Course Enrollment

& Classroom Scheduling System

Lecturer: Roeun Pacharoth

Student: Han Raby

Heng Oulong

Chea Chanminea

Hun Lyhorng

Lo Bunleang

Team Member & Roles

Han Raby

- Authentication
- Security
- Student-related features such as profile management and access control.

Lo Bunleang

- Schedule for both admin and student,
- Classroom management for both admin and student
- Timetable handling

Heng Oulong

- Course management for both admin and student
- Enrollment management for both admin and student

Chea Chanminea

- Database design
- Entity management along with admin dashboard and sidebar controllers

Hun Lyhorng

- User management features
- Home page controller for system navigation

INTRODUCTION

THE ESSENTIAL OF COURSE ENROLLMENT SYSTEM

A Course enrollment is essential in universities

B Manual systems are inefficient

C Digital systems improve accuracy

D Our project provides an online solution

Course Enrollment System

PROBLEM STATEMENT

WHAT EXACTLY IS WRONG AND WHY MUST WE FIX IT?

01

Manual enrollment causes delays

The first major problem is delay. Manual enrollment requires students to submit forms physically and wait for approval from administrators

02

High risk of data duplication

Another issue is data duplication. When data is recorded manually, the same student information may be entered multiple times.

03

Difficult to manage student records

Managing student records is also very difficult in manual systems. Information is often scattered across paper documents or multiple files.

04

No real-time enrollment tracking

Additionally, manual systems do not support real-time tracking of enrollments.

Project Objective

1. Digitally manages courses, lecturers, and enrollments
2. Enforces business rules automatically
3. Separates responsibilities between Admin and Student

Project Objective

4. Provides real-time feedback and validation

5. Supports modern UI and cloud-based resources

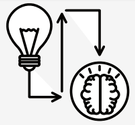
Project WorkFlow

Planning & Research

Requirements gathering, technology selection, team role assignment, research on enrollment systems

Development Sprints

5 sprints covering foundation, admin module, student module, advanced features, testing & refinement



Design & Architecture

ER diagram design, UI/UX mockups, system architecture planning, database structure finalization

Deployment & Documentation

System deployment preparation, user and technical documentation, presentation preparation

Development Sprints Breakdown

Sprint 1: Foundation & Authentication

Database entity setup, security configuration, authentication system, basic project structure with Gradle

Sprint 2: Admin Module Development

Admin dashboard, user management, course management, classroom management modules

Sprint 3: Student Module Development

Student profile management, course browsing and enrollment, schedule viewing, home page and navigation

Sprint 4: Advanced Features

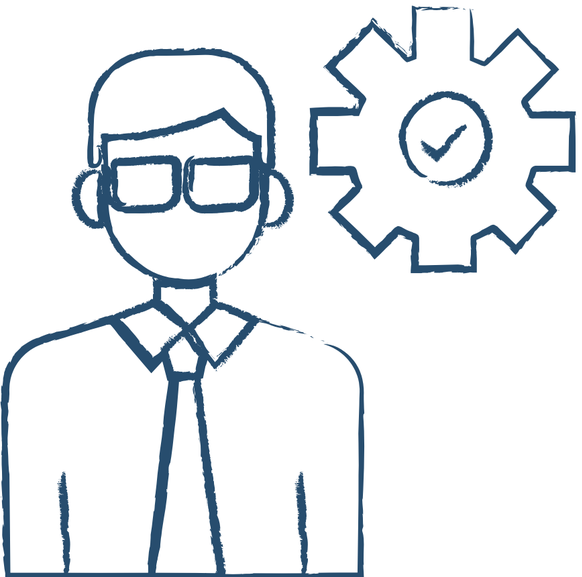
Enrollment processing system, schedule conflict resolution, integration between admin and student modules

Sprint 5: Testing & Refinement

Unit and integration testing, UI/UX refinement with Tailwind CSS, bug fixes and optimization

Scope Of System

- Admin-controlled course enrollment system
- Supports Admin and Student users
- Lecturer role used only for course assignment
- Manages users, courses, and enrollments, classroom and schedule



