PROJECT A

COMP2050 ASSIGNMENT 1

SOFTWARE REQUIREMENTS SPECIFICATION

Written by Ben Cavanagh - 46975373

Client Representative: Rumeth Herath - 47087854

MACQUARIE University
SYDNEY·AUSTRALIA

REVISION HISTORY

VERSION	DATE	CREATOR OF AND AGREER TO THE CHANGE	DESCRIPTION
1	13/08/2022	Ben Cavanagh	Started the SRS
2	13/08/2022	Ben Cavanagh	Completed 'product vision statement' and 'purpose'
3	14/08/2022	Ben Cavanagh	Completed 'scope'
4	14/08/2022	Ben Cavanagh	Completed context diagram and 'overall description'
5	15/08/2022	Ben Cavanagh	Completed 'product perspective'
6	15/08/2022	Ben Cavanagh	Completed 'user characteristics' and product functions'
7	16/08/2022	Ben Cavanagh	Completed 'constraints'
8	16/08/2022	Ben Cavanagh	Completed first draft of 'assumptions and dependencies'
9	16/08/2022	Ben Cavanagh	Completed first draft of 'specific requirements'
10	17/08/2022	Ben Cavanagh	Completed 'assumptions and dependencies'
11	23/08/2022	Ben Cavanagh	Completed 'references'
12	27/08/2022	Ben Cavanagh	Completed 'definitions, acronyms, and abbreviations'
13	04/08/2022	Ben Cavanagh	Completed 'assumptions and dependencies'
14	06/08/2022	Ben Cavanagh	Completed 'specific requirements'
15	10/09/2022	Ben Cavanagh	Completed 'table of contents'
16	10/09/2022	Ben Cavanagh	Finished the SRS

TABLE OF CONTENTS

INTRODUCTION

Project Vision Statement: "To promote clean national parks across New South Wales"

Purpose

This document is intended to be read by the client representative and the stakeholders of the project. The client has given the following task: to plan the software requirements for their new system in development. This process involved structured meetings with the client to gather and write a comprehensive set of requirements for their system.

The software requirements specification was written as a result of this process and a solution to the problem given. It will give an overview about the software aspect of the project as well as its goals. It will explain the functionality of the software of the project and its expected performance.

Scope

The document will provide:

- A general description of the service
- An explanation for how the service will meet the stakeholder needs
- A description of the project's functionality and features
- User characteristics
- A set of constraints giving limitations
- Assumptions and dependencies
- A detailed list of functional and non-functional requirements

The document will omit the technical aspects and the design of the software interface.

The purpose of a context diagram is to show how the code in the system will operate logically as a single process. They identify the data entering and the information leaving the system. The following context diagram visually shows the scope of the system.

