Analysis Report

In General

Why requirements analysis? The DevOps lifecycle consists of many lifecycles within itself. There starts with a need or opportunity, then there are stakeholders that are identified, then the requirements are elicited and built using different techniques, and then models are created and analyzed throughout the development process. The requirements process itself has a repeating lifecycle, that when the development starts to become fulfilled, requirements will continue to be elicited. Some things will need to change, and it is important to keep everything precise. Precision in terms of ensuring the requirements fulfil the client needs, and in terms of accurate language. It is inevitable that the client will want certain aspects of the system to be changed throughout the lifecycle, which will change some requirements, and that includes the requirements dependent on the initial changing requirements. If these changes are not kept up with, a developer that may not know of these changes will execute the requirement as it was originally intended. Ensuring requirements are precise and up to date are two reasons it is essential to continue combing through them, as processes are constantly evolving throughout the program lifecycle. Another reason to read through all the related requirements together is to ensure they form a cohesive image of what the final product should look like and that they aren't conflicting with one-another. It is possible, and highly likely, that in a large project, as requirements are being jotted down, some may overlap or conflict, as ideas that the client has may do the same thing. It is up to the project manager and developers to ensure that the requirements all fit together once they are ready to be looked at for implementation. Lastly, analyzing requirements needs to be done after their elicitation to ensure they are fitting to all the construction specifications outlined throughout the SWEBOK, such as ensuring they solve a problem, qualify measurable conditions, bounded by constraints, etc. This includes the precise wording, which is not all too natural, and would take at least one double-check, fixing any imprecise wording that may have initially been included. In the end, as mentioned in SWEBOK, the requirements must be precise enough to enable the requirements to be validated, their implementation verified, and the cost of each to be estimated. If done correctly, a uniform cost of time and effort can be estimated from gathering a sum of all requirements.

How are requirements analyzed? First, it is important, as previously mentioned, to re-read through the requirements created as-is. It is necessary to find and fix and conflicts between two or more requirements. This can be done by deleting over-lapping requirements or determining how one requirement may be the correct route, while the other was either a misunderstanding or a previous version of a requirement. The requirements need to interact with an organization and their environment as seamlessly as possible to ensure the software will be cohesive with their day-to-day lives, rather than become another task. Part of that is making sure the software properly represents the organization and does not conflict with ideals or become too complicated for the functions it is designed to assist. The software requirements hidden inside of the system requirements need to be drawn out as well. When the system needs to perform a task, usually that task can be broken down into multiple software requirements, which will be easier to follow during the development process. Requirements can be classified into two main groups, which are functional and nonfunctional. Functional describes functions that the software will execute. This can be elaborated as a software or feature. Nonfunctional describes a constraint or quality requirement. These can be further classified into performance, maintainability, safety, reliability, security, interoperability, or many other types of requirements, which are all included in the nonfunctional category. The classification of where the requirement is derived also needs to be included. This includes if the requirement is derived from another high-level requirement or a direct requirement. If the requirement is on the product or the process. A process requirement is a constraint on the development of the software, such as which process needs to be followed during development, while a product requirement is a need about the software itself. Another important component of classification is the priority of a requirement. This can be broken down into “low”, “med”, “high”, which would determine what needs to be accomplished first. These can be determined with feasibility, dependance, client-needs, etc. The requirements also need to be classified into scope. The last classification for a requirement is the volatility or stability, which would describe subsystems which may need to change due to external regulations, rules, standards, or any other decision which would affect the operability of that requirement.

