STUDENT MANAGEMENT SYSTEM

Software Requirements Specification

Course:
Software Engineering

Instructor:
Trần Thị Mỹ Tiên

Group:
Group 01

Faculty / Department:
Faculty of Information Technology and Electrical-Electronic Engineering

University:
University of Transport Ho Chi Minh City – UTH

Semester:
Semester 1 – Academic Year 2025–2026

Date:
January 202

Table of Contents

1. Functional Requirements

1.1 Common Module (Authentication & Security)

Applicable to: Student, Lecturer, Administrator
This module manages user login, session handling, and security functions to ensure safe access to the system for all user roles.

FR-01: User Login

- ➔ The system shall allow users to log in using their registered Email/User ID and Password.
- ➔ The system shall check the entered credentials with the user data stored in the database.
- ➔ If the credentials are valid, the system shall create a secure session and redirect the user to the dashboard based on their role.
- ➔ If the credentials are invalid, the system shall display an error message.
- ➔ The system shall limit the number of login attempts to prevent brute-force attacks (for example, a maximum of 5 failed attempts before temporary account lock).

FR-02: User Logout

- ➔ The system shall provide a Logout option in the main navigation menu.
- ➔ When the user selects Logout, the system shall end the current session, clear all session data, and redirect the user to the login page.

FR-03: Change Password

- ➔ The system shall allow logged-in users to change their password.
- ➔ Users must enter their current password, a new password, and confirm the new password.
- ➔ The system shall enforce password rules, such as:
    - At least 8 characters.
    - Include uppercase letters, lowercase letters, numbers, and special characters.
- ➔ After a successful password change, the system shall notify the user and record the action for audit purposes.

FR-04: Forgot Password Recovery

- ➔ The system shall provide a Forgot Password option on the login page.
- ➔ The user shall enter their registered Email/User ID.
- ➔ The system shall verify that the entered information exists in the database.

➔ If valid, the system shall generate a password recovery code or link with a limited time (15 minutes).
➔ The system shall send the recovery code or link through an external Email System.
➔ The user shall reset the password using the code or link, following the same password rules defined in **FR-03**.

1.2 Student Module

Actor: Student
This module provides functions that allow students to manage their academic activities, view personal and academic information, and interact with the system during their study period.

FR-05: View Personal Information
➔ The system shall allow students to view their personal profile.
➔ The profile shall include Student ID, Full Name, Date of Birth, Class, Department, Contact Information (Phone Number, Email, Address), and Enrollment Status.
➔ Sensitive information shall be displayed as read-only to prevent unauthorized changes.

FR-06: Update Personal Information
➔ The system shall allow students to update non-critical personal information, such as Phone Number, Address, and Email.
➔ Students shall not be allowed to change core academic information, including Student ID, Full Name, Date of Birth, Class, and Department.
➔ The system shall validate the updated information (for example, email format and phone number length).
➔ All changes shall be recorded for audit purposes.
➔ After a successful update, the system shall display a confirmation message.

FR-07: View Weekly Schedule
➔ The system shall generate and display a weekly timetable based on the student's registered courses.
➔ The timetable shall show Day, Time, Course Name, Room, and Lecturer.
➔ The schedule shall be presented in a visual calendar format, such as a weekly grid.

FR-08: View Academic Results

➔ The system shall calculate and display the student's cumulative GPA based on completed courses and credit weights.
➔ The system shall classify the academic performance (for example, Excellent, Good) based on GPA rules.
➔ A summary academic transcript shall be provided with semester-by-semester results.

FR-09: View Notifications

➔ The system shall provide a notification section for students.
➔ Notifications may include announcements, system messages, and alerts from the Administrator.
➔ Each notification shall include a timestamp and content.

1.3 Lecturer Module

Actor: Lecturer
This module supports lecturers in managing teaching schedules, student lists, attendance records, and grading activities.

FR-10: View Assigned Schedule

➔ The system shall display the lecturer's teaching schedule.
➔ The schedule shall include Class Name, Course Code, Room, Teaching Days, Time Slots, and Number of Enrolled Students.
➔ The system shall show the schedule for the current semester and allow filtering by week or month.

FR-11: View Student List

➔ The system shall allow the lecturer to select a class from their assigned schedule.
➔ The system shall display a list of enrolled students, including Student ID, Full Name, Contact Information, and a summary of attendance and grades.
➔ The student list shall support search and sorting functions.

FR-12: Enter Grades

➔ The system shall provide an interface for lecturers to enter grades for each student.
➔ Grades shall be entered for different components, such as Midterm Exam, Final Exam using a 0–10 scale.
➔ The system shall automatically calculate weighted averages based on predefined grading rules for the course.

➔ After grades are finalized, the system shall:
- Save and lock the grades to prevent further changes
- Notify affected students through the integrated app notification system

FR-13: Update Grades

➔ The system shall allow lecturers to update grades within a limited period before the semester ends.
➔ All grade updates shall be recorded in the system audit logs.
➔ The system shall notify students by notification directly on the app when their grades are changed.

1.4 Training Department Module

Actor: Administrator
This module supports the Academic Affairs Office in managing academic data, system configuration, and reporting for the entire academic system.

FR-14: Manage Semesters

➔ The system shall allow the Academic Affairs Officer to create, update, and archive academic semesters.
➔ Each semester shall include start date, end date, and status (for example, In Progress, Closed, Finished).
➔ When the semester status changes, the system shall automatically apply related system rules for student management within that semester, such as enabling or disabling the entry and update of student grades.

FR-15: Manage Courses (Subjects)

➔ The system shall allow the Academic Affairs Officer to manage the course catalog.
➔ Course management shall include creating, updating, and removing courses with information such as Course Code, Course Name, Credits, Description, Prerequisites, and Course Type (Core or Elective).
➔ Any updates shall be validated and applied to related classes, schedules, and student records.

FR-16: Manage Classes

➔ The system shall allow the Academic Affairs Officer to create, update, and delete classes for specific courses in each semester.
➔ Class details shall include Class Code, Maximum Student Capacity, Schedule (Day, Time, Room).

➔ The system shall enforce class capacity limits during student course registration.

FR-17: Assign Lecturer to Class

➔ The system shall allow the Academic Affairs Officer to assign or change a lecturer for a specific class.
➔ The system shall display a list of available lecturers within the department.
➔ Before assignment, the system shall check for schedule conflicts.
➔ After assignment, the system shall notify the lecturer by app.

FR-18: Manage Students

➔ The system shall allow the Academic Affairs Officer to manage student records.
➔ Functions shall include:
    - Creating new student records (including bulk import from CSV files)
    - Updating student status (Active, On Leave, Dropped Out, Graduated)
    - Deleting or archiving inactive student records
➔ The system shall automatically generate a unique Student ID for each new student.
➔ Required information shall include Full Name, Date of Birth, and Department, and the system shall also store student academic results.

FR-19: Manage Lecturers

➔ The system shall allow the Academic Affairs Officer to manage lecturer information.
➔ Lecturer management shall include creating new profiles, updating information (such as Contact Details and Qualifications), assigning departments, and deactivating accounts.
➔ The system shall support bulk operations.
➔ The system shall prevent deletion of lecturers who are currently assigned to active classes.

FR-20: Manage Announcements

➔ The system shall allow the Academic Affairs Officer to create, edit, and delete system announcements.
➔ The system shall send announcements to all students.
➔ Announcements shall be displayed on user dashboards

2. Non-Functional Requirements

2.1 Usability

NFR-01: The system shall provide a simple and clear user interface so that Students, Lecturers, and Administrators can use the system without special training.
NFR-02: Common academic tasks, such as viewing grades or registering courses, shall be completed within **no more than three user actions**.
NFR-03: The system shall use a consistent layout, color scheme, and navigation style across all application screens.

2.2 Performance

NFR-04: The system shall respond to normal user actions within **2 seconds** under standard operating conditions.
NFR-05: The system shall support at least **100 concurrent users** accessing the database without major performance slowdown.
NFR-06: Data retrieval operations, such as loading student records or course lists, shall be processed efficiently.

2.3 Security

NFR-07: The system shall require users to log in before accessing any academic functions.
NFR-08: The system shall restrict system functions based on user roles (Student, Lecturer, Administrator).
NFR-09: User passwords shall be stored securely and shall not be saved in plain text.
NFR-10: Only authorized users shall be allowed to create, update, or delete academic data.

2.4 Reliability

NFR-11: The system shall operate stably during normal use without unexpected crashes.
NFR-12: The system shall ensure the correctness and consistency of academic data, such as grades and enrollments.
NFR-13: The system shall record system errors to support debugging and system maintenance.

2.5 Maintainability

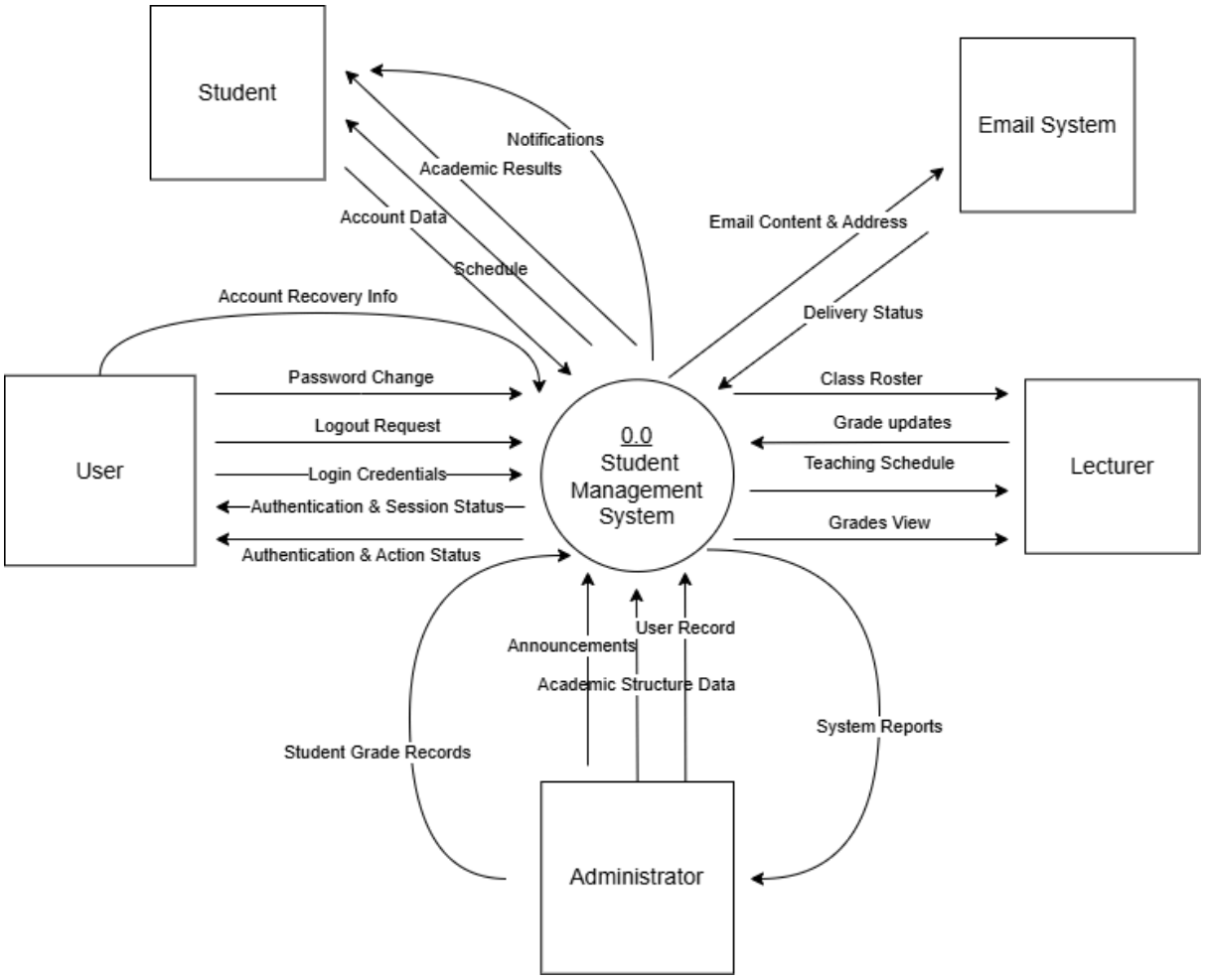
NFR-14: The system shall be designed in a modular way so that individual components can be updated or replaced easily.
NFR-15: The system source code shall follow consistent coding standards to improve readability and maintenance.
NFR-16: The system shall allow future extensions, such as adding new academic management features, with minimal changes.

2.6 Technical & Design Constraints

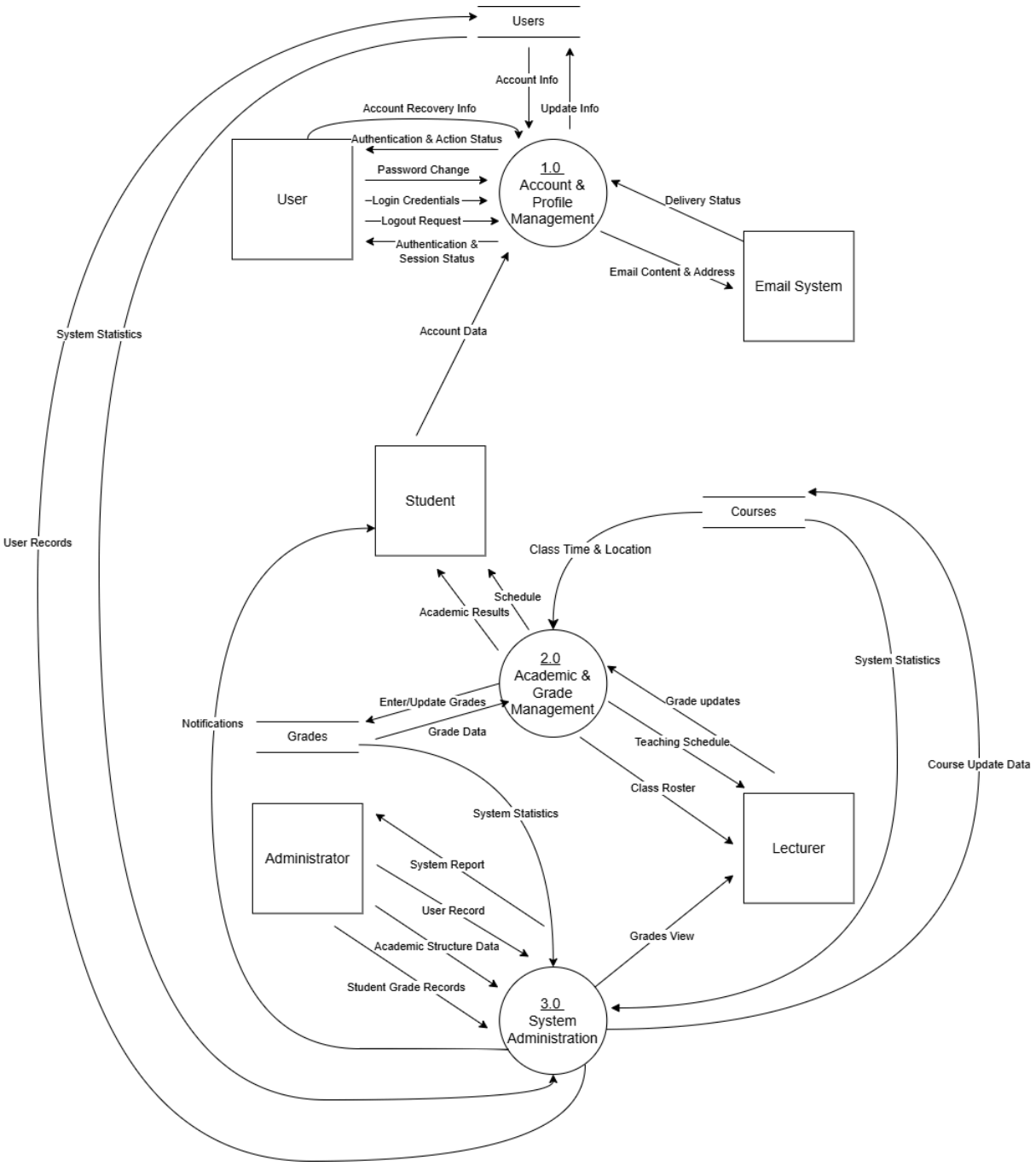
NFR-17: The system shall be developed as a **desktop application** using **Python**.
NFR-18: The system shall use **MySQL** as the database management system for storing academic and user data.
NFR-19: The system shall follow **Object-Oriented Programming (OOP)** principles to support modularity and maintainability.