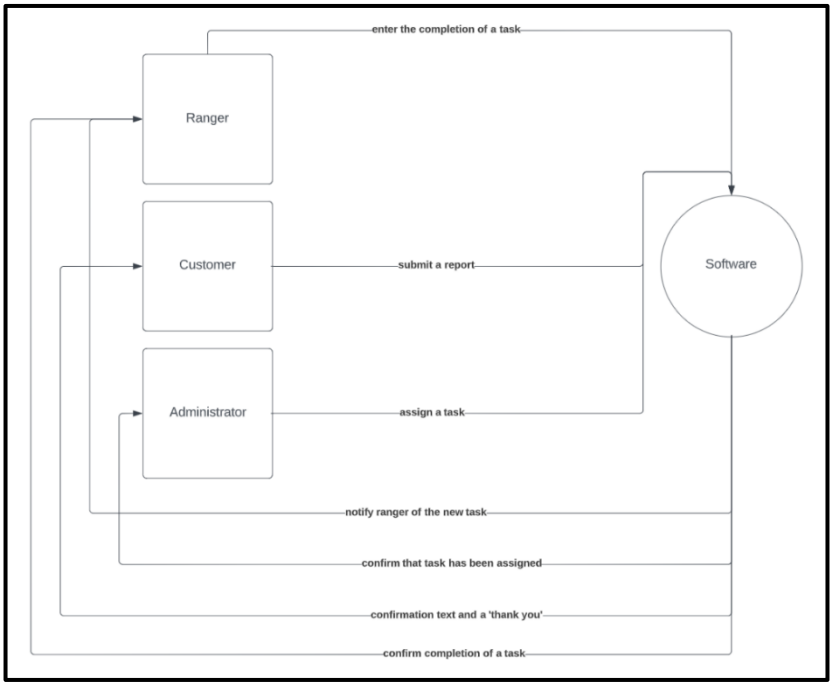

FIGURE 1

Definitions, Acronyms and Abbreviations

TERM	DEFINITION/DESCRIPTION
Agile Methodology	An approach to software development, where one iteration is made after another in small iterations
Administrator	A user who gives 'tasks' to certain rangers from a task list
Context Diagram	Shows how the system works as one process, like a high-level data flow diagram
Customer/Visitor	A user who can report a sighting of litter on the ground of a national park
Interaction Diagram	Shows the interactions between different participants inside a use case of a system
NSW National Parks	An app that can be purchased, providing information about national parks in NSW Australia
PPE	Personal Protective Equipment
Ranger	A user who receives tasks and carries out the litter removal process
Scope Creep	An undesirable effect of poor project management where the scope continues to grow after requirements should have been clearly defined
Tasks	Reports of litter sent from users
Use Case	A unique situation where a user will interact with the system
Use Case Diagram	Shows many of the most important use cases in a system

References

[1] "NSW National Parks and Wildlife Service | Home | NSW National Parks", NSW National Parks, 2022. [Online]. Available: https://www.nationalparks.nsw.gov.au/. [Accessed: 16- Aug- 2022].

OVERVIEW

Overall Description

At the time of writing, the *NSW National Parks* app has a range of functional features, from searching for nearby parks to providing safety rules. However, it lacks any kind of feature to report a sighting of rubbish or litter, increasing the difficulty for park rangers to do their jobs.

This is the main problem that is intended to be solved with the use of a new service for the app. The service will give users the ability to report any litter they see as they come across it. This information will be captured and sent to a ranger on the site, who will be able to remove the litter. A closer look at the functionality of the service is described in “Product Functions”.

Product Perspective

The product will be built on the already existing *NSW National Parks* application. The service accepts data input from other hardware: the existing fleet management tracking system used with *NSW National Parks*.

The product shall have three software user interfaces: one for each type of user. The product shall have a communications interface between rangers and administrators in their respective parks.

User Characteristics

There will be three main types of users who will use the product: regular customers, rangers, and administrators. The following section describes the characteristics of each type of user from the perspective of the service.

In the context of the system, a ranger is a qualified person that is able to pick up and remove litter safely. Different rangers will be responsible for different areas of a national park. A ranger may or may not have access to a personal vehicle, so the software will need to accommodate for both situations.

An administrator is a person from NPWS who will be responsible for allocating tasks between multiple rangers in a park. They will also need to be responsible for adding new resources into the system when required, such as equipment.

A visitor can be anyone visiting a national park, including a ranger or an administrator. They may want to send a report when litter is found in the park.

Product Functions

The service will allow any visitor to describe the type of litter, an estimated amount of litter, and other information. When the visitor sends the report, the date and location will be recorded automatically. The system will also determine if the litter is accessible by a vehicle or not.

The system will then send this information to NWPS, who can then direct a nearby ranger to remove the litter. Rangers will have their own user interface, allowing access to nearby reports of litter from customers. They will need to specify whether they have access to a vehicle or not. The service will allow the ranger to write a 'thank-you' note and possibly allow the visitor to have a free coupon if the ranger decides to. More detail of the product functionality is described in *Specific Requirements*, on page 8.

Constraints

The service in development has a set of limitations that must be considered. Limitations not imposed by the customer to the project are described as follows.

CONSTRAINT ID	CONSTRAINT
C1	From the visitor's perspective, the hardware of the product is limited to users' phones
C2	The user interfaces can only be connected to the interface of the *NSW National Parks Application*
C3	All interfaces must have aesthetics similar to/based on the *NSW National Parks* app
C4	The service must comply with the policies of *NSW National Parks*

Assumptions and Dependencies

The software will require a certain number of assumptions and dependencies prior to its commencement. A list of important assumptions are shown in the table below.

ASSUMPTION ID	ASSUMPTION
A1	All park visitors have access to a personal phone with Internet
A2	All park visitors have the app downloaded and are signed in
A3	All parks have rangers and administrators available to work

A set of dependencies are shown in the table below.

