CARLETON UNIVERSITY

cuACS

Requirements Analysis Document

Team QuackJaws
Jake Bauer
Ashlee Foureyes
Skyler Gubbels
Will Watt

Submitted to:
Dr. Christine Laurendeau
COMP3004 – Object-Oriented Software Engineering
School of Computer Science
Carleton University
February 12, 2019

Contents

1. Introduction

1.1 Purpose of the Carleton University Animal Care System (cuACS)

Animal shelters are tasked with the responsibility of providing care to animals in need while they await adoption into a loving home. Humans that are seeking the companionship of a pet can visit a shelter and choose an animal to adopt. However, with this process an issue often arises where a pet is adopted by a human with whom they are not fully compatible. This mismatch can be a result of a variety of reasons, including temperament, conflicting lifestyle requirements, and the physical and non-physical needs of both the pet and the potential adopter.

Carleton University Animal Care System (cuACS) aims to alleviate the issue of mismatching by providing a tool that automatically matches a pet to a potential owner based on compatibility. This compatibility measurement is based on matching the numerous physical and non-physical

traits applicable to the animals in the shelter as well as the traits and desires of potential owners. cuACS enables a smooth adoption process, ensures the experience will prove positive and match an animal which both fulfills a clients expectations and suits their lifestyle.

1.2 Document Overview

This document will outline and describe the functional and non-functional requirements of cuACS and the system models used to design the system. The purpose of this document is to organize and present the analyzed requirements and various models which will be used in the designing and building of the system.

The functional requirements will describe the interactions between the system and its environment independent of its implementation. These are presented in a table of numbered requirements accompanied with descriptions of the functionality required.

The non-functional requirements describe the components of the system that are not directly related to its functional behavior. This includes operability, reliability and interface requirements, among others. These are presented in the same way as the functional requirements.

Furthermore, this document will present an in-depth look into the functional requirements by providing a use case model and an object model in the form of a data dictionary with an accompanying class diagram. The use case model will show the potential ways that the system will be used and interacted with by those who would use the system. The data dictionary and class diagram which make up the object model will show the relationships between the high-level concepts or components manipulated by the system. These models break down the individual pieces which make up the system as a whole and detail them so that the purpose of each is clear.

Finally, there is an index of all figures and tables as well as a glossary of application domain-specific terms used in this document at the end for ease of navigation and convenience.

2. Proposed System

The cuACS system is made up of several interacting components. This section covers the functional requirements, non-functional requirements, and the models used to express the design and functionality of the system.

2.1 Overview

This section aims to detail first the requirements which led to the design of the system as a whole and then will detail the various components which will make up the system. First, there is a detailed breakdown of the functional requirements of the system then the non-functional requirements. Following that there is a section detailing the various use cases of the system including use case diagrams and table-based descriptions of each use case. Finally, there is an object model with a data dictionary and class diagram to explain the high level concepts which are manipulated by the system such as “Animal” and “Storage.”

2.2 Functional Requirements

Functional requirements describe the interactions between the system and its environment independent of its implementation. They describe what the system must be designed to do in the sense of defining the functionality it must have once it is implemented yet regardless of how it is implemented. The numbering scheme for functional requirements follows a pattern of identifying overall, major features with an FR-x.0.0 designation, with sub-requirements and more specific requirements being assigned number like FR-x.x.0 and FR-x.x.x respectively.

Table 1—Functional Requirements

Identifier	Description
FR-1.0.0	Staff must be able to view and manage the detailed information about all animals available for adoption and Clients must be able to view a list and the information of all animals for adoption.
FR-1.1.0	Staff must be able to add a new animal, being able to input detailed profile information when the animal is created. Detailed profile information includes the below in addition to the Name and ID of the animal.
FR-1.1.1	Physical traits for animals must be: • Species (Dog/Cat) • Breed • Size • Health (vaccination status, conditions, etc) • Age • Gender • Spayed/Neutered Status • Fur Length • Hypoallergenic Status
FR-1.1.2	Non-physical traits for animals (some may be applicable to clients as well) must be: • Easy to travel with • Good with children • Good with other animals • Level of energy

	• Useful for defending a home/property • Good with strangers • Good in crowds • Sleeping patterns (e.g. nocturnal) • Level of affection • Good with loud noises • Messiness • Previous home (e.g. street rescue) • Indoor or outdoor animal • Level of exercise needed
FR-1.2.0	Staff must be able to see a list of all animals available for adoption.
FR-1.2.1	Staff must be able to see detailed information about a selected animal.
FR-1.2.2	Clients must be able to see a list of all animals available for adoption.
FR-1.2.3	Clients must be able to see detailed information about a selected animal.
FR-1.3.0	Staff must be able to edit a chosen animal's information.
FR-2.0.0	Clients must be able to be added to the system and their information must be editable by both staff and the respective client.
FR-2.1.0	Staff must be able to add new clients to the system, creating their profile with basic contact information.
FR-2.1.1	A client's profile must consist of: Basic contact information, the client's personal information regarding their attributes/personality, and their matching preferences. The client will also be assigned an ID by the system which they cannot change.
FR-2.1.2	A client's contact information must consist of: • Full name • Full mailing address • Phone number • Email address
FR-2.1.3	A client's personal information must consist of: • Whether they have children • Whether they have other pets • Whether they travel a lot • Whether they exercise a lot • Whether they go out to crowded places often • What their typical sleeping patterns are like • Whether ot not they tend to make loud noises • Whether or not they are messy
FR-2.1.4	A client must be able to look at all of the physical and non-physical attributes of the animals and select which ones they absolutely must have, which ones they would prefer to have, which ones they feel neutral about, which ones they would rather not have, and which ones they absolutely will not accept. This makes up their matching preferences.

FR-2.2.0	Staff must be able to view a list of all clients.
FR-2.2.1	Staff must be able to view the detailed information of a selected client.
FR-2.3.0	Staff must be able to edit a chosen client's information.
FR-2.3.1	Clients must be able to view their own profile.
FR-2.3.2	Clients must be able to change the information on their own profile.
FR-3.0.0	The algorithm must be able to compute the optimal match between all clients and all animals and then output the list of matches with the ability to display detailed information as to why it arrived at that result. It must take into account the best interests of all animals in the shelter rather than producing a small set of optimal matches and a large set of sub-optimal matches.
FR-3.1.0	The algorithm must take into account both the full profiles of the clients and the full profiles of the animals when making matches.
FR-3.2.0	Once the algorithm has finished running, the system must display to staff a list of matches made. Each item in this list must include the names of the client and animal matched.
FR-3.3.0	When a staff member clicks on an item in the list of matches, a detailed view must appear showing the details of why the specific match was made. It must specify the exact rules used to compute the match and give data showing how/why the match was computed.
FR-3.4.0	A staff member must be able to launch an instance of the algorithm to compute matches.
FR-4.0.0	All animal and client data must be stored in a flat file system consisting of JSON or XML formatted data files. No database software will be used.

2.3 Non-Functional Requirements

Non-functional requirements describe the components of the system that are not directly related to its functional behaviour. This includes implementation requirements that define standards for system reliability, usability, performance, and more. These are all requirements that need to be taken into account when designing the system but do not directly influence what functionality the system is supposed to implement.

