

# National College of Ireland

BSHCSDE4

Software Development

Academic Year 2020/2021

Lorcan Murray

x16105834

x16105834@student.ncirl.ie

PetRescue

Technical Report

## Contents

Executive Summary.....	5
1.0 Introduction .....	5
1.1. Background .....	5
1.2. Aims.....	6
1.3. Technology.....	7
2.0 System.....	8
2.1. Requirements.....	8
2.1.1. Functional Requirements.....	8
2.1.1.1. Use Case Diagram .....	9
2.1.1.2. Requirement 1: User Registration .....	10
2.1.1.3. Description & Priority.....	10
2.1.1.4. Use Case .....	10
Requirement 2: User Login .....	12
2.1.1.5. Description & Priority.....	12
2.1.1.6. Use Case .....	12
2.1.1.7. Requirement 3: Report Pet .....	14
2.1.1.8. Description & Priority.....	14
2.1.1.9. Use Case .....	14
2.1.1.10. Requirement 4: Delete Report.....	16
2.1.1.11. Description & Priority.....	16
2.1.1.12. Use Case .....	16
2.1.1.13. Requirement 5: Edit Report .....	18
2.1.1.14. Description & Priority.....	18
2.1.1.15. Use Case .....	18
2.1.1.16. Requirement 6: View Gallery.....	20
2.1.1.17. Description & Priority .....	20
2.1.1.18. Use Case .....	20
2.1.1.19. Requirement 7: View Individual Profile .....	22
2.1.1.20. Description & Priority .....	22
2.1.1.21. Use Case .....	22
2.1.1.22. Requirement 8: Comment.....	24
2.1.1.23. Description & Priority .....	24
2.1.1.24. Use Case .....	24

2.1.1.25.	Requirement 9: Delete Comment .....	26
2.1.1.26.	Description & Priority .....	26
2.1.1.27.	Use Case .....	26
2.1.1.28.	Requirement 10: Send Direct Message .....	28
2.1.1.29.	Description & Priority .....	28
2.1.1.30.	Use Case .....	28
2.1.1.31.	Requirement 11: View Sent Messages .....	30
2.1.1.32.	Description & Priority .....	30
2.1.1.33.	Use Case .....	30
2.1.1.34.	Requirement 12: Logout.....	32
2.1.1.35.	Description & Priority .....	32
2.1.1.36.	Use Case .....	32
2.1.2.	Data Requirements .....	34
2.1.3.	User Requirements .....	36
2.1.4.	Environmental Requirements .....	37
2.1.4.1.	Client .....	37
2.1.4.2.	Server .....	37
2.1.4.3.	Development.....	37
2.1.5.	Usability Requirements.....	38
2.2.	Design & Architecture .....	38
2.3.	Implementation .....	41
2.4.	Graphical User Interface (GUI).....	50
2.5.	Testing.....	54
2.5.1.	Unit Testing.....	54
2.5.2.	Functional Testing.....	54
2.6.	Evaluation .....	54
3.0	Conclusions .....	56
4.0	Further Development or Research .....	57
5.0	References .....	57
6.0	Appendices.....	58
6.1.	Project Proposal.....	58
Objectives.....		60
Background .....		60
Technical Approach.....		61
Special Resources Required .....		61
Project Plan .....		62

Gantt Chart.....	63
Technical Details .....	64
Evaluation .....	64
6.2. Reflective Journals .....	65
October .....	65
Week 1 (26/09): .....	65
Week 2 (03/10): .....	65
Week 3 (10/10): .....	65
Week 4 (17/10): .....	65
Week 5 (24/10): .....	65
Reflection .....	66
November .....	66
Week 6 (31/10): .....	66
Week 7 (07/11): .....	66
Week 8 (14/11): .....	66
Week 9 (21/10): .....	66
Reflection .....	67
December.....	67
Week 10 (28/11): .....	67
Week 11 (05/12): .....	67
Week 12 (12/12): .....	67
Week 13 (19/12): .....	67
Week 14 (26/12): .....	67
Reflection .....	67
January .....	68
Week 15 (02/01): .....	68
Week 16 (09/01): .....	68
Week 17 (16/01): .....	68
Week 18 (23/01): .....	68
Reflection .....	68
February.....	69
Week 19 (30/01): .....	69
Week 20 (06/02): .....	69
Week 21 (13/02): .....	69
Week 22 (20/02): .....	69
Reflection .....	69

March .....	70
Week 23 (27/02): .....	70
Week 24 (06/03): .....	70
Week 25 (13/03): .....	70
Week 26 (20/03): .....	70
Week 27 (27/03): .....	71
Reflection .....	71
April .....	71
Week 28 (03/04): .....	71
Week 29 (10/04): .....	71
Week 30 (17/04): .....	71
Week 31 (24/04): .....	72
Reflection .....	72
6.3. Other materials used .....	73
6.3.1. Pet Rescue Test Plan .....	73
6.3.2. PetRescue System Evaluation Tasks .....	79
6.3.3. PetRescue System Evaluation Survey .....	80

## Executive Summary

The following document aims to serve as a technical report to outline in detail the specifications of 'PetRescue'; a Rich Internet Application (RIA) which has been developed as part of the year four software project module in the Software Development specialisation cohort.

PetRescue is a pet finder web application which offers registered, signed-in users the ability to report a missing, found, or stolen pet in the simplest possible form, while also providing a comments section under each missing pet's profile, allowing users to update each other instantly on any sightings or other helpful information relating to that pet. Leaving a comment automatically alerts the owner of the report via automated email, so no comment goes unseen. The application also provides direct messaging (DM) functionality, with a personal inbox to send and view private messages to and from other registered site users.

This report aims to highlight the technical specifications of the project, such as the architectural specifications which will be illustrated through visual representations of the system such as use case diagrams and Entity Relationship diagrams. The report also aims to outline the development process undertaken, as well as discuss what has been achieved in the final product compared to what was initially envisaged, and what future plans are in place for PetRescue.

It is intended that by the end of this document the reader will have a thorough grasp of the PetRescue application in its entirety from both a high and low-level perspective, and an understanding of what difficulties were encountered during the development process.

## 1.0 Introduction

### 1.1. Background

The idea for this project stems from a love of animals and a desire to help reduce the issue faced by pet-owners of losing a pet. It is a near-daily occurrence for people to encounter a wayward animal somewhere during the day, however it is not always possible, or advisable, to approach these animals and try to assist in returning them home. The PetRescue application aims to eliminate, or at the very least reduce the need to approach a stray animal in order to offer some assistance, therefore eliminating any potential risk involved in dealing with a possibly scared and unpredictable animal. To add to this, Ireland in recent years has seen a worrying spike in the number of reported animal thefts. These animals are then transported overseas and sold, often for cruel and illegal acts such as dog fighting. A resource to address these problems and help in some small way is something that offers real value, both to users of the application and to the animals it eventually helps rescue.

## 1.2. Aims

Pets going missing is far from a new phenomenon, however as discussed previously, in recent years pets (dogs in particular) are being stolen from their homes and sold for profit; a trend which has only increased over the last year. According to an RTE report on dog thefts, “animal and dog groups believe suspected dog thefts are linked with a bigger demand for dogs during the covid-19 pandemic” (McCormack, 2021). This is a trend in Ireland which is rising and can only be reduced by increasing awareness. Having a single resource such as PetRescue could greatly improve awareness of individual missing pets and eventually possibly reduce the trend of stolen pets by deterring would-be thieves.

When researching similar applications that already exist, very few could be found, with the only real competition being ‘lostandfoundpets.ie’. This site offers similar functionality to PetRescue, however is visually unappealing with harsh colours, and laden with adverts. It also falls short in terms of user experience. The site does not offer user registration and therefore contact details must be entered at the time of reporting a lost or found pet, which is a cumbersome task when reporting a found pet in particular, which the user may not be all that emotionally invested in. Another area PetRescue improves upon is contacting the report. While it is possible to contact a user on lostandfoundpets.ie, it must be done either by emailing the user from the site, or by phoning them. PetRescue does not display contact information on the site, and instead offers users a private inbox where messages can be sent between users and personal details exchanged if necessary. PetRescue also offers a comment section on each report, allowing users to quickly update a post with useful information such as a sighting of the animal. Once a comment is left, the owner of the report is instantly notified via automated email. Commenting is not something which is offered by lostandfoundpets.ie.

The overall aim for this project was to offer a platform which allowed registered users to log in and report a found or lost pet via the simplest possible process, while being visually appealing to the user to encourage interaction with the site. The following is a summation of the functionality offered by PetRescue:

- User Authentication
  - Register
  - Login
  - Logout
  - Password Recovery
- Report Missing, Found or Stolen
  - Description and image upload
  - Edit details of an existing report (user-specific)
  - Delete an existing report (user-specific)
- Gallery View (all reported pets)
- Pet Profile View (individual reported pet)
- Commenting
  - Post comment
  - Delete comment (user-specific)

- Generate email notification when comment is posted
- Messaging
  - Inbox
  - Create and send new message
  - View sent messages

### 1.3. Technology

**Frontend:** The user interface consists primarily of HTML, CSS, and JavaScript, with Bootstrap utilised for styling. The intention was for a stylish, simplistic UI to attract the user to interacting with the application and it is believed that this has been accomplished.

**Backend:** Initially the intention for this project was for the backend to be written in Python3. It was felt that having studied primarily java over the past 3 to 4 years, programming in Python would present new and exciting challenges. However, during development several significant issues arose which were proving costly in terms of time. After struggling with these issues for some time and following a helpful conversation with the project supervisor Rejwanul Haque, a decision was made to start over with the project in a different language. The chosen language was Ruby. Prior experience of Ruby was non-existent until the second semester of year four but having briefly used Ruby for another module and finding it much more workable, the decision was made and within a week the new Ruby version of the project had surpassed its Python predecessor in terms of progress.

**Framework:** As stated previously, the chosen programming language was changed late into the project, and with the change of programming language came a change to the framework. The initial framework was Django3, however when the language was changed to ruby, Rails was naturally chosen as the new framework. It is believed the issues faced initially with the Python project stemmed from an understanding (or lack thereof) of the Django framework. Working with Rails has proven to be much more fruitful, while still new and challenging.

**Version Control:** GitHub was utilised for version control. A connection was established from the Ubuntu terminal (for Windows) to the project's repository on github and each change that was made to the application during development was pushed and merged back to the master branch. Using version control proved invaluable on at least two occasions, where otherwise the changes made could have been detrimental to the project, however rollbacks were implemented, and the project recovered.

**Database:** This project utilises a PostgreSQL database. PostgreSQL was chosen due to its robust performance and reliability, as well as being a very popular object-relational database choice for Ruby on Rails projects in general.



**User Authentication:** Devise is used to handle all aspects of user authentication for PetRescue, including registration, log in and out functionality and token-based password recovery.

**Image Upload:** The process of reporting a missing or lost pet requires a user to submit a form with details relating to the missing animal, among those details is an image which is uploaded to the database. To handle these file uploads the Shrine gem was used, which allows implementation of validation rules for file uploads to restrict uploads to specific sizes and file types. For PetRescue, this was limited to jpeg, png and webp image files of no more than 5mb.

## 2.0 System

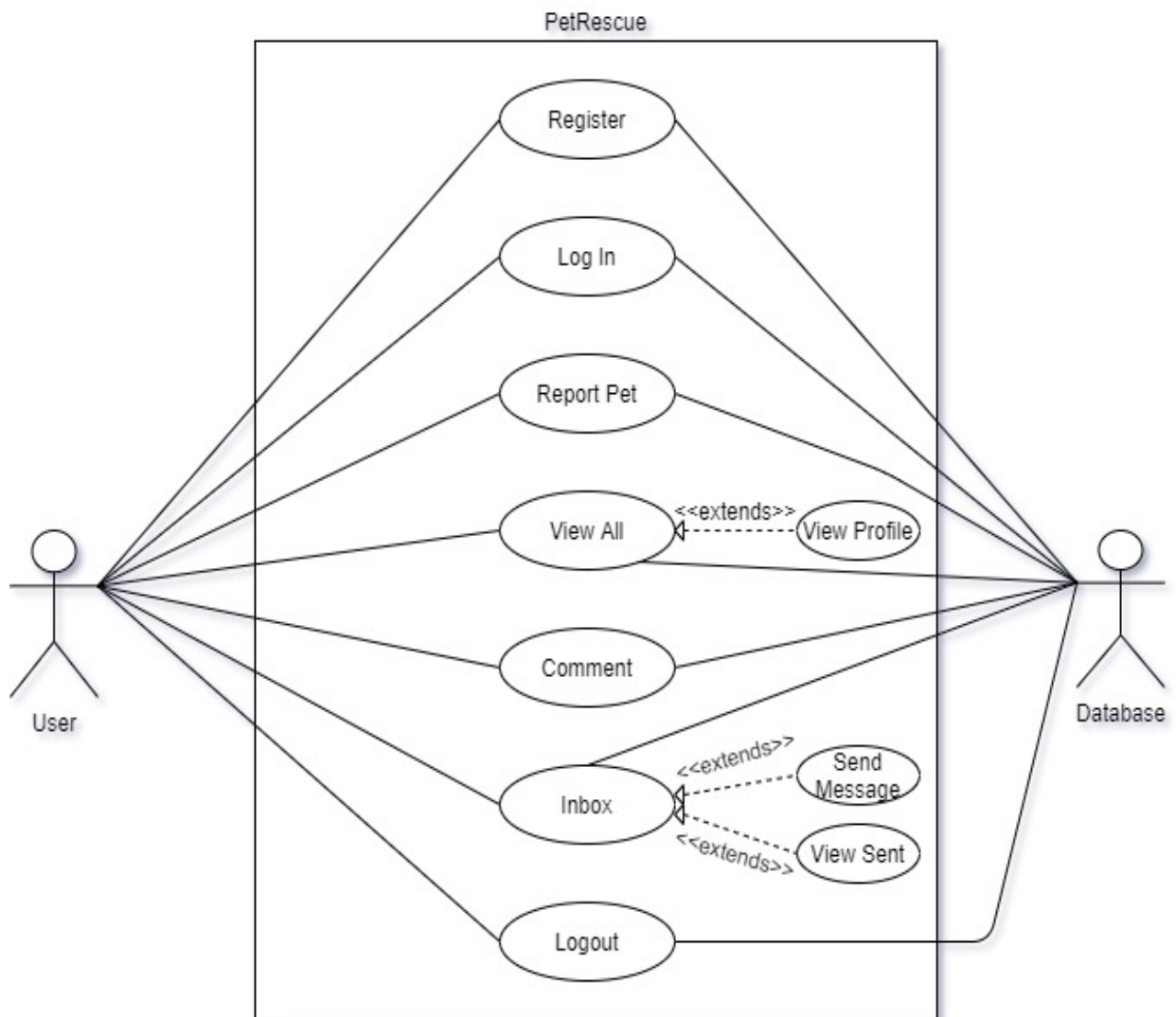
### 2.1. Requirements

All requirements for the system are detailed below. These requirements are all verifiable via user actions and should not require any prior experience of system usage.

#### 2.1.1. Functional Requirements

1. Landing page must offer user registration functionality.
2. Landing page must offer user login functionality.
3. Registered user must be able to report a missing pet.
4. Registered user must be able to delete pet report.
5. Registered user must be able to edit details of their pet report.
6. Registered user must be able to view gallery of missing pets.
7. Registered user must be able to view profile of individual missing pet.
8. Registered user must have the ability to comment on a pet report.
9. Registered user must be able to delete comment on a pet report.
10. Registered user must have the ability to browse inbox and view messages within.
11. Registered user must have the ability to send a private direct message to another registered user.
12. Registered user must have the ability to view sent messages.
13. Currently logged-in user must have the ability to log out.

### 2.1.1.1. Use Case Diagram



### 2.1.1.2. Requirement 1: User Registration

### 2.1.1.3. Description & Priority

This use case describes the user registration process. This process is vital as no further system interactions can take place with an unregistered user. Priority one.

### 2.1.1.4. Use Case

Register an unregistered user's details.

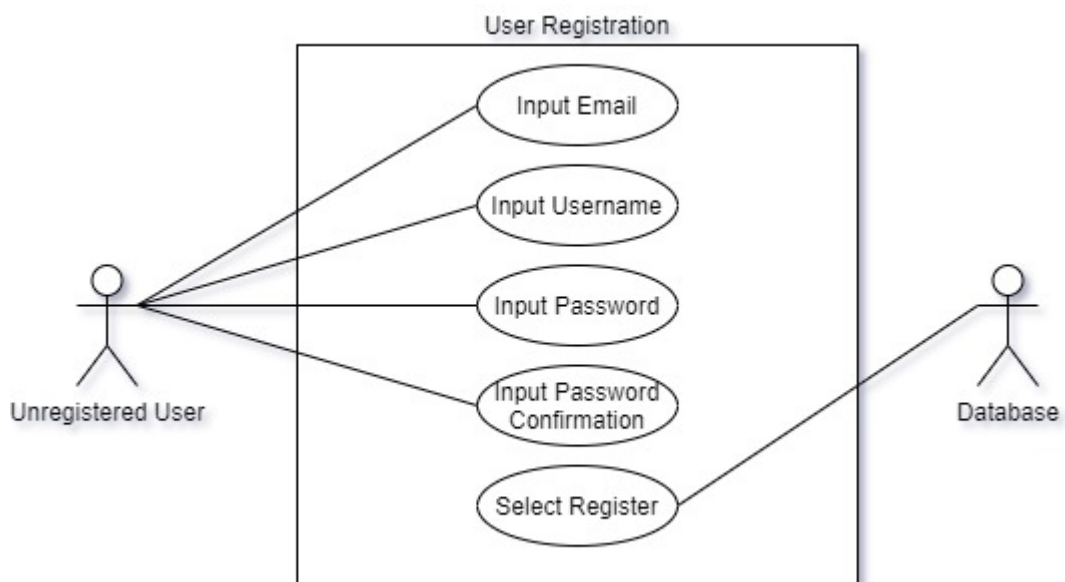
#### Scope

The scope of this use case is to register the details of an unregistered user.

#### Description

This use case describes the process of user registration.

#### Use Case Diagram



#### Flow Description

##### Precondition

User has navigated to the landing page of the application.

##### Activation

This use case starts when an unregistered user selects 'Register'.

##### Main flow

1. The system identifies that registration has been selected.
2. The user inputs username, a valid email address, password, and confirmation password.
3. The system verifies details entered in step two (See A1, A2, E1)

### **Alternate flow**

A1 : Email does not meet validation requirements.

1. The system returns appropriate message to user.
2. The use case continues at step two of main flow.

A2 : Password does not meet minimum requirements.

1. The system returns appropriate message to user.
2. The use case continues at step two of main flow.

### **Exceptional flow**

E1 : User attempts to Register with no Details.

1. System displays appropriate message to user.
2. No registration occurs.
3. The use case continues at step one of main flow.

### **Termination**

The use case is terminated when the user is registered with the system.

### **Post condition**

The user is brought to the homepage and system is in an idle state until further input from user.

## Requirement 2: User Login

### 2.1.1.5. Description & Priority

This use case describes the registered user login process. This process is vital as no further system interactions can take place for a logged-out user. Priority one.

### 2.1.1.6. Use Case

User Login.