DEPENDENCY ID	DEPENDENCY
D1	The product depends on the app itself to run
D2	The product depends on the GPS to find the user's location
D3	The app must have an internet connection
D4	The app is intended to use the fleet tracking systems to choose rangers for the job

SPECIFIC REQUIREMENTS

The following table shows the specific mandatory requirements outlined for the project. This includes the different requirement types and the fit criteria for each one.

NUMBER	TYPE	REQUIREMENT	FIT CRITERIA
1	**Functional**	Have an account sign-up user interface	The account sign-up interface must appear when called in 90% of test cases
2	**Functional**	Have an account log-in user interface	The account log-in interface must appear when called in 90% of test cases
3	**Functional**	Allow any user to create an account	100% of test cases must result in successfully creating account if the username is available
4	**Functional**	Allow an administrator to specify their national park location	The location of the national park must be correct and stored correctly in 100% of test cases
5	**Functional**	Allow any user to log-in	Provided the details are correct, the user must be able to sign into their account in 100% of test cases
6	**Functional**	Have three user interfaces for the three types of users of the system	The correct type of user interface must appear when called in 90% of test cases
7	**Functional**	Have a litter-reporting interface	The litter-reporting interface must appear in 95% of test cases when called for
8	**Functional**	Allow a customer to take a photo of their sighting of litter	100% of photos captured in test cases must be processed and uploaded correctly
9	**Functional**	Allow a customer to enter the quantity of litter	The system must only accept a reasonable numerical value as input in 100% of test cases
10	**Functional**	Allow a customer to enter a danger/hazard rating for the litter	The system must only allow one numerical selection as valid input in 100% of test cases
11	**Functional**	Record the date and time of the report	The date and time recorded must be correct in 100% of test cases
12	**Functional**	Record the park and location of the report	The location of the report must be 95% accurate in 100% of test cases

13	**Functional**	Identify the area of the park based on the location of the report	The park area must be 99% accurate in all test cases
14	**Functional**	Determine whether the report location can be travelled to by vehicle or not	The result must be 85% accurate in all test cases
15	**Functional**	Update the administrator's list of tasks	100% of uploaded tasks must be received and added to the appropriate task list
16	**Functional**	Allow an administrator to record all equipment in stock	100% of uploaded equipment must appear in the database
17	**Functional**	Allow an administrator to block 'fake' tasks from unreliable customers	100% of tasks must be successfully blocked by the administrator
18	**Functional**	Send a message back to the visitor stating the report cannot be used	100% of the messages must be received by the same visitor
19	**Functional**	Allow an administrator to allocate specific rangers to specific areas in the park	Data entered by the user must be 100% correct in the system
20	**Functional**	Inform an administrator of a suggested number of people required for a task	Accuracy for the recommendation must be at least 75% in all test cases
21	**Functional**	Allow an administrator to allocate a task to a specific ranger	100% of task allocations must be successful – appearing as a notification on the ranger's end
22	**Functional**	Allow an administrator to change a customer's account type to a ranger and vice versa	The success rate of all account type changing tests must be 100%
23	**Functional**	Allow a ranger to specify if they have a vehicle and which type	100% of vehicle-related data must appear in the ranger's relative park's database
24	**Functional**	Alert a ranger of a task required	The ranger must receive the notification in 100% of test cases
25	**Functional**	Inform a ranger of the PPE required for the task	The ranger must be sent a list of PPE in 100% of test cases
26	**Functional**	Allow a ranger to confirm the completion of a task	100% of task completion checks must be uploaded to the database
27	**Functional**	Allow a ranger to write a 'thank-you' note to the customer	100% of 'thank-you' messages must be delivered to the customer and stored in the database
28	**Functional**	Allow a ranger to give a customer a coupon if they wish to	100% of coupons sent out by rangers must be received by customers
29	**Non-Functional**	Security - keep all user data secure and confidential	There must be 0 reports of data breaches throughout the six months of operation
30	**Non-Functional**	Speed – operate fast enough so there are no delays	Interface-switching time must be less than two seconds in all test cases
31	**Non-Functional**	Compatibility – the product must be compatible with any type of phone	The service must be compatible with at least common 10 smartphone brands
32	**Non-Functional**	Availability – the system should be available for any user at any time they intend to use it	In 98% of all test cases, the system must be available to all end users
33	**Non-Functional**	Localisation – the system should accommodate for users who may speak different languages	In 95% of test cases, the translation between languages must be fully correct

The table below shows specific optional requirements recommended (but not mandatory) to include in the project.

