# CS 255 Project 2: System Design

# Tyler Ellis

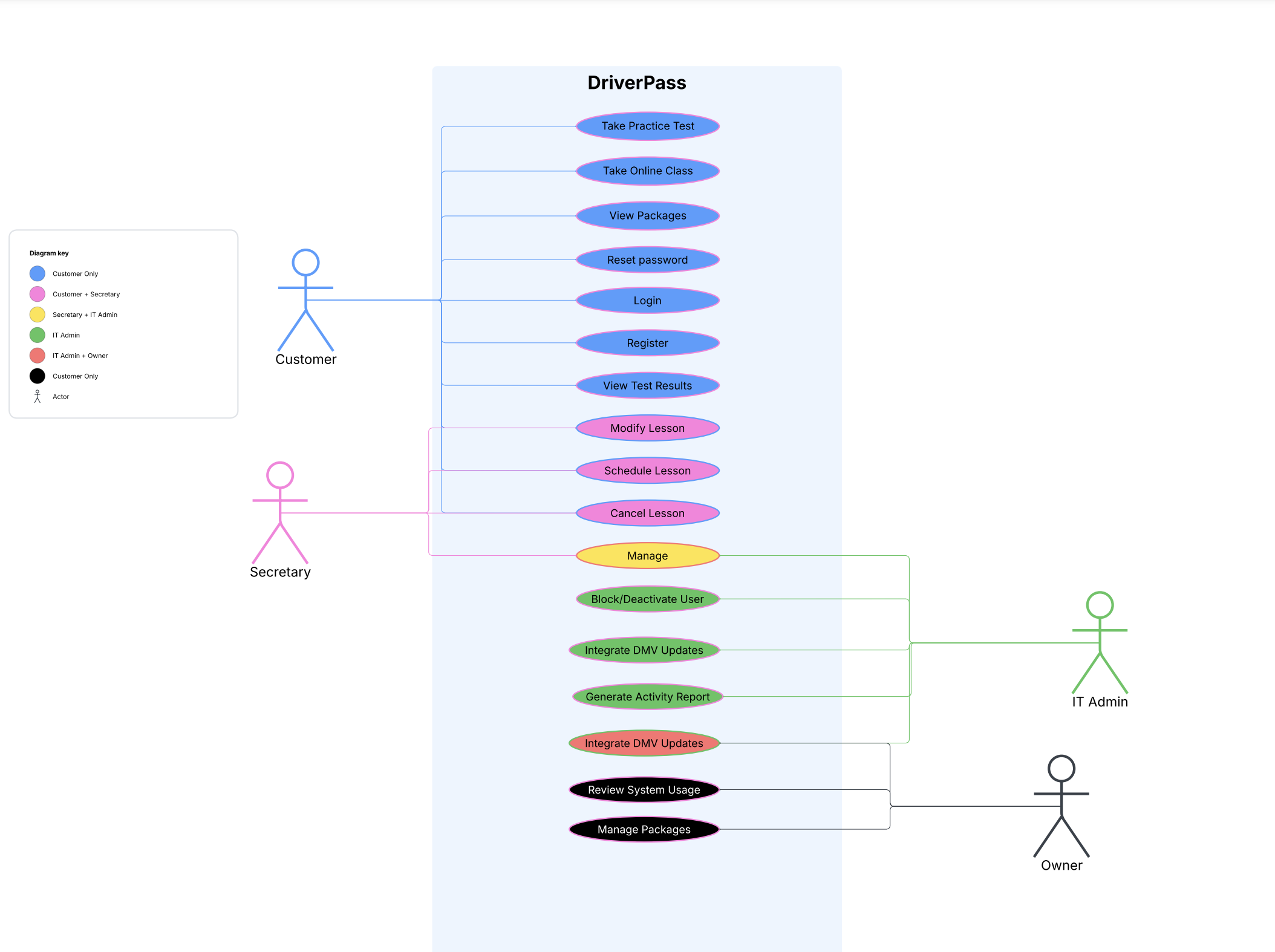
# [Tyler.Ellis1@snhu.edu](mailto:Tyler.Ellis1@snhu.edu)

