C868 – Software Capstone Project Summary

Task 2 – Section C

Capstone Proposal Project Name:	Christie’s Companions
Student Name:	Michael C Irick

Contents

Task 2 Part C – C868 Software Development Capstone

Application Design and Testing

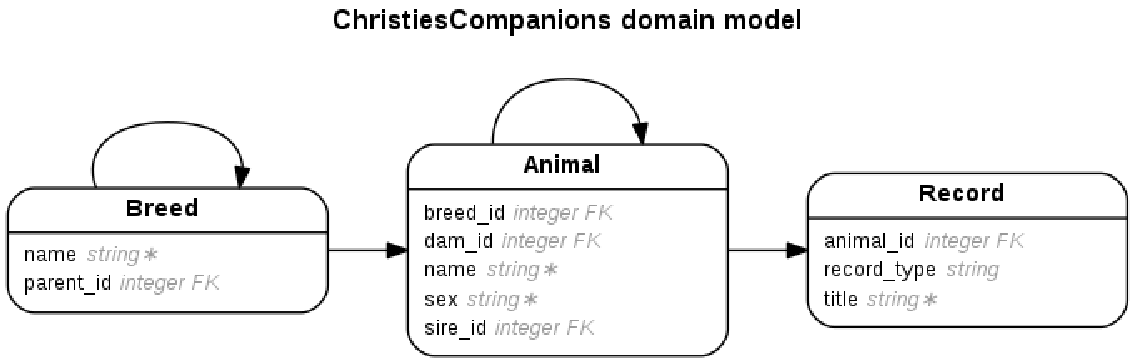
Design Document

Class Design

Below is the Entity-Relation Diagram that reflects the class design for the Christie’s Companions application. Due to the fairly standard implementation of MVC (Model/View/Controller) by the Ruby-on-Rails framework used in the project, only the classes involved in the core function of the application as a database will be shown and discussed.

The primary classes are Animal, Breed, and Record, while DashboardMessage and User are utilized in a secondary manner for presenting customized dashboard messages and restricting access to the administrator panel respectively. Each Breed has a one to many relationship with Animal representing all the animals belonging to that Breed, as well as a one-to-many relationship between Breed and itself representing an optional hierarchy of Breeds, i.e. Dog is the parent of Chihuahua. Animal has the inverse many-to-one relationship with Breed, two optional many-to-one relationships with Animal representing each Animal’s paternal and maternal relationships, and inverse one-to-many relationships with Animal for the Animal’s progeny, though only one of which will contain any relations. Record has a many-to-one relationship with Animal representing the various types of records, e.g. pedigree, vaccinations, that an Animal may have.

Figure 1. ERD



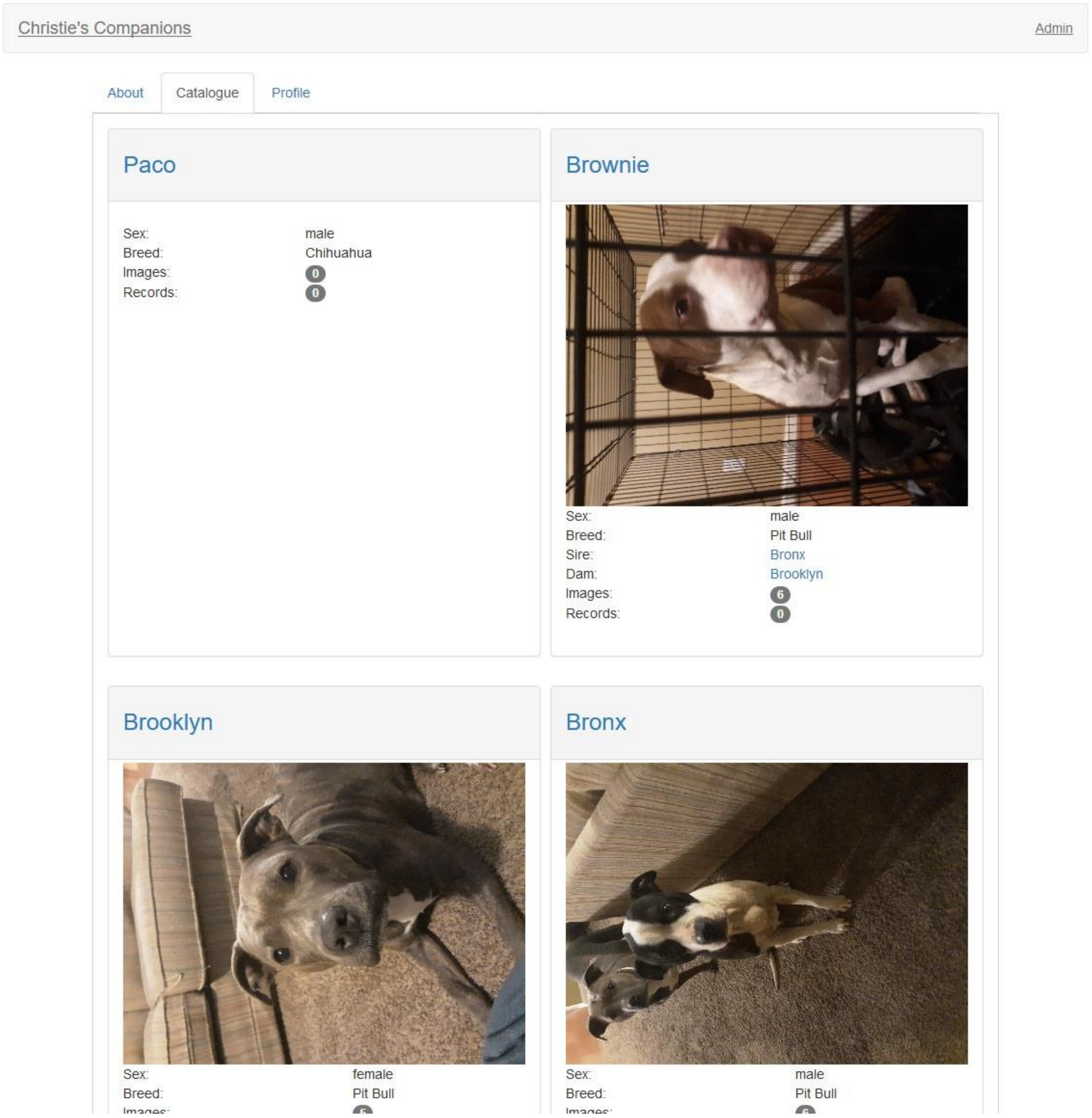
UI Design

Below are the low-fidelity versions of the user interface for the catalogue; high-fidelity user interface is out of scope for this project. Provided are the Catalogue and the Profile.

Figure 2 shows the main catalogue page. It shows each animal along with a profile picture, its sex, breed, sire, and dam, as well as how many images and how many records are available. By clicking the name of each animal, the view changes to the profile for that animal, see figure 3. At the top right hand side is a link to the admin panel, where new animals may be added.

Figure 3 shows for the selected animal the profile picture enlarged, sex, and breed. In addition to these basic attributes, there are 3 collapsible panels each for images, progeny, and records. These panels only appear if they contain data. The progeny panel uses the same components as the catalogue. The images panel shows larger versions of all attached images for the animal. Similarly, the records panel shows all the associated records with the animal.