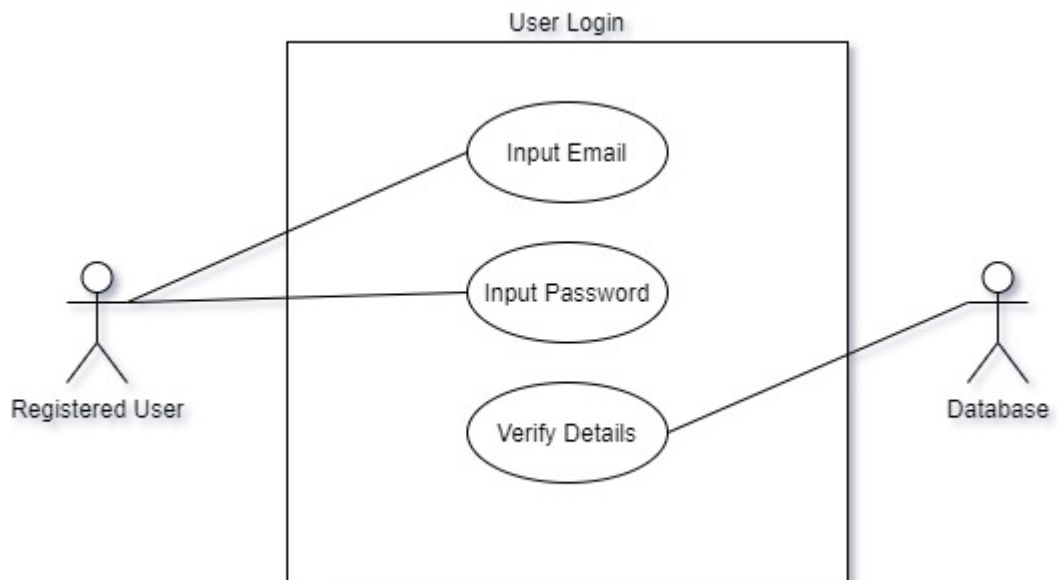
#### Scope

The scope of this use case is for a registered user to log in to the site.

#### Description

This use case describes the process of a registered user successfully logging in to the site.

#### Use Case Diagram



#### Flow Description

##### Precondition

User is registered.

##### Activation

This use case starts when a registered, logged out user selects 'Login'.

##### Main flow

1. User enters email into the email address field.
2. User enters password into the email address field.
3. User selects 'log in' (See E1).
4. System verifies credentials (See A1, A2).

### **Alternate flow**

A1 : Email is incorrect.

1. User remains logged out.
2. System displays appropriate message to user.
3. Use case continues from step one of the main flow.

A2 : Password is incorrect.

1. User remains logged out.
2. System displays appropriate message to user.
3. Use case continues from step two of main flow.

### **Exceptional flow**

E1 : User attempts to login with blank email or password.

1. System recognises data is missing.
2. User remains logged out.
3. Appropriate message is displayed to the user.
4. Use case continues at step one of main flow.

### **Termination**

The use case is terminated when the user is logged in successfully and user session has started.

### **Post condition**

The user is brought to the homepage and system is in an idle state until further input from user.

### 2.1.1.7. Requirement 3: Report Pet

### 2.1.1.8. Description & Priority

This use case describes the process of reporting lost, found or stolen pet. This process is important as it is the main purpose of the application. Priority one.

### 2.1.1.9. Use Case

Report Pet.

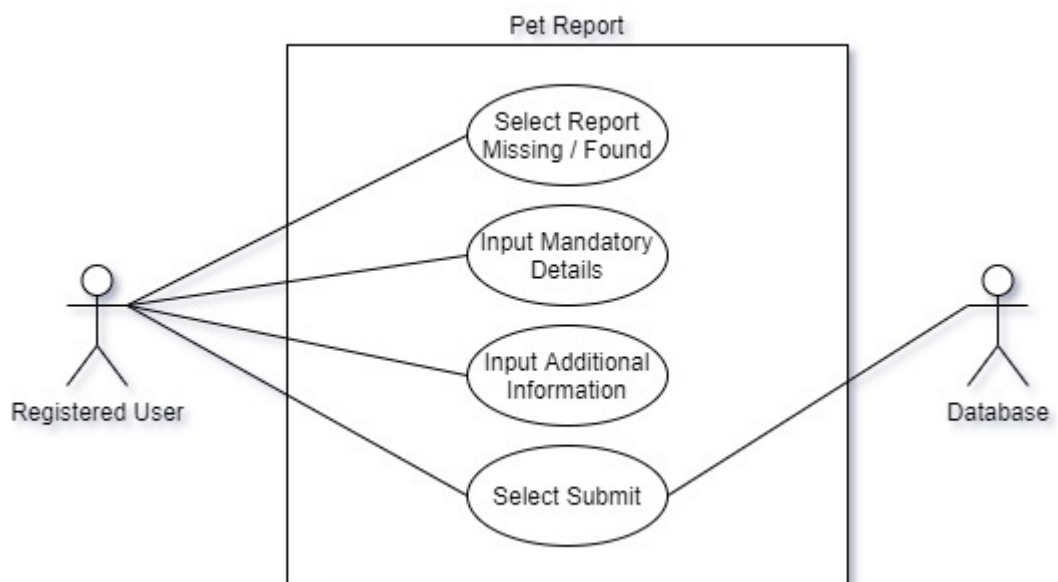
#### Scope

The scope of this use case is for a registered user to report a lost or found pet.

#### Description

This use case describes the process of reporting lost or found pet.

#### Use Case Diagram



#### Flow Description

#### Precondition

Registered user is logged in.

#### Activation

This use case starts when a logged-in user selects 'Report Missing / Found' from the home page.

### **Main flow**

1. User selects 'Report Missing / Found'.
2. User enters mandatory description details: (See E1).
  - a. Report Type
  - b. Date
  - c. Location
  - d. Animal
  - e. Breed
  - f. Colour
  - g. Image
3. User inputs Additional Information (See A1).
4. User selects Submit.
5. System imports details to DB.

### **Alternate flow**

A1 : No Additional Information entered.

1. User skips Additional Info field.
2. Use case continues at step five of main flow.

### **Exceptional flow**

E1 : User attempts to submit without some or all mandatory description details.

1. System recognises data is missing.
2. Appropriate message is displayed to user.
3. Use case continues at step two of main flow.

### **Termination**

The use case is terminated when the details are imported into the DB.

### **Post condition**

Message displayed to user confirming successful submission of report.



#### 2.1.1.10. Requirement 4: Delete Report

#### 2.1.1.11. Description & Priority

This use case describes the process of deleting a pet report. This process is of medium importance. A user should be allowed to delete any pet report posted by said user at any time. Priority two.

#### 2.1.1.12. Use Case

Delete Pet Report.

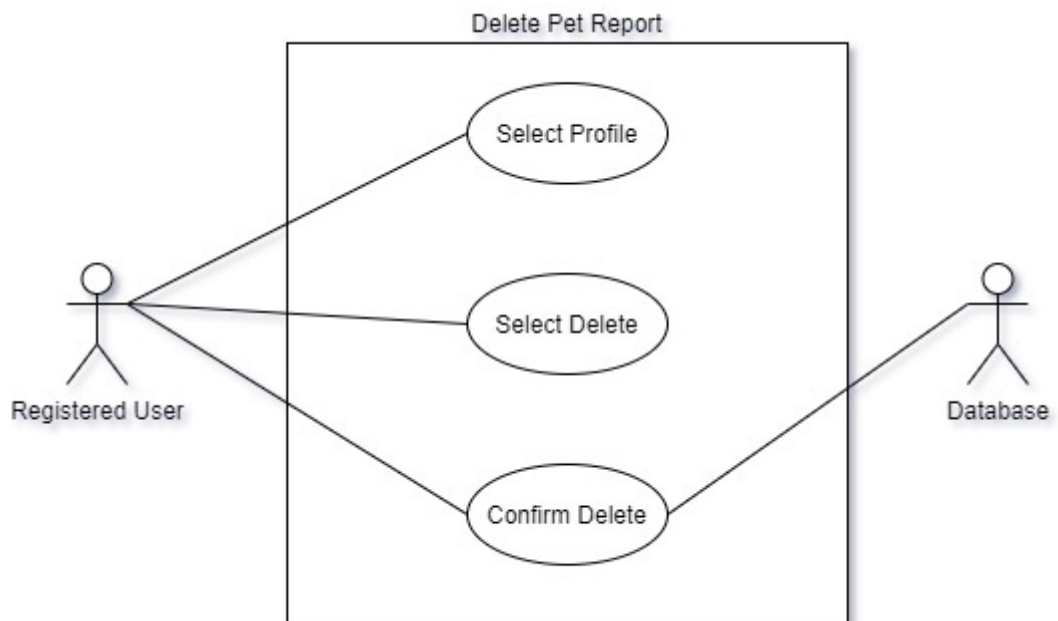
#### Scope

The scope of this use case is for a registered user to delete a pet report.

#### Description

This use case describes the process of deleting a pet report.

#### Use Case Diagram



#### Flow Description

#### Precondition

Registered has posted a Pet Report.

#### Activation

This use case starts when a logged-in user selects a Pet Report posted by that user.

**Main flow**

1. User selects the Pet Profile / Pet Report.
2. User selects 'Delete'.
3. User selects 'Yes' to delete confirmation (See A1).

**Alternate flow**

A1 : User rejects delete confirmation.

1. User selects 'No' to delete confirmation.
2. Use case resumes at step one of main flow.

**Exceptional flow**

No Exceptional Flow.

**Termination**

The use case is terminated when the details are deleted from the database.

**Post condition**

Message displayed to user confirming successful deletion of report.

### 2.1.1.13. Requirement 5: Edit Report

#### 2.1.1.14. Description & Priority

This use case describes the process of editing a pet report. This process is of medium importance. A user should be allowed to edit any details of a pet report posted by said user at any time. Priority two.

#### 2.1.1.15. Use Case

Edit Pet Report.

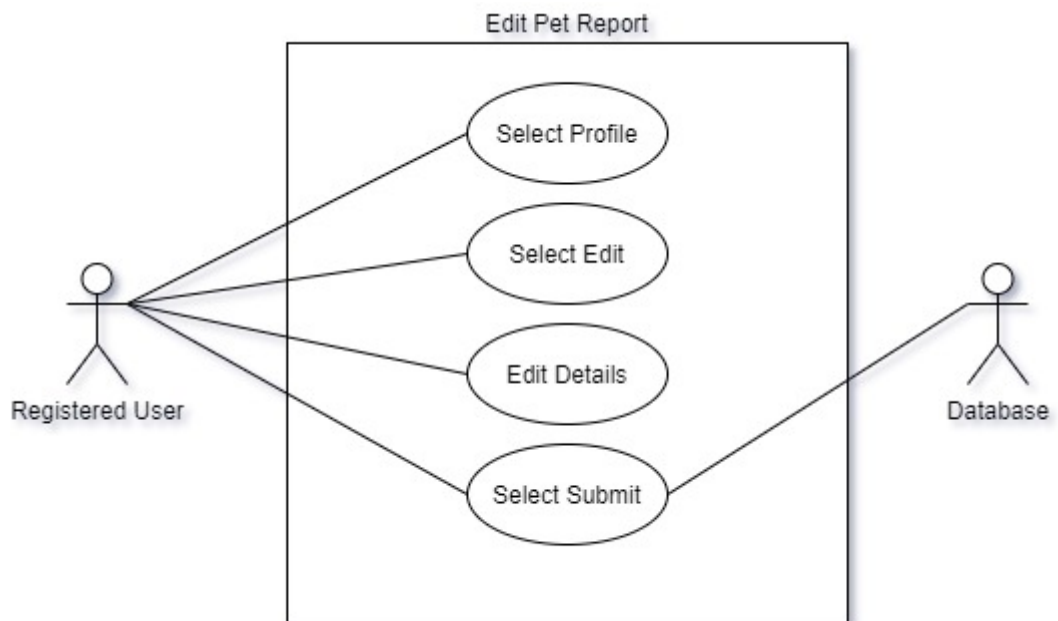
#### Scope

The scope of this use case is for a registered user to edit a pet report.

#### Description

This use case describes the process of editing a pet report.

#### Use Case Diagram



#### Flow Description

#### Precondition

Registered has posted a Pet Report.

#### Activation

This use case starts when a logged-in user selects a Pet Report posted by that user.

### **Main flow**

1. User selects the Pet Profile / Pet Report.
2. User selects 'Edit'.
3. User edits some or all details (See A1), (See E1).
4. User selects 'Submit'.

### **Alternate flow**

A1 : User makes no changes.

1. User does not edit any field.
2. Use case resumes at step four of main flow.

### **Exceptional flow**

E1 : User attempts to submit blank data for a mandatory field.

1. User edits a field and leaves field blank.
2. User selects 'Submit'.
3. Appropriate message is displayed to the user and no changes are committed to the database.
4. Use case resumes at step three of main flow.

### **Termination**

The use case is terminated when the details are edited, and changes committed to the database.

### **Post condition**

Message displayed to user confirming successful edit of report details.

### 2.1.1.16. Requirement 6: View Gallery

### 2.1.1.17. Description & Priority

This use case describes the process of viewing a gallery of all reported pets. This process is important as it is the main purpose of the application. Priority 1.

### 2.1.1.18. Use Case

#### View Gallery

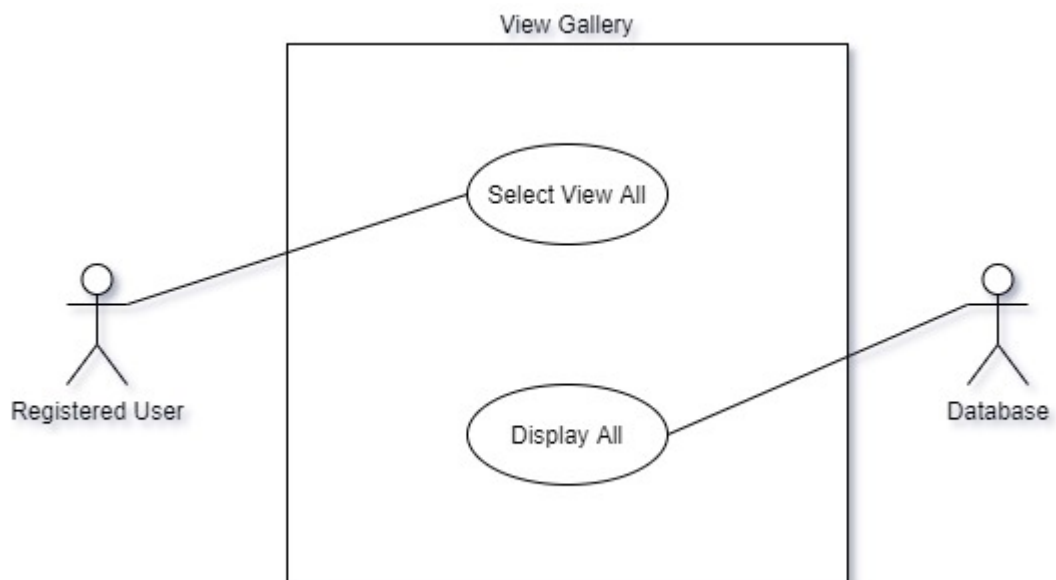
#### Scope

The scope of this use case is for a registered user to view all pets reported missing, found, or stolen.

#### Description

This use case describes the process of viewing all pets reported missing, found, or stolen in the gallery.

#### Use Case Diagram



#### Flow Description

#### Precondition

User is registered and logged in, or user is registered and logged out, or user is unregistered.

#### Activation

This use case starts when a user selects 'View All'.

#### Main flow

1. User selects 'View All'.
2. User is brought to the Gallery.

**Alternate flow**

No Alternate Flow.

**Exceptional flow**

No Exceptional Flow.

**Termination**

The use case is terminated when the gallery of missing pets is displayed to the user.

**Post condition**

System awaits further input in an idle state.

### 2.1.1.19. Requirement 7: View Individual Profile

#### 2.1.1.20. Description & Priority

This use case describes the process of viewing an individual missing pet profile. This process is important as it is the main purpose of the application. Priority one.

#### 2.1.1.21. Use Case

##### View Profile

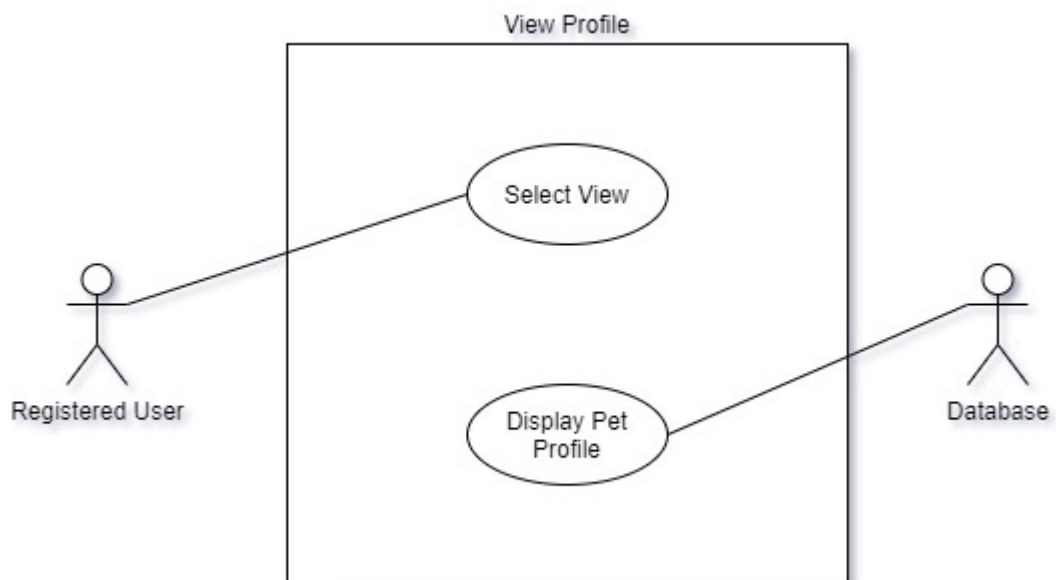
##### Scope

The scope of this use case is for a registered user to view an individual profile of a pet reported missing, found, or stolen.

##### Description

This use case describes the process of viewing an individual missing, found or stolen pet profile.

##### Use Case Diagram



##### Flow Description

##### Precondition

User has navigated to the Gallery.

##### Activation

This use case starts when a user selects 'View'.

##### Main flow

1. User selects 'View' on a specific profile / report from the Gallery.
2. User is brought to the reported pet profile.

**Alternate flow**

No Alternate Flow.

**Exceptional flow**

No Exceptional Flow.

**Termination**

The use case is terminated when the reported pet profile is displayed to the user.

**Post condition**

System awaits further input in an idle state.



#### 2.1.1.22. Requirement 8: Comment

#### 2.1.1.23. Description & Priority

This use case describes the process of commenting on an existing pet report. This process is of medium importance, it ensures users are updated as quickly as possible. Priority two.

#### 2.1.1.24. Use Case

Comment

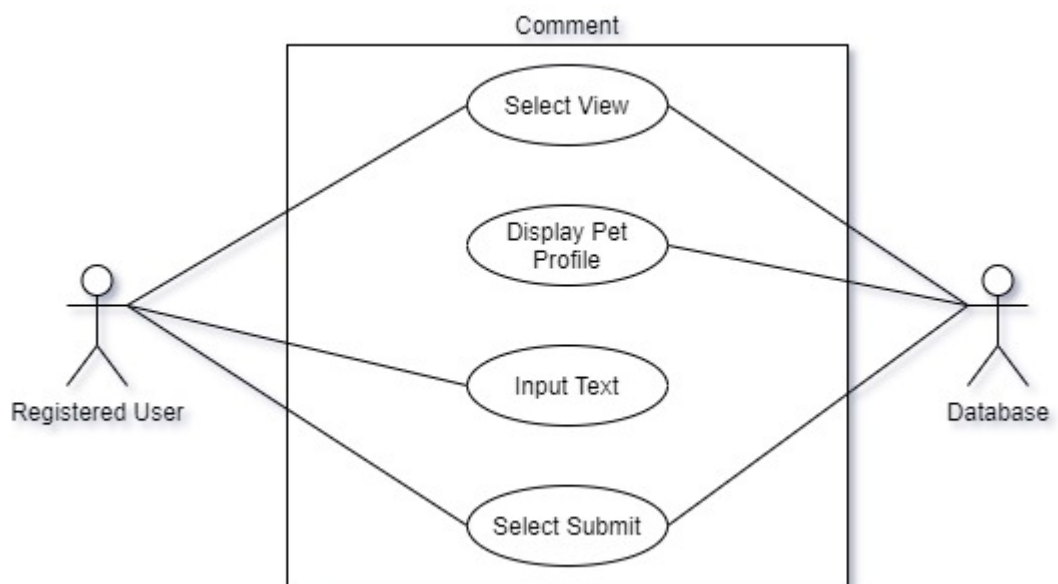
#### Scope

The scope of this use case is for a registered user to leave a comment on an existing pet report / profile.

#### Description

This use case describes the process of leaving a comment on an existing pet report / profile.

