# CS 255 Business Requirements Document - DriverPass

## System Components and Design

### Purpose

*What is the purpose of this project? Who is the client and what do they want their system to be able to do?*

* Client: DriverPass
* Objective: The DriverPass company aims to create an online Learning Management system that is specifically targeted at improving testing and passing scores on the local driving test by offering curated courses and hands on training to enrolled students. The students may also self-curate their learning plans, while having direct communications and feedback with their course instructors.

### System Background

*What does DriverPass want the system to do? What is the problem they want to fix? What are the different components needed for this system?*

* The driver for this development effort is there is a market opportunity to improve driver education and DMV passing percentages thru offer a personalized training program that includes both online classroom type training as well as on the road training.
* The system developed will allow the client to connect directly with the DMV to ensure the most accurate rules and testing materials are being utilized
* System Components:
  + The basic principle architectural model of the application revolves around a customized Learning Management System.
  + The LMS will be web-based UI, with limited offline reporting functionality.
  + The LMS will utilize a subscription service handler that allows student to manage their tier of enrollment thru upgrades to access additional content, or purchase hands-on training sessions
  + The subscription service intrinsically also establishes the need for a cart handler component to allow selection of the desired tier and handle payments and digital content enablement.
  + For the hands-on training, a calendar and resource reservation management component are required to support scheduling together a driver, a car, and an instructor, and this together will form a session. Each session much have options for bi-directional communications between driver and associated students.
  + Users will be able to perform self-service, or alternatively work with a proxy agent for all interactions with the system.
  + Provide monitoring of changing rules and conditions at the DMV and allowing content changes in response to changes at the DMV.
  + Dash board systems will be utilized to implement the companies desired user landing page which summarizes key information in a compact format.

### Objectives and Goals

*What should this system be able to do when it is completed? What measurable tasks need to be included in the system design to achieve this?*

*Client access*

* Provide access to learning content thru a Learning Management system interface (subscription restricted). Self and instructor assigned options
* Report generation for data analysis
* Associate drivers, customers and vehicles to calendar slots
* Subscription management – assign and manage user roles
* Feedback - Driver to customer, customer to driver
* Proxy administration (secretary or higher role)
* Business side dashboard, booking percentages, driver ratings, DMV pass rates, etc.

*Customer access*

* Self-scheduling – in car and online assets
* *Account self-management*
* Test administration – Practice exams administration
* User Dashboard

## Requirements

### Nonfunctional Requirements

#### Performance Requirements

*What environments (web-based, application, etc.) does this system need to run in? How fast should the system run? How often should the system be updated?*

* System runs on a web-based UI interface
* System is hosted on a cloud infrastructure
* Cloud provider responsible for maintenance and security
* System to be considered “fast responsive” relative to the current state of web technology, but not real-time
* Content updates available within 10 minutes from data entry unless restricted for release approvals

#### Platform Constraints

*What platforms (Windows, Unix, etc.) should the system run on? Does the back end require any tools, such as a database, to support this application?*

* The system shall be designed to be Windows compatible (frontend)
* The system shall be designed to be Mobile phone compatible (Android and Apple) (frontend)
* The system shall be Linux based (backend)
* The system shall utilize transaction level authentication (all client <-> server, database changes, database reads) to ensure user access restrictions are adhered to.
* The system shall implement a REST API compliant interface

#### Accuracy and Precision

*How will you distinguish between different users?* *Is the input case-sensitive? When should the system inform the admin of a problem?*

* The system shall utilize a traceable database scheme that is logged, replayable
* The system shall utilize databasing architecture with redundancy
* Each user shall be given a unique ID that shall be used for asset management, account details, logging, privilege escalation, etc.
* Each asset (car, appointment slot, etc.) shall have a unique ID used for linking assets together
* Inputs for user name and passwords shall case sensitive
* Account access shall use 2 factor Authentication when logging in.
* The system shall be Interlinked with DMV servers for content updates
* The system shall trigger a notification when DMV data changes
* Scheduling interface shall prevent double bookings thru asset tracking
  + Detection of booking collisions shall alert the users and admins for corrections
  + Server-side redundancy scripts shall be used for detecting non-UI induced double bookings (such as direct database edits)
    - Scripts shall run hourly (or more) and scan forward 1-week minimum duration
* System shall have an independently hosted watchdog to detect lockups or other issues
  + Watchdog shall have an alert system that pushes notification to admin/IT roles

