Software Requirements Specification

for

<Exam Management Database System>

Version 1.0 approved

Prepared by <Team 5>

<Information Technology Institute>

<22/5/2025>

Table of Contents

1. Introduction

1.1 Purpose

This document specifies the complete requirements for a database-centric Exam Management System designed to automate exam lifecycle processes, including:

- ***Exam creation** (randomized questions, time windows).*
- ***Student participation tracking** (answer submission, grading).*
- ***Reporting** (performance analytics, instructor workload).*

1.2 Document Conventions

Style	Usage
Bold	Key database entities (e.g., `Exam`).
`Monospace`	SQL objects (e.g., `usp_CreateExam`).
Italics	References (e.g., *ERD in Appendix B*).
ALL_CAPS	Constraints (e.g., `NOT NULL`).

1.3 Intended Audience and Reading Suggestions

Role	Focus Area
Database Architects	Schema design (Section 4.1).
Developers	Stored procedures/triggers (Section 3.2).
QA Engineers	Test cases for triggers (Section 5.3).

1.4 Project Scope

In-Scope

- ***Core Tables:** Exam, Student, Question_pool, Instructor.*
- ***Automation:** Grading via Calculate_Student_marks*
- ***Security:** Role-based access (RBAC) via user.role.*

Out-of-Scope

- *Frontend application development.*
- *Network infrastructure setup.*

1.5 References

1. ***ERD:*** **Examination Database System .png* (Appendix B).*
2. ***Object Catalog:*** *Database_Project_Objects_Documentation.docx.*
3. ***IEEE 830-1998 Standard.***

2. Overall Description

2.1 Product Perspective

*The **Exam Management Database System** is a **standalone, database-only solution** designed to serve as the **backend data repository and processing engine** for an educational institution's exam lifecycle. It **does not include a user interface** but is structured to support future integration with web or desktop applications via APIs or direct SQL access..>*

2.2 Product Features

Feature	***Key Objects***	***Priority***
Exam Creation	*CreateExam_Prc, Question_pool*	*High*
Grading Automation	*Calculate_Student_Degree, Student_Exam*	*High*
Reporting	*vw_ExamResults, sp_GetInstructorLoad*	*Medium*

2.3 User Classes and Characteristics

1. ***Database-Centric Architecture***
   - *The system is purely a relational database (MySQL/PostgreSQL) with no built-in UI.*
   - *All interactions occur via:*
     - ***Stored Procedures** (e.g., CreateExam_Prc, Calculate_Student_Degree).*
     - ***Views** (e.g., Top10StudentsPerCourse, vw_ExamResults).*
     - ***Direct SQL queries** (for admins/developers).*
2. ***Integration-Ready***
   - *Designed to be **called by external applications** (e.g., a web app using Python/Java).*
   - *Exposes **well-defined APIs** via stored procedures (e.g., usp_CreateExam).*
3. ***Self-Contained Business Logic***
   - ***Triggers** enforce critical rules (e.g., TRG_Prevent_Answer_After_Time blocks late submissions).*
   - ***Functions** handle calculations (e.g., Calculate_Student_Degree automates grading).*

2.4 Operating Environment

Component	Requirement
Database Server	MySQL 8.0+ or PostgreSQL 14+.
Storage	20GB (min), SSD recommended.

.

2.5 Design and Implementation Constraints

1. **Normalization:** 3NF (e.g., no transitive dependencies in Student_Exam).
2. **Triggers:** Enforce critical rules (e.g., TRG_Prevent_Answer_After_Time).

   Database-Only Constraint

   No Application Layer:

   The system **must not include UI components** (e.g., forms, dashboards).
   All interactions must occur via:

   Stored procedures (e.g., usp_CreateExam).
   Direct SQL queries (for admins).

   Rationale: Ensures separation of concerns; UI logic is delegated to future applications.

   No Built-in Authentication:

   Authentication/authorization is **limited to database roles** (e.g., GRANT SELECT TO 'instructor').

