# CS 255 System Design Document Template

## UML Diagrams

### UML Use Case Diagram

Description: Illustrates the interactions between actors (Student, Secretary, Instructor, Owner, IT Administrator) and the DriverPass system, including key use cases such as Register, Log In, Purchase Package, Schedule Lesson, Take Practice Test, Manage Users/Roles, and Generate Reports.

A diagram of a driver pass

AI-generated content may be incorrect.

### UML Activity Diagrams

Narrative: Student selects a package, checks availability, proposes time, system validates conflicts and eligibility, secretary confirms, system books lesson, and notifications are sent.

A diagram of a flowchart

AI-generated content may be incorrect.

Narrative: Student signs in, reviews packages, adds to cart, checks out through payment processor, system records purchase, updates entitlements, and sends receipt.

A diagram of a purchase package

AI-generated content may be incorrect.

### UML Sequence Diagram

Participants: Student UI, Schedule Service, Availability Service, Payment/Entitlement Service, Notification Service, Data Store.

Flow Summary: Student requests open slots → system queries instructor/vehicle availability → entitlement check ensures remaining lessons → reservation created → confirmation sent via email/SMS.

*A diagram of a sequence

AI-generated content may be incorrect.*

### UML Class Diagram

Classes & Attributes (sample, non‑exhaustive):

* User (userId, name, email, phone, role, status)
* Student extends User (licenseState, progressStatus)
* Instructor extends User (certifications, availabilityCalendarId)
* Secretary extends User
* Owner extends User
* Package (packageId, name, description, hoursIncluded, price, isActive)
* Purchase (purchaseId, studentId, packageId, amount, currency, paymentStatus, timestamp)
* Entitlement (studentId, packageId, lessonsRemaining, expiresOn)
* Vehicle (vehicleId, make, model, year, plate, status)
* Lesson (lessonId, studentId, instructorId, vehicleId, startTime, endTime, location, status)
* AvailabilitySlot (slotId, instructorId, vehicleId, startTime, endTime, isBooked)
* PracticeTest (testId, name, numQuestions, isActive)
* Notification (notificationId, channel, to, templateId, payload, status, timestamp)
* AuditLog (logId, actorId, action, resourceType, resourceId, timestamp, ip)

A diagram of a student

AI-generated content may be incorrect.

## Technical Requirements

## Architecture & Hosting

* Cloud‑hosted web application (e.g., AWS/Azure/GCP) with separate environments for dev, test, and prod.
* Three‑tier architecture: web (SPA), API layer (REST), and relational database.
* Stateless APIs with JWT‑based sessions; caching for read‑heavy endpoints.
* Infrastructure as Code for repeatable deployments.

## Software Stack

* Frontend: Responsive web app (HTML5, CSS, JavaScript framework).
* Backend: RESTful services with validation, logging, and rate limiting.
* Database: Relational DB (e.g., PostgreSQL or MySQL) with schema migrations.
* Messaging/Notifications: Email and SMS via third‑party gateways.
* Integrations: Payment processor (PCI‑compliant), DMV data lookups.

## Tools

* Lucidchart for UML diagrams.
* Git for version control; CI/CD pipeline for builds and deployments.
* Issue tracking and documentation (e.g., Jira/Confluence).
* Automated testing (unit, integration, end‑to‑end).

## Security

* Role‑based access control for Student, Secretary, Instructor, Owner, and IT Admin.
* Strong authentication with password policies and MFA option.
* TLS 1.2+ for data in transit; encryption at rest for sensitive data.
* Input validation, parameterized queries, and WAF to mitigate OWASP Top 10.
* Audit logging for privileged actions and payment events.
* Least‑privilege access to databases and secrets management.

## Performance & Scalability

* Horizontal scaling for stateless API nodes; connection pooling for DB.
* Page response targets under 2 seconds for P95 and API latency targets under 300 ms for P95 under normal load.
* Asynchronous processing for notifications and reporting.

## Reliability & Monitoring

* Automated nightly backups and point‑in‑time recovery for the database.
* Health checks, centralized logging, metrics, and alerting.
* Graceful degradation and retry policies on external integrations.

## Compliance & Privacy

* Payment data handled only by the PCI‑compliant processor; tokens stored, not card numbers.
* Data retention policy and privacy notice aligned with applicable regulations.

## Nonfunctional Requirements Mapping

* Usability: Clear student workflows and accessible design.
* Maintainability: Modular services and comprehensive tests.
* Portability: Cloud‑agnostic patterns where practical.
* Supportability: Runbooks for common operations.

# Appendices

Appendix A: Diagram Insertion Instructions

* Create diagrams in Lucidchart using the UML library.
* Export each diagram as PNG at high resolution.
* Paste the PNG in the labeled placeholder locations above.
* Ensure actor and system names match those referenced in this document.