# CS 255 Business Requirements Document Thompson

## System Components and Design

### Purpose

- Design and implement a system to enhance driver training for DriverPass.

- Improve the success rate of individuals taking driving tests.

Client:

- Liam, owner of DriverPass.

Client's Objectives:

- Provide Online Training:

- Online classes and practice tests.

- Road Training:

- Practical driving skills enhancement through on-the-road training.

Accessibility:

- Enable access to training materials:

- Anywhere (online and offline).

Reservation System:

- Implement a system for scheduling driving lessons:

- Customer preferences for day, time, and specific trainers.

User Management:

- Different User Types:

- Liam (Owner), Ian (IT Officer), Secretary, and Students.

- User-Friendly Interfaces:

- Registration, appointment scheduling, and resource access for students.

Security and Tracking:

- Data Security:

- Varying access rights for different user roles.

- Tracking Features:

- Monitor changes made by users.

- Maintain an activity log.

Integration with DMV:

- Establish a connection with the DMV:

- Receive updates on rules, policies, and sample questions.

- Ensure training content remains up-to-date and compliant.

Future Flexibility:

- Customize Training Packages:

- Allow Liam to modify and customize training packages.

- Disable Specific Packages:

- Ability to deactivate specific packages as needed.

Overall Goal:

- Create a comprehensive system:

- Improve the quality of driver training.

- Reduce the failure rate in driving tests.

- Provide efficient experience for both clients and the DriverPass team.

### System Background

What DriverPass Wants the System to Do (based on the interview transcript):

Training Services:

- Provide online classes and practice tests for driver training.

- Offer on-the-road training for customers who desire practical experience.

Data Accessibility:

- Allow access to data both online and offline.

- Enable the download of reports and information for offline use.

Security:

- Implement a secure system with different employee roles and varying access rights.

- Include the ability to reset passwords and block access, particularly for IT personnel.

Tracking:

- Implement detailed tracking for reservations, modifications, and activity.

- Clearly identify users responsible for changes to maintain accountability.

Reservations:

- Facilitate online and offline scheduling of driving lessons.

- Define three packages with varying hours and additional features.

- Ensure identification of matched drivers, times, and cars for each reservation.

User Types:

- Accommodate different user roles: Owner (management), IT Officer (maintenance and modification), Secretary (appointment scheduling), and Customers (online appointment management).

Registration Process:

- Simplify customer registration by providing personal information and credit card details.

- Include pickup and drop-off locations in the registration process.

Compliance:

- Stay informed and up to date with changes from the Department of Motor Vehicles (DMV).

- Receive notifications for any updates or changes in DMV rules, policies, or sample questions.

Interface:

- Develop a web-based system, preferably on the cloud.

- Design a user-friendly interface based on a specific sketch provided by Liam.

Future Features:

- Initially focus on current requirements; future features are not clearly defined.

The Problem They Want to Fix:

- Address the need for better driving training, recognizing a gap in the market where many individuals fail their driving tests at the Department of Motor Vehicles (DMV).

Different Components Needed for the System:

Online Training Platform:

- Module for online classes and practice tests.

On-the-Road Training Module:

- Integration for scheduling practical driving lessons.

Data Management System:

- System capable of handling data both online and offline.

- Ability to download reports and information.

Security Module:

- Role-based access control system with secure password management.

Tracking System:

- Activity tracking for reservations, modifications, and user actions.

Reservation System:

- Module for customers to schedule driving lessons online or offline.

- Three packages with varying hours and features.

User Management Module:

- Management of different user types with specific roles and permissions.

Registration System:

- Interface for customer registration, including personal information and credit card details.

Compliance Monitoring

- Connection to the DMV to receive updates regarding regulations, policies, and sample questions.

Interface Design:

- Web-based interface design based on the provided sketch.

Future Expansion Capability:

- Flexibility for future features and adjustments to packages or system modules.

### Objectives and Goals

Overall Objective:

- Improve driver training to boost the success rates of individuals undergoing driving tests.