# Professor Denise Washington

This template lays out all the different sections that you need to complete for Project Two. Each section has guidance to prompt your thinking. You will need to continually reference the interview transcript as you work to make sure that you are addressing your client’s needs. There is no required length for the final document. Instead, the goal is to complete each section based on what your client’s needs are. Remove this note when you are finished and replace all bracketed text with the relevant information.

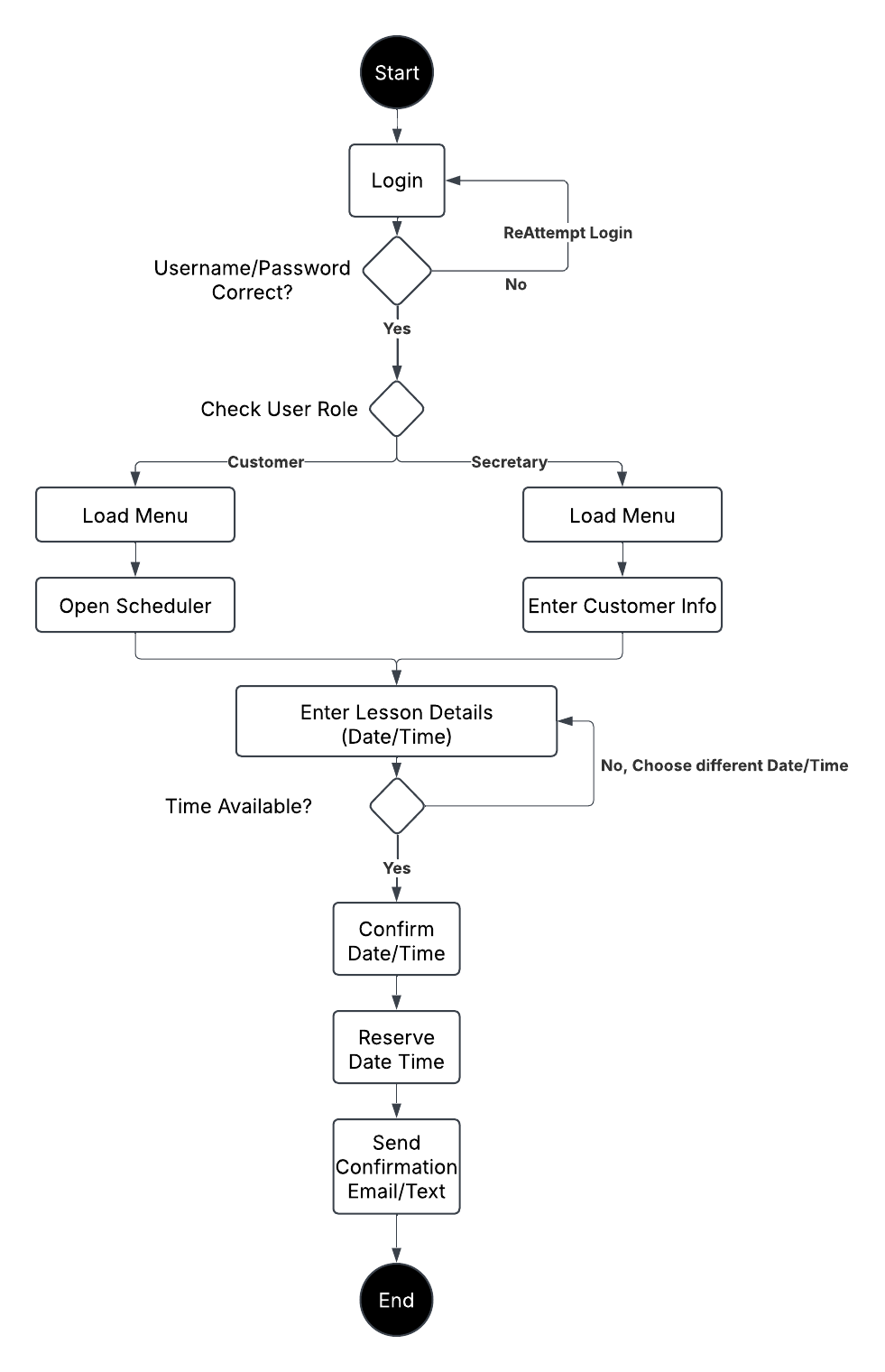
## UML Diagrams

### UML Use Case Diagram The use case diagram illustrates the various actors—Customer, Secretary, IT Admin, and Owner—and the system functions available to each. These include user registration, scheduling lessons, taking practice tests, integrating DMV updates, and generating reports.

  
Key Actors and Use Cases

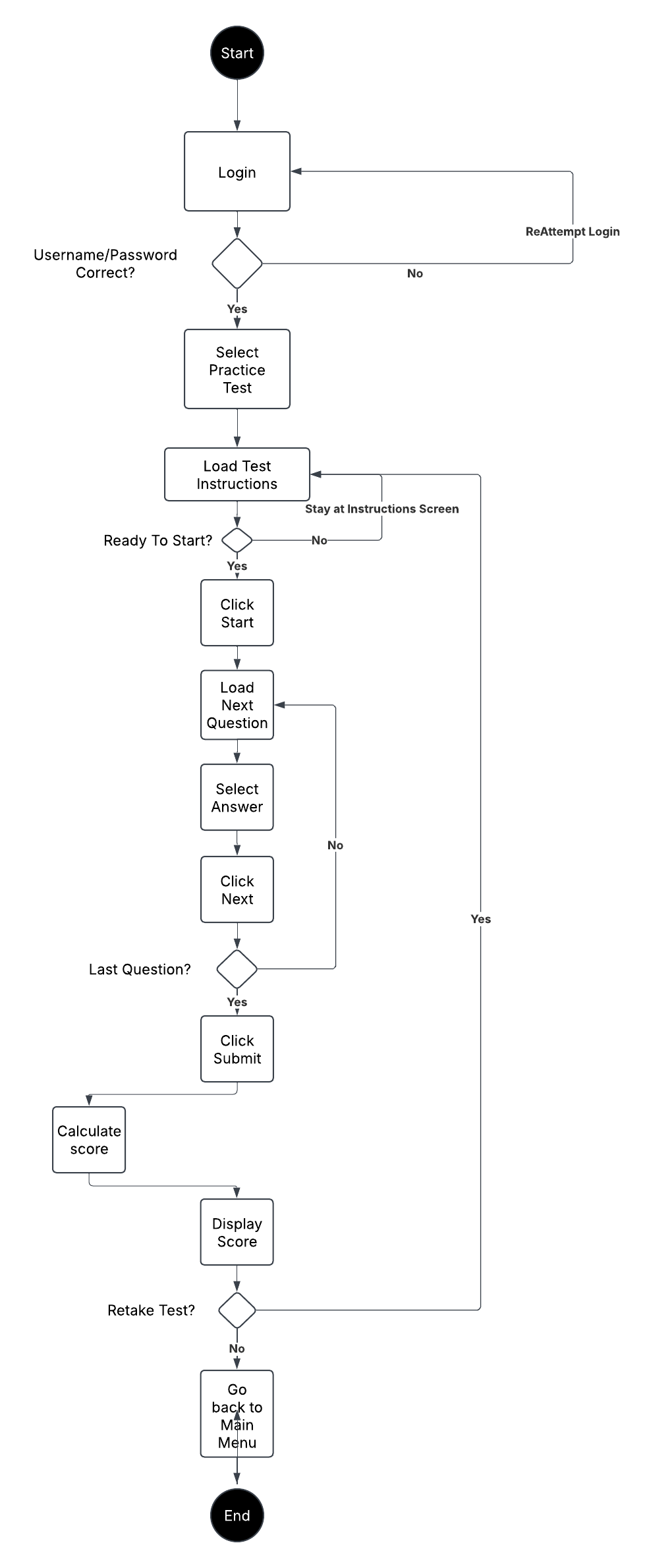
* Customer: Registers/logs in, schedules lesson, take practice tests, views test results.
* Secretary: Schedules lessons on behalf of customers, updates customer details, answers inquiries.
* IT Admin: Resets passwords, blocks users, integrates DMV updates, and oversees system operations.
* Owner: Has advanced privileges to enable or disable packages, review system usage, and view activity reports.

