# CS 255 System Design Document Template

## UML Diagrams

### UML Use Case Diagram

*A diagram of a customer

AI-generated content may be incorrect.*

The UML Use Case Diagram for the DriverPass system illustrates how different users and external systems interact with the platform to achieve specific goals. The diagram identifies four primary actors Customer, Secretary, IT Admin, and DMV System (External) each responsible for distinct functional interactions. Customers have the most extensive set of actions, including registering for an account, scheduling or canceling lessons, taking practice tests, and managing their progress. The Secretary and IT Admin roles reflect internal business functions, such as modifying or scheduling lessons, viewing activity logs, and managing user permissions and system configurations. The DMV System acts as an external entity that pushes rules and question updates to ensure compliance with state driving regulations. Each use case such as Schedule Lesson (2 hours) is clearly connected to related actions through include and extend relationships, which support modular functionality like assigning drivers or resetting passwords. This design ensures that the system aligns with DriverPass’s business goals by streamlining scheduling, testing, and administrative workflows, while also integrating external regulatory updates. The use of proper UML notation, clear actor roles, and logical relationships demonstrates how both user-facing and back-end processes are efficiently coordinated to meet client and user needs.

### UML Activity Diagrams

### *A diagram of a software process AI-generated content may be incorrect.*

The UML Activity Diagrams provide a clear and structured view of two key workflows within the DriverPass system: scheduling a lesson and taking a practice test. Both diagrams accurately depict the flow of control from user actions through system processes, incorporating decision points, actions, and outcomes in a logical sequence. The “Schedule Lesson (2 hours)” diagram illustrates how a customer logs in, selects a lesson time, and proceeds through availability checks, payment processing, and confirmation. By including decision nodes for availability and payment success, the diagram clearly outlines both the main path and alternate flows, such as handling unavailable slots or failed payments. This ensures the system can manage real-world scenarios while maintaining smooth operation. The “Take Practice Tests” diagram follows a similarly structured approach, showing the login process, test availability check, and progression through the test with decision points for errors and pass/fail outcomes. The use of initial and final nodes, swim lanes to separate responsibilities, and clear decision branches demonstrate strong adherence to UML standards. Overall, these diagrams effectively communicate system behavior, support transparency between technical and nontechnical stakeholders, and ensure the workflows align with DriverPass’s business goals of improving scheduling efficiency and student learning outcomes.

### UML Sequence Diagram

*A diagram of a service

AI-generated content may be incorrect.*

The UML Sequence Diagram for the “Schedule Lesson (2 hours)” use case effectively illustrates the step-by-step interaction between the student, various system components, and external services to complete the scheduling process. The diagram clearly identifies each participant such as the Web/App UI, Auth Service, Scheduler Service, Calendar, Notification Service, and Database and maps how messages flow sequentially between them. It begins with the student selecting a lesson time, followed by session validation and an availability check, ensuring secure and accurate booking. The use of alt frames to represent alternative scenarios, such as time unavailability and payment failure, demonstrates proper UML structure and captures real-world variability. By modeling conditional flows for availability and payment processing, the diagram provides a complete picture of both the main success path and exception handling. Additionally, incorporating notification and database interactions emphasizes how the system confirms appointments and maintains reliable records. Overall, this diagram not only adheres to UML standards using lifelines, activation bars, and clear message labels but also supports DriverPass’s business goals of providing a smooth, secure, and user-friendly scheduling experience.

### UML Class Diagram

A diagram of a user

AI-generated content may be incorrect.

The UML Class Diagram for the DriverPass system provides a structured representation of the core entities and their relationships, supporting the platform’s key scheduling, testing, and administrative functions. The diagram follows standard UML notation with classes, attributes, and associations that reflect the system’s functional requirements. At the top level, the User class is generalized into specialized roles Student, Secretary, Owner, and ITAdmin to ensure role-based access control and tailored functionality. The LessonReservation class serves as the central hub, linking students to instructors, vehicles, and payments through clearly defined associations. This enables efficient scheduling and resource allocation. The inclusion of PracticeTest and TestResult classes supports the learning and progress tracking features of DriverPass, while DMVUpdate and Notification ensure regulatory compliance and communication with users. Each association is represented with appropriate multiplicities, showing how entities interact in both one-to-one and one-to-many relationships. Overall, this class diagram provides a logical and organized structure for system development, enabling clear communication between technical and nontechnical stakeholders and ensuring the design meets client requirements for scheduling lessons, test preparation, administrative control, and integration with DMV updates.

## Technical Requirements

The technical requirements for the DriverPass system are designed to ensure a secure, scalable, and reliable environment that supports scheduling, testing, payments, and DMV integrations. The hardware infrastructure will rely on cloud-based or on-premises servers with adequate processing power, at least 8 GB of RAM, and SSD storage to handle concurrent user sessions and real-time updates. End users, including students, instructors, and administrators, can access the platform through desktops, laptops, or mobile devices with an internet connection and a modern browser.

The software stack will consist of a modern, responsive front end (e.g., React or Angular) and a secure, high-performance backend (e.g., Python, Java, or Node.js). A relational database such as PostgreSQL or MySQL will be used to store structured data related to accounts, reservations, test results, notifications, and DMV updates, supporting the relationships defined in the UML Class Diagram. Integrated services will include secure payment gateways and DMV APIs, as well as messaging services for real-time notifications via email or SMS.

From an infrastructure and security perspective, the system will use HTTPS encryption, secure authentication, and role-based access control to protect sensitive user data. Automated backups, disaster recovery planning, and performance monitoring will ensure reliability and business continuity. Development and deployment will leverage version control tools such as GitHub, CI/CD pipelines for rapid updates, and containerization with Docker to ensure consistent builds. This technical foundation directly supports the behaviors modeled in the UML diagrams such as lesson scheduling, payment flows, and rule updates and ensures DriverPass remains secure, efficient, and adaptable as the system scales.