner join Instructor I
          on ic.instructor_ID = I.ins_id
    WHERE ic.instructor_ID = @InstructorID
    GROUP BY c.crs_name, I.ins_name;
```

Purpose: Generate a report of all courses taught by a specific instructor, along with the number of students enrolled in each course.

Execution behavior:

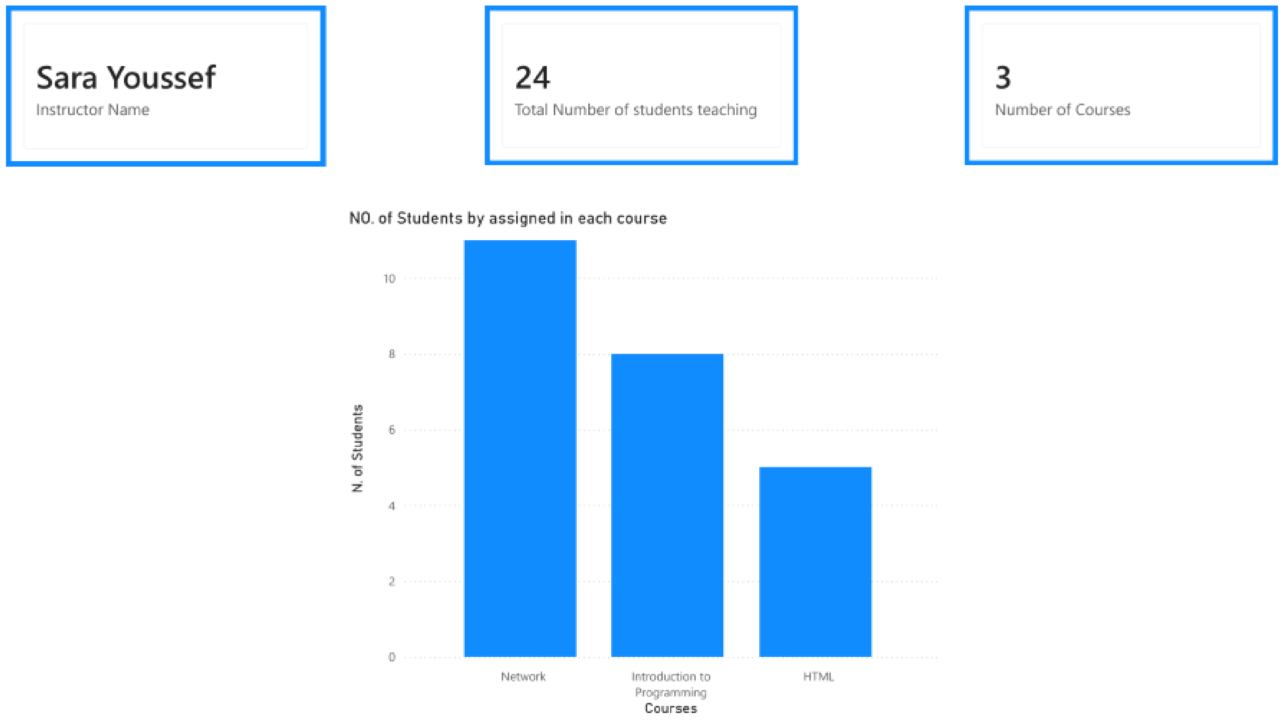
- The procedure starts from the instructor_Course table to find which courses are linked to the instructor ID you provide.
- It joins with the Course table to get the course names.
- It joins with the Instructor table to get the instructor's name.
- It uses a **LEFT JOIN** with st_crs_grade to count how many students are enrolled in each course (even if some courses have zero students, they will still appear).
- Finally, it groups the results by course name and instructor name so you get one row per course.

Input:

- @InstructorID → the instructor's ID (INT).

Output: A result set with three columns:

- Instructor name
- Course name
- Student count (number of students enrolled in that course).


Sara Youssef
Instructor Name

24
Total Number of students teaching

3
Number of Courses

Report 3 : Get courses names that an instructor teaches

Report 4

Report that takes course ID and returns its topics

SP_CourseTopicsReport

```
CREATE PROCEDURE [dbo].[SP_CourseTopicsReport]
    @CourseID INT
AS
    SELECT
        c.crs_name AS CourseName,
        t.topic_name AS TopicName
    FROM Course c
    INNER JOIN Topic t
        ON c.crs_id = t.crs_id
    WHERE c.crs_id = @CourseID;
```

Purpose: Retrieve all topics that belong to a specific course.

Execution behavior:

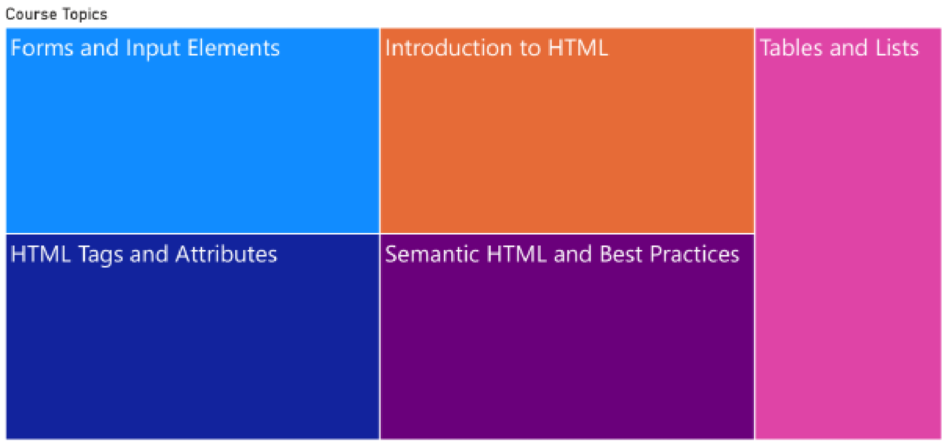
- The procedure takes a course ID (@CourseID).
- It looks in the Course table to find the course.
- It joins with the Topic table to get all topics linked to that course.
- It then returns one row for each topic inside that course.

Input:

- @CourseID → the ID of the course you want to see topics for.

Output: A result set with two columns:

- Course name
- Topic name



Report 4 : Topics in Course

Report 5

Report that takes exam number and returns the Questions in it and choices

SP_ExamQuestionsChoices

```
CREATE PROCEDURE [dbo].[SP_ExamQuestionsChoices] @ExamID INT
AS
BEGIN
    SELECT
        e.ex_ID,
        q.q_id,
        q.q_text,
        q.q_type,
        q.correct_choice,
        STRING_AGG(
            CASE WHEN c.choice_text <> q.correct_choice THEN
c.choice_text END, ', '
        ) WITHIN GROUP (ORDER BY c.ch_id) AS WrongChoices
    FROM Exam_Question e
    INNER JOIN Question q
        ON e.q_ID = q.q_id
    LEFT JOIN Choice c
        ON q.q_id = c.q_id
    WHERE e.ex_ID = @ExamID
    GROUP BY e.ex_ID, q.q_id, q.q_text,
q.correct_choice,q.q_type;
END;
```

Purpose: Retrieve all questions of a specific exam, along with their correct answer and all wrong choices.

