# CS 255 System Design Document

Alexander Falatine

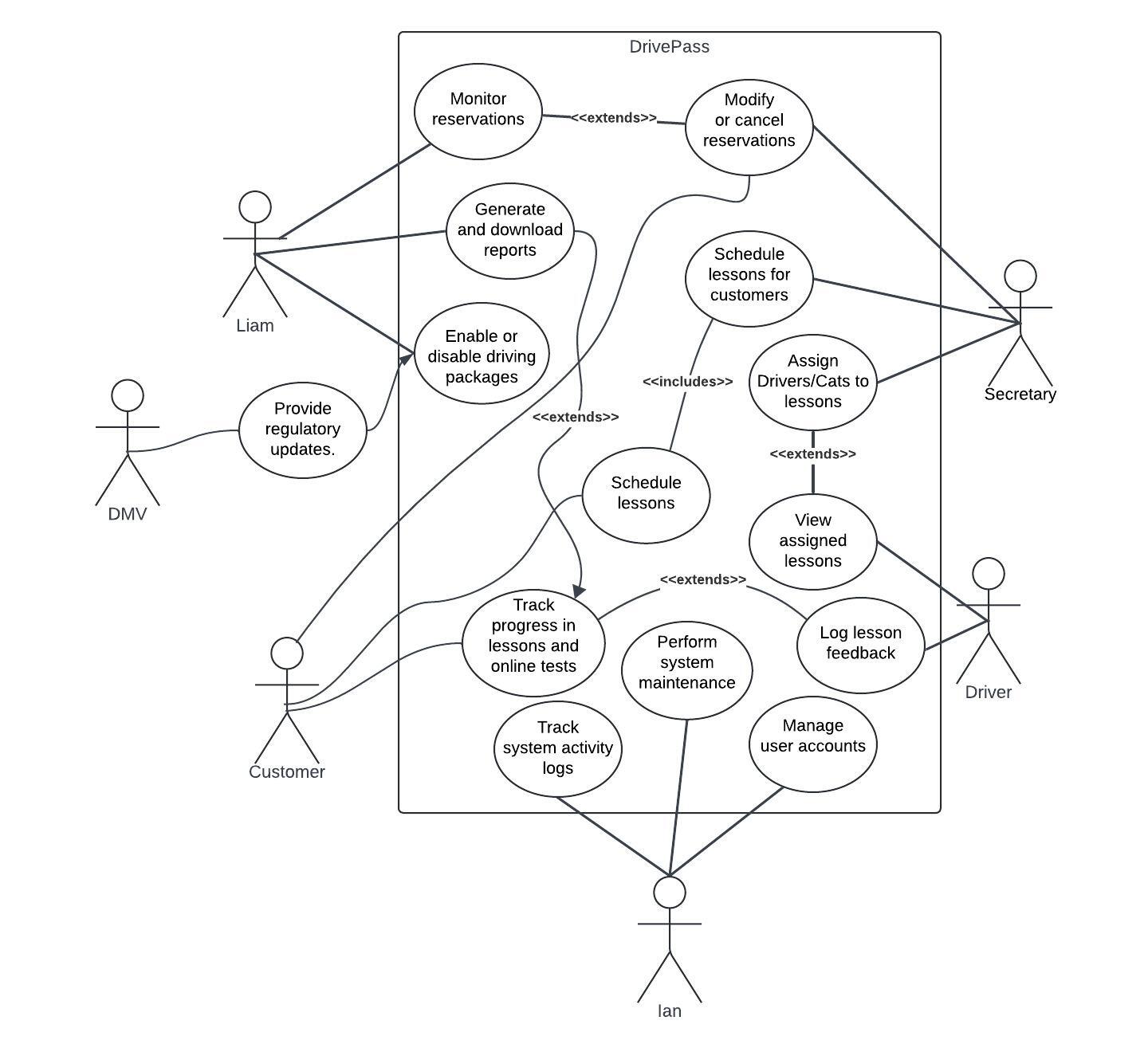
255 – System Analysis and Design

Dr. Goran Trajkovski

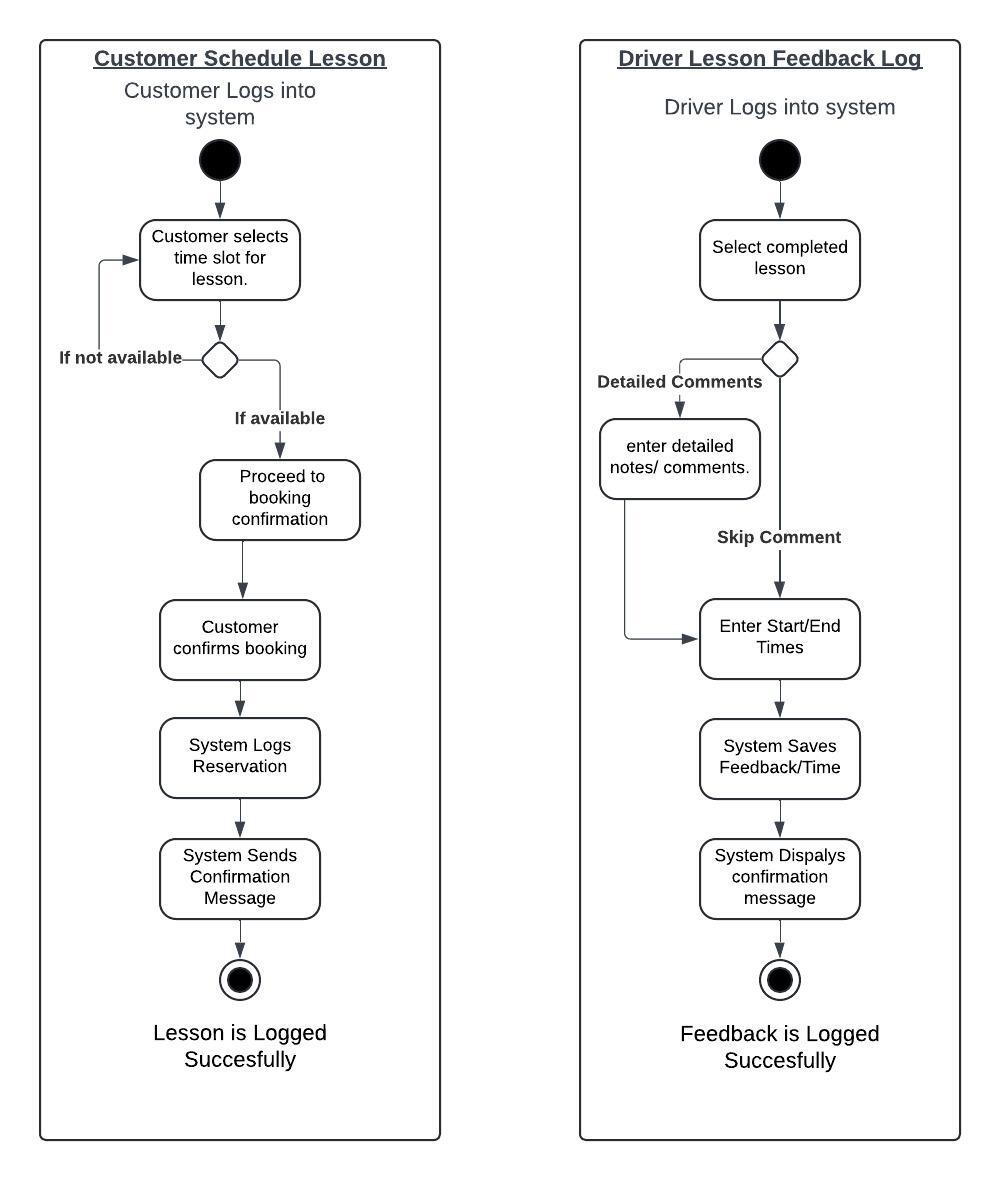
10/20/2024

## UML Diagrams

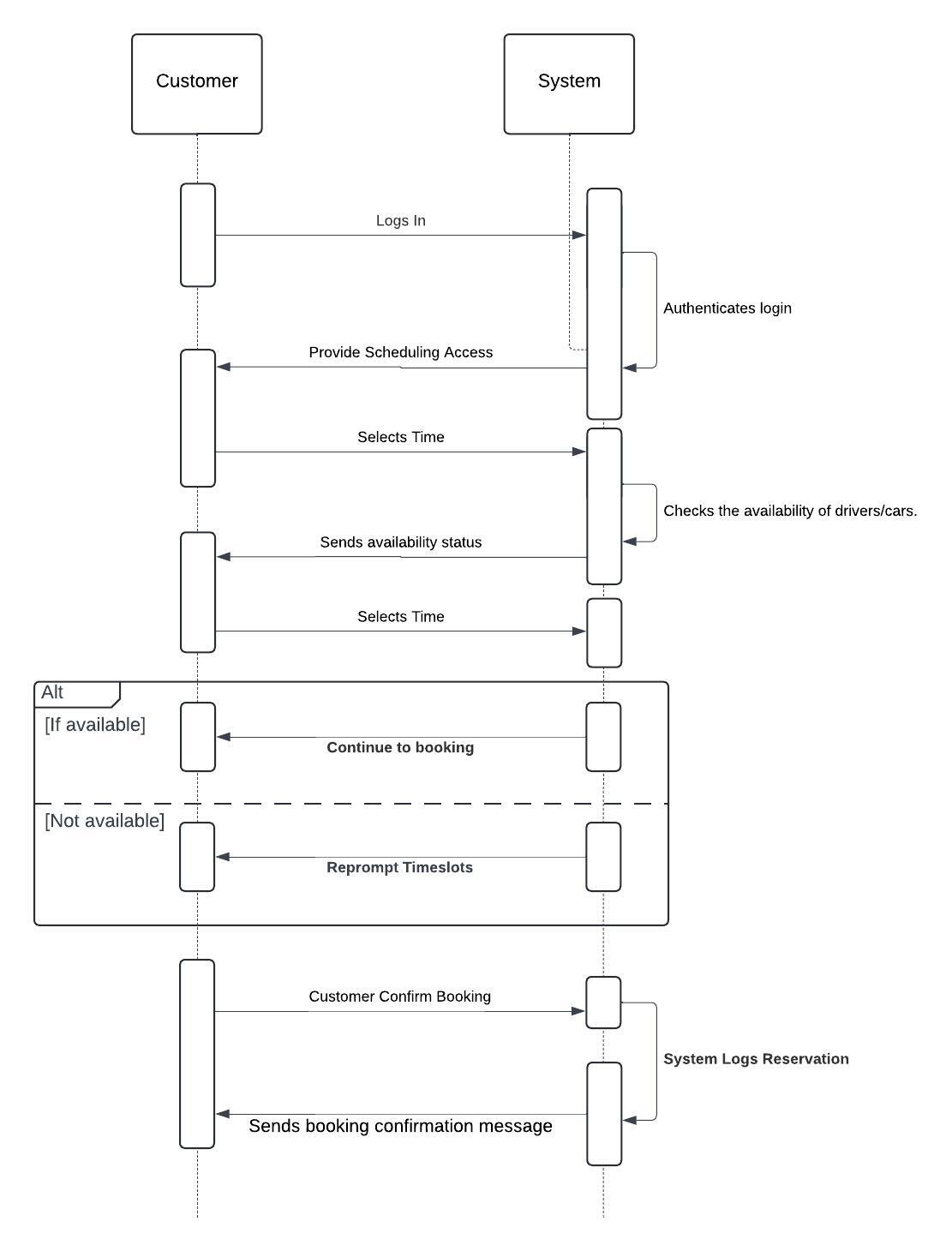
### UML Use Case Diagram



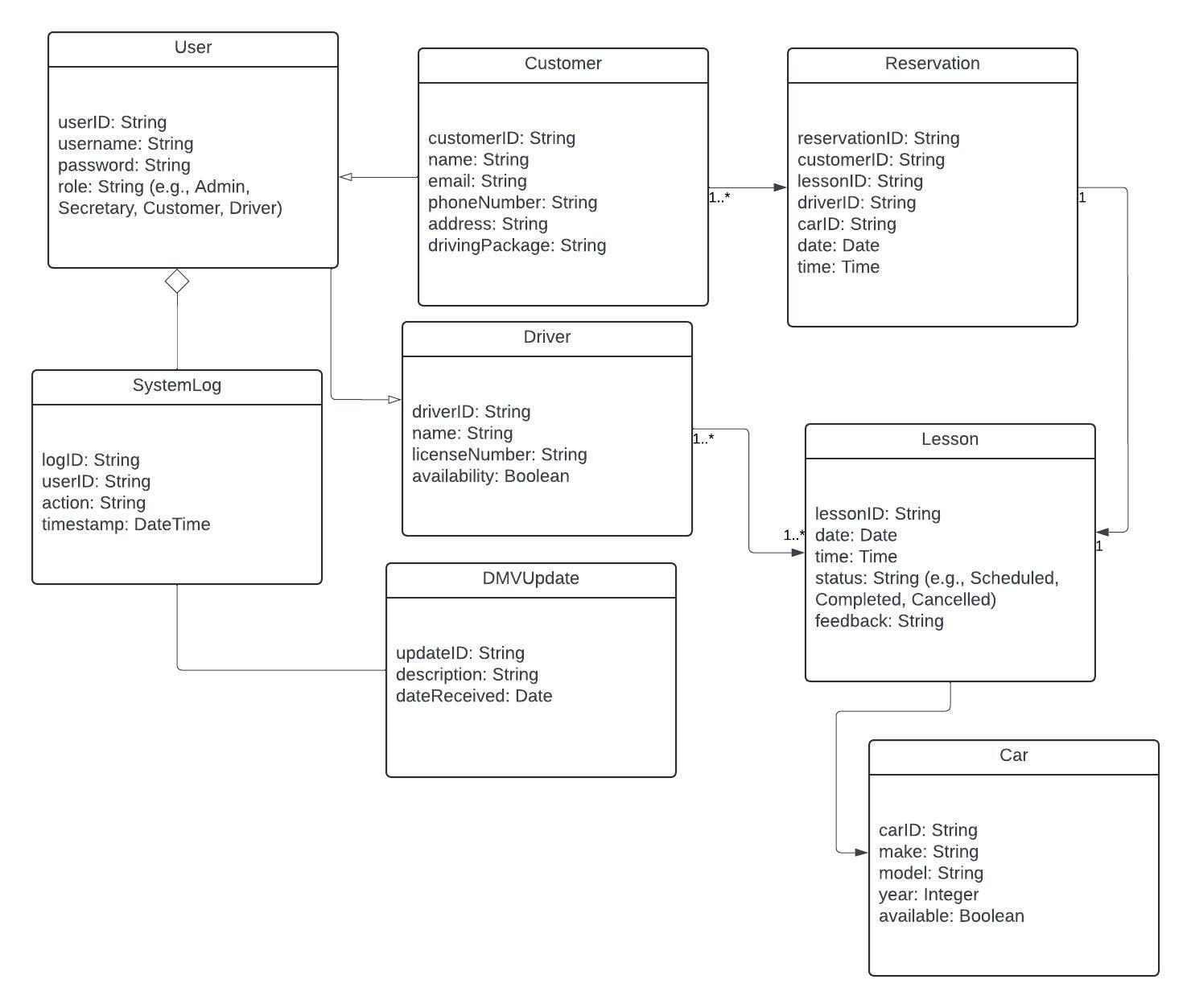
### UML Activity Diagrams



### UML Sequence Diagram



### UML Class Diagram



## Technical Requirements

**Hardware/Infrastructure Requirements:**

1. Server Devices:
   1. Cloud based server infrastructure (like Google Cloud, AWS, or Azure as examples) to host the backend system.
   2. Redundant servers to ensure high availability and load balancing which helps performance.
2. Client Devices:
   1. Desktops, laptops, tablets, and smartphones to access the system.
   2. Devices that support modern browsers (Chrome, Firefox, Safari, or Edge for instance).
   3. Drivers may need tablets or smartphones to log lesson feedback on the go for convenience.
3. Network Devices:
   1. High speed internet connection with secure routers and switches to support remote access for drivers.
   2. Firewalls and VPN access for IT personnel to manage the system securely.
4. Cloud Infrastructure:
   1. Google Cloud, AWS, or Microsoft Azure as mentioned prior to ensure scalability and fault tolerances.
   2. Cloud services for hosting applications and databases.
5. Security Infrastructure:
   1. SSL/TLS (Secure Sockets Layer/ Transport Layer Security) encryption to secure communication between the client and server.
   2. Role-Based Access Control (RBAC) to ensure secure user access.
6. Backup and Disaster Recovery:
   1. Automated cloud backups for all critical data, including user accounts, reservations, and lesson logs.
   2. Disaster recovery plan to restore system availability during outages.
7. Monitoring and Logging:
   1. Implement logging tools (e.g., ELK Stack) to monitor system activity and generate logs.
   2. System monitoring through tools like Prometheus and Grafana for real-time health checks.

**Software Requirements:**

1. Operating System:
   1. Server-side OS: Linux or Windows Server for backend operations.
   2. Client-side: Compatible with Windows, macOS, Linux, Android, and iOS.
2. Database Management System (DBMS):
   1. Relational database like MySQL, PostgreSQL, or SQL Server to store customer, lesson, and reservation data.
3. Web Server:
   1. Apache, Nginx, or IIS to handle HTTP requests and serve the frontend.
4. Frontend Tools:
   1. HTML, CSS, JavaScript for a responsive web interface.
   2. React or Angular for enhanced user experience and modular development.
5. Backend Tools:
   1. Node.js, Python (Django/Flask), or Java (Spring Boot) to handle API requests and implement business logic.
6. Third-Party Integration:
   1. DMV API integration to receive regulatory updates.
   2. Payment gateway integration (e.g., PayPal or Stripe) for handling customer payments.

**Specific Tools and Development Environment:**

1. Integrated Development Environment (IDE):
   1. Visual Studio Code, IntelliJ IDEA, or Eclipse for backend development.
   2. Frontend developers can use VS Code.
2. Version Control:
   1. Git for version control, hosted on platforms like GitHub or GitLab.
3. Testing Tools:
   1. Selenium or Postman for automated testing of the web interface and APIs.
   2. Unit testing tools such as JUnit (for Java) or PyTest (for Python) depending on the chosen code.
4. Project Management:
   1. Tools like Jira, Trello, or Asana to manage development tasks and timelines and be helpful in maintaining a structured approach to future projects.

**Performance and Scalability Requirements:**

1. Concurrency:
   1. The system must support at least 100 concurrent users without performance degradation.
2. System Updates:
   1. Scheduled maintenance windows during off-peak hours for system updates and upgrades.
3. Performance Metrics:
   1. Fast response time for data retrieval operations (< 2 seconds).
   2. Optimized database queries to handle large amounts of data efficiently.