### UML Activity Diagrams



Workflow Summary

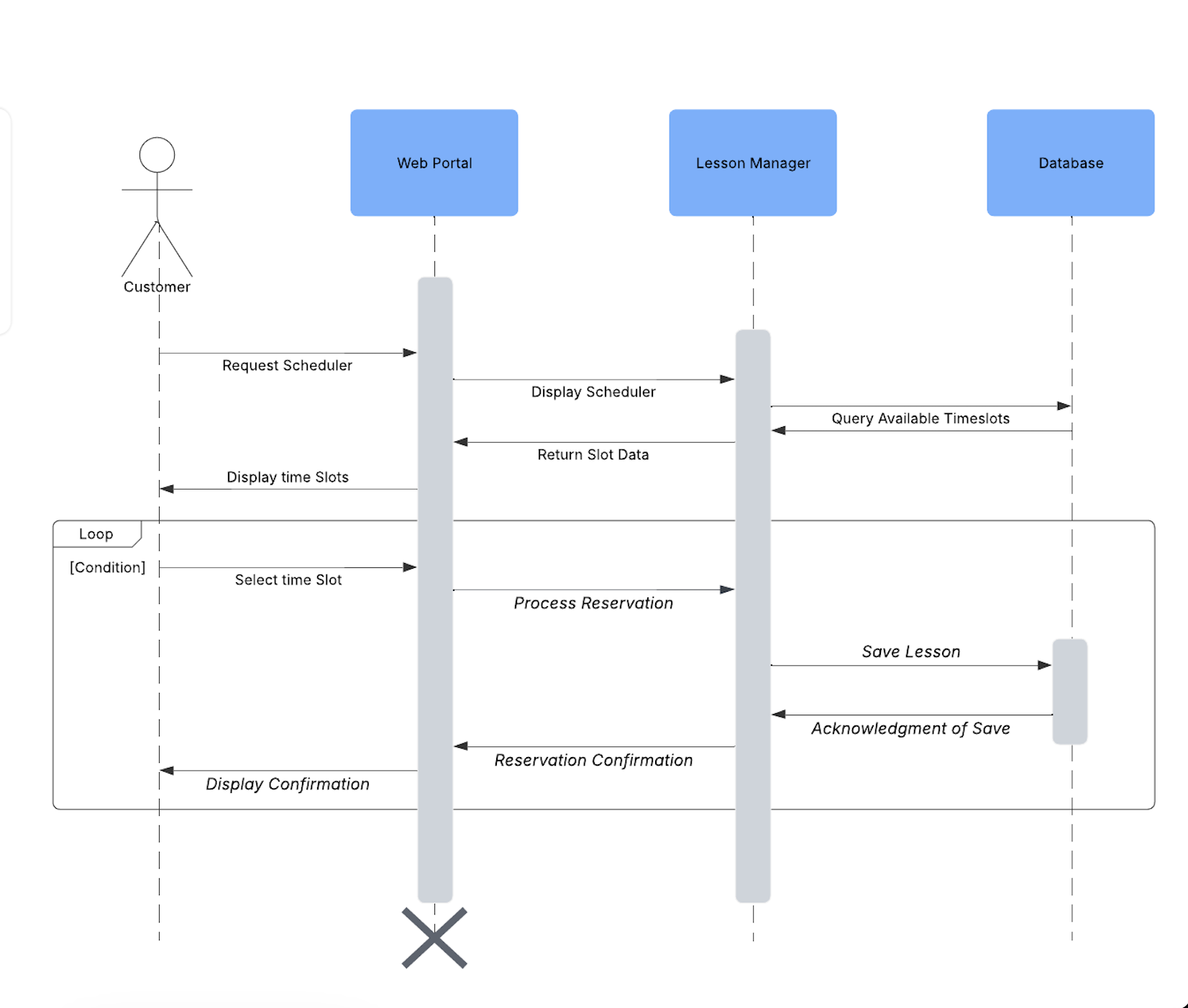
1. Start: The user (Customer or Secretary) Log into the service.
2. Role Verification: The System checks the user to determine which role to allow.
3. Menu Loading: The System loads the main menu.
4. Option Selection: The user selects the open schedule option.
5. Check Availability: The system verifies if a time slot is open.
6. Enter Details: The user confirms date, time, driver, or package if applicable.
7. Confirm Reservation: The system records the new lesson and notifies the user.
8. End: The lesson is scheduled, and the user may exit or continue with other tasks.



