# CS 255 System Design Document Template

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

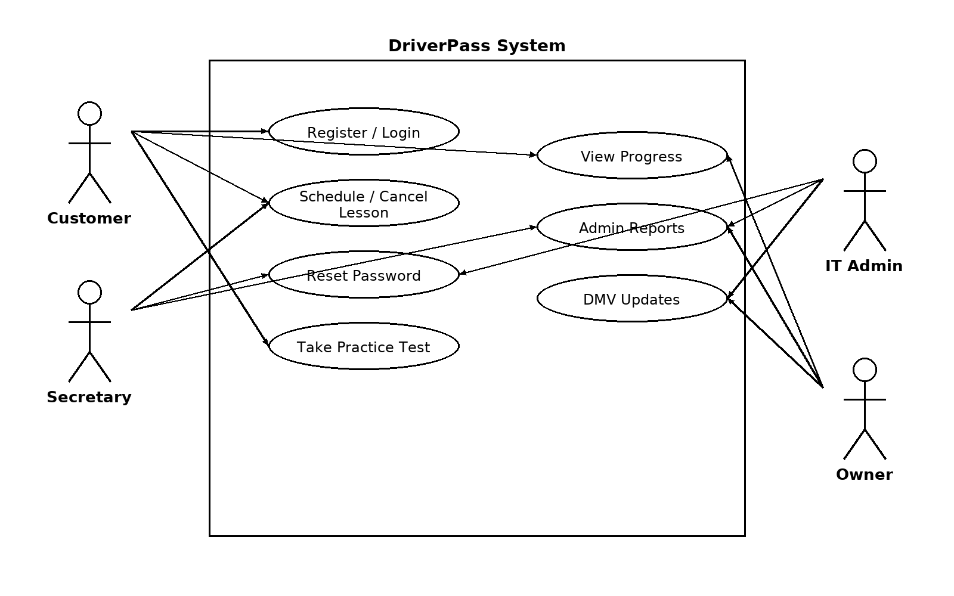
### UML Use Case Diagram

**Actors:**

* Customer
* Secretary
* IT Administrator
* Owner/Manager
* DMV System (external actor)

**Use Cases:**

* Register / Login
* Reset Password
* Schedule Lesson
* Cancel/Modify Lesson
* Take Practice Test
* View Progress
* Generate Activity Report
* Receive DMV Updates



### UML Activity Diagrams

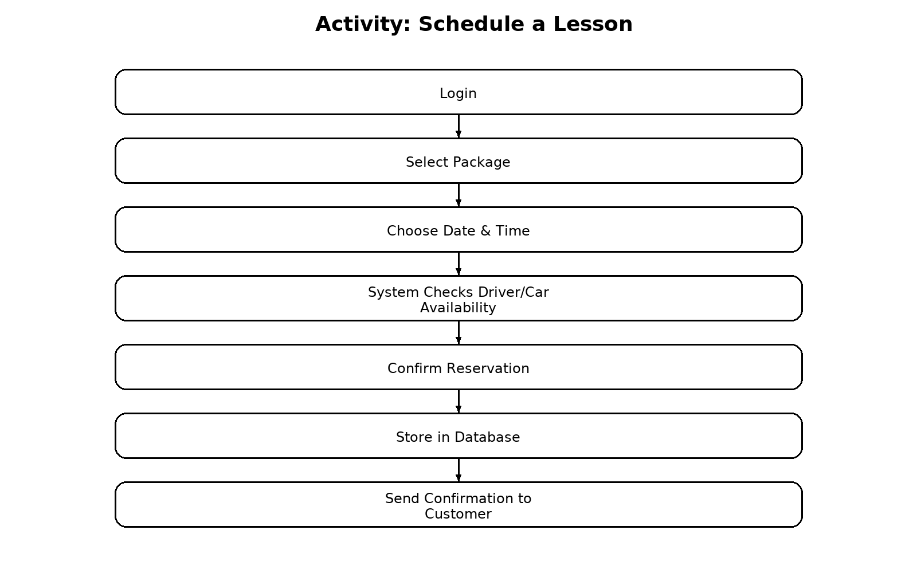
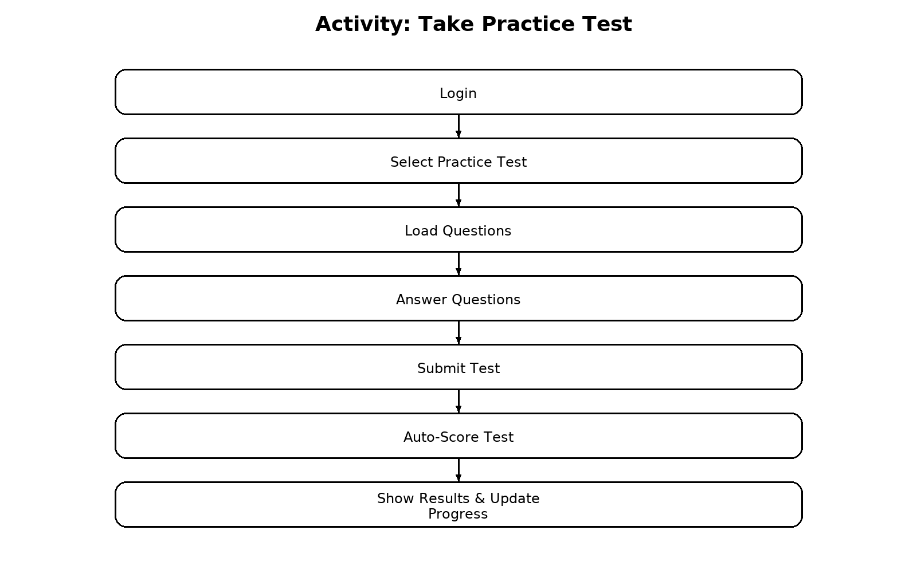
**Activity Diagram 1: Schedule a Lesson**

