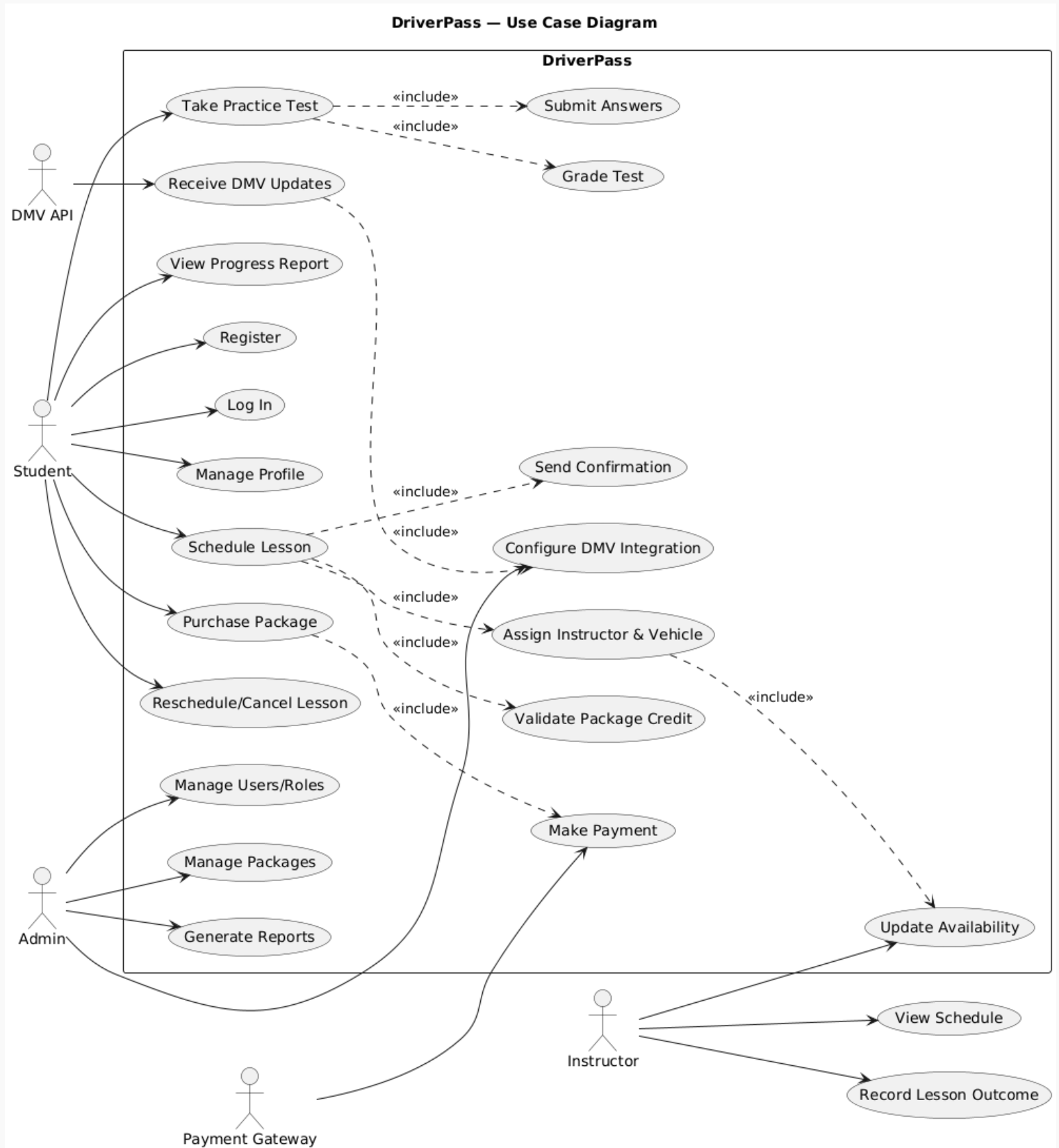
# CS 255 System Design Document Template