3. Data Flow Diagram

3.1 DFD Level 0

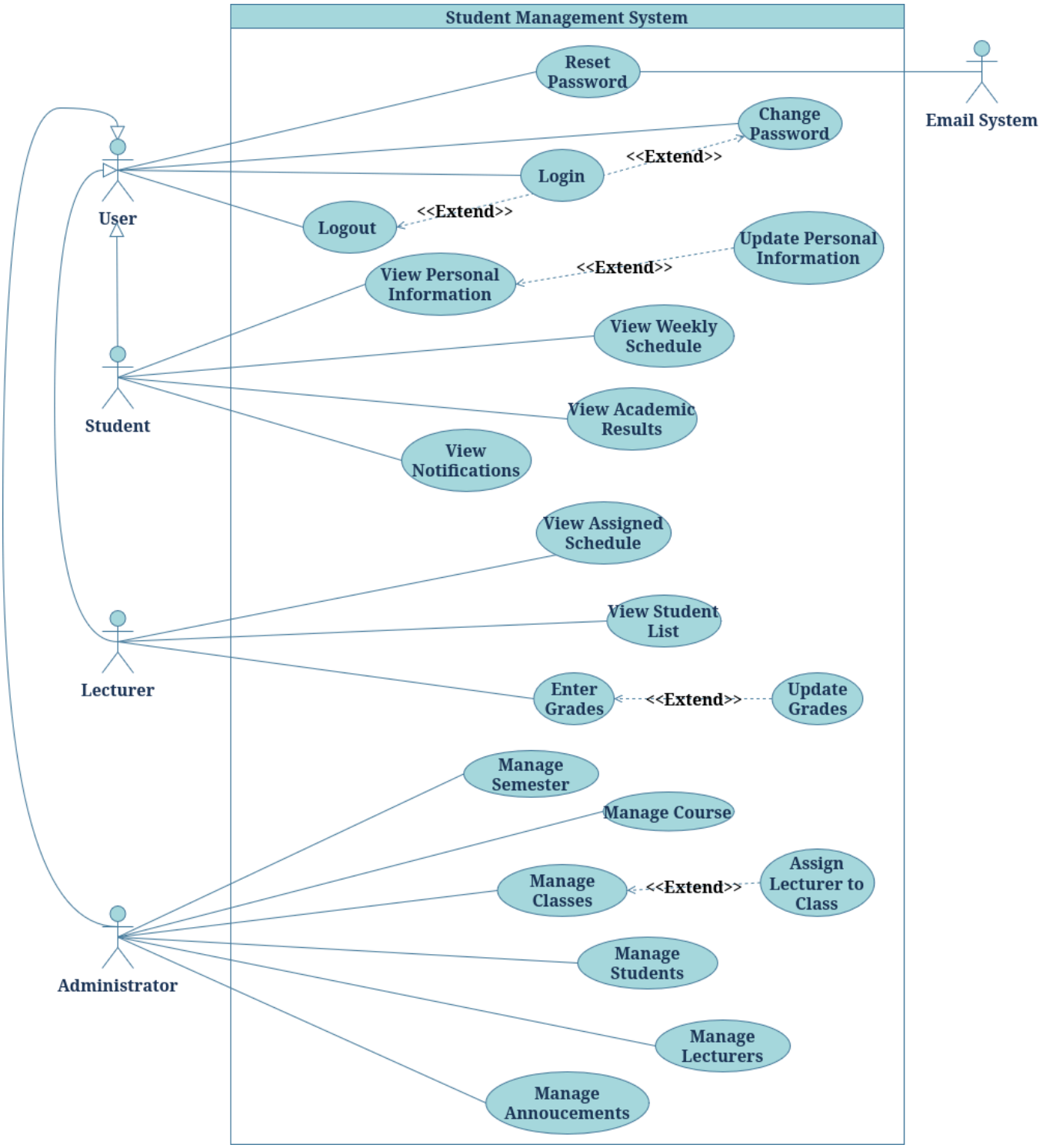


3.2 DFD Level 1



4. Use Case Diagram

4.1 Overview Diagram



4.2 List of Actors

STT	Actor	Description
1	User	A person who accesses the system without logging in. This actor represents general users who can view publicly available information provided by the system.
2	Student	A registered student of the university who interacts with the system after logging in to access academic-related information and services.
3	Lecturer	A university lecturer who interacts with the system in relation to teaching and academic management activities.
4	Administrator	A system administrator who represents the academic affairs office and is responsible for managing and maintaining the system at an overall level.
5	Email System	An external system that communicates with the Student Management System to support email-related communication such as notifications and account-related messages.

4.3 List of Use Cases

Use Case ID	Use Case Name	Primary Actor
UC01	Login	User
UC02	Logout	User
UC03	Change Password	User
UC04	Reset Password	User, Email System
UC05	View Personal Information	Student
UC06	Update Personal Information	Student
UC07	View Weekly Schedule	Student

UC08	View Academic Results	Student
UC09	View Notifications	Student
UC10	View Assigned Schedule	Lecturer
UC11	View Student List	Lecturer
UC12	Enter Grades	Lecturer
UC13	Update Grades	Lecturer
UC14	Manage Semesters	Administrator
UC15	Manage Students	Administrator
UC16	Manage Lecturers	Administrator
UC17	Manage Courses	Administrator
UC18	Manage Classes	Administrator
UC19	Assign Lecturer To Course/Class	Administrator
UC20	Manage Announcements	Administrator

4.4 Use Case Description

4.4.1 UC01 - Login

Description: This use case enables users to securely access the system by entering valid credentials.
After successful authentication, the system creates a session and redirects the user to the appropriate dashboard.
Priority: High
Relationships: Extends: UC02 - Logout, UC03 - Change Password

Basic Flow:
Step 1: The User accesses the login interface of the Student Management System.
Step 2: The User enters their registered **Email/User ID** and **Password**, then clicks the **"Login"** button.
Step 3: The system validates the entered credentials against the user data stored in the database.
Step 4: The system confirms that the account is valid and currently active.
Step 5: The system establishes a secure user session.
Step 6: The system redirects the User to the specific Dashboard corresponding to their assigned role (Student, Lecturer, or Administrator)..

Alternative Flows:

AF1 – Invalid Login Information

- **Condition:** The User provides an incorrect Email/User ID or Password.
- **AF1.1:** At Step 3, the system determines that the credentials do not match any existing records.
- **AF1.2:** The system displays the error message: **"Invalid username or password"**.
- **AF1.3:** The system directs the User back to Step 2 to re-enter their information.

AF2 – Temporary Account Lockout

- **Condition:** The User exceeds 5 failed login attempts.
- **AF2.1:** At Step 3, the system records the 5th failed attempt.
- **AF2.2:** The system updates the account status to **"Locked"** in the database.
- **AF2.3:** The system displays: **"Account locked due to multiple failed attempts. Please try again after 30 minutes."**

AF3 – Password Recovery Request

- **Condition:** The User chooses to reset their password instead of logging in.
- **AF3.1:** At Step 2, the User selects the **"Forgot Password"** option.
- **AF3.2:** The system initiates the password recovery process via UC04.

Pre-conditions: User has a valid registered account; system is available.
Post-conditions: User is authenticated and logged into the system; user session is created.

4.4.2 UC02 - Logout

Description: This use case allows users to safely exit the system.
The system terminates the active session and returns the user to the login page.
Priority: High
Relationships: Extended by: UC01 - Login

Basic Flow:
Step 1: The User selects the **"Logout"** option from the main navigation menu.
Step 2: The system receives the logout request and identifies the active session.
Step 3: The system terminates the current session and clears all temporary session data.
Step 4: The system updates the User's status in the database to record the session closure.
Step 5: The system redirects the User back to the Login Page.

Alternative Flows:

AF1 – Session Already Expired

- **Condition:** The user's session has timed out before they click the logout button.
- **AF1.1:** At Step 1, the system detects that the session is no longer valid.
- **AF1.2:** The system bypasses the manual termination steps.
- **AF1.3:** The system displays the notification: **"Session expired. Please log in again"**.
- **AF1.4:** The system automatically redirects the User to the Login Page.

Pre-conditions: User is logged into the system.
Post-conditions: User session is terminated; user is redirected to the login page.

4.4.3 UC03 - Change Password

Description: This use case allows authenticated users to update their password. The system validates the new password against security rules and confirms the change.
Priority: Medium
Relationships: Extended by: UC01 - Login

Basic Flow:
Step 1: The User navigates to the "Change Password" section within their personal profile.
Step 2: The User enters their current password, a new password, and confirms the new password.
Step 3: The system verifies the current password by cross-matching it with the user data stored in the database.
Step 4: The system validates the new password to ensure it meets the security rules (minimum 8 characters, including uppercase, lowercase, numbers, and special characters).
Step 5: The system updates the password in the database.
Step 6: The system displays a confirmation message: "Your password has been changed successfully".
Step 7: The system records the action in the audit logs for security purposes.

Alternative Flows:

AF1 – Incorrect Current Password

- **Condition:** The User enters a current password that does not match the system records.
- **AF1.1:** At Step 3, the system detects that the provided password is incorrect.
- **AF1.2:** The system displays an error message: "The current password you entered is incorrect. Please try again".
- **AF1.3:** The User is directed back to Step 2 to re-enter the information.

AF2 – New Password Validation Failure

- **Condition:** The new password is too weak or does not match the confirmation entry.
- **AF2.1:** At Step 4, the system identifies that the password does not meet complexity standards or the confirmation fails.
- **AF2.2:** The system displays the message: "Password too weak or confirmation does not match. Please follow the security rules".

- **AF2.3:** The User returns to Step 2 to provide a valid password.

Pre-conditions: User is logged into the system; user knows the current password.
Post-conditions: User password is updated successfully.

4.4.4 UC04 - Reset Password

Description: This use case supports users in recovering access to their accounts when they forget their password.
The system sends a time-limited recovery link or code via the Email System for password reset.
Priority: High
Relationships: None

Basic Flow:
Step 1: The User selects the "Forgot Password" option on the login page.
Step 2: The User enters their registered Email or User ID.
Step 3: The system verifies that the entered information exists in the database.
Step 4: The system generates a password recovery link or code with a 15-minute expiration limit.
Step 5: The system sends the recovery link/code to the User's registered email via the external Email System.
Step 6: The User accesses the link and enters a new password following the security rules.
Step 7: The system updates the password, saves the changes, and confirms the successful reset.

Alternative Flows:

AF1 – User Information Not Found

- **Condition:** The provided Email or User ID does not exist in the system.
- **AF1.1:** At Step 3, the system fails to find the account details.
- **AF1.2:** The system displays the message: "User ID or Email not found. Please check your information."
- **AF1.3:** The User is directed back to Step 2 to re-enter the data.

AF2 – Recovery Link Expired

- **Condition:** The User attempts to use the recovery link after the 15-minute window has passed.

- **AF2.1:** The system detects that the link is no longer valid.
- **AF2.2:** The system displays the message: "The recovery link has expired. Please request a new one."
- **AF2.3:** The User is redirected to Step 1 to restart the process.

Pre-conditions: User has a registered email address, user account exists.
Post-conditions: Password is reset, user can log in with the new password.

4.4.5 UC05 - View Personal Information

Description: This use case allows students to view their personal and enrollment information.
Sensitive academic data is presented in read-only mode to ensure data integrity.
Priority: Medium
Relationships: Extended by: UC06 - Update Personal Information

Basic Flow:
Step 1: The Student logs into the system.
Step 2: The Student navigates to the **"Personal Profile"** section from the dashboard.
Step 3: The system retrieves the student's data from the database.
Step 4: The system displays the following information: Student ID, Full Name, Date of Birth, Class, Department, Contact Information (Phone Number, Email, Address), and Enrollment Status.
Step 5: The system ensures that core academic details (Student ID, Name, etc.) are displayed in **read-only** mode

Alternative Flows:

AF1 – Data Retrieval Error

- **Condition:** The system fails to load the profile data due to a database connection issue.
- **AF1.1:** At Step 3, the system detects a failure in retrieving the account information.
- **AF1.2:** The system displays the message: **"Error: Unable to load profile. Please try again later."**
- **AF1.3:** The Student is directed back to the Dashboard.

Pre-conditions: Student is logged into the system.
Post-conditions: Student personal information is displayed.

4.4.6 UC06 - Update Personal Information

Description: This use case enables students to update permitted personal details such as contact information.
All changes are validated, saved, and recorded for auditing purposes.
Priority: Medium
Relationships: Extends: View Personal Information

Basic Flow:
Step 1: The Student views their profile (UC05) and selects the **"Update"** or **"Edit"** option.
Step 2: The Student enters new contact information, such as **Phone Number, Address, or Email**.
Step 3: The system validates the format of the updated data (e.g., correct email syntax and phone number length).
Step 4: The system saves the updated information to the database.
Step 5: The system records the changes in the **audit logs** for security tracking.
Step 6: The system displays a confirmation message: **"Profile updated successfully."**.

Alternative Flows:

AF1 – Invalid Data Format

- **Condition:** The Student enters an invalid email format or an incorrect phone number length.
- **AF1.1:** At Step 3, the system identifies the formatting error.
- **AF1.2:** The system highlights the invalid fields.
- **AF1.3:** The system displays the message: **"Update failed: Please enter a valid email and phone number."**
- **AF1.4:** The Student returns to Step 2 to correct the information.

