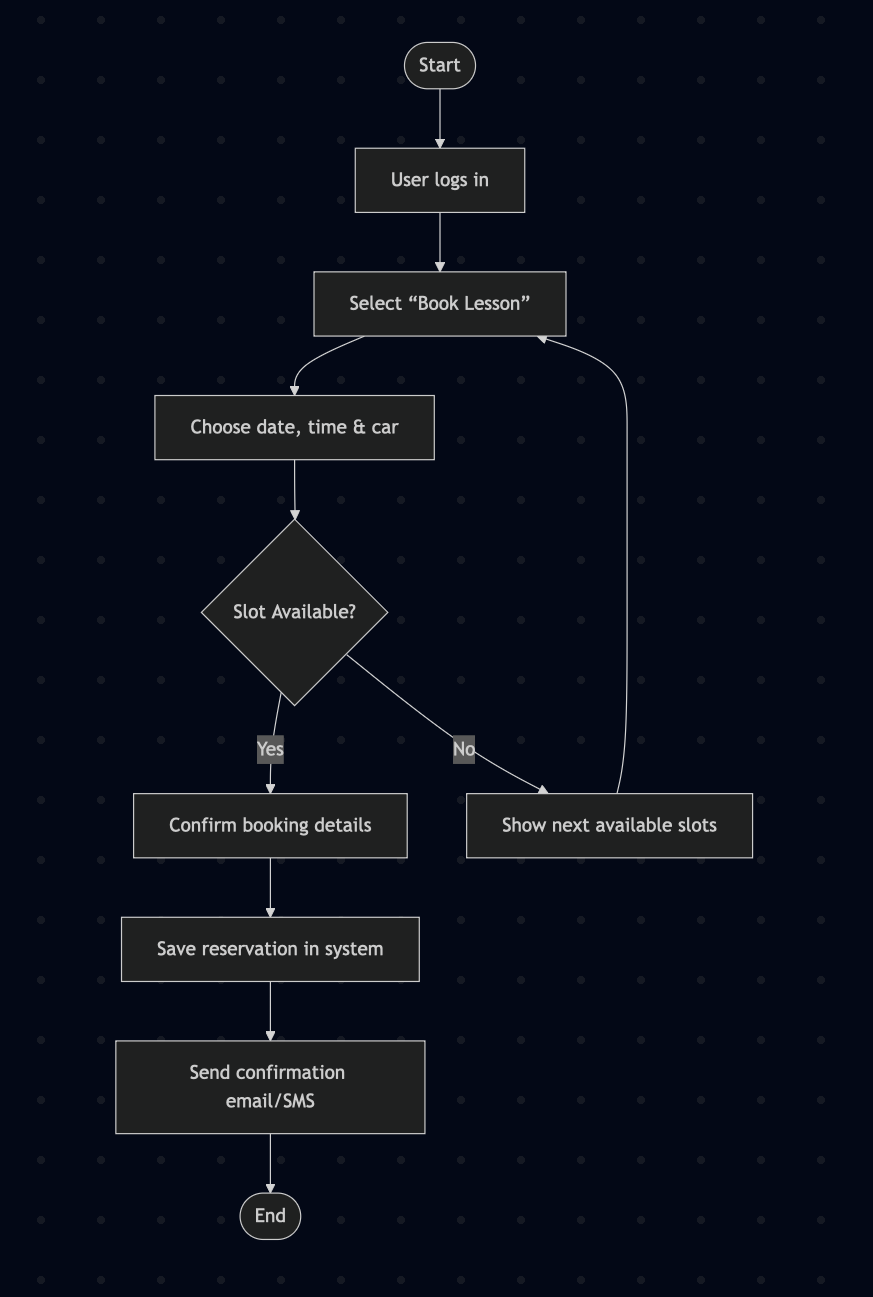
# CS 255 System Design Document Template