Execution behavior:

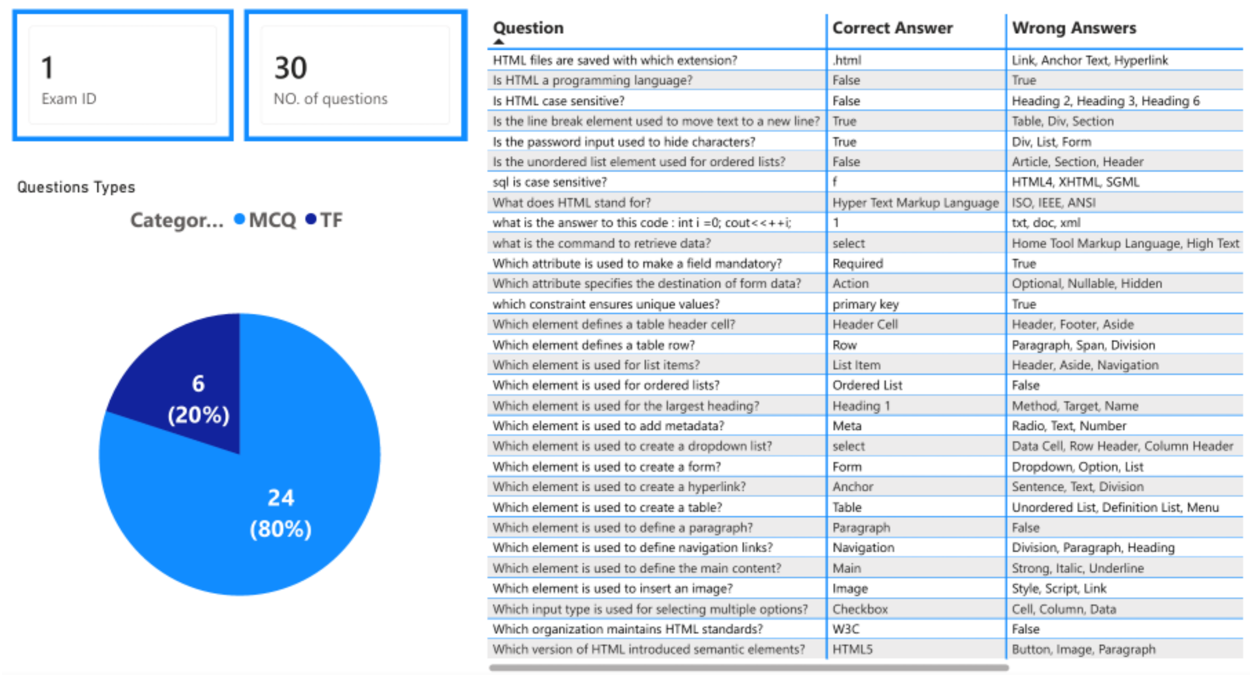
- The procedure takes an exam ID (@ExamID).
- It looks in the Exam_Question table to find all questions linked to that exam.
- It joins with the Question table to get the question text, type, and correct choice.
- It joins with the Choice table to collect all possible choices.
- It uses STRING_AGG to combine all wrong choices into one string, separated by commas, ordered by choice ID.
- Finally, it groups the results so you get one row per question with its correct and wrong choices.

Input:

- @ExamID → the exam's ID (INT).

Output: A result set with:

- Exam ID
- Question ID
- Question text
- Question type (MCQ, True/False, etc.)
- Correct choice
- Wrong choices (all combined in one column).



Question	Correct Answer	Wrong Answers
HTML files are saved with which extension?	.html	Link, Anchor Text, Hyperlink
Is HTML a programming language?	False	True
Is HTML case sensitive?	False	Heading 2, Heading 3, Heading 6
Is the line break element used to move text to a new line?	True	Table, Div, Section
Is the password input used to hide characters?	True	Div, List, Form
Is the unordered list element used for ordered lists?	False	Article, Section, Header
sql is case sensitive?	f	HTML4, XHTML, SGML
What does HTML stand for?	Hyper Text Markup Language	ISO, IEEE, ANSI
what is the answer to this code : int i =0; cout<<++i;	1	txt, doc, xml
what is the command to retrieve data?	select	Home Tool Markup Language, High Text
Which attribute is used to make a field mandatory?	Required	True
Which attribute specifies the destination of form data?	Action	Optional, Nullable, Hidden
which constraint ensures unique values?	primary key	True
Which element defines a table header cell?	Header Cell	Header, Footer, Aside
Which element defines a table row?	Row	Paragraph, Span, Division
Which element is used for list items?	List Item	Header, Aside, Navigation
Which element is used for ordered lists?	Ordered List	False
Which element is used for the largest heading?	Heading 1	Method, Target, Name
Which element is used to add metadata?	Meta	Radio, Text, Number
Which element is used to create a dropdown list?	select	Data Cell, Row Header, Column Header
Which element is used to create a form?	Form	Dropdown, Option, List
Which element is used to create a hyperlink?	Anchor	Sentence, Text, Division
Which element is used to create a table?	Table	Unordered List, Definition List, Menu
Which element is used to define a paragraph?	Paragraph	False
Which element is used to define navigation links?	Navigation	Division, Paragraph, Heading
Which element is used to define the main content?	Main	Strong, Italic, Underline
Which element is used to insert an image?	Image	Style, Script, Link
Which input type is used for selecting multiple options?	Checkbox	Cell, Column, Data
Which organization maintains HTML standards?	W3C	False
Which version of HTML introduced semantic elements?	HTML5	Button, Image, Paragraph

Report 5 : Exam Questions & Choices

Report 6

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

GetExamStudentAnswers

```
CREATE PROCEDURE [dbo].[GetExamStudentAnswers]
(
     @STUDENT_ID INT,
     @EXAM_ID INT
)
AS
BEGIN
     SELECT SEA.student_ID AS [Student ID], S.st_name AS
[Student Name],
          SEA.exam_ID AS [Exam ID], SEA.question_ID AS [Question
ID], Q.q_text AS [Question ], Q.q_type AS [Question Type],
          Q.correct_choice AS [Correct Answer],
SEA.student_Answer AS [student Answer], SEA.isCorrect AS
[Result],
          SEA.score AS [Student Score]
     FROM Student_Exam_Answer SEA
     INNER JOIN Question Q
     ON SEA.question_ID = Q.q_id
     INNER JOIN Student S
     ON S.st_id = SEA.student_ID
END
```

Purpose: Retrieve detailed information about a student's answers in a specific exam.

Execution behavior:

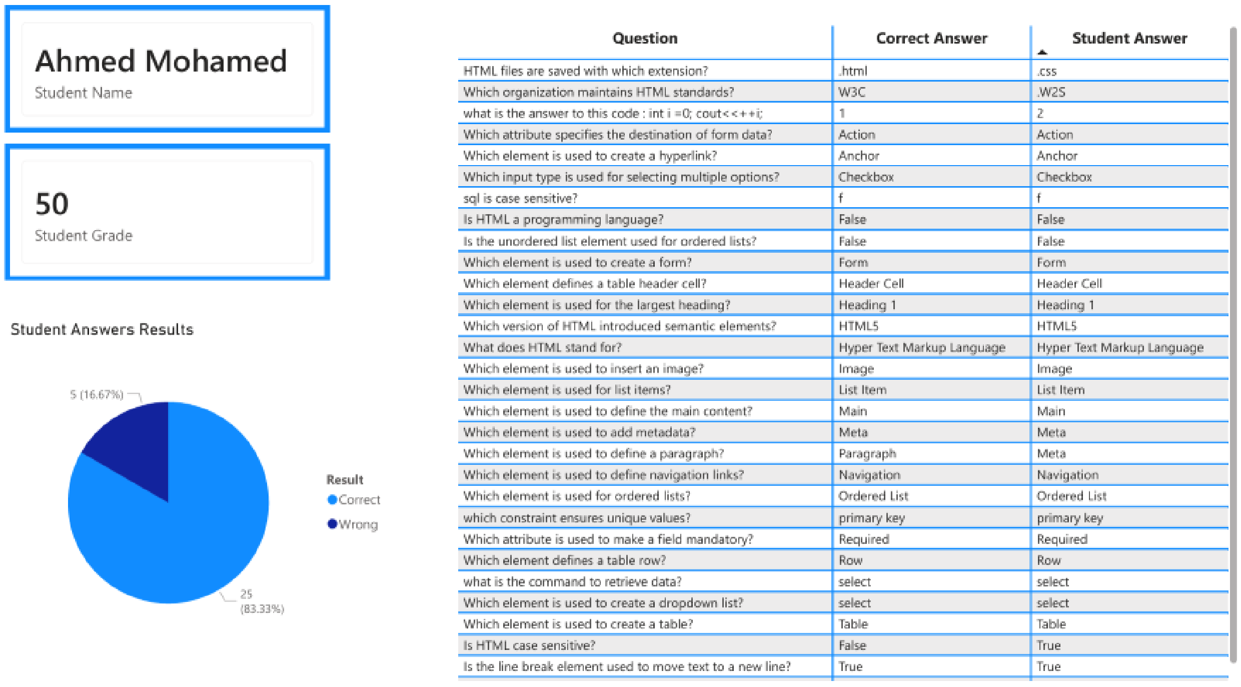
- The procedure takes two parameters: student ID and exam ID.
