# CS 255 System Design Document Template

## UML Diagrams

### UML Use Case Diagram

The DriverPass Use Case Diagram models the system interactions between users and the platform.  
**Actors:**

* **Student:** Registers an account, logs in, accesses training materials, takes practice tests, and schedules driving lessons or DMV tests.
* **Instructor/Admin:** Manages student progress, schedules, and availability; oversees the course content.
* **System:** Validates user credentials, stores results, and updates data automatically.

**Description:**  
The use case diagram shows how each user interacts with the system’s main features. For example, a student logs in, accesses online materials, and books a driving test. The instructor can view or modify schedules. This diagram highlights the system’s primary goal—allowing students to learn, practice, and schedule their tests efficiently through a single online platform.



### UML Activity Diagrams

#### Diagram 1:

#### This diagram details the sequence of steps a student takes to schedule a driving test.

**Process Summary:**

1. The student logs in to the DriverPass system.
2. The system verifies credentials and loads the student dashboard.
3. The student selects “Schedule Driving Test.”
4. The system checks available time slots and instructor availability.
5. The student selects a preferred date and time.
6. The system confirms the appointment and sends a notification.
7. The database records the scheduled test.

**Purpose:**  
This diagram demonstrates how DriverPass automates scheduling, reducing administrative workload and helping students manage appointments more efficiently.

#### **Diagram 2:**

This diagram outlines the steps a student follows to access and complete online learning modules.

**Process Summary:**

1. The student logs in to their DriverPass account.
2. The system verifies credentials and displays the learning dashboard.
3. The student selects a course or practice test.
4. The system retrieves and loads the content from the database.
5. The student completes lessons or exams.
6. The system saves the results and updates progress records.

**Purpose:**  
This activity diagram highlights the seamless process for students to access and complete training materials, ensuring a user-friendly and consistent learning experience.



### UML Sequence Diagram

#### **Sequence Diagram for Schedule Driving Test**

This diagram illustrates the communication flow between the **Student, System Interface, Database,** and **Instructor**.

**Interaction Summary:**

1. The student requests to schedule a driving test through the system interface.
2. The system sends a query to the database for available time slots.
3. The database returns available options.
4. The student selects a preferred date and time.
5. The system confirms the selection and sends an update to the instructor’s schedule.
6. The instructor is notified, and the confirmation is stored in the database.
7. The system displays a confirmation message to the student.

**Purpose:**  
This diagram demonstrates real-time communication between users and the system to ensure accurate scheduling and confirmation. It shows the logical order of interactions and how data flows between components.



### UML Class Diagram

The UML Class Diagram represents the system’s structure by showing the main classes, their attributes, and relationships.

**Classes and Attributes:**

* **Student**
  + Attributes: studentID, name, email, password, progressRecord
  + Relationships: Can schedule DrivingTest, access TrainingMaterial
* **Instructor**
  + Attributes: instructorID, name, availability, contactInfo
  + Relationships: Oversees DrivingTest
* **DrivingTest**
  + Attributes: testID, date, time, location, status, studentID
  + Relationships: Linked to Student and Instructor
* **TrainingMaterial**
  + Attributes: materialID, title, type, contentURL, difficultyLevel
  + Relationships: Accessed by Student
* **Account**
  + Attributes: username, passwordHash, role, lastLogin
  + Relationships: Associated with Student or Instructor
* **SystemManager**
  + Attributes: systemID, logHistory, backupStatus
  + Responsibilities: Maintains system integrity, manages updates, and handles notifications

**Purpose:**  
This class diagram shows how the DriverPass system organizes information about users, lessons, and tests. The relationships demonstrate how data and actions are connected, helping developers understand what each component must handle.

## Technical Requirements

#### **Hardware Requirements**

* **Server Infrastructure:**
  + Web server for hosting the DriverPass platform.
  + Database server for storing user data, test schedules, and learning materials.
* **Client Devices:**
  + Students: Desktop computers, laptops, or mobile devices with an internet connection.
  + Instructors: Desktop or tablet for managing schedules.
* **Network Equipment:**
  + Secure routers and switches with redundant backup to ensure uptime.

#### **Software Requirements**

* **Operating System:** Windows Server or Linux (Ubuntu) for server environments.
* **Database Management System:** MySQL or PostgreSQL for storing structured data.
* **Web Technologies:**
  + Frontend: HTML5, CSS, JavaScript, React or Angular for user interface.
  + Backend: Java (Spring Boot) or Python (Django/Flask) for application logic.
* **Development Tools:**
  + Lucidchart for UML diagrams.
  + Eclipse, IntelliJ IDEA, or Visual Studio Code for coding.
  + GitHub for version control and collaboration.
* **Browser Compatibility:** Chrome, Firefox, Safari, and Edge.

#### **Security Requirements**

* Encrypted connections using SSL/TLS to secure user data.
* Secure user authentication with salted password hashing.
* Role-based access control to separate permissions between students, instructors, and administrators.
* Regular database backups and monitoring to prevent data loss.
* Compliance with standard data protection regulations (e.g., FERPA for student data).

#### **Infrastructure Requirements**

* Cloud-based deployment using AWS, Azure, or Google Cloud.
* Load balancing to manage simultaneous access from multiple users.
* Automated scaling to accommodate user growth.
* Regular maintenance schedule for updates and patches.
* Logging and monitoring tools for system health and performance.