Figure 2. Catalogue



Christie's Companions

Admin

About | Catalogue | Profile

Paco

Sex:	male
Breed:	Chihuahua
Images:	0
Records:	0

Brownie

Sex:	male
Breed:	Pit Bull
Sire:	Bronx
Dam:	Brooklyn
Images:	6
Records:	0

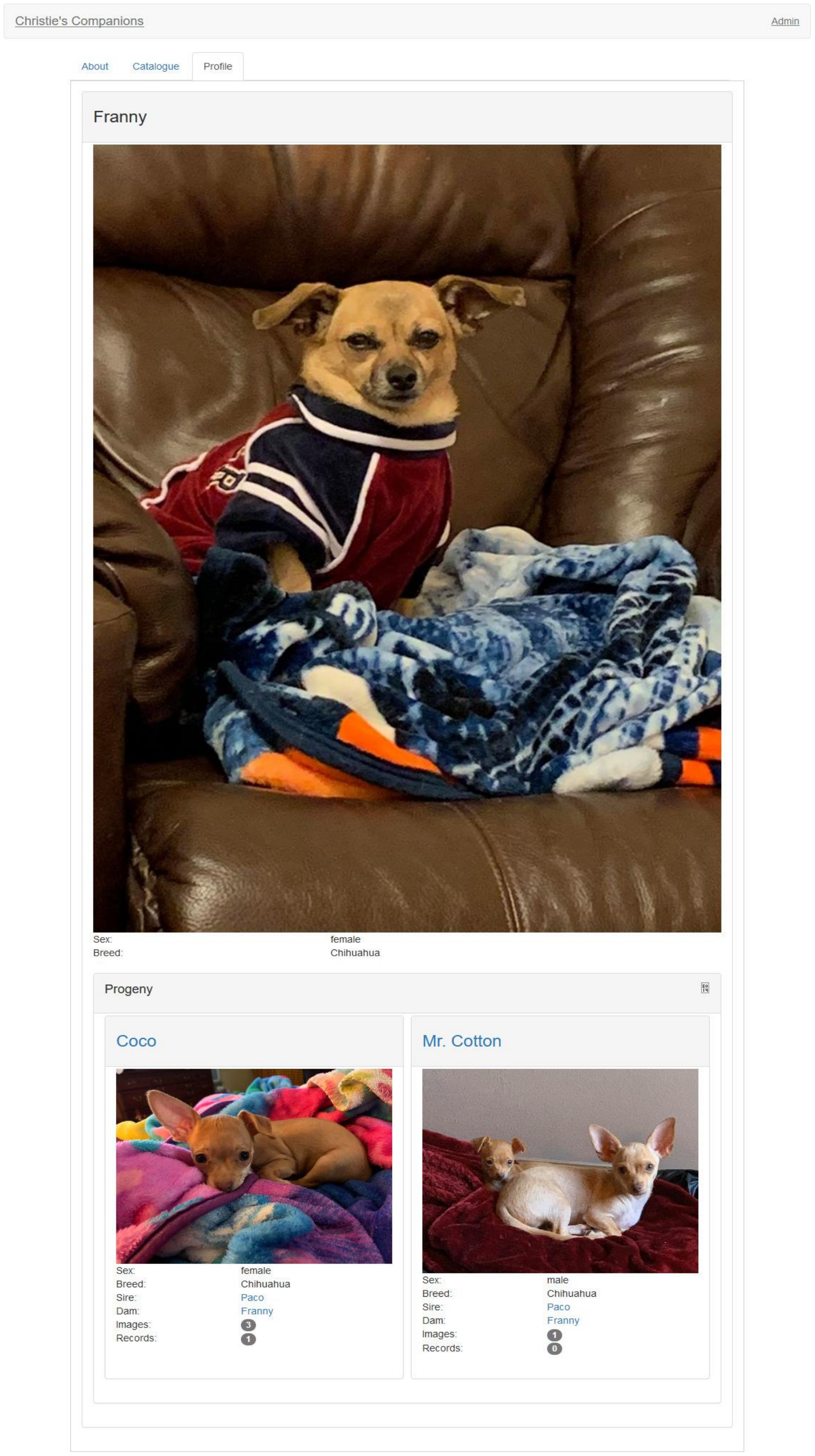
Brooklyn

Sex:	female
Breed:	Pit Bull

Bronx

Sex:	male
Breed:	Pit Bull

Figure 3. Profile



Christie's Companions Admin

About Catalogue Profile

Franny

Sex:	female
Breed:	Chihuahua

Progeny

Coco

Sex:	female
Breed:	Chihuahua
Sire:	Paco
Dam:	Franny
Images:	3
Records:	1

Mr. Cotton

Sex:	male
Breed:	Chihuahua
Sire:	Paco
Dam:	Franny
Images:	1
Records:	0

Unit Test Plan

Introduction

Purpose

To ensure that the validation rules for the classes for Animal, Breed, and Record are enforced correctly, a series of unit tests were developed using the MiniTest library for Ruby. Each unit test will create an invalid object of each class and then check for a specific field that is required. The test will pass only if the specific field is set.

Provide a brief description of the testing method(s) that you used and what the results it yielded. Also, what remediation was required if necessary and how it would be performed.

Overview

Since the application is designed to replace a paper-based record keeping system, it is vital to ensure that the required data is entered into the database. The validation rules get applied at the moment when a new record or changes to an existing record attempt to be persisted to the database which then sets an errors attribute on the record with messages of the failed validations; thus, by creating a new record missing the correct attribute and checking the error messages for the field to be tested, we can ensure that the database will only have valid data.

Being written in MiniTest, the test suite can be run from with the rails command which outputs the results of each test as well as a summary of all tests to the console.

Test Plan

Items

For each major class (Breed, Animal, and Record), there is a set of attributes that are required, and for each of which, there will be a test. The tests are as follows:

1. Breed
    1. Name
2. Animal
    1. Name
    2. Sex
    3. Breed
3. Record
    1. Animal
    2. Title

    This gives a total of six tests.

Features

To test each attribute, a new record object is created by calling the class method “new” and saving it to a variable. Then, the “save” method is called on the record object. Then the asset method is called which will cause the test to pass if the parameter evaluates to a “truthy” value. The parameter passed to asset is the negation of calling the “empty?” method on the records error messages mapped to the attribute tested.

Deliverables

On a successful test, a green dot (“.”) will print to the console; on a failure, a detailed error message and stack trace is shown. Then, after all tests have been run, a summary of successes to failures will be displayed.

