Online Course Management Platform

Web Based Information System Design
Database Management Systems Lab Assignment IV

Krishna Singha (23CS10034)
Shankit Kumar Das (23CS10066)
Sukumar Singh (23CS10069)
Suvadip Bhuiya (23CS10070)
Aman Tudu (23CS30004)

Team Name: TEAM BONGO-DB

ER Diagram

The ER diagram shows the structural design of the Online Course Management Platform database, including key entities and their relationships. It supports role-based users, course and content management, and enrollment tracking while ensuring data integrity and proper normalization.

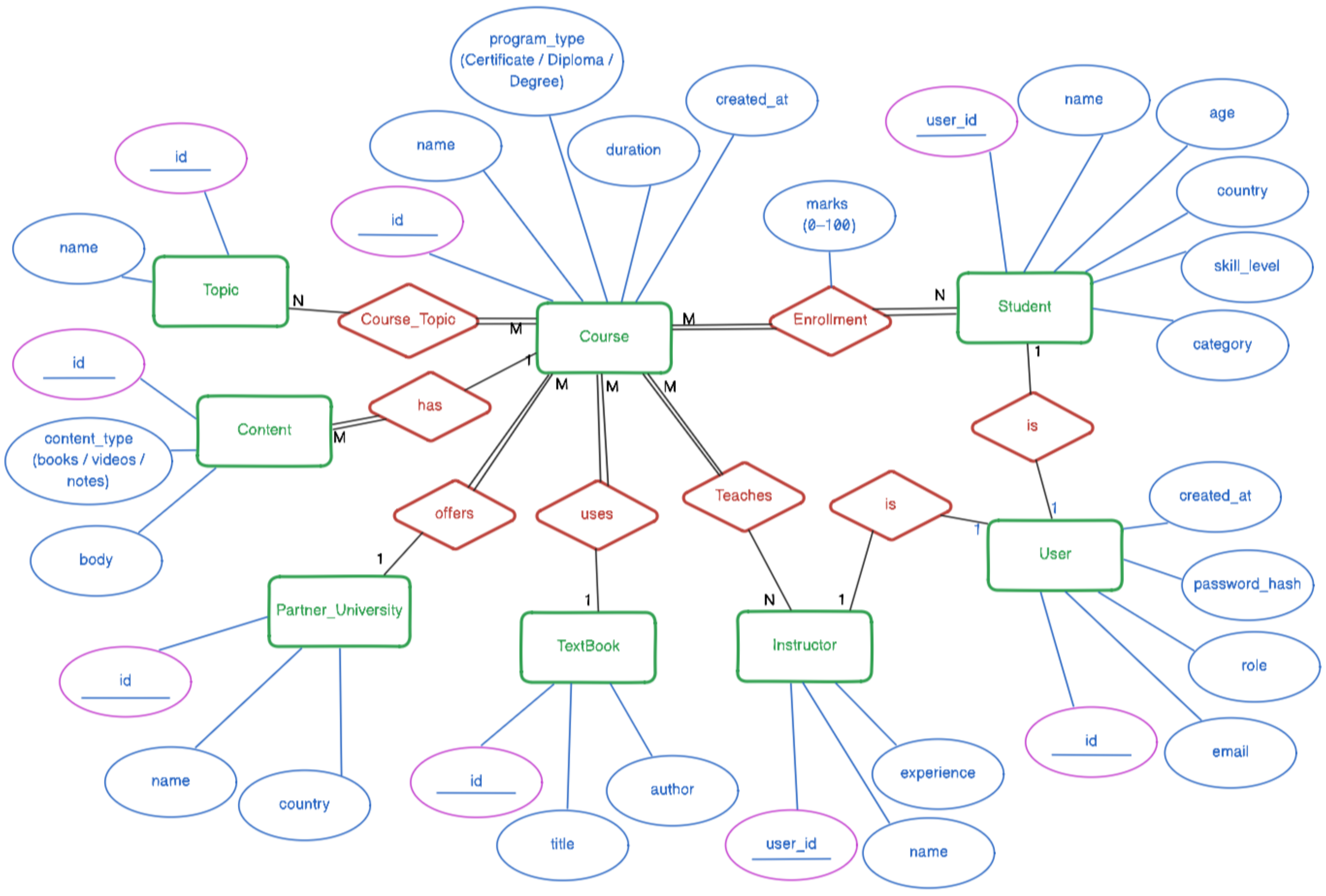


Figure 1: ER Diagram of the System

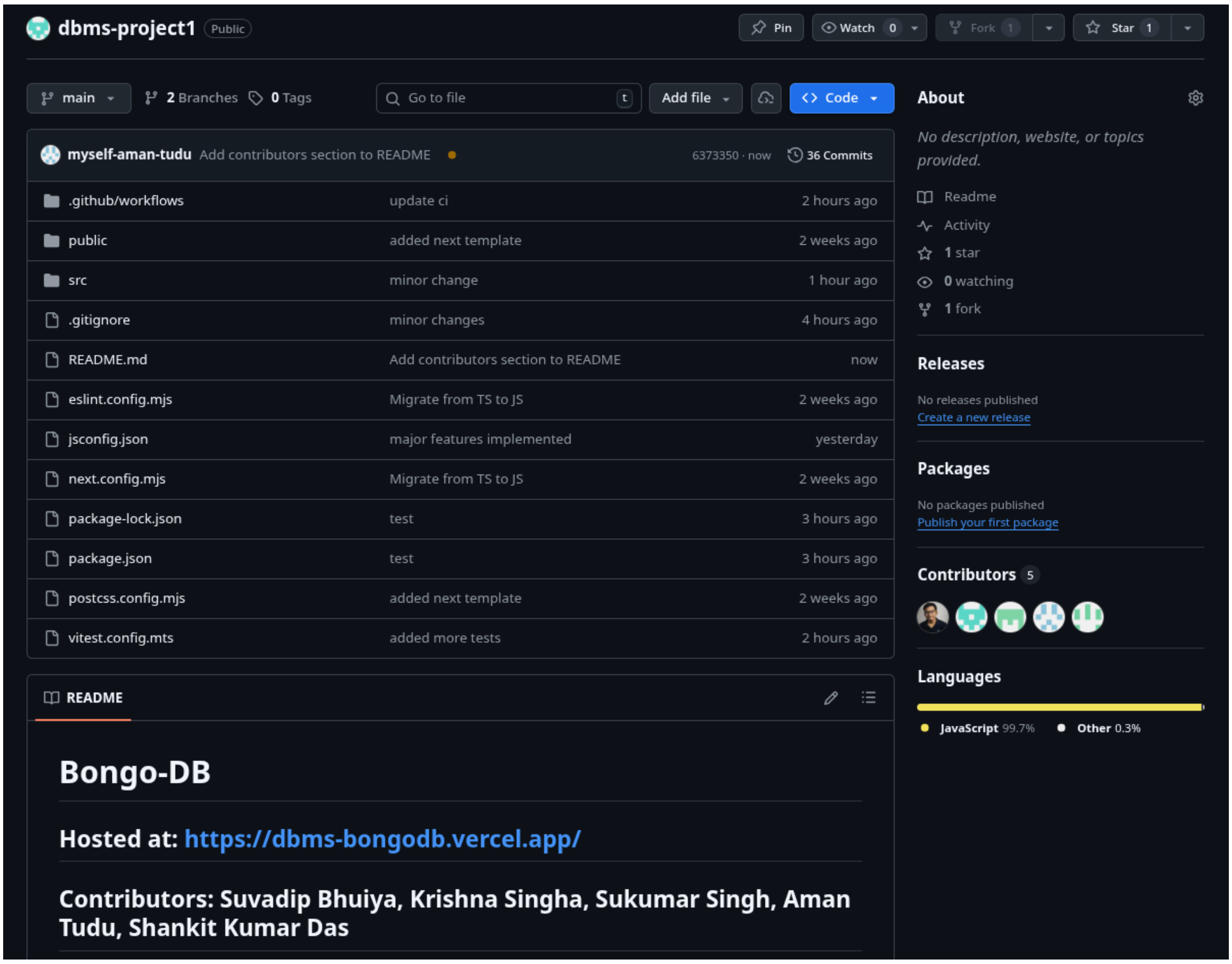
Project Repository

The complete source code, database schema, and implementation details for the **Team BONGO-DB** project can be found in our GitHub repository.