2.6 User Documentation

- *SQL Scripts: For schema deployment (CREATE TABLE scripts).*
- *SQL Script : For schema , structure and data.*

2.7 Assumptions and Dependencies

1. *Users are SQL-literate.*
2. *Authentication is handled externally.*
3. *Input data is pre-validated.*
4. *Instructors review auto-grades.*
5. *Single-tenant system.*

3. System Features

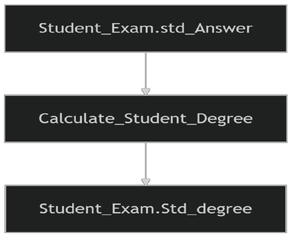
- ***3.1 Exam Creation Module***
- ***3.1.1 Functional Requirements***

ID	***Requirement***	***Implementation***	***Validation***
FR1-01	*Create exams with randomized questions.*	*SelectRandomQuestionsForExam (SP)*	*Verify via EXEC SelectRandomQuestionsForExam(ExamID, 10)*
FR1-02	*Enforce exam time windows.*	*TRG_Prevent_Answer_After_Time*	*Test with INSERT after EndTime.*

- ***3.1.2 Error Handling***
  FR1-03: *If TotalDegree > Course.MaxDegree, reject with error:*
  RAISE_ERROR('Exam degree exceeds course limit.');
- ***3.2 Grading Module***
- ***3.2.1 Functional Requirements***
-

ID	***Requirement***	***SQL Implementation***
FR2-01	*Auto-grade multiple-choice answers.*	*Calculate_Student_Degree (Function)*
FR2-02	*Log grade changes.*	*trg_Update_TotalDegree (Trigger)*

1.1 3.2.2 Data Flow



y.

3.1 Design and Implementation Constraints

3. **Normalization:** 3NF (e.g., no transitive dependencies in Student_Exam).
4. **Triggers:** Enforce critical rules (e.g., TRG_Prevent_Answer_After_Time).

Database-Only Constraint

No Application Layer:

The system **must not include UI components** (e.g., forms, dashboards).
All interactions must occur via:

Stored procedures (e.g., usp_CreateExam).
Direct SQL queries (for admins).

Rationale: Ensures separation of concerns; UI logic is delegated to future applications.

No Built-in Authentication:

Authentication/authorization is **limited to database roles** (e.g., GRANT SELECT TO 'instructor').

3.2 User Documentation

- *SQL Scripts: For schema deployment (CREATE TABLE scripts).*
- *SQL Script : For schema , structure and data.*

3.3 Assumptions and Dependencies

1. *Users are SQL-literate.*
2. *Authentication is handled externally.*
3. *Input data is pre-validated.*
4. *Instructors review auto-grades.*
5. *Single-tenant system.*

4. System Features

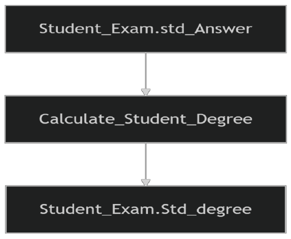
- ***3.1 Exam Creation Module***
- ***3.1.1 Functional Requirements***

ID	*Requirement*	*Implementation*	*Validation*
FR1-01	*Create exams with randomized questions.*	*SelectRandomQuestionsForExam (SP)*	*Verify via EXEC SelectRandomQuestionsForExam(ExamID, 10)*
FR1-02	*Enforce exam time windows.*	*TRG_Prevent_Answer_After_Time*	*Test with INSERT after EndTime.*

- ***3.1.2 Error Handling***
  FR1-03: *If TotalDegree > Course.MaxDegree, reject with error:*
  RAISE_ERROR('Exam degree exceeds course limit.');
- ***3.2 Grading Module***
- ***3.2.1 Functional Requirements***

ID	*Requirement*	*SQL Implementation*
FR2-01	*Auto-grade multiple-choice answers.*	*Calculate_Student_Degree (Function)*
FR2-02	*Log grade changes.*	*trg_Update_TotalDegree (Trigger)*

