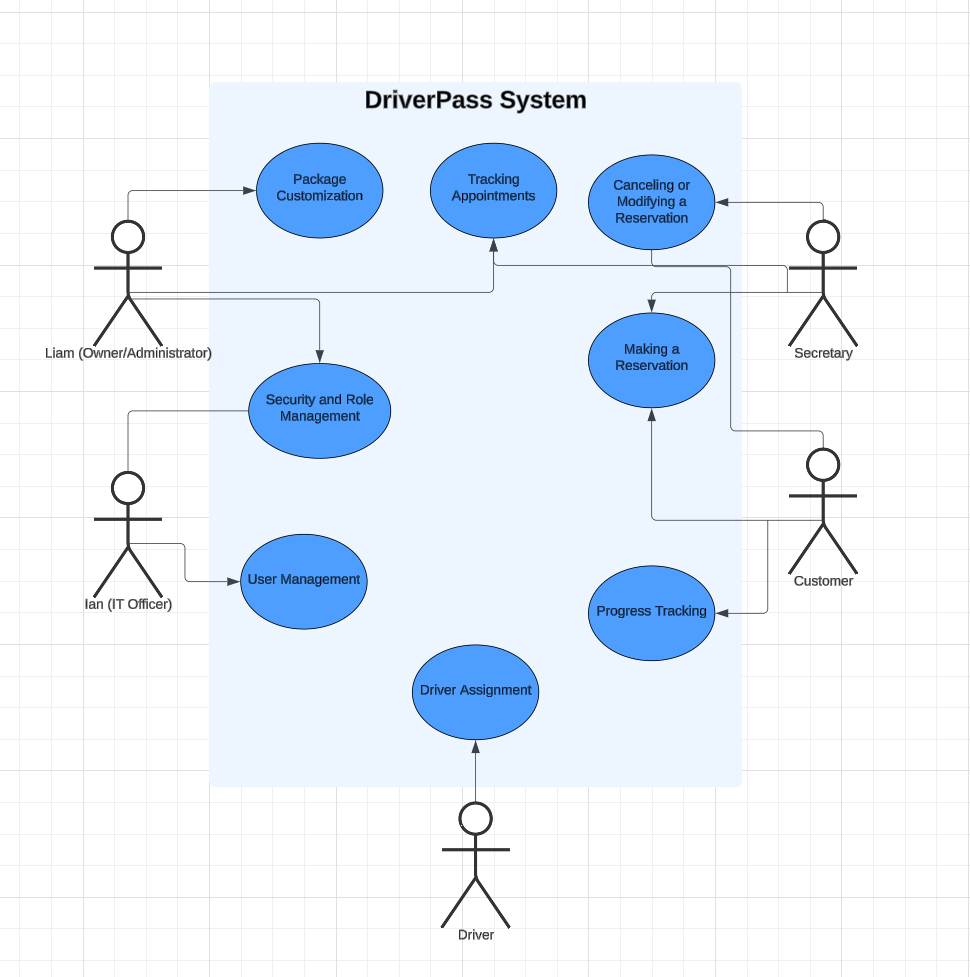
# CS 255 System Design Document Template

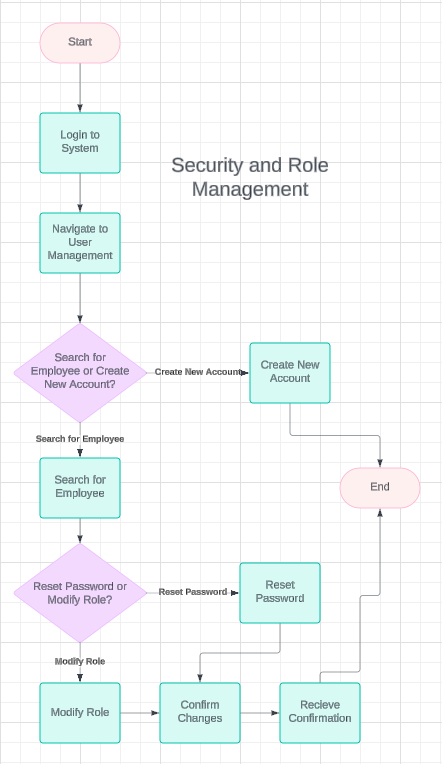
This template lays out all the different sections that you need to complete for Project Two. Each section has guidance to prompt your thinking. You will need to continually reference the interview transcript as you work to make sure that you are addressing your client’s needs. There is no required length 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.

