Carleton University Animal Care System (cuACS)

Requirements Analysis Document

Team: JSB
Justin Ward
Sabina Bialic
Brandon Ward

Submitted To:

Dr. Christine Laurendeau
COMP 3004 Object-Oriented Software Engineering
School of Computer Science
Carleton University

Feb-12-2019

Contents

Figures

Tables

1. Introduction

1.1 Purpose of System

The purpose of the Carleton University Animal Care System (cuACS) is to accurately match homeless animals in the shelter with loving humans who are looking to adopt a new addition into their family. In order to ensure that these animals and humans are fully compatible, the cuACS system shall provide a tool that automatically matches them together, in order to achieve an optimal set of fully compatible matches. These matches will take into account the best interests of all animals in the shelter and the qualities that humans are looking for in a companion, rather than producing a small subset of highly favourable matches and a large number of lesser compatible ones.

The cuACS system shall also provide the ability for the shelter staff to manage all the adoptable animals, view all the detailed profiles of the animals, the ability to manage all the clients, and view all the detailed profiles of the clients. As well, the system will allow the shelter staff the ability to assess animal-client compatibility and compute an optimal set of animal-client matches using the matching algorithm described previously. The clients on the other hand, will have the ability to view all the animals in the system, and view specific details of an animal in order to assess their physical and non-physical traits. As well, the clients will be able to edit their own personal profiles, which includes their personal information and matching preferences which pertain to the attributes that they are looking for in their potential new companion.

The most important aspect of the cuACS system is the animal-client matching (ACM) algorithm, as it is the basis of how animals and clients will be successfully matched with one another. One of the main goals of the cuACS system is to match clients and animals together in a way that produces a set of compatible matches to increase the likelihood of a client adopting an animal at the shelter. The ACM takes into account a specific client's matching preferences and use this to determine which animals in the system are best suited for the client. These matching preferences are made up of both physical attributes, along with unique non-physical attributes that pertain to each animal. Since every client has different criteria for the perfect companion, it is critical that the system accurately takes their preferences into account to match them with animals that have the best chance of meeting these criteria.

2.2 Overview of Document

The purpose of this document is to further understand the definition and scope of the problem in order to create a system that successfully meets all the needs of the users that shall be interacting with the system. This document aims to identify the problem that the system shall solve and to model the application domain by looking into the conditions that must be met for the system to be developed and implemented correctly. Provided are the functional requirements and non-functional requirements that are required by the user in order for the cuACS system to satisfy their business needs. As well, several system models are presented in order to obtain a further understanding of how the users will be interacting with the system, and what the expected behaviour is with each event. The use case model and object model analyze these interactions with the system, and provide a deeper understanding into the users' needs with the system. Finally, a glossary defining common terminology used consistently throughout this document is provided.

2. Proposed System

This section goes into further detail regarding the specific features that the proposed system shall have in order to be accepted by the user. It describes what the user expects the cuACS system to do, and is broken down into two subsections: functional requirements, and non-functional requirements.

2.1 Overview

The cuACS system that is being built will have a graphical user interface (UI) and will be developed using the Linux Ubuntu platform, with all source code having been written in C++. In addition, the system shall support two different types of users: the shelter staff, and the human clients looking to adopt an animal. The shelter staff will be given the ability to manage the shelter's adoptable animals, manage their human clients, as well as run an animal-client matching algorithm for which they will also be able to view the results of. The clients on the other hand will be given the ability to view a list of all the animals that are available for adoption, view specific details for a selected animal, and update their personal information, as well as their adoption matching preferences. In order for the cuACS system to be successfully accepted by the users, the specified requirements must be met.