AF2 – Unauthorized Field Modification Attempt

- **Condition:** The Student attempts to modify restricted academic fields (e.g., Student ID or Department).
- **AF2.1:** At Step 2, the system prevents any input into **read-only** fields.
- **AF2.2:** If an external script attempt is made, the system rejects the request during validation at Step 3.
- **AF2.3:** The system maintains the original data and informs the user that core information cannot be changed.

4.4.7 UC07 - View Weekly Schedule

Description: This use case enables students to view their weekly timetable including course names, rooms, and lecturers
Priority: Medium
Relationships: None

Basic Flow:
Step 1: The Student logs into the system and accesses the **"Weekly Schedule"** module.
Step 2: The system identifies the current date and retrieves the list of courses the student has registered for.
Step 3: The system generates a visual timetable based on the registration data.
Step 4: The system displays the schedule in a weekly grid format.
Step 5: The system presents specific details for each slot, including **Day, Time, Course Name, Room, and Lecturer**

Alternative Flows:

AF1 – No Registered Courses Found

- **Condition:** The Student has not yet registered for any courses for the current semester.
- **AF1.1:** At Step 2, the system detects an empty course registration list.
- **AF1.2:** The system stops the timetable generation process.
- **AF1.3:** The system displays the message: "No registered courses found for this week. Please check your enrollment status."

AF2 – Data Loading Failure

- **Condition:** The system cannot retrieve schedule data due to a database connection error.
- **AF2.1:** The system detects a retrieval error at Step 2.
- **AF2.2:** The system displays an error message: "Unable to load schedule. Please try again later."

Pre-conditions: Student is logged into the system.
Post-conditions: Weekly schedule is displayed.

4.4.8 UC08 - View Academic Results

Description: This use case allows students to view their complete academic records, including all courses taken and the corresponding grades.
Priority: High
Relationships: None

Basic Flow:
Step 1: The Student navigates to the **"Academic Results"** section from the dashboard.
Step 2: The system retrieves all completed course grades and their corresponding credit weights from the database.
Step 3: The system calculates the student's **cumulative GPA**.
Step 4: The system classifies the student's academic performance (e.g., **Excellent, Good**) based on established GPA rules.
Step 5: The system displays a summary academic transcript organized by individual semesters

Alternative Flows:

AF1 – Results Not Yet Finalized

- **Condition:** Academic results for the current semester have not been officially released or finalized by the lecturer.
- **AF1.1:** At Step 2, the system identifies that current semester grades are still in a "pending" or "unfinalized" status.
- **AF1.2:** The system displays the available previous results and adds a notice: "Current semester results are not yet finalized."

AF2 – Calculation Error

- **Condition:** Missing credit weights prevent the system from accurately calculating the GPA.
- **AF2.1:** At Step 3, the system detects missing data for specific courses.
- **AF2.2:** The system displays a warning: "GPA calculation is incomplete due to missing course data. Please contact the Training Department."

Pre-conditions: Academic results have been released; student is logged in.
Post-conditions: Academic results are displayed.

4.4.9 UC09 - View Notifications

Description: This use case enables students to receive and review system notifications and announcements.
Each notification includes relevant metadata such as time and read status.
Priority: Low
Relationships: None

Basic Flow:
Step 1: The Student logs into the system and navigates to the **Notification** section.
Step 2: The system retrieves all relevant notifications, including general announcements and system alerts from the Administrator.
Step 3: The system displays a list of notifications with metadata such as **timestamp** and **category** (e.g., Urgent or Informational).
Step 4: The Student selects a specific notification to read the full content.

Alternative Flows:

AF1 – No Notifications Available

- **Condition:** There are no notifications or announcements directed to the Student.
- **AF1.1:** At Step 2, the system finds no records in the notification database.
- **AF1.2:** The system displays the message: **"You have no new notifications at this time."**

AF2 – System Message Retrieval Failure

- **Condition:** The system cannot load notifications due to a server connection error.
- **AF2.1:** At Step 2, the system fails to communicate with the database.
- **AF2.2:** The system displays the message: **"Failed to load notifications. Please refresh the page."**

Pre-conditions: Student is logged into the system.
Post-conditions: Notifications are displayed; notifications are marked as read (if applicable).

4.4.10 UC10 - View Assigned Schedule

Description: This use case allows lecturers to view their teaching schedule for the semester.
The schedule includes assigned classes, time slots, and enrollment details.
Priority: Medium
Relationships: None

Basic Flow:
Step 1: The Lecturer logs into the system and accesses the **"Teaching Schedule"** module.
Step 2: The system retrieves the schedule for the current semester based on the Lecturer's ID.
Step 3: The system displays the teaching timetable, including **Class Name, Course Code, Room, Teaching Days, and Time Slots**.
Step 4: The Lecturer uses filtering options to view the schedule by **week** or **month**.
Step 5: The system updates the view to show the number of enrolled students for each class

Alternative Flows:

AF1 – No Classes Assigned

- **Condition:** The Lecturer is not assigned to any active classes for the current semester.
- **AF1.1:** At Step 2, the system detects an empty assignment record.
- **AF1.2:** The system displays the message: **"No assigned classes found for this semester."**

AF2 – Schedule Conflict Warning

- **Condition:** The system detects overlapping time slots in the assigned schedule.
- **AF2.1:** At Step 3, the system highlights the conflicting time slots in red.
- **AF2.2:** The system prompts the Lecturer to contact the Academic Affairs Office for resolution.

Pre-conditions: Student is registered for courses.
Post-conditions: Assigned schedule is displayed.

4.4.11 UC11 - View Student List

Description: This use case enables lecturers to view enrolled students in a selected class.
Student information is presented with support for searching and sorting.
Priority: Medium
Relationships: None

Basic Flow:
Step 1: The Lecturer selects a specific class from their assigned teaching schedule.
Step 2: The system retrieves the list of students enrolled in that class.
Step 3: The system displays the student records, including **Student ID, Full Name, Contact Information, and a summary of attendance/grades**.
Step 4: The Lecturer utilizes the **search** or **sorting** functions to locate specific students.
Step 5: The system updates the displayed list based on the chosen search criteria or sort order.

Alternative Flows:

AF1 – Empty Class Roster

- **Condition:** No students have registered for the selected class yet.
- **AF1.1:** At Step 2, the system identifies that the enrollment count is zero.
- **AF1.2:** The system displays the message: **"This class currently has no enrolled students."**

AF2 – Search Result Not Found

- **Condition:** The Lecturer enters search criteria that do not match any student in the list.
- **AF2.1:** At Step 4, the system filters the list but finds no matches.
- **AF2.2:** The system displays the message: **"No students found matching your search criteria."**

Pre-conditions: Lecturer is logged into the system; lecturer is assigned to at least one class.
Post-conditions: Student list is displayed.

4.4.12 UC12 - Enter Grades

Description: This use case allows lecturers to enter assessment scores for students. The system calculates weighted averages and finalizes grades upon completion.
Priority: High
Relationships: Extended by: UC13 - Update Grades

Basic Flow
Step 1: The Lecturer selects a specific class from their assigned teaching schedule.
Step 2: The system provides an interface for the Lecturer to input scores for each student.
Step 3: The Lecturer enters grades for various components on a scale of 0–10.
Step 4: The system automatically calculates the weighted averages based on the course's grading rules.
Step 5: The Lecturer finalizes the grades and submits them to the system.
Step 6: The system saves and locks the grades to prevent further unauthorized changes.
Step 7: The system automatically notifies the affected students through the integrated app notification system.

Alternative Flows:

AF1 – Invalid Grade Input

- **Condition:** The Lecturer enters a value outside the 0–10 scale.
- **AF1.1:** At Step 3, the system detects an out-of-range value.
- **AF1.2:** The system highlights the invalid field and prevents submission.
- **AF1.3:** The system displays the message: "Invalid input: Please enter a score between 0 and 10."

Pre-conditions: Lecturer is logged into the system; lecturer is assigned to the course.
Post-conditions: Grades are entered and saved.

4.4.13 UC13 - Update Grades

Description: This use case allows lecturers to modify previously entered grades. The system ensures modifications are only made while the semester is active and the individual grade records are not locked.
Priority: Medium

Relationships: Extends: UC12 - Enter Grade

Basic Flow:
Step 1: The Lecturer accesses the grading interface for a class where grades were previously entered.
Step 2: The system verifies the current date against the Semester End Date and checks if the **Semester Status**.
Step 3: The Lecturer modifies the specific grade components.
Step 4: The system updates the records.
Step 5: The system sends a notification to the affected student.

Alternative Flows:

AF1 – Semester Closed or Date Expired

- **Condition:** The semester status is finished or the current date has passed the "Semester End Date".
- **AF1.1:** At Step 2, the system identifies the semester is no longer active.
- **AF1.2:** The system disables the edit function and displays: **"Grade update period has expired."**.

Pre-conditions: Grades have been entered previously; lecturer is logged in;
the semester has not ended and the grade sheet has not been locked.
Post-conditions: Grades are updated; a notification is sent.

4.4.14 UC14 - Manage Semesters

Description: This use case enables administrators to create and control academic semesters.
Semester status changes automatically affect related system operations.
Priority: High
Relationships: None

Basic Flow:
Step 1: The Administrator (Academic Affairs Officer) accesses the Semester Management module.
Step 2: The Administrator selects an action: Create a new semester, Update an existing one, or Archive a finished semester.
Step 3: The Administrator enters the required details, including start/end dates.
Step 4: The Administrator sets the semester status (e.g., In Progress, Closed, Finished).

Step 5: The system validates the data and automatically applies related rules, such as enabling or disabling the entry and update of student grades.
Step 6: The system saves the changes and confirms the successful update.

Alternative Flows:

AF1 – Date Overlap Conflict

- **Condition:** The Administrator enters dates that overlap with an existing semester.
- **AF1.1:** At Step 5, the system identifies a scheduling conflict.
- **AF1.2:** The system displays the message: "Conflict detected: The selected dates overlap with another semester. Please review the schedule."

Pre-conditions: Administrator is logged into the system.
Post-conditions: Semester information is created, updated, or deleted.

4.4.15 UC15 - Manage Students

Description: This use case allows the Administrator to maintain student records, including creating new profiles, updating status, and performing bulk imports, while ensuring each student has a unique ID.
Priority: High
Relationships: None

Basic Flow:
Step 1: The Administrator accesses the Student Management module.
Step 2: The system displays the list of existing student records.
Step 3: The Administrator selects an action: Create a new record, Update student status (Active, On Leave, Dropped Out, Graduated), or Bulk Import from CSV.
Step 4: The Administrator enters or uploads the required information (Full Name, DoB, Student Status and Department).
Step 5: The system validates the provided data.
Step 6: For new students, the system automatically generates a unique Student ID.
Step 7: The system saves the student profile and academic result history in the database.

Alternative Flows:

AF1 – Bulk Import Failure

- **Condition:** The uploaded CSV file contains missing mandatory fields or invalid data.
- **AF1.1:** At Step 5, the system detects errors in the file structure.
- **AF1.2:** The system stops the import process and displays the message: "Import failed: Please ensure all required fields are correctly formatted in the CSV file."

Pre-conditions: Administrator is logged into the system.
Post-conditions: Student records are created, updated, or removed.

4.4.16 UC16 - Manage Lecturers

Description: This use case allows administrators to manage lecturer profiles, departments, and account status.
Priority: High
Relationships: None

Basic Flow:
Step 1: The Administrator accesses the Lecturer Management module.
Step 2: The system displays the list of existing lecturer profiles.
Step 3: The Administrator selects an action:

- Create a new lecturer profile.
- Update existing information (e.g., Contact Details, Qualifications).
- Assign a lecturer to a specific department.
- Deactivate a lecturer account.

Step 4: The Administrator enters the required data or performs bulk operations.
Step 5: The system validates the provided information.
Step 6: The system saves the changes to the database and confirms successful completion

Alternative Flows:

AF1 – Delete Active Lecturer Restricted

- **Condition:** The Administrator attempts to delete a lecturer who is currently assigned to active classes.
- **AF1.1:** At Step 5, the system identifies that the lecturer has active teaching assignments.
- **AF1.2:** The system blocks the deletion or deactivation process.

- **AF1.3:** The system displays the message: “Action failed: Cannot delete or deactivate a lecturer currently assigned to active classes.”

Pre-conditions: Administrator is logged into the system.
Post-conditions: Lecturer records are managed successfully.

4.4.17 UC17 - Manage Courses

Description: This use case enables administrators to maintain the course catalog, including credits and prerequisites
Priority: High
Relationships: None

Basic Flow:
Step 1: The Administrator accesses the Course Catalog management module.
Step 2: The system displays the list of all available courses.
Step 3: The Administrator selects an action:

- Create a new course.
- Update existing course details (e.g., Credits, Prerequisites).
- Remove a course from the catalog.

Step 4: The Administrator enters the required information: Course Code, Name, Credits, Description, Prerequisites, and Course Type.
Step 5: The system validates the input and checks for existing dependencies.
Step 6: The system applies updates to all related classes, schedules, and student records.
Step 7: The system saves the changes and confirms successful completion.

Alternative Flows:

AF1 – Invalid Course Data

- **Condition:** The Administrator enters invalid information (e.g., negative credit values or a duplicate Course Code).
- **AF1.1:** At Step 5, the system detects the incorrect or conflicting data.
- **AF1.2:** The system highlights the invalid fields and prompts for correction.
- **AF1.3:** The Administrator is directed back to Step 4

Pre-conditions: Administrator is logged into the system.

Post-conditions: Course information is managed successfully.

4.4.18 UC18 - Manage Classes

Description: This use case allows administrators to create and manage specific class sections, schedules, and capacities for each semester
Priority: High
Relationships: None

Basic Flow:
Step 1: The Administrator accesses the Class Management module.
Step 2: The system displays the list of classes filtered by course and semester.
Step 3: The Administrator selects an action:

- Create a new class for a specific course.
- Update class details (e.g., Room, Schedule, Capacity).
- Delete an existing class.

Step 4: The Administrator enters the required details: Class Code, Maximum Student Capacity, and Schedule (Day, Time, Room).
Step 5: The system validates the data and ensures capacity limits are respected.
Step 6: The system saves the changes and confirms successful completion.

Alternative Flows:

AF1 – Capacity Limit Conflict

- **Condition:** The Administrator attempts to set a maximum capacity lower than the number of students already enrolled.
- **AF1.1:** At Step 5, the system detects that the new capacity limit is insufficient for current enrollment.
- **AF1.2:** The system prevents the update and displays the message: “Update failed: Maximum capacity cannot be lower than the current number of enrolled students.”

Pre-conditions: Administrator is logged into the system.
Post-conditions: Class information is managed successfully.

