NORTH SOUTH UNIVERSITY

School of Engineering & Physical Sciences

Project Name:

OpenEdu: A Comprehensive Open-Source Learning Platform

Submitted To:

Department of Electrical & Computer Engineering (ECE)
Course Instructor: AKM Iqtidar Newaz [IQN]
Course: Software Engineering
Course ID: CSE327
Section: 8
Semester: Spring 2025

Submitted By:

Group Member's Name	ID
Moontaha Rawshan	2221035642
Mohammad Ishzaz Asif Rafid	2221370642
Abir Mahmood Rifat	2132677642
Mujtaba Muhammad Abrar	2122265642

OpenEdu: A Comprehensive Open-Source Learning Platform

Project Overview:

OpenEdu is an innovative online platform designed to streamline the management of educational content, user roles, and collaborative learning experiences. Inspired by GitHub's structured repository-based workflow, this LMS enables students, instructors, and administrators to interact efficiently in a structured digital learning environment. The system incorporates role-based permissions and organized educational resources such as lectures, videos, and notes.

Project Objectives:

The primary goal of this LMS is to provide an intuitive and scalable solution for online education by offering:

- A structured approach to **content management** with lectures, videos, and notes.
- **Role-based access control**, ensuring different levels of permissions for users.
- A centralized platform for **faculty, moderators, and administrators** to manage courses and users effectively.
- **Request-based user management** for handling appointments and content approvals.
- **Efficient organization of departments, courses, and faculties** under a structured hierarchy.

End Users:

The system is designed to accommodate multiple categories of users:

1. **Regular Users (Students)** – Can access course content, submit requests, and interact with the available resources.
2. **Faculty (Instructors)** – Responsible for managing course materials, lectures, videos, and notes.
3. **Moderators** – Review and approve faculty and content requests to maintain platform quality.

4. **Admins** – Manage course creation, faculty approvals, and oversee the overall functionality of the LMS.
5. **Masters (Super Admins)** – Oversee the entire system, appoint administrators, and manage department structures.

Functional and Non-Functional Requirements:

Functional Requirements:

1. User Management:

- Users can be assigned roles (Regular User, Faculty, Moderator, Admin, Master).
- Users can request access to specific functionalities (e.g., content approval, appointment requests).
- Moderators and admins can manage approval workflows for user requests.

2. Course and Content Management:

- Courses can be created and assigned to faculties.
- Regular Users can upload lectures, videos, and notes.
- Students can access and interact with available course materials.

3. Request Handling System:

- Users can submit different types of requests (e.g., content approval, appointment requests).
- Moderators and administrators can manage and process these requests.

4. Role-Based Access Control:

- Regular Users can view course materials.
- Regular Users can create content.
- Moderators can approve/reject requests related to faculty and content.
- Admins can oversee courses and faculty appointments.
- Masters (Super Admins) can manage high-level operations such as- department structuring.

5. Department and Faculty Management:

- Admins can organize courses, and faculties under a structured hierarchy.
- Masters can appoint administrators and manage department structures.

Non-Functional Requirements:

1. Product Requirements:

- Performance: The system must handle many concurrent users (e.g., students, faculty, and admins) without significant latency.

- Reliability: The system should have an acceptable failure rate, with a maximum downtime of 1 hour per month.

- Usability: The system should be intuitive and easy for all user roles (students, moderators, admins, and masters).

- Portability: The system should be accessible across different devices (desktop, mobile, tablet) and operating systems (Windows, macOS, Linux, iOS, Android).

- Maintainability: The system should be easy to maintain, with clear documentation and modular code structure to facilitate updates and bug fixes.

2. Organizational Requirements:

- Process Standards: The system must adhere to the organization's policies and procedures for content management and user roles.

- Implementation Requirements: The system should be developed using Python programming.

- Delivery Requirements: The system and its documentation must be delivered within 16 April 2025.

3. External Requirements:

- Interoperability: The system should interact seamlessly with external systems (e.g., student information systems).

- Legislative Requirements: The system must comply with data protection laws to ensure user data privacy and security. Users can't upload any illegal material.

- Ethical Requirements: The system should ensure that all content and interactions are ethical and acceptable to users and the general public (e.g., no offensive or inappropriate content).

- Security: The system must implement robust security measures to protect user data and prevent unauthorized access (e.g., encryption, role-based access control).

- Data Integrity: The system must ensure data integrity, with mechanisms to prevent data corruption or loss (e.g., regular backups and transaction logs).

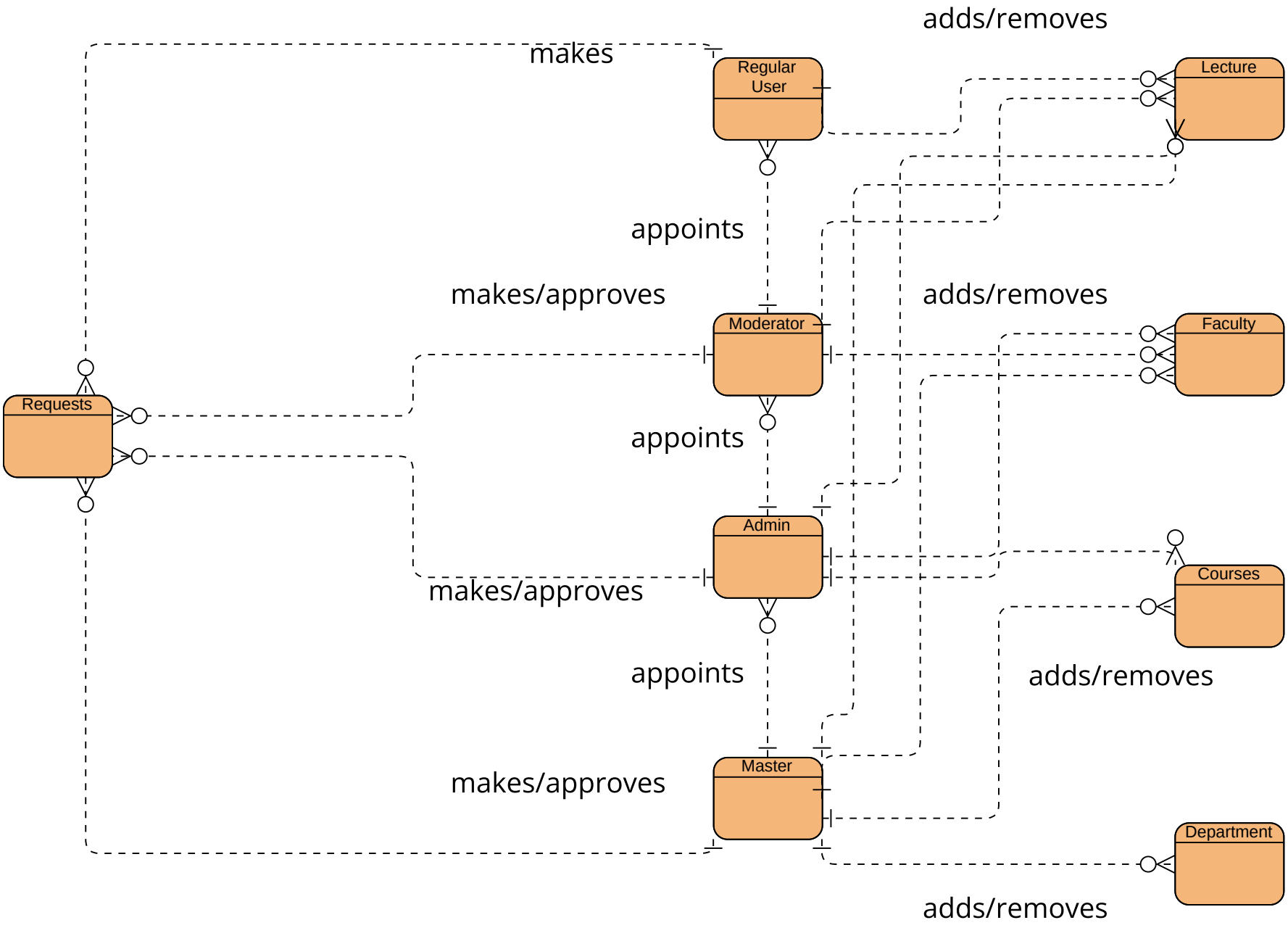
- User Experience: The system should provide a consistent and responsive user experience across all devices and browsers.