Table 2—Non-Functional Requirements

Identifier	Category	Description
NFR-1	Usability	The menu navigation system of the system must be easy to understand and intuitive to follow. It should use standard menus such as File, Edit, View, etc. As well as custom ones tailored to the program.
NFR-2	Usability	The function performed with every mouse click must be intuitive and easy to understand for a typical inexperienced home user. A home user should only have to search the help guide out of desire

		to see what the system can do and not necessity.
NFR-3	Usability	While the ACM algorithm is being run, a staff member or client should be able to do other tasks on the system. The algorithm should not lock up the program.
NFR-4	Reliability	The system may not be down for more than one (1) hour per week. This includes all updates and patches to the system which must be carried out as part of the normal operation of the system.
NFR-5	Reliability	The system must be able to cleanly recover from errors, display a message to the user who caused the error, and must not crash as a result of normal operation.
NFR-6	Reliability	The system must make backup copies of the data once every hour and must store all backup copies for the previous 24 hours, plus store one backup copy per day older than 24 hours up to one month previous.
NFR-7	Performance	The ACM algorithm should be able to match one hundred (100) clients with one hundred (100) animals in under one minute at least eighty-five (85) percent of the time.
NFR-8	Performance	The program should take no more than ten (10) minutes to launch under normal circumstances. The program can be considered launched when a staff member or client can interact with the system.
NFR-9	Performance	The system should be able to save and load client and animal profiles from the file-system without locking up the rest of the program.
NFR-10	Supportability	The cuACS system developers should provide, for two (2) years after the final delivery date of the product, support by phone, email, and in-person assistance for the system.
NFR-11	Supportability	Following the two (2) year support window, the software should be made available to be supported for additional time by the current maintainers or by a new company or by in-house IT staff.
NFR-12	Supportability	The system should be able to be scaled in the future to handle up to 10,000 clients and 10,000 animals.
NFR-13	Implementation	The language which must be used for the system's implementation is C++.
NFR-14	Implementation	The GUI of the system must be implemented using Qt.
NFR-15	Implementation	All persistent data must be stored in either a flat file system or using SQLite and must be loaded into the system on launch. If data is stored in a flat file system, it can either be stored in binary format or in plain text.
NFR-16	Implementation	All persistent data must be loaded into the system on launch.
NFR-17	Interface	Each major part of the application should be presented in its own

		window or tab within one window. For example, functionality for managing clients should not be in the same tab as the output results for the matching algorithm.
NFR-18	Interface	The interface for the system should be a professional-looking graphical user interface. It should not be a console-based interface.
NFR-19	Interface	Pop-up menus should be presented to the user when a significant process has been completed such as the completion of a run of the ACM algorithm.
NFR-20	Operations	The IT department of the shelter should be notified of system updates via e-mail.
NFR-21	Operations	System updates should be delivered to the IT department of the shelter via an online download portal. The IT department should be able to visit the cuACS developers' website to download the updates.
NFR-22	Operations	A shelter's IT department should be the managers of the system. They host the system using their resources, maintain, and update it using their staff.
NFR-23	Operations	The system should be deployed onto a single host computer in production.
NFR-24	Packaging	The system must come packaged with all necessary documentation to allow the IT department of the shelter to install it easily. It must also come packaged with a user help guide for staff and clients of the shelter.
NFR-25	Packaging	The system must come packaged as a binary file to be installed by the installation script. No compilation should be necessary on the part of the end-user.
NFR-26	Packaging	The system must come packaged with an easy installation script which will install the program by running a single command.
NFR-27	Packaging	The system must come pre-packaged with 25 complete animal profiles and 20 complete client profiles as sample profiles.
NFR-28	Legal	The source code for the system may not be open-sourced as the client wishes to keep the system private from competing shelters.
NFR-29	Legal	The system should comply with the local region's accessibility standards (e.g. Ontario's O. Reg. 191/11: INTEGRATED ACCESSIBILITY STANDARDS)
NFR-30	Legal	The system should be regional privacy law compliant. It should aim to comply with the GDPR should a similar law be enacted in the client's region. It should not store users' personal information in an insecure way and should handle it according to the laws of the country or region in which the software is operating.

2.4 System Models

System models are ways of representing the different functionality of a system in different ways. In this section, a use case model and an object model are provided. The purpose of the use case model is to enumerate the different ways in which external actors will interact with the cuACS system. The purpose of the object model is to design and specify the different major components which will make up the system's functionality.

2.4.1 Use Case Model

A use case model is a model which describes the different use cases of a system. That is to say, it describes the different ways that users of the system can interact with the system. It answers the question: "What are the things that a user can accomplish using this system?"

This section is organized into two main categories. The first category is made of of the high level use cases which describe the general functionality that the system provides to end-users. The second category is comprised of the detailed use cases which take the uses described by the high level use cases and break those down further to get a more specific flow of events. Describing both categories are a list of descriptions of the use cases, diagrams detailing the flow of events in the use cases more specifically, and table-based descriptions which provide a text-based rendition of the flow of events.

2.4.1.1 High Level Use Cases

The high level use cases are depicted in the diagram and descriptions below. These use cases cover the overall functionality of the system and putting them together provides a simple chart showing which end-users interact with which major parts of the system.

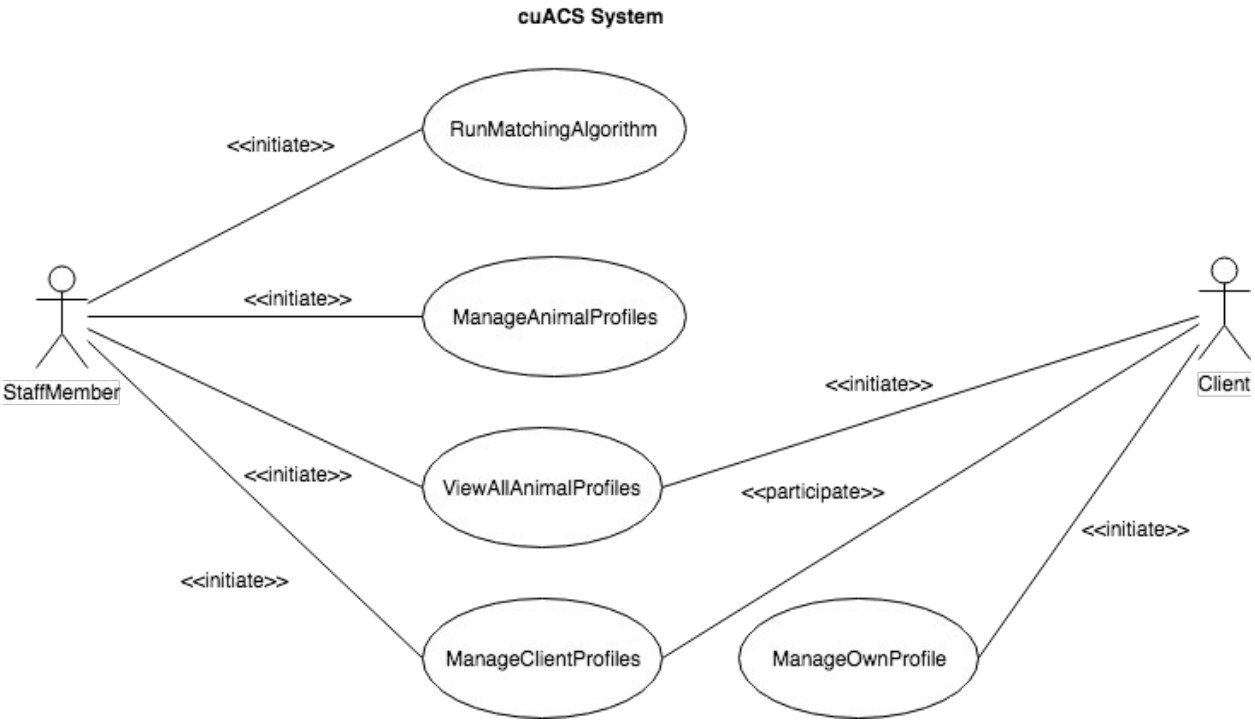

Figure 1—cuACS High Level Use Case Diagram

The following table provides a breakdown of each of the major use cases with a description that states the overall work that each use case is designed to accommodate.