4.4.19 UC19 - Assign Lecturer To Course/Class

Description: This use case enables administrators to assign specific lecturers to classes while checking for schedule conflicts.
Priority: High
Relationships: None

Basic Flow:
Step 1: The Administrator accesses the lecturer assignment interface.
Step 2: The system displays a list of available lecturers within the department.
Step 3: The Administrator selects a specific class and chooses a lecturer from the list to assign or change.
Step 4: The system automatically checks for any schedule conflicts for the selected lecturer.
Step 5: The Administrator confirms the selection to finalize the assignment.
Step 6: The system updates the class records in the database.
Step 7: The system automatically notifies the assigned lecturer through the application

Alternative Flows:

AF1 – Schedule Conflict Detected

- **Condition:** The selected lecturer is already assigned to another class at the same time.
- **AF1.1:** At Step 4, the system detects an overlapping time slot for the lecturer.
- **AF1.2:** The system prevents the assignment process.
- **AF1.3:** The system displays the message:
  "Assignment failed: The lecturer has a schedule conflict during this time slot. Please choose another lecturer or time.".

Pre-conditions: Administrator is logged into the system; Course/Class and Lecturer exist.
Post-conditions: Lecturer is assigned to the selected course/class.

4.4.20 UC20 Manage Announcements

Description: This use case allows the Administrator to create, modify, and remove system announcements, ensuring all students are informed and the announcements are visible on their dashboards.
Priority: Medium

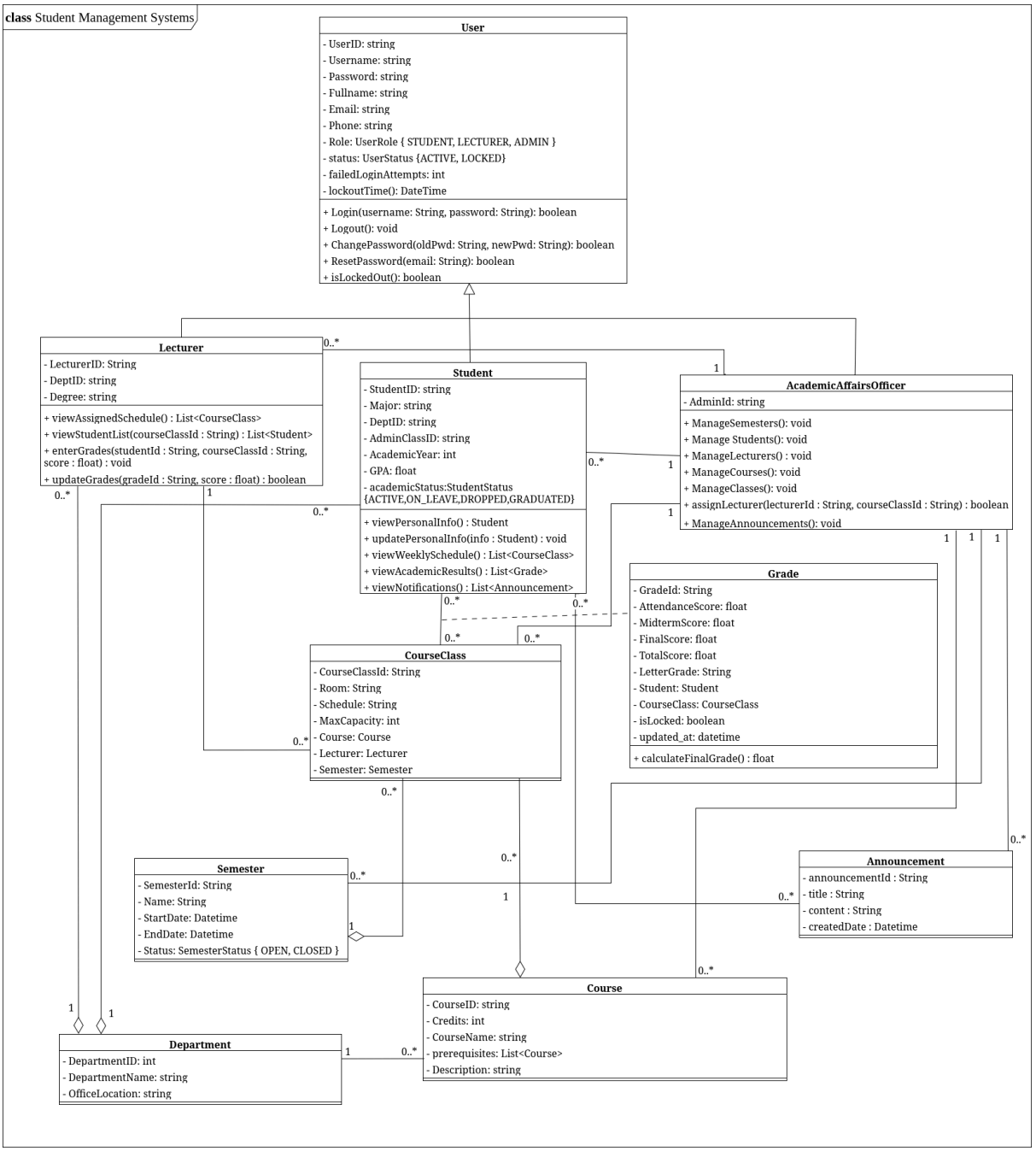
Relationships: None

Basic Flow:
Step 1: The Administrator accesses the Announcement Management module.
Step 2: The Administrator selects an action: **Create**, **Edit**, or **Delete** an announcement.
Step 3: The Administrator enters the announcement content.
Step 4: The system validates the announcement details.
Step 5: The system saves the announcement and ensures it is displayed on all student dashboards.

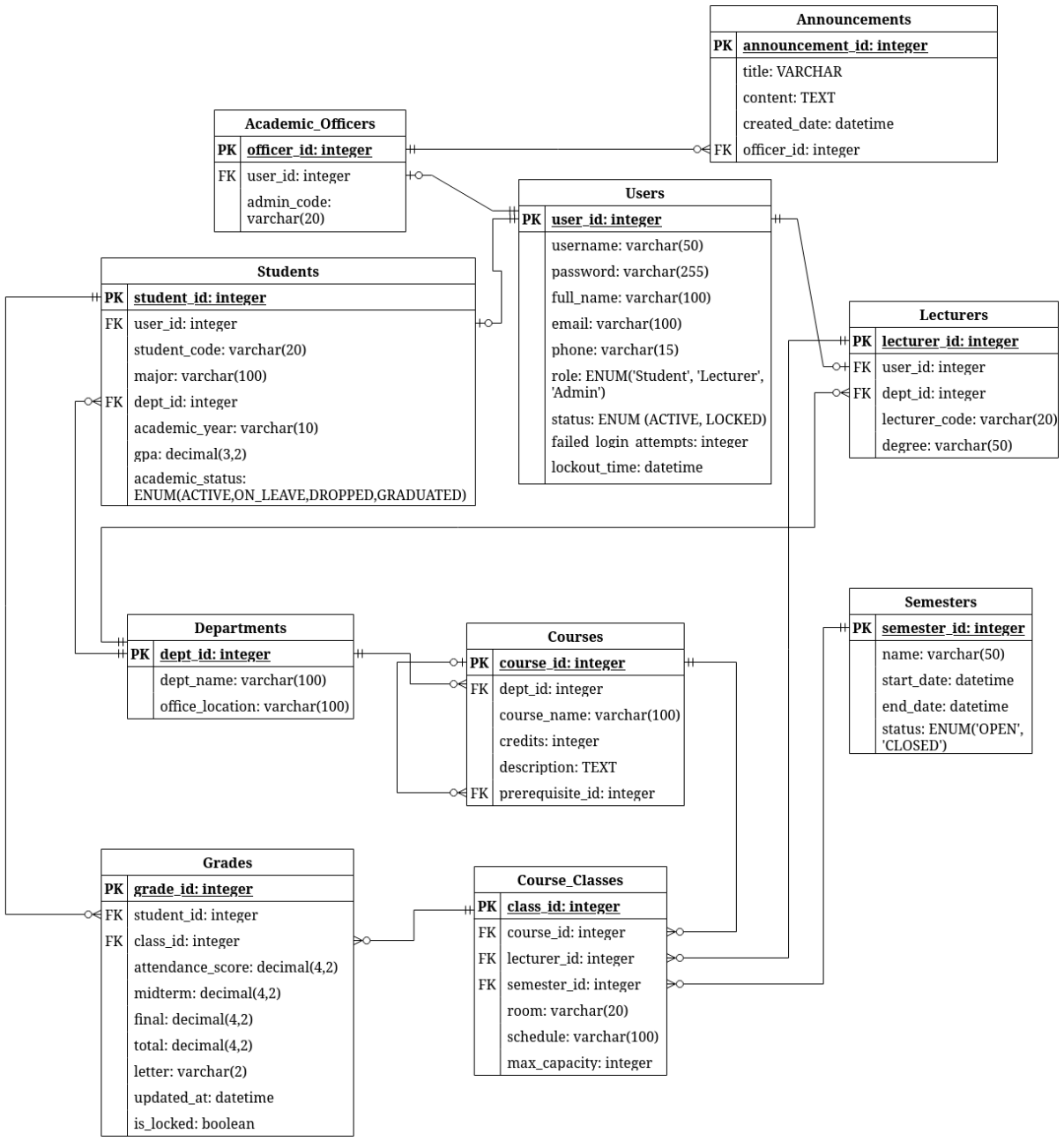
Alternative Flows: None

Pre-conditions: Administrator is logged into the system.
Post-conditions: Announcements are created, updated, or deleted.

5. Class Diagram



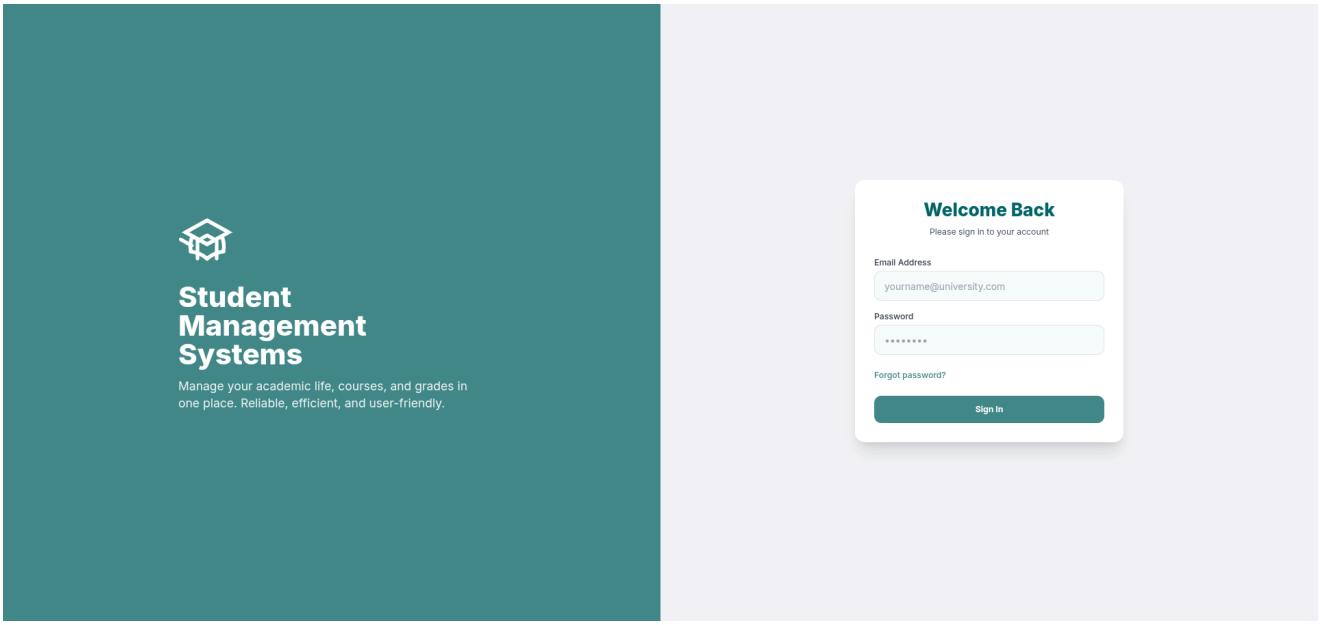
6. Data Model


Announcements
PK announcement_id: integer
title: VARCHAR
content: TEXT
created_date: datetime
FK officer_id: integer
Academic_Officers
PK officer_id: integer
FK user_id: integer
admin_code: varchar(20)
Users
PK user_id: integer
username: varchar(50)
password: varchar(255)
full_name: varchar(100)
email: varchar(100)
phone: varchar(15)
role: ENUM('Student', 'Lecturer', 'Admin')
status: ENUM (ACTIVE, LOCKED)
failed_login_attempts: integer
lockout_time: datetime
Students
PK student_id: integer
FK user_id: integer
student_code: varchar(20)
major: varchar(100)
FK dept_id: integer
academic_year: varchar(10)
gpa: decimal(3,2)
academic_status: ENUM(ACTIVE,ON_LEAVE,DROPPED,GRADUATED)
Lecturers
PK lecturer_id: integer
FK user_id: integer
FK dept_id: integer
lecturer_code: varchar(20)
degree: varchar(50)
Departments
PK dept_id: integer
dept_name: varchar(100)
office_location: varchar(100)
Courses
PK course_id: integer
FK dept_id: integer
course_name: varchar(100)
credits: integer
description: TEXT
FK prerequisite_id: integer
Semesters
PK semester_id: integer
name: varchar(50)
start_date: datetime
end_date: datetime
status: ENUM('OPEN', 'CLOSED')
Grades
PK grade_id: integer
FK student_id: integer
FK class_id: integer
attendance_score: decimal(4,2)
midterm: decimal(4,2)
final: decimal(4,2)
total: decimal(4,2)
letter: varchar(2)
updated_at: datetime
is_locked: boolean
Course_Classes
PK class_id: integer
FK course_id: integer
FK lecturer_id: integer
FK semester_id: integer
room: varchar(20)
schedule: varchar(100)
max_capacity: integer

