# CS 255 System Design Document Template

Stephen C. Pittman

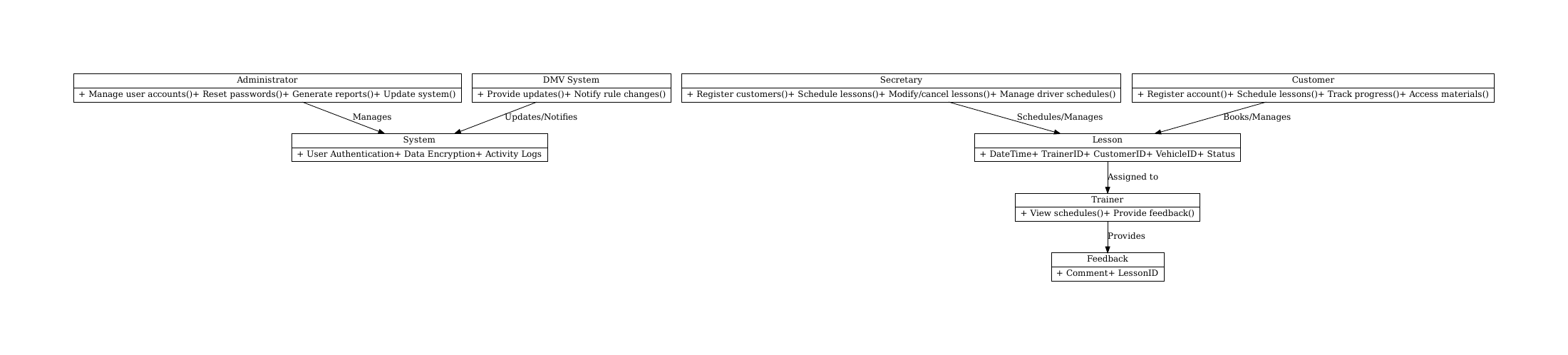
SNHU

CS 255

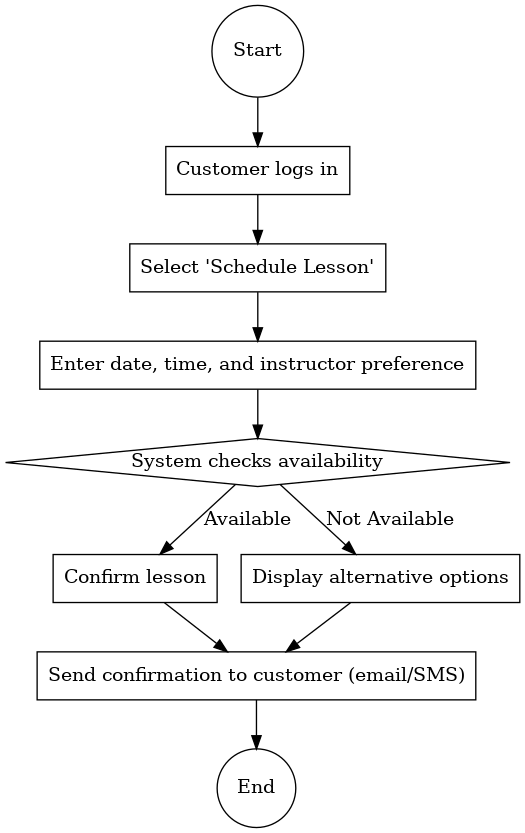
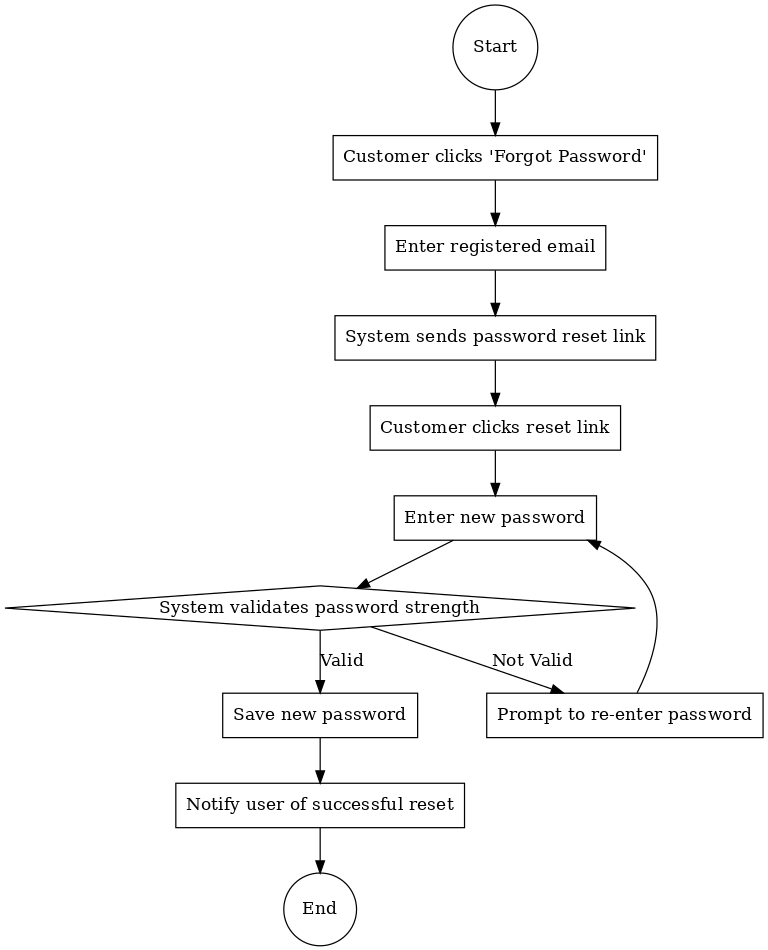
12/15/24

