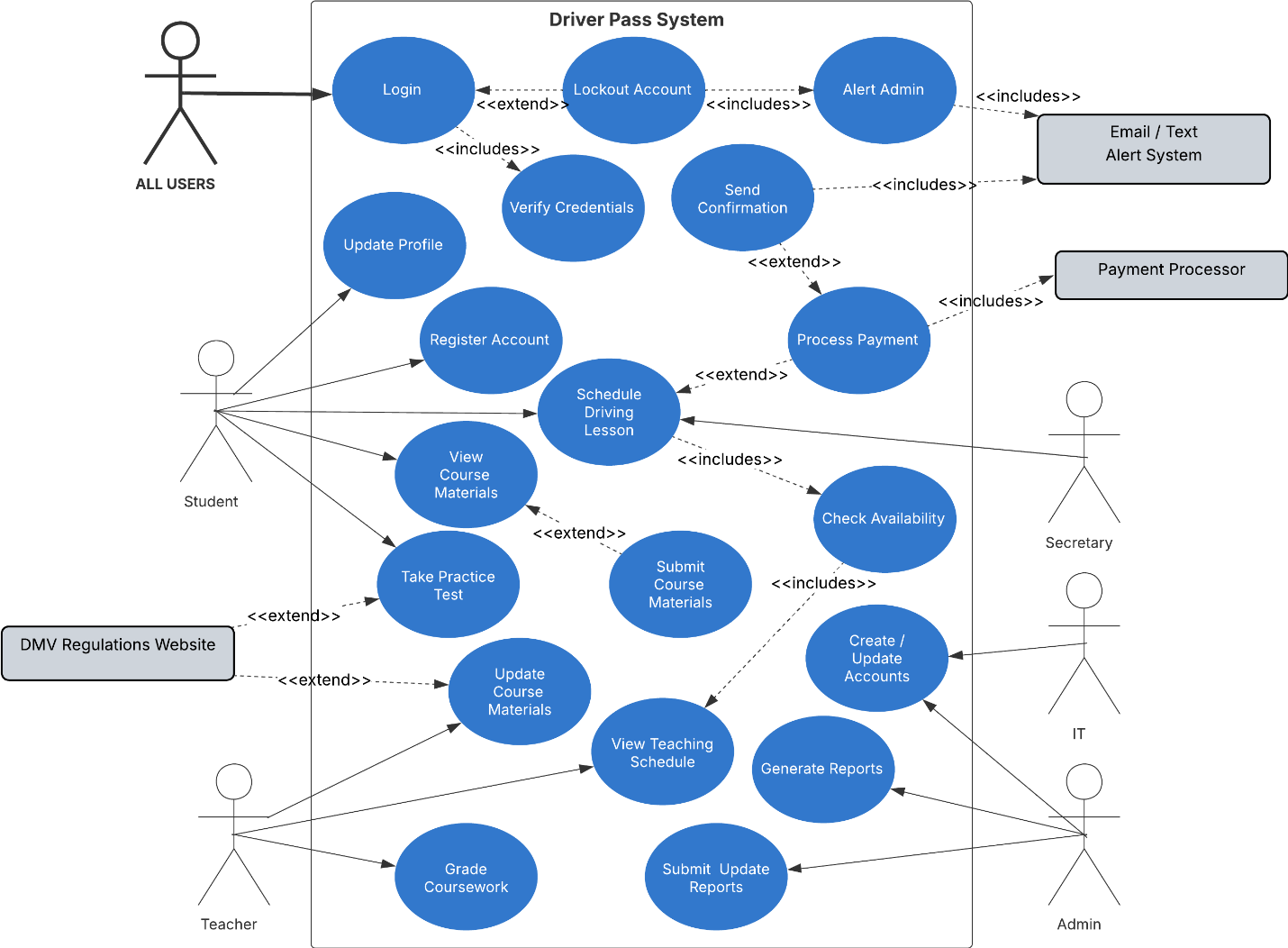
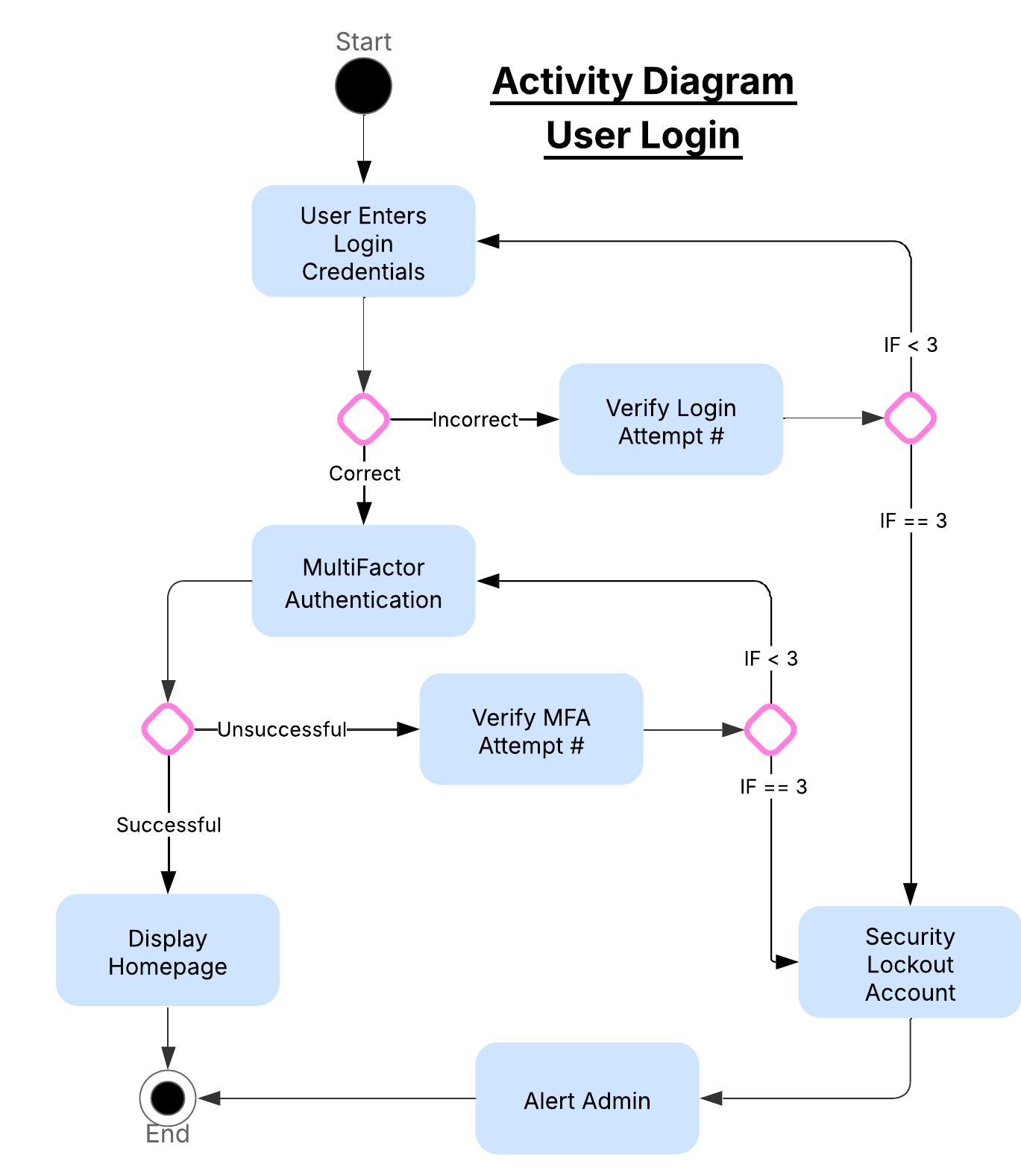
# CS 255 System Design Document Template