#### Adaptability

*Can you make changes to the user (add/remove/modify) without changing code? How will the system adapt to platform updates? What type of access does the IT admin need?*

* Features shall be based on a plug-in model where they can be added, removed, etc. without impacting the core functionality.
* User privilege escalation shall be managed thru the platform frontend
* Service updates (module level) shall be updatable via a live switchover (downtime less than 30 seconds)
* Service updates (system level) shall be updatable via a live switchover (downtime less than 30 seconds)
* Architecture shall be designed as suitable for scalability including multiple redundant backend servers including databasing.
* The system shall support localization in the form of DMV content that is user area specific.

#### Security

*What is required for the user to log in? How can you secure the connection or the data exchange between the client and the server? What should happen to the account if there is a “brute force” hacking attempt? What happens if the user forgets their password?*

* <Internal Concern> – Owner wants to be able to access the data offline in unrestricted (excel type) data format – Need to have a way to make data sanitized when prepared for offline usage
* User Roles specific access
  + Admin [unrestricted]
  + IT officer [unrestricted]
  + Secretary [make appointments, data entry, contact student]
  + Customer [make/cancel/modify appointments online, reset password]
  + Drivers [add student notes]
* Users may recover lost user name and passwords, reset 2FA will be mandated upon recovery
* All API transactions shall be tracked to the user [if logged in] with date, time, IP logging. Logs will be maintained for 2 years or as required by other constraints., whichever is longer.
* All sessions shall automatically time-out after 10 minutes of inactivity.
* Brute-force log-in detection (5 failed attempts) shall block the offending IP address and user account in an escalating duration of 0.25, 1, 6, 24 hours increments.
* API interface shall be hardened and tested to industry standards for unauthorized access and script injection, man-in-the middle detection, and etc based on best practices.
* All user information shall utilize encryption at all layers of transit and storage, with decryption keys using the unique user ID and a unique seed key per dataset.
  + No keys shall be stored in plain text format, but may be stored to a specific user
* Credit card information shall retain the card number and associated address only, requiring the user to enter the expiration and CVC code each transaction.

### Functional Requirements

*Using the information from the scenario, think about the different functions the system needs to provide. Each of your bullets should start with “The system shall . . .” For example, one functional requirement might be, “The system shall validate user credentials when logging in.”*

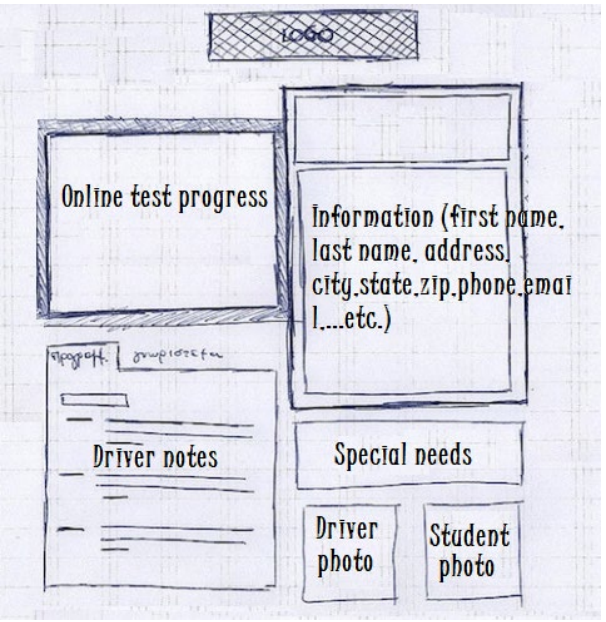
* The system shall validate user credentials when logging in
* The system shall validate the user privilege escalations prior to executing any api transaction
* The system shall allow authorized users to review all subscribed course materials
* The system shall allow non-customer roles to communicate with a single customer
* The system shall allow non-customer roles to communicate with all current customers (context – communicate holidays, other misc. closures, etc. without having to send to induvial customers 1 at a time)
* The system shall review local DMV content daily during early morning hours and alert to updated content if found.
* The system shall associate local DMV content to each user respectively (context - support localization of rules, tests, etc.)
* The system shall utilize local time zones for all user facing interactions, using GMT for all stored data including logging.
* The system shall enforce drivers to initialize the appointment with applicable vehicle checks, and log all relevant appointment data during the course of the day.
* The system shall allow proxy usage for scheduling and initial data entry only.
* The system shall utilize a modular framework that allows extensions to be added/updated/removed/etc. in real-time.
* The system shall utilize a live switch-over for system level updates.
* The system shall provide a double-booking prevention mechanism that actively monitors for double bookings for each appointment slot.
* The system shall provide security procedures/methods as outlined in the Security section of this document.
* The system shall require an approved photo to be loaded to each profile before being allowed to schedule an appointment as either a driver or as a customer.
* The system shall allow individual cars to be marked as out of service.
  + Before a car can be marked as out of service, all scheduled appointment slots must be transferred to another appointment option.
* The system shall sanitize all downloaded offline data to make it not personally identifiable (name/address/etc). Unique traceability can be maintained thru unique user IDs.