`https://github.com/suvadipbhuiya2004/dbms-project1`

Gihub Repository Image

Below are the visual representations of the implemented Online Course Management Platform:


dbms-project1 Public

myself-aman-tudu Add contributors section to README · 6373350 · now · 36 Commits

.github/workflows	update ci	2 hours ago
public	added next template	2 weeks ago
src	minor change	1 hour ago
.gitignore	minor changes	4 hours ago
README.md	Add contributors section to README	now
eslint.config.mjs	Migrate from TS to JS	2 weeks ago
jsconfig.json	major features implemented	yesterday
next.config.mjs	Migrate from TS to JS	2 weeks ago
package-lock.json	test	3 hours ago
package.json	test	3 hours ago
postcss.config.mjs	added next template	2 weeks ago
vitest.config.mts	added more tests	2 hours ago

About: No description, website, or topics provided. Readme · Activity · 1 star · 0 watching · 1 fork

Releases: No releases published. Create a new release

Packages: No packages published. Publish your first package

Contributors 5

Languages: JavaScript 99.7% · Other 0.3%

README

Bongo-DB

Hosted at: https://dbms-bongodb.vercel.app/

Contributors: Suvadip Bhuiya, Krishna Singha, Sukumar Singh, Aman Tudu, Shankit Kumar Das

Source Code: https://github.com/suvadipbhuiya2004/dbms-project1

Database Schema Description

Defined Enum Types

- **role_enum:** `ADMIN`, `STUDENT`, `INSTRUCTOR`, `DATA_ANALYST`
- **program_enum:** `CERTIFICATE`, `DIPLOMA`, `DEGREE`
- **content_enum:** `BOOK`, `VIDEO`, `NOTES`

Table: course

Column Name	Data Type	Attributes & Constraints
id	UUID	PK Default: `gen_random_uuid()`
name	TEXT	NOT NULL Part of Unique Index
program_type	program_enum	NOT NULL
duration	INT	NOT NULL Check: `duration > 0`
university_id	UUID	FK NOT NULL Ref: `partner_university(id)`
book_id	UUID	FK Ref: `textbook(id)`
created_at	TIMESTAMPTZ	NOT NULL Default: `now()`
Constraints	-	UNIQUE (name, university_id)

Table: content

Column Name	Data Type	Attributes & Constraints
id	UUID	PK Default: `gen_random_uuid()`
course_id	UUID	FK NOT NULL Ref: `course(id)` | On Delete: CASCADE
type	content_enum	NOT NULL
body	TEXT	Stores content payload
created_at	TIMESTAMPTZ	NOT NULL Default: `now()`

Table: topic

Column Name	Data Type	Attributes & Constraints
id	UUID	PK Default: `gen_random_uuid()`
name	TEXT	UQ NOT NULL

Table: users

Column Name	Data Type	Attributes & Constraints
id	UUID	PK Default: `gen_random_uuid()`
email	TEXT	UQ NOT NULL
password_hash	TEXT	NOT NULL
role	role_enum	NOT NULL Part of Composite Unique
created_at	TIMESTAMPTZ	NOT NULL Default: `now()`
Constraints	-	UNIQUE (id, role) *(Partition Key)*

Table: student

Column Name	Data Type	Attributes & Constraints
user_id	UUID	PK FK Ref: `users(id)`
role	role_enum	PK FK Ref: `users(role)` | Check: `role='STUDENT'`
name	TEXT	NOT NULL
age	INT	Check: `age > 0`
country	TEXT	Optional
skill_level	TEXT	Optional
category	TEXT	Optional

Table: instructor

Column Name	Data Type	Attributes & Constraints
user_id	UUID	PK FK Ref: `users(id)`
role	role_enum	PK FK Ref: `users(role)` | Check: `role='INSTRUCTOR'`
name	TEXT	NOT NULL
experience	INT	Check: `experience >= 0`
rating	NUMERIC(3,2)	Check: `0 <= rating <= 5`

Table: partner_university

Column Name	Data Type	Attributes & Constraints
id	UUID	PK Default: `gen_random_uuid()`
name	TEXT	UQ NOT NULL
country	TEXT	NOT NULL

Table: enrollment

Column Name	Data Type	Attributes & Constraints
student_id	UUID	PK FK Ref: `student(user_id)`
course_id	UUID	PK FK Ref: `course(id)`
marks	INT	Check: `0 <= marks <= 100`
created_at	TIMESTAMPTZ	NOT NULL Default: `now()`

Table: textbook

Column Name	Data Type	Attributes & Constraints
id	UUID	PK Default: `gen_random_uuid()`
title	TEXT	NOT NULL Part of Unique Index
author	TEXT	Part of Unique Index
Constraints	-	UNIQUE (title, author)

Table: teaches

Column Name	Data Type	Attributes & Constraints
inst_id	UUID	PK FK Ref: `instructor(user_id)`
course_id	UUID	PK FK Ref: `course(id)`

Table: course_topic

Column Name	Data Type	Attributes & Constraints
course_id	UUID	PK FK Ref: `course(id)`
topic_id	UUID	PK FK Ref: `topic(id)`

SQL Code: Database Schema Implementation

This appendix documents the structured SQL implementation of the database schema developed for the Online Course Management Platform. The schema has been implemented in PostgreSQL with minimal redundancy and strong referential integrity constraints.

