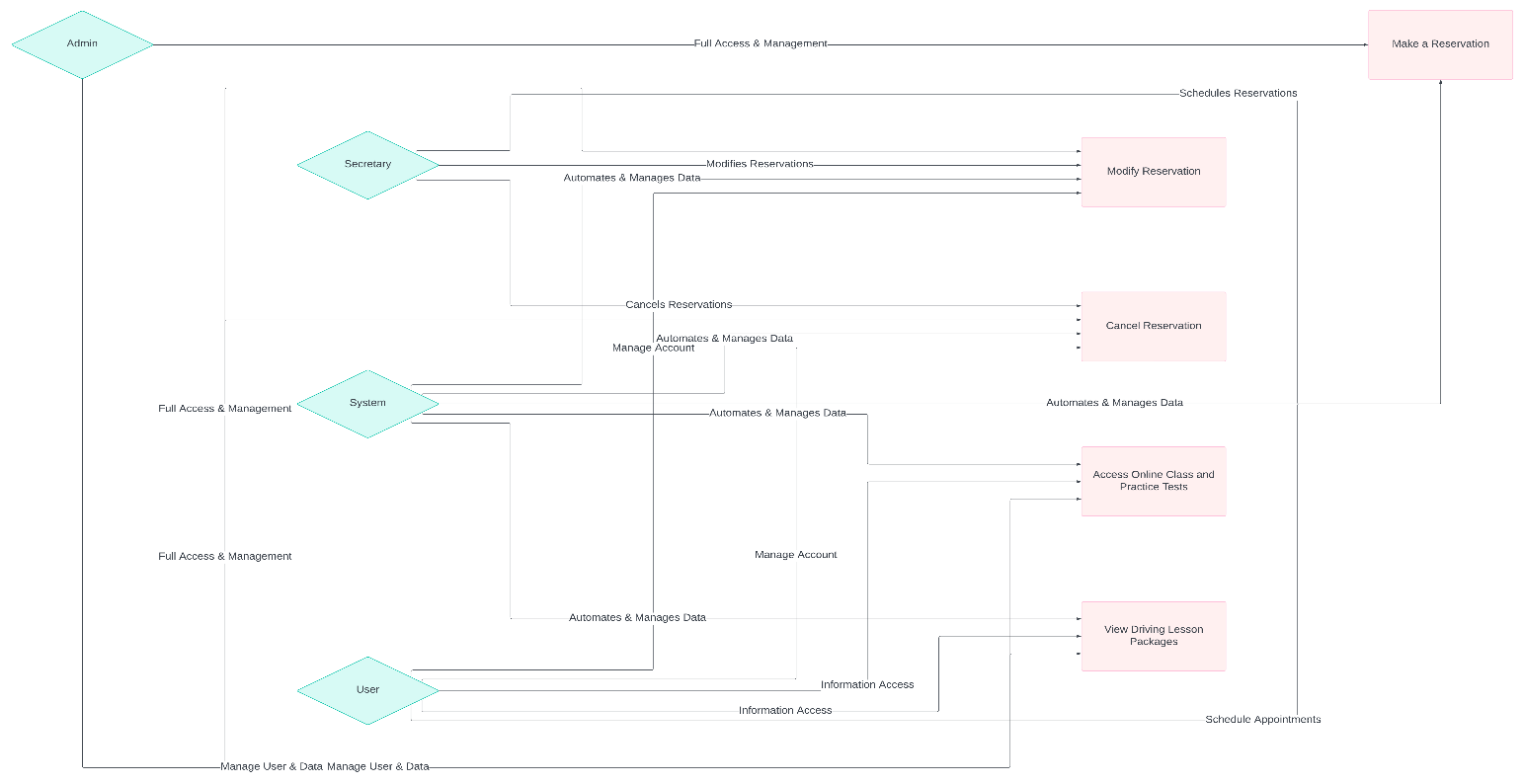
# CS 255 System Design Document Template