- It looks in the Student_Exam_Answer table to find all answers that student gave in that exam.
- It joins with the Question table to get the question text, type, and correct answer.
- It joins with the Student table to get the student's name.
- It then returns one row per question, showing both the student's answer and whether it was correct.

Input:

- @STUDENT_ID → the student's ID (INT).
- @EXAM_ID → the exam's ID (INT).

Output: A result set with:

- Student ID
- Student Name
- Exam ID
- Question ID
- Question text
- Question type (MCQ, True/False, etc.)
- Correct answer
- Student's answer
- Result (whether the answer is correct or not)
- Student's score for that question



Ahmed Mohamed
Student Name

50
Student Grade

Student Answers Results

Question	Correct Answer	Student Answer
HTML files are saved with which extension?	.html	.css
Which organization maintains HTML standards?	W3C	.W2S
what is the answer to this code : int i =0; cout<<++i;	1	2
Which attribute specifies the destination of form data?	Action	Action
Which element is used to create a hyperlink?	Anchor	Anchor
Which input type is used for selecting multiple options?	Checkbox	Checkbox
sql is case sensitive?	f	f
Is HTML a programming language?	False	False
Is the unordered list element used for ordered lists?	False	False
Which element is used to create a form?	Form	Form
Which element defines a table header cell?	Header Cell	Header Cell
Which element is used for the largest heading?	Heading 1	Heading 1
Which version of HTML introduced semantic elements?	HTML5	HTML5
What does HTML stand for?	Hyper Text Markup Language	Hyper Text Markup Language
Which element is used to insert an image?	Image	Image
Which element is used for list items?	List Item	List Item
Which element is used to define the main content?	Main	Main
Which element is used to add metadata?	Meta	Meta
Which element is used to define a paragraph?	Paragraph	Meta
Which element is used to define navigation links?	Navigation	Navigation
Which element is used for ordered lists?	Ordered List	Ordered List
which constraint ensures unique values?	primary key	primary key
Which attribute is used to make a field mandatory?	Required	Required
Which element defines a table row?	Row	Row
what is the command to retrieve data?	select	select
Which element is used to create a dropdown list?	select	select
Which element is used to create a table?	Table	Table
Is HTML case sensitive?	False	True
Is the line break element used to move text to a new line?	True	True

Report 6 : Student Answers in Exam

Interface

Technology Stack

Frontend Libraries & Tools

- **React 19.2.0** – Component-based UI and state management
- **Vite 7.2.4** – Fast development server and build tool
- **Tailwind CSS 3.4.19** – Utility-first CSS framework
- **Axios 1.13.2** – HTTP client for API communication
- **React Router DOM 7.11.0** – Client-side routing

User Roles

1. Admin

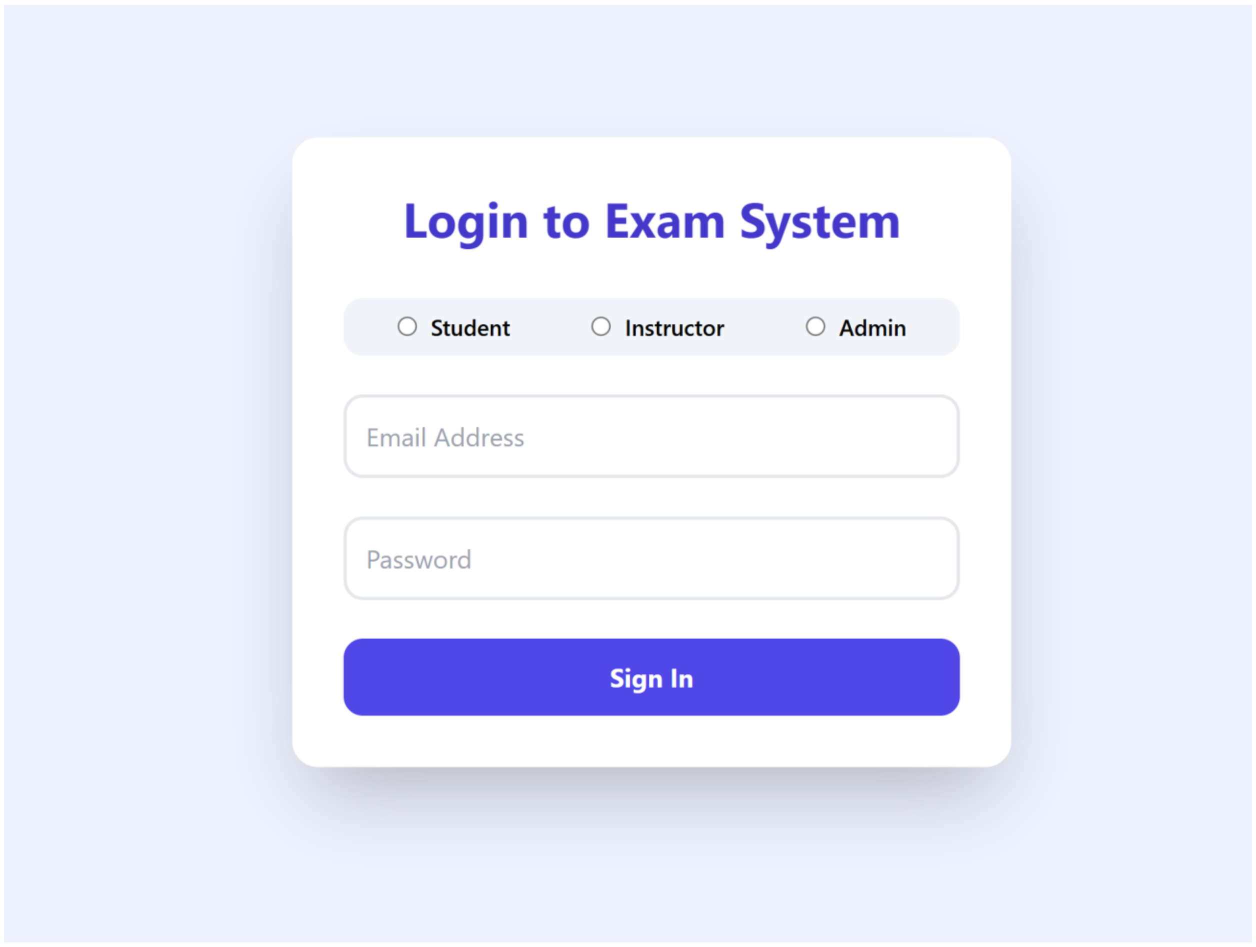
- Assign students to courses
- Display exams

2. Instructor

- Create exams
- Generate and regenerate questions
- Update exam settings (duration / question counts)
- Submit and finalize exams

3. Student

- Access assigned exams
- Take exams within a controlled time window
- Submit answers manually or automatically when time ends


Login to Exam System
Student
Instructor
Admin
Email Address
Password
Sign In

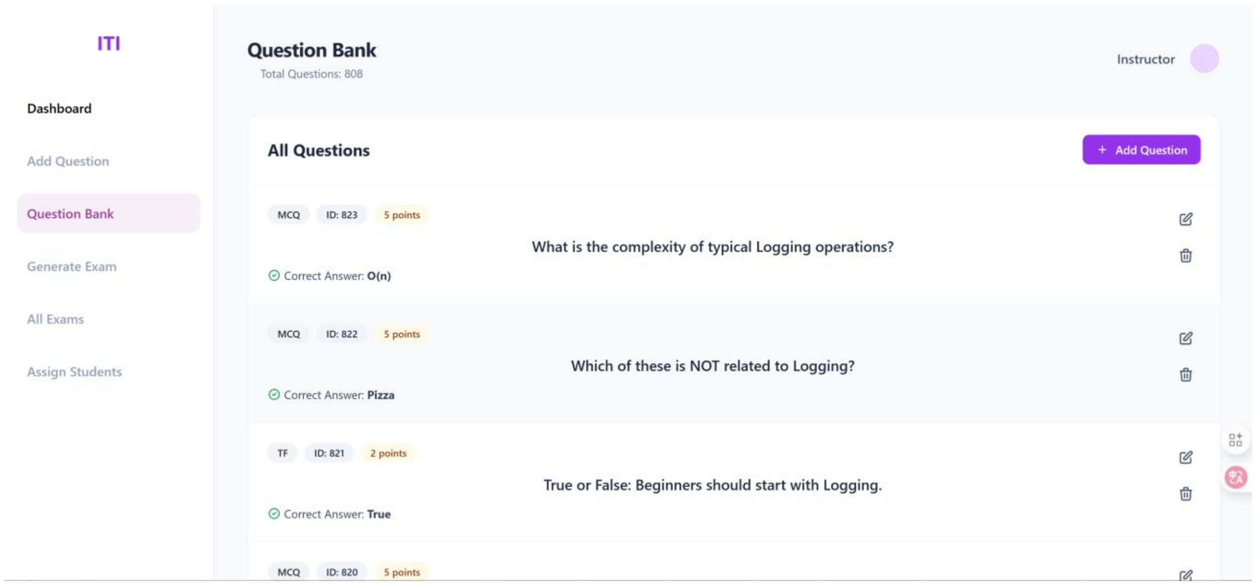
Login Page

Questions Management

Instructors have the ability to manage questions within the system, including adding new questions, editing existing ones, and deleting questions when needed. Each question can be categorized by its type, such as multiple-choice, **true/false**, or **short-answer**, allowing for better organization and flexible exam creation.

