# CS 255 System Design Document Template

Alex Simonian

02/20/2025

## UML Diagrams

### UML Use Case Diagram

A diagram of a driver pass system



### UML Activity Diagrams

Payment processing

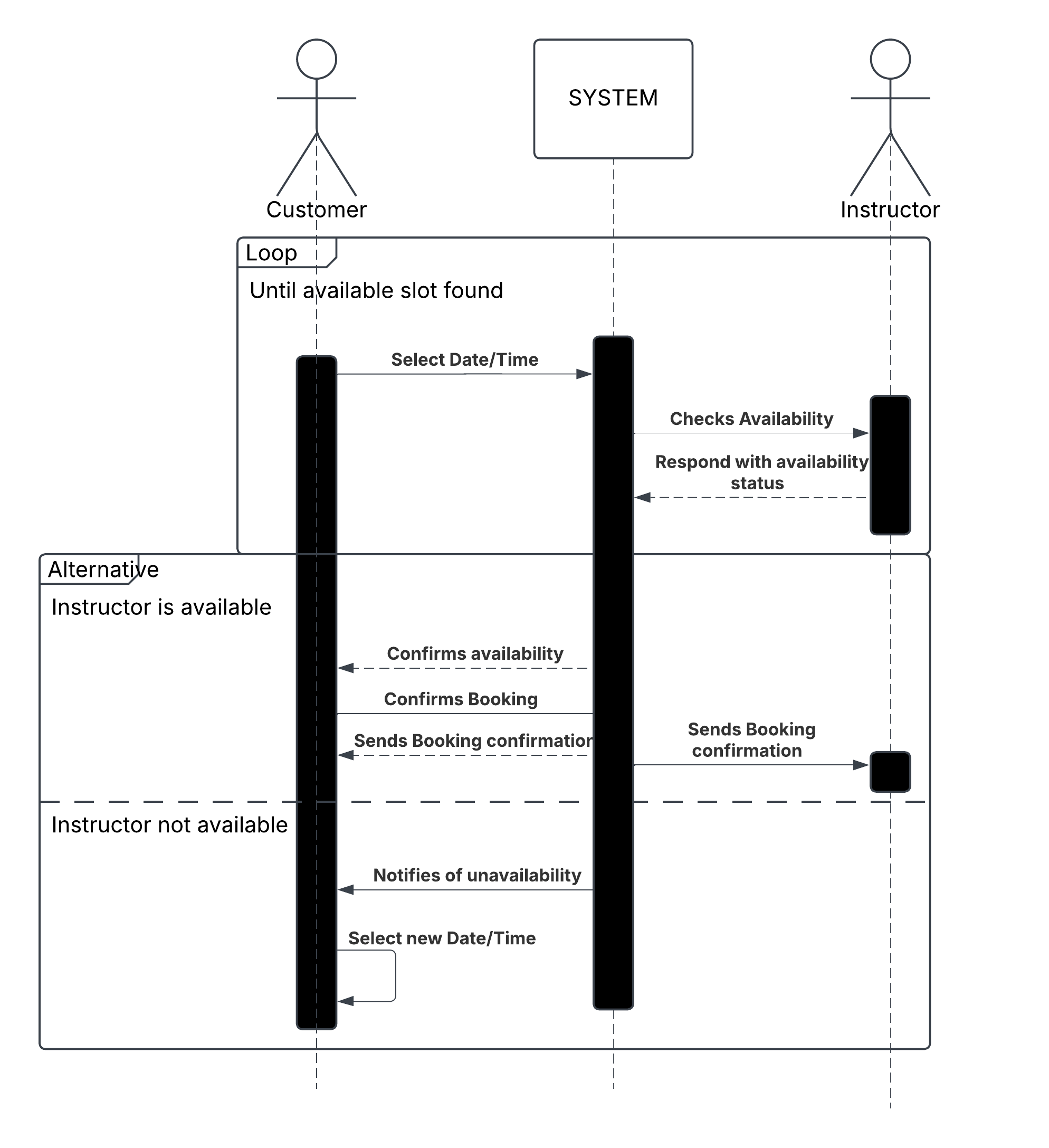
A diagram of a system

AI-generated content may be incorrect.

Customer booking driving lesson

### 

### UML Sequence Diagram – Customer booking lesson



### UML Class Diagram

A diagram of a software company

AI-generated content may be incorrect.

## Technical Requirements

**Hardware Requirements**

* **Servers**:
  + Cloud-based servers (e.g., AWS, Azure, or Google Cloud) to host the web application and database.
  + Scalable server infrastructure to handle peak traffic (e.g., during lesson bookings or practice test submissions).
* **Storage**:
  + Sufficient cloud storage for user data, training materials, and transaction logs.
  + Backup storage for disaster recovery and data redundancy.
* **Client Devices**:
  + The system must support access from desktops, laptops, tablets, and mobile devices.
  + Minimum hardware requirements for client devices (e.g., modern processors, sufficient RAM, and updated browsers).

**2. Software Requirements**

* **Operating System**:
  + The system should be platform-agnostic and support modern operating systems (Windows, macOS, Linux, iOS, Android).
* **Web Browsers**:
  + The system must be compatible with modern web browsers (Chrome, Firefox, Edge, Safari).
* **Database**:
  + A relational database management system (RDBMS) such as MySQL, PostgreSQL, or SQL Server for storing user data, lesson schedules, payments, and reports.
  + Support for NoSQL databases (e.g., MongoDB) for unstructured data like training materials or logs.
* **Development Frameworks**:
  + Front-end: HTML5, CSS3, JavaScript (React.js or Angular.js for dynamic user interfaces).
  + Back-end: Node.js, Python (Django/Flask), or Java (Spring Boot) for server-side logic.
  + API integration for third-party services (e.g., payment gateways, SMS/email notifications).

**3. Tools and Technologies**

* **Version Control**:
  + Git for version control and collaboration (e.g., GitHub or GitLab).
* **CI/CD Pipeline**:
  + Continuous Integration/Continuous Deployment (CI/CD) tools like Jenkins or GitHub Actions for automated testing and deployment.
* **Security Tools**:
  + SSL/TLS certificates for secure data transmission.
  + Encryption tools for sensitive data (e.g., AES-256 for database encryption).
  + Multi-factor authentication (MFA) tools for admin and IT roles.
* **Monitoring and Logging**:
  + Tools like Splunk or ELK Stack (Elasticsearch, Logstash, Kibana) for system monitoring and log analysis.
  + Real-time error tracking with tools like Sentry or New Relic.

**4. Infrastructure Requirements**

* **Cloud Hosting**:
  + The system must be hosted on a cloud platform (e.g., AWS, Azure, or Google Cloud) for scalability and reliability.
  + Use of load balancers to distribute traffic and ensure high availability.
* **Network Requirements**:
  + High-speed internet connectivity for real-time updates and seamless user experience.
  + Secure VPN access for administrators and IT staff managing the system.
* **Backup and Recovery**:
  + Automated daily backups of the database and critical system files.
  + Disaster recovery plan to restore the system in case of failure.

**5. Performance Requirements**

* **Response Time**:
  + The system should respond to user requests within **2 seconds** under normal load.
* **Uptime**:
  + The system must maintain **99.9% uptime** to ensure availability for users.
* **Scalability**:
  + The system should scale horizontally to handle up to **10,000 concurrent users** during peak times.

**6. Security Requirements**

* **Authentication**:
  + Role-based access control (RBAC) to ensure users only access features relevant to their role (e.g., customer, instructor, administrator, IT admin).
  + Multi-factor authentication (MFA) for admin and IT roles.
* **Data Protection**:
  + Encryption of sensitive data (e.g., passwords, payment information) both in transit and at rest.
  + Regular security audits and vulnerability assessments.
* **Compliance**:
  + The system must comply with data protection regulations (e.g., GDPR, CCPA) and industry standards (e.g., PCI DSS for payment processing).

**7. Integration Requirements**

* **Third-Party Services**:
  + Integration with payment gateways (e.g., Stripe, PayPal) for processing payments.
  + Integration with SMS/email notification services (e.g., Twilio, SendGrid) for booking confirmations and reminders.
* **DMV Integration**:
  + API integration with DMV systems to receive updates on policies and test changes.

**8. Maintenance Requirements**

* **Updates**:
  + Regular system updates to fix bugs, improve performance, and add new features.
  + Updates should occur during non-peak hours to minimize downtime.
* **Monitoring**:
  + Real-time monitoring of system performance, user activity, and error logs.
  + Automated alerts for critical issues (e.g., server downtime, failed payments).

The **DriverPass System** requires a robust, scalable, and secure infrastructure to support its functionalities. Key technical requirements include:

* **Cloud-based hosting** for scalability and reliability.
* **Modern web technologies** for front-end and back-end development.
* **Secure data storage and transmission** to protect sensitive user information.
* **Integration with third-party services** for payments, notifications, and DMV updates.
* **High performance and uptime** to ensure a seamless user experience.