#### Use Case Diagram



#### Flow Description

#### Precondition

Pet reports / profiles exist in the database.

#### Activation

This use case starts when a user selects 'View' on a pet report / profile.

**Main flow**

1. User selects 'View' on a specific profile / report from the Gallery.
2. User is brought to the reported pet profile.
3. User enters text in the 'comment' section of the profile (See E1).
4. User selects 'Submit'.

**Alternate flow**

No Alternate Flow.

**Exceptional flow**

E1: User attempts to submit a blank comment.

1. User does not enter text into the comment section.
2. User selects 'Submit'.
3. Appropriate message is displayed to the user and the comment is not posted.
4. Use case resumes at step two of the main flow.

**Termination**

The use case is terminated when the commented is posted to the database.

**Post condition**

User's comment is displayed in comments section.

### 2.1.1.25. Requirement 9: Delete Comment

### 2.1.1.26. Description & Priority

This use case describes the process of deleting an existing comment on an existing pet report. This process is of medium importance, knowing users have the option to delete a comment will encourage users to post comments. Priority two.

### 2.1.1.27. Use Case

Delete Comment

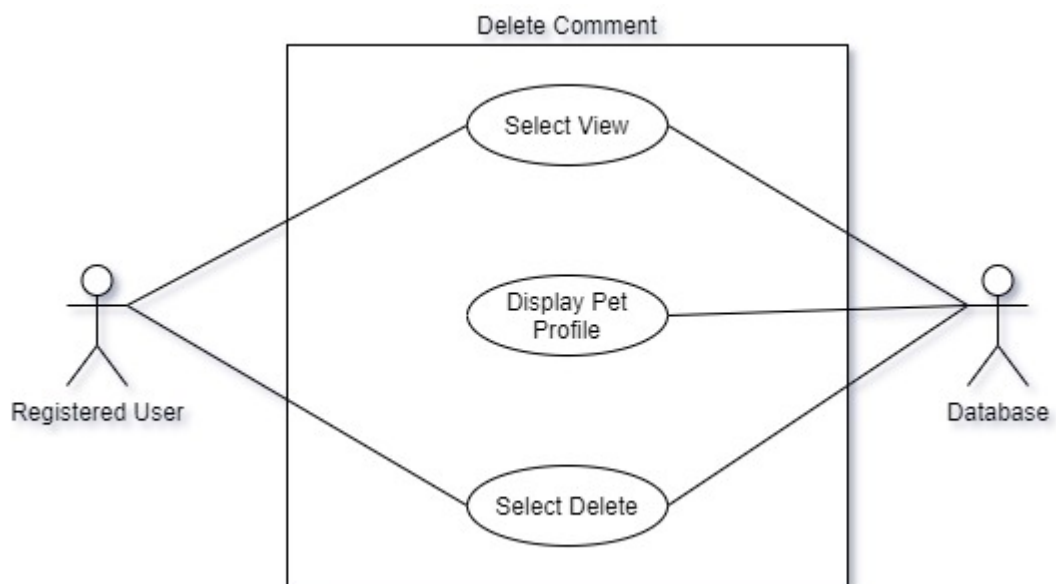
#### Scope

The scope of this use case is for a registered user to delete an existing comment on an existing pet report / profile.

#### Description

This use case describes the process of deleting an existing comment on an existing pet report / profile.

#### Use Case Diagram



#### Flow Description

#### Precondition

A comment associated to the user exists for a pet report / profile.

#### Activation

This use case starts when a user selects 'View' on a pet report / profile.

**Main flow**

1. User selects 'View' on a specific profile / report from the Gallery.
2. User is brought to the reported pet profile.
3. User scrolls to comments section and selects 'Delete' beside said user's comment.

**Alternate flow**

No Alternate Flow.

**Exceptional flow**

No Exceptional Flow.

**Termination**

The use case is terminated when the user's comment is deleted from the database.

**Post condition**

User's comment is no longer displayed in the comments section.

### 2.1.1.28. Requirement 10: Send Direct Message

#### 2.1.1.29. Description & Priority

This use case describes the process of sending a direct message to another registered user. This is of high importance as it is the primary method of users exchanging details privately to return a found dog to an owner, for example. Priority one.

#### 2.1.1.30. Use Case

Send Message

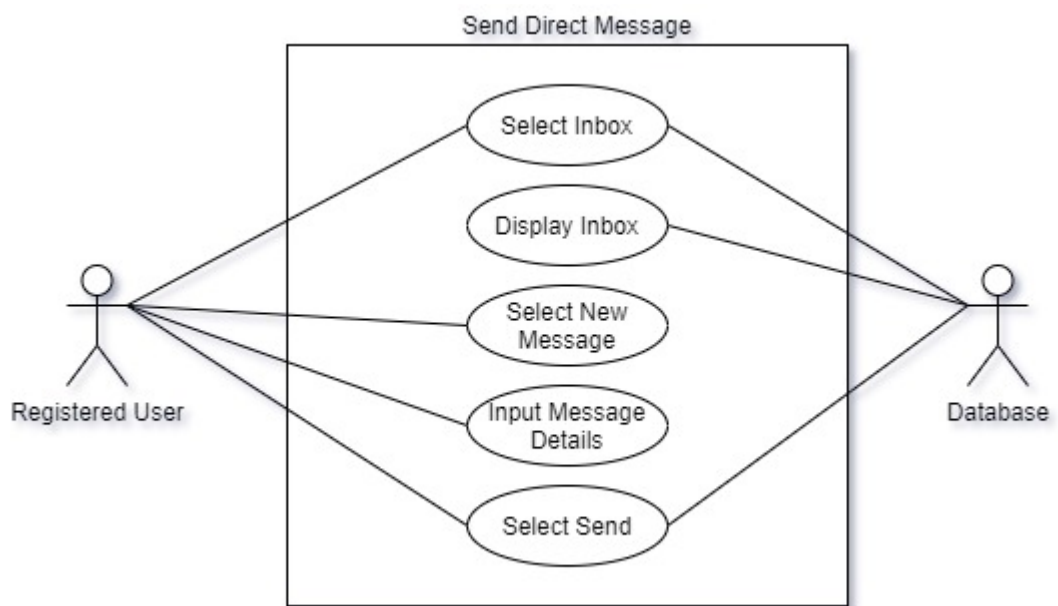
#### Scope

The scope of this use case is for a registered user to send a message to another registered user.

#### Description

This use case describes the process of sending a direct message to another registered user.

#### Use Case Diagram



#### Flow Description

#### Precondition

User is registered.

#### Activation

This use case starts when a user selects 'Inbox' from the navigation bar.

**Main flow**

1. User selects 'Inbox'.
2. User is brought to the inbox.
3. Existing messages to the current user are displayed.
4. User selects 'New Message'.
5. User enters a subject, message body, and selects a recipient (See E1).
6. User selects 'Send'.

**Alternate flow**

No Alternate Flow.

**Exceptional flow**

E1: User leaves mandatory information blank.

1. User leaves subject, message body, or recipient blank.
2. User selects 'Send'.
3. Appropriate message is displayed to the user.
4. Use case resumes at step five of the main flow.

**Termination**

The use case is terminated when the user's message is posted to the database.

**Post condition**

The message can be viewed by the recipient from the recipient's inbox.

### 2.1.1.31. Requirement 11: View Sent Messages

### 2.1.1.32. Description & Priority

This use case describes the process of viewing messages previously sent by the current user. Priority two.

### 2.1.1.33. Use Case

#### View Sent Messages

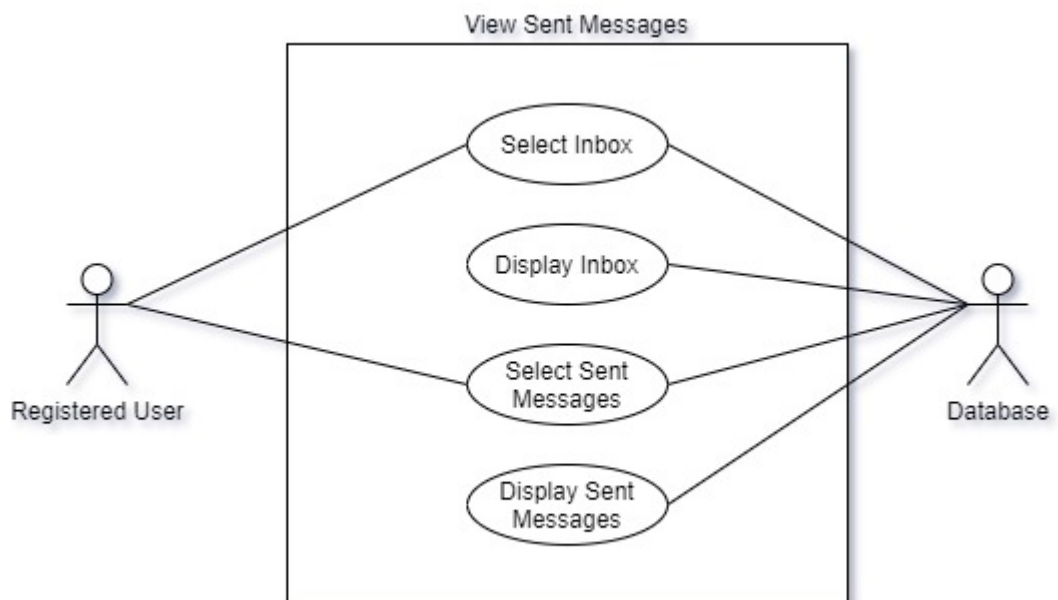
#### Scope

The scope of this use case is for a registered user to view sent messages.

#### Description

This use case describes the process of viewing all messages sent from the current user to other users.

#### Use Case Diagram



#### Flow Description

#### Precondition

User has sent message(s).

#### Activation

This use case starts when a user selects 'Inbox' from the navigation bar.

**Main flow**

1. User selects 'Inbox'.
2. User is brought to the inbox.
3. Existing messages to the current user are displayed.
4. User selects 'View Sent'.
5. User is brought to the sent box.
6. All Sent items are visible and include the subject, date and time, and user the message was sent to.

**Alternate flow**

No Alternate Flow.

**Exceptional flow**

No Exceptional Flow.

**Termination**

The use case is terminated when the user's sent messages are displayed.

**Post condition**

System directs user to sentbox and awaits in an idle state for further input from user.



#### 2.1.1.34. Requirement 12: Logout

#### 2.1.1.35. Description & Priority

This use case describes the process of accessing the inbox. This process is useful as vital as a logged in user must have the ability to end the session. Priority 1.

#### 2.1.1.36. Use Case

Logout

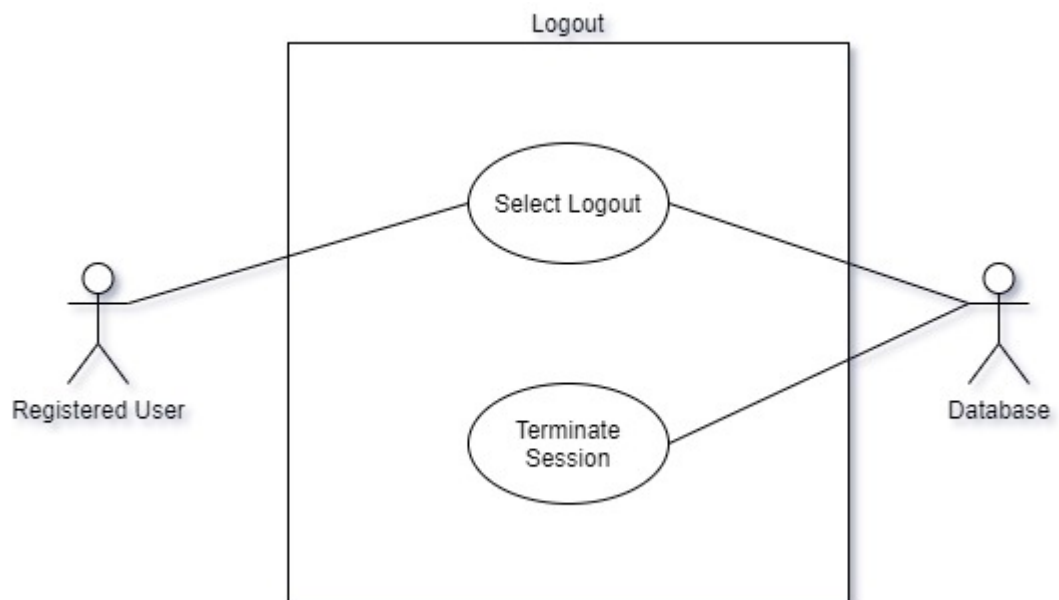
#### Scope

The scope of this use case is for a logged in user to log out.

#### Description

This use case describes the process of logging out.

#### Use Case Diagram



#### Flow Description

#### Precondition

Registered user is logged in.

#### Activation

This use case starts when a user selects 'Logout'.

**Main flow**

1. User selects Logout.
2. Session is terminated.
3. Appropriate message is displayed to the user.

**Alternate flow**

No Alternative Flow.

**Exceptional flow**

No Exceptional Flow.

**Termination**

The use case is terminated when the session is terminated.

**Post condition**

User is brought to login screen and system remains idle awaiting further input.

### 2.1.2. Data Requirements

PetRescue is built around an object-relational database, consisting of five models as follows. These models and their relationships are as follows:

- Users

- o has\_many :pet\_reports
- o has\_many :comments
- o has\_many :messages

- Pet\_Reports

- o belongs\_to :user
- o has\_many :comments

- Comments

- o belongs\_to :user
- o belongs\_to :pet\_report

- Messages

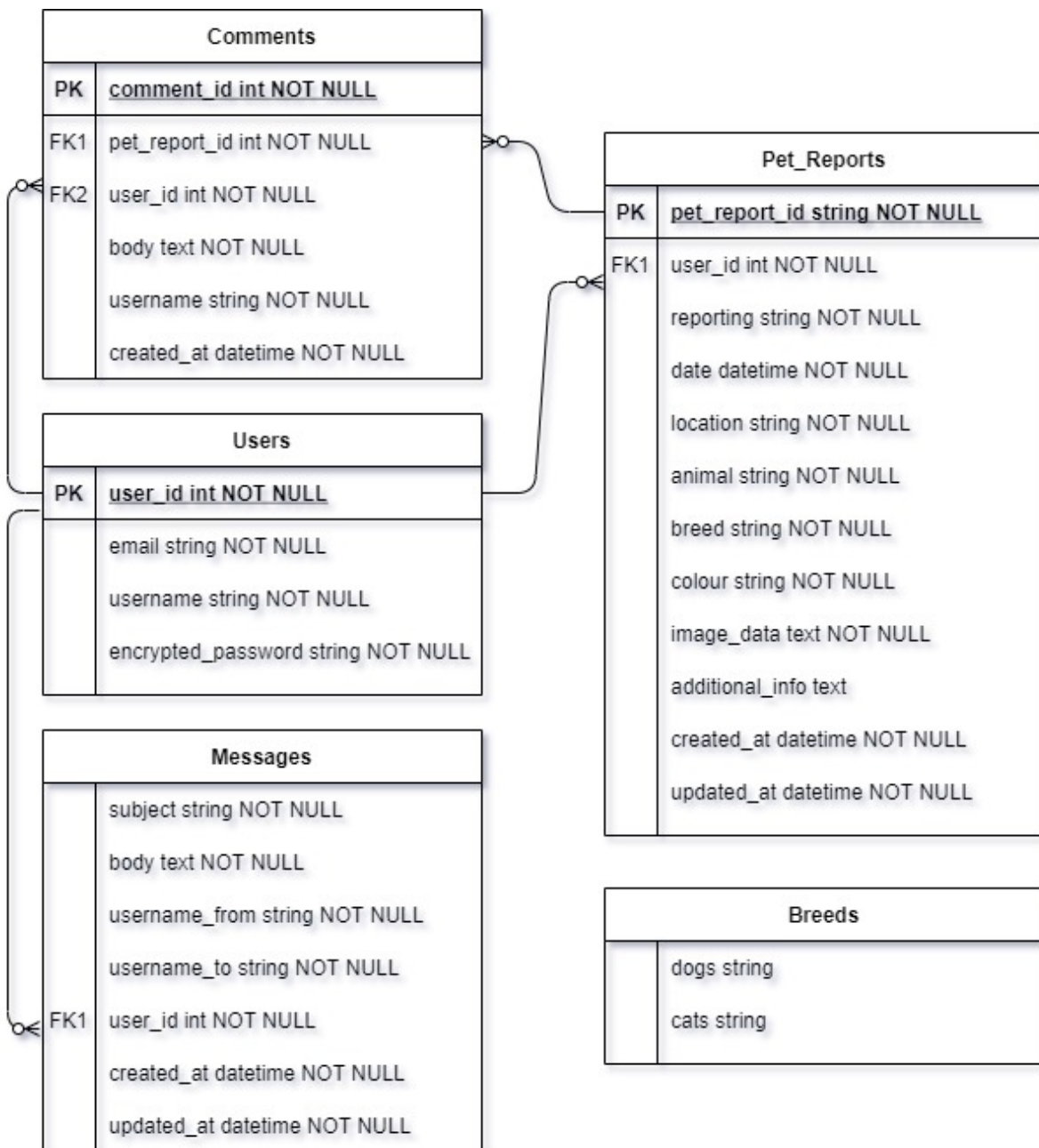
- o belongs\_to :user

- Breeds (This is a standalone model, used to populate the breed dropdown for the pet report form)

One important data requirement for PetRescue is consistency of data. The application was intended to improve on the offering of its competitor by reducing the amount of input required by a user to report a missing pet. This reduction of user input means that the data that is captured is absolutely vital. If one mandatory field is missing it results in the report being too generic, offering little or no value to users. To handle this validation is performed at the application level. Each mandatory field is specified in the model as being a requirement, therefore missing data in that field will result in failure to commit to the database. Similarly, to ensure integrity (and security) of the data, validation is performed on certain fields, where failure to do so could result in bad data reaching the database. One such example is the file attachment included in the pet report form. Without proper validation a user could simply upload text, resulting in a sub-standard pet report being published, or submit excessively large files clogging up the database, or even worse submit something malicious which is then consumed by the database and the application. For these reasons validation is handled in the image\_uploader.rb file to ensure only secure, valid data is reaching the database. This validation is handled as follows:

```
Attacher.validate do
  #must be of type jpeg, png or webp
  validate_mime_type %w[image/jpeg image/png image/webp]
  #must be no more than 5mb
  validate_max_size 5*1024*1024
end
```

The following entity relationship diagram depicts the various models and data types that exist in the database, and the relationship (where it exists) between them.



### 2.1.3. User Requirements

All user requirements for the PetRescue web application are detailed in the following format:

**'As a ... I'** : This represents the stakeholder the requirement is aimed at.

**'Must / Should / Want'** : This represents the priority : Must = High, Should = Medium, Want = Low.

**'So that...'** : This represents the overall aim/outcome of the requirement.

This format breaks each requirement down in to three easily readable and understandable segments to ensure maximum efficacy of the requirements.

All user requirements are details below:

As a...I	Must/Should/Want...	So that...
User	Must have the ability to register	I can use my account credentials to log in
User	Must have the ability to log in	I can access the application
User	Must be able to recover my password	I can access the site again in the event that I have forgotten my password
User	Must have the ability to log out	I can end my session
User	Must have the ability to report a lost, found, or stolen pet	I can post the report and get information back from other users
User	Should have the ability to upload an image when reporting	My report contains an image which improves the chances of someone recognising the pet from the post
User	Want to only enter necessary and relevant details when reporting	My report is completed with ease in a short amount of time
User	Must be able to delete a pet report I have posted	I can remove the post once the animal has been found or returned
User	Must be able to edit a pet report I have posted	I can change the details if accidentally posted incorrectly
User	Should be able to comment on an existing pet report	I can update the reporter of the pet report on any information (such as a recent sighting) I have quickly and easily
User	Must be able to delete my comment on an existing pet report	My comment no longer is displayed in the comments section

User	Want to receive an email when someone comments on my pet report	I am notified instantly on the comment and can react quickly and accordingly
User	Want an attractive front end	The application is enjoyable to use
User	Want a user-friendly UI that is easily navigable	I can easily accomplish what I set out to do when using the application
User	Must have the ability to send messages to other users	I can make a private enquiry to a user about a pet report
User	Must have an inbox	I can view messages sent to me by other users
User	Should have the ability to view messages sent by me	I am able to keep track of who/what I sent messages to/about
User	Should have the ability to reply to a message I have received	I can communicate with other users to resolve enquiries

#### 2.1.4. Environmental Requirements

##### 2.1.4.1. Client

As PetRescue is a web-based application the only environmental requirement at the client side is an active internet connection. PetRescue has been tested across multiple browsers and is built with responsiveness in mind, so desktop, laptop and mobile devices should all be compatible.