APIs for Questions management:

- `POST /api/questions` – Add a question
- `PUT /api/questions/:id` – Update a question
- `DELETE /api/questions/:id` – Delete a question
- `GET /api/questions/:examId` – Get all questions



Question Bank
Total Questions: 808

All Questions

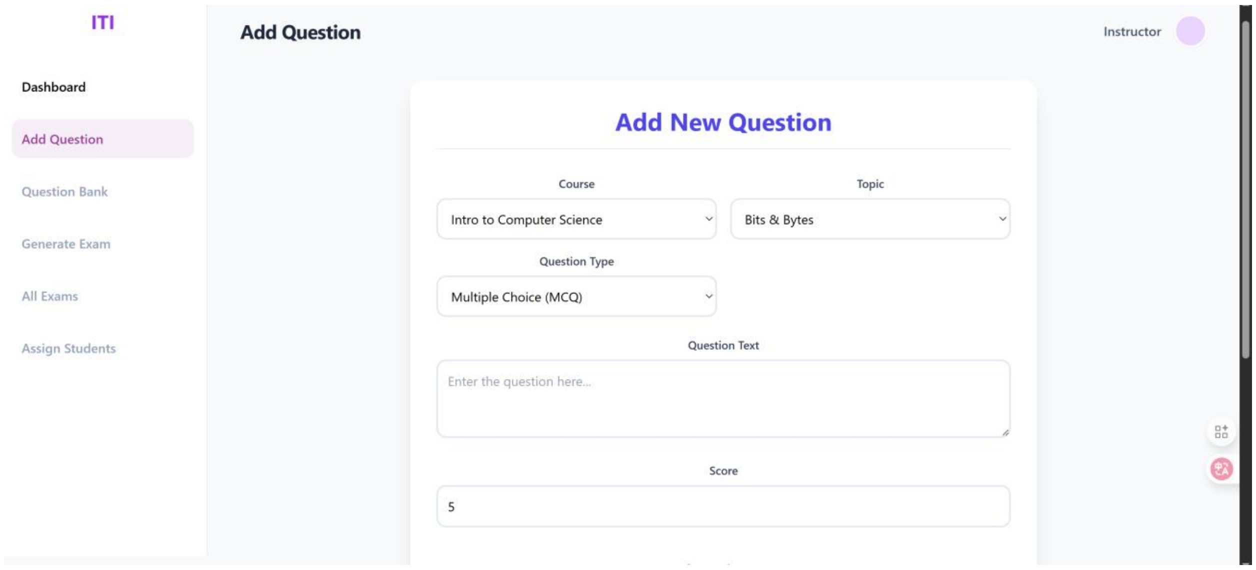
MCQ ID: 823 5 points
What is the complexity of typical Logging operations?
Correct Answer: O(n)

MCQ ID: 822 5 points
Which of these is NOT related to Logging?
Correct Answer: Pizza

TF ID: 821 2 points
True or False: Beginners should start with Logging.
Correct Answer: True

MCQ ID: 820 5 points

Question Bank



ITI

Dashboard

Add Question

Question Bank

Generate Exam

All Exams

Assign Students

Add Question

Instructor

Add New Question

Course: Intro to Computer Science

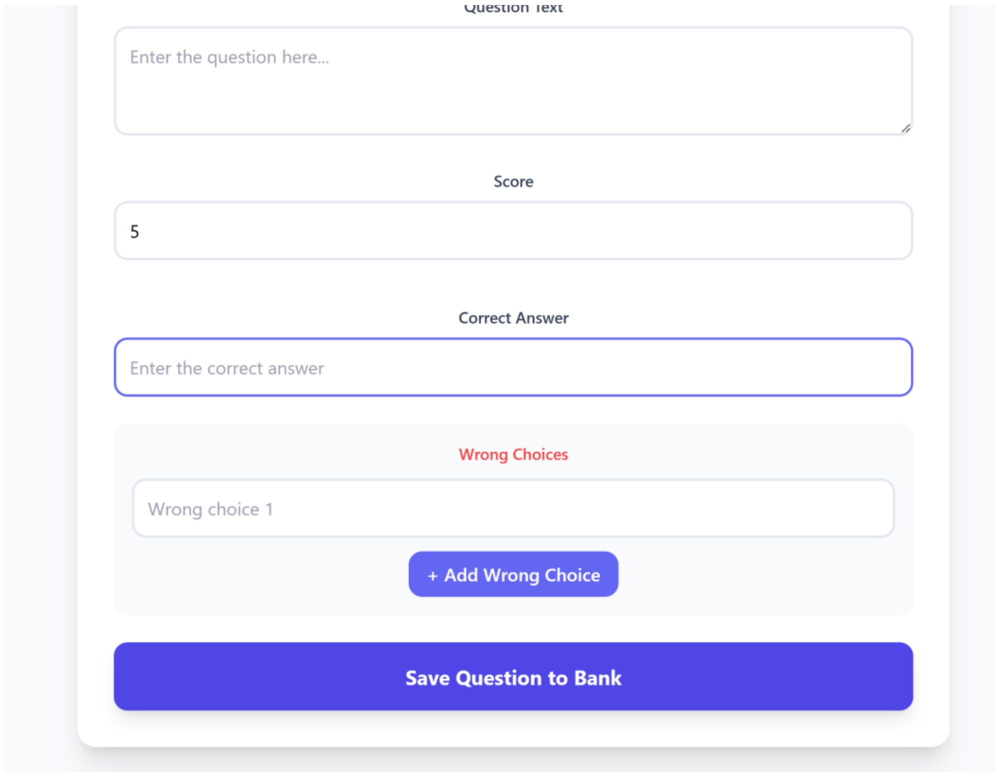
Topic: Bits & Bytes

Question Type: Multiple Choice (MCQ)

Question Text: Enter the question here...

Score: 5

Adding New Question to the Question Bank



Question Text: Enter the question here...