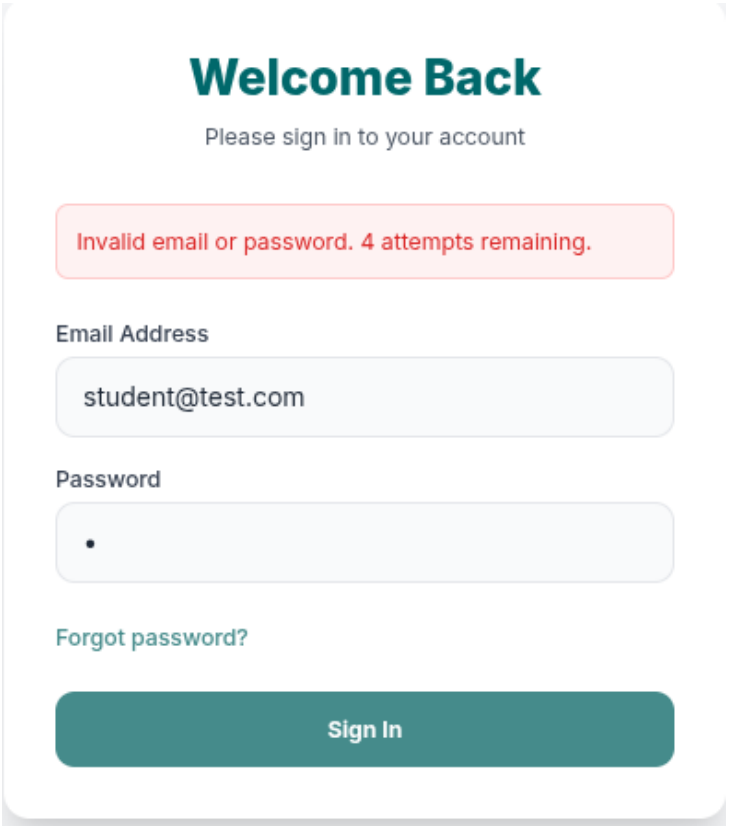
7. Interface Design Description

7.1 Authentication Interfaces

7.1.1 Login Interface

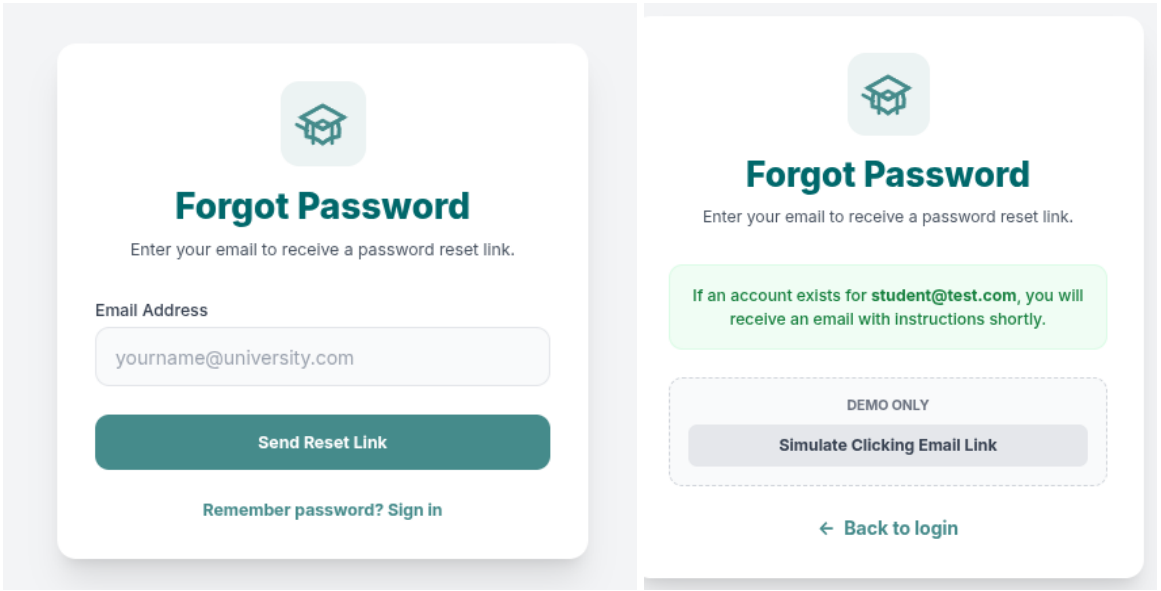


- The Login screen allows users to enter their registered Email/User ID and Password.
- The screen features a Login button and a Forgot Password link.



- Error messages are displayed immediately if the login information is invalid.
- Upon successful login, users are redirected to a dashboard specific to their role.

7.1.2 Forgot Password Interface

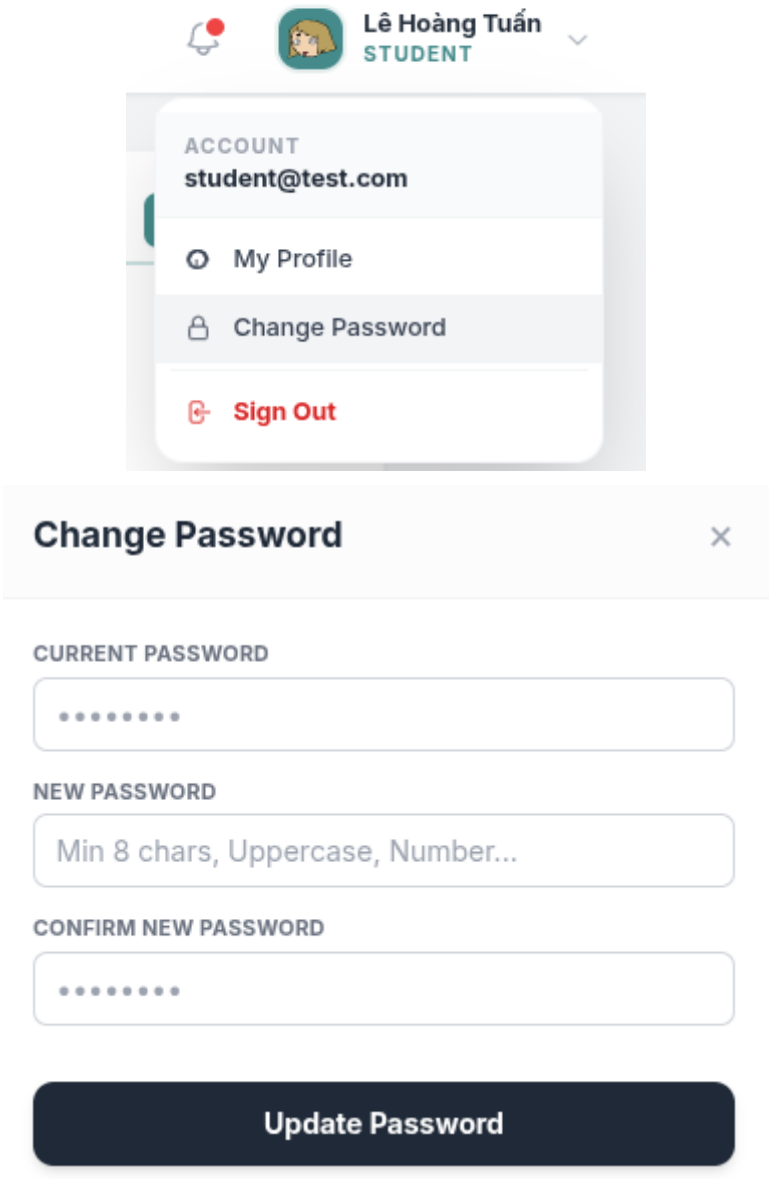


- Users enter their registered Email or User ID to initiate recovery.
- The system provides clear instructions for the password recovery process.

- A confirmation message is displayed once the recovery request is successfully submitted.

7.1.3 Change Password Interface

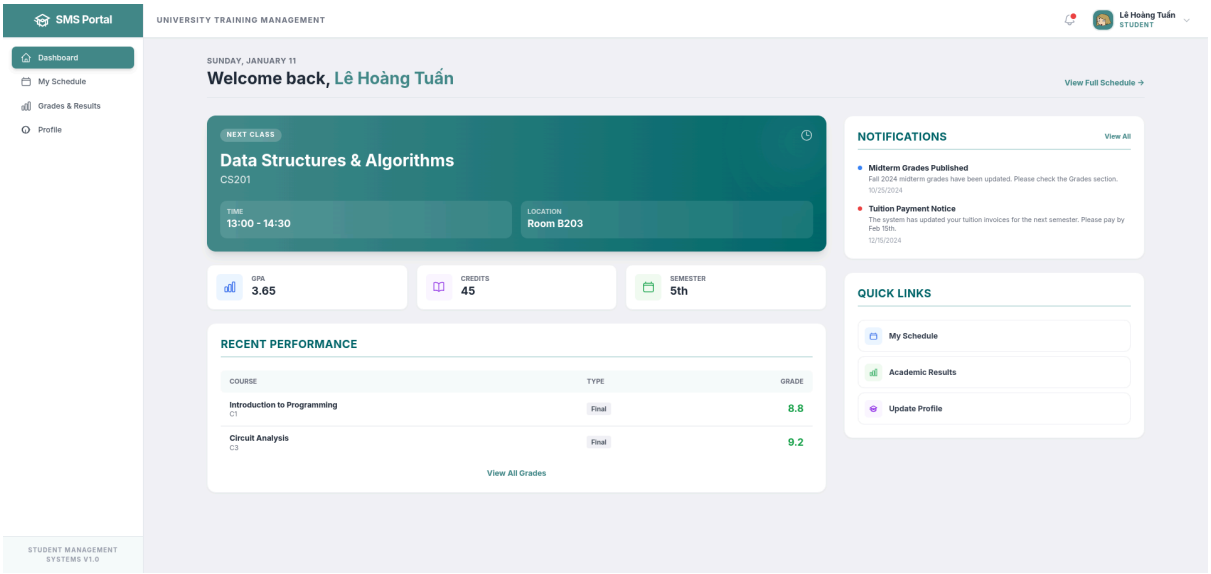
After logging into the system, the user can access the **Change Password** function from the **Profile menu**.



- The interface allows users to enter their current password, a new password, and confirm it.
- The system validates the input and updates the password, then displays a success or error message.

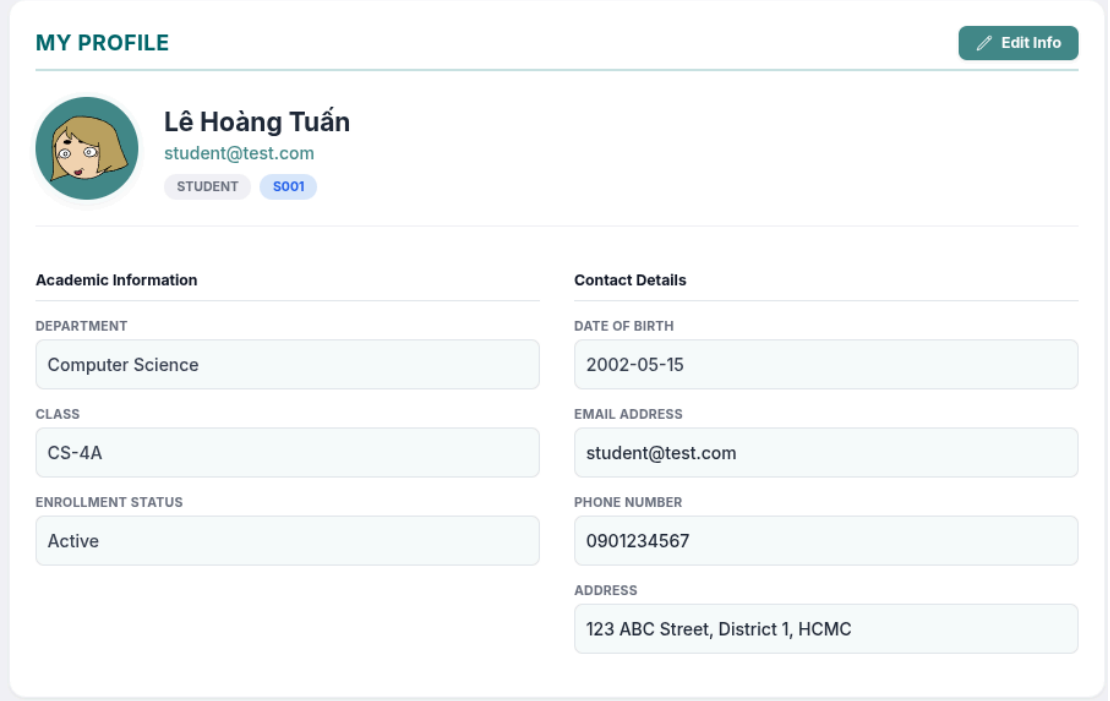
7.2 Student Interfaces

7.2.1 Student Dashboard



- The dashboard serves as the primary screen after a **student** logs in.
- It displays recent notifications, a summary of registered courses, and quick access to main functions.
- Important announcements are highlighted to ensure they are easily viewed by the student.

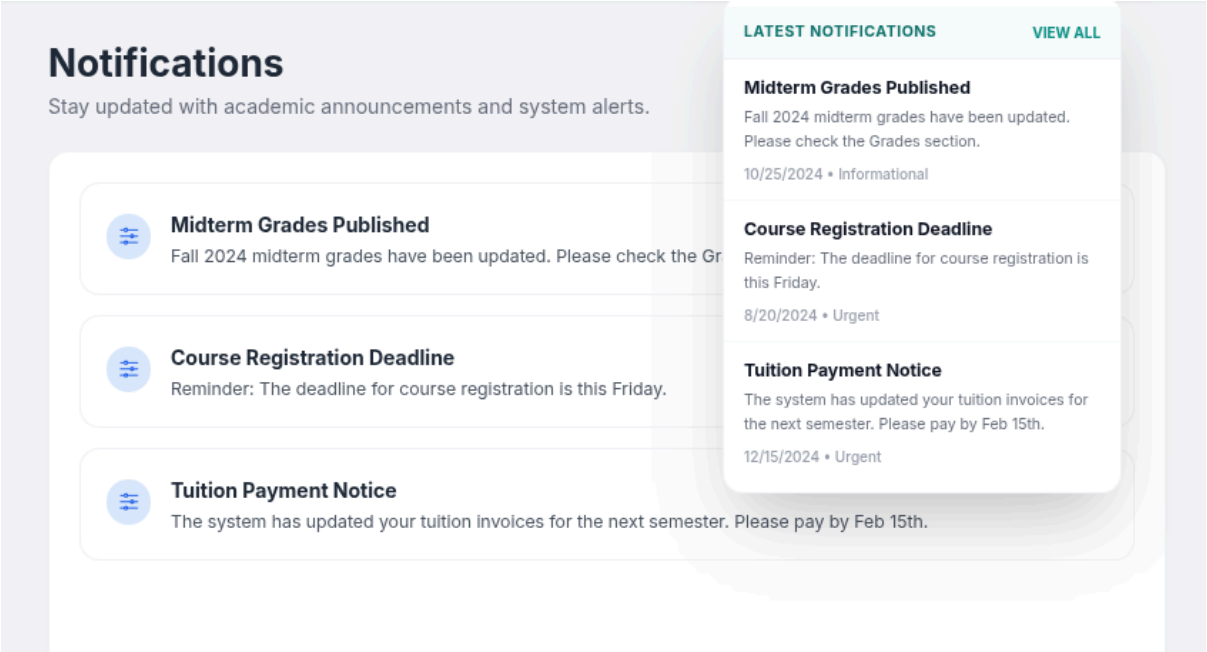
7.2.2 Personal Information Interface



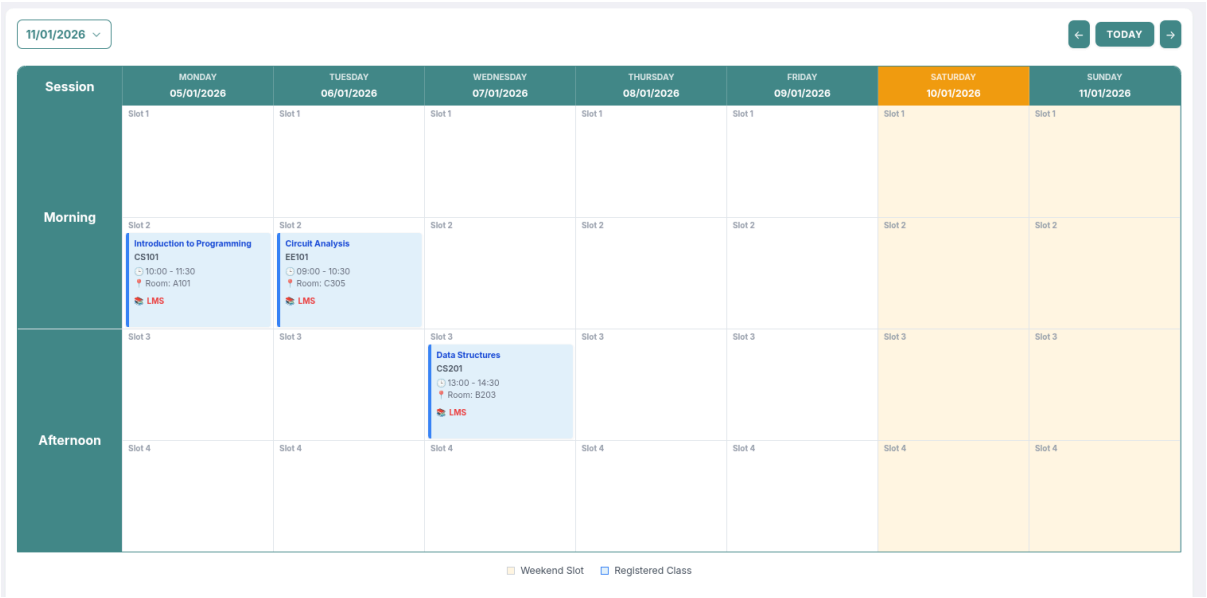
- This interface displays profile details like Date of Birth, Full Name, Class, and Department.
- Users can edit specific fields such as Phone Number, Address, and Email.
- Read-only fields are clearly marked to prevent confusion.
- A confirmation message appears after any successful update to personal information.

