# CS 255 Business Requirements Document

## 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?*

* DriverPass wants to revolutionize the way people prepare for their driving tests by offering a hybrid solution: online classes, practice exams, and in person training
* The purpose of this project is to develop a centralized, secure, and cloud based system that allows DriverPass staff and clients to easily manage lessons, monitor progress, and adapt to DMV updates
* The system should let users register, schedule or cancel lessons, access materials, and track progress across packages online and on the road
* It also needs to allow the admin team to manage user accounts, track activity logs, generate reports, and sync with DMV updates

### 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?*

* DriverPass identified a gap in the market: too many people fail their DMV tests due to lack of preparation
* Their solution is to combine online tools with in person lessons
* The system needs to support various users owner, IT admin, secretary, instructors, and students with different access rights
* Core components include:
  + User account management (sign up, password recovery, role based access)
  + Online lesson and test progress tracking
  + Appointment scheduling and matching (student, driver, vehicle)
  + DMV integration for updates
  + Administrative dashboard with activity logs and reporting
  + Flexible package offerings with the ability to disable them
  + Objectives and Goals

### 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?*

* The system should allow:
  + Students to register, book/cancel/modify lessons, and track their progress online
  + Staff (admin/secretary) to input and view client data, schedule lessons, and assign drivers
  + Instructors to log comments, lesson times, and review progress
  + The business to view system usage, monitor changes, and generate printable reports
  + Automatic password recovery and role based access control
  + Real time syncing with DMV updates (or receive notifications)
* Goals must be measurable:
  + A student should be able to complete registration in under 5 minutes
  + A secretary should be able to schedule a lesson in 3 steps or fewer
  + Activity logs should clearly show the who/what/when of all modifications
  + All systems should work on mobile and desktop via a browser

## Requirements

### Nonfunctional Requirements

*In this section, you will detail the different nonfunctional requirements for the DriverPass system. You will need to think about the different things that the system needs to function properly.*

#### 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?*

* The system needs to run entirely web based and feel fast no one should be sitting there watching a loading bar crawl just to reschedule a lesson
* It should be optimized for mobile and desktop browsers, especially since users might be accessing it from home, work, or on the go
* Updates to lesson content, DMV policy syncs, and security patches should happen automatically or with minimal disruption, I'd expect at least monthly backend maintenance and real time updates when syncing DMV materials

#### 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?*

* This system needs to run cross platform Windows, macOS, and mobile browsers at the very least
* The backend should run securely on a cloud platform (like AWS or Azure) and rely on a scalable relational database (SQL based) to manage users, appointments, lessons, and package data
* No local install should be required, everything should be accessible via browser, no matter the OS

#### 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?*

* Each user will be distinguished by a unique login (email + password combo), and roles will determine what they see and what they can do
* Input fields should be case sensitive when it matters (like passwords) and forgiving when it doesn't (like names or addresses)
* Any failed login attempts, account lockouts, or mismatched lesson data should flag the admin or IT role immediately through the dashboard

#### 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?*

* User management (like adding or deactivating users) needs to happen through an admin panel no touching the code just to fire someone or add a new trainer
* The system should be loosely coupled so platform updates don’t break core features. Think flexible, not fragile
* Ian, the IT guy, should have god mode access: full control over resetting accounts, disabling packages, and updating content when needed

#### 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?*

* Login should require strong passwords, and users should confirm with email verification. Optional 2FA would be a huge plus
* Data exchanged between the user and the server must be encrypted, SSL/TLS all the way
* If someone brute forces login attempts, the account should autolock after a set number of tries, and notify admin and user immediately
* Forgot your password? No problem a secure, time sensitive reset link should be emailed, with CAPTCHA verification to block bots

### 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 allow customers to book, modify, or cancel driving lessons through their account
* The system shall allow secretaries to create appointments on behalf of customers over the phone or in person
* The system shall match a customer with a car, a driver, and a time slot based on availability
* The system shall display customer progress across lessons, including test status (passed, failed, in progress)
* The system shall allow administrators to view and print activity logs
* The system shall notify users and staff of upcoming appointments and changes
* The system shall disable registration for a training package if deactivated by admin
* The system shall allow the DMV to push updates to the lesson content or test questions
* The system shall store lesson notes, time logs, and instructor comments under each student's record

### 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 interface should be clean, intuitive, and responsive.. nobody wants to zoom in or click five times just to schedule a ride
* There are a few user types: students, secretaries, instructors, admin/IT, and the owner
* Students need access to booking, package selection, progress tracking, and profile editing
* Secretaries need a quick interface for managing customer info and scheduling
* Instructors need a space to log time, notes, and see their upcoming sessions
* Admin needs full oversight, including user management, logs, and system controls
* Interaction will happen through web browsers on both computers and mobile phones think cloud first, device flexible

### 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?*

* I'm assuming users have stable internet access and basic computer skills.. enough to book lessons and reset passwords online
* I’m also assuming our team will build the system with mobile... first design in mind, given how often people rely on phones to manage appointments
* The DMV will provide an API or portal we can connect to for their rule updates, even if it’s manual entry at first
* User behavior will follow expected patterns (no one is trying to book 12 lessons at once on accident)

### 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?*

* This system won’t support offline editing of lesson data, it needs a web connection to sync everything properly
* Budget and time constraints mean we probably can’t build a full drag and drop interface or allow full non dev customization of lesson packages.. some updates will still need a developer
* The business logic layer will need extra testing to avoid bugs in appointment conflicts or tracking mismatches
* We are relying on third party tools (cloud hosting, email services) that may have their own outages or limits outside our control

### Gantt Chart