Tasks

1. Setup the tests in the “test/models” directory according to MiniTest specification.
2. Create the code to be tested.

3. Run the test suite with the command "bundle exec rails test".
4. Evaluate the result and fix any failed tests.

Needs

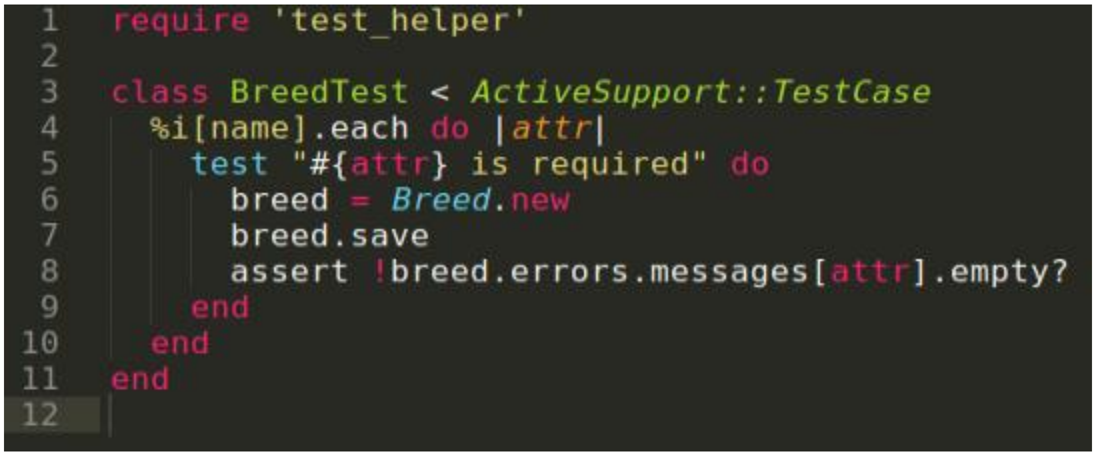
The testing library used was MiniTest version 5.13.0, though any version greater than or equal to 5.1 but less than 6.0 will suffice.

Pass/Fail Criteria

As described above, by checking whether the list of error messages for the tested attribute is empty, we can determine whether the record's attribute is valid. A failing test would feature an error message for the tested attribute.

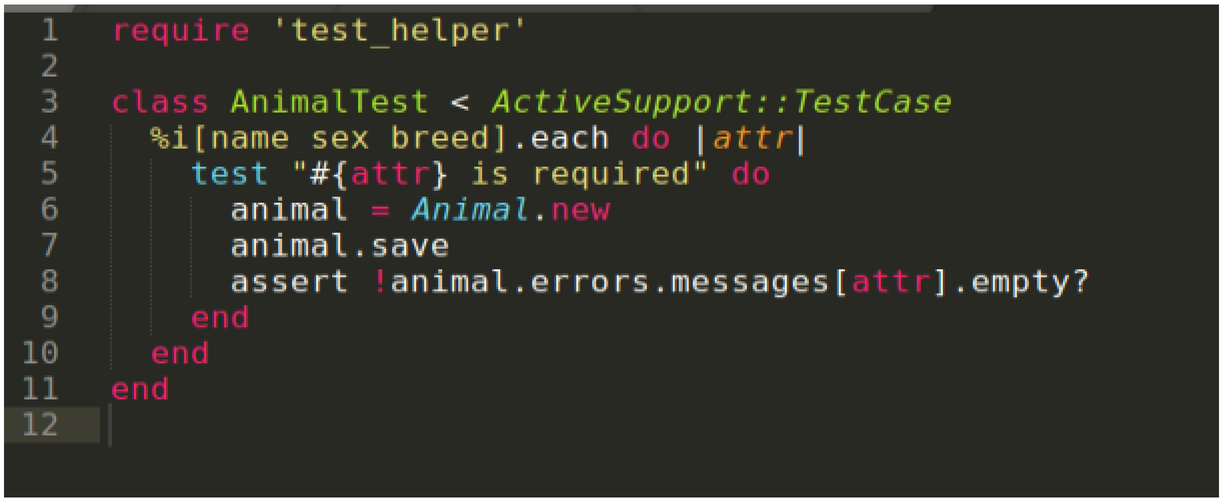
Specifications

Figure 4: screenshot of the Breed name test from file "test/models/breed_test.rb":



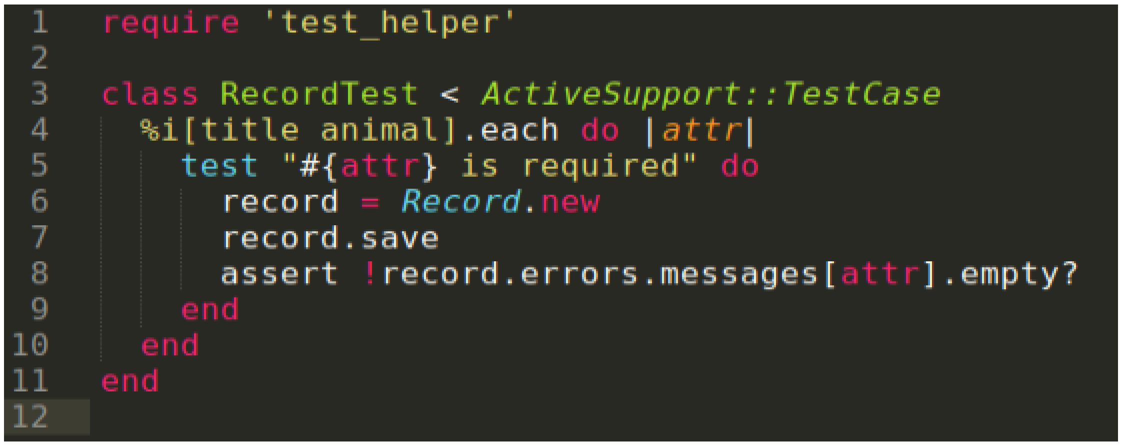
```ruby
require 'test_helper'

class BreedTest < ActiveSupport::TestCase
  %i[name].each do |attr|
    test "#{attr} is required" do
      breed = Breed.new
      breed.save
      assert !breed.errors.messages[attr].empty?
    end
  end
end

```

Figure 5: screenshot of the Animal name, breed, sex tests from file “test/models/animal_test.rb”:



```ruby
require 'test_helper'

class AnimalTest < ActiveSupport::TestCase
  %i[name sex breed].each do |attr|
    test "#{attr} is required" do
      animal = Animal.new
      animal.save
      assert !animal.errors.messages[attr].empty?
    end
  end
end
```

Figure 6: screenshot of the Record title, animal tests from file “test/models/record_test.rb”:



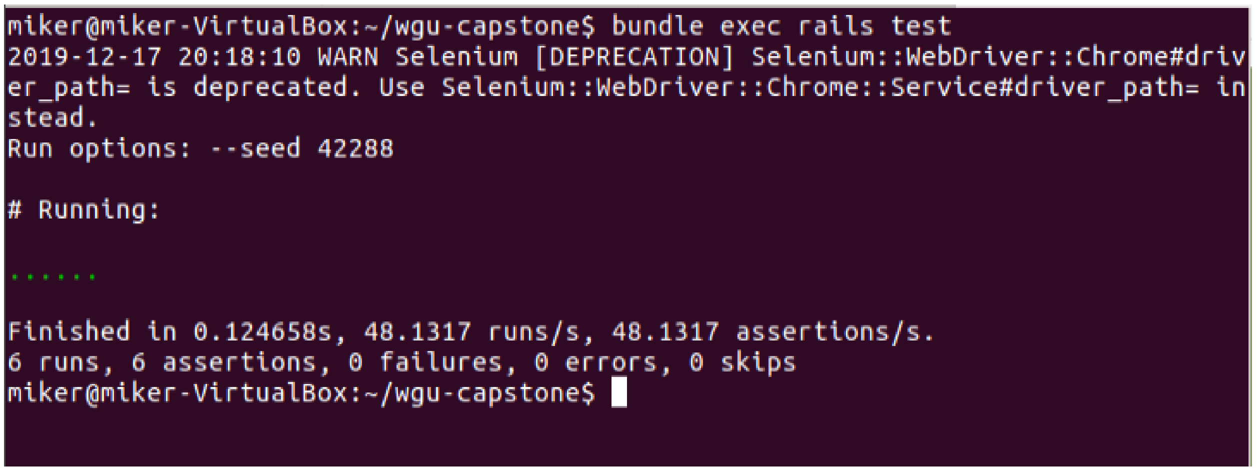
```ruby
require 'test_helper'

class RecordTest < ActiveSupport::TestCase
  %i[title animal].each do |attr|
    test "#{attr} is required" do
      record = Record.new
      record.save
      assert !record.errors.messages[attr].empty?
    end
  end
end
```