## UML Diagrams

### UML Use Case Diagram



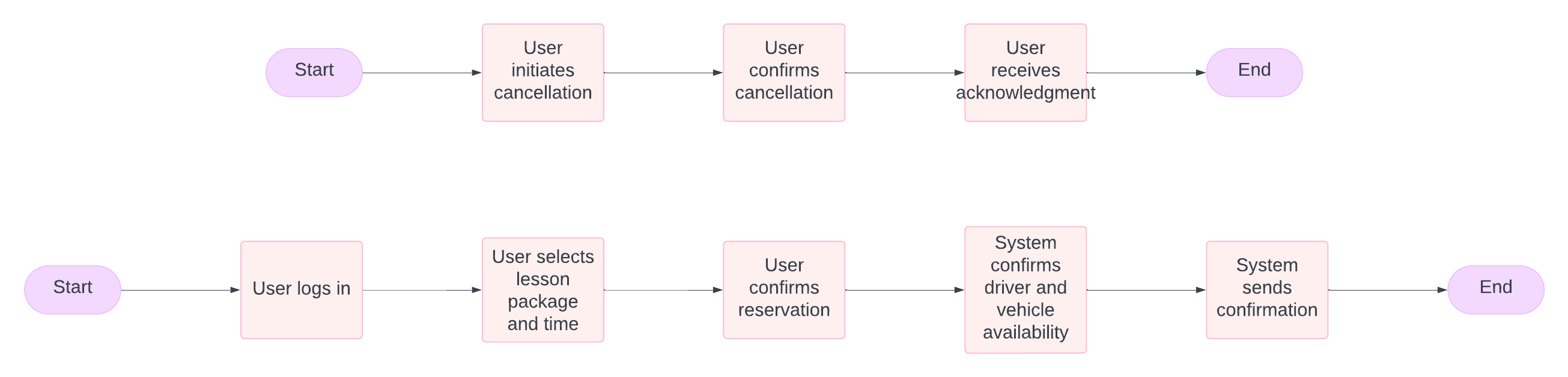
 **Actors**:

* Admin (full access, user and data management)
* User (appointment scheduling, account management)
* Secretary (schedules, modifies, or cancels reservations for users)
* System (automates tasks and manages data)

 **Use Cases**:

* Make a Reservation
* Modify Reservation
* Cancel Reservation
* View Driving Lesson Packages
* Access Online Class and Practice Tests​