The following sections include:

- Enumerated type definitions
- Core entity tables
- Role-specific extensions
- Relationship (many-to-many) mappings
- Indexing strategies for optimization

---

1 Database Implementation

This section presents the SQL implementation of the Online Course Management Platform database schema. The schema is designed to ensure data integrity, normalization, and support for role-based access control.

1.1 Database Extension and Enum Definitions

The following definitions initialize required PostgreSQL extensions and define enumerated types used for role management, program categorization, and content classification.

```sql
CREATE EXTENSION IF NOT EXISTS "pgcrypto";
-- ENUMS
CREATE TYPE role_enum AS ENUM (
    'ADMIN',
    'STUDENT',
    'INSTRUCTOR',
    'DATA_ANALYST'
);
CREATE TYPE program_enum AS ENUM (
    'CERTIFICATE',
    'DIPLOMA',
    'DEGREE'
);

CREATE TYPE content_enum AS ENUM (
    'BOOK',
    'VIDEO',
    'NOTES'
);
```

1.2 User and Role-Specific Tables

The users table serves as the central entity for managing authentication credentials and fundamental identity information of all platform participants. It stores essential details such as unique user identifiers, email addresses, encrypted passwords, assigned roles, and account creation timestamps. This design ensures secure authentication and enables role-based access control (RBAC) within the system.

```sql
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name TEXT NOT NULL,
    email TEXT UNIQUE NOT NULL,
    password_hash TEXT NOT NULL,
    role role_enum NOT NULL,
    created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE TABLE students (
    user_id UUID PRIMARY KEY REFERENCES users(id) ON DELETE
       CASCADE,
    age INT CHECK (age > 0),
    country TEXT,
    skill_level TEXT,
    category TEXT
);

CREATE TABLE instructors (
    user_id UUID PRIMARY KEY REFERENCES users(id) ON DELETE
       CASCADE,
    experience INT CHECK (experience >= 0),
    rating NUMERIC(3,2) CHECK (rating BETWEEN 0 AND 5)
);
```

1.3 Academic Entities

These tables collectively define the academic framework of the platform by modeling core educational entities such as partner universities, textbooks, subject topics, and structured courses. They establish the foundational relationships required to organize programs, associate learning resources, and maintain institutional affiliations within the system.

```sql

CREATE TABLE partner_university (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name TEXT UNIQUE NOT NULL,
    country TEXT NOT NULL
);
```

```sql

CREATE TABLE textbooks (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    title TEXT NOT NULL,
    author TEXT,
    UNIQUE (title, author)
);

CREATE TABLE topics (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name TEXT UNIQUE NOT NULL
);

CREATE TABLE courses (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name TEXT NOT NULL,
    description TEXT,
    program_type program_enum NOT NULL,
    duration INT NOT NULL CHECK (duration > 0),
    university_id UUID NOT NULL REFERENCES partner_university(id)
       ,
    book_id UUID REFERENCES textbooks(id),
    created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
    UNIQUE (name, university_id)
);

CREATE INDEX idx_course_university ON courses(university_id);
```

1.4 Course Content Management

The contents table is responsible for managing and organizing course-related learning materials, including books, instructional videos, and academic notes, each of which is systematically associated with a specific course to ensure structured content delivery and proper academic resource management within the platform.

```sql
CREATE TABLE contents (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    course_id UUID NOT NULL REFERENCES courses(id) ON DELETE
       CASCADE,
    type content_enum NOT NULL,
    body TEXT,
    created_at TIMESTAMPTZ NOT NULL DEFAULT now()
);

CREATE INDEX idx_content_course ON contents(course_id);
```