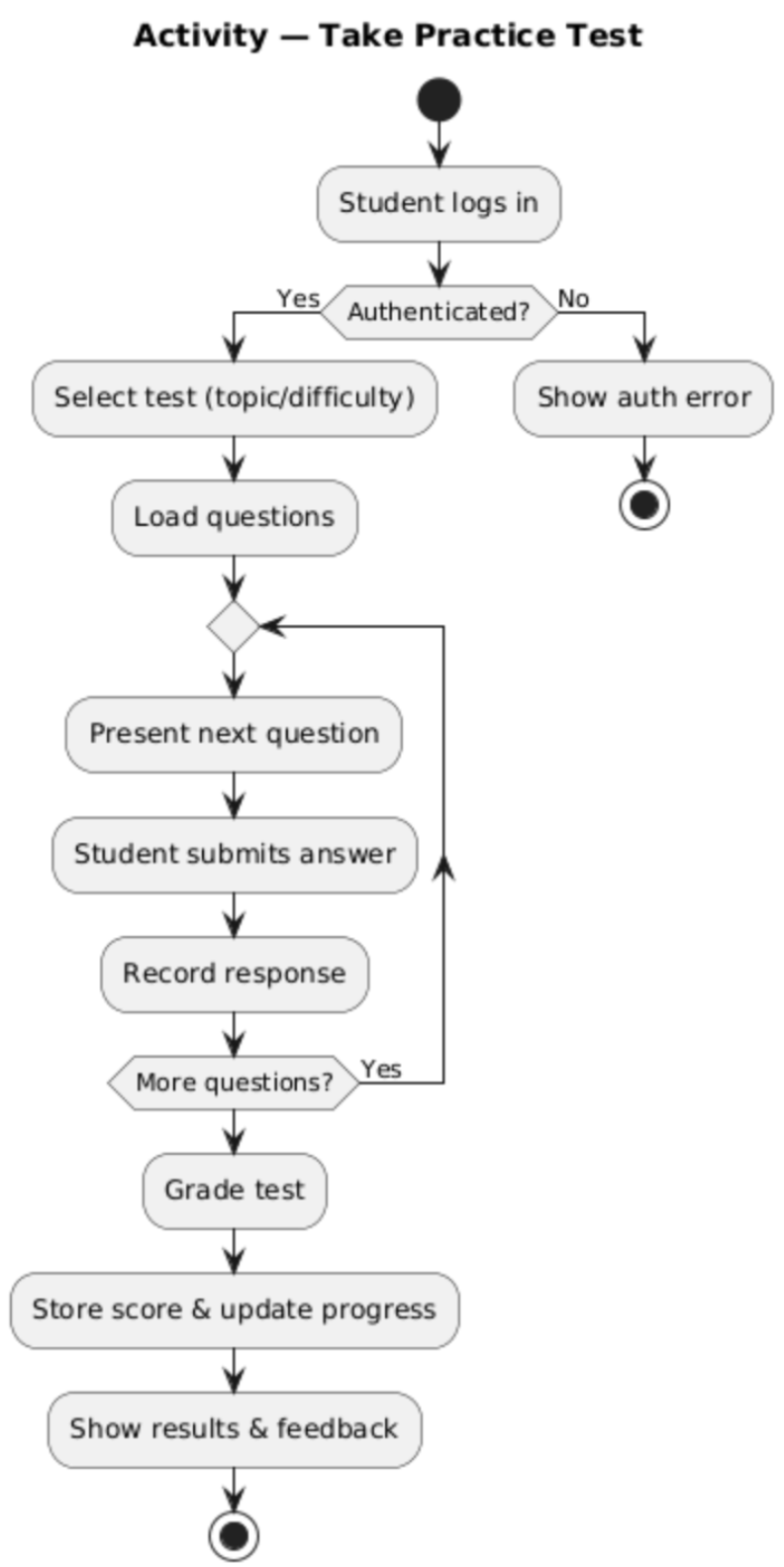
This template lays out all the different sections that you need to complete for Project Two. Each section has guidance to prompt your thinking. You will need to continually reference the interview transcript as you work to make sure that you are addressing your client’s needs. There is no required length for the final document. Instead the goal is to complete each section based on what your client’s needs are. Remove this note when you are finished, and replace all bracketed text with the relevant information.  
  