Table 3—High Level Use Case Descriptions

UC-01	**RunMatchingAlgorithm**	StaffMembers launch an instance of the animal-client matching algorithm and view its output.
UC-02	**ManageAnimalProfiles**	StaffMembers view, add, and alter animal profiles.
UC-03	**ViewAllAnimalProfiles**	StaffMembers and Clients view a list of all animals which are registered in the shelter. They also view detailed profiles of selected animals.
UC-04	**ManageClientProfiles**	StaffMembers manage the profiles of Clients including adding new profiles and editing existing ones.
UC-05	**ManageOwnProfile**	Clients manage their own profile, adding and removing information as desired.

2.4.1.2 High Level Use Case Flow of Events

For the RunMatchingAlgorithm use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember is logged into the cuACS system, describes the process of a StaffMember running an instance of the matching algorithm, and then details that the two possible outputs are an error condition, MissingProfilesError, and a successful matching output.

Use Case Identifier	UC-01
Use Case Name	**RunMatchingAlgorithm**
Participating Actor(s)	StaffMember
Flow of Events	1. The StaffMember launches an instance of the ACM algorithm to match Clients and animals. 2. The StaffMember views a basic list of the matches resulting from running the algorithm. 3. The StaffMember chooses a desired match to view detailed information explaining why the match was made by the algorithm.
Entry Condition(s)	• The StaffMember is logged into cuACS.
Exit Condition(s)	• The matches are available to be seen by StaffMembers upon completion of a running instance of the algorithm OR • There is a MissingProfilesError which causes the operation to abort.
Quality Requirement(s)	• While the algorithm is being run, a progress bar must appear showing the progress of matching. • A running instance of the algorithm must happen in the background and must not lock up the program.
Traceability	FR-3.0.0, FR-3.1.0, FR-3.2.0, FR-3.4.0, NFR-3

For the ManageAnimalProfiles use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember is logged into the cuACS system, describes the different things that a StaffMember can do to manage animal profiles, and then details that the two possible overall outputs are an error condition, FileError, and a successful saving of the animal's profile to persistent storage.

Use Case Identifier	UC-02
Use Case Name	**ManageAnimalProfiles**
Participating Actor(s)	StaffMember
Flow of Events	1. StaffMember chooses whether or not to add or alter an existing animal

	profile 2. If the StaffMember chooses to add a new animal profile, they fill out the animal's information 3. If the StaffMember chooses to alter an existing animal's profile, they view the animal's profile and edit the fields they wish to change. 4. Both choices result in the StaffMember saving the new or altered animal profile to persistent storage.
Entry Condition(s)	• The StaffMember is logged into cuACS.
Exit Condition(s)	• The new or altered animal profile is saved to persistent storage OR • The system presents the StaffMember with a FileError, aborting the process and restoring the pre-change state of the system.
Quality Requirement(s)	-
Traceability	FR-1.1.0, FR-1.2.0, FR-1.3.0, NFR-5

For the ViewAllAnimalProfiles use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember or Client is logged into the cuACS system, details what they can see with regards to animal profiles, and details that the two possible overall outputs are an error condition, FileError, and a successful viewing of an animal's profile.

Use Case Identifier	UC-03
Use Case Name	**ViewAllAnimalProfiles**
Participating Actor(s)	Initiated by StaffMember OR Initiated by Client
Flow of Events	1. StaffMember or Client chooses to view all of the animal profiles by accessing the relevant part of the user interface. 2. The StaffMember or Client select a specific animal to view more detailed information about that animal.
Entry Condition(s)	• The StaffMember, or Client, is logged into cuACS.
Exit Condition(s)	• The profile is displayed to the initiator OR • The system presents the initiator with a FileError, aborting the process
Quality Requirement(s)	-
Traceability	FR-1.2.0, FR-1.2.1, FR-1.2.2, FR-1.2.3, NFR-5

For the ManageClientProfiles use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember is logged into the cuACS system, describes the different things that a StaffMember can do to manage client profiles, and then details that the two possible overall outputs are an error condition, FileError, and a successful saving of the client's profile to persistent storage.

Use Case Identifier	UC-04
Use Case Name	**ManageClientProfiles**
Participating Actor(s)	Initiated by StaffMember Communicates with Client
Flow of Events	1. A StaffMember views the list of all clients. 2. A StaffMember chooses to add or edit an existing Client profile. 3. If the staff member chooses to add a Client profile, they create one and fill out all of the basic information necessary for the profile to be created. 4. If the StaffMember chooses to edit a Client's profile, they choose the specific Client's profile. 5. Then the StaffMember edits the fields that need to be changed. 6. Finally, for both choices, the Client profile is saved to persistent storage. 7. The Client can now view and edit their new or altered profile.
Entry Condition(s)	• The StaffMember is logged into cuACS.
Exit Condition(s)	• The new or altered Client profile is saved to persistent storage.
Quality Requirement(s)	• A Client's profile data must be stored securely to comply with local privacy laws.
Traceability	FR-2.0.0, FR-FR-2.1.0, FR-2.2.0, FR-2.2.1, FR-2.3.0, NFR-30

For the ManageOwnProfile use case, the following shows the flow of events through this use case. This use case can only be invoked if a Client is logged into the cuACS system, describes the different things that a Client can do to manage their own profile, and then details that the two possible overall outputs are an error condition, FileError, and a successful saving of the Client's own modified profile to persistent storage.

Use Case Identifier	UC-05
Use Case Name	**ManageOwnProfile**
Participating Actor(s)	Initiated by Client
Flow of Events	1. A Client sees their own profile 2. The Client makes a change to one or more fields in their profile

	information. 3. The Client saves their updated profile to persistent storage.
Entry Condition(s)	• The Client is logged in to their cuACS account.
Exit Condition(s)	• The Client's updated profile is saved to persistent storage.
Quality Requirement(s)	• A Client's profile data must be stored securely to comply with local privacy laws.
Traceability	FR-2.3.1, FR-2.3.2, NFR-30

2.4.1.3 Detailed Use Cases

Detailed use cases provide a much finer-grained look at the use cases of the system. They allow a more microscopic look at the flow of events which are normally obscured in a high level use case. Below are the detailed use case diagrams which accompany each of the above high level use cases.

