Southern New Hampshire University

DriverPass System and Business Requirements

Prepared for: Dr. Phillip Davis
Prepared by: Alexander Ahmann

Prepared on: Oct. 7, 2024

System Components and Design

Purpose

The purpose of this *report* is to plan out and document the software engineering task posed by the client "DriverPass." They are a company that offers services to students of automobile operation, and this report draws from an interview (after CS-255 n.d.) between two of their representatives[1] and two members of our software development team.[2] This document will outline and elaborate on the problem

[1] They are called "Ian" and "Liam."
[2] They are called "Sam" and "Jennifer."

posed by the clients, a well-defined software engineering problem based on discussions with the client's representatives, and the components and design of the system needed, non-functional requirements, and functional requirements.

System Background

Regarding the *project*, its purpose of this project is to develop a system that will allow DriverPass to teach online classes and administer practice tests to their students. The specific problem posed by DriverPass is that their students are struggling to pass exams. To correct this issue, they want to create an online platform that will allow students to take online classes and do practice exams. Their hope is that giving students the option to have on-demand course content delivered through a computer system will help them with passing exams that require the memorisation and processing of information regarding driving.

This kind of computing system should allow for people with the appropriate authorisation[3] to access student information from a central server. There should also be a component for authentication and authorisation based on a role-based access control setup. An accounting component must also be included for teachers, technicians and other staff to track the progress of each student.

With authentication and roles, a "super user" is needed who has full access to the web application, and can reset passwords, read any kind of information on the system, and/or write any kind of information to the system. A lower tier of "teacher" or "moderator" is needed, and finally a "student" tier is needed.

The client also requested a "reservation" feature, which would allow the end user to reserve in-person driving lessons. These "reservations" are to come in three packages:

- **Package One**: Six hours in a car with a trainer
- **Package Two**: Eight hours in a car with a trainer and an in-person lesson where we explain the DMV rules and policies
- **Package Three**: Twelve hours in a car with a trainer, an in-person lesson where we explain the DMV rules and policies—plus access to our online class with all the content and material. The online class also includes practice tests

Furthermore, the client mentioned having only ten (10) automobiles, meaning that the web application would have to account for the number of available cars that can be rented.

Objectives and Goals

From the conversation, I have devised the following objective: "*invent a web application (and optional mobile application) that will deliver content, in the form of online courses and practice tests, to students*." Design the system to have a role-based access control.

The following components are to be included in the web application:

[3] People such as "Liam."

- A role-based access control that gives privileges to end-users based on what permissions that they have been granted.
    - In particular, a "super user" role who can reset passwords, and read and write any kind of data into the system.
- A "reservations" feature that will allow students to make appointments, or "reservations," for in-person driving lessons.
    - A selection of packages, three (3) to be specific, that students can choose from when renting automobiles.
    - A feature to account for the limited supply of automobiles (the client mentioned having only 10 available cars).
- An accounting system that will measure student usage of the web application system, such as:
    - Their access to the coursework materials.
    - Their performance on practice tests.
    - How many times have they took practice tests.
    - Their "reservations" and a system to account for available automobiles.

Requirements

Nonfunctional Requirements

This section will discuss "nonfunctional requirements," which are system requirements that are not explicitly defined in the system. Such stuff includes performance and platform requirements, their restraints, the kind of accuracy in precision, adaptability, and cybersecurity.

Performance Requirements

The web application will run in a cloud environment4 and run on a Linux distribution. The web application should, in theory, be as fast as possible, and not use up too much non-volatile storage space – to satisfy the time-space tradeoff.[5] The exact solution to this problem will require more investigation, and I propose using optimization techniques to formalise the problem and derive the solution.[6]

Regarding software updates, third party software should be updated through the Linux distribution's update mechanism, and new versions of our in-house web application should be pushed onto the cloud platform through a DevOps-style continuous integration and continuous deployment mechanism.

Accuracy and Precision

"Liam" mentioned needing to segregate different users of the web application into different "roles" that give each user the appropriate privileges needed when using said web application.[7] System Administrators are needed – or a kind of user who can reset all passwords, create or delete accounts,

[4] At the request of Ian, DriverPass' chief technician.
[5] See Oechslin (n.d).
[6] See Pierre (2012).
[7] CS-255 (n.d., p. 2)

grant or revoke privileges of other accounts, block access to unauthorised users, and monitor all activity on the web application – including low-level debugging information.[8]

Another kind of user, one who is not quite as privileged as the system administrator, which I will call the "moderator," is needed. This moderator should be able to have some access to the coursework materials, grade students, and track their progress.[9] Finally, a third kind of user, who is the primary consumer of the service being offered, is the "student." The student should have the option to register for accounts on the web application, get "car-rental packages," access video lectures and other learning resources, and take practice quizzes. The "student" role should have the least privileges.

