**CS-255 System Analysis And Design**

**Module 7**

**Project Two**

**System Design Document**

**Submitted By:**

HAMNA KHALID

ID: 2902671

[hamna.khalid@snhu.edu](mailto:hamna.khalid@snhu.edu)

**Submitted To:**

PROFESSOR PARUL HIRPARA

p.hirpara@snhu.edu

Southern New Hampshire University

Thursday, October 16, 2025

**System Design Document**

This System Design Document (SDD) provides a comprehensive overview of the proposed design for the **DriverPass** system. It builds upon the Business Requirements Document (BRD) completed earlier in the course, translating the client’s requirements into a visual and technical blueprint. The document contains five key **UML diagrams**: a Use Case Diagram, two Activity Diagrams, a Sequence Diagram, and a Class Diagram, each representing different perspectives of the DriverPass system. It also outlines the **technical requirements**, including hardware, software, and infrastructure specifications, necessary for successful system implementation.

This design ensures that DriverPass can deliver a secure, user-friendly, and efficient platform for students preparing for DMV exams, while also providing robust administrative and scheduling tools for staff.

# **Purpose**

The purpose of this document is to clearly define how the DriverPass system will function, interact, and perform once implemented. It serves as a technical roadmap for developers and a reference for stakeholders to visualize the system’s processes. By including structured models and detailed requirements, this document bridges the gap between business needs and technical design, ensuring alignment between the client’s expectations and the final product.

The SDD will:

* Illustrate **user interactions** and system behavior using UML diagrams.
* Describe **core workflows** such as scheduling driving lessons and taking online practice tests.
* Define **system structure** through class modeling and object relationships.
* Specify **technical, software, and security requirements** essential for deployment.

# **System Overview**

DriverPass is a blended online and on-the-road training system designed to reduce DMV test failure rates. The system enables students to prepare for exams through interactive learning materials and hands-on driving lessons, all while allowing administrators and instructors to efficiently manage users, schedules, and compliance with DMV regulations.

This document focuses on four primary areas of system design:

1. **Use Case Diagram**:

Outlines major user interactions and system boundaries.

1. **Activity Diagrams**:

Detail step-by-step workflows for key use cases.

1. **Sequence Diagram**:

Demonstrates the order of operations and communications between components.

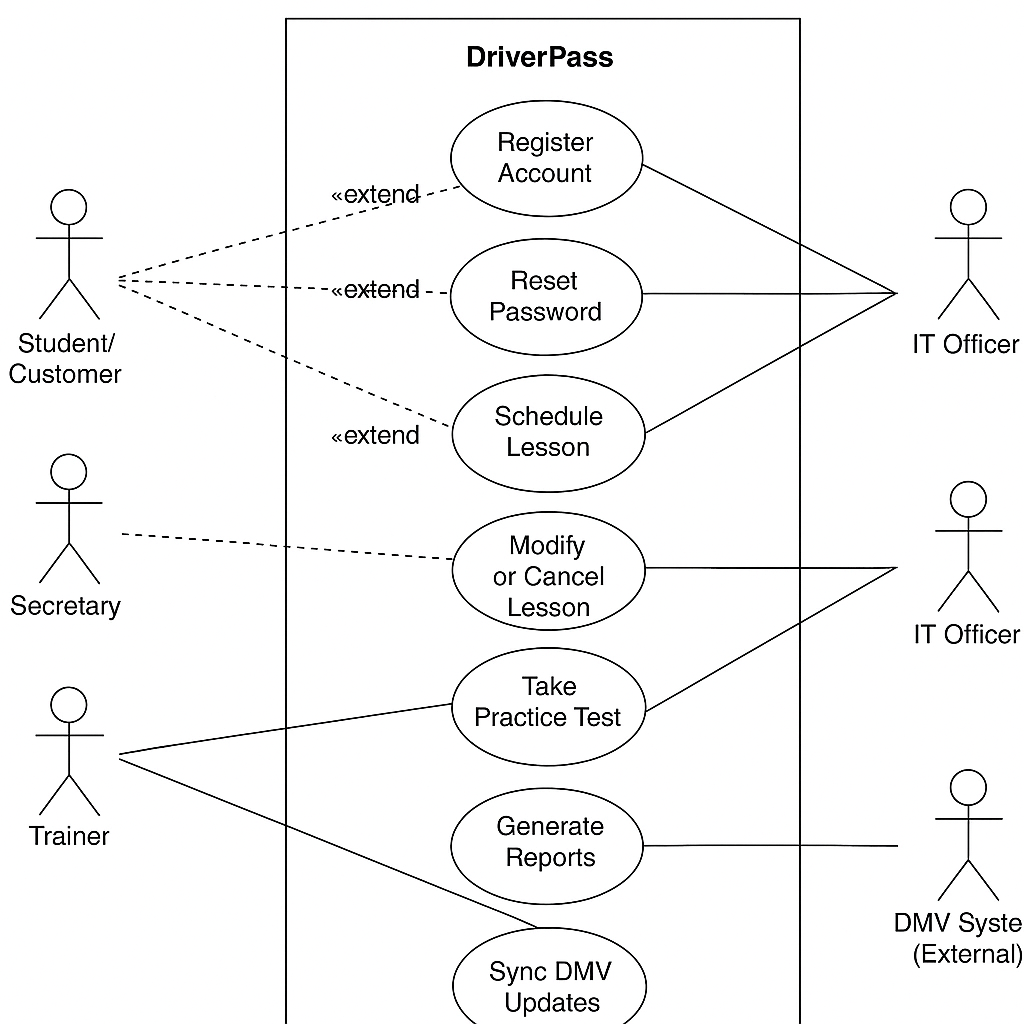
1. **Class Diagram**:

Defines the system’s object structure and relationships.

# **UML Diagrams**

## **UML Use Case Diagram**

The UML Use Case Diagram provides an overview of how users interact with the DriverPass system. It identifies the main actors and their associated functions, illustrating the boundaries of the system and shared dependencies between actions.



**Fig 1: Use Case Diagram**

**Actors** include:

* **Student/Customer:**

Registers, logs in, schedules lessons, takes practice exams, and tracks progress.

* **Secretary:**

Manages reservations and customer records for in-person and phone bookings.

* **Trainer:**

Records lesson completion and student performance notes.

* **IT Officer:**

Handles system configuration, user management, and password resets.

* **DMV System (External):**

Sends updates on laws, test questions, and driving regulations.

**Key Use Cases:**

The key use cases are as follows:

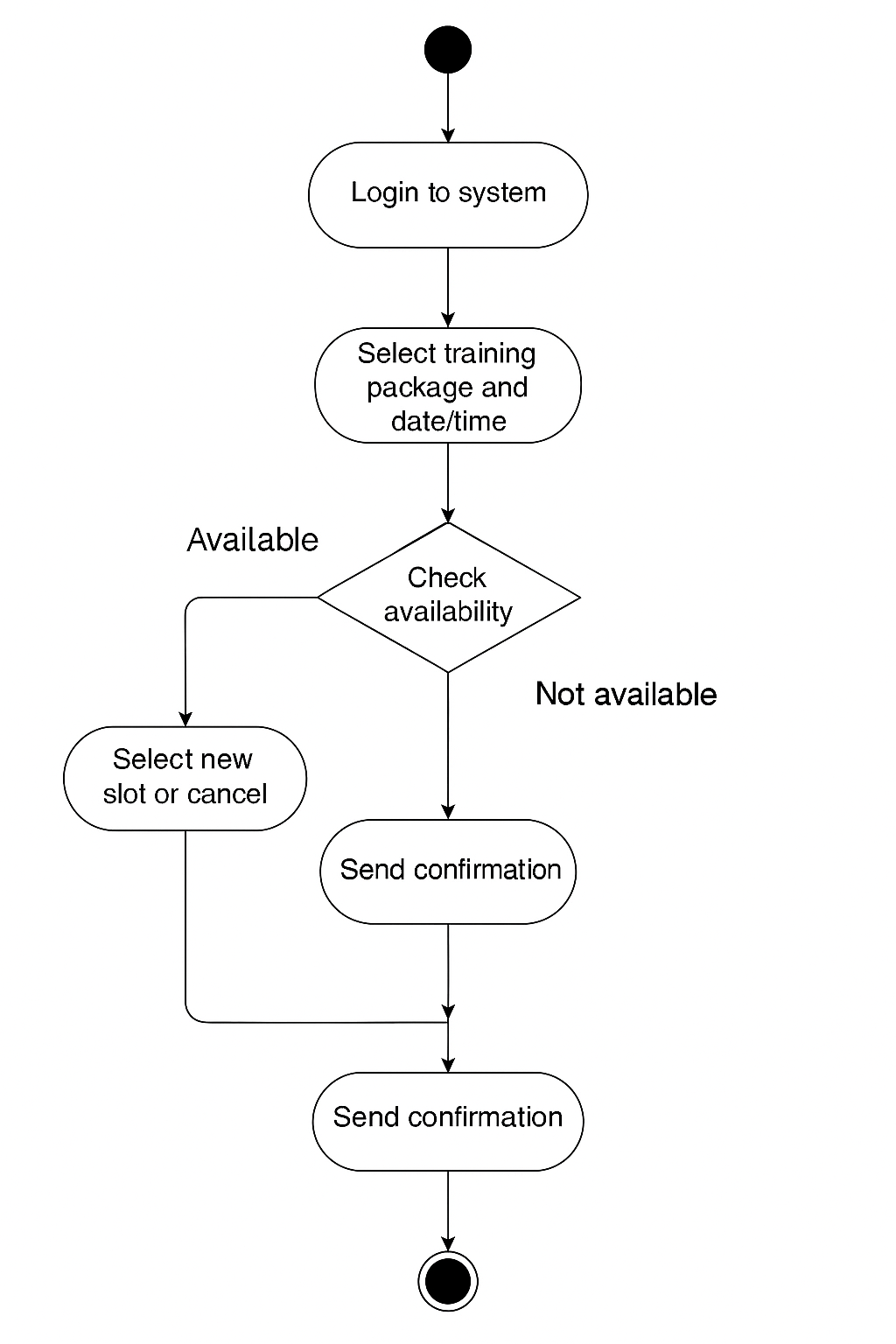
* Register Account
* Login
* Reset Password
* Schedule Lesson
* Modify or Cancel Lesson
* Take Practice Test
* View Progress
* Manage Accounts
* Generate Reports
* Sync DMV Updates

## **UML Activity Diagrams**

Activity Diagrams describe the internal flow of actions for specific processes in the DriverPass system. Each diagram focuses on a single workflow, showing decisions, concurrency, and outcomes.

### **2.1 Activity Diagram: Schedule Lesson**

This activity diagram represents how students or secretaries schedule driving lessons. The process begins when a user logs into the system and selects a training package. The system checks trainer and vehicle availability. If a suitable slot exists, it books the lesson and sends confirmation; otherwise, it prompts the user to choose another time or cancel.

