# CS 255 Business Requirements Document

**DriverPASS BRD**

*Author: Quazi Mainuddin*

*Date: 08/14/2025*

*Document: DriverPass BRD*

## System Components and Design

### Purpose

* DriverPass is a startup focused on improving driver education and DMV test pass rates.
* The purpose of this project is to design and develop a cloud-based web system to support DriverPass’s services, It should include:
  + online classes
  + practice tests
  + appointment scheduling
  + in-person driving sessions
* The client wants the system to support online and mobile access, role-based user permissions, activity tracking, and appointment scheduling.
* The system should also allow exporting data for offline access and enable admin control over training package availability.

### System Background

* DriverPass seeks to address the high failure rate of DMV driving tests due to lack of comprehensive training options.
* Their goal is to provide flexible, quality driver education through a combination of:
  + Online driver education and practice tests
  + In-person lessons explaining DMV policies
  + On-the-road driving instruction in scheduled time blocks
* The system must include:
  + Role-based user access (Admin, IT Officer, Secretary, Student)
  + Package management (three training packages with future expandability)
  + Reservation/scheduling system for lessons
  + Online course tracking and test results
  + Integration with DMV updates
  + Secure account creation, login, and password reset functionality

### Objectives and Goals

* The system should allow students to:
  + Register, log in, and reset passwords online
  + Schedule, cancel, or modify driving lessons
  + View lesson history, driver comments, and training progress
  + Access and complete online practice tests and lessons
* The system should enable administrators and staff to:
  + Manage user accounts and permissions
  + Track lesson reservations and driver assignments
  + Export reports (e.g., student progress, appointments)
  + Modify which training packages are active or inactive
  + Monitor system activity logs (who made, changed, or canceled bookings)
* The system should:
  + Notify staff when DMV policies or content are updated
  + Securely handle personal and payment data
  + Run on cloud infrastructure with minimal local IT burden

## Requirements

### Nonfunctional Requirements

#### Performance Requirements

* The system shall operate as a cloud-based web application, accessible via standard modern browsers (Chrome, Firefox, Safari).
* The system must respond to user interactions (login, scheduling, test submissions) within 2 seconds under normal load.
* Data should be backed up daily, and maintenance updates should be scheduled during non-peak hours with minimal downtime (less than 1 hour monthly).

#### Platform Constraints

* The system must be browser-accessible on Windows, macOS, Android, and iOS platforms.
* The backend will rely on a relational database (e.g., MySQL or PostgreSQL) and be hosted on a scalable cloud infrastructure (e.g., AWS or Azure).
* Development must use standard web technologies (HTML5, CSS, JavaScript, RESTful APIs).

#### Accuracy and Precision

* The system shall enforce strict role-based access control (RBAC) using unique usernames and passwords.
* All form inputs must be validated both client- and server-side for completeness and correct formats (e.g., phone numbers, credit cards).  
  The system will notify administrators via email/log entries for invalid login attempts, data conflicts, or errors during reservation processing.

#### Adaptability

* Admin users shall be able to activate or deactivate training packages via the interface without code changes.  
  IT admin should have the ability to reset accounts and block access via an admin dashboard.
* The system must be built modularly to support future feature additions (e.g., new lesson types, reporting formats).

#### Security

* All connections between clients and the server must be secured via HTTPS.
* User accounts must be protected by hashed passwords and optional two-factor authentication (2FA) for staff accounts.
* After 5 failed login attempts, the system shall temporarily lock the account and notify the IT admin.
* The system will provide password reset functionality via email verification.

### Functional Requirements

* The system shall allow users to register, log in, and reset passwords.
* The system shall allow students to schedule, modify, and cancel lesson appointments online.
* The system shall display available time slots, trainers, and vehicle availability.
* The system shall allow admins to manage and deactivate training packages.
* The system shall allow secretaries to schedule appointments by phone or in person.
* The system shall maintain an activity log for all reservation actions (create, cancel, modify).
* The system shall record student test results, track online class progress, and display trainer notes.
* The system shall import updates from the DMV regarding rule changes and practice test content.

