

# Examination System

## Team Members

N	Name
1	Ahmed Mohamed Reda Salah
2	Aya Fathy Mohamed Ibrahim
3	Manar Abdelmnam Elmeslmany
4	Mohamed Fayez Abdelatif Mabrouk
5	Mennatullah Ibrahim Elsaid Hamed
6	Mahmoud Mohamed Ahmed Salama

## **Abstract:**

The Examination System Database project aims to design and implement a comprehensive database solution that manages examinations within a training environment using Microsoft SQL Server. The system provides a centralized question pool with multiple question types (multiple-choice, true/false, and text-based) that instructors can use to generate exams either manually or randomly. It ensures flexibility in exam creation while maintaining data integrity and consistency through constraints, triggers, and stored procedures.

The database manages courses, instructors, students, branches, tracks, and intakes, with proper relationships to support real-world training scenarios. Instructors can create and schedule exams for their courses, assign degrees to questions, and monitor student performance. Students can only access their assigned exams during the specified time, and the system automatically evaluates objective questions while offering review options for subjective answers.

From a technical perspective, the system employs optimized indexes, file groups, and backup strategies to enhance performance and reliability. SQL Server roles and permissions ensure secure access control, with distinct accounts for administrators, training managers, instructors, and students. Daily automated backups safeguard data, while views, functions, and stored procedures provide a user-friendly interface, eliminating the need for direct queries.

The next step is to upload the database into Microsoft Azure, where the data is securely stored and managed through Azure services. Once the data is available in Azure, it is seamlessly connected and ingested into Microsoft Fabric. This integration allows us to use

Fabric's unified analytics capabilities for data transformation, modeling, reporting, and advanced insights.

The last step is to design Dashboards for analysis and extract insights ,so it is easier to read the data from graphs.And we build user friendly website where managers , instructors and students can login and access their data regarding the grades , tracks and branches where students are assigned to and instructors can track the progress of their students.

## **Introduction:**

In modern educational and training environments, examinations play a vital role in assessing student knowledge, skills, and overall performance. Traditional examination methods, which rely heavily on manual question preparation, paper-based exams, and manual grading, are often time-consuming, prone to errors, and inefficient in managing large numbers of students and courses. Moreover, ensuring exam integrity, fair evaluation, and secure storage of results becomes increasingly challenging without a structured and automated system.

The Examination System Database project addresses these issues by providing a centralized, secure, and automated examination management solution built using Microsoft SQL Server. The system enables training managers, instructors, and students to interact seamlessly through well-defined roles and permissions. A question pool is maintained to allow instructors to generate exams either manually or randomly, with support for different question types such as multiple-choice, true/false, and text-based questions. The system automatically grades objective questions while offering instructors tools to review and grade subjective answers. Providing reports,dashboards and a website so managers and instructors can

track the progress for students , grades in each track and department.

## **Problem Statement:**

Current examination processes face multiple challenges:

- 1-Manual exam creation and grading lead to delays, inefficiency, and errors.
- 2-Lack of a centralized system for managing courses, instructors, students, and exams results in poor data organization.
- 3-Ensuring exam fairness and preventing unauthorized access to exam data is difficult without role-based security.
- 4-Backup and recovery processes are often neglected, leading to risks of data loss.
- 5-Scalability issues arise when handling multiple courses, branches, intakes, and large student groups.
- 6- Creating the model in star schema view to improve the performance
- 7-Creating reports so the managers and instructor can easily access the data of the students and the grades in a user friendly way
- 8- Designing Dashboards in a clear way to be easy to understand
- 9-Developing a website for managers, students and instructors as they can login in to access their data using a friendly GUI.

The proposed Examination System Database seeks to solve these problems by offering an integrated, automated, and reliable platform that supports the entire examination lifecycle—from question creation to student result reporting ,while maintaining high performance, security, and data integrity.

## **System Requirements:**

### **Functional Requirements:**

- 1-User Management
- 2-Course & Training Management
- 3-Question Pool Management
- 4-Exam Creation & Scheduling
- 5-Exam Execution and Result Management
- 6-Security & Access Control
- 7-Backup & Recovery

### **Non-Functional Requirements:**

#### **Performance:**

- 1-Use indexes for faster query execution and high performance.

2-Support efficient random question selection and result calculation.

### **Scalability:**

1-Database design supports multiple courses, exams, instructors, and students without performance degradation.

### **Reliability & Availability:**

1-We do daily automated backups to ensure data safety and quick recovery.

2-We ensure that exams and student answers are never lost during execution.

### **Security:**

1-we use SQL Server authentication and permissions to enforce role-based access.

2-We protect student exam answers and grades from unauthorized access.

3-Implement constraints and triggers to preserve data integrity.

### **Maintainability:**

1-Use naming conventions for all objects (tables, procedures, triggers, functions).

2-Centralize business logic in stored procedures/functions for easier updates.

### **Usability:**

1-Provide views and stored procedures to simplify user interaction.

2-Ensure that instructors, managers, and students can perform tasks without writing SQL queries.

### **Data Integrity:**

1-Use constraints, foreign keys, and triggers to maintain valid relationships between entities.

2-Prevent invalid data entry (e.g., exam marks exceeding course max degree).

## **Entity Relationship Diagram(ERD):**

These are the following entities:

1-Instructor : Ins\_ID (PK) ,Name ,Email,UserName,Birthdate,  
,Gender,AcademicRank,Hire\_date.

2-Student: Student\_ID (PK) ,Name ,Email,UserName,Gender  
,Status,Address,College,University.

3-Exam : Exam\_ID (PK) , Date, start time, end time, total time , Status , Corrective .

