# CS 255 System Design Document – Francisco Ortega

## DriverPass System UML Diagrams

### UML Use Case Diagram

A diagram of a diagram

Description automatically generated

Figure - UML Use Case Diagram for DriverPass System

### UML Activity Diagrams

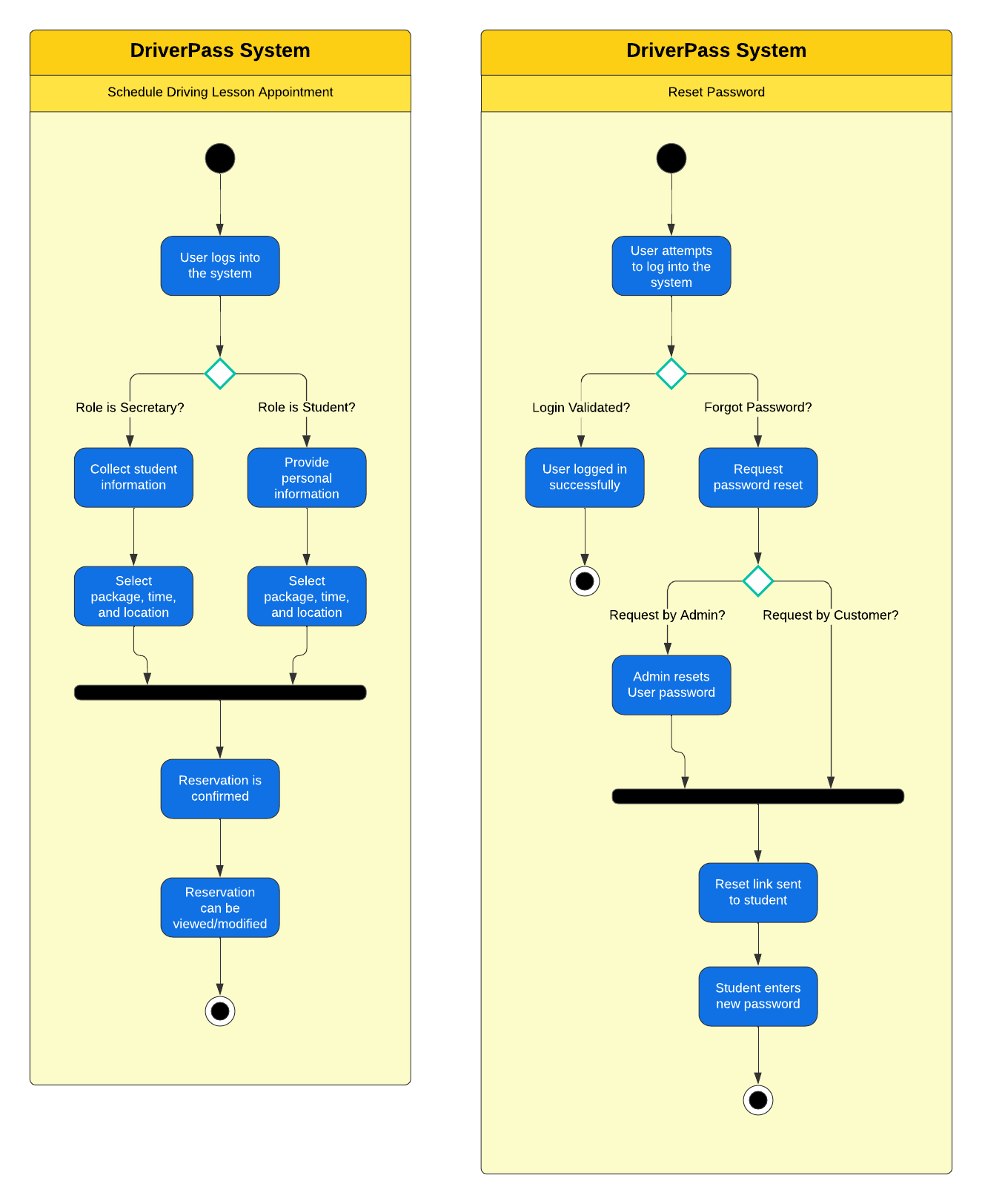


Figure - UML Activity Diagrams for DriverPass System Use Cases – Appointment Scheduling and Reset Password

