# CS 255 System Design Document

Edwin Martinez

# Introduction

The DriverPass system design aligns with the business requirements by combining both object-oriented and process-oriented views. The UML diagrams demonstrate user interaction, workflows, and system structure, while the technical requirements outline the necessary infrastructure to ensure security, reliability, and scalability. This design provides a clear blueprint for developers to implement and deliver a robust system for DriverPass.

## Use Case Diagram

The use case diagram captures the functional requirements of DriverPass. It shows how different actors (Student, Admin, IT Admin, Secretary, and DMV System) interact with the system. Students can create accounts, log in, schedule and cancel appointments, take practice exams, and view progress. Admins manage training packages, view logs, and generate reports. IT Admins handle user roles and account blocking, while Secretaries register students and update student information. The DMV system provides policy/test updates.

## Activity Diagrams

Two use cases were expanded into activity diagrams:

### Schedule Appointment

The student logs in, views available slots and instructors, selects a slot, and the system checks for conflicts and lesson credits. If credits are insufficient, the system prompts the student to purchase more. Once confirmed, the reservation is created, credits decremented, and a confirmation is sent.

### Take Practice Exam

The student logs in, starts an exam instance, and questions are pulled from the question bank. Students answer questions until completion or timeout. The exam is graded, results are stored, and scores/recommendations are shown to the student.

## Sequence Diagram

The sequence diagram illustrates the Schedule Appointment use case at a detailed interaction level. It shows the student interacting with the web application, which communicates with the authentication service, scheduling service, and balance service. The flow includes login validation, slot retrieval, conflict detection, credit verification, reservation, and confirmation. Alternative and loop fragments handle login retries, slot conflicts, and insufficient credits.