NUMBER	TYPE	REQUIREMENT	FIT CRITERIA
1	**Functional**	Locate all the nearby bins for a visitor/ranger	All bins must appear on the visitor's user interface in 90% of test cases
2	**Functional**	Allow a customer to select a litter type from a list of options	All pre-written litter types must be present in 100% of test cases
3	**Functional**	Allow an existing user (e.g., a ranger) to merge their existing *NSW National Parks* account with the new service	In all test cases, account-merging processes must be 100% successful
4	**Functional**	Allow a visitor to save a report and then close the report interface or app	99% of reports must be saved and are able to be re-opened in all test cases

[end of Software Requirements Specification]

USE CASE DIAGRAM

Use case diagrams show the interactions between the main actors of system and the various scenarios on which each actor uses the system.

The following is a use case diagram, showing the system needs from multiple perspectives. These perspectives are from the regular customer, the ranger, and the administrator.

The use case stating, “maintain/update resource list” means that administrators should be able to update their litter-removal resources in stock. It also means administrators can add/remove sections of the park on the database.

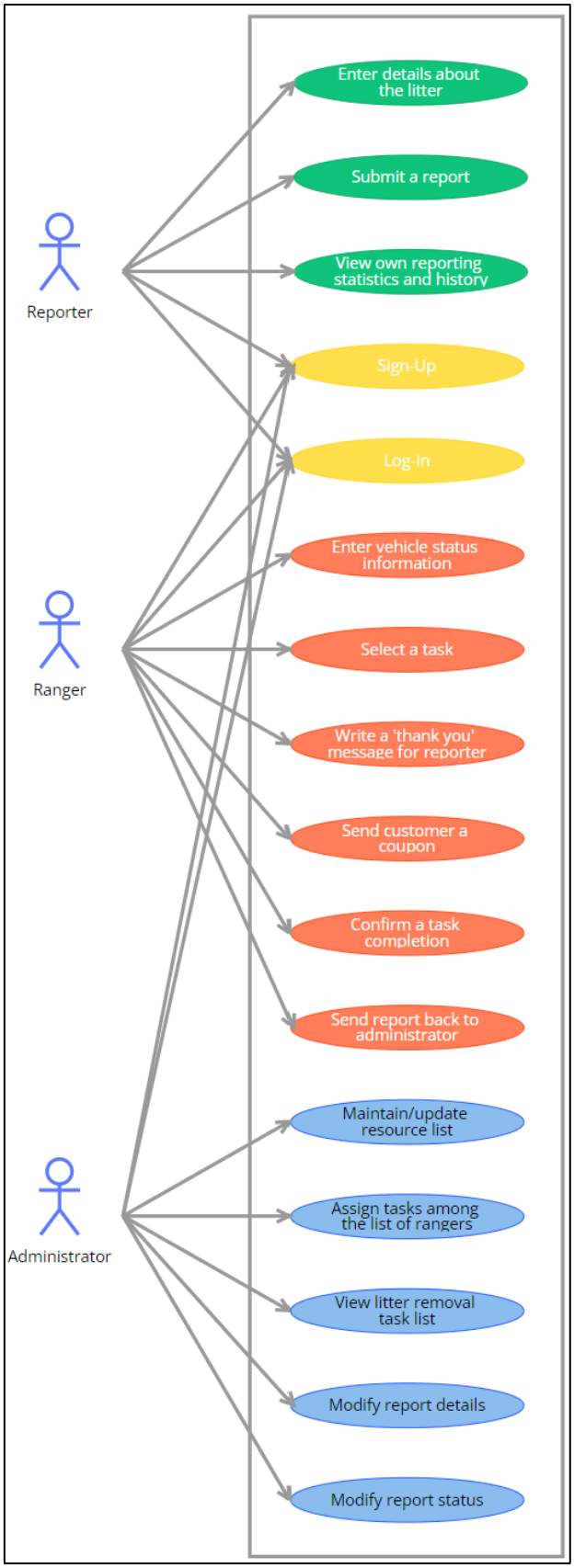

FIGURE 2

Use Case One

Use Case	**Entering details about the litter**
Goal	**A customer is able to successfully enter all relevant information into the system and the report is uploaded to the administrator database**
Preconditions	**The customer has the app downloaded, is signed in and the service is running without issues**
Success End Condition	**The goal is achieved**
Failed End Condition	**The reporting interface is closed and the report made by the user is not saved**
Primary Actors	**The customer reporting the litter**
Secondary Actors	**The system**
Trigger	**The customer selects the option on the service to report litter**
Description / Main Success Scenario	**1: The customer takes a photo of the litter photo, quantity, hazard rating** **2: The customer enters the approximate quantity of litter** **3: The customer gives the litter a hazard rating and then submits the report** **4: The system takes a snapshot of other relevant information** **5: The system sends the report to the administrator database**
Alternative Flow: Cancelling a report	**1: The customer may/may not enter certain information** **2: The customer clicks the cancel button** **3: The customer is brought back to their main user interface**

Use Case One Description

In the first use case, any visitor user will be able to report litter by entering all relevant details into the system. The primary mandatory requirement in this case is to allow a visitor to record a sighting of litter and upload it to the respective park's task list. Without implementation of this requirement, the service as a whole fails to carry out its objective.

It is also mandatory for the visitor to be able to enter a range of relevant supporting information. This should include an approximate quantity of litter, an estimated hazard rating, and the type of litter. The type of litter can be categorised by the user with a list of pre-set litter types. This can range from regular rubbish to dead animals to fallen trees. However, if no options best suit the type of litter, then there should be an option for the user to describe specifically what it is. After this information is entered into the report by the user, they should also be able to write any other information they want the administrator/ranger to know about the litter.