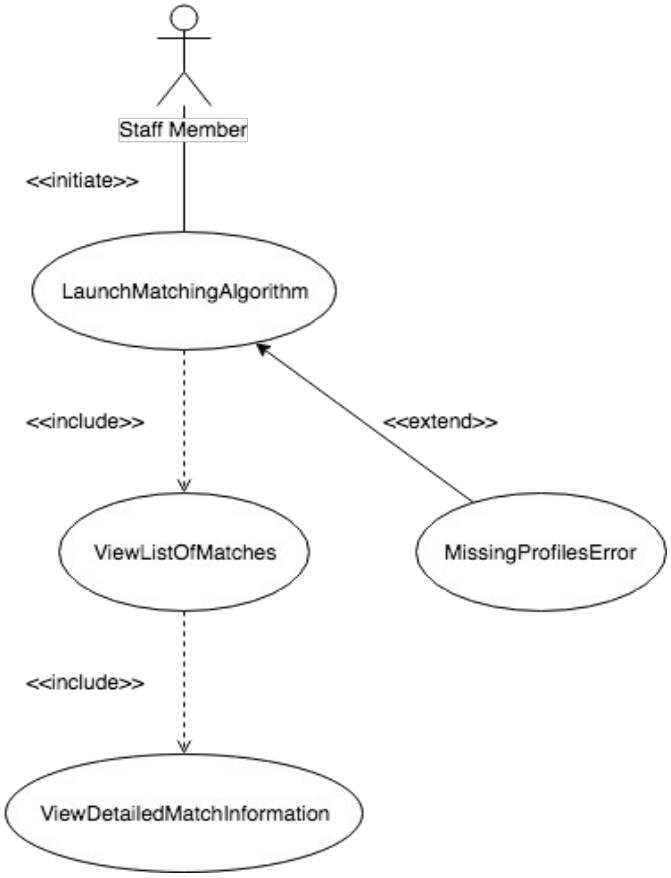

Figure 2—RunMatchingAlgorithm Detailed Use Case Diagram

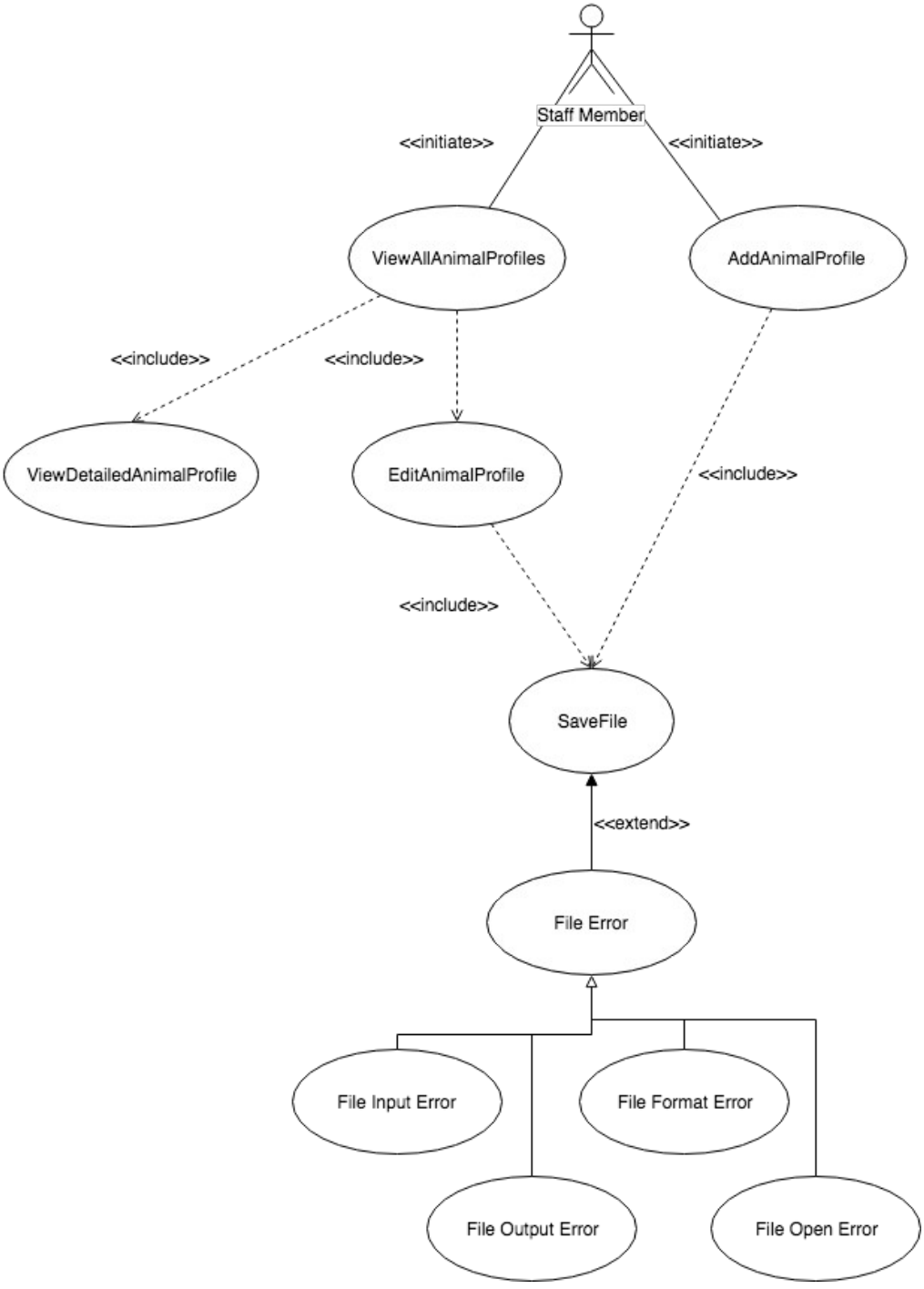

Figure 3—ManageAnimalProfiles Detailed Use Case Diagram, including ViewAllAnimalProfiles From the Perspective of a StaffMember

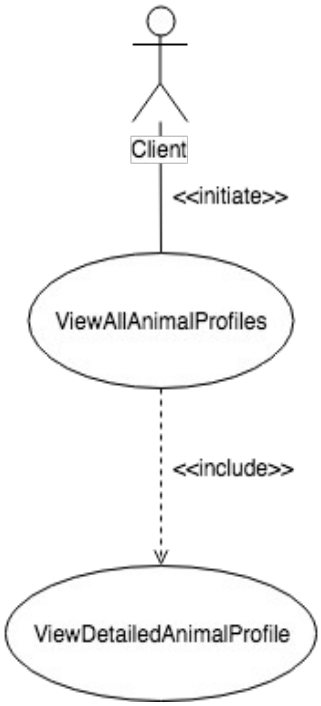

Figure 4—ViewAllAnimalProfiles From the Perspective of a Client Detailed Use Case Diagram

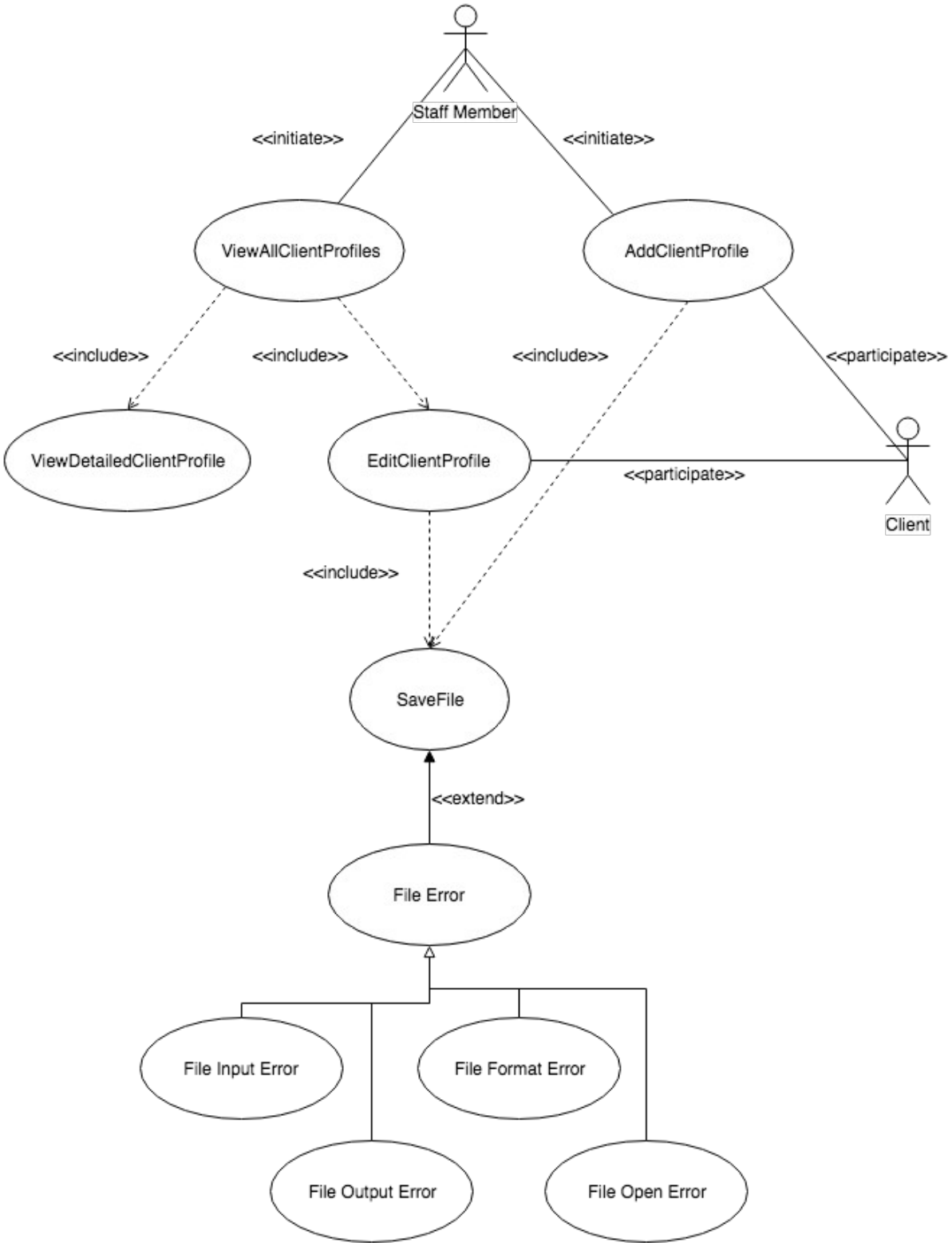

Figure 5—Manage Client Profiles Detailed Use Case Diagram

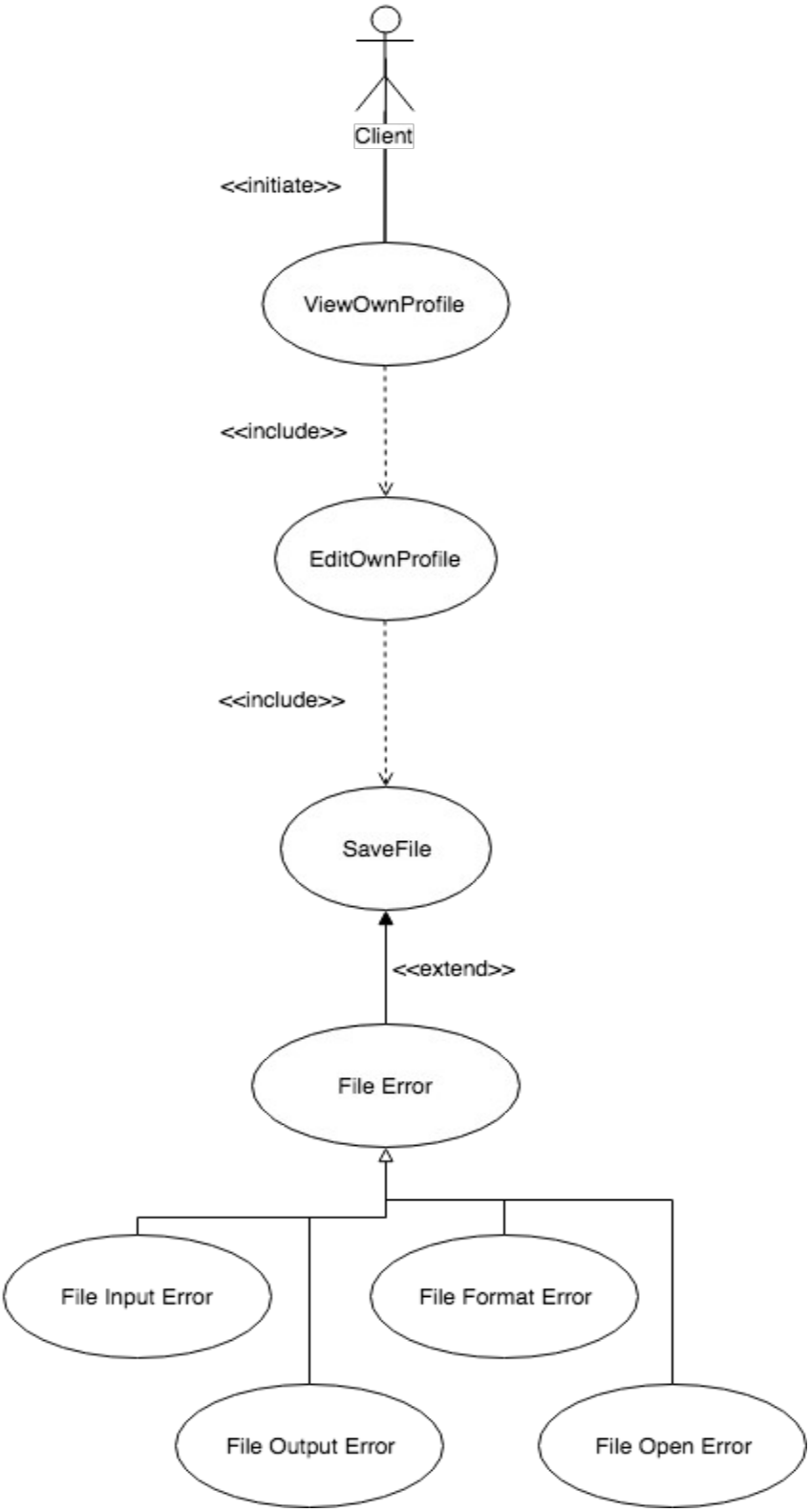

Figure 6—Manage Own Profile Detailed Use Case Diagram

The following table provides a breakdown of each of the detailed use cases represented in the above diagrams with a description that states the overall work that each use case is designed to accommodate.

Table 4—Detailed Use Case Descriptions

UC-06	**LaunchMatchingAlgorithm**	StaffMembers launch an instance of the animal Client matching algorith which runs in the background.
UC-07	**ViewListOfMatches**	StaffMembers see a list of matched Clients and animals output by the algorithm.
UC-08	**ViewDetailedMatchInformation**	StaffMembers select one match from the list and view the detailed output about why the match was made by the algorithm.
UC-09	**AddAnimalProfile**	A StaffMember adds a new animal profile, filling out all necessary information.
UC-10	**EditAnimalProfile**	A StaffMember edits an animal's profile, changing one or more fields.
UC-11	**AddClientProfile**	A StaffMember adds a new client profile, filling out all the necessary information.
UC-12	**EditClientProfile**	A StaffMember edits an existing client's profile
UC-13	**ViewAllAnimalProfiles**	A StaffMember OR a Client view a list of all the animals registered in the system.
UC-14	**ViewDetailedAnimalProfile**	A StaffMember OR a Client view detailed ithe detailed profile of a selected animal.
UC-15	**ViewAllClientProfiles**	A StaffMember views a list of all the clients registered in the system.
UC-16	**ViewDetailedClientProfile**	A StaffMember views detailed information about a selected client.
UC-17	**ViewOwnProfile**	A Client views their own complete profile.
UC-18	**EditOwnProfile**	A Client makes desired modifications to the fiels that make up their profile.
UC-19	**MissingProfilesError**	The system reports that insufficient profiles are available to run the matching algorithm.
UC-20	**SaveFile**	The system saves a profile file to persistent

		storage.
UC-21	**FileError**	There was an error processing a selected profile file (generalizes FileOpenError, FileFormatError, FileInputError, FileOutputError). The error is reported to the user and the operation is aborted.
UC-22	**FileFormatError**	The system reports that the file containes a corrupted or incomplete data format.
UC-23	**FileInputError**	The system reports that the file could not be read.
UC-24	**FileOutputError**	The system reports that the file could not be written to.
UC-25	**FileOpenError**	The system reports that the file could not be opened.

2.4.1.4 Detailed Use Case Flow of Events

For the LaunchMatchingAlgorithm use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember is logged into the cuACS system, and shows that the StaffMember can choose to launch an instance of the matching algorithm, and then details that the two possible overall outputs are an error condition, MissingProfilesError, and a successful run of the algorithm which outputs match information.

Use Case Identifier	UC-06
Use Case Name	**LaunchMatchingAlgorithm**
Participating Actor(s)	Initiated by StaffMember
Flow of Events	1. A StaffMember launches an instance of the matching algorithm.
Entry Condition(s)	• The StaffMember must be logged in to cuACS.
Exit Condition(s)	• The algorithm finishes running OR • The algorithm throws a MissingProfilesError
Quality Requirement(s)	• The algorithm should not lock up the system while it is running. • The algorithm should be able to match 100 clients with 100 animals in under one minute at least 85% of the time.
Traceability	FR-3.0.0, NFR-3, NFR-7

For the ViewListOfMatches use case, the following shows the flow of events through this use case. This use case can only be invoked if an instance of the algorithm has been run, and shows that the StaffMember can choose to view the resulting list of matches from the latest run of the algorithm.

Use Case Identifier	UC-07
Use Case Name	**ViewListOfMatches**
Participating Actor(s)	Initiated by StaffMember
Flow of Events	1. A StaffMember chooses the option to view a list of all the matches made by running the latest instance of the matching algorithm.
Entry Condition(s)	• An instance of the algorithm must have been run.
Exit Condition(s)	• The list of matches is shown to the StaffMember
Quality Requirement(s)	-
Traceability	FR-3.2.0

For the ViewDetailedMatchInformation use case, the following shows the flow of events through this use case. This use case can only be invoked if an instance of the algorithm has been run, and shows that the StaffMember can choose to view more detailed information about a specific match from the list of matches generated by the latest run of the algorithm.

Use Case Identifier	UC-08
Use Case Name	**ViewDetailedMatchInformation**
Participating Actor(s)	Initiated by StaffMember
Flow of Events	1. A StaffMember selects a specific match from the list of matches. 2. The StaffMember selects the option to view detailed information about the selected match.
Entry Condition(s)	• An instance of the algorithm must have been run.
Exit Condition(s)	• The detailed match information is shown to the StaffMember
Quality Requirement(s)	-
Traceability	FR-3.0.0

For the AddAnimalProfile use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember is logged into the cuACS system, and shows that the StaffMember can choose to add an animal profile by selecting that option, filling

out the information, and then saving the file. This will either result in the file being saved or a FileError.

Use Case Identifier	UC-09
Use Case Name	**AddAnimalProfile**
Participating Actor(s)	Initiated by StaffMember
Flow of Events	1. A StaffMember selects the option to add a new animal. 2. The StaffMember fills out all of the necessary information to create the animal's profile. 3. The StaffMember saves the animal's new profile into the system.
Entry Condition(s)	• The StaffMember must be logged in to cuACS.
Exit Condition(s)	• The new profile is successfully saved OR • There is a FileError which aborts the operation.
Quality Requirement(s)	-
Traceability	FR-1.1.0

For the EditAnimalProfile use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember is logged into the cuACS system, and shows that the StaffMember can choose to edit a specific animal's profile by selecting and viewing that animal's profile, going into an editing mode, filling out the information, and then saving the file. This will either result in the file being saved or a FileError.

Use Case Identifier	UC-10
Use Case Name	**EditAnimalProfile**
Participating Actor(s)	Initiated by StaffMember
Flow of Events	1. A StaffMember selects a specific animal's profile. 2. The StaffMember opens the detailed view of the selected animal's profile. 3. The StaffMember turns on a profile editing mode. 4. The StaffMember edits the desired fields in the animal's profile. 5. The StaffMember saves the altered profile.
Entry Condition(s)	• The StaffMember must be logged in to cuACS.
Exit Condition(s)	• The altered profile is successfully saved OR • There is a FileError which aborts the operation.
Quality Requirement(s)	-

Traceability	FR-1.3.0

For the AddClientProfile use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember is logged into the cuACS system, and shows that the StaffMember can choose to add a client profile by selecting that option, filling out the information, and then saving the file. This will either result in the file being saved or a FileError.

Use Case Identifier	UC-11
Use Case Name	**AddClientProfile**
Participating Actor(s)	Initiated by StaffMember Communicates with Client
Flow of Events	1. A StaffMember selects the option to add a new Cient. 2. The StaffMember fills out all of the necessary information to create the Client's profile. 3. The StaffMember saves the Client's new profile into the system.
Entry Condition(s)	• The StaffMember must be logged in to cuACS.
Exit Condition(s)	• The new profile is successfully saved OR • There is a FileError which aborts the operation.
Quality Requirement(s)	• A Client's profile data must be stored securely to comply with local privacy laws.
Traceability	FR-2.1.0, NFR-30

For the EditClientProfile use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember is logged into the cuACS system, and shows that the StaffMember can choose to edit a specific Client's profile by selecting and viewing that Client's profile, going into an editing mode, filling out the information, and then saving the file. This will either result in the file being saved or a FileError. Furthermore, the Client will then have a new or altered profile with which they can also change.

Use Case Identifier	UC-12
Use Case Name	**EditClientProfile**
Participating Actor(s)	Initiated by StaffMember Communicates with Client
Flow of Events	1. A StaffMember selects a specific Client's profile. 2. The StaffMember opens the detailed view of the selected Client's profile.

	3. The StaffMember turns on a profile editing mode. 4. The StaffMember edits the desired fields in the Client's profile. 5. The StaffMember saves the altered profile.
Entry Condition(s)	• The StaffMember must be logged in to cuACS.
Exit Condition(s)	• The altered profile is successfully saved OR • There is a FileError which aborts the operation.
Quality Requirement(s)	• A Client's profile data must be stored securely to comply with local privacy laws.
Traceability	FR-2.3.0, NFR-30

For the ViewAllAnimalProfiles use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember or a Client is logged into the cuACS system, and shows that the StaffMember or Client can view a list of all of the animals registered in the system.

Use Case Identifier	UC-13
Use Case Name	**ViewAllAnimalProfiles**
Participating Actor(s)	Initiated by StaffMember OR Initiated by Client
Flow of Events	1. A StaffMember or a Client selects the option to view all animals.
Entry Condition(s)	• The StaffMember is logged into cuACS OR • The Client is logged into cuACS.
Exit Condition(s)	• The list or all registered animals is displayed to the initiator.
Quality Requirement(s)	-
Traceability	FR-1.2.0, FR-1.2.2

For the ViewDetailedAnimalProfile use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember or a Client is logged into the cuACS system, and shows that the StaffMember or Client can view a specific animal's detailed profile information by selecting a specific animal out of the list of animals.

Use Case Identifier	UC-14
Use Case Name	**ViewDetailedAnimalProfile**
Participating Actor(s)	Initiated by StaffMember OR Initiated by Client

Flow of Events	1. A StaffMember or a Client selects, from the list of all animal profiles, a specific animal profile. 2. The StaffMember or Client selects the option to view more detailed information about that animal.
Entry Condition(s)	• The StaffMember is logged into cuACS OR • The Client is logged into cuACS.
Exit Condition(s)	• The detailed profile for the selected animal is displayed to the initiator.
Quality Requirement(s)	-
Traceability	FR-1.2.1, FR-1.2.3

For the ViewAllClientProfiles use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember is logged into the cuACS system, and shows that the StaffMember can view a list of all of the Clients registered in the system.

