PART 1: Overview & Software Requirements Specification.

1) Introduction:

a) Purpose:

The purpose of this Learning Management System is to develop a platform that makes learning more efficient and advanced for students while providing teachers with the ability to manage the educational content.

b) Project Scope:

The system will provide functionalities for teachers to create and manage courses and exams, students to register for courses and take exams and admins can add and manage users. The system also collects feedback from teachers, students and administrators.

c) Glossary and Abbreviations:

LMS: Learning management system.

SRS: Software Requirements specification.

Admin: The person responsible for maintaining and managing the system.

UI: User interface.

SQL: Structured Query Language for managing a relational database.

Query: Request for data or information from the database.

Backend: The server side of the system.

UMI: Unified Modeling Language used in system design to develop UML diagrams.

Feedback: Responses or reviews about the students and faculty members Performance.

Skill gap analysis: The difference between the skills a person has and the skills needed to achieve a specific goal.

d) List of the System Stakeholders:

Students, Faculty Members and Administrators.

e) References:

Project Description File.

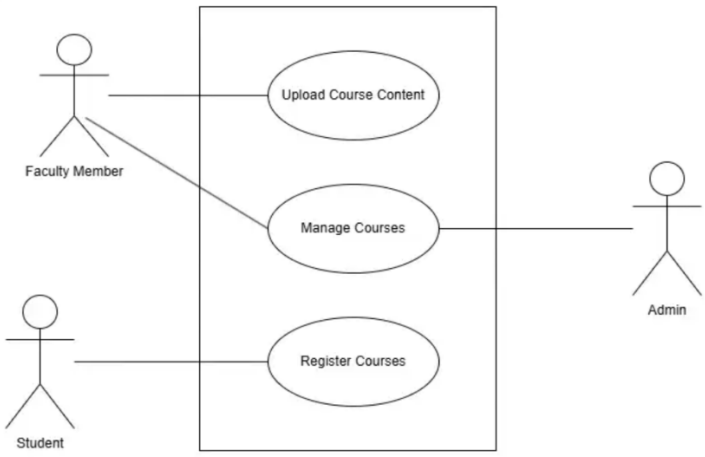
Lecture notes from Dr. Amr Ghoneim CS251 Software Engineering 1 course.

2) Functional Requirements:

a) User Requirements Specification:

1. The Admin should be able to:
    - Manage faculty members and students.
    - Manage courses (course administration).
    - Manage and generate reports.
    - Manage evaluations (add the list of questions to decide the performance of the faculty member).
    - Rate the faculty member according to a question list and their performance.
2. The Faculty Member should be able to:
    - Upload and manage course content (course delivery).
    - Track student progress and submissions.
    - View comments and feedback given by their students.
    - Rate their co-teachers.
3. The Student should be able to:
    - Register courses and access course materials.
    - Take exams and assessments.
    - View their scores, transcripts, and certificates.
    - Give their comments and feedback.

This is a simple use case diagram illustrating the relationship between the actors (students, faculty members, admins) and the courses.



b) System Requirements Specification:

- The System should include at least 3 major parts: an Administration Module, a Student Module, and a Faculty Member Module.
- The System should be able to:
  - Manage users, courses, and roles.
  - Deliver online courses and education by having a Course management system.
  - Maintaining employee records.
  - Generate reports, certificates, and transcripts.
  - Generate a course calendar.
  - Send Messages, Notifications, and Reminders.
  - Include an assessment engine for pre/post-testing.
  - Include a teaching performance evaluation module which handles:
    - The process of adding and managing evaluation questions.
    - Collecting comments, feedback and ratings from students, admins, and teachers.
    - Display evaluation results, final ratings and comments.

c) Requirements' Priorities (using the MoSCoW scheme):

a) User Requirements:

Requirement	**Priority**
Admin can manage faculty members and students	Must Have
Admin can manage courses	Must have
Admin can manage and generate reports	Should Have
Admin can manage evaluations	Must Have
Admin can rate faculty members	Should Have
Faculty member can upload and manage course content	Must Have
Faculty member can track student progress and submissions	Should Have
Faculty member can view comments and feedback given by students	Must Have
Faculty member can rate their co-teachers	Could Have
Students can register courses and access course materials	Must Have
Students can take exams and assessments	Must Have
Students can view their scores, transcripts, and certificates	Should Have
Students can give their comments and feedback	Must Have

b) System Requirements:

Requirement	Priority
The system must include 3 major parts: Admin, Student, Faculty Member Module	Must Have
The system can manage users, courses and roles	Must Have
The system can deliver online courses by having a course management system	Must Have
The system can maintain employee records	Should Have
The system can generate reports, certificates, and transcripts	Should Have
The system can generate a course calendar	Could Have
The system can send messages, notifications, and reminders	Should Have
The system must include an assessment engine for pre/post-testing	Must Have
The system must include a teaching performance evaluation module	Must Have

3) Non-functional requirements:

Performance:

b) The system shall load and initialize all functionalities within **2 seconds** at maximum under normal operating conditions.

to set the Requirements to **mathematical specification** we should declare:

Step 1: Identify Key Elements in the Requirement:

Process: system initialization.

Constraint: Max time = 2 seconds.

Condition: Under normal conditions.

Step 2: Define Variables

T*init* = time to initialize the system

T*max*= maximum allowed time (2 seconds)

NOC: a Boolean condition indicating "normal operating conditions"

Step 3: Use Math/Logic

If NOC = true ⇒ T*init* ≤ 2

c) when the system is started, all core and functionalities must be fully loaded and responsive within 2 second, using testing tools (browser dev tools) under normal operating conditions.

d) To meet the requirement of loading all core functionalities within 2 seconds, the system architecture must be optimized for speed and efficiency.
they may include some points:

-selecting high-performance database.

-Applying asynchronous processing.

-Using front-end framework and minimize large dependencies.

Operational:
b) the system should be available at any time will the user access to it.

c) the system should always be available 24/7 excluding planned maintenance windows.

d) To ensure the system is **available at any time when the user accesses it**, the system architecture must:

Design the system for automatic recovery with database replication and backup strategies.

Implement high availability using load balancing and failover systems to ensure service continuity during server failures or high traffic***.***

Security:

b) The system must enforce role-based access control (RBAC), ensuring that only authorized users (admin, faculty, students) can access specific system functionalities based on their roles.
c) the **access levels** and **permissions** of each user role need to be clearly defined:

admin: should be able to access all system modules and perform all administrative actions (user management, course creation, reporting) without restriction.

Unauthorized access attempts by non-admin users (faculty, students) to admin functions should be blocked, and logged as security events.

Faculty members: should have access to their own courses, assignments, and student evaluation data. Unauthorized attempts to access admin or student-only functionalities should be blocked and logged for audit purposes.

Students: should only be able to view and interact with content for courses they are enrolled in. Unauthorized access attempts to faculty or admin areas should be blocked and logged for monitoring

NFRs-US

Category	Usability-multilingual support
Requirement	The system shall support multiple languages, including at minimum English and Arabic, to accommodate users from different language backgrounds.
Fit Criteria	All interface elements (labels, buttons, error messages) must be fully translated into English and Arabic. Verification will be done through UI testing in each supported language.
Source	Stakeholder Request, Target User Demographics, (in our project we selected from the project details)
Precondition	The user will select the language, and the system will load it.
Postcondition	The selected language will apply with correct text direction
The system architecture	-Use **language resource files**. -Design UI components adaptable to text direction. -Ensure **database fields** support multilingual content (product descriptions in multiple languages).

4) Design & Implementation Constraints:

Technical constraints:

Technical constraints significantly impact design projects because they dictate how far designers can push creative and innovative boundaries.

The LMS must be a web-based application, accessible via modern browsers (Chrome, Firefox, Safari, Edge).

Some examples include:

- Device : screen sizes, etc.
- Accessibility constraints: how voice control and screen readers impact design decisions.
- Performance constraints: the impact of user bandwidth/Internet connectivity, product servers, and tech stacks.
- Tech stack constraints: how front-end and back-end tech impact the design process

Data Constraints:

- System must maintain complete logs of evaluation submissions and edits.
- Reports should include aggregated scores, individual comments, and trend analysis.

Financial constraints:

Financial constraints impact many areas of the design process, including human resources, tools, user research, project scope, and technology. While many see financial constraints as a roadblock, they often drive creative thinking and design innovation through bootstrapping and workarounds.

Some ways financial constraints impact the design process include:

- Limiting the scope of each discipline (research, wireframing, prototyping, interviews, testing, etc.)
- Limits the number of iterations and testing rounds.
- Specifies what tools designers use.
- Determines the size and skill level of the design team.

Legal and regulatory constraints:

Legal constraints impact content and user data the most regarding UX projects. These laws change depending on the country, so designers rely on advice from legal counsel and stakeholders and they also must stay constantly updated with evolving law information to ensure that their designs remain compliant.

Some examples of how legal constraints impact design include:

- Industry-specific regulations: some industries, like financial and healthcare, have laws about privacy and security that significantly impact design–for example, login and authentication procedures.

5) System Evolution:

a) Anticipated Changes

Over time, we can expect various changes that will affect how an LMS should work. Here are the major anticipated changes:

1. **Hardware& Technology Evolution**
    - Faster processors, High upfront costs for server hardware and maintenance, Limited scalability and accessibility, High quality Ram
    - Responsive design for seamless use across devices. such as laptops, smartphones, tablets
    - Adoption Ai tools and reals-time video tools

Expected Change:

LMS should support more advanced features like real-time video lectures, Al tools

2. **Changing User Needs(requirement)**
    - Users (students, faculty, admins) may demand more personalized and engaging experience.
    - More institutions are moving toward blended learning, mobile-first platforms, and integration with other tools like Zoom, Google Classroom, or AI chatbots
3. **More Advanced Evaluation and Feedback Tools**
    - Demand for smarter performance tracking and analytics (AI-based suggestions, dashboards).

Expected Change:

Teacher evaluation modules need to evolve with better analytics and smarter feedback mechanisms.

b) How These Anticipated Changes Should Affect System design:

1. Modular and Scalable Architecture

- Each part (admin module, student module, faculty module, evaluation module) should be a separate, plug-and-play component.
- Easy to update or replace parts without affecting the whole system

Example: **If video assessment becomes a new requirement, a new module can be added without reworking the entire system.**

2. Mobile and Cross-Device Compatibility

- Design the LMS using responsive web design.
- Provide mobile apps or mobile-optimized interfaces.

Example: **Students should be able to complete quizzes and receive notifications on phones or tablets**

3.Customizable Evaluation and Reporting Tools

- Allow admins to add/remove questions, change rating logic, or view performance over time.
- Support flexible roles (students rate faculty, admins rate teachers, co-teachers rate each other, etc.).

Example: **New evaluation formats (like video feedback or peer reviews) can be added over time.**

4. Security and Data Privacy Layers

- Add role-based access controls, encrypted data storage, secure login, etc.
- Provide audit trails and data logs.

Example: **Ensure that students can't view other students' feedback or results, but admins can.**

6) The requirements discovery approaches:

1. Document Analysis:

Reviewing existing documentation and records to extract useful information.

Example: Analyzing the project description/scenario.

2. Use Case Analysis:

Describing specific scenarios/use-cases to see how the system would be used.

Example: Define Scenarios like “A student gives feedback after completing a course.”

3. Comparative Analysis:

Studying existing systems to extract from them useful information.

Example: Checking out how other LMS Platforms like “Blackboard” or “Canvas” implement the Teacher Evaluation Module.

7) Requirements Validation Techniques:

1) Requirements Reviews:

After conclusion the requirements, we meet with stakeholders to review the requirements and ensure they match what they want.

Example 1: Set up interview with Teachers to show “Add Course requirement” in terms of the data required to be stored about the courses.

2) Create Prototyping:

Create a prototype interface to present it to the client to get feedback and ensure that what this he would it.

Example 1: Create prototype “Admin Dashboard” and present it to client and get feedback.

Example 2: Collect opinions from students about “Student Dashboard” and to know their feedback about collected requirements in terms of “Usability”.

3) Checking User Scenarios:

Checking user scenarios and ensuring they match all requirements that have been collected.

Example 1: Collected “User Scenarios” from teachers and reviews it to determine requirements

Example 2: For Scenario “Teacher Add New course” we validate it by:

1) Checking all functions like add videos , add quiz it works well and performs requirements that must be in place.

2) Present it to teachers to ensure that requirements match their real workflow.

4) Gathering Test Cases:

Gathering test cases to test code to ensure the code performs what is required.

Example 1: Equip some student data with some wrong and duplicate data to test adding accounts to student by Admin.

Example 2: Attempt to login with wrong password to test login validation

Example 3: Attempt to add a new course with duplicate data to validate the system prevent duplication or not.

Part 2: System Design & Models :

8) Functional Diagrams:

a) Use-Case Diagram

b) Detailed Use-Cases Description

Login description

ID Number – Name:	Login
Goal:	Allow user to join our systems
Preconditions:	The user has internet access.
Postconditions:	1. The user is successfully authenticated. 2. The user gains access to their account and authorized features.
Initiator(s):	User
Main Success Scenario:	1. User opens the application or website. 2. User navigates to the login screen. 3. User enters their username/email and password. 4. System verifies the credentials. 5. If the credentials are valid, the user is redirected to their dashboard. 6. The system logs the login event for security purposes.
Alternative Scenarios:	1. *Invalid Credentials:* The user enters incorrect login information, and the system displays an error message. 2. *Forgot Password:* The user clicks on "Forgot Password" to reset their password via email. 3. *Account Locked:* The user exceeds maximum login attempts, and the system temporarily locks the account.

Verify password description

ID Number – Name:	Verify password
Goal:	The goal of this use case is to verify that the password provided by a user matches their stored credentials during login.
Preconditions:	1. The user must be authenticated or in the process of logging in. 2. The user must enter a password to verify their identity.
Postconditions:	1. The system confirms whether the entered password matches the stored password. 2. If valid, the user is allowed to proceed with the action . 3. If invalid, the system denies the action and provides an appropriate error message..
Initiator(s):	System
Main Success Scenario:	1. The system prompts the user to enter their current password. 2. The user inputs their password. 3. The system hashes the input and compares it to the stored password hash. 4. If the passwords match, the user is verified and allowed to proceed. 5. A success message is displayed or the action is completed.
Alternative Scenarios:	Incorrect Password*:* The password does not match, and the system shows an error message.

Display error password description

ID Number – Name:	Display error password
Goal:	To inform the user when an incorrect or invalid password is entered.
Preconditions:	1. The user attempts to log in or verify their password. 2. The system compares the entered password with the stored password. 3. The entered password is incorrect.
Postconditions:	1. An appropriate error message is displayed to the user. 2. The user is prompted to re-enter the correct password.
Initiator(s):	System
Main Success Scenario:	1. The user inputs an incorrect password. 2. The system compares the input against the stored credentials. 3. The system detects the mismatch. 4. A user-friendly error message is displayed
Alternative Scenarios:	None specifically defined.

Edit profile description

ID Number – Name:	Edit profile
Goal:	To allow users to update their personal information.
Preconditions:	1. The user must be logged into their account.
Postconditions:	1. The user's updated profile information is saved successfully. 2. The system reflects the updated information in all relevant views..
Initiator(s):	User.
Main Success Scenario:	1. User logs into the application. 2. User navigates to the "Profile" or "Account Settings" section. 3. User clicks “Edit” to update profile fields (e.g., name, phone, etc.). 4. User makes changes and clicks “Save.” 5. System validates the input (e.g., proper email format). 6. System updates the database and confirms changes to the user.
Alternative Scenarios:	No Changes Made*:* If the user saves without editing anything, the system may return a message like “No changes detected.”

Rate faculty member description

ID Number – Name:	Edit profile
Goal:	To allow users to provide feedback by rating faculty members based on their performance, teaching quality, or other relevant criteria.
Preconditions:	1. The user must be logged into the system. 2. The user must have access to the faculty member's profile or course.
Postconditions:	1. The rating is successfully submitted and saved in the system. 2. The rating may be included in the faculty member’s overall evaluation or performance dashboard.
Initiator(s):	User.
Main Success Scenario:	1. User logs into the system. 2. User navigates to the faculty member’s profile or course evaluation page. 3. User selects a rating and optionally adds a comment. 4. User submits the rating. 5. The system validates and saves the feedback. 6. A confirmation message is displayed.
Alternative Scenarios:	1. *Duplicate Rating:* If the user already rated the same faculty member, the system prevents multiple submissions or allows only one update.

Logout description

ID Number – Name:	Edit profile
Goal:	To allow users to exit from the system.
Preconditions:	The user is logged in.
Postconditions:	The session is ended, and the user is redirected to the login page.
Initiator(s):	User.
Main Success Scenario:	1.The user clicks "Log Out." 2.The system terminates the session. 3.The user is redirected to the login page.
Alternative Scenarios:	None specifically defined.

Manage Role description

ID Number – Name:	Manage Role
Goal:	To allow admin to assign, modify, or roles such as(admin, faculty member, student) for users.
Preconditions:	1. The administrator must be authenticated and authorized to manage user roles
Postconditions:	1. The selected user's role is updated successfully. 2. The changes are reflected immediately in the user's access and permissions.
Initiator(s):	Admin.
Main Success Scenario:	1. Admin logs into the system. 2. Admin navigates to the role management section. 3. Admin selects a user to manage his role. 4. Admin chooses a new role or modifies existing roles.
Alternative Scenarios:	No Changes Made: If the admin saves without editing anything.

Manage User description

ID Number – Name:	Manage User
Goal:	To allow administrators to view, search, edit, or delete user accounts to maintain proper user control and system integrity.
Preconditions:	1. The administrator must be logged in and have the necessary permissions. 2. The target user(s) must exist in the database.
Postconditions:	1. The selected user information is updated, deactivated, or deleted as requested. 2. System reflects changes immediately across relevant modules.
Initiator(s):	Admin.
Main Success Scenario:	1. Admin logs into the system. 2. Admin navigates to the "Manage Users" section. 3. Admin can choose an action (add, edit, delete, etc.). 4. System validates the action and applies changes. 5. A success message is shown and the event is logged.
Alternative Scenarios:	1. User Not Found*:* The system displays an error if the user does not exist. 2. Failed Update*:* Due to system or database issues, the action fails and an error message is shown. 3. Action Confirmation Required: For sensitive actions like deletion, the system prompts for confirmation.

Add User description

ID Number – Name:	Add User
Goal:	Allow an admin to add a new user (student or faculty) to the system.
Preconditions:	1. Admin is logged into the. 2. Admin has necessary authorization.
Postconditions:	1.New user account is created and stored in the system.
Initiator(s):	Admin.
Main Success Scenario:	1. Admin logs in. 2. Navigates to the user management section. 3. Selects "Add User". 4. Fills in required user details. 5. Submits the form. 6. System confirms user creation.
Alternative Scenarios:	1. Missing or invalid input: System prompts for corrections. 2. Duplicate user detected: System warns and halts creation.

Delete User description

ID Number – Name:	Delete User
Goal:	Allow an admin to delete a user (student or faculty) from the system.
Preconditions:	1. Admin is logged into the system. 2. Admin has necessary authorization. 3. The user to be deleted exists in the system.
Postconditions:	1. The user account is permanently removed from the system.
Initiator(s):	Admin.
Main Success Scenario:	1. Admin logs in. 2. Navigates to the user management section. 3. Selects the user to delete. 4. Confirms the deletion. 5. System removes the user and displays confirmation.
Alternative Scenarios:	1. Attempt to delete non-existent user: System shows error.

Edit User description

ID Number – Name:	Edit User
Goal:	Allow an admin to modify details of an existing user (student or faculty) in the system.
Preconditions:	1. Admin is logged into the system. 2. Admin has necessary authorization. 3. The user exists in the system.
Postconditions:	1. The user's updated information is saved and reflected in the system.
Initiator(s):	Admin.
Main Success Scenario:	1. Admin logs in. 2. Navigates to the user management section. 3. Selects a user to edit. 4. Updates the user details. 5. Saves the changes. 6. System confirms the update.
Alternative Scenarios:	1. Attempting to edit user information while the information is present.

Assign course to teacher description

ID Number – Name:	Assign Course to Teacher
Goal:	Allow an admin to assign a course to a teacher in the system.
Preconditions:	1. Admin is logged into the LMS. 2. The teacher and course exist in the system.
Postconditions:	1. The selected teacher is assigned to the specified course.
Initiator(s):	1. Admin
Main Success Scenario:	1. Admin logs in. 2. Navigates to the course assignment section. 3. Selects a course. 4. Selects a teacher. 5. Confirms the assignment. 6. System links the teacher to the course and shows a success message.
Alternative Scenarios:	1. Course or teacher does not exist: System displays an error. 2. Course already assigned: System warns and requests confirmation for reassignment.

Generate reports description

ID Number – Name:	Generate report
Goal:	To allow administrators to schedule, generate, and access reports based on system data, enabling performance tracking, auditing, and decision-making.
Preconditions:	1. The administrator must be logged in and have the necessary permissions. 2. Relevant data must be available and up-to-date in the system.
Postconditions:	1. The requested report is successfully generated and available for download, viewing, or distribution. 2. Scheduled reports are generated automatically at the specified times and delivered as configured.
Initiator(s):	Admin.
Main Success Scenario:	1. Admin logs into the system. 2. Admin navigates to the "Reports" section. 3. Admin selects the type of report to generate. 4. Admin submits the report generation request. 5. System processes the request and generates the report. 6. The report is made available to the admin for viewing, downloading, or distribution.
Alternative Scenarios:	No Data Available: The system displays an error if there is insufficient data for the requested report. Invalid Scheduling Parameters: If the scheduling information is invalid (e.g., incorrect dates), the system prompts the admin to correct it. Delivery Failure: If the report cannot be emailed or saved to a target location, the system notifies the admin and logs the issue.

Create evaluation question description

ID Number – Name:	Create evaluation question.
Goal:	To allow administrators to create a list of questions that will be used to evaluate the teachers.
Preconditions:	1. The administrator must be logged into the system with permissions to manage review/rating templates. 2. There must be an interface for managing questions
Postconditions:	1.A new list of questions is successfully added and available for the user to evaluation . 2.The questions are saved in the database and linked to the appropriate evaluation process.
Initiator(s):	Admin.
Main Success Scenario:	1.Admin logs into the system. 2.Admin navigates to the "Manage Rating Questions" section. 3.Admin selects "Add New List." 4.Admin enters each question manually or uploads a prepared list. 5.Admin reviews and confirms the entered questions. 6.System saves the new list of questions and associates it with the corresponding evaluation form or template. 7.System displays a confirmation message and makes the questions available for immediate use.
Alternative Scenarios:	Duplicate Questions: If identical questions already exist, the system warns and asks whether to continue, update, or cancel..

Manage evaluation description

ID Number – Name:	Manage evaluation
Goal:	To allow administrators to view, edit, and delete evaluations, ensuring proper review .
Preconditions:	1. The administrator (or evaluator) must be logged into the system with the appropriate permissions. 2. Evaluation templates or rating questions must be available in the system
Postconditions:	1. Evaluations are successfully updated, or deleted as requested. 2. The system reflects changes immediately across reports and dashboards.
Initiator(s):	Admin.
Main Success Scenario:	1. Admin logs into the system. 2. Admin navigates to the "Manage Evaluation" section. 3. Admin chooses an action: edit an existing one, or delete an evaluation. 4. Admin fills out or updates evaluation responses (ratings, comments, feedback). 5. Admin submits the evaluation. 6. System saves the evaluation and updates related reports and statistics.
Alternative Scenarios:	Failed Save in database.

Delete evaluation description

ID Number – Name:	Delete evaluation
Goal:	To allow administrators to delete evaluations, ensuring proper review .
Preconditions:	1. The administrator (or evaluator) must be logged into the system with the appropriate permissions. 2. Evaluation templates or rating questions must be available in the system
Postconditions:	1. Evaluations are successfully deleted as requested. 2. The system reflects changes immediately across reports and dashboards.
Initiator(s):	Admin.
Main Success Scenario:	1. Admin logs into the system. 2. Admin navigates to the "Manage Evaluation" section. 3. Admin delete an evaluation. 4. Admin submits the evaluation. 5. System saves the evaluation and updates related reports and statistics.
Alternative Scenarios:	Failed Save in database.

Edit evaluation description

ID Number – Name:	Edit evaluation
Goal:	To allow administrators to view, edit ensuring proper review .
Preconditions:	1. The administrator (or evaluator) must be logged into the system with the appropriate permissions. 2. Evaluation templates or rating questions must be available in the system
Postconditions:	1. Evaluations are successfully edited as requested. 2. The system reflects changes immediately across reports and dashboards.
Initiator(s):	Admin.
Main Success Scenario:	1. Admin logs into the system. 2. Admin navigates to the "Manage Evaluation" section. 3. Admin edit an existing one. 4. Admin submits the evaluation. 5. System saves the evaluation and updates related reports and statistics.
Alternative Scenarios:	Failed Save in database.

Manage course description

ID Number – Name:	Manage course

Goal:	Allows administrators to create, view, update, and delete courses, which are then displayed to users.
Preconditions:	1. The administrator must be logged into the system with course management permissions. 2. Basic course categories or departments must exist if courses are to be associated with them.
Postconditions:	1. Courses are successfully created, updated, or deleted as needed. 2. The updated course information is reflected immediately in course catalogs, schedules, and user-facing views.
Initiator(s):	1. Admin .
Main Success Scenario:	1. Admin logs into the system. 2. Admin navigates to the "Manage Courses" section. 3. Admin chooses an action: create a new course, edit an existing course, or delete a course. 4. Admin submits the course details. 5. System validates the input and saves the course details. 6. System confirms the successful action and updates the course listing.
Alternative Scenarios:	Missing Required Fields. failed save due to database.

Add course description

ID Number – Name:	Add course
Goal:	Allow an Admin to add a new course to the system.
Preconditions:	1. user is logged into the. 2. user has necessary authorization.
Postconditions:	1.New course is created and stored in the system.
Initiator(s):	1.Admin.
Main Success Scenario:	1. user logs in. 2. Navigates to the course management section. 3. Selects "Add course". 4. Fills in required user details. 5. Submits the form. 6. System confirms user creation.
Alternative Scenarios:	1. Duplicate course detected: System warns and halts creation.

Delete course description

ID Number – Name:	Delete course
Goal:	Allow an Admin to delete a course from the system.
Preconditions:	1. user is logged into the system. 2. user has necessary authorization. 3. The course to be deleted exists in the system.
Postconditions:	1. The course is permanently removed from the system.
Initiator(s):	1.Admin.
Main Success Scenario:	1. Admin logs in. 2. Navigates to the course management section. 3. Selects the course to delete. 4. Confirms the deletion. 5. System removes the user and displays confirmation.
Alternative Scenarios:	1. Attempt to delete non-existent course: System shows error.

Edit course description

ID Number – Name:	Edit course
Goal:	Allow an admin to modify details of an existing course in the system.
Preconditions:	1. user is logged into the system. 2. user has necessary authorization. 3. The course exists in the system.
Postconditions:	1. The course's updated information is saved and reflected in the system.
Initiator(s):	1.Admin.
Main Success Scenario:	1. Admin logs in. 2. Navigates to the course management section. 3. Selects a course to edit. 4. Updates the course details. 5. Saves the changes. 6. System confirms the update.
Alternative Scenarios:	1. Attempting to edit user information while the information is present.

Announce schedule exam description

ID Number – Name:	Announce schedule exam
Goal:	To allow faculty members or administrators to announce the schedule of an upcoming exam.
Preconditions:	1. The exam must be created and saved. 2. User must be authorized to schedule exams.
Postconditions:	1. The exam schedule is published and viewable by students. 2. Notifications may be sent automatically.
Initiator(s):	1.Admin 2. Faculty member.
Main Success Scenario:	1.User logs in and navigates to a course or exam module. 2. Clicks “Announce Exam Schedule.” 3. Confirm the schedule set by system. 4. System publishes the schedule and sends notifications (if applicable).
Alternative Scenarios:	Scheduling Conflict: System notifies if there's a conflict with another exam. Missing Exam: If no exam is selected, system prompts for one.

Track course progress description

ID Number – Name:	Track Course Progress
Goal:	Allow faculty to view a student's progress in their course
Preconditions:	1. faculty member is logged into the system. 2.There is at least one student in the teacher's course..
Postconditions:	1. The faculty member is shown progress indicators like completed modules, grades.
Initiator(s):	1. faculty member.
Main Success Scenario:	1. faculty member logs in. 2. Navigates to the course dashboard. 3. Selects a course. 4. System displays progress data (e.g., completed lessons, grades, submissions).
Alternative Scenarios:	1. No progress recorded yet: System displays a message like “No activity recorded.” 2. Course not found or access denied: System shows an error message.

Create exam description

ID Number – Name:	Create exam
Goal:	To allow faculty members to create exams for their courses.
Preconditions:	1.The faculty member must be logged in. 2.The faculty member must have an active course assigned.
Postconditions:	1. The exam is successfully created and saved in the system. 2. It becomes available for scheduling and distribution.
Initiator(s):	Faculty member.
Main Success Scenario:	1. Faculty logs in and selects a course. 2. Clicks "Create Exam." 3. Enters exam details (title, instructions, time limit, etc.). 4. Adds questions (manual input or import). 5. Clicks "Save." 6. System confirms creation and stores the exam.
Alternative Scenarios:	Incomplete Exam Details: System prompts for completion. Question Format Errors: The system requests corrections.

Upload material description

ID Number – Name:	Upload material.
Goal:	To allow faculty members to upload learning materials (PDFs, slides, videos) to their respective courses.
Preconditions:	1. The faculty member must be logged into the system. 2. The course must exist and be assigned to the faculty member.
Postconditions:	1. The material is successfully uploaded and becomes available to students enrolled in the course. 2. The material is stored in the course repository.
Initiator(s):	Faculty member.
Main Success Scenario:	1. Faculty member logs and Navigates to the course dashboard. 2. Selects "Upload Material." 3. Chooses the file or video and enters metadata and Clicks "Upload." 4. The system saves the material and confirms successful upload.
Alternative Scenarios:	material Too Large: System shows error if the material exceeds allowed size.

Upload assignment description

ID Number – Name:	Upload assignment .
Goal:	To allow faculty members to upload learning assignment to their respective courses.
Preconditions:	1. The faculty member must be logged into the system. 2. The course must exist and be assigned to the faculty member.
Postconditions:	1. The assignment is successfully uploaded and becomes available to students enrolled in the course. 2. The assignment is stored in the course repository.
Initiator(s):	Faculty member.
Main Success Scenario:	1. Faculty member logs and Navigates to the course dashboard. 2. Selects "Upload assignment ." 3. Chooses the file and enters metadata and Clicks "Upload." 4. The system saves the file and confirms successful upload.
Alternative Scenarios:	File Too Large: System shows error if the file exceeds allowed size. Unsupported Format: System prompts the user to upload a supported file type.

Upload video description

ID Number – Name:	Upload video.
Goal:	To allow faculty members to upload learning materials (videos) to their respective courses.
Preconditions:	1. The faculty member must be logged into the system. 2. The course must exist and be assigned to the faculty member.
Postconditions:	1. The video is successfully uploaded and becomes available to students enrolled in the course. 2. The video is stored in the course repository.
Initiator(s):	Faculty member.
Main Success Scenario:	1. Faculty member logs and Navigates to the course dashboard. 2. Selects "Upload video." 3. Chooses the video and enters metadata and Clicks "Upload." 4. The system saves the video and confirms successful upload.
Alternative Scenarios:	video Too Large: System shows error if the video exceeds allowed size.

Check assignment description

ID Number – Name:	Check Assignment.
Goal:	Enable a faculty member to review assignments they have submitted by students .
Preconditions:	1. Faculty member logs into the system. 2. The faculty member is assigned to at least one course with existing assignments.
Postconditions:	1. Faculty can view assignment details, student submissions.
Initiator(s):	1. Faculty Member
Main Success Scenario:	1. Faculty logs in. 2. Navigates to their course dashboard. 3. Selects "Assignments." 4. System displays a list of assignments associated with the course. 5. Faculty views submission statuses and assignment parameters..
Alternative Scenarios:	1. No assignments have been created yet: System indicates "No assignments found."2. Faculty lacks permissions for selected course: Access is denied with an appropriate error message.

Rate co-teacher description

ID Number – Name:	Rate Co-Teacher
Goal:	Allow a faculty member to rate or provide feedback on a co-teacher they are collaborating with in a course.
Preconditions:	1. Faculty is logged into the system. 2. The faculty member shares a course with the co-teacher. 3. Rating feature is enabled by the institution.
Postconditions:	1. rating is recorded in the system for review or reporting.
Initiator(s):	1. Faculty Member
Main Success Scenario:	1. Faculty logs in. 2. Navigates to the shared course page. 3. Selects the co-teacher rating section. 4. Submits a rating. 5. System confirms submission and stores the data.
Alternative Scenarios:	1. Co-teacher not found or not associated with the course: System displays an error.

Submit assignment description

ID Number – Name:	Submit Assignment
Goal:	Allow a student to submit an assignment for a course through the system.
Preconditions:	1. Student is logged into the system. 2. The assignment is available for submission. 3. The student is registered in the course.
Postconditions:	1. The assignment file is successfully uploaded and recorded in the system.
Initiator(s):	1. Student
Main Success Scenario:	1. Student logs in. 2. Navigates to the course and assignment section. 3. Selects the assignment. 4. Uploads the required file(s). 5. Clicks "Submit." 6. System confirms successful submission.
Alternative Scenarios:	1. File format or size is invalid: System shows error and prompts correction. 2. Assignment deadline has passed: System prevents submission and displays a message.

View final rating description

ID Number – Name:	View final rating.
Goal:	Allow a faculty member to view his final rating or evaluation score of a student or co-teacher..
Preconditions:	1.Faculty member is logged into the LMS. 2.Final ratings are already calculated and available.
Postconditions:	1.Final rating is displayed to the faculty member.
Initiator(s):	Faculty member.
Main Success Scenario:	1 Faculty member logs into the system. 2 Faculty member navigates to final rating . 3 Faculty member reviews the final rate and related details.
Alternative Scenarios:	There is no rating.

View comments description

ID Number – Name:	View comments.
Goal:	Enable a faculty member to view comments provided by students, reviewers.
Preconditions:	1.Faculty member is logged into the system. 2.Comments are available and assigned to relevant content.
Postconditions:	1.Comments are displayed to the faculty member.
Initiator(s):	Faculty member.
Main Success Scenario:	1.faculty member logs into the system. 2.faculty navigates to section with feedback. 3.faculty member view the comments.
Alternative Scenarios:	There is no comments.

Answer evaluation question description

ID Number – Name:	Answer evaluation question.
Goal:	Allow a faculty member and student to answer a question that is intended to be rated, and receive feedback or a score.
Preconditions:	1.The student is logged into the LMS. 2.The question is available and marked for rating.
Postconditions:	1.The student's answer is saved.
Initiator(s):	1.Student. 2.faculty member.
Main Success Scenario:	1.user logs in. 2.Navigates to evaluation. 3.Answer the question. 4.Save the answers.
Alternative Scenarios:	Incomplete question Details: System prompts for completion.

Register course description

ID Number – Name:	Register course.
Goal:	Allow a student to register available courses into their academic schedule.
Preconditions:	1. The student is logged into the system. 2. The registration period is open.
Postconditions:	1. Selected courses are registered and added to the student's schedule. 2. Course availability is updated.
Initiator(s):	Student.
Main Success Scenario:	1. Student logs in. 2. Navigates to the course registration page. 3. Selects desired courses. 4. System validates prerequisites and schedule. 5. Student confirms registration. 6. Courses are added to schedule.
Alternative Scenarios:	Course full, time conflict, or prerequisites not met: System displays error and prompts for alternate selection.

Take exam description

ID Number – Name:	Take exam.
Goal:	Allow a student to access and complete an online exam within the system.
Preconditions:	1. The student is logged into the LMS. 2. The exam is scheduled and available. 3. The student is eligible to take the exam.
Postconditions:	1. The student's answers are saved. 2. The exam is submitted for grading.
Initiator(s):	Student.
Main Success Scenario:	1. Student logs in. 2. Navigates to the exam section. 3. Selects the assigned exam. 4. Answers all questions. 5. Submits the exam. 6. System confirms submission.
Alternative Scenarios:	1. Exam not available: system displays an error. 2. Incomplete answers: system prompts to review. 3. Session timeout: system auto-submits saved answers.

View schedule course description

ID Number – Name:	View schedule course.
Goal:	Allow a student to view the list of courses they have registered for in the current academic schedule.
Preconditions:	1. The student is logged into the system. 2. The student has registered for at least one course.
Postconditions:	1. The system displays the student's current schedule with course details.
Initiator(s):	Student.
Main Success Scenario:	1. Student logs in. 2. Navigates to the "My Schedule" section. 3. System displays the list of registered courses with time and location details.
Alternative Scenarios:	No courses registered: System displays a message indicating that no courses are currently scheduled.

View schedule exam description

ID Number – Name:	View schedule exam.
Goal:	Allow a student to view upcoming exam dates, times, and locations for their registered courses.
Preconditions:	1. The student is logged into the system. 2. Exam schedules have been published.
Postconditions:	1. The system displays the list of scheduled exams for the student.
Initiator(s):	Student.
Main Success Scenario:	1. Student logs in. 2. Navigates to the "Exam Schedule" section. 3. System retrieves and displays the list of scheduled exams.
Alternative Scenarios:	No exams scheduled: System displays a message indicating that no exams are currently scheduled.

Check course registered description

ID Number – Name:	Check Course Registered
Goal:	Allow a student to view the list of courses they are registered in.
Preconditions:	1. The student is logged into the system. 2. The student has registered for at least one course.
Postconditions:	1. The student sees a list of currently registered courses.
Initiator(s):	1. system
Main Success Scenario:	1. Student logs in. 2. Navigates to the "My Courses" or "Registered Courses" section. 3. System displays the list of registered courses.
Alternative Scenarios:	1. Student has not registered for any courses: System displays a message saying "No registered courses found."

Search for courses description

ID Number – Name:	Search for courses exam.
Goal:	Allow a student to search for available online courses filters such as course name, department, instructor.
Preconditions:	1. The user is logged into the system. 2. Course data is available in the system,
Postconditions:	1. A list of matching courses is displayed to the user.
Initiator(s):	Student.
Main Success Scenario:	1. User logs in. 2. Navigates to the "Search Courses" section. 3. Enters search keywords or selects filters. 4. System displays a list of courses that match the criteria.
Alternative Scenarios:	1. No matching results: System displays "No courses found." 2. Invalid search input: System prompts the user to enter valid search criteria.

Drop course description

ID Number – Name:	Drop course exam.
Goal:	Allow a student to remove a previously registered course from their schedule within the allowed drop period
Preconditions:	1. The student is logged into the system. 2. The drop period is currently active. 3. The student has at least one registered course.
Postconditions:	1. The course is removed from the student's schedule. 2. The course seat availability is updated. 3. The student's academic record is updated accordingly.
Initiator(s):	Student.
Main Success Scenario:	1. Student logs in. 2. Navigates to "My Courses" section. 3. Selects the course to drop. 4. Confirms the drop action. 5. System removes the course from the schedule and updates records.
Alternative Scenarios:	1. Drop period has ended: System blocks the action and shows an error. 2. Attempt to drop a mandatory course: System prevents the drop and notifies the student. 3. Technical issue: System shows an error and advises retry.

Print certificate description

ID Number – Name:	Print certificate exam.
Goal:	Allow a student to generate and print a certificate of course completion or academic achievement from the system.
Preconditions:	1. The student is logged into the system. 2. The student has completed the required course(s) or program. 3. Certificate generation is enabled by the system.
Postconditions:	1. A downloadable and printable certificate is generated and displayed.
Initiator(s):	Student.
Main Success Scenario:	1. Student logs in. 2. Navigates to "Certificates" section. 3. Clicks on "Generate Certificate." 4. System creates and displays the certificate. 5. Student downloads or prints it.
Alternative Scenarios:	1. Courses not completed: System notifies the student that the certificate is unavailable. 2. Certificate generation error: System displays a failure message and logs the error.

View grades description

ID Number – Name:	View grades exam.
Goal:	Allow a student to view their grades for completed courses, assignments, and exams.
Preconditions:	1. The student is logged into the system. 2. Grades have been entered and published by instructors.
Postconditions:	1. The system displays the student's grades .
Initiator(s):	Student.
Main Success Scenario:	1. Student logs in. 2. Navigates to the "Grades" section. 3. System retrieves and displays grades for all completed assessments and courses.
Alternative Scenarios:	1. Grades not yet published: System displays a message indicating grades are not available. 2. System error: An error message is shown and user is advised to try again later.