Procedures

As shown above, each test is created dynamically by iterating over a list of attributes, then setting up the test using MiniTest specifications. This approach allows each test to be consistent with the others for the same class while also reducing the amount of code that needs to be written and maintained.

Results

The following image shows the result of running the test suite on the command line. There are six green dots to indicate that all tests have passed as well as a summary that shows the percentage of passing tests as 100%.



```
miker@miker-VirtualBox:~/wgu-capstone$ bundle exec rails test
2019-12-17 20:18:10 WARN Selenium [DEPRECATION] Selenium::WebDriver::Chrome#driv
er_path= is deprecated. Use Selenium::WebDriver::Chrome::Service#driver_path= in
stead.
Run options: --seed 42288

# Running:

......

Finished in 0.124658s, 48.1317 runs/s, 48.1317 assertions/s.
6 runs, 6 assertions, 0 failures, 0 errors, 0 skips
miker@miker-VirtualBox:~/wgu-capstone$ 
```

Source Code

The source code can be found in the archive file wgu-capstone-master.zip.

Link to Live Version

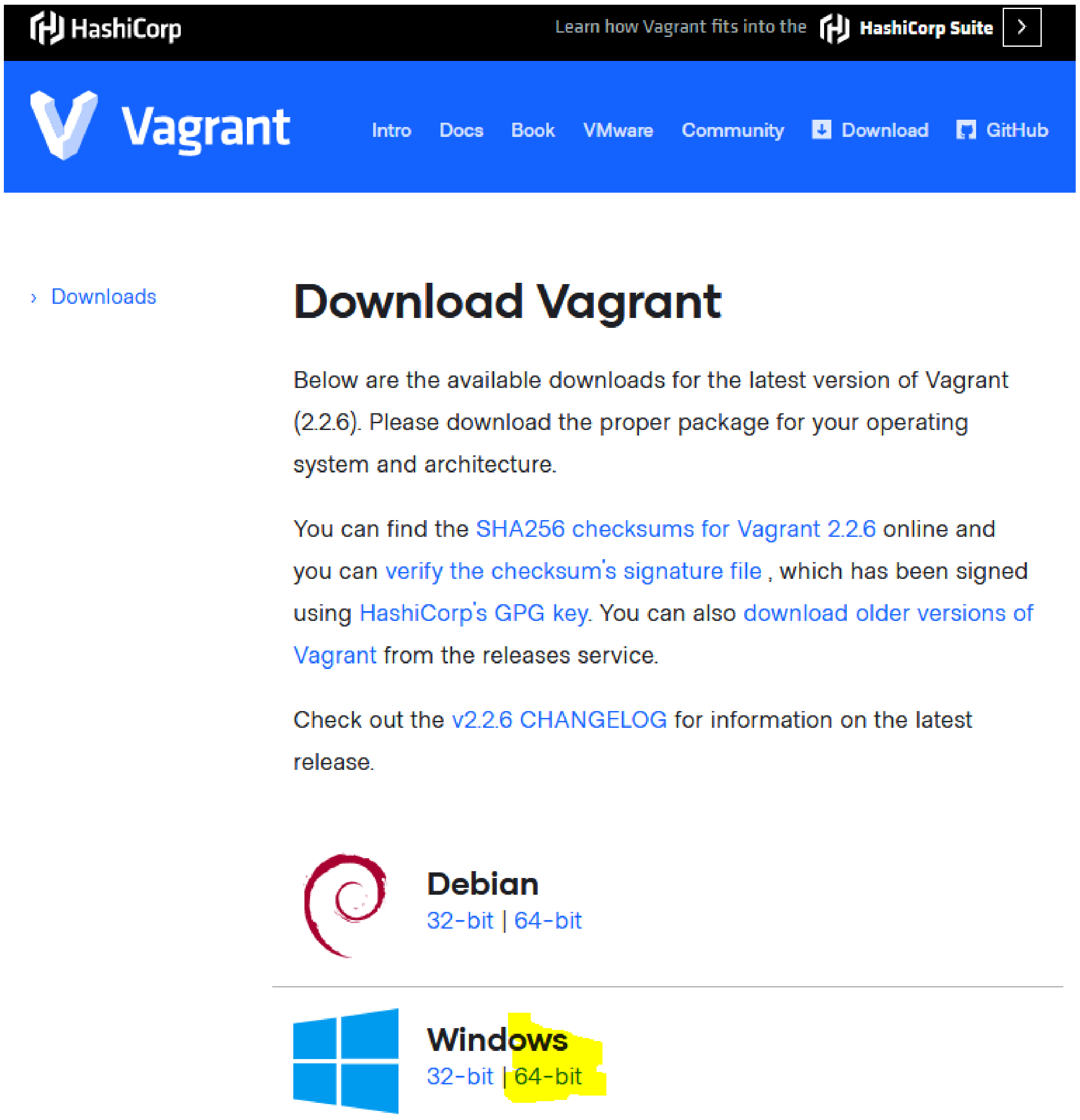
The live version can be found at http://christiescompanions.hatchie.studio

User Guide

This guide will include instructions on how to install the application for development as well as login to manage the data of the application.

Installation and Startup

First, we must install Vagrant to setup our development environment. Vagrant is a tool for managing virtual machine configuration and setup. Download the latest 64-bit version of Vagrant from https://www.vagrantup.com/downloads.html.



Download Vagrant

Below are the available downloads for the latest version of Vagrant (2.2.6). Please download the proper package for your operating system and architecture.