Another important portion of requirements analysis is conceptual modeling. This is where the development team can create some visual aid or systematic relationship mapping of the project in varying scopes. This is useful to test and see where each requirement lies within the overall project, what may depend on another requirement, and develop a better, general understanding of the system as a whole. Within the development team, these models can be used to go back and change the previously mentioned characteristics and classifications. Once these requirements are re-analyzed using the conceptual model, an accepted (easy to follow) model can be brought to the client to further elicit requirements and determine what, if any, changes would need to be made in the planning before beginning the development. A few types of conceptual models mention in SWEBOK are use case diagrams, data flow models, state models, goal-based models, user interactions, object models, and data models. All of these are notations for the Unified Modeling Language (UML). When deciding which modeling techniques to use, there are 3 factors which need to be considered. One is the nature of the problem being solved by the proposed system, the expertise of the engineers and what they are comfortable with, and the process requirements for the customer, which would override the previously mentioned factor. What is recommended is beginning with modeling of the software context, which describes the relationship between the software and its external environment.

For TSPA – Unified System

For this project, a general idea of the daily system routines was understood to determine what the opportunity could be for a new system. Generally, when for a single task, multiple systems are used, that is a good indicator of a problem that needs to be solved. That is the general goal of this system. After building the initial requirements, they were then re-looked at through a checklist for precise wording. This is when they were also determined to be function or nonfunctional and if they were process or product requirements. After they were built and used for concept diagrams, they were given relationships with one-another where applicable. At this point, any conflicting requirements should have been either nullified or reconfigured.

Requirements List

Key	Summary	Description	T	Linked Issues	Labels
SPA-39	The system shall log users out of their accounts upon submission of beginning of system maintenance.	The system will display a warning message to all users continuously 15 minutes prior to maintenance, and after that 15 minutes has passed, all users will be logged out of their accounts to ensure no data leakage or data-flow during database changes.			functional, product
SPA-38	In case of maintenance during working hours, the system shall redirect users to the modified home page, with a prompt of estimated time and a phone number.	This will allow users to call regarding any needs they may have so the appointments can be handled on the local database.			functional, product
SPA-37	Upon submission of permission change by the Director, the system shall log that user off of their profile before submitting changes to the database.	Logging a user off of their profile before submitting the changes will keep from any user permission dualities occurring from them navigating their profile after the change in permissions.			functional, product
SPA-36	The system shall provide the Director a view for granting a specified student additional client permissions for special appointments, upon completion of specialized training	This time frame is dependent on the director's action. A later version will send the director a request once the student updates their Graduation Tracker with the specialized training completion.			functional, product
SPA-35	Upon the scheduling of a client's first appointment, the system shall create a new user profile unique to that client.	The unique identifier for the customers will be a phone number and unique ClientID. This is the profile that will be updated with any new student/ educator comments.			functional, product
SPA-34	If a student is not checked out within one hour of their completed class time, the system shall log them out with a notification to	The message will be sent via email upon the automatic checkout to the corresponding educators and administration.			functional, product

	administration via their web portal.	This is to be later updated as an in-system user message, which will appear when they open their smart-phone application. [To be completed in mobile release]	
SPA-33	When the admissions department accepts a student into the program, the system shall create a profile for that student for storing graduation progress.	Basic information such as name, phone number, and email will be stored, as well as create a unique Student ID.	functional, product
SPA-32	The system shall send data to the cloud-based server first before storing it in the locally-based database.	Sending all data to the cloud-based server is to ensure the local network/ hardware is not the source of any bottle-necking in the service. This is due to limited network and database features on-site. This is the new alternative to a previously mentioned constraint. (SPA-23 To Do)	nonfunctional, product
SPA-31	When a user is viewing any tab, accessible information shall be 2 or fewer interactions from the current view.	This is to prevent any convoluted user interface with hard-to-find data. All data required by the user needs to be less than 3 “clicks” away from their current view on any specified tab.	functional, product
SPA-30	When a user logs in, the database shall respond with the user class before sending other user information.	It is important for the universal portal to determine user class before sending other related data to ensure they are being given only the information pertinent to their user class. Other user classes can view sensitive information.	nonfunctional, product
SPA-19	Upon submission of updated schedules, the system shall display the schedule in the calendar view to display 3 weeks prior and a year ahead.	When 3 weeks have passed, the information will be stored in a database and external excel file for record-keeping.	functional, product