Give a comment description

ID Number – Name:	Give a comment.
Goal:	Allow a student to rate a teacher and leave a comment based on their experience in a course.
Preconditions:	1. The student is logged into the system. 2. The student is/was enrolled in a course taught by the teacher. 3. The rating period is open.
Postconditions:	1. The rating and comment are submitted and stored in the system. 2. Feedback becomes available to academic administrators or relevant staff.
Initiator(s):	Student.
Main Success Scenario:	1. Student logs in. 2. Navigates to the "Rate My Instructor" section. 3. Selects a completed course. 4. Enters a comment. 5. Submits the feedback. 6. System stores the input and confirms submission.
Alternative Scenarios:	1. Feedback period closed: System prevents submission and notifies the student. 2. Student not enrolled in the course: Access is denied. 3. System error: Displays error message and suggests retry.

View transcript description

ID Number – Name:	View transcript.
Goal:	Allow students to view their academic transcript, including completed courses, grades, credit hours, and GPA.
Preconditions:	1. The student is logged into the academic portal system. 2. The transcript data is available and up to date in the system.
Postconditions:	1. The student successfully views their academic transcript. 2. The system may log the access or allow printing/export.
Initiator(s):	Student.
Main Success Scenario:	1. Student logs into the portal. 2. Navigates to the "Transcript" or "Academic Records" section. 3. System fetches and displays the student's transcript. 4. Student reviews the transcript. 5. Optionally, the student prints or downloads the transcript.
Alternative Scenarios:	Transcript Unavailable: System displays an error message if the transcript cannot be retrieved.

Watch online course description

ID Number – Name:	Watch online course.
Goal:	Allow students to view their academic transcript, including completed courses, grades, credit hours, and GPA.
Preconditions:	1. The student is logged into the system. 2. The course content (videos) is published and available.
Postconditions:	1. The video course is marked as viewed or in progress. 2. Student's viewing activity is logged for tracking/completion.
Initiator(s):	Student.
Main Success Scenario:	1. Student logs into the system. 2. Navigates to "My Courses" section. 3. Selects the course to watch. 4. The system loads and streams the video content. 5. The student watches the video. 6. The system updates viewing progress.
Alternative Scenarios:	Video Not Loading: The system displays an error and suggests checking the internet connection or trying again later.

Download online course description

ID Number – Name:	Download online course.
Goal:	Allow students to download course materials (e.g., videos, PDFs, presentations) for offline access.
Preconditions:	1. The student is logged into the system . 2. The student is enrolled in the course. 3. Downloadable content is available.
Postconditions:	1. Selected course materials are downloaded to the student's device. 2. The system may log or track the download activity.
Initiator(s):	Student.
Main Success Scenario:	1. Student logs into the system. 2. Navigates to "My Courses" and selects a course. 3. Student clicks on the download icon next to available materials. 4. The system prepares and initiates the download. 5. Files are saved to the student's device for offline use.
Alternative Scenarios:	Download Restricted**:** The content is not available for download; the system notifies the student.

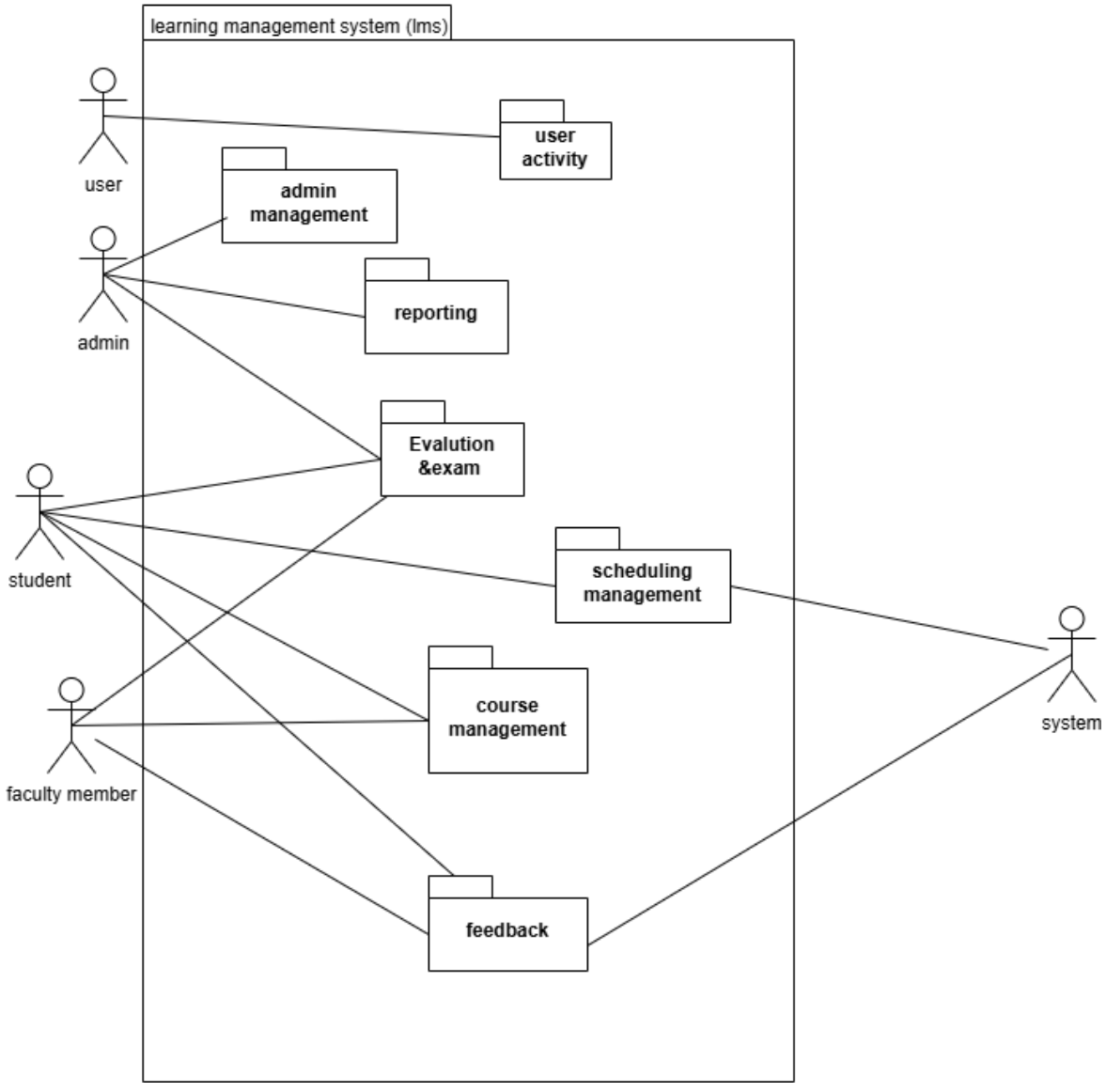
Send notification description

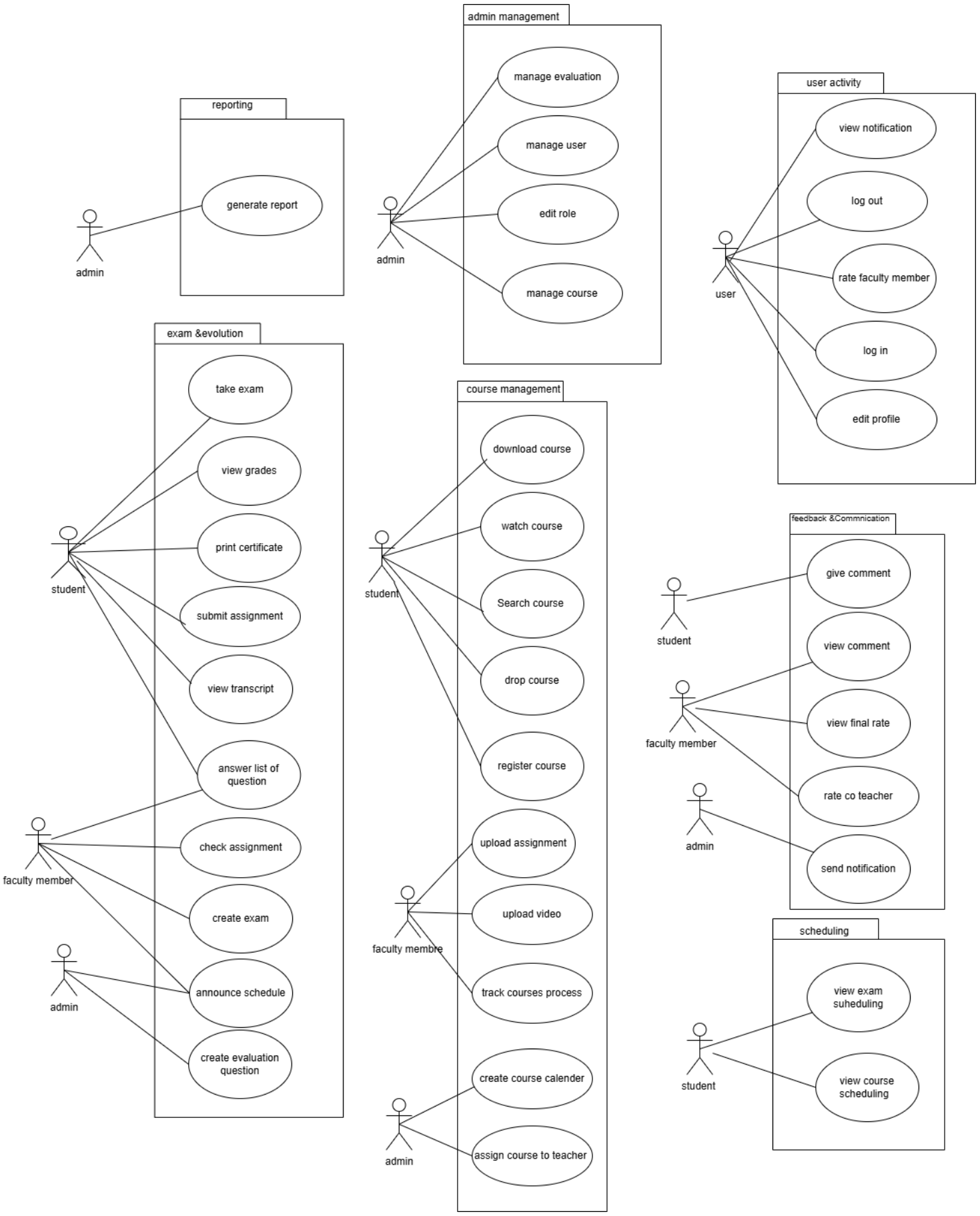
ID Number – Name:	Send notification.
Goal:	Enable the system to send notifications to students about important updates, deadlines, or exam.
Preconditions:	1. Sender system has appropriate access rights. 2. The recipient faculty member or student is registered in the system.
Postconditions:	1. Notification is delivered to the recipient via the selected channel(s). 2. Notification is logged in the system for record-keeping.
Initiator(s):	Admin.
Main Success Scenario:	1. The sender system creates a notification message. 2. Selects target recipients (all students, a faculty member) 3. Sends the notification. 4. System delivers the notification and confirms success.
Alternative Scenarios:	invalid Recipient: System fails to find the recipient; an error is displayed.

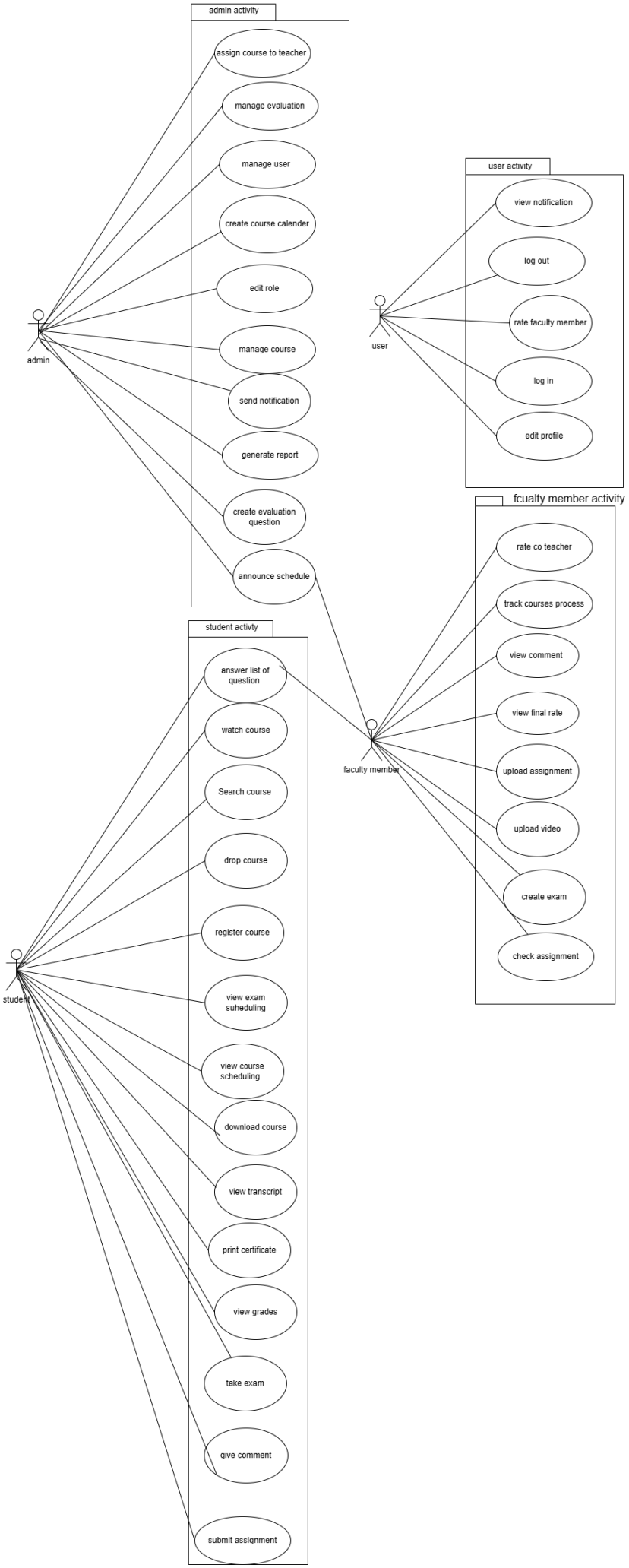
create course calendar description

ID Number – Name:	Make course create.
Goal:	Allow administrators or department heads to assign time slots, instructors, and rooms for each course for the upcoming term.
Preconditions:	1. The list of courses, instructors, rooms, and student capacities is available. 2. Scheduling permissions are granted to the user.
Postconditions:	1. The course schedule is created.. 2. Students and faculty can view their assigned timetables.
Initiator(s):	Admin.
Main Success Scenario:	1. Admin logs into the system. 2. Navigates to the "Course Scheduling" module. 3. Assigns instructors, rooms, and time slots to each course. 4. System checks for instructor/time/room conflicts. 5. Finalized schedule is saved .
Alternative Scenarios:	instructor Conflict: Instructor is assigned two overlapping sessions; system displays a warning.

c) Package Diagram grouping relevant Use Cases into Packages.

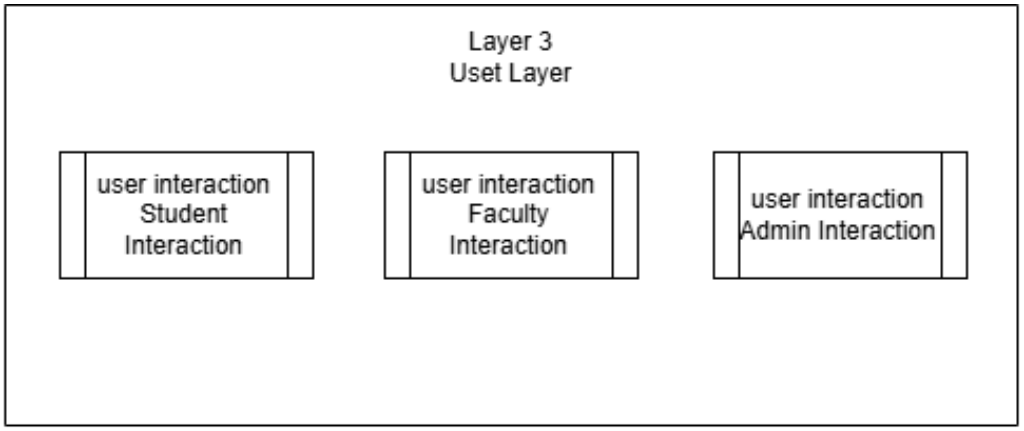


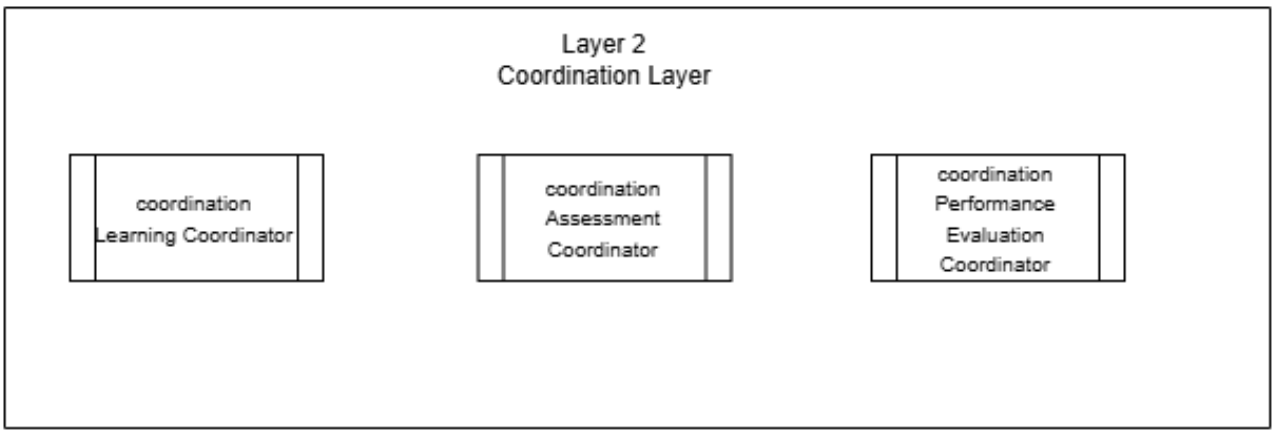

reporting
generate report
admin
admin management
manage evaluation
manage user
edit role
manage course
admin
user activity
view notification
log out
rate faculty member
log in
edit profile
user
exam &evolution
take exam
view grades
print certificate
submit assignment
view transcript
answer list of question
check assignment
create exam
announce schedule
create evaluation question
student
faculty member
admin
course management
download course
watch course
Search course
drop course
register course
upload assignment
upload video
track courses process
create course calender
assign course to teacher
student
faculty membre
admin
feedback &Commnication
give comment
view comment
view final rate
rate co teacher
send notification
student
faculty member
admin
scheduling
view exam suheduling
view course scheduling
student

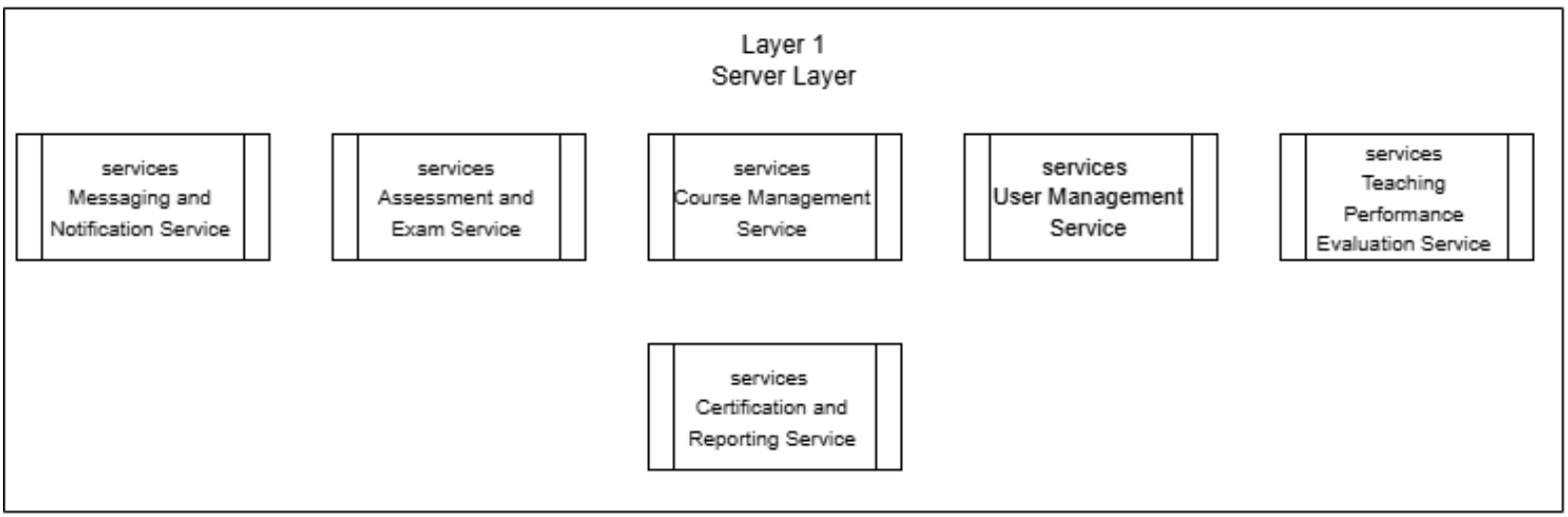

admin activity
assign course to teacher
manage evaluation
manage user
create course calender
edit role
manage course
send notification
generate report
create evaluation question
announce schedule
admin
user activity
view notification
log out
rate faculty member
log in
edit profile
user
fcualty member activity
rate co teacher
track courses process
view comment
view final rate
upload assignment
upload video
create exam
check assignment
faculty member
student activty
answer list of question
watch course
Search course
drop course
register course
view exam suheduling
view course scheduling
download course
view transcript
print certificate
view grades
take exam
give comment
submit assignment
student

9) Structural & Behavioural Diagrams:

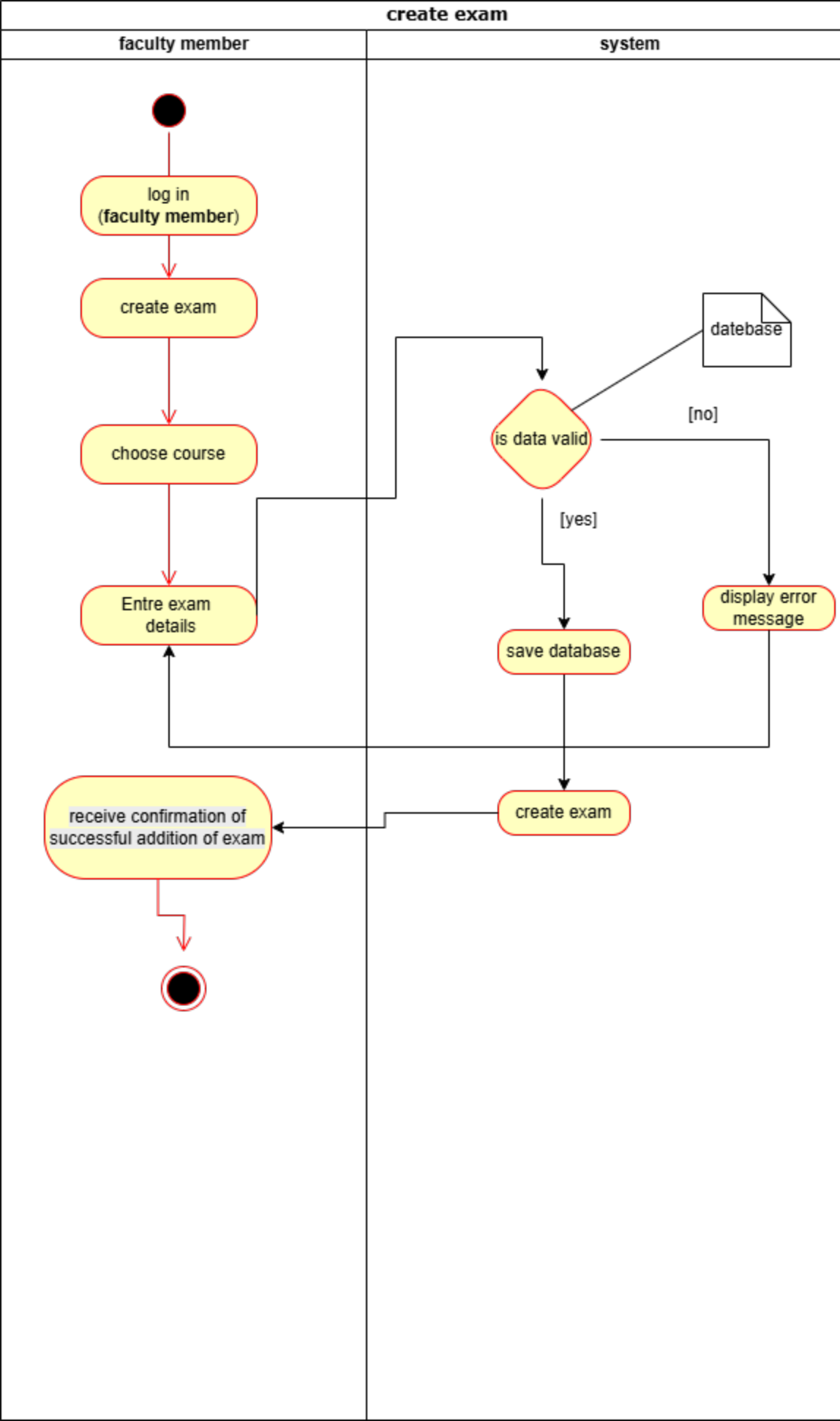
a) System Architecture

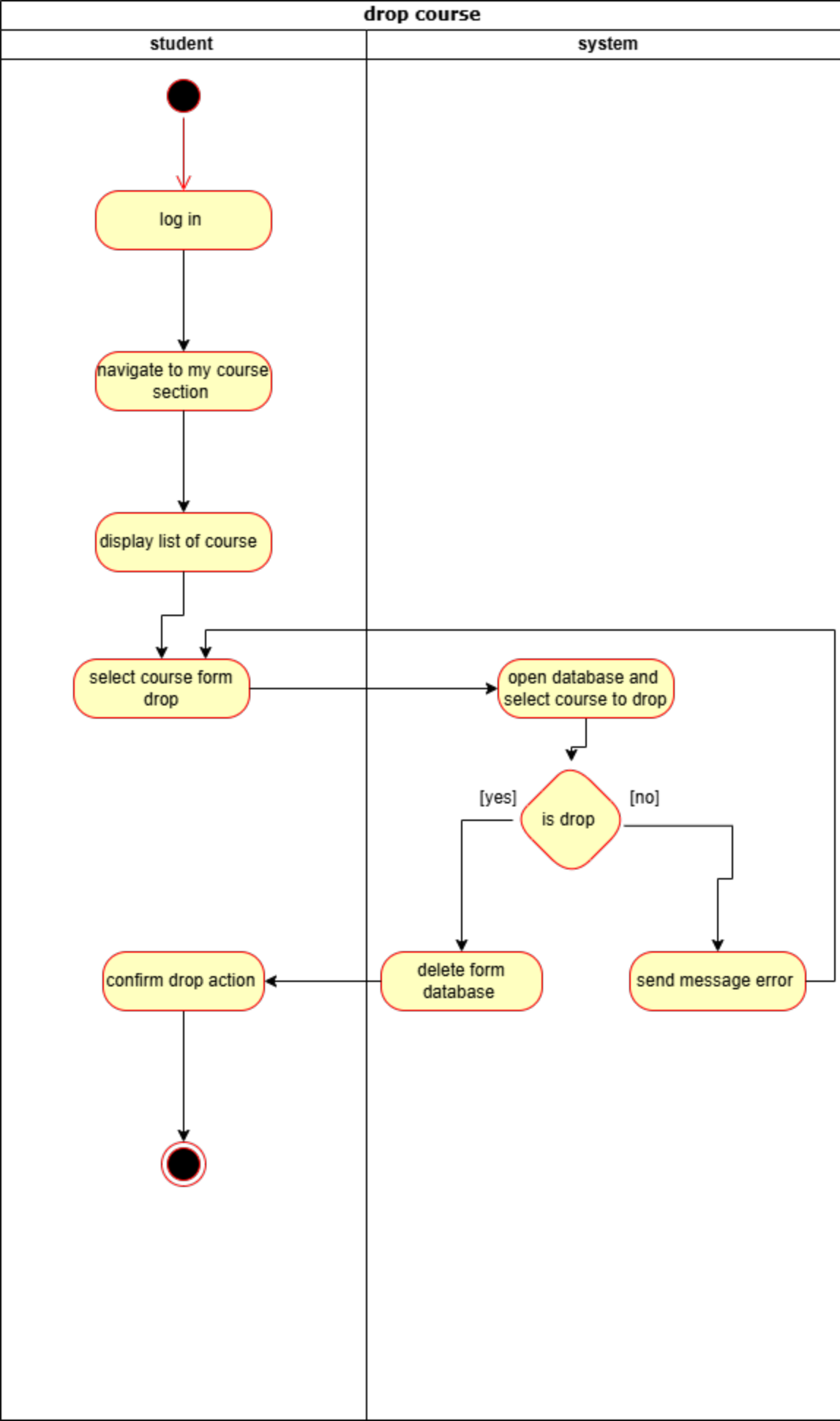


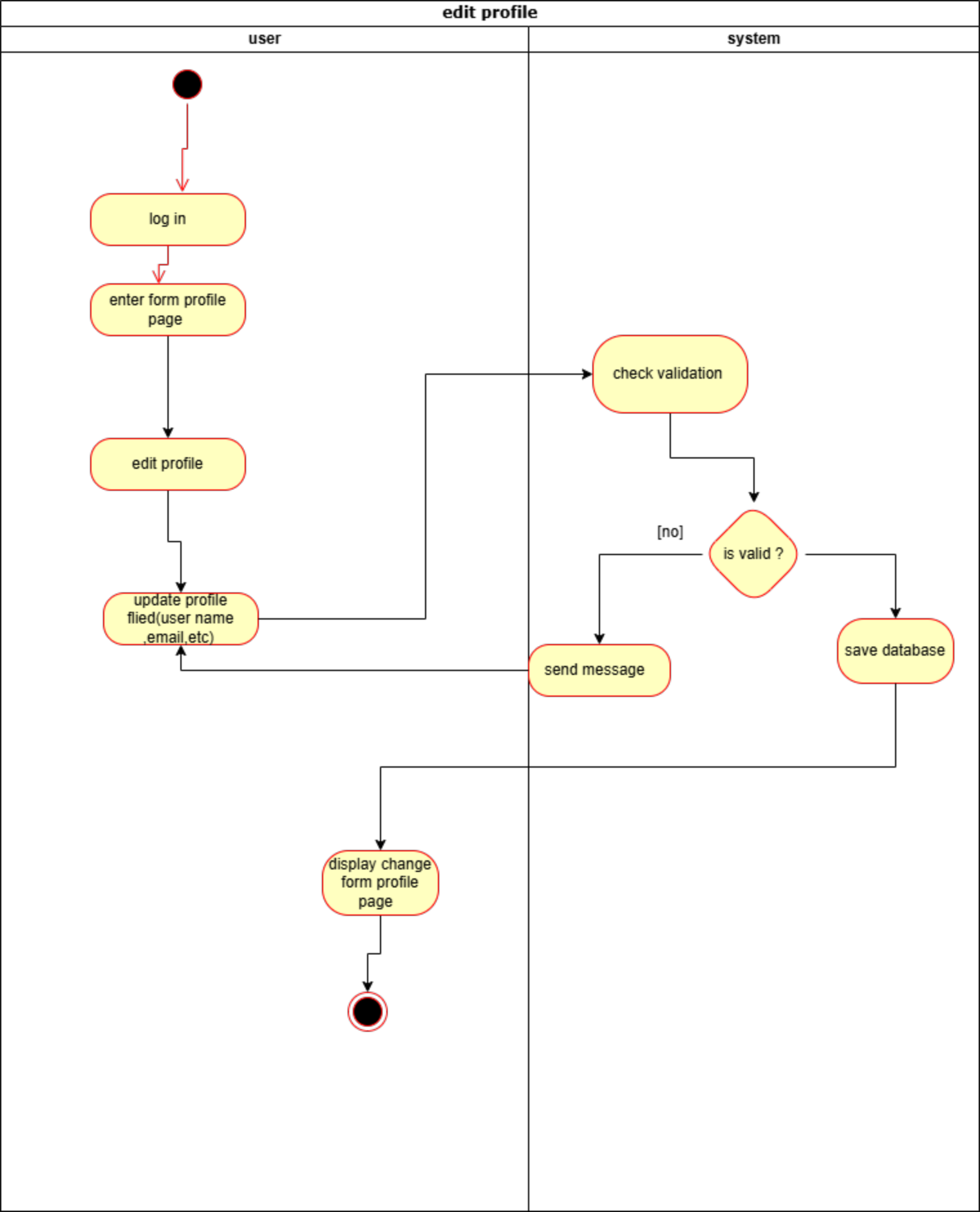




b) Activity Diagrams

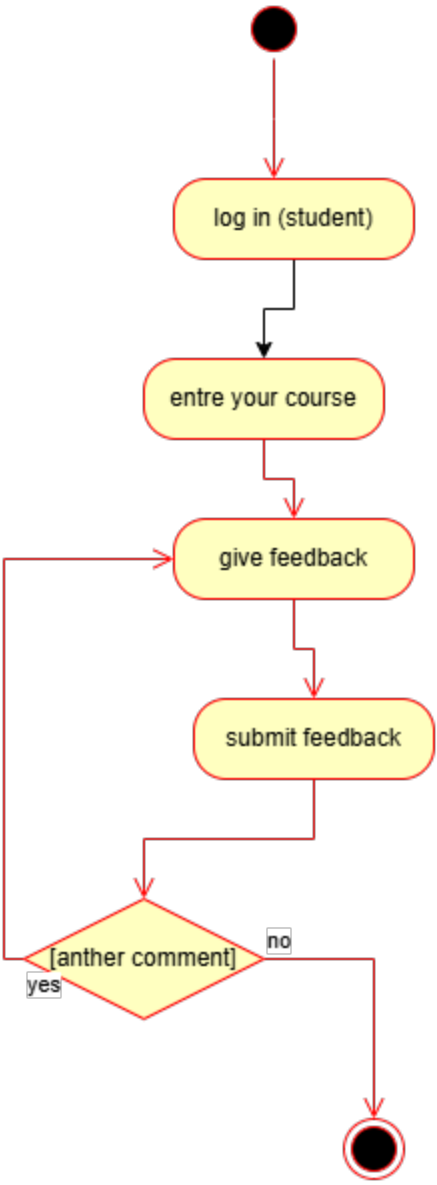

create exam
faculty member
system
log in
(faculty member)
create exam
choose course
Entre exam
details
datebase
is data valid
[no]
[yes]
display error
message
save database
create exam
receive confirmation of
successful addition of exam

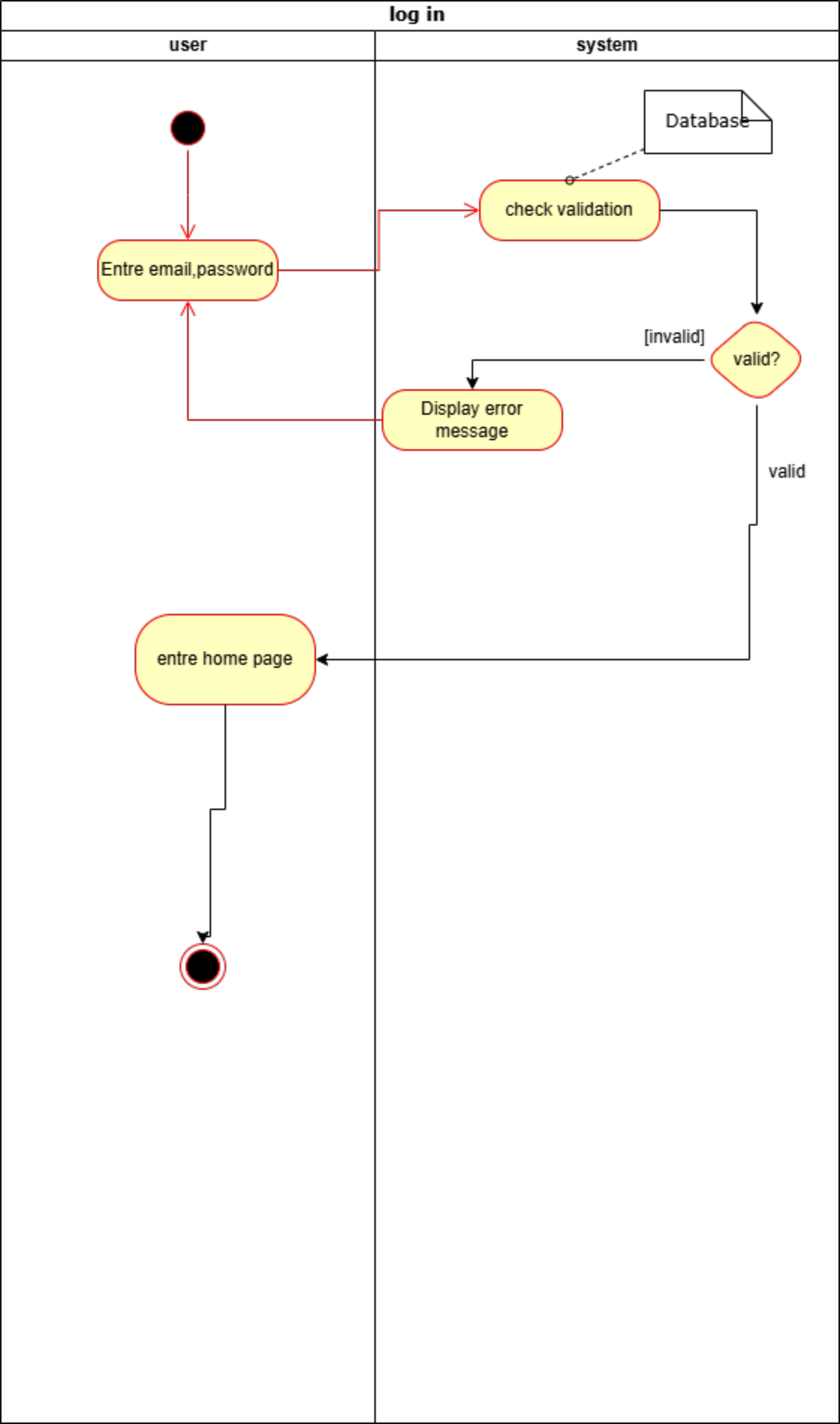

drop course
student
system
log in
navigate to my course section
display list of course
select course form drop
open database and select course to drop
is drop
[yes]
[no]
confirm drop action
delete form database
send message error

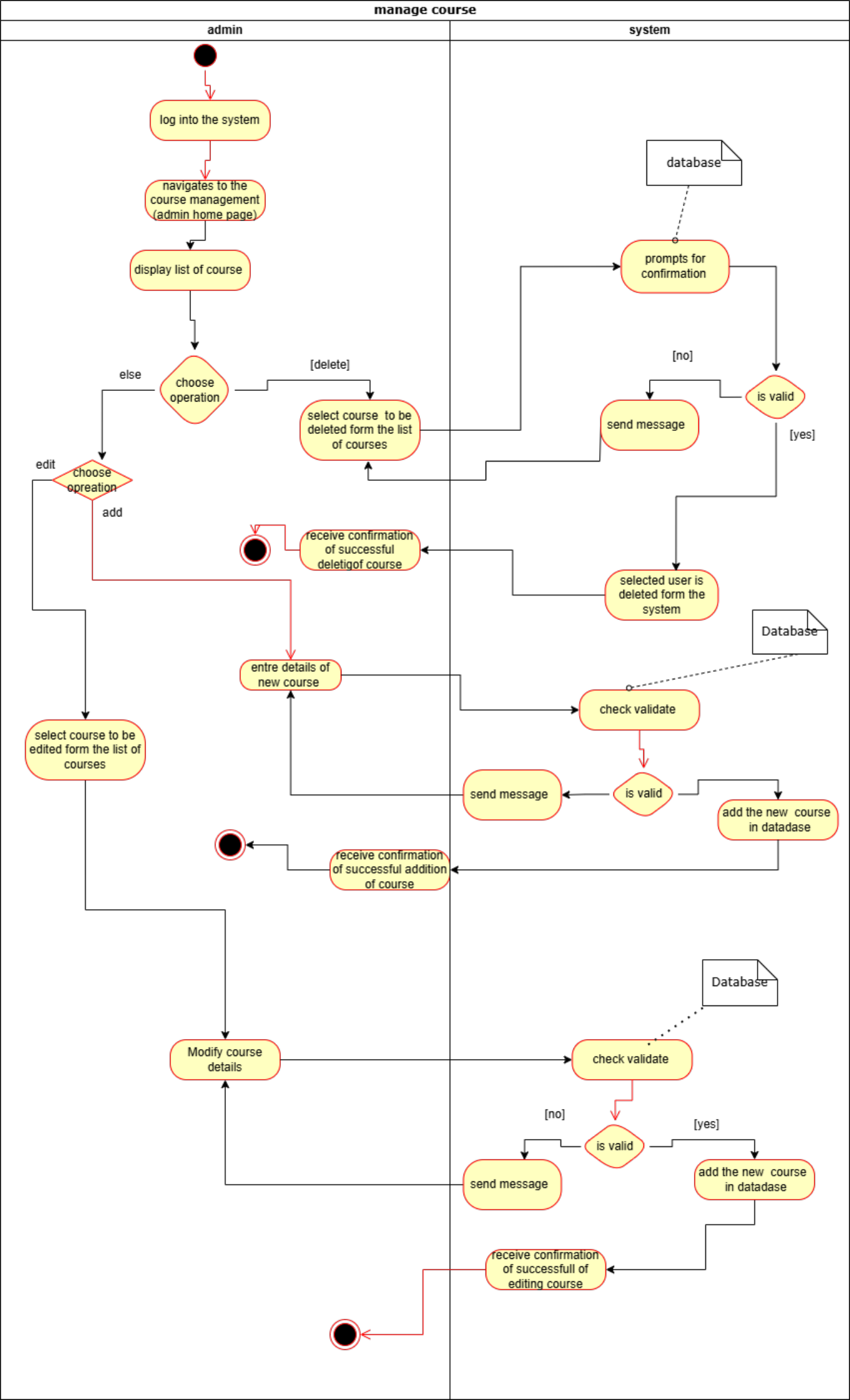

edit profile
user
system
log in
enter form profile page
edit profile
update profile flied(user name ,email,etc)
check validation
[no]
is valid ?
send message
save database
display change form profile page