Problem to Solve:

- High failure rate (more than 65%) among driving test applicants.

- Inadequate tools for effective driving training.

System Functionality:

Online Training:

- Provide online classes.

- Offer practice tests for better preparation.

On-the-Road Training:

- Facilitate hands-on, on-the-road training sessions.

Accessibility:

- Access to training materials from anywhere:

- Online and offline capabilities.

Reservation System:

- Enable customers to schedule driving lessons:

- Specify day, time, and preferred trainer.

- Online scheduling or office visits.

User Management:

- Different User Types:

- Liam (Owner), Ian (IT Officer), Secretary, and Students.

- User-Friendly Interfaces:

- Registration, appointment scheduling, and resource access for students.

Security and Tracking:

- Data Security:

- Different access rights for different users.

- Tracking Features:

- Monitor changes made by users.

- Maintain an activity log.

Integration with DMV:

- Establish a connection with the DMV:

- Receive updates on rules, policies, and sample questions.

Future Flexibility:

- Customize Training Packages:

- Allow Liam to modify and customize training packages.

- Disable Specific Packages:

- Ability to deactivate specific packages as needed.

Components Needed:

Online Training Module:

- Classes and practice test features.

On-the-Road Training Module:

- Practical driving sessions.

- Reservation System:

- Interface for scheduling lessons.

- User Management System:

- Different user roles and interfaces.

Security and Tracking System:

- Access control and activity tracking.

Integration Module:

- Connectivity with DMV for updates.

Flexibility Module:

- Customization of training packages.

Overall System Interface:

- User-friendly interface for seamless interactions.

## Requirements

### Nonfunctional Requirements

#### Performance Requirements

System Environments:

Web-Based:

- The system should run on a web-based environment to ensure accessibility from different devices and locations.

- Utilize cloud infrastructure for flexibility and scalability.

Offline Access:

- Consider providing offline access for users, allowing them to access training materials even without an internet connection.

Cross-Device Compatibility:

- Ensure the system's compatibility spans across a variety of devices, including both computers and mobile devices.

Performance Requirements:

Speed of the System:

- The system should provide a responsive and efficient user experience.

- Online classes and practice tests should load promptly.

- Real-time scheduling and updates for driving lessons.

Transaction Processing:

- Transactions, such as reservation requests or updates, should be processed swiftly to maintain user satisfaction.

Update Frequency:

DMV Updates:

- The system should promptly receive updates from the DMV to stay current with rules, policies, and sample questions.  
 - Frequency: As soon as the DMV releases updates.

System Enhancements:

- Regular updates to the system for bug fixes, security enhancements, and new features.

- Frequency: Regular intervals, e.g., quarterly, as needed.

Data Backup:

- Regular backup schedules to prevent data loss.

- Frequency: Daily or as per business continuity requirements.

Security Patching:

- Timely application of security patches to ensure the system's resilience against potential threats.

- Frequency: As security patches are released, with a proactive approach to addressing vulnerabilities.

User Interface Improvements:

- Periodic updates to the user interface to enhance user experience and meet evolving design standards.

- Frequency: As needed based on user feedback and industry trends.

Training Content:

- Regular updates to training content to align with DMV regulations and improve effectiveness.

- Frequency: Semi-annually or as per DMV policy changes.

#### Platform Constraints

System Platforms:

Web Platforms:

- The system must be compatible with well-known web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

- Cross-browser compatibility to ensure a consistent experience for users on different platforms.

Operating Systems:

- The system should be platform-agnostic, capable of running on various operating systems, such as Windows, Unix, and Linux.

- Compatibility with both desktop and mobile operating systems.

Backend Requirements:

Database:

- Utilize a robust and scalable database management system (DBMS) to store and manage data efficiently.

- Examples: MySQL, PostgreSQL, or MongoDB, depending on specific data needs.

Server-Side Technologies:

- Employ server-side scripting languages and frameworks for backend development.

- Examples: Node.js, Django, Ruby on Rails, or Flask.

Cloud Services:

- Consider hosting the application on cloud platforms for scalability and accessibility.

- Examples: Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform.

Integration Tools:

- Implement tools for seamless integration with external entities, such as the DMV for updates.

- Use APIs and web services for smooth communication.

Security Tools:

- Integrate security tools to ensure data protection and user authentication.

- Examples: SSL certificates, encryption protocols, and secure coding practices.

Backup Solutions:

- Implement a reliable backup solution to prevent data loss.

- Regularly backup data to secure storage.

Development Tools:

- Use version control systems for collaborative development and code management.

- Examples: Git, SVN.

Monitoring and Logging Tools:

- Implement tools for system monitoring, error tracking, and logging.

- Examples: Prometheus, ELK Stack (Elasticsearch, Logstash, Kibana).

Containerization:

- Consider containerization for efficient deployment and management.

- Examples: Docker.

#### Accuracy and Precision

User Roles:

- Differentiate users based on assigned roles: Liam (Owner), Ian (IT Officer), Secretary, and Students.

- Assign specific access rights and permissions to each role.

Authentication:

- Implement a secure authentication system, requiring unique credentials for each user.

- Utilize username/password combinations for user login.

Multi-Factor Authentication (MFA):

- Enhance security by considering MFA for critical roles, such as Liam and Ian.

User Profile Management:

- Allow users to manage their profiles with essential information.

User Interfaces:

- Design tailored user interfaces based on roles, presenting relevant functionalities for each user type.

Input Sensitivity:

Username and Password:

- Generally, consider usernames and passwords as case-sensitive to enhance security.

- Clearly communicate case-sensitivity during user registration and login.

Data Entry:

- For other user inputs (e.g., names, addresses), consider case-insensitivity to improve user experience.

System Problem Notification:

Real-time Alerts:

- Implement real-time monitoring for critical system components.

- Send immediate alerts to the admin or relevant personnel for issues affecting system functionality, such as downtime or data inconsistencies.

Automated Email Alerts:

- Set up automated email alerts for less critical issues that do not require immediate attention.

- Provide detailed information about the problem and steps for resolution.

Dashboard Notifications:

- Include a dashboard with a notification area for admins to check for system status and alerts.

Error Logging:

- Implement comprehensive error logging to track issues and generate reports for admin review.

- Utilize log files to capture the details of errors and anomalies.

User Feedback Mechanism:

- Enable users to report issues directly through the system.

- Admins can be notified of user-reported problems through an internal messaging system or email.

Scheduled System Health Checks:

- Conduct regular system health checks and notify the admin of any deviations from expected performance.

- Schedule automated checks during non-peak hours to minimize user impact.

#### Adaptability

Admin Dashboard:

- Create an administrative dashboard equipped with user management functionalities.  
 - Allow admins to add, remove, or modify users through a user-friendly interface.

Role-Based Access Control (RBAC):

- Implement RBAC to allow the admin to assign roles and permissions dynamically without altering the code.

- Provide flexibility to customize user roles based on changing organizational needs.

Configuration Settings:

- Include configuration settings for user attributes (e.g., name, email, role) that can be adjusted without code changes.

- Admins can update user details through configuration options.

Audit Trails:

- Incorporate audit trails to track user management actions.

- Admins can review changes made to user profiles for accountability.

Adaptation to Platform Updates:

APIs and Web Services:

- Design the system with modular components and expose functionalities through APIs and web services.

- Allow for seamless adaptation to platform updates without disrupting core functionalities.

Containerization:

- Utilize containerization (e.g., Docker) to encapsulate the application and its dependencies.

- Simplifies deployment and ensures consistent behavior across different environments.

Continuous Integration/Continuous Deployment (CI/CD):

- Establish Continuous Integration/Continuous Deployment (CI/CD) pipelines for automated testing and deployment.

- Facilitates swift integration of updates and ensures minimal downtime.

Versioning:

- Employ version control for the application code and APIs.

- Enable the system to gracefully handle updates without affecting existing users.

IT Admin Access:

Full System Access:

- The IT admin should have full access to the backend, database, and server configurations.

