# CS 255 System Design Document

## UML Diagrams

### UML Use Case Diagram

*A diagram of a company's flowchart

AI-generated content may be incorrect.*

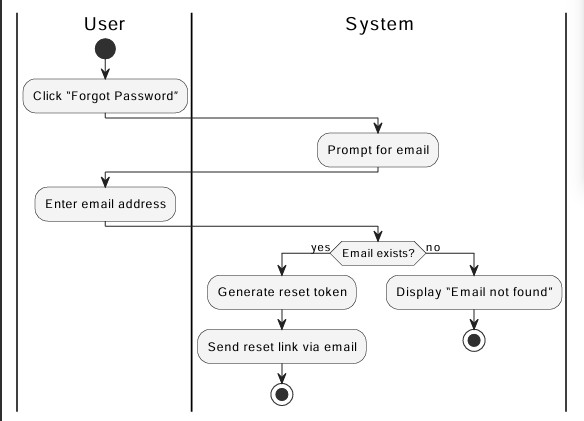
### UML Activity Diagrams

*Activity Diagram for “Make Reservation”*

*A diagram of a service

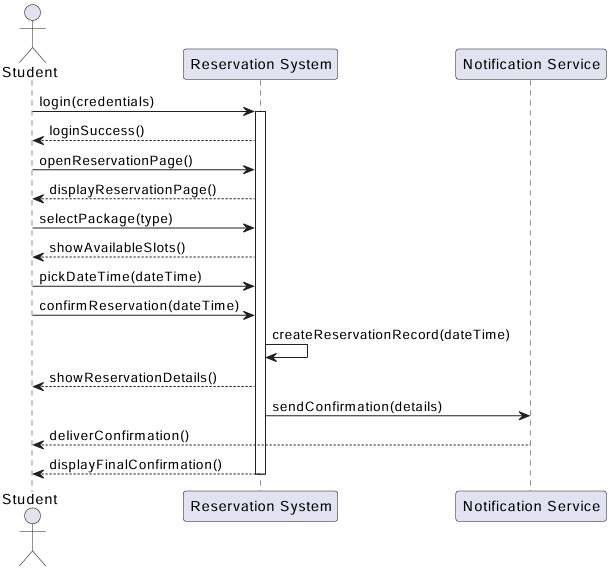
AI-generated content may be incorrect.*

*Activity Diagram for “Reset Password”*

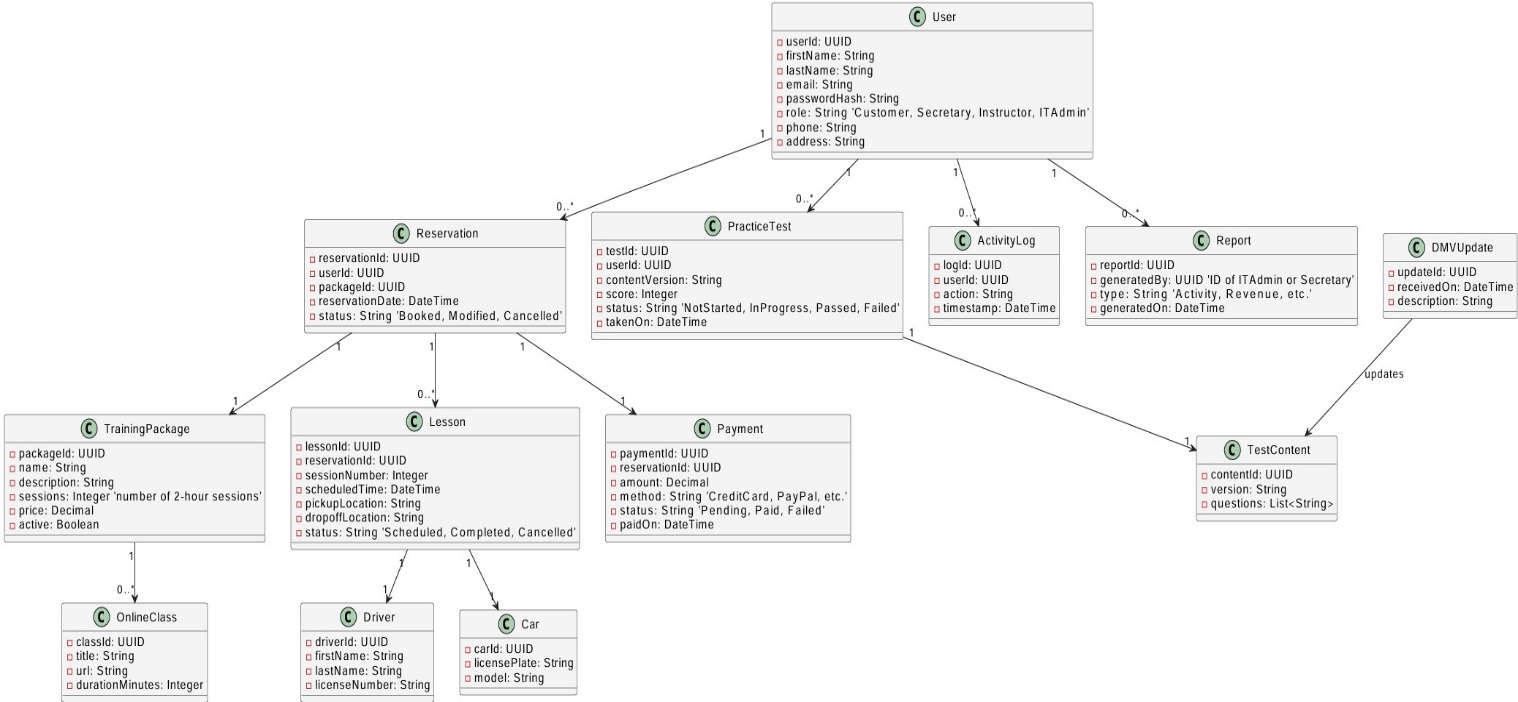
**

### UML Sequence Diagram

*UML Sequence Diagram*

*Use case: Make Reservation  
*

### UML Class Diagram

*UML Class Diagram system design  
*

## Technical Requirements

*-You need nothing but a browser, no special apps or installs*

*-We need a handful of cloud services, servers, database, file storage, and email to make everything fast, reliable, and secure*

*-Under the hood, we will use standard frameworks, containers, and automated pipelines so the system stays healthy, up to date, and easy for our team to maintain*

*1. Cloud Servers & Hosting*

*Our whole system lives “in the cloud” on secure servers that we rent instead of buying and maintaining ourselves. Think of these as always on computers somewhere else that handle every request.. whether someone’s logging in, booking a lesson, or pulling a report. We will put a traffic director (a load balancer) in front of them to spread work evenly, so nobody’s waiting around while one server is slammed and another sits idle.*

*2. Central Database & Storage*

*All of our data user profiles, lesson schedules, package details, activity logs, practice test questions goes into a single, reliable database in the cloud. This is the engine room: it keeps everything organized, searchable, and safe. We will also have a secure file bucket (a cloud folder) for any documents or report exports you want to download and edit at home.*

*3. Web & Mobile Access*