##### 2.1.4.2. Server

At the time of writing PetRescue has not yet been deployed. PetRescue is a Ruby application built on the Rails framework and utilises a PostgreSQL database. That being the case, a PaaS provider who support this object-relational database should be chosen to host the site, such as Heroku. PetRescue relies on a number of gems for different aspects of the site, these gems are dependencies which should be present in the gemfile and the gems installed once deployed. PetRescue captures most of its data from user input, however in order to populate the breeds lists which are used by the pet report form to select a breed, a seed must be run in order to seed the breed data from the seed.rb file into the database.

##### 2.1.4.3. Development

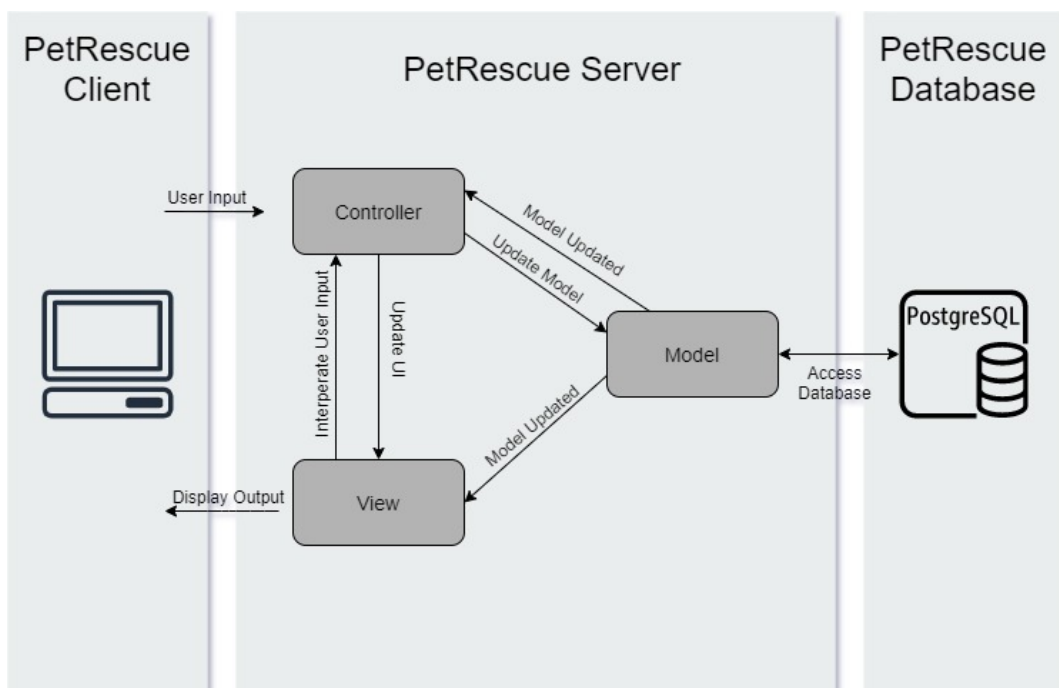
During development of PetRescue there were a number of environmental requirements. As development was done solely on a Windows laptop it was necessary to run all commands through an Ubuntu terminal for Windows and the application would run on localhost on port 3000. The mailer responsible for the sending of automated emails was configured specific to the development environment using smtp as the delivery method on port 587 and connecting to a Gmail account that was specifically set up for the project. The Gmail account initially would not send the mails due to the source sender being less secure, so 'Less Secure App Access' had to be enabled in the account in order for the mailer to work.

### 2.1.5. Usability Requirements

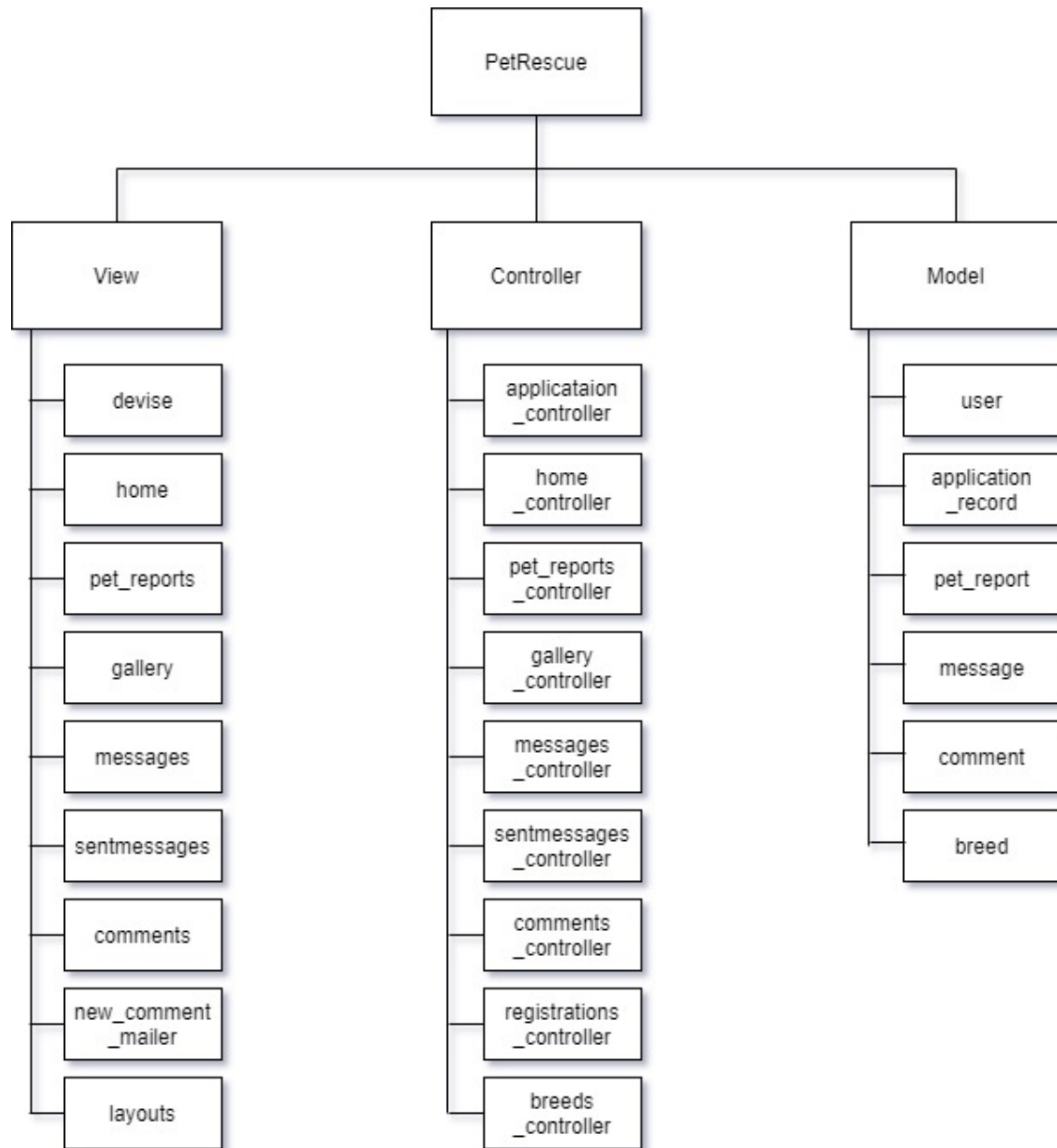
The main usability requirement for PetRescue was that the application should be designed so that an inexperienced user with no prior knowledge of the application can easily use all features of the site end to end. Since the development of PetRescue, this requirement has been verified by multiple test users of varying demographics who were given tasks to complete on the site. In all cases the users completed their assigned task on the first attempt and within a reasonable timeframe. This requirement was achieved through pre-planning and designing a clean and minimalist user interface with logical process flows for the key functionalities of the site. These process flows will be discussed further on in this document, under the Design and Architecture section.

### 2.2. Design & Architecture

PetRescue is a Ruby on Rails web application, and as a result it has been developed using the Model View Controller (MVC) architectural design pattern. “The Model-View-Controller is an architectural pattern that separates an application into three main logical components: the model, the view and the controller; each built to handle specific development aspects of an application” (MVC Framework – Introduction – Tutorialspoint, 2021). The front-end is developed in the Views, meaning it is responsible for all aspects of the application which are presented to or interacted with by the user. These user interactions are taken by the View and passed to the Controller, which then asserts what to do with this input and informs the model, and/or view. The Model is where data and the logic associated with that data are maintained. This architectural model is popular with many frameworks as it provides separation of concerns which makes code easier to maintain by reducing complexity. The below diagram has been created to visually represent PetRescue’s architectural implementation of the MVC design pattern:

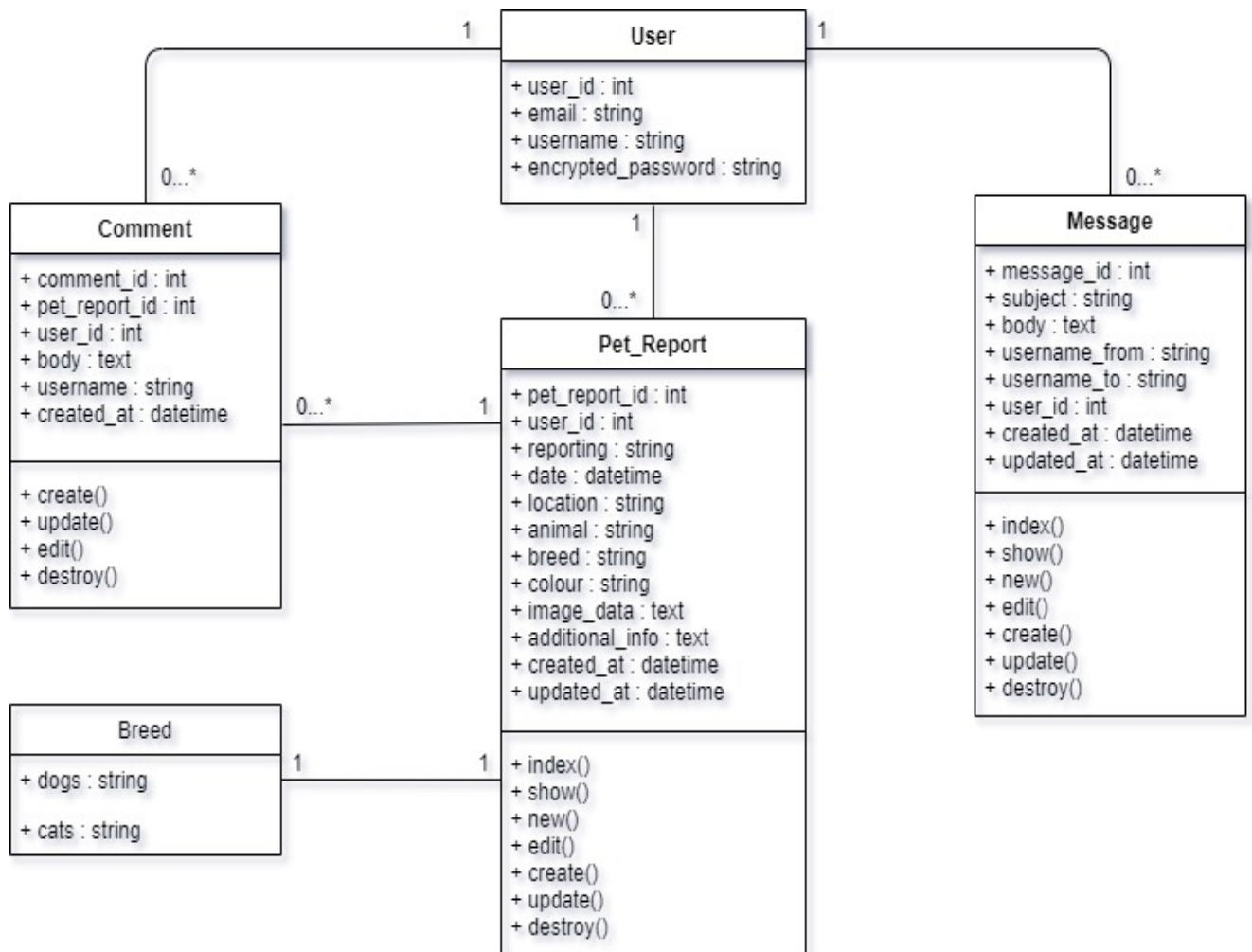


To aid the above visual representation, a structural breakdown of PetRescue can be seen below. This details the specific models, views and controllers that make up the PetRescue web application.





The following class diagram shows PetRescue's structure in terms of its classes, and the attributes, methods, and relations of those classes:



### 2.3. Implementation

There are several pieces of functionality throughout PetRescue which will be discussed throughout this document. However, there are three key features which the application is heavily dependent upon, as combined they make up the sole purpose of the application.

#### Pet Report Process

First and foremost is the process of reporting a pet missing, found, or stolen. To do this a user must complete a form which is then submitted to the database. Although unregistered users can view currently missing, found, or stolen pets, in order to access the form to post a pet report themselves, a user must be registered. To achieve this a check is carried out in the controller for pet reports, to ensure that the current user has been authenticated:

```
before_action :authenticate_user!, except: :index
```

As can be seen from this snippet, the controller checks that the user is authenticated before any method is accessed, with the exception of the index method, meaning unauthenticated users can access index only, and can view currently missing, found, or lost pets. The form itself is a partial, with the code written in the view and shared by both 'new.html.erb' and 'edit.html.erb'. As part of this form, the user is required to submit an image of the pet they are reporting. This was the most difficult part of the pet reporting process to implement. In order to attach the image as a file attachment, the shrine plugin was installed, and the shrine gem added to the gemfile. Shrine.rb was added to the initializers directory and this specifies what plugins are required, such as active record, and where to store uploaded files, for both permanent and cached storage. An image uploader class was created to implement validation around the file type and size that could be uploaded, it was then specified to include this class in the pet\_report model, along with the attribute 'image' and this attribute was included in the list of permissible parameters in the pet report controller. The following code snippets walk through the image uploader as part of the overall form submission.

App/config/initializers/shrine.rb:

```
require "shrine"
require "shrine/storage/file_system"
require "shrine/storage/memory"

if Rails.env.test?
  Shrine.storages = {
    cache: Shrine::Storage::Memory.new, # temporary
    store: Shrine::Storage::Memory.new, # permanent
  }
else
  Shrine.storages = {
    cache: Shrine::Storage::FileSystem.new("public", prefix: "uploads/cache"), # temporary
    store: Shrine::Storage::FileSystem.new("public", prefix: "uploads"), # permanent
  }
}
```

```

end

Shrine.plugin :activerecord #or sequel
Shrine.plugin :cached_attachment_data # for retaining the cached file across f
orm redispays
Shrine.plugin :restore_cached_data # re-
extract metadata when attaching a cached file
Shrine.plugin :validation
Shrine.plugin :validation_helpers
Shrine.plugin :derivatives

```

App/uploaders/image\_uploader.rb

```

require "image_processing/mini_magick"

class ImageUploader < Shrine

  #resize images : code from https://shrinerb.com/docs/processing
  Attacher.derivatives do |image|
    magick = ImageProcessing::MiniMagick.source(image)

    {
      large: magick.resize_to_limit!(800, 800),
      medium: magick.resize_to_limit!(500, 500),
      small: magick.resize_to_limit!(200, 200),
    }
  end

  #add validation for file uploads
  Attacher.validate do
    #must be of type jpeg, png or webp
    validate_mime_type %w[image/jpeg image/png image/webp]
    #must be no more than 5mb
    validate_max_size 5*1024*1024
  end
end
end

```

App/models/pet\_report.rb:

```
include ImageUploader::Attachment(:image)
```

App/controllers/pet\_reports\_controller:

```

def pet_report_params
  params.require(:pet_report).permit(:reporting, :date, :location, :animal
, :breed, :colour, :image, :additionalInfo, :user_id)

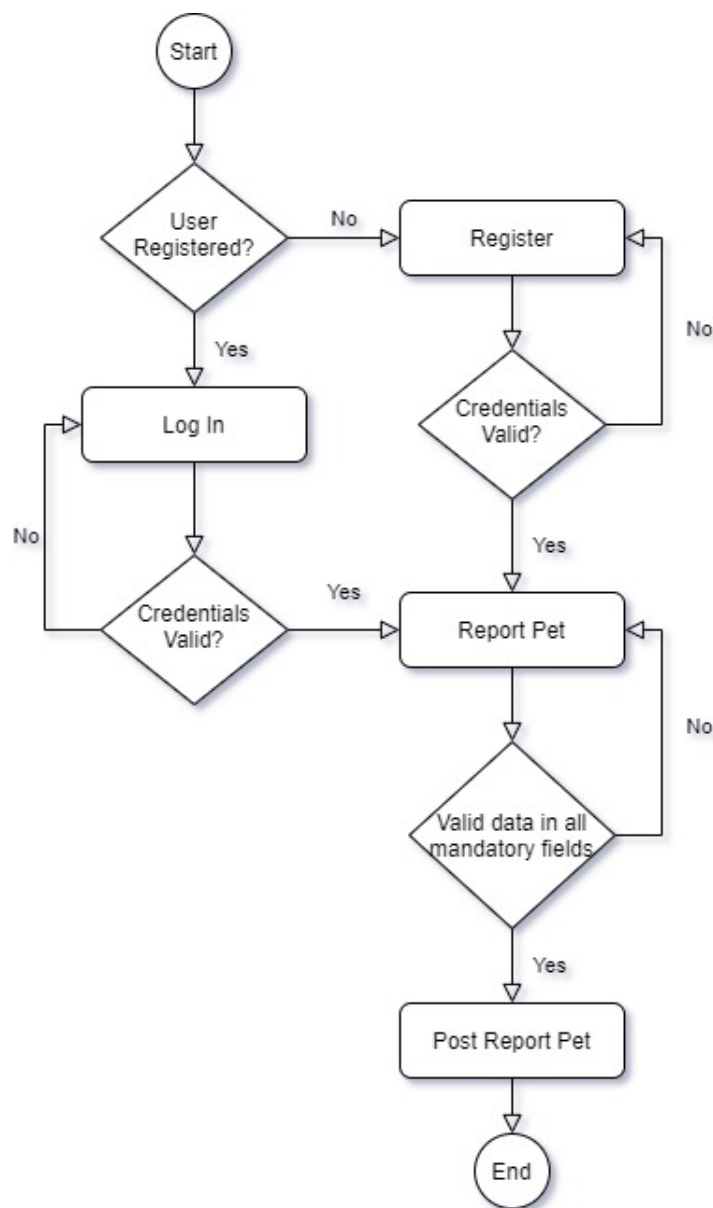
```

```
end
```

App/views/pet\_report/\_form.html.erb:

```
<%= image_tag f.object.image_url if f.object.cached_image_data %> <!--  
if image is cached, display it so user is aware-->  
<%= f.hidden_field :image, value: @pet_report.cached_image_data %> <!--  
cache image data: prevents image being lost if form fails to upload (validatio  
n failure etc.)-->  
<%= f.file_field :image %>
```

The overall process of reporting a pet can be seen in the following process flow:



## Commenting

Another key feature of PetRescue is the ability for users to comment on a pet report, with that comment automatically generating and sending an email to the owner of the report. This is a hugely important piece of functionality, as retrieving a missing pet can be a time-sensitive issue, and the quickest possible way for users to update each other on sightings of a pet is through this method. To tackle this functionality, a comment scaffold was generated, with a relationship established between comment, pet\_report and user:

Comment model:

```
class Comment < ApplicationRecord
  belongs_to :user
  belongs_to :pet_report
end
```

User model:

```
has_many :comments
```

Pet\_report model:

```
has_many :comments
```

The create and destroy methods to allow creation and deletion of a comment were established in the comments controller. Both methods call the pet report model to get the current pet reports ID and assign it to the comment's 'pet\_report\_id' attribute. It also assigns the comments user\_id attribute based on the current\_user method provided by devise, so the reporter of the comment is established. If the comment is created and saved to the database, then the mailer (new\_comment\_mailer.rb) and its method (new\_comment) are called, using the '.deliver\_now' method provided by Application Mailer to send the automated email (designed in views/new\_comment\_mailer/new\_comment.html.erb).