4.1 Data Flow



5. Database Design

5.1.1 4.1 Schema Details

5.1.1.1 Key Tables

Table	**Columns (PK in bold)**	**Constraints**
Exam	**ExamID**, CourseID, StartTime, TotalDegree	CHECK (TotalDegree <= Course.MaxDegree)
Student_Exam	**Exam_id**, **Std_id**, Std_degree	FOREIGN KEY (Std_id) REFERENCES Student

5.1.2 Indexing Strategy

Table	Indexed Column	Purpose
Exam	CourseID	Speed up course-specific queries.
Student_Exam	Std_id	Accelerate student grade lookup.

6. Other Nonfunctional Requirements

6.1.1 Performance

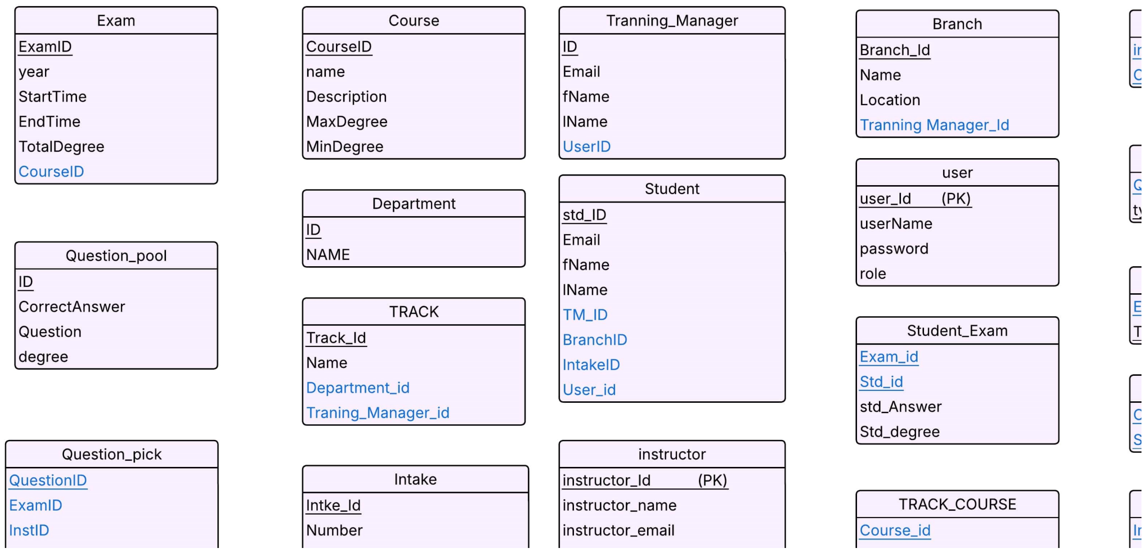
Scenario	Requirement
Concurrent exam submissions	200+ transactions/sec.
Report generation (vw_ExamResults)	<3s response time.