### User Interface

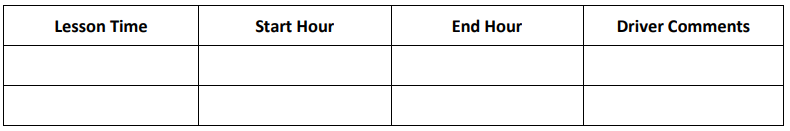
*What are the needs of the interface? Who are the different users for this interface? What will each user need to be able to do through the interface? How will the user interact with the interface (mobile, browser, etc.)?*

*The system shall implement the following UI features and data collections.*

* Web UI
  + Pages to Build
    - Summary page (default page upon customer login) [login required]



* + - Driver Feedback Sheet



* + - Test Progress Sheet
      * Time Taken, score, status [not taken, in progress, failed, passed]
    - Data input form (aka new user registration) [no login required]
      * First, Last name
      * Address
      * Phone
      * State
      * CC number + expiry + security code
      * Preferred pickup location (drop down list?)
      * Preferred Dropoff location (drop down list?)
    - Contact us [Requires login]
    - User management [Admin/IT officer only restricted access]
    - Contact student [Secretary/Driver/IT/Admin only]
    - Password Reset [no login required]
    - Appointment management [Driver/IT/Admin restricted access] [login required]
      * Car details
        + Condition checked out
        + Condition checked in
      * Driver details
      * Customer details
      * General appointment notes
        + Pickup time
        + Pickup location
        + Dropoff time
        + Dropoff location
    - Package Purchase cart [login required]
      * Purchase packages
        + Option 1 -Six hours in car
        + Option 2 – Eight hours in a car + In person lesson
        + Option 3 – Twelve hours in car + In person lesson + Online Class Library access + Online practice tests
        + Add/edit/remove package options [admin roles only] – Extension package later
        + Disable sales of a given package
* Appointment Scheduling [login required]
  + 10 cars
    - 1 driver per car
    - <TBD> students per car
  + 2 hour blocks, scheduled online
  + Schedule slots – include
    - time (self/secretary)
    - location (self/secretary)
    - Driver (self/secretary)
    - Car (company only)
* Reports
  + Car history report [Driver/IT/Admin restricted access] [login required]
  + User history report [IT/Admin restricted access] [login required]
  + Driver history report [IT/Admin restricted access] [login required]
  + <consider> Direct Database editing (still logged!) [IT/Admin restricted access] [login required]
* Raw Data Download(excel) < see security concerns> - Needs sanitized data output

### Assumptions

*What things were not specifically addressed in your design above? What assumptions are you making in your design about the users or the technology they have?*