2.2 Functional Requirements

Functional requirements specify the details of a function that the system must support; it describes the interactions between the system and the external environment, which is independent of the system's implementation. The cuACS system supports two types of users:

- The *shelter staff* should be able to manage the shelter's adoptable animals, manage their human clients, and have access to animal-client compatibility to compute an optimal set of animal-client matches
- The *client* should be able to view a list of all animals that are available for adoption, view specific details for a selected animal, and have the ability to edit their profile where they can update their personal information and matching preferences

Table 1 - Functional Requirements

FR-01	The cuACS system must support two categories of accounts; staff members and clients
FR-01-01	*Staff members must be able to launch the cuACS system*
FR-02	Staff members must be able to log in to existing accounts within the cuACS system
FR-03	Staff members must be able to create accounts for new staff members

FR-04	Staff members must be able to launch the ACM (animal-client matching) algorithm
FR-04-01 FR-04-02	*Staff members must be able to view the results of the ACM* *Staff members must be able to see specific match results*
FR-05	Staff members can view a list of all animal profiles
F-05-01 F-05-02 F-05-03	*Staff members can view specific animal profiles* *Staff members can edit specific animal profiles* *Staff members can add animal profiles*
FR-06	Staff members can view a list of all client profiles
FR-06-01 FR-06-02 FR-06-03	*Staff members can view specific client profiles* *Staff members can add client profiles* *Staff members must be able to save the user's profiles*
FR-07	Clients must be able to log into existing accounts within the cuACS system.
FR-08	Clients must be able to edit their own profile
FR-08-01 FR-08-02 FR-08-03	*Clients must be able to edit their personal information* *Clients must be able to edit their matching preferences* *Clients must be able to save their profiles*
FR-09	Clients must be able to view a list of all animal profiles
FR-09-01	*Clients must be able to view a detailed profile of a selected animal*

2.3 Non-Functional Requirements

Non-functional requirements place restrictions on the system that is being developed, and it specifies the external constraints that the system must meet. They describes aspects of the system that are not directly related to the functional behavior of the system, and include a variety of requirements that apply to different aspects of the system. There are several categories of non-functional requirements: usability, reliability, interface, supportability, performance, legal, packaging, implementation, and operation.

Table 2 - Non-Functional Requirements

NFR-01	Usability: clear, detailed instructions on how to install and configure the system should be provided

NFR-02	Usability: save operations should be confirmed to the user
NFR-03	Usability: user interface should be graphical
NFR-03	Usability: user interface should match Carleton University colors
NFR-04	Usability: the system must be easily learned by both categories of users
NFR-05	Usability: all error messages should be descriptive and suggest appropriate solutions
NFR-06	Reliability: save operations save directly to the database in case of system failure
NFR-07	Reliability: users should be able to securely log into their accounts
NFR-08	Reliability: the system should be recoverable in the event of a system failure
NFR-09	Interface: the data must be stored in a relational table-based style to conform to the SQL database
NFR-10	Interface: the system must use query language commands to retrieve data from the SQLite database
NFR-11	Interface: the system must use Data Definition Language (DDL) to define how the data is stored within the SQLite database
NFR-12	Interface: the system must use Data Modification Language to modify the data within the SQLite database.
NFR-13	Supportability: the system should be adaptable for other animal shelters
NFR-14	Supportability: the staff should be able to add animals into the system without it having to shut down temporarily
NFR-15	Supportability: the ACM algorithm must be easily adapted to implement additional traits
NFR-16	Performance: there should not be more than a 5-second delay to save and update changes made to a client's profile
NFR-17	Performance: the ACM algorithm must run in under 5 seconds
NFR-18	Performance: the login operation should finish within 10 seconds, whether it's successful or times out
NFR-19	Legal: the client should be allowed to access and edit their personal information as specified in the Municipal Freedom of Information and Protection of Privacy Act, R.S.O. 1990, c. M.56, s. 36.
NFR-20	Legal: the client's personal information should not be disclosed as specified in the

	Freedom of Information and Protection of Privacy Act R.S.O. 1990, c. F.31, s. 21 (3)
NFR-21	Legal: the system shall not implement any copyrighted code as specified in the Copyright Act Section 42(1)(a)
NFR-22	Packaging: the source code, data files, and configuration scripts, as well as installation, build and launching instructions must be delivered in a single .tar file
NFR-23	Packaging: the system must be installed with a single command
NFR-24	Packaging: the system must be launchable with a single command
NFR-25	Implementation: the system must be written in C++
NFR-26	Implementation: the system's graphical user interface will be created using Qt
NFR-27	Implementation: the system needs to work on the class VM
NFR-28	Operation: The system should be accessible by clients and staff
NFR-29	Operation: The staff should only be able to add or edit one animal at a time
NFR-30	Operation: The system should be run by one single command

2.4 System Models

This section contains the complete functional specifications, including illustrations of the user interface of the system and the navigational paths representing the sequence of events that occur within the system.

