ITI Examination System

Team Members

Arwa Alaa

Mahmoud Mohamed

Abdelrahman yehia

Mazen Mohamed

Hisham Ahmed

DESCRIPTION FOR Examination System Database.

The Examination System Database is designed to manage and facilitate the examination processes within an educational institution. It serves various stakeholders including training managers, instructors, and students, allowing them to perform operations related to course management, question pools, exam administration, student answer tracking, and results evaluation.

Purpose: Manages user authentication and roles within the system.

System Requirements for the Examination Management System

1. Question Pool & Exam Creation

1. System should provide a question pool for instructors to select exam questions.
2. Questions can be of type Multiple Choice, True/False, or Text-based.
3. The system should store the correct answers for Multiple Choice and True/False questions.
4. Text-based questions should store the best accepted answer and use text functions & regex to check student responses.
5. Instructors should be able to review and manually grade text-based answers.

2. User Roles & Management

Training Manager can:

1. Manage branches, tracks, and intakes.
2. Add and edit students' personal data and assign them to intakes, branches, and tracks.

Instructors can:

Teach one or more courses.

Create exams only for courses they teach.

Select students for exams and define the exam date, start time, and end time.

Students can:

Access the system and take exams only within the scheduled time.

3. Course & Exam Management

The system should store course details including:

Course Name, Description, Max Degree, Min Degree.

Instructors can create exams by:

1. Selecting random or manual questions from the question pool.
2. Assigning marks per question, ensuring total marks do not exceed the course's max degree.

Each exam should store:

1. Exam Type (Regular Exam or Corrective).
2. Associated Intake, Branch, Track, and Course.
3. Start & End Time, Total Duration, and Allowance Options.

The system should track which instructor created an exam, and store it by Year, Course, and Instructor.

4. Student Exam Management & Grading

1. Students should only be able to see and take exams at the assigned time.
2. The system should store student responses for each exam.
3. Automatic grading for Multiple Choice and True/False questions.
4. Instructors should manually review and grade text-based answers.
5. The system should calculate the final score for each student in the course.

5. Authentication & Security

The system should have login accounts for

1. Training Managers
2. Instructors
3. Students

SP:Name	Desc	Done By
sp_AddIntake	This stored procedure inserts a new intake record into the `intake` table with details like name, start date, end date, year, and active status.	Arwa
sp_UpdateIntake	The stored procedure updates an existing intake record in the `Intake` table based on the provided `intake_id`. Optional parameters allow updating the intake name, start date, and active status while preserving existing values if NULL is passed.	Arwa
sp_AddDepartment	The stored procedure inserts a new department record into the `Department` table with the given department name.	Arwa
sp_UpdateDepartment	The stored procedure updates the name of an existing department in the `Department` table based on the provided `Dept_id`.	Arwa

sp_AddTrack	The stored procedure inserts a new track record into the `Track` table with the given track name and associated department ID.	Arwa
sp_UpdateTrack	he stored procedure updates an existing track record in the `Track` table based on the provided `track_id`. Optional parameters allow updating the track name and department ID while preserving existing values if NULL is passed.	Arwa
sp_AddBranch	The stored procedure inserts a new branch record into the `Branch` table with the given branch name and location.	Arwa
sp_UpdateBranch	The stored procedure updates an existing branch record in the `Branch` table based on the provided `branch_id`. Optional parameters allow updating the branch name and location while preserving existing values if NULL is passed.	Arwa
sp_AddStudent	The stored procedure inserts a new student record into the `Student` table with details such as first name, last name, email, phone, birth date, username, and password.	Arwa

sp_OpenClass	The stored procedure inserts a new class record into the `Class` table with the specified track ID, intake ID, and branch ID.	Arwa
sp_UpdateStudent	The stored procedure updates the track, intake, and branch information for an existing student in the `Student` table based on the provided `std_id`.	Arwa
sp_AddInstructor	The stored procedure inserts a new instructor record into the `Instructor` table with the given name, salary, address, phone, username, password, and training manager status.	Arwa
proc_CreateExam	The stored procedure allows an instructor to create an exam by randomly selecting a specified number of True/False, Multiple Choice, and Text questions from the `question` table for a particular course. It checks if the instructor is allowed to create an exam based on the active intake and course-year match. If the conditions are met, it inserts the exam details into the `exam` table and associates the selected questions with the exam in the `Exam_Question` table.	Arwa