## UML Diagrams

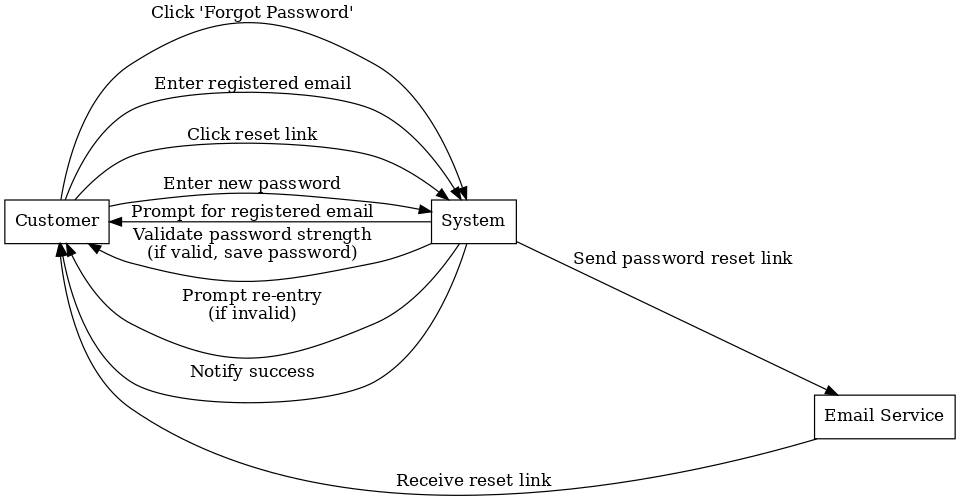
### UML Use Case Diagram



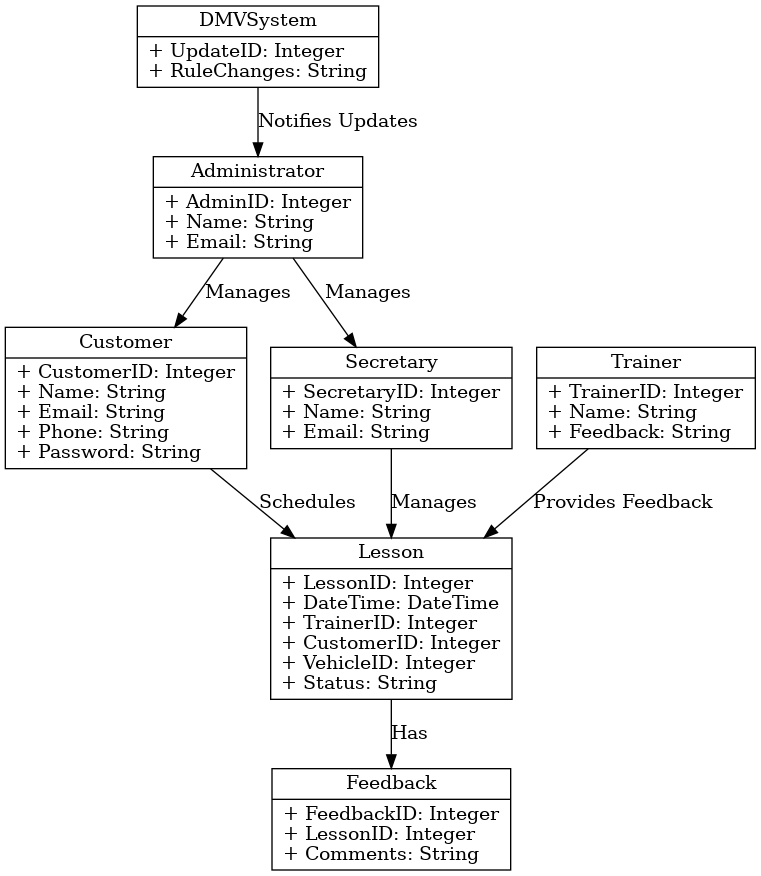
### UML Activity Diagrams



### UML Sequence Diagram



### UML Class Diagram



## Technical Requirements

Based on the functional and nonfunctional requirements outlined in the business requirements document, the proposed system's technical requirements include the necessary hardware, software, tools, and infrastructure to ensure efficient and reliable operation. The system will utilize cloud servers hosted on platforms such as AWS, Azure, or Google Cloud Platform to guarantee scalability and reliability. These servers should have a minimum of a quad-core CPU, 16 GB RAM, and 1 TB of expandable storage, with an uptime of 99.9% to meet performance and availability standards. On the client side, the system will support desktops, laptops, smartphones, and tablets, all requiring compatibility with modern web browsers and stable internet connections. High-speed internet and secure network infrastructure, including firewalls, are essential for protecting against unauthorized access.

The software stack for the system will include a Linux-based operating system, such as Ubuntu or CentOS, for server environments. The database will use a relational model like MySQL or PostgreSQL to efficiently manage user accounts, lesson scheduling, and activity tracking. The application will be hosted on web servers such as Nginx or Apache HTTP Server, with backend development utilizing Python frameworks like Django or Flask or Java frameworks like Spring Boot for secure API development. The frontend will leverage JavaScript frameworks such as React.js or Angular to deliver a dynamic and responsive user interface. To ensure secure communication and data protection, the system will use TLS for encrypted HTTPS connections and password hashing techniques like bcrypt or Argon2.

To support development and maintenance, the system will rely on tools such as Visual Studio Code, IntelliJ IDEA, or Eclipse for coding and debugging, along with Git for version control. Testing will be conducted using tools like Selenium for user interface testing, Postman for API validation, and OWASP ZAP for vulnerability scanning. Additionally, monitoring tools such as Datadog or New Relic will track system performance and identify potential issues in real time.

The infrastructure will be fully cloud-based, leveraging AWS EC2 or Azure App Services for hosting, along with AWS S3 or Azure Blob Storage for secure file storage and daily automated backups. Integration with external systems, such as DMV databases, will be achieved through APIs, enabling rule updates and notifications. These technical requirements ensure the system is scalable, secure, and aligned with DriverPass’s objectives while meeting performance and adaptability needs.