SPA-18	When a new schedule is created, administrators shall have the ability to edit the calendar view beginning two days from the current date.	If a schedule needs to be changed within the two-day window, the Director will need to accept it, as well as any past dates, along with a non-removable log to show the edits. The administrators will have an entire calendar year from the current date to edit upcoming schedules.	SPA-17	functional, product
SPA-17	The Scheduling system shall display an Employee work schedule after it has been posted by administration.	This would be viewable by an employee or administrator at any time, through a web application, displaying up to two weeks prior to the current date.	SPA-18	functional, product
SPA-16	Upon completion of a student appointment, an educator shall have access to verify the appointment within a 30-minute window of the scheduled time.	The Director will have ultimate access to re-open and edit any past appointments as well as add notes. The educator and students will also be able to add notes BEFORE and 30 minutes following the verification of completion.	SPA-7 , SPA-12	functional, product
SPA-15	The Director shall have access to view and edit the running total of products at any time to account for purchasing replacements when a threshold of on-hand is crossed.			functional, product
SPA-14	The Product Tracker system shall allow support staff to edit the running total of products available, used, and sold throughout the given workday.			functional, product

SPA-13	The system shall allow a student to view their overall graduation progress from their profile at any time, updated per appointment.	This will be accomplished through an “Individual Graduation Tracker” tab in the overall system that will appear for all students. This will only show the individual student’s information in regards to progress to graduation such as how many of each type of appointment they have completed and what certifications/ exams they have passed.	SPA-9	functional, product
SPA-12	Upon successful completion of a client appointment, the system shall store the category of appointment under that student's profile as a categorical count for graduation tracking, within 5 minutes of verification.	When the scheduled appointment is finished with the student, an educator will verify the appointment is complete, and that is when the appointment will go from “in-progress” to “completed”, and will be stored under the category of appointment, such as hair dye, hair cut, and can fulfill multiple appointment types.	SPA-16	nonfunctional, product
SPA-11	When a student checks in, the system shall store the name of the student along with the current time, in order of arrival.	When a student signs in using the finger-print scanner, the corresponding information (name, time and date, signed in/out) will be sent to the database, which is grabbed by the user (if they have access) when they load the viewable item.		functional, product

When evaluating each requirement, they are checked through a checklist described in the MITRE table and in the “Characteristics of a Set of Requirements” from 29148.

The MITRE table includes the following foundations to check

- The system-level technical requirements are traceable to the user requirements.

- Each system requirement describes something relevant: a function the system must perform, performance a function must provide, a constraint on the design, or a reference such as to an interface definition.

- The level of detail that the requirements provide about system functionality is appropriate.
- The requirements are sufficient to describe what the overall system must do, what its performance must be, and what constraints an engineer should consider.
- The requirements include any legal or regulatory constraints within which the system must perform.
- The requirements include enterprise architecture constraints within which the system must integrate (or toward which the system is desired to migrate).
- Environmental design requirements are specified.
- All external interfaces for the system are included. Major internal interfaces may also be included if they are important to system modularity, or future growth in capability.
- Requirement statements use the word "shall"
- Requirements statements are unambiguous.
- Performance requirements statements (including logistics/sustainment/support) are quantifiable, testable, and/or verifiable.
- If objective performance values are included as goals, ensure they are clearly identified and distinguished from firm requirements.
- The operational and support environment is described and defined.
- The requirements include appropriate use of Government and industry specifications, standards, and guides.
- Verification approaches for all system performance and sustainability requirements are complete and appropriate.
- Every requirement must have a verification method identified.
- If a requirement cannot easily be verified by a direct inspection, measurement, or one time demonstration of the requirement, the verification requirement should include an expanded test criteria description to ensure that there is no disagreement later in the program.

And the requirements also need to follow these following characteristics:

1. Include all requirements types relevant to the system under consideration

2. Account for requirements in all stages of the life cycle
3. Involve all stakeholders in the requirements elicitation, capture, and analysis activity.

