Project: cuACS

Christine Laurendeau

Problem Description

Animal shelters provide a crucial service for the homeless animals awaiting adoption into a loving home, as well as for the humans who enjoy the comfort and companionship that a beloved pet can offer. However, one problem with the animal adoption process is that it often allows animals to be adopted by humans with whom they are not fully compatible. A shelter's animals and the human clients who adopt them may be mismatched in a variety of ways, including in terms of temperament, needs, and expectations. The *Carleton University Animal Care System (cuACS)* proposes to address this issue by providing a tool that automatically matches together, based on compatibility, shelter animals and the human clients who wish to adopt them. The goal of the *cuACS* system is to generate an optimal set of matches, where a match consists of an animal available for adoption and a human client who is well suited to adopt it. An optimal set of matches will take into account the best interests of all animals in the shelter, rather than producing a small subset of highly favourable matches and a large number of lesser compatible ones.

The main features of the *cuACS* system include:

- the ability to manage the shelter's adoptable animals and their detailed profile information
- the ability to manage the shelter's human clients and their profile information, including personal data and matching preferences; and
- the ability to assess animal-client compatibility and compute an optimal set of animal-client matches.

The *cuACS* system supports two categories of users: clients and shelter staff. Staff members can add new animals, view and edit each animal's profile information, as well as view client profiles and add new clients. They can also launch the *animal-client matching (ACM) algorithm* that computes the optimal set of animal-client matches. Clients can view the detailed profile of a selected animal, and they can edit their own client profile. In addition to basic contact information, each client's profile includes two types of data: the client's own personal information (the *attributes,* or qualities, that describe the client), and the client's *matching preferences*, which consist of the attributes that the client is looking for in an adopted animal. The ACM algorithm uses the profiles of all clients and animals in order to compute the best possible set of matches between animals and clients.

Features

Clients can view a list of all animals available for adoption, and they can view the detailed profile of a selected animal. Clients can also edit their own profile, where they can update their personal information and their matching preferences.

Shelter staff can view a list of all animals available for adoption, and they can view the detailed profile of a selected animal. They can add a new animal and edit any animal's profile information. Shelter staff can also view a list of all existing clients, and they can view the detailed profile of a selected client. They can also add new clients. Most importantly, staff members can launch the execution of the ACM algorithm and view the resulting matches. The ACM algorithm uses animal and client profiles to compute the best matches, and it outputs the match results. These results must consist of both a summary of matches, and the details of a selected match. The summary information indicates the names of the matched pairs of animals and clients. The detailed information for a given match specifies the exact rules that were used to compute that match, as well as the data supporting how and why that specific match was computed.

Technical Specifications

The Linux Ubuntu platform, as provided in the official course virtual machine (VM), will be used as the test bed for evaluating the *cuACS* system. All source code must be written in C++, and it must be designed and implemented at the level of a 3rd year undergraduate Computer Science student, using advanced OO techniques such as polymorphism and design patterns.

User Interface (UI)

The *cuACS* user interface (UI) will preferably be graphical in nature. A console-based UI will be an acceptable alternative, but will not earn full marks for any of the coding deliverables. In general, user features should be easily navigable, either as menu items and/or pop-up menus. The look-and-feel of the *cuACS* system should be professional and consistent with commercially available UIs.

Data Storage

The *cuACS* system will run on a single host, with all its data stored on that same host. Every development team implementing the *cuACS* system will design their own animal and client data representation. Data must be stored in persistent storage (flat files or SQL database) and loaded into the UI when *cuACS* launches. Modifications to data in the UI must be saved to persistent storage after every change.

Data storage, both in memory and in persistent storage, must be organized for ease of retrieval and efficient use of space. There should be no duplication of information anywhere. Persistent data may be stored in flat files, or any other mechanism available in the VM provided, including the Qt SQLite library. The *cuACS* system must be delivered already configured with a *minimum* of 25 complete animal profiles and 20 complete client profiles.

Animal-Client Matching (ACM) Algorithm

Every development team implementing the *cuACS* system will define a set of attributes (or qualities) that they believe are important in matching together compatible animals and human clients and likely to result in a successful adoption. Some attributes will be physical, such as type of animal, breed, gender, age, and size. Non-physical attributes will relate to temperament, personality traits, habits, etc. Specific values for these attributes will be included in every animal profile. A client profile will contain specific values for the attributes that are relevant to humans in the adoption process, as part of the client's personal data. Each client profile must also contain the client's matching preferences, which are the values for the attributes that the client is looking for in an adopted animal.

Once a development team has defined a set of attributes, they must design their own *cuACS* animal-client matching (ACM) algorithm. The algorithm must use its own unique set of rules for matching together animal and client profiles, based on the attributes and matching preferences contained in those profiles. Each development team's ACM algorithm will be evaluated based on the variety, originality, and appropriateness of its matching rules and attributes.

In addition to the basic physical attributes (type of animal, breed, gender, age range, etc), the algorithm must use a *minimum* of 12 (twelve) separate and unique non-physical attributes. All the attributes defined in the animal and client profiles must be used by the algorithm's rules to compute the best matches. This includes a client's own attributes, as well as their matching preferences.

Delivery and Deployment

Delivery and deployment of the *cuACS* system must be *turnkey*. This means that the user must be able to install all the software for the project with a single command, build it with a single command, and then launch the project with a single command. Projects that do not conform to this requirement will not be graded.

Team Project

All Deliverables

All deliverables must be submitted in *cuLearn* before the due date and time. It is the responsibility of every team member to ensure that the work is submitted on time, and that the submission includes all the correct files.

Documentation deliverables must be submitted as a PDF document. They must be typed and legible, and they must be **professional** documents, including a cover page, page numbers, table of contents, table of figures, section numbers and names, etc. All diagrams and tables must be introduced and explained in the text. Documents that do not conform to these specifications will not be graded.

Coding deliverables must be delivered as a single tar file consisting of all source code, data files, and configuration scripts, as well as installation, build, and launch instructions. Coding submissions that cannot be downloaded into a fresh directory and un-tar'ed, then successfully built with a single command in the course VM, and successfully launched with a single command in the course VM, will not be graded.

The expected content of all deliverables will be posted in *cuLearn* and discussed in class.

Deliverable #1

The work submitted for this deliverable will consist of:

- a soft copy of the Requirements Analysis Document, in PDF format
- the implementation of selected feature(s)

Deliverable #2

The work submitted for this deliverable will consist of:

- a soft copy of the ACM algorithm design document, in PDF format
- an in-class presentation of the ACM algorithm design
- a soft copy of the presentation slides, in PDF format
- the implementation of selected feature(s)

Deliverable #3

The work submitted for this deliverable will consist of:

- a soft copy of the System Design Document, in PDF format
- the implementation of selected feature(s)

Deliverable #4

The work submitted for this deliverable will consist of:

- a soft copy of the revised ACM algorithm design document, in PDF format
- the implementation of selected feature(s)