### UML Sequence Diagram

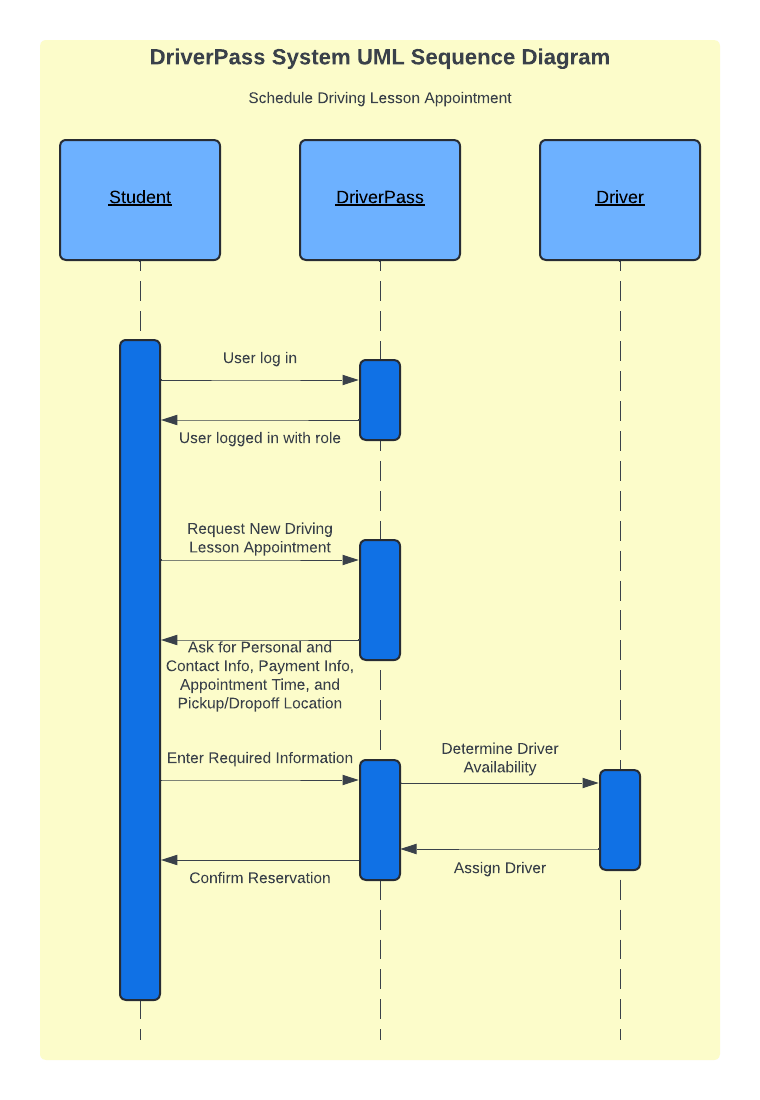
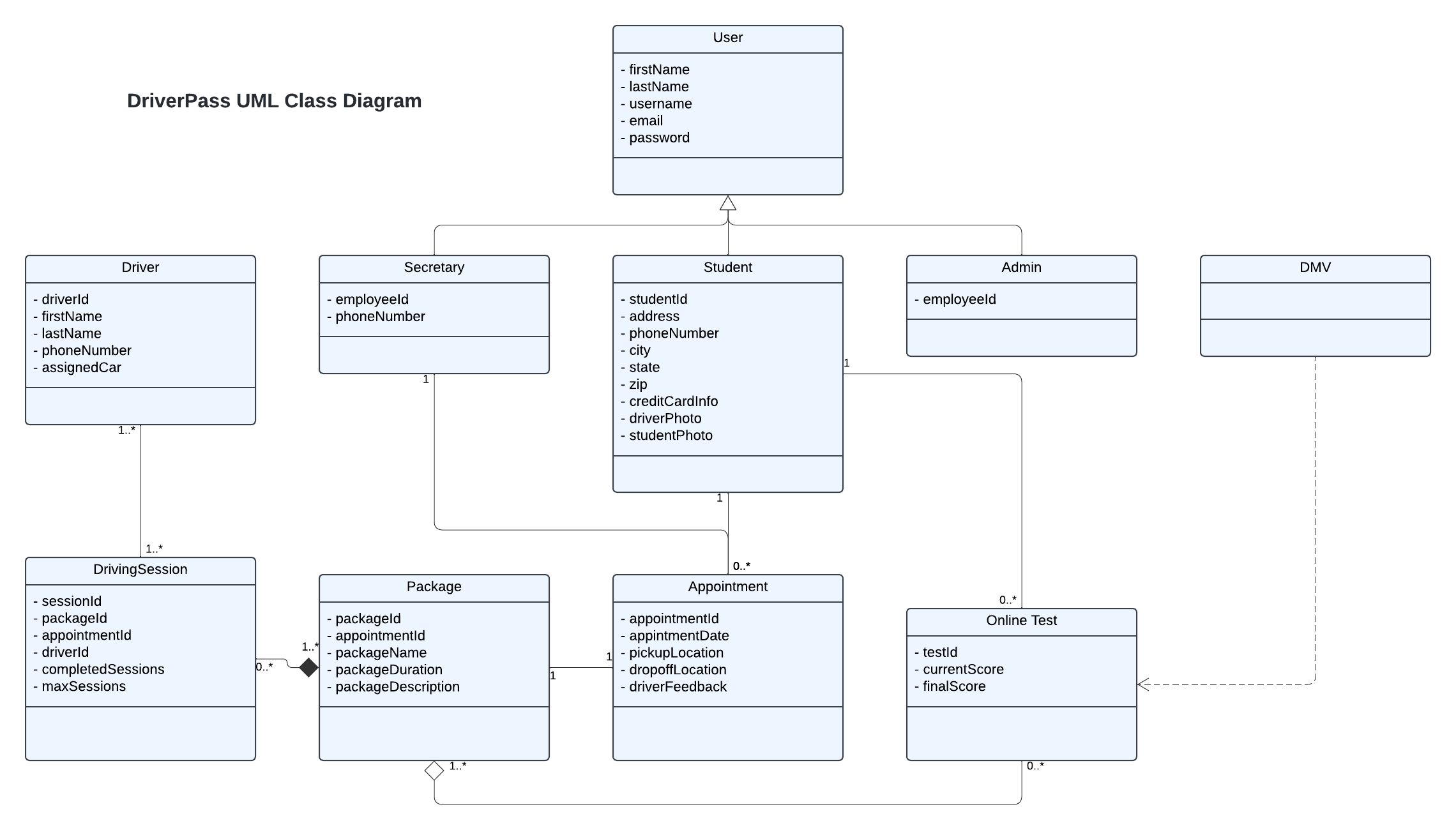
**