Score: 5

Correct Answer: Enter the correct answer

Wrong Choices

Wrong choice 1

+ Add Wrong Choice

Save Question to Bank

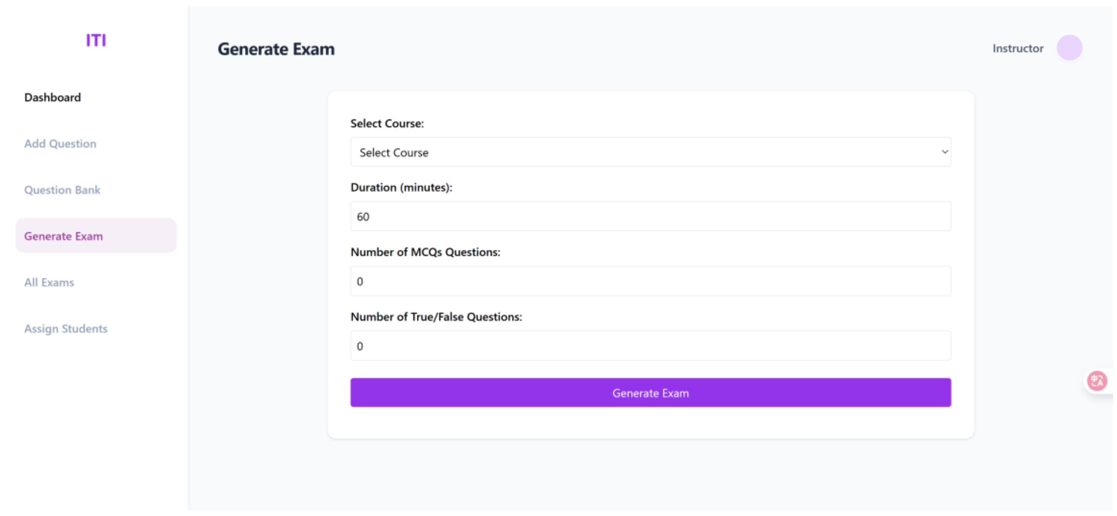
Inserting Answers to Question

Exam

Exam Lifecycle (Core Flow)

Step 1: Create Empty Exam

1. Instructor selects course and duration
2. Frontend sends: POST /api/exams
3. Backend creates an empty exam record



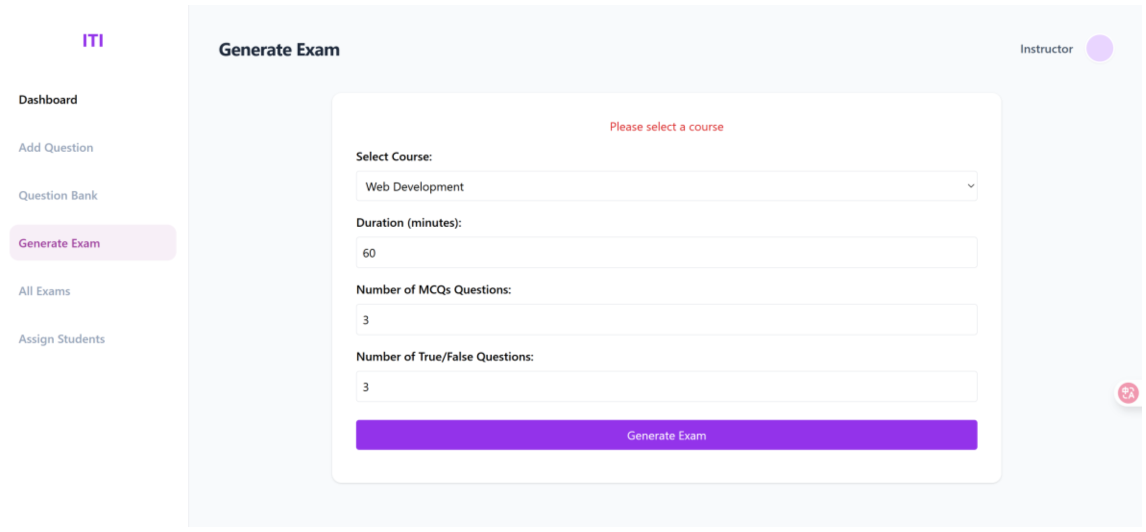
Select Course: Select Course

Duration (minutes): 60

Number of MCQs Questions: 0

Number of True/False Questions: 0

Generating Exam



Please select a course

Select Course: Web Development

Duration (minutes): 60

Number of MCQs Questions: 3

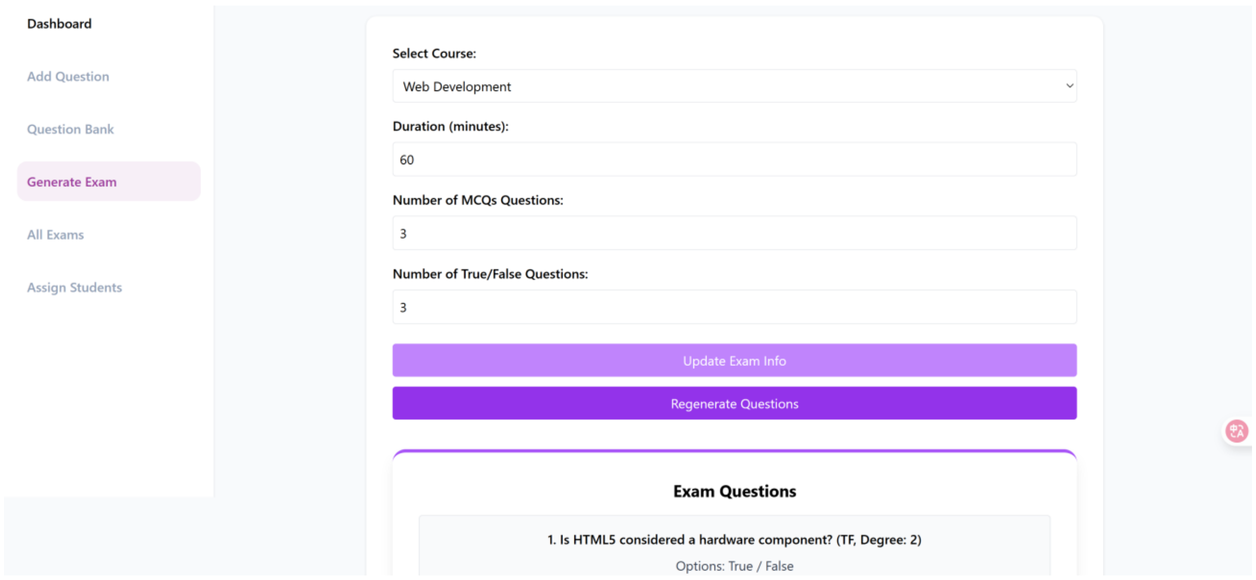
Number of True/False Questions: 3

Choosing number and type of questions

Step 2: Generate Exam Questions

1. Instructor specifies number of MCQ and True/False questions
2. Frontend sends: POST /api/exams/generate
3. Questions are attached to the existing exam

Important: Frontend **does not regenerate automatically** to avoid duplicated questions.