5-Users: User\_ID, Email, Password\_hash, last\_login, User\_type, is\_active , created\_at, updated\_at.

6-Question pool : ID (PK) , Body , Degree , type, model\_answer.

7-Choices : Ch\_ID (PK) , Body , Is\_Correct, Choice\_letter.

8-Track: Track\_ID (PK), Track\_Name, No\_courses.

9-Course : Crs\_Id (PK) , crs\_Name , Description , Max Degree , Min Degree.

10-Department : Dept\_ID (PK) , Dept\_Name.

11-Branch : Branch\_ID (PK) , Branch Name, Location.

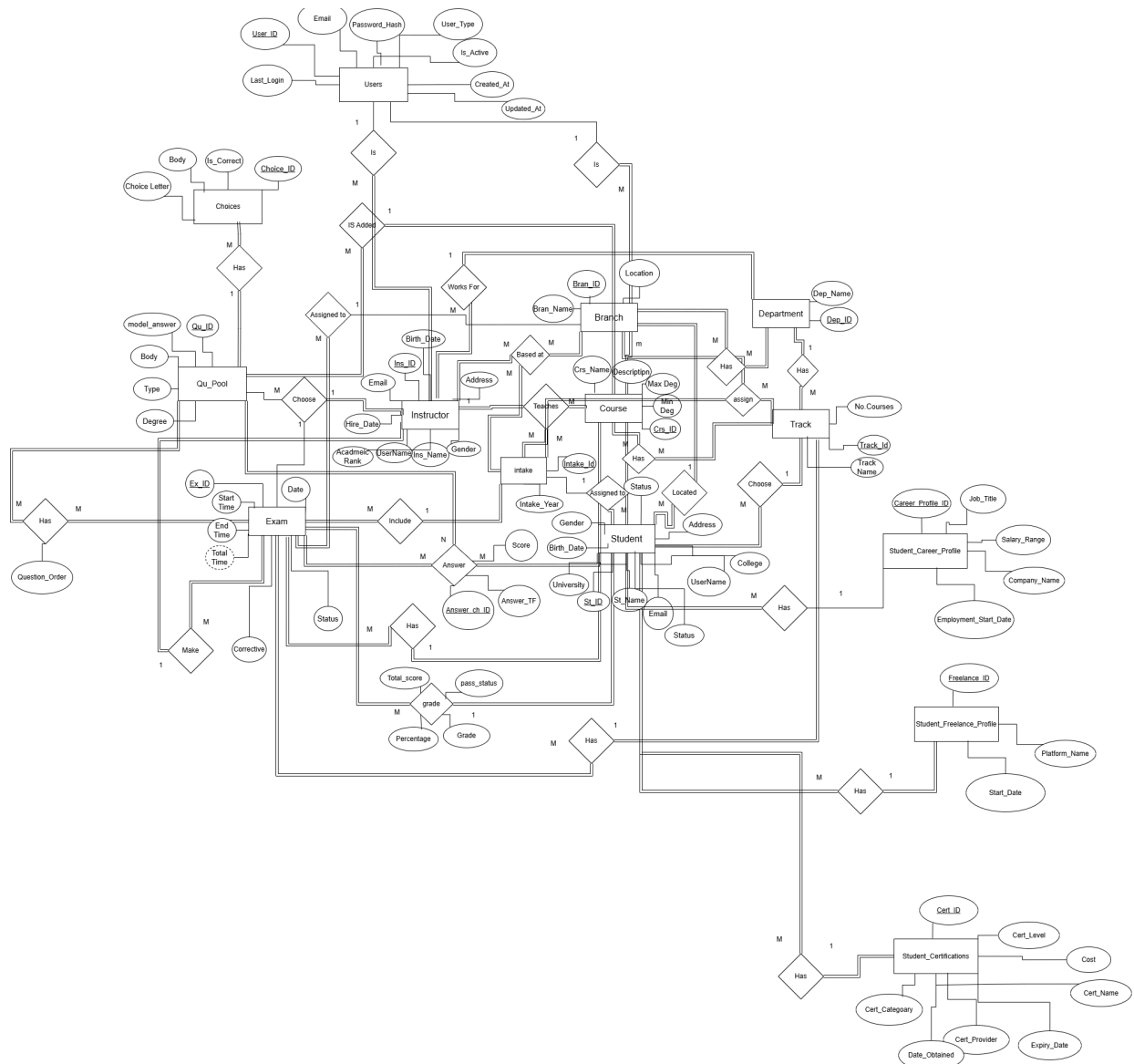
12-Intake : ID (PK) , Year.

13-Student\_Career\_Profile : career\_profile\_ID , job\_title , Salary\_range, company\_name , Employment\_start\_date.

14-Student\_Freelance\_Profile : Freelance\_ID , Platform\_Name , Start\_Date.

15-Student\_Certifications: Cert\_ID, Cert\_Level , cost, Cert\_name , Cert\_category, Cert\_provider , Data\_obtained, Expiry\_date.