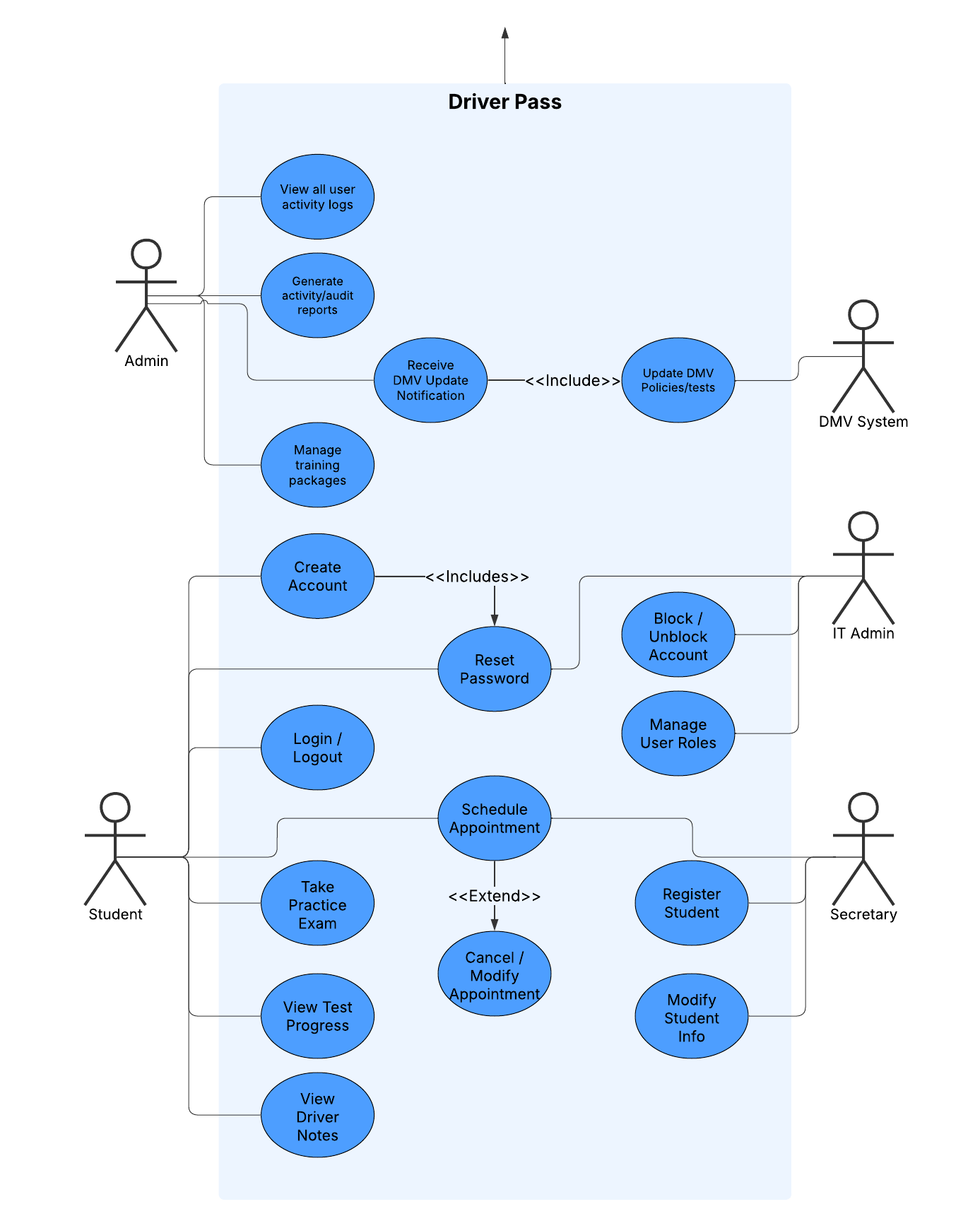
## Class Diagram

The class diagram represents the main objects in the DriverPass system and their relationships:

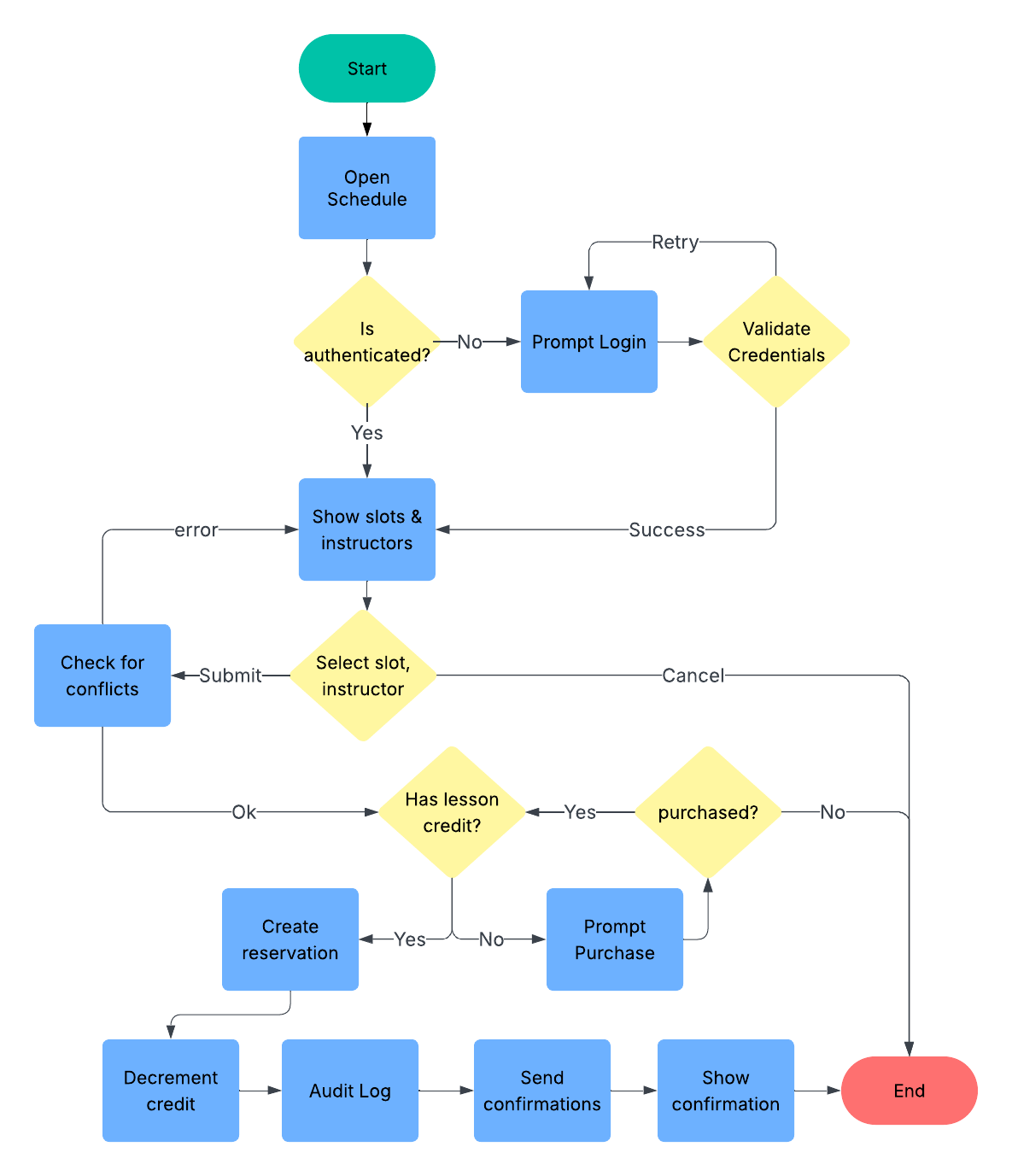
* User (base class) with three subclasses: Student, Instructor, and Admin.
* Student can schedule/cancel appointments, take practice exams, and view results.
* Instructor can view and update their schedule.
* Admin manages users and generates reports.
* Appointment, Schedule, PracticeExam, ExamResult, Question, Report are supporting classes.