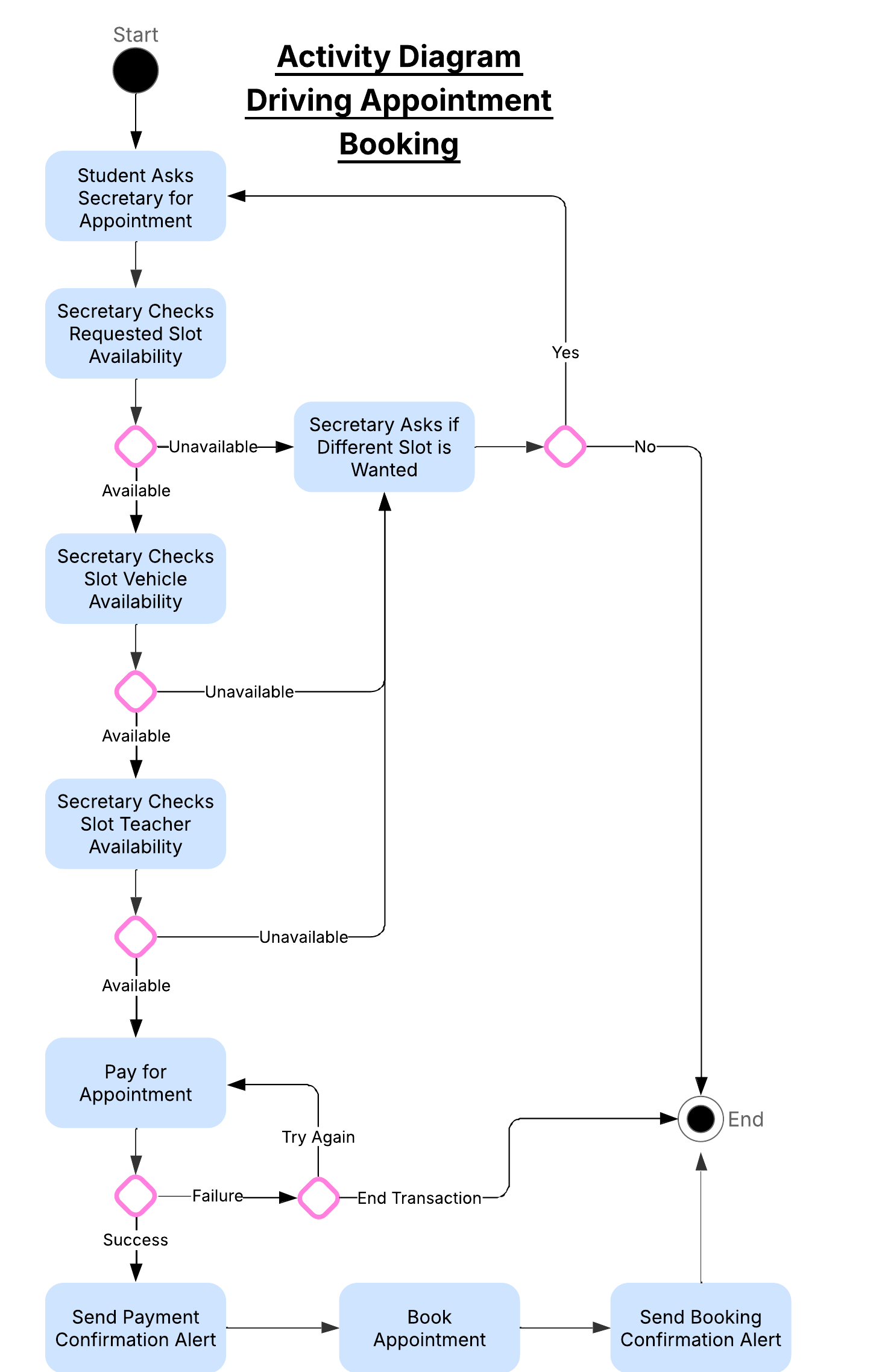
## UML Diagrams

### UML Use Case Diagram

**

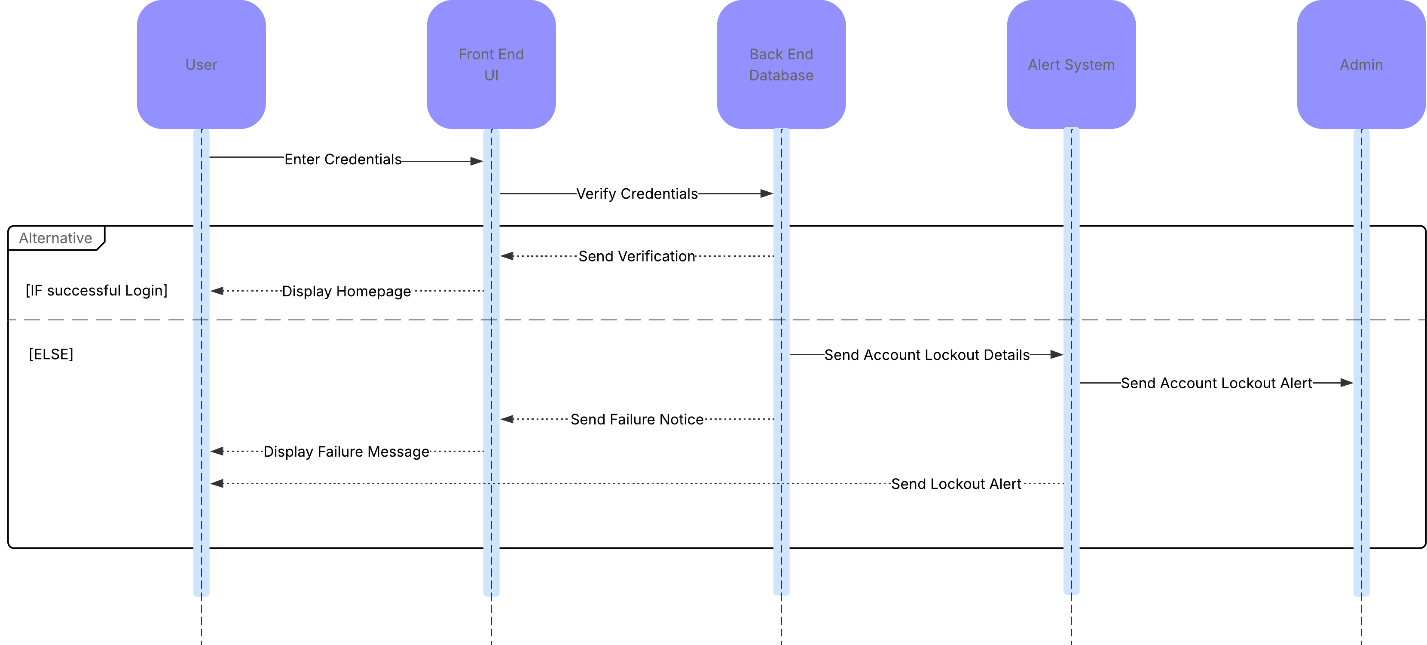
### UML Activity Diagrams



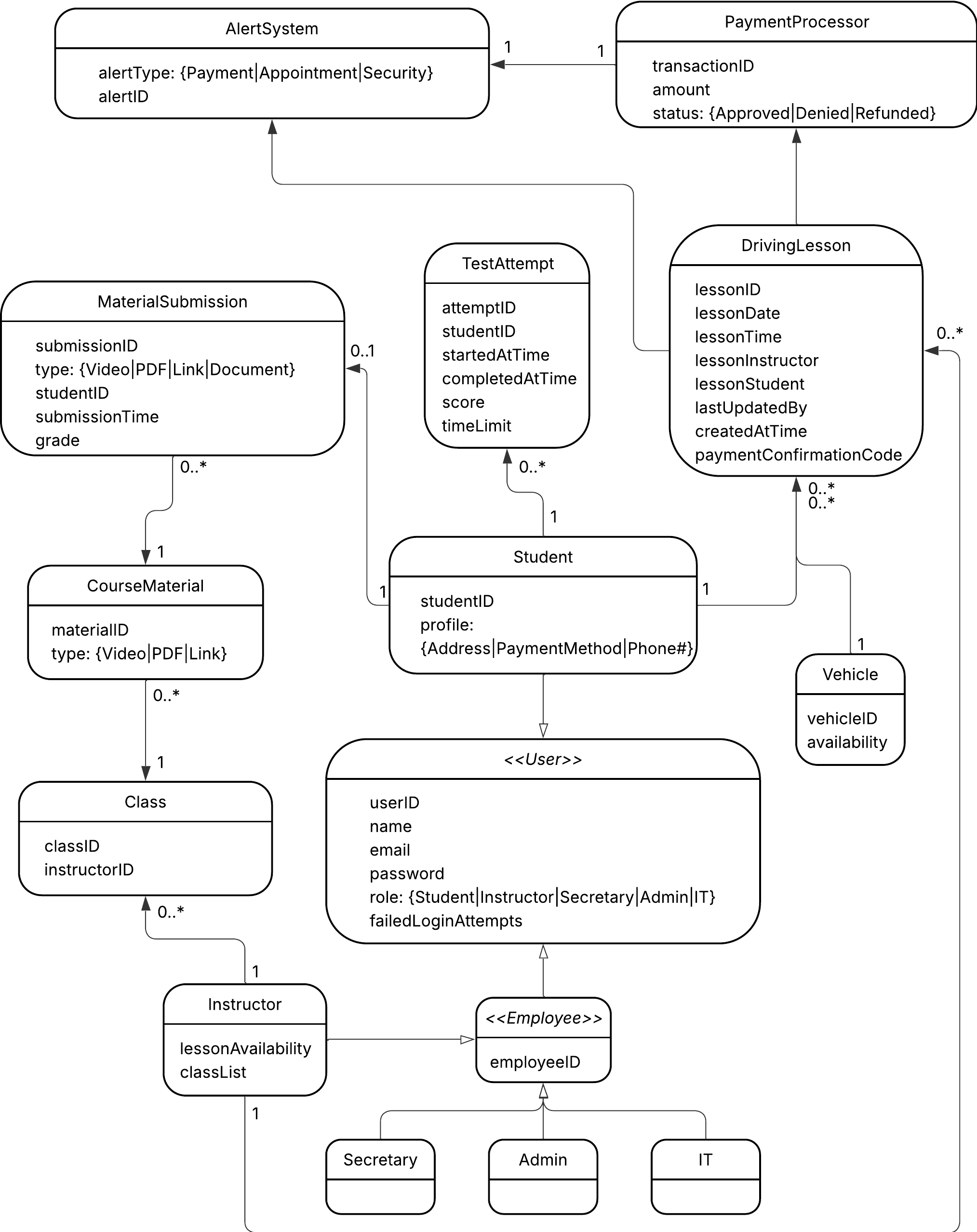


### UML Sequence Diagram

Sequence Diagram for Use Case User Login



### UML Class Diagram

**

## Technical Requirements

The system is going to be deployed as a secure, cloud-hosted web application with a responsive interface for desktop, tablet, and mobile devices and users. The client needs a reliable and secure login system with account protection and security holds, with security notifications. They also need functionality for scheduling lessons, payments, confirmations, course material access, course material grading, tests, and course material submissions. Roles for users in the system will be either Student, Instructor, Secretary, Admin, or IT. One specialty Admin function must be the ability to download and upload reports to update student information. Keeping the solution web-based ensures consistent behavior across platforms and aligns with the original intent to provide a single system that integrates registration, learning activities, and appointment scheduling in one place.