Based on the information sent, it can greatly help administrators with assigning tasks accordingly. Their judgements can be based on the equipment readily available (including PPE), the group of rangers located closest to the report location based on fleet vehicle tracking, and the rangers' vehicle types (if any).

Use Case Two

Use Case	**Signing up to use the service**
Goal	**A user is able to register an account and choose whether they are a customer, ranger, or administrator. Verification will be needed if 'ranger' or 'administrator' is selected**
Preconditions	**The user has downloaded the app**
Success End Condition	**The goal is achieved**
Failed End Condition	**An error occurs, and the user account is not created**
Primary Actors	**The user intending to register a new account for the service**
Secondary Actors	**The system**
Trigger	**The user selects the sign-up user interface**
Description / Main Success Scenario	**1: The user enters their desired username** **2: If the username is available, the user enters their password** **3: The user enters their email address and/or phone number** **4: The user confirms their information and signs up**
Alternative Flow: Username unavailable	**1: The user enters their desired username** **2: The username is unavailable, and the system asks the user to try another username** **3: The process repeats until an available username is found**

Use Case Two Description

In the second use case, a user should be able to create a new account by signing up. They can specify what type of user they will be. However, if they choose either a ranger or administrator for a specific park, they will need verification.

It is recommended that users can simultaneously merge their existing *NSW National Parks* account if the customer already has an account. However, if the user chooses not to, they can create an account by entering their preferred username and a password consisting of characters, uppercase, lowercase, and numerical characters.

Use Case Three

Use Case	**Assigning tasks among a list of rangers**
Goal	**An administrator is able to successfully send tasks/reports to individuals or groups of rangers in their respective national parks**
Preconditions	**The administrator/s have installed the service, have their database running successfully and have a minimum of one task ready to be assigned**
Success End Condition	**The goal is achieved**
Failed End Condition	**The tasks are not sent out and still remain in the database**
Primary Actors	**The administrator sending the task/s**
Secondary Actors	**The system**
Trigger	**The administrator opens their personalised administrator interface**
Description / Main Success Scenario	**1: The administrator looks at the tasks stored in the database** **2: The administrator selects one or more tasks** **3: The administrator selects one or more park rangers to assign the task/s to** **4: The administrator confirms their selection and sends the tasks out to the rangers**
Alternative Flow: Cancelling	**1: The administrator looks at the tasks stored in the database** **2: The administrator may/may not select tasks** **3: The administrator clicks the 'cancel' button, and the tasks remain unresolved in the database**

Use Case Three Description

In case three, the system administrator assigns a set of tasks (which are reports from customers) to specific rangers in specific park areas. This interface should only be accessed by registered administrators of national parks.

Once a report is made from a user, the list of tasks will be updated with the new task added. Administrators can view these reports and make judgements about which tasks to assign to specific rangers. There should be an option for an administrator to easily remove reports from the list if visitors choose to misuse the service.