**By: Brandon Patrick**

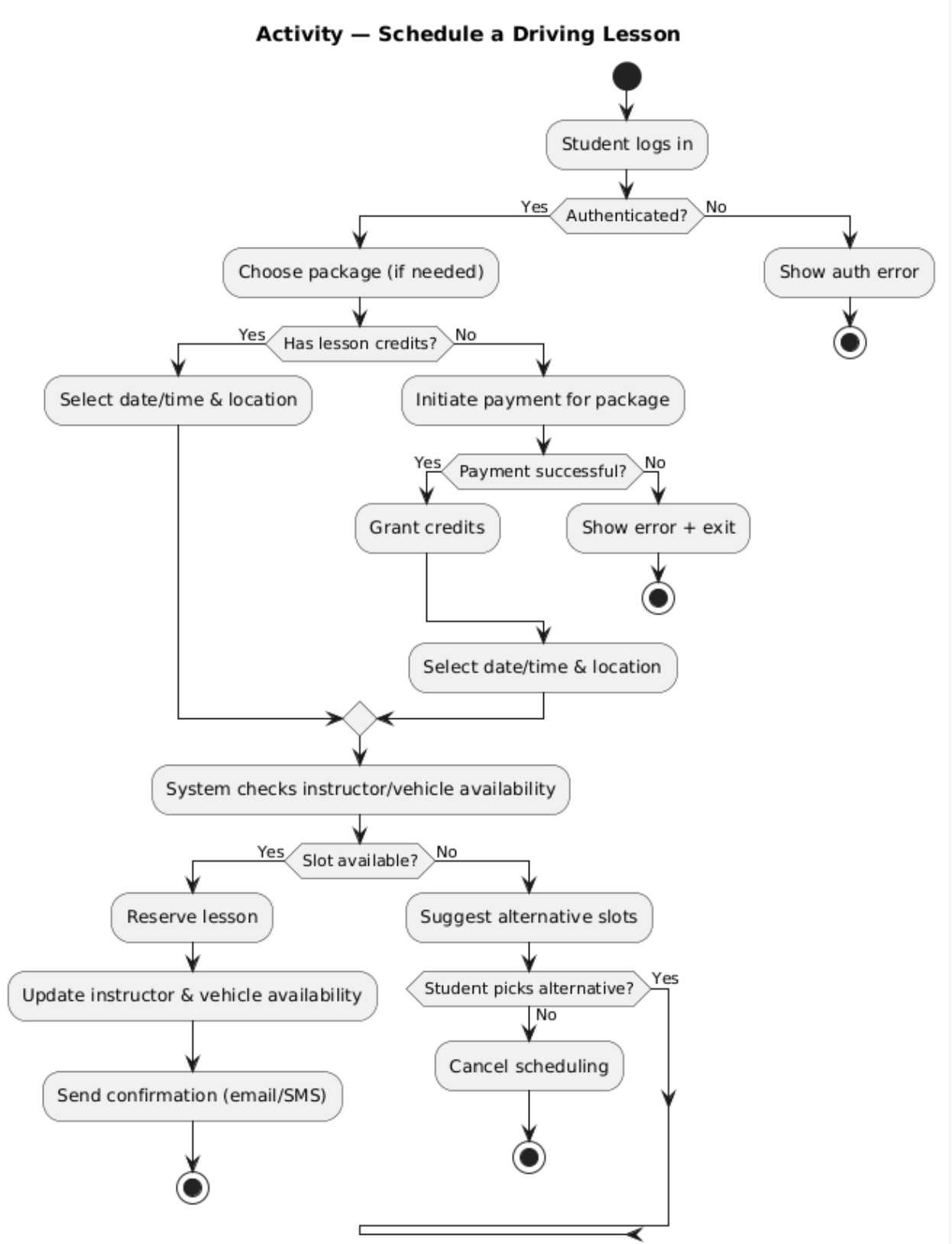
## UML Diagrams

### UML Use Case Diagram

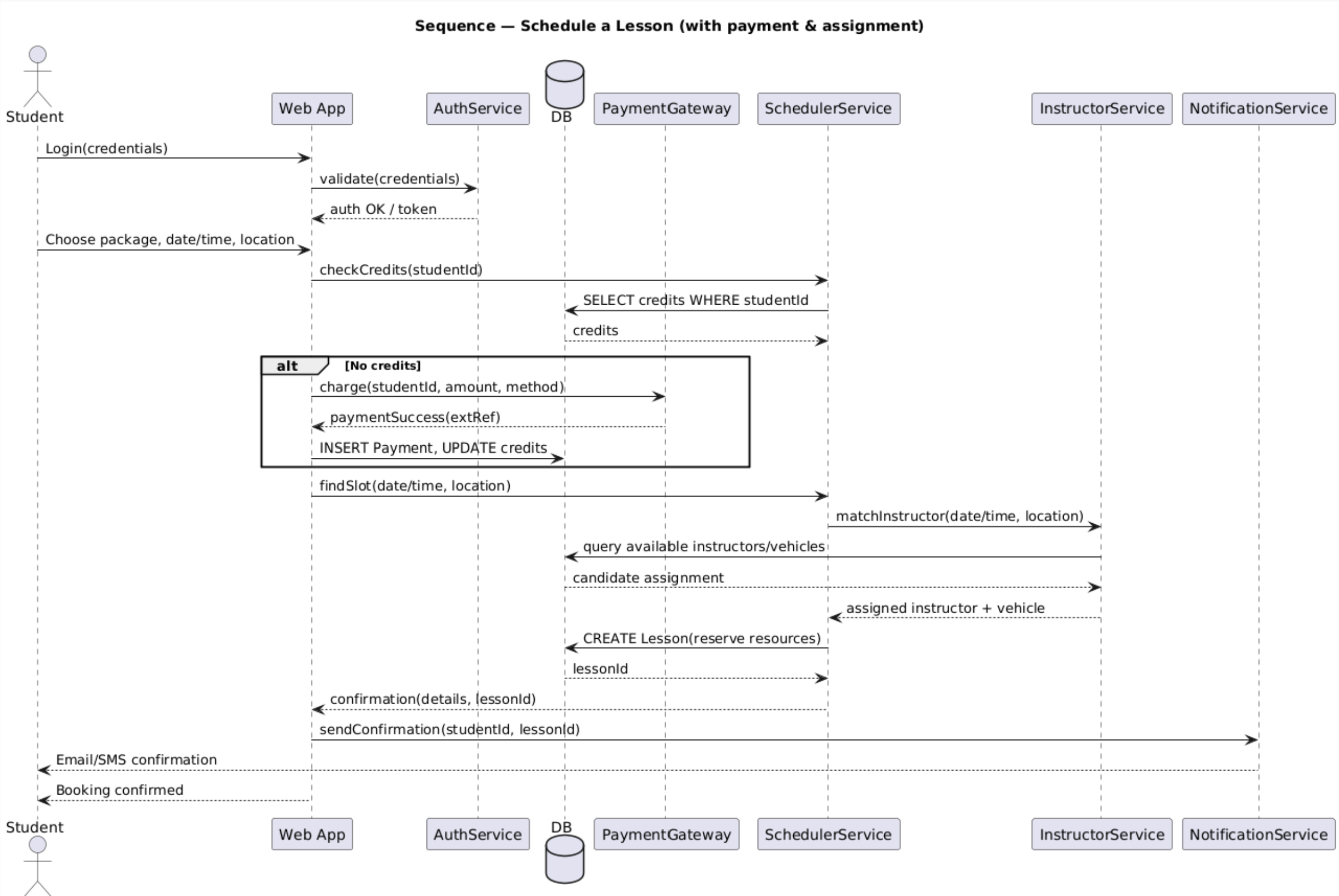


### UML Activity Diagrams

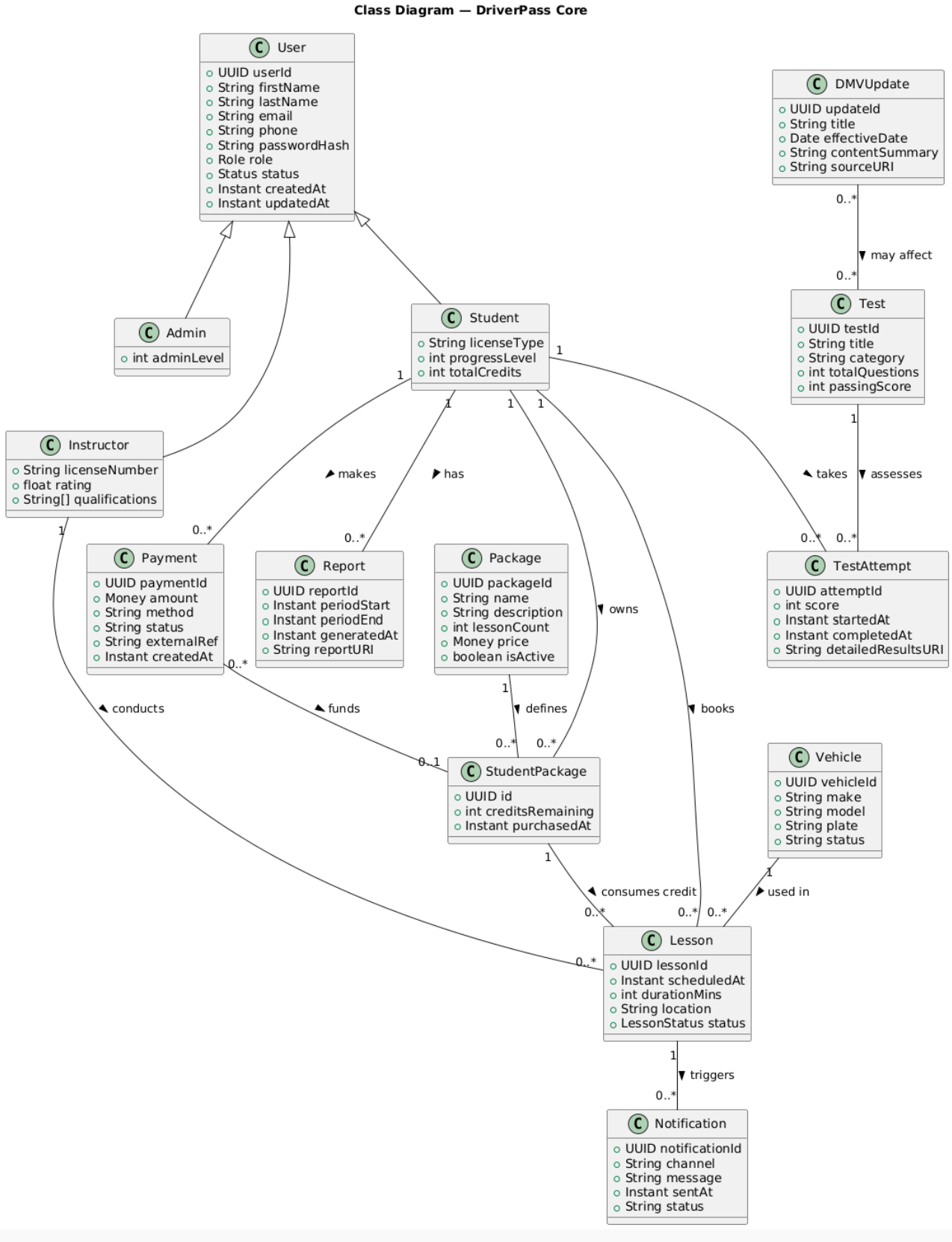




### UML Sequence Diagram



### UML Class Diagram



## Technical Requirements

## **Hardware & Hosting**

* **Cloud hosting** on AWS/Azure/GCP with at least two availability zones. Start with a small autoscaling group (2–3 app instances, t-shirt size ~2 vCPU/4–8 GB RAM each) behind a managed load balancer.
* **Managed database** (e.g., Amazon RDS/Azure Database) with automatic backups and multi-AZ failover; baseline: 2 vCPU/8 GB RAM, 100–200 GB SSD storage to start.
* **Object storage** (S3/Blob Storage) for reports, images, and exported CSV/XLS files.
* **Caching layer** (Redis/MemoryStore) to offload frequent reads (test banks, schedules).