****

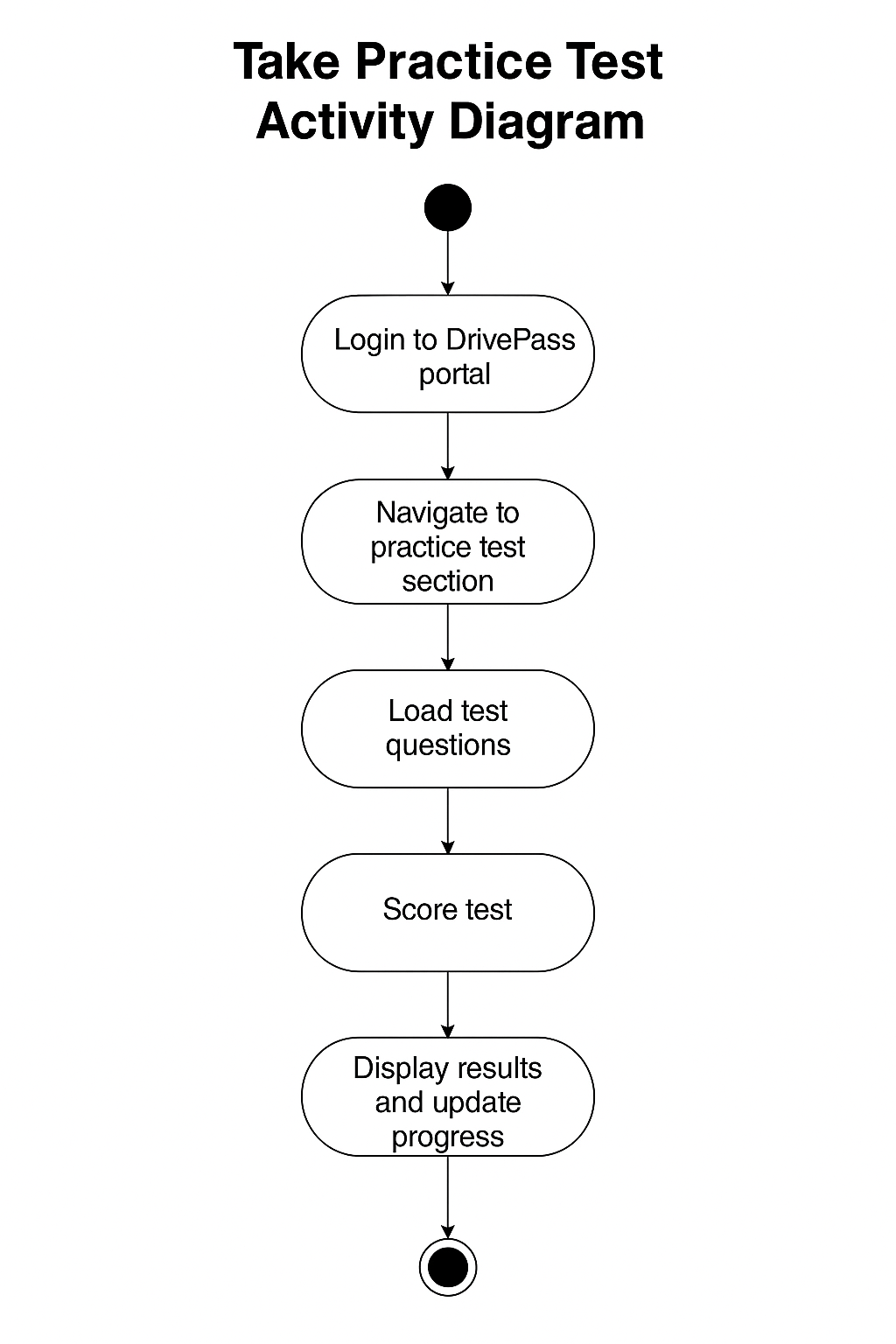
**Fig 2.1:Activity Diagram ( Schedule Lesson )**

Key flow steps for this activity are as follows:

1. Login to the system.
2. Select training package and desired date/time.
3. System checks availability of trainers and vehicles.
4. If available, the reservation is confirmed and stored in the database.
5. User receives confirmation via the system or email.
6. If unavailable, the user may select a new slot or cancel.

### **2.2 Activity Diagram: Take Practice Test**

This activity diagram illustrates the process of taking an online practice test within DriverPass. It shows the logical flow from starting the test to receiving a score and progress update.

