

Index Corruption – Assignment IV

DBMS Lab – Assignment IV

This document describes the Student Academic Management System (DBMS Lab Assignment IV).

1 Name and Roll Number of Group Members

Group Name: Index Corruption

Members:

- Ashutosh Sharma – 23CS10005
- Sujal Anil Kaware – 23CS30056
- Parag Mahadeo Chimankar – 23CS10049
- Kshetrimayum Abo – 23CS30029
- Kartik Pandey – 23CS30026

2 ER Diagram

The complete Entity-Relationship Diagram for the Student Academic Management System is shown below.

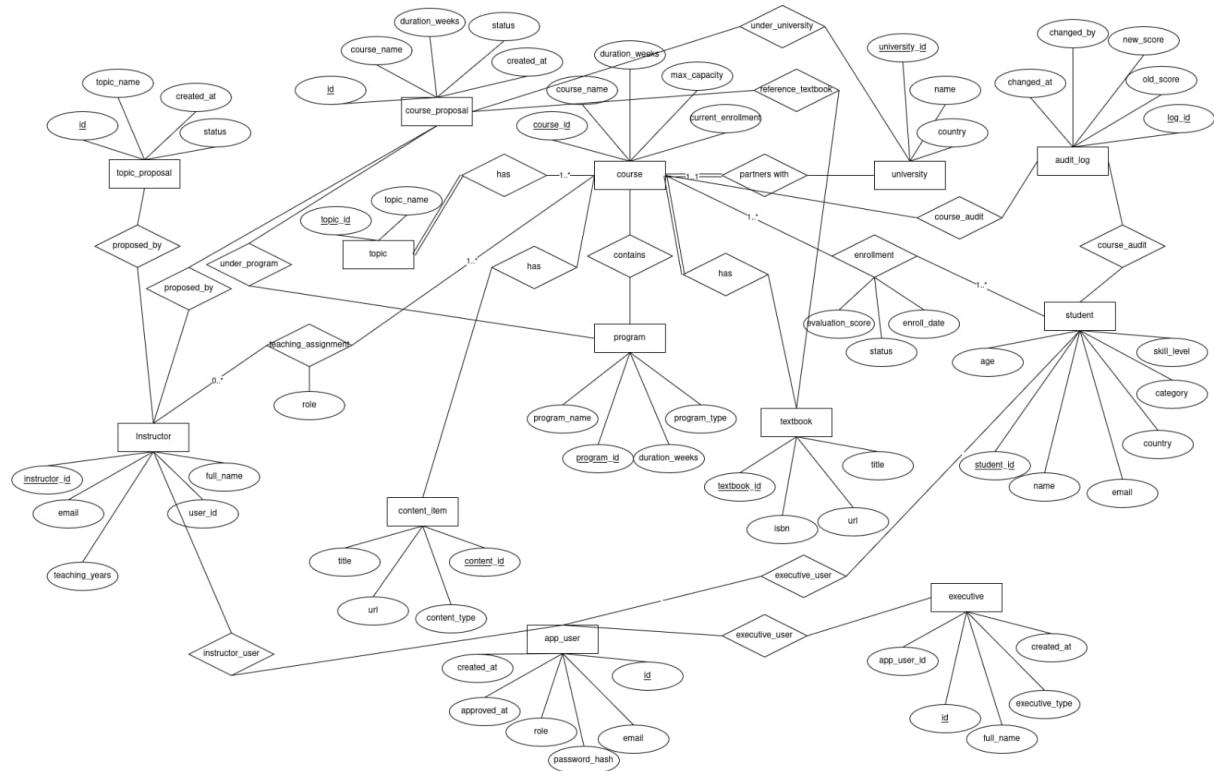


Figure 1: Entity-Relationship Diagram for Student Academic Management System

3 Table Schema

The system uses a PostgreSQL database with the following tables. The schema supports universities, programs, courses, students, instructors, enrollments, teaching assignments, content items, and role-based access control.

3.1 RBAC and Users

Table	Description
app_user	Roles-based access control: <code>id</code> (PK), <code>email</code> (unique), <code>password_hash</code> , <code>role</code> (admin / instructor / student / analyst), <code>created_at</code> , <code>approved_at</code> . Links to <code>student.email</code> or <code>instructor.email</code> .