Key Features & Functionalities

Admin

. Create, update, and delete user accounts

. Assign roles and manage user status

. Create, update, and delete courses

. Assign lecturers to courses

. Manage student enrollments, classroom, and schedule

. View system data and dashboards

Student

. Register and log in to the system

. View available courses

. Enroll in or withdraw from courses

. View / edit personal profile and enrollment history

. View schedule

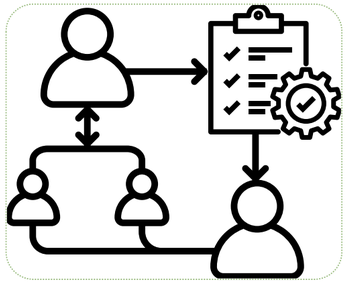
Lecturer

. Used for assigning responsibility to courses

. Does not have login access or system features

Users & Roles Design

Admin



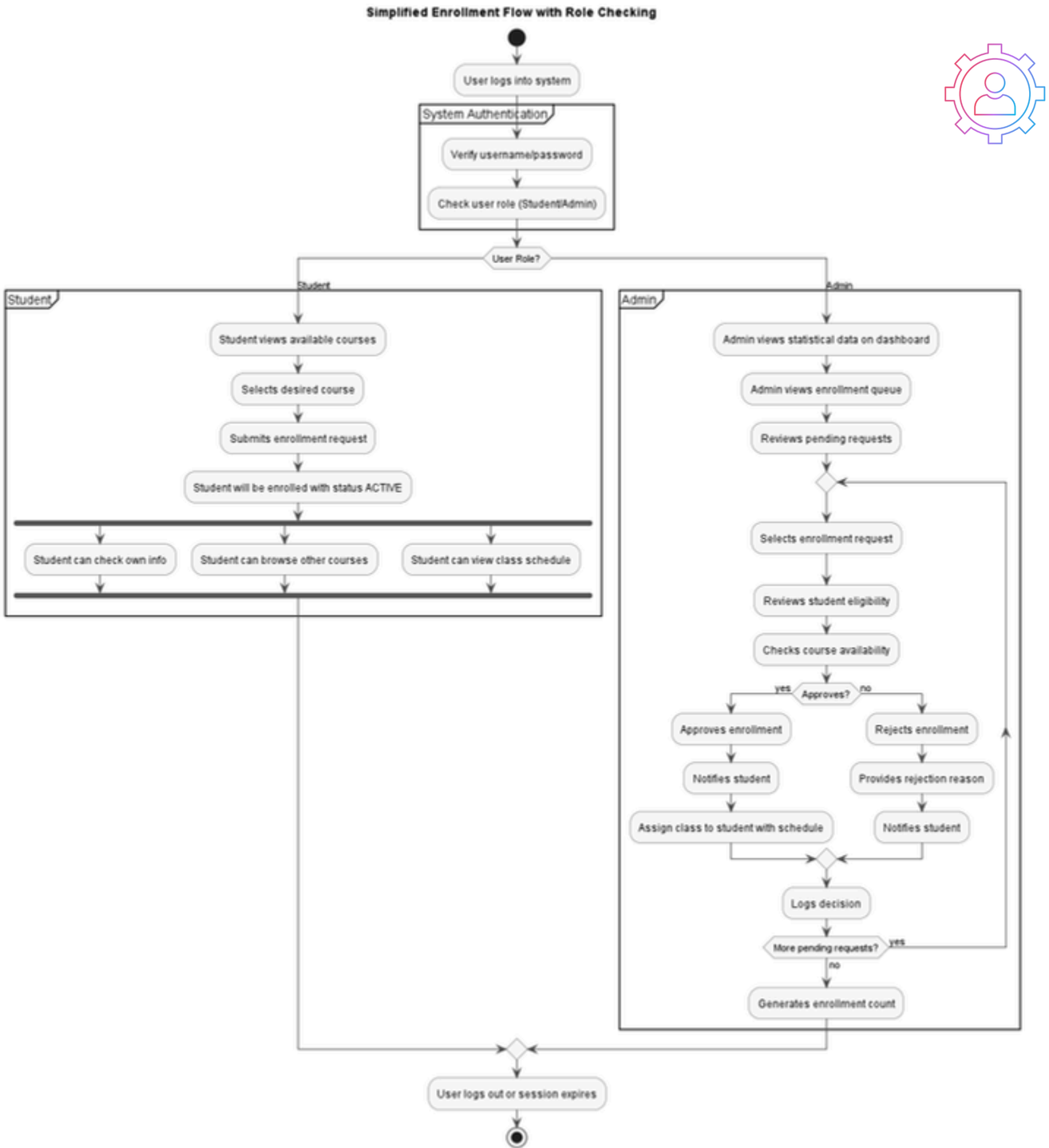
Student

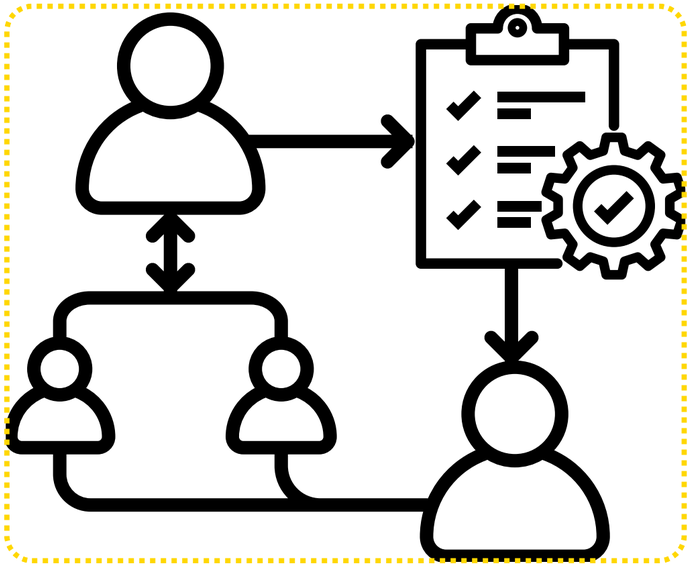
Role-based access control (RBAC) using Spring Security

Users table stores all personal info + role_id (foreign key)