****

**Fig 2.2:Activity Diagram ( Take Practice Test )**

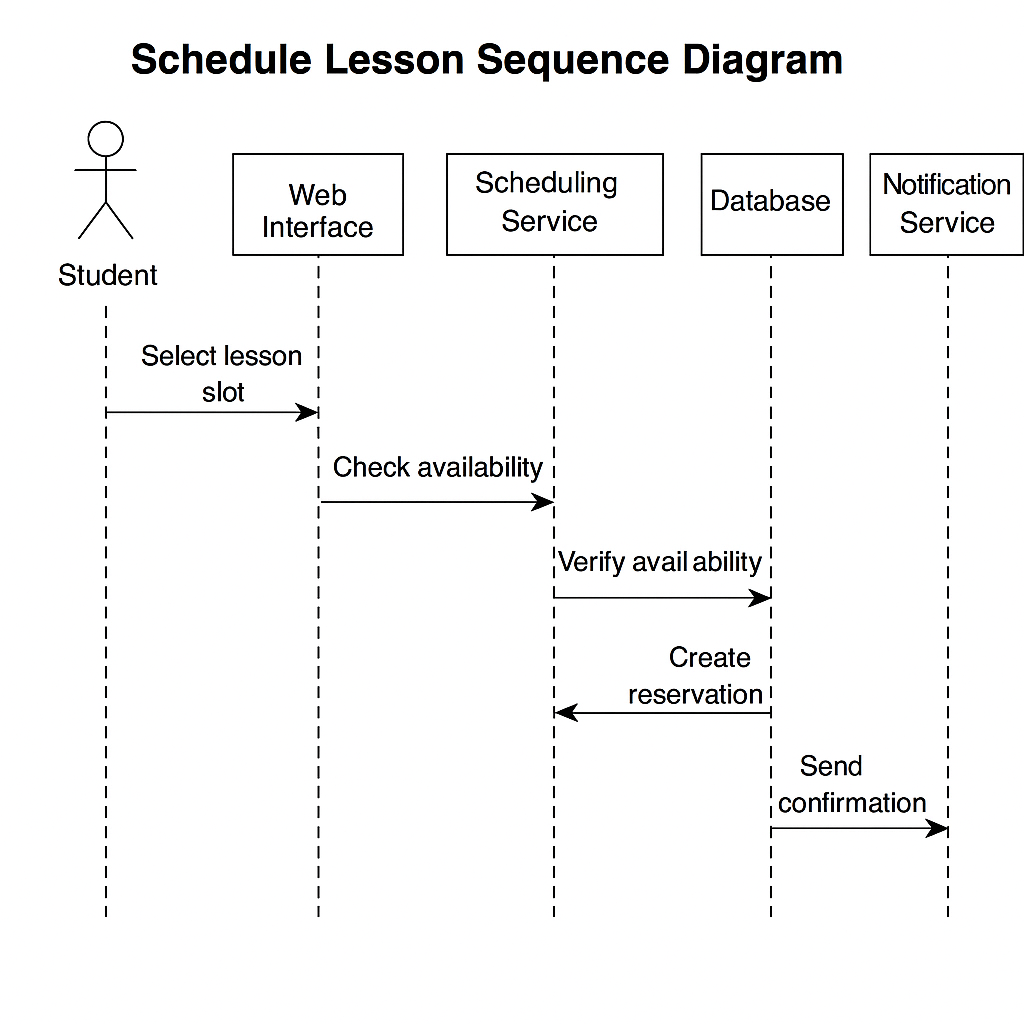
Key flow steps for this activity are as follows:

1. Student logs into the DriverPass portal.
2. Navigates to the practice test section.
3. System loads test questions and tracks responses.
4. After submission, the system scores the test.
5. Results are displayed, and progress is updated in the student’s profile.

This process enhances user engagement and provides students with instant feedback, improving their learning outcomes.

## **UML Sequence Diagram**

The Sequence Diagram shows how different system components communicate to complete the “**Schedule Lesson**” use case. It represents the flow of messages between the Student, Web Interface, Scheduling Service, Database, and Notification Service.