2.4.1 Use Cases

Use cases represent the functionality of a system and are a sequence of events that describe all the possible events that the actors interacting with the system can partake in, focusing on the external behaviour of the system. In the cuACS system, there are two actors that shall interact with the system, therefore use cases need to be developed to show the events of each of those actors as they interact with the system.

Use Case Overview

Figure 1 depicts a high-level use case for the cuACS system, defining the boundaries of the system as the actors interact with it. Along with the high-level use case diagram, table 3 provides a written description outlining, from the user's perspective, how the system's behaviour shall respond to each request.

Figure 1 - High Level Use Case Diagram

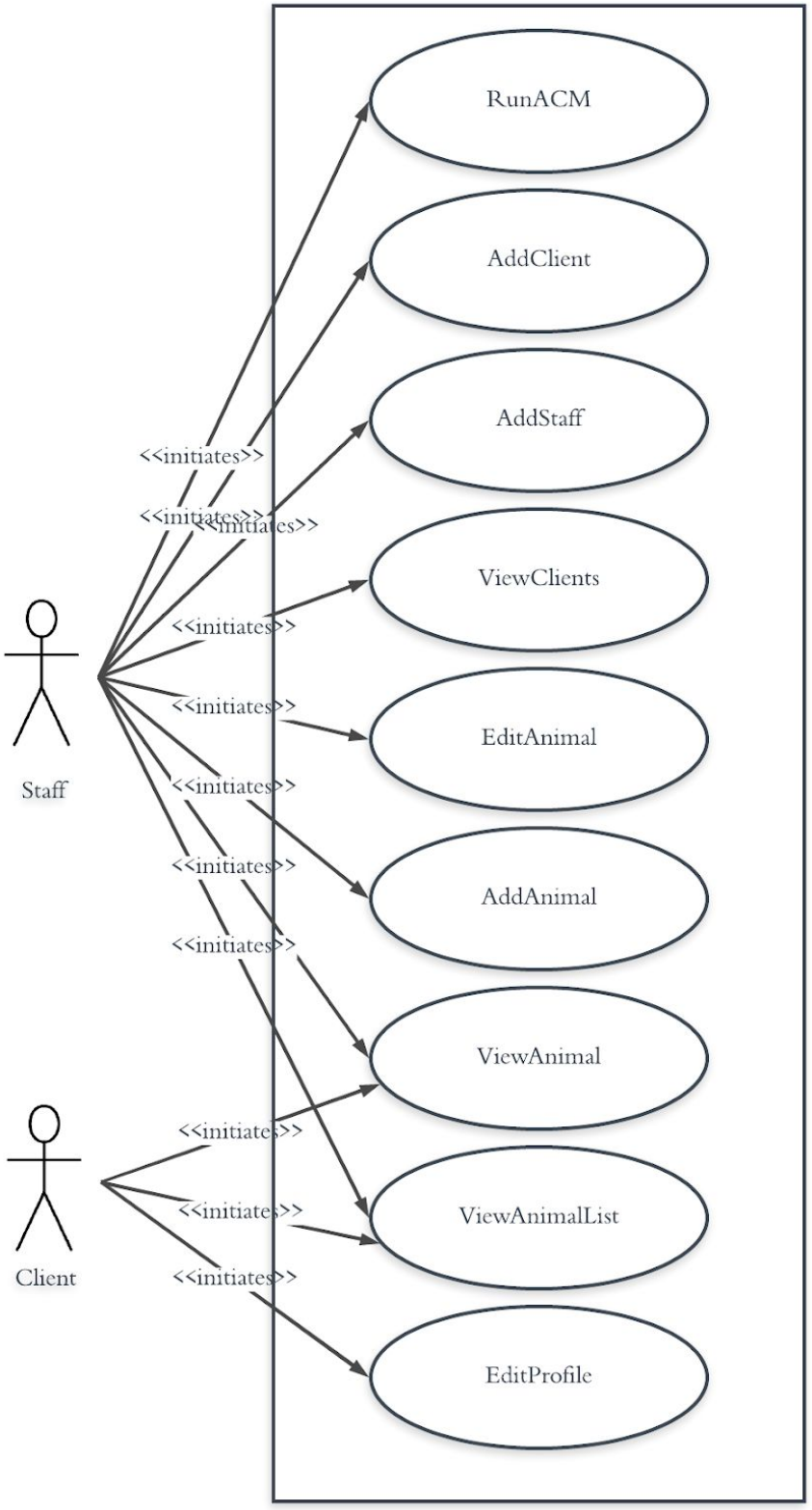

Table 3 - High-Level Use Case Descriptions

UC-01	RunACM	The staff member runs the ACM algorithm (generalizes the scenario ViewMatches, ViewAnimal)
UC-02	AddStaff	The staff member adds a staff profile
UC-03	ViewClients	The staff member views all client profiles (generalizes the scenario AddClient, ViewClientDetails)
UC-04	EditAnimal	The staff member edits the details for an animal
UC-05	AddAnimal	The staff member adds an animal
UC-06	ViewAnimal	The client views a specific animal's profile
UC-07	ViewAnimalList	The staff member and the client views all the animals (generalizes the scenario ViewAnimal)
UC-08	EditProfile	The client views their profile (generalizes the scenario EditPersonalInfo, EditMatchingInfo)

In addition to a high-level use case for the cuACS system, detailed use cases for each subsystem are provided; that is, a detailed use case for how the client and staff interacts with the system, and a detailed use case for how the shelter staff interact with the system. Figures 2-7 outline how both the clients and the staff members interact with the cuACS system in order to achieve the functional requirements of the system. Note that UC-3 ViewClients does not have a detailed use case, since there is only one step to that use case.

Figure 2 - Detailed RunACM Diagram

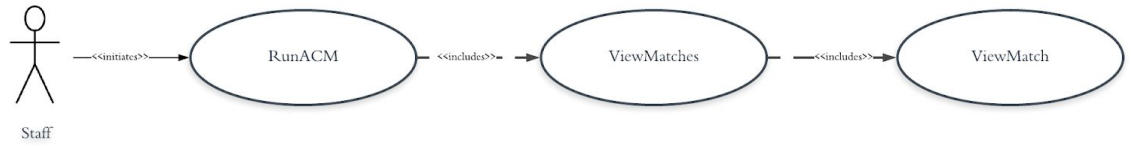

Figure 3 - Detailed AddStaffDiagram

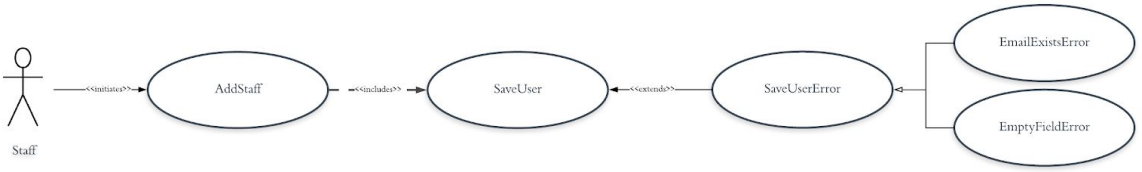

Figure 4 - Detailed AddClient Diagram

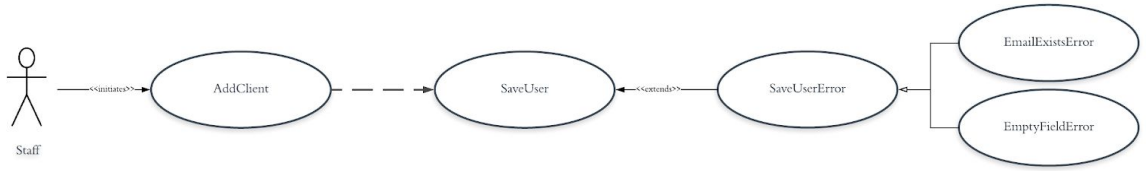

Figure 5 - Detailed EditAnimal Diagram

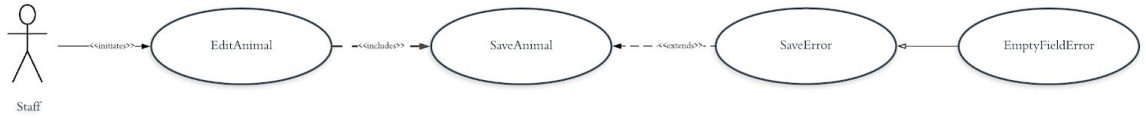

Figure 6 - Detailed AddAnimal Diagram

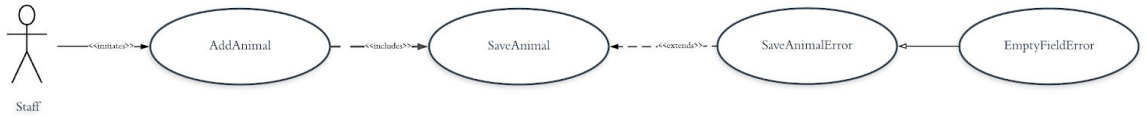

Figure 7 - Detailed EditProfile Diagram

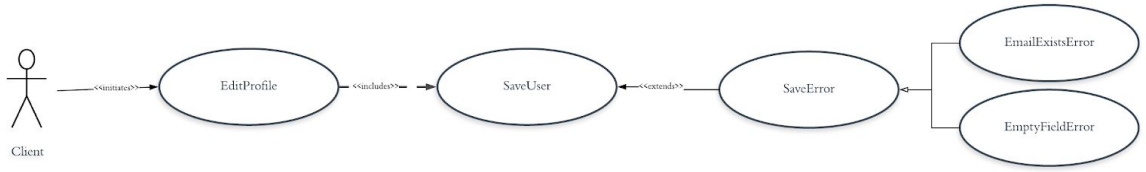

Table 4 - Detailed Use Case Descriptions

UC-09	ViewMatches	The staff member can view ACM results
UC-10	ViewMatch	The staff member can view a specific match from the ACM results
UC-11	SaveUser	The client saves changes made to their profile
UC-12	SaveUserError	There was an error processing the changes made to the client (generalizes EmailExistsError, EmptyFieldError). The error is reported to the user, and the user is prompted to try again
UC-13	EmailExistsError	The system reports that the current email is already being used in the system

UC-14	EmptyFieldError	The system reports that there was an empty field
UC-15	AddClient	The staff member adds a client profile
UC-16	SaveAnimal	The staff member saves the changes made to the animal
UC-17	SaveAnimalError	There was an error processing the changes made to the animal (generalizes EmptyFieldError). The error is reported to the user, and the user is prompted to try again

Use Case Flow of Events

The flow of events describe the sequence of events that occur within each use case that was presented previously in table 3 and table 4. Along with the flow of events are the participating actors, entry conditions, exit conditions, quality requirements, and traceability for each use case.

Table 5 - Use Case Flow of Events

Use Case Identifier	UC-01
Name	RunACM
Participating Actor	Staff
Flow of Events	1. The staff member clicks a Run ACM button 2. The system runs the ACM
Entry Condition	The staff member clicks a button.
Exit Condition	The ACM finishes running and displays a list of matches
Quality Requirements	The algorithm must run in under 5 seconds
Traceability	FR-04

Use Case Identifier	UC-02
Name	AddStaff
Participating Actor	Staff
Flow of Events	1. The staff member clicks an Add Staff Member button 2. The staff member enters the email address associated with the new account
Entry Condition	The staff member clicks a button.

Exit Condition	The new staff member is added into the system.
Quality Requirements	
Traceability	FR-03

Use Case Identifier	UC-03
Name	ViewClients
Participating Actor	Staff
Flow of Events	1. The staff member clicks a View Client Profiles button 2. The system displays all the client profiles in the system
Entry Condition	The staff member clicks a button
Exit Condition	
Quality Requirements	
Traceability	FR-06

Use Case Identifier	UC-04
Name	EditAnimal
Participating Actor	Staff
Flow of Events	1. The staff members views a specific animal's profile 2. The staff member modifies the information about that specific animal 3. The staff member saves the changes 4. If an error occurs with saving the information, the system notifies the staff member
Entry Condition	The staff member selects an animal to edit
Exit Condition	The staff member saves the changes
Quality Requirements	
Traceability	F-05-02

Use Case Identifier	UC-05

Name	AddAnimal
Participating Actor	Staff
Flow of Events	1. The staff member views all the animals in the system 2. The staff member elects a button to add a new animal into the system 3. The staff member fills out all the information about that animal 4. The staff member saves the changes 5. If an error occurs with saving the information, the system notifies the staff member
Entry Condition	The staff member clicks a button.
Exit Condition	The new animal is added into the system.
Quality Requirements	
Traceability	F-05-03

Use Case Identifier	UC-06
Name	ViewAnimal
Participating Actor	Client, Staff
Flow of Events	1. The user clicks on a specific animal from the animal list 2. The system displays information for the specified animal
Entry Condition	The user clicks on an animal from a list.
Exit Condition	The user elects to go back.
Quality Requirements	
Traceability	F-05-01, FR-09-01

Use Case Identifier	UC-07
Name	ViewAnimalList
Participating Actor	Client, Staff
Flow of Events	1. The user clicks a View Animals button 2. The system displays the animals that are available in the system
Entry Condition	The user clicks a button.

Exit Condition	
Quality Requirements	
Traceability	FR-05, FR-09

Use Case Identifier	UC-08
Name	EditProfile
Participating Actor	Client
Flow of Events	1. The system displays Client profile 2. The system offers 2 menu options: Edit Personal Info, Edit Matching Info 3. If Client chooses Edit Personal Info, the system will display their personal info for editing 4. If Client chooses Edit Matching Info the system will display their matching info for editing
Entry Condition	The client has elected to view their profile
Exit Condition	The client has elected to go back to the main screen
Quality Requirements	
Traceability	NFR-19

Use Case Identifier	UC-09
Name	ViewMatches
Participating Actor	Staff
Flow of Events	1. The system displays a list of animals based off of the results of the ACM for a given client
Entry Condition	The staff member ran the ACM.
Exit Condition	
Quality Requirements	The time it takes to view the matches after running the ACM should not exceed more than 5 seconds.
Traceability	FR-04-01

Use Case Identifier	UC-10
Name	ViewMatch
Participating Actor	Staff
Flow of Events	1. The staff member is able to view specific match details
Entry Condition	The staff member ran the ACM.
Exit Condition	
Quality Requirements	The time it takes to view the matches after running the ACM should not exceed more than 5 seconds.
Traceability	FR-04-02

Use Case Identifier	UC-11
Name	SaveUser
Participating Actor	Client
Flow of Events	1. The system saves their Client's personal information 2. If save was successful the system will display a confirmation dialog 3. If save was not successful the system will display a Save Error (includes use case **SaveUserError**)
Entry Condition	Client elects to save
Exit Condition	The updated info is saved to the database
Quality Requirements	The user must be logged with client account
Traceability	FR-08-03

Use Case Identifier	UC-12
Name	SaveUserError
Participating Actor	Client
Flow of Events	1. The system notifies the user that an error has occurred with saving the information (generalizes **EmailExistsError** and **EmptyFieldError**).