Select Course: Web Development

Duration (minutes): 60

Number of MCQs Questions: 3

Number of True/False Questions: 3

Exam Questions

1. Is HTML5 considered a hardware component? (TF, Degree: 2)

Options: True / False

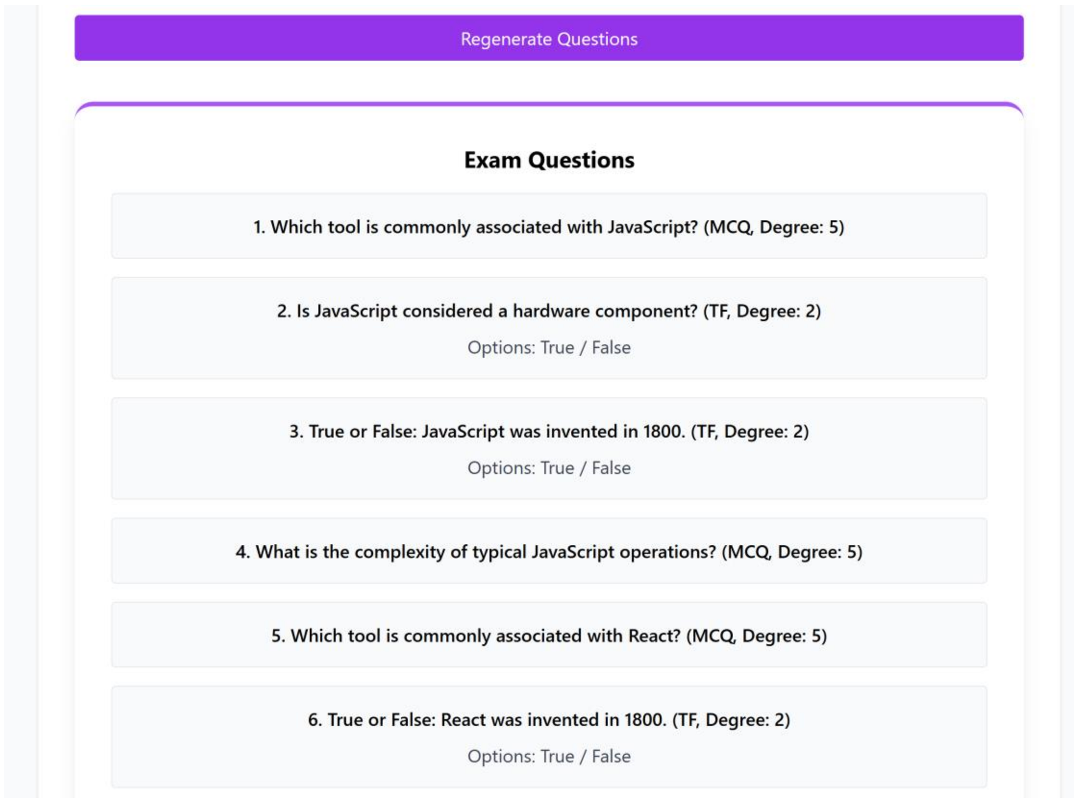
Generated Exam

Step 3: Update Exam

1. Instructor updates:
   - Duration
   - MCQ / TF counts
2. Frontend sends: PUT /api/exams/:id

Step 4: Regenerate Exam

1. Instructor may regenerate questions manually
2. Old questions are replaced (not appended)
3. Controlled action to avoid accidental duplication


Regenerate Questions

Exam Questions

1. Which tool is commonly associated with JavaScript? (MCQ, Degree: 5)

2. Is JavaScript considered a hardware component? (TF, Degree: 2)
Options: True / False

3. True or False: JavaScript was invented in 1800. (TF, Degree: 2)
Options: True / False

4. What is the complexity of typical JavaScript operations? (MCQ, Degree: 5)

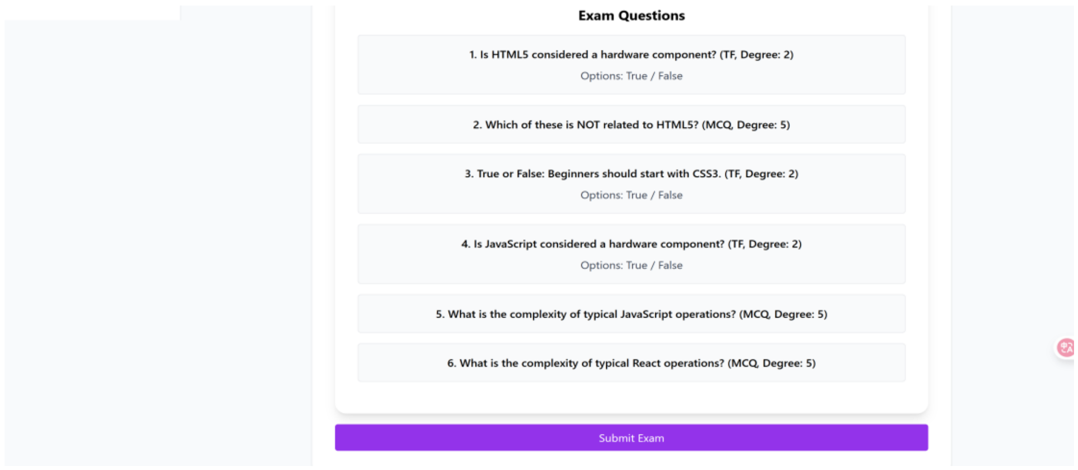
5. Which tool is commonly associated with React? (MCQ, Degree: 5)

6. True or False: React was invented in 1800. (TF, Degree: 2)
Options: True / False

Exam Questions

Step 5: Submit Exam (Instructor)

1. Locks exam for students : POST /api/exams/submit


Exam Questions

1. Is HTML5 considered a hardware component? (TF, Degree: 2)
Options: True / False

2. Which of these is NOT related to HTML5? (MCQ, Degree: 5)

3. True or False: Beginners should start with CSS3. (TF, Degree: 2)
Options: True / False

4. Is JavaScript considered a hardware component? (TF, Degree: 2)
Options: True / False

5. What is the complexity of typical JavaScript operations? (MCQ, Degree: 5)

6. What is the complexity of typical React operations? (MCQ, Degree: 5)

Submit Exam

Exam Submission

Student Exam Flow

Step 1: Fetch Exam & Questions

GET /api/student/exam/:id/questions

Response includes:

- startTime
- durationMinutes
- questions[]

Step 2: Secure Timer Logic (Server-Based)

Timer **does NOT use localStorage**

If student refreshes → timer continues correctly

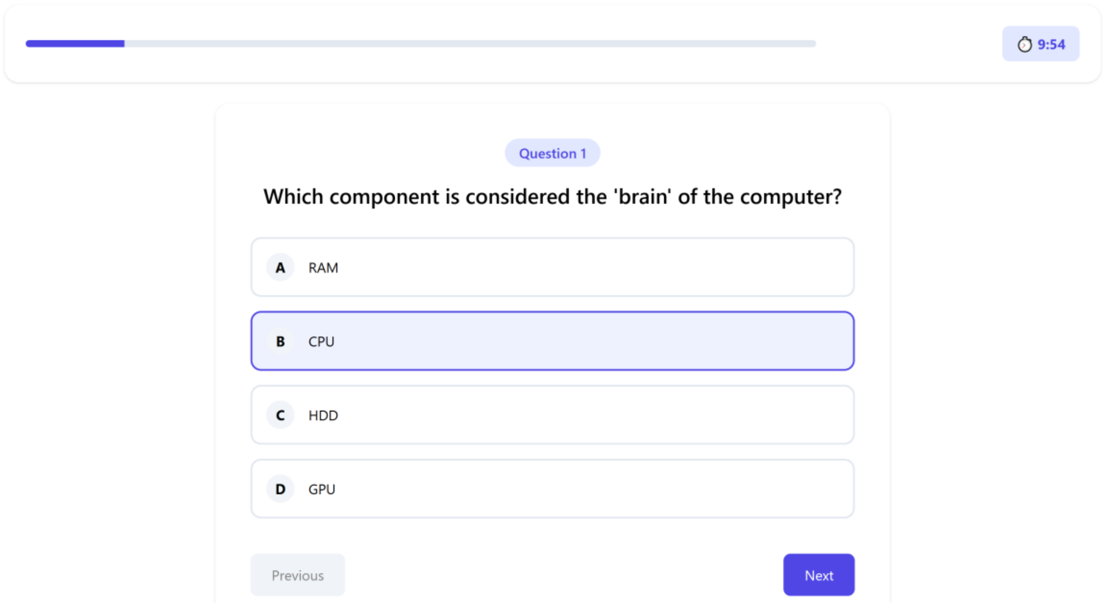
Step 3: Answering Questions

- Student answers stored locally (for navigation only)
- Options shuffled per question

Step 4: Auto Submit

- When remaining time reaches zero:

POST /api/student/exam/submit



Which component is considered the 'brain' of the computer?

