# CS 255 Business Requirements Document Template

## 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?*

* The purpose of this project is to design a system for DriverPass that provides students with tools to prepare for their driving tests effectively.
* DriverPass, the client, wants the system to offer online practice exams, on-the-road training scheduling, and progress tracking to improve student pass rates.

### 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 aims to address the issue of students failing their driving tests due to inadequate preparation tools.
* The system should provide online classes, practice exams, and flexible scheduling for driving lessons.
* Components include user accounts, practice test modules, scheduling tools, reporting tools, and administrative interfaces.

### 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 take online practice exams and track their progress.
* The system must enable scheduling of on-the-road training sessions with assigned drivers and cars.
* The system should provide administrative tools for tracking reservations, managing users, and handling updates.
* Security and data integrity should be ensured for user and business information.

## 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?*

* The system must operate on web-based platforms and mobile devices.
* It should respond to user actions within 2 seconds on average.
* System updates should occur monthly to ensure functionality and security.

#### 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 should run on major operating systems, including Windows, macOS, iOS, and Android.
* Backend support must include a secure, cloud-based database for user and transaction data storage.

#### 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?*

* User accounts must be distinguished by unique email addresses.
* Passwords are case-sensitive to enhance security.
* System administrators must be notified immediately of any errors or failed transactions.

#### 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?*

* IT administrators should be able to add, remove, or modify user roles without altering core system code.
* The system must accommodate platform updates with minimal downtime.
* The interface must be adaptable for both desktop and mobile devices.

#### 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?*

* Users must log in using a unique email and secure password.
* All data exchanges between the client and server must be encrypted using SSL.
* After three failed login attempts, user accounts should lock temporarily, requiring manual or automated password reset.
* Users must have the ability to reset passwords securely through email authentication.

### 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 upon login.
* The system shall allow users to schedule, modify, and cancel driving lessons.
* The system shall enable students to take practice exams and receive feedback.
* The system shall generate activity reports for administrators to track system usage and performance.
* The system shall integrate with DMV updates to ensure test relevance.

### 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 must provide intuitive navigation for users of varying technical skills.
* Students must be able to access practice exams, view progress, and schedule lessons through a web browser or mobile app.
* Administrators need access to manage user accounts, view system logs, and update content.
* Drivers must have access to schedules and customer notes via a secure interface.

### 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?*

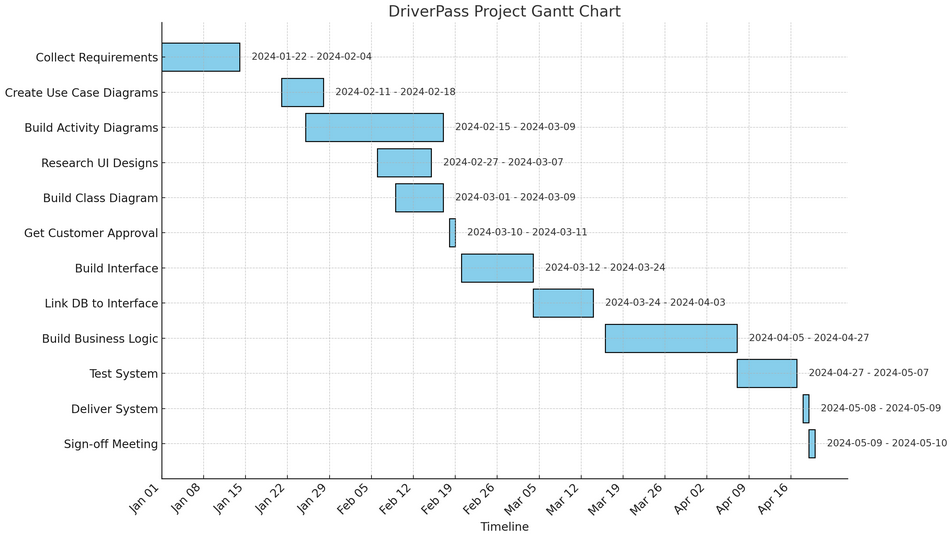
* Users will have access to devices with internet connectivity to use the system.
* DriverPass will provide the necessary resources to train staff on system operations.
* Customers will follow the password and security protocols to maintain account integrity.

### 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?*

* The system depends on reliable third-party integrations, such as payment gateways and DMV updates.
* Budget constraints may limit the extent of features included in the initial version.
* Time constraints may impact the ability to test all scenarios comprehensively.

### Gantt Chart

**