Figure - UML Sequence Diagram for DriverPass System Appointment Scheduling Use Case

### UML Class Diagram

**

## DriverPass System Technical Requirements

DriverPass boasts a wide range of functional and nonfunctional requirements for its system to reliably deliver the expected service to its users. These range from purely operational to carefully considered resiliency options that maximize user experience.

In terms of hardware, there are no strict requirements for the system apart from the phone, printer, and other peripherals that the Admins or Secretary might need to carry out their functions. Since the application will be deployed on the cloud, the cloud service provider will handle the physical servers themselves unless DriverPass opts for a hybrid solution. Apart from this required hardware, the client will only need a desktop or mobile device that can reliably access the Internet via a supported browser of choice.

As for software, DriverPass will need to ensure that its system can run on all major operating systems, i.e. Windows, macOS, Linux, iOS, and Android. This is also true for all major web browsers, including Chrome, Firefox, Edge, and Safari. A trusted database solution will be needed to ensure that customer information is reliably stored, appointments are appropriately added, updated or removed, and that packages and driver information is up to date. A relational database solution like PostgreSQL would work for the needs of the DriverPass system. Since the application will be web-based, programming languages like HTML, CSS, and JavaScript will be mandatory. However, there is flexibility in frameworks like React, Angular, Spring, and other web-based stacks. Most importantly, however, integration with the DMV will require use of an API to ensure compliance checks and updates can be made regularly and without disruption.

Tools needed for development will be a reliable IDE like Visual Studio Code or IntelliJ, based on the desired tech stack. Familiarity with logging and alarming tools provided by the cloud service provider will also be required to ensure that reports can be generated. Additional cloud infrastructure like load balancers should be considered to ensure that the application is highly available and resilient, especially when there is a surge in demand on the app. Overall, the system can be designed effectively when planned with just enough resources to meet the needs of the DriverPass team and users but without exceeding budget or timeline expectations.