

Group Project

High Level Design



SuBmitted By:

Amritpal Kaur

Changkai Zou

Han Wang

Yi Wu

**1. Introduction**

**About the System**  
The Academic Exchange Platform is a Java based web application that enables academic professionals and institutions to connect for educational collaboration and teaching opportunities. Academic Institutions can post course offerings, while Academic Professionals can search, view, and request to teach these courses based on expertise.

**Purpose of the Document**  
This document provides a high-level design overview of the Academic Exchange Platform, detailing the system's architecture, components, design approach, and testing methodology. It is intended as a blueprint for developers and stakeholders to understand the structural and functional aspects of the platform.

**2. Targeted Audience**

This document is intended for:

* **Developers**: For understanding the design and implementation guidelines.
* **System Architects**: To review and refine architectural decisions.
* **Project Managers**: For tracking project development and ensuring alignment with requirements.
* **Quality Assurance (QA) Team**: For reference to design elements during testing.
* **End Stakeholders**: For insight into the functional and technical capabilities of the system.

**3. Scope**

**In-Scope**

* Functional requirements, including registration, profile management, course listing, search, and notifications.
* Core architectural components, such as Servlets, JSP, and the database design.
* Use case, class, and component diagrams.
* Security and deployment considerations.

**Out-of-Scope**

* Advanced UI/UX features not specified in the requirements.
* Integration with third-party APIs not covered by the project requirements.
* Future scalability considerations beyond the initial deployment.

**4. Application Architecture**

**High-Level Architecture**  
The platform has Servlets serving as controllers, JSPs as the view layer, and Java classes and DAOs as the model for managing data. JDBC handles direct database interaction, with MySQL for data persistence.

A screenshot of a computer

Description automatically generated

**Main Components**

* **Presentation Layer (Client)**:

A web interface for user interactions, allowing Academic Professionals and Academic Institutions to interact with the system.

* **Service Layer**:

**Authentication Service**: Manages login/logout functionality.

**Profile Management Service**: Handles profile creation and updates for both Academic Professionals and Academic Institutions.

**Course Management Service**: Allows institutions to create, edit, and manage courses.

**Search and Filtering Service**: Provides search functionality for Academic Professionals to filter courses.

**Notification Service**: Sends notifications to users regarding the status of their requests.

* **Data Access Layer**:

Interfaces with the relational database (AEP) to perform CRUD (Create, Read, Update, Delete) operations for all entities, including users, courses, and requests.

* **Database**:

Stores user data, course listings, search logs, and notifications.

**6. Business Architecture**

**Use Case Diagrams**

A diagram of a diagram

Description automatically generated

* **User Registration and Authentication Use Case**: Covers the registration and authentication of both Academic Professionals and Academic Institutions.
* **Profile Completion:** Upong successful registration, users complete their profiles. Academic professionals add details like their current institution, academic position, and area of expertise. Academic institutions add information such as their address and course offerings.
* **Course Search By Academic Professional Use Case**: Academic professionals can search for available courses based on criteria such as institution name, course code, title, term, schedule, and delivery method. This allows professionals to find relevant opportunities aligned with their expertise.
* **Request to Teach Use Case**: After finding a suitable course, an academic professional can submit a request to teach it. The institution then receives the request and has the option to accept or reject it based on the professional’s qualifications.
* **Notification System Use Case**: Upon accepting or rejecting a request to teach, the system sends a notification to the academic professional. Notifications inform professionals about the status of their teaching requests.
* **Profile and Course Update Use Case**: Users can update their profiles, and institutions can modify course listings as needed to keep information current (e.g., changing course schedules or updating qualifications).
* **User Logout**: After completing their tasks on the platform, users can log out securely to protect their account information.

**7. Detailed Design**

**a. Class Diagrams**

In this section, you would define the classes needed for each layer of the architecture, especially focusing on the Service Layer and Data Access Layer. Here’s an outline of the primary classes:

1. **Authentication Classes**

* User: Base class (interface) with properties like userId, email, and password.
* AcademicProfessional (extends User): Includes additional fields like currentInstitution, position, areaOfExpertise.
* AcademicInstitution (extends User): Includes fields like institutionName, address, and currentCourses.

