Design Document

Vu Ai Thanh Truong Gia Bao Nguyen Tuan Anh

20-05-2025

Table of contents

1 Conceptual & Logical Design

The project is aimed to provide a yet minimal but still functional of a management application for a education institution. The main actors that interacts to the app contains: `Administrators`, `Instructors`, and `Students`. There activities will go around the `Courses` at school.

1.1 Functional and Non-functional Designs

1.1.1 Functional

1.1.1.1 User Authentication and Authorization:

- The system must allow users (students, instructors, and admins) to log in using email and password credentials.
- Users must be assigned roles (student, instructor, admin) with role-specific access to features (e.g., students can enroll/unenroll, instructors can view/manage courses, admins can manage users and courses).

1.1.1.2 Course Management

- The system must display a list of courses with filters by course name, department, and schedule, supporting pagination (10 courses per page).
- Users can view detailed course information (e.g., ID, name, department, instructor, location, schedule, semester, availability) on a course detail page.
- Instructors and admins can add, edit, and delete courses, while students can only view course details.
- The system must track and display enrolled students for each course.

1.1.1.3 Enrollment Management:

- Students must be able to enroll in a course if availability is greater than zero and they are not already enrolled, with a confirmation dialog before submission.
- Students must be able to unenroll from a course if they are enrolled, with a confirmation dialog before submission.

- The system must update course availability automatically when students enroll or unenroll.
- The system must prevent enrollment if the course is full (availability ≤ 0) or if the student is already enrolled.

1.1.1.4 User Interface and Navigation:

- The system must provide a responsive web interface using Bootstrap for consistent styling across devices.
- Each role (student, instructor, admin) must have a dedicated dashboard with role-specific options.
- Flash messages must be displayed to provide feedback on actions (e.g., success or error messages for login, enrollment).

1.1.1.5 Analytics

- For management roles like `Administrators` and `Instructors`, they will have access to several dashboards to report about the courses at the institution, or the courses that the instructors are involving in.

1.1.2 Non-functional

- **Performance**:
  - Having high uptime, with all errors are being handled to prevent crashes.
  - Queries should execute within a reasonable time, with minimal complexity. Current target is 0.5 seconds for a dataset of 1000 enrollments.
- **Security**:
  - Passwords are encrypted when being stored in the database, with supports of password hashing function `bcrypt`.
  - Access control systems based on roles are designed properly, prevent unauthorized information access or disclosure.
  - Preventions or mitigations of common web security vulnerabilities as well as applications using SQL database system: SQL Injection, Cross-site Scripting, etc.
- **Scalability**:
  - The application is scalable with increasing numbers of users in an acceptable level, against around 300 users under peak time.
- **Usability**:

  - Provide easy-to-navigate interface, with responsive User Interface with `Bootstrap` library.
- **Reliability**:
  - Ensure data integrity with normalized database design (3NF) and constraints.