1.5 Relationship Tables (Many-to-Many Associations)

The following tables implement many-to-many relationships within the database schema, enabling structured associations between core entities of the platform. Specifically, they manage course-to-topic mappings to support subject categorization, instructor-to-course assignments to facilitate teaching responsibilities, and student enrollments to track participation and academic performance.

```sql
CREATE TABLE course_topics (
    course_id UUID REFERENCES courses(id) ON DELETE CASCADE,
    topic_id UUID REFERENCES topics(id) ON DELETE CASCADE,
    PRIMARY KEY (course_id, topic_id)
);

CREATE INDEX idx_course_topics_topic ON course_topics(topic_id);

CREATE TABLE teaches (
    instructor_id UUID REFERENCES instructors(user_id) ON DELETE
       CASCADE,
    course_id UUID REFERENCES courses(id) ON DELETE CASCADE,
    PRIMARY KEY (instructor_id, course_id)
);

CREATE INDEX idx_teaches_course ON teaches(course_id);

CREATE TABLE enrollments (
    student_id UUID REFERENCES students(user_id) ON DELETE
       CASCADE,
    course_id UUID REFERENCES courses(id) ON DELETE CASCADE,
    marks INT CHECK (marks BETWEEN 0 AND 100),
    created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
    PRIMARY KEY (student_id, course_id)
);

CREATE INDEX idx_enroll_course ON enrollments(course_id);
```

List of Functionalities Implemented

1. User Authentication & Role Based Authorization

The system implements a secure and scalable user authentication mechanism to ensure that only authorized users can access the platform. The authentication system is designed to support multiple user roles while maintaining security and session integrity.

Technology Stack Used

The authentication module is implemented using the following technology stack:

- **Frontend:** Next.Js, Tailwind-Css
- **Backend:** Next.Js
- **Database:** PostgreSQL
- **Authentication Mechanism:** JSON Web Token (JWT)
- **Password Security:** Password hashing using bcrypt

Password Security

Passwords are never stored in plain text. During registration, user passwords are hashed using a secure hashing algorithm (bcrypt) before being stored in the database.

When a user attempts to log in, the entered password is compared with the stored hashed password using secure comparison methods. This ensures protection against password theft and database compromise.

JWT-Based Authentication

The system uses JSON Web Tokens (JWT) to manage authenticated sessions.

JWT is a compact, URL-safe token format used to securely transmit information between the client and server. After successful login:

1. The server verifies user credentials.
2. If valid, a JWT token is generated.
3. The token contains encoded user information such as:

    - User ID
    - User Role

4. The token is sent to the client.
5. The client stores the token and includes it in future requests.

Protected routes on the server verify this token before granting access. This ensures secure session management without storing session data on the server.

User Roles

The system supports four distinct user roles:

- Student
- Instructor
- Data Analyst
- Admin

All users log in through the same login interface. Role-based access is determined after authentication using the role stored in the JWT token.

Sign-Up and Access Control Policy

The system enforces controlled registration rules:

- **Students and Instructors:** Can register independently using the public sign-up form.
- **Admin and Data Analyst:** Cannot self-register.
- Only an existing Admin has the authority to create new Admin or Data Analyst accounts.

This controlled registration mechanism prevents unauthorized privilege escalation and ensures administrative integrity.

Logout Mechanism

Logout is implemented by removing the JWT token from the client side. Once removed, protected routes can no longer be accessed without re-authentication.

The authentication system ensures secure login, controlled access, password protection, and scalable session management for all user types.

2. Course Management

The Course Management module enables the system administrator to create, update, and manage courses offered on the platform. This functionality ensures structured organization of academic content and proper assignment of instructional responsibilities.

Administrative Control

Only users with the **Admin** role are authorized to perform course management operations. This restriction ensures centralized control and prevents unauthorized modifications.

The Administrator can:

- Create new courses
- Update course details
- Delete or deactivate courses
- Assign instructors to courses

Course Attributes

Each course is categorized and structured using the following attributes:

- **Course Title**
- **Program Type** (e.g., Undergraduate, Postgraduate, Certification, Diploma)
- **Duration** (e.g., 6 weeks, 3 months, 1 semester)
- **Associated University**
- **Course Description**

These attributes allow structured classification and easier searching/filtering of courses within the platform.

Instructor Assignment

The system supports a flexible instructor allocation model:

- A single course may have **multiple instructors**.
- An instructor may be assigned to **multiple courses**.

This establishes a **many-to-many relationship** between courses and instructors.

Only the Administrator has the authority to assign or modify instructor allocations for any course. This ensures proper oversight and prevents unauthorized role escalation.