There can be an option to send a team of rangers to complete a task if the amount of litter is in large quantities or the object is large in size or weight. It is important for administrators to determine which tasks go to which people so that priorities can be made if there are high levels of danger and rangers avoid these tasks.

The administrator will know which rangers are closest to the report based on the trackers on the fleet tracking data.

SEQUENCE (INTERACTION) DIAGRAM

The following is a sequence diagram (also called an *interaction* diagram). The use case selected to be chosen for this sequence diagram is *use case one* (the litter information-entering use case), as it is the most important.

The diagram visualises the interactions between three participants in the system. These are the customer, the system and the NPWS administrator. The system allows the customer to enter in any relevant information and then sends it on to the park administrator/s.

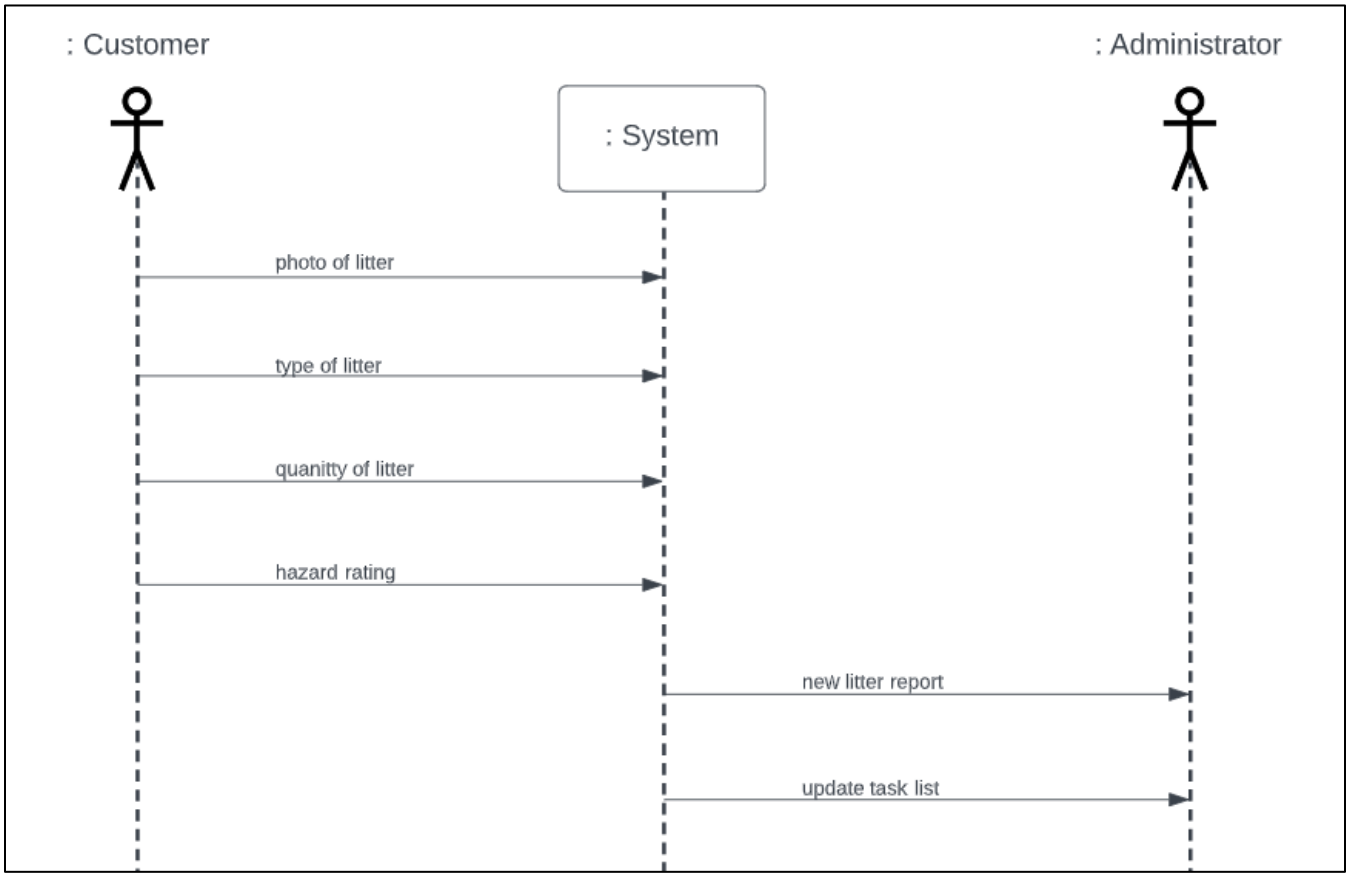

FIGURE 3

PRIORITISATION EXPLANATION

It was important to determine the most important use cases. This was done by observing all use cases from the diagram and writing down the use cases that would prevent the system from operating if removed from the diagram. When this was done, it was found that the three most important use cases were the following:

- Entering details about the litter
- Signing up and logging into the service
- Assigning tasks among the list of rangers on the system

REPORT - TECHNIQUES

In order to obtain clear requirements for the project, various techniques needed to be applied. These techniques included holding regular meetings/interviews/user stories, brainstorming, scenario role-playing and questionaries. The following section will describe how these techniques were used to identify and detail the requirements gathered from the client.

The requirements-elicitation process is more effective when numerous clients of different types are interviewed. This is because feedback from all types of users of the system can be received. However, due to project constraints this could not be changed. Instead, only one client could be interviewed to elicit the user requirements.

Initially, regular interviews and meetings were organised with the client. The interviews which took place were in the form of user stories. Role playing was also carried out This technique was applied in order to discover the features the client representative most desired to include in the system.

After each meeting with the client, his ideas of features were taken into consideration and analysed. Brainstorming sessions were then performed to consider as many different use case scenarios as possible considering the user stories. After the use cases were written, a set of requirements were written for the use cases.

The requirements were then split into two categories: ones that were mandatory and optional. Some of the requirements were good, but beyond the scope and current needs of the client. If they were implemented in the initial release, they might cause scope creep and unnecessary delays to launch the system. Therefore, they have been suggested for a second version.

After a few meetings with the client, a questionnaire was written for the client in order to further develop the requirements. The purpose of the questionnaire was to verify the features discussed by the client, and to allow a choice of a basic graphical user interface that may be changed further in development.

After several meetings with the client, the final meeting for the requirements of first version of the system was held. During the meeting, the client representative was asked to prioritise and rank all of the written requirements, finalising the scope to avoid scope creep. Then, the verification process occurred, and the customer signed-off to confirm the requirements.

In summary, multiple techniques and strategies were applied in order to elicit the client's desired features into functional and non-functional requirements. Firstly, regular meetings were established and carried out. User stories were used so that the client could clearly describe the essential system features. Brainstorming sessions were done to consider as many use case scenarios as possible, considering the desired features. Questionnaires were given to the client to verify each of the requirements written.

REPORT – THE CLIENT

The client representative of the project is Rumeth Herath. It was essential to work closely with him so that the requirements defined would be able to best suit his needs. This section will describe a summary of the experience when working with the client, including both the main positive and negative aspects.

During the interview and meeting stages, it was clear that Rumeth put a lot of thought into what features he wanted the system to have. His needs for the system, as well as desired features for future versions, were stated explicitly and plainly most of the time. As a software engineer, it was relatively easy to translate the needs into a set of functional and non-functional requirements.

During the last meeting when the final requirements were discussed, it appeared to be difficult for Rumeth to prioritise the written list of requirements. The list of requirements can be seen to be considerably extensive and proved to be difficult to compare some. In only a few times, he was also relatively vague when discussing his needs for the system and lacking in detail.

Overall, it was fairly easy to negotiate and compromise with Rumeth. A few features that he suggested would not work due to different reasons. The software engineer put forward requirements that best work with the intended features. He explained to Rumeth why the features would not work, and he was willing to accept the proposed alternative solutions.

In summary, it was a pleasant experience working with the client representative of the project. Rumeth played a significantly important role in the requirement-gathering process for the system. However, he could have improved by providing more detailed user stories, so that the software engineer could transform them into more suitable requirements for his needs.

PLAN TO FURTHER DEVELOP REQUIREMENTS

If the development of the project was to be continued, the existing requirements would need to be further developed and refined in order to achieve a project with features that all users are satisfied with. The following section describes a fundamental plan to continue building the requirements in the future.

Firstly, client representatives of each user type will be found. Regular meetings will be scheduled with them to discuss a range of information about the project. The purpose of these meetings will not be to discuss any 'new' features. The time period for gathering features is finished for the first version of the project. The purpose of these future meetings is to break down the current requirements into as much detail as possible. In this way, it can be ensured that the 'right' requirements are being implemented.