* American English is the default language, with localizations being out of scope for the initial development efforts.
* User can produce a list of all scheduled appointments from the report generator
* Drivers can produce a list of all scheduled appointments from the report generator
* customer can self-subscribe to all course work and training
* customer can self-update payment information
* customer buy-up will automatically upgrade subscription level
* Customers can self-delete their accounts, relevant ID and customer names will remain for record keeping.
* Customer re-enrolling will require a new unique ID once an account has been deleted.
* Users will be required to confirm initial account creation via an automated email sent to the email address provided.
* Engineering staff can supersede the requirements in this document with written authorization from the client owner or IT contacts.
* Security protocols will outrank user interface conveniences/requirements.
* Exported data will be limited to a maximum of 30 days span.

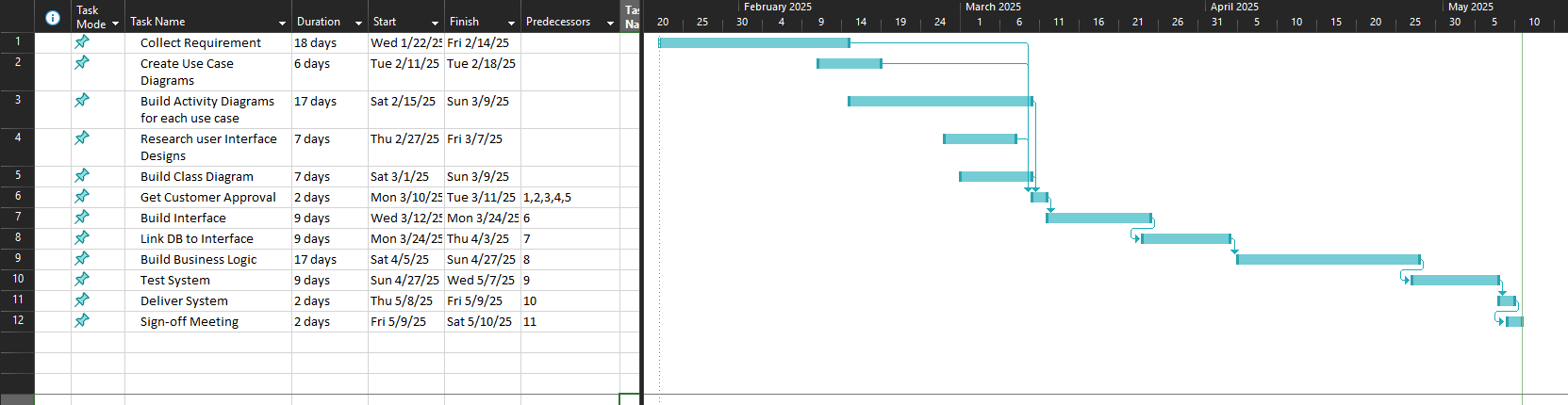
### Limitations

*Any system you build will naturally have limitations. What limitations do you see in your system design? What limitations do you have as far as resources, time, budget, or technology?*

* Data editing shall be only available while actively connected and logged into the server.
* Servers and services shall only be designed for USA workflows
* ADA services excepting closed captions, shall not be part of the initial deployment as the drivers testing are required to have reasonably corrected sight.
* Mobile device services shall utilize the default device browser adaptations initially. These device types shall be restricted to the latest Samsung and Apple models with other manufacturers and phone models considered as a non-required added compatibility.