7.2.7 View Notifications Interface

Students can access the **View Notifications** interface to view system announcements and academic notifications. Notifications include title, content summary, timestamp.



7.2.8 Weekly Schedule Interface



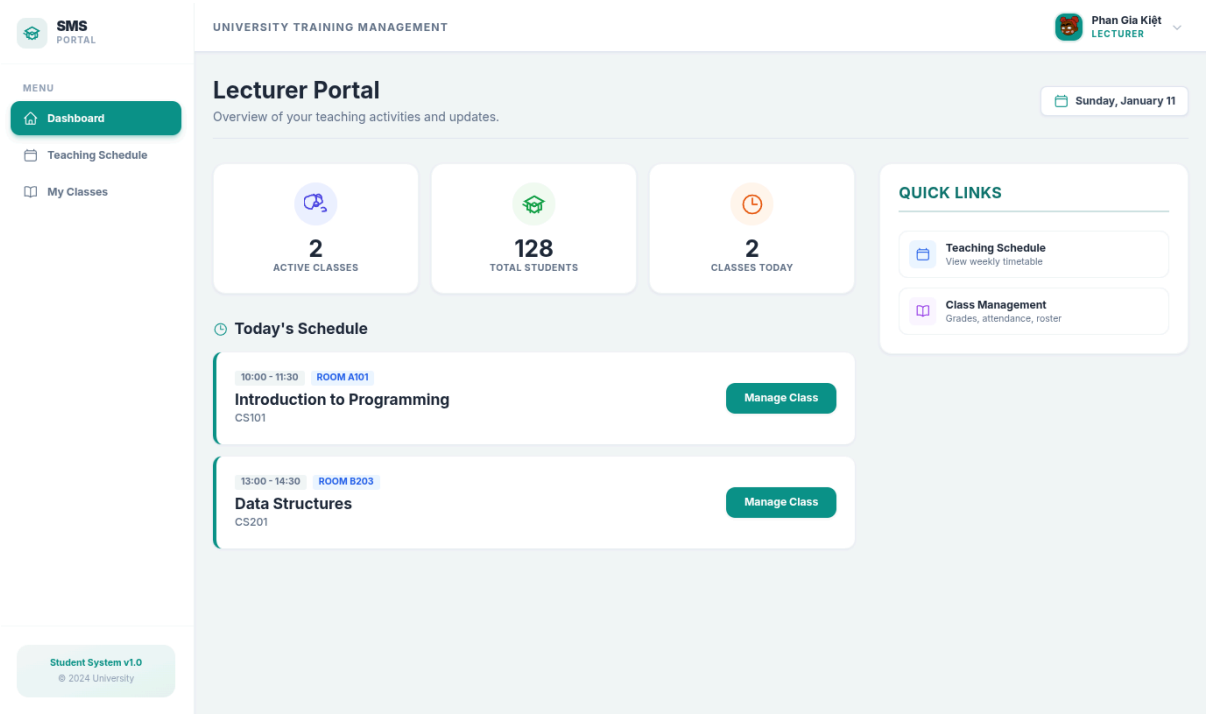
- The student's timetable is displayed in an organized weekly calendar format.
- Each class block contains the course name, time, room number, and lecturer name.
- The schedule is designed to be visually intuitive and easy to read.

7.2.9 Grades and Academic Results Interface

- This interface displays grades organized by course and semester.
- It provides a breakdown of component scores alongside the final grades.
- Academic results include the student's GPA and overall performance classification.
- Students can also view their historical academic results from previous semesters.

7.3 Lecturer Interfaces

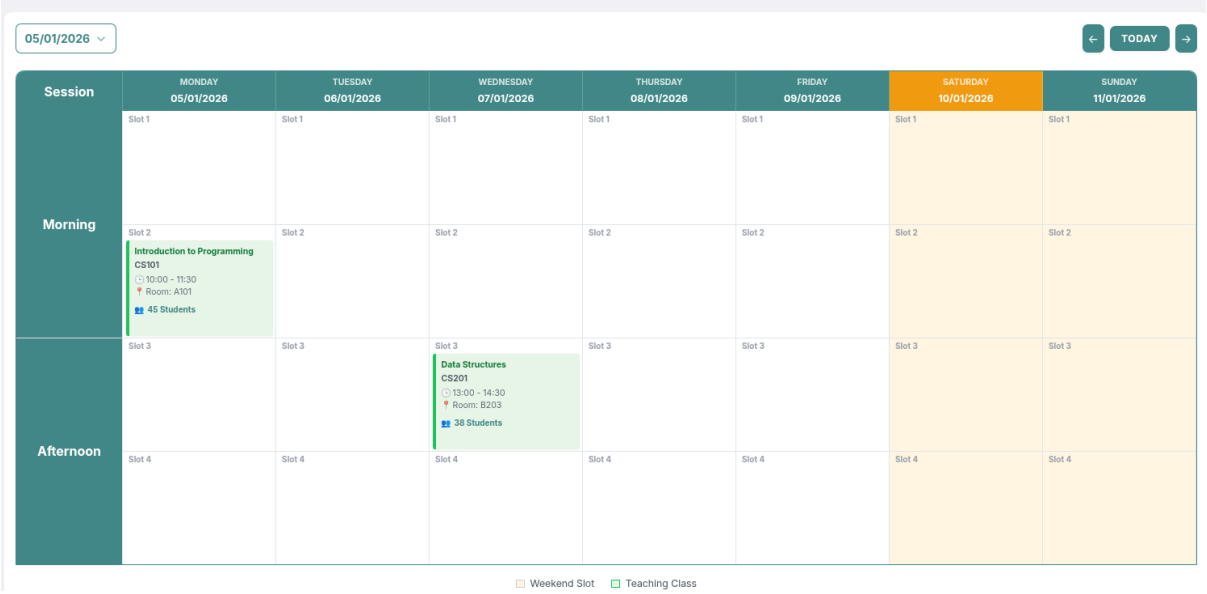
7.3.1 Lecturer Dashboard



- Displays an overview of the lecturer’s teaching activities.

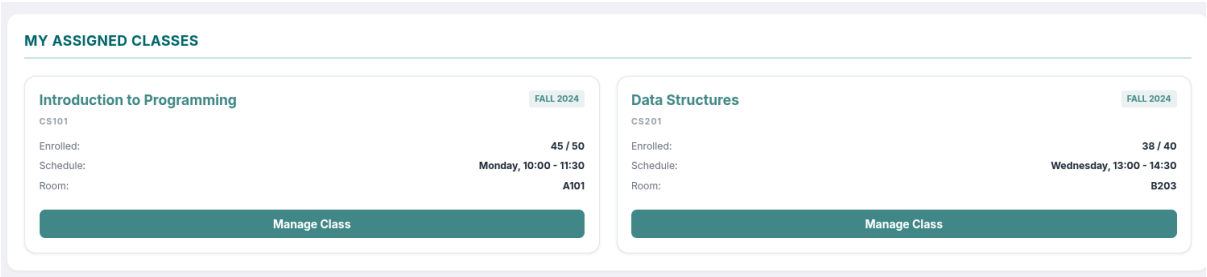
- Provides quick navigation to **Teaching Schedule**, and **Class Management** through the sidebar menu.

7.3.2 Teaching Schedule Interface



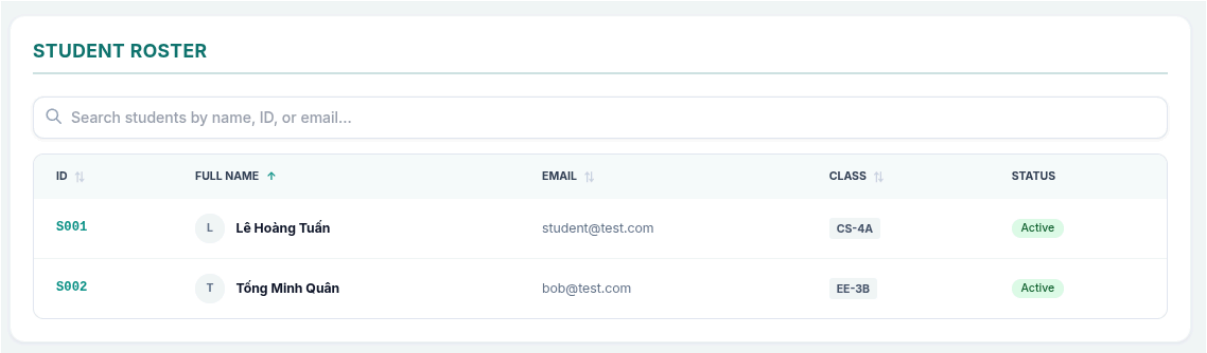
- Displays the lecturer's teaching schedule for the current semester.
- The schedule includes Course Name, Class Code, Room, Teaching Days, and Time Slots.
- Classes are presented in a clear list or timetable-style layout.
- Allows lecturers to select a class directly from the schedule to access **Class Management** functions.

7.3.3 Class Management



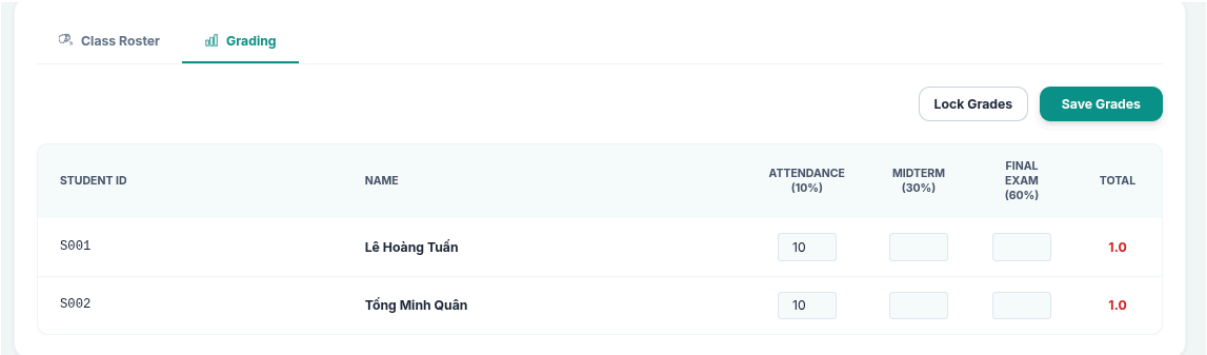
- The Class Management interface allows lecturers to manage all teaching-related activities for each assigned class.
- After selecting a class, the system displays the class title and provides three main functional sections: **Roster**, and **Grading**.

7.3.4 Class Management – Roster Interface



- Displays a list of students enrolled in the selected class.
- Includes Student ID, Full Name, and basic enrollment information.
- Supports search and sorting to quickly locate students.
- Provides navigation back to the class list.

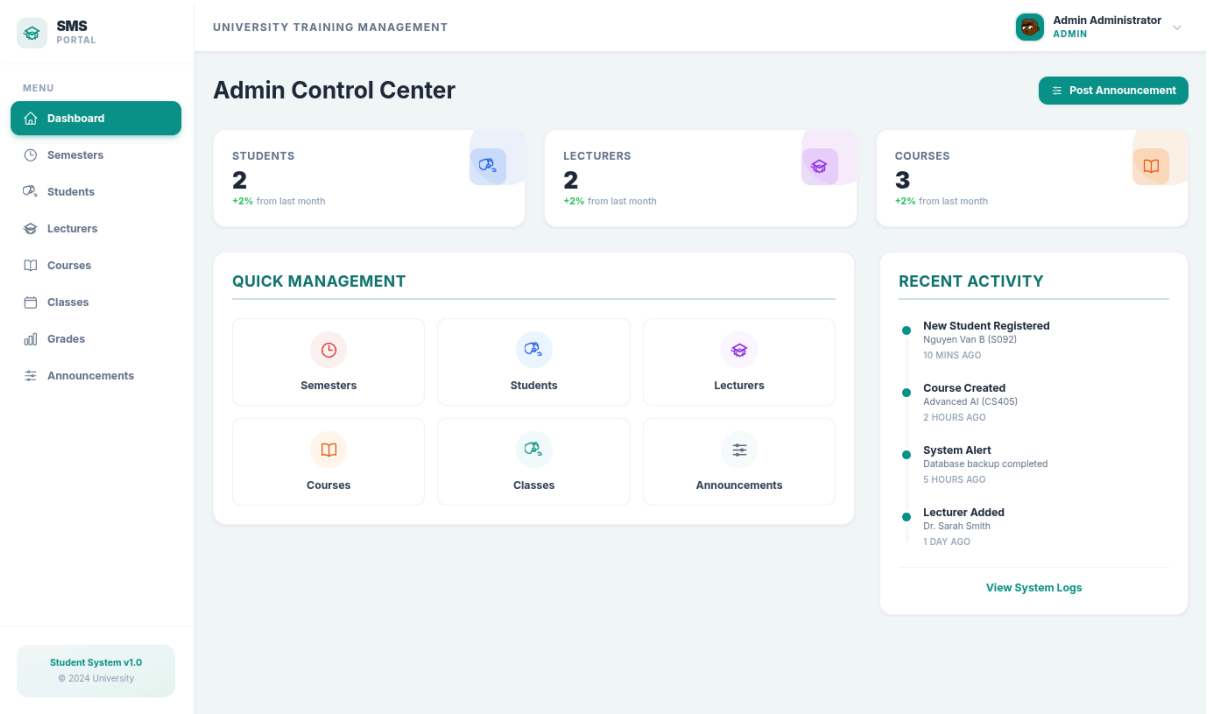
7.3.5 Class Management – Grading Interface



- Allows lecturers to enter and manage grades for students in the selected class.
- Grade components are clearly labeled (Attendance, Midterm, Final Exam).
- The system automatically calculates total scores.
- Displays confirmation messages after grades are saved or updated.

