CS 255 System Design Document – DriverPass System Design

Name: Christopher Phillips

Date: February 23, 2025

**I. Introduction**

The DriverPass System dais built to be your all-in-one solution for driver training. It brings together online learning modules, practice tests, and the ability to book on-the-road training sessions while also keeping you updated with the latest DMV information in real-time. Designed for a wide-range of users from students and secretaries to owners, IT staff, and driving instructors it makes sure everyone gets secure access and that all important data is tracked efficiently

**II. UML Diagrams**

A. Use Case Diagram

This diagram captures the interactions between all key actors (Customer, Secretary, Owner, IT Officer, Driver/Instructor, and DMV System) and the system’s core functionalities, such as creating accounts, booking reservations, selecting training packages, processing payments, and integrating DMV updates. It also stresses that customers must be verified through login before accessing reservation or course content.

B. Activity Diagrams

The Make/Modify/Cancel Reservation diagram walks you through each step of the reservation process from choosing a package and assigning a driver to handling payments. It also shows what happens if the payment fails along the way. The Reset Password diagram covers how the system checks your reset token and gives you an error message if anything goes wrong.

C. Sequence Diagram

This diagram maps out the back-and-forth communication during a reservation. It shows both successful payment scenarios and what happens if a payment doesn’t go through, making sure every possible outcome is addressed.

D. Class Diagram

This diagram showcases the five core classes User, Reservation, Package, Payment, and DMVUpdate and how they connect. For example, a single Reservation points to exactly one Package, but that same Package can be linked to several Reservations.

**III. Technical Requirements**

Below are the technical requirements we’ve chosen to ensure DriverPass meets all client expectations. The client wants to prioritize security, have for future growth, and have an experience that’s simple for everyone to use.

A. Hardware Requirements

Cloud-Based Server:

The system will be hosted on a cloud-based server (e.g., AWS, Azure) with sufficient CPU and memory to handle simultaneous user logins, reservation requests, and database queries. End users (customers, secretaries, owners) only need a standard PC or mobile device with internet access and a modern web browser to access DriverPass.

Specifications: A minimum of 4 virtual CPUs, 16 GB of RAM, and high-performance storage will be provisioned to handle simultaneous user logins, reservation requests, and database queries.

Mobile and Desktop Support:

End users (customers, secretaries, owners) will access DriverPass using standard PCs or mobile devices with modern web browsers. Mobile devices must have sufficient processing power (e.g., a modern smartphone with at least 2GB RAM) to run the application smoothly.

B. Software and Tools

* Operating System: A Linux-based server OS (e.g., Ubuntu) or a Windows Server environment.
* Web Server: Nginx or Apache to serve the web application.
* Database: A relational database such as MySQL or PostgreSQL for storing user data, reservations, packages, and payment records.
* Programming Language/Framework: Python or Java-based solution with a framework (e.g., Django, Spring) for rapid web development and robust security features.
* CASE Tool: Lucidchart or any UML-compliant tool for maintaining design diagrams.
* Payment Integration: A secure, PCI-compliant payment gateway (e.g., Stripe or PayPal).
* DMV Integration: Web service or REST API calls to retrieve updated driving rules or test questions.

Programming Language/Framework:

The application will be developed using either Python with the Django framework or Java with the Spring framework. Both frameworks support rapid development, robust security, and scalability.

* CASE Tool: Lucidchart will be used to create and maintain UML diagrams because of its collaborative features and ease of use.
* Payment Integration: Integration with a PCI-compliant payment gateway such as Stripe or PayPal is essential to securely process credit card transactions.
* DMV Integration: The system will utilize DMV data by using web services or REST APIs, so any new rules, policies, or test questions become instantly available to our users. This way, everything stays up to date and accurate.

C. Infrastructure and Security

* Secure Data Transmission: All communications between the client and server will be secured using HTTPS/SSL. This ensures that sensitive data, such as credit card details and personal information, is encrypted during transit.
* Role-Based Access Control (RBAC): The system will implement RBAC to enforce different permission levels across user types (e.g., Customer, Secretary, Driver, IT Officer, and Owner). Each role will have clearly defined access rights, ensuring that only authorized actions are permitted.
* Audit Logging: Detailed logs will record all changes to reservations, account modifications, and system configurations. These logs will be stored securely and will be used to generate administrative and activity reports.
* Password Security: Passwords will be hashed using industry-standard algorithms and stored in an encrypted format in the database to protect user credentials.

D. Scalability and Availability

* Horizontal Scaling: The system should support additional users without a significant drop in performance. Automatic or manual scaling may be used during peak traffic (e.g., near DMV test dates or holidays).
* High Availability: The design incorporates load balancing and failover mechanisms to ensure minimal downtime. This includes automatic scaling and redundancy measures to maintain service during hardware failures or traffic spikes.
* Data Synchronization: Although the system is web-based, provisions for offline read-only access (with data synchronization upon reconnection) may be implemented to mitigate issues when stable internet connectivity is not available.

E. Functional vs. Non-Functional Requirements

Functional Requirements

These requirements cover the system’s main tasks. In DriverPass, they include:

* Creating user accounts, logging in, and resetting passwords
* Booking new reservations, as well as changing or canceling them
* Selecting a package and assigning both a car and a driver
* Handling payments and generating detailed activity reports for admins
* Staying in sync with DMV updates so users always get the latest information

Non-Functional Requirements:

* These specify the quality attributes of the system. For DriverPass, non-functional requirements include:
* High performance and responsiveness (e.g., system response time under three seconds).
* Scalability to support a growing user base without performance degradation.
* Robust security measures (e.g., encryption, RBAC, audit logging).
* User-friendly design that is accessible from both desktop and mobile devices.