****

**Fig 3: Sequence Diagram ( Schedule Lesson )**

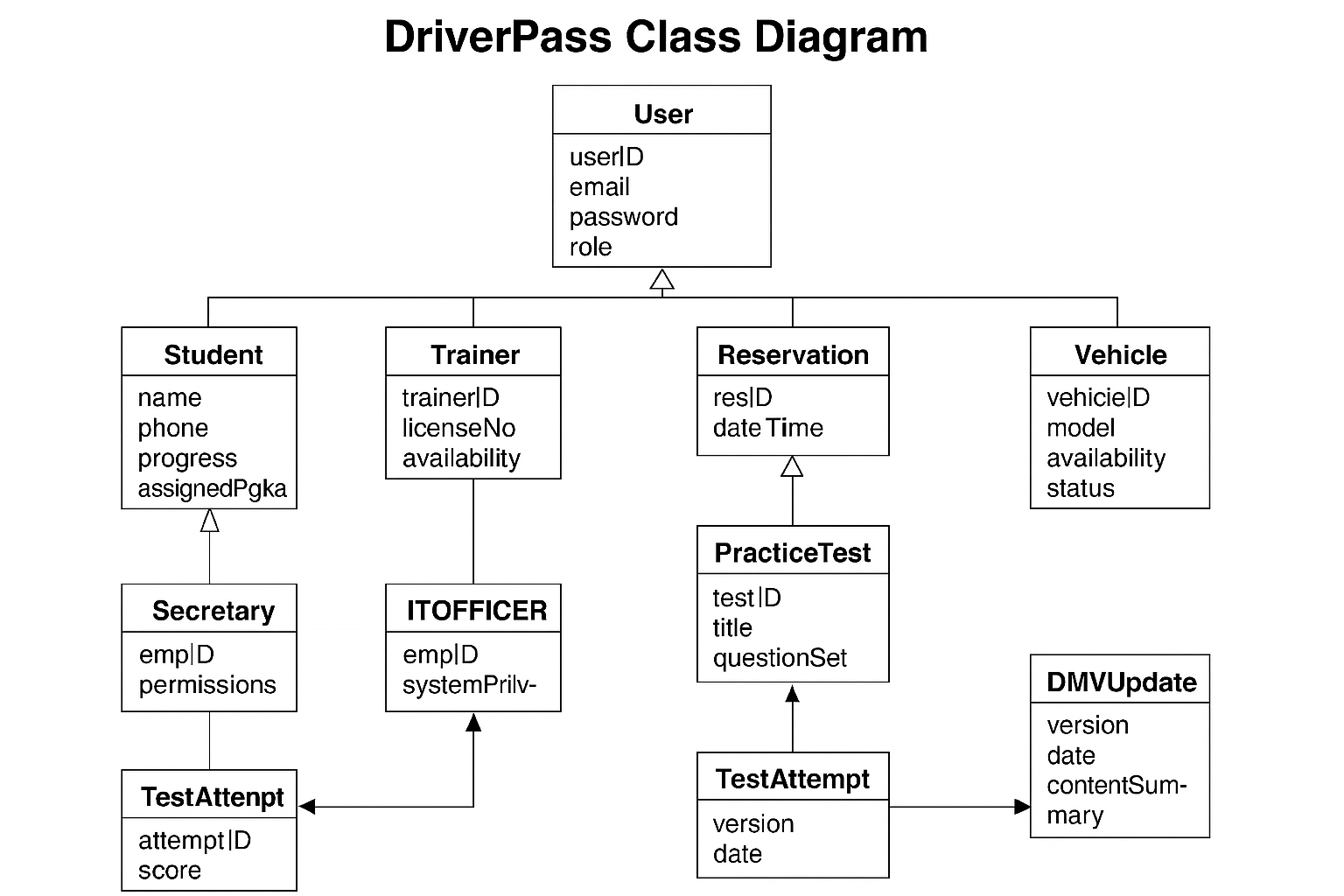
The flow summary for this sequence diagram is as follows:

1. Student selects a lesson slot via the web interface.
2. System checks trainer and vehicle availability.
3. Database verifies scheduling constraints.
4. If successful, a reservation is created.
5. The system sends confirmation to the student.

This sequence ensures efficient, error-free scheduling and keeps all stakeholders informed.

## **UML Class Diagram**

The Class Diagram defines the core objects, attributes, and relationships that make up the DriverPass system. It establishes the static structure upon which all dynamic actions occur.