## **Software Stack**

* **Backend**: Modern web framework (e.g., Java/Spring Boot or Node.js/Express). Provides REST APIs for auth, scheduling, exams, reporting.
* **Frontend**: Responsive web app (React or Vue) supporting desktop and mobile browsers (latest Chrome, Edge, Safari, Firefox).
* **Database schema**: Relational (PostgreSQL or MySQL). Core tables: Users, Roles, Students, Instructors, Packages, StudentPackages (credits), Lessons, Vehicles, PracticeTests, TestAttempts, Payments, ActivityLog, Notifications, DmvUpdates.
* **Authentication/Authorization**: JWT or OAuth2; role-based access control (student, instructor, secretary, admin, owner).
* **Background jobs**: Worker service for grading, notifications, report generation, and DMV sync.

## **Integrations**

* **Payment Gateway** (Stripe/Authorize.Net): tokenized card entry (DriverPass never stores PAN), webhooks for confirmations/refunds.
* **DMV content feed**: Secure REST API or webhook to ingest updated rules/questions; version content and log changes.
* **Email/SMS**: Transactional provider (Amazon SES, SendGrid, Twilio) for confirmations, reminders, password resets.

## **Security & Compliance**

* **Transport security**: HTTPS (TLS 1.2+) everywhere; HSTS; modern ciphers.
* **Secrets management**: Cloud KMS/Key Vault; never hard-code secrets.
* **Data protection**: At-rest encryption (DB/storage), password hashing (bcrypt/Argon2), PII minimization, field-level encryption for sensitive metadata if stored.
* **Account safety**: Password complexity policy, MFA option, lockout after failed attempts, secure password reset via time-bound token.
* **Auditability**: ActivityLog for create/modify/cancel operations and admin actions; immutable timestamps and actor IDs.
* **Payments**: Reduce PCI scope by using gateway-hosted fields/checkout; store only tokens and last-4/brand.

## **Reliability, Performance & Scale**

* **Availability target**: ≥99.5% initial; blue/green or rolling deployments.
* **Autoscaling**: Scale on CPU, latency, or queue depth; CDN for static assets.
* **Caching**: Redis for read-heavy data; client-side caching/ETags.
* **Backups & DR**: Automated daily DB snapshots (35-day retention), object storage lifecycle rules, quarterly restore tests; RTO ≤ 4 hrs, RPO ≤ 24 hrs to start.

## **Observability & Operations**

* **Logging**: Structured JSON logs; central aggregation (CloudWatch/Log Analytics); redact PII.
* **Metrics/Tracing**: APM (New Relic/DataDog/OpenTelemetry) for API latency, error rates, queue times.
* **Alerting**: On-call alerts for uptime, 5xx spikes, payment/webhook failures, DMV sync errors.
* **CI/CD**: Git-based workflow, automated tests, vulnerability scans (SCA/SAST), container/image signing; staged environments (dev, test, prod).

## **Environment & Tools**

* **Development**: IDEs (IntelliJ/VS Code), Postman/Insomnia for API tests, Jest/JUnit for unit tests, Cypress/Playwright for UI tests.
* **Issue tracking**: Jira/GitHub Projects; documentation in Confluence/Notion.
* **Diagramming**: Lucidchart for UML (use case, activity, sequence, class).

## **Network & Access**

* **VPC** with private subnets for app/DB, public subnets for load balancer/CDN.
* **Access control**: Least-privilege IAM roles; VPN/Zero Trust for staff consoles; IP allowlists for admin endpoints.

## **Browser & Device Support**

* **Browsers**: Latest 2 versions of Chrome, Edge, Safari, Firefox.
* **Mobile**: Responsive layout; optional PWA for notifications/offline stubs (view-only).

## **Nonfunctional Targets (initial)**

* **Auth** < 400 ms p95; **Core API** < 500–700 ms p95 under expected load.
* **Concurrency**: 500–1,000 active users baseline; horizontal scale path documented.
* **Data retention**: Test attempts and audit logs retained ≥ 2 years (configurable).