Course Lifecycle Management

The Administrator can manage the lifecycle of a course by:

- Activating or deactivating courses
- Updating course metadata
- Reassigning instructors when necessary

This structured management mechanism ensures scalability and consistency in handling academic offerings.

The Course Management module ensures centralized administrative control, structured categorization, and flexible instructor allocation, enabling efficient academic program organization within the system.

3. Student Course Enrollment

The Student Course Enrollment module allows students to browse available courses and enroll directly through the platform.

Course Search and Viewing

Students can:

- Search for courses based on keywords, program type, or university
- View course details such as description, duration, and assigned instructors

To ensure efficient and fast search operations, **database indexing** has been implemented on frequently queried attributes such as course title, program type, and associated university. This significantly improves query performance and reduces search time, especially when handling large datasets.

Enrollment Process

Once a student selects a course:

- The student can enroll through the enrollment interface.
- The system records the enrollment in the database.
- A student may enroll in multiple courses.

Access to course materials is granted only after successful enrollment.

This module ensures a smooth enrollment experience while maintaining efficient database performance through optimized indexing techniques.

4. Instructor Content Upload and Evaluation

The Instructor Content Management module allows instructors to upload and manage academic resources related to the courses assigned to them.

Content Upload

Only instructors assigned to a specific course are authorized to upload and manage its learning materials. The system allows instructors to upload:

- Lecture notes
- Assignments
- Notes, Books
- External resource links (e.g., reference materials or video links)

Each uploaded resource is associated with a specific course and is accessible only to students enrolled in that course.

Student Evaluation

In addition to content management, instructors are responsible for evaluating students enrolled in their courses. The system allows instructors to:

- Assign marks to individual students
- Update or modify marks when necessary

Marks are stored securely in the database and are linked to both the student and the respective course.

This module ensures structured content delivery and academic evaluation while maintaining role-based access control.

5. Analytics and Statistics Module

The Analytics and Statistics module is one of the core components of the system. It provides structured insights into course performance, student outcomes, and overall academic trends. This module is accessible exclusively to users with the **Data Analyst** role and is designed to support data-driven academic decision making.

Purpose of the Module

The primary objective of this module is to transform raw academic data into meaningful statistical insights. It aggregates course, enrollment, and evaluation data to generate measurable indicators such as:

- Total student enrollment per course
- Course popularity
- Student performance metrics
- Grade distribution analysis
- Instructor contribution and activity
- Geographic distribution of students

All analytics are generated dynamically using structured database queries and aggregation functions.

Course-Level Overview

For each course, the Data Analyst dashboard provides high-level summary statistics, including:

- **Total Students Enrolled** – Indicates the overall enrollment count for the course.
- **Average Marks** – Computed using aggregate functions (e.g., AVG) on student marks.
- **Pass Rate** – Percentage of students scoring above the passing threshold.
- **Highest Score** – Maximum marks obtained in the course.

These metrics provide a quick performance snapshot of any selected course.

Grade Distribution Analysis

The system categorizes student marks into predefined grade brackets:

- A (90–100)
- B (75–89)
- C (60–74)
- D (50–59)
- F (0–49)

For each grade range, the dashboard displays:

- Number of students in that range
- Visual distribution bars for comparative understanding

This allows the Data Analyst to evaluate:

- Academic difficulty level of the course
- Distribution fairness
- Whether performance is skewed toward high or low grades

If most students fall under high-grade brackets (A or B), the course may indicate strong performance or easier assessment criteria. Conversely, clustering in lower brackets may indicate course difficulty or instructional gaps.

Top Performers Ranking

The system also identifies and ranks the highest-performing students in a course.

For each top performer, the dashboard displays:

- Rank position
- Student name
- Country
- Obtained marks

This ranking system enables the Data Analyst to:

- Identify academic excellence
- Compare performance trends across different courses
- Analyze consistency in high achievers

Ranking is generated using sorted queries based on marks in descending order.

Student Distribution by Country

The module includes geographic analytics to display the distribution of enrolled students based on their country.

For each country, the dashboard presents:

- Number of enrolled students
- Comparative visual representation

This enables analysis of:

- International diversity within a course
- Regional participation trends
- Expansion opportunities for specific geographic regions

Such insights are particularly useful for institutional strategy and outreach planning.