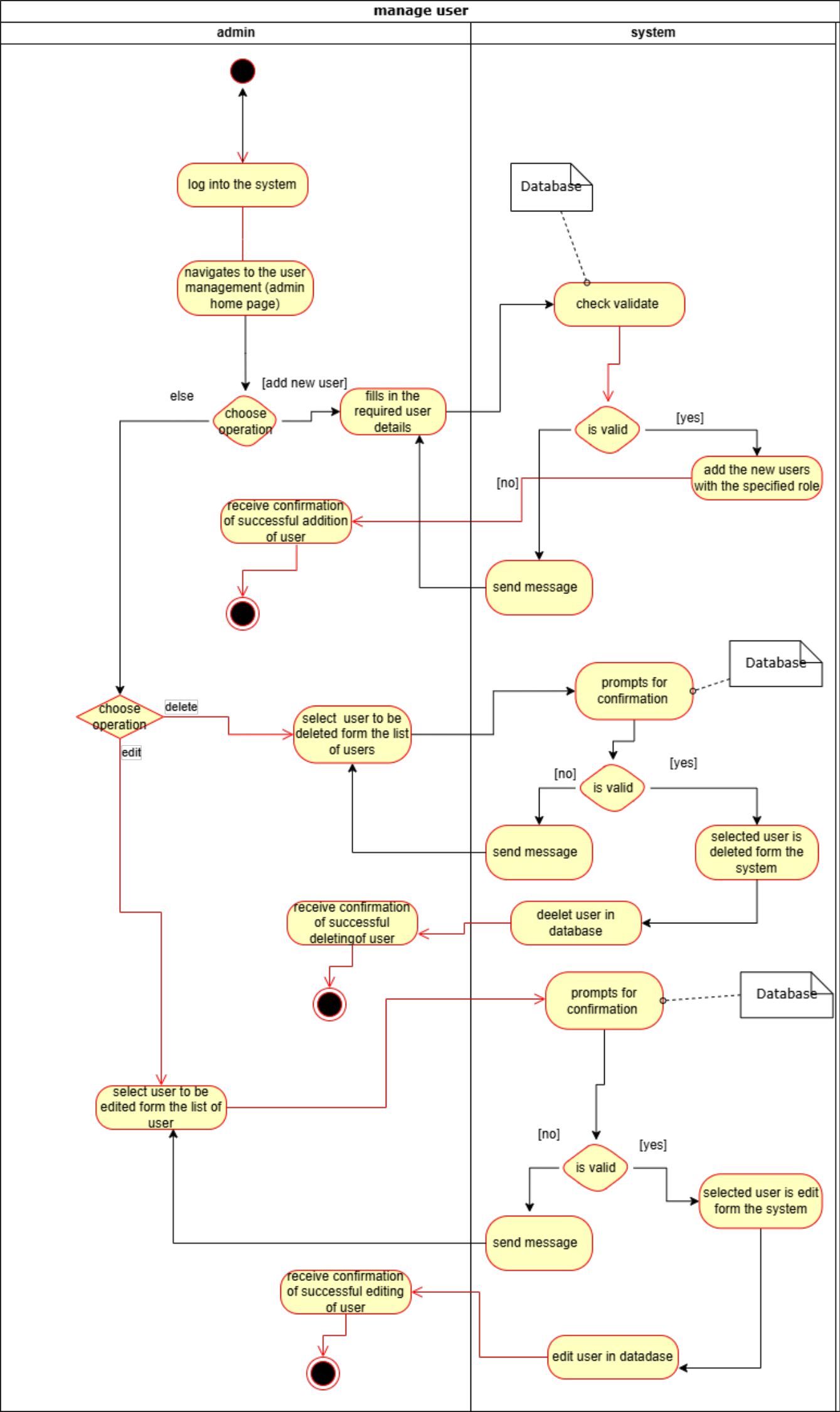
submit feedback

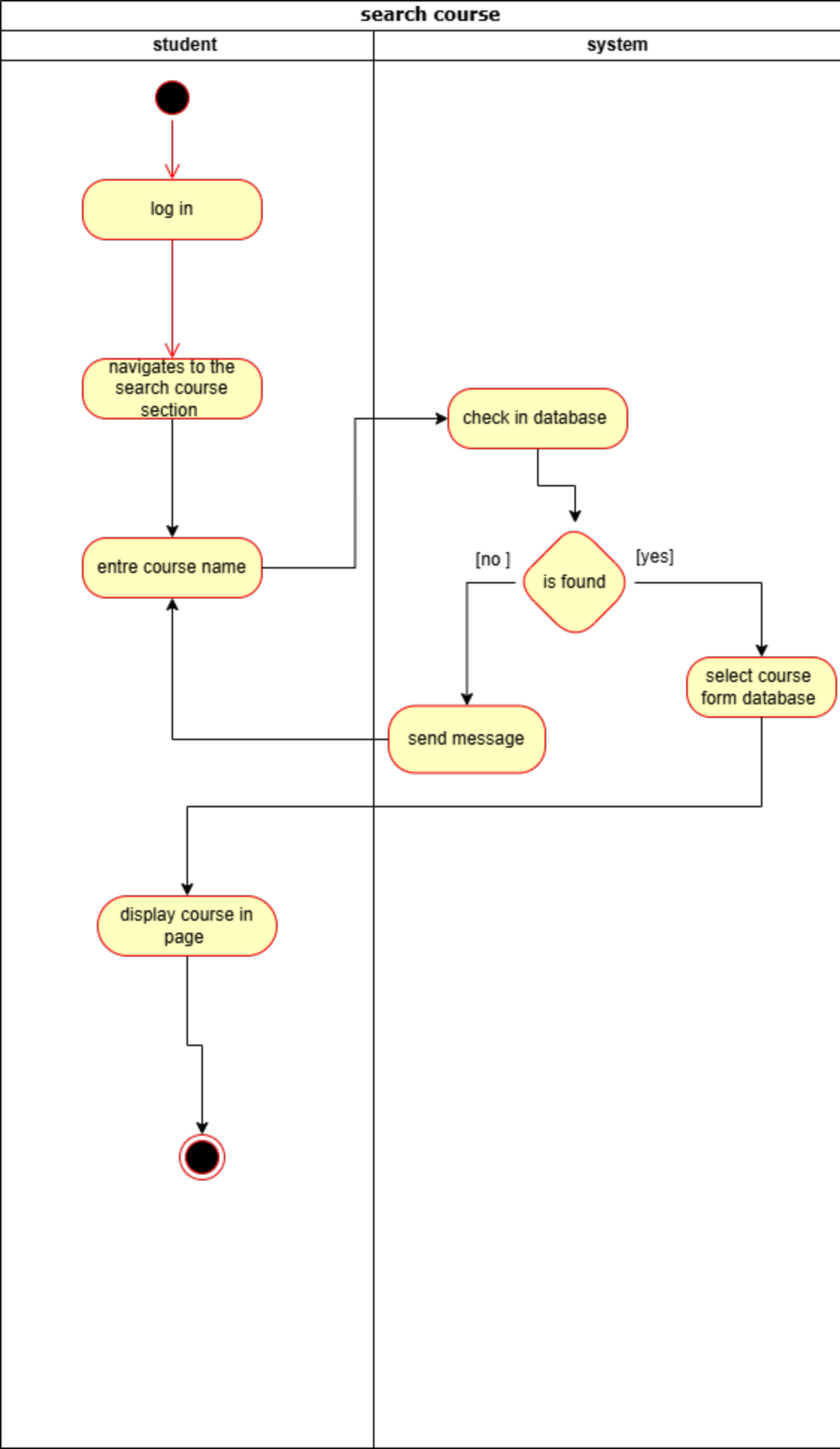
user (student)




log in
user
system
Database
check validation
Entre email,password
[invalid]
valid?
Display error message
valid
entre home page


manage course
admin
system
log into the system
navigates to the course management (admin home page)
display list of course
choose operation
else
[delete]
edit
choose opreation
add
select course to be edited form the list of courses
select course to be deleted form the list of courses
database
prompts for confirmation
[no]
is valid
send message
[yes]
receive confirmation of successful deletigof course
selected user is deleted form the system
entre details of new course
Database
check validate
is valid
send message
add the new course in datadase
receive confirmation of successful addition of course
Modify course details
Database
check validate
[no]
is valid
[yes]
send message
add the new course in datadase
receive confirmation of successfull of editing course


manage user
admin
system
log into the system
navigates to the user management (admin home page)
choose operation
[add new user]
else
fills in the required user details
Database
check validate
is valid
[yes]
add the new users with the specified role
[no]
receive confirmation of successful addition of user
send message
choose operation
delete
edit
select user to be deleted form the list of users
prompts for confirmation
Database
[no]
is valid
[yes]
send message
selected user is deleted form the system
receive confirmation of successful deletingof user
deelet user in database
prompts for confirmation
Database
select user to be edited form the list of user
[no]
is valid
[yes]
selected user is edit form the system
send message
receive confirmation of successful editing of user
edit user in datadase

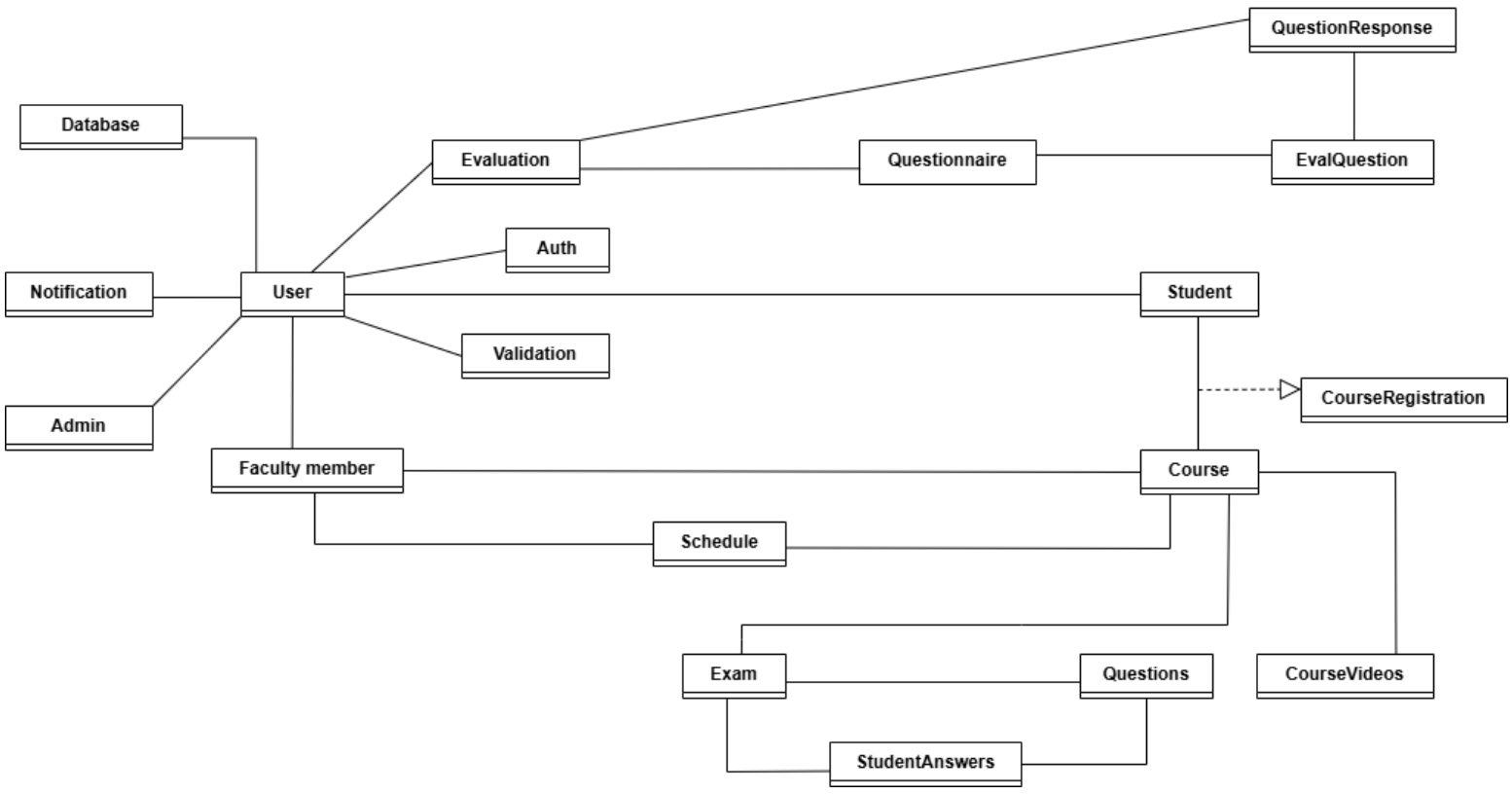

search course
student
system
log in
navigates to the search course section
entre course name
check in database
is found
[no]
[yes]
send message
select course form database
display course in page

c) Based on the Activity Diagrams, the List of User Interfaces required for the System and the corresponding users of each interface.

User interface (use case)	Description	Corresponding users
Log in page (log in)	Provide secure for user logins	• Admin • Faculty member • Student
Manage course page (manage course)	Allow admin , faculty member to manage course by addition or deleting or edition them within system	• Admin • Faculty member
Manage user page (manage user)	Allow admin to manage users by addition or deleting them with different role within system	• Admin
Profile page (edit profile)	Allow user to edit profile by edit user information	• Admin • Faculty member • Student
Course page (create exam)	To allow faculty member to create exam for their courses	• Faculty member
Rate my instructor section (give comment)	Allow student to rate teacher and leave a comment based on their experience in course	• Student
My course page (drop course)	Allow student to remove a previously course from their	• Student

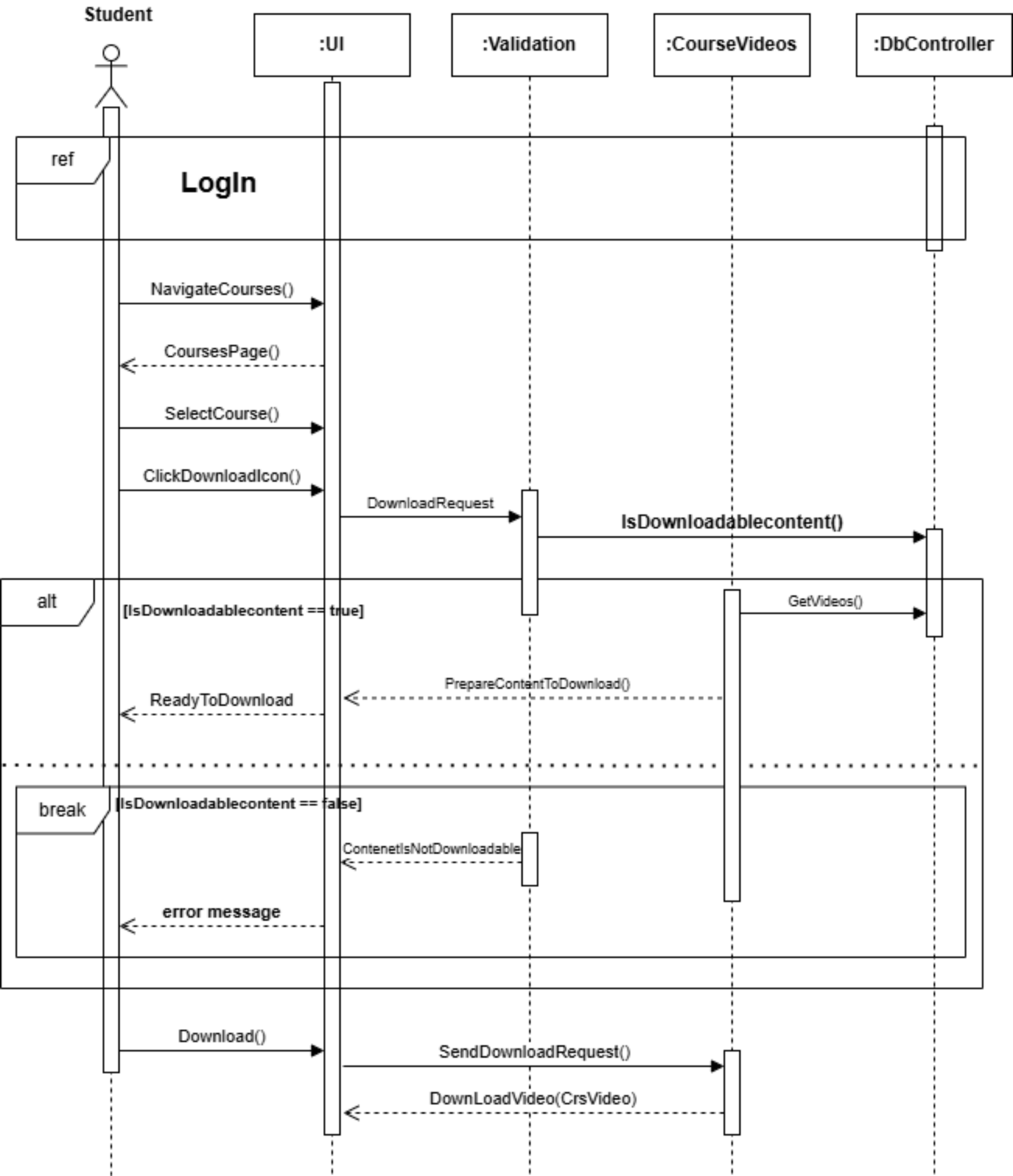
	schedule within the allowed drop period	
Search course section (search course)	Allow student to search for available online course such as name course , department ,instructor	• Student

d) Class Diagram 1: An initial version based on the requirements and Use Case/Activity diagrams

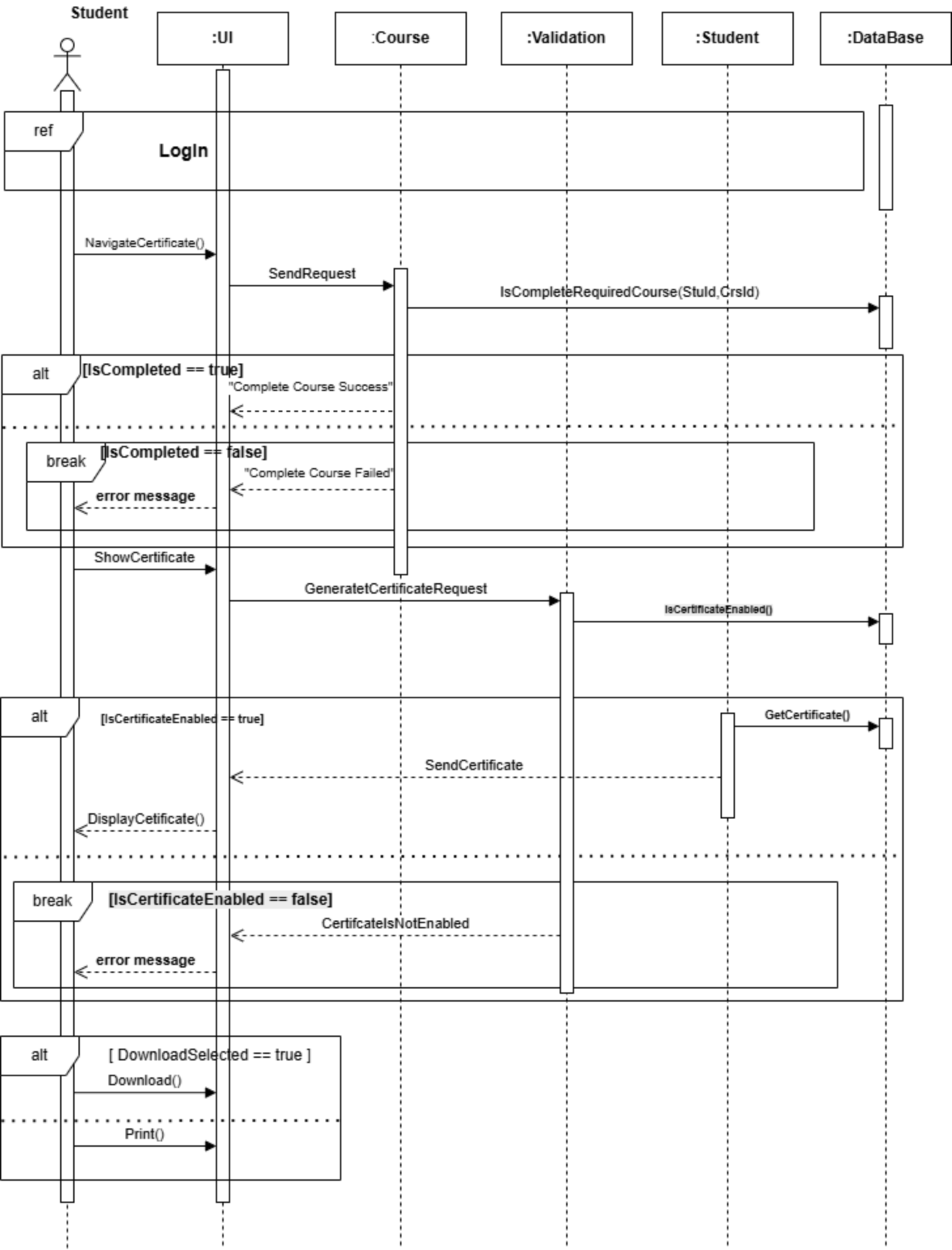


e) Sequence Diagram(s)

