Index Corruption – Assignment 5

DBMS Lab – Assignment 5

Abstract. This report presents the Student Academic Management System developed for DBMS Lab Assignment 5. The system is a full-stack web application that supports four roles—Student, Instructor, Admin, and Analyst—with JWT-based authentication and a PostgreSQL backend.

Repository: `https://github.com/ashuwhy/dbmslab`
Screenshots: `https://ashuwhy.com/dbmslabscreenshots`

1 Screenshot Preview

Figure 1 shows a preview of the application.

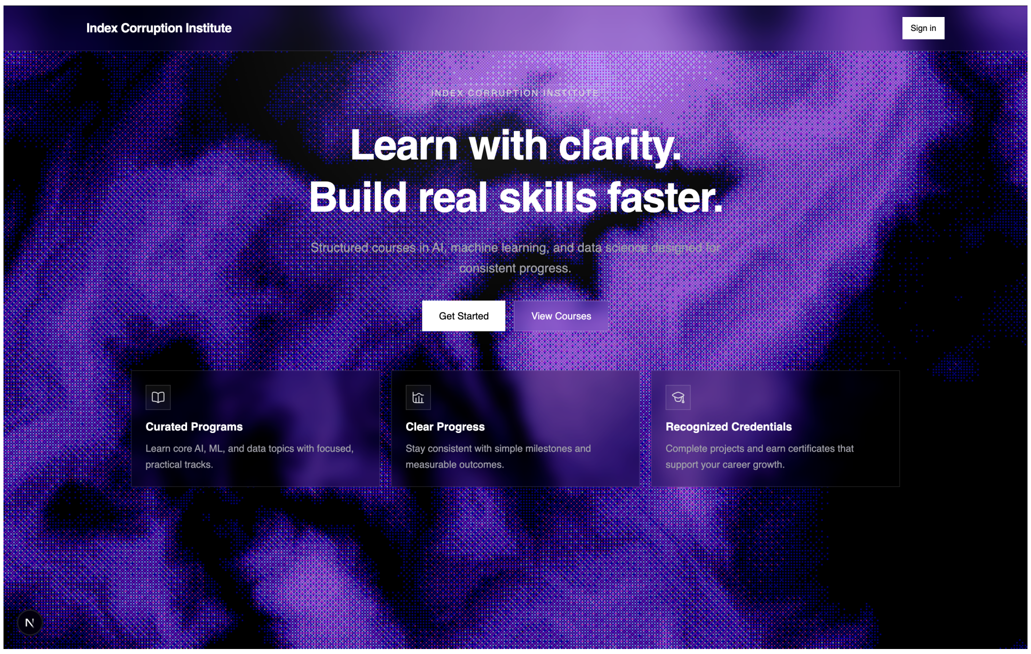
Index Corruption Institute
Sign in
INDEX CORRUPTION INSTITUTE
Learn with clarity.
Build real skills faster.
Structured courses in AI, machine learning, and data science designed for consistent progress.
Get Started
View Courses
Curated Programs
Learn core AI, ML, and data topics with focused, practical tracks.
Clear Progress
Stay consistent with simple milestones and measurable outcomes.
Recognized Credentials
Complete projects and earn certificates that support your career growth.

Figure 1: Application screenshot preview

2 Name and Roll Number of Group Members

Group No.: 1
Group Name: Index Corruption

Members:

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

3 ER Diagram

The complete Entity-Relationship Diagram for the Student Academic Management System is shown below. (Place the file `er.png` or `er.jpeg` in the same folder as this `main.tex` when compiling.)