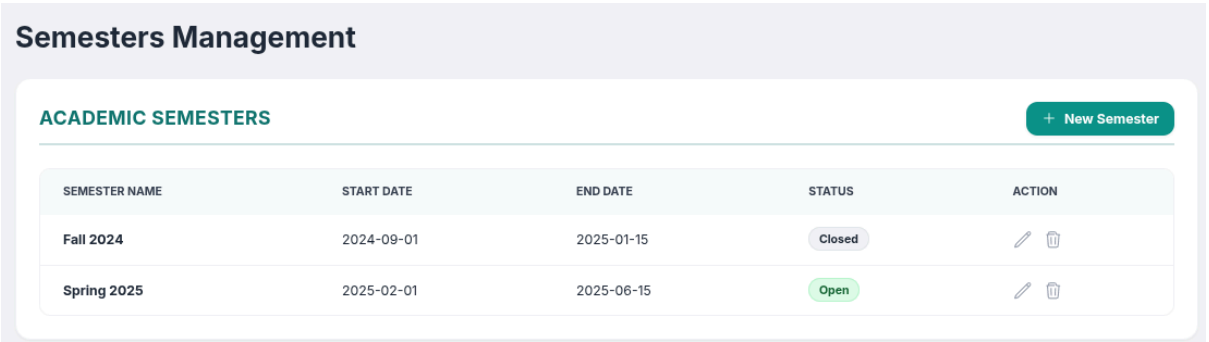
7.4 Administrator Interfaces

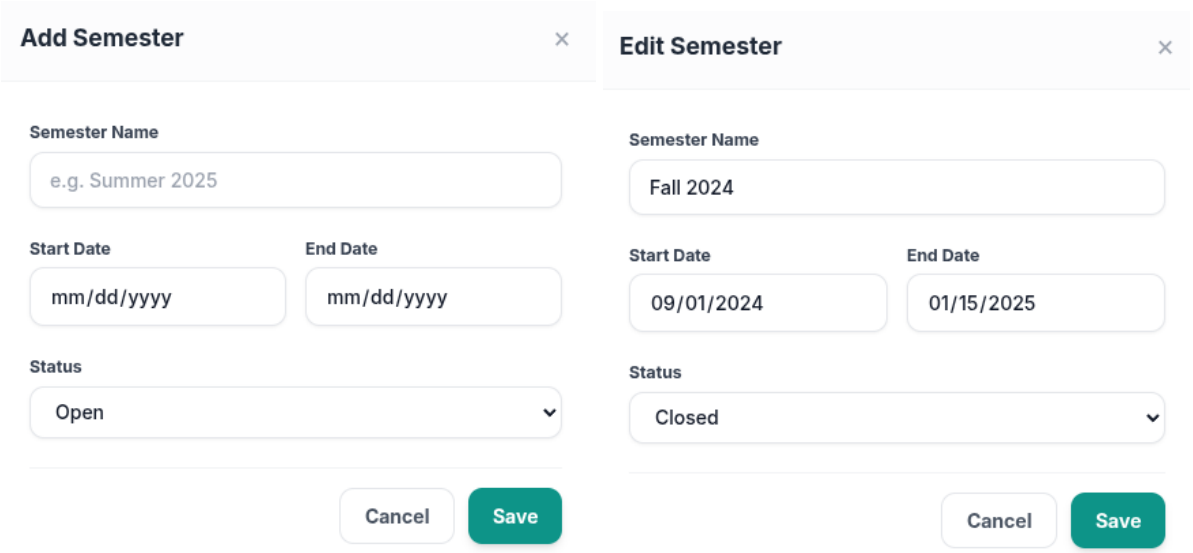
7.4.1 Administrator Dashboard



- The dashboard provides centralized access to all system management functions.
- It displays real-time system statistics and important administrative notices.

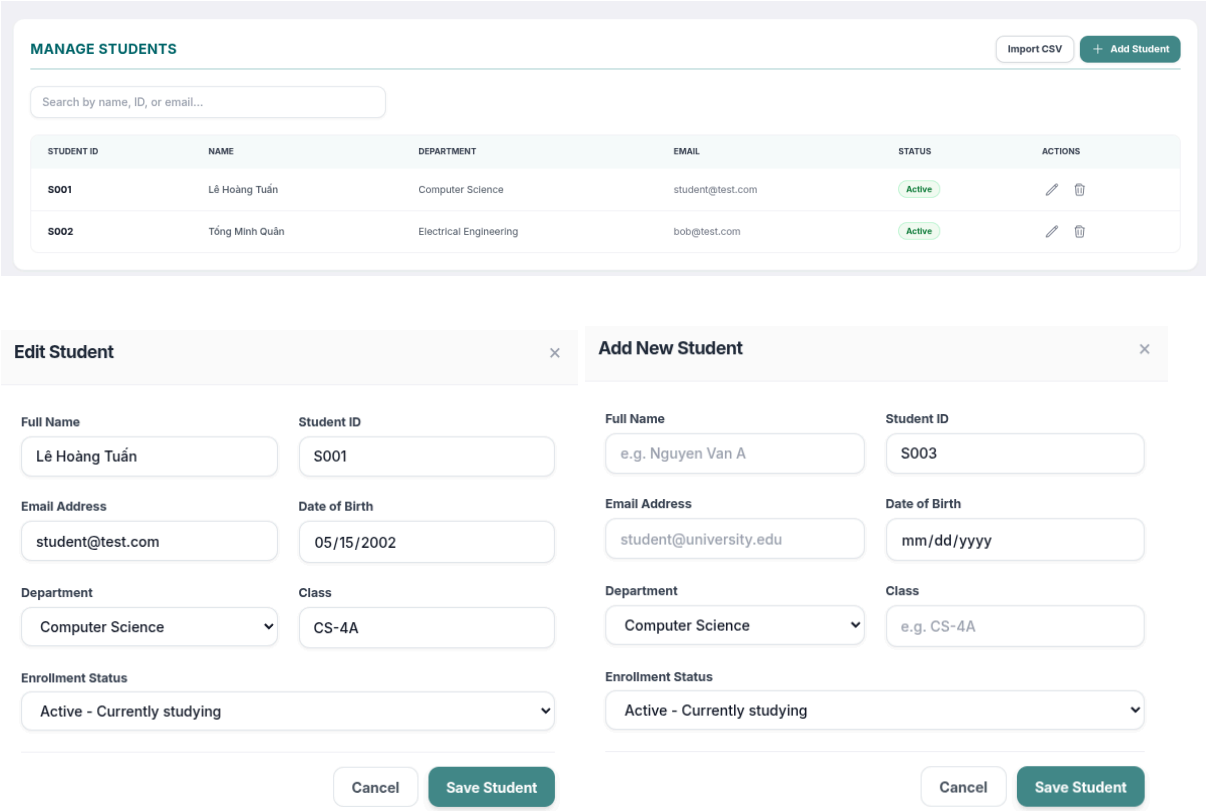
7.5.2 Manage Semesters



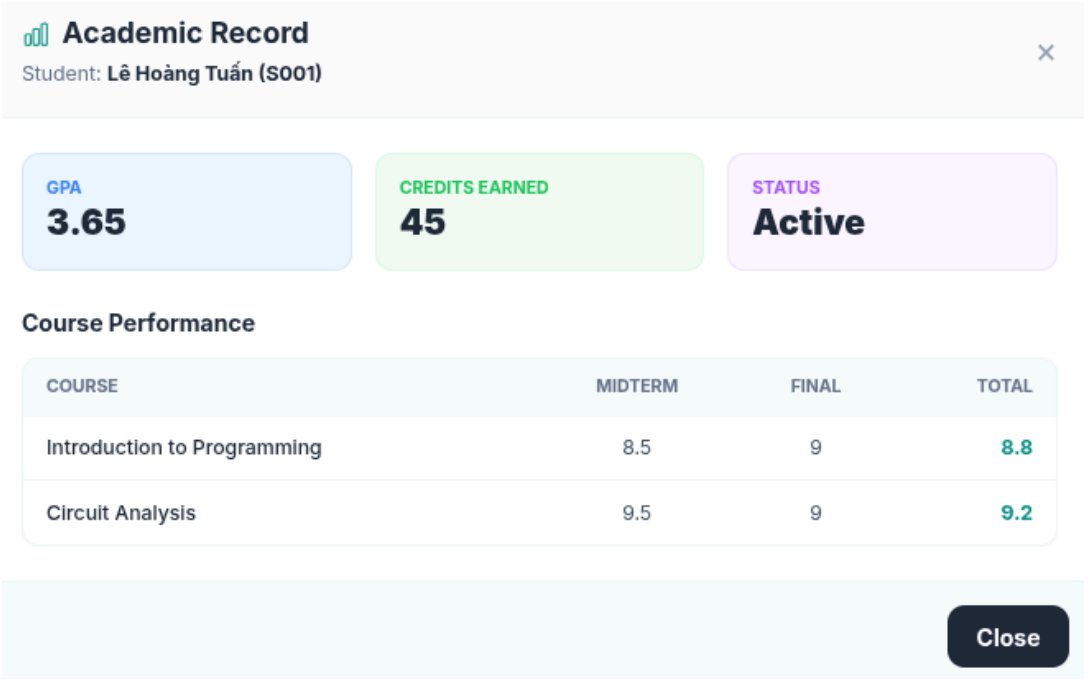


- The system allows the Administrator to create, update, and archive academic semesters.
- Each semester includes start date, end date, and status.
- Changes in semester status automatically affect system functions such as course registration availability.

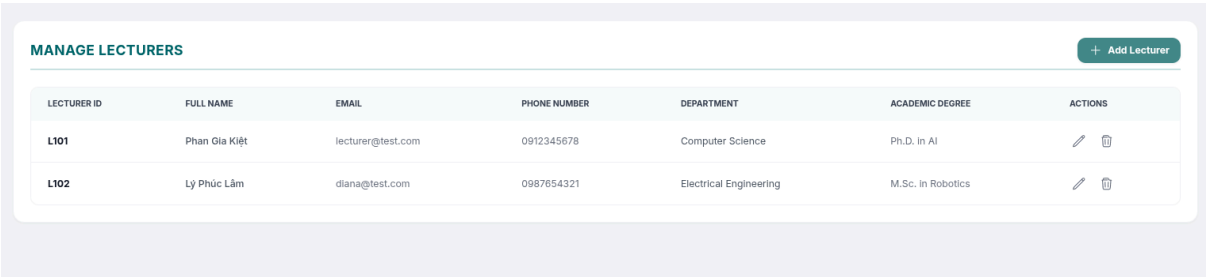
7.5.3 Manage Students

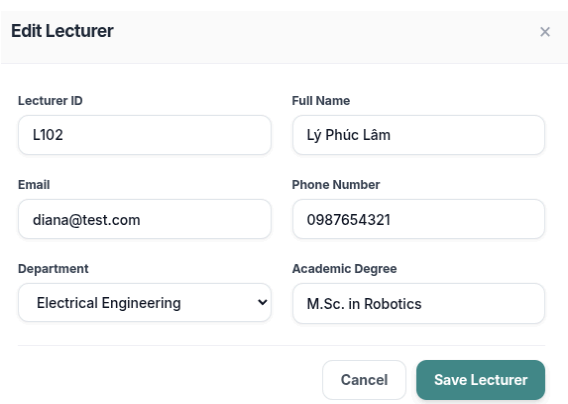


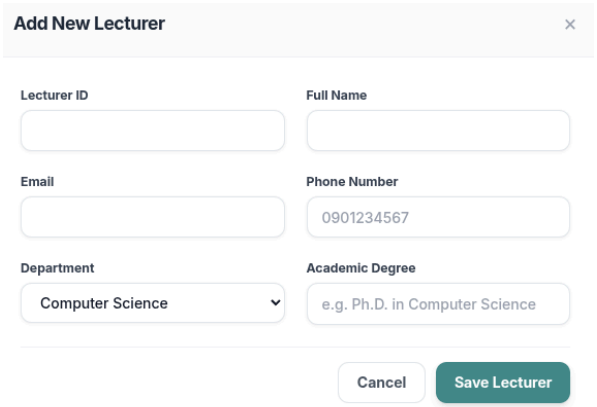
- The system allows the Administrator to manage student records.
- Functions include creating new student accounts, updating student status, and archiving inactive records.
- Bulk operations such as CSV import are supported.
- Viewing the academic results (grade board) of each student