Entry Condition	An information save operation failed.
Exit Condition	The requested operation is aborted.
Quality Requirements	
Traceability	NFR-05

Use Case Identifier	UC-13
Name	EmailExistsError
Participating Actor	Client
Flow of Events	1. The system notifies the client that the email is already associated with a user in the system (specializes use case **SaveUserError**).
Entry Condition	An information save operation failed.
Exit Condition	The requested operation is aborted.
Quality Requirements	
Traceability	NFR-05

Use Case Identifier	UC-14
Name	EmptyFieldError
Participating Actor	Staff, Client
Flow of Events	1. The system notifies the staff member or the client that there is a missing field (specializes use case **SaveUserError, SaveAnimalError**).
Entry Condition	An information save operation failed.
Exit Condition	The requested operation is aborted.
Quality Requirements	
Traceability	NFR-05

Use Case Identifier	UC-15

Name	AddClient
Participating Actor	Staff
Flow of Events	1. The staff member clicks an Add Client button 2. The staff member enters the email associated with the client's account
Entry Condition	The staff member clicks a button.
Exit Condition	The new client is added into the system.
Quality Requirements	
Traceability	FR-06-02

Use Case Identifier	UC-16
Name	SaveAnimal
Participating Actor	Staff
Flow of Events	1. The system saves their animal's information 2. If save was successful the system will display a confirmation dialog 3. If save was not successful the system will display a Save Error (includes use case **SaveAnimalError**)
Entry Condition	Client elects to save
Exit Condition	The updated info is saved to the database
Quality Requirements	The user must be logged with client account

Use Case Identifier	UC-17
Name	SaveAnimalError
Participating Actor	Staff
Flow of Events	1. The system notifies the user that an error has occurred with saving the animal's information.
Entry Condition	An information save operation failed.
Exit Condition	The requested operation is aborted.

Quality Requirements	
Traceability	NFR-05

2.4.2 Object Model

The analysis object model focuses on the specific entities that are manipulated by the system, and their attributes and relationships with different entities. To show these features of the system, UML class diagrams are used to create analysis object models, and they focus on the entity, boundary, and control objects of the system. As seen in figure 8, the cuACS system can be represented in a UML class diagram (ignoring boundary and control objects) along with table 6 which shows a corresponding data dictionary for the UML diagram.