****

**Fig 4: Class Diagram**

### **Key Classes:**

The key classes for this class diagram are as follows:

* **User (Base Class):**

userID, email, password, role

* **Student (Subclass):**

name, phone, progress, assignedPackage

* **Trainer (Subclass):**

trainerID, licenseNo, availability

* **Secretary (Subclass):**

empID, permissions

* **ITOfficer (Subclass):**

empID, systemPrivileges

* **Reservation:**

resID, dateTime, trainer, vehicle, status

* **Vehicle:**

vehicleID, model, availability, status

* **PracticeTest:**

testID, title, questionSet

* **TestAttempt:**

attemptID, score, duration, status

* **DMVUpdate:**

version, date, contentSummary

### **Relationships:**

The relationship between core objects is as follows:

* *User → Student/Trainer/Secretary/ITOfficer* (Inheritance)
* *Student 1..*\*\* → Reservation (Association)
* *Reservation 1* → Trainer, Vehicle (Aggregation)
* *Student 1..*\*\* → TestAttempt (Composition)
* *TestAttempt 1* → PracticeTest (Association)

# **Technical Requirements**

The technical design ensures that DriverPass operates securely, efficiently, and scalably across all supported platforms. These requirements define the necessary environment, tools, and standards to ensure proper deployment and long-term maintainability.

## **Hardware Requirements**

* Cloud-based hosting (AWS EC2, Azure App Service, or equivalent).
* Redundant servers with automated failover and daily backups.
* End-user access via desktops, laptops, tablets, or smartphones.
* Reliable internet connectivity with secure HTTPS protocol.

## **Software Requirements**

* **Frontend:**

HTML5, CSS3, JavaScript (React or Angular recommended).

* **Backend:**

Python (Flask/Django) or Java (Spring Boot).

* **Database:**

MySQL or PostgreSQL relational database.

* **API Integration:**

RESTful endpoints for DMV sync and notifications.

* **Operating Systems:**

Windows, macOS, Linux (cross-compatible).

* **Browsers Supported:**

Chrome, Firefox, Edge, Safari.

## **Security Requirements**

* All communication secured via HTTPS (SSL/TLS).
* Strong password encryption (bcrypt or AES-256).
* Multi-factor authentication for staff accounts.
* Automatic account lock after 3 failed attempts.
* Role-Based Access Control (RBAC) for Students, Staff, and Admins.
* Logging and auditing of all system activities.
* Third-party secure payment gateway integration (PCI DSS compliant).

## **Infrastructure And Deployment**

* Hosted in a **cloud-native environment** with 99.9% uptime.
* Continuous Integration/Continuous Deployment (CI/CD) pipelines.
* API gateway for DMV and notification services.
* Auto-scaling for performance optimization.
* Data encryption at rest and in transit.

## **Maintenance And Scalability**

* Modular architecture for easy updates.
* Scalable backend for increasing student load.
* Support for future mobile app integration.
* Monitoring through AWS CloudWatch or Azure Monitor.

# **Conclusion**

The DriverPass system design effectively combines process and object modeling to deliver a flexible and reliable solution that meets all client needs. The UML diagrams define clear user interactions, workflows, and structural relationships, ensuring developers have a precise roadmap for implementation.

Through its technical design, DriverPass guarantees **secure access**, **real-time DMV updates**, **efficient scheduling**, and **user-friendly operation**. The design is scalable and future-ready, ensuring DriverPass can evolve as the organization grows and technology advances.

# **Acronyms And Full Forms**

This section explains the abbreviations used throughout the document to ensure clarity for all readers.

* DMV – Department Of Motor Vehicles
* **UML** – Unified Modelling Language
* **SDD** – System Design Document
* **BRD** – Business Requirements Document
* **UI** – User Interface
* **API** – Application Programming Interface
* **RBAC** – Role-Based Access Control
* **CI** – Continuous Integration
* **CD** – Continuous Deployment
* **SNHU** – Southern New Hampshire University

# **References**

* SNHU. (2025). *CS-255 Module Five: Project One: Business Requirements Document.*
* SNHU. (2025). *CS-255 Project Two Guidelines and Rubric.*
* Object Management Group (OMG). (2024). *Unified Modeling Language (UML) 2.5.1 Specification.*
* AWS Architecture Center. (2024). *Best Practices for Cloud Security and Scalability.*
* DriverPass Client Interview Transcript (2025). Southern New Hampshire University.