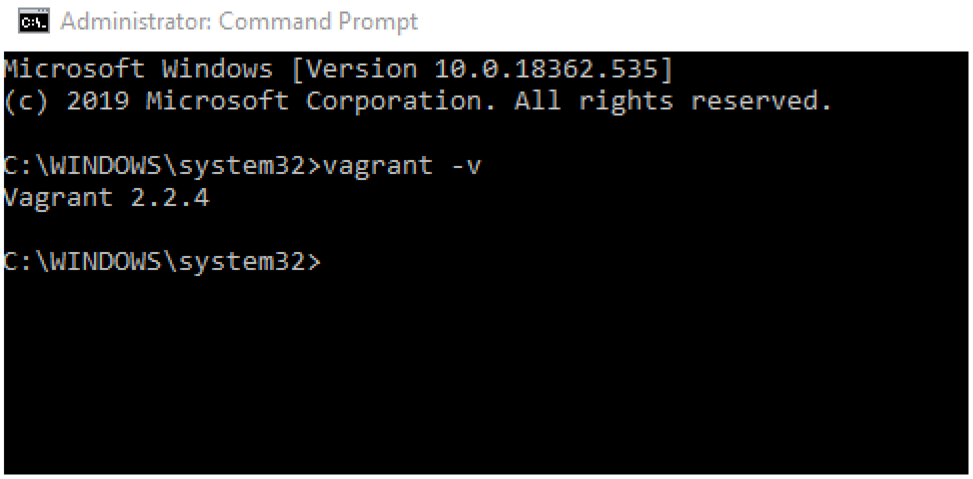
You can find the SHA256 checksums for Vagrant 2.2.6 online and you can verify the checksum's signature file , which has been signed using HashiCorp's GPG key. You can also download older versions of Vagrant from the releases service.

Check out the v2.2.6 CHANGELOG for information on the latest release.

Debian
32-bit | 64-bit

Windows
32-bit | 64-bit

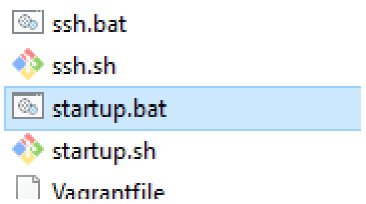
After downloading and running the installer with the default options, you may verify that the installation was succesful by opening a command prompt as administrator and typing "vagrant -v" to display the version:



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18362.535]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>vagrant -v
Vagrant 2.2.4

C:\WINDOWS\system32>
```

Next, run the appropriate installation file in the source folder:

- Windows: run the startup.bat file
- Linux: run the startup.sh script

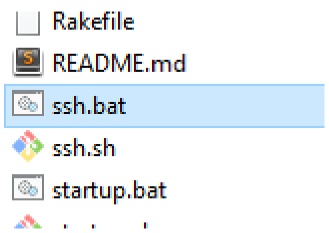


The installation of the development environment may take sometime (15-20 minutes). It will setup an Ubuntu virtual machine running PostgreSQL with the Ruby environment for the web application. When the install is complete, the program will exit. When the virtual machine is not running, run the startup file again to launch it.

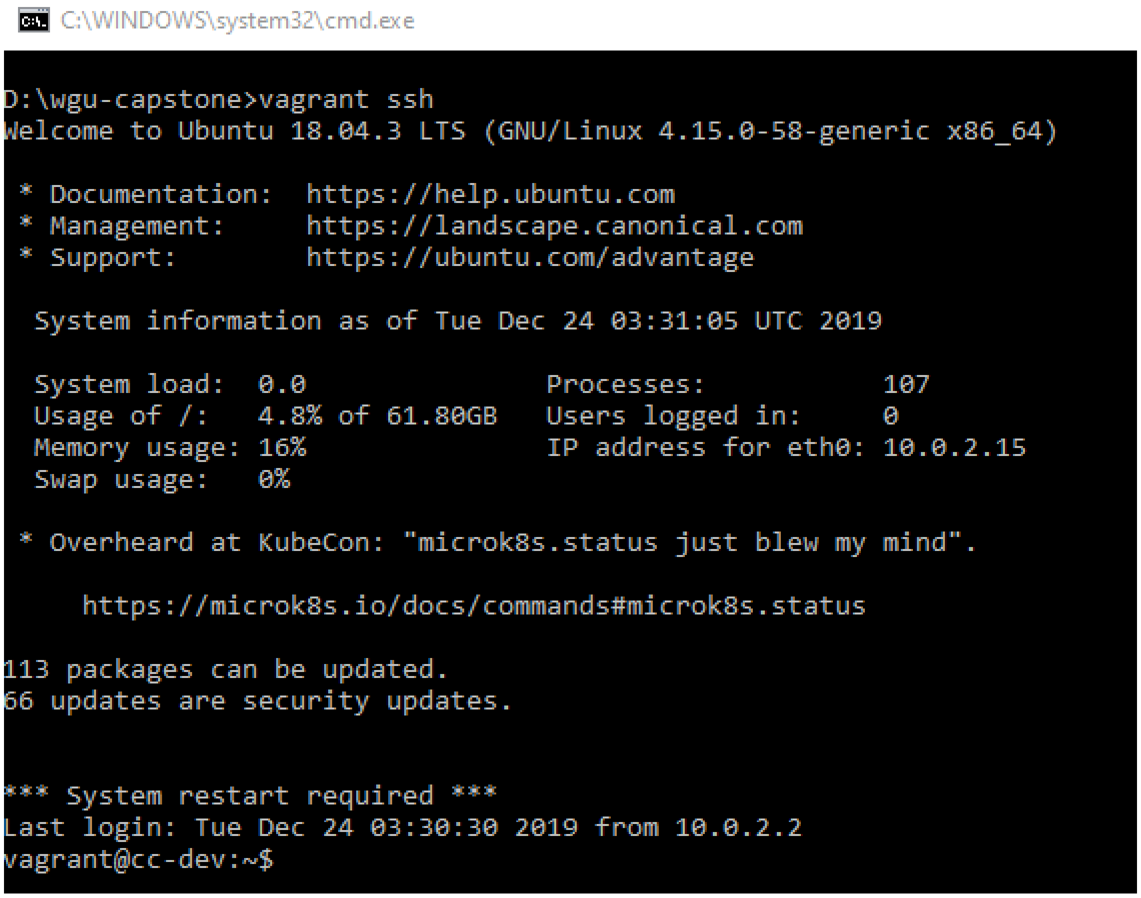
Development

To open the development console, run the appropriate file from the source folder:

- Windows: run ssh.bat file
- Linux: run the ssh.sh file

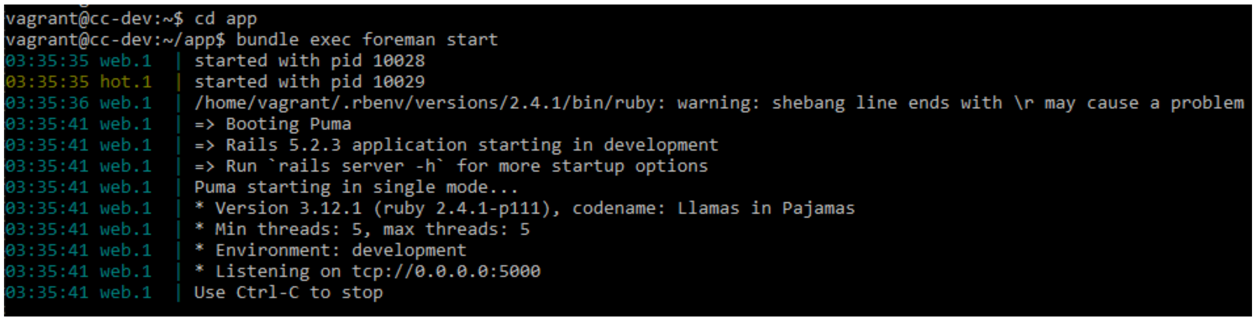


Development Console:



```
C:\WINDOWS\system32\cmd.exe

D:\wgu-capstone>vagrant ssh
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Dec 24 03:31:05 UTC 2019

  System load:  0.0                Processes:           107
  Usage of /:   4.8% of 61.80GB   Users logged in:     0
  Memory usage: 16%                IP address for eth0: 10.0.2.15
  Swap usage:   0%

 * Overheard at KubeCon: "microk8s.status just blew my mind".

     https://microk8s.io/docs/commands#microk8s.status

113 packages can be updated.
66 updates are security updates.


*** System restart required ***
Last login: Tue Dec 24 03:30:30 2019 from 10.0.2.2
vagrant@cc-dev:~$
```

From here, you can start the web application in development mode by navigating to the app directory, then running “bundle exec foreman start”.

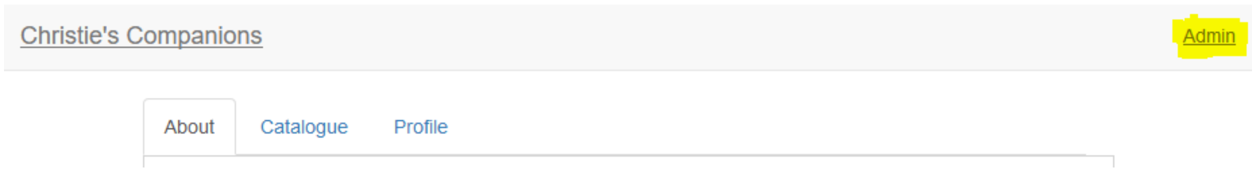


```
vagrant@cc-dev:~$ cd app
vagrant@cc-dev:~/app$ bundle exec foreman start
03:35:35 web.1  | started with pid 10028
03:35:35 hot.1  | started with pid 10029
03:35:36 web.1  | /home/vagrant/.rbenv/versions/2.4.1/bin/ruby: warning: shebang line ends with \r may cause a problem
03:35:41 web.1  | => Booting Puma
03:35:41 web.1  | => Rails 5.2.3 application starting in development
03:35:41 web.1  | => Run `rails server -h` for more startup options
03:35:41 web.1  | Puma starting in single mode...
03:35:41 web.1  | * Version 3.12.1 (ruby 2.4.1-p111), codename: Llamas in Pajamas
03:35:41 web.1  | * Min threads: 5, max threads: 5
03:35:41 web.1  | * Environment: development
03:35:41 web.1  | * Listening on tcp://0.0.0.0:5000
03:35:41 web.1  | Use Ctrl-C to stop
```

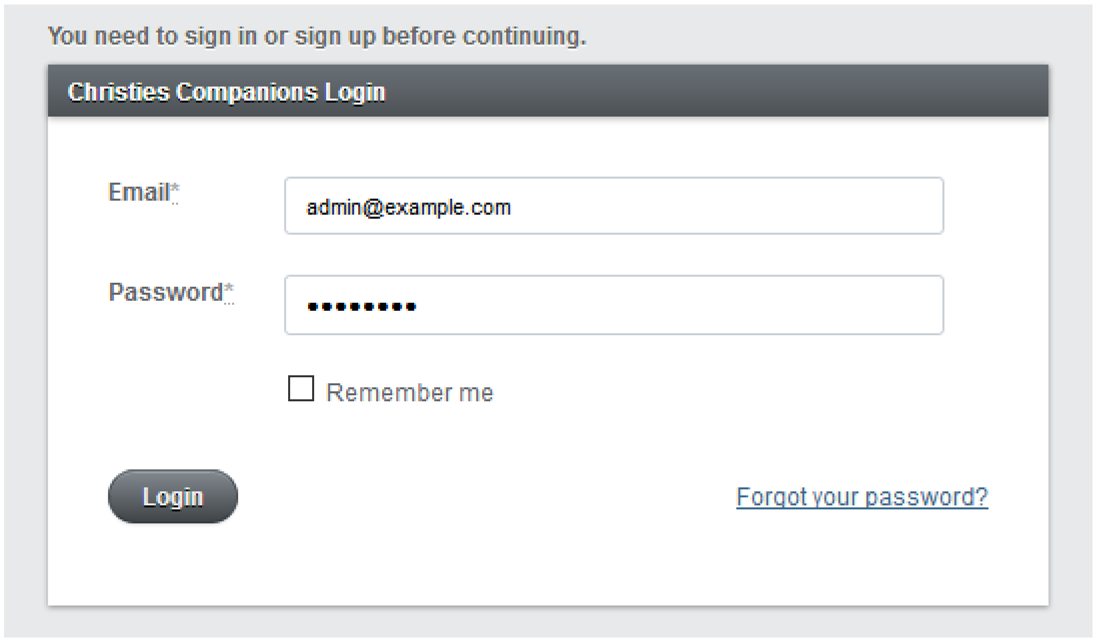
Then, point your web browser to http://localhost:5000.

Administration

After navigating to the front-page, clicking on the Admin link in the top right corner will bring you to the sign in page.



The default credentials are email address `admin@example.com` and password `password`.

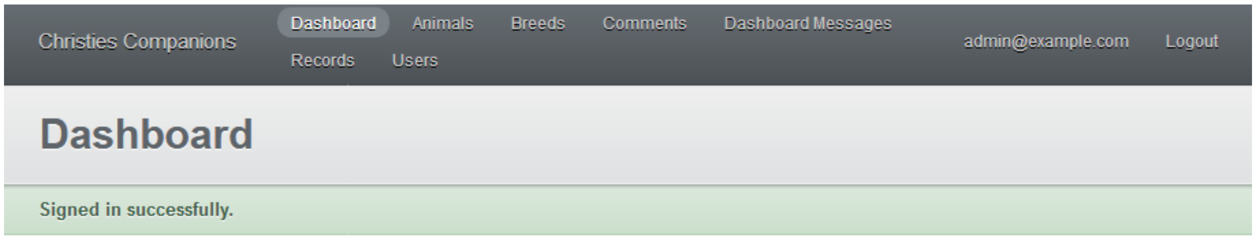