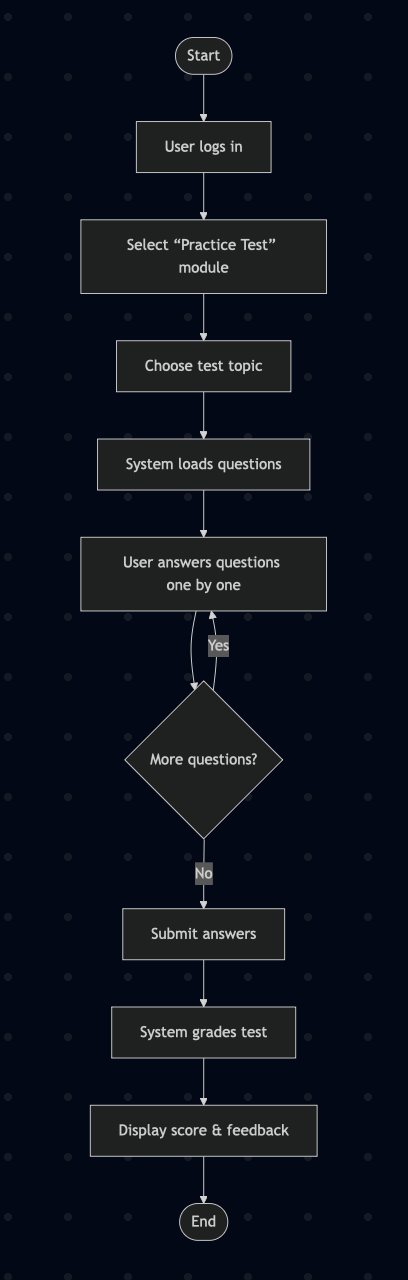
## UML Diagrams

### UML Use Case Diagram

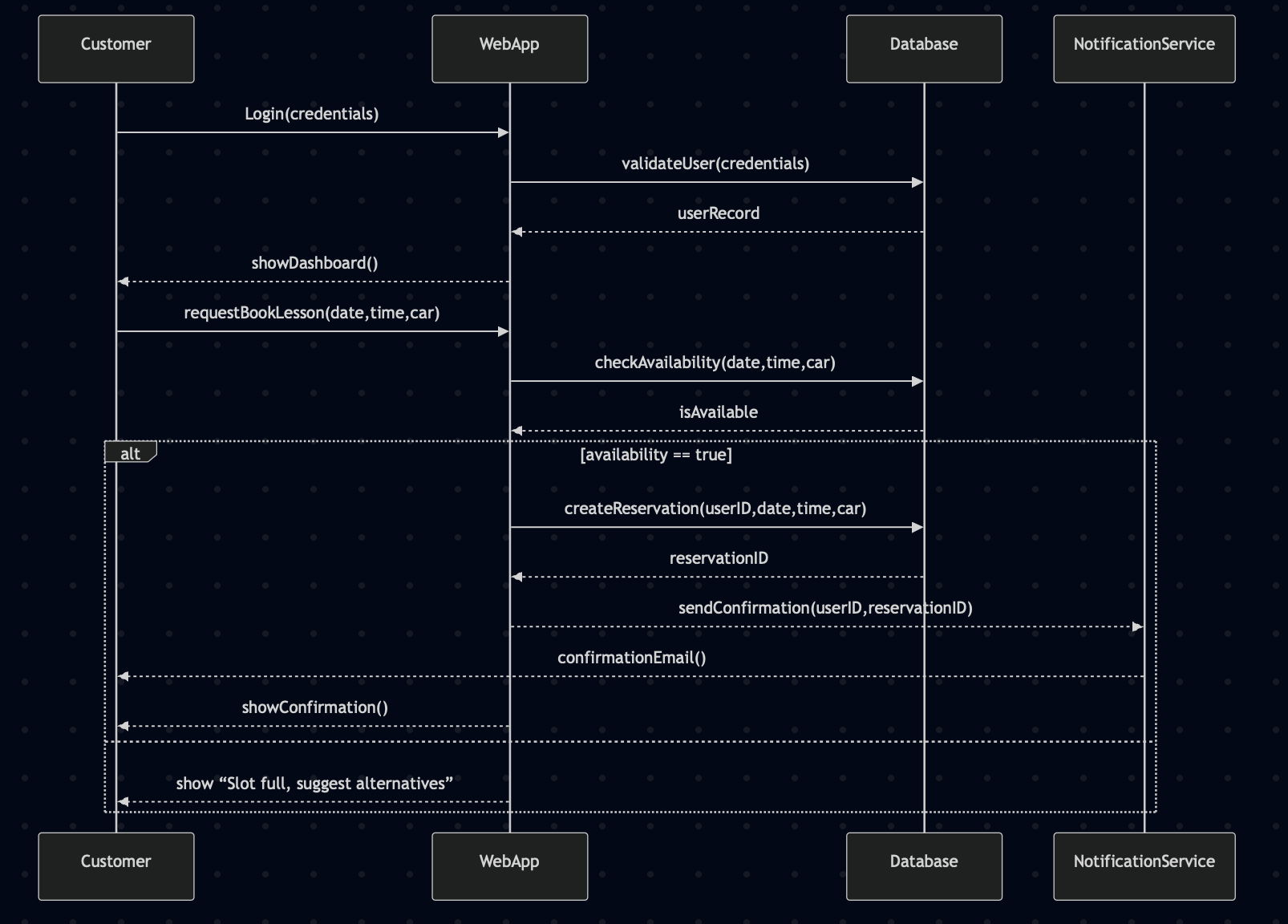
### 

### UML Activity Diagrams

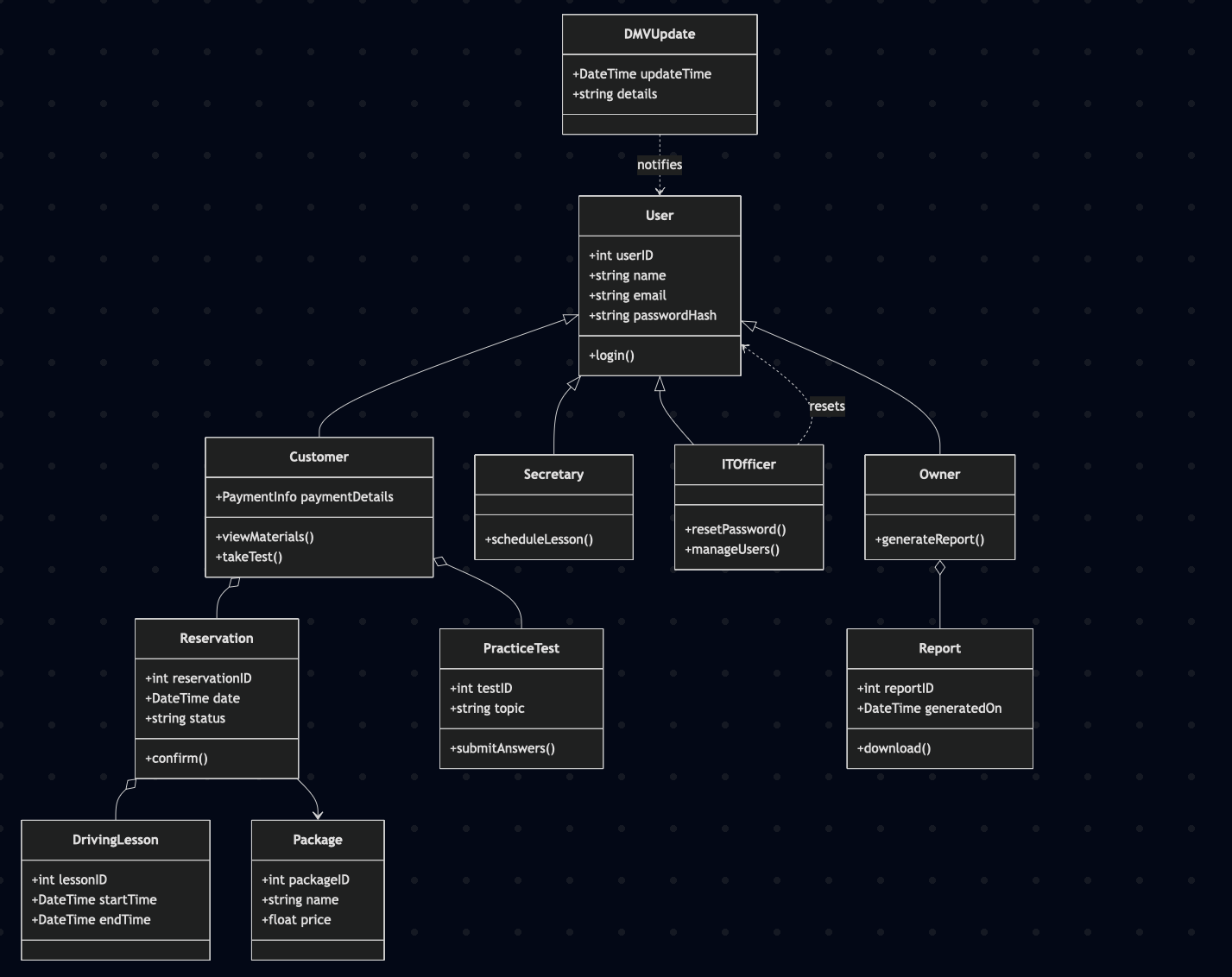




### UML Sequence Diagram



### UML Class Diagram



## Technical Requirements

**Hardware Requirements**

* A set of Linux‐based servers, each equipped with at least 4 vCPUs and 16 GB of RAM, will host the web application and related services.
* Solid‑state drives arranged in RAID 1 will provide fast, fault‑tolerant storage for the database.
* A dedicated network firewall (physical or virtual) must protect internal traffic and enforce access rules.
* End users will need modern desktops, laptops, tablets, or smartphones running up‑to‑date browsers (Chrome, Firefox, Safari, Edge).

**Software Requirements**

* The client interface will use a JavaScript framework in the browser and communicate over HTTPS.
* The server‑side component will expose RESTful APIs in a statically typed language or popular server‑side JavaScript runtime.
* PostgreSQL will be the primary relational database, storing users, reservations, lesson records, and test results.
* Nginx will act as a reverse proxy, handling SSL/TLS termination and forwarding requests to application servers.

**Development & Deployment Tools**

* Git will manage source code with a shared remote repository that supports pull requests and code reviews.
* A continuous integration/delivery pipeline will automatically build, run tests, and deploy new releases when merging with the main branch.
* An issue‐tracking system will log feature requests, bugs, and support tickets for the team to triage.
* Monitoring and alerting tools will continuously check server health metrics (CPU, memory, response time) and notify staff of critical conditions.

**Infrastructure Requirements**

* All services will run in a cloud environment capable of auto‑scaling and multi‑region failover for high availability.
* A load balancer will distribute incoming traffic across multiple application servers to keep performance steady under heavy use.
* A content delivery network will cache static assets (images, scripts, stylesheets) at edge locations to improve page load times.
* Automated backup routines with point‑in‑time recovery will protect the database and enable fast restoration after data loss.