From a platform perspective the application will run in a modern browser and be hosted on a managed cloud provider such as AWS or Azure, using Linux runtime to support common server technologies and deployment tooling. This choice was made to achieve scalability, security, and easier maintenance over time. Performance targets follow usability expectations set in earlier reports, typical pages should load within about two seconds under normal load, including during busier periods. These expectations map directly to the nonfunctional requirements defined for responsiveness and broad device/browser compatibility, as well as preference for cloud hosting. A centralized database is required to store student records, bookings, payments, and audit data, and the environment should produce daily logs for operational transparency.

The front end runs in all major modern browsers and is built with web technologies LIKE HTML5/CSS/JavaScript to remain compatible across all devices. The back end uses a Linux hosted stack like Node.js to expose secure APIs, enforce role based access, and implement login protections including encryption, lockouts on repeated failures, and password recovery. The application supports a web interface for all roles and includes CSV import/export to help admins manage and report data without direct database access.

Various tools will be utilized by the system like built-in health checks, centralized logging, and performance analytics so IT can monitor outages or abnormal behavior. Admin users also use CSV tools for report exports/imports and a dedicated QA/test environment before production releases. Notifications in the form of email/SMS are used for confirmations and reminders tied to bookings and policy updates.

The infrastructure needed is largely supported by the third party cloud hosting platform. This is one of the benefits of cloud hosting, you do not need to host your own server or have any data farms on site. This cloud infrastructure also supports auto scaling the application as the user base grows. The alert system and payment processor are also third party entities that will be integrated with, but not physically housed.