Use Case Identifier	UC-15
Use Case Name	**ViewAllClientProfiles**
Participating Actor(s)	Initiated by StaffMember
Flow of Events	1. A StaffMember selects the option to view all Client profiles.
Entry Condition(s)	• The StaffMember must be logged in to cuACS
Exit Condition(s)	• A list of all Client profiles is shown to the StaffMember.
Quality Requirement(s)	-
Traceability	FR-2.2.0

For the ViewDetailedClientProfile use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember is logged into the cuACS system, and shows that the StaffMember can view a specific Client's detailed profile information by selecting a specific Client out of the list of all Clients registered in the system.

Use Case Identifier	UC-16
Use Case Name	**ViewDetailedClientProfile**
Participating Actor(s)	Initiated by StaffMember
Flow of Events	1. A StaffMember selects, from the list of Client profiles, a specific Client profile. 2. The StaffMember selects the option to view more detailed

	information about that Client.
Entry Condition(s)	• The StaffMember must be logged in to cuACS
Exit Condition(s)	• The StaffMember is shown the detailed profile for the selected Client.
Quality Requirement(s)	-
Traceability	FR-2.2.1

For the ViewOwnProfile use case, the following shows the flow of events through this use case. This use case can only be invoked if a Client is logged into their account on the cuACS system, and shows that the Client can view their own profile information.

Use Case Identifier	UC-17
Use Case Name	**ViewOwnProfile**
Participating Actor(s)	Initiated by Client
Flow of Events	1. The Client chooses the option to view their own profile.
Entry Condition(s)	• The Client must be logged into their cuACS account.
Exit Condition(s)	• The Client is shown their own profile.
Quality Requirement(s)	-
Traceability	FR-2.3.1

For the EditOwnProfile use case, the following shows the flow of events through this use case. This use case can only be invoked if a Client is logged into their account on the cuACS system, and shows that the Client can edit their own profile information. This will either result in a FileError or it will result in the Client successfully saving their altered profile.

Use Case Identifier	UC-18
Use Case Name	**EditOwnProfile**
Participating Actor(s)	Initiated by Client
Flow of Events	1. A Client selects the option to edit their own profile. 2. The Client makes the desired changes to their profile. 3. The Client saves their altered profile.
Entry Condition(s)	• The Client must be logged into their cuACS account.
Exit Condition(s)	• The altered profile is successfully saved OR • There is a FileError, aborting the operation.

Quality Requirement(s)	• The intended format of the data in each of the Client's profile information fields should be easy to understand from a Client's perspective. For example, if entering a phone number, the client should not have to guess whether a format like (XXX) XXX-XXXX or a format like XXX-XXX-XXXX is correct.
Traceability	FR-2.3.2, NFR-2

For the MissingProfilesError use case, the following shows the flow of events through this use case. This use case can only be invoked if a StaffMember had tried to run the matching algorithm and the algorithm saw that it had insufficient profiles to match against. This results in an abortion of the operation.

Use Case Identifier	UC-19
Use Case Name	**MissingProfilesError**
Participating Actor(s)	StaffMember
Flow of Events	1. The system notifies the StaffMember that there are insufficient profiles of either Clients or animals to run the ACM algorithm.
Entry Condition(s)	• A "Launch ACM Algorithm" operation failed due to a lack of profiles to match against.
Exit Condition(s)	• The algorithm does not run and the operation is aborted.
Quality Requirement(s)	• The error should be presented to the user in an easy-to-understand manner which clearly explains what caused the error to happen.
Traceability	NFR-5, NFR-2

For the SaveFile use case, the following shows the flow of events through this use case. This use case is invoked when a Client or StaffMember wants to save a profile. This could be a Client having altered their own profile, a StaffMember adding a new animal, or anything in between. This results in either the file being saved or a FileError, in which case the file operation is aborted.

Use Case Identifier	UC-20
Use Case Name	**SaveFile**
Participating Actor(s)	Initiated by StaffMember OR Initiated by Client

Flow of Events	1. The initiator invokes the file saving system by saving whichever profile they are currently working on.
Entry Condition(s)	• A new or altered profile file must be available to be saved.
Exit Condition(s)	• The file is successfully saved OR • The system returns a FileError aborting the process.
Quality Requirement(s)	-
Traceability	FR-4.0.0

For the FileError use case, the following shows the flow of events through this use case. This use case can only be invoked if a file operation failed. This results in an abortion of the operation, putting the system back in the state it was before the error.

Use Case Identifier	UC-21
Use Case Name	**FileError**
Participating Actor(s)	StaffMember OR Client
Flow of Events	1. The system notifies the StaffMember or Client that an error has occurred with the chosen file operation.
Entry Condition(s)	• A file processing operation failed.
Exit Condition(s)	• The requested operation is aborted, reverting the state of the system to one before the change had taken place.
Quality Requirement(s)	• The error should be presented to the user in an easy-to-understand manner which clearly explains what caused the error to happen.
Traceability	NFR-5, NFR-2

For the FileFormatError use case, the following shows the flow of events through this use case. This use case can only be invoked if a file operation failed due to the file being read having an improper or corrupted data format. This results in an abortion of the operation, putting the system back in the state it was before the error.

Use Case Identifier	UC-22
Use Case Name	**FileFormatError**
Participating Actor(s)	StaffMember OR Client
Flow of Events	1. The system notifies the StaffMember or Client that the profile file is corrupt or has an incomplete data format (specializes use case

	FileError).
Entry Condition(s)	• The loading of a profile failed because the saved information is in an improper format.
Exit Condition(s)	• The profile load operation is aborted.
Quality Requirement(s)	• The error should be presented to the user in an easy-to-understand manner which clearly explains what caused the error to happen.
Traceability	NFR-5, NFR-2

For the FileInputError use case, the following shows the flow of events through this use case. This use case can only be invoked if a file operation failed due to a failure to read a file. This results in an abortion of a profile load operation, putting the system back in the state it was before the error.

Use Case Identifier	UC-23
Use Case Name	**FileInputError**
Participating Actor(s)	StaffMember OR Client
Flow of Events	1. The system notifies the StaffMember or Client that the profile file could not be read (specializes use case **FileError**).
Entry Condition(s)	• A file read operation failed.
Exit Condition(s)	• The profile load operation is aborted.
Quality Requirement(s)	• The error should be presented to the user in an easy-to-understand manner which clearly explains what caused the error to happen.
Traceability	NFR-5, NFR-2

For the FileOutputError use case, the following shows the flow of events through this use case. This use case can only be invoked if a file operation failed due to an error writing the file to persistent storage. This results in an abortion of the operation, putting the system back in the state it was before the error. The file is not written.

Use Case Identifier	UC-24
Use Case Name	**FileOutputError**
Participating Actor(s)	StaffMember OR Client
Flow of Events	1. The system notifies the StaffMember or Client that the profile file

	could not be written to persistent storage (specializes use case **FileError**).
Entry Condition(s)	• A file write operation failed.
Exit Condition(s)	• The profile save operation is aborted, reverting profiles to their previous, unchanged state.
Quality Requirement(s)	• The error should be presented to the user in an easy-to-understand manner which clearly explains what caused the error to happen.
Traceability	NFR-5, NFR-2

For the FileOpenError use case, the following shows the flow of events through this use case. This use case can only be invoked if a file operation failed due to a failure to open a file for reading or writing. This results in an abortion of the operation, putting the system back in the state it was before the error.