# ERD:



## Mapping:

**Mapping and changing every entity to table , their attributes to column and breaking the relations into tables:**

### **1-Department ↔ Track**

- Dep\_ID as a foreign key in Track table
- **Relationship:** 1 Department → many Tracks (1:N)

### **2-Instructor ↔ Department**

- Dept\_ID as a foreign key in Instructor table
- **Relationship:** 1 Department → many instructors (1:N)

### **3-Choices ↔ Question\_Pool**

- Qu\_ID as a foreign key in choices table
- **Relationship:** 1 question → many choices(1:N)

### **4-Course ↔ Instructor ↔ Intake**

- New table for Crs\_ID, Ins\_ID, Intake\_ID
- **Relationship:** Ternary

### **5- Department ↔ Branch**

- New table for Dept\_ID, Bran\_ID
- **Relationship:** Many Departments ↔ Many Branches (M:N)

## **6- Exam ↔ Branch, Course, Intake, Instructor, Track**

- Bran\_ID, Crs\_ID, Intake\_ID, Inst\_ID, Track\_ID as foreign keys in Exam table
- **Relationship:** 1 Branch, Course, Intake, Instructor, Track ↔ Many exams (1:N)

## **7- Exam ↔ Corrective**

- Ex\_ID as a foreign key in Exam\_Corrective table
- **Relationship:** 1 Exam ↔ 1 corrective (May-Must)

## **8- Student ↔ Corrective**

- St\_ID as a foreign key in Exam\_Corrective table
- **Relationship:** 1 students → many correctives (1:N)

## **9-Exam ↔ Questions**

- New table for Ex\_ID QuP\_ID
- **Relationship:** Many Exams ↔ Many Questions (M:N)

## **10- Instructor ↔ Exam ↔ Question Pool**

- New table for Inst\_ID, Ex\_ID, QuP\_Id
- **Relationship:** Ternary Relation

## **11- Intake ↔ Instructor ↔ Branch**

- New table for intake\_ID, Inst\_ID, Bran\_ID
- **Relationship:** Ternary

## **12- Question Pool ↔ Course**

- Crs\_ID as a foreign key in Qu\_Pool table
- **Relationship:** 1 Course ↔ Many question

### **13- Student ↔ Exam ↔ Question\_Pool**

- New table for St\_ID, Ex\_ID, QuP\_ID
- **Relationship:** Ternary

### **14- Student ↔ Intake, Track, Branch**

- Intake\_ID, Track\_ID, Bran\_ID as a foreign key in Student table
- **Relationship:** 1 intake, Track, Branch ↔ Many Students

### **15- Track ↔ Course**

- New table for Track\_ID, Crs\_ID
- **Relationship:** Many tracks ↔ Many Courses

### **16- Users↔Student**

- Student\_Id as a foreign key in users table
- **Relationship:** 1user↔Many students

### **16- Users↔Instructor**

- Instructor\_Id as a foreign key in users table
- **Relationship:** 1user↔Many instructors

### **17- Student\_Career\_Profile↔Student**

- Student\_Id as a foreign key in Student\_Career\_profile table
- **Relationship:** many student↔ 1 Career profiles

### **17- Student\_Freelance\_Profile↔Student**

- Student\_Id as a foreign key in Student\_Freelance\_profile table

- **Relationship:**many student ↔ 1 Freelance profiles

### 17- Student\_Certifications ↔ Student

- Student\_Id as a foreign key in Student\_Certifications table
- **Relationship:**many student ↔ 1 Certification

--	--

--	--	--

**Track**

<u>Track_id</u>	Track_Name	No_Courses	Dep_Id
			-----

**Track\_Crs**

Track_id	Crs_Id
-----	-----

**Student**

<u>ST_ID</u>	ST_Name	Email	user_Name	Date_of_Birth	Gender	Address	College	University	Intake_ID	Track_id	Status	Bran_ID
									-----	-----		-----

**Course**

<u>Crs_Id</u>	Crs_Name	Description	Min_Deg	Max_Deg

**Inst\_Exam\_QuP**

Ins_ID	Ex_Id	Qu_ID
-----	-----	-----

**Instrucror**

<u>Ins_ID</u>	Ins_Name	Email	user_Name	Date_of_Birth	Gender	Hiring_date	Address	Academic_Rank	Dep_Id
									-----

**Exam**

<u>Ex_Id</u>	Date	start_time	End_time	Corrective	Exam_Status	Crs_ID	Intake_ID	Ins_ID	Track_id	Bran_ID
						-----	-----	-----	-----	-----

**Qu\_Pool**

<u>Qu_ID</u>	Body	Type	Degree	Crs_ID
				-----

**Exam\_QuP**

Ex_Id	Qu_ID	Question_order
-----	-----	

**Choises**

Body	Is_Correct	Qu_ID
		-----

**Stu\_Exam\_QUP**

ST_ID	Ex_Id	Qu_ID	Answer_TF	Score	Answer_Ch_ID
-----	-----	-----			-----

**Exam\_Corrective**

<u>Corrective_ID</u>	Corrective_Data	Start_Time	End_Time	Is_Completed	Score	ST_ID	Ex_Id
						-----	-----

**Intake\_Ins\_Branch**

Intake_ID	Ins_ID	Bran_ID
-----	-----	

**Dept\_Branch**

Dep_Id	Bran_ID
-----	

**Student\_Carrer\_Profile**

<u>Career_Profile_ID</u>	Company_Name	Job_Title	Employment_start_date	Salary_Range	ST_ID
					-----

**Student\_Freelance\_Profile**

# Data Definition Language(DDL):

## 1. Database Creation

- Database Name: **Examination\_System\_Data**
  - Filegroups:
    - Primary: Holds main objects.
    - EXAM\_DATA\_FG: Dedicated for large exam-related data (Exam, Questions, Answers).
  - Log File: Separate transaction log file.
  - File Settings: Initial size 500MB, with automatic growth configured.
- 

## 2. Core Entities and Tables

### Branch

- Stores branch information (e.g., Cairo, Alexandria).

### Department

- Stores academic departments (e.g., CS, IT).

### Dept\_Branch

- Junction table mapping departments to branches (many-to-many).