Workflow Summary

1. Start: Customer selects a practice test.
2. Load Questions: The system retrieves and displays test questions (updated by DMV).
3. Answer Questions: The customer completes the test.
4. Submit and Score: System calculates score, records the result, and updates progress.
5. End: User views results or exits the test module.

### UML Sequence Diagram



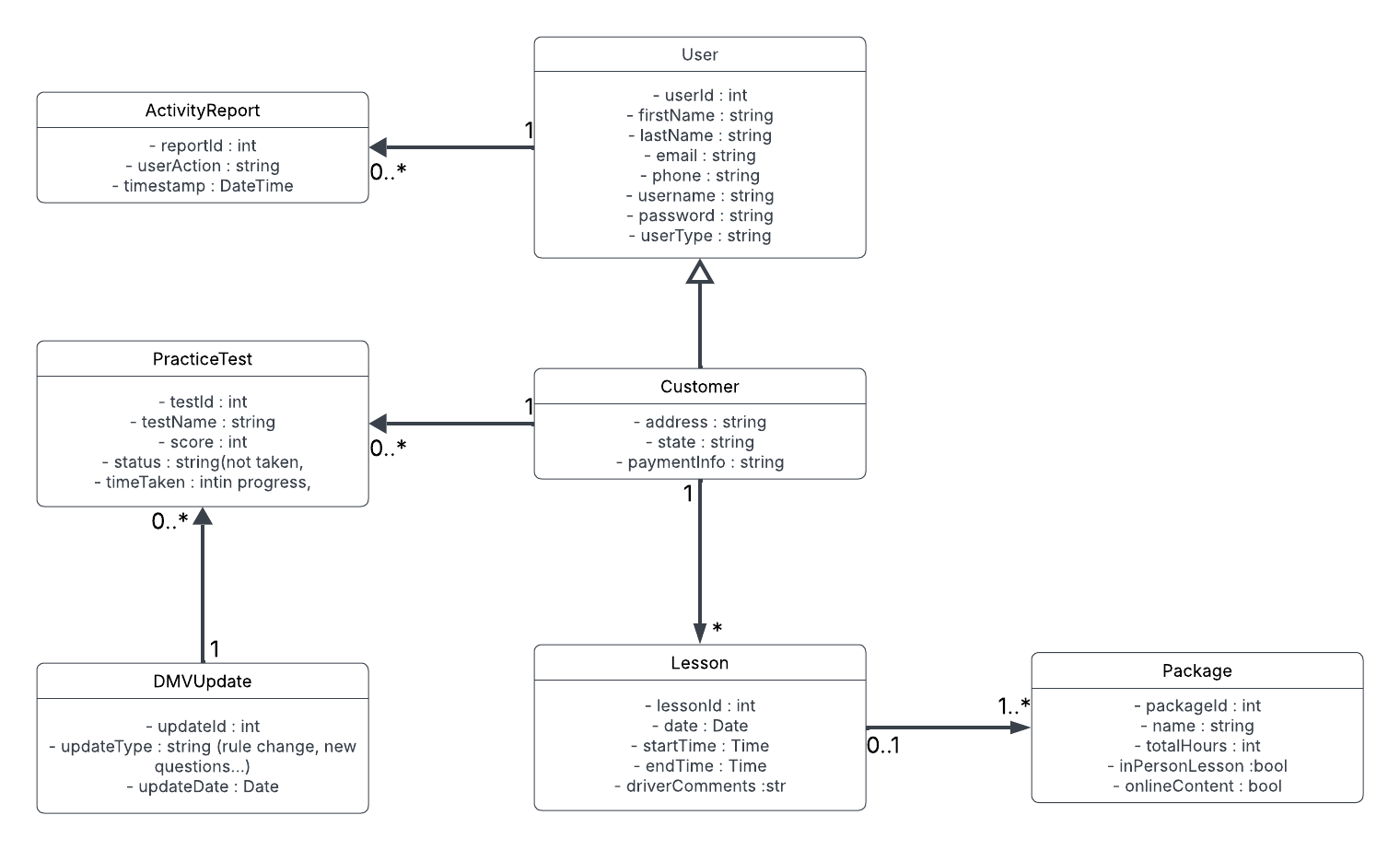
Participants

* Customer (or Secretary): Initiates scheduling action.
* System: Receives user requests and processes logic.
* Database: Stores lesson reservations, checks for conflicts.

Message Flow

1. Customer sends a “Request to Schedule Lesson” to the System.
2. System queries the Database for the requested time.
3. Database returns availability status.
4. Customer confirms the time slot.
5. System updates Database with the new reservation details.
6. Database confirms success; System notifies the Customer.

### UML Class Diagram



Classes and Relationships

* User: Parent class with attributes such as userId, username, password, userType.
* Customer (inherits from User): Adds address, state, paymentInfo.
* Lesson: Represents a scheduled driving lesson, including date, time, and optional driver comments. Linked to a Customer.
* Package: Specifies different training packages (6, 8, 12 hours, etc.). A Lesson may reference a Package.
* PracticeTest: Tracks the test name, score, time taken, status. Associated with a Customer.