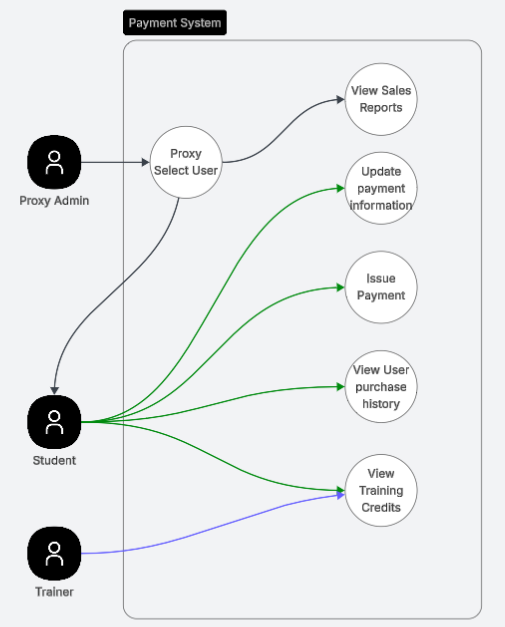
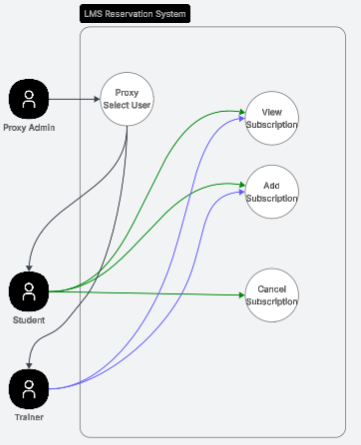
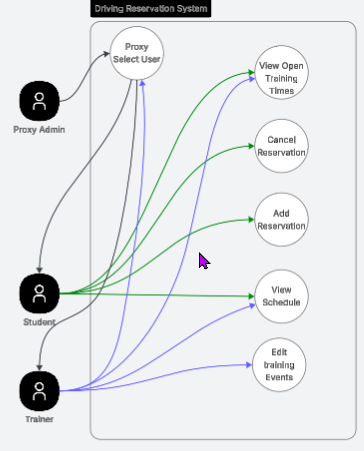
**Gantt Chart**

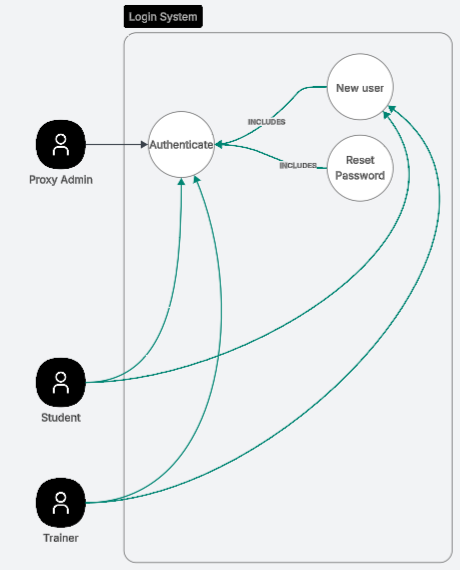
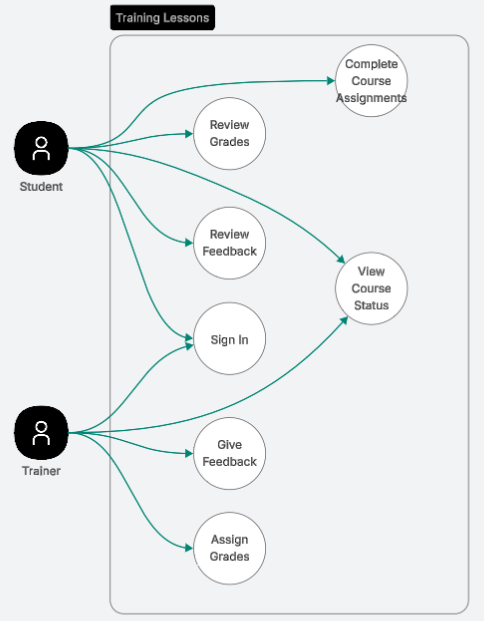
*Please include a screenshot of the GANTT chart that you created. Be sure to check that it meets the plan described by the characters in the interview.*



