# CS 255 System Design Document Template

### Project Overview

**Project Name:** DriverPass System Design

**Project Description:** DriverPass is an innovative online platform designed to streamline the process of learning to drive and obtaining a driver's license. The system aims to provide a seamless experience for students, instructors, and administrators by offering a range of services, including lesson scheduling, online training, payment processing, and performance tracking. By integrating these services into a single platform, DriverPass ensures that the entire process, from booking lessons to passing the driving test, is efficient, user-friendly, and accessible.

**Project Objectives:** The primary objectives of the DriverPass system are as follows:

1. **Streamline the Lesson Scheduling Process:** Allow students to easily book, reschedule, or cancel driving lessons with available instructors, ensuring flexibility and convenience.
2. **Facilitate Online Learning:** Provide students with access to a comprehensive range of online learning materials and practice tests, helping them prepare effectively for both the written and practical driving exams.
3. **Simplify Payment Processing:** Integrate secure payment processing within the system, enabling students to pay for lessons and tests seamlessly.
4. **Enhance Communication:** Improve communication between students, instructors, and administrators through automated notifications, feedback systems, and real-time updates.
5. **Ensure Compliance:** Maintain up-to-date training materials and test requirements in line with the latest regulations from the Department of Motor Vehicles (DMV).
6. **Provide Administrative Oversight:** Equip administrators with tools to manage user accounts, monitor system performance, generate reports, and ensure the overall functionality and security of the platform.

**System Scope:** The DriverPass system is designed to cater to three primary user groups: students, instructors, and administrators. Each group will have access to specific functionalities tailored to their needs. The system will also integrate with external systems, such as a secure payment gateway and the DMV's regulatory database, to provide a comprehensive service offering. The system is intended to be scalable, allowing for future enhancements and the addition of new features as the platform grows.

**Assumptions:**

* All users will have access to the internet and a compatible device to use the DriverPass platform.
* The system will be developed using modern web technologies and will adhere to industry standards for security and data privacy.
* The DMV's database will be accessible for integration with the DriverPass system for real-time updates on regulations and test requirements.

**Constraints:**

* The system must be operational within the allocated budget and timeline.
* All data must be stored securely and comply with local and federal data protection regulations.
* The system must be able to handle peak traffic periods, particularly during high-demand times for driving lessons and tests.

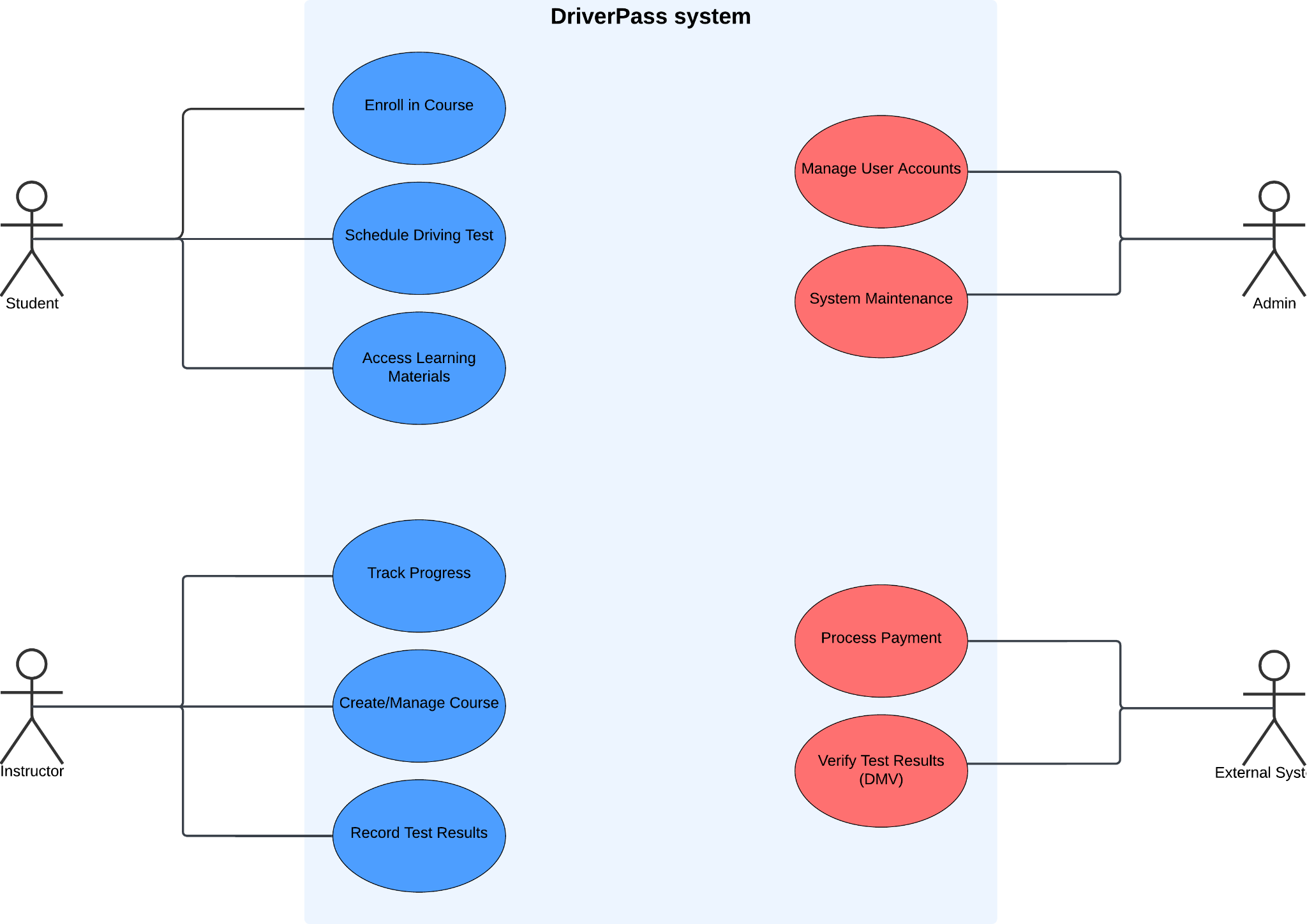
## UML Diagrams

### UML Use Case Diagram

The UML Use Case Diagram for the DriverPass system outlines the primary actors and their interactions with the system’s key functionalities. The actors identified include Students, Instructors, Administrators, and the DMV System. Each actor has specific roles and use cases they interact with:

* **Students:** Can sign up, log in, book lessons, take online tests, view progress, and make payments.
* **Instructors:** Manage schedules, provide feedback, and track student progress.
* **Administrators:** Oversee the entire system, manage user accounts, generate reports, and ensure system integrity.
* **External Systems:** Verifies test results and ensures compliance with driving regulations.

The use case diagram ensures that all necessary interactions are captured, making it clear how each user will interact with the system.

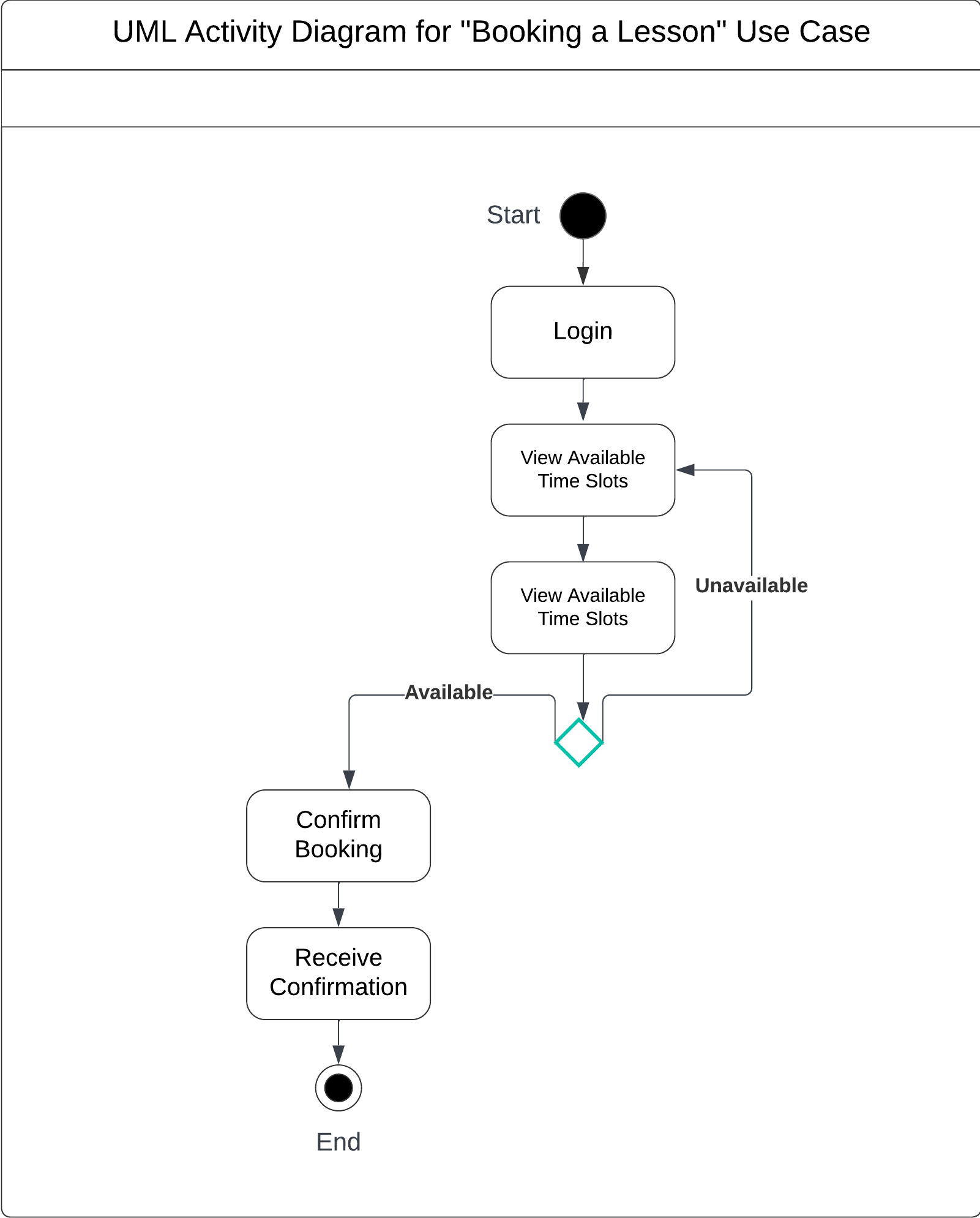


### UML Activity Diagrams

Two UML Activity Diagrams have been created for the DriverPass system, each corresponding to a critical use case:

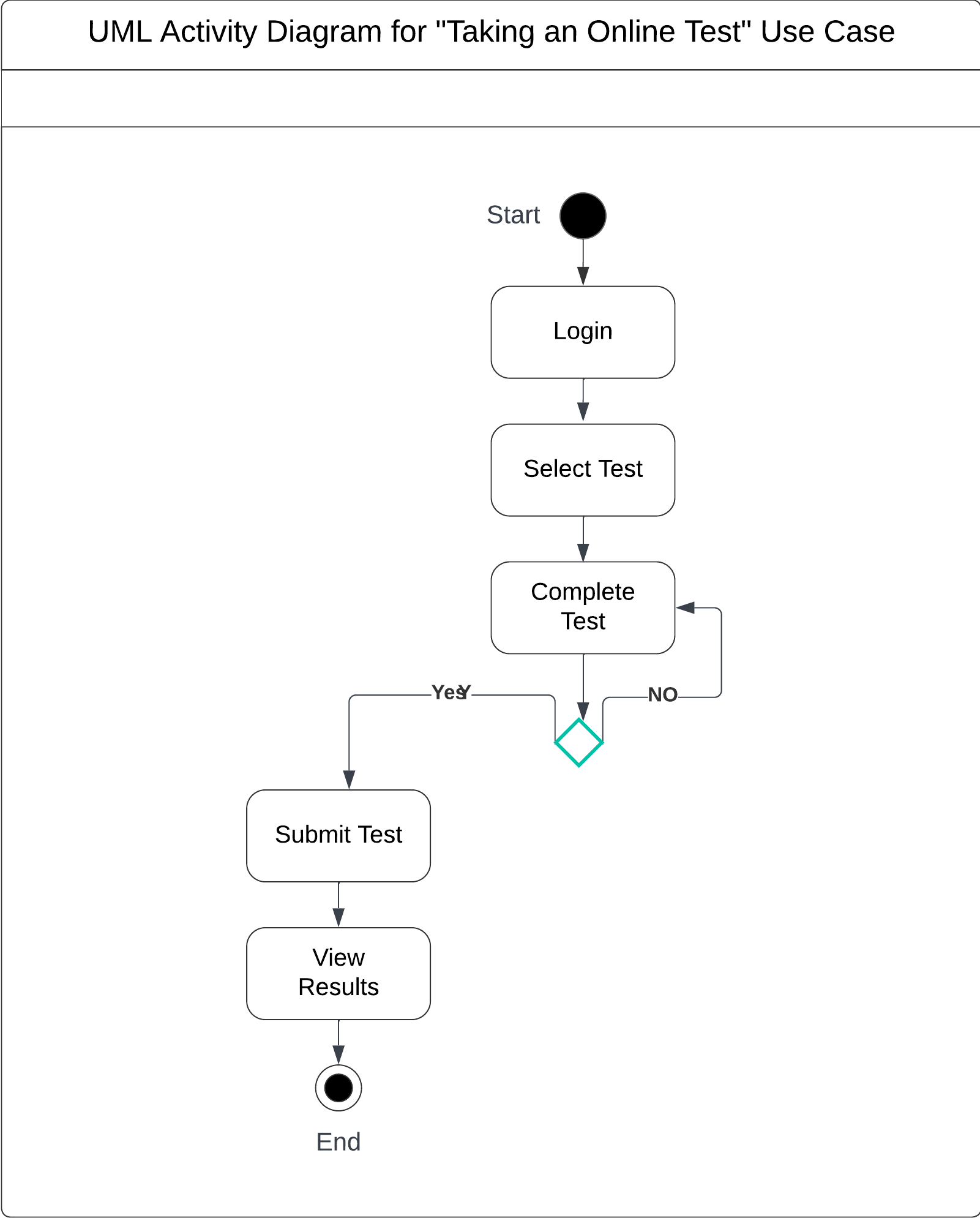
**Activity Diagram 1: Student Booking a Lesson**

This diagram captures the process a student follows to book a driving lesson. It starts from logging in to the system, selecting an available time slot, confirming the booking, and receiving a confirmation notification.



**Activity Diagram 2: Student Taking an Online Test**

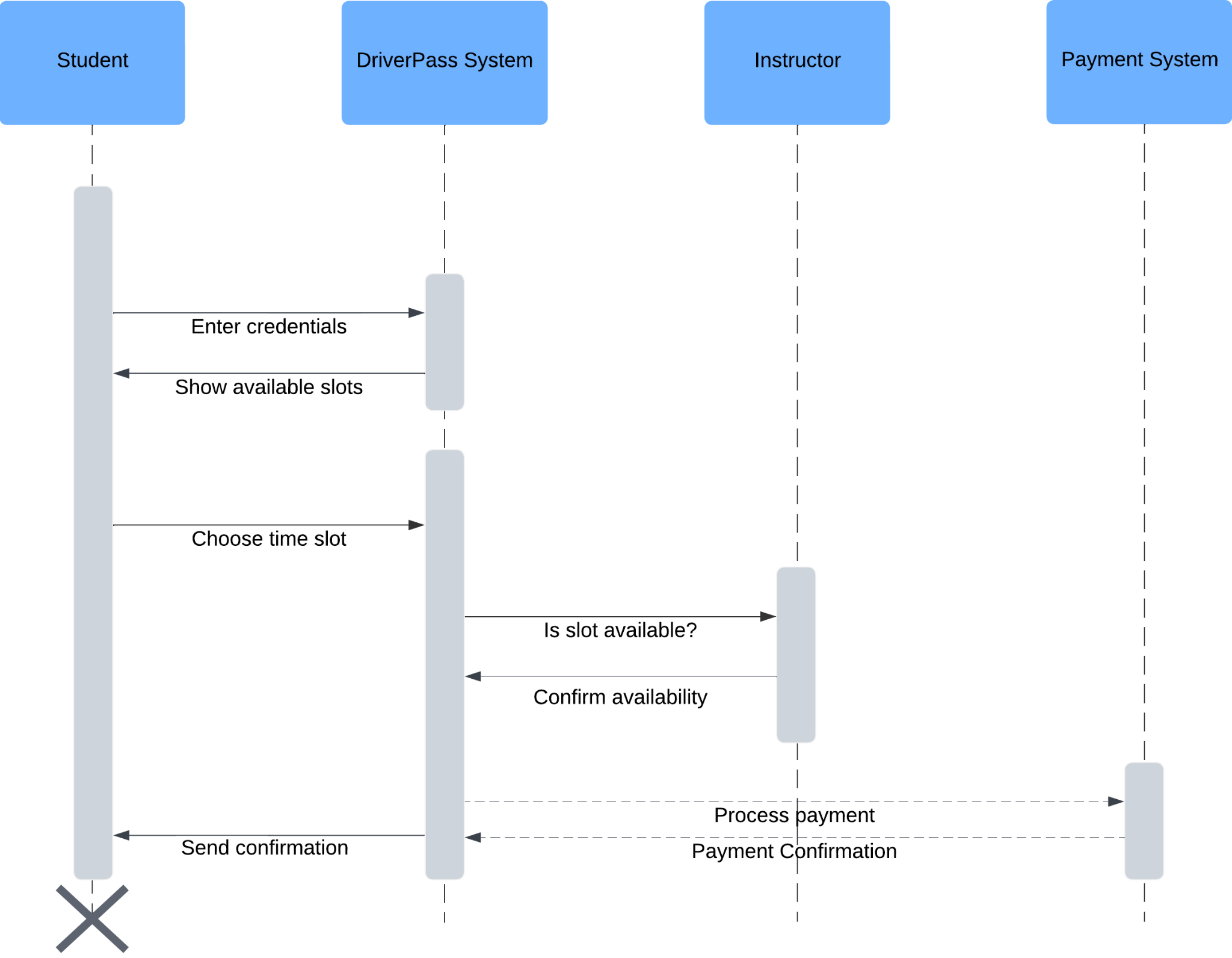
This diagram outlines the steps involved in a student taking an online driving test. It includes logging in, selecting the test, answering questions, submitting the test, and receiving the results.



Both activity diagrams clearly depict the step-by-step flow of these processes, ensuring that the system functions as intended for each of these critical operations.

### UML Sequence Diagram

A UML Sequence Diagram has been created to represent the interactions between the student, system, instructor, and the DMV system during the process of booking a lesson and receiving feedback. This diagram shows the sequence of messages exchanged between the student, the system, and other relevant actors. It starts with the student booking a lesson, the system confirming availability, notifying the instructor, and concluding with the DMV system updating the student’s progress after the lesson is completed.

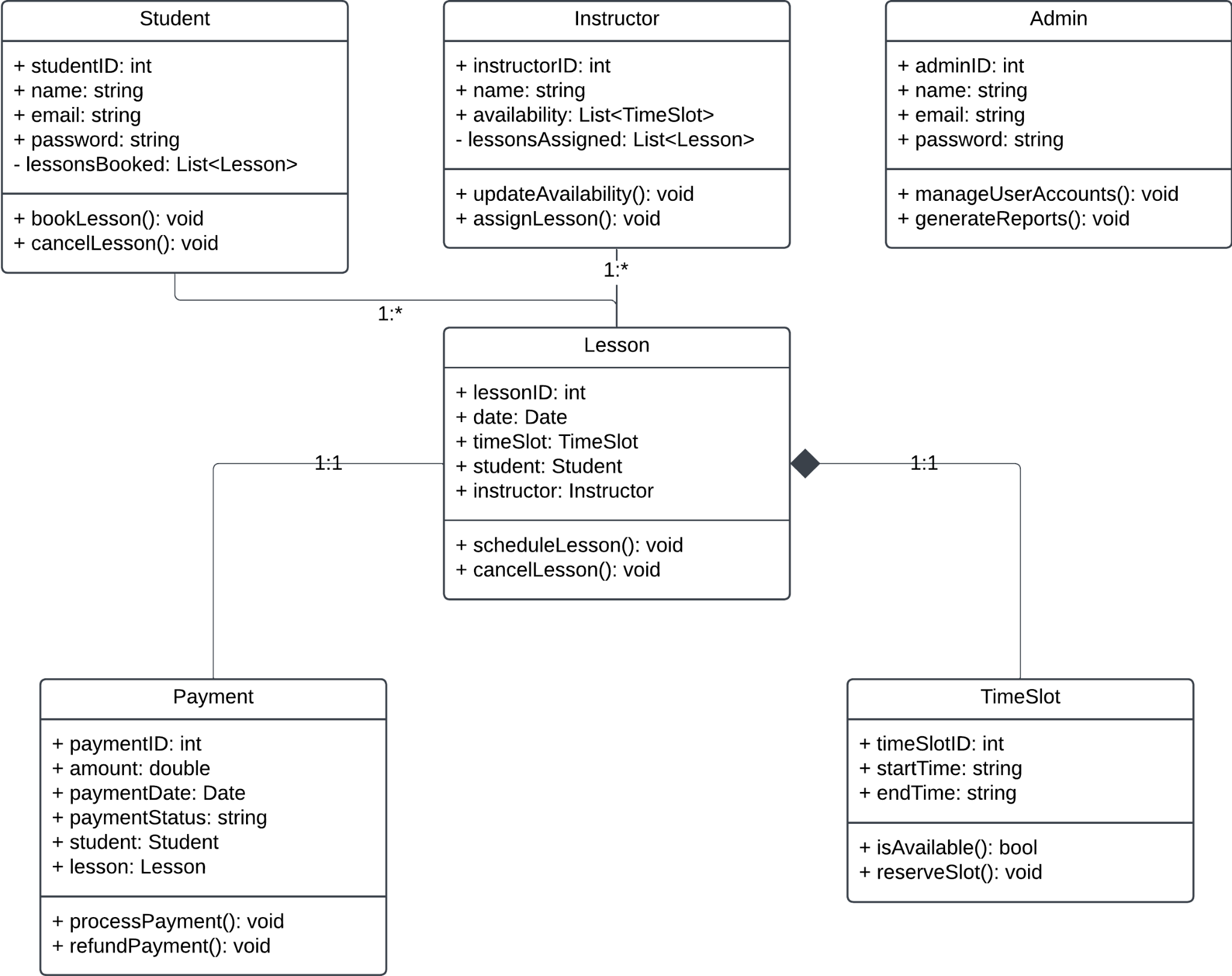


### UML Class Diagram

**Description:**The UML Class Diagram illustrates the structure of the DriverPass system in terms of its classes, attributes, and relationships. The diagram includes the following key classes:

* **Student:** Attributes include StudentID, Name, Email, and LessonHistory.
* **Instructor:** Attributes include InstructorID, Name, Availability, and Feedback.
* **Admin:** Attributes include AdminID, Name, and Permissions.
* **Lesson:** Attributes include LessonID, Date, Time, Instructor, and Student.
* **Payment:** Attributes include PaymentID, Amount, Date, and PaymentMethod.
* **Test:** Attributes include TestID, Date, Result, and Student.