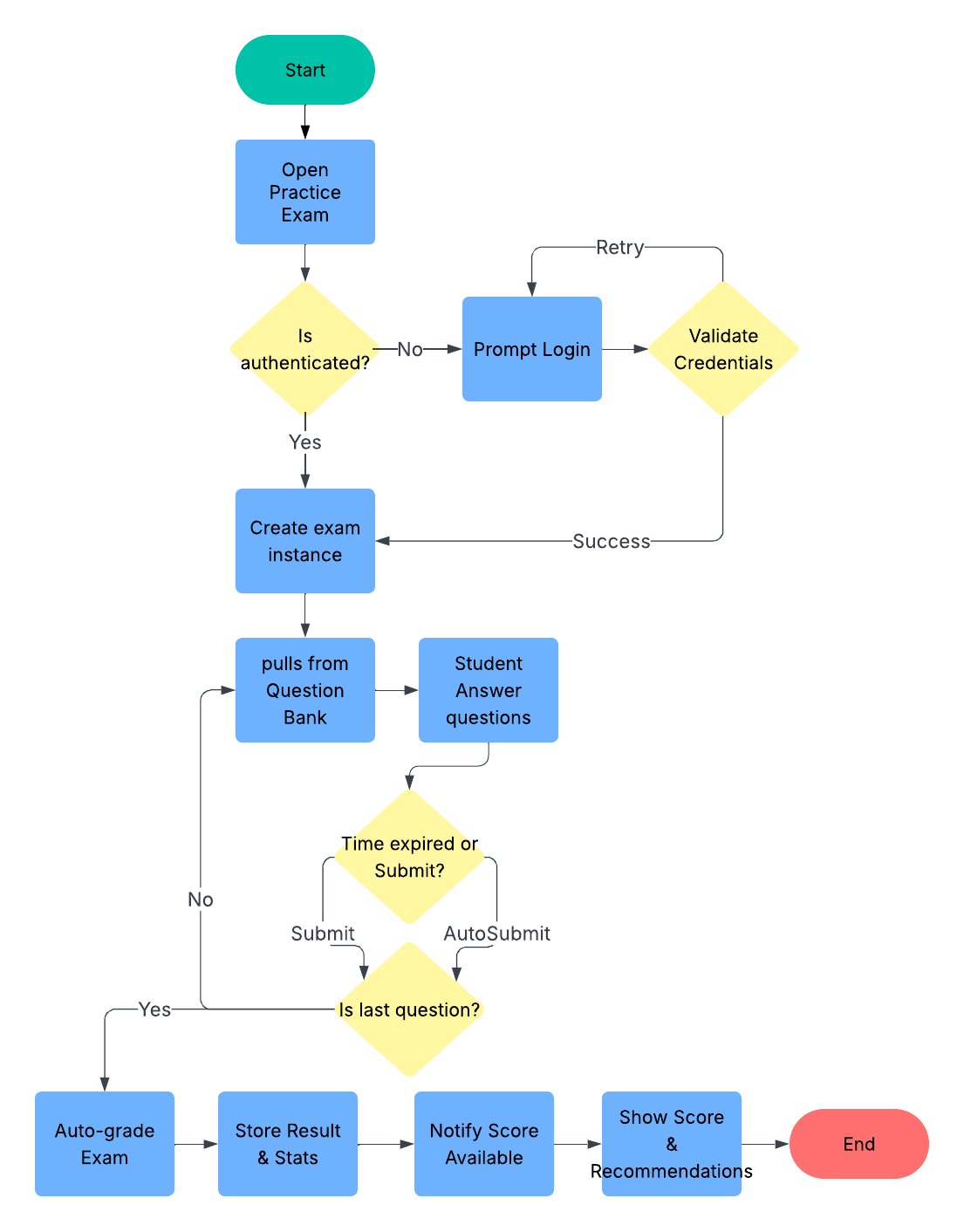
## UML Diagrams

### UML Use Case Diagram

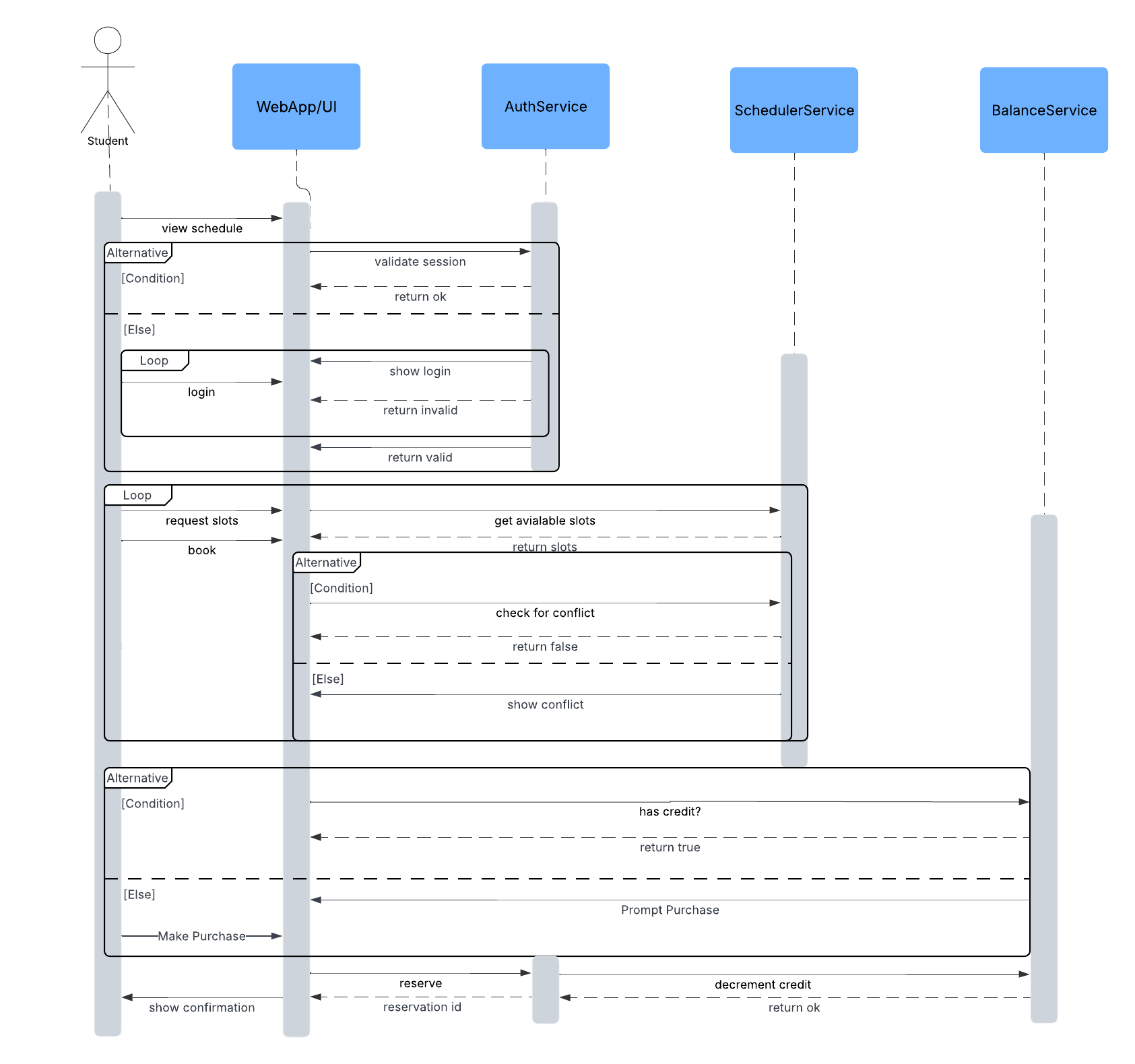
**

### UML Activity Diagrams

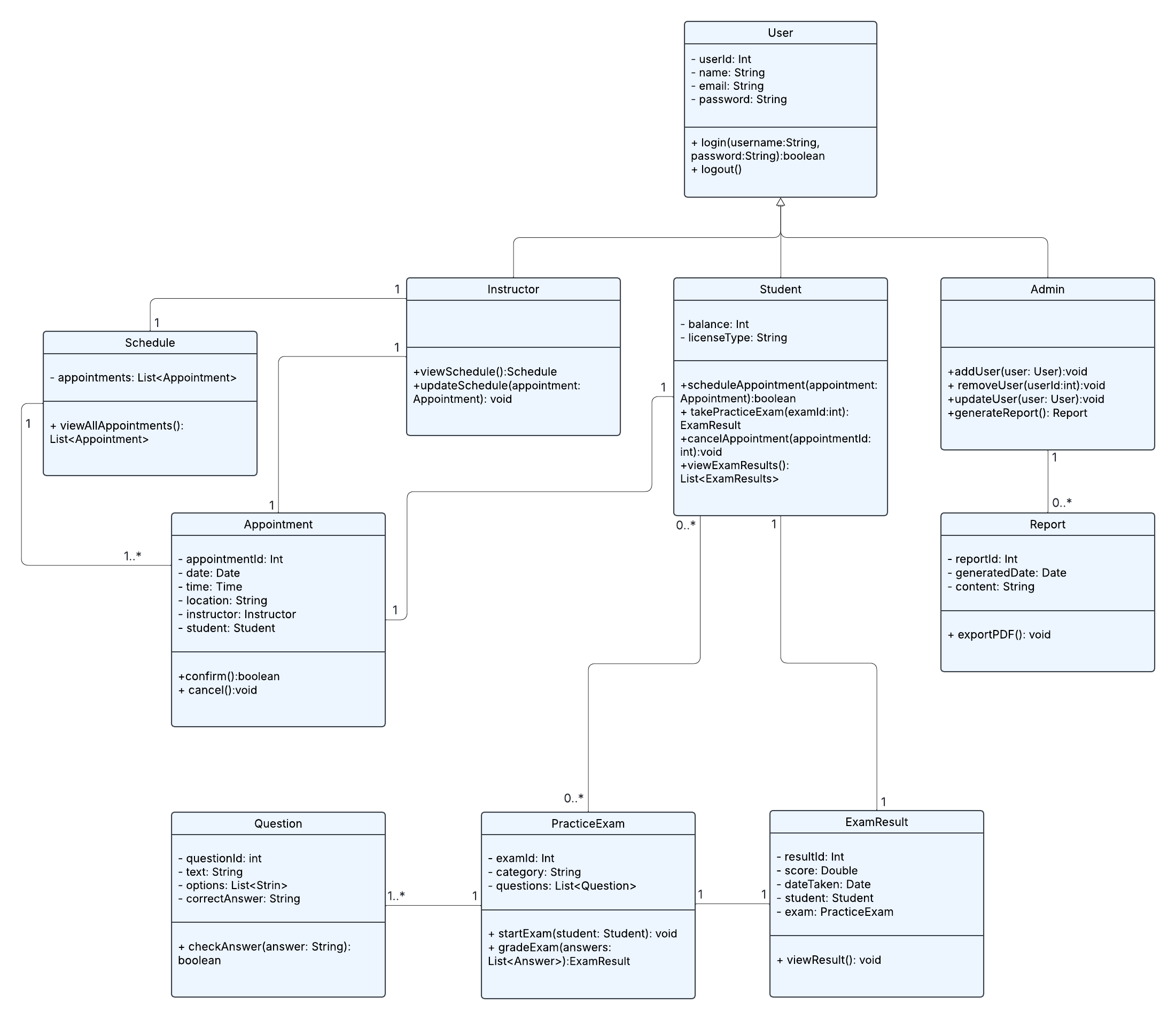
**

**

### UML Sequence Diagram

**

### UML Class Diagram

**

## Technical Requirements

## Functional Requirements

* **Student Functions:** Account creation, login/logout, scheduling and canceling lessons, taking practice exams, viewing progress and notes.
* **Admin Functions:** User management, package management, system audit logs, generating reports.
* **Instructor Functions:** Viewing and updating teaching schedules.
* **System Functions:** DMV update integration, user role management, conflict detection, automated grading.

## Nonfunctional Requirements

* **Performance:** The system must support concurrent access by multiple students scheduling and taking exams without noticeable delay.
* **Scalability:** Must handle growth in the number of students, appointments, and practice exams.
* **Security:** User authentication, secure password storage, role-based access control, and encrypted communication.
* **Reliability:** High availability with data backups and recovery mechanisms.
* **Usability:** Simple, web-based user interface accessible across devices.

## Technical Infrastructure

### Hardware:

* Cloud-hosted servers (recommended: AWS/Azure) with load balancing.
* Relational database server (MySQL or PostgreSQL) for user, appointment, and exam data.

### Software:

* **Backend:** Java or C# for business logic.
* **Frontend:** Web application with HTML/CSS/JavaScript (React/Angular).
* **Database:** Relational DBMS.

### Tools:

* **UML modeling:** Lucidchart.
* **Version control:** Git.
* Development IDEs (Eclipse, Visual Studio Code).

### Integration:

* External DMV system APIs for test updates.
* Email/SMS notification service for confirmations.

### Networking:

* Secure HTTPS protocol.
* Firewalls and VPN access for administrators.