App/mailers/new\_comment\_mailer.rb:

```
class NewCommentMailer < ApplicationMailer

  def new_comment(comment)
    @comment = comment
    @pet_report = @comment.pet_report

    mail to: @pet_report.user.email,
        subject: "New Comment on Your PetRescue Post"
  end
end
```

App/controllers/comments\_controller.rb:

```
def create
  @pet_report = PetReport.find(params[:pet_report_id]) # find pet report
  @comment = @pet_report.comments.create(comment_params)
  @comment.user_id = current_user.id if current_user # assign the user id of
the current user to the comment
  @comment.username = current_user.username

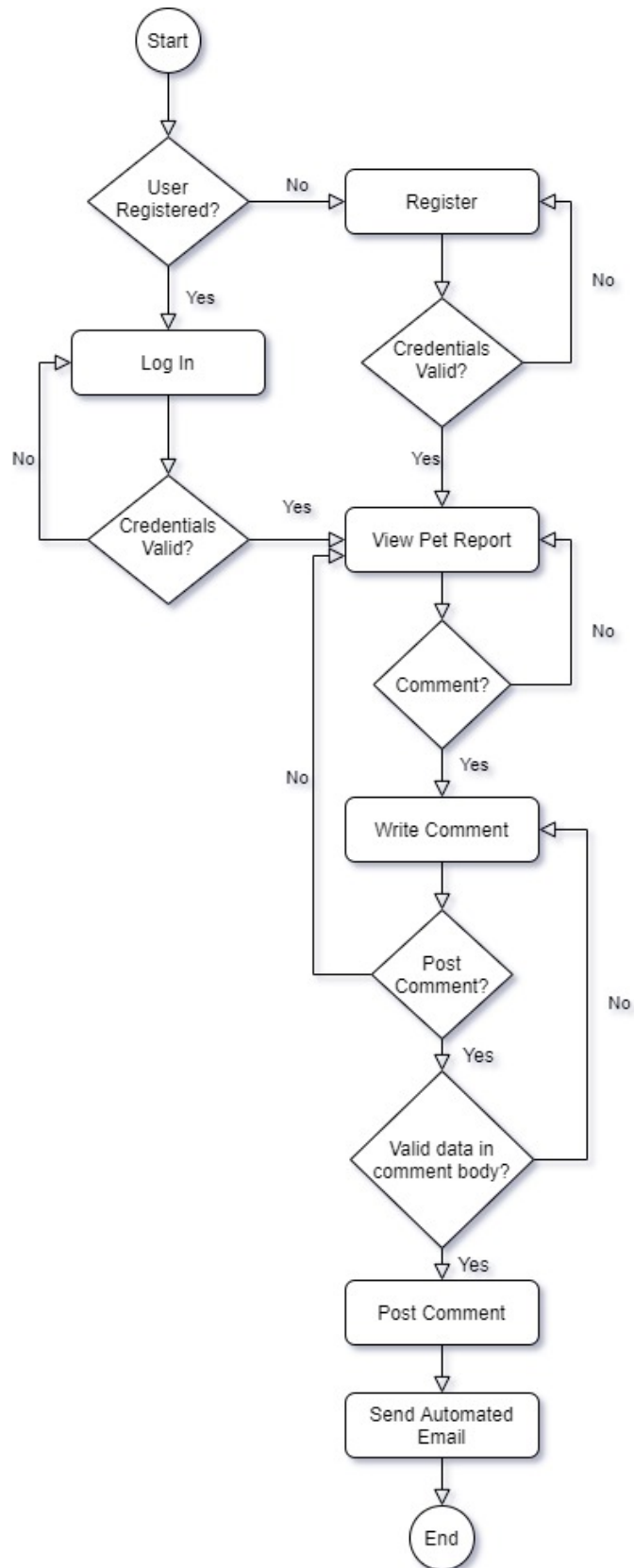
  if @comment.save
    NewCommentMailer.new_comment(@comment).deliver_now # send email notifi
cation to owner of the original post
    flash[:success] = 'Comment Posted'
    redirect_to pet_report_path(@pet_report)
  else
    flash[:danger] = 'Could Not Post Comment'
  end
end
end
```

The connection details for the mailer are set up per environment, in development for example, the connection details are as follows:

```
config.action_mailer.delivery_method = :smtp
config.action_mailer.smtp_settings = {
  address: "smtp.gmail.com",
  port: 587,
  domain: "example.com",
  authentication: "plain",
  enable_starttls_auto: true,
  #username and password in application.yml, which is included in gitignore
so credentials are not in public repo
  user_name: ENV["MAIL_USERNAME"],
  password: ENV["MAIL_PASSWORD"]
}
config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }
```

Note that the username and password details are stored in the application.yml file which is specified to be ignored by github for security, so sensitive details are not pushed up to the public repository where this application lives.

The flow of the comments feature can be seen in the below process flow diagram:



## Messaging Process

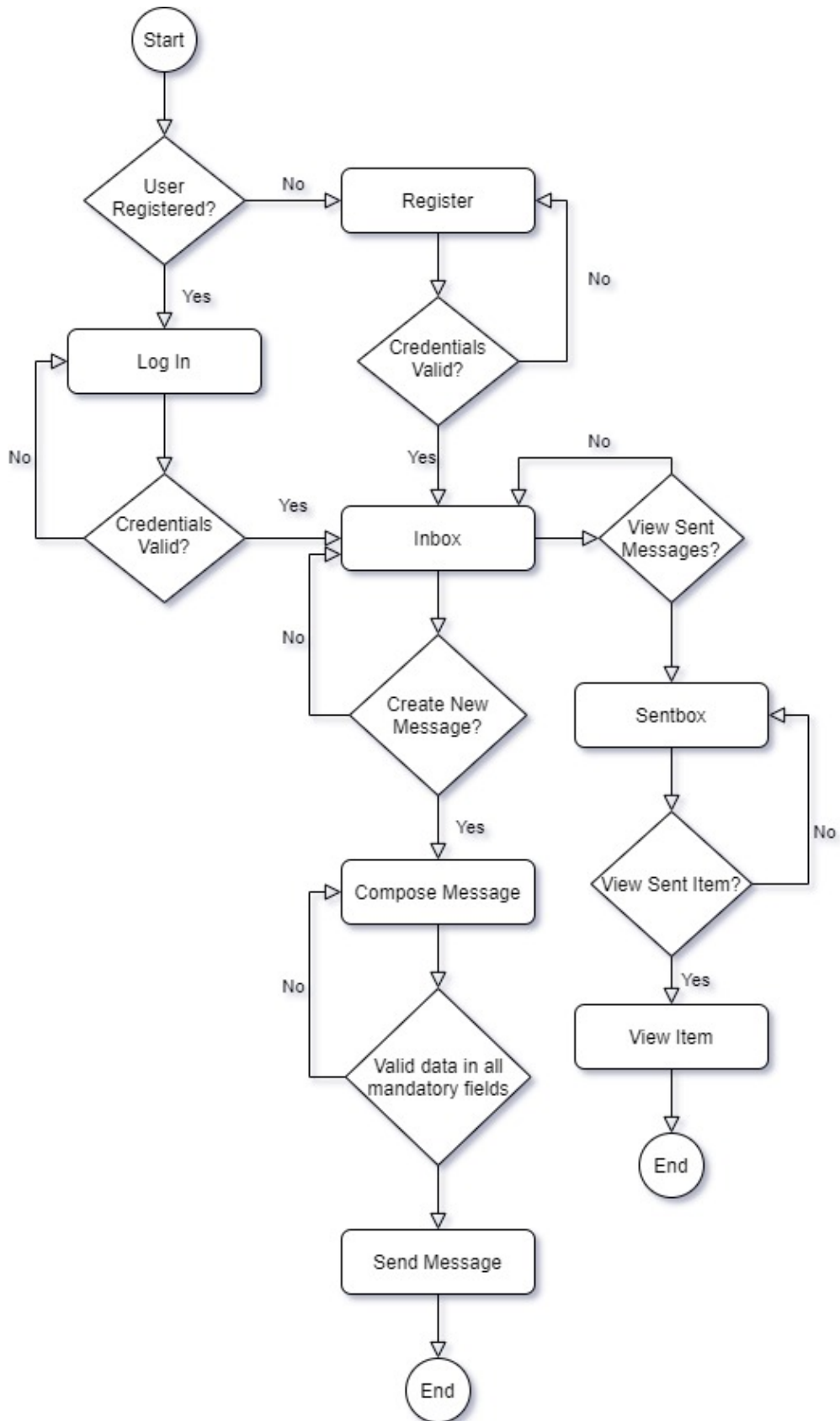
The third key feature of PetRescue is its inbox, allowing users to communicate more sensitive details to each other privately such as phone numbers or addresses, where a public comment would not be suitable. To implement direct messaging, a number of avenues were explored, however as this was one of the later pieces of functionality to enter development and time was becoming more and more limited, it was decided that the simplest option would be to generate a messages scaffold with `username_to`, `username_from` and `user_id` attributes (among others). Users post to the messages table, but it was implemented using a simple conditional statement in the view that the only messages that would display are messages with `username_to` equalling the current user's username. This way, a generic inbox is tailored to only display items that are intended for the current user. Similarly, for the sent box, only items with `username_from` equalling the current user's username are displayed. This can be seen in the following code snippet:

App/views/messages/index.html.erb:

```
<tbody>
  <% @messages.each do |message| %>
    <!-- only show messages sent to the current user -->
    <% if message.username_to == current_user.username %>
      <tr class="table-light">
        <td><%= message.subject %></td>
        <td><%= message.username_from %></td>
        <td><%= message.created_at %></td>
        <td><%= link_to 'View', message, class:"btn btn-outline-
dark" %></td>
        <td><%= link_to 'Delete', message, method: :delete, data: { confirm:
'Are you sure?' }, class:"btn btn-outline-danger" %></td>
      </tr>
    <% end %>
  <% end %>
</tbody>
```



The following process flow diagram describes the flow of sending a message:



Some of the more minor pieces of functionality are still worth mentioning. For example, the search feature which allows users to search for missing, found, or stolen pets by breed. This was implemented by creating a search method in the pet\_report model, which is then called from the gallery controller and utilised by the view.

App/models/pet\_report.rb:

```
#Search Function
def self.search(search)
  if search
    where(["breed LIKE ?", "%#{search}%"])
  else
    all
  end
end
```

App/controllers/gallery\_controller:

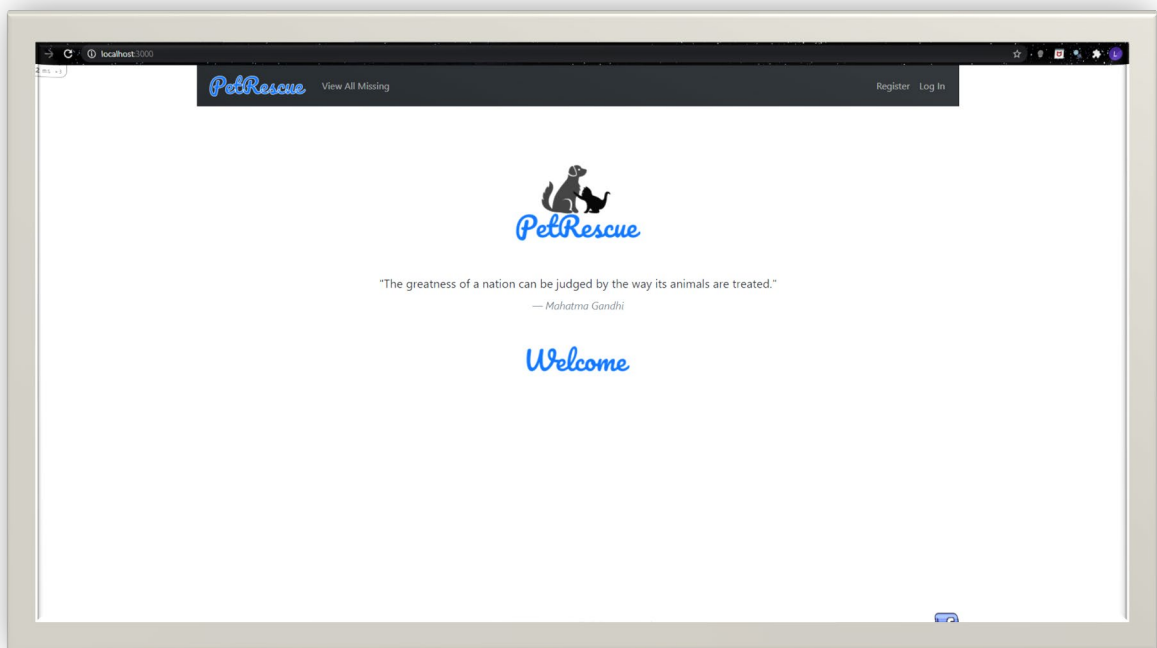
```
def index
  @pet_reports = PetReport.search(params[:search])
end
```

App/views/gallery/index:

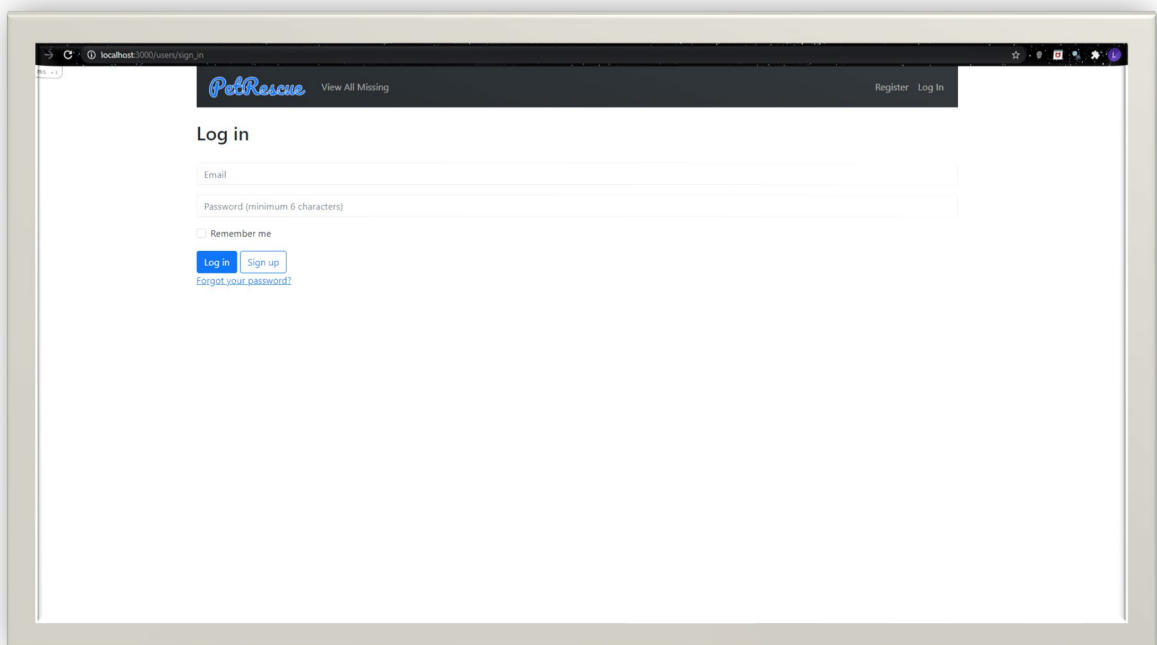
```
<!-- Search Missing Pets -->
<%= form_tag gallery_index_path, class:"form-inline my-2 my-lg-0", :method => 'get' do %>
  <%= text_field_tag :search, params[:search], class:"form-control mr-sm-2", placeholder:"Search by Breed" %>
  <%= submit_tag "Search", class:"btn btn-outline-secondary my-2 my-sm-0" %>
<% end %>
```

## 2.4. Graphical User Interface (GUI)

Landing Page – Presented to the user upon navigation to the site:



Log In – Presented to the user upon selecting 'Log In'. Similar to register and forgot password pages:



Report Pet Form – Presented to the user upon selection of ‘Report Missing / Found’:

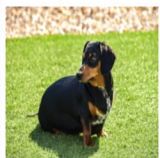

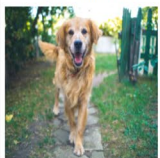
The screenshot shows a web browser window with the URL `localhost:3000/pet_reports/new`. The page title is "Report a Missing or Found Pet". The navigation bar includes "PetRescue", "Report Missing/Found", "View All Missing", "Inbox", and "Log Out". The form contains the following fields:

- Reporting:
- Date: 2021, May, 14, 17:00
- Location:
- Animal:
- Breed:
- Colour:
- File upload: "Choose file" button, "No file chosen" text
- Additional info:
- Buttons: "Create Pet report" (green), "Back" (grey)

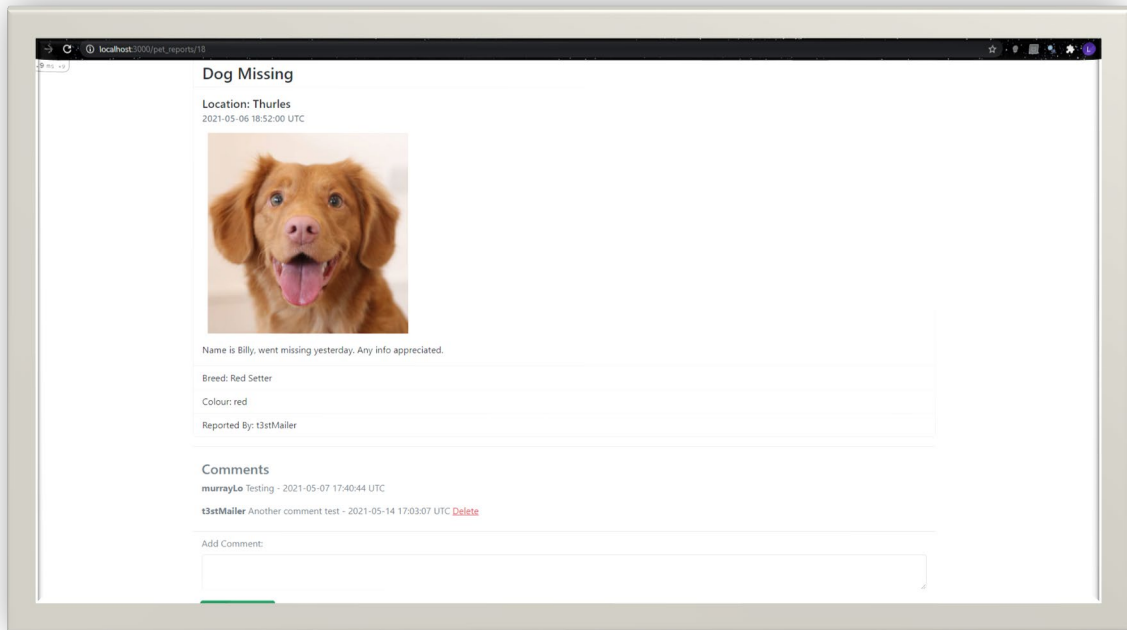
View Gallery – The main gallery view displaying all existing pet reports. Also contains the ‘search by breed’ functionality:

The screenshot shows a web browser window with the URL `localhost:3000/gallery/index`. The page title is "Currently Missing". The navigation bar is the same as in the previous screenshot. The page content includes:

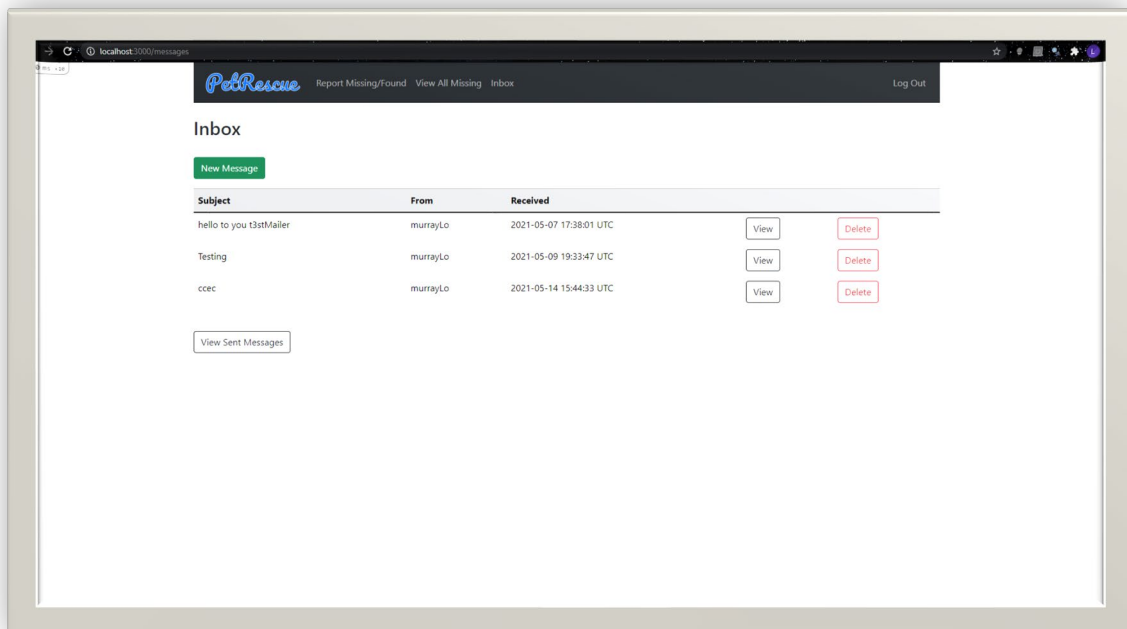
- Search by Breed:  Search
- Report a Lost or Found Pet:
- Three report cards, each with a "View" button and a timestamp:

Found	Stolen	Missing
<b>Arklow</b> Dog, Dachshund	<b>Dublin</b> Dog, Alaskan Malamute	<b>Fermoy</b> Dog, Labrador Retriever
		
<input type="button" value="View"/>	<input type="button" value="View"/>	<input type="button" value="View"/>
Reported 2021-04-27 22:49:00 UTC	Reported 2021-04-28 06:39:00 UTC	Reported 2021-04-30 08:01:00 UTC
<b>Stolen</b>	<b>Missing</b>	<b>Missing</b>

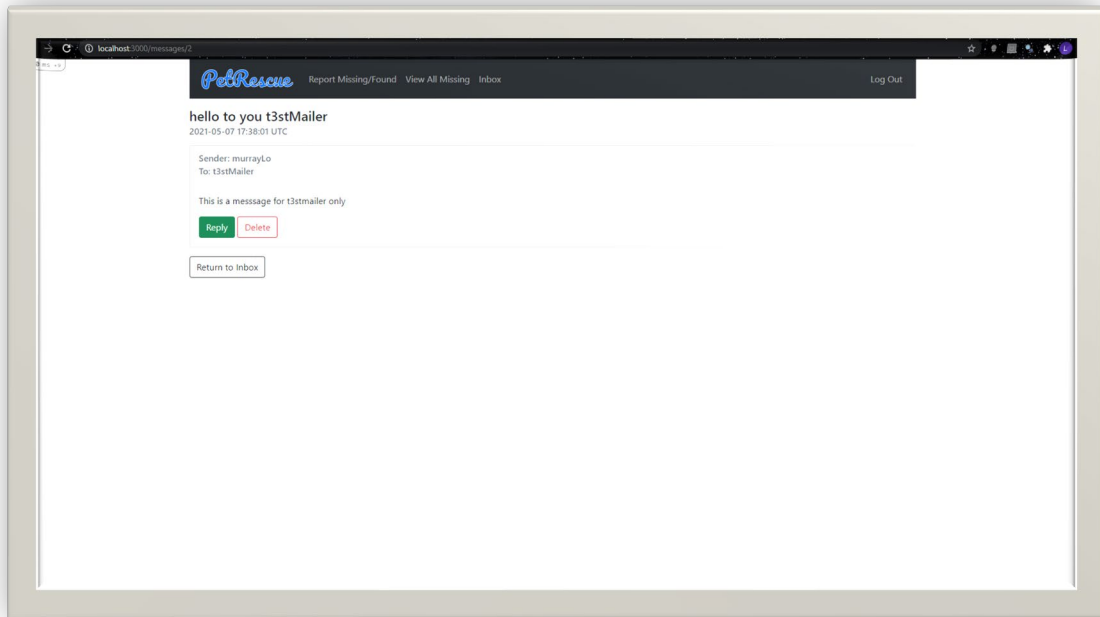
View Individual Pet Report – Presented to the user upon selecting ‘View’ on a pet report from the gallery. Displays additional information about the pet, and contains comments section to display and post comments:



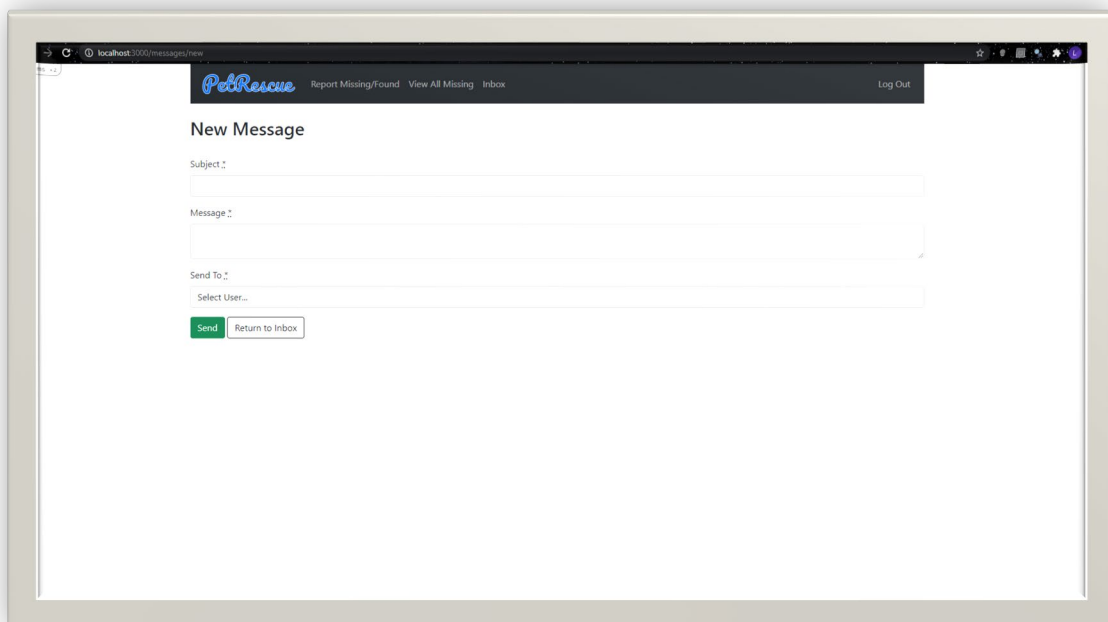
Inbox – Presented to the user upon selection of ‘Inbox’ in the navbar. Contains all messages sent to the current user:



**View Message** – Presented to the user upon selecting ‘View’ on a message from the inbox. Displays all message details (subject, message, sender, receiver, sent date) and allows deletion of the message or to send a reply:



**Send New Message** – Presented to the user upon selection of ‘New Message’ from the inbox, or ‘Reply’ from within a message:



## 2.5. Testing

Manual testing was carried out in different phases for the PetRescue application. The decision was made to manually test the site as an emphasis was put on exploratory and exception testing, which is better executed with the human eye rather than automation. According to Tommy Wyher of DevOps Zone, “Human testers can quickly identify when something looks “off.” Automated test scripts do not pick up these visual issues. When a tester interacts with software as a user would, they are able to discover usability issues and user interface glitches. Automated test scripts can’t test for these things” (Wyher, 2016). The testing for PetRescue was executed as follows:

### 2.5.1. Unit Testing

Unit testing was carried out continuously throughout the development of PetRescue. Each change that was made during development was tested immediately after its implementation and the code adjusted until testing of the specific change was successful. Upon completion of a development phase, a regression test was executed before pushing code to the github repository and merging back to the master branch.

### 2.5.2. Functional Testing

Upon completion of development, a functional test plan was drafted which covered testing of all functionalities of the application, as well as all end-to-end processes. Test cases and results are logged in the test plan which can be seen in the appendix of this document. (*See Appendix 6.3.1.*).

## 2.6. Evaluation

In order to evaluate the system in its entirety, a number of users were selected to carry out two designated tasks each on the system. Each user was timed during execution of their tasks and the results logged (*tasks and results can be seen in appendix 6.3.2.*). The actual times taken by the users to complete each task were compared to the expected timeframe for each task, which were identified by an experienced system user completing the tasks. This allows for identification of areas within the application which could be improved upon, or highlight area’s which have been designed correctly, with the average user in mind. In order to evaluate the system fairly, the users executing these tasks should span a wide demographic in terms of age, gender, and technical capability. However, as the system was being tested locally, the number of users that could partake in the exercises was limited to family members, due to covid restrictions. That being the case five family members were identified which still satisfied the need for a broad range in age, gender, and technical ability among users. The users executing evaluation scenarios are as follows (note no personal details for any user are disclosed):

User	Gender	Age Bracket	Technical Ability	System Experience
User 1	Female	65 - 69	Low	None
User 2	Male	60 - 64	High	None
User 3	Female	30 - 34	Moderate	None
User 4	Female	30 - 34	Moderate	None
User 5	Male	25 - 29	High	None

From the results of this evaluation technique, as can be seen in the appendix, it is clear that the system poses no issues to experienced or inexperienced users, with three of the ten tasks falling outside of the expected completion timeframe, however not by long enough to cause concern in any instance.

Following the execution of the tasks discussed above, each user was given approximately five minutes to explore the site at their leisure and advised to interact with the site as much as possible. Each user was then asked to complete a short survey on their experience using PetRescue (*Survey and results can be seen in appendix 6.3.3.*). The intention behind this was to give a more detailed insight into the specific areas of the site the users felt could either be improved upon, or were satisfactory in terms of features, design, and usability. The results show that overall users were satisfied with the system in terms of usability, design, and functionality, with one hundred percent of users answering that they would return to use the PetRescue site in the future.



### 3.0 Conclusions

The initial idea for the PetRescue project was to build a web application which could potentially offer genuine value to people in allowing them to report a missing, found, or stolen pet. Although issues were encountered along the way, resulting in a complete overhaul of the project; changing programming language and framework mid-way through the second semester of this year, the final product has achieved what it set out to achieve, as verified by independent users, and almost all functionality initially described in the project proposal has been implemented. Given the timeframe this project had to be completed in because of the change, there is immense pride taken from the final product.

Based on the extensive testing and evaluation carried out on the site, it is safe to say that PetRescue is a robust site that has achieved what it set out to achieve; that is, to improve upon the offering of its only competitor 'lostandfoundpets.ie' by offering users a simpler, more effective method of communicating a missing or found pet to as many people in as possible in the shortest amount of time. Features such as the comments section with automated email notifications and private messaging are another reason PetRescue stands out from the competition.

One feature initially proposed was to implement GPS to select the user's location when reporting a pet. Unfortunately, this was not implemented in the final product due to time constraints, however there is no doubt that development of this web application will continue after the project's submission, and GPS location is something which will almost certainly be implemented in a future release. It is felt that the positive experience taken from development of this application will result in further exploration of Ruby and the Rails framework and will most likely lead to development of other Ruby on Rails projects in the future.

## 4.0 Further Development or Research

As discussed above, given more time there is one feature which was initially proposed for this project but could not be implemented before the deadline. That is to offer GPS location as part of the pet report form. User should be able to select the location field and have their location displayed for selection. This was investigated during development of this project, but again, time constraints towards the submission deadline meant that enough time could not be dedicated to ensuring this was implemented properly. There is significant work in implementing such a feature which involves pulling the users current latitude and longitude coordinates and using reverse geocoding to find the location. There is no doubt that this will be implemented in a future release of PetRescue as development continues to further improve the site and make it a viable product for production.

## 5.0 References

McCormack, C., 2021. *Animal groups warn of rise in numbers of dog thefts*. [online] RTE.ie. Available at: <<https://www.rte.ie/news/2020/0709/1152224-dogs-stolen/>> [Accessed 8 May 2021].

Tutorialspoint.com. 2021. *MVC Framework - Introduction - Tutorialspoint*. [online] Available at: <[https://www.tutorialspoint.com/mvc\\_framework/mvc\\_framework\\_introduction.htm](https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm)> [Accessed 9 May 2021].

Wyher, T., 2016. *5 Reasons Why Manual Testing Is Still Very Important - DZone DevOps*. [online] dzone.com. Available at: <<https://dzone.com/articles/5-reasons-why-manual-testing-is-still-very-importa>> [Accessed 11 May 2021].

## 6.0 Appendices

### 6.1. Project Proposal



# National College of Ireland

## Project Proposal

Pet Rescue

05/11/2020

Programme Title: BSHCSDE4

Specialisation: Software Development

Academic Year 2020/2021

Student Name: Lorcan Murray

Student Number: 16105834

Email: x16105834@student.ncirl.ie

## Contents

1.0	Objectives.....	60
2.0	Background .....	60
3.0	Technical Approach.....	61
4.0	Special Resources Required .....	61
5.0	Project Plan .....	62
	Gantt Chart.....	63
6.0	Technical Details .....	64
7.0	Evaluation .....	64

## Objectives

The core objectives for the Pet Rescue project are to develop a Rich Internet Application (RIA) which will allow users to create an account, log in, and register details of an animal they would like to report either missing, or found. The process of reporting a lost or found animal involves the optional uploading of a photo, along with a description comprised of data taken from a number of user input fields such as species, breed, colour etc. as well as date/time and a 'last seen' location, based on GPS location. Users can browse all lost or found animals, select a thumbnail of any animal listed as lost or found which will pull that animal's profile and display all details on that animal to the user. A registered user can message the poster of a lost or found animal in order to enquire further or exchange contact details etc. Users will also receive a notification if for example an animal is reported found and the details match an animal which had previously been reported missing. In addition, users will have the ability to browse a personal inbox in order to view and reply to messages from other registered users. Aside from functionality, a key objective of this project is design. The application must be attractive, and of a sleek, simplistic design that is user friendly and easily navigable. The data required for reporting lost or missing animals should be strictly restricted to only relevant, necessary information, so as not to burden or overwhelm users with multiple unnecessary fields to fill in.

## Background

The idea for this project essentially stems from being a dog owner, and lover of animals in general. I have often found myself in a situation whereby I encounter a dog that looks possibly stray but am unable to do anything about it, as I may be short on time for example. In this scenario the only option is to move on and hope the animal is ok. In general, it is not advisable to approach an animal when one is encountered in this manner, even a family pet, as the animal may be nervous or scared and is therefore unpredictable and should be considered potentially dangerous. The proposed Pet Rescue application eliminates the need to approach a stray animal in the street, while still giving users peace of mind by offering the ability to do their part in helping to reunite the animal with its owners. Following the inception of this idea some research was undertaken to investigate whether anything similar already exists. Surprisingly, very few offerings could be found – most attempts to find an application to report a lost or found pet resulted in finding sites relating to pet adoptions. One site which did seem to have similar functionality to PetRescue was 'lostandfoundpets.ie'. Although the core functionality is similar, the lostandfoundpets site is cumbersome and laden with ads. When attempting to report a lost or found pet a lot of the details required were unnecessary and off-putting from the perspective of a user reporting a found pet, who may not have a hugely invested interest in reporting the pet in the first place. The aim of pet PetRescue is to make the task of reporting a missing or found pet as effortless as possible. The project is one which I feel I will be emotionally invested in, and as such will put great emphasis on not only the functionality of the site, but also on the look and feel from an end user perspective.

### Technical Approach

It is my intention that this project will be written in Python, with a html/css bootstrapped front-end to ensure optimal design. I have chosen Python as my preferred language for a number of reasons, the primary reason being that I currently have limited experience in using Python but am very aware of its versatility and capabilities, not least of which are the vast array of libraries which can be utilised. To compliment this, I intend on using the Django web framework with virtualenv for package management. Django is a fully-featured server-side web framework which is written in Python, and is one of the more popular choices as a framework for Python projects. My understanding is that Django's MVT (model-view-template) architecture is slightly more confined when compared to other, more architecturally loose frameworks such as Flask, however this offers better project stability and performance. Again, I have no prior experience in using Django, so this will be a challenge in itself but one I look forward to delving deeper into. In terms of the main database for this project, at the time of writing, a decision has not yet been finalised on what database system to use, however I am currently leaning towards PostgreSQL. PostgreSQL is an object-relational database which I understand from research is well known to be highly reliable and robust in terms of performance and works well with the Django framework. I also intend to use GitHub for version control.

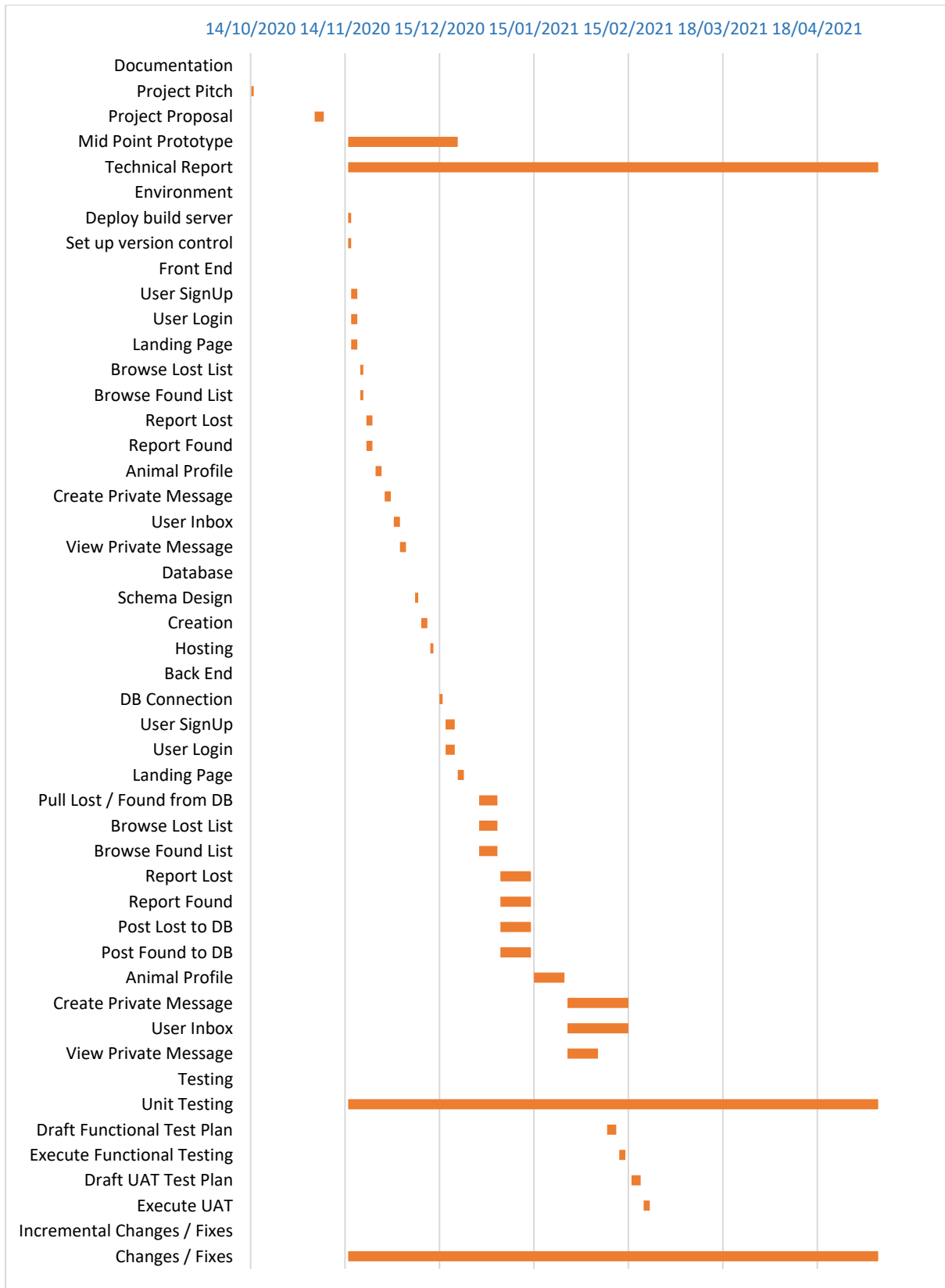
### Special Resources Required

It is my intention that no special resources will be required for this project other than a virtual environment. This will avoid the need to install Python packages globally and thus ensure system tools remain unaffected.