## Track

- Defines tracks/programs (e.g., AI, Software Development).
- Linked to a department.

## Intake

- Represents academic year/intake.

## Student

- Stores student data (name, username, email).
- Linked to Intake, Track, and Branch.

## Course

- Stores course details (name, description, min/max degrees).
- Validation: **Max\_Deg > Min\_Deg.**

## Instructor

- Stores instructor data (name, username, email).

- Linked to a department.
- 

### 3. Exam and Assessment

#### Exam

- Stores exam details (date, start/end time, corrective flag, status).
- Linked to Branch, Course, Intake, Instructor, and Track.
- Constraints:
  - End time > Start time.
  - Duration between 30–240 minutes.
  - Status limited to Scheduled, In Progress, Completed, Cancelled.

#### Question Pool (Qu\_Pool)

- Repository of exam questions.
- Supports MCQ and True/False types.
- Default degree = 5.

#### Choices

- Stores choices for MCQs.
- Marks correct answer(s).

## **Exam\_Questions**

- Maps exams to selected questions with a defined order.

## **Student Exam Answers (Stu\_Exam\_QUP)**

- Stores students' answers per exam question.
- Supports either MCQ choice or True/False (mutually exclusive).
- Stores score per question.

## **Exam\_Corrective**

- Stores corrective (make-up) exam details.
- Linked to original exam and student.
- Duration 30–240 minutes.
- Tracks completion status and score.

---

## **4. Mapping / Relationship Tables**

## **Crs\_Inst\_Intake**

- Links courses, instructors, and intakes.

## **Track\_Crs**

- Links tracks to courses.

## **Inst\_Exam\_QuP**

- Links instructors to exams and the questions they provided.

## **Intake\_Ins\_Branch**

- Maps intakes, instructors, and branches.

---

## **5. Key Features**

- **Normalization:** Proper many-to-many mappings using junction tables.
- **Constraints:** Data validation through CHECK constraints (exam duration, degrees, statuses).
- **Scalability:** Large exam/question/answer data placed in separate filegroup (**EXAM\_DATA\_FG**).

- **Integrity: Extensive use of foreign keys and composite primary keys to enforce consistency.**

## Stored Procedures:

Stored Procedure	Description
SP_OpenNewIntake	<p><b>Purpose:</b> Opens a new intake in the system by adding a record to the Intake table.</p> <p><b>Parameters:</b> @Intake_Year INT - The academic year of the new intake to be created.</p> <p><b>Logic &amp; Rules:</b> - Only targets the Intake table. - Accepts only the intake year as input; Intake_ID is auto-generated. - Rejects duplicate entries if the Intake_Year already exists. - Prevents opening an intake for a past year (year &lt; current max Intake_Year). - Inserts the new intake if all checks pass and returns the new Intake_ID.</p> <p><b>Output:</b> - Prints a success message with the newly created Intake_ID and Year. - Prints an error message and aborts if any validation fails.</p>
SP_insert_intake_branch_tracks	<p><b>Purpose:</b> Inserts multiple tracks for a specific branch in the latest intake.</p> <p><b>Functionality:</b> 1. Takes @intake_id, @branch_id, and @track_ids (comma-separated). 2. Validates that: - @intake_id exists and equals the latest intake</p>

	<p>(MAX(Intake_ID)).</p> <ul style="list-style-type: none"> <li>- @branch_id exists in the Branch table.</li> <li>- All provided @track_ids exist in the Track table.</li> </ul> <p>3. Prevents duplicate (Intake_ID, Branch_ID, Track_ID) combinations.</p> <p>4. Inserts valid records into Track_Branch_Intake.</p> <p>5. Rolls back and raises errors for invalid or duplicate entries.</p>
SP_AssignInstructorToBranchIntake	<p><b>Purpose:</b></p> <p>Purpose of this stored procedure is to <b>manually assign instructor to specific branch in specific intake</b> , but it does it carefully by checking if the instructor is already assigned before to the specific branch and intake both.</p> <hr/> <p>◆ <b>Step-by-step Explanation:</b></p> <p><b>1) Input Parameters</b></p> <ul style="list-style-type: none"> <li>● <b>@InstructorID:</b> The ID of the instructor you want to assign.</li> <li>● <b>@BranchID:</b> The ID of the branch you want to assign an instructor to it.</li> <li>● <b>@IntakeID:</b> The ID of the intake you want to assign an instructor to.</li> </ul> <p><b>2) Check if the instructor not assigned before</b></p> <ul style="list-style-type: none"> <li>● It looks up the</li> </ul>

	<p><b>Intake_Ins_Branch</b> table to make sure the instructor with <b>@InstructorID</b> not assigned before at a specific branch with <b>@BranchID</b> in specific intake with <b>@IntakeID</b> .</p> <ul style="list-style-type: none"> <li>• If yes, it prints message : <i>"Instructor is already assigned to this Branch and Intake."</i></li> </ul> <p>3) Assign the instructor to the table: Adds the instructor , branch and the intake to <b>Intake_Ins_Branch</b> table</p>
<p><b>SP_ViewStudentGradesByInstructor</b></p>	<p><b>Purpose:</b> This stored procedure allows an <b>instructor</b> to view the <b>grades of all students</b> in a specific exam they created. It retrieves each student's total score, compares it to the <b>maximum course degree</b>, and determines whether the student has <b>passed</b> or <b>needs a corrective exam</b>.</p> <hr/> <p>♦ <b>Step-by-step Explanation</b></p> <p>1) Input Parameters</p> <ul style="list-style-type: none"> <li>• <b>@InstructorID</b>: The ID of the instructor you will write.</li> <li>• <b>@ExamID</b>: The ID of the exam you want to see the grades of the student who took that exam.</li> </ul>