proc_CreateExamManually	The stored procedure assigns an existing exam to a specific class by linking the exam with a track, branch, and intake, along with the exam's start time, end time, and date. It retrieves the year from the `intake` table based on the provided `intake_id` and inserts the exam details into the `Exam_Class` table.	Arwa
proc_AssignStudentToExam	The stored procedure allows an instructor to create an exam manually for a specific course. It first checks if the instructor is authorized to create an exam for the given course based on the active intake and course-year match. If the conditions are met, it inserts the exam into the `exam` table. If not, it prints an error message	Arwa
proc_SolveExam	The stored procedure assigns a question to an exam by inserting the question into the `Exam_Question` table with the specified exam ID, question ID, and mark. It first checks if the provided exam ID exists and if the question has already been assigned to the exam. If either condition is not met, it prints an appropriate message. Upon successful addition, it confirms the question has been added to the exam.	Arwa

GetTotalDegreeForStudentExam	The procedure records a student's answer to a specific exam question, checks if the answer is correct based on the question type (multiple choice or true/false), and updates the student's exam result accordingly. It also ensures that the student has an entry in the `exam_student` table and prevents overwriting the previous results by adding marks incrementally for correct answers. The procedure calculates the total degree a student has earned in a specific exam, validates the input, and checks if the student has a valid entry in the `Exam_Student` table. It sums the student's final results and compares them with the total marks required for passing, returning the total degree and a message indicating whether the student passed or failed the exam.	Mahmoud
instrucrors_courses_PROC	stored procedure for fetching instructor details and their courses looks mostly correct. However, there are a few minor improvements and corrections that can be made for better clarity and functionality: Show Course TEACH by Instractor_ID	mahmoud
show_BranchTrack_proc @branch_name	Show Tracks in this Branch Your procedure to show the branches and their associated tracks is on the right track! However, I recommend incorporating a few best practices for better readability and efficiency, as well as improving error handling. Here's	mahmoud

show_TrackCourses_proc	Your stored procedure for showing courses associated with a specific track is almost correct, but there are a few adjustments needed for better Show Course in this Track	Mahmoud
get_Departments	Execute department_veiw;	Mazen
SP_GetIntakeData	Execute intake_veiw After validate data	mahmoud
`GetTrackBranchIntake`	Execute track_branch_intack_view After validate data	Mahmoud
`GetQuestionsByExamAndIntake`	Execute exam_quetion_view After validate data	mazen

`ValidateStudentResult`	Execute student_degree_view After validate degree	Mazen

Trg:Name		Desc
trg_CheckTotalMarkInsteadOf	This trigger is designed to check if the total marks exceed the maximum degree for a course when new records are inserted. I noticed that your code is almost complete but is missing a few finishing touches, such as the CATCH block to handle any potential errors. Below is the complete and corrected code for your trigger:	Mahmoud
EnforceExamTime	EnforceExamTime is designed to enforce that answers cannot be submitted outside the allowed exam time	Mahmoud

trg_PreventInvalidInsert	trigger trg_PreventInvalidInsert is designed to prevent the insertion of rows into the Answer table if the student_id and exam_id do not exist in the Exam_Student table. However, there are a few issues and improvements that can be made to ensure the trigger works correctly and efficiently	Mahmoud
InsertTrack	Your trigger InsertTrack is designed to print a welcome message for each new track inserted into the track table	Mahmoud
insert_branch	Your trigger InsertBranch is designed to print a welcome message for each new track inserted into the branch table	Mahmoud
insert_department	Your trigger Insertd department is designed to print a welcome message for each new track inserted into the department table	Mahmoud

insert_intake	Your trigger Insert intake is designed to print a welcome message for each new track inserted into the intake table	Mahmoud