When storing data, the case sensitivity of alphabetical characters should be taken into account. Some data entries should be case sensitive, like passwords.[10] Other kinds of inputs, like email addresses, do not necessarily need to be case sensitive. A *role-based access control system* (RBAC) should be used as a mechanism for authorisation.[11]

Platform Constraints

Regarding the back end of the DriverPass web application, the end-user should not have to run too many clients side plugins[12] to access the content stored on the web application. Most of the "accounting" work should be done on the server side, so a relational *database management system*[13] (or relational DBMS) and web script processor[14] should be installed onto the web server stack.

Regarding the client-side of the DriverPass web application, the end-user, whether a system administrator, moderator, or student, should not need to install third-party tools in order for it to function. The only apparatus needed to access the web application is a standard web browser like Mozilla Firefox or Google Chrome.

The only exception that I can think of is an SSH client for the system administrator in case the web application is not functioning properly and cannot be accessed through a web browser. An SSH client can be installed on the system administrator's computer system so that they can access the server hosting the web application is being served from, so that problems can be mended.

Adaptability

---

[8] Non-technical system administrators may not be interested in debugging information. But this kind of information can be helpful for technicians authorised by the system administrators in debugging or troubleshooting. Additional features should be added to assist non-technical system administrators in not having to concern themselves with technical details.

[9] As per the request of "Liam" (CS-255, n.d., p. 2).

[10] I intend for passwords to be put through a cryptographically secure hashing function. A single change in a bit that composes the string must result in a completely new hash. Therefore, the passwords are case sensitive.

[11] See Computer Security Resource Center (n.d.).

[12] Some examples of superfluous plugins include the (now depreciated) Adobe Flash, Adobe Shockwave, Java Web Client, et cetera.

[13] Examples include MySQL or SQLite.

[14] Examples include PHP and Python Django.

DriverPass' technical team wants the web application to work in a cloud environment and does not want to have to deal with low-level maintenance.[15] Therefore, we should implement cloud automation to handle such details.

Nonetheless, a SSH service should be installed to act as a "backdoor" in the event that a critical failure occurs, and technicians need to mend the situation. The system will also update software using the chosen UNIX distribution's mechanism for software updating, or if updating our software product, new versions of the codebase will be deployed to it via a continuous integration and continuous deployment system.

Security

As described by Liam, a user is segregated into different roles: a role for a "big boss" administrator, another for technicians, another for the secretary, and another for "students." The authentication method should be proportional to the level of access: for example, an administrator should need multi-factor authentication with strict time limits, technicians should be subjected to the same level of need to prove their identity, whereas the "secretary" and "student" should only need a password, and the time limits to use the web application should be more relaxed.

My proposed technical requirement for security comes in two forms: prevention and mitigation. Regarding prevention, rate limiting should be put into place to lower the chances of a brute force or fuzzing attack of being successful. The cloud system should only allow twenty requests per second (20 req/sec).[16] Brute force and application fuzzing tooling relies on making many requests at a high rateuntil a vulnerability is discovered, implying that rate limiting will prevent, or at least slow down, such attacks.

Continuing the theme of preventing attacks, the introduction of standard tools like firewalls, intrusion prevention systems and endpoint detection response will be helpful in preventing malicious attackers from getting full control of the system. Using memory-safe languages like Java or Rust will prevent code injection at the "native binary" level, safe coding procedures like prepared statements and non-alphanumeric character filtering will mitigate web application-level attacks like SQL Injection and cross-site scripting, and passwords should be stored as a cryptographically secure hash to make it more difficult to crack passwords if they get leaked.

Regarding mitigation: I posit that computer systems will eventually be compromised by a malicious hacker. Therefore, our in-house technical team should inform DriverPass if a breach is to occur and put forward a *digital forensics and incident response team* to find and eradicate malicious hackers who were able to breach our systems.

[15] Quote from Ian, their technician (CS-255 n.d.): "The system needs to run off the web, preferably over the cloud. We do not want to deal with backup and security; we need that to be taken care of. We need our focus to be on running the business with minimal technical problems."
[16] Jin et al. (n.d.) reports that "normal" clients make HTTP requests at about 20 requests per second.

Regarding password management, if “students” forget their password, then the administrator or a technician part of DriverPass should be able to reset it. If, by some bizarre series of unfortunate events, all administrators and technicians forget their password, then a SSH backdoor should be in place for in-house developers to login and reset their passwords.

Functional Requirements

The web application system will:

- Supply educational content and practice exams to the “students”, with content being developed or curated by the administrators and teachers.
- Allow an administrator or technician to reset passwords.
    - Contains a backdoor for in-house developers to reset passwords in an unlikely event.
- Allow students to schedule rentals for automobiles that will allow for them to do “in person” practice.
    - Include a queue and accounting system that will take the limited number of rental cars available.
- Allow administrators and technicians to block student access if need be.
- Include a mechanism by which to process financial payments.

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 needs to present information and controls depending on the role of the user logging into the system. Regarding the:

- "Big Boss” System Administrator:
    - They need to be able to track all aspects of the student accounts and other accounts regardless of roles.
    - They need to be able to download student data, and other logging data, into a tabular format (like Microsoft Excel or a simple CSV format).
    - They need to be able to reset passwords.
- Technician Accounts:
    - Needs to be able to reset passwords.
- Student Accounts:
    - Needs to be able to access educational content in the form of videos and other resources.
    - Needs to be able to take practice quizzes.
    - Needs to be able to make requests for “in person” driving and rental of a limited set of cars at the DriverPass facility.
    - Needs to have an interface to enter their personal details and financial information to purchase the products.

