# CS 255 System Design Document

## UML Diagrams

### UML Use Case Diagram

A diagram of a driver pass

Description automatically generated

### UML Activity Diagrams

This first activity diagram represents the use case of logging into the system.

A diagram of a system

Description automatically generated

This second activity diagram represents the use case of purchasing a package.

A diagram of a package

Description automatically generated

### UML Sequence Diagram

This sequence diagram represents the use case of purchasing a package.

A diagram of a system

Description automatically generated

### UML Class Diagram

This class diagram only includes attributes, not methods.

A diagram of a software company

Description automatically generated

## Technical Requirements

When it comes to infrastructure, the most important thing is having servers that can successfully handle the system without interruptions. First, the system should be accessible over the cloud, meaning that we will need cloud-based servers to host the system and its databases. There are many cloud-based servers to choose from, but the most popular ones are Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure. They each have their pros and cons, but they would all be solid choices to ensure efficiency and reliability. The system will also need security tools for encryption, firewall protection, and vulnerability scans, as well as automated backups to ensure that data doesn’t get lost or stolen.

In terms of hardware, the system will be accessible by pretty much any device with an internet connection, given that it meets the minimum specifications. This includes desktop computers, laptops, phones, tablets, etc. To run the system smoothly, this device should have at least a dual-core processor, 4 GB of RAM, and 25 GB of storage. If it doesn’t meet these requirements, then the system will have trouble operating efficiently.

When it comes to software requirements, we need a few different applications for building and testing the system, as well as third-party applications for payment processing and scheduling. For back-end work such as building the application, we will need a programming language and an IDE to create the code, and then testing software to test the code’s functionality. We will also need software to manage our databases and keep track of any code changes. As far as third-party applications, we will need one for processing customers' payments, and another for handling scheduling tasks. Lastly, when it comes to front-end work, we will need software for creating the user interface (UI) and user experience (UX).