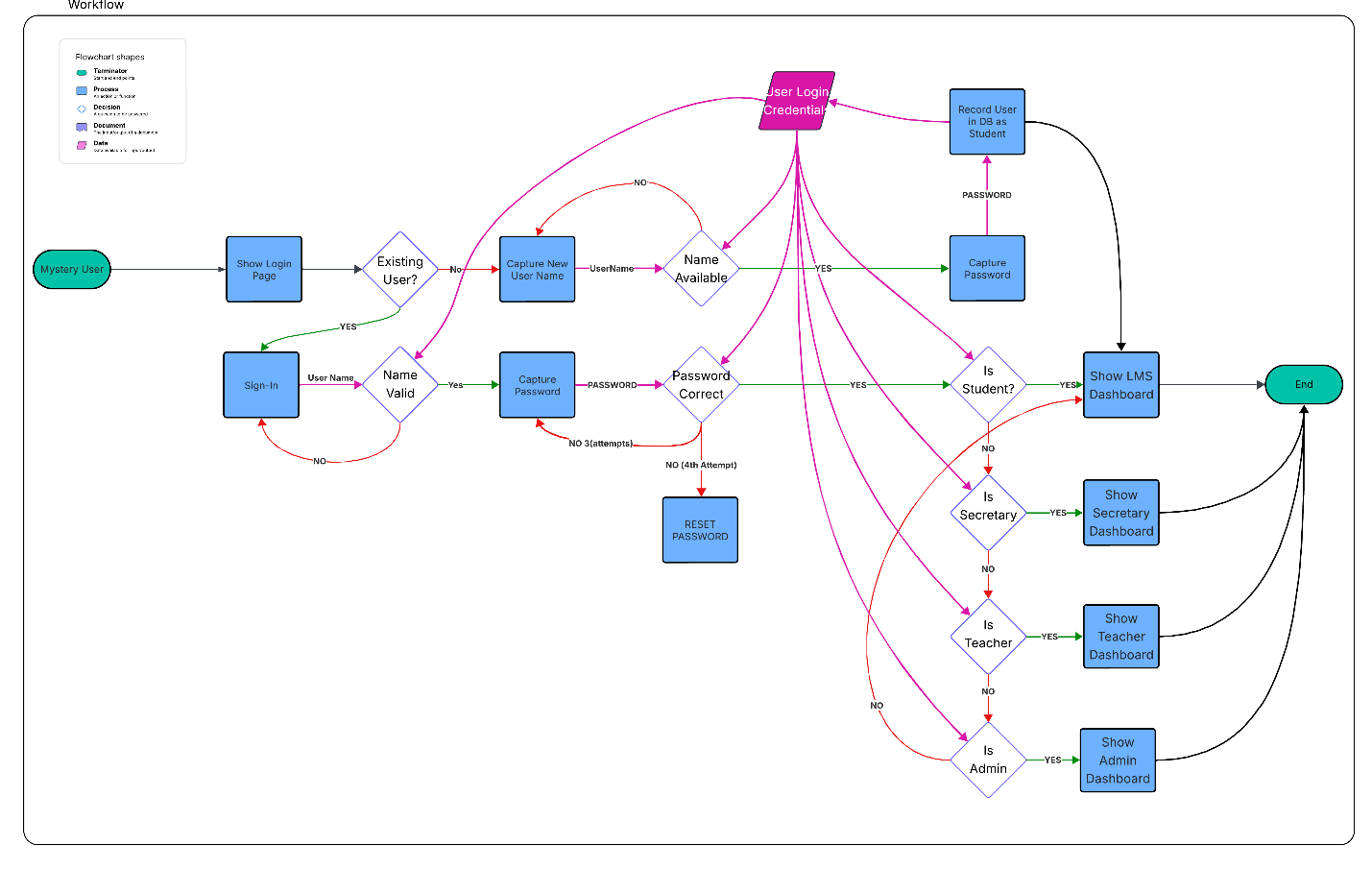
**UML Use Case Diagram**

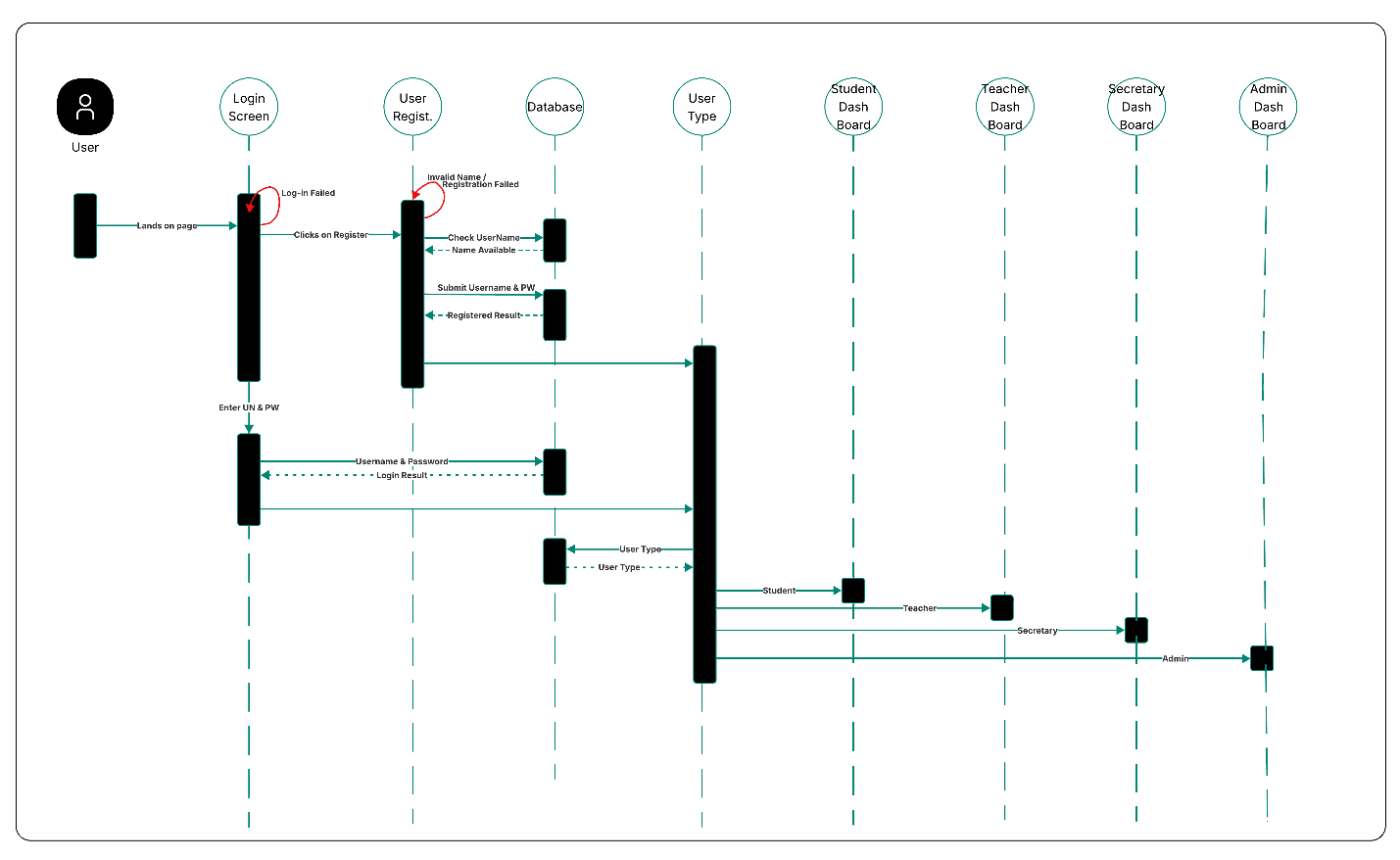
The use cases were broken down into key functions to aid in understanding as follows.