The set includes individual requirements which are:

- consistent, non-overlapping/conflicting, and unique.
- Feasible when considering acceptable risks and constraints.
- Comprehensible and clear within the scope of the project they are describing
- Able to be validated to prove completion within constraints.

All of these will help to prevent requirements creep during the life cycle along with any unnecessary redundancies and incomplete portions.

Characteristics of this set of system/software requirements

- Complete - All requirements in the set fully describe one specific requirement. Any additional information which may be needed for clarification, to avoid wordiness and breaking the standard requirements format, that information is included in the description.
- Consistent - Each requirement in the set is one-to-one with some aspect of the system. To maintain this, I avoided terms like "and" which would show that it could be broken down into multiple requirements. If it depends on another requirement, it is noted in the description or dependencies.
- Feasible - Each requirement was specifically created with this characteristic in mind. If the feasibility is low (relatively), then it does not describe some critical functionality of the system and its priority is set below medium.
- Comprehensible - Clarity is relative, but each requirement was written in such a way that somebody without advanced technical knowledge can understand what is required. This was double-checked by reading each off to a friend that is not in a CS major.
- Able to be validated - For each requirement, there is some measurement that will show when the requirement has been accomplished. If it is not immediately obvious in the requirement itself, then it will be clarified in the description.

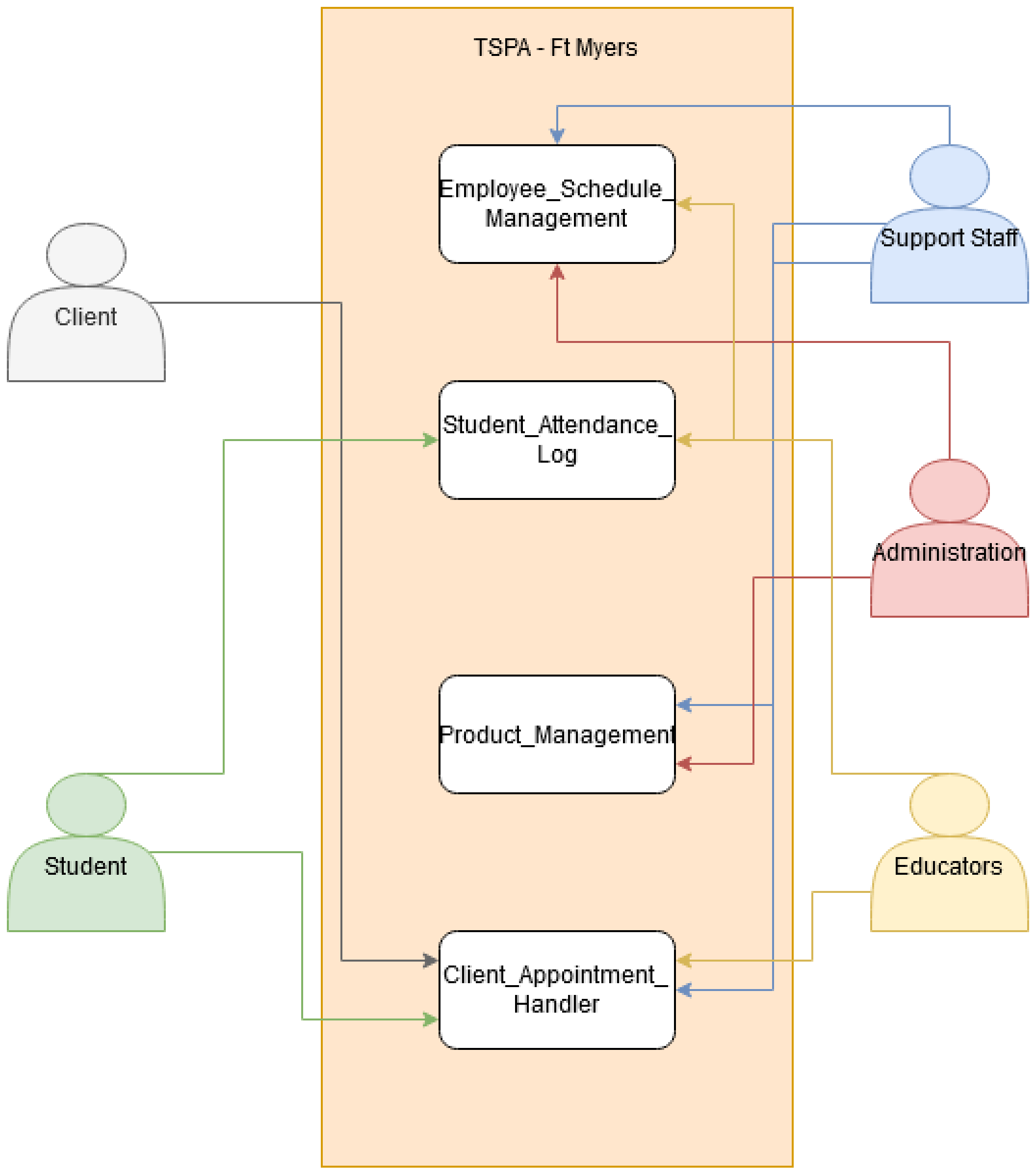
TSPA Use Cases

ID:	**TSPA0001**
Title:	Employee_Schedule_Management
Description:	Web view for administration to manage employee scheduling.
Primary Actor:	Administrator
Preconditions:	Administration creates an excel document with dates and times to email to each of the employees.
Postconditions:	Administration has management access over a shared view with employees which they can create the schedule and the employees can view, without the need for emailing.
Main Success Scenario:	There will be a portal which administrators and employees can view a shared calendar. This calendar will show dates and times of each employee. The administrator will have access to a year in advance for scheduling and will input the schedule in with the excel sheet. This is where the system will process the excel sheet into the calendar for ease of use. The calendar view will have filters for each employee class to make it easier to view the schedules for specific types of employees.
Extensions:	If the administrator does not stick to a specified format for the excel sheet, it would throw off how the system displays the view in the calendar. Due to this, there will need to be manual editing in the calendar view for members with administrator access.
Frequency of Use:	Each schedule is posted for the month. It will be posted by the administrator in that frequency to be viewed throughout the month.
Status:	No Progress
Owner:	Christopher Frank
Priority:	VERY HIGH