Relationships between these classes are depicted, such as the association between Students and Lessons, and between Lessons and Instructors. The diagram provides a clear overview of the system’s structure and how data is organized within it.



## Technical Requirements

#### **1. Hardware Requirements**

* **Servers:**
  + **Web Server:** A web server will be required to host the DriverPass application. This server must be robust enough to handle multiple simultaneous users, especially during peak hours when many students may be scheduling lessons or taking online tests.
  + **Database Server:** A separate database server will be needed to store user information, lesson schedules, test results, and payment details. This server should have high availability and redundancy to prevent data loss.
* **Client Devices:**
  + **End-User Devices:** The system must be compatible with various end-user devices, including desktop computers, laptops, tablets, and smartphones. Users (students, instructors, admins) should be able to access the system seamlessly from any of these devices.
* **Backup Hardware:**
  + **Backup Storage:** Regular backups of the database and system configuration files should be stored on secure, redundant hardware to ensure data recovery in case of a system failure.

#### **2. Software Requirements**

* **Operating Systems:**
  + **Server-Side OS:** The web and database servers should run on a stable, secure operating system such as Linux (e.g., Ubuntu Server) or Windows Server.
  + **Client-Side OS:** The system should support all major operating systems used by end users, including Windows, macOS, Android, and iOS.
* **Web Server Software:**
  + **Apache/Nginx:** The web server should use either Apache or Nginx to serve the application, ensuring efficient handling of HTTP requests and load balancing.
* **Database Management System (DBMS):**
  + **MySQL/PostgreSQL:** The system should use a reliable DBMS like MySQL or PostgreSQL to manage relational data efficiently, including user information, lesson schedules, and test results.
* **Development Tools:**
  + **Programming Languages:** The system will be developed using languages such as JavaScript (for front-end development), Node.js or PHP (for back-end development), and SQL (for database queries).
  + **Integrated Development Environment (IDE):** Developers will use an IDE like Visual Studio Code or JetBrains WebStorm to write, test, and debug the application code.
* **Security Software:**
  + **SSL/TLS Certificates:** SSL/TLS certificates must be implemented on the web server to encrypt data transmitted between the server and client devices, ensuring secure communication.
  + **Firewall and Anti-Virus:** A robust firewall and anti-virus software should be installed on the servers to protect against unauthorized access and malware attacks.

#### **3. Network Infrastructure**

* **Internet Connectivity:**
  + **High-Speed Internet:** Reliable, high-speed internet connectivity is essential for the web server and database server to ensure that users experience minimal latency and downtime.
* **Load Balancers:**
  + **Load Balancing Solutions:** To distribute incoming traffic evenly across multiple servers, a load balancer should be implemented. This will help prevent server overloads and ensure continuous availability of the system.
* **Virtual Private Network (VPN):**
  + **VPN for Remote Access:** Admins and developers requiring remote access to the servers should use a secure VPN to ensure that all remote connections are encrypted and protected from potential threats.

#### **4. Security Measures**

* **User Authentication:**
  + **Multi-Factor Authentication (MFA):** Implement MFA for users accessing the system, particularly for admin and instructor accounts, to add an extra layer of security beyond just username and password.
* **Data Encryption:**
  + **Encryption Standards:** All sensitive data, including user credentials, payment details, and personal information, should be encrypted using AES-256 or a comparable encryption standard.
* **Regular Security Audits:**
  + **Audit Frequency:** Conduct regular security audits, both internally and externally, to identify potential vulnerabilities and ensure compliance with industry standards.

#### **5. Scalability and Performance**

* **Scalability:**
  + **Horizontal and Vertical Scaling:** Design the system architecture to support both horizontal scaling (adding more servers) and vertical scaling (upgrading existing servers) as the user base grows.
* **Performance Monitoring:**
  + **Monitoring Tools:** Use performance monitoring tools like New Relic or Datadog to track system performance, including load times, server health, and response times. This will help identify bottlenecks and ensure that the system performs optimally under various conditions.