3.2 Core Domain Tables

Table	Key Columns
university	<code>university_id</code> (PK), <code>name</code> (unique), <code>country</code>
program	<code>program_id</code> (PK), <code>program_name</code> , <code>program_type</code> , <code>duration_weeks_or_months</code>
course	<code>course_id</code> (PK), <code>course_name</code> (unique), <code>duration_weeks</code> , <code>current_enrollment</code> , <code>university_id</code> (FK), <code>program_id</code> (FK), <code>textbook_id</code> (FK)
textbook	<code>textbook_id</code> (PK), <code>title</code> , <code>isbn</code> (unique), <code>url</code>
topic	<code>topic_id</code> (PK), <code>topic_name</code> (unique)
course_topic	<code>course_id</code> (FK, PK), <code>topic_id</code> (FK, PK)

3.3 People and Assignments

Table	Key Columns
student	<code>student_id</code> (PK), <code>email</code> (unique), <code>full_name</code> , <code>age</code> (≥ 13), <code>country</code> , <code>category</code> , <code>skill_level</code>
instructor	<code>instructor_id</code> (PK), <code>full_name</code> , <code>email</code> (unique)
enrollment	<code>student_id</code> (FK, PK), <code>course_id</code> (FK, PK), <code>enroll_date</code> , <code>evaluation_score</code> , <code>status</code>
teaching_assignment	<code>instructor_id</code> (FK, PK), <code>course_id</code> (FK, PK), <code>role</code>
content_item	<code>content_id</code> (PK), <code>course_id</code> (FK), <code>content_type</code> , <code>title</code> , <code>url</code>
audit_log	<code>log_id</code> (PK), <code>student_id</code> (FK), <code>course_id</code> (FK), <code>old_score</code> , <code>new_score</code> , <code>changed_by</code> , <code>changed_at</code>

3.4 Proposals

Table	Key Columns
course_proposal	proposal_id (PK), proposed_by (FK to instructor), course_name, duration_weeks, university_id (FK), program_id (FK), textbook_id (FK), status (pending/approved/rejected), created_at
topic_proposal	proposal_id (PK), proposed_by (FK to instructor), topic_name, status (pending/approved/rejected), created_at

3.5 Key Relationships

- **Course** belongs to one University and one Program; has one Textbook; many ContentItems; many Enrollments; many TeachingAssignments; many Topics via course_topic.
- **Student** has many Enrollments; email links to app_user for login.
- **Instructor** has many TeachingAssignments; email links to app_user for login.
- **Enrollment** is a many-to-many between Student and Course with enroll_date, evaluation_score, and status.
- **Teaching assignment** is a many-to-many between Instructor and Course with role.

3.6 Database Constraints and Features

- **Check Constraints:** student.age ($\geq 13, \leq 100$), enrollment.evaluation_score (0–100), course.duration_weeks (> 0)
- **Indexes:** Strategic indexes on student.country, enrollment.evaluation_score, course.duration_weeks, course.course_name
- **Database Triggers:** trg_auto_enrollment_count automatically maintains course.current_enrollment
- **Pessimistic Locking:** SELECT ... FOR UPDATE used in enrollment operations to prevent race conditions

4 List of Functionalities Implemented

The system implements a complete academic management platform with four user roles (Student, Instructor, Admin, Analyst), each with distinct capabilities enforced via JWT-based Role-Based Access Control (RBAC).

4.1 Student Functionalities

- **Registration:** Self-registration with name, email, age (≥ 13), country, skill level, and password. No admin approval required.
- **Browse Courses:** Search for available courses with name, topic, program type, university, and duration.
- **Apply for Courses:** Submit enrollment applications. Applications start with status = pending. They can track pending and rejected enrollment applications.
- **Course Detail Page:** Access course materials (content items), textbooks, topics, and personal evaluation scores.