The requirements will be broken down from their current form by asking specific questions and ensuring specific responses are given in return. It is inevitable that the clients will mention new features during these meetings, which should be welcomed but not be focussed on at this point in time. This is because scope creep will occur, causing delay in the release of mandatory functionality. Instead, their ideas will be recorded and be considered future versions of the product.

After the requirements have been broken down into as much detail as possible, the user interfaces will be discussed with the clients. The software engineer will use wireframing tools to design multiple types of each different user interface. Each interface will have different functionality depending on the user type. For example, visitors will be able to see their reporting history and make a new report. The ranger interface will show what tasks they have available, their past reports, their specified vehicle type, their park, and park area. Surveys will be taken so all clients of each type can decide their preferred user interface out of various options.

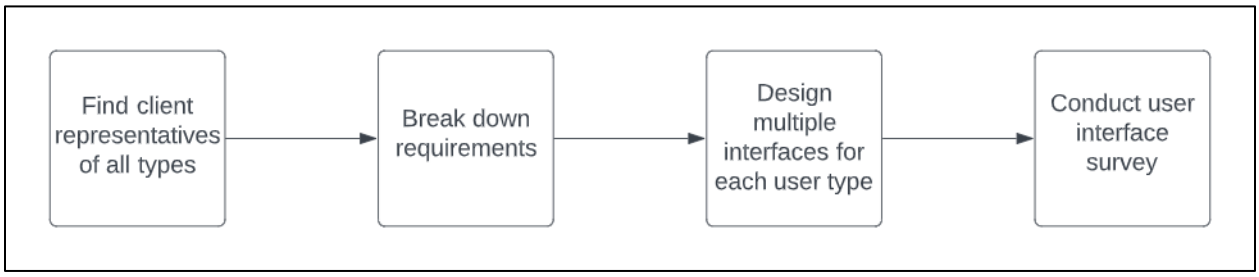

FIGURE 4

REPORT FOR PROJECT B

As part of the assessment for COMP2050, it was required for the software engineer (Ben Cavanagh) to also become the client representative of another project, while the client representative of this project (Rumeth Herath) was the software engineer of the other project. This section will describe the extent of how much an impact was made in being a client for the same person.

The software engineer believed that overall, he did well in expressing his needs for the bicycle-sharing app. In his meetings with Rumeth, he was able to describe in detail what features were the most important to him in order to meet his needs. He was particularly proud of imagining various scenarios where other people may want to use the app for different purposes. He was also able to describe a range of optional features that would be highly beneficial to include in the app.

However, just as Rumeth found it difficult to prioritise the requirements for his system, the client representative of Project B was also challenged in prioritising the list of requirements for his system. After the final meeting for Project B, he believed he could have performed as a client better if he gave more detailed feedback, and focussing less on the technical work, as this is for the software engineer of Project B.

It was an enjoyable experience to be the client representative of another person's project. The client representative of Project B performed well in his role. He gave Rumeth (the software engineer) a range of ideas about the project's features and uses. However, he could have performed better as a client if he had given more descriptive feedback.

LOG OF INTERACTIONS

The following table describes every interaction that happened between the software engineer and the client representative. The date and time, duration, topic, and the main discussions were recorded.

There were seven sessions between the software engineer and the client representative. During the same time period, the same topics were discussed for Project B, where the software engineer and client roles were reversed.

DATE AND TIME	DURATION	TOPICS	MAIN DISCUSSIONS
10/8/22 – 5pm	2 hours	- System goal definition	The overall goal of the system, planning the regular meeting dates and times
13/8/22 – 12pm	2 hours	- Main system features	The most important features the system should include
17/8/22 – 5pm	2 hours	- Visitors	The specific system features that are focussed on the visitor users
20/8/22 – 12pm	2 hours	- Rangers	The features of the system that focus on the rangers
24/8/22 – 5pm	2 hours	- Administrators	The features of the system that focus on the administrator users
27/8/22 – 12pm	2 hours	- Interfaces	The general idea of the interfaces and basic designs, the questionnaire
31/8/22 – 5pm	3 hours	- Prioritising requirements - Validation of requirements - Scope sign-off	Ranking the requirements created, agreeing on the scope of the project, and signing off