Roles table: role_id, role_name (enum), description

Users & Roles Design

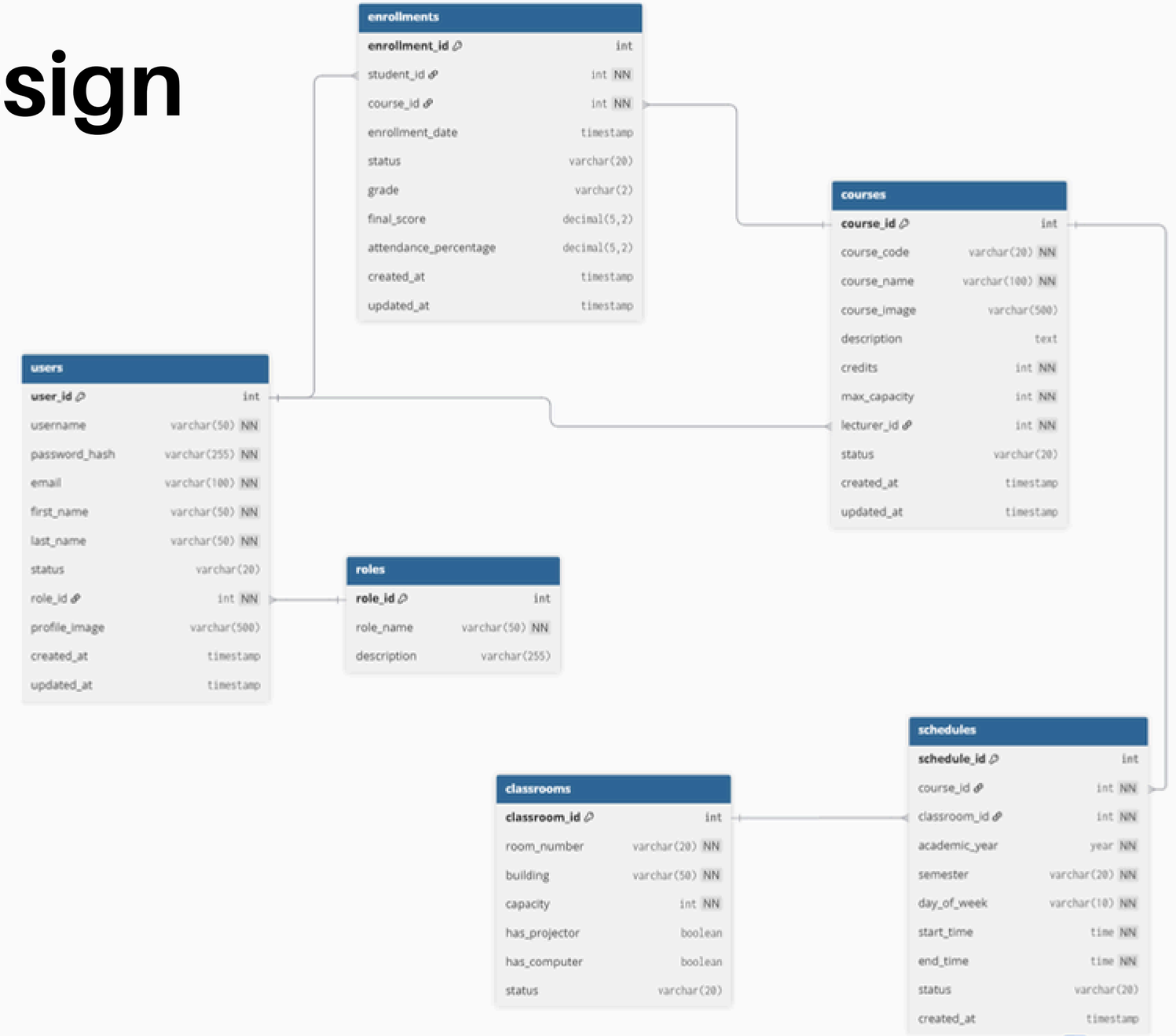




Database Design

DB

- 6 main entities: users, roles, courses, classrooms, schedules, enrollments
- Normalized to 3NF (no redundancy, no duplicate data)
- Uses foreign keys, unique constraints, indexes for performance



API Endpoint Design

RESTful style (even for Thymeleaf, endpoints follow REST)

Secured with Spring Security (role-based)

Uses HTTP methods (GET, POST)

RBAC

- /admin/** → ADMIN only
- /student/** → STUDENT only
- Public: /login, /register

Ex: GET/admin/dashboardAdmin

- Description (overview + stats)
- Role(ADMIN)
- ResponseType(HTML)

API Endpoint Design

Role	Number of Endpoints	Examples
Public	2	/login, /register
Student	10	/student/profile, /student/courses
Admin	20	/admin/users, /admin/courses
Other	3	/home, /logout
Total	**35**	

API Endpoint Design

#	HTTP Method	URL Path	Description	Controller File	Role Access (via @PreAuthorize)	Returns (HTML/Redirect)
1	GET	/student/profile	Show student profile (personal info + enrollments)	StudentProfileController	STUDENT	student/profile
2	POST	/student/profile/update	Update profile (name, email, DOB)	StudentProfileController	STUDENT	redirect:/student/profile
3	POST	/student/profile/upload-image	Upload profile image	StudentProfileController	STUDENT	redirect:/student/profile
4	POST	/student/profile/drop-enrollment	Drop a course enrollment	StudentProfileController	STUDENT	redirect:/student/profile
5	GET	/student/home	Student homepage (list all courses)	StudentController	STUDENT	student/home

API Endpoint Design

#	HTTP Method	URL Path	Description	Controller File	Role Access (via @PreAuthorize)	Returns (HTML/Redirect)
6	GET	/student/courses	List all available courses (with search)	StudentCourseController	STUDENT	student/course/courses
7	GET	/student/courses/{id}	View details of a single course	StudentCourseController	STUDENT	student/course/course-details
8	GET	/admin/adminDashboard	Admin dashboard with stats	AdminController	ADMIN	admin/adminDashboard
9	GET	/classrooms	List all classrooms (admin)	AdminController	ADMIN	admin/classroom/list
10	GET	/classrooms/create	Show create classroom form	AdminController	ADMIN	admin/classroom/form
11	GET	/schedules	List all schedules (admin)	AdminController	ADMIN	admin/schedule/list

API Endpoint Design

#	HTTP Method	URL Path	Description	Controller File	Role Access (via @PreAuthorize)	Returns (HTML/Redirect)
12	GET	/schedules/create	Show create schedule form	AdminController	ADMIN	admin/schedule/form
13	GET	/admin/schedules	List all schedules	AdminScheduleController	ADMIN	admin/schedule/list
14	GET	/admin/schedules/new	Show create schedule form (modal fragment)	AdminScheduleController	ADMIN	admin/schedule/form :: scheduleForm
15	POST	/admin/schedules	Create new schedule	AdminScheduleController	ADMIN	redirect:/admin/schedules
16	GET	/admin/schedules/{id}/edit	Show edit schedule form (modal fragment)	AdminScheduleController	ADMIN	admin/schedule/form :: scheduleForm

API Endpoint Design

#	HTTP Method	URL Path	Description	Controller File	Role Access (via @PreAuthorize)	Returns (HTML/Redirect)
17	POST	/admin/schedules/{id}	Update existing schedule	AdminScheduleController	ADMIN	redirect:/admin/schedules
18	POST	/admin/schedules/{id}/delete	Delete a schedule	AdminScheduleController	ADMIN	redirect:/admin/schedules
19	GET	/admin/enrollments	List all enrollments	AdminEnrollmentController	ADMIN	admin/enrollments
20	GET	/admin/enrollments/student/{studentId}	View enrollments for a specific student	AdminEnrollmentController	ADMIN	admin/enrollments
21	GET	/admin/enrollments/create	Show create enrollment form	AdminEnrollmentController	ADMIN	admin/enrollment-create

API Endpoint Design

#	HTTP Method	URL Path	Description	Controller File	Role Access (via @PreAuthorize)	Returns (HTML/Redirect)
17	POST	/admin/schedules/{id}	Update existing schedule	AdminScheduleController	ADMIN	redirect:/admin/schedules
18	POST	/admin/schedules/{id}/delete	Delete a schedule	AdminScheduleController	ADMIN	redirect:/admin/schedules
19	GET	/admin/enrollments	List all enrollments	AdminEnrollmentController	ADMIN	admin/enrollments
20	GET	/admin/enrollments/student/{studentId}	View enrollments for a specific student	AdminEnrollmentController	ADMIN	admin/enrollments
21	GET	/admin/enrollments/create	Show create enrollment form	AdminEnrollmentController	ADMIN	admin/enrollment-create

API Endpoint Design

#	HTTP Method	URL Path	Description	Controller File	Role Access (via @PreAuthorize)	Returns (HTML/Redirect)
22	POST	/admin/enrollments/create	Create new enrollment	AdminEnrollmentController	ADMIN	redirect:/admin/enrollments
23	POST	/admin/enrollments/delete/{id}	Delete an enrollment	AdminEnrollmentController	ADMIN	redirect:/admin/enrollments
24	GET	/admin/users	List all users	AdminUserController	ADMIN	admin/users
25	GET	/admin/users/create	Show create user form	AdminUserController	ADMIN	admin/user-create
26	POST	/admin/users/create	Create new user	AdminUserController	ADMIN	redirect:/admin/users
27	POST	/admin/users/{id}/status	Toggle user status (active/inactive)	AdminUserController	ADMIN	redirect:/admin/users

API Endpoint Design

#	HTTP Method	URL Path	Description	Controller File	Role Access (via @PreAuthorize)	Returns (HTML/Redirect)
28	POST	/admin/users/{id}/delete	Delete a user	AdminUserController	ADMIN	redirect:/admin/users
29	GET	/student/classrooms	List all classrooms (for student)	StudentClassroomController	STUDENT	student/classroom/list
30	GET	/student/enrollments	View my own enrollments	StudentEnrollmentController	STUDENT	student/enrollment/enrollments
31	POST	/student/enrollments/enroll	Enroll in a course	StudentEnrollmentController	STUDENT	redirect (back to referer)
32	POST	/student/enrollments/drop/{id}	Drop an enrollment	StudentEnrollmentController	STUDENT	redirect (back to referer)
33	GET	/home	Redirect to role-based dashboard	HomeController	Authenticated	redirect (/admin/adminDashboard or /student/home)
34	GET	/login	Show login/register page	AuthController	Public	auth/LoginForm

System Architecture

BROWSER

User clicks button, fills form, views page

↕

CONTROLLER

Entry point: Maps URLs to methods, receives DTO, calls Service

↕

DTO (Data Transfer Object)

Simple class with validation rules (@NotNull, @NotBlank)

↕

SERVICE

Business logic, validation, conflict detection, converts DTO ↔ Entity

↕

REPOSITORY

Database access: Extends JpaRepository, auto-generated SQL

↕

DATABASE

MySQL / PostgreSQL - Stores data and executes queries

Technology Stack

BACKEND

Java (JDK 21+)

Core programming language

Spring Boot

Application framework

Spring MVC

Controllers & request handling

Spring Data JPA (Hibernate)

ORM and database interaction

Spring Security

Authentication & authorization (Admin / Student roles)

Thymeleaf

Server-side HTML template engine

Technology Stack

FRONTEND

HTML5

Page structure

Thymeleaf

Dynamic rendering from backend data

Tailwind CSS

UI styling and responsive layout

Font Awesome

Icons (calendar, users, status indicators)

Live Demo

Conclusion

- The project successfully implements a modern layered system architecture using Spring Boot.
- Clear separation of concerns between Controller, Service, Repository, and Database.
- Role-based access ensures secure interaction for Admin and Student users.
- The system improves data consistency, maintainability, and scalability.
- Demonstrates how real-world academic systems can be built using clean architecture principles.

FUTUER ENHANCEMENT

Email Notifications

- Notify users of schedule changes
- Enrollment confirmations

1

Export to PDF

2

- Download and print schedules
- Student timetable reports

Lecturer Assignment

- Assign lecturers to schedules
- Check lecturer conflicts

3

Performance Optimization

4

- Database indexing
- Caching for faster loading

Thank You