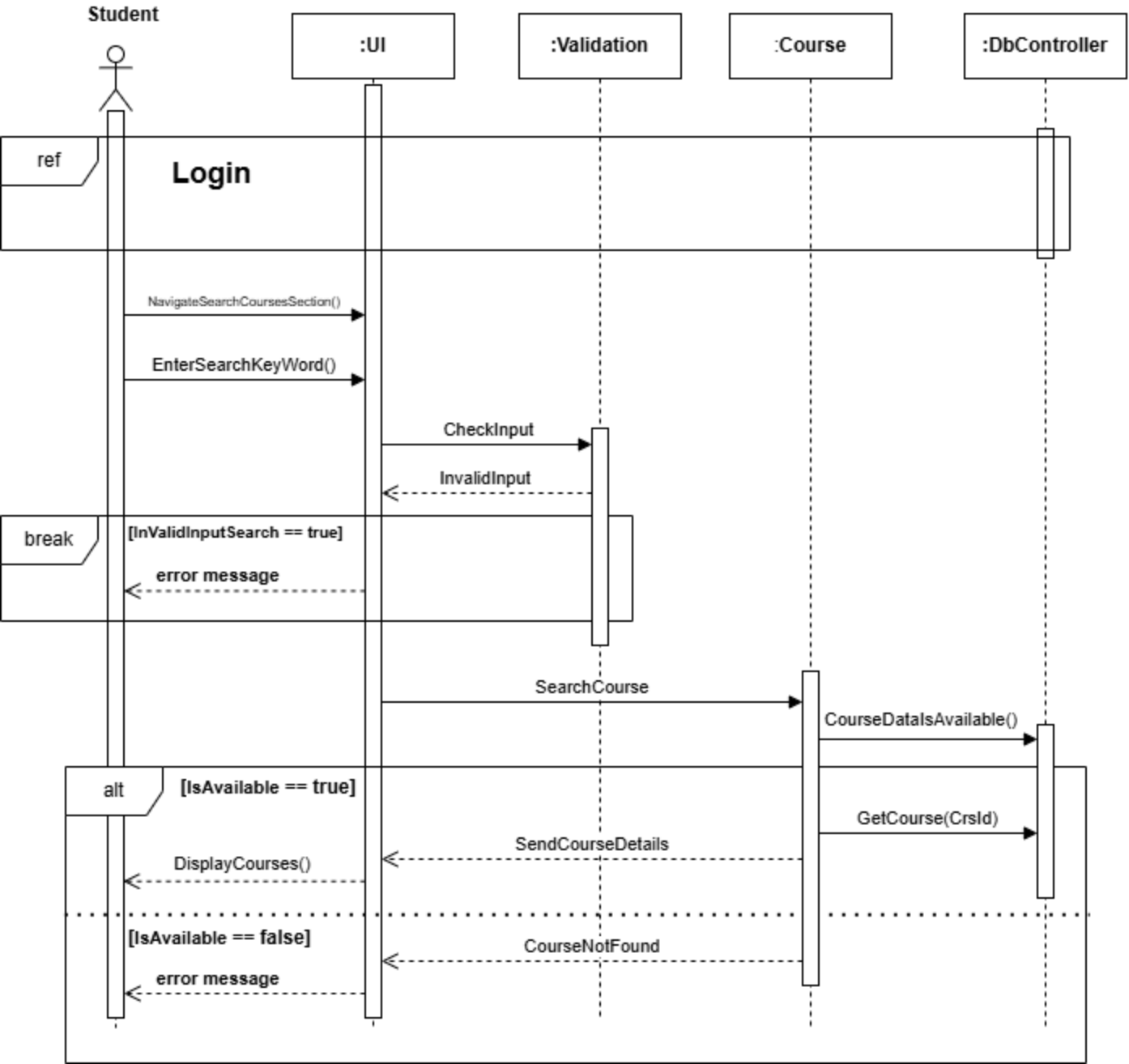
Download Online Course



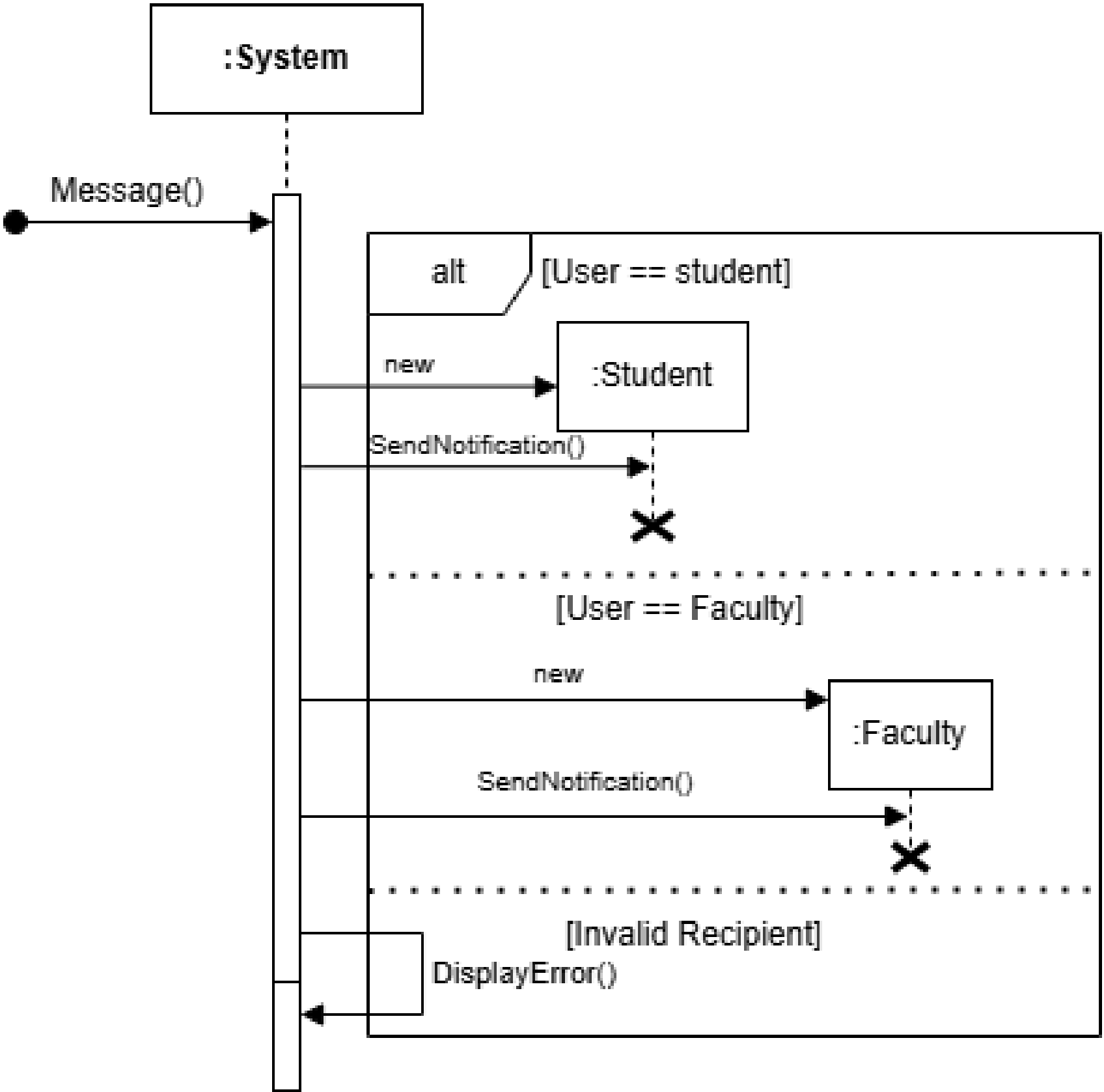
Print Certificate



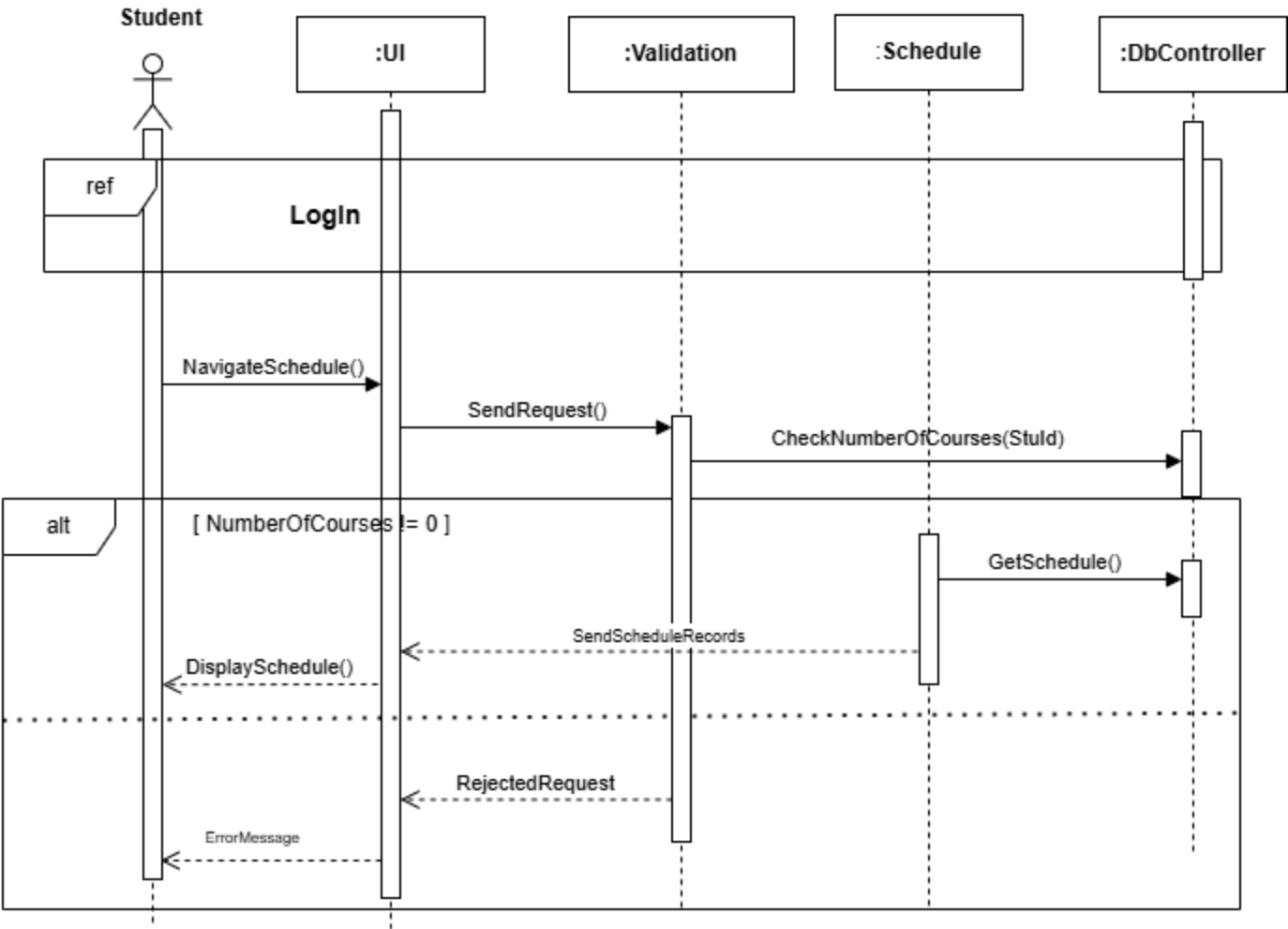
Search For Course



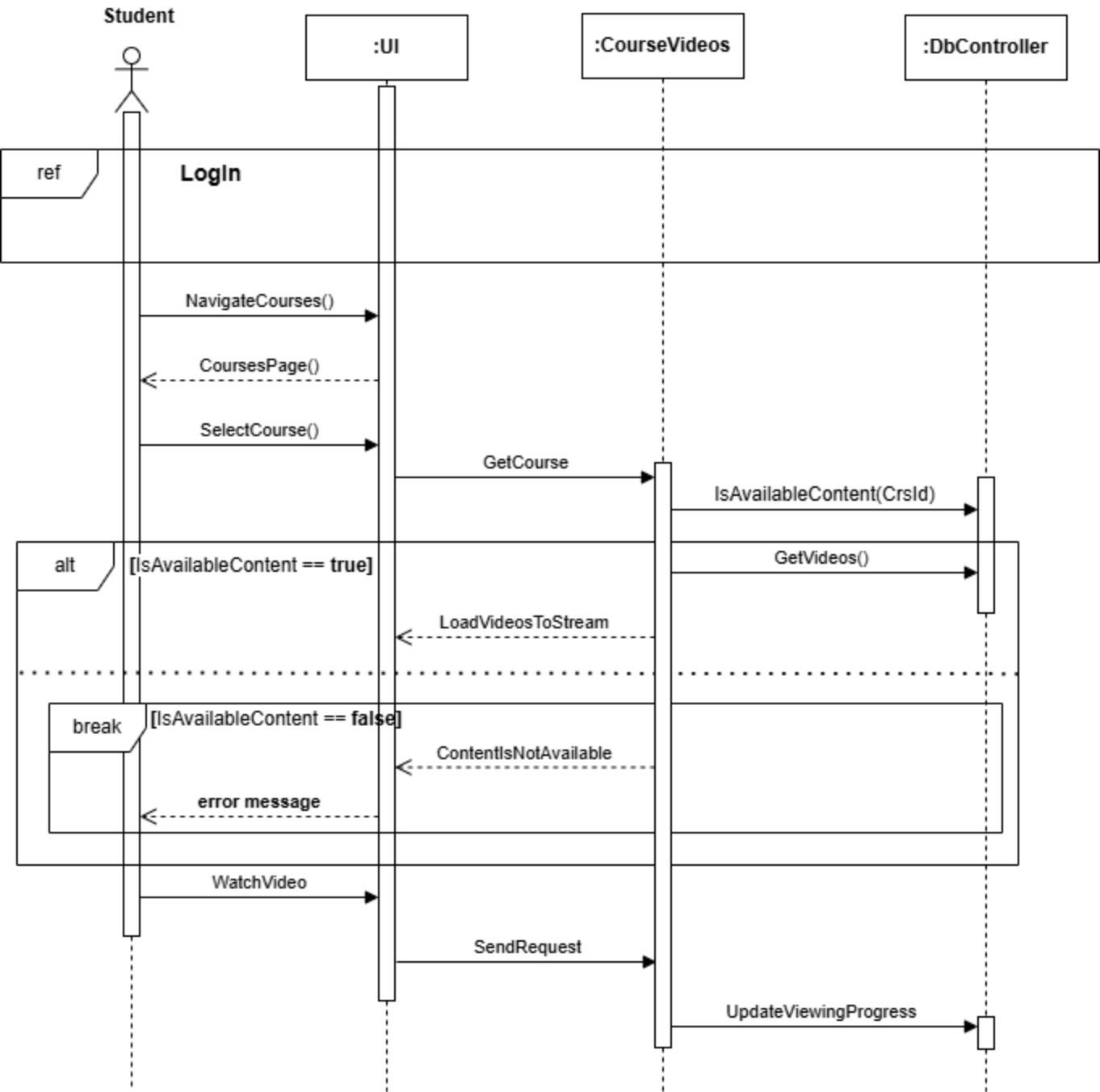
Send Notification



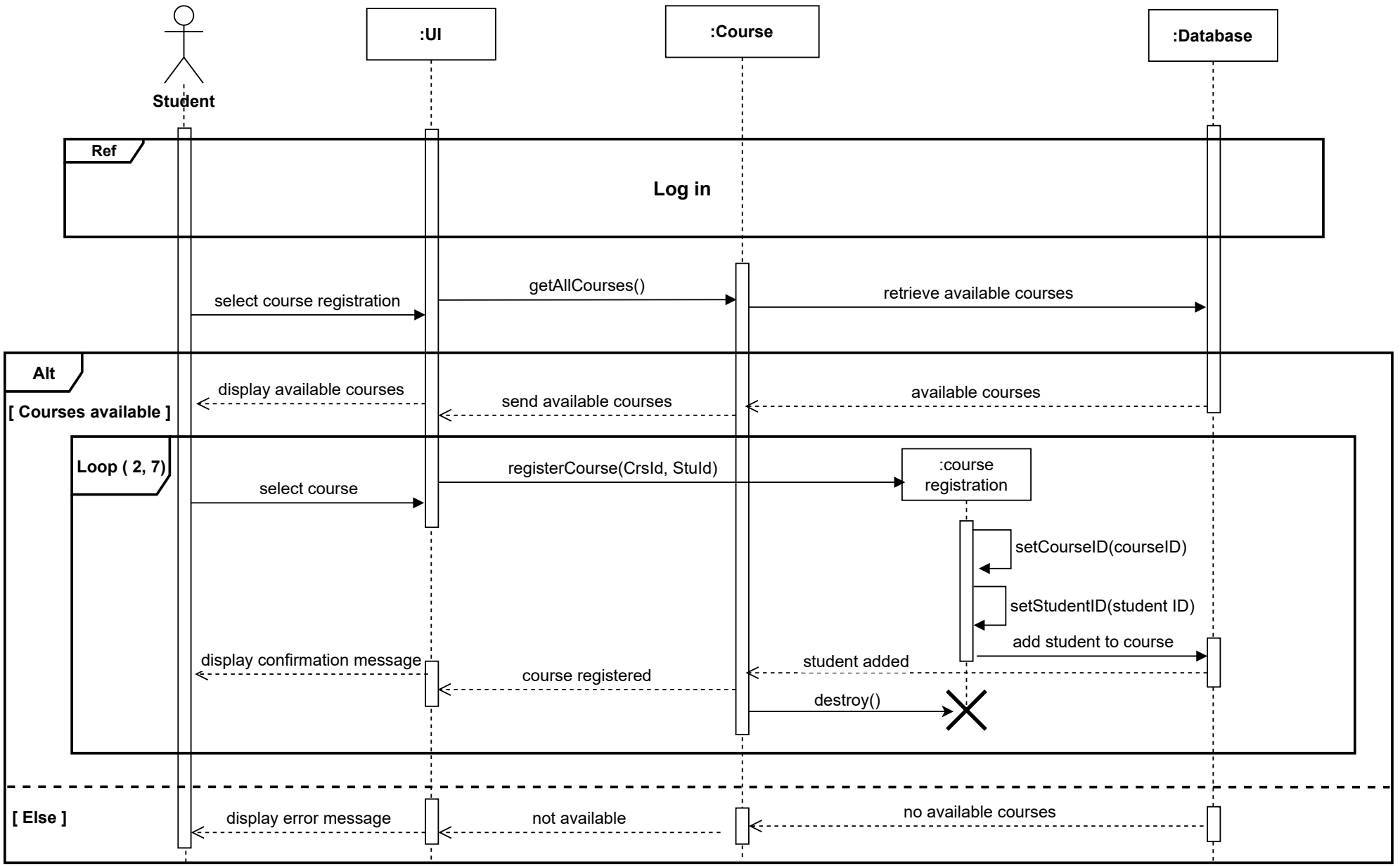
View Schedule Course



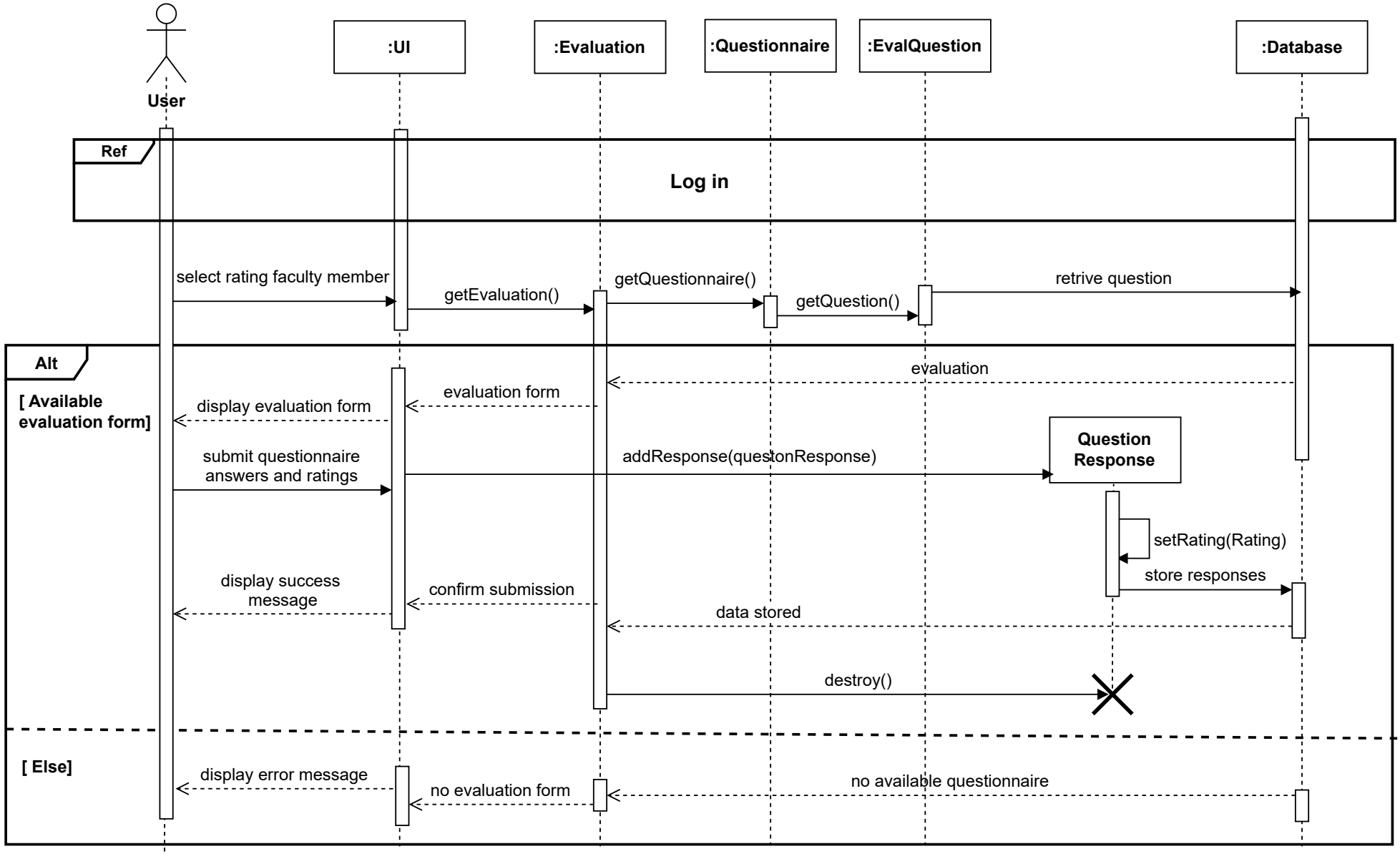
Watch Online Course



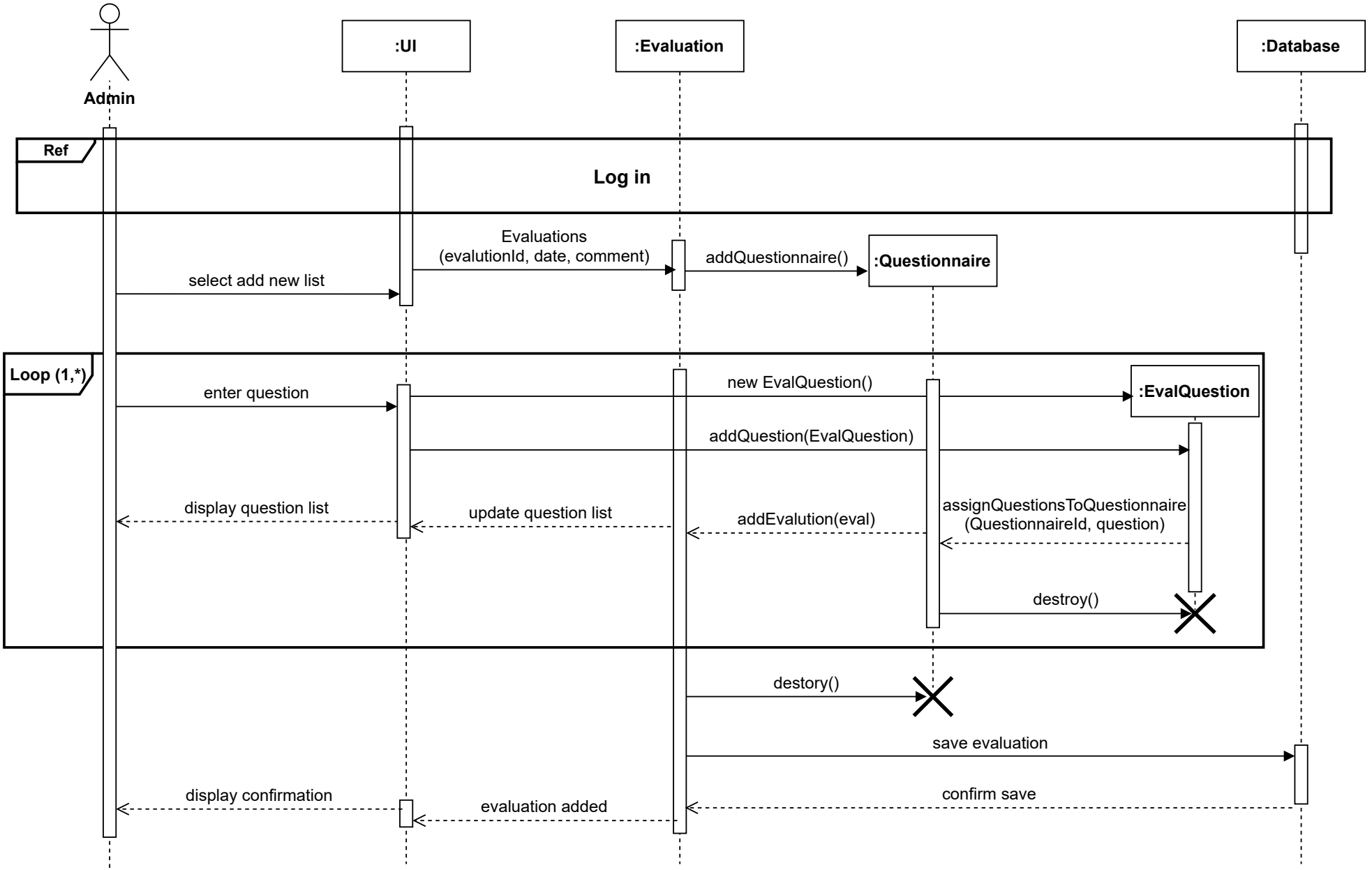
Register courses



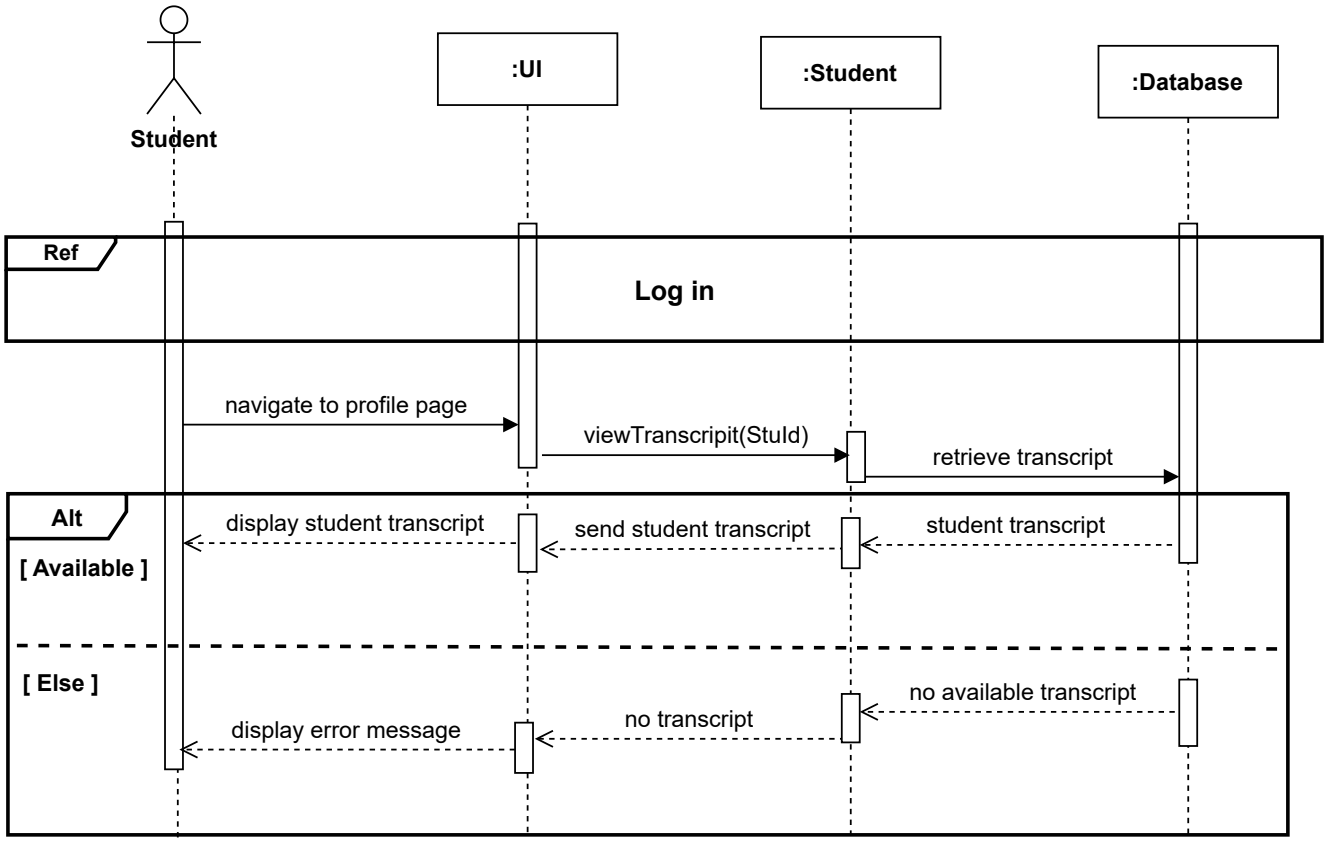
Rate faculty member



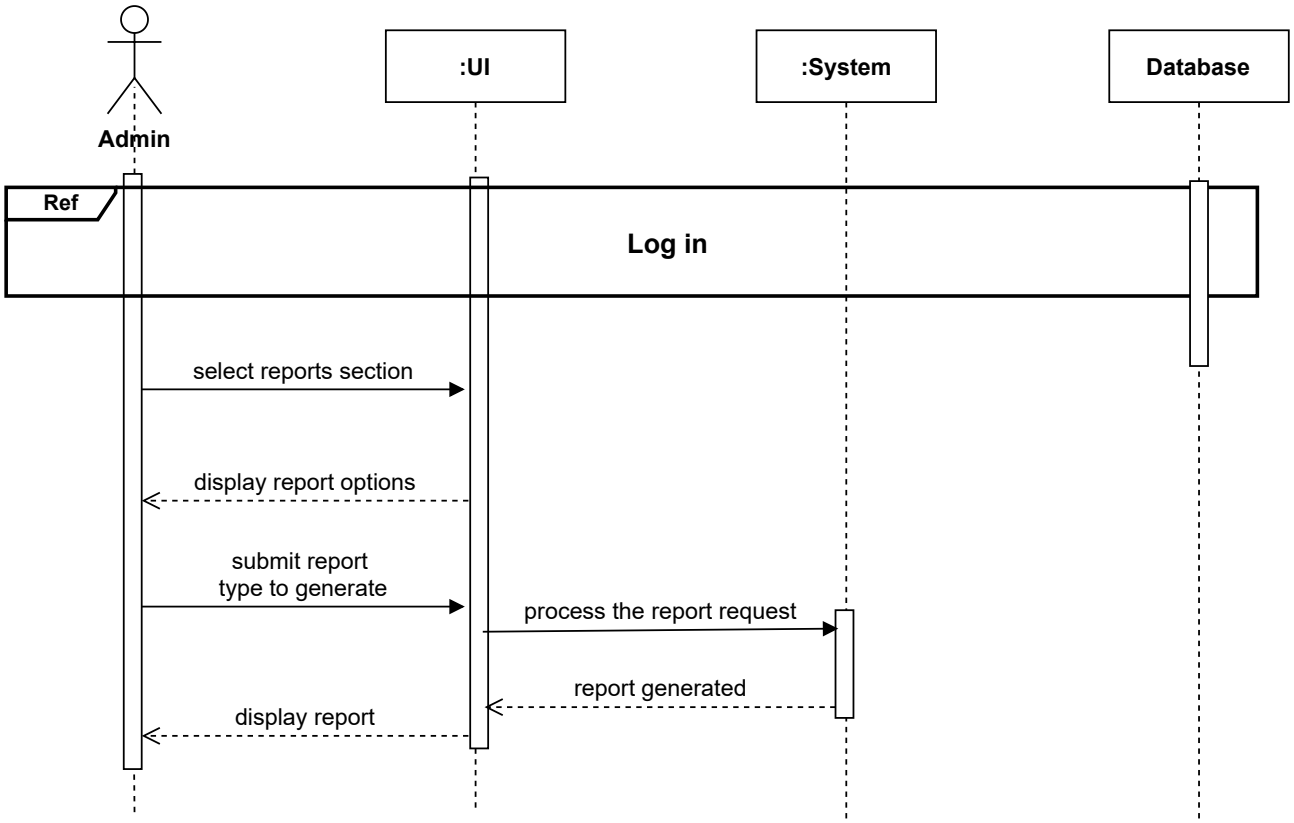
Create Evolutions Questions



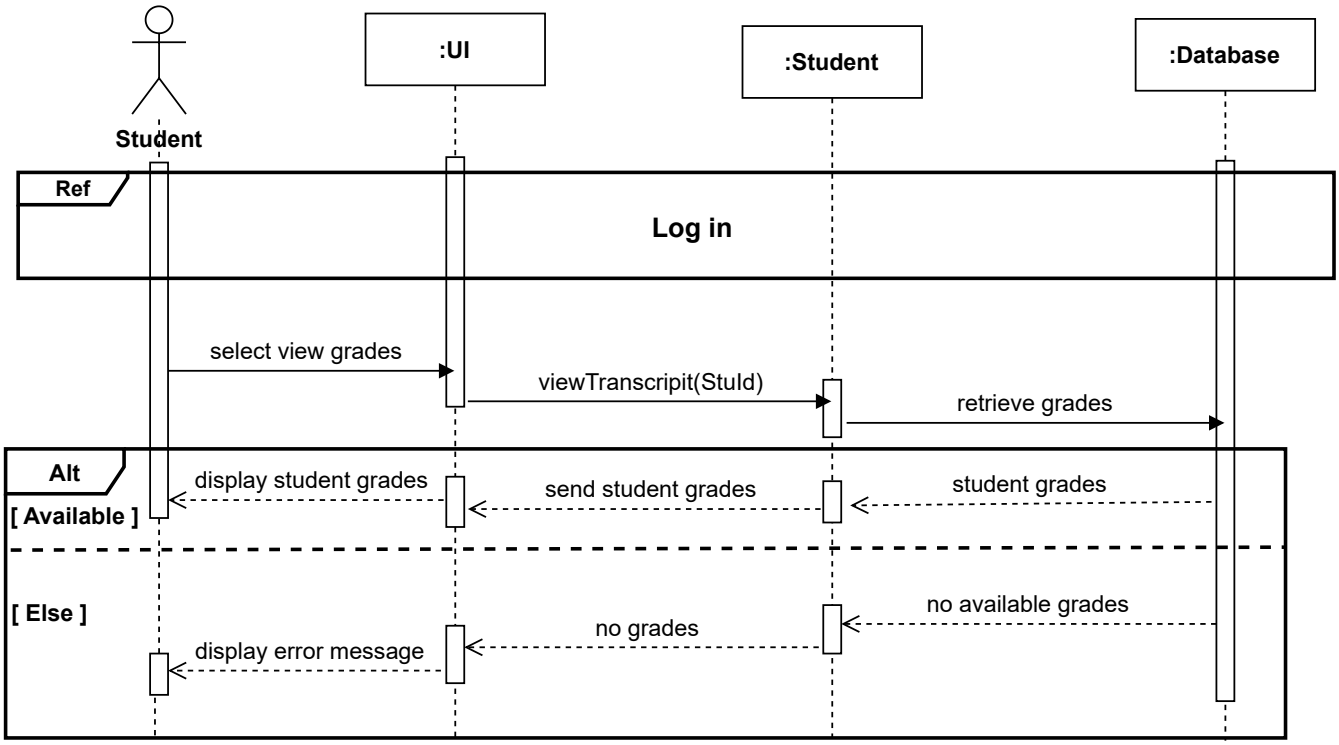
View transcript



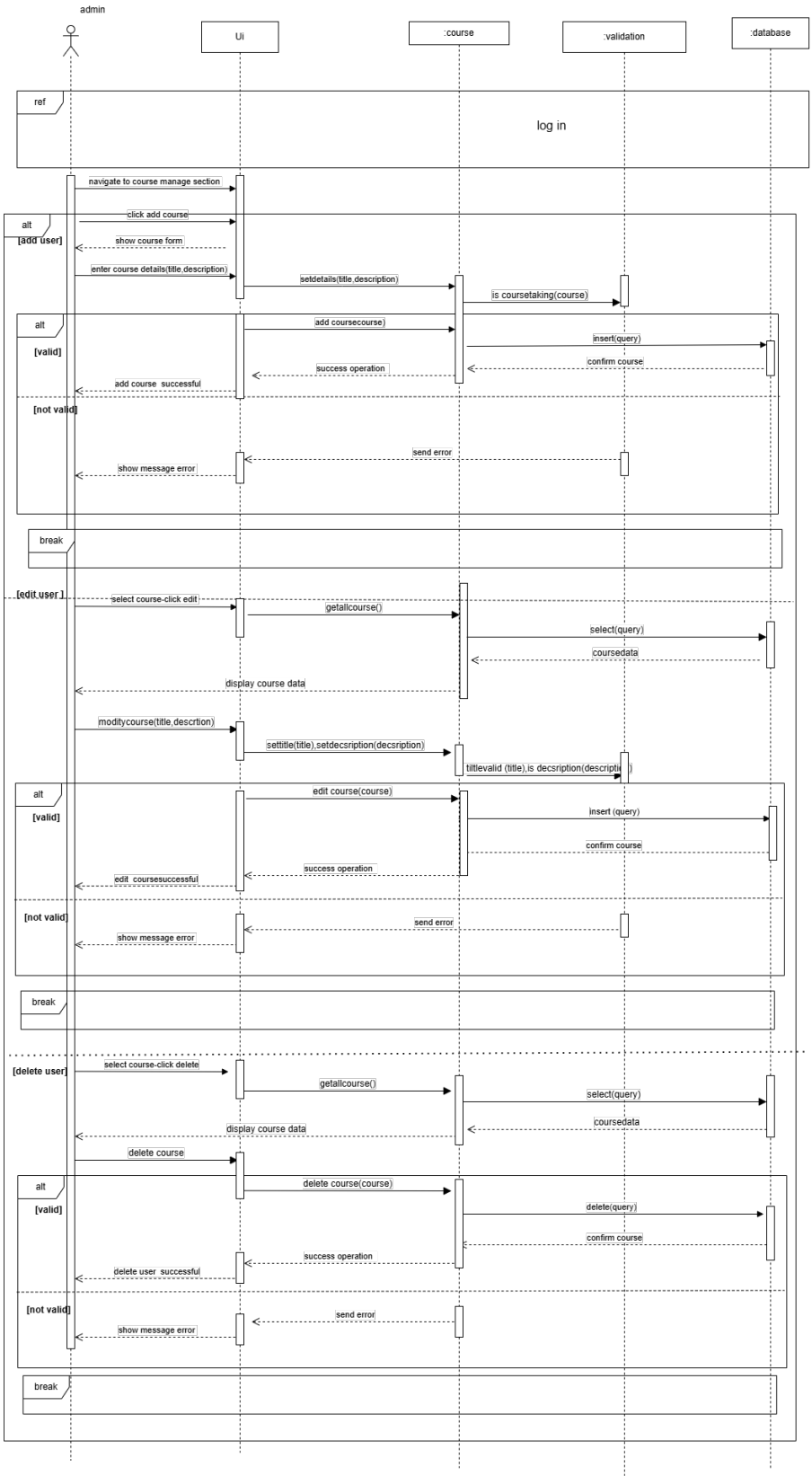
Generate reports



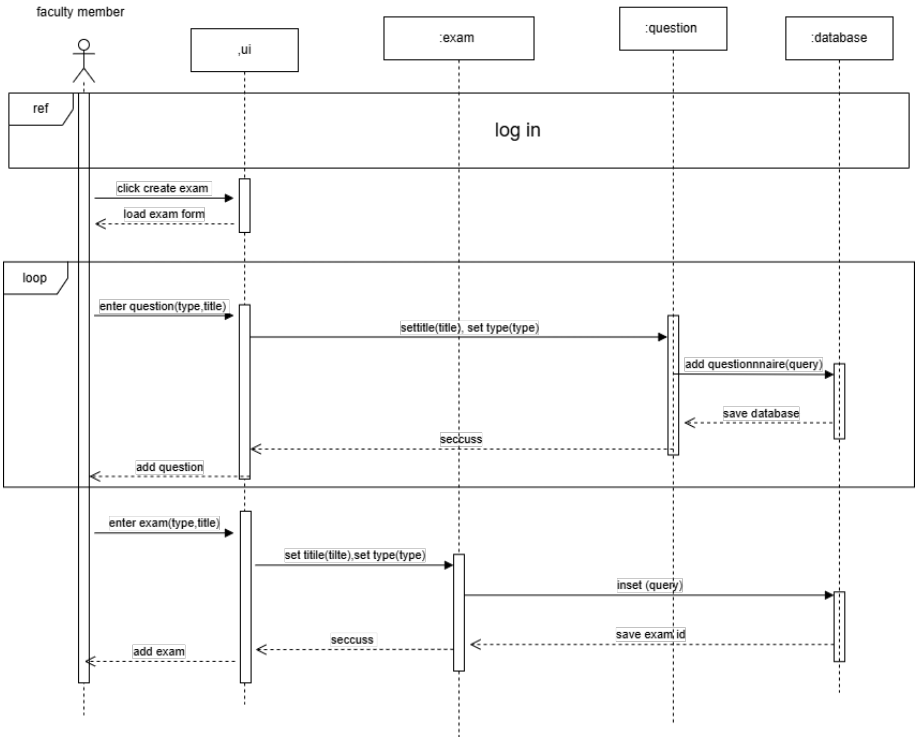
View grades


Student
:UI
:Student
:Database
Ref
Log in
select view grades
viewTranscripit(StuId)
retrieve grades
Alt
[Available]
display student grades
send student grades
student grades
[Else]
no available grades
no grades
display error message