You need to sign in or sign up before continuing.

Christies Companions Login

Email* admin@example.com

Password* •••••••••

The administration panel allows authorized users to edit the various types of data in the application, as well as add and remove access to users. Each of the index views can be accessed via a link in the top navigation bar.

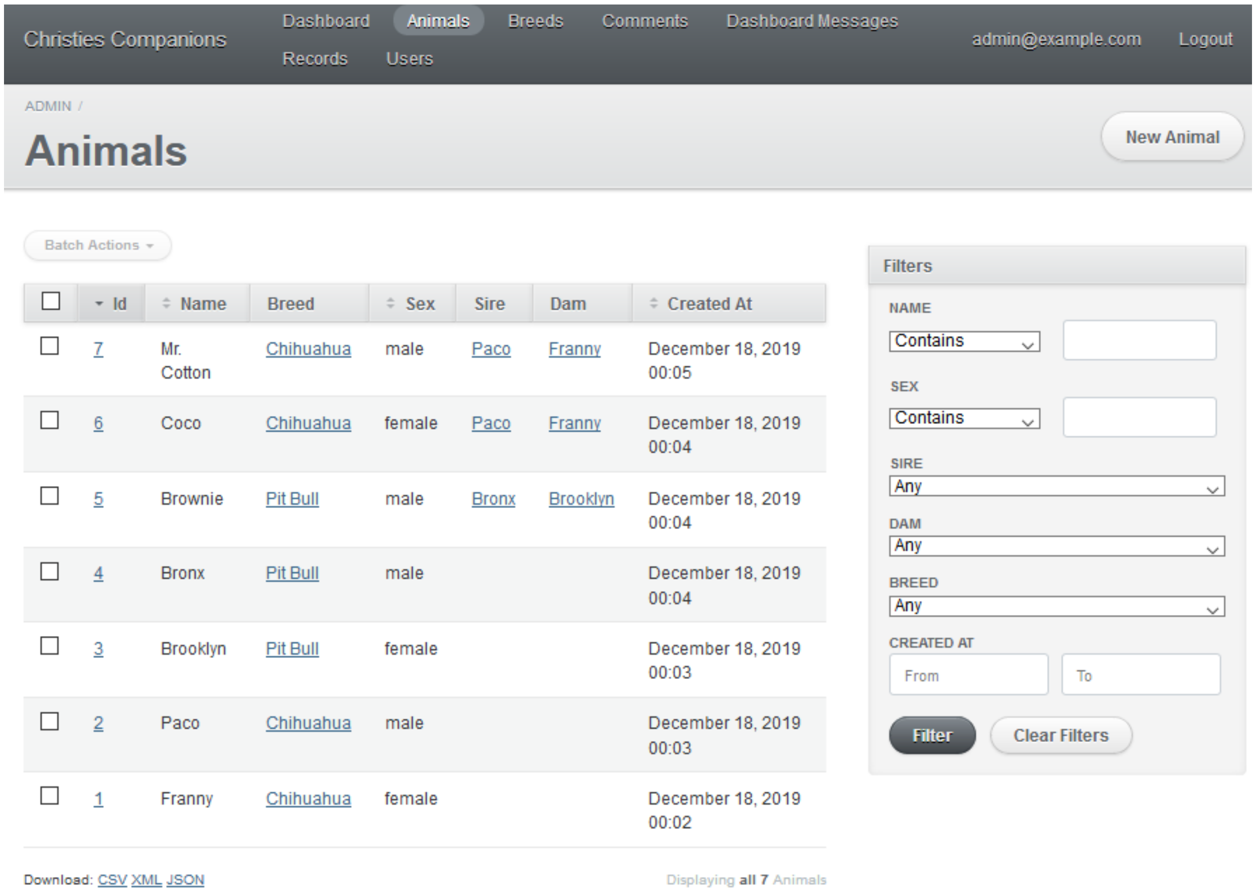


Dashboard

Signed in successfully.

The index views show a quick high-level view of each class of data, for example, Animal. It features links to show more detailed information about each object by clicking on the ID number.

One can also delete multiple objects by checking the left-hand box on each row, and selecting delete selected from the batch actions button. Filters are also available as well as download capability in CSV, XML, and JSON allowing simple reporting functionality, for example, finding all the animals of a particular sire.



Animals

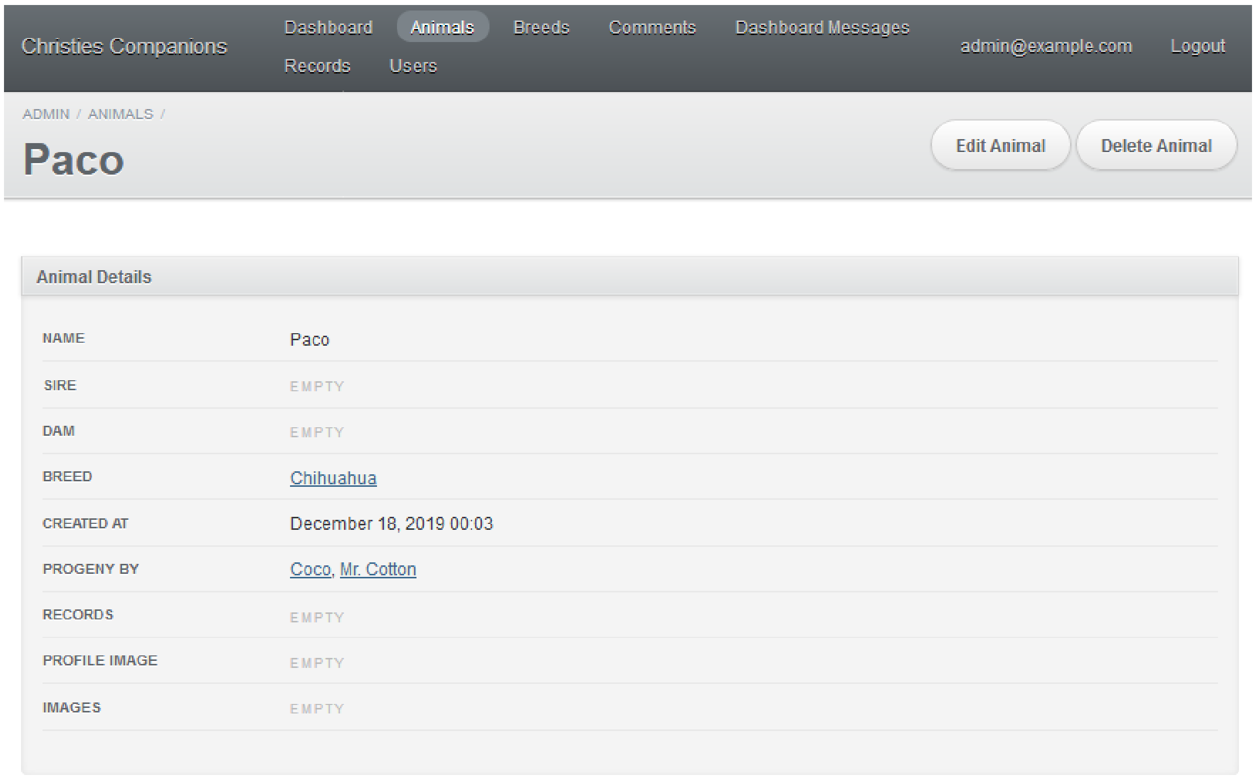
☐	Id	Name	Breed	Sex	Sire	Dam	Created At
☐	7	Mr. Cotton	Chihuahua	male	Paco	Franny	December 18, 2019 00:05
☐	6	Coco	Chihuahua	female	Paco	Franny	December 18, 2019 00:04
☐	5	Brownie	Pit Bull	male	Bronx	Brooklyn	December 18, 2019 00:04
☐	4	Bronx	Pit Bull	male			December 18, 2019 00:04
☐	3	Brooklyn	Pit Bull	female			December 18, 2019 00:03
☐	2	Paco	Chihuahua	male			December 18, 2019 00:03
☐	1	Franny	Chihuahua	female			December 18, 2019 00:02

Download: CSV XML JSON