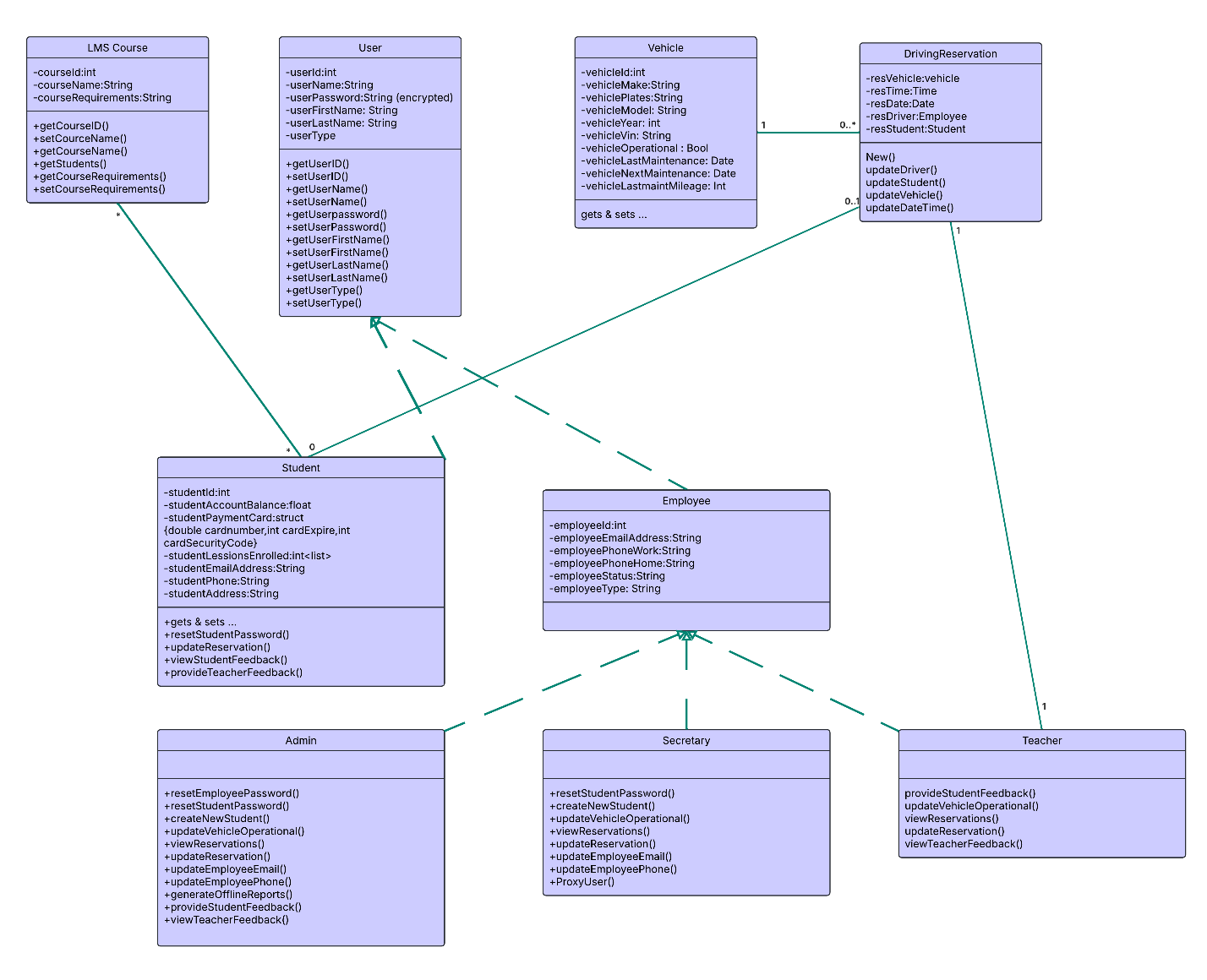
 

**UML Activity Diagram**

For the UML Activity Diagram, the user Login page is detailed here, which includes both unknown users (diagram 1) and known users (Diagram 2) with all account types ****

**UML Sequence Diagram**

**UML CLASS Diagram**



Technical Requirements

* Hardware
  + Cloud Server with USA priority services
    - Linux servers
  + User Access device
    - mobile or PC platform
* Software
  + Linux Server
    - REST interface
    - Transaction based user authentication
  + Database package
    - widely adopted
    - regular security patches
  + LMS tool
  + LMS content
  + Security firewall
    - Widely adopted
    - Regular security patches
* Tools
  + Laptops for developers
  + IDEs with licenses
  + Git repository
  + JIRA bug trackers
  + Open Source Libraries
* Misc
  + Client sample Dataset
  + Development Schedule
  + Development diagrams/charts