	<p>2) It ensures Only grades for exams assigned to the given instructor are displayed.</p> <p>3) Students' scores are calculated using aggregate function (Sum) across <b>Stu_Exam_QUP</b> Table.</p> <p>4) It checks If a student's total grade is less than 60% of the course maximum degree, message will be printed "Needs Corrective Exam", otherwise "Passed".</p> <p>5) Prevents null scores from affecting results by using <b>ISNULL</b> if there is null value it will replace it by Zero</p>
sp_ShowInstructorCourseQuestions	<p><b>Purpose:</b> Displays all available questions for a specific course to an instructor, only if the instructor is officially assigned to that course in the current intake.</p> <p><b>Parameters:</b>  @Inst_ID - Instructor ID  @Crs_ID - Course ID</p> <p><b>Logic:</b>  1. Validate instructor and course existence.  2. Verify assignment via Crs_Inst_Track_Branch_Intake.  3. If valid, return all questions for that course.  4. Otherwise, raise an appropriate error.</p> <p><b>Use Case:</b> Used by instructors when</p>

	selecting or composing their exams.
sp_ViewAssignedExams	<p><b>Purpose:</b> Retrieves all exams assigned to a given student (by ST_ID). The procedure validates:</p> <ul style="list-style-type: none"> <li>- The student exists in the system.</li> <li>- The student is enrolled in the latest intake (MAX Intake_ID).</li> </ul> <p>If valid, it returns all assigned exams with details such as:</p> <ul style="list-style-type: none"> <li>- Exam ID</li> <li>- Course name</li> <li>- Exam date and time</li> <li>- Exam type (Normal or Corrective)</li> <li>- Branch name</li> <li>- Instructor name (if available)</li> <li>- Exam status (Scheduled, Completed, etc.)</li> </ul> <p>If the student does not exist or is not in the latest intake, an appropriate error message is raised.</p> <p><b>Parameters:</b> @ST_ID INT --&gt; Student ID to check assigned exams for.</p> <p><b>Output:</b> Result set of assigned exams with related metadata.</p>
sp_ViewExamPaper	<p><b>Purpose:</b> Displays the full exam paper (questions + answers) for a specific student, course, and exam.</p> <p><b>Logic Summary:</b></p> <ol style="list-style-type: none"> <li>1. Validate student existence.</li> <li>2. Validate that the student belongs to the latest intake.</li> </ol>

	<p>3. Validate course existence.  4. Ensure the student takes this course (same intake, track, branch).  5. Ensure the exam belongs to the same course.  6. (Optional) Validate that the current date/time is within the exam window.  7. Return 20 questions with their answers in the exam order.</p> <p><b>Parameters:</b></p> <p>@St_ID INT → Student ID  @Crs_ID INT → Course ID  @Ex_ID INT → Exam ID</p> <p><b>Output:</b></p> <ul style="list-style-type: none"> <li>- Question_Order</li> <li>- Question_Body</li> <li>- Answer_ID</li> <li>- Answer_Body</li> </ul>
<p>SP_CreateExamByInstructor</p>	<p><b>Purpose:</b></p> <p>This stored procedure allows an <b>instructor</b> to create an exam for a specific <b>course, branch, intake, and track</b>. It automatically validates exam conditions, assigns random questions from the course's question pool, and links them to both the <b>exam</b> and the <b>instructor</b>. The procedure also ensures that only <b>authorized instructors</b> can create exams and that exam rules (duration, corrective/normal) are respected.</p> <hr/> <p>◆ <b>Step-by-step Explanation</b></p> <p>1) Input Parameters</p>

- |  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <ul style="list-style-type: none"><li>• <b>@InstructorID:</b> The ID of the instructor who wants to create the exam.</li><li>• <b>@Crs_ID:</b> The ID of the course for which the exam is being created.</li><li>• <b>@ExamDate:</b> The date on which the exam will be held.</li><li>• <b>@StartTime:</b>The exam start time.</li><li>• <b>@EndTime</b>The exam end time (must make exam exactly 60 or 120 minutes).</li><li>• <b>@Branch_ID</b>The branch where the exam is assigned.</li><li>• <b>@Intake_ID</b>The intake (batch/semester) for which the exam is created.</li><li>• <b>@Track_ID</b>The track within the intake where this exam applies.</li><li>• <b>@NumQuestions</b>The number of random questions to include in the exam.</li><li>• <b>@IsCorrective</b>Flag to determine if the exam is normal (0) or corrective (1).</li></ul> |
|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

	<p><b>1)Instructor authorization:</b></p> <p><b>Only instructors officially assigned to teach the course in the given intake can create exams.</b></p> <p><b>2)Track validation:</b> <b>The specified track must be linked to the chosen course.</b></p> <p><b>3)Exam duration validation:</b> <b>The exam duration must be exactly 1 hour (60 minutes) or 2 hours (120 minutes).</b></p> <p><b>4)One Normal exam per Intake:</b> <b>A course can have only one normal exam per intake. If another is needed, it must be created as a corrective exam.</b></p> <p><b>5)Randomized question selection:</b> <b>Questions for the exam are chosen randomly from the question pool of the course.</b></p> <p><b>6)Question–Instructor linkage:</b> <b>Ensures exam questions are tied back to the instructor who created the exam.</b></p> <p><b>7)Error handling:</b> <b>Any errors during exam creation are caught, and an error message</b></p>
--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	is printed instead of failing silently.
sp_ViewStudentGradesByEmail	<p><b>Description:</b> Returns a detailed grade report for a student identified by their email. The report includes course, exam type (Normal/Corrective), pass status (Pass/Fail), grade score, and intake year. Works across all courses and exams the student has taken.</p> <p><b>Parameters:</b> @Email NVARCHAR(255) – The student's registered email address.</p> <p><b>Logic:</b> 1. Validate that the student exists. 2. Fetch all graded exams for that student (from Student_Grades table). 3. Join with Exam, Course, and Intake tables for full context. 4. Return a detailed list of: <ul style="list-style-type: none"> <li>- Course Name</li> <li>- Exam Type (Normal/Corrective)</li> <li>- Exam Date</li> <li>- Grade</li> <li>- Pass Status (Pass/Fail)</li> <li>- Intake Year</li> <li>- Track / Branch (optional, for reporting)</li> </ul> </p>