Use Case Identifier	UC-25
Use Case Name	**FileOpenError**
Participating Actor(s)	StaffMember OR Client
Flow of Events	1. The system notifies the StaffMember or Client that the profile file could not be opened (specializes use case **FileError**).
Entry Condition(s)	• A file open operation failed.
Exit Condition(s)	• The requested operation (either save or load a profile) is aborted, reverting profiles to their previous, unchanged state.
Quality Requirement(s)	• The error should be presented to the user in an easy-to-understand manner which clearly explains what caused the error to happen.
Traceability	NFR-5, NFR-2

2.4.2 Object Model

The object model describes the high level concepts which are manipulated by the system. The two components of this model included in this document are the data dictionary and the class diagram.

2.4.2.1 Data Dictionary

The data dictionary is a dictionary of the objects which make up the underlying system. It gives a name, a set of attributes and associations, and a definition or description for each object. This is particularly helpful to enumerate all of the objects which play a major role in the system.

Table 5—Data Dictionary for the cuACS System

Class Number	Traceability	Object Name	Attributes and Associations	Definition/Description
CN-01	UC-02, UC-03, UC-07, UC-09, UC-10, UC-13, UC-14	Animal	- idNumber - name - breed - size - age - species - gender - isHypoAllergenic	- Animals, within the shelter, ready for adoption. Each animal has a set of attributes (unique physical, and non-physical), as well as a name. The set of non-physical and physical attributes are used in determining potential matches with clients, using the ACM algorithm.
CN-02	UC-05, UC-04, UC-07, UC-11, UC-12, UC-13, UC-14, UC-15, UC-16, UC-17, UC-18	Client	- Physical traits - Non-physical traits - Unique client ID number	- An individual wishing to adopt an animal from the shelter. Each individual has their own physical characteristics, as well as non-physical characteristics that are used in determining a potential match with an animal, using the ACM algorithm.
CN-03	UC-01, UC-02, UC-03, UC-04, UC-06, UC-07, UC-08, UC-09, UC-10, UC-11, UC-12, UC-13, UC-14, UC-15, UC-16,	StaffMember	- Unique Staff ID number - staff member personal traits	- An individual who is employed by the Shelter. Can also be a client of the shelter.
CN-04	UC-04, UC-05, UC-11, UC-12, UC-	Client_Management_System	- Client	- An instance of the Client Management System, responsible for accessing client records and

	15, UC-16, UC-17, UC-18, UC-19			information, as well as sending new records/information to the client storage device - Can be accessed by clients and staff members
CN-05	UC-02, UC-03, UC-09, UC-10, UC-13, UC-14, UC-19	Animal_Management_System	- Animal	- An instance of the Animal Management System, responsible for accessing animal records/information from the storage device. Also responsible for sending edits, changes and new information about the Animals to the storage device. - Can be accessed by staff members only
CN-06	UC-01, UC-06, UC-19, UC-07, UC-08	Control	- storage - view - fileSaver	- An instance of the Carleton University Animal Care System matching algorithm (cuACS), launched by a staff member - the object responsible for the flow and the control of various entity objects, such as animal, staffMember, and client, as well as interactions with the storage and saving
CN-07		View		- object responsible for the sending data to and from the graphical user interface
CN-08		Storage	- numElements - animalList - largestID	- the system used to store animals, clients, and other entity objects
CN-9	UC-20, UC-21, UC-22, UC-23, UC-24, UC-25	FileSaver		- object for reading and writing data to and from the data file storage

2.4.2.2 Class Diagram

The class diagram seen below provides a graphical representation of the Data Dictionary. It shows the attributes of and associations between all of the objects that make up the system in a way that is easy to understand and in which relationships such as multiplicity, inheritance and ownership can be easily expressed.

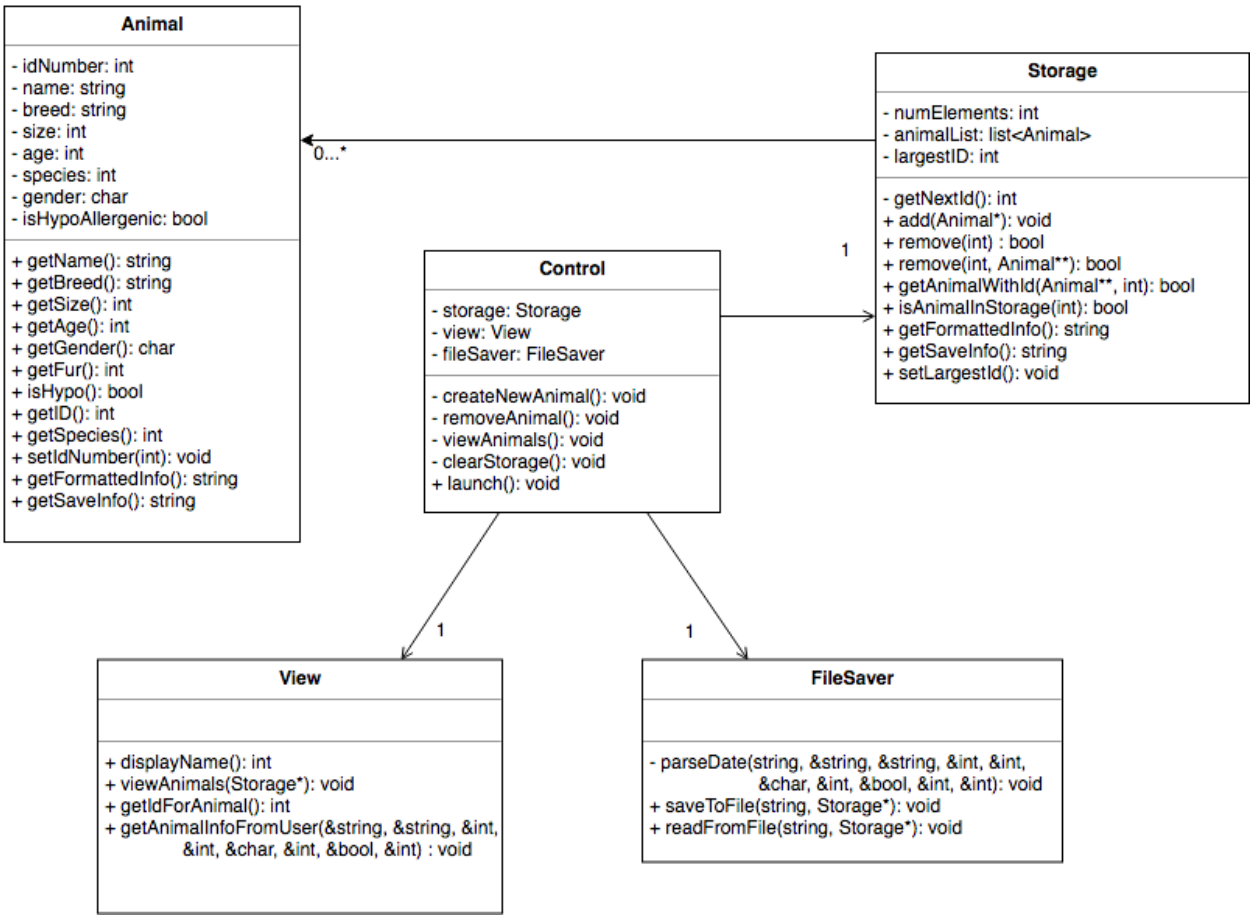

Figure 7—Object Model Class Diagram

3. Glossary of Terms Used

What follows is a collection of the application domain-specific terms used in this document in alphabetical-order.

Term	Definition/Explanation

Animal	An animal which resides in the animal shelter.
Client	A person who is looking to adopt a pet.
StaffMember	A person who is employed by the animal shelter.
Persistent Storage	Storage that persists even if the system is turned off and on again.
Profile	A collection of information, stored as a file, containing both physical and non-physical information identifying an animal or a Client.

4. Index of Tables

5. Index of Figures