- Required to perform maintenance tasks, troubleshoot issues, and ensure system health.

Database Management:

- Access and manage the database to perform tasks such as data updates, backups, and optimizations.

Server Configuration:

- Configure server settings and ensure the proper functioning of server-side components.

- May involve adjusting resource allocations or handling server-specific issues.

Security Configuration:

- Manage security configurations, including user access controls, firewalls, and encryption protocols.

- Responsible for implementing and updating security measures to protect the system.

Integration and API Management:

- Oversee integrations with external services, including updates to DMV data.

- Ensure the smooth functioning of APIs and web services.

User Management:

- Use admin privileges to handle user management tasks, including adding, modifying, or removing users.

- May also involve troubleshooting user-related issues.

#### Security

Credentials:

- Users need a valid combination of username and password to log in.

- Ensure secure password policies (e.g., complexity requirements) for enhanced security.

Multi-Factor Authentication (MFA):

- Implement MFA as an additional layer of security.

- Users may need to verify their identity through a secondary method (e.g., one-time codes sent to their mobile device).

Account Activation:

- For new users, require an account activation step (e.g., email verification) to ensure the legitimacy of the account.

Securing Connection/Data Exchange:

SSL/TLS Encryption:

- Employ SSL/TLS protocols to encrypt data exchanged between the client and the server.

- Ensure that all communications, including login credentials, are transmitted securely.

HTTPS:

- Host the system on HTTPS to provide a secure communication channel.

- Secure data in transit by implementing encryption to prevent eavesdropping and mitigate man-in-the-middle attacks.

Secure Socket Layer (SSL) Certificates:

- Regularly update and renew SSL certificates to maintain a secure connection.

Data Validation and Sanitization:

- Incorporate input validation and data sanitization techniques to thwart injection attacks and enhance system security.

Handling Brute Force Hacking Attempts:

Account Lockout:

- Implement account lockout mechanisms after a specified number of failed login attempts.

- Temporarily disable the account to thwart brute force attacks.

IP Blocking:

- Automatically block or throttle IP addresses associated with repeated failed login attempts.

- Prevents an attacker from making numerous logins attempts from the same IP.

- Notification to Admin:

- Set up notifications to alert admins about multiple failed logins attempts for proactive monitoring.

Password Recovery:

Forgot Password Feature:

- Implement a "Forgot Password" feature allowing users to reset their passwords.

- This may involve answering security questions or receiving a reset link via email.

Email Verification:

- Use email verification for password recovery requests to confirm the user's identity.

Temporary Passwords:

- Optionally, provide users with temporary passwords for immediate access, with a requirement to change it upon login.

Account Unlocking:

- Include a process for unlocking user accounts that may be temporarily locked due to multiple failed login attempts.

### Functional Requirements

User Authentication:

- Ensure that the system validates user credentials during the login process.  
 - The system shall provide support for multi-factor authentication to enhance overall security measures.

User Roles and Permissions:

- The system shall differentiate between user roles, including Liam (Owner), Ian (IT Officer), Secretary, and Students.

- Admins shall be able to dynamically assign roles and permissions without code changes.

User Management:

Admins can add, remove, or modify user accounts through an admin dashboard.

- The system shall utilize role-based access control (RBAC) to manage user privileges.

Data Security:

- The system shall encrypt data exchanged between the client and the server using SSL/TLS protocols.

- Passwords shall adhere to secure complexity requirements.

Account Lockout Mechanism:

- The system shall implement an account lockout mechanism after a specified number of failed login attempts.

- Temporarily locked accounts shall be automatically unlocked after a predefined period.

IP Blocking:

- The system shall automatically block, or throttle IP addresses associated with repeated failed login attempts.

User Registration:

- New users shall undergo an account activation process, such as email verification.

- User registration shall include capturing essential information, including username and password.

Password Recovery:

- The system shall provide a "Forgot Password" feature for users to reset their passwords.

- Password recovery requests shall involve email verification to confirm user identity.

Real-time Alerts for Admin:

- The system shall send real-time alerts to the admin for critical issues affecting system functionality.