## Project Plan

Task	Start Date	Duration (days)	End Date
<b>Documentation</b>			
Project Pitch	14/10/2020	1	15/10/2020
Project Proposal	04/11/2020	3	07/11/2020
Mid Point Prototype	15/11/2020	36	21/12/2020
Technical Report	15/11/2020	174	08/05/2021
<b>Environment</b>			
Deploy build server	15/11/2020	1	16/11/2020
Set up version control	15/11/2020	1	16/11/2020
<b>Front End</b>			
User SignUp	16/11/2020	2	18/11/2020
User Login	16/11/2020	2	18/11/2020
Landing Page	16/11/2020	2	18/11/2020
Browse Lost List	19/11/2020	1	20/11/2020
Browse Found List	19/11/2020	1	20/11/2020
Report Lost	21/11/2020	2	23/11/2020
Report Found	21/11/2020	2	23/11/2020
Animal Profile	24/11/2020	2	26/11/2020
Create Private Message	27/11/2020	2	29/11/2020
User Inbox	30/11/2020	2	01/12/2020
View Private Message	02/12/2020	2	04/12/2020
<b>Database</b>			
Schema Design	07/12/2020	1	08/12/2020
Creation	09/12/2020	2	11/12/2020
Hosting	12/12/2020	1	12/12/2020
<b>Back End</b>			
DB Connection	15/12/2020	1	16/12/2020
User SignUp	17/12/2020	3	20/12/2020
User Login	17/12/2020	3	20/12/2020
Landing Page	21/12/2020	2	23/12/2020
Pull Lost / Found from DB	28/12/2020	6	03/01/2021
Browse Lost List	28/12/2020	6	03/01/2021
Browse Found List	28/12/2020	6	03/01/2021
Report Lost	04/01/2021	10	14/01/2021
Report Found	04/01/2021	10	14/01/2021
Post Lost to DB	04/01/2021	10	14/01/2021
Post Found to DB	04/01/2021	10	14/01/2021
Animal Profile	15/01/2021	10	25/01/2021
Create Private Message	26/01/2021	20	15/01/2021
User Inbox	26/01/2021	20	15/01/2021
View Private Message	26/01/2021	10	05/01/2020
<b>Testing</b>			
Unit Testing	15/11/2020	174	08/05/2021
Draft Functional Test Plan	08/02/2021	3	11/02/2021
Execute Functional Testing	12/02/2021	2	14/02/2021
Draft UAT Test Plan	16/02/2021	3	19/02/2021
Execute UAT	20/02/2021	2	22/02/2021
<b>Incremental Changes / Fixes</b>			
Changes / Fixes	15/11/2020	174	08/05/2021

## Gantt Chart





The above is a preliminary Gantt chart outlining the estimated effort in days for each project component. The schedule was drafted on the basis of full availability, however this is subject to change due to factors external to this project, such as other college projects, work, and family life, which may impact the estimated timelines outlined above. However, the schedule allows for delays with the months of March and April left available for any work which may be still outstanding at that point. The plan details the intention to complete front-end development as well as creation of the database by Christmas, to allow for a tangible prototype for the project's mid-point presentation. Back-end development then begins towards the end of December. The chart visualises the increased effort that is expected once back-end development begins. While unit testing will be an ongoing iterative process, the application will undergo a full functional test plan with a suite of functional test cases drafted prior to test execution. Similarly, once drafted, an independent potential user of the application will execute the user acceptance testing test plan to confirm that the application will satisfy the needs and expectations of the average user.

### Technical Details

While the front-end will consist of HTML, CSS and JavaScript, utilising bootstrap for an aesthetically pleasing design, the back-end processes and functionality will be programmed in Python. This will all sit within the Django web framework, utilising virtual environments to manage any package dependencies, thus eliminating the risk of clashes between pre-existing system components and required Python packages. As outlined in section 3.0, the selection of a database system has not yet been confirmed, with multiple options available and under review, currently more research is needed before a final decision can be made. That being said, the current front-runner is PostgreSQL as a RDBMS.

### Evaluation

System evaluation for this project will be completed via execution of a number of different testing methods. All components will be continually unit tested during development. A functional test plan will be drafted and executed with an emphasis on exception testing to ensure reliability under use by the average user. As well as this, it is my intention to have a user acceptance test plan drafted, which will be executed by an independent potential user of the app. This will likely be carried out by family members or friends – it should be noted that no personal information is required for UAT execution, so there are no ethical or GDPR implications to this.

## 6.2. Reflective Journals

October

Week 1 (26/09):

This week was very much focused on trying to get to grips with the new modules in this semester, and trying to get back into the studying midframe, which was difficult having come from several months of free time. Not much work was done outside of college hours, however a lot of time was spent thinking about ideas for a software project. One or two ideas came to mind but were dismissed as they were either impractical in terms of implementation, or already existed. I found myself getting frustrated by my lack of solid ideas and found this bringing on some anxiousness about the year ahead.

Week 2 (03/10):

This week, aside from labs during college hours, my extra-curricular efforts this week were invested in beginning CA1 and CA2 for 'Introduction to Artificial Intelligence'. Monday evening was spent researching different techniques that can be used in games of chess and beginning to draft up a report on those techniques. On Wednesday myself and two classmates arranged a team's call to discuss CA2. Following this call, I began to work on CA2, I did not get a whole lot done and come about 11pm decided to finish up for the night. I continued to work on this on Friday evening and again on Sunday and made some progress on pawn movements, however I found the logic difficult to fully understand so progress on this CA is slow. I also dedicated nearly 2 hours on Sunday to thinking and researching ideas for my software project and finally nailed down something which I wanted to pursue – a pet finder app.

Week 3 (10/10):

On Monday evening I sat down for an hour and a half to further develop my idea for my software project. I began by hand-drawing simple wireframes for the basic ideas I had, which I found helped give me clarity on the functionality I wanted from the app, as well as giving me further ideas for additional functionality that could be included. I also spent roughly an hour that evening working on CA2 for Intro to AI module, but again, progress on this is slow as I am finding myself spending more time trying to understand the logic than I am coding. On Wednesday I typed up a sort of script for my project pitch and on Thursday evening after class I recorded and submitted my pitch.

Week 4 (17/10):

Sunday was spent working on CA2 for intro to AI. I made some progress on pawn movements, having both black and white pawns moving as per the rules of the game, except for 1 issue with black pawns not changing to a queen when taking a white piece on Y=0. I was still working on this Sunday evening when my partner unexpectedly went into labour (3.5 weeks early). On the morning of Monday 19<sup>th</sup> my daughter was born, however due to some complications she was admitted to NICU. For this reason, my college work suffered quite a bit for the rest of the week, as my time was spent visiting between 11am and 7pm each day.

Week 5 (24/10):

Again, not much was done this week. Thankfully, my daughter was discharged from NICU on Monday, however most of time this week was dedicated to her, with feeding every 3 hours etc. I found it difficult to devote time to college. I did however get a limited window of time on Wednesday and Friday evening to start working on my CA for Multimedia and Mobile Application Development. Luckily, the lecturer for this module has extended the submission deadline to 8<sup>th</sup> November for this CA, so I can catch up on the time I missed out on the previous week.

## Reflection

This month has had its ups and downs. In general, I feel I need to find more time to dedicate to college work, as I feel I am falling behind a little bit. At the time of writing, I have not heard back on whether my project pitch was successful, so I will not be able to progress in any way with my software project until this is confirmed.

## November

### Week 6 (31/10):

This week saw the first (of many) submission of a substantial CA, namely CA1 for Strategic Management in which a PESTEL analysis was undertaken. Although it took some time to complete, I found this CA to be ok. In comparison to some of the deliverables for other modules this CA was far less cumbersome, and I was happy to be able to get it done and submitted in an efficient manner. I was informed this week that my project pitch was accepted and had my first supervisor meeting on Wednesday (4<sup>th</sup> Nov) with Frances Sheridan, who was filling in for Paul Hayes, who is my assigned supervisor. Frances gave me some valuable feedback on my project pitch, advising that the idea was good, but encouraging me to try to develop the idea a bit more. I found this meeting to be very helpful.

### Week 7 (07/11):

This week I submitted the project proposal. I found the project proposal to be a lengthy process, mainly due to working out expected timeframes and graphing it up as a gantt chart. I did however find that doing this gave me some good perspective on time vs deliverables, which otherwise may have slipped my mind and caused issues further down the line. Along with the proposal I submitted the ethics declaration form – albeit a late submission, as I had forgotten to submit the ethics while submitting the project proposal, however I contacted Frances and was advised to submit the ethics form in the next ethics submission point, which I did.

### Week 8 (14/11):

This was a busy week which saw more CA submissions – namely CA1 for Introduction to Artificial Intelligence and CA1, part 1 and 2 for Web Services and API Development. CA1 for Intro to AI was a short report on 3 different AI methodologies which can be implemented in a game of chess. All in all, this was a nice CA which caused little difficulty. For CA1 in Web Services I managed to get the submission in, however I found that the CA caused me great difficulty – part 1 specifically. Part 2 was a document with questions to answer. I found this to be a relatively simpler exercise and was happy enough with the final submission for part 2.

### Week 9 (21/10):

Again, this week saw another CA deadline. I submitted CA1 for the Data Application Development module which involved gathering datasets and doing up a report based on the findings from the data. For this CA I found the report proved more difficult than expected, due to the IEEE format that was expected for the document. I had not written a report in IEEE before and found it took substantially longer to complete than a standard report, however the CA was submitted as expected. This week on Wednesday (25<sup>th</sup> Nov.) I met with my newly appointed supervisor Rejwanul Haque. Rejwanul went over my project idea and gave some suggestions which may help improve the overall

idea. One such idea was a word comparison to notify users of a matching description to one they had posted. This is something I will investigate further, as I think it could be a great idea.

### Reflection

Having gotten through some CA's which initially seemed like a daunting task, I found that I am more driven to keep on top of the deliverables this semester. I found the supervisor meetings were very helpful and look forward to future meetings, which I believe will occur fortnightly.

### December

#### Week 10 (28/11):

This week was busy in terms of college work. With multiple deliverables due in the coming weeks pressure to complete project work is mounting. My focus this week was trying to complete the chess project for the Intro to AI module, as its due on the 18<sup>th</sup> and I currently do not have CA2 finished, nor have I started CA3.

#### Week 11 (05/12):

This week I began focusing on other assignments. Namely, my project for Mobile APP development and my groups team project for the web services and API development module. Work is currently incredibly busy, and I am finding myself working until after 11pm some nights which is impacting my college work, however so far, I am confident all deliverables will be met.

#### Week 12 (12/12):

This week I submitted my effort for the chess CA's for the Introduction to Artificial Intelligence module, on 18<sup>th</sup> Dec. Unfortunately, I struggled with CA2 too much, and although I got it completed in the end, it did not leave me with enough time to even attempt CA3, and as a result nothing was submitted for CA3. I am happy with my submission for CA2 however, as a few weeks ago I was seriously questioning whether I would be able to complete it.

#### Week 13 (19/12):

This week we submitted our team project for the Web Services and API Development module. This was submitted on Sunday 20<sup>th</sup> and myself and my teammates were working on the final few deliverables (report and presentation) up to a few hours before the deadline, however we were happy with the outcome. The midpoint presentation for the 4<sup>th</sup> year software project was also due this week. Unfortunately, not much work had commenced on my software project at that point, so I had not much tangible software to demo.

#### Week 14 (26/12):

This week I submitted my project for Multimedia and Mobile App Development, as well as my project for the Data Application Development module. The latter was tight in terms of submitting before the deadline, but I managed to get the report complete with a few minutes to spare before the deadline. I am also working on TABA's this week for Intro to AI and Strategic Management.

### Reflection

This was one of the busiest months I can remember. As discussed in the journal entries above, multiple deliverables were submitted this month, while work and family life still had to keep going. I am finding there are not enough hours in the day to get everything done in the timely manner I

would have always done up until 4<sup>th</sup> year, however so far almost everything has been submitted on or before the deadline.

## January

### Week 15 (02/01):

This week was dedicated to TABA's. In general I found the TABA's to be a much less stressful experience compared to the usual end-of-semester examinations, however the workload for these assignment-based assessments was substantial. My cohort, BSHCSDE4, had two TABA's due, one for the 'Introduction to Artificial Intelligence' module, and the other for 'Strategic Management'. Although both presented challenges, I found the Strategic Management TABA took more effort. I was pleased with the result for both TABA's, and both were submitted on time.

### Week 16 (09/01):

This was the first week in a very long time I attempted to close the laptop, forget about college, and focus on spending time with my family. This was a much-needed break from the stress of studying and working on projects and allowed me to devote my time and attention to my 12-week-old daughter. I feel this was one of the most important weeks of the year, as it not only gave me the time with my daughter I had been missing out on up until now, but it also allowed me to recharge the batteries, so to speak, and get myself prepared for the final stretch of a very tough 4 years in college.

### Week 17 (16/01):

This week I attempted to ease myself back into a college mind frame, starting some light work on my software project. Work was done on the sign-up functionality and was happy enough with how that progressed, now having a registration form that successfully posts new users to the DB. As with the previous week, I wanted to devote as much time as possible to my family before returning to classes, so I have not progressed my software project much further than the registration functionality.

### Week 18 (23/01):

This week saw a return to classes. So far, I am surprised to see the projected workload for this final semester seems to be slightly lighter than the previous semester, which is a welcome revelation. Aside from my software project, the biggest workload I can see on the horizon is the project for the cloud application development module. This is a project worth 100% of the module, and from what I can see it looks to have a workload not far off the main software project so it will be key to get this project started as soon as possible once more details become available.

## Reflection

This was a bit of a mixed month, beginning with the stress and workload involved in submitting the TABA's, then followed by a week off to unwind and easing back into college work by getting a start on coding my software project, then back to classes at the end of the month. So far, I am happy about how this final semester has started. After the stress of the previous semester, falling behind on my college work due to the birth of my daughter, I am determined this year to get an early start where possible on any assignments and projects due. I am feeling motivated to complete the last stretch of my time in NCI as diligently as possible.

## February

### Week 19 (30/01):

This week I began working on the project for cloud application development. This is a massive project, worth 100% of the module, so I am eager to get a start on this, however the installation of the Rails framework is proving difficult, and at the time of writing I have not yet successfully gotten the server running on localhost. The problem seems to be with the database connection, I have tried a multitude of things to fix this but so far have not resolved the issue and continue to work on this. I have also started to work on a group assignment for the Usability Design module. This project does not seem like it will cause any issues, however myself and my teammate wanted to get this underway as it is a sizeable document and if left until closer to the deadline could cause stress that could otherwise be easily avoided. I also got some minor work done on my software project.

### Week 20 (06/02):

This week I continued work on my Software Project. At this point I have user registration and login/logout functionality working, as well as having run my first migration to the DB, so I now have a model to accept a report of a lost/found pet from a user – but have yet to implement this functionality for the front end. A major breakthrough this week means I am now up and running with rails for my ruby project for Cloud App Dev module. As discussed, last week saw a lot of frustration around setting up rails. The issue was down to a DB issue. When creating the app via terminal, only a 'database' attribute was populated in the database.yml file. When I manually edited that file and specified a username, password, host, and port, I was able to rake the DB and run the server successfully.

### Week 21 (13/02):

This week was largely dedicated to making significant strides in the various projects. I began to work on my Ruby on Rails project, getting the app set up and the server running. I also dedicated a decent chunk of time on the usability design case study. The document is shaping up nicely and feel we should get this submitted well before the deadline. I have spent some time working on my software project this week, I now have the form to submit a missing/found report working, however have run in to an issue attempting to submit an image to the DB as part of the upload, which I am hoping will not cause me too much grief however at the time of writing I have not yet resolved the issue.

### Week 22 (20/02):

This week myself and my teammate completed our case study for CA1 in the usability design module. Submitting this early gave me a boost as I feel it is one step closer to the final goal. I progressed my work on my ruby on rails project this week also, creating the MVC for the different aspects of the project and getting CRUD functionality working for what will be an admin account (although no accounts implemented yet).

## Reflection

This month has once again been a busy month. The struggle between college, work and family life has not gotten any easier, however I have started to focus on the fact that I am on the final stretch now and submitting the first assignment of the semester has given me a boost and encouraged me to proactively work on other projects. I will be giving my all over the next few months.

March

Week 23 (27/02):

This week I continued work on my cloud app, so far, it is going well but still only basic functionality. I also began working on the gRPC project for the Distributed Systems module. I am finding this module to be quite tough so I imagine the project will be challenging. So far, I have set up the project using IntelliJ IDEA, and have decided to use Gradle as the build tool, rather than maven. I chose this as I believe Gradle is a slightly newer technology and would be beneficial to learn. Myself and my teammate also began work on our Usability Design project this week. I also attempted to work on my software project this week, as the panic is starting to set in. I am still having difficulties with Python in Django but hopefully will overcome these once I invest more time into the project.

Week 24 (06/03):

This was a busy week centred around the usability design project, as it is a big project workwise and the deadline is getting close. Myself and my teammate work well together, so I am not too concerned about getting this completed. We have delegated work to each other for different parts of the project, based on our strengths and weaknesses and have left each other to work on our respective parts and it is looking like we are almost finished. I have also tried to dedicate some time to my other projects this week, cloud app is coming together slowly (I feel once I have time to properly devote to this project that it will turn out quite well), the gRPC project I am struggling with somewhat, and my main software project is plagued with errors to the point that I am considering starting over.

Week 25 (13/03):

This week myself and my teammate submitted our project for usability design. It is the last deliverable for that module so feels like a big milestone and should hopefully allow us to dedicate more time to other projects with one module now down. I progressed with my cloud app project, getting user authentication working, including session management. I also spent time working on the gRPC project and have managed to get a simple unary call working in java. I have made a decision this week to start my software project over from scratch. It is very late to be starting a project that is due for submission in May, but if I do not restart the project, I may end up struggling with what I have and end up wasting more time. As well as restarting the project, I am going to code it in Ruby on Rails, instead of Python in Django. Although I have never used Ruby before this semester, I have made strides in my cloud app project, which is a Ruby on Rails project, and I feel a lot of more comfortable using this over Python, so I will arrange a meeting with my project supervisor to discuss this.

Week 26 (20/03):

This week I met with my project supervisor Rejwanul Haque. I explained that I was struggling with the Python project and that I would like to start over in Ruby. Rejwanul said this would be fine and that if I had any issues to contact him, which was reassuring. So, I have now set up a project in ruby and created a scaffold for my main model. I have found over this semester that I enjoy coding in ruby, so I look forward to making strides with my project. I also progressed work on my gRPC project. Having struggled to implement a server streaming service I have finally gotten one working. I also completed some work on my cloud app project and finally have this looking like a real application.

### Week 27 (27/03):

This week I Submitted my gRPC project. I was unhappy with how it turned out as I could not get several requirements working, but the core functionality is there for 3 different streaming services, so I am hoping this is enough to get some decent marks. I dedicated significant time this week to my main software project and am delighted with the results so far. Having been plagued with issues in Python I am now enjoying working on this project in Ruby. At the time of writing, I have basic CRUD functionality working for my main model and have implemented some basic user authentication, which needs further work. My cloud app project has taken a backseat this week to allow me to progress my main software project, but I will resume work on this next week.

### Reflection

Another busy month, however significant milestones have been achieved – All deliverables for usability design have been submitted, the gRPC project has been submitted with just one TABA left for that module, advancements have been made on my cloud app project and most importantly my main software project is in good shape for the first time, so all in all I am feeling confident everything is on track.