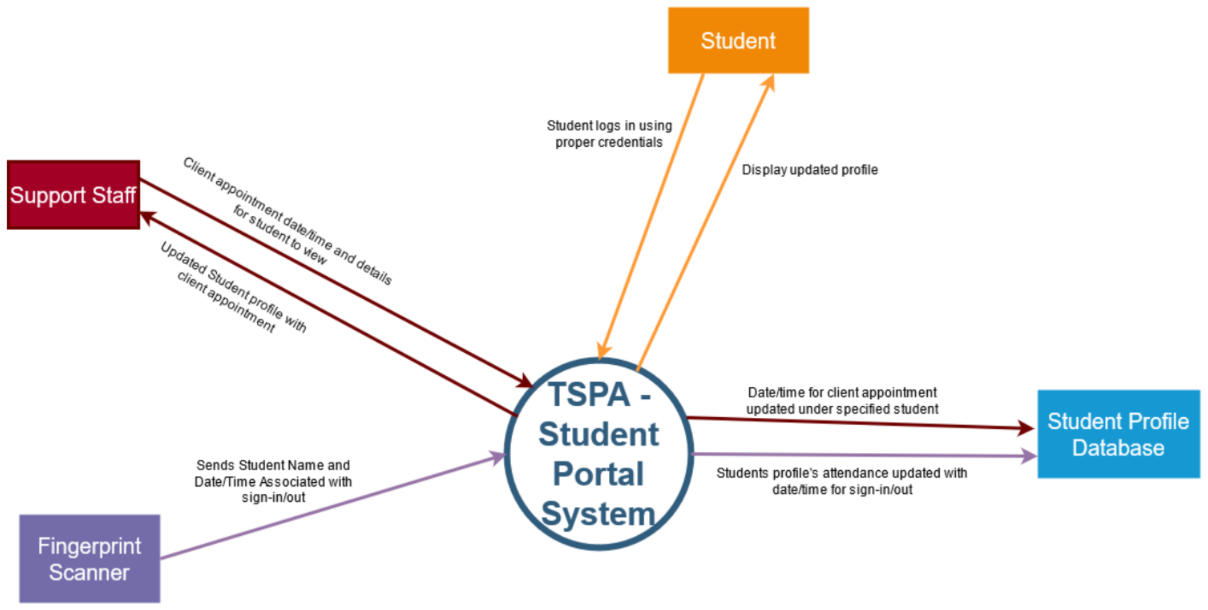
ID:	**TSPA0002**
Title:	Student_Attendance_Log
Description:	Automatically track and save Student attendance
Primary Actor:	Student
Preconditions:	Students use a fingerprint scanner which logs the date and time for their entry, breaks and exits. These logs are manually taken and copied into an Excel sheet for data keeping. From there, the attendance will be viewed for each student to ensure they meet hours for graduation requirements.
Postconditions:	Students use finger-print scanner which logs the date and time for their entry, breaks and exits. These logs will be automatically collected into the system and profiled under each student. Tracking each log, as well as automatically compiling them into total_hours_per_week/semester.
Main Success Scenario:	When students use the fingerprint scanner it is to indicate they are signing in for the day, taking a break, or signing out for the day. When the students use it for the first time (for each day), the system will log the date and time of the entry, and until the next entry is logged, the time will be compiled for their total time. When they use it again to check out, the time and date will be logged, along with stopping that compilation of total time.
Extensions:	If a student leaves for the day and forgets to sign out, the system would continue to count and would actually mark them as checking out the next day when they sign in, with all of that time at night counted for them. To combat this, there will be an automatic time-out function which would sign them out, and alert administration that they didn't sign out within an hour of their scheduled class hours ending.
Frequency of Use:	The students sign in and out throughout the day, each day, weekday and weekend. This all depends on if the business is operational (business as usual).
Status:	No Progress
Owner:	Christopher Frank
Priority:	HIGHEST

ID:	**TSPA0002**
Title:	Student_Attendance_Log
Description:	Automatically track and save Student attendance
Primary Actor:	Student
Preconditions:	Students use a fingerprint scanner which logs the date and time for their entry, breaks and exits. These logs are manually taken and copied into an Excel sheet for data keeping. From there, the attendance will be viewed for each student to ensure they meet hours for graduation requirements.
Postconditions:	Students use finger-print scanner which logs the date and time for their entry, breaks and exits. These logs will be automatically collected into the system and profiled under each student. Tracking each log, as well as automatically compiling them into total_hours_per_week/semester.
Main Success Scenario:	When students use the fingerprint scanner it is to indicate they are signing in for the day, taking a break, or signing out for the day. When the students use it for the first time (for each day), the system will log the date and time of the entry, and until the next entry is logged, the time will be compiled for their total time. When they use it again to check out, the time and date will be logged, along with stopping that compilation of total time.
Extensions:	If a student leaves for the day and forgets to sign out, the system would continue to count and would actually mark them as checking out the next day when they sign in, with all of that time at night counted for them. To combat this, there will be an automatic time-out function which would sign them out, and alert administration that they didn't sign out within an hour of their scheduled class hours ending.
Frequency of Use:	The students sign in and out throughout the day, each day, weekday and weekend. This all depends on if the business is operational (business as usual).
Status:	No Progress
Owner:	Christopher Frank
Priority:	HIGHEST