- Alerts shall include detailed information about the problem for quick resolution.

Role-Based Interface:

- The system shall present role-based interfaces for different user types.

- User interfaces shall display functionalities relevant to each role.

SSL/TLS Certificate Management:

- The system shall regularly update and renew SSL/TLS certificates to maintain secure connections.

Integration with DMV:

- The system shall establish a secure connection with the DMV to receive updates on rules, policies, and sample questions.

- Updates from the DMV shall be encrypted for data security.

Flexibility in User Packages:

- The system shall allow customization of driving training packages by Liam without code changes.

- Liam shall be able to disable specific training packages if necessary.

System Health Checks:

- The system shall conduct regular health checks and notify the admin of any deviations from expected performance.

- Automated checks shall occur during non-peak hours to minimize user impact.

Cross-Device Compatibility:

- The system shall be compatible with various devices, including computers and mobile devices.

- Ensure a consistent user experience across different platforms.

Offline Access:

- The system shall provide offline access to training materials, allowing users to study without an internet connection.

User-Friendly Interface:

- User interfaces shall be designed to be intuitive and user-friendly.

- The system shall include an admin dashboard for convenient management.

Continuous Integration/Continuous Deployment (CI/CD):

- The system shall implement CI/CD pipelines for automated testing and deployment.

- Facilitate swift integration of updates with minimal downtime.

### User Interface

User-Friendly Design:

- The interface needs to be intuitive and user-friendly to accommodate users with varying technical proficiency.

Role-Based Access:

- Implement role-based interfaces to cater to the needs of different users, including Liam (Owner), Ian (IT Officer), Secretary, and Students.

Dashboard for Admin (Liam):

- Admins should have a dashboard providing an overview of system health, alerts, and critical information.

- Liam's dashboard should include features for managing user roles, modifying training packages, and receiving notifications.

User Profile Management:

- Users, including students and the secretary, need an interface to manage their profiles.

- Features should include updating personal information, changing passwords, and viewing training progress.

Training Package Customization (Liam):

- Liam should have an interface to customize driving training packages, including adding or disabling specific packages.

Reservation Management:

- Both students and the secretary need an interface to make, modify, or cancel driving lesson reservations.

- The interface should display available time slots and allow for convenient scheduling.

Lesson Progress Tracking:

- Students should have an interface to track their progress, including completed lessons, test scores, and upcoming sessions.

- The interface should provide detailed information on lessons, including start and end times and driver comments.

Communication Features:

- The secretary should have an interface for managing communications, including receiving inquiries, scheduling calls, and confirming appointments.

- Liam may need an interface for communicating critical information or updates.

Interface for DMV Updates (IT Officer - Ian):

- Ian needs an interface to manage the integration with the DMV, ensuring timely updates on rules, policies, and sample questions.

Web-Based Interface:

- The primary interface should be web-based, accessible through standard web browsers.

- Ensure cross-browser compatibility for users using different browsers.

Mobile Accessibility:

- Design the interface to be responsive, enabling users to access the system on various devices, including mobile phones and tablets.

Cloud-Based System:

- Host the system on the cloud to provide easy and secure access from any location.

- Utilize cloud infrastructure for scalability and reliability.

Offline Access (Limited):

- Provide limited offline access to training materials for users who may not always have an internet connection.

- Key functionalities, such as making reservations, may require an internet connection.

Email Notifications:

- Implement email notifications for important updates, appointment confirmations, or alerts.

- Users should have the option to receive notifications via email.

Interactive Dashboards:

- Admins should interact with dynamic dashboards that allow real-time monitoring and management of the system.

- Dashboards should include visualizations, charts, and key metrics.

Forms and Input Fields:

- Use forms and input fields for data entry, such as user registration, reservation details, and training package customization.

User Support Features:

- Provide in-app guidance, tooltips, and help features to assist users in navigating and using the interface effectively.

- Include a help center or knowledge base for more extensive support.