sp_StudentSubmitExam	<p><b>Purpose:</b> Allows a student to submit answers for a specific exam.</p> <p>✓ <b>Functionality:</b></p> <ol style="list-style-type: none"> <li>1. Takes parameters: <ul style="list-style-type: none"> <li>- @St_ID (Student ID)</li> <li>- @Crs_ID (Course ID)</li> <li>- @Ex_ID (Exam ID)</li> <li>- @AnswerList (Comma-separated list of 20 Choice_IDs ordered by question)</li> </ul> </li> <li>2. Validates: <ul style="list-style-type: none"> <li>- Student exists in the latest intake.</li> <li>- Exam and course match the student's track, branch, and intake.</li> <li>- Student has not already submitted this exam.</li> <li>- Exactly 20 answers are provided.</li> </ul> </li> <li>3. Inserts answers by matching Exam_QUP.Question_Order (1–20) to the position of each provided answer.</li> <li>4. Optionally checks exam time window (commented for flexibility).</li> <li>5. Rolls back transaction if any validation fails.</li> </ol>
sp_Grade_Unprocessed_Exams	<p><b>Purpose:</b> Grades all unprocessed (ungraded) exams for students who have completed all their exam answers in the latest intake.</p>

	<p><b>Logic Summary:</b>  Finds all (ST_ID, Ex_ID) in Stu_Exam_Qup not in Student_Grades.  Checks that the student belongs to the latest intake.  Ensures all exam questions are answered.  Calculates score, percentage, grade, and pass/fail status.  Inserts results into Student_Grades.</p>
sp_ViewInstructorStudentsGrades	<p><b>Description:</b>  Displays all students' grades for courses taught by a given instructor in the current (latest) intake. Supports optional filters by Course ID and Branch ID for more focused results.</p> <p><b>Parameters:</b>  - @Inst_ID INT  Instructor ID (required)  - @Crs_ID INT = NULL  Optional Course ID filter  - @Bran_ID INT = NULL  Optional Branch ID filter</p> <p><b>Logic:</b>  1. Validate that the instructor exists.  2. Identify the latest intake ID (MAX(Intake_ID)) from the Intake table.  3. Validate that the instructor is teaching in this latest intake (exists in Crs_Inst_Intake table with this Intake_ID).</p>

	<p>4. Ensure that graded exams exist for this instructor in this intake.</p> <p>5. Retrieve all student grades with optional filters for Course and Branch.</p> <p>6. Display:</p> <ul style="list-style-type: none"><li>- Student Name</li><li>- Course Name</li><li>- Exam Type</li></ul> <p>(Normal/Corrective)</p> <ul style="list-style-type: none"><li>- Grade and Pass/Fail Status</li><li>- Intake Year, Branch, Track</li></ul> <p><b>Notes:</b></p> <ul style="list-style-type: none"><li>- Uses Student_Grades, Exam, Course, Student, Intake, Branch, Track.</li><li>- Ensures instructor belongs to the current active intake only.</li></ul>
--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## SP\_Assign\_Student\_To\_Course

### Purpose:

This procedure is designed to **manually assign a student to a specific course**, but it does so carefully by checking whether the student and course exist, and whether the course already belongs to the student's track.

---

#### ◆ Step-by-step Explanation

##### 1) Input Parameters

- **@StudentID**: The ID of the student you want to assign.
- **@CourseID**: The ID of the course you want to assign.

##### 2) Check if the student exists

- It looks up the **Student** table to make sure the student with **@StudentID** exists.
- If not, it raises an error: *"Student does not exist."*

##### 3) Check if the course exists

- It looks up the **Course** table to ensure the course with **@CourseID** exists.
- If not, it raises an error: *"Course does not exist."*

##### 4) Get the student's track

- Every student belongs to a **Track**.
- The procedure retrieves the

	<p><b>Track_ID</b> of the student.</p> <p>5) Check if the course already belongs to that track</p> <ul style="list-style-type: none"><li>• It searches the <b>Track_Crs</b> table (relationship between tracks and courses).</li><li>• If the course is not in the student's track, it inserts a new row into <b>Track_Crs</b>, effectively adding that course to the track.</li><li>• If the course is already there, it just prints a message: <i>"Course is already in student's track."</i></li></ul> <p>6) Error handling</p> <ul style="list-style-type: none"><li>• The procedure is wrapped in a <b>TRY...CATCH</b> block.</li><li>• If something unexpected goes wrong (e.g., constraint violation), it prints the error message instead of crashing.</li></ul>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