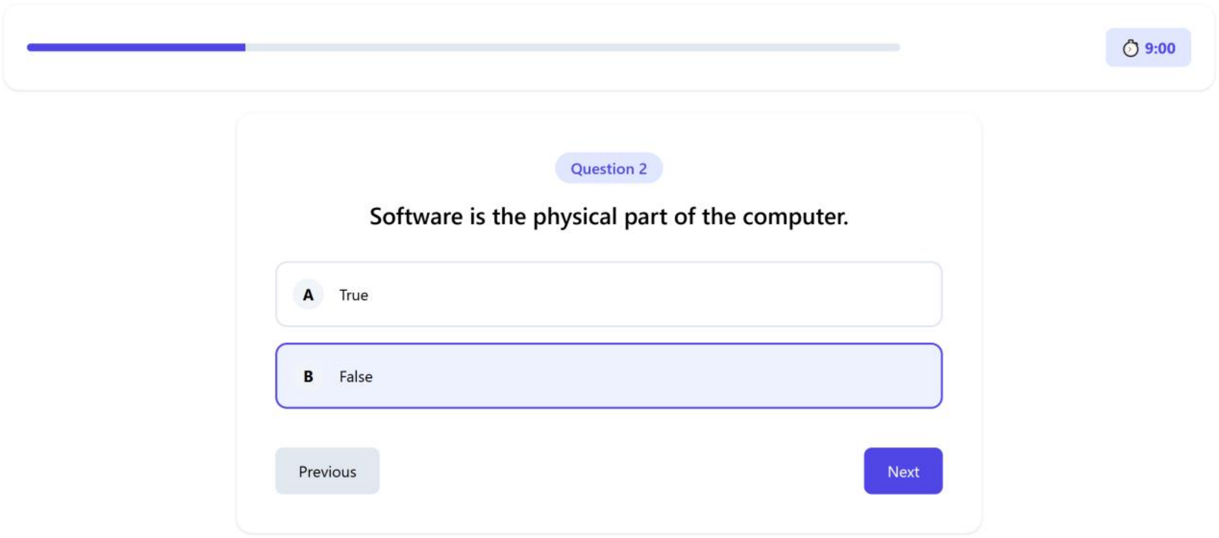
A RAM

B CPU

C HDD

D GPU

Exam View



9:00

Question 2

Software is the physical part of the computer.

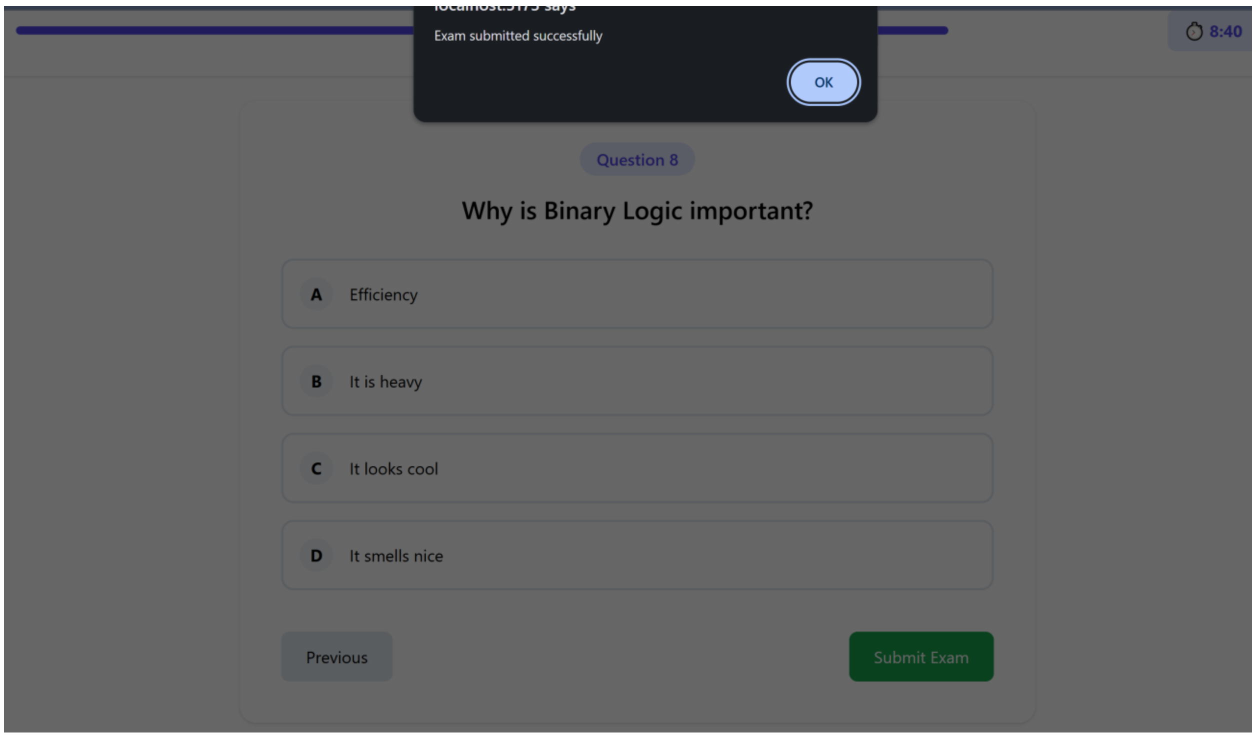
A True

[✓ B False]

Previous

Next

Time Slider during exam solving



localhost:5173 says

Exam submitted successfully

OK

8:40

Question 8

Why is Binary Logic important?

A Efficiency

B It is heavy

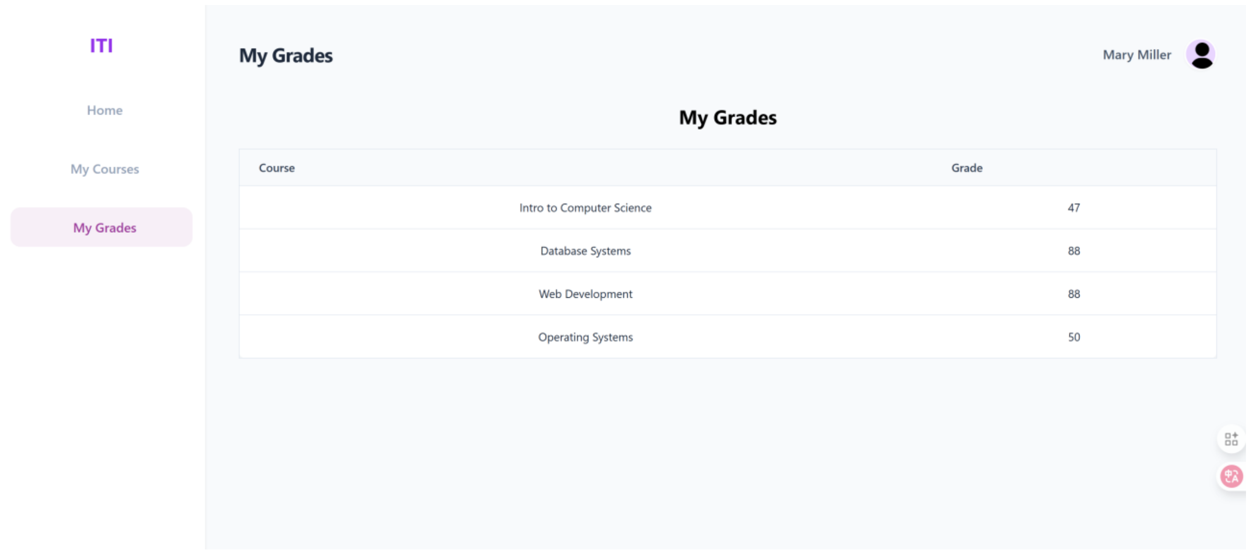
C It looks cool

D It smells nice

Previous

Submit Exam

Submitting Exam


ITI

Home

My Courses

My Grades

My Grades

Mary Miller

My Grades

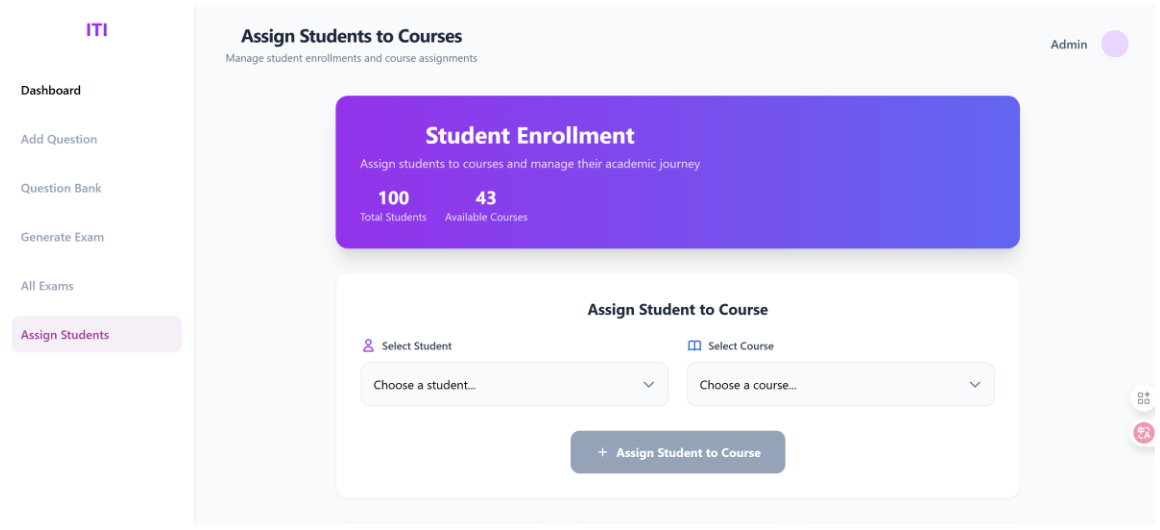
Course	Grade
Intro to Computer Science	47
Database Systems	88
Web Development	88
Operating Systems	50

Grades Dashboard

Assign Student to Course

API Usage from Frontend Perspective:

POST /api/courses/assign
Body: { st_id, crs_id }
Used by: **Admin / Instructor**


ITI

Dashboard

Add Question

Question Bank

Generate Exam

All Exams