Course Popularity and Enrollment Trends

Course popularity is measured using total enrollment count. By comparing enrollment across courses, the Data Analyst can:

- Identify highly demanded programs
- Detect under-enrolled courses
- Recommend improvements or restructuring

Enrollment metrics are derived using COUNT aggregation queries on the enrollment table.

Student Performance Metrics

The system provides statistical measures that allow performance tracking at multiple levels:

- Average performance per course
- Maximum score comparison
- Pass percentage analysis

These metrics help determine:

- Overall academic effectiveness
- Assessment quality
- Instructor performance impact

Instructor Activity Insights

Although instructors manage course content and evaluations, the Data Analyst can observe indirect indicators of instructor contribution through:

- Student performance outcomes
- Grade distribution patterns
- Course enrollment numbers

Courses with consistently strong performance and high enrollment may indicate effective instructional engagement.

Database Aggregation and Query Optimization

All analytics are computed using structured SQL aggregation functions such as:

- COUNT()
- AVG()
- MAX()
- GROUP BY clauses

Indexes implemented on course identifiers and enrollment tables ensure efficient computation even with large datasets.

Significance of the Module

This module elevates the system beyond a simple course management platform by introducing data-driven evaluation capabilities. It enables:

- Evidence-based academic decisions
- Transparent performance monitoring
- Institutional performance comparison
- Strategic planning support

The Analytics and Statistics module ensures that academic data is not only stored but actively utilized to derive meaningful insights, making the platform both operationally efficient and analytically powerful.

6. Form Handling, Data Validation, and Responsive User Interface

This section describes the system's implementation of structured web forms, validation mechanisms, and responsive interface design to ensure usability, correctness, and accessibility across devices.

Web Form Handling

The platform uses structured web forms to facilitate user interaction and data submission. Forms are implemented for the following operations:

- User registration (Student and Instructor)
- User login
- Course creation and modification (Admin)
- Content upload (Instructor)
- Course enrollment (Student)

Each form communicates with the backend through secure HTTP requests. Submitted data is processed by server-side controllers, which interact with the database to perform the required operations. Role-based access control ensures that only authorized users can access specific forms.

Data Validation Mechanisms

To ensure data integrity and security, the system implements both client-side and server-side validation.

Client-Side Validation:

- Required field checks
- Email format validation
- Password strength validation
- Input length constraints

Client-side validation improves user experience by providing immediate feedback before form submission.

Server-Side Validation:

- Verification of required fields
- Role-based authorization checks
- Duplicate record prevention (e.g., duplicate email registration)
- Data type and format validation

Server-side validation ensures that malicious or manipulated inputs cannot bypass security controls. Parameterized queries are used to prevent SQL injection attacks.

Responsive User Interface

The platform is designed with a responsive layout to ensure compatibility across desktops, tablets, and mobile devices.

Key interface features include:

- Adaptive layouts using flexible grid systems
- Scalable components and dashboards
- Mobile-friendly navigation
- Role-specific dashboards customized for:
  - Admin
  - Student
  - Instructor
  - Data Analyst

Each user role is presented with a tailored dashboard containing only relevant features and statistics. This improves usability, reduces clutter, and enhances workflow efficiency.

By integrating structured form handling, comprehensive validation mechanisms, and responsive interface design, the system ensures secure data processing, improved user experience, and accessibility across multiple device types.

Programming Languages Used

The primary language used for developing the web application will be **JavaScript**, which will be utilized in both frontend and backend development through the Next.js framework and related libraries. For database design and data management, **SQL** will be used to define schemas, manage relational data, and perform queries within the PostgreSQL database system.

Front-End and Back-End Tools Used

Frontend Technologies

- **Next.js** – Will be used to build the web interface, handle routing, and support server-side rendering.
- **Tailwind CSS** – Will be used for styling and responsive UI design.
- **Zustand** – Will be used for client-side state management.
- **Lucide React** – Additional frontend libraries for UI animation and interface components.

Backend Technologies

- **Next.js** - will be used to handle API routes, process server-side logic, and manage communication between the frontend and the database.
- **PostgreSQL** – Will be used as the relational database system.
- **JWT (JSON Web Token)** – Will be used for secure authentication and session handling.
- **Vitest** - will be used as a testing framework to perform unit and integration testing, ensuring the reliability and correctness of application components.
- **bcrypt, jsonwebtoken** – Additional backend libraries for authentication and security handling.

Deployment

- **Vercel** – Will be used for hosting and deployment of the web application.