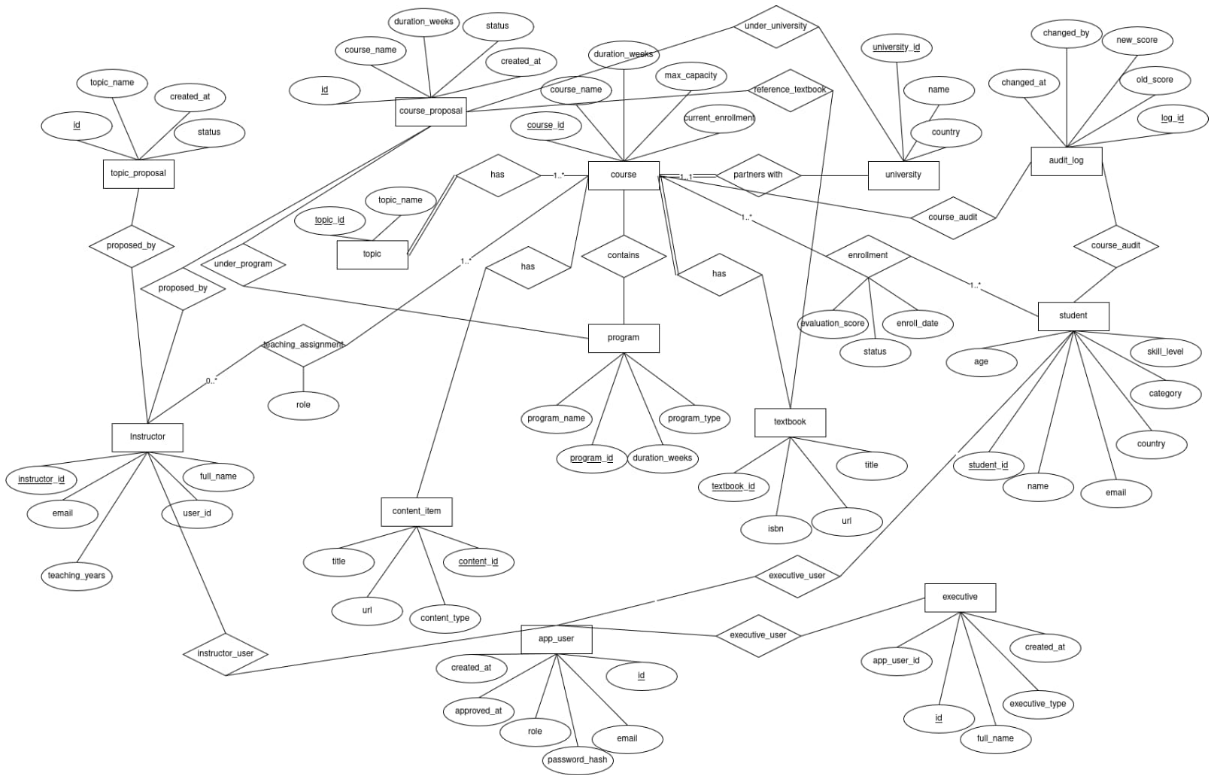

Figure 2: Entity-Relationship Diagram for Student Academic Management System

4 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, role-based access control, and proposal workflows.

4.1 RBAC and Users

Table	Description
`app_user`	RBAC: `id` (PK), `email` (unique), `password_hash`, `role` (admin / instructor / student / analyst), `created_at`, `approved_at`. Links to `student.email`, `instructor.email`, or `executive` via `app_user_id`.
`executive`	Admin and analyst users: `id` (PK), `app_user_id` (FK to `app_user`, unique), `full_name`, `executive_type` (admin / analyst), `created_at`.

4.2 Core Domain Tables

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

4.3 People and Assignments

Table	Key Columns
`student`	`student_id` (PK), `email` (unique), `full_name`, `age` (≥ 13, ≤ 100), `country`, `category`, `skill_level`
`instructor`	`instructor_id` (PK), `full_name`, `email` (unique), `teaching_years`, `user_id` (FK to `app_user`)
`enrollment`	`student_id` (FK, PK), `course_id` (FK, PK), `enroll_date`, `evaluation_score` (0–100), `status` (pending/approved/rejected)
`teaching_assignment`	`instructor_id` (FK, PK), `course_id` (FK, PK), `role`
`content_item`	`content_id` (PK), `course_id` (FK), `content_type`, `title`, `url`
`audit_log`	`log_id` (PK), `student_id` (FK), `course_id` (FK), `old_score`, `new_score`, `changed_by`, `changed_at`

4.4 Proposals

4.5 Key Relationships

- **Course** belongs to one University and one Program; has one Textbook; has `max_capacity` and `current_enrollment`; 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` and `user_id` link to `app_user` for login.
- **Executive** (admin/analyst) links one-to-one to `app_user` via `app_user_id`.
- **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`.

Table	Key Columns
`course_proposal`	`id` (PK), `instructor_id` (FK), `course_name`, `duration_weeks`, `university_id` (FK), `program_id` (FK), `textbook_id` (FK), `status` (pending/approved/rejected), `created_at`
`topic_proposal`	`id` (PK), `instructor_id` (FK), `topic_name`, `status` (pending/approved/rejected), `created_at`

4.6 Database Constraints and Features

- **Check Constraints:** `student.age` ($\geq$ 13, $\leq$ 100), `enrollment.evaluation_score` (0–100), `course.duration_weeks` (> 0), `program.duration_weeks_or_months` (> 0), proposal `status` values.
- **Indexes:** On `student.country`, `enrollment.evaluation_score`, `enrollment(course_id, evaluation_score)`, `course.duration_weeks`, `course.course_name`, `course_proposal.status`, `topic_proposal.status`, `executive.executive_type`, `instructor.user_id`.
- **Database Triggers:** `trg_auto_enrollment_count` (function `fn_update_enrollment_count`) automatically maintains `course.current_enrollment` on INSERT/DELETE of `enrollment`.
- **Pessimistic Locking:** `SELECT ... FOR UPDATE` used in enrollment operations to prevent race conditions.

5 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).

5.1 Student Functionalities

- **Registration:** Self-registration with name, email, age ($\geq$ 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.

5.2 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.

5.3 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.

5.4 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.

5.5 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.

5.6 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.

6 List of Front-end Tools Used

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

6.1 Core Framework and Language

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

6.2 Styling and UI Components

- **Tailwind CSS 4:** Utility-first CSS framework for rapid UI development
- **tailwindcss-animate:** Animation utilities for Tailwind
- **Radix UI:** Headless, accessible UI component primitives
  - `@radix-ui/react-dialog` for modals
  - `@radix-ui/react-label` for form accessibility
  - `@radix-ui/react-scroll-area` for custom scrollbars
  - `@radix-ui/react-slot` for composition
- **Lucide React:** Icon library for consistent, customizable icons
- **Huge Icons:** `@hugeicons/react` and `@hugeicons/core-free-icons` for additional icons
- **class-variance-authority (cva):** Type-safe variant-based component styling
- **clsx** and **tailwind-merge:** Conditional class names and Tailwind class merging

6.3 Development and Build Tools

- **ESLint:** JavaScript/TypeScript linter for code quality
- **eslint-config-next:** Next.js-specific linting rules
- **npm:** Package manager for dependency management
- **PostCSS** with `@tailwindcss/postcss` for Tailwind CSS 4

6.4 Project Structure

- `/apps/web` – Next.js frontend application
  - Role-specific pages: `admin/`, `analyst/`, `instructor/`, `student/`
  - Authentication pages: `login/`, `signup/` (student, instructor, analyst)
  - Shared components (`components/`, `app/components/`) and UI primitives (`components/ui/`)

6.5 Backend Integration

- `/apps/api` – FastAPI: Python-based REST API backend
  - JWT Authentication: Token-based authentication for secure API access
  - PostgreSQL: Relational database with SQLAlchemy ORM