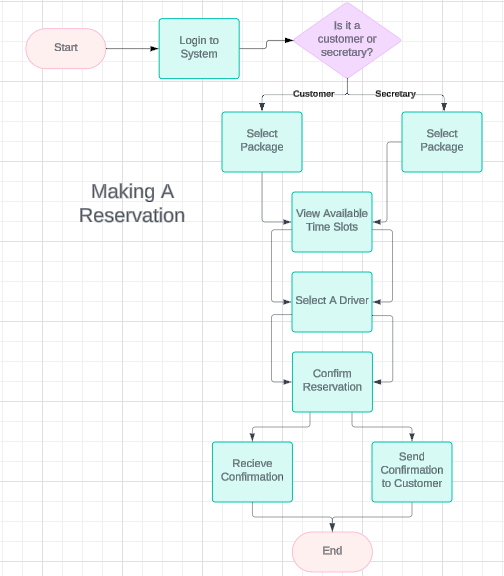
## UML Diagrams

### UML Use Case Diagram

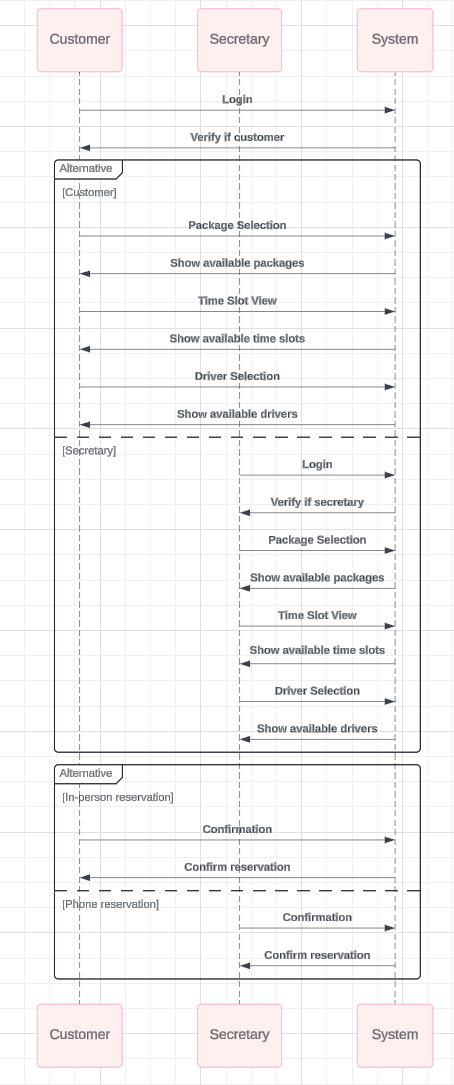


### UML Activity Diagrams

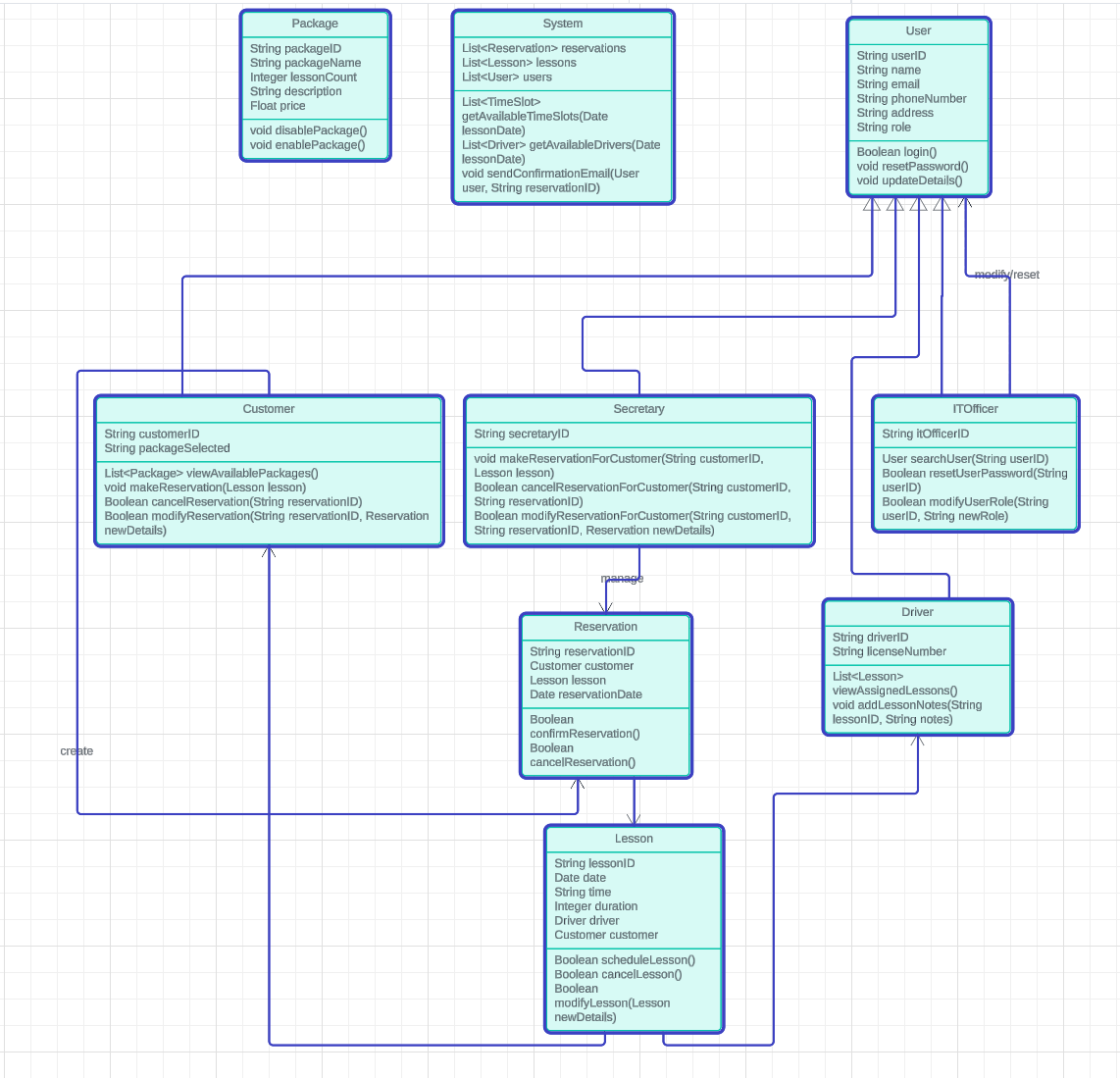




### UML Sequence Diagram



### UML Class Diagram



## Technical Requirements

#### **A. Hardware Requirements**

1. **Server Infrastructure**:
   1. **Web Servers**: Servers to host the web application. These should be high-availability servers that can handle a peak load of at least 1000 concurrent users.
   2. **Database Servers**: Dedicated database servers for storing user data, reservation details, driver schedules, etc.
   3. **Load Balancers**: For distributing traffic across multiple servers to ensure high availability.
   4. **Backup Hardware**: External storage or cloud storage solution for regular backups and disaster recovery.
2. **Client Devices**:
   1. **Customer/Secretary Devices**: Regular laptops, desktops, or mobile devices (smartphones, tablets) capable of accessing the web interface.
   2. **Driver Devices**: Tablets or smartphones for drivers to log in, view lessons, and add notes.

#### **B. Software Requirements**

1. **Operating System**:
   1. **Server OS**: Linux (e.g., Ubuntu or CentOS) or Windows Server depending on the server environment.
   2. **Client OS**: Not constrained by operating system; compatible with Windows, macOS, Android, and iOS.
2. **Web Application Framework**:
   1. **Backend**:
      1. **Node.js** (for handling backend logic and API requests).
      2. **Django/Flask (Python)** or **Spring Boot (Java)** as alternatives for robust backend frameworks.
   2. **Frontend**:
      1. **React.js** or **Angular** for building dynamic, user-friendly interfaces.
      2. **Bootstrap** for responsive UI across devices (desktops, tablets, and mobile phones).
3. **Database**:
   1. **Relational Database**: PostgreSQL or MySQL to handle structured data like user information, reservations, lessons, and driver schedules.
   2. **Cloud Database (optional)**: AWS RDS or Google Cloud SQL for managed database services.