### April

#### Week 28 (03/04):

This week my main focus has been the cloud application development project. I have made good progress with this project and have achieved a lot of the functionality I intended to implement. I also spent some time this week working on my main project. This is progressing nicely however I have had some issues with image uploads via Active Storage and have been looking into shrine instead, which I will attempt to implement over the coming days.

#### Week 29 (10/04):

This week I submitted the cloud application development project. The final product was missing some of the requirements, however in general I was happy with how it turned out. I was unable to get the app hosted, after days of unsuccessful attempts, so I am hoping this does not cost me too many marks, either way another module is now complete. Having gotten the cloud project submitted, I have a lot more time to dedicate to my main software project. I still have a TBAA for the Distributed Systems module so will need to do some study for that, but this week I managed to get image uploads working successfully with Shrine. This is a big step forward as without it the application is useless.

#### Week 30 (17/04):

This week I have started to do some study for the Distributed Systems TBAA, however most of my time is dedicated to the software project. I now have the pet reports complete and styled the way I want them, with edit and delete functionality possible only for the owner of the report. This took some time to get working so I am happy it is done, as it is important. I am currently working on functionality for users to comment on reports and link comments to users. So far, I have the comments working, but am having difficulty associating the user to the comment.



### Week 31 (24/04):

This week I completed comments, with user associations and allowing deletion of the comment only for the owner of the comment. I am now working on mailers for the site. I have successfully implemented mailers for users requesting to reset their password, and at the time of writing am moving on to an automatic email being sent to the owner of a report when a new comment is posted. I ran in to an issue with sending mails for password resets which took some time to resolve, and turned out to be a configuration with the Gmail account the mails were being sent from, not allowing access from an insecure app. Once the setting was disabled the mails were sending without issue. I have also dedicated time this week to studying for the DS TBAA, which is taking place on 1<sup>st</sup> May.

### Reflection

As more deliverables are being submitted the pressure is easing slightly and I am enjoying working on my software project and being able to dedicate the time to it that it requires. Once the DS TBAA is complete on May 1<sup>st</sup> I will be working exclusively on the software project, so I am hopeful that I will then be able to implement everything I had hoped I would for this project.

## 6.3. Other materials used

### 6.3.1. Pet Rescue Test Plan

ID	Functional Area	Test Case	Description	Steps	Expected Results	Actual Results	Result
1.1.	User Authentication	Unregistered User	Test that unregistered user can only access gallery	<ol style="list-style-type: none"> <li>1. Navigate to site</li> <li>2. Do not log in</li> <li>3. Select 'View All Missing'</li> <li>4. Select 'Report a Lost or Found Pet'</li> <li>5. Select 'View All Missing' again</li> <li>6. Select 'View' on a Pet Report</li> </ol>	<ol style="list-style-type: none"> <li>1. User can access the Gallery</li> <li>2. When 'Report a Lost or Found Pet' is selected a message is displayed to the user advising they must be logged in and user is redirected to the Log In page</li> <li>3. When 'View' is selected, a message is displayed to the user advising that they must be logged in and user is redirected to the Log In page</li> </ol>	As Expected	Pass
1.2.	User Authentication	Registered User	Ensure a registered user can log in	<ol style="list-style-type: none"> <li>1. Navigate to the site</li> <li>2. Select Log In</li> <li>3. Input an invalid username and password and select Log In</li> <li>4. Input a valid username and password and select log in</li> </ol>	<ol style="list-style-type: none"> <li>1. Invalid details return an error message to the user and user is not logged in</li> <li>2. Valid details return a successful message and user is logged in</li> </ol>	As Expected	Pass
1.3.	User Authentication	Register	Ensure a user can register	<ol style="list-style-type: none"> <li>1. Navigate to the site</li> <li>2. Select Register</li> <li>3. Attempt to leave all fields blank and Sign Up</li> <li>4. Enter valid data in all fields and Sign Up</li> </ol>	<ol style="list-style-type: none"> <li>1. Error message displayed and registration is unsuccessful when fields are left blank</li> <li>2. User is successfully registered when valid data entered and user selects Sign Up</li> </ol>	As Expected	Pass

1.4.	User Authentication	Log Out	Ensure a user can log out of the site successfully	<ol style="list-style-type: none"> <li>1. While logged in to the site, select Log Out in the navbar</li> <li>2. Navigate to the gallery and select 'View' and 'Report a Pet Missing or Found'</li> </ol>	<ol style="list-style-type: none"> <li>1. Message is displayed to the user confirming they have been logged out and user is redirected to the landing page</li> <li>2. After log out, the user is unable to access the View or Report Missing or Found options from the Gallery</li> </ol>	As Expected	Pass
1.5.	User Authentication	Log In Post Register	Ensure a newly registered user can log in	<ol style="list-style-type: none"> <li>1. Using the credentials from test case 1.3. attempt to log in to the site</li> </ol>	<ol style="list-style-type: none"> <li>1. Log in successful</li> </ol>	As Expected	Pass
1.6.	User Authentication	Forgot Password	Ensure Password Recovery is functioning as expected	<ol style="list-style-type: none"> <li>1. While logged out, select Log In</li> <li>2. Do not enter any details</li> <li>3. Select Forgot your Password</li> <li>4. Enter a valid email address you can access and select 'Reset Password'</li> <li>5. Check inbox of the email address entered</li> <li>6. Follow the link in the email and enter a new password</li> <li>7. Log out of the application and log back in with the new password</li> </ol>	<ol style="list-style-type: none"> <li>1. Email is received with a link to reset password</li> <li>2. New password accepted</li> <li>3. User can log in with recovered password</li> </ol>	As Expected	Pass
2.1.	Pet Report	Report a missing pet	Ensure that a pet can be reported missing found or stolen	<ol style="list-style-type: none"> <li>1. Log in to the site</li> <li>2. Select Report Missing / Found</li> <li>3. Enter valid details into all fields</li> <li>4. Select Submit</li> </ol>	<ol style="list-style-type: none"> <li>1. Pet report is successfully posted</li> </ol>	As Expected	Pass

2.2.	Pet Report	Cancel Pet Report	Ensure a Pet Report can be cancelled before it is posted	<ol style="list-style-type: none"> <li>1. Log in to the site</li> <li>2. Select Report Missing / Found</li> <li>3. Enter valid details into all fields</li> <li>4. Select Back</li> </ol>	<ol style="list-style-type: none"> <li>1. User is returned to the Gallery page</li> <li>2. Pet Report is not posted</li> </ol>	As Expected	Pass
2.3	Pet Report	Edit Pet Report	Ensure the details of an existing pet report can be edited	<ol style="list-style-type: none"> <li>1. Select an existing pet report posted by the current user</li> <li>2. Select edit</li> <li>3. Edit some details and save the changes</li> </ol>	<ol style="list-style-type: none"> <li>1. Pet report is updated with the new details</li> </ol>	As Expected	Pass
2.4.	Pet Report	Delete Pet Report	Ensure an existing pet report can be deleted	<ol style="list-style-type: none"> <li>1. Select an existing pet report posted by the current user</li> <li>2. Select delete</li> <li>3. Cancel the warning</li> <li>4. Repeat and confirm the warning</li> </ol>	<ol style="list-style-type: none"> <li>1. When user selects delete and cancels the warning, the pet report is not deleted</li> <li>2. When user OK's the warning, the pet report is permanently deleted</li> </ol>	As Expected	Pass
3.1.	Gallery	Search Gallery	Ensure the search functionality in the gallery is working as expected	<ol style="list-style-type: none"> <li>1. Ensure user is logged in</li> <li>2. Navigate to the gallery</li> <li>3. Type a breed which matches the breed in an existing pet report into the search bar and select search</li> <li>4. Repeat the above, this time just partially type the breed and select search</li> </ol>	<ol style="list-style-type: none"> <li>1. Any pet reports which have a breed matching the breed entered into the search bar are returned</li> <li>2. Any pet reports which have a breed containing the partially typed breed entered into the search bar are returned</li> </ol>	As Expected	Pass
3.2.	Gallery	Report from Gallery	Ensure the option to report a lost or found pet is working from the gallery view	<ol style="list-style-type: none"> <li>1. Ensure user is logged in</li> <li>2. Navigate to the gallery</li> <li>3. Select 'Report a Lost or Found Pet'</li> <li>4. Complete the form and post the report</li> <li>5. Check the gallery for the new report</li> </ol>	<ol style="list-style-type: none"> <li>1. The new report is posted and is visible in the gallery</li> </ol>	As Expected	Pass

4.1.	Comments	Post a Comment	Ensure that a registered user is able to post a comment	<ol style="list-style-type: none"> <li>1. Ensure user is logged in</li> <li>2. Navigate to an individual pet report</li> <li>3. Scroll down to the comments section</li> <li>4. Enter some text in the 'Add Comment' box</li> <li>5. Select 'Post Comment'</li> </ol>	<ol style="list-style-type: none"> <li>1. Comment is posted</li> <li>2. Newly posted comment is visible in the comments section</li> </ol>	As Expected	Pass
4.2.	Comments	Delete a Comment	Ensure that a user can delete their comment	<ol style="list-style-type: none"> <li>1. Ensure user is logged in</li> <li>2. Navigate to the comment posted in test case 4.1.</li> <li>3. Select Delete</li> <li>4. Cancel the warning</li> <li>5. Repeat the above steps and OK the warning</li> </ol>	<ol style="list-style-type: none"> <li>1. Cancelling the warning does not delete the comment and comment is still visible in comments section</li> <li>2. OK the warning deletes the comment and the comment is permanently removed from the comments section</li> </ol>	As Expected	Pass
4.3.	Comments	Comment Notification	Ensure the owner of a pet report receives an email notification when a comment is posted on their report	<ol style="list-style-type: none"> <li>1. Log in and navigate to a pet report which was posted by a user account whose email address you have access to</li> <li>2. Post a comment</li> <li>3. Check the inbox of the pet report owners email address</li> </ol>	<ol style="list-style-type: none"> <li>1. An email is received into the pet owners inbox</li> </ol>	As Expected	Pass
5.1.	Messages	View Message	View a message from the inbox	<ol style="list-style-type: none"> <li>1. Ensure user is logged in</li> <li>2. Navigate to inbox</li> <li>3. Select 'View' on a message</li> </ol>	<ol style="list-style-type: none"> <li>1. The message is displayed to the user</li> </ol>	As Expected	Pass
5.2.	Messages	Send Message	Send a message	<ol style="list-style-type: none"> <li>1. Navigate to the inbox</li> <li>2. Select 'New Message'</li> <li>3. Enter message details and select 'Send'</li> </ol>	<ol style="list-style-type: none"> <li>1. Message is sent to the recipient</li> </ol>	As Expected	Pass

5.3.	Messages	Delete Message	Delete a message from the inbox	<ol style="list-style-type: none"> <li>1. Navigate to inbox</li> <li>2. Select 'Delete' on a message</li> <li>3. Cancel the warning</li> <li>4. Confirm the warning</li> </ol>	<ol style="list-style-type: none"> <li>1. Cancelling the message does not delete the message</li> <li>2. Confirming the message deletes the message and the message is removed from the inbox permanently</li> </ol>	As Expected	Pass
5.3.	Messages	Reply Message	Reply to a message	<ol style="list-style-type: none"> <li>1. Navigate to inbox</li> <li>2. Select View on a message</li> <li>3. Select Reply</li> <li>4. Enter details and select Send</li> </ol>	<ol style="list-style-type: none"> <li>1. Reply message is sent to the user</li> </ol>	As Expected	Pass
5.4.	Messages	Sentbox	View previously sent messages	<ol style="list-style-type: none"> <li>1. Navigate to inbox</li> <li>2. Select view sent</li> </ol>	<ol style="list-style-type: none"> <li>1. User is brought to the sent box</li> <li>2. All messages sent by the current user are displayed</li> </ol>	As Expected	Pass
5.5.	Messages	View Sent Message	View details of a previously sent message	<ol style="list-style-type: none"> <li>1. Navigate to inbox</li> <li>2. Select view sent</li> <li>3. Select View on a sent message</li> </ol>	<ol style="list-style-type: none"> <li>1. The sent message is displayed to the user</li> </ol>	As Expected	Pass
6.1.	Exceptions	Security	Attempt to access site while logged out via URL manipulation	<ol style="list-style-type: none"> <li>1. Ensure user is logged out</li> <li>2. Enter a specific route into the URL, ie /pet_reports/12</li> </ol>	<ol style="list-style-type: none"> <li>1. The user is not able to access the specific pet report with an ID of 12</li> </ol>	As Expected	Pass
6.2.	Exceptions	Pet Report	Delete other users pet report	<ol style="list-style-type: none"> <li>1. Ensure user is logged in</li> <li>2. Navigate to a pet report which was not posted by the current user</li> <li>3. Attempt to delete the pet report</li> </ol>	<ol style="list-style-type: none"> <li>1. There is no option to delete the pet report</li> <li>2. Deletion of a pet report not owned by the current user is not possible</li> </ol>	As Expected	Pass
6.3.	Exceptions	Comment	Delete other users' comment	<ol style="list-style-type: none"> <li>1. Ensure user is logged in</li> <li>2. Navigate to a pet report with comments</li> <li>3. Attempt to delete a comment which was not posted by the current user</li> </ol>	<ol style="list-style-type: none"> <li>1. There is no option to delete the comment unless it was posted by the current user</li> </ol>	As Expected	Pass

6.4.	Exceptions	Input Fields	Exception test on all input fields	<p>1. For all mandatory input fields, attempt to leave the field blank</p> <p>2. for all fields of type int, attempt to submit text</p>	<p>1. Cannot leave mandatory fields blank</p> <p>2. Int fields will not accept text - form submission fails</p>	As Expected	Pass
------	------------	--------------	------------------------------------	---	---	-------------	------

### 6.3.2. PetRescue System Evaluation Tasks

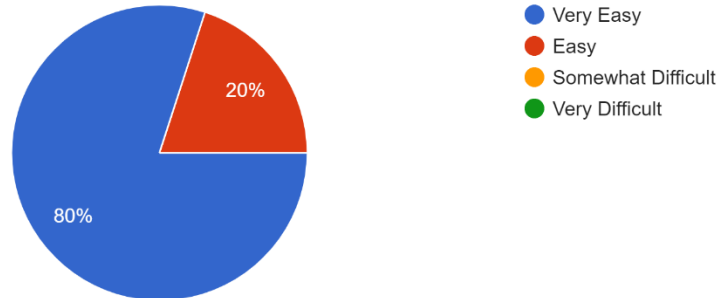
	Task	Preconditions	Expected Time	Actual Time	Comments
<b>User 1</b>	Post a Pet Report	User is registered	50 to 60 secs	54.5 secs	
<b>User 1</b>	View a Sent Message	Registered account with message(s) sent	20 to 30 secs	37.2 secs	User initially selected message in inbox in error.
<b>User 2</b>	Comment on Pet Report	User is registered	30 to 40 secs	40.17 secs	
<b>User 2</b>	Reset Password	User is registered	50 to 60 secs	50.85 secs	
<b>User 3</b>	Send Message	User is registered	30 to 40 secs	48.35 secs	
<b>User 3</b>	Log Out	User is registered	10 to 20 secs	17.9 secs	
<b>User 4</b>	Register Account	User is not registered	30 to 40 secs	28.31 secs	
<b>User 4</b>	Delete a Pet Report	User's account has previously posted pet report	20 to 30 secs	25.59 secs	
<b>User 5</b>	Edit a Pet Report	User's account has previously posted pet report	30 to 40 secs	27.16 secs	
<b>User 5</b>	Delete a Comment	User's account has previously posted comment	20 to 30 secs	22.76 secs	



### 6.3.3. PetRescue System Evaluation Survey

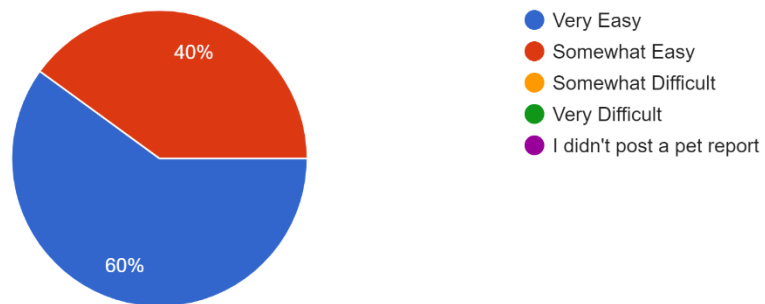
How easy or difficult was it to register and login to the site?

5 responses



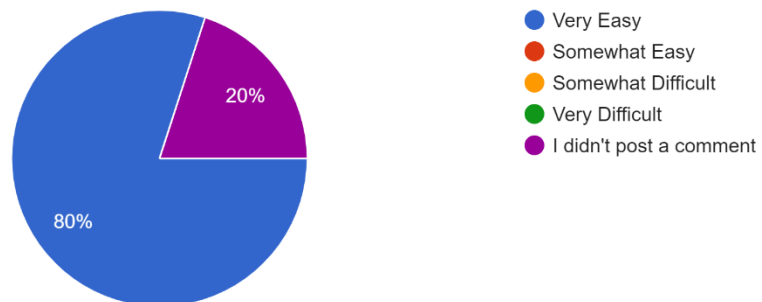
How easy or difficult was it to report a pet?

5 responses

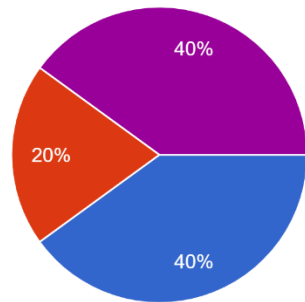


How easy or difficult was it to post a comment?

5 responses

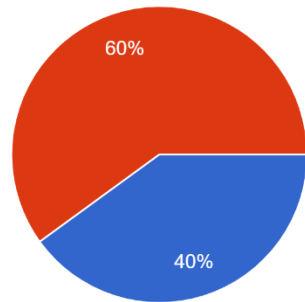


How easy or difficult was it to send a message?  
5 responses



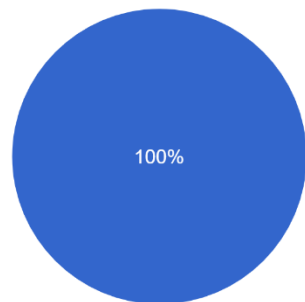
- Very Easy
- Somewhat Easy
- Somewhat Difficult
- Very Difficult
- I didn't send a message

Overall how would you describe the look of the site?  
5 responses



- Very Nice
- Somewhat Nice
- Somewhat Poor
- Very Poor

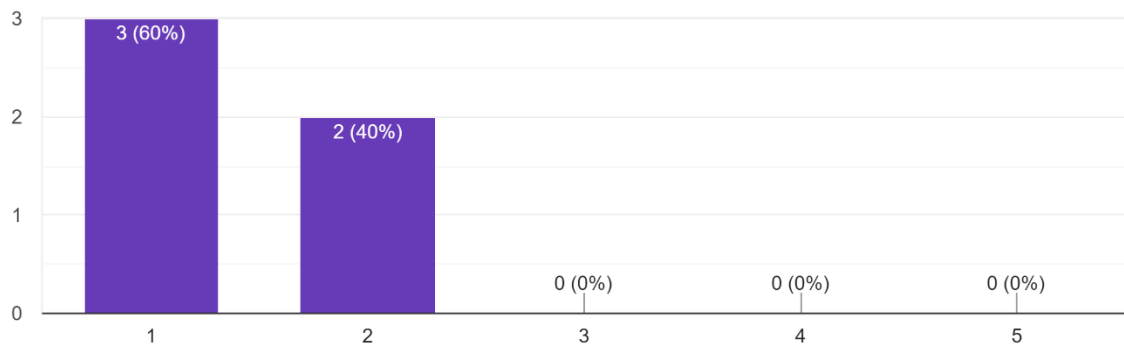
How did you find navigating through the site?  
5 responses



- Very Easy
- Somewhat Easy
- Ok
- Somewhat Difficult
- Very Difficult

On a scale of 1 to 5, how difficult was it to complete the tasks assigned to you?

5 responses



Would you return to use PetRescue in the future?

5 responses