Assign Students

Assign Students to Courses

Manage student enrollments and course assignments

Admin

Student Enrollment

Assign students to courses and manage their academic journey

100 Total Students

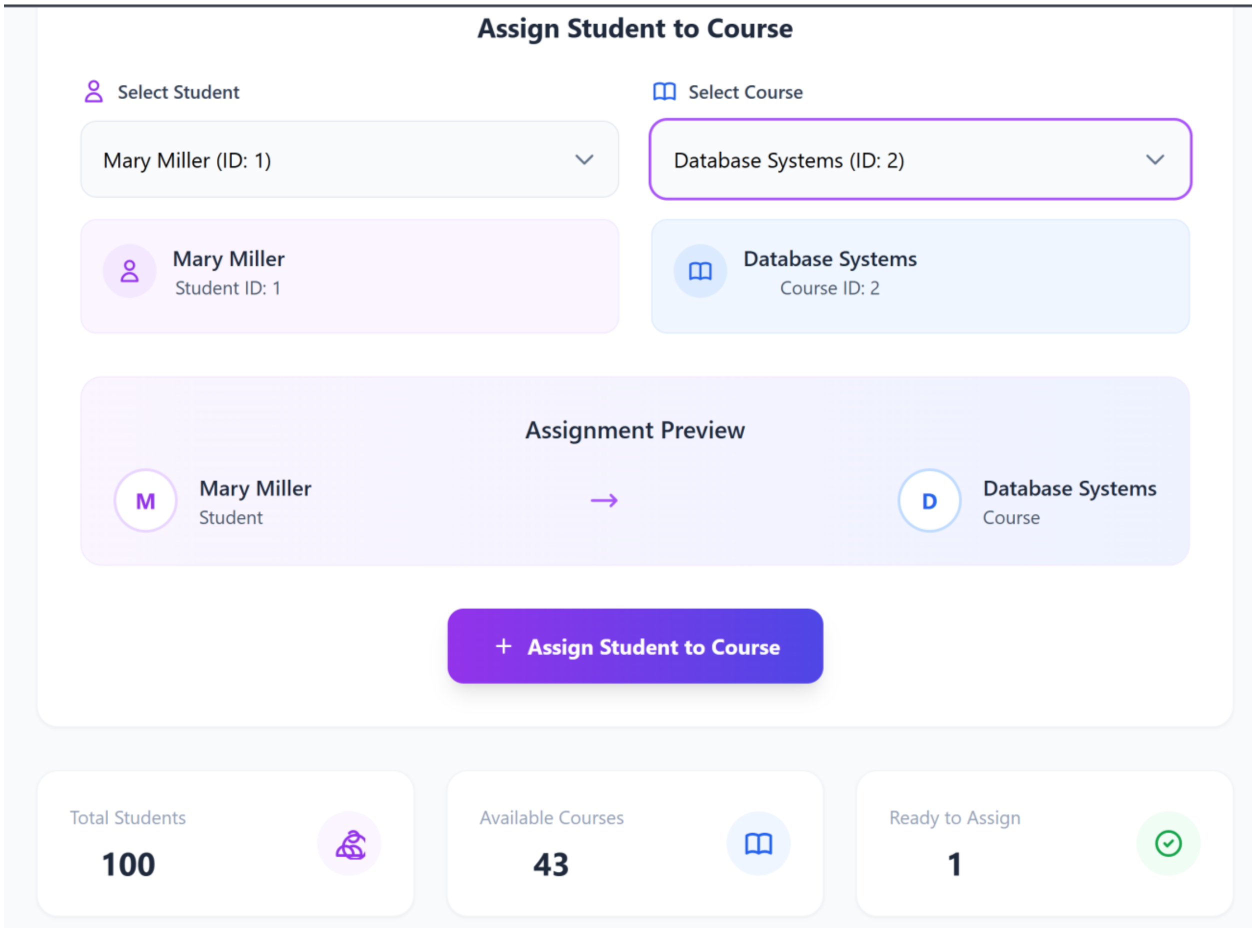
43 Available Courses

Assign Student to Course

Select Student: Choose a student...

Select Course: Choose a course...

Courses Assignement Dashboard



Assign Student to Course

Select Student: Mary Miller (ID: 1)

Select Course: Database Systems (ID: 2)

Mary Miller
Student ID: 1

Database Systems
Course ID: 2

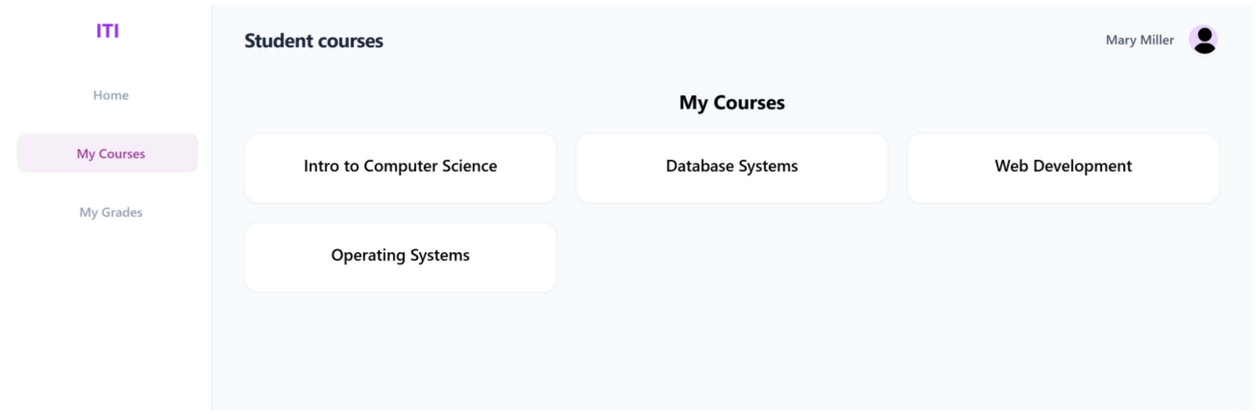
Assignment Preview

Mary Miller
Student
→
Database Systems
Course

+ Assign Student to Course

Total Students	Available Courses	Ready to Assign
100	43	1

Assigning Student to Course



ITI

Home
My Courses
My Grades

Student courses — Mary Miller

My Courses

- Intro to Computer Science
- Database Systems
- Web Development
- Operating Systems

Student Courses Dashboard