6.1.2 Security

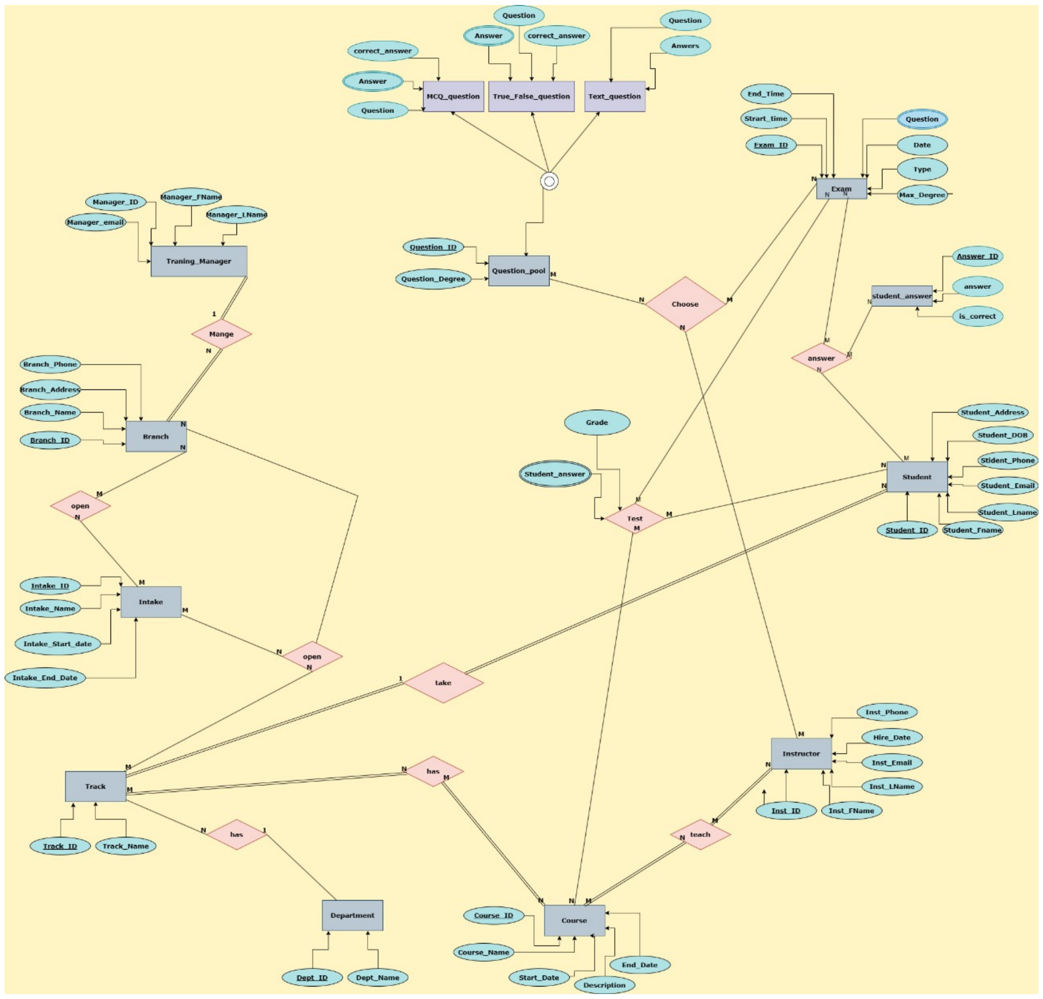
Control	Implementation
Password hashing	bcrypt in user.password.
Row-level security	GRANT SELECT ON vw_StudentInfo TO 'instructor'

6.1.3 Test Cases

Test ID	Scenario	Expected Result
TC-01	Submit answer after EndTime	Trigger blocks with error



ER diagram:


Question
Answer
correct_answer
Question
Answers
correct_answer
Answer
MCQ_question
True_False_question
Text_question
Question
End_Time
Strart_time
Exam_ID
Question
Date
Type
Max_Degree
Exam
Manager_ID
Manager_FName
Manager_LName
Manager_email
Traning_Manager
Question_ID
Question_Degree
Question_pool
Choose
Answer_ID
answer
is_correct
student_answer
answer
Mange
Branch_Phone
Branch_Address
Branch_Name
Branch_ID
Branch
Grade
Student_answer
Test
Student_Address
Student_DOB
Stident_Phone
Student_Email
Student_Lname
Student_ID
Student_Fname
Student
open
Intake_ID
Intake_Name
Intake_Start_date
Intake_End_Date
Intake
open
take
Inst_Phone
Hire_Date
Inst_Email
Inst_LName
Inst_ID
Inst_FName
Instructor
Track
Track_ID
Track_Name
has
has
teach
Department
Dept_ID
Dept_Name
Course_ID
Course_Name
Start_Date
Description
End_Date
Course
1
N
M