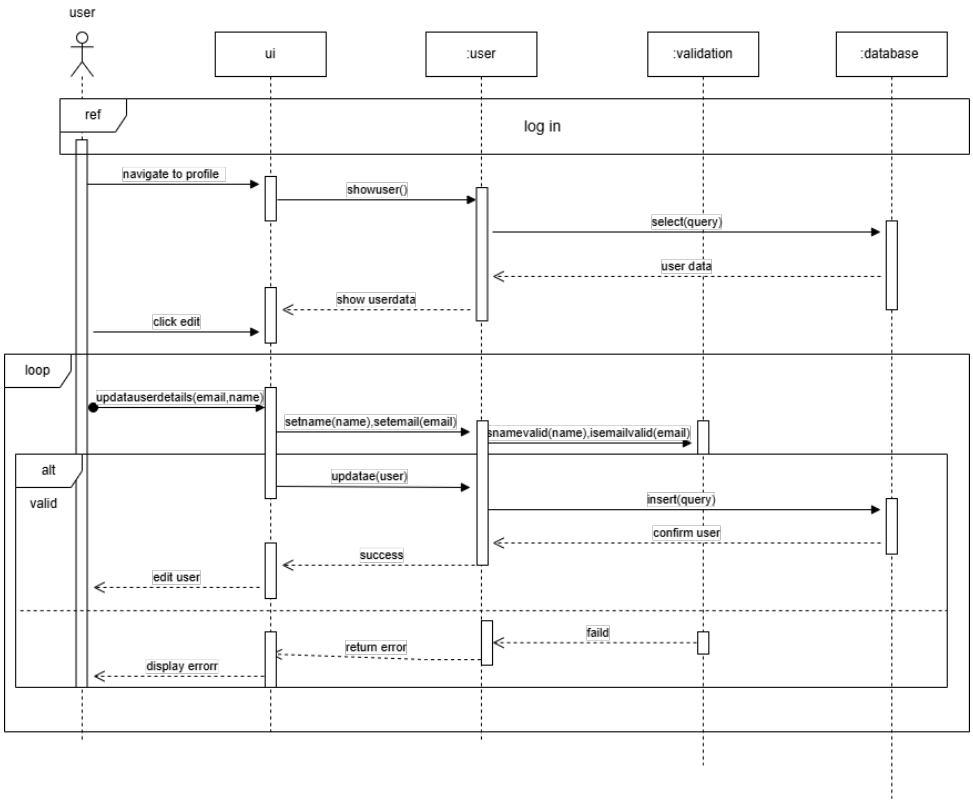
Course Management



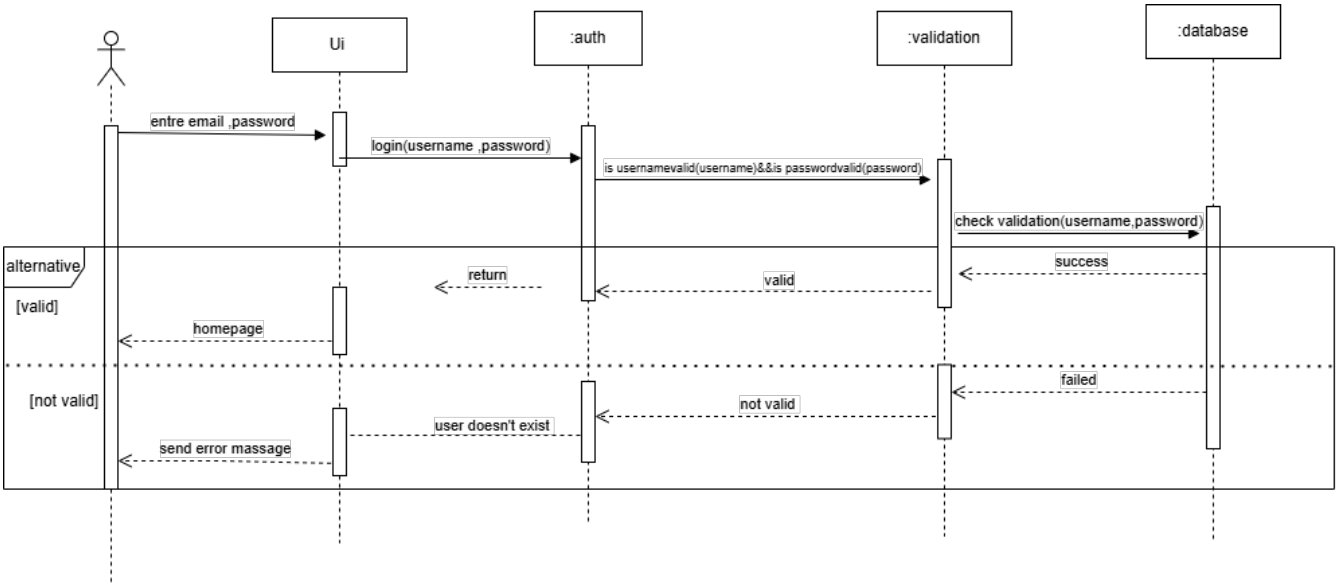
Create Exam



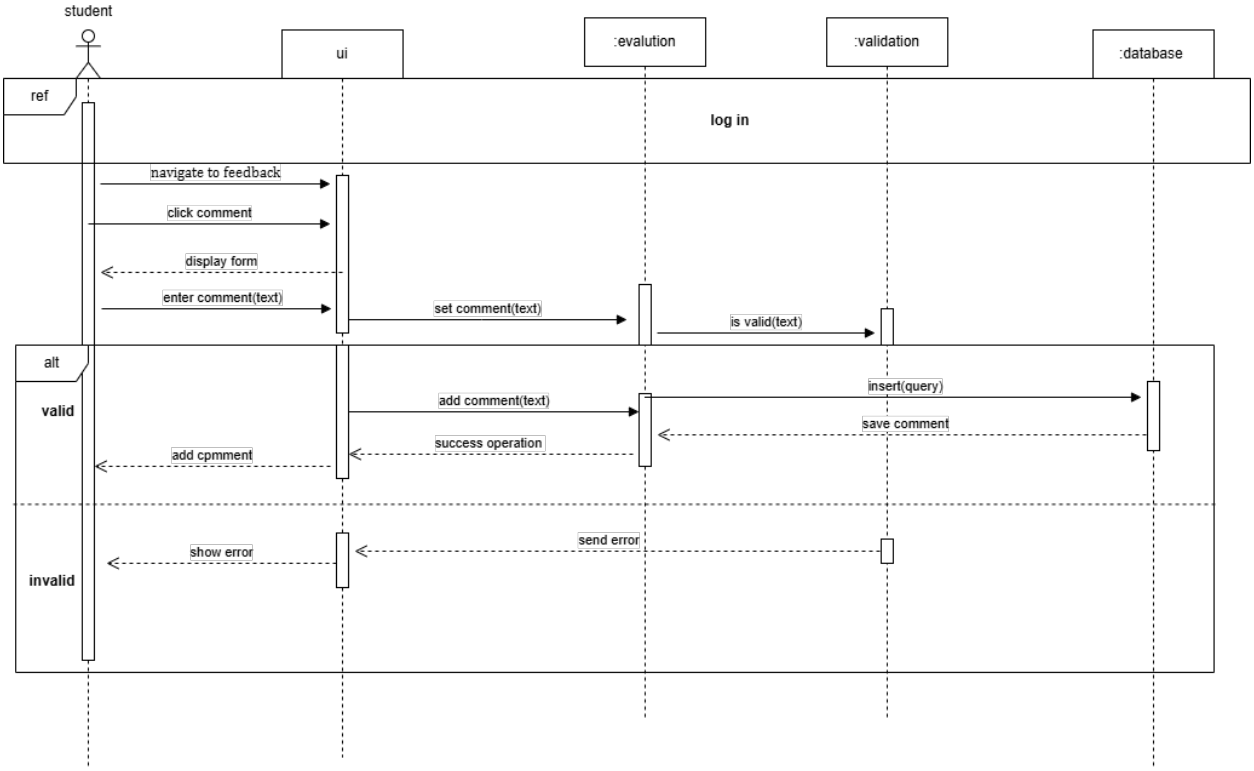
Edit Profile



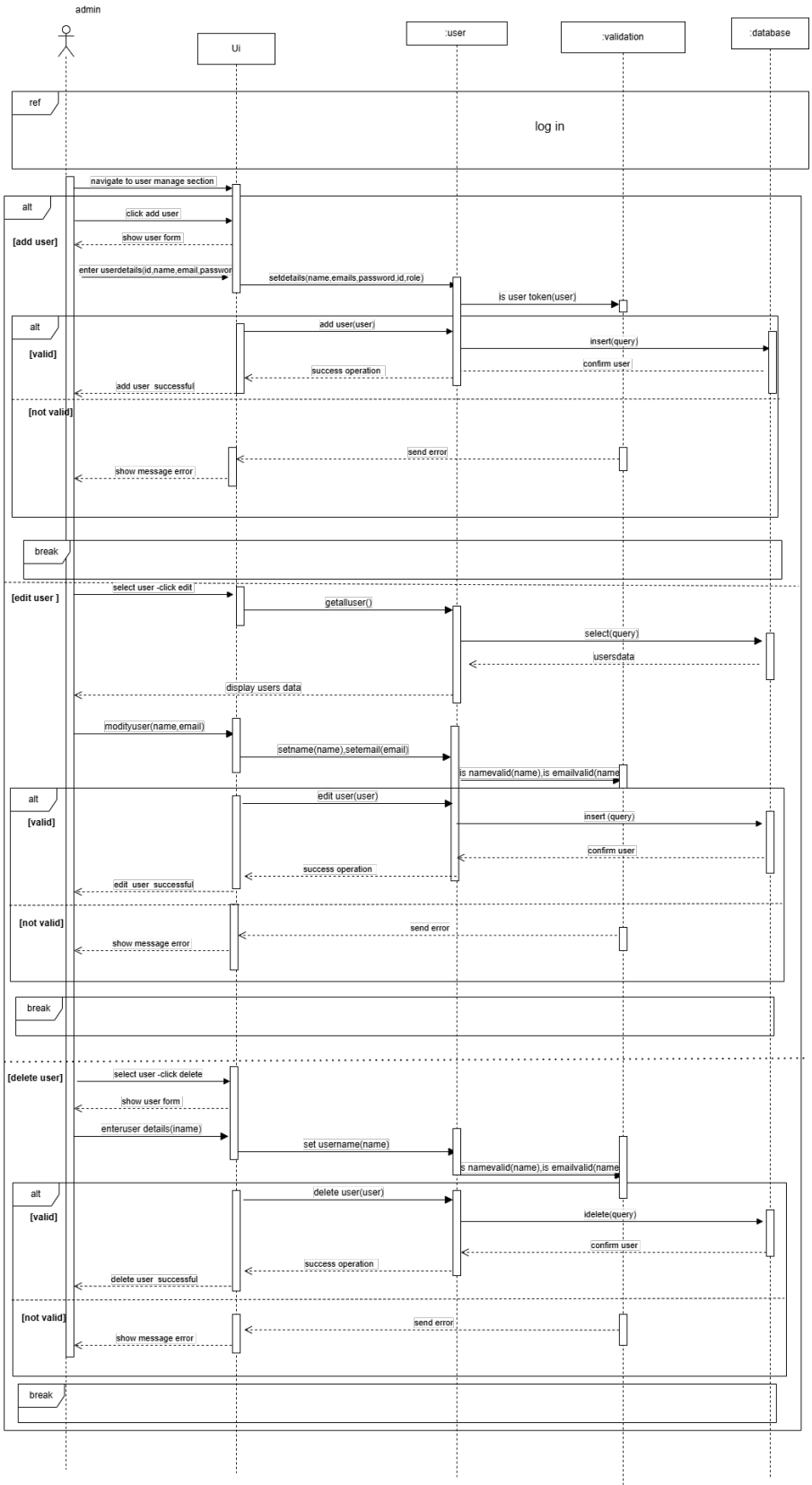
Log In



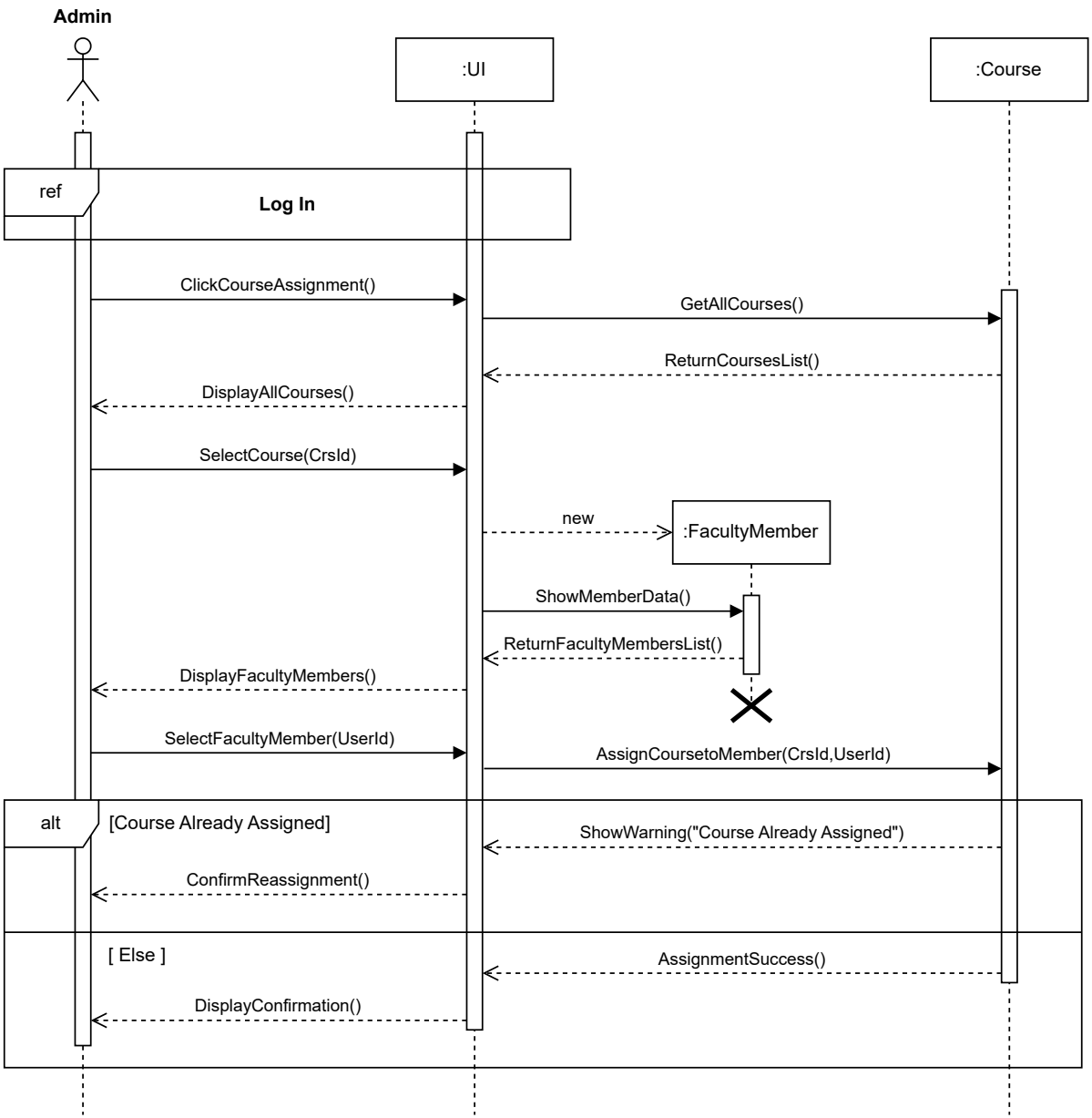
Submit Feedback



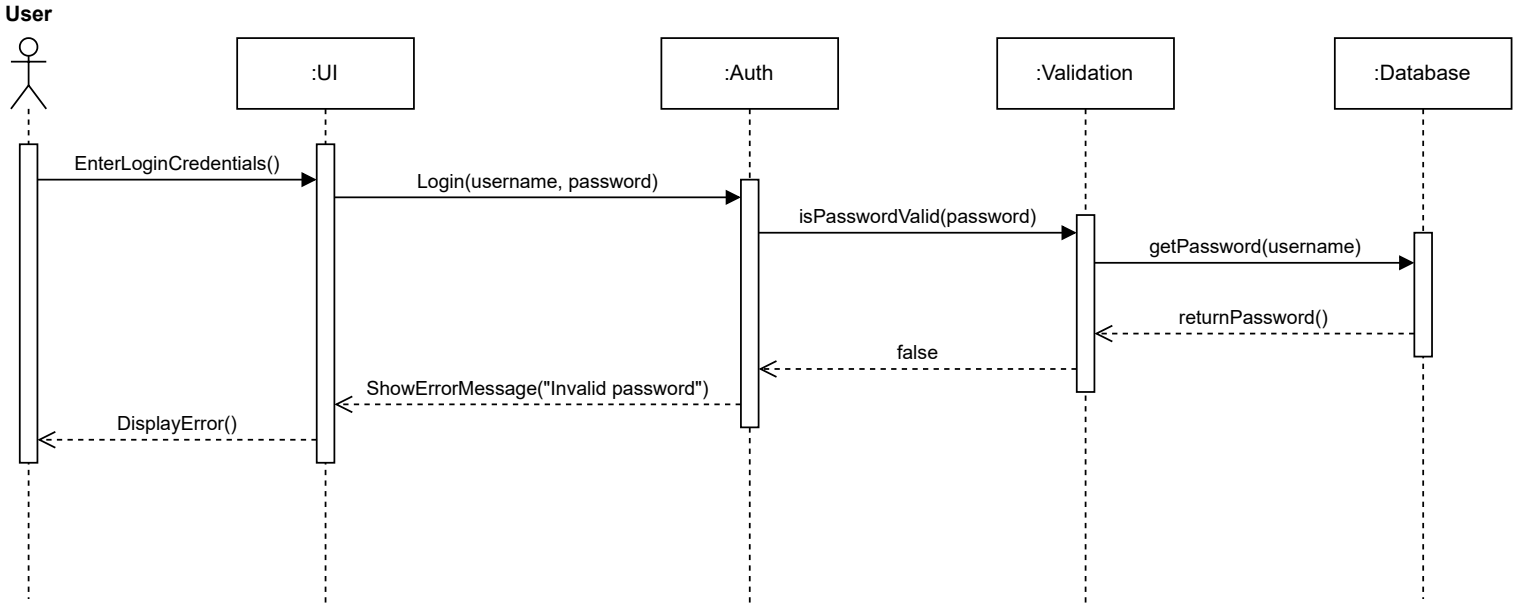
User Management



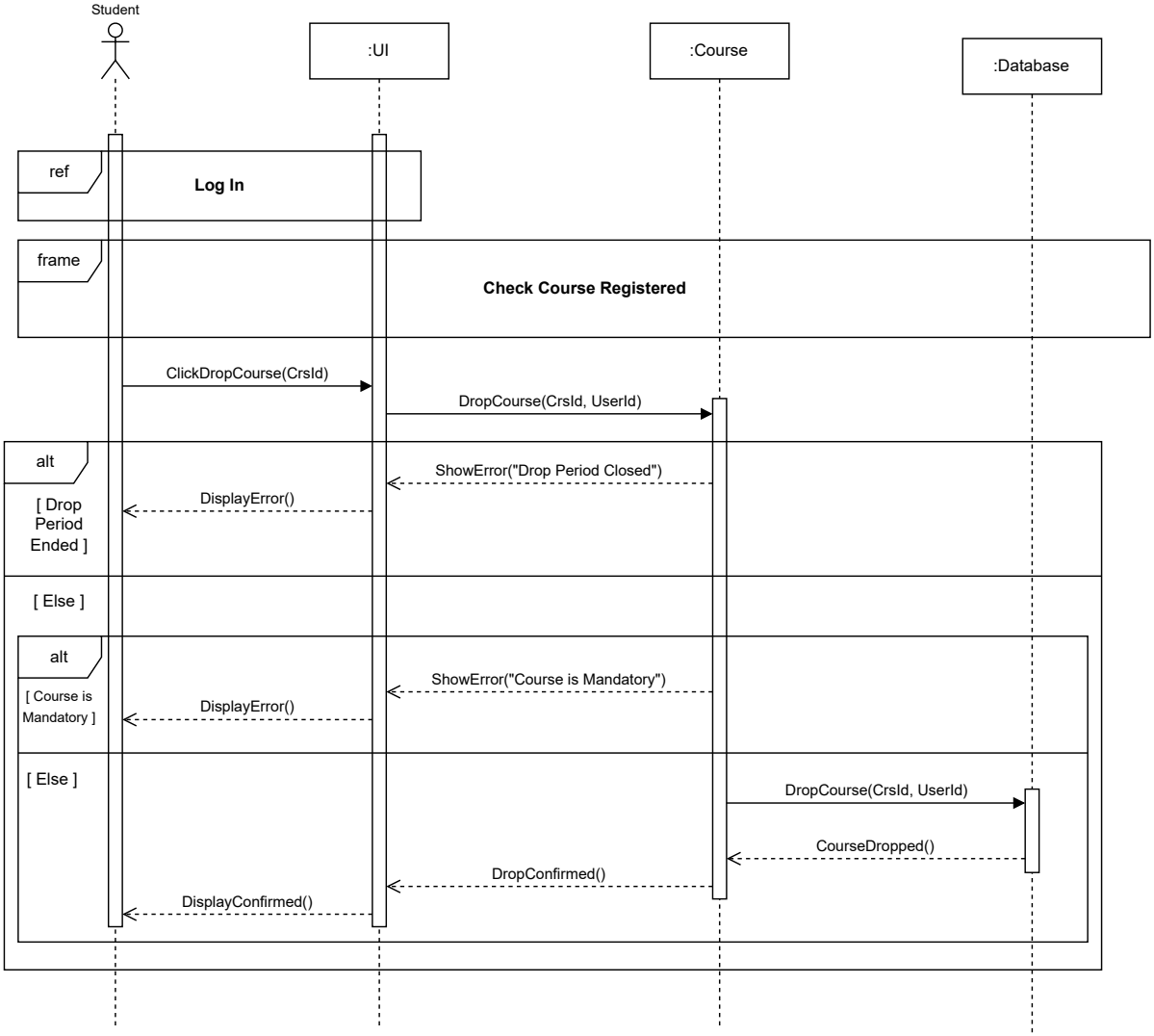
Assign Courses To Teacher



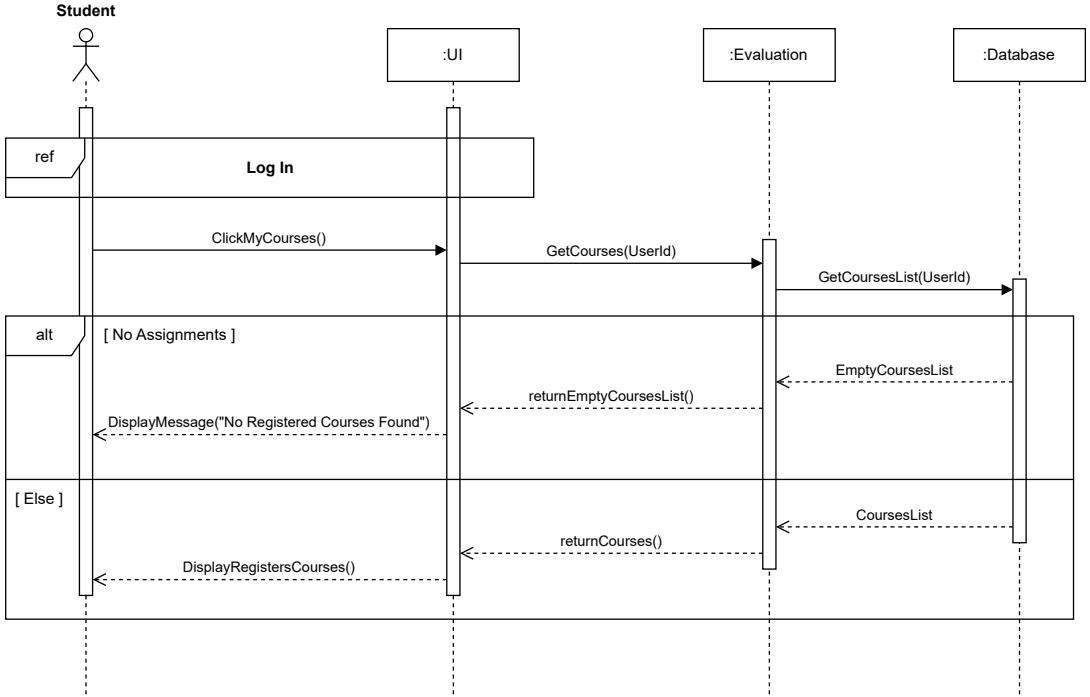
Display Error Password



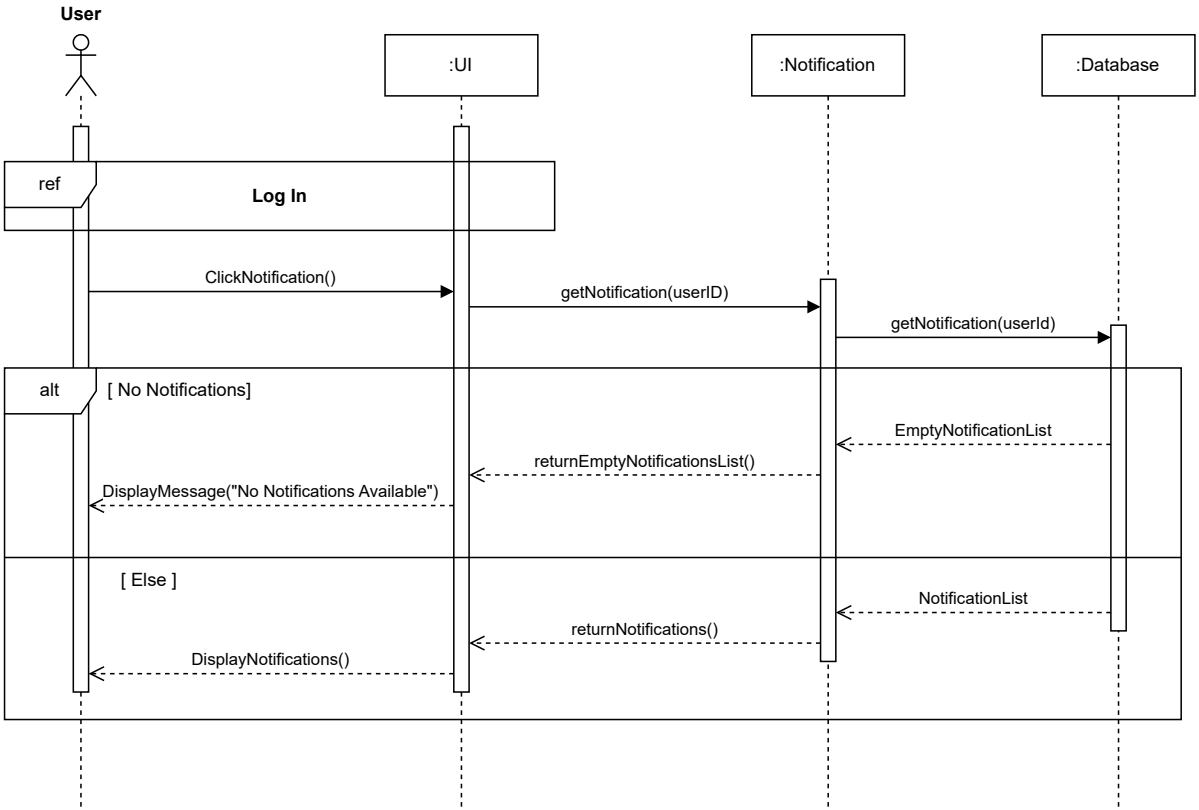
Drop Course


Student
:UI
:Course
:Database
ref
Log In
frame
Check Course Registered
ClickDropCourse(CrsId)
DropCourse(CrsId, UserId)
alt
[Drop Period Ended]
ShowError("Drop Period Closed")
DisplayError()
[Else]
alt
[Course is Mandatory]
ShowError("Course is Mandatory")
DisplayError()
[Else]
DropCourse(CrsId, UserId)
CourseDropped()
DropConfirmed()
DisplayConfirmed()

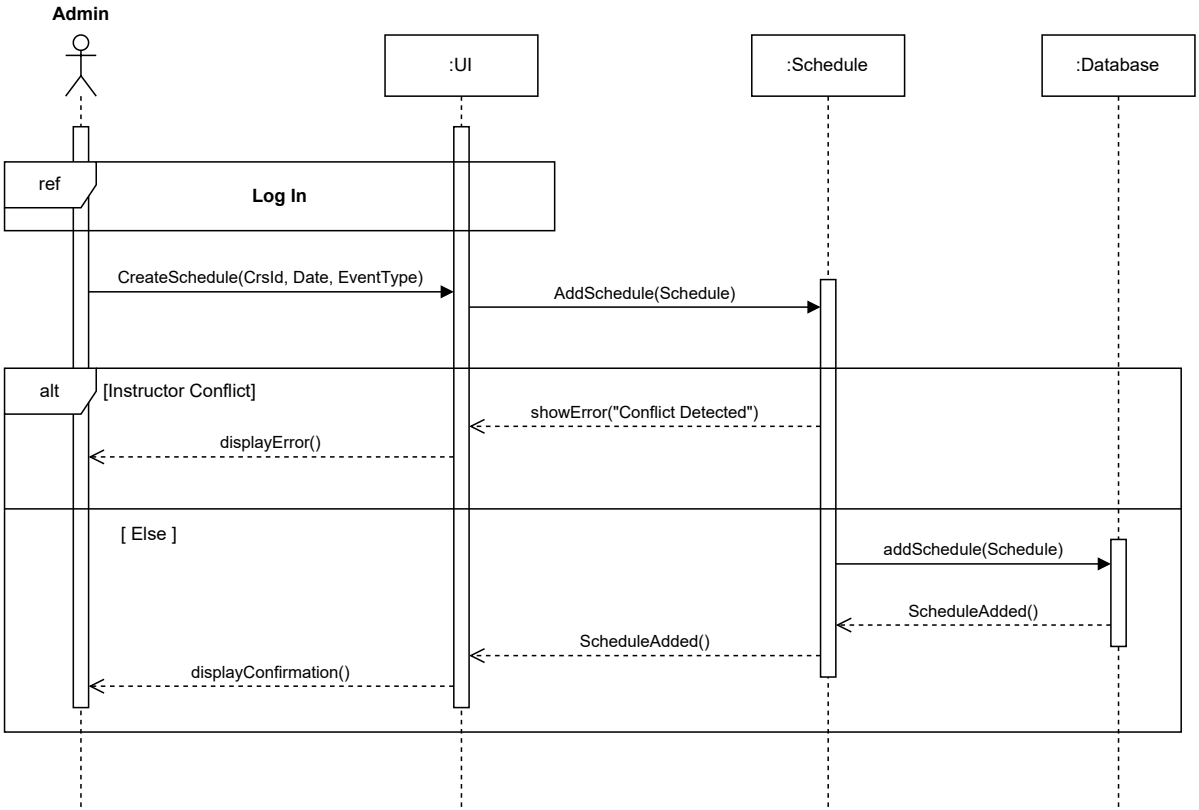
Check Course Registered



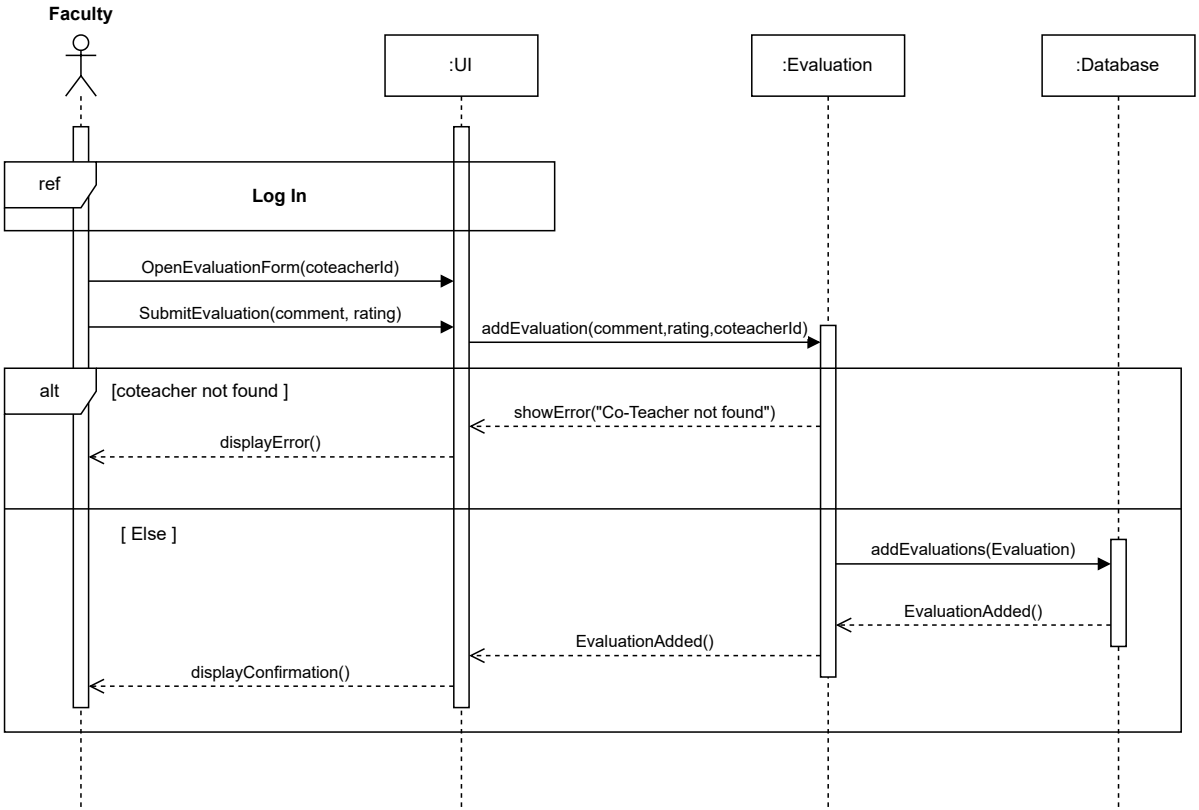
View Notification



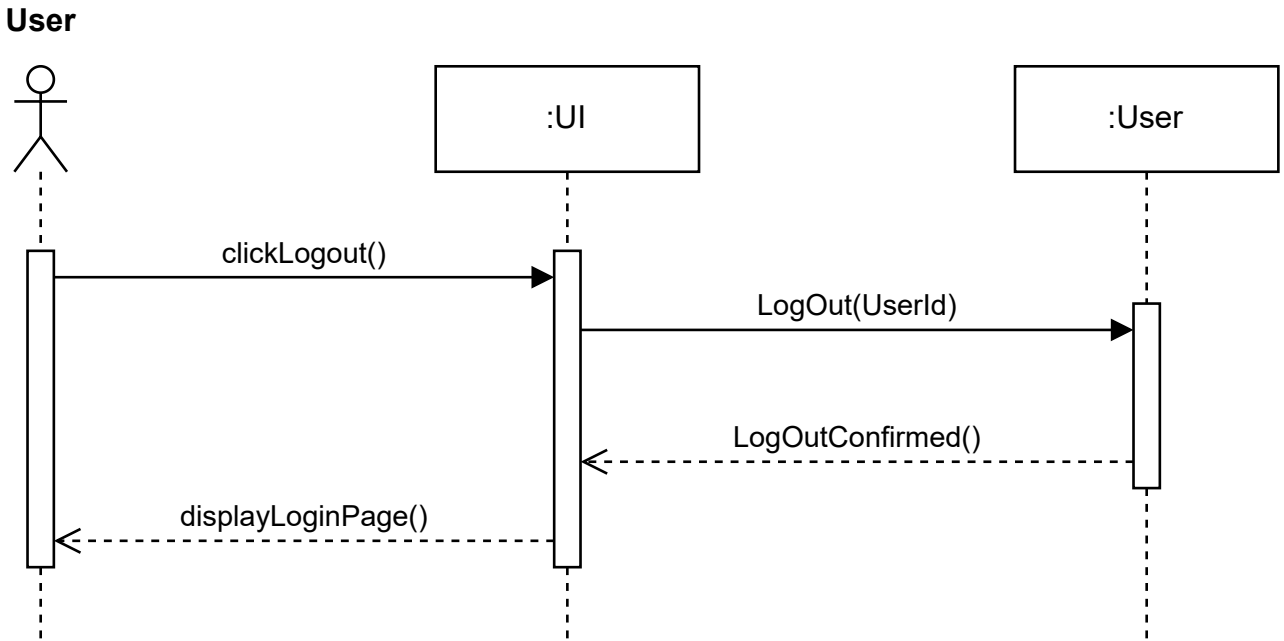
Create Course Calendar



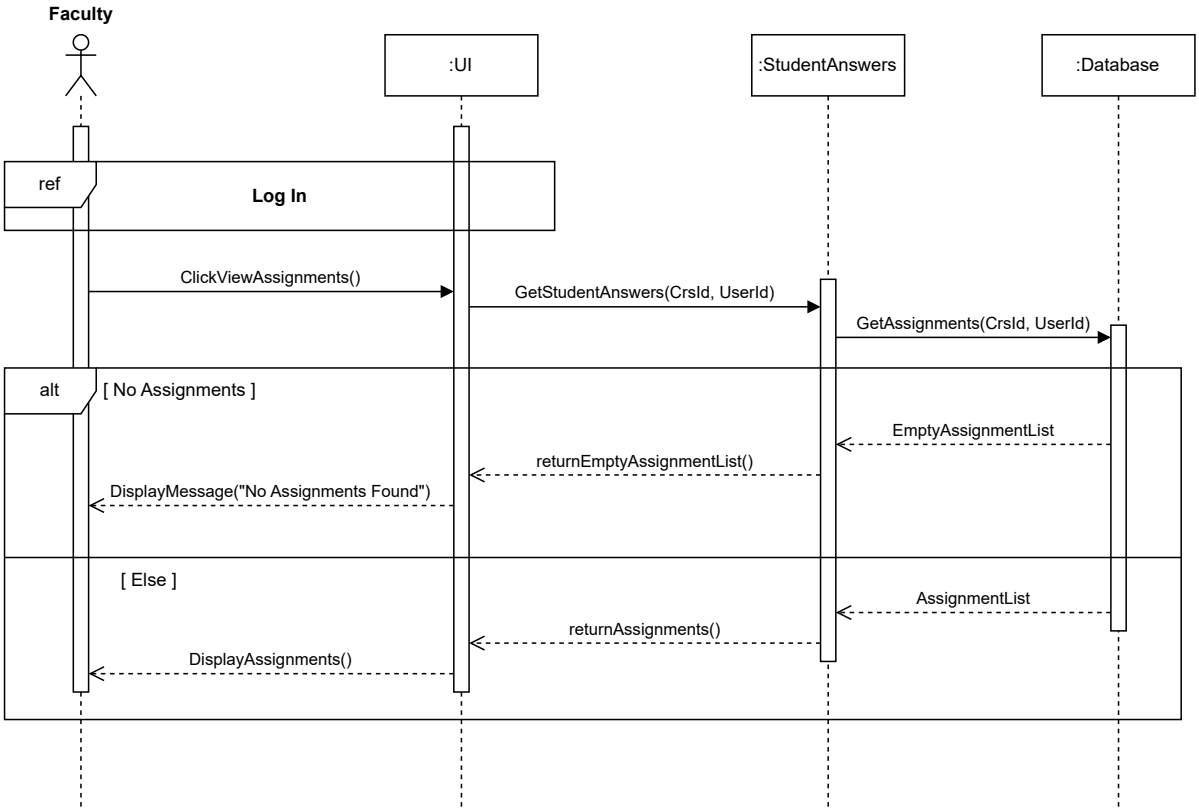
Rate Co-Teacher



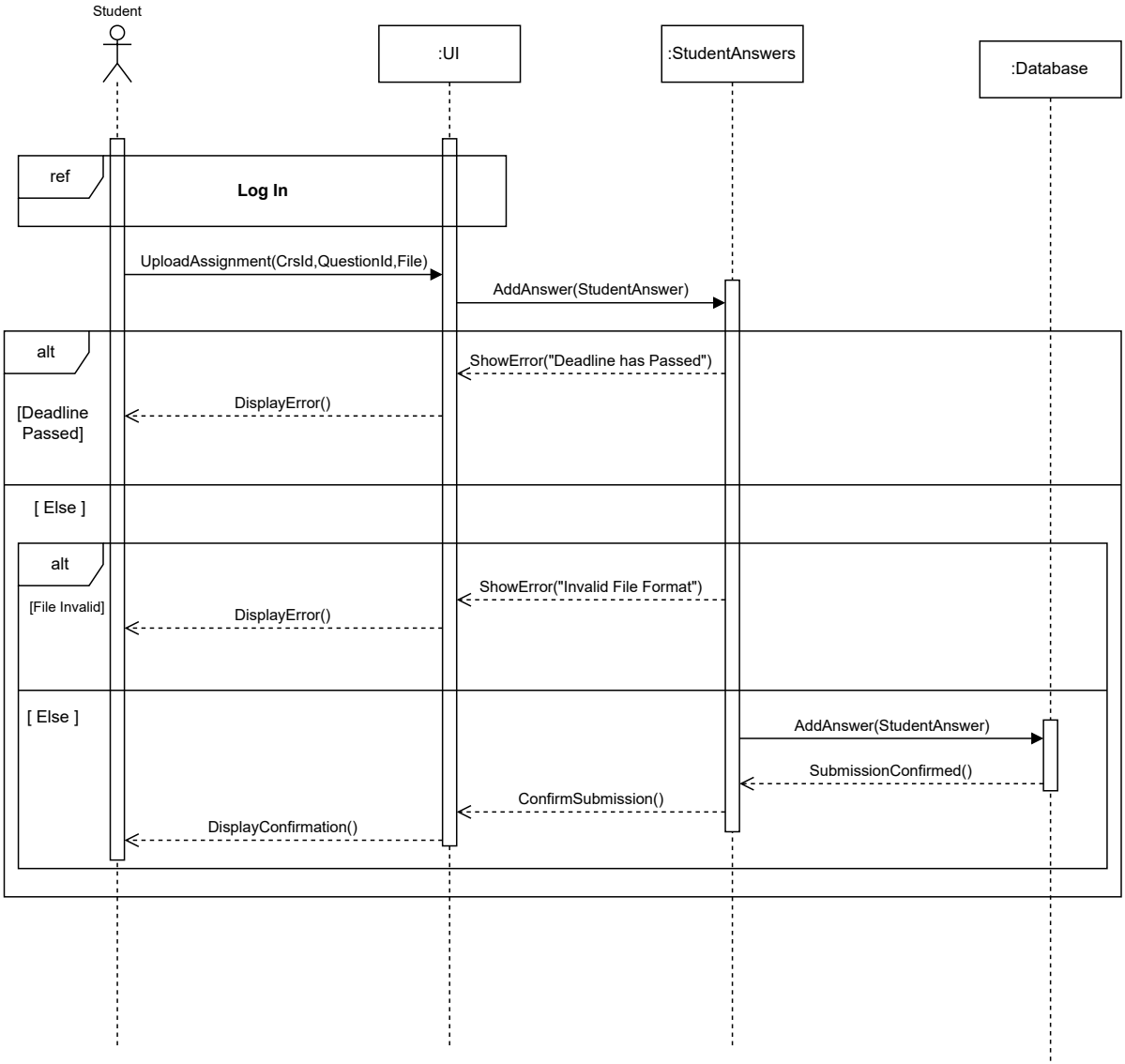
Log Out


User
:UI
:User
clickLogout()
LogOut(UserId)
LogOutConfirmed()
displayLoginPage()