*Students, secretaries, instructors and managers will never need to install anything. Everyone just opens a modern browser (Chrome, Safari, Edge) on a laptop, tablet or phone and signs in. The system automatically adjusts its layout so it works just as well on a phone screen as on a big desktop monitor.*

*4. Application Layer & APIs*

*Behind the scenes, our “brains” live in an application framework.. essentially a set of building blocks that handle things like user authentication, booking logic, and syncing with the DMV update feed. Whenever someone clicks “Make Reservation” that request zips up to our application, which talks to the database, checks availability, and then confirms or gently explains why a slot isnt open.*

*5. Email & Notifications*

*We will plug in a trusted email service (like SendGrid or Amazon SES) so the system can automatically send password reset links, lesson confirmations, and activity alerts without exposing our own mail servers to spam filters or delivery headaches.*

*6. Security & Backups*

*Every bit of data in motion is wrapped in a secure envelope (SSL/TLS), so nobody can eavesdrop on passwords or personal details. Passwords themselves are scrambled with a one way hash before we ever store them. We will also configure automatic backups likw snapshots of our database and files, so we can restore everything to a known good state if something ever goes wrong.*

*7. Developer & Operations Tools*

*On the operations side, we will use version control (GitHub or similar) to track every change to our code and configuration. We will containerize our application (Docker) so it runs the same way everywhere, whether on a developer's laptop or in production. Automated pipelines (CI/CD) will test, build, and deploy updates at the push of a button, keeping downtime to near zero.*

*8. Monitoring & Maintenance*

*To keep an eye on system health, we will plug into a lightweight monitoring service that watches server load, error rates, and response times. If something spikes.. maybe a sudden rush of students booking weekend lessons, we will get an alert so we can scale up resources or investigate before it affects users.*