7.5.3 Manage Lecturers

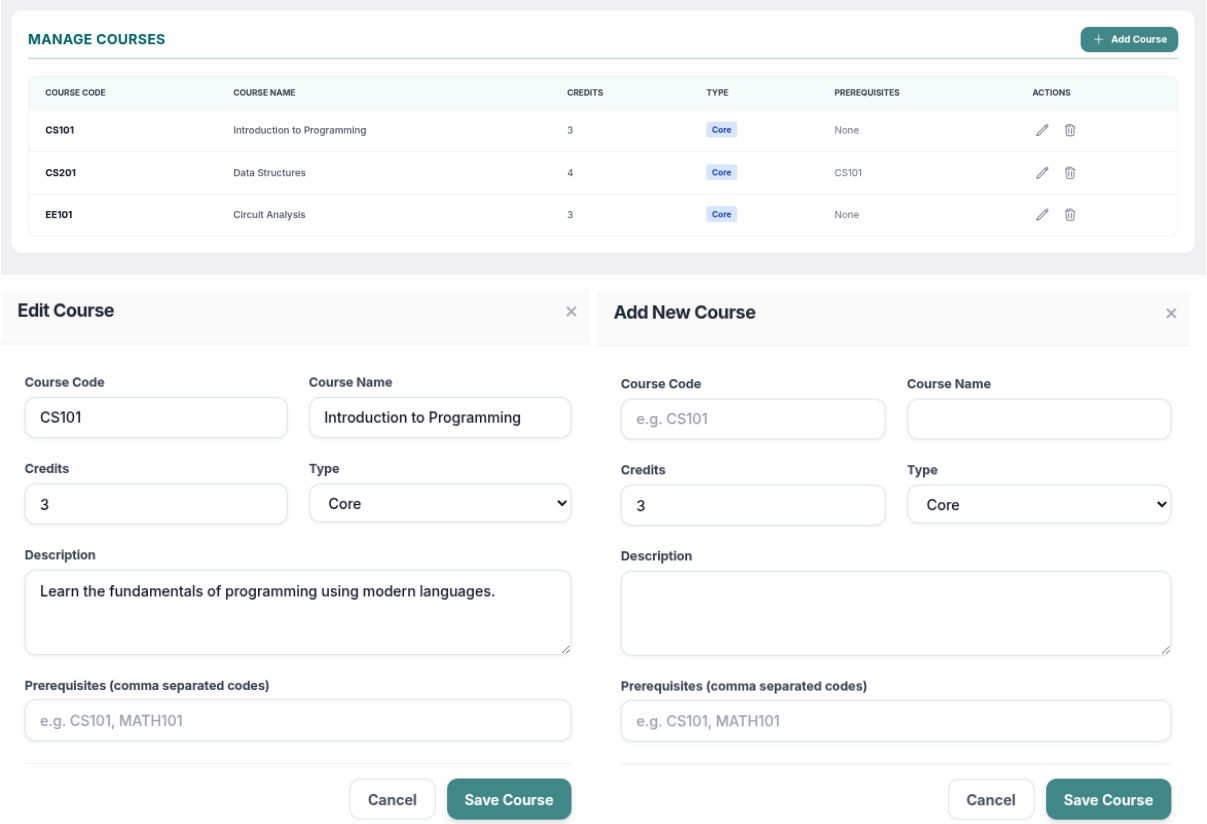






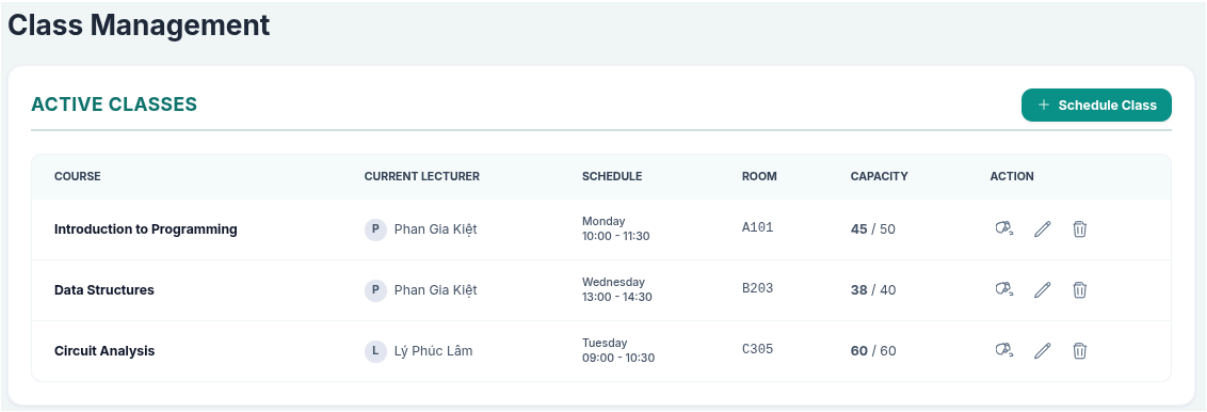
- The system allows the Administrator to manage lecturer profiles.
- Functions include creating, updating, assigning departments, and deactivating lecturer accounts.
- The system prevents deletion of lecturers who are assigned to active classes.

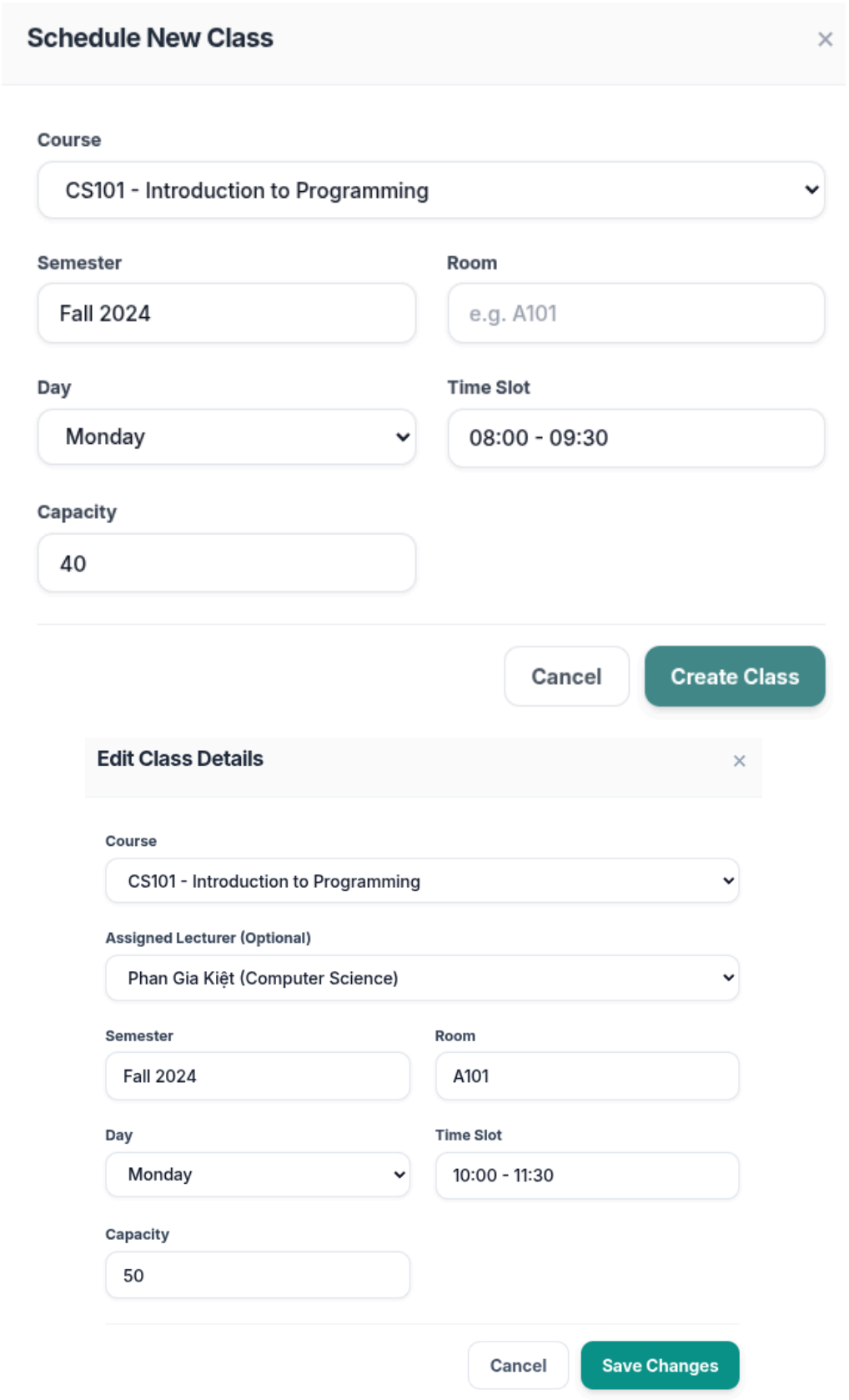
7.5.5 Manage Course



- The system allows the Administrator to create, update, and delete courses in the course catalog.
- Course information includes course code, name, credits, description, prerequisites, and course type.
- All changes are validated and applied consistently across related classes and student records.

7.5.6 Manage Classes

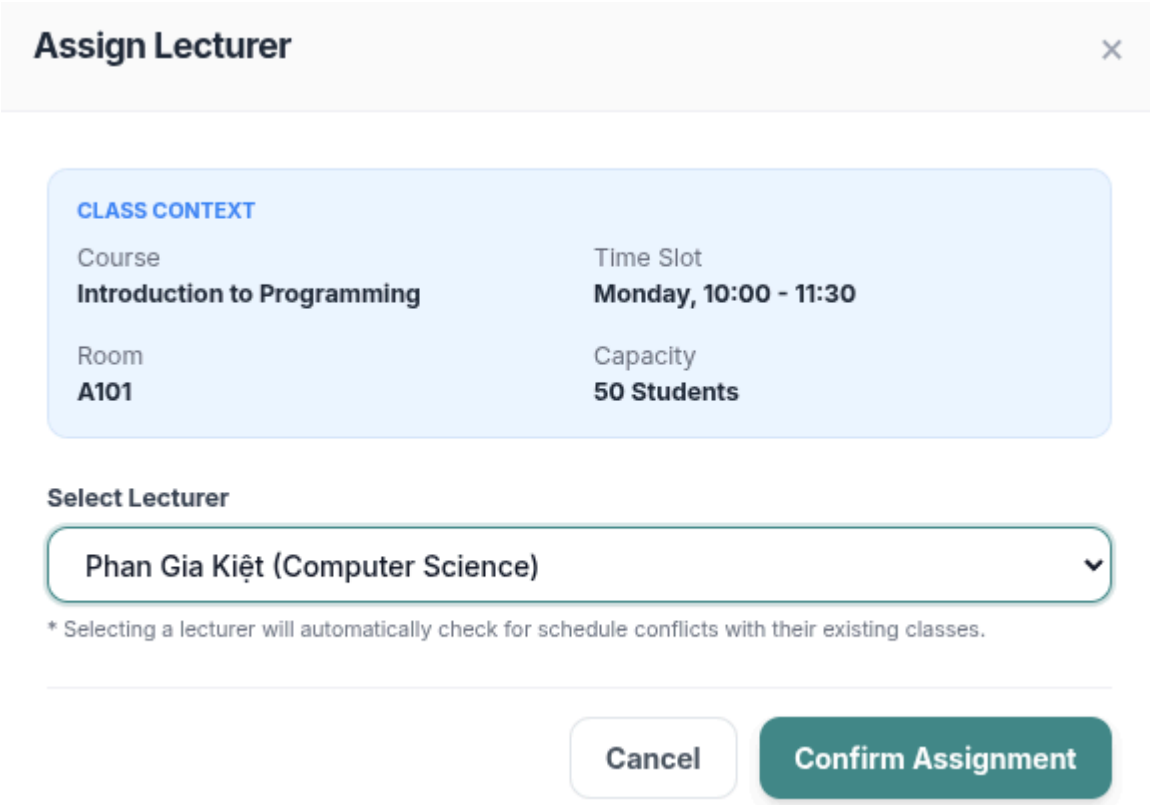




- The system allows the Administrator to create and manage classes for each course and semester.

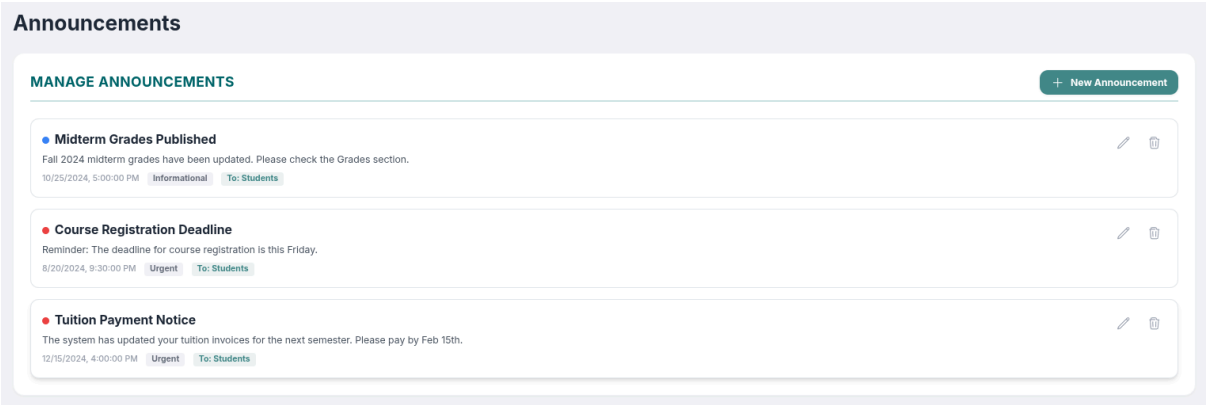
- Class details include class code, schedule, room, and maximum student capacity.

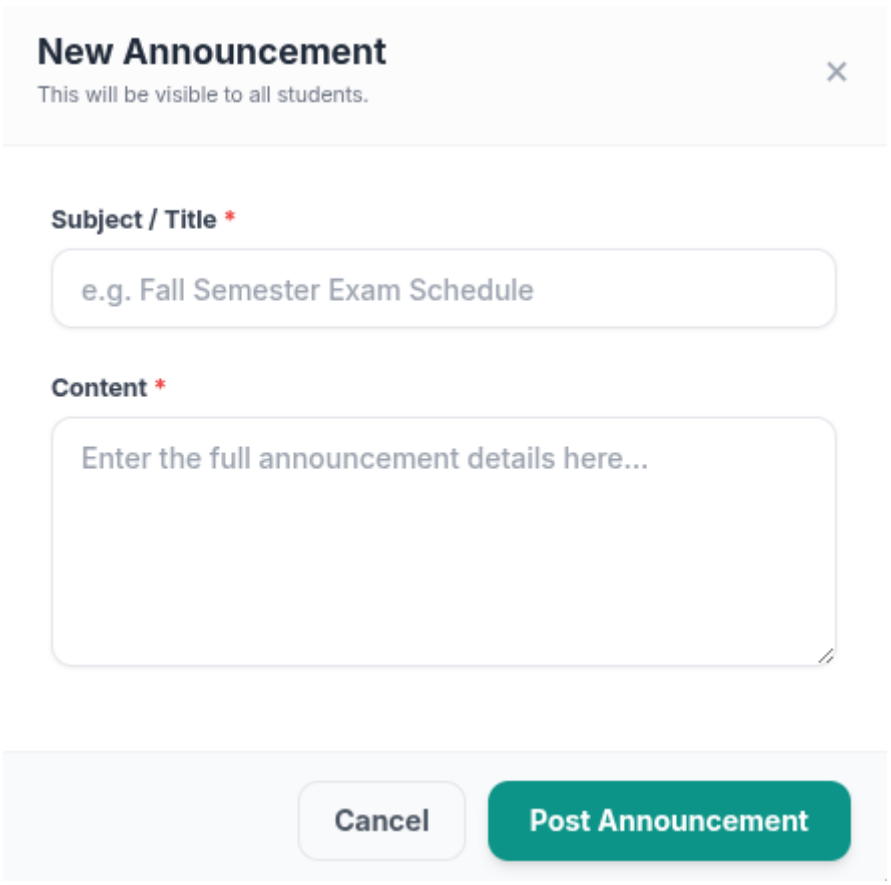
7.5.7 Assign Lecturer to Class



- The system allows the Administrator to assign or change a lecturer for a specific class.
- The system displays available lecturers and checks for schedule conflicts before assignment.

7.5.9 Manage Announcements




New Announcement
This will be visible to all students.
Subject / Title *
e.g. Fall Semester Exam Schedule
Content *
Enter the full announcement details here...
Cancel
Post Announcement