Therefore, **OpenEdu** is a structured and scalable approach to online education, offering seamless interaction among students, instructors, and administrators. By implementing **hierarchical user roles, structured content management, and efficient request-handling**, this system provides a complete digital learning experience that ensures academic integrity and accessibility for all users.

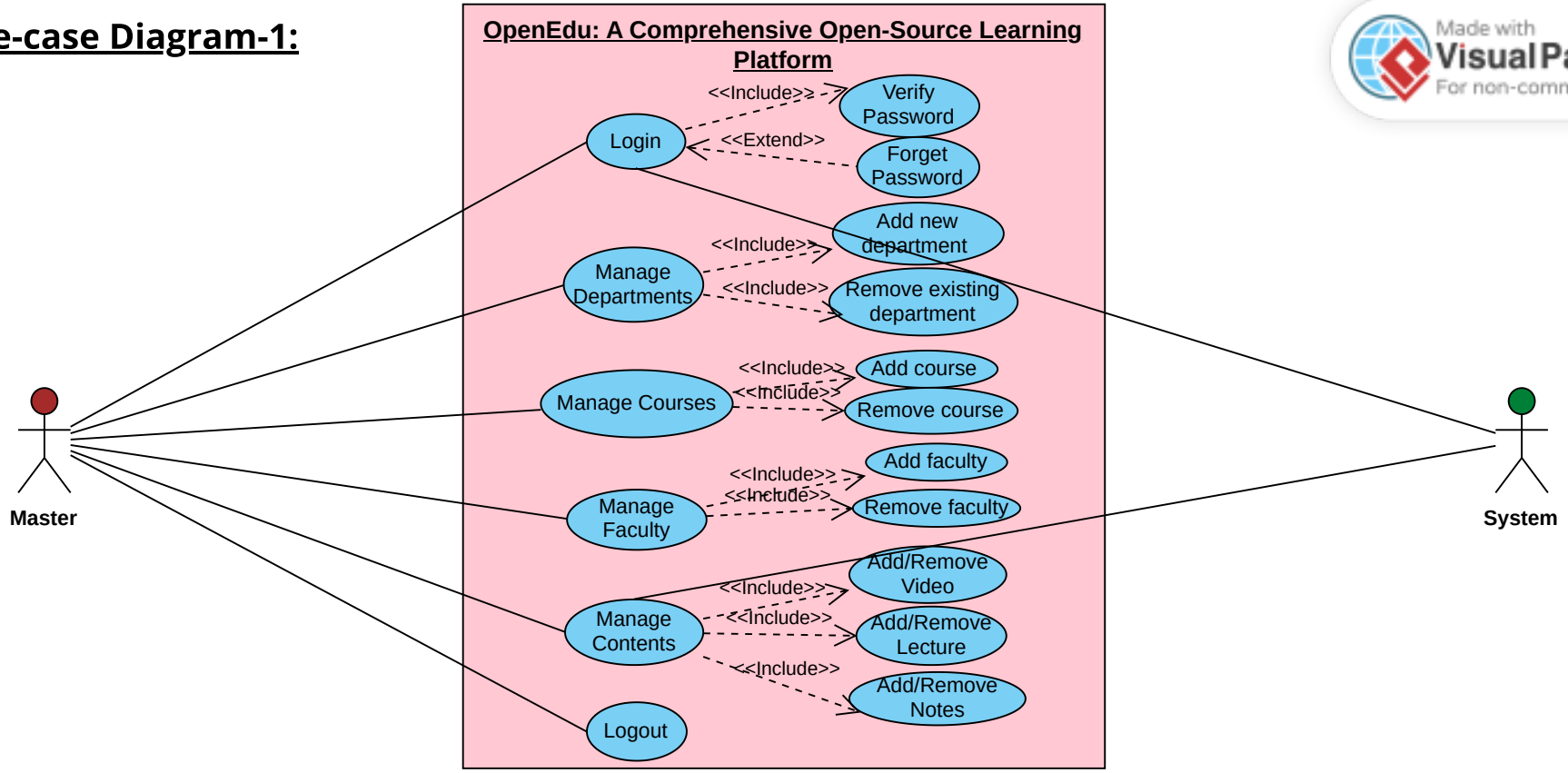
GitHub Repository Link:

https://github.com/EzazAsif/OpenEdu-A-Comprehensive-Open-Source-Learning-Platform

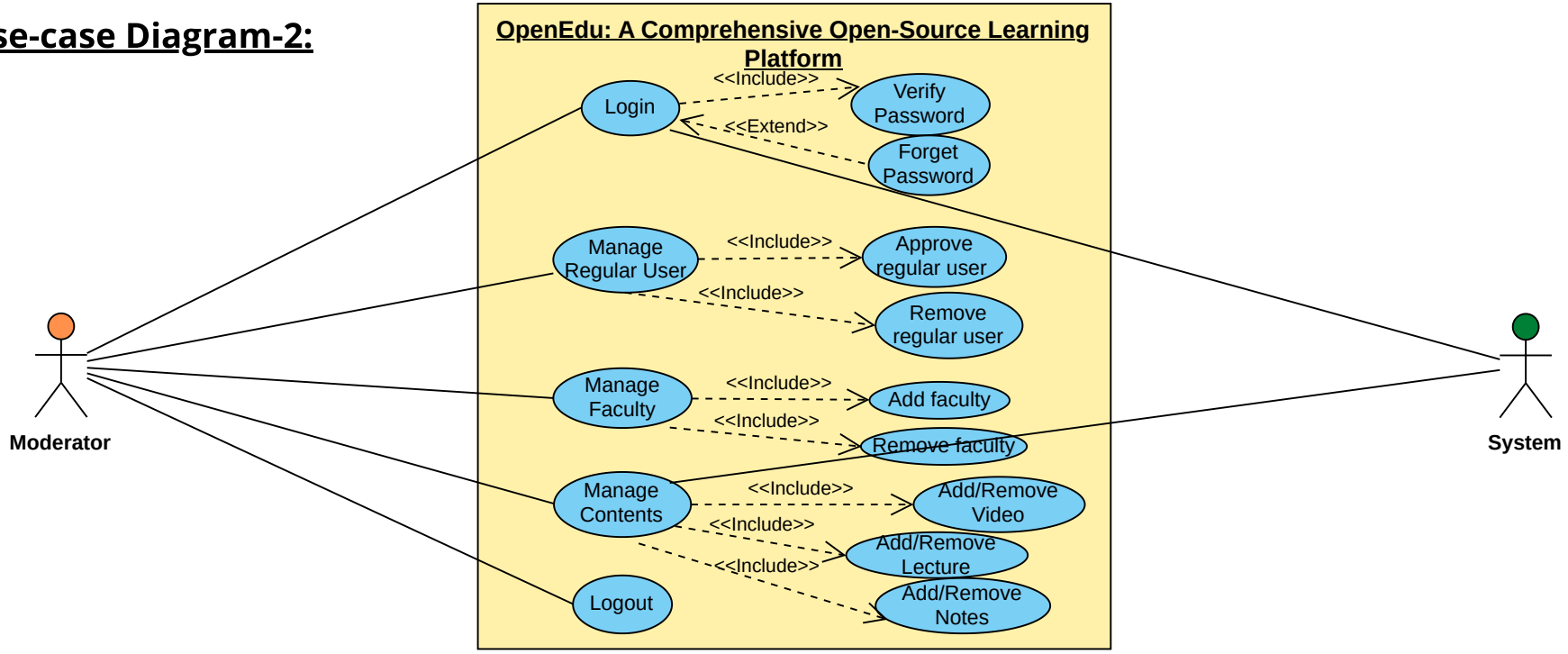
Entity-Relationship (ER) Diagram:



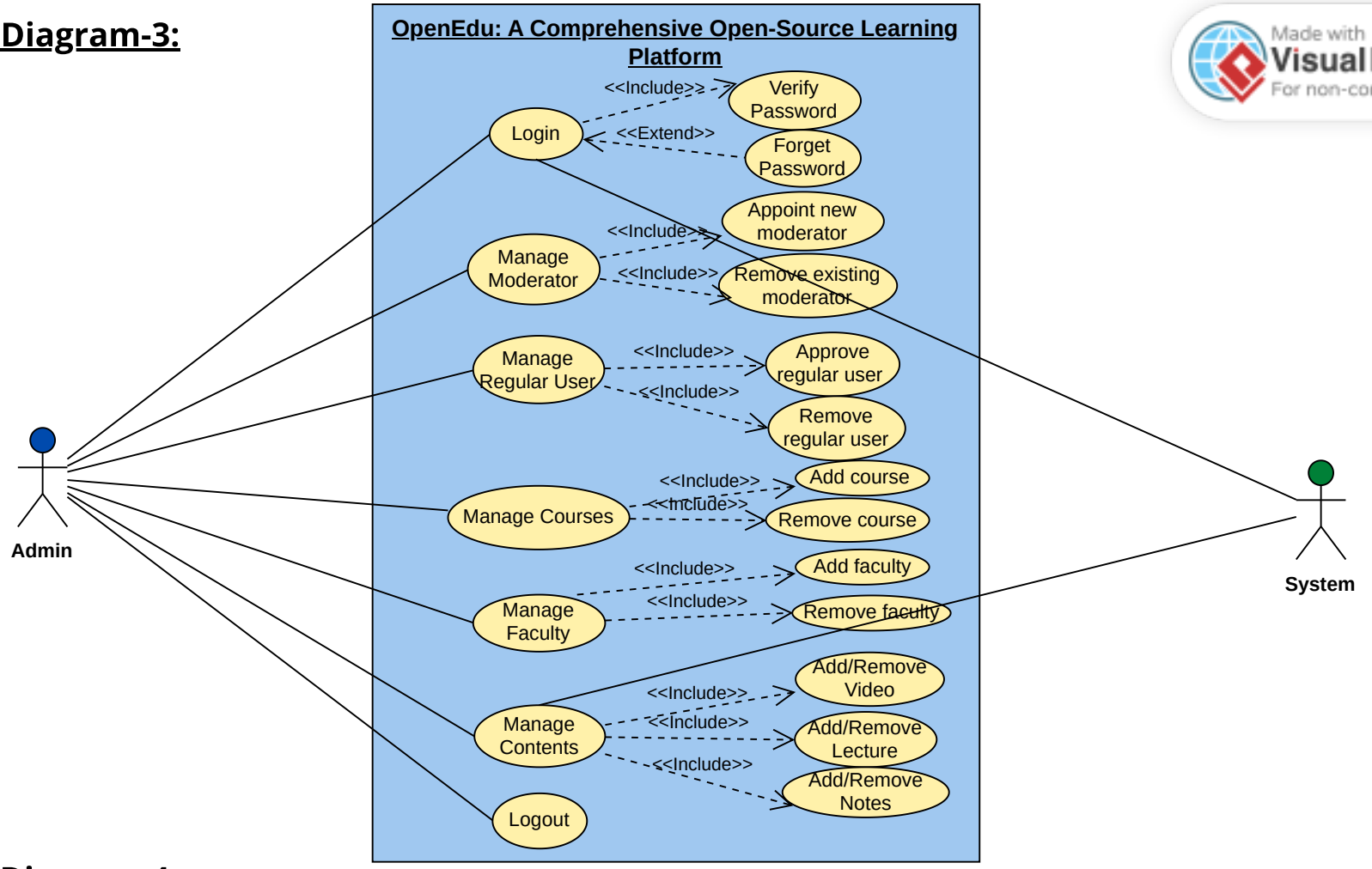
Use-case Diagram-1:



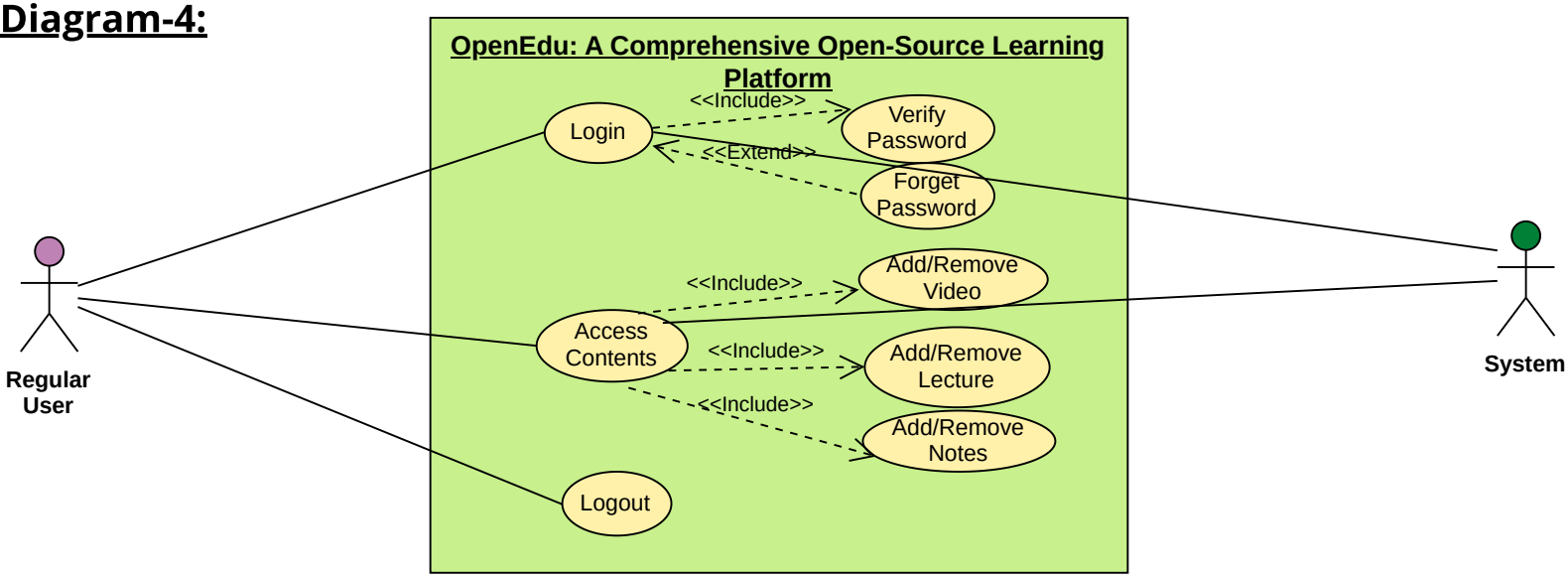
Use-case Diagram-2:



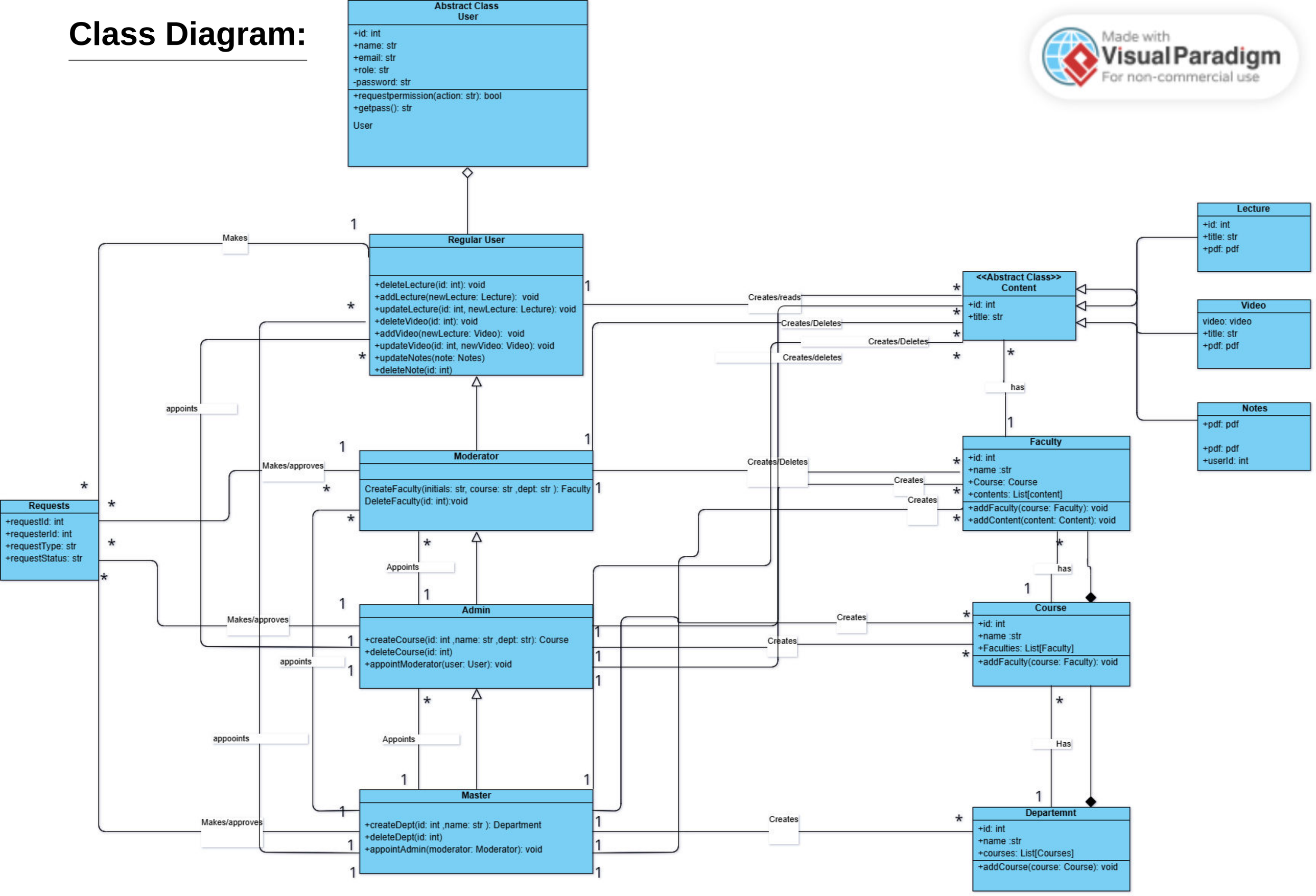
Use-case Diagram-3:



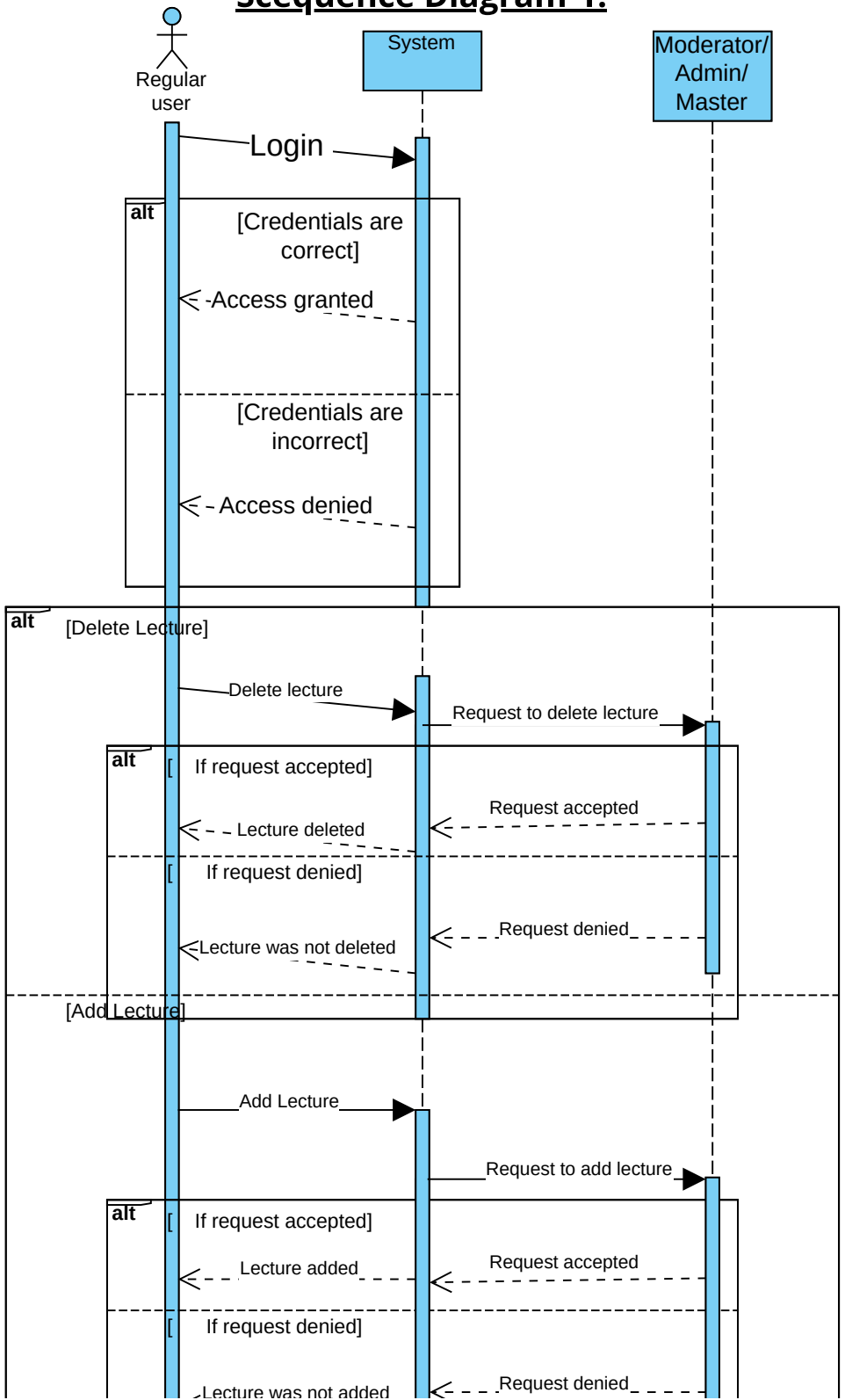
Use-case Diagram-4:

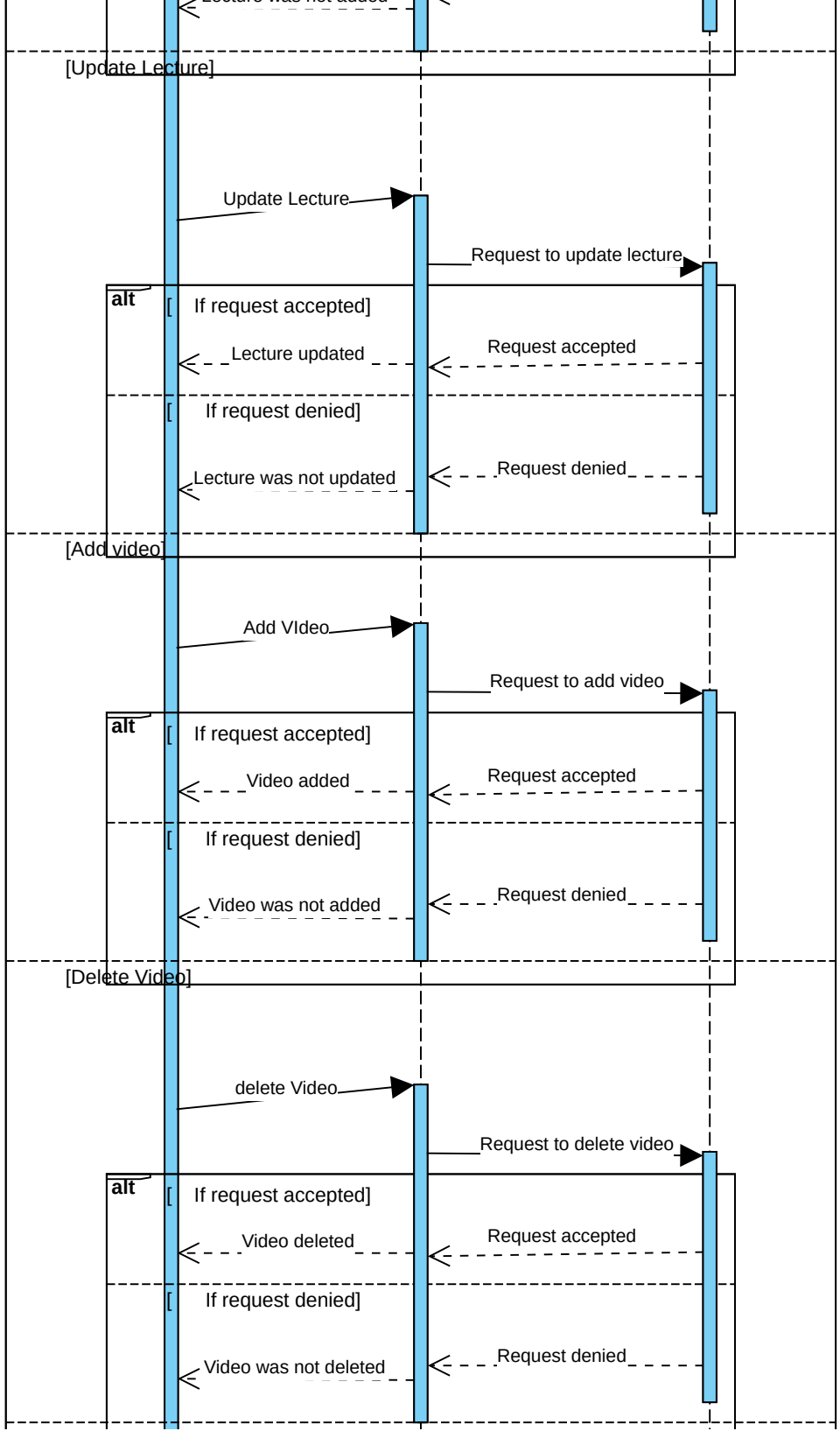


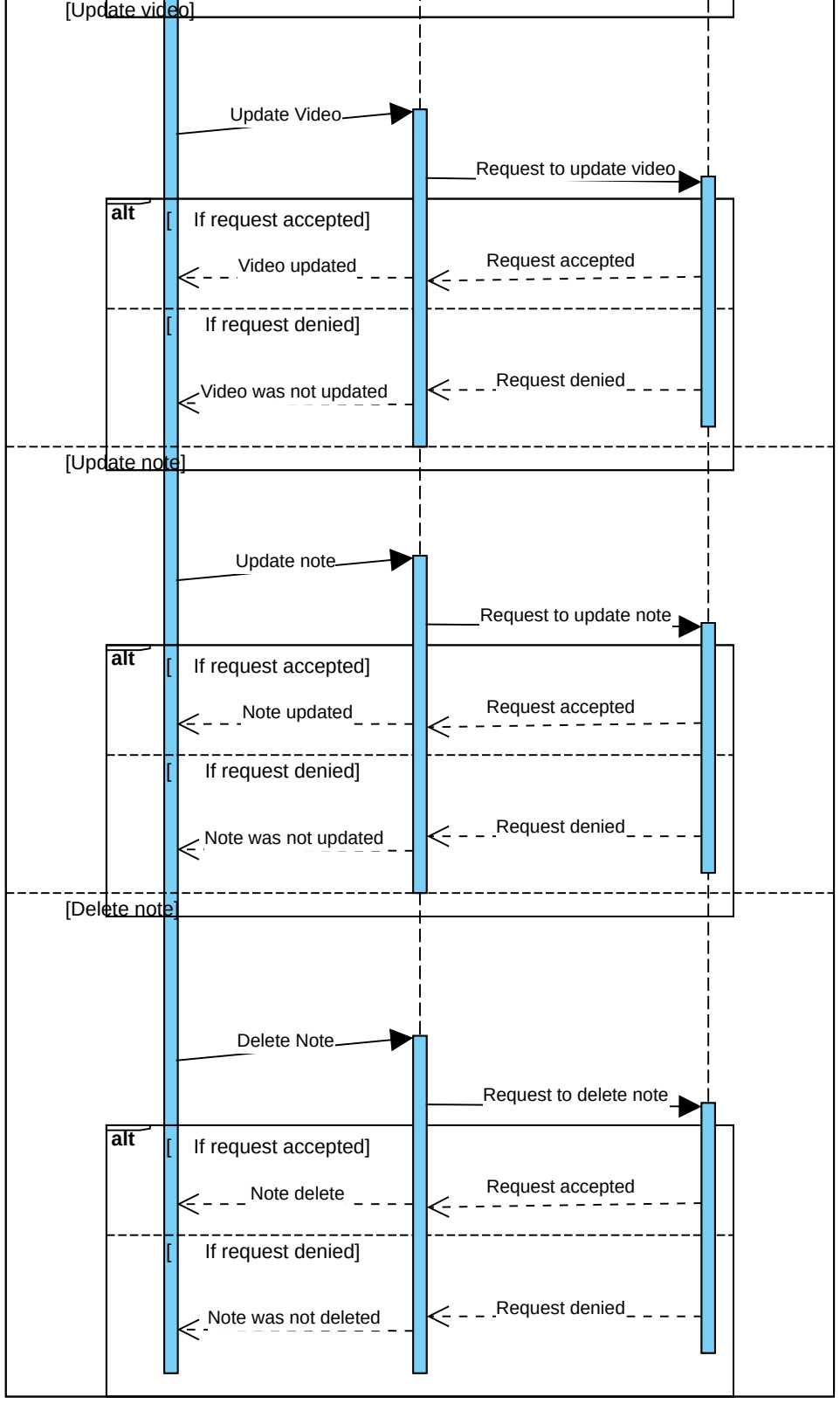
Class Diagram:



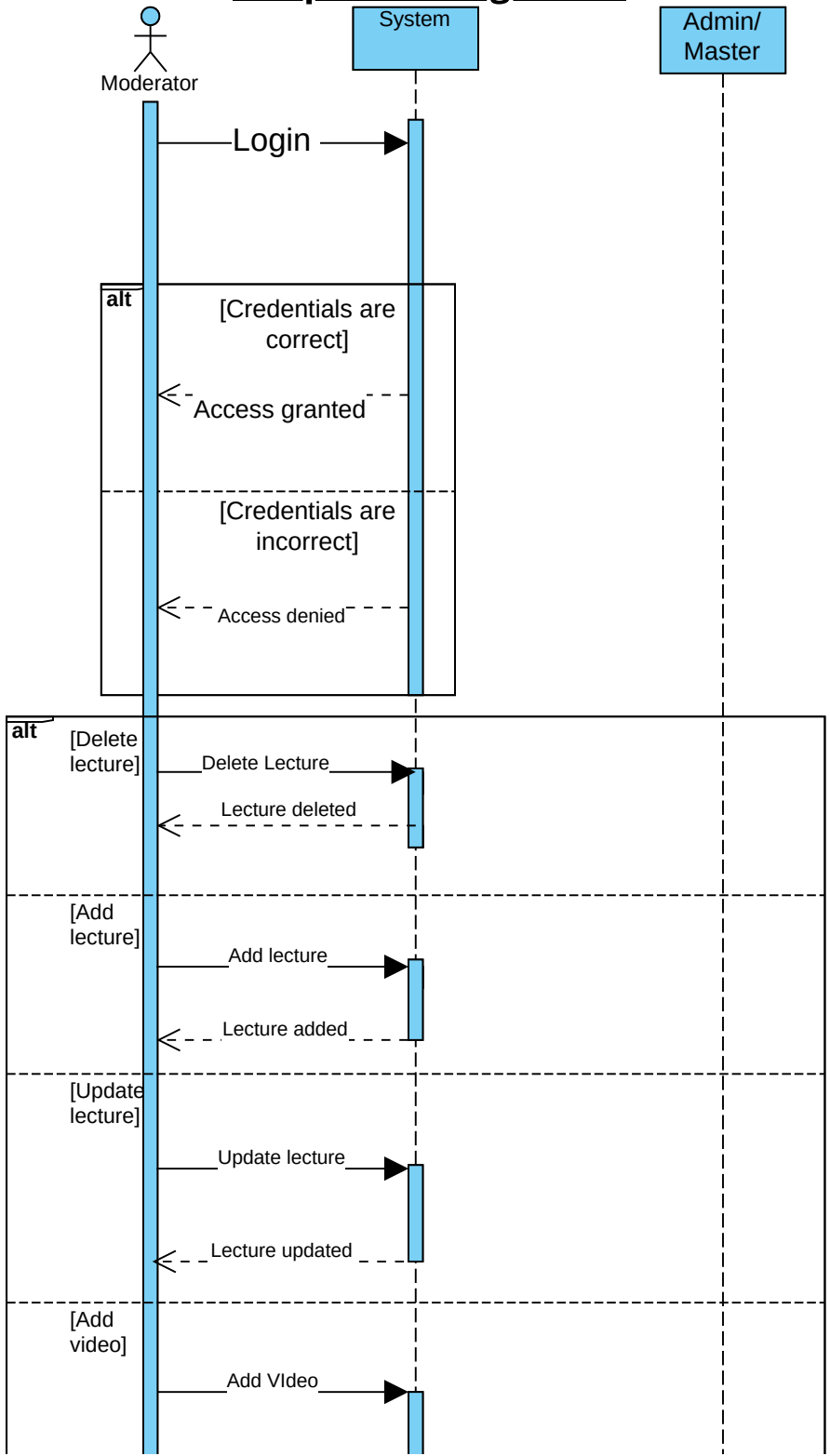
Scequence Diagram-1:

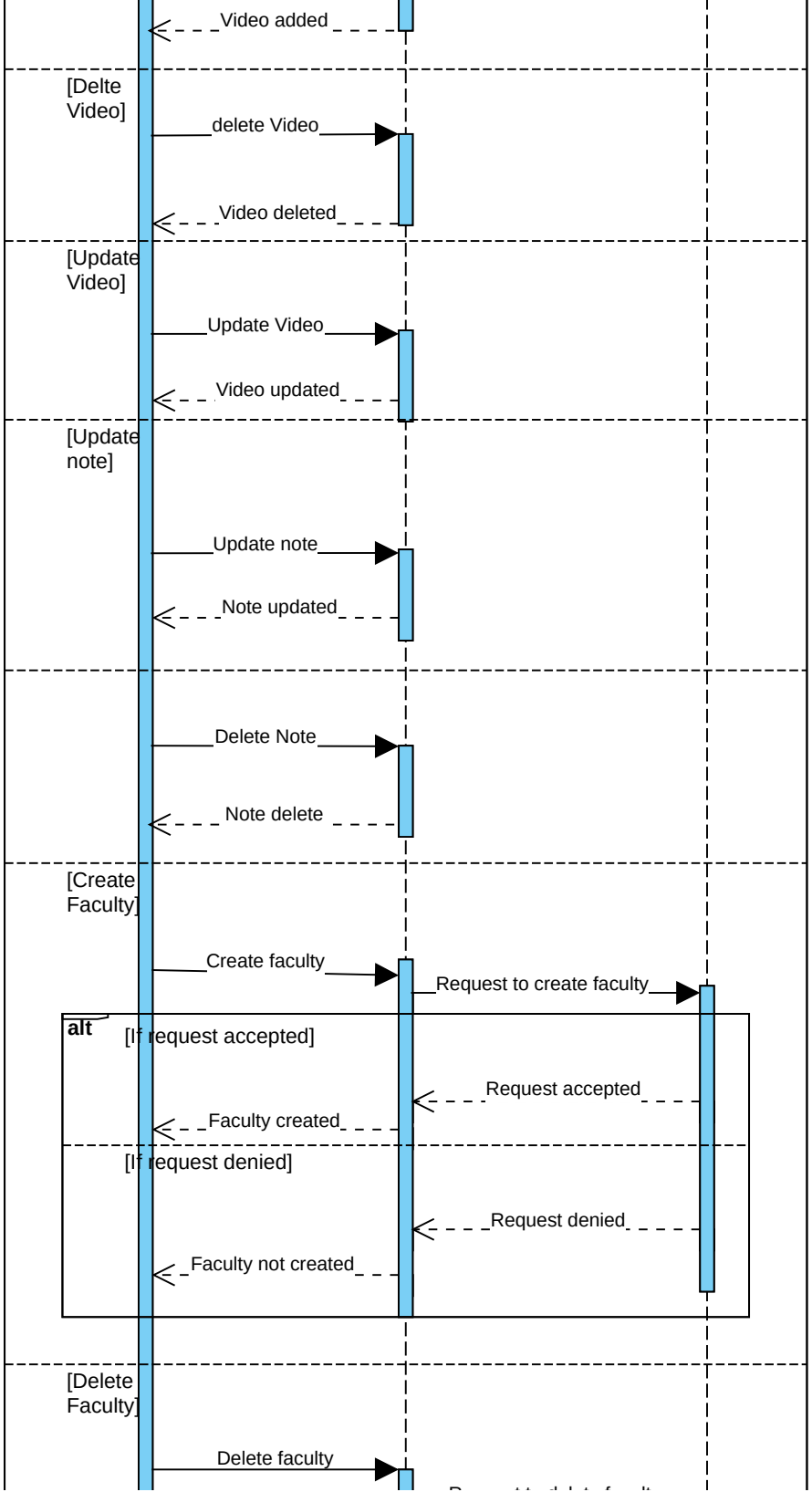


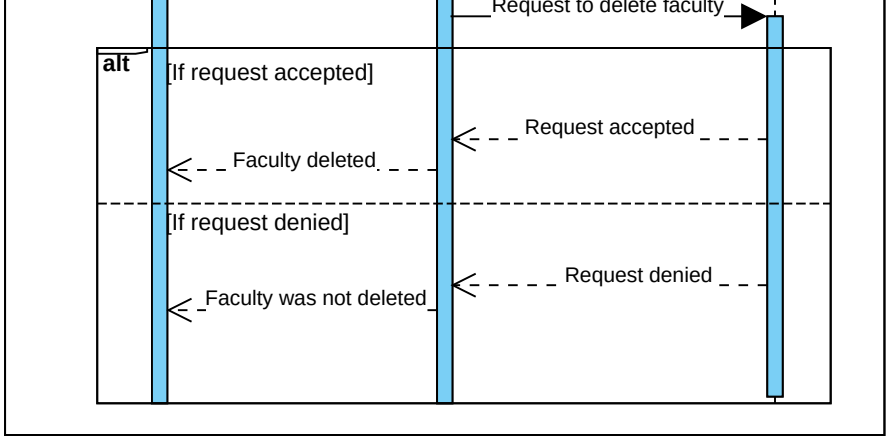




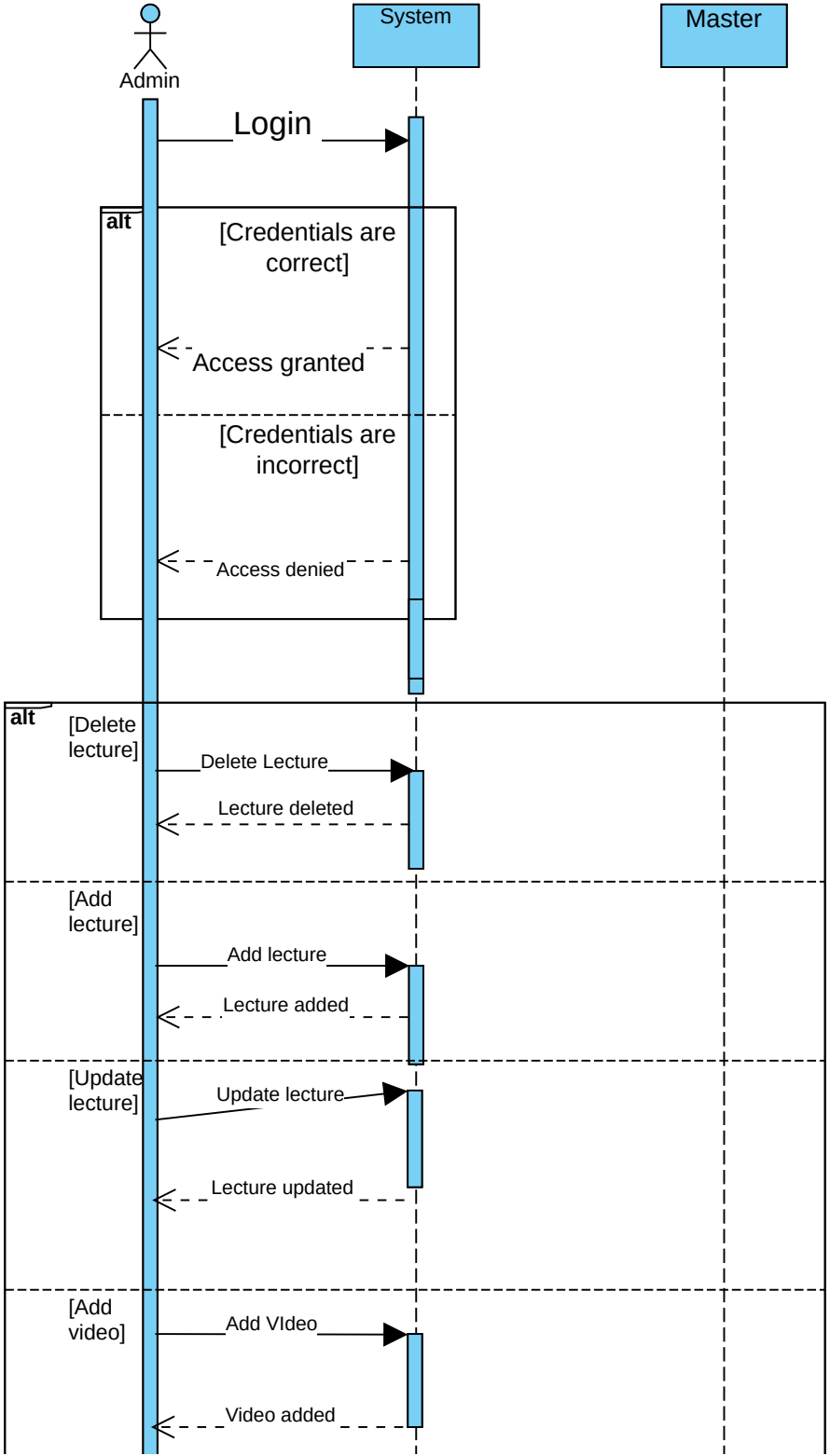
Scequence Diagram-2:

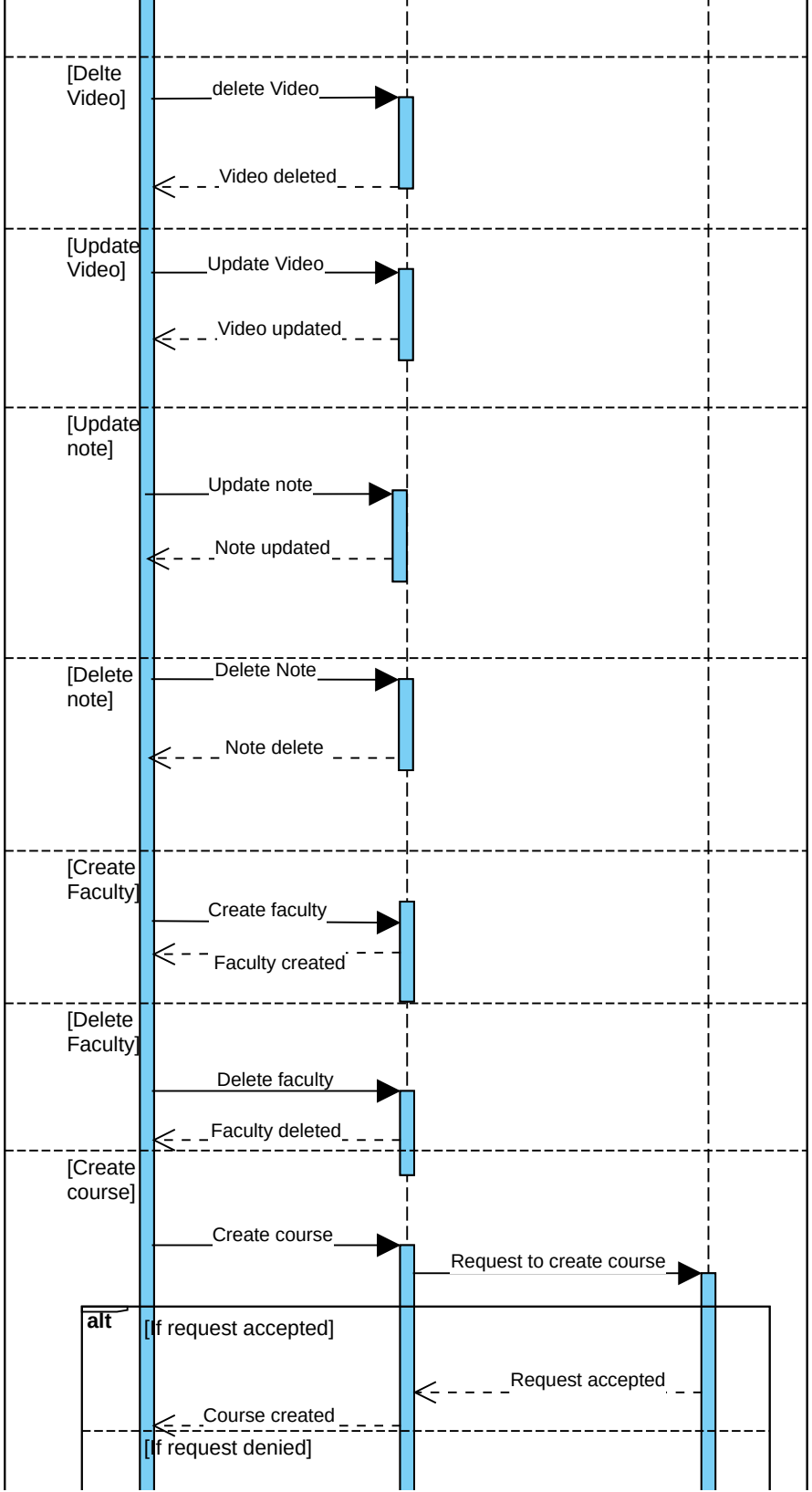


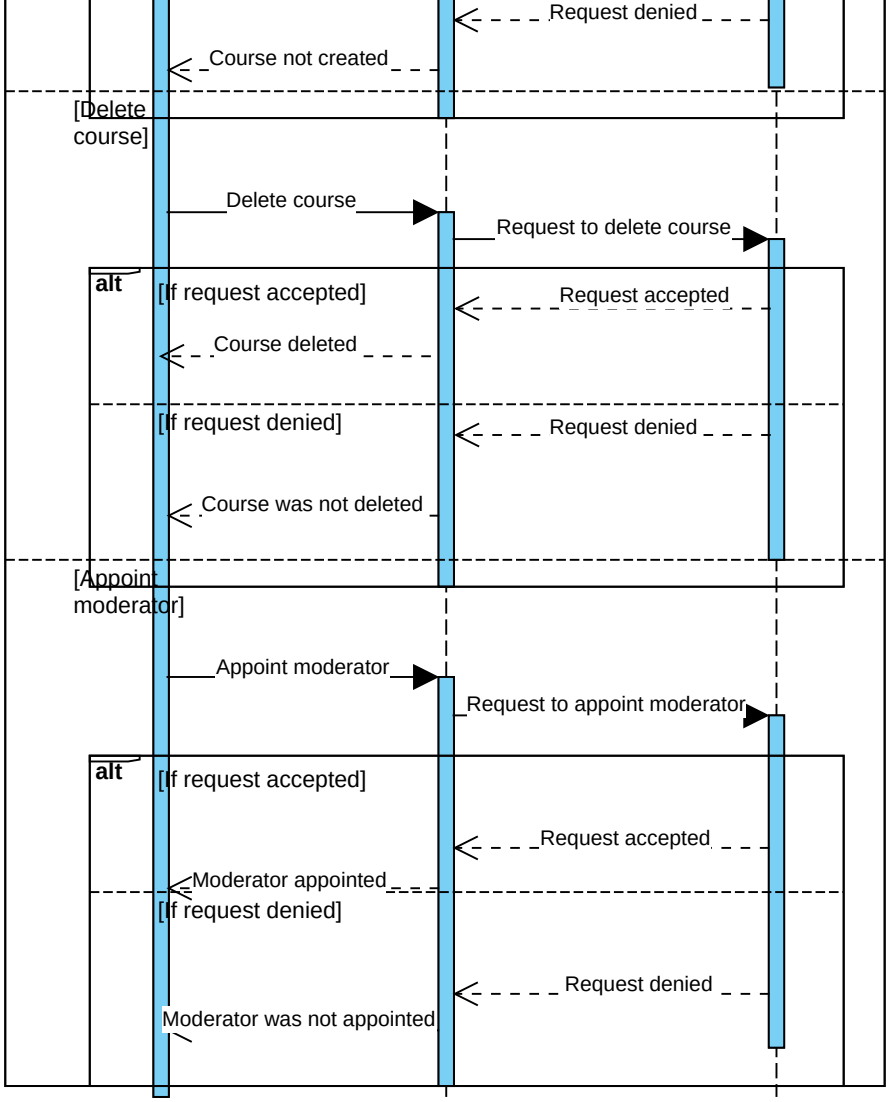




Scequence Diagram-3:







Scequence Diagram-4: