# CS 255 System Design Document Template

This template lays out all the different sections that you need to complete for Project Two. Each section has guidance to prompt your thinking. You will need to continually reference the interview transcript as you work to make sure that you are addressing your client’s needs. There is no required length for the final document. Instead the goal is to complete each section based on what your client’s needs are. Remove this note when you are finished, and replace all bracketed text with the relevant information.

## UML Diagrams

### UML Use Case Diagram

A diagram of a driver pass

Description automatically generated

### UML Activity Diagrams

Access Online Classes:

A black background with white text

Description automatically generated

Make Reservations:

A diagram of a software application

Description automatically generated with medium confidence

### UML Sequence Diagram

A diagram of a customer

Description automatically generated

### UML Class Diagram

A screenshot of a computer

Description automatically generated

## Technical Requirements

**Technical Requirements:**

Hardware:

1. **Server**: A robust server infrastructure to host the web-based system. This should be scalable to handle increasing user loads.
   * CPU: Multi-core processors for efficient multitasking.
   * RAM: Minimum of 16GB, scalable as needed.
   * Storage: SSDs for faster data retrieval, with a minimum of 1TB storage to start.
2. **Backup Systems**: Redundant servers and storage solutions to ensure data integrity and availability.
3. **Networking Equipment**: High-speed routers and switches to ensure fast and reliable data transfer.

Software:

1. **Operating System**: A stable server OS like Linux (Ubuntu Server or CentOS) for hosting the application.
2. **Database Management System (DBMS)**: PostgreSQL, as mentioned, to manage and store all the data efficiently.
3. **Web Server**: Apache or Nginx to serve the web application.
4. **Backend Framework**: A framework like Django or Flask for Python, or Express for Node.js, to handle server-side operations.
5. **Frontend Framework**: React or Angular for building a dynamic and responsive user interface.
6. **SSL Certificate**: To ensure encrypted communication between the client and server.
7. **Development Tools**: Integrated Development Environments (IDEs) like Visual Studio Code or PyCharm, version control systems like Git, and collaboration tools like GitHub or Bitbucket.

Infrastructure:

1. **Cloud Hosting**: Consider using cloud platforms like AWS, Google Cloud, or Azure to host the application. This provides scalability, reliability, and a suite of tools to monitor and manage the application.
2. **Content Delivery Network (CDN)**: To ensure faster content delivery to users worldwide.
3. **Backup and Recovery**: Regular backups should be scheduled, and a recovery plan should be in place.

Security:

1. **Firewalls**: To protect the system from unauthorized access and potential threats.
2. **Intrusion Detection Systems (IDS)**: To monitor and detect any malicious activities.
3. **Data Encryption**: Ensure data at rest and in transit is encrypted using industry-standard protocols.
4. **Regular Security Audits**: To identify and rectify potential vulnerabilities.
5. **User Authentication and Authorization**: Implement strong password policies, two-factor authentication, and role-based access controls.

Miscellaneous:

1. **API Integration**: For integrating with DMV for real-time updates and compliance checks.
2. **Payment Gateway**: Integration with a secure payment gateway for processing transactions.
3. **Monitoring Tools**: Tools like Grafana or Prometheus to monitor server health, performance, and user activity.
4. **Logging**: Implement comprehensive logging mechanisms to track system activities, errors, and user interactions.