1. **Profile Management Classes**

* ProfileService(interface): Manages profile data, including creation and updates.
* AcademicProfessionalProfile: Specific class for academic professionals’ profiles, with properties like educationBackground and expertise.
* InstitutionProfile: Class for institution profiles, including address and courseOfferings.

1. **Course Management Classes**

* Course: Represents course details such as courseId, title, code, term, schedule, deliveryMethod, preferredQualifications, and compensation.
* CourseService: Handles CRUD operations for Course objects.
* RequestToTeach: Class that represents a teaching request, with properties like requestId, courseId, professionalId, status.

1. **Search and Filtering Classes**

* SearchService: Provides search functionality with methods like searchByInstitution, searchByCourseCode, and searchBySchedule.
* SearchFilter: Contains criteria like institutionName, courseCode, schedule, and deliveryMethod.

1. **Notification Classes**

* NotificationService: Manages notifications, with methods for creating, updating, and querying notifications.
* Notification: Represents notifications with fields like notificationId, userId, message, status, timestamp.

1. **Data Access Classes (DAOs)**

* UserDAO, CourseDAO, RequestDAO, etc.: Data access objects for each primary entity ( User, Course, RequestToTeach). Each DAO handles CRUD operations for its respective entity.
* DatabaseConnection: Manages the JDBC connection to the MySQL database.

**b. Component Diagram**

A diagram of a company

Description automatically generated

1. **Data Architecture**
   * 1. **Users Table**

* Fields: UserID (PK), Name, Email, Password, UserType (e.g., academic professional or institution).
* Description: Stores basic user information. The UserType differentiates between academic professionals and institutions.
  + 1. **AcademicProfessionals Table**
* Fields: ProfessionalID (PK, FK to Users), Institution, AcademicPosition, EducationBackground, AreaOfExpertise.
* Description: Extends the Users table to hold details specific to academic professionals.
  + 1. **AcademicInstitutions Table**
* Fields: InstitutionID (PK, FK to Users), Address, CoursesOffered.
* Description: Extends the Users table for institutions, storing additional fields specific to academic institutions.
  + 1. **Courses Table**
* Fields: CourseID (PK), InstitutionID (FK to AcademicInstitutions), Title, Code, Term, Schedule, DeliveryMethod, QualificationsRequired, Compensation.
* Description: Stores details about courses offered by institutions. The InstitutionID establishes a relationship with the AcademicInstitutions table.
  + 1. **Requests Table**
* Fields: RequestID (PK), CourseID (FK to Courses), ProfessionalID (FK to AcademicProfessionals), RequestDate, Status (pending, accepted, or rejected).
* Description: Tracks teaching requests made by academic professionals for specific courses. The Status field indicates the outcome of the request.
  + 1. **Notifications Table**
* Fields: NotificationID (PK), RequestID (FK to Requests), ProfessionalID (FK to AcademicProfessionals), NotificationText, NotificationDate.
* Description: Stores notifications sent to professionals, linking to a specific request for status updates.

A diagram of a computer

Description automatically generated with medium confidence

**9. Security Architecture**

* User Authentication: Passwords are stored securely in the Users table using a hashing algorithm.
* Access Control: Each user role (academic professional or institution) has specific privileges within the application, enforced by the backend.
* Data Integrity: Foreign key constraints ensure referential integrity, while transactions in CRUD operations prevent partial data updates.

**10. Deployment Architecture**

The platform will run on a web server (Apache Tomcat) hosting the JSP and Servlet components, with MySQL as the database on the backend.

**11. Testing Model**

**Unit Testing with JUnit**

* Unit tests for core functionality such as registration, authentication, and request processing.
* Tests organized in a separate test package to ensure modularity and maintainability.

**Integration Testing**

* Integration tests between Servlets and DAOs to ensure proper data retrieval and processing.

Top of Form