SP_Assign_Students_To_Track	<p><b>Purpose:</b></p> <p>This procedure is used to <b>assign multiple students at once</b> to a specific track. Instead of assigning one by one, you can pass a comma-separated list of student IDs, and it updates all of them to the given track.</p> <p>This procedure is a <b>bulk assign tool that lets you quickly move one or multiple students into a specific track</b>, while ensuring the track and all students exist before updating.</p>
SP_AutoAssignCorrectiveExam	<p><b>Purpose:</b></p> <p>The purpose of this stored procedure is <b>to automatically assign corrective (make-up) exams to students who failed their original exams</b>.</p> <p><b>It ensures that:</b></p> <ul style="list-style-type: none"> <li>• <b>Only students who scored below the minimum required degree get corrective exams.</b></li> <li>• <b>Corrective exams are scheduled automatically (7 days later, from 9 AM to 11 AM).</b></li> <li>• <b>No duplicate corrective exams are created for the same student and exam.</b></li> </ul>

SP\_assign\_ins\_crs\_intake

**Purpose:**

In specific intake, each instructor must be assigned to a specific course

**Steps:**

- 3 variables needed (  
@CourseID ,  
@InstructorID,  
@Intake\_id
- First: checking for existence of Courseid, instructorid and intake\_id entered
- After that check that all of them have not entered together before
- Finally: Insertion commands

SP\_examdetails\_St

**Purpose:**

To enter the students id and reach to their exam details (id, course name, Date, start time, End time and its status

**Steps:**

- One variable needed  
**StudentID**
- First: Select columns needed from many tables (joins used)
- Ordering them by date
- Finally: check by enter any student ID

## Triggers:

Triggercountdegrees	<p><b>Purpose:</b> To make sure that the count of degrees that instructor choose=100</p> <p><b>Action:</b> Raise error that count of degrees &lt;&gt; 100</p>
TRG_EnforceExamTimeWindow	<p><b>Purpose:</b> This trigger ensures that student exam answers inserted into the <b>Stu_Exam_QUP</b> table follow strict exam rules. It prevents invalid or unauthorized answer submissions, enforces the exam time window, and guarantees that students can only submit valid answers once per question.</p> <p><b>Action:</b> Raise error "Cannot submit answers outside the exam time frame."</p>
TR_Student_PreventDelete	<p><b>Purpose:</b> To prevent deletion of students who have exam records in the system, thereby maintaining referential integrity and preventing orphaned data.</p>

<p>TR_Stu_Exam_QUP_ValidateDegree</p>	<p><b>Purpose:</b> To enforce a business rule that every question must be worth exactly 5 points in the student exam scoring system, ensuring consistent grading across all exams.</p>
<p>TR_Qu_Pool_PreventDelete</p>	<p><b>Purpose:</b> The purpose of the trigger <b>TR_Qu_Pool_PreventDelete</b> is to control and protect deletion of questions from the <b>Qu_Pool</b> table. This trigger prevents accidental deletion of questions that are already used in exams, while allowing safe deletion of unused questions and their choices.</p>
<p>TR_Exam_Corrective_ValidateDate</p>	<p><b>Purpose:</b> The purpose of the trigger <b>TR_Exam_Corrective_ValidateDate</b> is to enforce a business rule on corrective exams so that their scheduled date is always after the original exam date This trigger ensures data integrity by making sure that a corrective exam can only be scheduled after its corresponding original exam date.</p>

## Views:

<b>v_EligibleStudentsForExam</b>	<p><b>Purpose:</b></p> <p>This view is designed to provide a <b>list of students eligible to take an exam</b>, along with details about the exam, course, intake, track, and branch.</p> <p>It also checks whether a student has <b>already taken the exam</b> and marks their eligibility status accordingly.</p>
<b>V_StudentAnswers</b>	<p><b>Purpose:</b></p> <p>To views the name of the students, question body and their answers</p>
<b>v_ExamSchedule</b>	<p><b>Purpose:</b></p> <p>To display detailed information about scheduled exams.</p> <p>Includes exam date, time, duration (calculated in minutes), branch, track, instructor, and whether it's corrective.</p> <p>Useful as an exam timetable/report for students, instructors, and admins.</p>

v_ExamChoices	<p><b>Purpose:</b></p> <p>To show all choices for each question in an exam.</p> <ul style="list-style-type: none"> <li>• Includes choice text, whether it's correct, and course name.</li> <li>• Useful for validating exams (e.g., checking correct/incorrect choices).</li> </ul>
v_TeachingAssignments	<p><b>Purpose:</b></p> <p>To show which courses are assigned to each instructor.</p> <ul style="list-style-type: none"> <li>• Includes instructor details, course, department, branch, track, intake, and total courses assigned.</li> <li>• Useful for workload management and academic planning.</li> </ul>
v_CorrectiveExams	<p><b>Purpose:</b></p> <p>To monitor corrective (make-up) exams.</p> <ul style="list-style-type: none"> <li>• Shows corrective exam details, student info, original exam, original score, corrective score, result (passed/failed), and how many days after the original exam it was scheduled.</li> </ul>

	<ul style="list-style-type: none"> <li>• Useful for tracking student improvements and compliance with exam policies.</li> </ul>
<b>v_ExamResultsByStudent</b>	<p><b>Purpose:</b></p> <p>To calculate the final result of each student per exam.</p> <ul style="list-style-type: none"> <li>• Summarizes total score, pass/fail status, number of questions, average score per question.</li> <li>• Compares student's score with Min_Degree (passing threshold).</li> <li>• Useful for exam reports, certificates, and analytics.</li> </ul>
<b>v_ExamQuestions</b>	<p><b>Purpose:</b></p> <p>To list the questions assigned to each exam.</p> <ul style="list-style-type: none"> <li>• Displays exam ID, course, question details (text, type, degree, order), and instructor who created it.</li> <li>• Helps in reviewing exam contents and auditing question assignments.</li> </ul>

v_QuestionPool	<p><b>Purpose:</b></p> <p>To display detailed information about scheduled exams.</p> <ul style="list-style-type: none"> <li>• Includes exam date, time, duration (calculated in minutes), branch, track, instructor, and whether it's corrective.</li> <li>• Useful as an exam timetable/report for students, instructors, and admins.</li> </ul>
----------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Business Intelligence Layer:

### Power BI Dashboards:

#### Dashboard 1: Intake Overview

- Key metrics: Total students, Total Intakes
- Visuals: KPI cards, Bar Charts, Slicers

#### Dashboard 2: Intake Performance Analysis

- Total failed students and succeeded ones in every year
- Total failed students and succeeded ones by each branch
- Total failed students and succeeded ones by each track

#### Dashboard 3: Student Portal

- Personal grade history
- Progress tracking
- Comparative performance

#### **Dashboard 4: Student-Freelance**

- Key metrics : total freelance revenue , total freelancers , freelance platforms,average freelance income
- Visuals:KPIS , Slicers, bar charts, area chart

#### **Dashboard 5: Branch Performance Analytics**

- Key metrics : Pass rate , fail rate , Total success , total failed
- Visuals:KPIS , Slicers, bar charts

#### **Dashboard 6: Department Performance Insights**

- Key metrics : student to instructor ratio , average score
- Visuals:KPIS , Slicers, bar charts, Gauge , scatter chart

#### **Dashboard 7: Track Academic Performance**

- Key metrics : total students ,total passed students,pass rate , average score
- Visuals:KPIS , Slicers, bar charts , line chart

#### **Dashboard 8: Instructor**

- Key metrics : Total instructors , Active instructors , pass rate , average score
- Visuals:KPIS , Slicers, bar charts, pie chart ,line chart,treemap

#### **Dashboard 9: Course Performance metrics**

- Key metrics :Total students , Total courses , Total success , total failures
- Visuals:KPIS , Slicers, bar charts

#### **Dashboard 10: Exam Overview**

- Key metrics : total students , average student score , average age , high risk students
- Visuals:KPIS , Slicers, bar charts, pie chart , line chart , doughnut chart

## **SSRS Reports**

### **1. Exam Information Report**

**Description:** Displays comprehensive exam details, including question text and multiple-choice options for a specific exam.

**SP Logic:** Utilizes [dbo].[Sp\_GetExamQu] with an @Exam\_ID parameter. The core logic employs the STRING\_AGG function to concatenate multiple answer choices

into a single, readable column. It joins Qu\_Pool, Exam\_Qup, and Choices tables to ensure correct question ordering and data integrity.

## **2. Students By Department Report**

**Description:** Categorizes and lists students based on their enrolled departments for administrative tracking.

**SP Logic:** Powered by [dbo].[SP\_GetStudentsByDepartment] using @DeptNo as a filter. It performs an INNER JOIN between Student, Track, and Department tables. The report design features a grouping mechanism by Department Name to provide a structured and clean visual output.

## **3. Branch Tracks Report**

**Description:** Identifies which branches offer specific training tracks and organizes them by intake cycles.

**SP Logic:** Uses [dbo].[Sp\_GetBranchesByTrack] to link branches, tracks, and intake periods via multiple joins. It filters data by @Track\_ID and applies a descending sort on Intake\_ID. The report implements \*Nested Grouping\* in SSRS to visualize the hierarchy between intakes and branches.

## **4. Instructor Courses Report**

**Description:** Analyzes instructor workload by listing assigned courses and the total number of students per course.

**SP Logic:** Executes [dbo].[Sp\_GetInstructorCourses] featuring an aggregation logic with COUNT(St\_ID). It utilizes a GROUP BY clause on Instructor and Course names to ensure statistical accuracy. The report is grouped by Intake\_ID to show chronological teaching assignments.

## **5. Student Profile (Certificates & Freelance)**

**Description:** Provides a holistic view of a student's professional achievements, including certifications and freelance platforms.

**SP Logic:** Implements [dbo].[Sp\_GetStudentCEF] using LEFT JOIN to ensure student data is displayed even if certificates or freelance info are missing. It uses the ISNULL function to replace null values with user-friendly text like "No Certificate," maintaining a professional report layout.

## **6. Student Final Grades Report**

**Description:** Generates a student's academic transcript, translating numerical scores into descriptive grade statuses.

**SP Logic:** Driven by [dbo].[SP\_GetStudentGrades] which incorporates a CASE statement to categorize scores into ranks (e.g., Excellent, Very Good). It joins Student\_Grades, Exam, and Course tables. This conditional logic adds significant business value by providing immediate qualitative insights into student performance.

## Web Application

### Web Portal Architecture

#### Technology Stack

- Frontend: (React/Vite build tool/TailwindCSS)
- Backend: (Python 3.12 with FastAPI/SQLAlchemy ORM)
- Authentication: JWT/OAuth 2.0
- Hosting: Azure App Service

#### User Interfaces

##### Student Portal

- Login screen
- Dashboard with upcoming exams
- Exam Taken , passed and failed
- Grade viewing
- Courses and certificates viewing
- Profile management

##### Manger Portal

- Dashboard Indicating performance overview and current intake
- System-wide exam overview
- Checking courses in each track
- View registered students by track or by branch
- Manage all instructor information and statistics
- Track all student grades across the exams (filtration by track or by branch)
- Overview Students certificates across the whole institute
- Tracking professional work experience and freelance platforms for all the students