Assumptions

Regarding the server side of the application:

- We are using a cloud system that can scale its hardware resources as needed.
- The system is running a Linux, or some other *NIX-like, operating system.

Regarding the client side of the application:

- The end-user has a modern web browser, like Mozilla Firefox, Microsoft Edge, or Google Chrome, that can render HTML5 and CSS3 and can execute JavaScript client-side.
- System Administrators and Technicians have SSH clients that can allow themselves, or our in-house developers, to login to the cloud hosting in the event that the web application is not functioning properly.

Limitations

The following are limitations that I can think of:

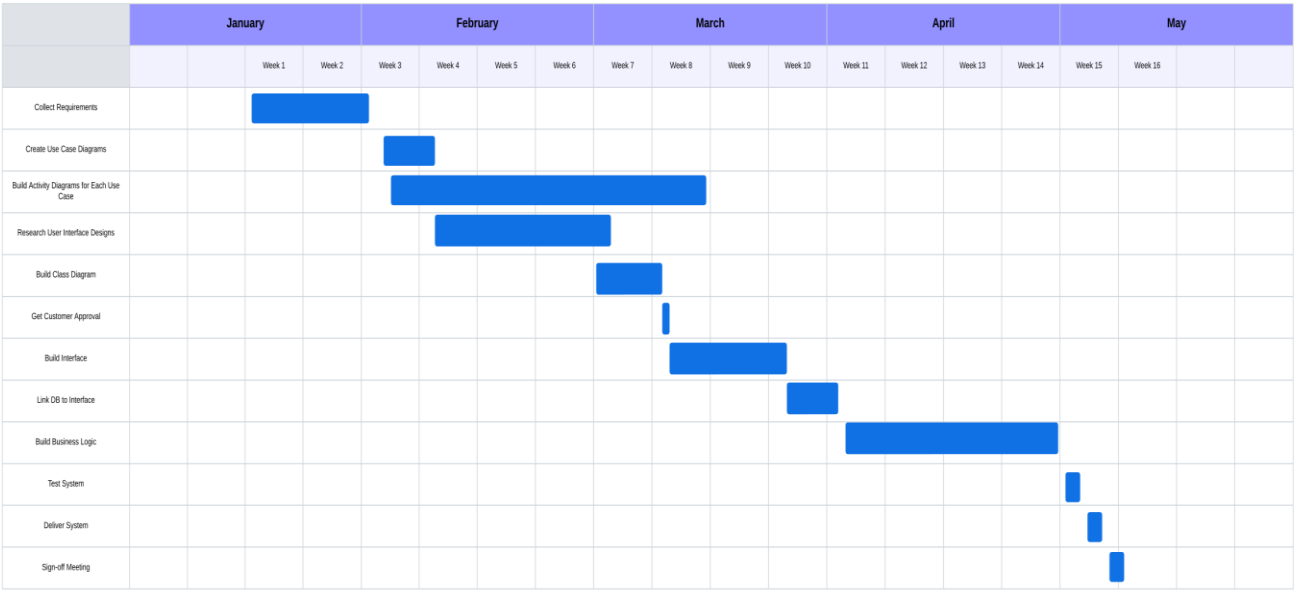
- The cloud computing provider is not able to scale depending on the number of students enrolling or content uploaded onto its non-volatile storage devices.
- The student does not know how to use a computer system.[17]
    - Or, rather, the student does not know how to use specifically our web application, for whatever reason.

Timeline for System Development

The following Gantt chart is a depiction of the schedule by which to proceed with developing the web application:

[17] This implies that there should be some traditional ”pencil and paper” form that a student can fill out, which will be inputted into the system by a technician.

DriverPass Web Application Deployment Schedule



References

- CS-255 (n.d.). *DriverPass Interview Transcript*.
- Computer Security Resource Center (n.d.). *role-based access control (RBAC)*. National Institute of Standards and Technology. Retrieved on Oct. 1, 2024, from: https://csrc.nist.gov/glossary/term/role_based_access_control
- Jinghe, J., Nodir, N. Im, C., & Nam, S. Y. (n.d.). *Mitigating HTTP GET Flooding Attacks through Modified NetFPGA Reference Router*. Retrieved on Oct. 7, 2024 from: https://www.researchgate.net/publication/267384975_Mitigating_HTTP_GET_Flooding_Attacks_through_Modified_NetFPGA_Reference_Router
- Pierre, D. A. (2012). *Optimization Theory with Applications*. Dover Publications.
- Oechslin, P. (n.d.). *Making a Faster Cryptanalytic Time-Memory Trade-Off*. Archived at: https://web.archive.org/web/20120204073703/http://lasecwww.epfl.ch/pub/lasec/doc/Oech03.pdf