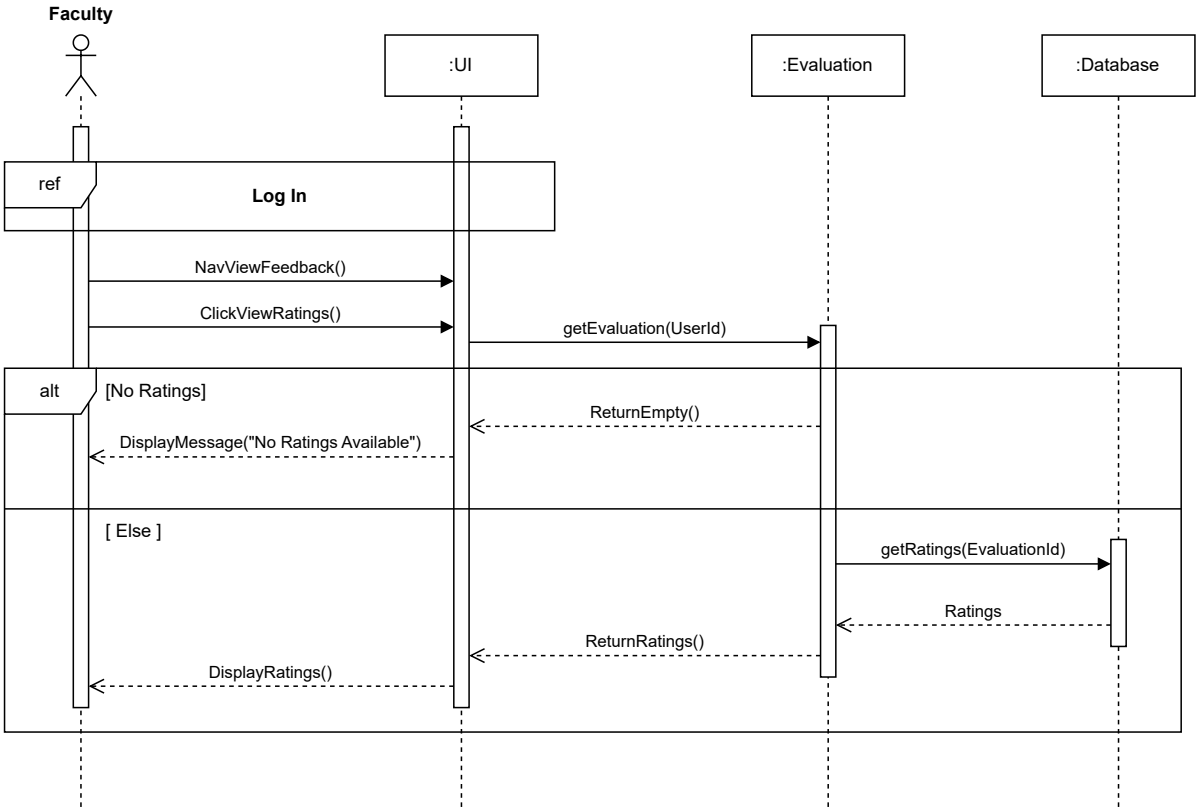
Check Assignments



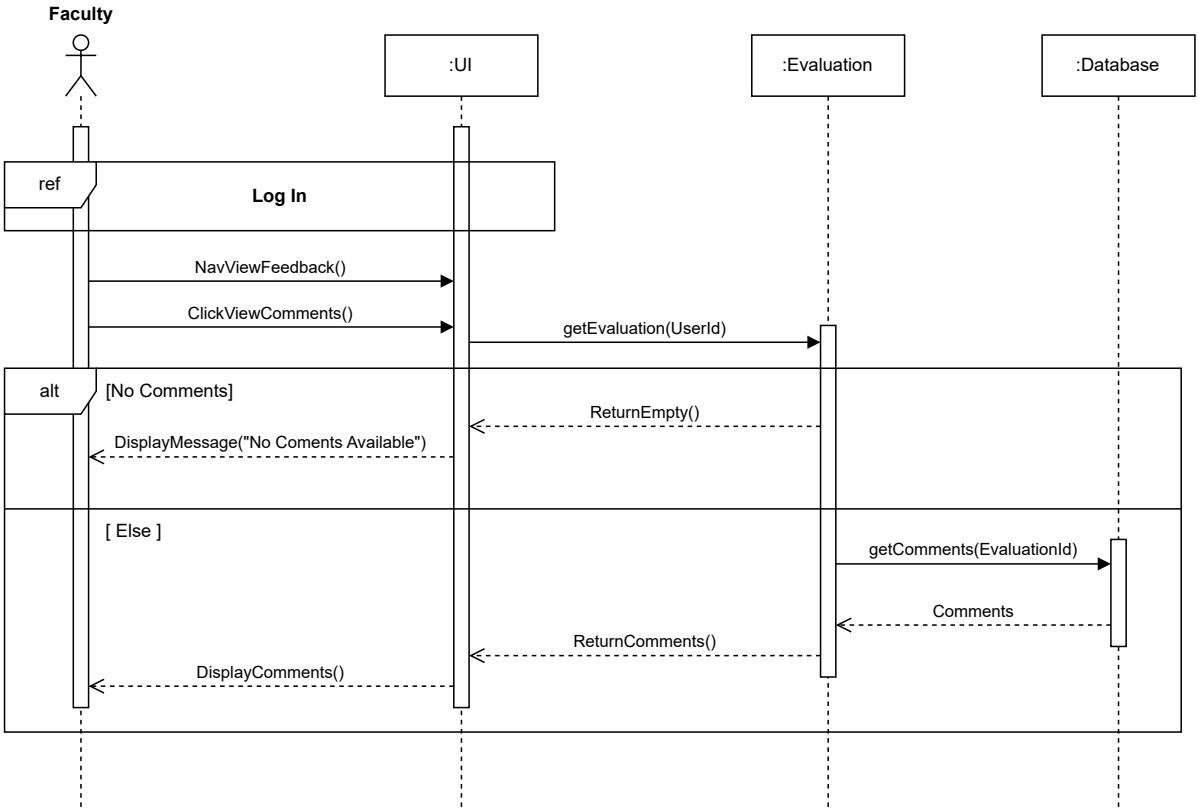
Submit Assignments



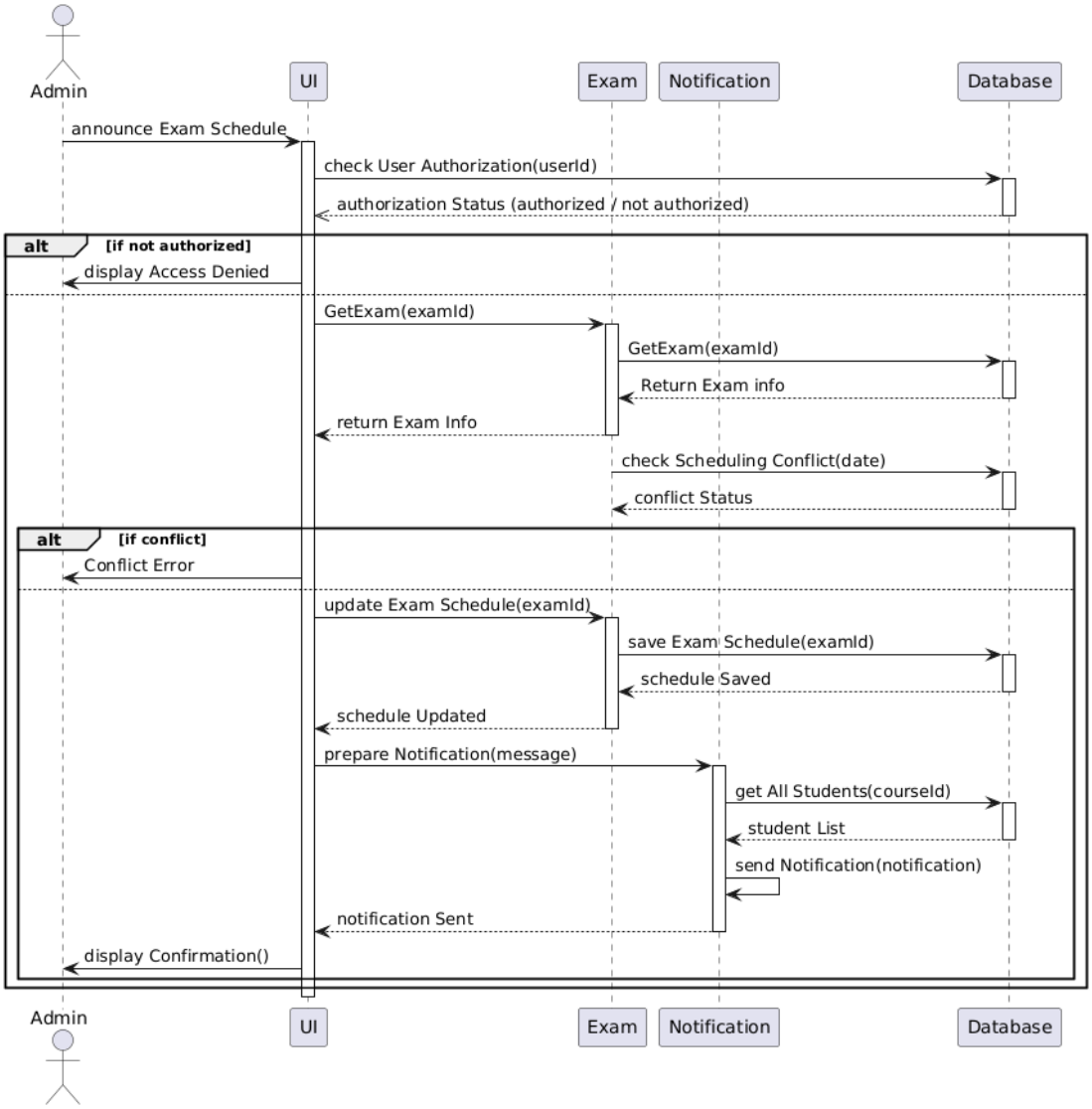
View Final Rate



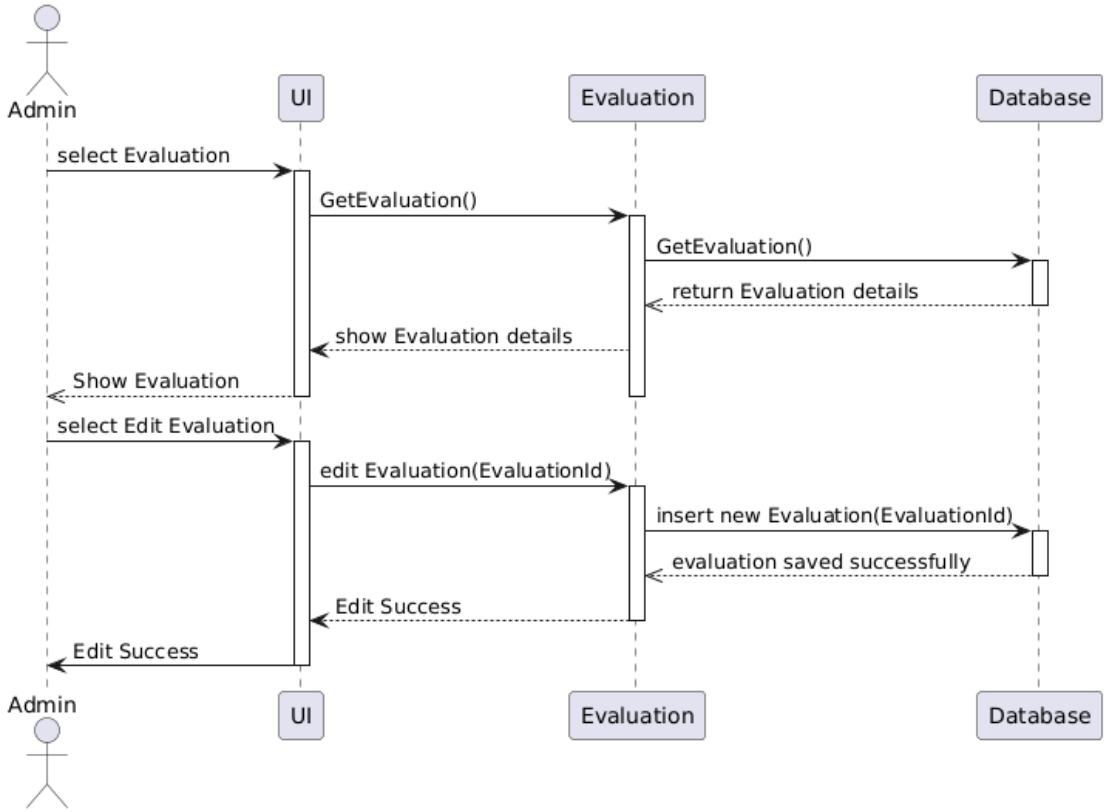
View Comments



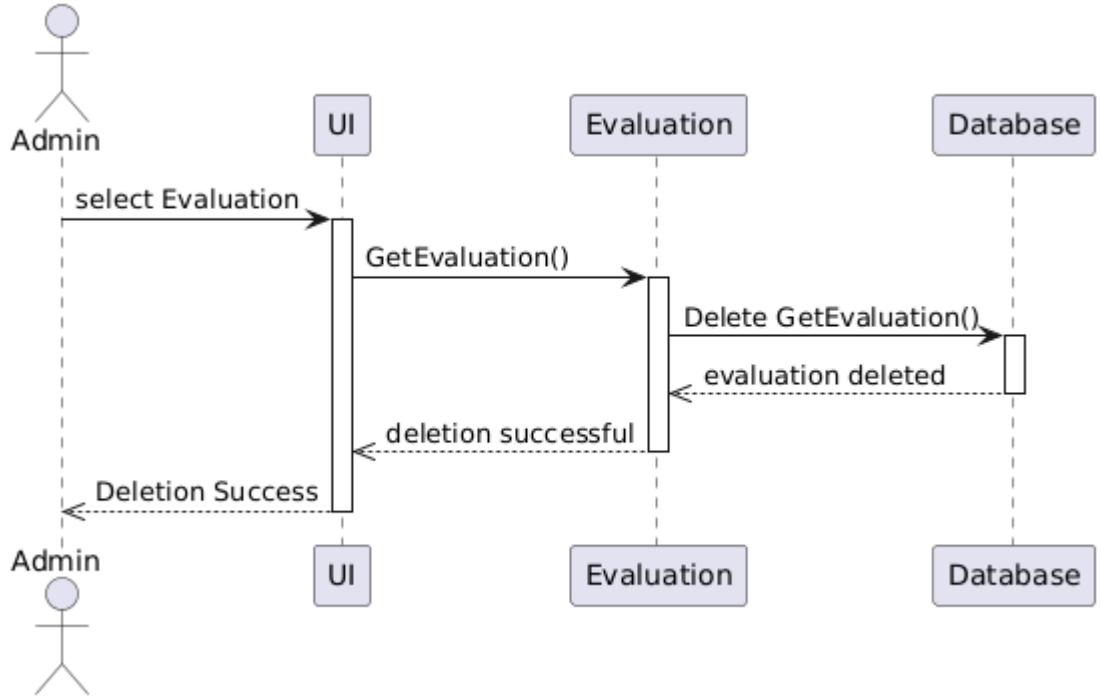
Use Case : Announce Schedule Exam



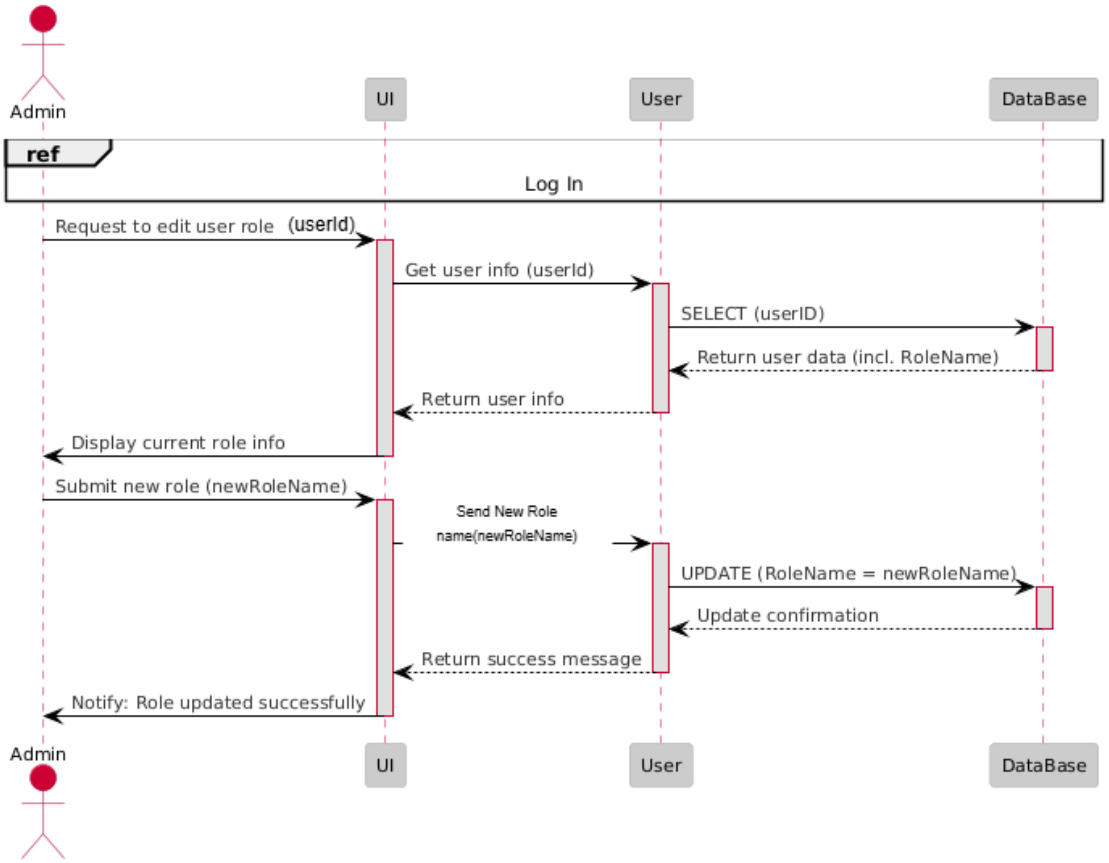
Use Case :Edit Evaluation



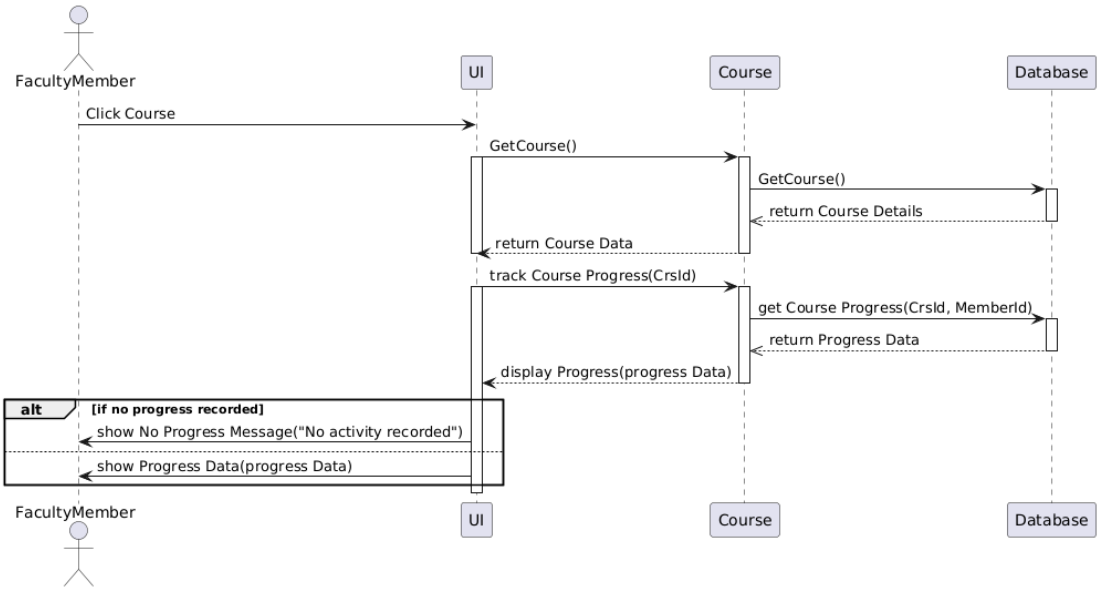
Use Case : Delete Evaluation



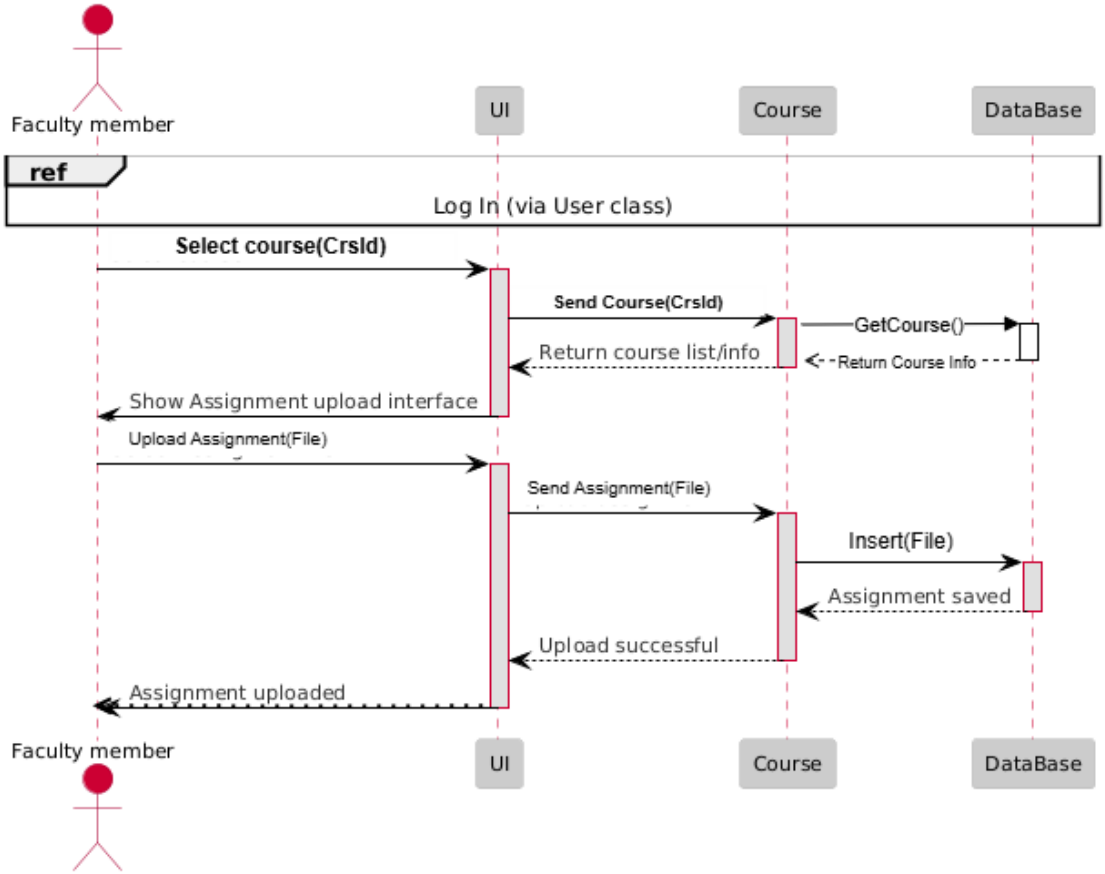
Use Case : Edit Roles



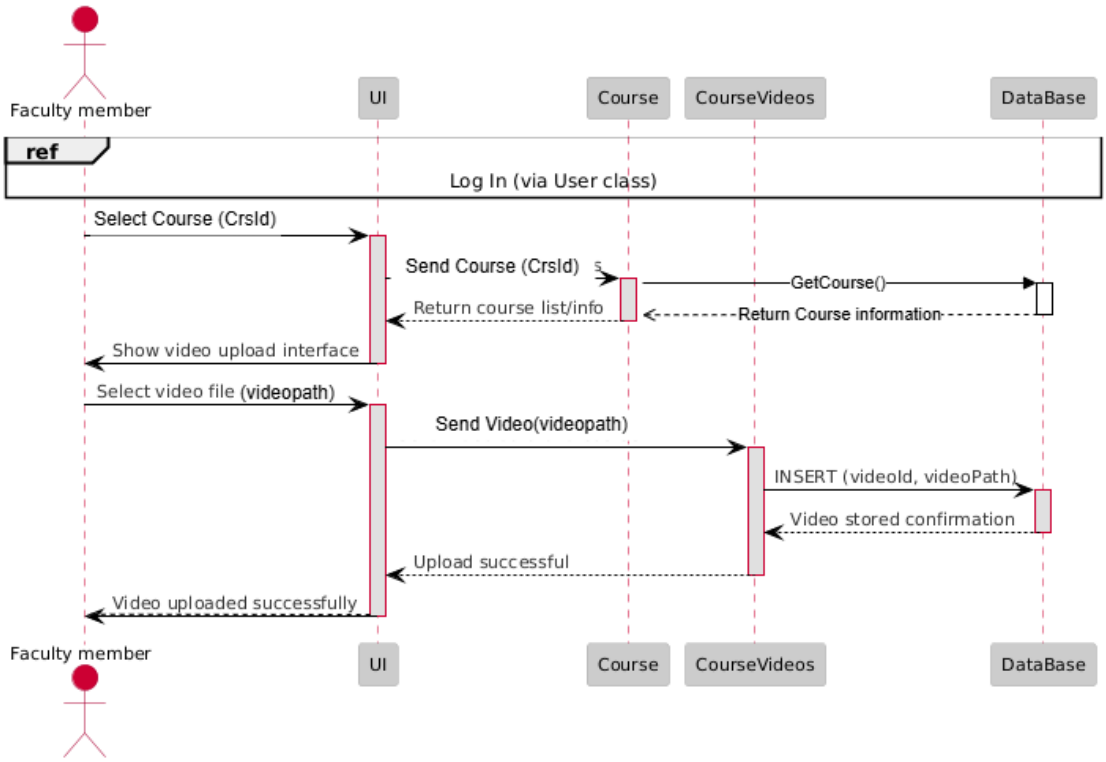
Use Case : Track Course Prograce



Use Case : Upload Assignment

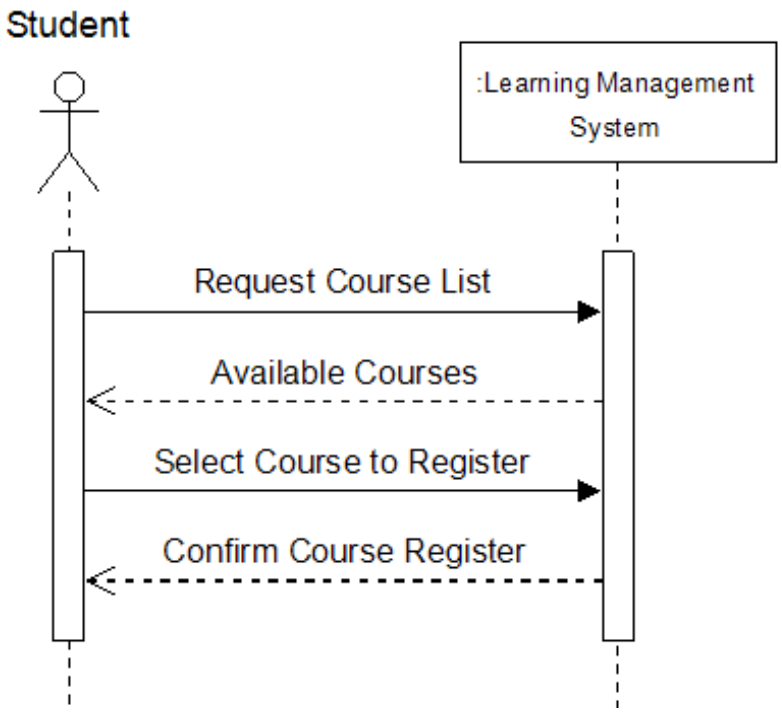


Use Case : Upload Video

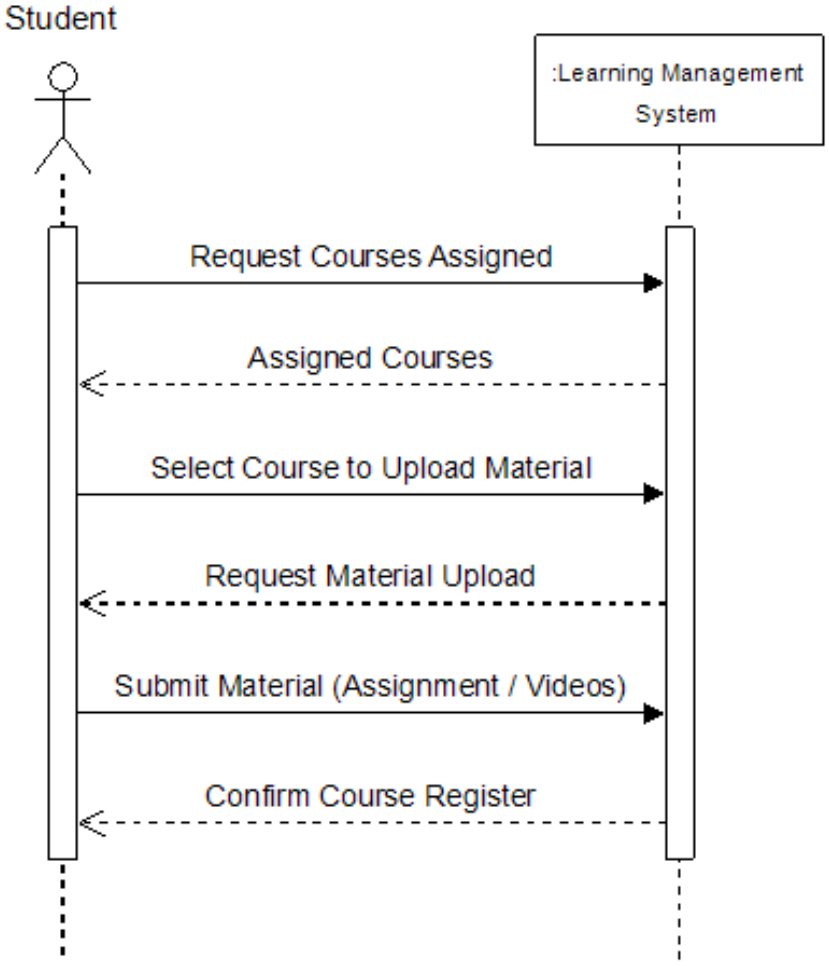


f) System Sequence Diagrams (SSDs)

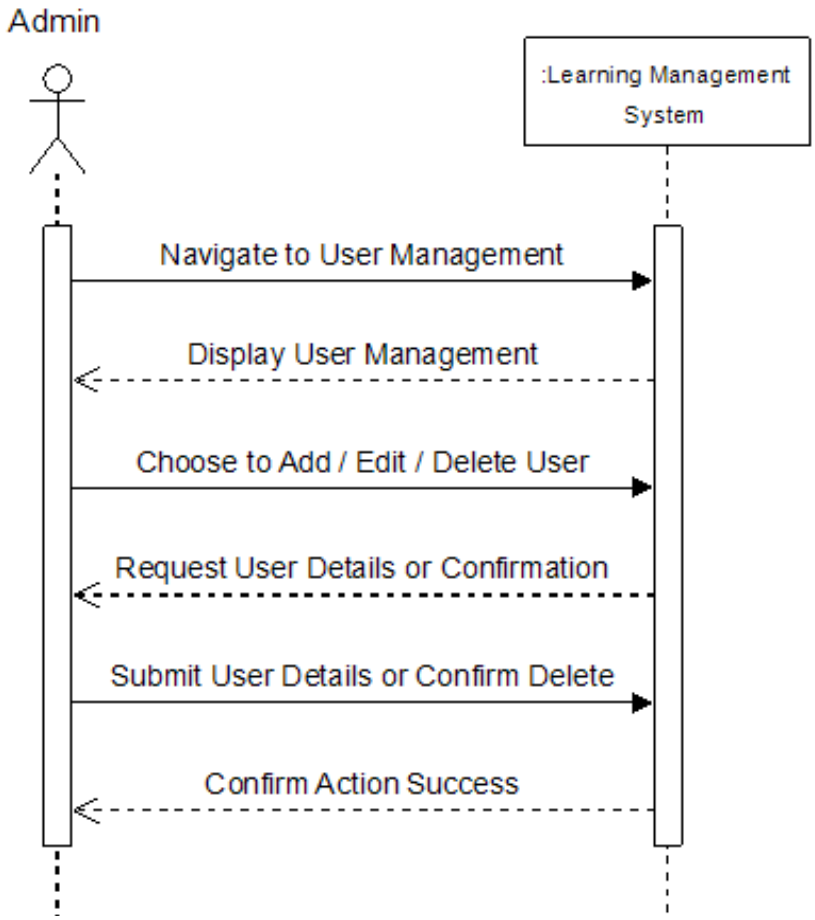
Register Courses



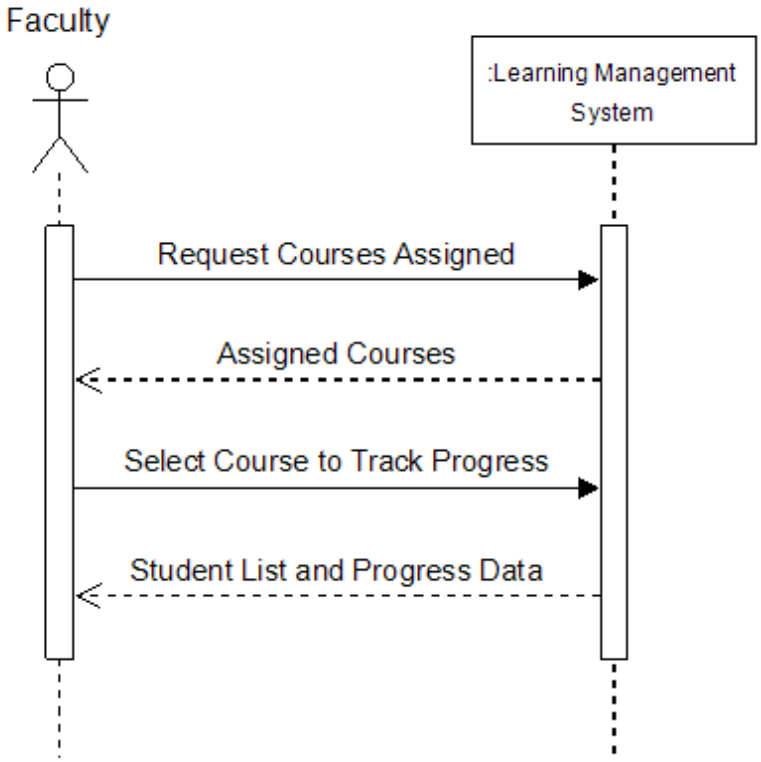
Upload Material



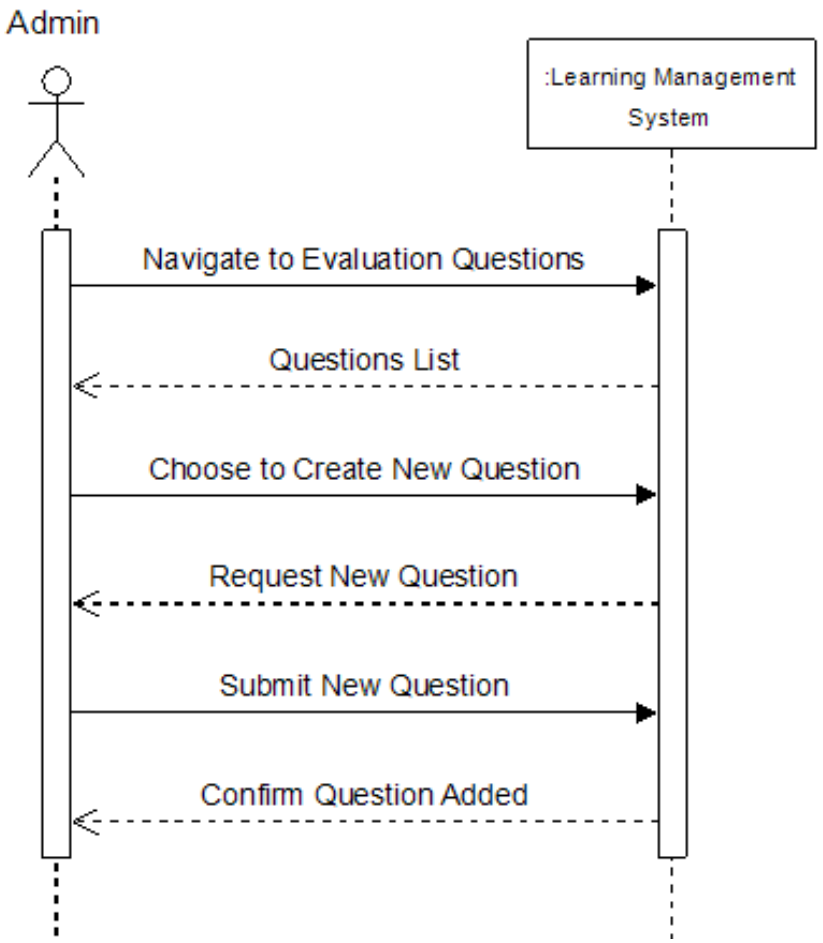
Manage User



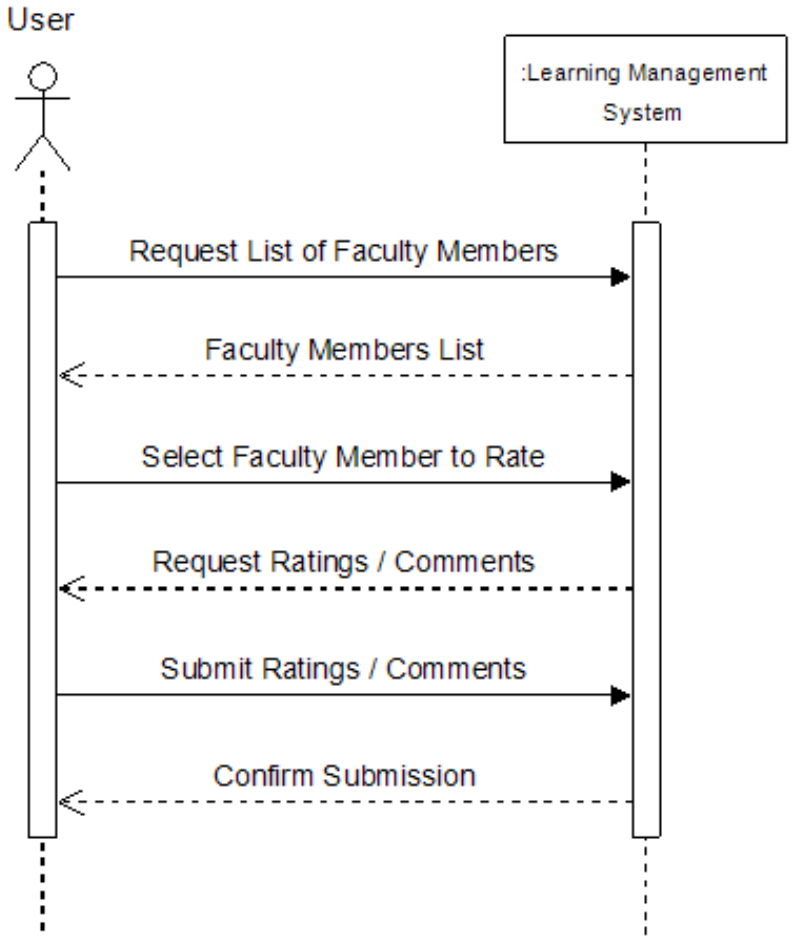
Track Course Progress



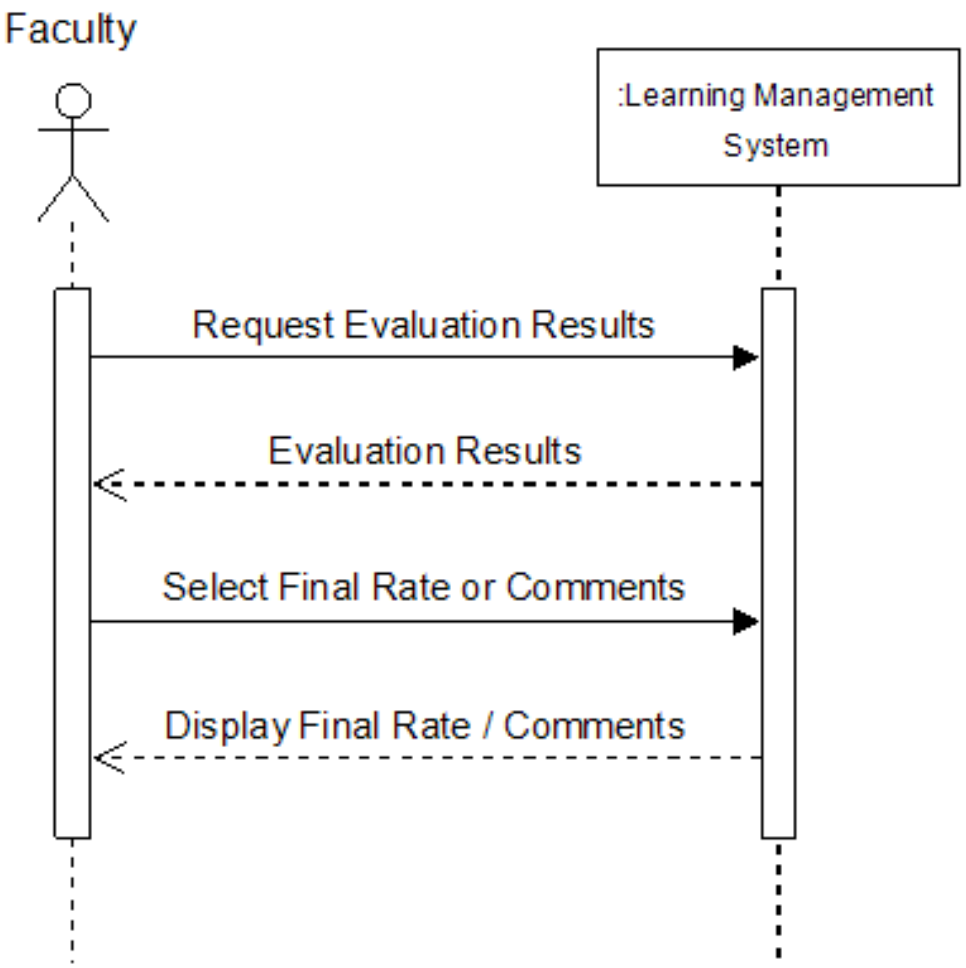
Create Evaluation Questions



Rate Faculty Member

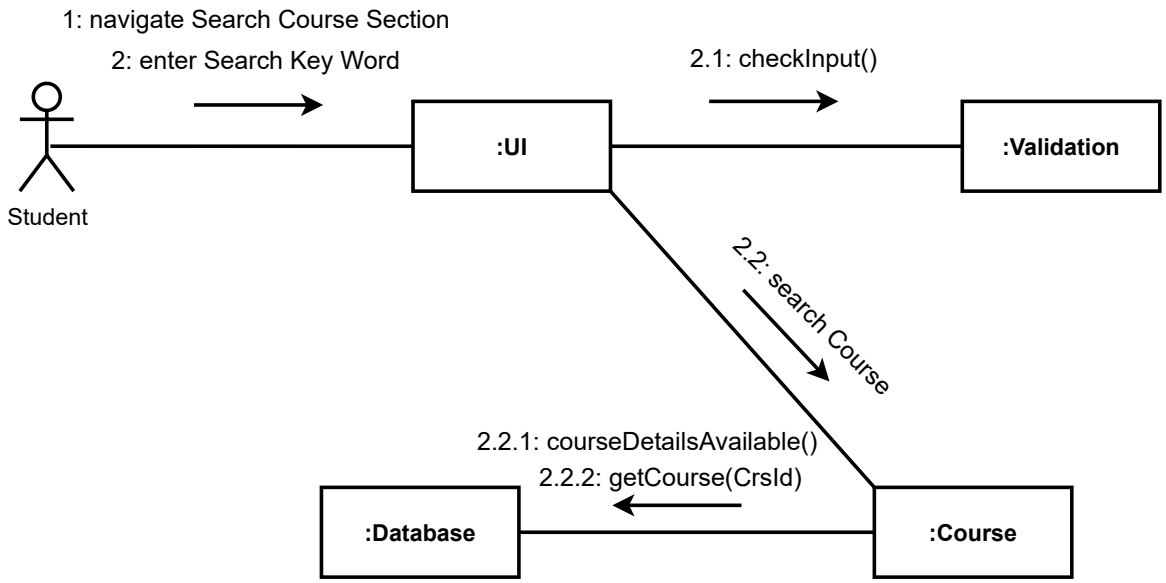


View Feedback

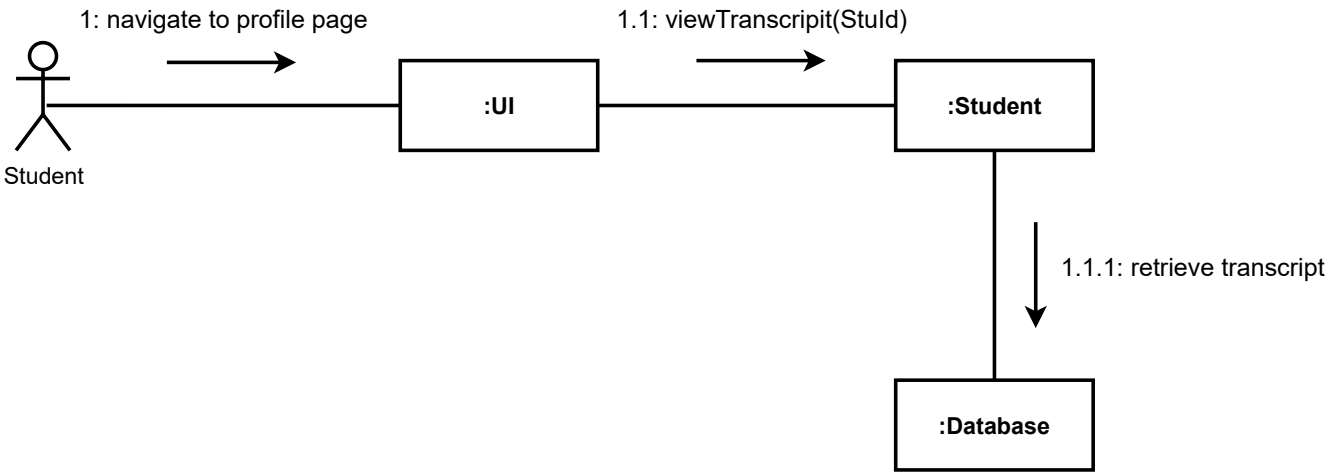


g) Collaboration/Communication Diagram(s)