Figure 8 - UML Class Diagram

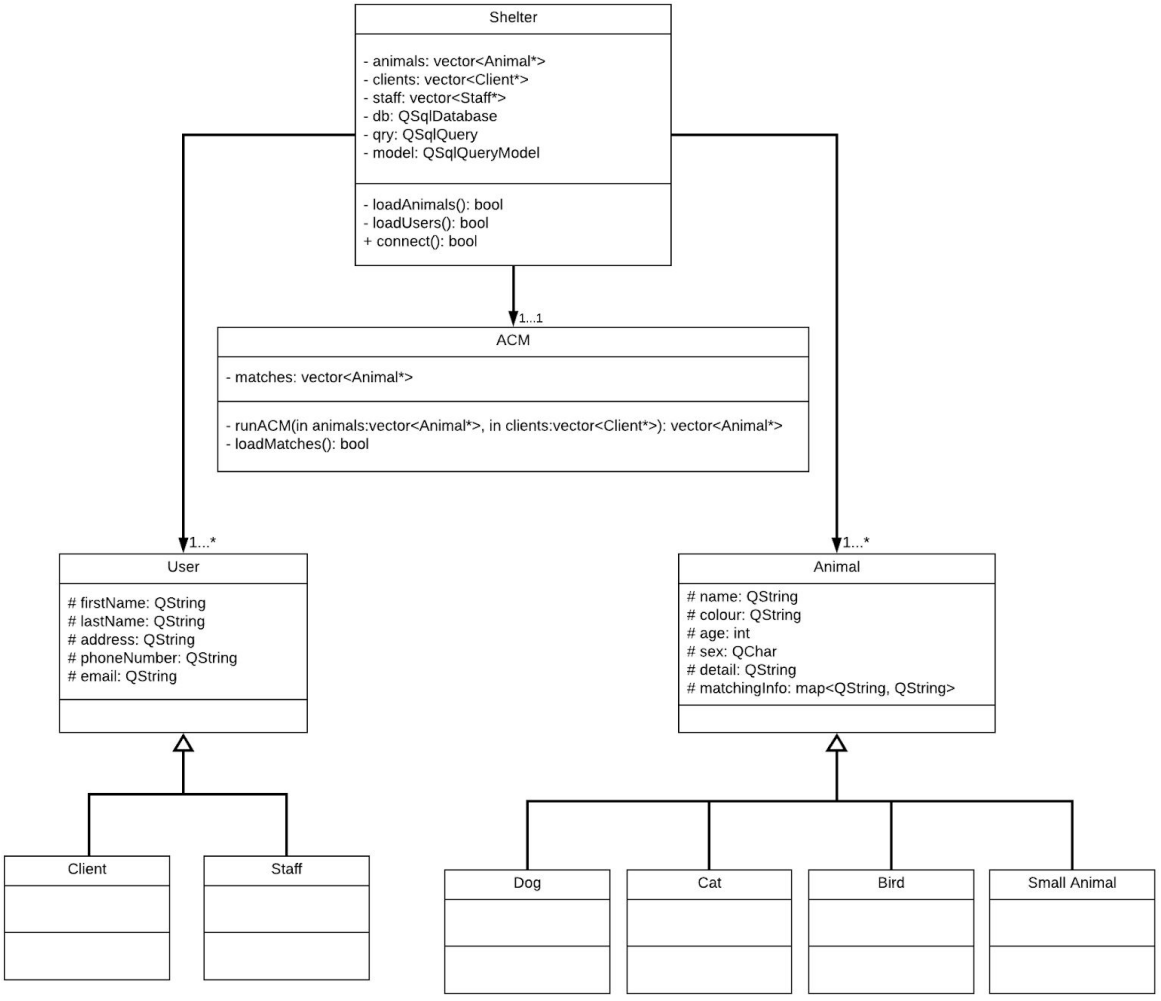

Table 6 - Data Dictionary

ID	Class	Class Description	Attributes	Associations	Traceability
C-01	Shelter	Shelter class to represent the cuACS shelter	Animals Clients Staff DB Qry Model	One shelter has many animals, many users, and one ACM	UC-02 UC-03 UC-04 UC-05 UC-06 UC-07 UC-08 UC-09 UC-10 UC-11 UC-15 UC-16
C-02	ACM	ACM class to represent the animal-client matching algorithm	Matches	One ACM belongs to one shelter	UC-01 UC-09 UC-10
C-03	User	User class to represent the users of the system; base class for Client and Staff	First Name Last Name Address Phone Number Email	Many users belong to one shelter, and a user can be either a client or a shelter staff	UC-02 UC-03 UC-07 UC-08 UC-11 UC-15
C-04	Client	Client class that represents the clients of the system; derived from User class		Client is a type of user	UC-03 UC-07 UC-08 UC-11 UC-15
C-05	Staff	Staff class that represents the shelter staff of the system; derived from User class		Staff is a type of user	UC-02 UC-07 UC-08 UC-11
C-06	Animal	Animal class to represent the animals in the system; base class for Dog, Cat, Bird, and Small Pet	Name Colour Age Sex Detail Matching Info	Many animals belong to one shelter, and an animal can be either a dog, cat, bird, or a small animal	UC-04 UC-05 UC-06 UC-16
C-07	Dog	Dog class to represent		Dog is a type of Animal	UC-04

		the dogs in the system; inherits from Animal class			UC-05 UC-06 UC-16
C-08	Cat	Cat class to represent the cats in the system; inherits from Animal class		Cat is a type of Animal	UC-04 UC-05 UC-06 UC-16
C-09	Bird	Bird class to represent the birds in the system; inherits from Animal class		Bird is a type of Animal	UC-04 UC-05 UC-06 UC-16
C-10	Small Animal	Small Animal class to represent the small pets in the system; inherits from Animal class		Small Animal is a type of Animal	UC-04 UC-05 UC-06 UC-16

3. Glossary

ACM	Animal-client matching algorithm
Client	Human user that is looking to adopt an animal
cuACS	Carleton University Animal Care System; the system to be developed
Shelter	Refers to shelter for whom the system is being made for; in this case, refers to the cuACS organization
Staff	Human user that manages the system
User	A user of that system; general term to referring to both client and staff users