### User Interface

* User Roles: Student, Secretary, Admin (Liam), IT Officer (Ian)
* Student Interface: Accessible via browser/mobile; schedule lessons, view progress, take tests, and reset password.
* Secretary Interface: Desktop-based form entry to create/update student profiles, schedule appointments.
* Admin/IT Interface: Manage users, generate reports, configure packages, monitor system logs.
* Interfaces must be responsive, simple to navigate, and intuitive, especially for mobile users.

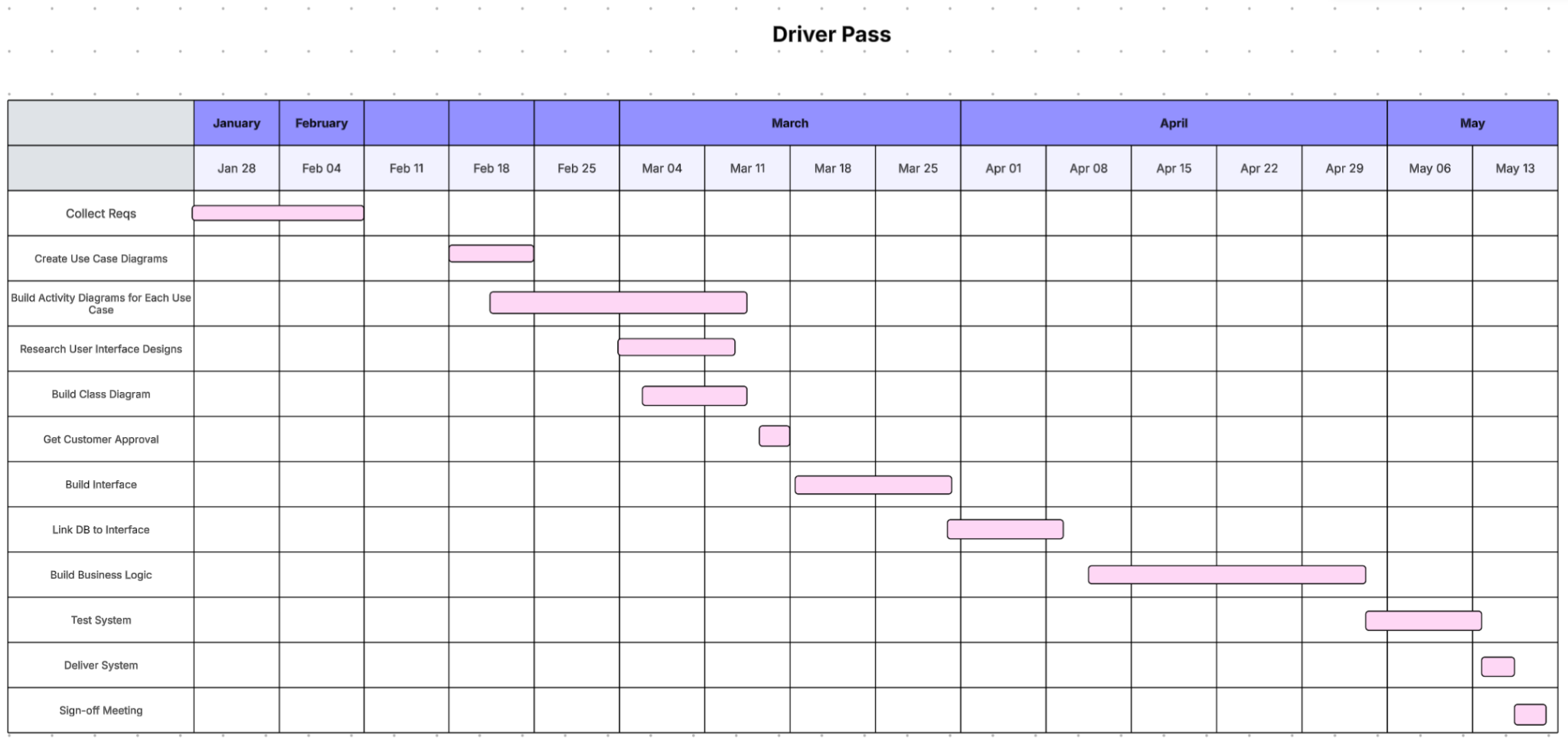
### Assumptions

* Students have access to a stable internet connection and modern browser.
* Credit card processing will be handled through a third-party provider.
* The DMV provides a public or authenticated API for updates, or alternatively sends update files periodically.
* The user base is expected to grow, so scalability is considered in the design.

