# CS 255 System Design Document Template

## UML Diagrams

### UML Use Case Diagram

A diagram of a driver pass

Description automatically generated

### UML Activity Diagrams

A diagram of a software

Description automatically generated

A diagram of a flowchart

Description automatically generated

### UML Sequence Diagram

A diagram of a process

Description automatically generated

### UML Class Diagram

A diagram of a program

Description automatically generated

## Technical Requirements

**Hardware Requirements**

The hardware requirements depend on both the server-side infrastructure and the client-side devices used by students, instructors, and administrators.

**Server-Side Requirements:**

* **Cloud-Based Servers**:
  + The system should be hosted on a reliable cloud infrastructure like **Amazon Web Services (AWS)**, **Google Cloud Platform (GCP)**, or **Microsoft Azure**. These platforms offer scalable computing resources that can be adjusted as user demand increases.
  + **Virtual Machines (VMs)** for running the application backend and database.
  + **Storage**: The system requires cloud storage for lesson data, test results, and user information. Services like **AWS S3** or **Google Cloud Storage** can be used for this purpose.

**Client-Side Requirements:**

* **User Devices**:
  + Students, instructors, and administrators will access the system via web browsers. The system should be compatible with **desktops, laptops, tablets, and mobile phones** running modern operating systems like **Windows**, **macOS**, **Linux**, **iOS**, and **Android**.
  + Devices should support modern web browsers like **Google Chrome**, **Firefox**, **Safari**, and **Microsoft Edge** to ensure compatibility.

**Software Requirements**

**Backend:**

* **Programming Languages**: The backend should be developed using **Java**, **Python (Django/Flask)**, or **Node.js** to handle business logic and database interactions efficiently.
* **Web Framework**: A reliable web framework like **Django** (Python), **Spring Boot** (Java), or **Express.js** (Node.js) will be required to create REST APIs for interacting with the frontend.
* **Database Management**:
  + A relational database like **MySQL**, **PostgreSQL**, or **SQL Server** is necessary for storing structured data, including user accounts, lessons, exam results, and schedules.
  + **NoSQL databases** such as **MongoDB** can also be considered for storing less structured data, like feedback or lesson logs.
* **API Integration**:
  + The system will require APIs to integrate with external services like **payment gateways**, DMV databases for regulatory compliance updates, and email services for sending notifications.

**Frontend:**

* **Frontend Framework**: The user interface will be built using **React.js**, **Angular**, or **Vue.js** to create a responsive and dynamic user experience.
* **HTML5, CSS3, and JavaScript**: Standard web technologies will be used to create interactive and user-friendly web pages that are compatible with various screen sizes.
* **Mobile-Responsive Design**: The frontend must be mobile-responsive, ensuring that users can access the system seamlessly from both mobile devices and desktop browsers.

**Other Software Requirements:**

* **Security Tools**:
  + **SSL/TLS Encryption**: The system must use SSL certificates to secure all communication between the client and the server.
  + **OAuth or JWT (JSON Web Tokens)**: To ensure secure user authentication and session management.
  + **Data Encryption**: Sensitive information (e.g., passwords, personal data) should be encrypted at rest using tools like **bcrypt** for hashing passwords.
  + **Access Control**: Implement role-based access control (RBAC) to ensure different users (students, instructors, admins) have appropriate access levels.

Tools and Technologies

Development Tools:

Integrated Development Environments (IDEs) like Visual Studio Code, PyCharm, or IntelliJ IDEA will be used for code development.

Version Control: Git and GitHub/GitLab will be used for version control and collaboration among developers.

Continuous Integration/Continuous Deployment (CI/CD): Tools like Jenkins, Travis CI, or CircleCI will be required for automating testing and deployment.

Database Tools:

Database Management System (DBMS): Tools like MySQL Workbench or pgAdmin for managing and interacting with the database.

API Tools:

Postman or Insomnia for testing REST APIs and integration endpoints.

Testing Frameworks:

Unit testing and integration testing will be performed using JUnit (for Java), PyTest (for Python), or Jest (for JavaScript).

**Infrastructure Requirements**

**Cloud Infrastructure:**

* **Cloud Hosting**: The system will be deployed on cloud platforms like **AWS**, **Google Cloud**, or **Azure** for scalability, performance, and availability.
* **Server Configuration**:
  + **Load Balancers** to distribute traffic across multiple servers to prevent overload and improve performance.
  + **Auto-scaling** capabilities to handle high demand and adjust resources based on system load.

**Database Infrastructure:**

* **High Availability**: The database should be set up in a high-availability (HA) architecture with failover capabilities to ensure system reliability.
* **Backups**: Automated database backups should be scheduled to ensure data recovery in case of failure.

**Security Infrastructure:**

* **Firewalls**: To protect against unauthorized access, firewall rules should be configured to restrict access to only authorized IP addresses.
* **Monitoring and Logging**:
  + **CloudWatch** (AWS) or **Stackdriver** (Google Cloud) to monitor the health and performance of the system.
  + Logs for user activity (e.g., bookings, modifications) should be stored and accessible for audit purposes.

**Email Notification Services:**

* **Email Integration**: The system will need an email service such as **SendGrid**, **AWS SES**, or **Mailgun** to send notifications, reminders, and confirmations to users regarding their lesson bookings and exam results.

**Scalability:**

* **Auto-scaling** mechanisms should be enabled to dynamically adjust the resources (e.g., CPU, memory) allocated to the application to ensure smooth functioning as the number of users increases.
* **Database Sharding** may be used to scale the database horizontally as the system grows.

**Performance and Reliability Requirements**

* **Uptime**: The system should have an uptime of 99.9%, meaning it must be operational nearly all the time to accommodate student bookings and training activities.
* **Response Time**:
  + The system should be optimized for fast response times, with most actions (e.g., booking a lesson or taking a test) completed within 2-3 seconds.
  + Database queries and API calls should be optimized to ensure performance even during peak traffic.
* **Redundancy**: The infrastructure should ensure redundancy through multi-zone deployment to minimize the risk of downtime or data loss.
* **Disaster Recovery**: A disaster recovery plan should be in place, with regular backups of the database and cloud-based recovery mechanisms to restore service in case of failure.