* DMVUpdate: Maintains the information on new rules or policies from the DMV.
* ActivityReport: Logs user actions in the system for accountability and auditing.

## Technical Requirements

Hardware

* Cloud Hosting: Allocate virtual instances on a platform like AWS, Azure, or Google Cloud with minimum specs of 2–4 CPU cores and 8 GB RAM.
* Local Workstations: Each staff or developer workstation should have at least 8 GB RAM, a modern CPU, and reliable network connectivity.

Software

* Operating System: Linux for production (Ubuntu, CentOS, etc.) due to stability and security features.
* Database: MySQL or PostgreSQL to store user data, lessons, and test results.
* Web Server/Framework: Nginx or Apache for HTTP requests, and a back-end language (Java, Python, PHP, or C Sharp) to handle business logic.

Tools

* CASE Tool: Lucidchart for all UML diagram creation and maintenance.
* Version Control: Git for managing code changes and collaboration.
* Security and Monitoring: HTTPS/TLS encryption, role-based access, and logs to track suspicious activity.

Infrastructure

* Integration: A scheduled job or API that periodically checks for DMV updates and pulls new rules or question banks.
* Backup and Recovery: Automated daily or weekly backups of the database, stored offsite for redundancy.
* Maintenance: Routine patching and updates quarterly or sooner for critical security issues.