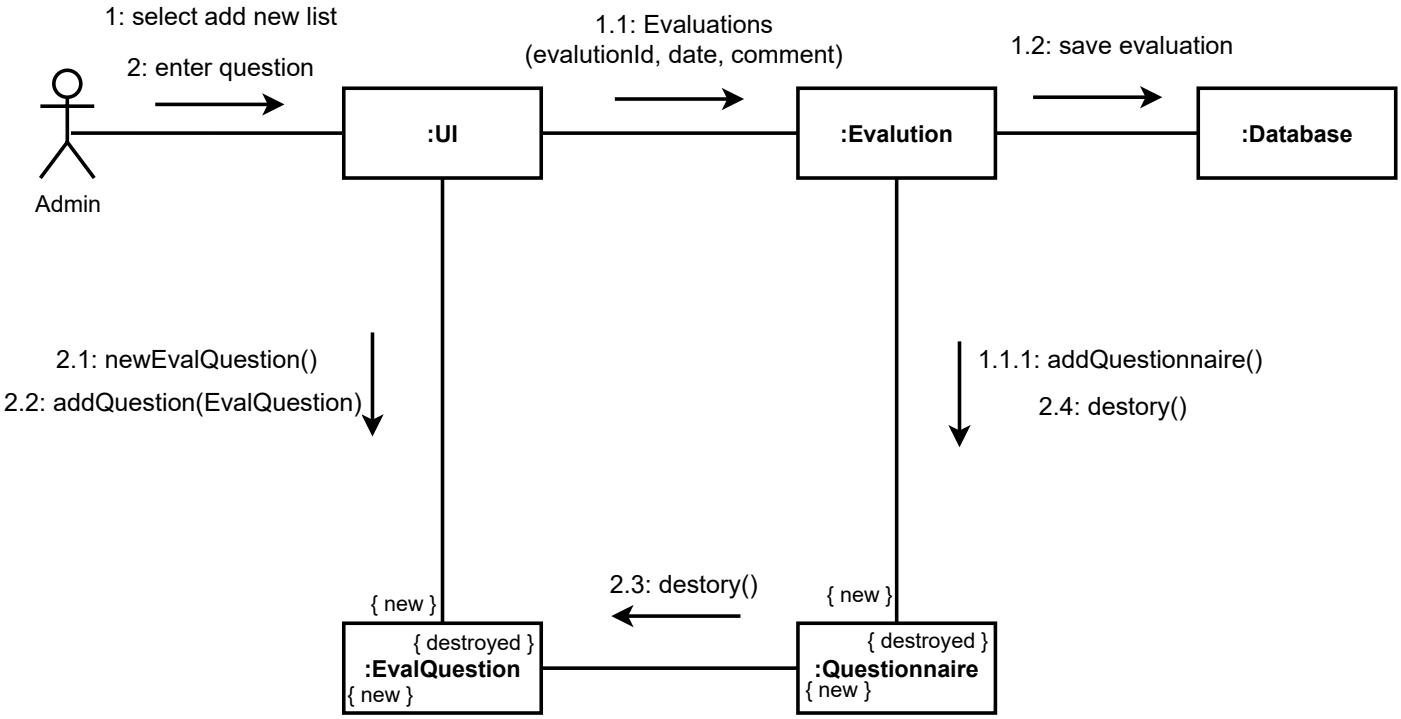
Search for course



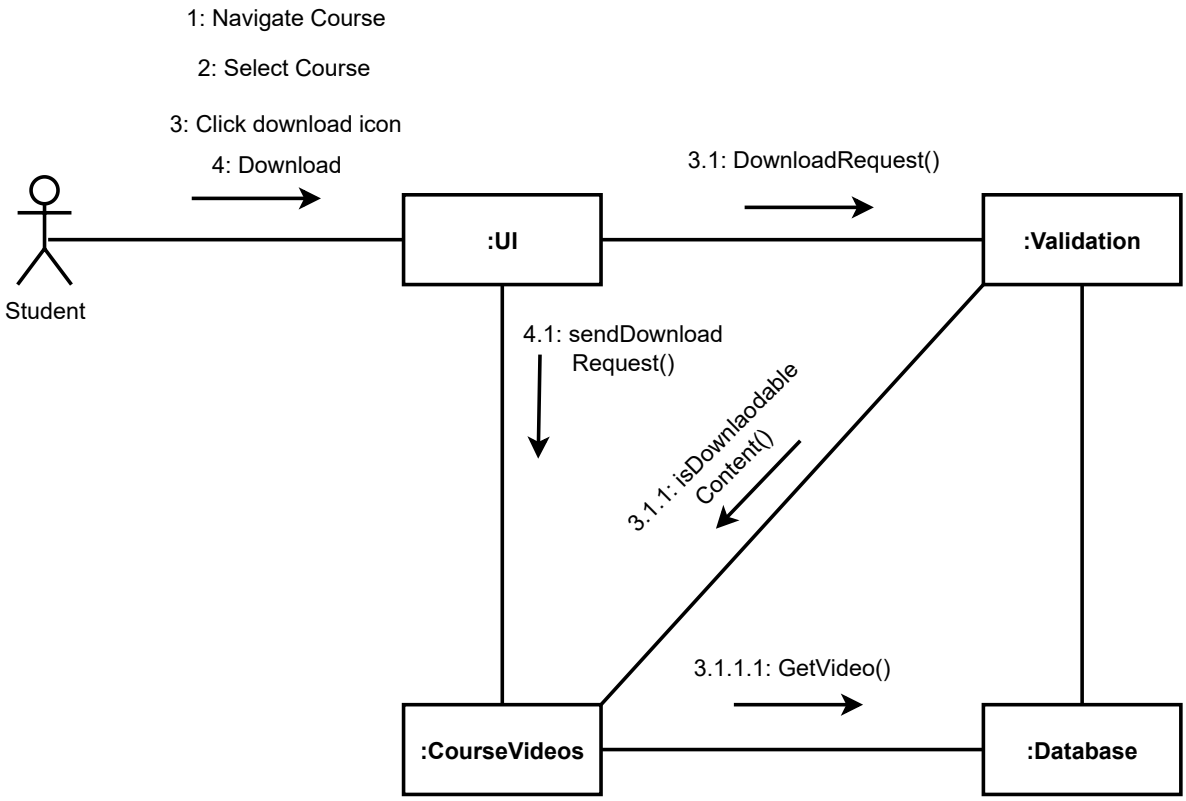
View transcript



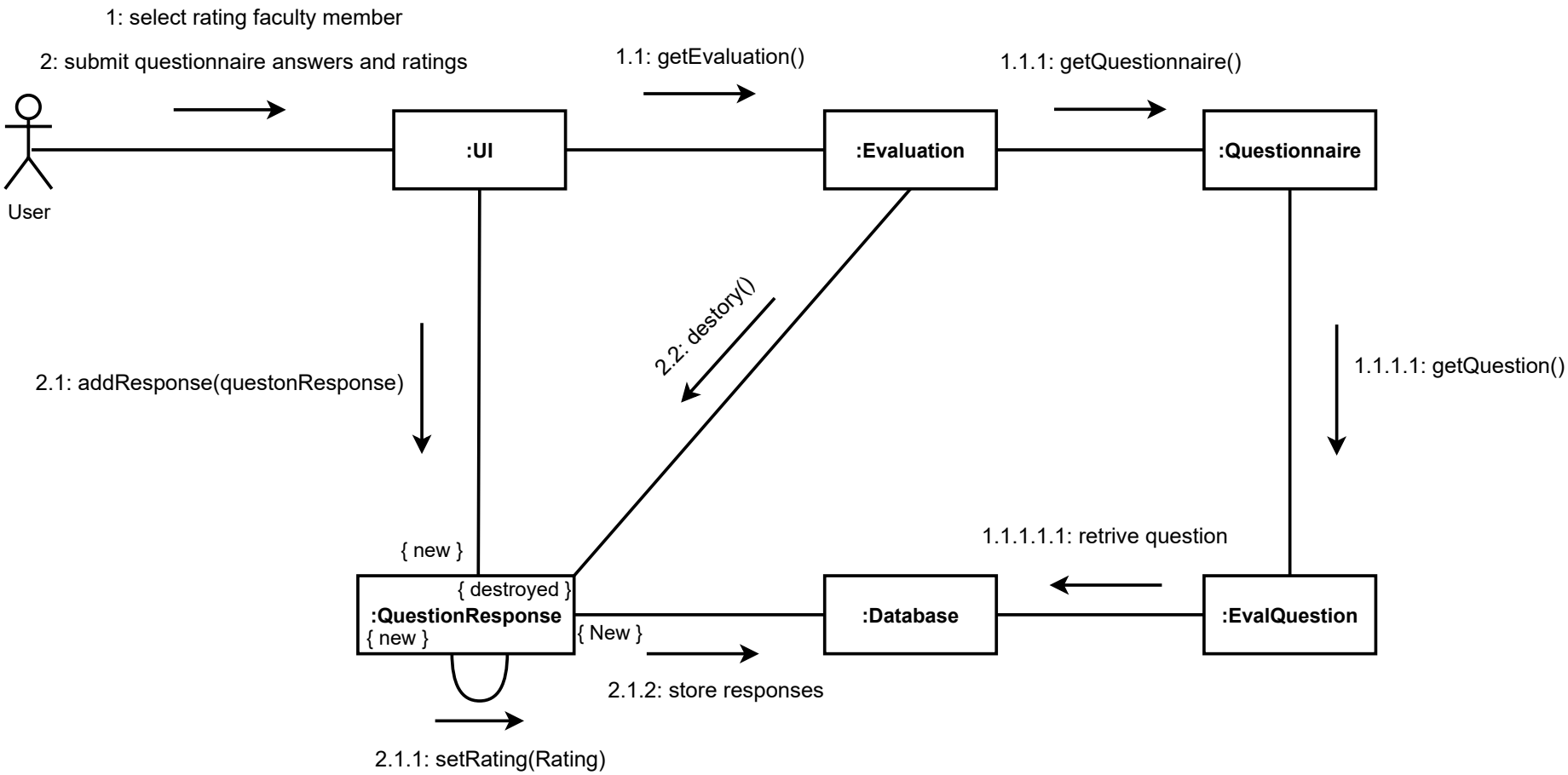
Create Evolutions Questions



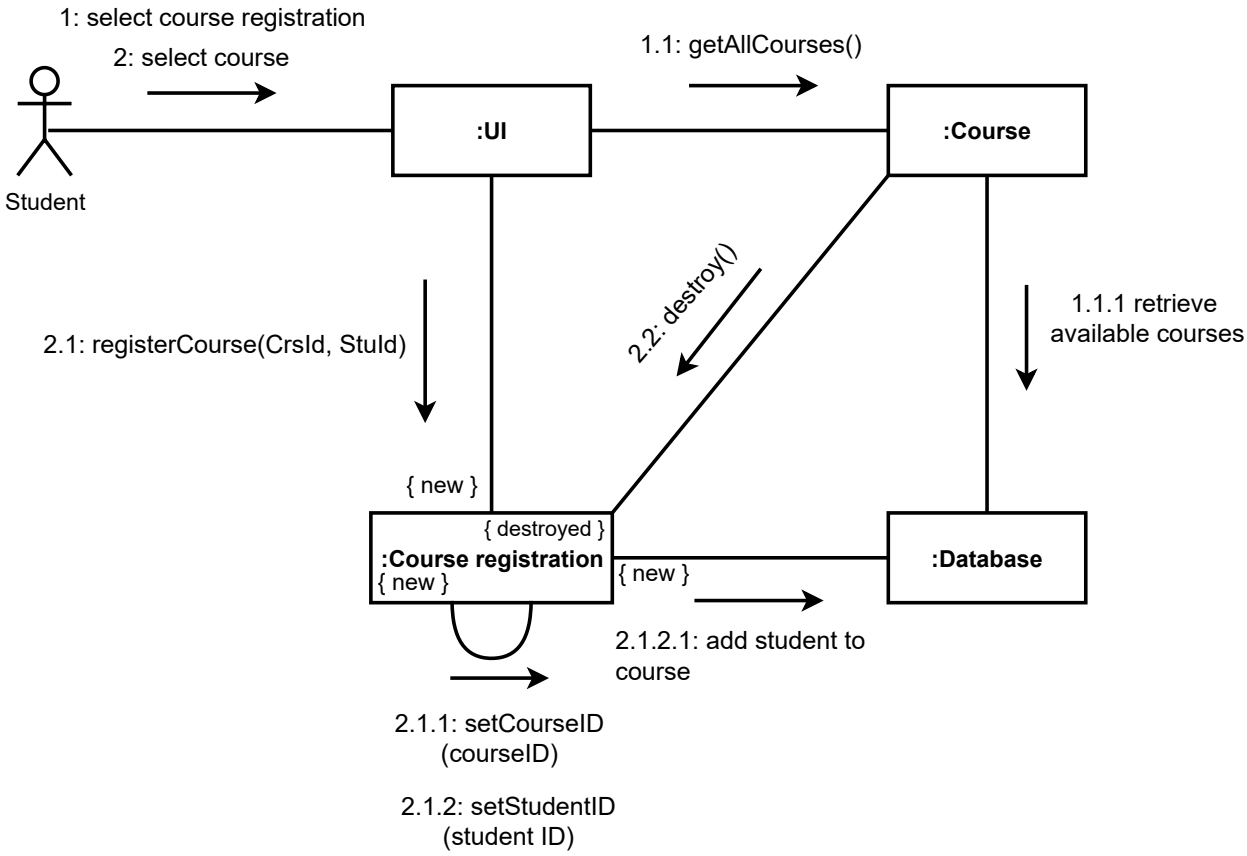
Download Online Course



Rate faculty member



Register courses



h) Which strategy (or strategies) did you use to implement the use cases: One Central Class, Actor Class, or Use-Case class? Please explain your choice and the potential advantages/disadvantages of your design.

1 Actor Class:

In this strategy, each actor (user role) in the system has a corresponding class. Each actor class handles the use cases that are relevant to that actor.

Advantages:

Modularity: Changes to one actor's use cases won't affect others.

Clarity: It's clear which use cases apply to which actors.

Disadvantages:

Redundancy: If different actors have similar use cases, there may be duplicated code.

Complexity: The number of classes can grow quickly with the number of actors.

---

2 Use-Case Class:

This strategy involves creating a separate class for each use case. Each class encapsulates all the logic for that use case.

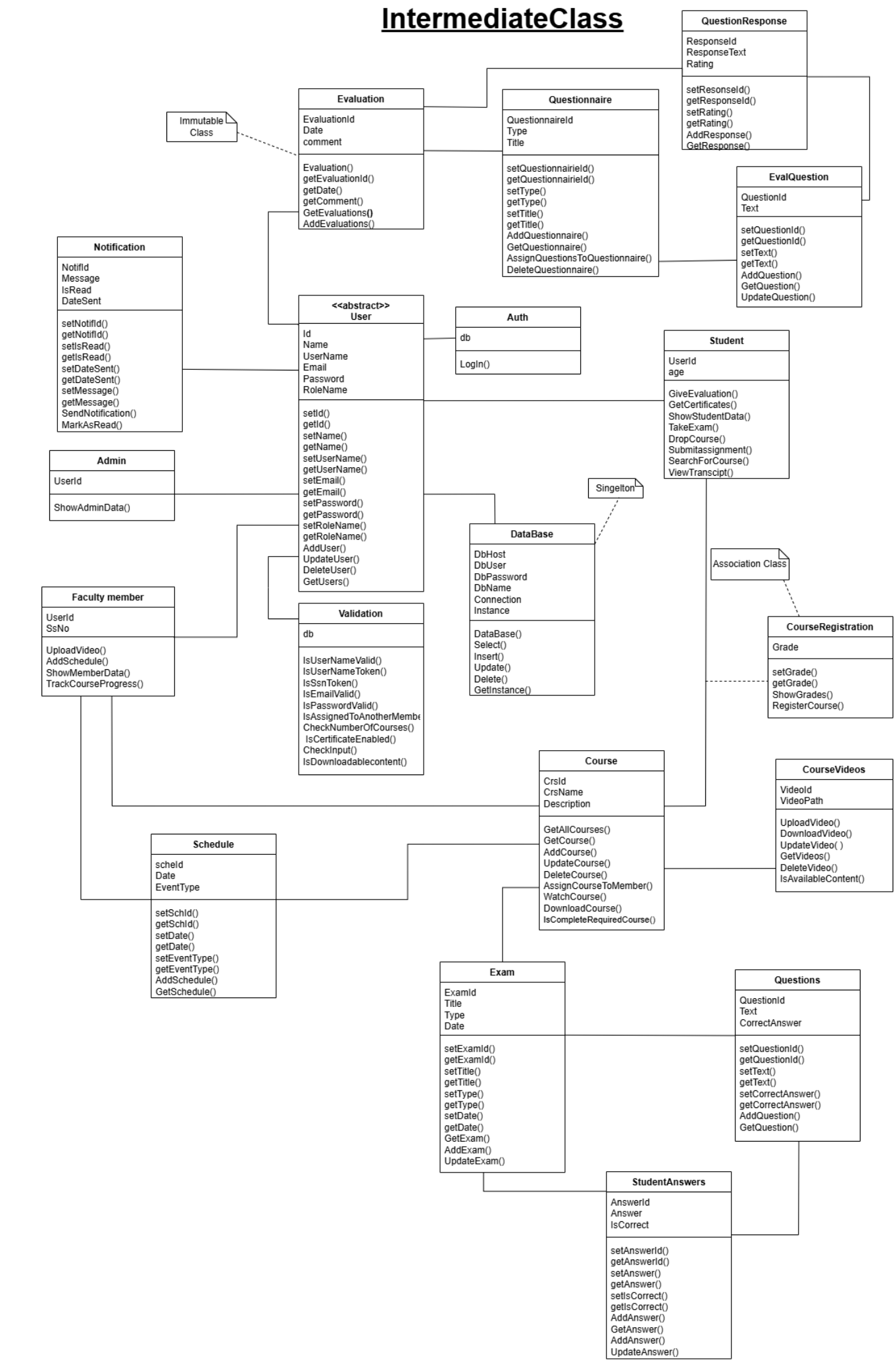
Advantages:

High cohesion: Each class has a single, well-defined responsibility. Easy to modify: Changes to one use case won't affect others.

Disadvantages:

Potential for many small classes: If there are many use cases, there will be many classes, which can make the system harder to understand. Communication overhead: If use cases need to share information, they'll need to communicate through a shared medium, which can add complexity.

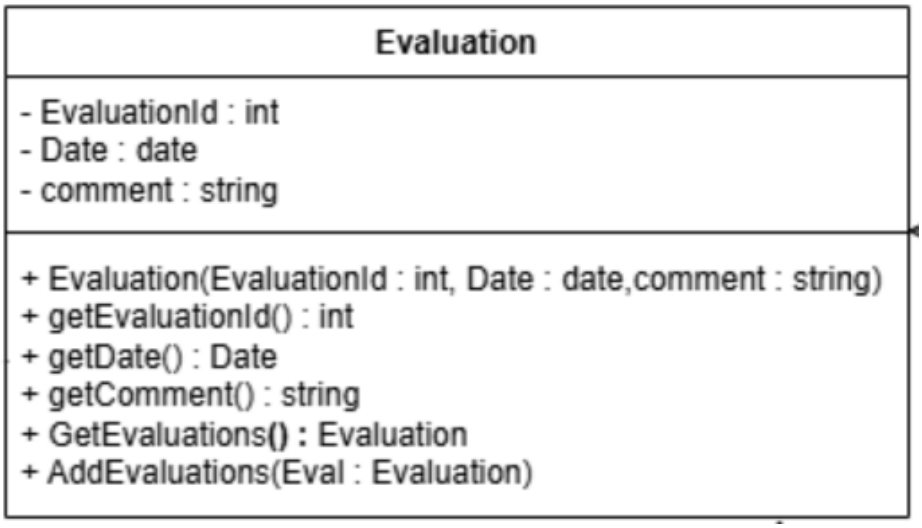
i) Class Diagram 2: An intermediate version based on the interaction diagrams



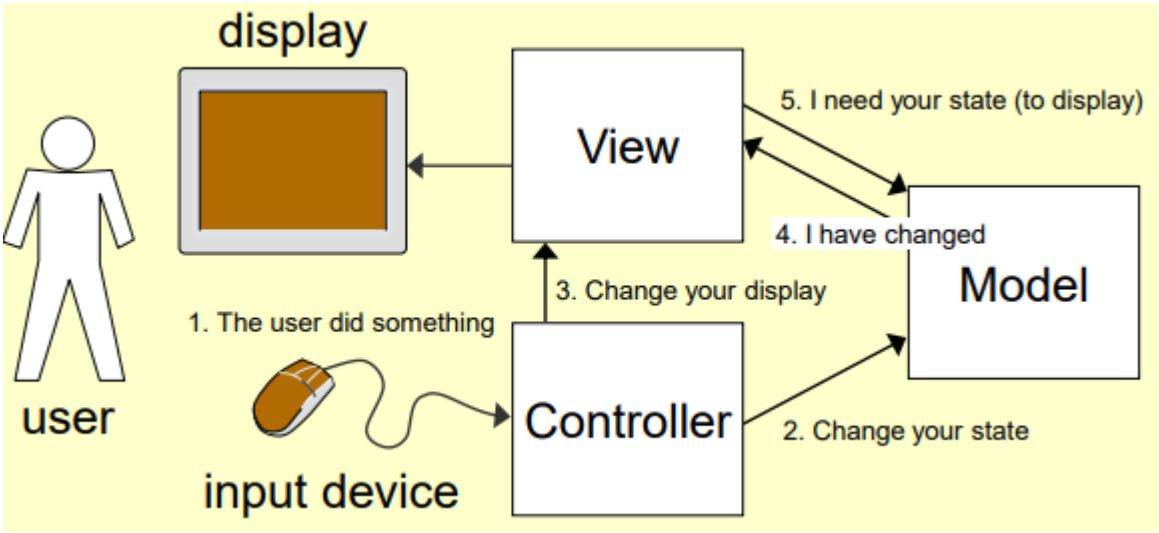
j) Three Design Patterns Applied

Design Patterns Description

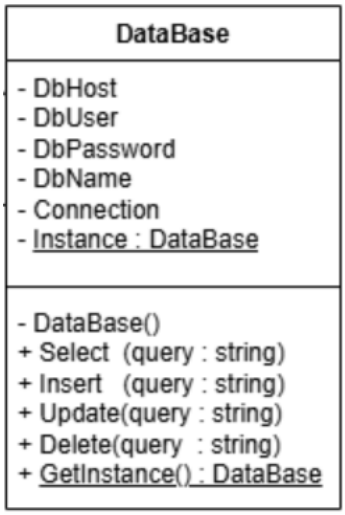
- **Name:** Immutable design Pattern.
- **Context:** The Evaluation object that it's state can't be changed after the object is created.
- **Problem:** How can we ensure data integrity and prevent accidental or unauthorized changes to the object state once it has been created?
- **Forces:** There must be no loopholes that would allow illegal modification of an immutable object.
- **Solution:**
  - Ensure the evaluation constructor is the only way you can set the data member's values.
  - The evaluation methods that have access to it's variables must not be able to change them.
- **References:** The Immutable design pattern gained widespread adoption after Bloch's influential chapter in 'Effective Java' highlighted its benefits for concurrent programming and system stability.



- **Name:** Models views controller (MVC) design Pattern.
- **Context:** Dividing a feature into three primary logical components model, view and controller. Each component handles a development aspect independently ensuring that a change in one doesn't affect the others.
- **Problem:** How can we structure the course feature so that it handles the data management, UI and input handling.
- **Forces:** The three components should not affect each other.
- **Solution:**
  View: Handles how the course is presented at the page.
  Model: Manages the course data and business logic.
  Controller: Manages user input and updates between the Model and the View.
- **References:** The MVC design pattern was introduced by Trygve Reenskaug in the 1970s while working on Smalltalk at Xerox PARC. Later popularized through the book Design Patterns: Elements of Reusable Object-Oriented Software by Gamma et al. in 1994

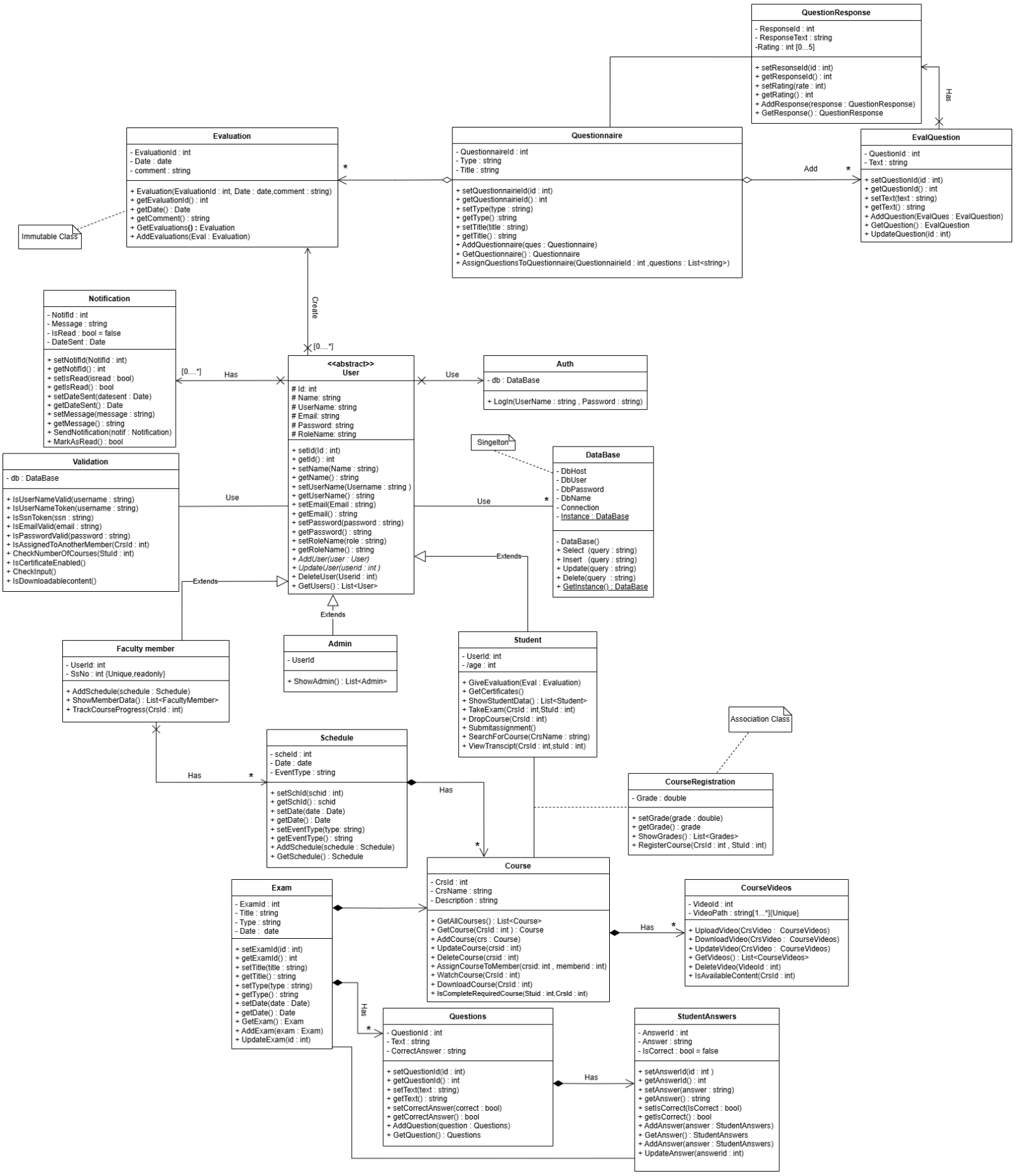


- **Name:** Singleton design pattern
- **Context:** The application needs a single, shared access point to manage database connections efficiently and consistently.
- **Problem:** How can we ensure that only one instance of the Database exists and is used throughout the application, while providing a global access to it?
- **Forces:**
  - The singleton object should be accessible to classes that need it.
  - The public constructor cannot guarantee that no more than one instance will be created.
  - Creating multiple database instances can lead to high resource consumption.
- **Solution:**
  - Having a private constructor to prevent direct instantiation and make sure no other class can create an instance.
  - A public static method that on its first call creates the only instance of the singleton class and stores it's reference in a static private variable for future access.
- **References:** Singleton design pattern was introduced by the "Gang of Four" in their book *Design Patterns: Elements of Reusable Object-Oriented Software*, published in 1994. This pattern has since become a fundamental concept in software design.



k) Class Diagram 3: The final version, after applying the design pattern(s) and any other modifications

Final Class Diagram



l) [1 Bonus Mark] Define a design smell, highlight a design smell in your design, and suggest how it can be avoided.

A design smell is a symptom in OO Software Design that indicates a problem in the structure of classes that makes the design harder to maintain and understand, it is like a warning that something needs restructuring or to be simplified, there are many examples of design smells like God Class, Duplicate Code, Shotgun Surgery, Data Clumps.

A design smell in our design is God Class: the User class in the Class Diagram is overloaded with too many methods and responsibilities such as managing authentication details, dealing with user info, methods related to evaluations and notifications, and UI support methods.

It can be avoided by splitting the methods and responsibilities across multiple classes for example: moving authentication methods to separate authenticationservice class, moving methods related to evaluations and notifications to separate services classes. This makes the User class more focused with better maintainability and more clarified.

m) [1 Bonus Mark] Read about Class Structuring Criteria and classify all the classes in your class diagram into one of the different categories of application classes. The general categories are an Entity Class, a Boundary Class, a Control Class, or an Application Logic Class. Some of those categories are further categorised.

1.Entity Classes: *These represent the core business objects in your system, typically corresponding to our system and persistent data objects*

use in class diagram :

- *QuestionResponse* (Response data model)
- Evaluation (Evaluation records)
- Questionnaire (Questionnaire structure)
- Notification (Notification messages)
- User (User profiles)
- *FacultyMember* (Academic staff data)
- Admin (Administrator data)
- Schedule (Event scheduling)
- *CourseRegistration* (Course enrollment records)
- Course (Course information)
- Exam (Exam details)
- *CourseVideos* (Course video resources)
- Questions (Assessment questions)
- *StudentAnswers* (Student response data)
- Student (Student profiles)

2.Boundary Classes: These classes are responsible for interaction with actors or external systems. They often handle input and output

use in class diagram :

- Auth (Authentication interface)
- Notification (Notification delivery interface)
- *DataBase* (Database connection handler)
- *And any class interaction with actor*

3.Control Classes: These classes coordinate between entity and boundary classes, controlling the flow of data and coordinating interactions. They often implement the application's business logic. In some categorizations

use in class diagram :

- DataEase (Data access operations controller)
- Validation (Input validation coordinator)
- CourseRegistration (Course enrollment logic)
- Evaluation (Evaluation process manager)

Control classes divide :

- **Transaction Control**: Manages **individual business transactions** or use cases — short-lived operations like course registration, exam submission, or answer evaluation.
- **Session Control**: Manages **longer workflows or sessions**, coordinating multiple steps or transactions, often representing user interactions over time.

4.Application Logic Classes: These classes encapsulate the logic of your application, typically implementing algorithms and business rules. They are often used by control classes to perform specific operations. In a more detailed classification

use in class diagram :

4.1 Data Access Layer

- DataEase (Implements CRUD operations***)***

4.2 Business Rules

- Validation (Username/email/password rules***)***

4.3 Service Classes

- *CourseVideos* (Video upload/download service)
- Exam (Exam management service)

Application Logic Classes divided into

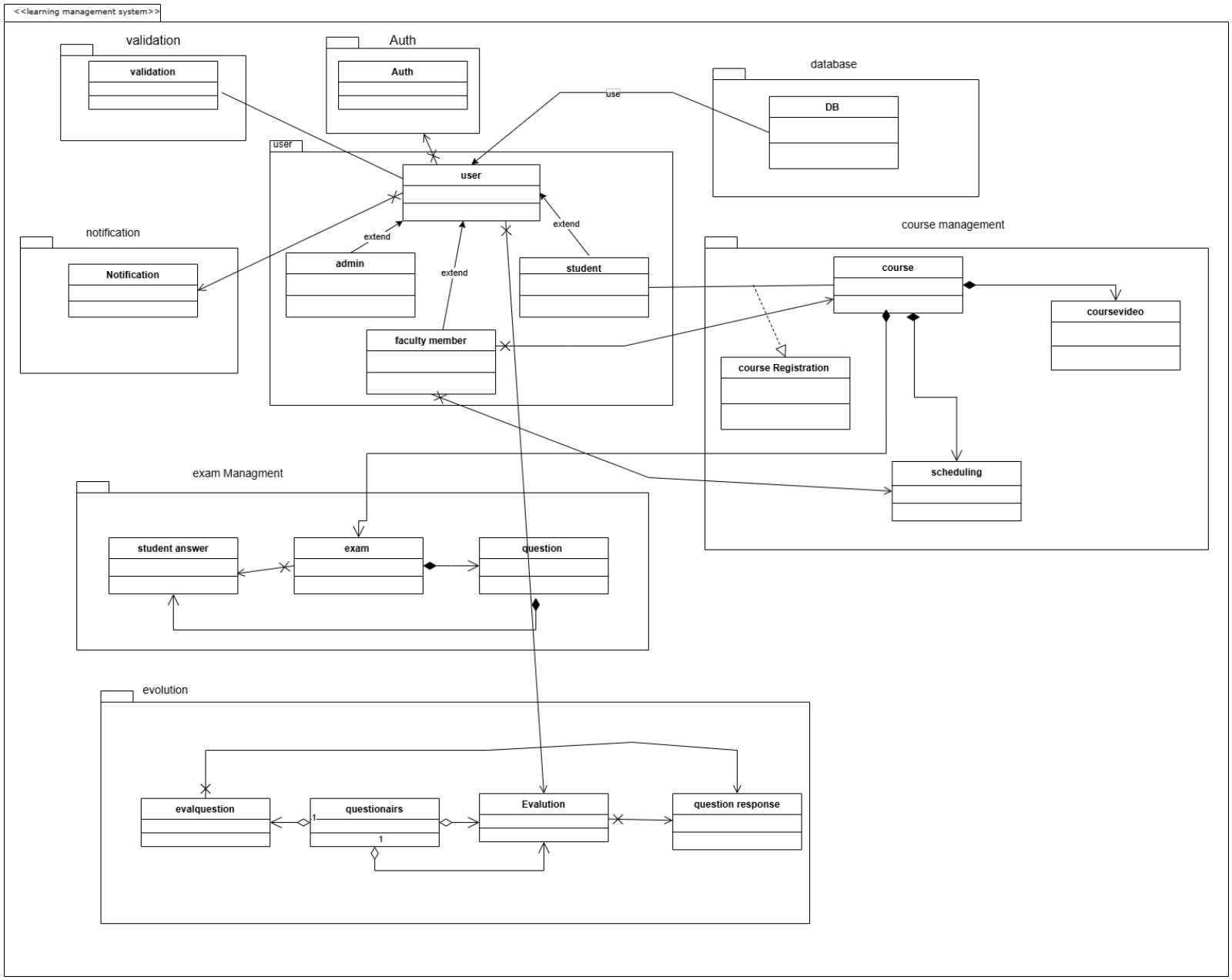
- ***Pure Fabrication Classes***
  *These are **not modeled after real-world concepts**, but are "invented" to achieve a better design (e.g., for cohesion, separation of concerns, reusability, or to avoid violating SRP or high coupling).*
- ***Low-Level Classes***
  *These classes typically perform **technical system tasks** – such as connecting to*

a database, managing sessions, or sending notifications – often ***used by higher-level application or domain logic*** .

n) Did you rely on Forks or Cascades in your interaction diagrams? Give an example of your choice and mention its advantage(s)/disadvantage(s).

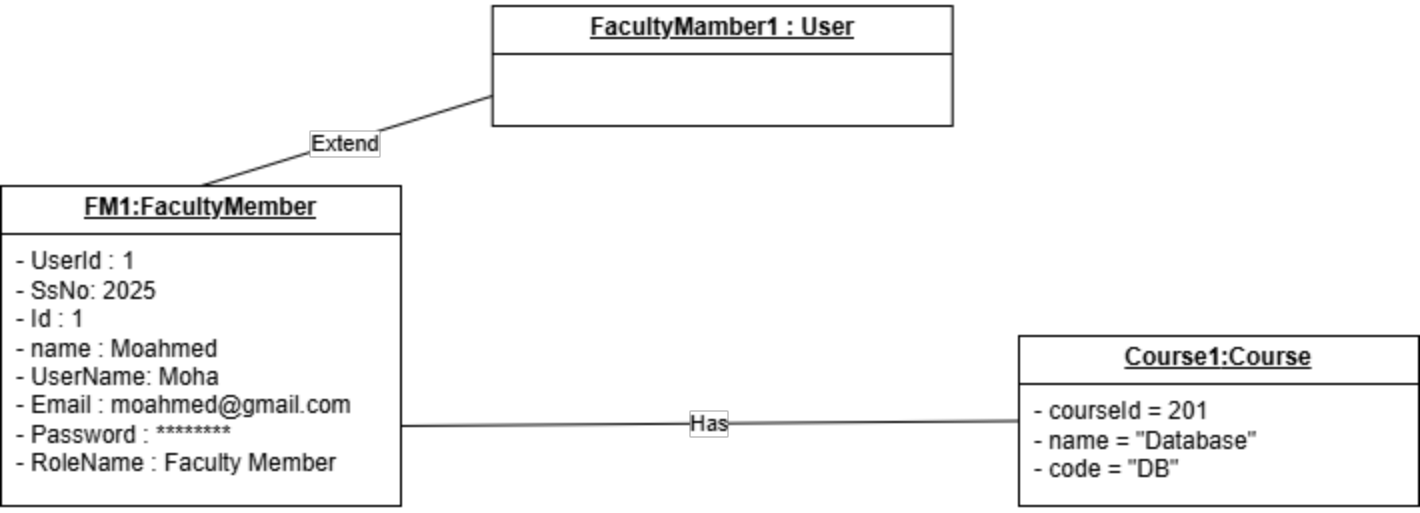
No we didnt rely on Forks or Cascades in interaction diagrams.

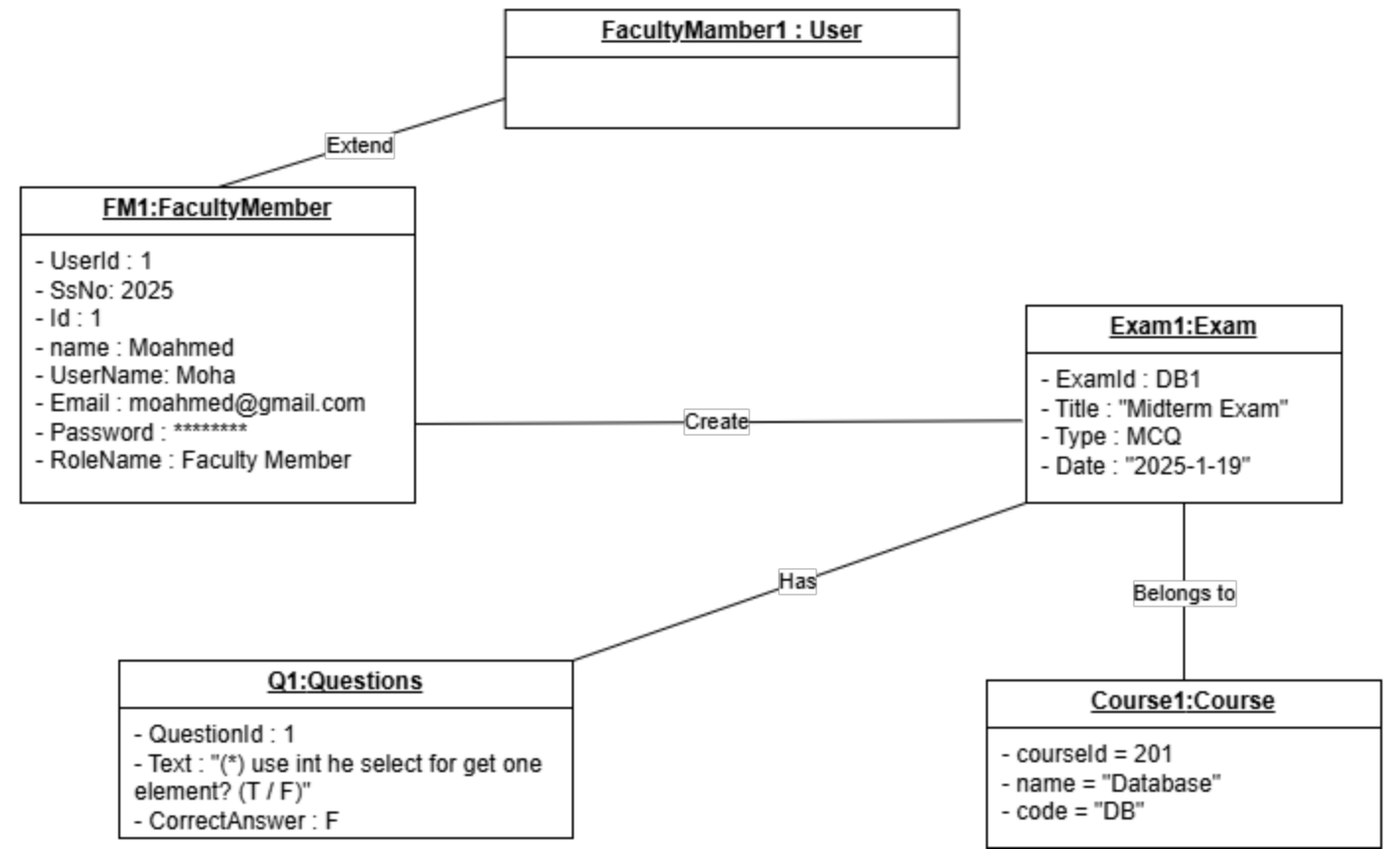
o) Package Diagram grouping relevant Classes into Packages.