* Customer logs in
* Selects training package
* Chooses date and time for lesson
* System checks driver and car availability
* Reservation confirmed and stored in database
* Customer receives confirmation

(Insert Lucidchart activity diagram for Scheduling a Lesson here)

**Activity Diagram 2: Take Practice Test**

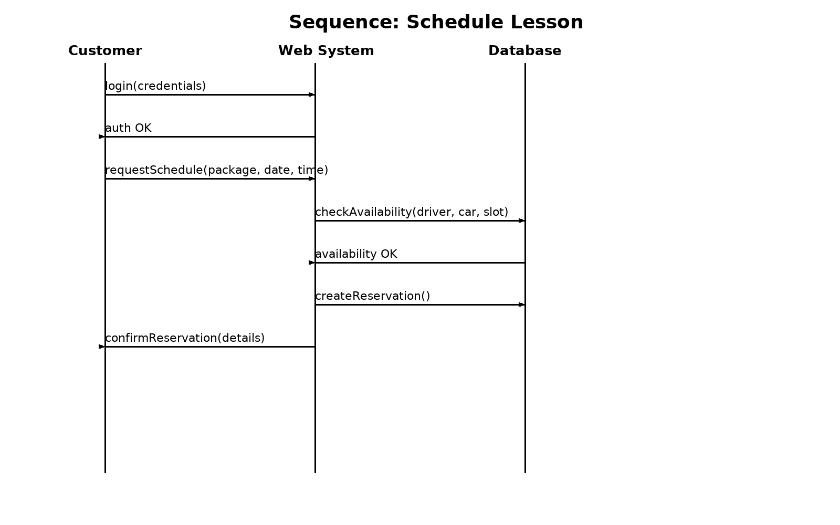
* Customer logs in
* Selects a practice test
* System loads test questions
* Customer completes answers
* System scores test
* Result displayed to customer
* Progress recorded in system



### UML Sequence Diagram

**Sequence: Schedule Lesson**

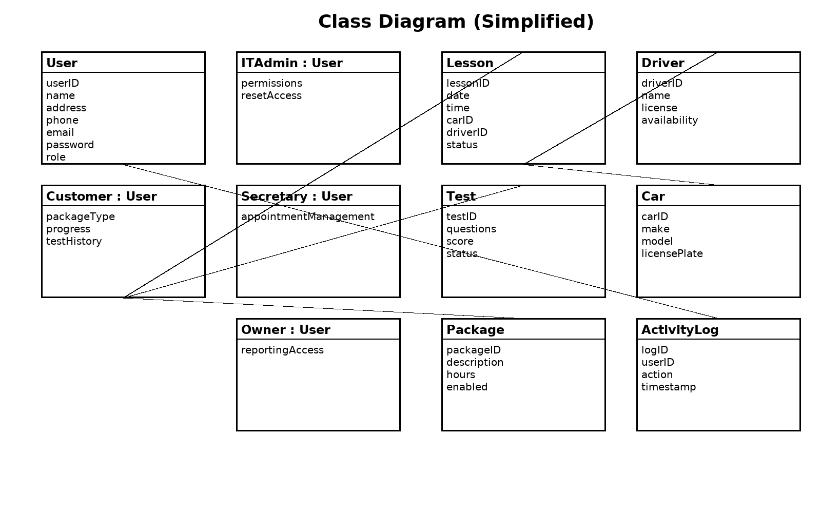
1. **Customer** → Login → **System**
2. **Customer** → Request Lesson → **System**
3. **System** → Check Availability → **Database**
4. **Database** → Return Availability → **System**
5. **System** → Confirm Lesson → **Customer**
6. **System** → Update Reservation → **Database**

****

### UML Class Diagram

**Classes and Attributes:**

* **User**: userID, name, address, phone, email, password, role
* **Customer (extends User)**: packageType, progress, testHistory
* **Secretary (extends User)**: appointmentManagement
* **ITAdmin (extends User)**: permissions, resetAccess
* **Owner (extends User)**: reportingAccess
* **Lesson**: lessonID, date, time, carID, driverID, status
* **Driver**: driverID, name, license, availability
* **Car**: carID, make, model, licensePlate
* **Test**: testID, questions, score, status
* **Package**: packageID, description, hours, enabled/disabled
* **ActivityLog**: logID, userID, action, timestamp



## Technical Requirements

**Hardware**

* Cloud-hosted infrastructure with scalable web servers.
* Secure storage for customer and scheduling data.
* Backup servers for redundancy and disaster recovery.

**Software**

* Web application stack (HTML, CSS, JavaScript frontend; backend in Java, Python, or Node.js).
* Relational database (MySQL or PostgreSQL) to manage users, reservations, and packages.
* Secure API integration with DMV for updates and compliance.

**Tools**

* Lucidchart for UML modeling.
* GitHub/Bitbucket for version control.
* Encryption tools for secure credential storage and data exchange.
* Monitoring and logging tools to track activity and system performance.

**Infrastructure**

* Cloud deployment (AWS, Azure, or Google Cloud).
* HTTPS protocol for all data communication.
* Role-based access control for employees and customers.
* Password recovery system with secure email verification.
* Audit logging for user and administrator actions.
* Regular system updates, patches, and weekly backups.

**Summary**

For DriverPass, this design offers a web-based platform that facilitates secure user management, online testing, lesson scheduling, and DMV updates. It is user-friendly for administrators, employees, and customers while satisfying both functional and nonfunctional requirements, offering role-based permissions, and guaranteeing adherence to DMV standards.