### Assumptions

Not Specifically Addressed:

Accessibility Requirements:

- Incorporate specific accessibility features for users with disabilities, including support for screen readers, voice commands, and alternative input methods.

Localization and Internationalization:

- Considerations for adapting the interface to different languages, regions, and cultural preferences.

User Training and Onboarding:

- Strategies for user training and onboarding to ensure users understand the system's functionalities and features.

Data Backup and Recovery:

- Detailed plans for data backup, recovery procedures, and disaster recovery strategies.

User Feedback Mechanisms:

- Implementation of user feedback mechanisms to gather insights for continuous improvement.

Scalability Planning:

- Strategies for scalability to accommodate potential growth in the number of users, transactions, or data volume.

Assumptions:

User Technical Proficiency:

- Assuming a reasonable level of technical proficiency among users to navigate web-based interfaces and perform basic tasks.

Internet Connectivity:

- Assuming users, particularly students and the secretary, have consistent internet connectivity for regular system access.

Device Availability:

- Assuming users have access to devices such as computers or mobile devices with modern web browsers.

Security Awareness:

- Assuming users are aware of basic security practices, such as safeguarding passwords and recognizing potential phishing attempts.

DMV Integration Availability:

- Assuming the availability and cooperation of the DMV for seamless integration, updates, and notifications.

Cloud Service Reliability:

- Assuming the reliability and availability of the chosen cloud service for hosting the system.

Regular Maintenance:

- Assuming routine system maintenance, updates, and patches are part of the operational plan.

Communication Preferences:

- Assuming user preferences for communication channels (e.g., email notifications) align with the implemented features.

Training Package Flexibility:

- Assuming the need for training package customization by Liam is limited to predefined configurations and not extensive code-level modifications.

Regulatory Compliance:

- Assuming compliance with relevant regulations and standards related to driving training systems and online services.

### Limitations

Limited Customization Without Development Intervention:

- While the system allows Liam to customize training packages, significant modifications or the addition/removal of modules may require development intervention. The system is not designed for extensive non-developer customization.

Assumption of Internet Connectivity:

- The system assumes consistent internet connectivity for users. Offline access is limited to certain functionalities, and key features like making reservations may require an online connection.

Security Measures:

- While the system incorporates security measures like encryption and account lockout, it may not cover all security threats. Continuous monitoring and updates are essential to address evolving security challenges.

Assumption of DMV Cooperation:

- The system design assumes cooperation and availability of the DMV for integration. Delays or changes in DMV processes may impact the system's ability to receive timely updates.

Accessibility Considerations:

- The initial design does not explicitly address detailed accessibility features for users with disabilities. Future iterations should prioritize accessibility to ensure inclusivity.

Limitations in Resources, Time, Budget, or Technology:

Resource Constraints:

- Limited availability of human resources for development, testing, and ongoing maintenance.

Time Constraints:

- The project timeline, as outlined in the schedule planning, may face challenges due to unforeseen circumstances, changes in requirements, or unexpected delays.

Budget Constraints:

- Budget limitations may impact the selection of technologies, tools, and infrastructure, potentially limiting the scalability or features that can be implemented.

Technology Stack:

- The choice of technology stack may have limitations in terms of scalability, flexibility, or compatibility with future advancements.

User Training:

- The system design assumes users possess a reasonable level of technical proficiency. Adequate resources and time for user training may be limited.

Regulatory Changes:

- Changes in driving regulations or industry standards may require system updates, which might be subject to regulatory approval and could introduce delays.

Testing Coverage:

- Comprehensive testing is crucial, but resource and time constraints may limit the extent of testing coverage, potentially leading to undetected issues.

Integration Challenges:

- Integrating with external services, especially the DMV, may pose challenges due to varying data formats, system downtimes, or changes in their APIs.

Scalability Challenges:

- The system's scalability may be limited by the chosen infrastructure or technology stack, potentially affecting performance under increased user loads.

External Dependencies:

- Dependencies on external factors, such as third-party services or regulatory bodies, may introduce uncertainties that could impact the system's functionality.

### Gantt Chart

.