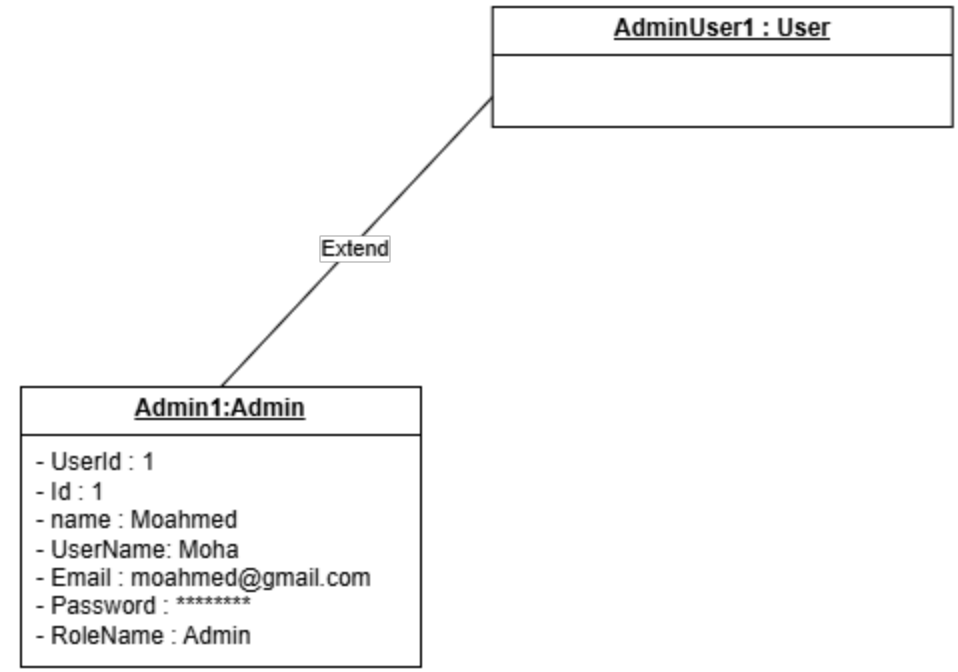
p) Object Diagrams

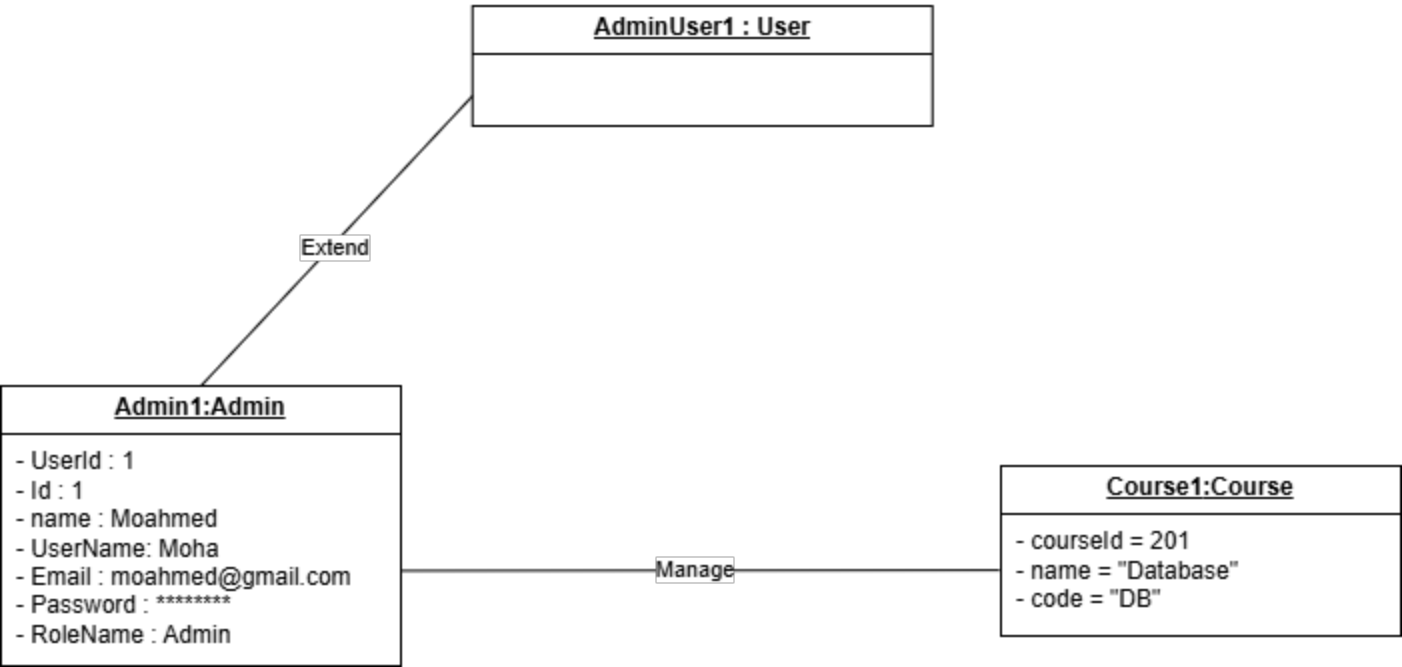
Use Case : Create Exam



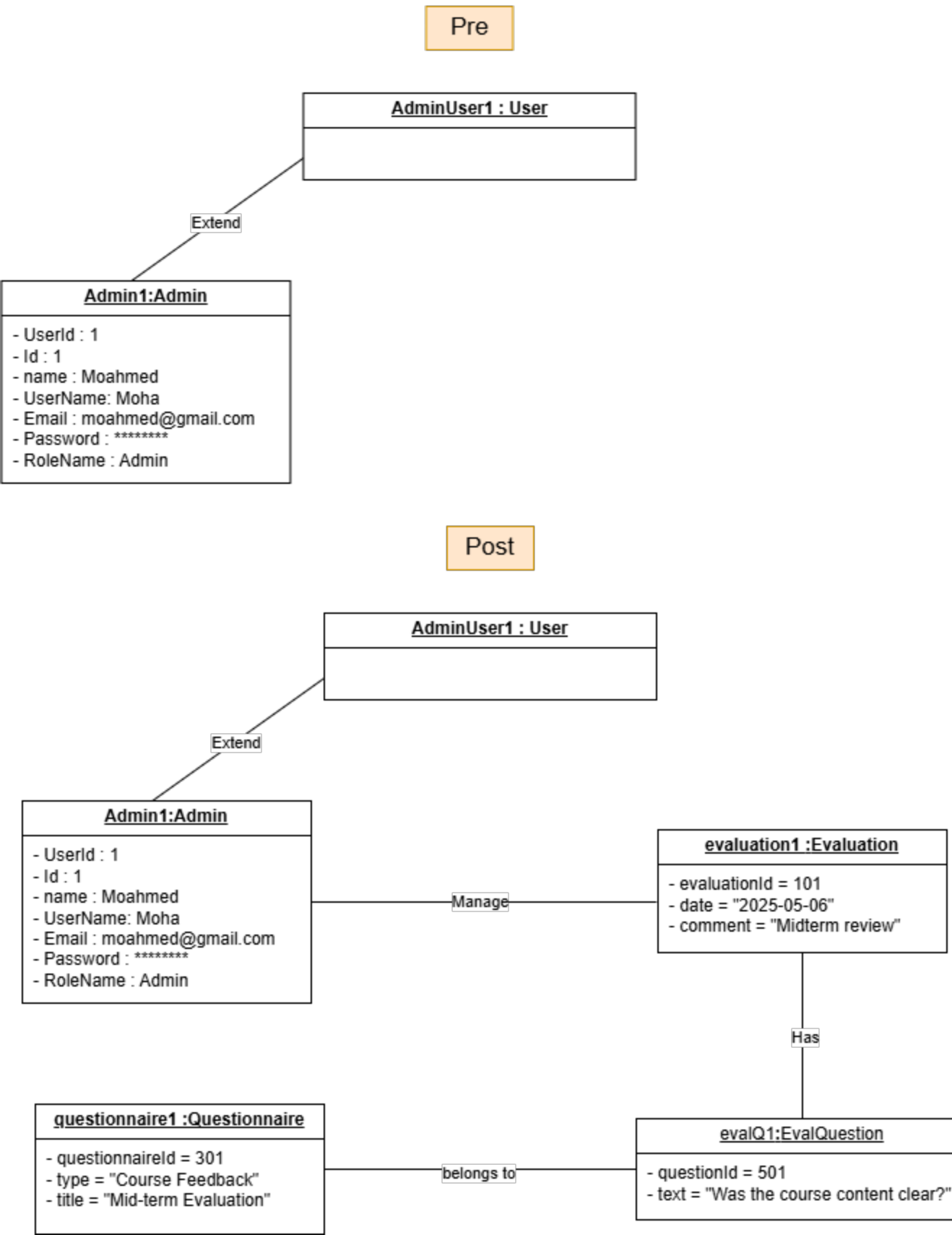


Use Case : Manage Course





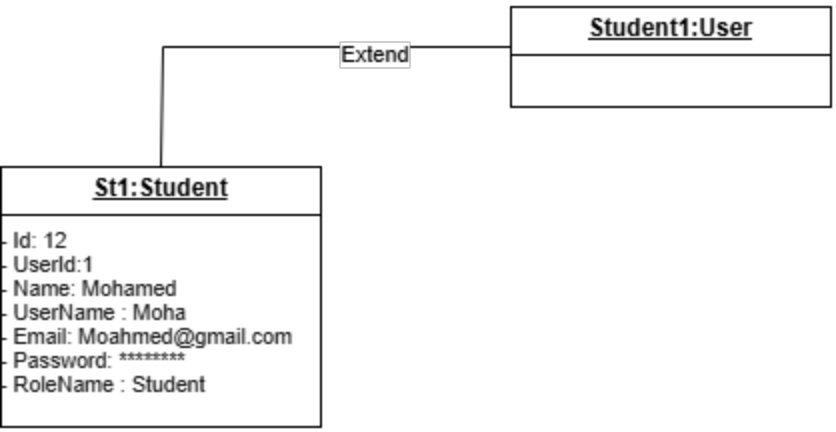
Use Cases : Manage Evaluation and Create Evaluations Questionnaire

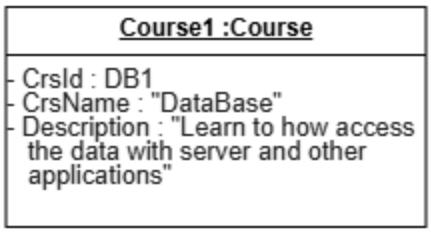


Use Case : Register Course

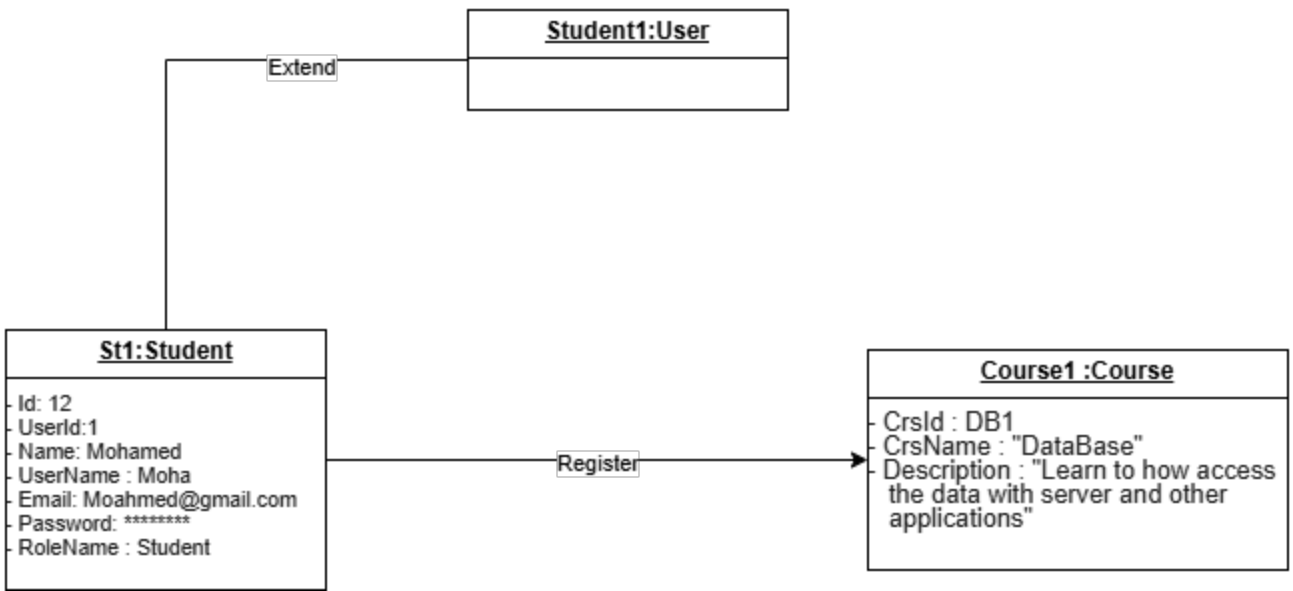
Pre

Pre :The student is logged into the system. and The registration period is open.
Post : Selected courses are registered and added to the student's schedule.
2. Course availability is updated.

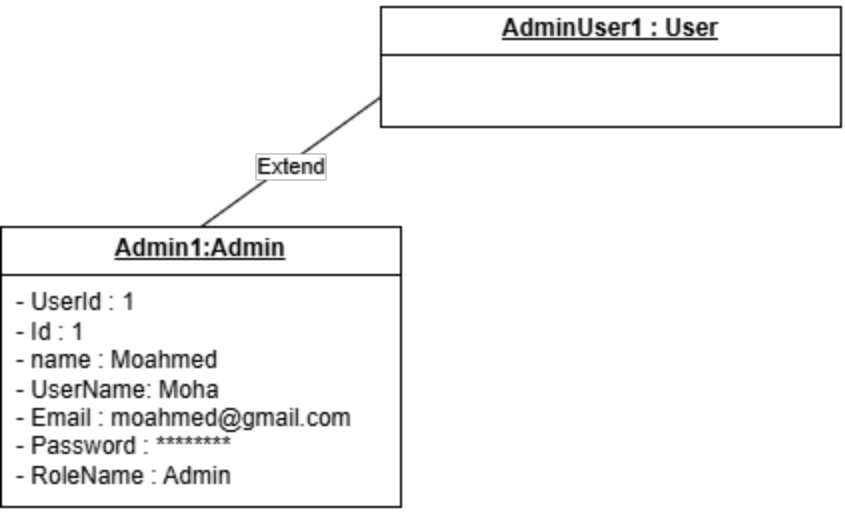


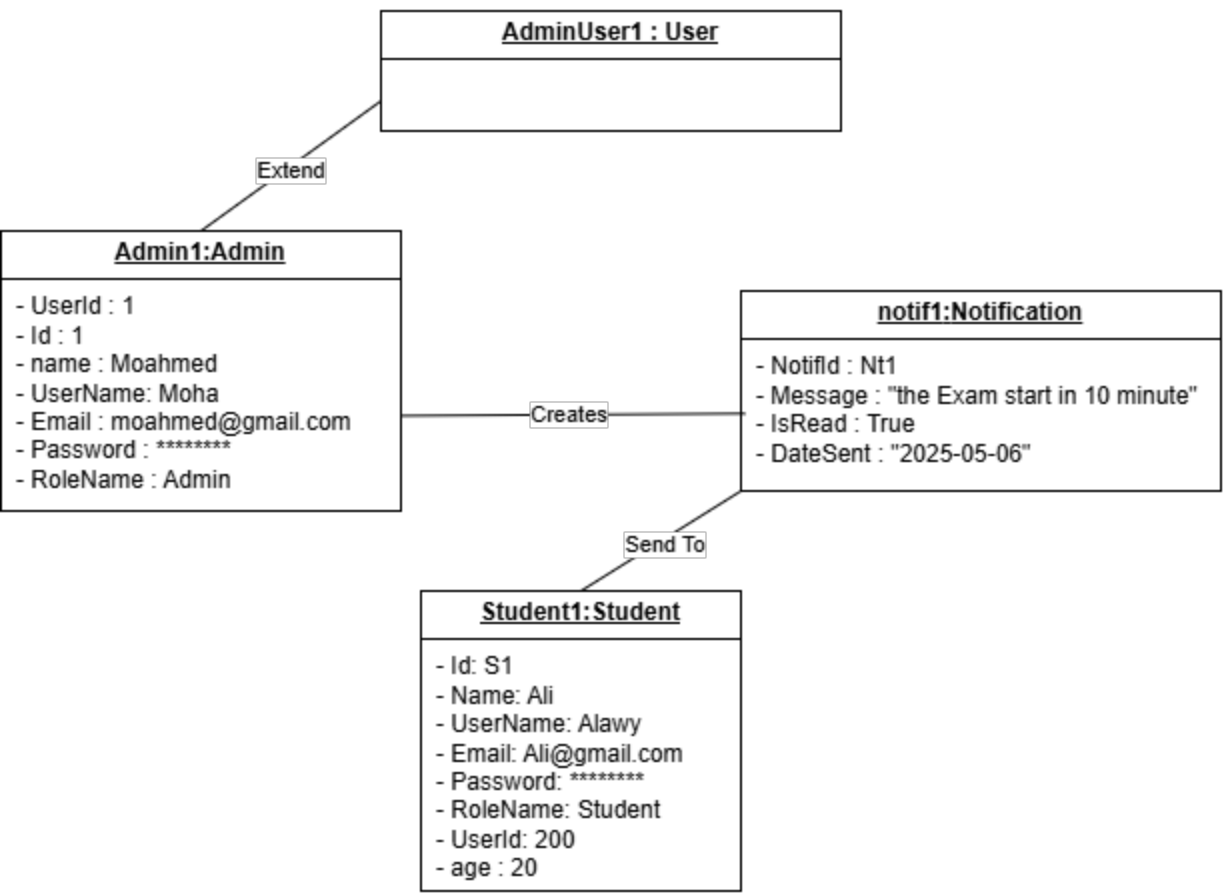


Post

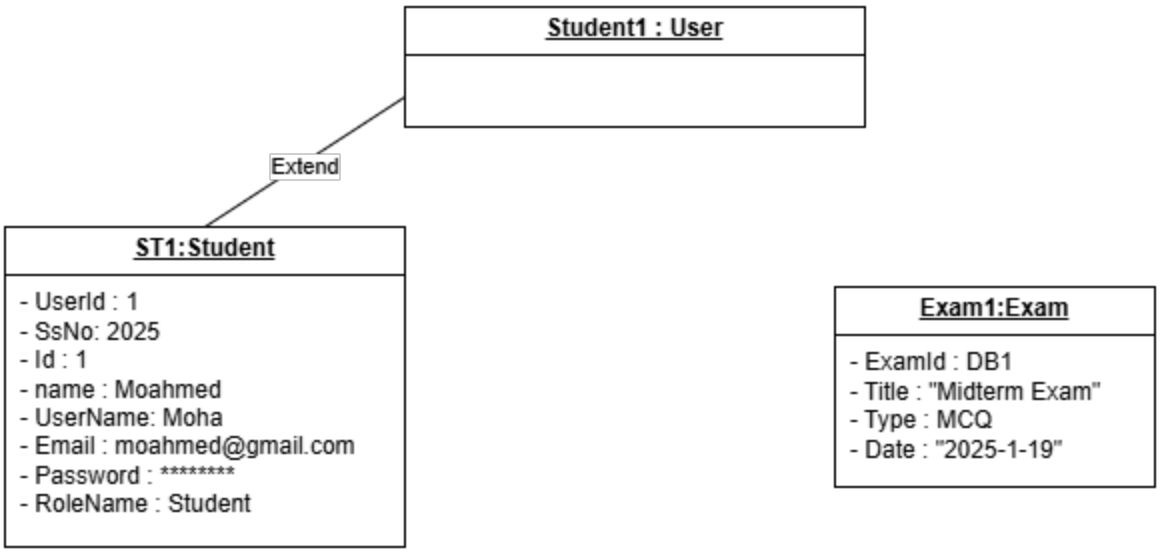


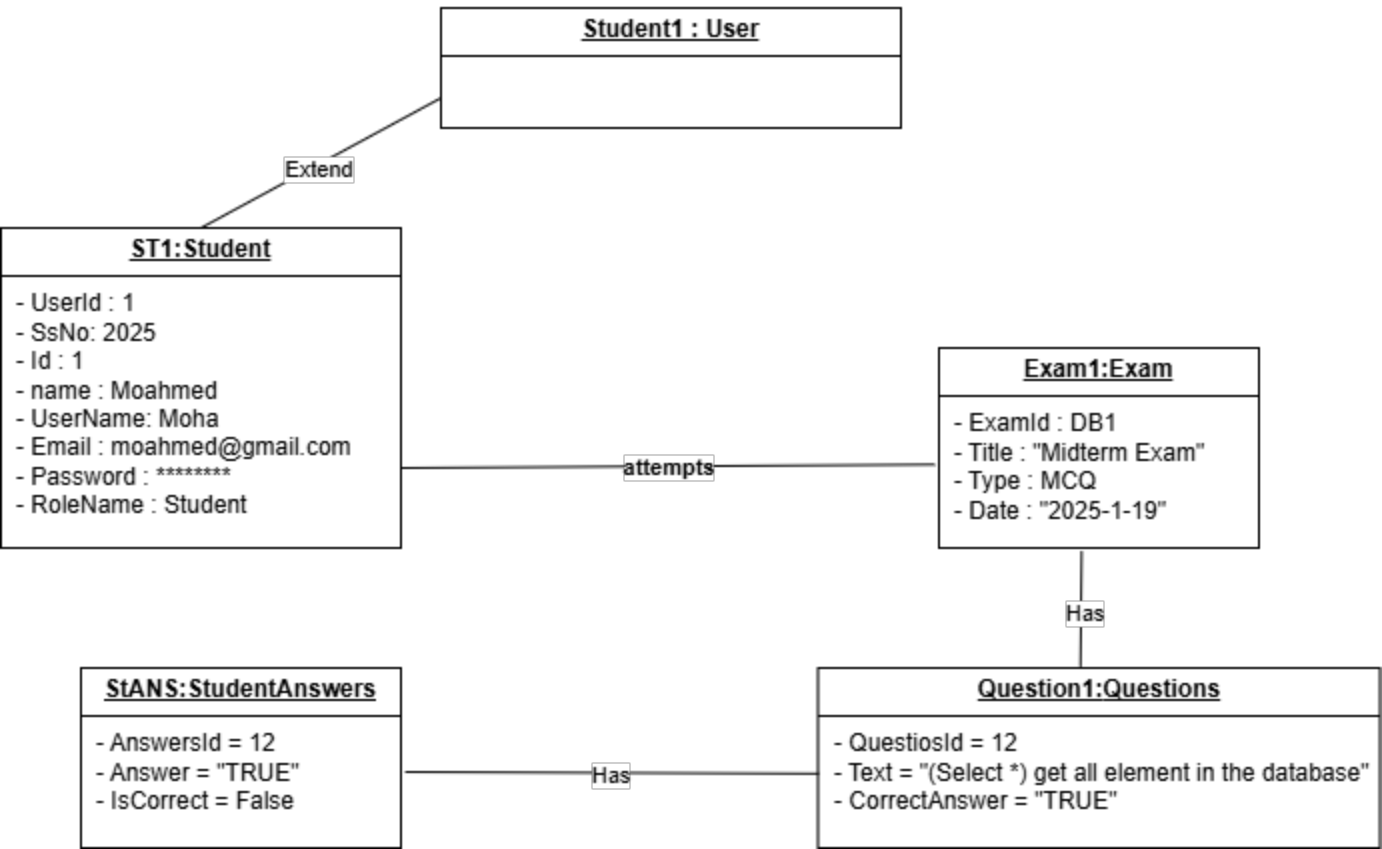
Use case : Send Notivications





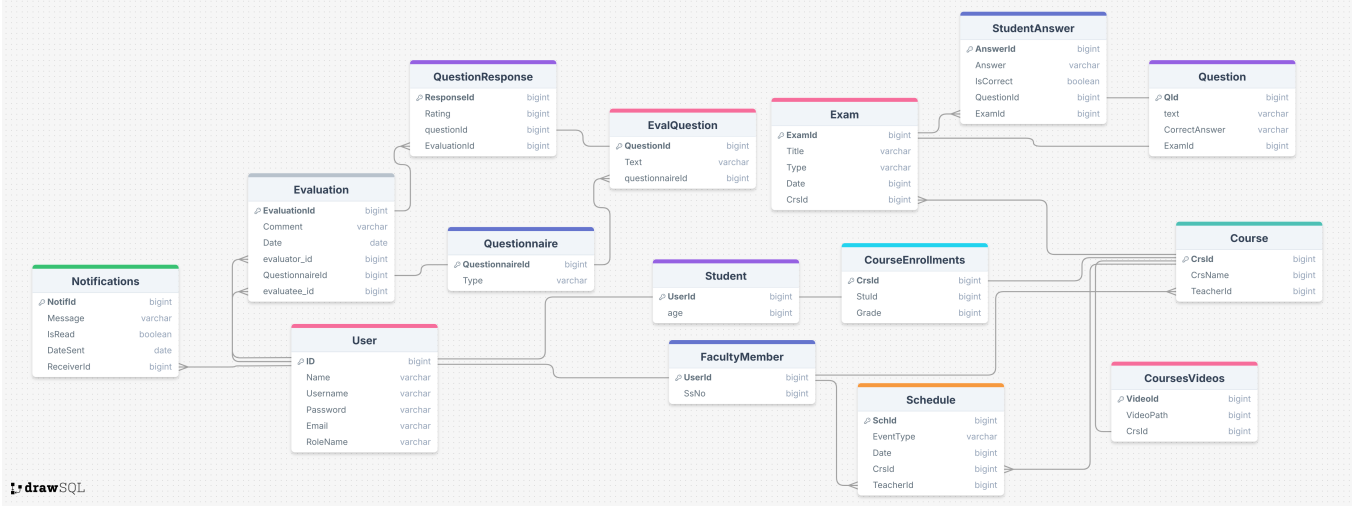
User Case : Take Exam



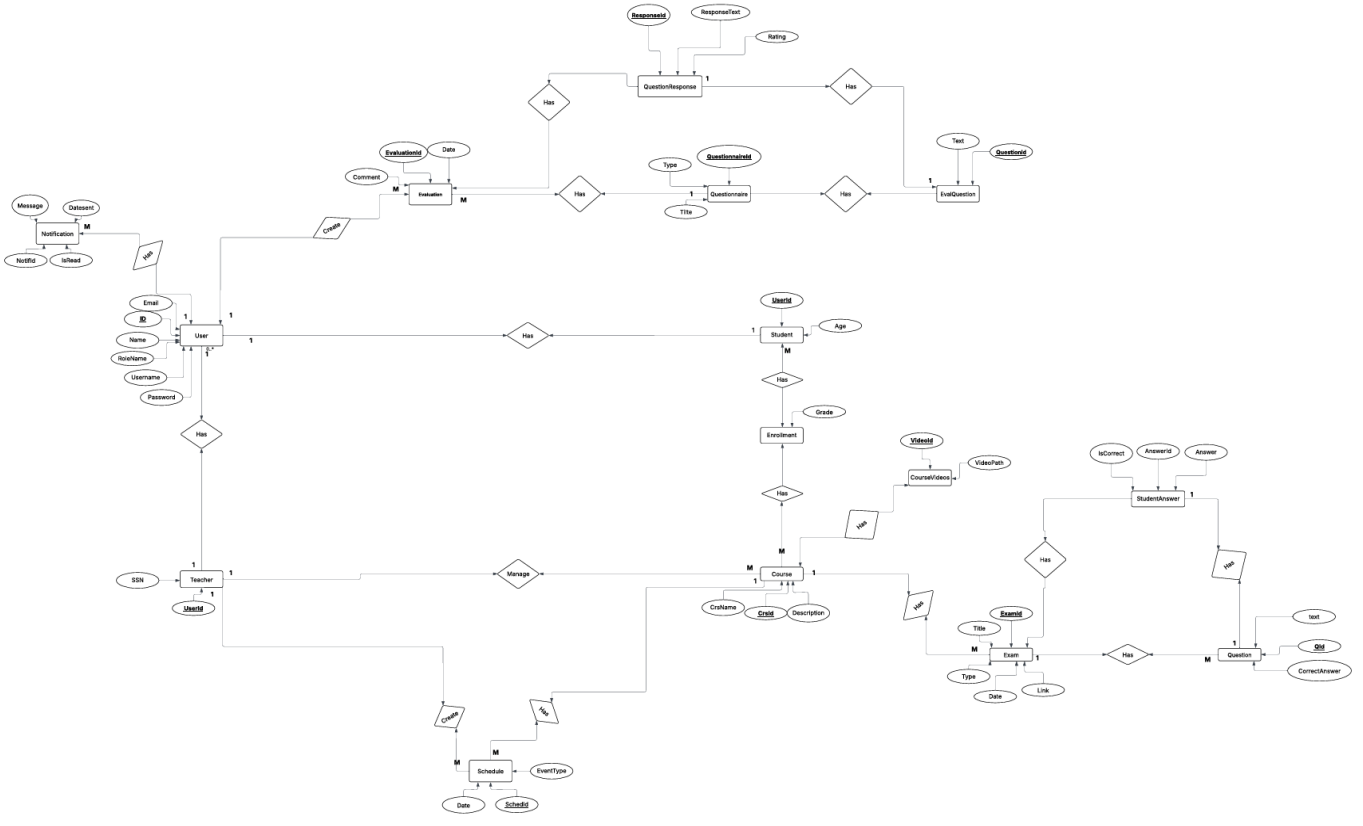


q) Database Specification

Tables



ERD



PART 4: Complexity & Testing

12) Are there pairs of Software Quality Factors that are not independent in your system? Give an example.

Usability and Performance: These two factors can often influence each other.

Usability refers to how easy it is for users to learn and use the system. In a digital healthcare system, this could involve factors like how intuitive the user interface is, how easy it is to navigate through different screens, and how clear the instructions are.

Performance refers to how well the system performs its functions. This could involve factors like how quickly the system responds to user inputs, how efficiently it processes data, and how reliably it operates.

If a system is designed to be highly performant, with fast response times and efficient data processing, it can enhance the usability of the system because users can complete their tasks quickly and smoothly. However, focusing too much on performance might lead to a complex user interface that can hinder usability.

On the other hand, if a system is designed with a strong focus on usability, with an intuitive and user-friendly interface, it might require more computational resources or lead to longer response times, which could impact performance.

Example:

Consider the Validation class:

- Centralizing all validation in one class might introduce extra function calls or conditional checks across the system.
- This could slightly degrade Performance, especially if validation rules are complex or called frequently in loops.

Improving Maintainability by adding abstraction and modularity (e.g., more helper classes like Validation)
Might reduce Performance due to overhead.

13) Calculate the LOC and CCM (Cyclomatic Complexity Metric) for the main functions in your system.

To clarify: LOC is every line of code in a piece of code (excluding lines that are empty, contains only comments, contains only closing brackets or html tags)

And CCM is starting with a base value 1 and adding 1 for each: decision point, logical operators, loops, exception handling

Manage Users (ManageUsers.php):

LOC: 256 lines including php backend logic (61) and HTML structure (195)

CCM: 8

Register Courses (studentEnrolCourses.php):

LOC: 130 lines including php backend logic (20) and HTML structure (110)

CCM: 5

Upload Material (UploadCourseMaterial.php):

LOC: 200 lines including php backend logic (45) and HTML structure (155)

CCM: 8

Create Evaluation Questions (AddQuestionnaire.php):

LOC: 126 lines including php backend logic (28) and HTML structure (98)

CCM: 4

View Feedback (ViewFeedbacks.php):

LOC: 202 lines including php backend logic (22) and HTML structure (180)

CCM: 6

Rate Faculty Member (studentFeedback.php):

LOC: 135 lines including php backend logic (30) and HTML structure (105)

CCM: 7

14) For the classes in your system, calculate all the following OO Complexity Metrics
a) WMC (Weighted Methods per Class)
b) DIT (Depth of Inheritance Tree)
c) NOC (Number of Children)
d) CBO (Coupling Between Objects)
e) RFC (Response for Class)
f) LCOM (Lack of Cohesion of Methods)

---

a) Weighted Methods per Class (WMC)

Equation:
WMC = Σ Ci where Ci = complexity of method i (assumed as 1 per method)

Class	**No. of Methods (WMC)**
User	10
Student	6
Admin	1
FacultyMember	4
Schedule	8
Course	9
Exam	10
Questions	9
StudentAnswers	6
CourseRegistration	4
CourseVideos	4
Evaluation	6
Questionnaire	8
EvalQuestion	6
QuestionResponse	6
Notification	5
Validation	6
Auth	2
Database	7

b) Depth of Inheritance Tree (DIT)

Definition: Maximum length from node to root.

Class	DIT
User	0
Student	1
Admin	1
FacultyMember	1
Schedule	0
Course	0
Exam	0
Questions	0
StudentAnswers	0
CourseRegistration	0
CourseVideos	0
Evaluation	0
Questionnaire	0
EvalQuestion	0
QuestionResponse	0
Notification	0
Validation	0
Auth	0
Database	0

---

c) Number of Children (NOC)

Definition: Number of immediate subclasses.

Class	NOC
User	3 (Student, Admin, FacultyMember)
Others	0

---

d) Coupling Between Objects (CBO)

Equation: **CBO = FanOut + FanIn**

Class	CBO Reasoning	CBO
User	Uses Validation, Notification	2
Student	Uses Evaluation, CourseRegistration, Course, Schedule	3
Admin	Uses Schedule	1
FacultyMember	Uses Schedule	1
Schedule	Uses Course, Exam	2
Course	Uses Questions	1
Exam	Uses Questions	1
Questions	Uses StudentAnswers	1
StudentAnswers	None	0
CourseRegistration	Uses Course	1
CourseVideos	None	0
Evaluation	Uses Questionnaire	1
Questionnaire	Uses EvalQuestion	1
EvalQuestion	None	0
QuestionResponse	None	0
Notification	None	0
Validation	Uses Database	1

Class	CBO Reasoning	CBO
Auth	Uses Database	1
Database	None	0

---

e) Response for Class (RFC)

Equation: **RS = M ∪ R**

Class	RFC Reasoning	RFC
User	10 (all own methods) + 2 (Notification, Validation)	12
Student	6 (own) + 3 (Eval, CourseReg, Schedule)	9
Admin	1	1
FacultyMember	4 + 1 (Schedule)	5
Schedule	8 + 2 (Course, Exam)	10
Course	9 + 1 (Questions)	10
Exam	10 + 1 (Questions)	11
Questions	9 + 1 (StudentAnswers)	10
StudentAnswers	6	6
CourseRegistration	4 + 1 (Course)	5
CourseVideos	4	4
Evaluation	6 + 1 (Questionnaire)	7
Questionnaire	8 + 1 (EvalQuestion)	9
EvalQuestion	6	6
QuestionResponse	6	6
Notification	5	5
Validation	6 + 1 (Database)	7
Auth	2 + 1 (Database)	3
Database	7	7

---

f) Lack of Cohesion of Methods (LCOM)

Equation: **LCOM = (P − Q) / (1 − Q)** where P = #pairs of methods not sharing attributes, Q = #pairs sharing at least 1 attribute

For simplicity and conservative estimate, assuming moderate cohesion:

Class	**LCOM (qualitative)**
User	Low
Student	Medium
Admin	High
FacultyMember	Medium
Schedule	Medium
Course	Medium
Exam	Low
Questions	Medium
StudentAnswers	Medium
CourseRegistration	Medium
CourseVideos	High
Evaluation	Medium
Questionnaire	Medium
EvalQuestion	Medium
QuestionResponse	Medium
Notification	Medium
Validation	Medium
Auth	Low
Database	Low

Note:

- WMC assumes method complexity = 1 per method.
- CBO is approximated based on associations, uses/arrows.
- RFC includes own methods + directly invoked methods/classes.
- LCOM is qualitatively estimated due to lack of explicit attribute-method mapping in diagram.

15) Considering Report for at least 6 main functions in your system. For each function, consider path testing by determining a set of test cases (the value of the function's parameters) such that each path through the function is executed at least once. Whit e-Box Testing, generate a Unit-Testing Test

Function 1: Add Question

Purpose: Teacher adds a question to a specific exam.

Paths Identified:

Path 1: Successful addition.

Path 2: Database connection failure.

Path 3: Query execution failure.

Path 4: Missing question text or correct answer.

Path 5: Invalid exam ID.

Test Cases:

Test case	Question Text	Correct Answer	Exam ID	Expected output	Actual output
1	"What is PHP?"	"A scripting language"	101	"Question added successfully!"	"Question added successfully!"
2	""	"Correct"	102	"Failed to add question."	"Failed to add question."
3	"What is PHP?"	""	103	"Failed to add question."	"Failed to add question."
4	"What is PHP?"	"A scripting language"	-1	"Failed to add question."	"Failed to add question."
5	"What is PHP?"	"A scripting language"	"abc"	"Failed to add question."	"Failed to add question."
6	"What is PHP?"	"A scripting language"	104	"Database connection error."	"Database connection error."

Function 2: Delete User

Purpose: Admin deletes a user data from the database.

Paths Identified:

Path 1: Successful deletion.

Path 2: Database connection failure.

Path 3: Deletion query execution failure.

Path 4: Invalid user ID.

Path 5: User ID not found.

Test Cases:

Test case	User ID	Expected output	Actual output
1	101	(Successful Deletion)	(Successful Deletion)
2	-1	"Error in Query"	"Error in Query"
3	“abc”	"Error in Query"	"Error in Query"
4	0	"Error in Query"	"Error in Query"
5	999	(User Not Found)	(User Not Found)
6	102	"Database connection error."	"Database connection error."

Function 3: Get transcript

Purpose: Retrieves the registered courses and their grades from the database.

Paths Identified:

Path 1: Successful retrieval of transcript data.

Path 2: Database connection failure.

Path 3: Query execution failure.

Path 4: Invalid student ID.

Path 5: No grades recorded.

Path 6: Student has registered courses with grades available.

Path 7: Student has registered courses but no grades available.

Test Cases:

Test case	Student ID	Courses Registered	Grades Available	Expected output	Actual output
1	101	Yes	Yes	Array of courses with grades	Array of courses with grades
2	102	Yes	No	Empty array (no grades available)	Empty array
3	-1	No	Not available	false (Invalid ID)	false
4	"abc"	No	Not available	false (Invalid ID)	false
5	0	No	Not available	false (Invalid ID)	false
6	103	No	Not available	Empty array (no courses)	Empty array
7	104	Yes	Yes	"Database connection error."	"Database connection error."

Function 4: Get courses assigned to member.

Purpose: Retrieves the list of courses assigned to a specific faculty member from the database.

Paths Identified:

Path 1: Successful retrieval of assigned courses .

Path 2: Database connection failure.

Path 3: Query execution failure.

Path 4: Invalid faculty ID.

Path 5: No courses assigned.

Path 6: Non-existent faculty ID.

Test Cases:

Test case	Faculty ID	Courses Assigned	Expected output	Actual output
1	201	Yes	Array of assigned courses	Array of courses
2	202	No	Empty array (no courses assigned)	Empty array
3	-1	Not available	false (invalid ID)	false
4	“abc”	Not available	false (invalid ID)	false
5	0	Not available	false (invalid ID)	false
6	999999	No	Empty array (non-existent faculty)	Empty array
7	203	Yes	Error in Query	Error in Query (database connection failure)

Function 5: Send Notification.

Purpose: Retrieves all notifications for a given user ID from the database.

Paths Identified:

Path 1: Successful retrieval of notifications.

Path 2: Database connection failure.

Path 3: Query execution failure.

Path 4: Invalid user ID.

Path 5: User ID with no notifications.

Path 6: Non-existent user ID.

Test cases:

Test case	User ID	Notifications Available	Expected output	Actual output
1	101	Yes	Array of notifications	Array of notifications
2	102	No	Empty array (no notifications)	Empty array
3	-1	Not available	Error at query (invalid ID)	Error at query
4	“abc”	Not available	Error at query (non numeric id)	Error at query
5	0	Not available	Error at query (invalid ID)	Error at query
6	999999	No	Empty array (non-existent user)	Empty array
7	303	Yes	Error at query (database connection issue)	Error at query

Function 6: Get calendar.

Purpose: Retrieves the calendar of events for a student based on their registered courses.

Paths Identified:

Path 1: Student has more than 0 registered courses.

Path 2: Student has 0 registered courses.

Path 3: Database connection failure.

Path 4: Query execution failure.

Path 5: Invalid student ID.

Path 6: Non-existent student ID.

Test cases:

Test case	Student ID	Number of Courses	Expected output	Actual output
1	101	3	Array of calendar events	Array of events
2	102	0	false (error message: Must Number of Courses More than 0)	false (error msg)
3	-1	Not available	false (invalid ID)	false (error msg)
4	“abc”	Not available	false (invalid ID)	false (error msg)
5	0	Not available	false (invalid ID)	false (error msg)
6	999999	0	false (non-existent student, no courses)	false (error msg)
7	103	2	Error in Query (database connection failure)	Error in Query

16) Considering Black-Box Testing, generate a Functionality System-Testing Test Report for at least 6 main functions in your system. For each function, consider boundary testing by determining a set of test cases (the value of the function’s parameters) from the extreme ends or boundaries between partitions of the input values.

BlackBoxtesting

1.AddStudent Function.

Description: The "AddStudent" function is responsible for registering a new Student by the Admin.

in the system by inserting their details into the "Student" table. It takes four parameters: "Username", "name", "email", and "password".

Function	Input Parameters	Expected Result
1. AddStudent	Username = "Jo" Name="jo" Email="jo@gmail.com" , Password="Abc12"(invalid username)	Reject: "username must at least 3 letters"
	Username = "Joe" Name="jo" Email="jo@gmail.com" , Password="Abc12"(invalid username)	Rejected: The username is already in use by someone else.
	Username = "Jon" Name="" Email="jo@gmail.com", Password="Abc12" ,(invalid name)	Reject : must have at least 1 char
	Username = "Jon" Name="jo" Email="jo@gmail.com", Password="abc12" ,(invalid password)	Reject : the first char must be capital.
	Username = "Jon" Name="jo" Email="jo@gmail.com", Password="Abc" ,(invalid password)	Reject: Must have a number.
	Username = "Jon" Name="jo" Email="jo@.com", Password="Abc12" ,(invalid email)	Reject: Invalid format
	Username = "Jon" Name="jo" , Email="jo@gmail.com", Password="Abc12" .	Accept

2.AddExam Function.

Description: The "AddExam" function is responsible for adding a Exam by the Faculty member.

in the system by inserting their details into the "Exam" table. It takes four parameters: "title", "crsId", "Date", and "type".

Function	Input Parameters	Expected Result
1. AddExam	Title= " " ,type = post , date =2025/12/5 , CrsId = 1 . (title empty)	Reject: "Name must at least 3 letters"
	Title = "quiz" , type = post , date =2024/12/5 CrsId = 1 . (date is invalid)	Reject: the date is invalid.
	Title = "quiz" , type = 123 , date =2024/12/5 CrsId = 1 . (type is invalid)	Reject :Type mustn`t be number
	Title = "quiz" , type = "" , date =2024/12/5 CrsId = 1 . (type is invalid)	Reject : type must have at least 3
	Title= "123 " ,type = post , date =2025/12/5 , CrsId = 1 . (title empty)	Reject :title mustn`t be number
	Title= "123 " ,type = post , date =2025/12/5 , CrsId = 1 . (title empty)	Accepted.

3.Evaluate co-teacher Function.

Description: The " Evaluate co-teacher " function is responsible for adding evaluation by the Faculty member.
in the system by inserting their details into the "Evaluation" table. It takes four parameters: "evaluatee_Id", "evaluator_Id", "Date", and "Comment".

Function	Input Parameters	Expected Result
1.Evaluate co-teacher	evaluatee_Id=4(navigate the evaluatee) ,evaluator_ID=12(take from database), date=2025/6/6,comment="mlmsflsfmd"	Accepted
	evaluatee_Id=4 ,evaluator_ID=12, date=2024/6/6,comment="mlmsflsfmd" (invalid date)	Reject: The date cant be in the past.
	evaluatee_Id=4 ,evaluator_ID=12, date=2025/5/13,comemnt="mlmsflsfmd" (invalid date)	Reject: The date cant be in the same day you are submit the evaluation.
	evaluatee_Id=4 ,evaluator_ID=12, date=2025/6/6,comment=" " (invalid comment)	Reject: the comment must be at least 10 chars.

4.uploadCoursevideo

Description: The "uploadCoursevideo " function is responsible for upload Video by the Faculty member.
in the system by inserting their details into the "coursevideos" table. It takes two parameters: "course_Id", "targetfile".

Function	Input Parameters	Expected Result
1. UploadVideo	Course_Id=5(navigate the course), Target file="mkkk.jpj" (invalid target file)	Reject : the target file should be vedio.
	Course_Id=5(navigate the course), Target file="mkkk.mp4"	Accept

5.AddCourse

Description: The "AddCourse " function is responsible for add New course by the Admin and Faculty member.
in the system by inserting their details into the "courses" table. It takes two parameters: "course_name", "description".

Function	Input Parameters	Expected Result
1. AddSCourse	Course_name ="database" Description="" .(invalid description)	Reject : description must be at least 10 chars.
	Course_name =" " Description="mnbhfasdcee" . (invalid course_name)	Reject : name must be at least 4 chars.
	Course_name ="database " Description="mnbhfasdcee" .	Accept

6.AddFeedback

Description: The " AddFeedback " function is responsible for add feedback by the Admin and Faculty member.
in the system by inserting their details into the "evaluation" table. It takes two parameters: "faculty_ID", " comment".

Function	Input Parameters	Expected Result
1. AddSFeedback	Submit with all fields empty	Reject .
	Submit with faculty selected, no comments	Reject : comment must be at least 10 chars.
	Submit with faculty selected, comments = "mdjsuwlkchh"	Accept