View:Name	Desc	
department_veiw	The department_view creates an encrypted view displaying the name column from the Department table as dept_name.	Mahmoud
intake_veiw	The intake_view creates an encrypted view that selects and renames columns from the intake table: name to intake_name, start_date to st_date, end_date to enddate, and year to Year. The WITH ENCRYPTION option hides the view's definition from the metadata.	Mahmoud
branch_veiw	The branch_view creates an encrypted view that renames Name to branch_name and location to branch_loc from the branch table.	Mahmoud

track_dept_veiw	The track_dept_view creates an encrypted view that selects the Name column from the track table and renames it as both track_name and dept_name. It joins the track table with the Department table based on matching Dept_Id.	Mahmoud
track_branch_intack_view	The track_branch_intake_view creates an encrypted view that joins the track, branch, intake, and Class tables, selecting and renaming columns as 'Track Name', 'Branch Name', and 'Intake Name'. There is a possible issue in the join condition for Intake_Id.	Mahmoud
Course_veiw	The Course_view creates an encrypted view that selects and renames columns from the Course table: C_Name to course_name, description to course_desc, min_degree to min_deg, and max_degree to max_deg.	Mahmoud
student_veiw	The student_view creates an encrypted view that combines fname and lname into st_name, and selects email as st_email, phone, birth_date as BD, and Age as age from the student table.	Mahmoud

instructor_veiw	The instructor_view creates an encrypted view that selects columns from the instructor table: Ins_Name, Address (renamed as Address), phone, and Salary.	mahmoud
exam_quetion_view	The exam_question_view creates an encrypted view that joins the exam, Question, and Exam_Question tables. It selects Que_Text from the Question table, type (renamed as Question_type) from Question, type (renamed as exam_type) from exam, and mark from Exam_Question. The view is based on matching Exam_Id and Qus_Id.	mahmoud
course_Track_view	The course_track_view creates an encrypted view that joins the track, Course_Track, and course tables. It selects the Name from track (renamed as Track_name) and C_Name from course (renamed as course), based on matching Track_Id and Course_Id.	Mahmoud

instructor_course_view	The instructor_course_view creates an encrypted view that joins the instructor, instructor_course, and course tables. It selects Ins_Name from instructor (renamed as instructor_name), C_Name from course (renamed as course), and year from instructor_course (renamed as year), based on matching Ins_ID and CourseID.	mahmoud
student_degree_view	The student_degree_view creates an encrypted view that joins the exam, student, and Exam_Student tables. It selects the concatenated fname and lname as student_name, type from exam (renamed as exam_type), and final_result from Exam_Student, based on matching Exam_id and student_Id.	mahmoud
StudentFinalResultsByCourse	The StudentFinalResultsByCourse view retrieves the final results of students for each course. It joins the student, Exam_Student, exam, Instructor, Instructor_course, and Course tables, selecting the student's full name (CONCAT(Std.Fname, ' ', Std.lname)), course name (Cors.C_Name), and the final result (E_Std.final_result).	Mahmoud

ExamDetailsByTrackBranchIntakeDepartment	The ExamDetailsByTrackBranchIntakeDepartment view retrieves detailed exam information, including the ExamId, TrackName, BranchName, IntakeName, and DepartmentName. It joins the Exam_Class, branch, intake, track, and Department tables based on their respective IDs to provide these details.	mazen
AllowExamOptions	The AllowExamOptions view retrieves the ExamId and corresponding AllowanceOptions for each exam. It joins the exam table with the ExmAllowanceOptions table based on matching Exam_Id, selecting the exam ID and its allowance options.	Mazen
StudentsEnrolledInCourses	The StudentsEnrolledInCourses view retrieves the full names of students and the courses they are enrolled in. It joins the student, Class, track, Course_Track, and Course tables to match students with their respective courses based on track and course IDs.	mazen

InstructorsTeachingCourses	The InstructorsTeachingCourses view retrieves the names of instructors, the courses they teach, and the corresponding year. It joins the Instructor, Instructor_course, and Course tables based on matching Ins_ID and CourseID to provide this information.	mazen
All index in project	Create index to some table	Hisham