4.2 Instructor Functionalities

- **Registration:** Register with name, email, years of experience, and password. Requires admin approval before login.
- **Dashboard:** View his/her created courses with student counts.
- **Student Management:** View enrolled students and pending applications per course. Approve or reject enrollment applications.
- **Grading:** Set or update evaluation scores (0–100) for enrolled students. All grade changes are logged in `audit_log`.
- **Audit Log Viewing:** View complete history of grade changes per course.
- **Content Management:** Add content items (books, videos, notes with URLs) to courses and delete existing content.
- **Topic Management:** Link approved topics to courses or remove topic associations.
- **Course Proposals:** Propose new courses with name, duration, university, program, and textbook. Proposals require admin approval.
- **Topic Proposals:** Propose new topics requiring admin approval.
- **Course Analytics:** View score distribution, average score, and total students per course.

4.3 Admin Functionalities

- **Dashboard:** View system-wide statistics (total users, courses, students, instructors, enrollments).
- **User Management:** List all users, create new users with any role (automatically creates linked records), delete users with cascading removal.
- **Registration Approvals:** Approve or reject pending instructor and analyst registrations by setting `approved_at` timestamp.
- **Course Proposal Approvals:** Review and approve/reject course proposals. On approval, system creates course, assigns proposing instructor, and links topics.
- **Topic Proposal Approvals:** Review and approve/reject topic proposals. Creates new topic records on approval.
- **University Management:** Create new university records for course creation.

4.4 Analyst Functionalities

- **Registration:** Register with name, email, and password. Requires admin approval before login.
- **Overall Statistics:** Total courses, students, enrollments, and platform-wide average evaluation score.
- **Most Popular Course:** Identify course with highest enrollment, with optional university-based filtering.
- **Enrollments per Course:** Breakdown of enrollment counts for each course.
- **Average Score by Course:** Average evaluation scores ranked across all courses.
- **Top Indian Student by AI Average:** Highest-performing Indian student in AI-related courses (using ILIKE pattern matching).
- **Courses by University:** Course count distribution across universities.
- **Students by Country:** Geographic distribution of student body.
- **Skill Level Distribution:** Breakdown of students by skill level (beginner, intermediate, advanced).
- **Top Courses:** Highest-enrollment courses with average scores.

4.5 Advanced Reports (Analyst Module)

- **Module Analytics:** Cohort analysis showing average scores and student counts per program type.
- **Instructor Performance Index (IPI):** Computes ratio of instructor's average score to global topic average, identifying high- and low-performing instructors per topic.
- **At-Risk Students:** Identifies students with average evaluation score below configurable threshold (default: 40).
- **Topic Trends:** Ranks topics by total enrollment volume to identify trending subjects.

4.6 Advanced Database Features

- **Database Triggers:** PL/pgSQL trigger for automatic `current_enrollment` maintenance.
- **Audit Logging:** Complete audit trail for grade changes with old/new scores, timestamps, and responsible user.
- **Pessimistic Locking:** `SELECT . . . FOR UPDATE` for safe concurrent enrollment operations.
- **Check Constraints:** Database-level validation for age, scores, and durations.
- **Indexes:** Performance optimization on frequently queried columns.

5 List of Front-end Tools Used

The frontend is built with modern web technologies providing a responsive, type-safe user interface.

5.1 Core Framework and Language

- **Next.js 16:** React framework with App Router for server-side rendering and routing
- **React 19:** Component-based UI library
- **TypeScript:** Strongly-typed JavaScript for improved developer experience and code quality

5.2 Styling and UI Components

- **Tailwind CSS 4:** Utility-first CSS framework for rapid UI development
- **Radix UI:** Headless, accessible UI component primitives
 - Dialog component for modals
 - Label component for form accessibility
 - Scroll Area component for custom scrollbars
 - Other accessible primitives
- **Lucide React:** Icon library for consistent, customizable icons

5.3 Development Tools

- **ESLint:** JavaScript/TypeScript linter for code quality
- **eslint-config-next:** Next.js-specific linting rules
- **npm:** Package manager for dependency management

5.4 Project Structure

- /apps/web – Next.js frontend application
 - Role-specific pages: admin, analyst, instructor, student
 - Authentication pages: login, registration
 - Shared components and utilities

5.5 Backend Integration

- **FastAPI:** Python-based REST API backend
- **JWT Authentication:** Token-based authentication for secure API access
- **PostgreSQL:** Relational database with SQLAlchemy ORM