1.2 Entity-Relationship Diagram

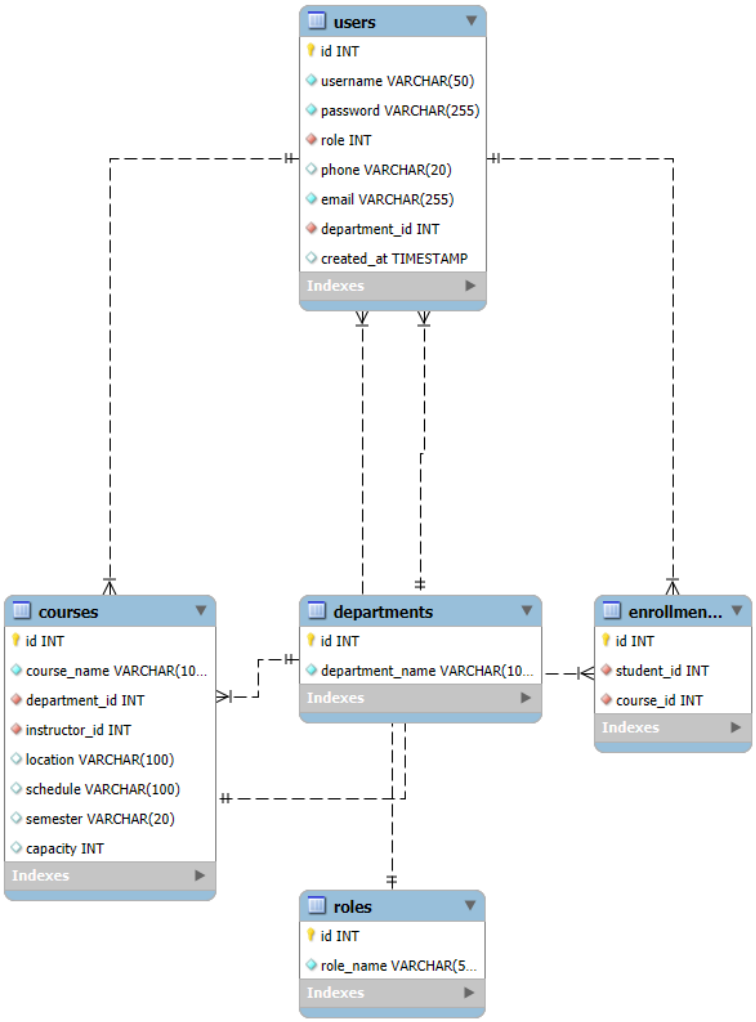

Figure 1: Entity-Relationship Diagram of the project

1.3 Proof of Normalization Form

1.3.1 First Normal Form (1NF):

- All tables have primary keys
- All attributes are atomic
- No repeating groups

1.3.2 Second Normal Form (2NF):

- All non-key attributes are fully functionally dependent on their primary keys
- No partial dependencies exist

1.3.3 Third Normal Form (3NF):

- No transitive dependencies
- Proper use of foreign keys for references
- Related data is properly normalized into separate tables

2 Physical Schema Definition

2.1 Complete DDL scripts of all tables

The full script is in our GitHub repository at this link.

2.2 Definitions of views, indexes, partitioning strategy

2.2.1 Views

Currently, there is one view `view_course_details` in the database. It provides a simplified and denormalized view of course information with instructor details. This view joins `courses` and `users` tables to provide course information along with the instructor's name, making it easier to retrieve complete course information without writing complex joins in application queries.

2.2.2 Indexes

The database schema defined several indexes on operations that are queried frequently during the application lifecycle.

1. `idx_users_email` on users(email) - Improves login and user lookup operations
2. `idx_users_role` on users(role) - Enhances queries filtering users by role type
3. `idx_courses_instructor` on courses(instructor_id) - Optimizes searches for courses by instructor
4. `idx_enrollments_student_course` on enrollments(student_id, course_id) - A composite index that improves enrollment lookups and enforces the unique constraint preventing duplicate enrollments

3 Task Division & Project Plan

3.1 Task Division

Member	Role	Detailed Description
Vu Ai Thanh	Project Leader, Main Developer	+ Design the database system & blueprints of the application+ Implement core functions in administrator's role & analytics
Truong Gia Bao	Main Developer	+ Implement core functions in students & instructor roles+ Design UX/UI of the application

Member	Role	Detailed Description
Nguyen Tuan Anh	Developer	+ Refactoring minor issues on codebases+ Documentation writer

3.2 Project Plan

The project spans from May 6 to May 26, 2025, with the following timeline:

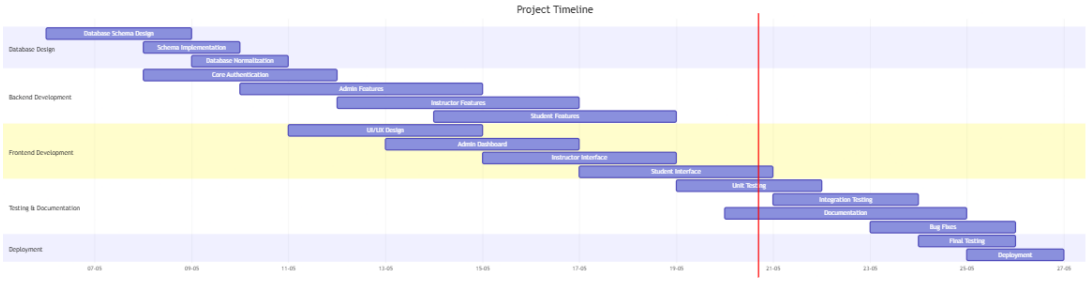

Figure 2: Gantt Chart of the Project

This timeline shows the major phases and tasks of the project, with overlapping periods where team members work in parallel on different components. The project is structured to begin with database design and core backend development, followed by frontend implementation, and concluding with thorough testing and documentation.

4 Supporting Documentation

4.1 Design Rationale

4.1.1 Tables

The project is centering around the normal operations of course enrollment in education institutes. Therefore, our database designs will involves three main actors:

- Administrators: People who manage the main direction of courses in the institutes
- Instructors: People who are in charge of more micro levels: teach, and manage the classes.
- Students: People who take the courses.

From these descriptions, we are adding `Courses` that are provided by the school, taught by instructors; as well as `Enrollments` to solve the involvement of students into the courses. There are several supporting tables such as `roles` and `departments` to provide the access control system in the business logic of the application, or to bring the application more realistic.

4.1.2 Keys

As any other systems, all objects in our database is managed via their primary key (via ID of administrators, instructors, students, courses, etc.). With the enrollments of student, as it is a many-to-many relationship, we bridge them with one more table: `Enrollments`. The

foreign keys referenced in the tables showing the relationship between the objects in the system: Each students & instructors are in departments of the institute, each students' courses must be managed by a department and an instructors, etc.

4.2 Notes on data loading approaches

In the implementation, we are using random data to provide a quick test bench for the application. The data generate script is available at the project folder. In the future, the script can increase up the load for security & performance testing.