   3. **NoSQL (optional)**: MongoDB for flexibility in storing unstructured data like logs, notes, or tracking data.
4. **Middleware**:
   1. **API Gateways**: Used for handling and routing external requests for package selection, reservations, and driver assignment.
   2. **Caching Services**: Redis or Memcached for caching frequently requested data, such as available lesson slots.
5. **Security**:
   1. **Authentication/Authorization**: OAuth 2.0 for secure user authentication (using Google, Facebook, etc.).
   2. **Encryption**: SSL/TLS certificates for encrypting communication between the web app and users (HTTPS).
   3. **Database Encryption**: AES-256 encryption for sensitive data, especially payment details.
   4. **Firewall & Security Tools**: Cloudflare or AWS Shield to protect against DDOS attacks and ensure secure traffic management.
6. **Tools and Frameworks**:
   1. **Version Control**: Git/GitHub or Bitbucket for source code management.
   2. **Continuous Integration/Continuous Deployment (CI/CD)**: Jenkins or GitLab CI for automating the build, testing, and deployment of the application.
   3. **Monitoring Tools**: New Relic or Datadog for performance monitoring and logging.
7. **Compliance Tools**:
   1. **Payment Gateway**: Stripe or PayPal for handling secure credit card transactions and PCI-DSS compliance.
   2. **DMV Integration**: APIs or webhooks for retrieving updated information from the DMV regarding driving test requirements.

#### **C. Tools for Development & Maintenance**

1. **Development Environment**:
   1. **IDE**: Visual Studio Code or JetBrains IntelliJ for developing and debugging the system.
   2. **Project Management**: Jira or Trello for tracking development progress, sprints, and issues.
   3. **Collaboration**: Slack or Microsoft Teams for communication between developers and stakeholders.
2. **Testing**:
   1. **Unit Testing**: Jest (for JavaScript), PyTest (for Python), or JUnit (for Java).
   2. **Integration Testing**: Postman for API testing.
   3. **User Acceptance Testing (UAT)**: A staging environment that mirrors production for testing by end-users before deployment.