### UML Activity Diagrams

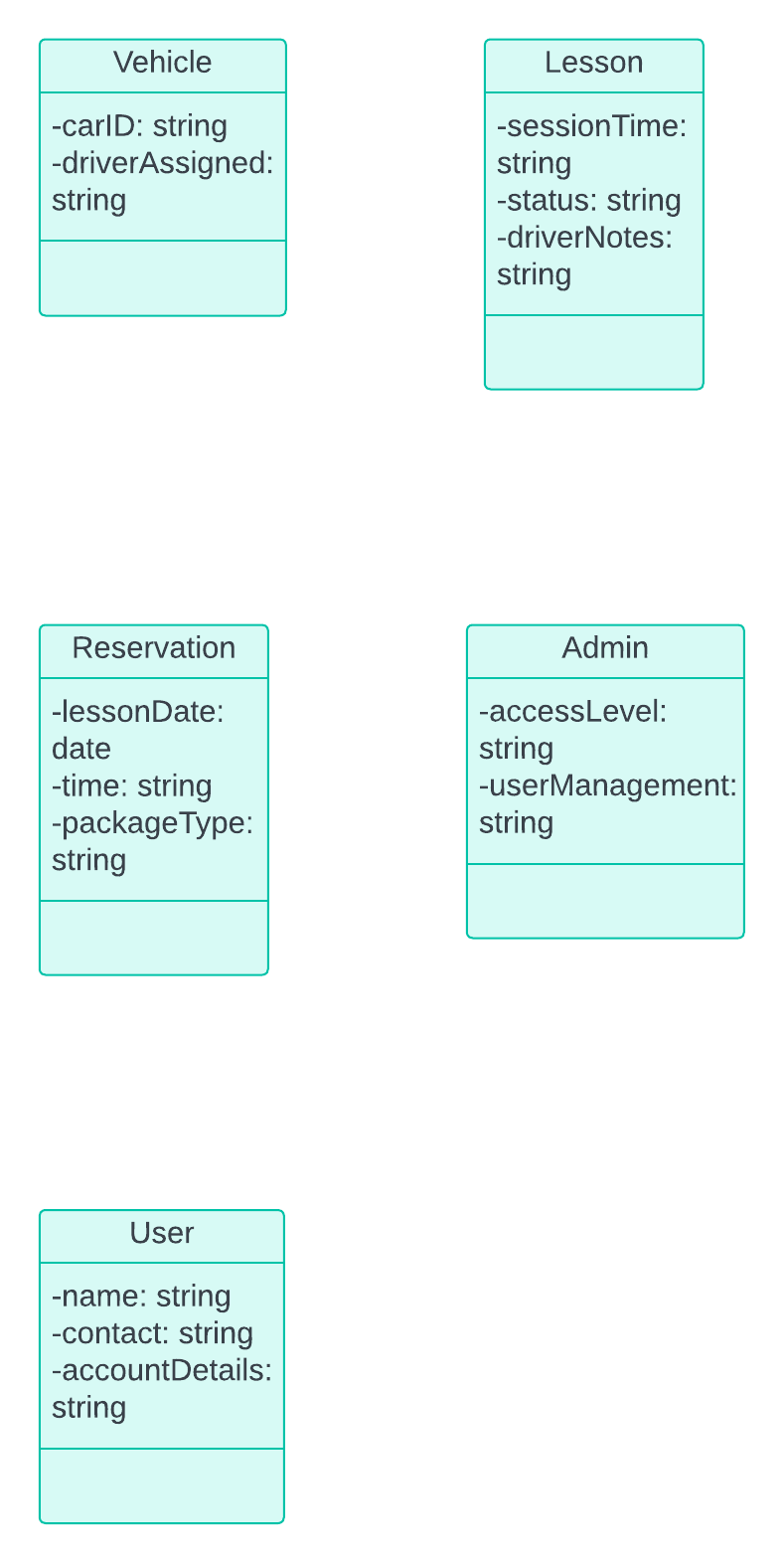
**

### UML Sequence Diagram

A diagram of a system

Description automatically generated

### UML Class Diagram



## Technical Requirements

**1. Hardware Requirements**

* **Server Infrastructure**:
  + **Cloud Hosting Server**: A scalable cloud server to host the web application, database, and file storage. This ensures remote accessibility, on-demand scaling, and low maintenance needs.
  + **Backup and Redundancy**: Cloud solutions with regular data backups and redundancy to prevent data loss and support system recovery, ensuring consistent availability.
* **Client Devices**:
  + **Employee Devices**: Desktop or laptop computers for administrators and secretaries to access the management interface.
  + **User Devices**: The system must be compatible with mobile devices, desktops, and tablets, enabling users to book and manage lessons from any device with internet access.

**2. Software Requirements**

* **Operating System**:
  + The cloud server should support a robust, server-optimized operating system, such as **Linux** or **Windows Server**.
* **Application Software**:
  + **Web Application Framework**: A framework like **Django (Python)** or **Node.js (JavaScript)** to support dynamic content and user interactions.
  + **Database Management System (DBMS)**: A relational database, such as **MySQL**, **PostgreSQL**, or a cloud-based solution like **AWS RDS**, for managing user, appointment, and lesson data. This ensures data integrity, quick query execution, and secure data handling.
  + **Authentication & Security Modules**: Libraries or modules to handle secure authentication and session management, such as **OAuth** for user authentication and **JWT (JSON Web Token)** for session tokens.
  + **APIs for External Integration**: The system may integrate with DMV or third-party services for rule and policy updates, utilizing REST or GraphQL APIs for real-time data synchronization.
* **Data Encryption & Security**:
  + **Data Encryption**: Secure user and payment information using **SSL/TLS** for data in transit and **AES-256** for data at rest.
  + **User Session Management**: **Token-based session management** to ensure secure login sessions, minimizing the risk of unauthorized access.

**3. Tools and Development Environment**

* **Development Environment**:
  + **Integrated Development Environment (IDE)**: IDEs like **Visual Studio Code** or **PyCharm** to streamline the development process.
  + **Version Control**: **Git** for version control, hosted on platforms like **GitHub** or **GitLab** for collaborative development, code backup, and version history.
  + **Testing Frameworks**: Testing tools such as **JUnit** (Java), **pytest** (Python), or **Jest** (JavaScript) for unit testing, ensuring the system performs as expected.
* **Diagramming & Modeling Tools**:
  + **Lucidchart** or similar CASE tools to create and update UML diagrams for design documentation.

**4. Infrastructure Requirements**

* **Web-Based Infrastructure**:
  + The system should run on a cloud-hosted **web server**, ensuring it’s accessible from any location with internet access. This reduces the need for on-premises hardware and allows for automatic updates, maintenance, and backups managed by the cloud provider.
* **Database**:
  + **Cloud Database**: Hosted on a secure cloud platform like **AWS RDS** or **Google Cloud SQL** to handle scalability and provide automated backup and recovery options.
  + **Relational Data Structure**: A relational database design to organize users, lessons, reservations, and packages in a structured manner that allows for easy data retrieval and reporting.
* **Network and Security**:
  + **Firewall Protection**: Firewalls at the network level to monitor and control incoming and outgoing traffic, adding a layer of protection.
  + **Role-Based Access Control (RBAC)**: Permissions are configured based on user roles (e.g., Admin, Secretary, Customer), restricting sensitive data access to only authorized users.
  + **Data Backup and Recovery**: Regular data backups with a defined recovery protocol to prevent data loss and ensure continuity.