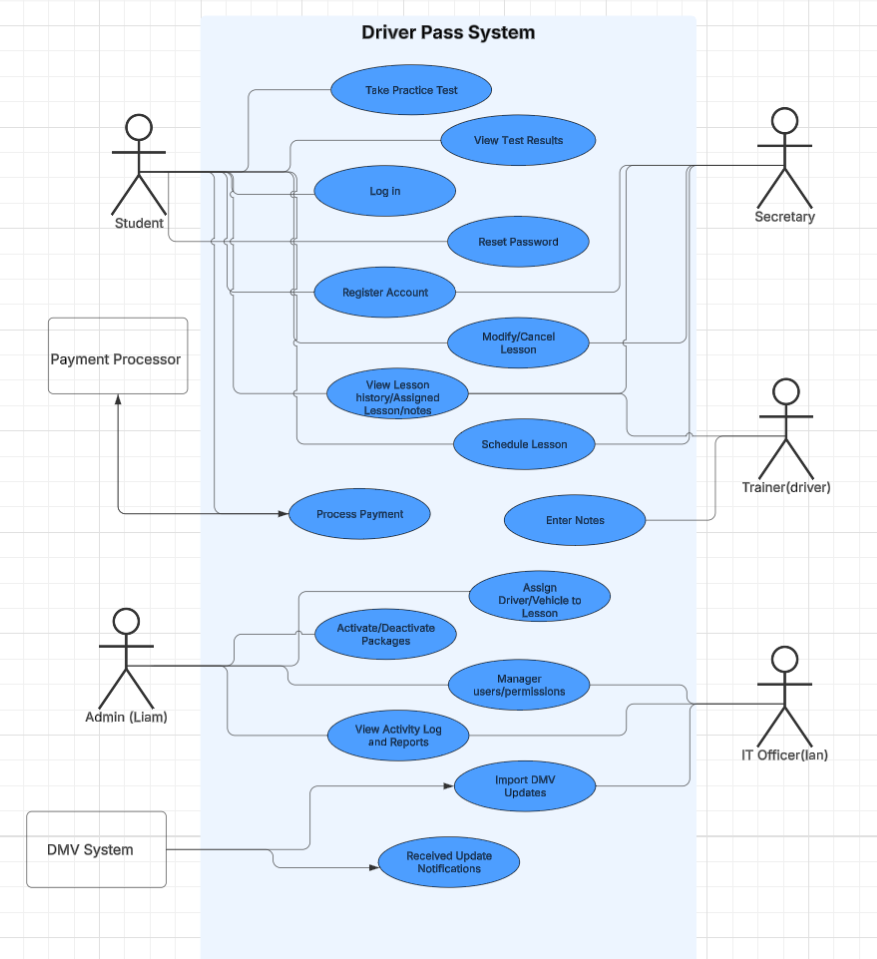
### Limitations

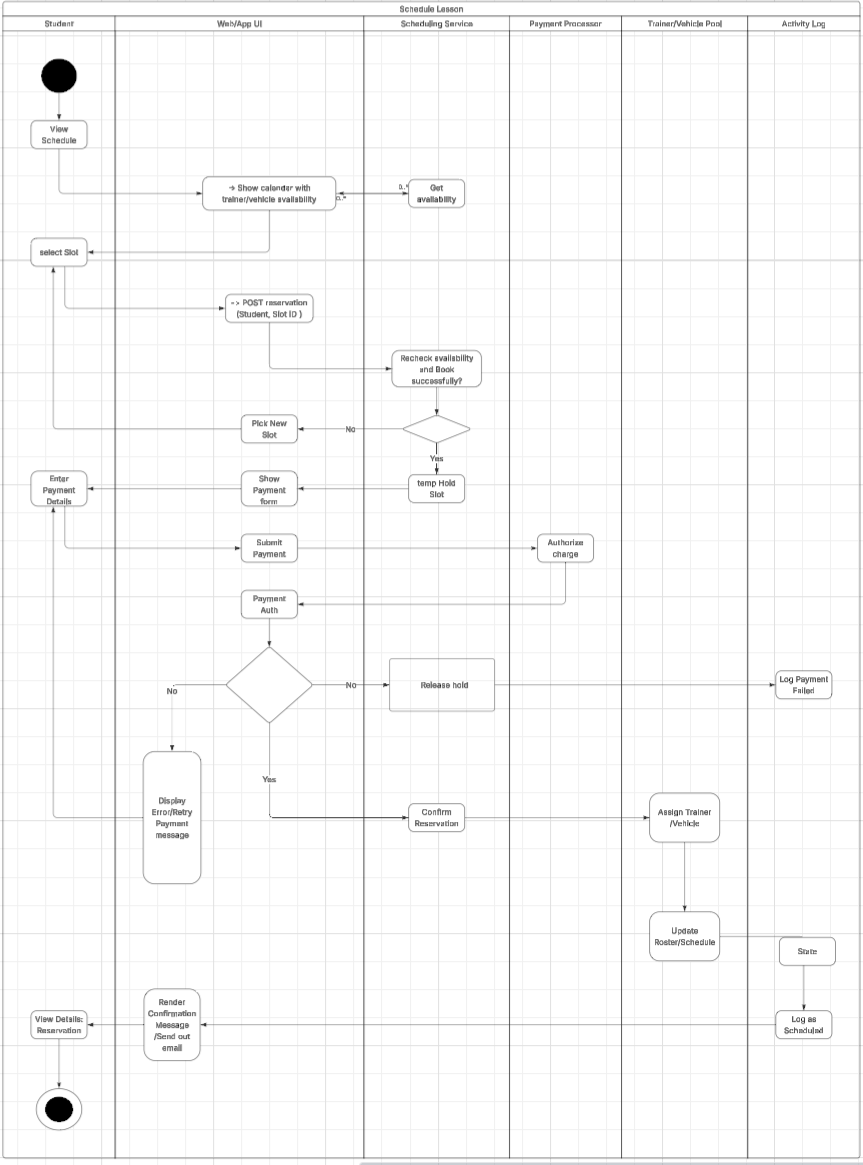
* Real-time DMV integration depends on external API availability and may be delayed or manual initially.
* The system cannot be modified by non-technical staff; changes to business logic require developer involvement.
* The initial release may not include mobile apps, but will support mobile browsers.
* Budget and timeline may limit the level of interface polish or future enhancements in the MVP.