Displaying **all 7** Animals

NAME: Contains
SEX: Contains
SIRE: Any
DAM: Any
BREED: Any
CREATED AT: From To

Clicking on the ID number on the index page, will bring you to the show view, which shows more data than the index view, for example, an animal's images and records, as well as administrator comments for each object. There are buttons here to delete the object as well as edit the object.

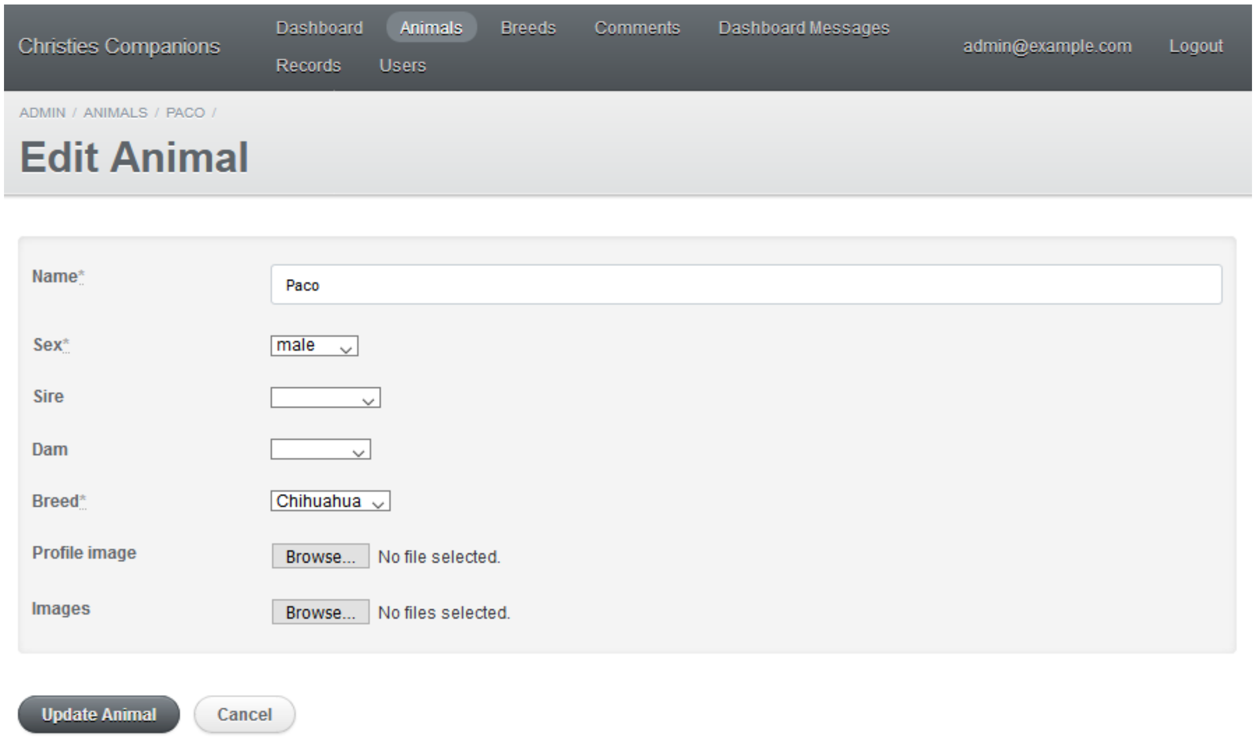


Paco

Animal Details

NAME	Paco
SIRE	EMPTY
DAM	EMPTY
BREED	Chihuahua
CREATED AT	December 18, 2019 00:03
PROGENY BY	Coco, Mr. Cotton
RECORDS	EMPTY
PROFILE IMAGE	EMPTY
IMAGES	EMPTY

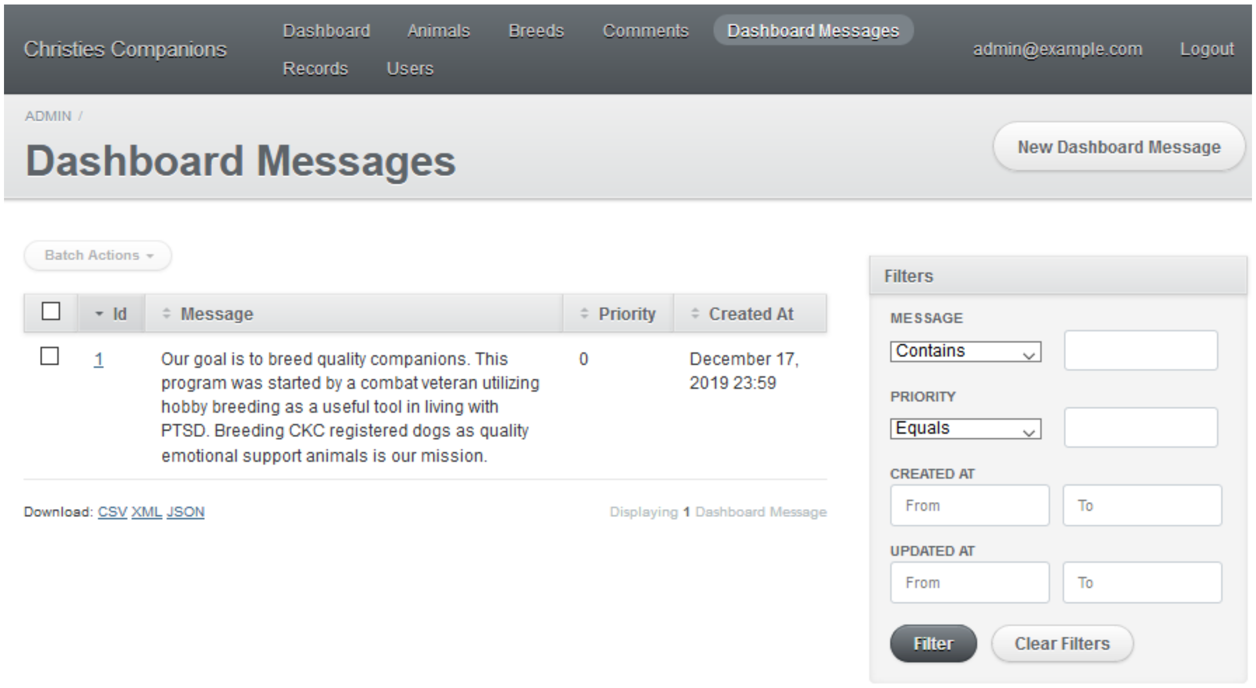
The edit view contains a form with validations specific to that class.



Edit Animal

Name*	Paco
Sex*	male
Sire	
Dam	
Breed*	Chihuahua
Profile image	Browse... No file selected.
Images	Browse... No files selected.

Of particular note are the Dashboard Messages. This changes the text of the About tab on the public catalog. They are ordered by their priority attribute.



Dashboard Messages

	Id	Message	Priority	Created At
☐	1	Our goal is to breed quality companions. This program was started by a combat veteran utilizing hobby breeding as a useful tool in living with PTSD. Breeding CKC registered dogs as quality emotional support animals is our mission.	0	December 17, 2019 23:59

Displaying 1 Dashboard Message

MESSAGE: Contains
PRIORITY: Equals
CREATED AT: From / To
UPDATED AT: From / To