# DriverPass System Design Document

## UML Diagrams

### UML Use Case Diagram

The Use Case Diagram identifies the different user roles—Student, Secretary, IT Officer, DMV, and Driver—and their interactions with the system. It showcases how the system enables functionalities such as scheduling lessons, tracking progress, and receiving updates. By visualizing these relationships, we ensure all client requirements are addressed in a user-friendly manner.

A diagram of a driver pass

Description automatically generated

### UML Activity Diagrams

1. Scheduling Lessons

This diagram outlines how students and secretaries can schedule lessons. It starts with logging into the system and ends with confirmation of the booking. This clear step-by-step process ensures ease of use and flexibility for different user needs.

A screenshot of a computer

Description automatically generated

1. Tracking Test Progress

This diagram shows how students log into the system to check their progress. It includes key actions such as viewing test results, tracking status (e.g., passed, failed, or in progress), and reviewing feedback. This helps students stay informed and better prepared for their tests.

A screenshot of a diagram

Description automatically generated

### UML Sequence Diagram

The sequence diagram illustrates the interactions between the student, the system, and the database during lesson scheduling. It demonstrates how the system confirms availability, processes the booking, and provides confirmation to the user, ensuring seamless communication.

A diagram of a diagram

Description automatically generated

### UML Class Diagram

The class diagram organizes the main components of the system, including:

* User: Represents students, secretaries, and IT officers with details such as name, role, and contact info.
* Lesson: Includes lesson details like date, time, driver, and student.
* Package: Covers different packages available, including hours and costs.
* Test: Tracks test details such as scores, status, and completion time.
* DMVUpdate: Stores updates from the DMV, including rules and requirements. This structure ensures that the system is well-organized and efficient.

A diagram of a computer

Description automatically generated

## Technical Requirements

**Hardware**

* **Servers**: The system requires a web server for hosting and a database server to store user, lesson, and test data. These servers are essential for ensuring the system is accessible and reliable for all users.
* **User Devices:** The system supports desktops, tablets, and smartphones, allowing users to access the platform from a variety of devices for convenience and ease of use.

**Software**

* **Operating System**: Linux servers are chosen for their stability, scalability, and cost-effectiveness, ensuring smooth operation of the system.
* **Database**: MySQL or PostgreSQL is used for managing relational data, ensuring reliable and efficient storage of user, lesson, and test information.
* **Programming Language**: Python or Java is selected for backend development, providing robustness and flexibility for system functionalities.
* **Web Framework:** Django or Flask is used to create a user-friendly and interactive interface, enhancing the overall user experience.

**Tools**

* **Development Tools**: Integrated Development Environments (IDEs) such as IntelliJ IDEA or Visual Studio Code are utilized to write and debug code effectively, streamlining the development process.
* **Diagram Tools**: Lucidchart is used for creating clear and professional UML diagrams, helping communicate the system’s design effectively.
* **Version Control**: Git is employed for tracking code changes and enabling collaboration among team members, ensuring efficient project management.

**Infrastructure**

* **Cloud Hosting**: Hosting on platforms like AWS or Azure ensures scalability, high availability, and reliable data backups, reducing risks of data loss.
* Security:
  + Role-based access control (RBAC) ensures users only access features relevant to their roles, protecting sensitive data.
  + Encryption (TLS) safeguards data transmitted between users and the system, ensuring privacy and security.
  + Secure password storage and recovery mechanisms protect user accounts and enhance trust in the system.

This design balances functionality, security, and scalability to meet the client's needs while being prepared for future growth.