### Gantt Chart

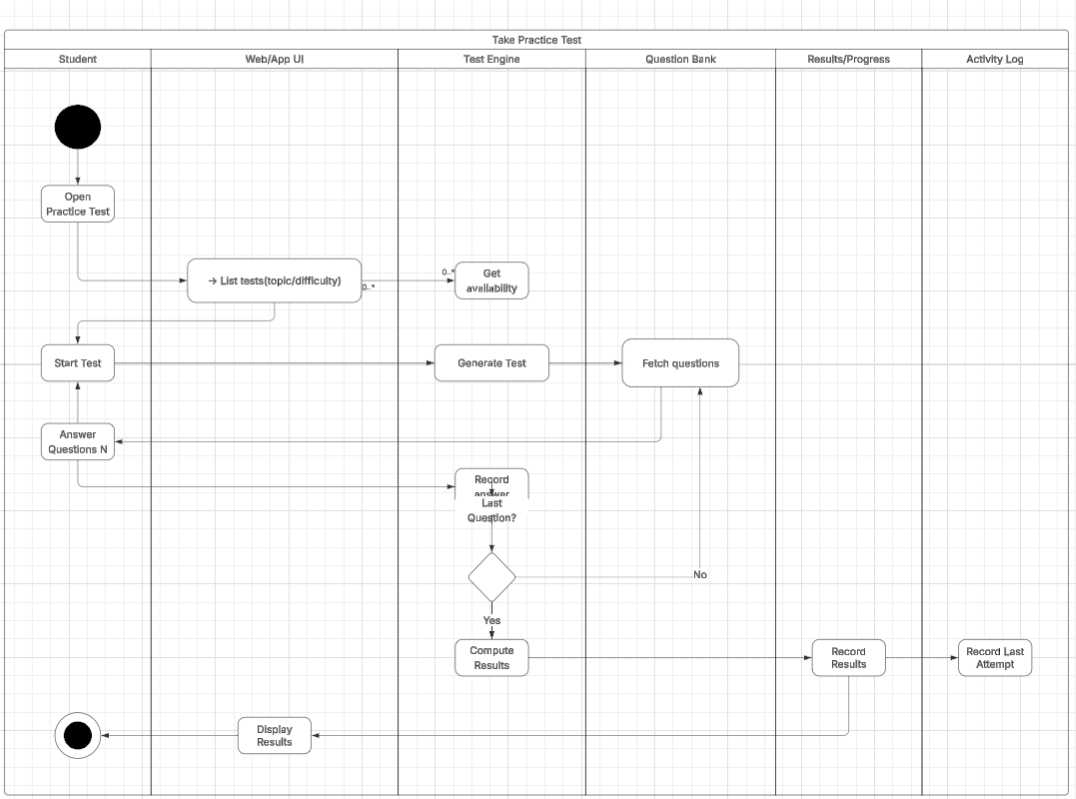
**

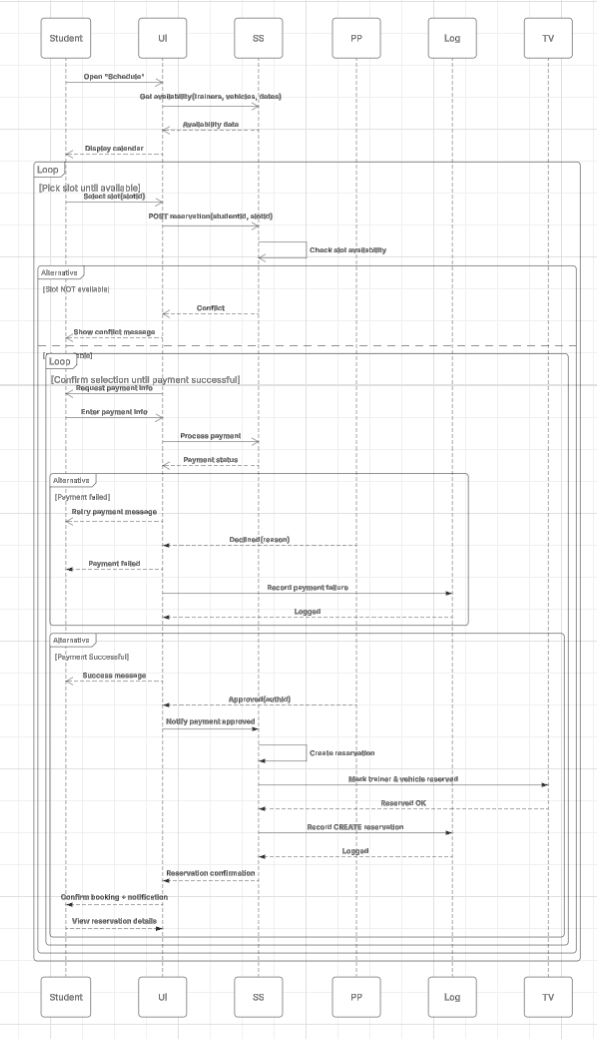
Use Case Diagram:

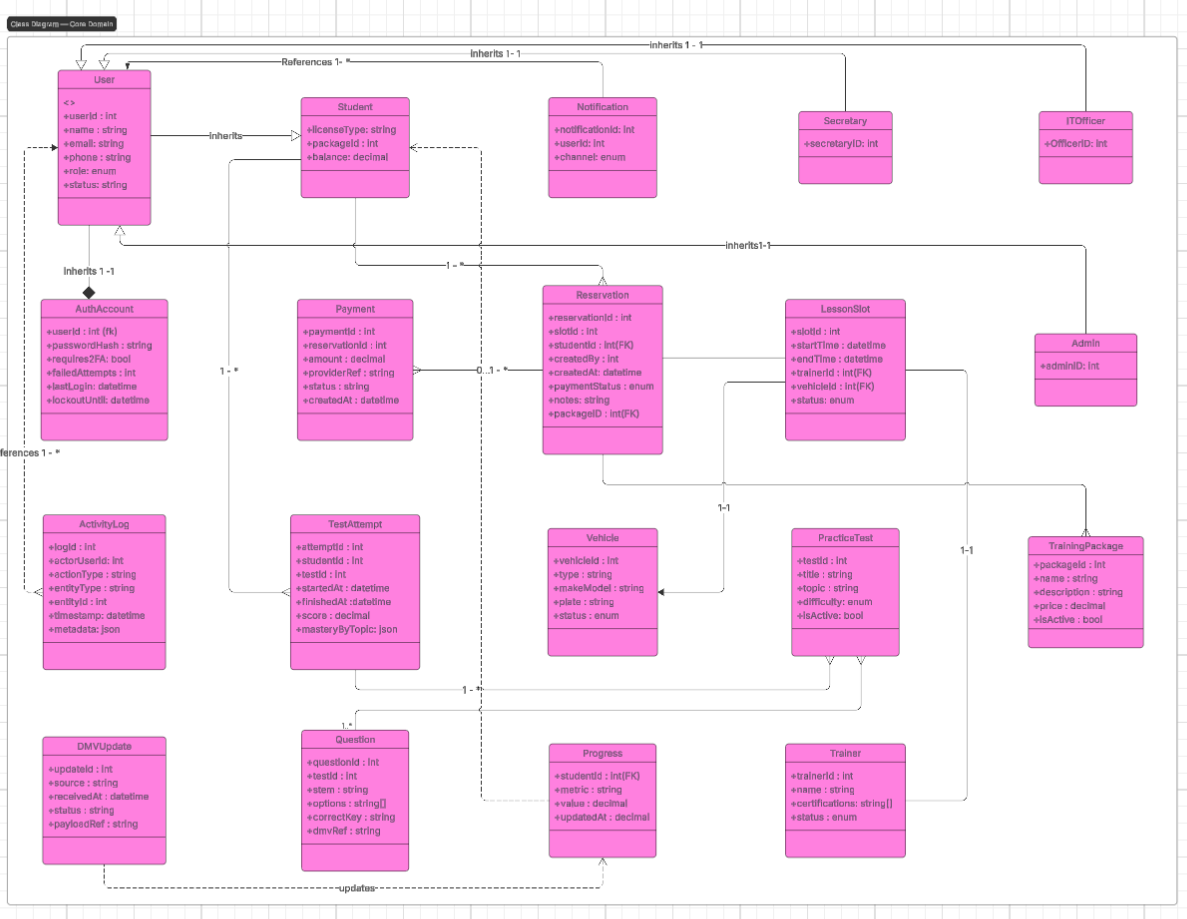


UML Activity Diagram: Schedule a lesson  
  


UML Activity Diagram : Take Practice Test



UML Sequence Diagram:  
  


UML sequence diagram:  
  


# Technical Requirements

## Platform & Infrastructure

* **Cloud-hosted, scalable web application** with RESTful APIs; supports Chrome/Firefox/Safari; responsive for mobile. *(Matches “cloud-based web app, modern browsers, mobile access.”)*
* **Relational database** (MySQL or PostgreSQL) for core data (users, reservations, tests, logs). *(You specified an RDBMS.)*
* **Daily backups** with restore verification; **planned maintenance** with <1 hr monthly downtime target.
* **Performance SLO:** common actions (login, scheduling, test submit) respond in ≤2 seconds under normal load.

## Security & Access

* **TLS/HTTPS** everywhere; **hashed passwords**; **optional 2FA** for staff; **RBAC** across Student/Secretary/Admin/IT roles.
* **Account lockout** after 5 failed attempts; **email-based password reset** with verification.
* **Activity logging** of reservation lifecycle and critical admin actions.

## Integrations

* **Payment processing** via a third-party provider (tokenization, authorization, capture, refunds).
* **DMV content updates** via API or periodic files; admins notified when rules/content change.

## Application Components

* **Front end:** HTML5/CSS/JavaScript SPA (or SSR), forms with client-side validation. *(Matches “standard web technologies, client/server validation.”)*
* **Back end:** REST API services for Auth, Scheduling, Payments, Tests, Admin/Packages, DMV Updates.
* **Background jobs:** nightly backups, DMV import processors, reminder notifications.

## Tools

* **Issue tracking & CI/CD** consistent with modular, future-extensible architecture in the BRD. *(Modularity & adaptability)*