ID:	**TSPA0004**
Title:	Product_Management
Description:	Track amount/ type of product on-hand.
Primary Actor:	Support Staff
Preconditions:	To track how much product is on-hand and what has been sold or used by the students, the support staff will count each item and track it on a sheet of paper to be filled into an Excel sheet later.
Postconditions:	There will be a running track of product in the system which the staff will double-check. Each item includes an optional photo and a running total for on-hand, sold, or used.
Main Success Scenario:	When a client purchases product, the support staff will simply include the amount sold in the system, on the product view. At the end of the day, the support staff will count the items used as usual and also mark that in the system.
Extensions:	The data will be collected and stored for administrator analysis to see how much students are using and how much product is being sold. The log will also track which staff kept track for better accountability.
Frequency of Use:	Each time a client purchases an item and each day during use count.
Status:	No Progress
Owner:	Christopher Frank
Priority:	LOW


TSPA - Ft Myers
Employee_Schedule_Management
Student_Attendance_Log
Product_Management
Client_Appointment_Handler
Client
Student
Support Staff
Administration
Educators

The TSPA Unified System incorporates many functions of their day-to-day operations including all possible user classes from the students, clients, and educators, to the director. One portion of the system is the student system which tracks their graduation progress to become viewable at any time on their own personal profile. That is described in the Level 0 data flow diagram, or context diagram below.


Student
Student logs in using proper credentials
Display updated profile
Support Staff
Client appointment date/time and details for student to view
Updated Student profile with client appointment
TSPA - Student Portal System
Date/time for client appointment updated under specified student
Student Profile Database
Students profile's attendance updated with date/time for sign-in/out
Sends Student Name and Date/Time Associated with sign-in/out
Fingerprint Scanner

This diagram was used to represent and analyze the interactions between the users and the student system for a few different operations that are described in some of the requirements.

Critical Thinking

A good first place to start is with the purpose. My purpose for building this is to solve the problem, which also addresses the "Question at Issue" portion of the Elder Paul Critical Thinking model. The problem at-hand is the unavailability of information for any user class unable to access an excel file at the front desk. The other issue is the non-uniformity amongst the variety of data logging methods. Bringing all of these together would create an easier workflow for TSPA's day-to-day operations. The information is gathered through the elicitation process. This required asking formulated questions which gave me an understanding of how their usual day works and the repeated functions that happen throughout. This also provided me with the point of view from each user class for each function. For example, when a client wants to get a hair cut, I had to imagine all of the functions that would take place from the client calling and setting an appointment, to checking in, receiving it, and leaving. This required the interaction from no less than 4 different user classes, which all play a vital role in that process. Some assumptions needed to be made from that point of view to incorporate certain functionalities and data that were not available before, such as leaving notes about a specific client, which would be viewable for whoever needed to help that client next. The consequence of that added functionality is, if they had a specific issue before, it shouldn't happen again, thus creating a better experience for everyone. The inferences made here is that for each requirement, included added functionalities that didn't exist before, it should make the stakeholders which are interacting with it, happier than they would without it. The core concept/ principle used for every requirement is to make sure it flows seamlessly with their existing day-to-day interactions and if it could be cumbersome to the user, fix it or get rid of it completely.

Ethical and Professional Responsibilities in Engineering

Every decision made in engineering is made to affect multiple people outside of the realm of the development process. This means, in some way, changing their lives. When making these decisions, there needs to be a logical and systematic thought process which would ensure the outcome is better than what would have been the case without whatever is being implemented. As engineers, it is our goal to make people's lives easier, more efficient, or more fun. If some designed solution is being used, but does not accomplish the previously mentioned goals, that engineer failed his/herself and all other stakeholders involved. Failure is a necessity to learning, but if anything is well-thought out enough, it should be mitigated in a reasonable fashion.   In terms of this TSPA system, it is my goal to allow the students to have the peace of mind they can track their progress to graduation, give the clients a better experience, also giving the overall business a better reputation, and allowing all managed data to exist in one, centralized location. The goal is to accomplish this in a manner which would seamlessly exist in their day-to-day operations before its implementation.